WHITE PAPER

Industry

Solutions Reference Architecture

Red Hat* OpenShift Container Platform 4.9 for Network Function Containerization Infrastructure

A reference architecture for Red Hat* OpenShift Container Platform running on 3rd Gen Intel® Xeon® Scalable processors with Intel® next generation Network Adapter that provides a guaranteed base level performance for real world Containerized Network Functions

Authors

Timothy Miskell

Ai Bee Lim

Shivapriya Hiremath

Red Hat* Reviewers

Bertrand Rault

Paul Lancaster

William Caban

Laney Badulis

1 Executive Summary

Over the past few decades the network transformation has been happening at a rapid rate with network functions moving from fixed function towards virtualized functions and recently the industry trends are pushing for another transformation into Containerized Network Functions (CNFs) for large scale cloud deployment fueled by the 5G services roll out to millions of subscribers.

Intel® and Red Hat* co-developed this high-performance reference architecture using Red Hat* OpenShift Container Platform 4.9 on 3rd Generation Intel® Xeon® Scalable processors Platform with Intel® Ethernet Network Adapter E810, in this case specifically E810-2C-QDA2. This combination enables the deployment of performant and low-latency container-based networking workloads onto different footprints, such as bare metal, virtual, private cloud, public cloud, or a combination of these, in either a centralized data centric location or at the network edge.

Table of Contents

Figures

Tables

2 Solution Brief

2.1 Business Challenge

The amount of data that traverse through the network kept increasing fueled by COVID pandemic, the rising trend in high-definition content delivery streaming services and immersive gaming user experience. On top of that, the industry currently is rolling out 5G solutions to provide higher throughput with lower latency to enable more use cases for user rich experience on consumption of these services. The demand on network infrastructures continues to be pressured over the course of this change. In order to stay agile and competitive, communication service providers need to look at economics of cloud approach to offer network services in order to keep up with the growing demand.

Investment needs to be made towards an infrastructure that will allow ability to develop applications and deploy services quickly from data center to the cloud, all the way to the edge of the network so that services are closer to the user. Intel® and Red Hat* defined a better together solution that will address this business challenge.

2.2 Solution Value

Business transformation requires automation, containers, and a modern infrastructure. That's exactly what communication service providers obtain when they deploy Intel's reference architecture for Red Hat* OpenShift Container Platform 4.9 for hybrid-multi cloud, or network cloud workloads. With this reference architecture, communication service providers can move to a modern, cloud-native infrastructure that meets today's demands and staying agile. Here are a few of the benefits of this solution:

- Take advantage of a validated, yet customizable design. Whatever the workload, this verified design helps organizations deploy data center infrastructure quickly and efficiently with less tuning—potentially reducing total costs and speeding time to deployment.
- Accelerate and simplify application development. Modern applications have a lot of moving parts, and there are many different concepts developers need to be aware of. This complexity can slow down innovation. OperatorHub is an easy-to-use catalog of operators (a method of packaging, deploying, and managing a Kubernetes-native application) from the Kubernetes community and Red Hat* partners.
- Easily scale your workloads. The combination of Red Hat* OpenShift Container Platform, Red Hat* OpenShift Data Foundation, OperatorHub, and Intel® technology makes it easy to scale a variety of workloads. These include databases, event streaming, video streaming, telecommunications service provider operations, data analytics, AI, and machine learning. The modular nature of the architecture enables developers to quickly add capacity, expand clusters, and extend capabilities.
- Meet growing storage and network needs. As the amount of data explodes in every industry, storing, managing and moving that data becomes increasingly challenging. Intel® Optane™ SSDs with OpenShift Data Foundation can store metadata and/or act as data cache to accelerate storage systems based on SATA and NAND SSDs. It can also help eliminate the storage penalty typical of infrastructures that use low-cost, high-capacity drives. Moving of the data is critical in order to provide high network throughput as part of the infrastructure.

2.3 Solution Architecture Highlights

This reference architecture delivers a turnkey, end-to-end solution using the latest Intel® platform technologies (see Figure 1) to deliver a production-ready foundation. The solution simplifies network cloud deployment at all network locations, adopt the latest best practices, and provides a stable, highly available infrastructure for running deployed services and applications. It also helps to provision and deploy a highly available OpenShift Container Platform 4.9 cluster either on-premises or in a hybrid cloud, i.e. multi-vendor and/or integrated near and far edge deployments, with both the registry and the application pods backed by OpenShift Data Foundation. The solution is powered by highly scalable 3rd Gen Intel® Xeon® Scalable processors and supported by Intel® Optane™ technology, Intel® Ethernet networking products and Intel® QuickAssist Technologies (Intel® QAT) acceleration technologies.

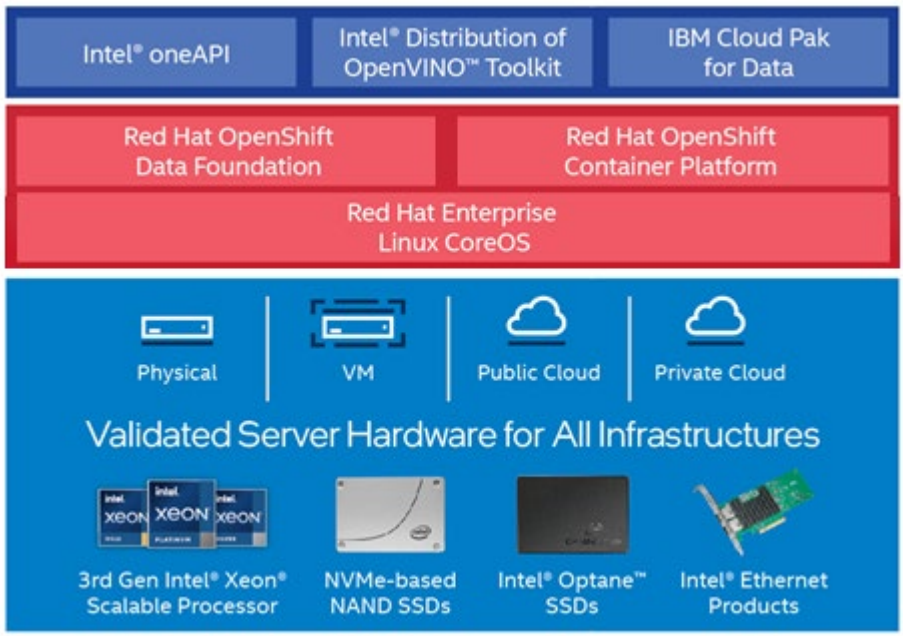

Figure 1. The Red Hat* OpenShift Container Platform is Optimized for Intel® Technologies.

2.4 A Closer Look at Red Hat* OpenShift Container Platform

OpenShift Container Platform provides a consistent and security-enabled Kubernetes cloud-native, hybrid-multi cloud experience (see Figure 2). It accommodates a large, scalable mix of microservices-oriented applications and their dependent components. OpenShift Container Platform uses the Container Runtime Interface–Open Container Initiative engine and Kubernetes-based orchestration. It provides container-as-a-service (CaaS) and platform-as-a-service (PaaS) workflows for developers and existing applications. OpenShift Container Platform provide high-performance, scalable infrastructure for various workload and use-cases. The following sections describe a few notable components of the overall platform.

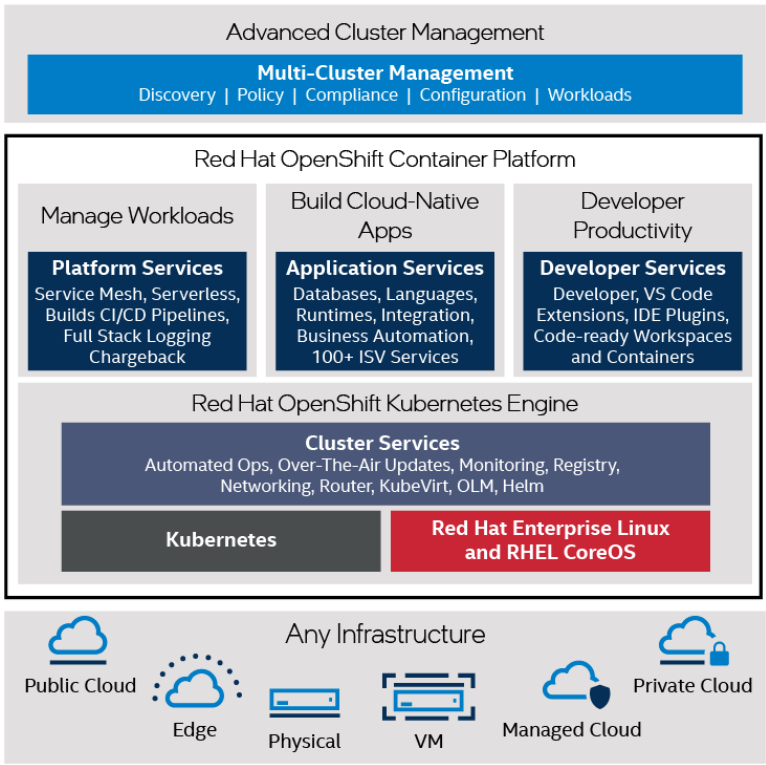

Figure 2. Red Hat* OpenShift Container Platform Helps Communication Service Providers Develop, Deploy, and Manage Innovative Applications at Scale.

2.4.1 OpenShift Marketplace

Developers and Kubernetes administrators can use Red Hat* Marketplace to gain automation advantages while enabling the portability of the services across Kubernetes environments. Developers can choose operators for a wide variety of tasks, including AI and machine learning, databases, integration and delivery, logging and tracing, monitoring, networking, security, storage, and streaming and messaging. Once installed on a cluster, operators are listed in the OpenShift Container Platform Developer Catalog, providing a self-service experience. Developers do not need to be an expert in applications but ease of install the operators needed to accomplish goals of the application or services. As a result, developers can spend more time in solving critical business needs and less on installing and maintaining the underlying infrastructures.

2.5 Example Use Case

As mentioned earlier, this reference architecture is targeted for a network cloud workload specifically communication service provider operations like Network Functions Virtualization (NFV).

2.6 Learn More

You may also find the following resources useful:

- 3rd Gen Intel® Xeon® Scalable processors
- Intel® Optane™ SSDs
- Intel® Ethernet products
- Red Hat* OpenShift Container Platform

Find the solution that is right for your organization. Contact your Intel® representative.

3 Implementation Guide

3.1 Introduction

The previous pages discussed the business value of using Intel® technology with OpenShift Container Platform, along with a high-level look at the technologies used in the solution. In this section, more detail is provided about those technologies, the seismic interpretation use case, and the steps required to run the use case experiment.

3.2 Key Technologies

3.2.1 3rd Gen Intel® Xeon® Scalable Processors

Intel's latest processors for data center workloads are 3rd Gen Intel® Xeon® Scalable processors. They are packed with performance- and security-enhancing features, including the following:

- Enhanced per-core performance, with up to 40 cores in a standard socket
- Enhanced memory performance with support for up to 3200 MT/s DIMMs (2 DIMMs per channel)
- Database compression with Intel® Vector Byte Manipulation Instructions
- Increased memory capacity with up to eight channels
- Support for Intel® Optane™ PMem 200 series
- Built-in AI acceleration with enhanced performance of Intel® Deep Learning Boost
- Faster inter-node connections with three Intel® Ultra Path Interconnect links at 11.2 GT/s
- More, faster I/O with PCI Express 4 and up to 64 lanes (per socket) at 16 GT/s
- Hardware-enhanced security of Intel® Crypto Acceleration

3rd Gen Intel® Xeon® Scalable processors offer new hardware-enhanced security features:

- Intel® Platform Firmware Resilience uses an Intel® FPGA to help protect, detect, and correct platform firmware.
- Intel® Secure Hash Algorithm (SHA) Extensions are designed to improve the performance of SHA-1 and SHA-256 on Intel® processors.
- Total Memory Encryption provides full memory encryption to help protect against physical attack.

3.2.2 Intel® Ethernet Products

Intel® Ethernet products are the foundation for server and appliance connectivity. They provide broad interoperability, critical performance optimizations, and increased agility for communications, cloud, and enterprise IT network solutions. Intel® provides data centers worldwide with innovative Ethernet components and solutions that are extensively tested for network interoperability,

reliability, and performance. Intel® Ethernet controllers, adapters, and accessories deliver speeds from 1 to 100 GbE—with versatile capabilities to optimize workload performance.

Intel® Ethernet 800 Series offers:

- **Higher Bandwidth** as Intel's first NIC with PCIe* 4.0 and 50Gb PAM4 SerDes
- **Improved Application Efficiency** with Application Device Queues (ADQ), Dynamic Device Personalization (DDP)
- **Versatility** with Flexible speeds: 2x100/50/25/10GbE, 4x25/10GbE, or 8x10GbE
- **RDMA support** for both iWARP and RoCEv2 providing a choice in hyper-converged networks

3.2.3 Intel® Network Adapter with DPDK*

Intel® networking products deliver continuous innovation for high throughput and performance for networking infrastructure. Intel® Network Adapter with DPDK* provides highly optimized network virtualization and fast data path packet processing. DPDK* supports many performance-sensitive telecommunications use cases running on this implementation.

3.2.4 Dynamic Device Personalization

By complementing 2nd Gen Intel® Xeon® Scalable processors with Intel® SSDs and Intel® Ethernet 700 Series Network Adapters, this reference implementation can help enterprises address storage bottlenecks and better utilize CPU resources. 10, 25, 40, and 100 GbE options can be deployed where necessary to help provide balanced system performance that scales well and delivers low latency.

The ability to reconfigure network controllers for different network functions on-demand, without the need for migrating all VMs from the server, avoids unnecessary loss of compute for VMs during server cold restart. It also improves packet processing performance for applications/VMs by adding the capability to process new protocols in the network controller at run time.

This kind of on-demand reconfiguration is offered by Intel® Ethernet 700 Series' Dynamic Device Personalization capability. This capability in the Intel® Ethernet 700 Series devices to load an additional firmware profile on top of the device's default firmware image, to enable parsing and classification of additional specified packet types so these packet types can be distributed to specific queues on the NIC's host interface using standard filters. Software applies these custom profiles in a non-permanent, transaction-like mode, so that the original network controller's configuration is restored after NIC reset or by rolling back profile changes by software. Using APIs provided by drivers, personality profiles can be applied by the DPDK*. Support for kernel drivers and integration with higher level management/orchestration tools is in progress.

Dynamic Device Personalization can be used to optimize packet processing performance for different network functions, native or running in virtual environment. By applying a Dynamic Device Personalization profile to the network controller, the following use cases could be addressed:

- New packet classification types (flow types) for offloading packet classification to network controller:
  - New IP protocols in addition to TCP/UDP/SCTP (examples include IP ESP and IP AH)
  - New UDP protocols, such as MPLSoUDP and QUIC
  - New TCP subtypes, like TCP SYN-no-ACK
  - New tunneling protocols like PPPoE and GTP-C/GTP-U
- New packet types for packet identification; these are reported on the packet's RX descriptor:
  - IPv6, GTP-U, IPv4, UDP, PAY4
  - IPv4, GTP-U, IPv6, UDP, PAY4
  - IPv4, GTP-U, PAY4
  - IPv6, GTP-C, PAY4
  - MPLS, MPLS, IPv6, TCP, PAY4

3.2.5 Intel® QAT

Intel® QuickAssist Technology (Intel® QAT) consists of a hardware accelerator to offload lookaside cryptographic and compression/decompression co-processing services. These services are accessible using Intel® QAT-related APIs that communicate via PCI configuration space access and associated rings stored in system memory. These features require an out-of-tree driver provided by Intel®, and all Intel® QAT functionality is directly supported by Intel®.

3.2.5.1 Cryptographic Functions

- Cipher operations:
  - Advanced Encryption Standard
  - Data Encryption Standard/Triple DES (3DES/TDES)
  - RC4
- Hash operations:
  - SHA-1, MD5

  - SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - SHA-3
  - ZUC
- Authentication operation:
  - HMAC, AES-XCBC, AES-CCM, AES-GCM, AES-XTS
- Cipher-Hash Combined Operation
- Key Derivation Operation
- Wireless Cryptography:
  - KASUMI, SNOW 3G, ZUC

3.2.5.2 Public Key Functions

- RSA Operation
- Diffie-Hellman Operation
- Digital Signature Standard Operation
- Key Derivation Operation
- Elliptic Curve Cryptography: ECDSA and ECDH

3.2.5.3 Compression/Decompression Functions

- Deflate

3.3 Red Hat* OpenShift Container Platform Reference Design

Tables 1– 4 provide a guide for assessing conformance to Intel's reference architecture for OpenShift Container Platform (both the master node and worker node configurations). It is expected that all required resources to implement a software-defined infrastructure reside within each server instance. For a system to conform to the reference architecture, all requirements in these tables must be satisfied, with the exception of Table 2, which provides an alternative deployment option (not referenced in the rest of this document) for use cases that benefit from external data nodes, separate from the Compute/Worker nodes. Table 2 provides a Base design; a Plus design is also available. For a system to conform to the reference architecture, all requirements in these tables must be satisfied.

Table 1. Required Hardware Bill of Materials

Ingredient	Master Node Requirement	Worker Node Base Requirement	Worker Node Plus Requirement	Storage Base Node (Capacity Optimized)	Storage Plus Node (Performance Optimized)
CPU	Intel® Xeon® Gold 5318N CPU 20c 2.0GHz 135W (SST-PP Config 2)	Intel® Xeon® Gold 5318N CPU 24c 2.1GHz 150W	Intel® Xeon® Gold 6338N CPU 32c 2.2GHz 185W	Intel® Xeon® Gold 6330N CPU 28c 2.2GHz 165W	Intel® Xeon® Gold 6338N CPU 32c 2.2GHz 185W
Memory	256GB – Required	256GB - Required	512GB – Required	256GB - Required	256GB - Required
Intel® Optane™ PMEM	Not Required	Not Required	Recommended	SSD for Metadata, Caching	PMEM for Metadata, Caching
NIC	2x Intel® Ethernet Network Adapter E810-CQDA2 or E810-XXVDA2	2x Intel® Ethernet Network Adapter E810–CQDA2or E810-XXVDA2	2x Intel® Ethernet Network Adapter E810–CQDA2 or E810–2CQDA2	2x Intel® Ethernet Network Adapter E810–CQDA2 or E810-XXVDA2	2x Intel® Ethernet Network Adapter E810–CQDA2 or E810-XXVDA2
LOM	Required	Required	Required	Required	Required
Intel® QAT	Recommended	Required	Required	Recommended	Recommended
Storage for Boot Application Storage	Recommended 2 x 4.0TB Intel® D7-P5510	Recommended 2 x 4.0TB Intel® D7-P5510	Recommended 4 x 4.0TB Intel® D7-P5510	Required 8x 16TB Intel® P5316 QLC NAND	Required 6 x 4.0TB Intel® D7-P5510

Table 2. Network Switches

Hardware	Recommendation
1x Cisco Nexus3000 C3232C Chassis (Nexus 9000 Series), NXOS: version 9.2(1)	Required

Table 3. Firmware Versions (All Required)

Ingredient	Version
BIOS	SE5C6200.86B.0020.P34.2107301450
3rd Gen Intel® Xeon® Scalable processor platforms	0xd0002e0
BMC	2.85.a963a1e7
Intel® Ethernet Network Adapter E810	2.30

Table 4. Software Bill of Materials

Software	Version	Recommendation
Red Hat* OpenShift Container Platform	4.9	Required
HAProxy*	2.2.15	Recommended
Dnsmasq*	2.79	Recommended
NGINX*	1.20	Required
VPP*	22.02	Required
Intel® QAT	1.7.l.4.10.0-00014	Recommended
DPDK*	21.11	Required
SR-IOV Network Operator	4.9.0-202202211206	Required
Node Feature Discovery	4.9.0-202202211131	Recommended

3.4 Security

This reference architecture must implement and enable Intel® Boot Guard Technology to ensure that the firmware is verified in the boot phase. This makes it possible to establish the hardware root of trust. With UEFI Secure Boot Methodology, the secure boot can be extended into the OS and further extend the chain of trust into loading of signed kernel modules.

Alternatively, all Intel® architecture-based solutions recommend installing the TPM, which enables administrators to secure platforms for a trusted (measured) boot with known trustworthy (measured) firmware and OS. The TPM also enables local and remote attestation by third parties to advertise such known good conditions (assuming the presence of Intel® Trusted Execution Technology).

3.4.1 Side Channel Mitigation

This reference architecture has been verified for Spectre and Meltdown exposure using Spectre and Meltdown Mitigation Detection Tool v0.42, which verifies that firmware and OS updates protect against known attacks:

- Spectre Variant 1
- Spectre Variant 2
- Spectre Variant 3
- Spectre Variant 3a
- Spectre Variant 4
- Foreshadow (SGX)
- L1 terminal fault
- Foreshadow-NG (OS)
- Foreshadow-NG (VMM)
- MDS
- ZombieLoad v2
- TAA
- No eXcuses
- iTLB multihit
- MCEPSC

4 NFV Performance Requirements

This section provides information necessary to verify the performance metrics for the Intel® Reference Implementation for Cloud-Native Hybrid-Multicloud and SDN/NFV Built Using Red Hat* OpenShift Container Platform, to ensure that there are no anomalies. The current Base and Plus solutions were tested with specific hardware and software configurations.[21] For performance baseline and performance verification, the Turbo enabled with C-states enabled settings are used to gather data for maximum performance configuration.

DPDK* is a core platform technology component for Master Node, Base Worker Node, and Plus Worker Node configurations. As such, it is expected that a compliant system must implement the DPDK* software and meet performance metrics. Figure below illustrates SDN and NFV workload examples across all network locations, including centralize sites, regional points of presence

(PoPs), remote central offices (COs), access COs, or customer premises. Each network location has different network throughput requirements as well as thermal and power conditions. Intel® provides platform architecture solutions that span all levels of network location deployment with the same consistent Intel® architecture software investment.

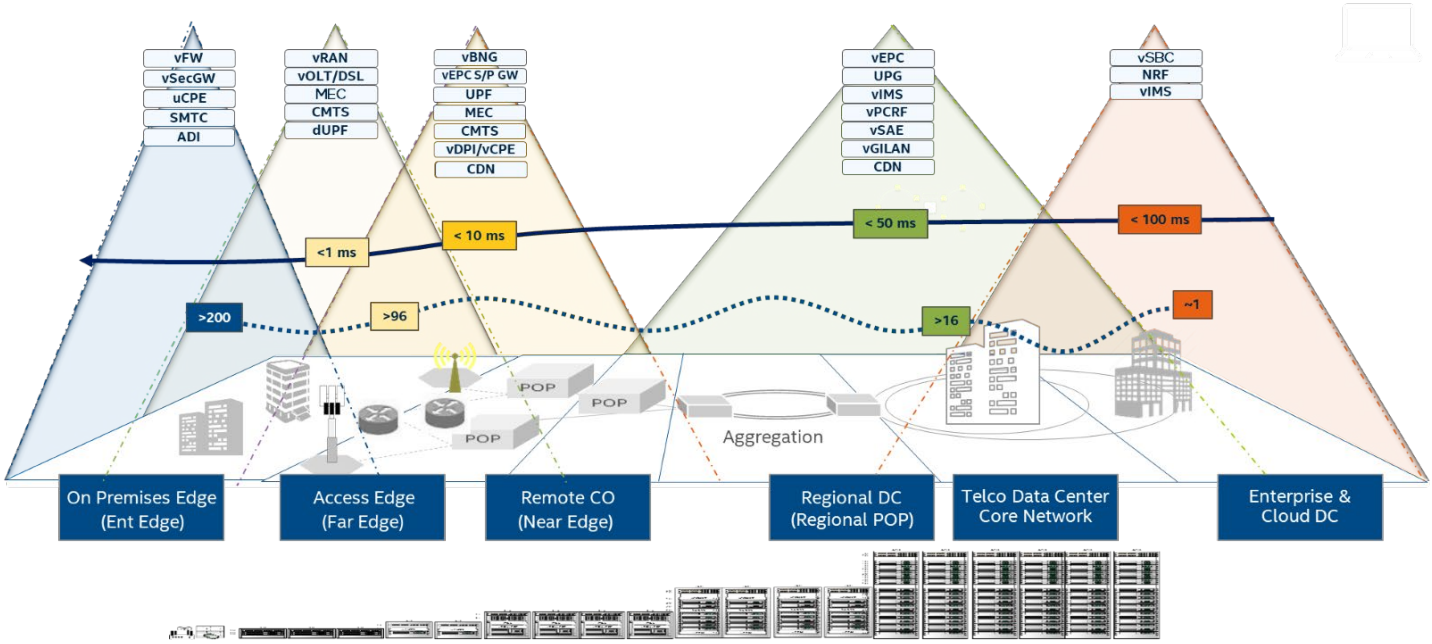

Figure 3. SDN and NFV Workloads Per Network Location Deployment

DCI Node	Distributed Continuous Integration[1] node to automate deployment of RHOCP in cluster mode
Provisioning Node	Provisioning node to deploy RHOCP onto and act as a front end for the master and worker nodes
Master Node 1	Master node as part of a 3 node HA cluster that acts a controller
Master Node 2	Master node as part of a 3 node HA cluster that acts a controller
Master Node 3	Master node as part of a 3 node HA cluster that acts a controller
Worker Node 1	Worker node where CNFs are deployed

[1] For further information regarding DCI, refer to: https://doc.distributed-ci.io

Figure 4. Red Hat* OpenShift 4.9 Deployment Overview

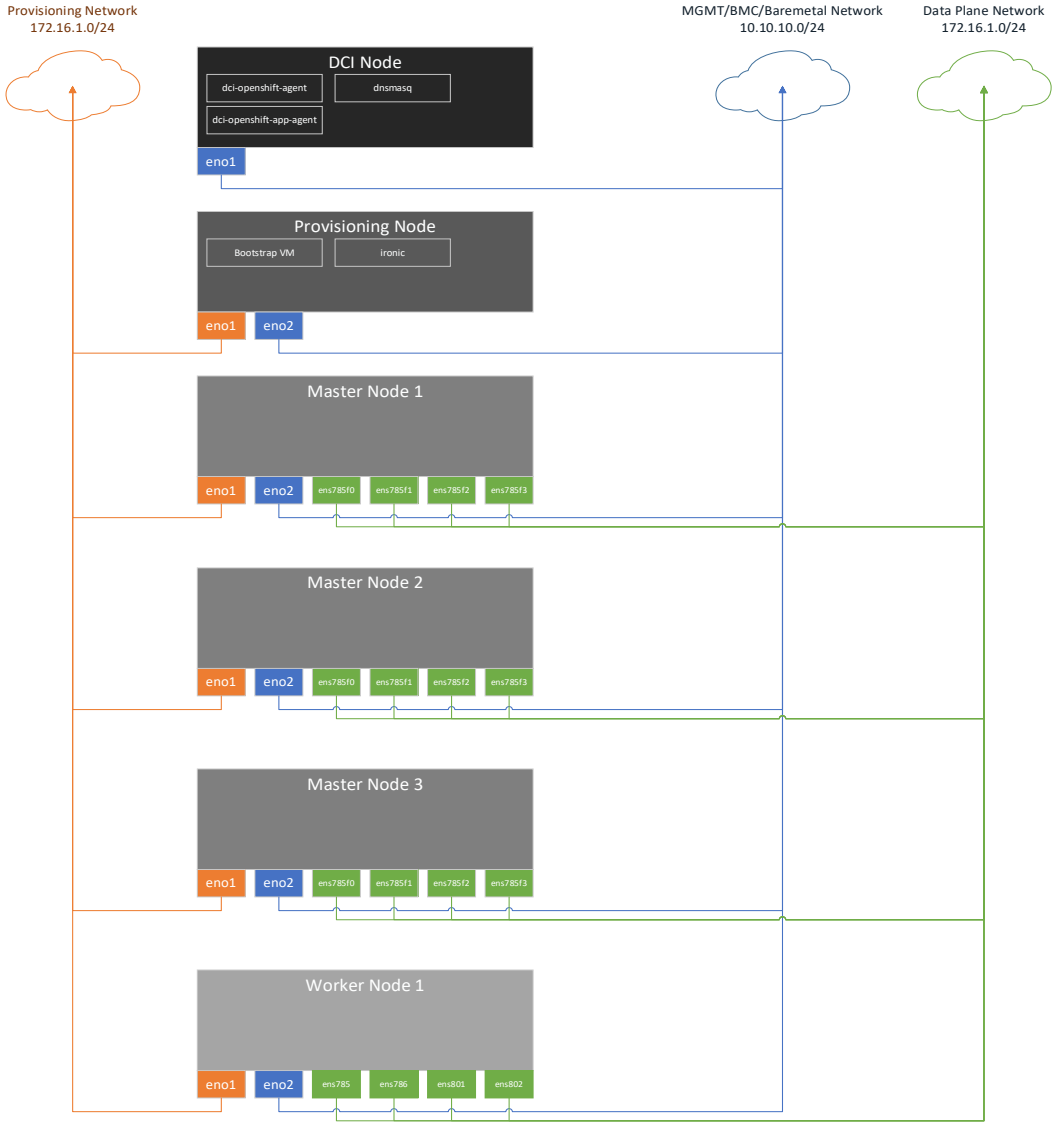

Figure 5. Red Hat* OpenShift 4.9 Network Topology Overview

4.1 Performance Baseline Requirement

There are a few applications that are required to be executed once the system has been configured per the bill of materials, BIOS configuration, Intel® Advanced Technology configuration, and software stack configuration. The output of these applications provides a performance baseline on the system's expected latency performance, memory bandwidth, and jitter. If the latency performance and memory bandwidth performance are outside the range as tabulated above, revisit the Cluster Configuration (A.2, A.3)section to verify the validity of system components, BIOS settings, and software components.

4.1.1 Cyclictest

This reference implementation must demonstrate the NFVI system latency for the wake-up time of the threads running in the container as specified in Table 1- 4. To validate conformance to this requirement, benchmarking must be performed using the cyclictest application running in a container. Appendix E.3 in 639782 provides information on running the cyclictest application.

Table 5. Cyclic Test Performance Requirements

Cyclictest Performance in Latency (Executed Inside the Container)	1 CPU	8 CPUs
Minimum	<5 µs	<10 µs
Average	<10 µs	<15 µs

[a]A real-time kernel is available from Red Hat* for any workload that requires strict low latency and deterministic behavior.

4.1.2 Memory Latency Checker – Reference Only

The Memory Latency Checker (MLC) is also required (Table 5 and Table 6). Download the latest version and execute the application, unzip the tarball package, and go into the Linux folder and execute ./mlc or ./mlc_avx512.

Table 6. Memory Latency Checker Local and Remote NUMA Performance Requirements

	3rd Generation Intel® Xeon® Scalable Processor	
Memory Latency Checker	**Local NUMA Node**	**Remote NUMA Node**
Idle Latency	75 ns	130 ns
Memory Bandwidth Between Nodes within the System **Using read-only traffic type**	55000	149000

Table 7. Memory Latency Checker Peak Injection Memory Bandwidth Requirements

	3rd Generation Intel® Xeon® Scalable Processor	
Peak Injection Memory Bandwidth Using All Threads **Peak Injection Memory Bandwidth (1 MB/s)** **Using All Threads**	**Base Configuration**	**Plus Configuration**
All Reads	290000	290000
3:1 Reads-Writes	260000	260000
2:1 Reads-Writes	250000	250000
1:1 Reads-Writes	220000	220000
Stream-Triad-Like	250000	260000
Loaded Latencies Using Read-only Traffic Type with Delay=0	210	270
L2-L2 HIT Latency	50	50
L2-L2 HITM Latency	50	50
Remote Socket L2-L2 HITM Latency Data Address on Writer Socket	110	110
Remote Socket L2-L2 HITM Latency Data Address on Reader Socket	115	130

4.1.3 Jitter – Reference Only

The jitter application measures the variability of latency in the execution of a user space dummy loop with a dummy operation. Use the following steps to download the tool, build, and execute the tool, targeting an idle core:

```
1. git clone https://gerrit.fd.io/r/pma tools
2. cd pma tools/jitter
3. make
4. ./jitter -c 2 -i 200
```

In the output, review the Inst_Jitter column; this should range from 5K to 100K if **Max Performance Profile with Turbo Mode** is enabled. When using **Deterministic Performance** in a BIOS setting, the jitter should not exceed 10K.

4.1.4 Intel® QAT cpa_sample_code

The Intel® QAT cpa_sample_code should be executed to ensure that the Intel® QAT functionality is tested to the expected performance (Table 17). This test is not required for the Controller Node.

Table 8. Intel® Quick Assist Technology® CPA Sample Code Performance Requirements

Intel® QAT cpa_sample_code	**Compression[a]**	**Encryption[b]**	**RSA2048[c]**	**PCIe* Width**
Base Worker Node Configuration				
Intel® C626 Chipset	24 Gb/s	40 Gb/s	100 K/Ops	x16
Intel® QAT 8970 (PCIe*) AIC or Equivalent Third-Party Intel® C626 Series Chipset Intel® QAT-enabled PCIe* AIC	34 Gb/s	40 Gb/s	100 K/Ops	x8
Plus Worker Node Configuration				
Intel® C627 Chipset	54 Gb/s	100 Gb/s	100 K/Ops	x16
Intel® C628 Chipset	54 Gb/s	100 Gb/s	100 K/Ops	x16

[a] Performance to be measured at 8 KB packet size
[b] Performance to be measured at 4 KB packet size
[c] Performance to be measured at 2 KB packet size

4.1.5 OpenSSL Speed Benchmark

The OpenSSL speed benchmark should be executed to perform Bulk Encryption with AES128-CBC HMAC-SHA1 and Public Key Exchange with RSA2048. Verify that the expected results are as shown in Table 9 to obtain a baseline for the performance of the system. This test is not required for the Master Node. Refer to Section 5.1.2 in 639782 for an example of verification information.

Table 9. Intel® Quick Assist Technology® OpenSSL Speed Benchmark Requirements

OpenSSL Speed Benchmark OpenSSL Benchmark	AES128-CBCHMAC-SHA1[a]	RSA2048 (1 Core)	
	Intel® QAT	Intel® QAT	Software
Base Configuration	40 Gbps	100 K sign/s	1 K sign/s
Plus Configuration	100 Gbps	100 K sign/s	1 K sign/s

a Performance to be measured at 16 KB packet size

4.2 Packet Processing Performance

Systems compliant with this reference implementation must demonstrate a minimum packet processing performance as specified in Table 10, implementing DPDK* to optimize packet processing performance. To validate conformance to the packet processing performance requirements, benchmarking must be performed using the DPDK* L3 Forwarding application.

The RFC2544, Benchmarking Methodology for Network Interconnect Devices Zero Packet Loss test case, is used to validate conformance. This test is used to determine the target platform throughput as defined in RFC1242, Benchmarking Terminology for Network Interconnection Devices. For this requirement, the RFC2544 test case uses DPDK* L3 Forwarding application (see Appendix F.2 in 639782) as the test application. Refer to Appendix F.2 in 639782 and the Benchmarking Methodology for Network Interconnect Devices, RFC2544, and Benchmarking Methodology for Network Interconnect Devices, RFC1242.

This reference implementation requires that the following KPIs be collected to ensure that these network use cases are meeting the expected performance.

- SR-IOV Function DPDK* application
- SR-IOV Function CNI application
- Topology manager for Kubernetes

Table 10. DPDK* L3 Forwarding RFC2544 Performance Requirements

Network KPIs	Test Cases	Packet Rate Line Rate with Packet Size 256B
25 Gbps NIC	PF Pass-through	90%
	SRIOV DPDK* APP	90%
	SRIOV CNI APP	90%
2x 25 Gbps NIC	PF Pass-through	90%
	SRIOV DPDK* APP	90%
	SRIOV CNI APP	90%

4.3 VPP*-IPSec for Secure Transport

4.3.1 Overview

The Vector Packet Processor (VPP*) application is a production grade virtual Switch (vSwitch) that leverages the Data Plane Development Kit (DPDK*) in order to scale performance across queues and cores, and supports deployment in baremetal, virtualized as Virtual Network Function (VNF), and containerized environments as a Containerized Network Function (CNF). Internet Protocol Security (IPSec) is a means to create an encrypted L3 tunnel between two endpoints over an insecure channel. VPP* provides the ability to create IPSec tunnel, and offers support for a variety of engines to perform symmetric encryption and authentication either through software or hardware, including native support, OpenSSL, the Intel® IPSec Multi-Buffer Library, along with Quick Assist Technology (QAT) hardware. Furthermore, IPSec supports multiple ciphers including for example AES-128-CBC SHA1-HMAC as well as AES-128-GCM, which can be preconfigured statically or dynamically between two endpoints. VPP* IPSec is typically leveraged for Firewall or VPN applications deployed on Network Function Virtualization Infrastructure (NFVI).

4.3.2 Test Setup

The figure below illustrates how the VPP* CNFs are deployed onto a Worker Node Plus Configuration. The benchmarks focus on single socket performance on 3rd Generation Intel® Xeon® Scalable Processors. In this case, two VPP* containers are deployed per socket, specifically VPP* Endpoint 0 (VPP* EP0) and VPP* Endpoint 1 (VPP* EP1). Each VPP* container is allocated a pair of 100 GbE Virtual Functions (VFs) from a single Physical Function (PF). For the forward flow sent to VPP* EP0, the first VF receives cleartext

traffic from the traffic generator, in this case Pktgen, VPP* encrypts the traffic with the corresponding cipher and afterwards sends the encrypted packets to the other VF. Similarly, for VPP* EP1, the first VF receives encrypted traffic, VPP* decrypts the traffic with the corresponding cipher and afterwards sends the clear text traffic to the other VF. The flow in the reverse direction performs the same operations in reverse order.

The benchmarks compare VPP* IPSec aggregate throughput performance for two engines, as well as for two ciphers. In particular we compare the VPP* with and without Intel® IPSec Multi-buffer library support, as well as the AES-128-CBC-SHA-1 along with the AES-128-GCM ciphers. Furthermore, Pktgen is configured to send at the maximum possible rate, and the benchmarks measure the Maximum Receiver Rate (MRR) of the end-to-end network.

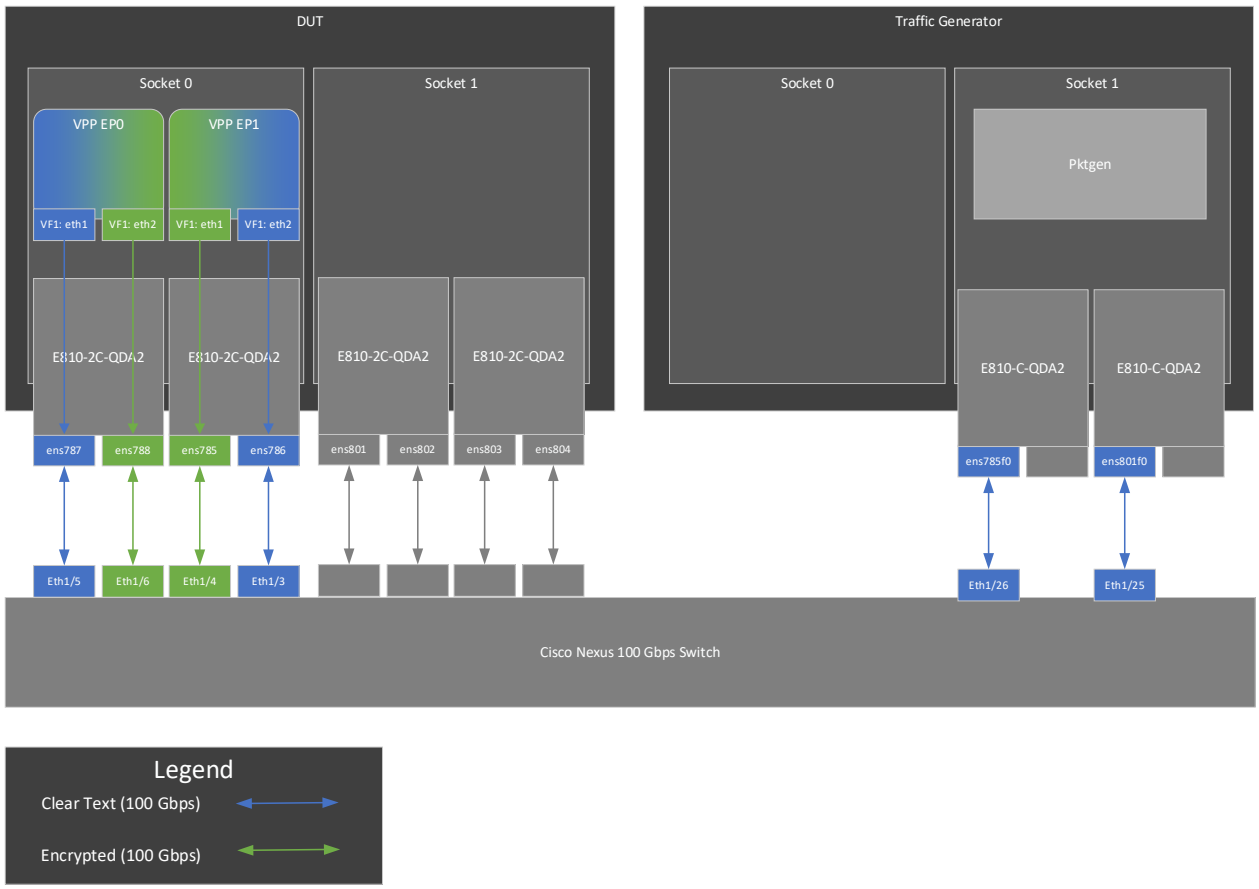

Figure 6. Test Setup - VPP*-IPSec for Secure Transport

Table 11. VPP* Device Under Test Configuration

Component	DUT Configuration
NICs	4x E810-2C-QDA2 (2x E810-2CQDA2 / Socket)
VFs	2x VFs allocated to each VPP* container
VPP* iavf PMD CPUs	9T
VPP* vCPUs	10C/20T allocated to each VPP* container (1C/2T Control Plane, 9C/18T Data Plane, 9T per VF)
Idle Cores/Threads	44C/88T
RXQs/TXQs per VF	9/9
Traffic Generator	Pktgen (200 Gbps TX / Socket, 200 Gbps TX / Server)
Traffic Profile	Maximum Receiver Rate, Clear text IPv4+TCP (16 Flows sent to each VPP* container)

https://www.intel.com/content/www/us/en/products/details/servers/server-systems/server-system-m50cyp.html

4.3.3 Test Results

The figure below displays the aggregate throughput for VPP* IPSec with the AES-128-GCM cipher, with the native VPP* crypto handler for a variety of packet sizes ranging from 64 B up to 1450 B. In this case, performance scales up to approximately 187.5 Gbps with the native crypto handler for a 1450 B packet size. In addition, performance scales up to approximately 40.9 MPPS with the native crypto handler for a 64 B packet size.

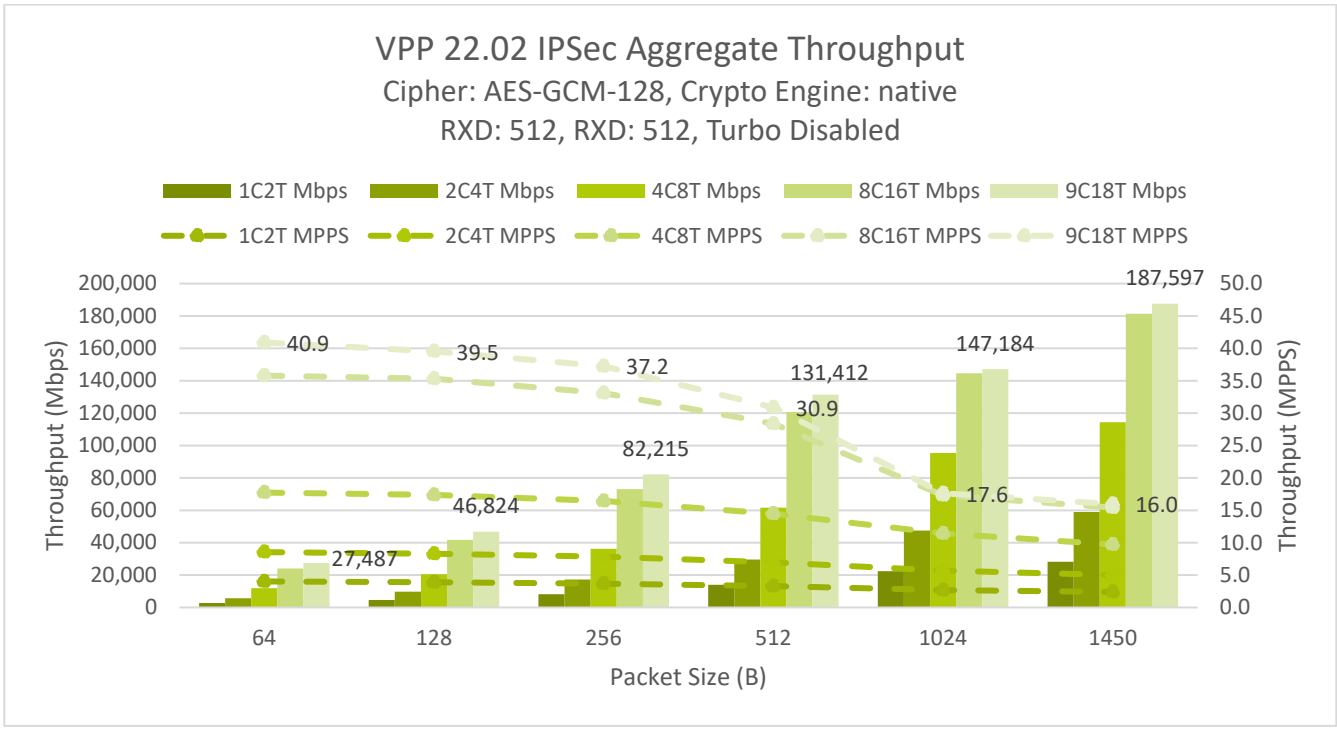

Figure 7. Test Results - IPSec Aggregate Throughput

4.4 NGINX* Application for Web Proxy Applications

4.4.1 Overview

NGINX* is an open source, production grade origin HTTP/HTTPS web server and Content Delivery Network (CDN) web caching application. NGINX* is designed to be orchestrated and deployed at scale, for example as Virtual Machines running in a Red Hat* OpenStack environment, or as containers running in a Red Hat* OpenShift environment. In addition, NGINX*, which supports the Linux TCP/IP stack, supports SR-IOV either as a VNF or CNF via the kernel driver.

4.4.2 Test Setup

The test environment consists of a Device Under Test (DUT) with 4x NGINX* CNFs deployed on a single socket. Each NGINX* origin web server container includes 1x 100 GbE VF, with each VF created from a unique PF. Each NGINX* container reserves 15 threads for a total of 60 threads consumed for the given socket. To ensure that the system is not client bound, 3x traffic generators are included, each with the ability to generate up to 200 Gbps of HTTP requests. The benchmarking tool leveraged in this case is Vegeta, a software based HTTP/HTTPS client, which is deployed baremetal onto each of the traffic generators.

For the purposes of the benchmarks, the Vegeta clients are started simultaneously and each target at most 2x NGINX* origin web servers. For each iteration of the benchmark, the Vegeta clients target a specific object size ranging from 1KiB up to 10 MiB, and also target a specific throughput rate in terms of transactions per second (TPS) ranging from the default of 50 TPS up to 51,200 TPS. For each iteration, the KPI of interest is the achieved aggregate throughput in terms of both TPS and Gbps.

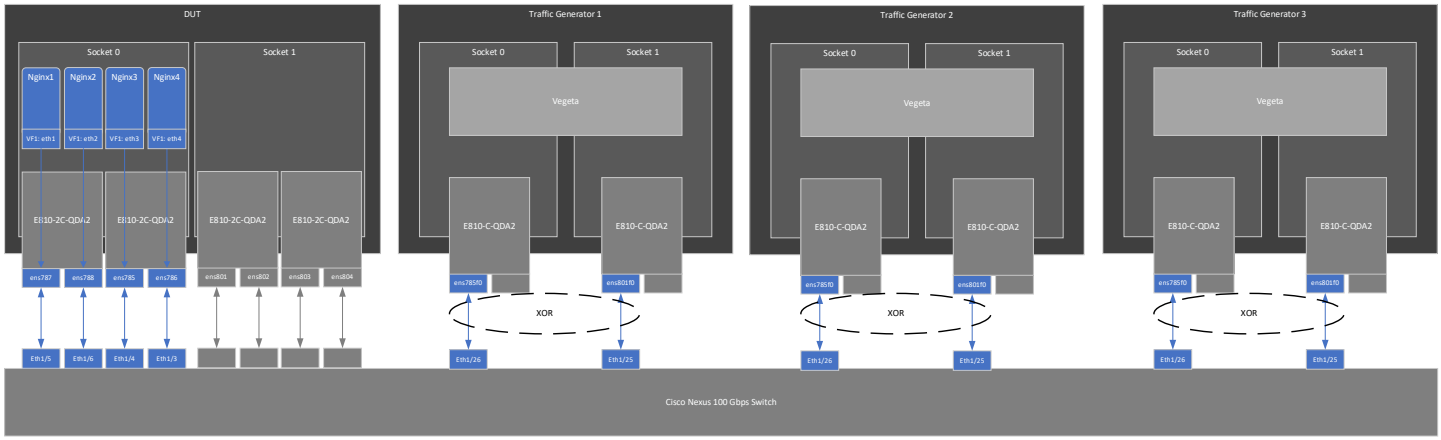

Figure 8. Test Setup - NGINX* Application for Web Proxy Applications

Table 12. NGINX* Device Under Test Configuration

Component	DUT Configuration
NICs	4x E810-2C-QDA2 (2x E810-2C-QDA2 / Socket)
NGINX* Instances	4 / Socket (4 total)
SR-IOV	4x 100 Gbps VFs (1x VF per CNF, 1x VF / PF)
vCPUs	15T per NGINX* CNF (30C/60T total)
Idle Cores/Threads	34C/68T
NGINX* Client	Vegeta (100 Gbps TX / Socket, 200 Gbps TX / Server)
Object Sizes (KiB)	1, 64, 128, 256, 1024, 4096, 8192, 10240

4.4.3 Test Results

The figure below presents the aggregate throughput across all Vegeta clients in terms of Transactions Per Second (TPS). In general, the achievable throughput in TPS decreases as the object size increases. In this case, the aggregate throughput reaches up to approximately 204,423 TPS for a 1 KiB object size.

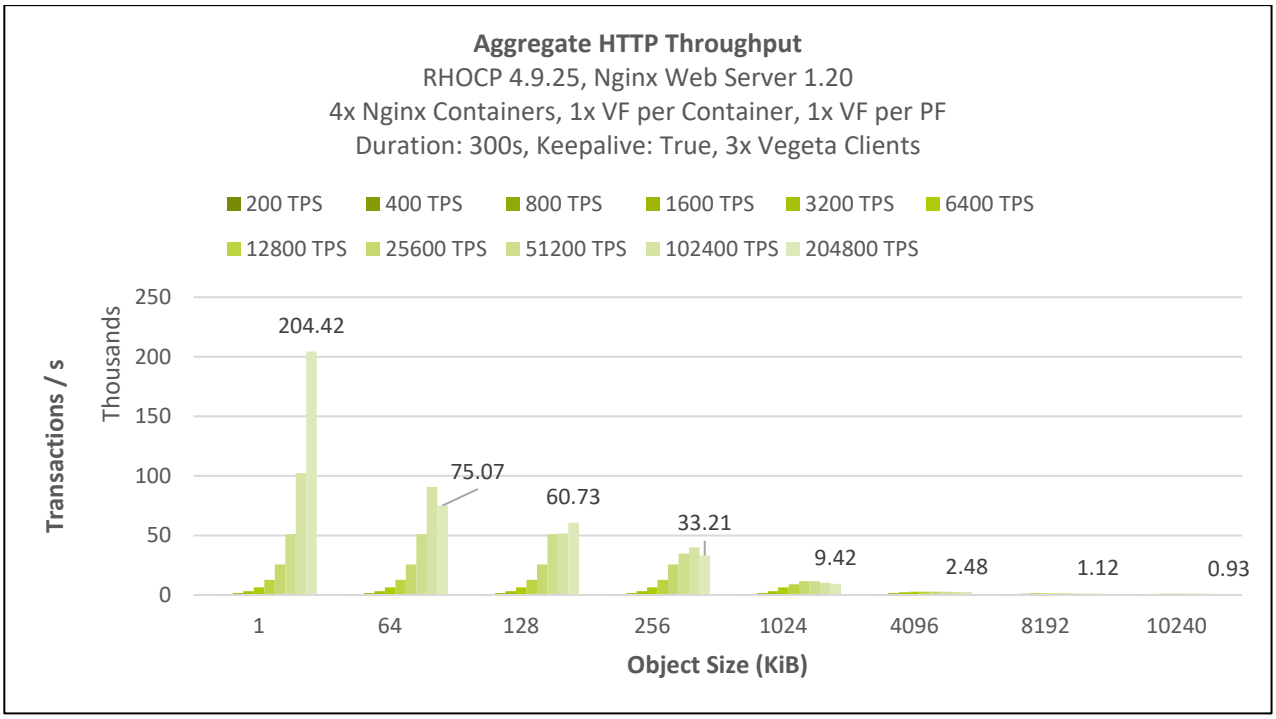

Figure 9. Test Results – Aggregate HTTP Throughput

Appendix A Installation Steps and Scripts

A.1 BIOS Settings

Table A1 provides BIOS settings that are applicable to Compute/Worker/Storage nodes; Table 2 provides detailed settings for BIOS configurations that maximize deterministic performance.

Table 13. System BIOS Settings for Compute/Worker/Storage Nodes

Ingredient	Setting	Recommendation
Intel® Hyper Threading Technology	Enabled	Required
CPU Power/Performance Policy	Performance	Required
CPU Workload Configuration	Balanced	Recommended
Intel® Turbo Boost Technology	Enabled	Recommended
Intel® Speed Shift technology	HWP native	Recommended
Intel® Turbo Boost Technology/Hardware P-State energy performance preference	EPP/EPB settings balanced	Recommended
Three-way mirroring	With the least overhead on processing power	Recommended

Table 14. Advanced BIOS Settings.

Note: Use either column with deterministic performance or Turbo mode enabled in this table to gather performance data required for conformance. Some solutions may not provide the BIOS options as documented in this table. Therefore, for this reference architecture, the BIOS should be set to "Max Performance" profile with Virtualization.

Menu (Advanced)	Path to BIOS Setting	BIOS Setting	Recommendation
Socket Configuration	Processor Configuration	Hyper-Threading	Enable
		X2APIC	Enable
		VMX	Enable
		Uncore frequency scaling	Enable
		Uncore frequency	800MHz to 2.4GHz
	CPU P State Control	EIST PSD Function	HW_ALL
		Boot Performance Mode	Max. Performance
		AVX License Pre-Grant	Disable
		AVX ICCP Pre Grant Level	NA
		AVX P1	Nominal
		Energy Efficient Turbo	Enable
		GPSS timer	0us
		Turbo	Enable
		Intel® SpeedStep®(Pstates) Technology	Enable
	Frequency Prioritization	RAPL Prioritization	Disable
	Hardware PM State Control	Hardware P-States	Native with no Legacy Support
		EPP enable	Disable
	CPU C State Control	Enable Monitor Mwait	Enable
		CPU C1 Auto Demotion	Enable
		CPU C1 Auto unDemotion	Enable
		Processor C6 or CPU C6 Report	Enable
		Enhanced Halt State (C1E)	Enable
		OS ACPI Cx	ACPI C2
	Energy Performance Bias	Power Performance Tuning	OS Controls EPB
		Workload Configuration	Balanced
	Package C State Control	Package C State	C6 Non Retention
		Dynamic L1	Enable
Memory Configuration		Memory Configuration	8-way interleave
		Enforce POR	Enable
		Memory DIMM Refresh Rate	2x
Platform Configuration	Miscellaneous Configuration	Serial Debug Message Level	Minimum
	PCI Express* Configuration	PCIe* ASPM Support	Per Port
	PCI Express* Configuration	PCIe* ASPM	Disable
	PCI Express* Configuration	ECRC generation and checking	Enable
Server Management		Resume on AC Power Loss	Power On
System Acoustic and Performance Configuration		Set Fan Profile	Performance

A.2 OS Configuration

The Bastion node and infrastructure node must use Red Hat* Enterprise Linux (RHEL) 8.4 as the OS. RHEL requires an enabled Red Hat* subscription to run properly.

The Bootstrap, Control Plane, and Compute nodes must use the Red Hat* Enterprise Linux CoreOS (RHCOS) as the OS. RHCOS settings can be modified only in a limited manner. This controlled immutability allows OpenShift Container Platform to store the latest state of RHCOS systems in the cluster, so it is always able to create additional nodes and perform updates based on the latest RHCOS configuration.

A.3 Red Hat* OpenShift Container Platform Configuration

Installing OpenShift Container Platform requires, at a minimum, the following nodes:

- **One Bastion mode.** Used to provision the bootstrap node. The Bastion node is used later to manage the cluster while it is in production use. It hosts the PXE, DHCP, DNS, and HTTP servers.
- **Infrastructure node(s).** For enterprise-level, high-availability deployment of Red Hat* OpenShift Container Platform 4.9, it is recommended to have at least two enterprise-level, commercially supported L4 load balancers, such as those available from NGINX*, F5, or Avi Networks.
- **One Bootstrap node.** Used to deploy OpenShift Container Platform. The Bootstrap node can be reinstalled to become a Compute node after the cluster is installed.
- **Three Control Plane nodes.** Used to manage the Worker nodes and the Kubernetes pods in the cluster.
- **At least two Compute (or worker) nodes.** Worker nodes can be added to or deleted from a cluster if doing so does not compromise the viability of the cluster. At least two viable Worker nodes must always be operating.

Table 15 shows the minimum resource requirements. Installing OpenShift Container Platform with OpenShift Data Foundation requires at least three Storage nodes. Storage can be either provisioned from dedicated nodes or shared with compute services. These nodes can be added later, after cluster deployment.

For more information about OpenShift Container Platform installation, see OpenShift Container Platform 4.9 Documentation. For recommended and validated hardware configuration, see the earlier Implementation Guide section.

Table 15. Minimum Resource Requirements for Red Hat* OpenShift Container Platform 4.9 Nodes

Node	OS	Minimum CPU Cores	RAM	Storage	IOPS
DCI Node	Red Hat* Enterprise Linux 8.4	4	32 GB	180 GB	300
Provisioning Node	Red Hat* Enterprise Linux 8.4	4	16 GB	100 GB	300
Bootstrap VM	Red Hat* Enterprise Linux CoreOS	4	16 GB	100 GB	300
Master Node	Red Hat* Enterprise Linux CoreOS	4	16 GB	100 GB	300
Worker Node	Red Hat* Enterprise Linux CoreOS	2	8 GB	100 GB	300

A.4 VPP* IPSec Container Deployment

The following section provides additional details on how to deploy VPP* IPSec as a CNF on RHOCP 4.9.

A.5 SR-IOV Network Operator Configuration

Install the SR-IOV Network Operator through the OpenShift UI by navigating to “OperatorHub” under the “Operators” section. Within the “Filter by keyword” dialog box enter “SR-IOV Network Operator”. Select the “SR-IOV Network Operator” and click the “Install” button.

Once installed a SriovNetworkNodePolicy will need to be created in order to create and bind Virtual Functions (VFs) on each of the worker nodes. From the OpenShift UI navigate to “Installed Operators” under the “Operators” section. From the project filter drop down box select “All Projects” and click on the “SR-IOV Network Operator”. Confirm the SR-IOV configuration status on each of the worker nodes in the cluster by clicking on the “Sriov Network Node State” tab. Select each worker node and click on the “YAML” tab. From the YAML, review the list of available interfaces, the current driver loaded for each interface, and confirm that the field “totalvfs” for each of the corresponding interfaces is non-zero. In this case, create one Virtual Function per Physical Function, each bound to the vfio-pci driver. Navigate to the “Details” tab for the SR-IOV Network Operator and click on the “Create Instance” button under the “Sriov Network Node Policy” section. In the “Form view” enter the following set of details for each VF. As an example to create a VF from PF ens785 :

- Name: ens785-vf
- Nic Selector
  - Pf Names
    - Value: ens785

- Resource Name: vppep0clearvf
- Device Type: vfio-pci
- Num Vfs: 1
- Mtu:9000

All remaining fields can be left at their default value assignments. Once finished click the “Create” button. From the “Details” tab click on the “Sriov Network Node State” section and confirm the SriovNetworkNodeState status on the corresponding PF to ensure that the each of the VF groups was successfully created.

As an example:

```
…
spec:
  dpConfigVersion: '1933744'
status:
  interfaces:
…
    - driver: ice
      vendor: '8086'
      name: ens785
      linkSpeed: 100000 Mb/s
      mtu: 1500
      mac: 'b4:96:91:93:fc:48'
      deviceID: '1592'
      linkType: ETH
      pciAddress: '0000:4b:00.0'
      totalvfs: 256
    - driver: ice
      vendor: '8086'
      name: ens786
      linkSpeed: 100000 Mb/s
      mtu: 1500
      mac: 'b4:96:91:93:fc:4c'
      deviceID: '1592'
      linkType: ETH
      pciAddress: '0000:4e:00.0'
      totalvfs: 256
…
```

Disable MAC spoof checking and enable trust mode on all of the VFs that have been created via an SriovNetwork custom resource definition. Navigate to the “Details” tab on the SR-IOV Network Operator and click the “Create Instance” button under the “Sriov Network” section. From the “Form view” enter the following set of details for each VF. As an example, to customize the VF from PF ens785:

- Name: vppep0clearvfcrd
- Resource Name: vppep0clearvf
- Trust: on
- Spoof Chk off

Once completed click the “Create” button.

A.6 Container Image

To build the VPP* container image, create the following Dockerfile:

```
FROM ubuntu:20.04

RUN apt-get update && apt-get install -y --no-install-recommends \
                apt-transport-https \
                ca-certificates \
                curl \
                gnupg \
                iproute2 \
                iputils-ping \
        && rm -rf /var/lib/apt/lists/*

ARG REPO
ARG VPP_VERSION

WORKDIR /vpp

COPY get-vpp.sh /get-vpp.sh

RUN set -eux; \
        /get-vpp.sh; \
        apt-get update && apt-get install -y -V ./*.deb; \
        dpkg-query -f '${Version}\n' -W vpp > /vpp/version; \
```

```
        rm -rf vom*.deb vpp-dbg*.deb; \
        rm -rf /var/lib/apt/lists/*;

RUN mkdir -p /var/log/vpp

RUN apt update
RUN apt-get -y install iproute2
RUN apt-get -y install python3
RUN apt-get -y install git
RUN apt-get -y install ethtool
RUN git clone http://dpdk.org/git/dpdk dpdk-21.11 && cd ./dpdk-21.11 && git checkout v21.11
CMD ["/usr/bin/vpp", "-c", "/etc/vpp/startup.conf"]
```

Once completed, from the OpenShift CLI run:

```
oc start-build --from-file=Dockerfile
```

A.7 Pod Configuration

To launch each VPP* instance, create a YAML script that allocates 2x VFs along with 9C/18T. As an example, for VPP* endpoint 0:

```
apiVersion: v1
kind: Pod
metadata:
  name: vpp-ep0-app
  labels:
    app: vpp
  namespace: default
spec:
  securityContext:
    allowPrivilegeEscalation: true
    privileged: true
  containers:
    - name: vpp-ep0
      image: 'image-registry.openshift-image-registry.svc:5000/default/<vpp-image-name>'
      ports:
        - containerPort: 8080
          protocol: TCP
      resources:
        limits:
          cpu: "18"
          openshift.io/vppep0clearvf: "1"
          openshift.io/vppep0encvf: "1"
        requests:
          cpu: "18"
          openshift.io/vppep0clearvf: "1"
          openshift.io/vppep0encvf: "1"
```

A.8 NGINX* Container Deployment

The following section provides additional details on how to deploy NGINX* as a CNF on RHOCP 4.9.

A.8.1 SR-IOV Network Operator Configuration

Install the SR-IOV Network Operator through the OpenShift UI by navigating to “OperatorHub” under the “Operators” section. Within the “Filter by keyword” dialog box enter “SR-IOV Network Operator”. Select the “SR-IOV Network Operator” and click the “Install” button.

Once installed a SriovNetworkNodePolicy will need to be created in order to create and bind Virtual Functions (VFs) on each of the worker nodes. From the OpenShift UI navigate to “Installed Operators” under the “Operators” section. From the project filter drop down box select “All Projects” and click on the “SR-IOV Network Operator”. Confirm the SR-IOV configuration status on each of the worker nodes in the cluster by clicking on the “Sriov Network Node State” tab. Select each worker node and click on the “YAML” tab. From the YAML, review the list of available interfaces, the current driver loaded for each interface, and confirm that the field “totalvfs” for each of the corresponding interfaces is non-zero. In this case, create one Virtual Function per Physical Function, each bound to the vfio-pci driver. Navigate to the “Details” tab for the SR-IOV Network Operator and click on the “Create Instance” button under the “Sriov Network Node Policy” section. In the “Form view” enter the following set of details for each VF. As an example to create a VF from PF ens785 :

- Name: ens785-vf
- Nic Selector
  - Pf Names
    - Value: ens785
- Resource Name: nginx1vf
- Device Type: netdevice
- Num Vfs: 1
- Mtu:9000

All remaining fields can be left at their default value assignments. Once finished click the “Create” button. From the “Details” tab click on the “Sriov Network Node State” section and confirm the SriovNetworkNodeState status on the corresponding PF to ensure that the each of the VF groups was successfully created. As an example:

```
…
spec:
  dpConfigVersion: '1933744'
status:
  interfaces:
…
    - driver: ice
      vendor: '8086'
      name: ens785
      linkSpeed: 100000 Mb/s
      mtu: 1500
      mac: 'b4:96:91:93:fc:48'
      deviceID: '1592'
      linkType: ETH
      pciAddress: '0000:4b:00.0'
      totalvfs: 256
    - driver: ice
      vendor: '8086'
      name: ens786
      linkSpeed: 100000 Mb/s
      mtu: 1500
      mac: 'b4:96:91:93:fc:4c'
      deviceID: '1592'
      linkType: ETH
      pciAddress: '0000:4e:00.0'
      totalvfs: 256
…
```

Configure a static IP subnet for each of the VFs using IPAM. From the “Details” tab for the SR-IOV Network Operator, click on “Create Instance” under the “Sriov Network” section. Enter the appropriate static IP subnet configuration settings under the IPAM section. As an example YAML template for the network associated with the VF attached to interface ens785 :

```
apiVersion: sriovnetwork.openshift.io/v1
kind: SriovNetwork
metadata:
  annotations:
    operator.sriovnetwork.openshift.io/last-network-namespace: default
  resourceVersion: '4457791'
  name: ens785-vf-network
  uid: 1c1c9c2e-ffc7-40f6-87b7-971d7bb6fce1
  creationTimestamp: '2022-02-15T21:27:50Z'
  generation: 2
  managedFields:
    - apiVersion: sriovnetwork.openshift.io/v1
      fieldsType: FieldsV1
      fieldsV1:
        'f:spec':
          .: {}
          'f:ipam': {}
          'f:linkState': {}
          'f:networkNamespace': {}
          'f:resourceName': {}
          'f:spoofChk': {}
          'f:trust': {}
      manager: Mozilla
      operation: Update
      time: '2022-02-15T21:27:50Z'
    - apiVersion: sriovnetwork.openshift.io/v1
      fieldsType: FieldsV1
      fieldsV1:
        'f:metadata':
          'f:annotations':
            .: {}
            'f:operator.sriovnetwork.openshift.io/last-network-namespace': {}
          'f:finalizers':
            .: {}
            'v:"netattdef.finalizers.sriovnetwork.openshift.io"': {}
      manager: sriov-network-operator
      operation: Update
      time: '2022-02-15T21:27:50Z'
  namespace: openshift-sriov-network-operator
  finalizers:
    - netattdef.finalizers.sriovnetwork.openshift.io
spec:
```

```
  ipam: >-
    {  "type": "host-local",  "subnet": "172.16.4.0/24",  "rangeStart":
    "172.16.4.2",  "rangeEnd": "172.16.4.11",  "routes": [{    "dst":
    "0.0.0.0/0"  }],  "gateway": "172.16.4.1"}
  linkState: auto
  networkNamespace: default
  resourceName: nginx_dataplane
  spoofChk: 'on'
  trust: 'off'
```

Enable MAC spoof checking and disable trust mode on all of the VFs that have been created via an SriovNetwork custom resource definition. Navigate to the "Details" tab on the SR-IOV Network Operator and click the "Create Instance" button under the "Sriov Network" section. From the "Form view" enter the following set of details for each VF. As an example, to customize the VF from PF ens785:

- Name: nginx1vfcrd
- Resource Name: nginx1vf
- Trust: off
- Spoof Chk on

Once completed click the "Create" button.

A.8.2 Container Image

To build the NGINX* container image, create the following Dockerfile:

```
FROM registry.access.redhat.com/ubi8/ubi-init

ENV NAME=nginx \
    NGINX_VERSION=1.20 \
    NGINX_SHORT_VER=120 \
    VERSION=0

ENV SUMMARY="Platform for running nginx $NGINX VERSION or building nginx-based application" \
    DESCRIPTION="Nginx is a web server and a reverse proxy server for HTTP, SMTP, POP3 and IMAP \
protocols, with a strong focus on high concurrency, performance and low memory usage. The container \
image provides a containerized packaging of the nginx $NGINX_VERSION daemon. The image can be used \
as a base image for other applications based on nginx $NGINX_VERSION web server. \
Nginx server image can be extended using source-to-image tool."

LABEL summary="${SUMMARY}" \
      description="${DESCRIPTION}" \
      io.k8s.description="${DESCRIPTION}" \
      io.k8s.display-name="Nginx ${NGINX_VERSION}" \
      io.openshift.expose-services="8080:http" \
      io.openshift.expose-services="8443:https" \
      io.openshift.tags="builder,${NAME},${NAME}-${NGINX_SHORT_VER}" \
      com.redhat.component="${NAME}-${NGINX_SHORT_VER}-container" \
      name="ubi8/${NAME}-${NGINX_SHORT_VER}" \
      version="1" \
      com.redhat.license_terms="https://www.redhat.com/en/about/red-hat-end-user-license-agreements#UBI"
\
      maintainer="SoftwareCollections.org <sclorg@redhat.com>" \
      help="For more information visit https://github.com/sclorg/${NAME}-container" \
      usage="s2i build <SOURCE-REPOSITORY> ubi8/${NAME}-${NGINX_SHORT_VER}:latest <APP-NAME>"

ENV NGINX_CONFIGURATION_PATH=${APP_ROOT}/etc/nginx.d \
    NGINX_CONF_PATH=/etc/nginx/nginx.conf \
    NGINX_DEFAULT_CONF_PATH=${APP_ROOT}/etc/nginx.default.d \
    NGINX_CONTAINER_SCRIPTS_PATH=/usr/share/container-scripts/nginx \
    NGINX_APP_ROOT=${APP_ROOT} \
    NGINX_LOG_PATH=/var/log/nginx \
    NGINX_PERL_MODULE_PATH=${APP_ROOT}/etc/perl

RUN yum -y install iproute
RUN yum -y install ethtool
RUN yum -y install pciutils

RUN yum -y module enable nginx:$NGINX_VERSION && \
    INSTALL_PKGS="nss_wrapper bind-utils gettext hostname nginx nginx-mod-stream nginx-mod-http-perl" &&
\
    yum install -y --setopt=tsflags=nodocs $INSTALL_PKGS && \
    rpm -V $INSTALL_PKGS && \
    yum -y clean all --enablerepo='*'

CMD nginx -g "daemon off;"
```

Once completed, from the OpenShift CLI run:

```
oc start-build --from-file=Dockerfile
```

A.8.3 Pod Configuration

To launch each NGINX* instance, create a YAML script that allocates 1x VF along with 15T. As an example, for NGINX* instance 1:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-app
  labels:
    app: nginx
  namespace: default
  annotations:
    k8s.v1.cni.cncf.io/networks: ens785-vf-network
spec:
  securityContext:
    allowPrivilegeEscalation: true
    privileged: true
  containers:
    - name: nginx1
      image: 'image-registry.openshift-image-registry.svc:5000/default/<nginx-image-name>'
      ports:
        - containerPort: 8080
          protocol: TCP
      resources:
        limits:
          cpu: "15"
          openshift.io/nginx1vf: "1"
        requests:
          cpu: "15"
          openshift.io/nginx1vf: "1"
```

A.9 Bare-Metal Installation

A.9.1 User-Provisioned Infrastructure Configuration

Before installing a OpenShift Container Platform cluster, the underlying infrastructure must be provided. Follow these steps, which are essential to install OpenShift Container Platform:

1. Provide a static IP address configuration for each node in the cluster to establish a network connection between all nodes in the cluster.
2. Configure load balancers. Here is an example configuration file for HAProxy*:

```
frontend openshift-api-server-onprem1
    bind 172.30.4.111:6443
    default_backend openshift-api-server-onprem1
    mode tcp
    option tcplog
frontend machine-config-server-onprem1
    bind 172.30.4.111:22623
    default_backend machine-config-server-onprem1
    mode tcp
    option tcplog
frontend ingress-http-onprem1
    bind 172.30.4.111:80
    default_backend ingress-http-onprem1
    mode tcp
    option tcplog
frontend ingress-https-onprem1
    bind 172.30.4.111:443
    default_backend ingress-https-onprem1
    mode tcp
    option tcplog
backend openshift-api-server-onprem1
    balance source
    mode tcp
    server bootstrap 172.30.4.104:6443 check
    server master1 172.30.4.101:6443 check
    server master2 172.30.4.102:6443 check
    server master3 172.30.4.103:6443 check
backend machine-config-server-onprem1
    balance source
    mode tcp
    server bootstrap 172.30.4.104:22623 check
    server master1 172.30.4.101:22623 check
    server master2 172.30.4.102:22623 check
```

```
    server master3 172.30.4.103:22623 check
backend ingress-http-onprem1
    balance source
    mode tcp
    server master1 172.30.4.101:80 check
    server master2 172.30.4.102:80 check
    server master3 172.30.4.103:80 check
    server worker1 172.30.4.104:80 check
    server worker2 172.30.4.105:80 check
backend ingress-https-onprem1
    balance source
    mode tcp
    server master1 172.30.4.101:443 check
    server master2 172.30.4.102:443 check
    server master3 172.30.4.103:443 check
    server worker1 172.30.4.104:443 check
    server worker2 172.30.4.105:443 check
```

3. Provide a DNS configuration for each node in the cluster and required OpenShift DNS records. The example below shows the configuration file for Dnsmasq*:

```
# main DNS entries for apps & infra - OpenShift-specific
address=/api.onprem1.ocp.public/api.onprem1.ocp.public/172.30.4.111
address=/api-int.onprem1.ocp.public/api-int.onprem1.ocp.public/172.30.4.111
address=/lb.onprem1.ocp.public/lb.onprem1.ocp.public/172.30.4.111
address=/etcd-0.onprem1.ocp.public/etcd-0.onprem1.ocp.public/172.30.4.101
address=/etcd-1.onprem1.ocp.public/etcd-1.onprem1.ocp.public/172.30.4.102
address=/etcd-2.onprem1.ocp.public/etcd-2.onprem1.ocp.public/172.30.4.103
srv-host=_etcd-server-ssl._tcp.onprem1.ocp.public,etcd-0.onprem1.ocp.public,2380,0,10
srv-host=_etcd-server-ssl._tcp.onprem1.ocp.public,etcd-1.onprem1.ocp.public,2380,0,10
srv-host=_etcd-server-ssl._tcp.onprem1.ocp.public,etcd-2.onprem1.ocp.public,2380,0,10
# DNS entries for OpenShift hosts
address=/master1.onprem1.ocp.public/master1.onprem1.ocp.public/172.30.4.101
address=/master2.onprem1.ocp.public/master2.onprem1.ocp.public/172.30.4.102
address=/master3.onprem1.ocp.public/master3.onprem1.ocp.public/172.30.4.103
address=/ocs1.onprem1.ocp.public/ocs1.onprem1.ocp.public/172.30.4.107
address=/ocs2.onprem1.ocp.public/ocs2.onprem1.ocp.public/172.30.4.108
address=/ocs3.onprem1.ocp.public/ocs3.onprem1.ocp.public/172.30.4.109
address=/worker1.onprem1.ocp.public/worker1.onprem1.ocp.public/172.30.4.104
address=/worker2.onprem1.ocp.public/worker2.onprem1.ocp.public/172.30.4.105
```

4. Configure the ports for your nodes and verify that the ports shown in the following table were opened:

Table 16. Firewall Port Configuration Requirements

All Nodes to All Nodes

Protocol	Port	Description
ICMP	N/A	Network reachability tests
TCP	9000-9999	Host-level services, including the node exporter on ports 9100-9101 and the Cluster Version Operator on port 9099
	10250-10259	The default ports that Kubernetes reserves
	10256	openshift-sdn
UDP	4789	VXLAN and GENEVE
	6081	VXLAN and GENEVE
	9000-9999	Host-level services, including the node exporter on ports 9100-9101
TCP/UDP	30000-32767	Kubernetes NodePort

All Nodes to the Control Plane

Protocol	Port	Description
TCP	2379-2380	etcd server, peer, and metrics ports
	6443	Kubernetes API

A.10 Creating the OpenShift Manifest and Ignition Configuration Files

1. Provide an install-config.yaml file. Here is an example file:

```
apiVersion: v1
baseDomain: example.com
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 2
controlPlane:
  hyperthreading: Enabled
  name: onprem1
  replicas: 3
metadata:
  name: cluster
networking:
  clusterNetworks:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  none: {}
fips: false
pullSecret: '{"auths": …}'
sshKey: 'ssh-ed25519 AAAA…'
```

2. Next, generate the OpenShift manifests for the cluster:

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

3. Modify the <installation_directory>/manifests/cluster-scheduler-02-config.yml OpenShift manifest file to prevent pods from being scheduled on the Control Plane nodes. Open the <installation_directory>/manifests/cluster-scheduler-02-config.yml file and locate the mastersSchedulable parameter. Set its value to False.
4. Then, generate the ignition configuration files:

```
$ ./openshift-install create ignition-configs --dir=<installation_directory>
```

A.11 Creating Red Hat* Enterprise Linux CoreOS Nodes

Red Hat* Enterprise Linux CoreOS-ready nodes must be provided before installing the OpenShift Container Platform cluster. The Red Hat* Enterprise Linux CoreOS nodes can be created using ISO image or network PXE booting.

A.12 Installing a Bare-Metal Red Hat* OpenShift Container Platform

1. Call the installer to check the cluster creation process:

```
$ ./openshift-install wait-for bootstrap-complete --dir=<installation_directory> //
--log-level=info
```

2. To finish the installation process, all certificate signing requests must be approved.
3. After successful installation, the oc get nodes command should return a list of nodes that are ready to work:

```
$ oc get nodes
NAME                                STATUS  ROLES   AGE     VERSION
cluster1-2jlqr-master-0             Ready   master  46m     v1.18.3+6c42de8
cluster1-2jlqr-master-1             Ready   master  46m     v1.18.3+6c42de8
cluster1-2jlqr-master-2             Ready   master  45m     v1.18.3+6c42de8
cluster1-2jlqr-worker-westeurope1-jjmkp  Ready    worker  24m     v1.18.3+6c42de8
cluster1-2jlqr-worker-westeurope1-kwzr6  Ready    worker  25m     v1.18.3+6c42de8
cluster1-2jlqr-worker-westeurope2-41rjg  Ready    worker  25m     v1.18.3+6c42de8
cluster1-2jlqr-worker-westeurope2-z7pdt  Ready    worker  23m     v1.18.3+6c42de8
cluster1-2jlqr-worker-westeurope2-mjmfd  Ready    worker  24m     v1.18.3+6c42de8
cluster1-2jlqr-worker-westeurope2-zmf21  Ready    worker  25m     v1.18.3+6c42de8
```

SOLUTION BRIEF ENDNOTES

IMPLEMENTATION GUIDE ENDNOTES

Notices & Disclaimers

Intel technologies may require enabled hardware, software, or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.