USER GUIDE

Intel Corporation

Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor and Intel® Xeon® D Processor Reference Architecture User Guide Release V22.01

Authors

Francis Cahill

Octavia Carotti

Tom Counihan

Dave Cremins

Calin Gherghe

Joel A. Gibson

Shivapriya Hiremath

Radoslaw Jablonski

Konrad Jankowski

Veronika Karpenko

Patryk Klimowicz

Krystian Mlynek

Dana Nehama

Michael O'Reilly

Seungweon Park

Dan Parsons

Michael Pedersen

Dmitrii Puzikov

Syed Faraz Ali Shah

Mark Liu

1 Introduction

The **Container Bare Metal Reference Architecture (BMRA)** is part of the Network and Cloud Edge Reference System Architectures portfolio. It is a cloud-native, forward-looking Kubernetes-cluster template solution for network implementations.

This document contains instructions on installation, configuration, and use of networking and device plug-in features for setting the BMRA Release 22.01 in multiple deployment scenarios. By following this document, it is possible to set up a Kubernetes cluster and configure it automatically using open-source Ansible playbooks provided by Intel for optimized deployments across network locations and use cases. The playbooks enable users to perform automated, predicted deployments for decreasing installation time from days to hours.

The document provides the following information to enable best-practice deployments:

- Part 1 (Sections 2 – 5): Details about the supported hardware and software components and the Ansible scripts implementation.
- Part 2 (Sections 6 – 12): Step-by-step instructions on how to build your BMRA flavor using the Ansible scripts. **If you wish to start building your BMRA right away, you may directly go these sections and start automatically provisioning the BMRA configuration of your choice.**
- Part 3 (Section 13): BMRA application examples.
- Part 4 (Appendix A): The BMRA Release Notes.
- Part 5 (Appendix B): Abbreviations

The hardware platforms supported by BMRA 21.01 are based on the 2nd and **3rd Generation Intel® Xeon® Scalable processors**, **Intel® Xeon® D processors**, Intel® Ethernet Controller, Intel® QuickAssist Technology (Intel® QAT), Intel® Server GPU, and other advanced platform technologies.

The software components supported are comprised of open-source Cloud Native software delivered by Intel and its partners in CNCF communities (e.g. Kubernetes, Telegraf, Istio, FD.io) and by Intel forward-looking innovation for overcoming barriers to networking adoption in Cloud. For details, refer to the *Network and Cloud Edge Reference System Architectures Portfolio User Manual*.

This guide provides step-by-step instructions for delivering multiple BMRA flavors. The generation of each BMRA flavor is completely automated, using an open-source Ansible script that is defined by a Configuration Profile. Network locations impose different hardware, software, and configuration specifications due to varying workloads, cost, density, and performance requirements. A Configuration Profile determines how a BMRA flavor should be optimally deployed given those network and workload requirements.

BMRA Configuration Profiles are network location-specific:

- **On-Premises Edge Configuration Profile** – Typical Customer Premises deployment supporting, for example, Content Delivery Network (CDN) and Smart City scenarios.
- **Remote Central Office-Forwarding Configuration Profile** – Near Edge deployments supporting fast packet-forwarding workloads such as Cable Modem Termination System (CMTS), User Plane Function (UPF) and Application Gateway Function (AGF).

- **Regional Data Center Configuration Profile** – Central-office location typical Configuration Profile. Currently tailored exclusively for Media Visual Processing workloads such as CDN Transcoding.

Configuration Profiles that are not location-specific to enable flexible deployments:

- **Basic Configuration Profile** – Minimum BMRA Kubernetes cluster setup.
- **Full Configuration Profile** – Complete BMRA setup based on all software features.
- **Storage Configuration Profile** – A BMRA flavor supporting MinIO Object Storage.

Following are the primary new and updated hardware ingredients and software features introduced in Release 22.01. For additional details, refer to the BMRA Release Notes.

Hardware Platform:

- Support Intel Xeon D-1700 and D-2700 processors for public usage
- Support Taylors Falls Reference Design for Intel Xeon D-1700
- Updated Intel® Ethernet 700 and 800 Network Adapter drivers

Configuration Profiles:

- Introduce Storage Configuration Profile to support high-performance object storage with MinIO operator/console
- Support for rendering BMRA Configuration Profile files from template

Security:

- Updated Intel® Software Guard Extensions (Intel® SGX) Software Development Kit
- Updated Key Management Reference Application version to 1.4

Service Mesh:

- Microservices architectures with Istio service mesh (using istioctl for deployment) and Envoy integrated with KMRA version 1.4

Packet Processing

- Updated to latest DPDK LTS 21.11

Power Management Kubernetes Operator:

- Support for Power Manager, a Kubernetes Operator, designed to expose and utilize Intel-specific power management technologies in a Kubernetes environment

Kubernetes:

- Updated Kubernetes to v1.22.3
- Updated Node Feature Discovery (NFD)
- Support for Platform Aware Scheduling, adding to Telemetry Aware Scheduling, GPU Aware Scheduling
- Removed support for CPU Manager for Kubernetes (CMK)
- Updated other SW versions

Experience Kits, the collaterals that explain in detail the technologies enabled in BMRA release 22.01, including benchmark information, are available in the following locations:

Intel Network Builder:

- https://networkbuilders.intel.com/intel-technologies/network-transformation-exp-kits
- https://networkbuilders.intel.com/intel-technologies/container-experience-kits

Table of Contents

Figures

Tables

Document Revision History

Three previous editions of the BMRA document were released, starting April 2019.

- Covered 2nd Generation Intel® Xeon® Scalable processors
- Covered 2nd Generation and 3rd Generation Intel® Xeon® Scalable processors
- Covered 2nd Generation and 3rd Generation Intel® Xeon® Scalable processors and Intel® Xeon® D processor

REVISION	DATE	DESCRIPTION
001	February 2022	Initial release.
002	March 2022	Updated a few URLs.

1.1 Terminology

Here are some key concepts and terminology used throughout this document:

- A **Reference Architecture** provides a template solution for a specific implementation, typically using industry best practices and the latest technology developments.
- **Container Bare Metal Reference Architecture (BMRA)** – A Reference Architecture that implements a container bare metal deployment model of a Kubernetes cluster. This guide provides an Intel implementation of a Kubernetes cluster that supports containers on a bare metal platform using open-source software.
- **Reference Architecture Configuration Profile** – A Configuration Profile defines (1) a specific set of hardware ingredients used to build a Kubernetes cluster (the hardware BOM), (2) software modules and capabilities that enable the system design (the software BOM), and (3) specific configuration of those hardware and software elements. In this document, we discuss BMRA Configuration Profiles that have been defined and optimized per network location.
- A **Reference Architecture Flavor** is an instance of a Reference Architecture generated by implementing a Configuration Profile specification. In this document, we discuss a set of BMRA flavors defined per network location.
- **Reference Architecture Ansible Playbook** – A set of scripts that prepare, configure, and deploy your Kubernetes cluster. This document discusses Reference Architecture Ansible scripts designed for BMRA. The Ansible scripts implement the BMRA Configuration Profiles and generate BMRA flavors per network location.

1.2 Reference Documentation

The *Network and Cloud Edge Reference System Architectures Portfolio User Manual* contains a complete list of reference documents.

Collaterals, including technical guides and solution briefs that explain in detail the technologies enabled in BMRA release 22.01, are available in the following locations: https://networkbuilders.intel.com/intel-technologies/network-transformation-exp-kits and https://networkbuilders.intel.com/intel-technologies/container-experience-kits.

Table 1. Reference Documents

REFERENCE	SOURCE
Network and Cloud Edge Reference System Architectures Portfolio User Manual	https://networkbuilders.intel.com/solutionslibrary/reference-system-architectures-portfolio-user-manual
Virtual Machine for 2nd Generation and 3rd Generation Intel Xeon Scalable Processor Reference Architecture User Guide	https://networkbuilders.intel.com/solutionslibrary/virtual-machine-for-2nd-generation-and-3rd-generation-intel-xeon-scalable-processor-reference-architecture-user-guide

Part 1:
Reference Architecture Components and Deployment Guidelines:
Ansible Playbooks
Hardware Components
Software Ingredients
Recommended Configurations

2 Reference Architecture Deployment

This chapter explains how a BMRA flavor is generated and deployed. The process includes installation of the hardware setup followed by system provisioning using Ansible playbook.

2.1 BMRA Architecture

The BMRA is a Kubernetes cluster that can be configured to support a flexible number of control nodes and worker nodes. To deploy the BMRA the following elements must be defined and configured:

1. Hardware components
2. Software ingredients
3. Specific hardware and software configuration parameters
4. Configuration Profile that describes the combined hardware, software, and configuration setup
5. Ansible playbooks that implement the BMRA per the Configuration Profile definition

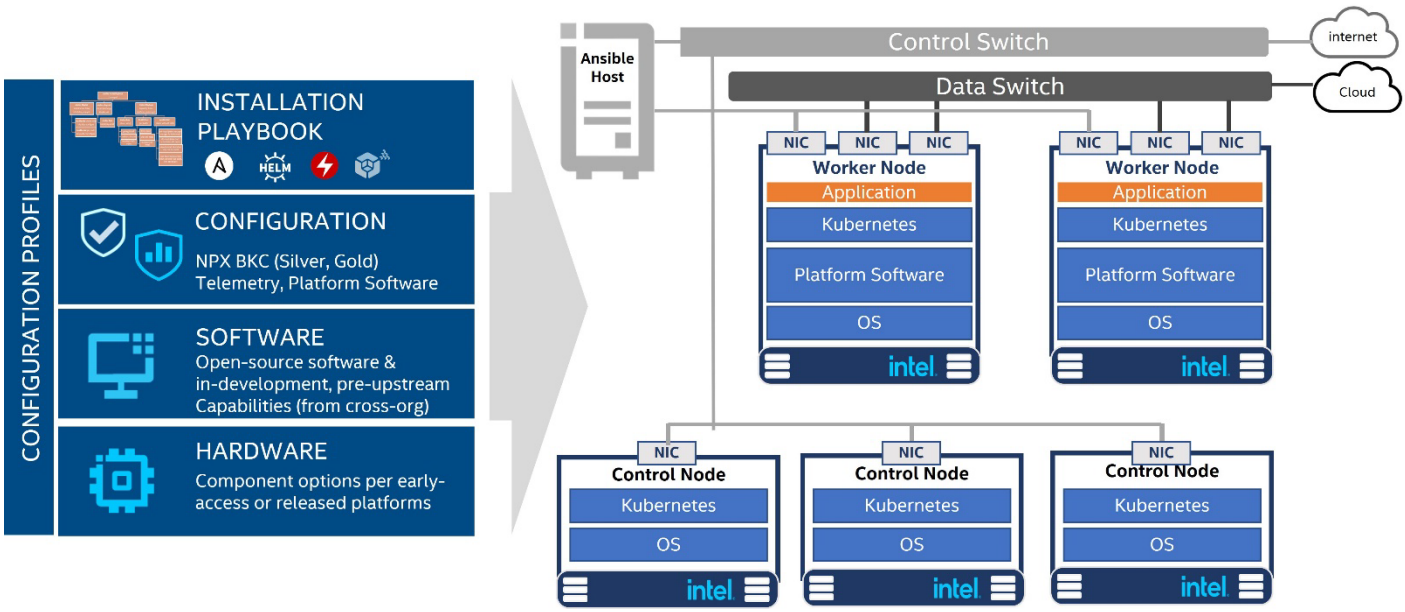

Figure 1. Container Bare Metal Reference Architecture (BMRA) Illustration and Applicable Elements

2.2 Configuration Profiles

Network deployments vary by location. Each location imposes different hardware and software specifications and configurations due to varying workloads, cost, density, and performance requirements.[1] To address these matters, the BMRA supports the concept of Configuration Profiles, each of which determines how a BMRA flavor can be generated. BMRA 22.01 continues to support the following Reference Architecture Configuration Profiles.

Three of the Reference Architecture Configuration Profiles are network location-specific:

- **On-Premises Edge Configuration Profile** – Small cluster of stationary or mobile server platforms, ranging from one to four servers. Usage scenarios include data collection from sensors, local (edge) processing, and upstream data transmission. Sample locations are hospitals, factory floors, law enforcement, media, cargo transportation, power utilities. This Configuration Profile recommends a Kubernetes cluster hardware configuration, software capabilities, and specific hardware and software configurations that typically support enterprise edge workloads used in SMTC deployments, CDN, and Ad-insertion.
- **Remote Central Office-Forwarding Configuration Profile** – Clusters ranging from a half rack to a few racks of servers, typically in a pre-existing, repurposed, unmanned structure. The usage scenarios include running latency-sensitive applications near the user (for example, real-time gaming, stock trading, video conferencing). This Configuration Profile addresses a Kubernetes cluster hardware, software capabilities, and configurations that enable high performance for packet forwarding packets. In this category, you can find workloads such as UPF, vBNG, vCMTS, and vCDN.

[1] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

- **Regional Data Center Configuration Profile** – The Regional Data Center consists of a management domain with many racks of servers, typically managed and orchestrated by a single instance of resource orchestration. Usage scenarios include services such as content delivery, media, mobile connectivity, and cloud services. This Configuration Profile is tailored exclusively and defined for Media Visual Processing workloads such as CDN Transcoding.

Three additional Reference Architecture Configuration Profiles that are not location-specific enable flexible deployments per need:

- **Basic Configuration Profile** – Minimum BMRA Kubernetes cluster setup.
- **Full Configuration Profile** – Complete BMRA setup supporting all available software features. This profile is targeting developers and deployers that are looking to evaluate, control, and configure the software and hardware ingredients and dependencies.
- **Storage Configuration Profile** – Kubernetes-native high-performance object store supports deploying MinIO tenants onto private and public cloud infrastructures ("Hybrid" Cloud). The API is also compatible with Amazon Web Services Simple Storage Service (S3).

2.3 Reference Architecture Installation Prerequisites

Before the Ansible playbook can begin, you must identify the required hardware components, ensure hardware connectivity, and complete the initial configuration, for example BIOS setup. This section helps you get ready for your Ansible installation.

This section describes the minimal system prerequisites needed for the Ansible Host and Kubernetes control nodes and worker nodes. It also provides a list of steps required to prepare hosts for successful deployment. These instructions include:

- Hardware BOM selection and setup
- Required BIOS/UEFI configuration, including virtualization and hyper-threading settings
- Network topology requirements – a list of necessary network connections between the nodes
- Installation of software dependencies needed to execute Ansible playbooks
- Generation and distribution of SSH keys that are used for authentication between Ansible Host and Kubernetes cluster target servers

After satisfying these prerequisites, Ansible playbooks for 2nd Generation Intel Xeon Scalable processors, 3rd Generation Intel Xeon Scalable processors, and Intel Xeon D processors can be downloaded directly from the dedicated GitHub page (https://github.com/intel/container-experience-kits/releases) or cloned using the Git.

2.3.1 Hardware BOM Selection and Setup for Control and Worker Nodes

Before software deployment and configuration of BMRA, administrators must deploy the physical hardware infrastructure for their site. To obtain ideal performance and latency characteristics for a given network location, Intel recommends the following hardware configurations:

- Control Nodes - Review Section 3.1 for recommended Control node assembly.
- Worker Nodes – Refer to the following sections for recommended Worker node assembly:
  - Base Worker Node – Review Section 3.2 to satisfy base performance characteristics.
  - Plus Worker Node – Review Section 3.3 to satisfy plus performance characteristics.

Section 7, BMRA Basic Configuration Profile Setup, Section 8, BMRA Full Configuration Profile Setup, Section 9, BMRA On-Premises Edge Configuration Profile Setup, Section 10, BMRA Remote Central Office-Forwarding Configuration Profile Setup, Section 11, BMRA Regional Data Center Configuration Profile Setup, and Section 12, Storage Configuration Profile Setup contain details about the hardware BOM selection and setup.

2.3.2 BIOS Selection for Control and Worker Nodes

Enter the UEFI or BIOS menu and update the configuration as listed in Section 6 and Table 20, which describe the BIOS selection in detail.

2.3.3 Operating System Selection for Ansible Host, Control, and Worker Nodes

The following Linux operating systems are supported for Control and Worker Nodes:

- RHEL for x86_64 Version 8 (8.5)
- Ubuntu 20.04 LTS (20.04.3)
- Ubuntu 21.04 (21.04)

For all supported distributions, the base images are sufficient when built using the "Minimal" option during installation. In addition, the following must be met:

- The Control and Worker Nodes must have network connectivity to the Ansible Host.
- SSH connections are supported. If needed, on Ubuntu, install SSH Server with the following commands (internet access is required):

```
# sudo apt update
# sudo apt install openssh-server
```

2.3.4 Network Interface Requirements for Control and Worker Nodes

The following list provides a brief description of different networks and network interfaces needed for deployment.

- Internet network
  - Ansible Host accessible
  - Capable of downloading packages from the internet
  - Can be configured for Dynamic Host Configuration Protocol (DHCP) or with static IP address
- Management network and Calico pod network interface (This can be a shared interface with the internet network)
  - Kubernetes control and worker node inter-node communications
  - Calico pod network runs over this network
  - Configured to use a private static address
- Tenant data networks
  - Dedicated networks for traffic
  - SR-IOV enabled
  - Virtual function (VF) can be DPDK bound in pod

2.4 Ansible Playbook

The Reference Architecture is configured and provisioned automatically using Ansible scripts[2]. Each Configuration Profile has a dedicated Ansible playbook. This section describes how the Ansible playbooks allow for an automated deployment of a fully functional BMRA cluster, including initial system configuration, Kubernetes deployment, and setup of capabilities as described in Section 2.5.

2.4.1 Ansible Playbook Building Blocks

The following components make up the BMRA Ansible playbooks.

Note: Ansible playbooks for 2nd Generation Intel Xeon Scalable processors, 3rd Generation Intel Xeon Scalable processors, and Intel® Xeon® D processors are open source and available here.

Configuration Files provide examples of cluster-wide and host-specific configuration options for each of the Configuration Profiles. With minimal changes these Configuration Files can be used directly with their corresponding playbooks.

For default values refer to the Configuration Profile-specific sections: BMRA Basic Configuration Profile Setup; BMRA Full Configuration Profile Setup; BMRA On-Premises Edge Configuration Profile Setup; BMRA Remote Central Office-Forwarding Configuration Profile Setup; BMRA Regional Data Center Configuration Profile Setup, and Storage Configuration Profile Setup.

- `inventory.ini`
- `group_vars/all.yml`
- `host_vars/node1.yml`

Ansible Playbooks act as a user entry point and include all relevant Ansible roles and Helm charts. Top-level Ansible playbooks exist for each Configuration Profile, which allows lean use case-oriented cluster deployments. Each playbook includes only the Ansible roles and configuration files that are relevant for a given use case. See High Level Ansible Playbooks in Figure 2.

- `playbooks/basic.yml`
- `playbooks/full_nfv.yml`
- `playbooks/on_prem.yml`
- `playbooks/remote_fp.yml`
- `playbooks/regional_dc.yml`
- `playbooks/storage.yml`

Additionally, a Cluster Removal Playbook exists to optionally remove an existing cluster in case user wants to try different deployment models.

- `playbooks/redeploy_cleanup.yml`

Each of these playbooks encompasses **Ansible Roles** grouped into three main execution phases, which are depicted in Figure 2 and further explained in the next section:

- Infrastructure Setup
- Kubernetes Deployment
- Capabilities Setup

[2] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

Note that several Capabilities Setup roles include nested Helm charts for easier deployment and lifecycle management of deployed applications as well as a group of **Common Utility Roles** that provide reusable functionality across the playbooks.

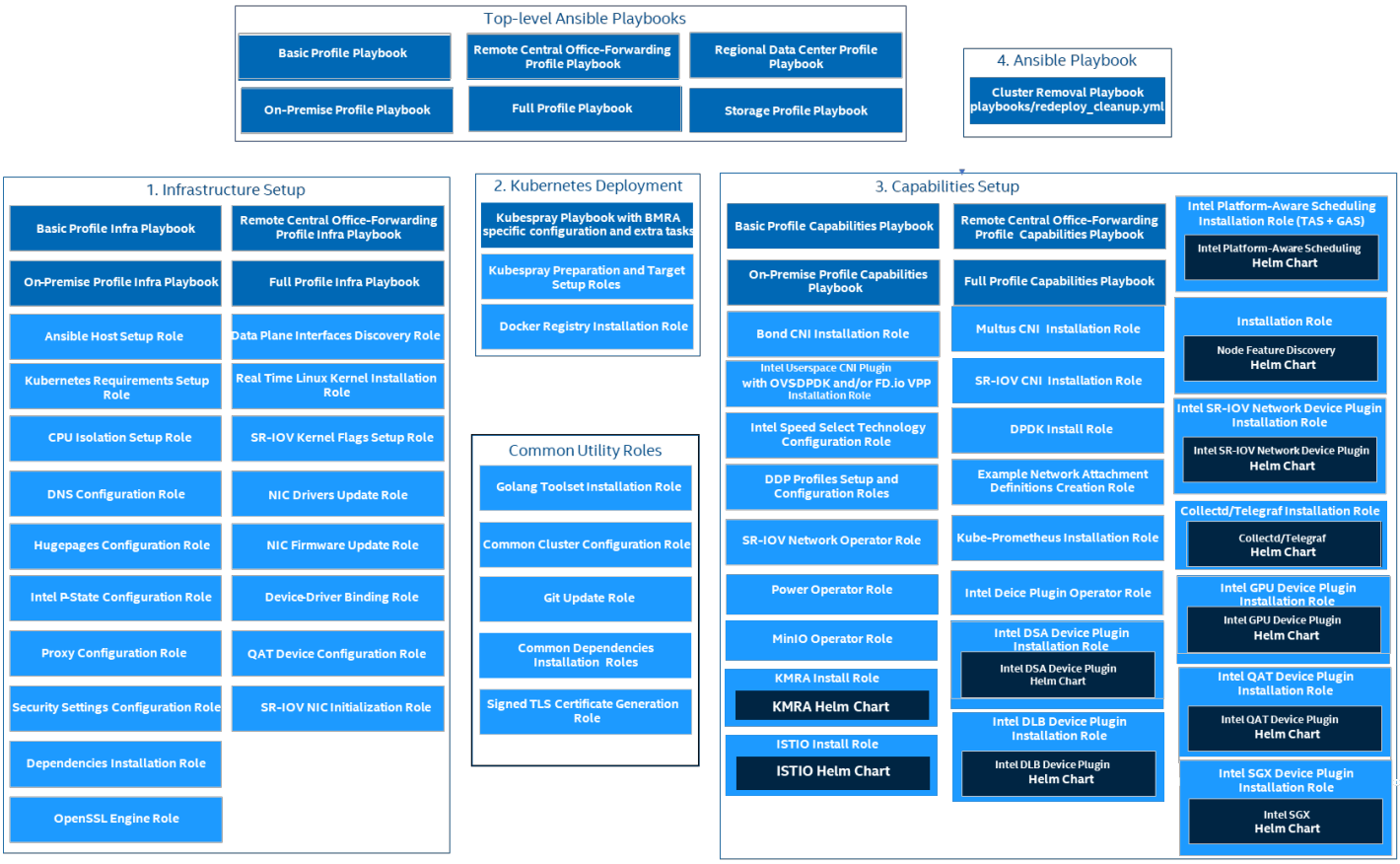

Figure 2. High Level BMRA Ansible Playbooks Architecture[3]

2.4.2 Ansible Playbook Phases

Regardless of the selected Configuration Profile, the installation process always consists of three main phases:

1. **Infrastructure Setup** (sub-playbooks located in `playbooks/infra/` directory)
   These playbooks modify kernel boot parameters and apply the initial system configuration for the cluster nodes. Depending on the selected Configuration Profile this includes:
   - Generic host OS preparation, e.g., installation of required packages, Linux kernel configuration, proxy and DNS configuration, and modification of SELinux policies and firewall rules.
   - Configuration of the kernel boot parameters according to the user-provided configuration in order to configure CPU isolation, SR-IOV related settings such as IOMMU, hugepages, or explicitly enable/disable Intel P-state technology.
   - Configuration of SR-IOV capable network cards and QAT devices. This includes the creation of virtual functions and binding to appropriate Linux kernel modules.
   - Network Adapter drivers and firmware updates, which help ensure that all latest capabilities such as Dynamic Device Personalization (DDP) profiles are enabled.
   - Intel® Speed Select Technology (Intel® SST) configuration, which provides control over base frequency.
   - Installation of Dynamic Device Personalization profiles, which can increase packet throughput, help reduce latency, and lower CPU usage by offloading packet classification and load balancing to the network adapter.
2. **Kubernetes Setup** (located in `playbooks/k8s/` directory)
   This playbook deploys a high availability (HA) Kubernetes (K8s) cluster using Kubespray, which is a project under the Kubernetes community that deploys production-ready Kubernetes clusters. The Multus container network interface (CNI) plugin, which is specifically designed to support multiple networking interfaces in a Kubernetes environment, is deployed by

[3] Refer to https://software.intel.com/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

Kubespray along with Calico and Helm. Preferred security practices are used in the default configuration. On top of Kubespray, there's also a container registry instance deployed to store images of various control-plane Kubernetes applications such as TAS, CMK, or device plugins.

3. **BMRA System Capabilities Setup** (sub-playbooks in the `playbooks/intel` directory):
   Advanced networking technologies, enhanced platform awareness, and device plugin features are deployed by this playbook using operators or Helm charts as part of the BMRA. The following capabilities are deployed:
   - Device plugins that allow using, for example, SR-IOV, QAT, and GPU devices in workloads running on top of Kubernetes.
   - SR-IOV CNI plugin, Bond CNI plugin, and Userspace CNI plugin, which allow Kubernetes pods to be attached directly to accelerated and highly available hardware and software network interfaces.
   - Native CPU Manager for Kubernetes (replacement for CMK), which performs a variety of operations to enable core pinning and isolation on a container or a thread level.
   - Node Feature Discovery (NFD), which is a Kubernetes add-on to detect and advertise hardware and software capabilities of a platform that can, in turn, be used to facilitate intelligent scheduling of a workload.
   - Telemetry Aware Scheduling, which allows scheduling workloads based on telemetry data.
   - Full Telemetry Stack consisting of collectd, Kube-Prometheus, and Grafana, which gives cluster and workload monitoring capabilities and acts as a source of metrics that can be used in TAS to orchestrate scheduling decisions.
   - MinIO operator/console, which supports deploying MinIO tenants onto private and public cloud infrastructures ("Hybrid" Cloud).

2.5 Deployment using Ansible Playbook

This section describes common steps that need to be executed in order to obtain the BMRA Ansible Playbooks source code, prepare target servers, configure inventory and variable files, and deploy the BMRA Kubernetes cluster.

2.5.1 Prepare Target Servers

For each target server that will act as a control or worker node, you must make sure that it meets the following requirements:

- Install Python 3. The following example assumes that the host is running RHEL. Other operating systems may have slightly different installation steps:
  ```
  yum install python3
  ```
- Internet access on all target servers is mandatory. Proxies are supported and can be configured in the Ansible vars.
- BIOS configuration matching the desired state is applied. For details, refer to the specific Configuration Profile section for your profile: BMRA Basic Configuration Profile Setup, BMRA Full Configuration Profile Setup, BMRA On-Premises Edge Configuration Profile Setup; BMRA Regional Data Center Configuration Profile Setup, and Storage Configuration Profile Setup.

For detailed steps on how to build the Ansible Host, refer to Section 6.1.

2.5.2 Prepare Ansible Host and Configuration Templates

Perform the following steps:

1. Log in to your Ansible host (the one that you will run these Ansible playbooks from).
2. Install packages on Ansible Host. The following example assumes that the host is running RHEL. Other operating systems may have slightly different installation steps:
   ```
   yum install python3
   pip3 install –upgrade pip
   pip3 install ansible==3.4.0
   ```
3. Enable passwordless login between all nodes in the cluster.
   Create authentication SSH-Keygen keys on Ansible Host:
   ```
   ssh-keygen
   ```
4. SSH is used by Ansible Host to communicate with each target node. Configure the same SSH keys on each machine. Copy your generated public keys to all the nodes from the Ansible Host:
   ```
   ssh-copy-id root@<target_server_address>
   ```
5. Clone the source code and change working directory.
   ```
   git clone https://github.com/intel/container-experience-kits/
   cd container-experience-kits
   ```
   Check out the latest version of the playbooks – using the tag from Table 23, for example:
   ```
   git checkout v22.01
   ```
   Note: Alternatively go to https://github.com/intel/container-experience-kits/releases, download the latest release tarball, and unarchive it:
   ```
   wget https://github.com/intel/container-experience-kits/archive/v22.01.tar.gz
   tar xf v22.01.tar.gz
   cd container-experience-kits-22.01
   ```
6. Initialize Git submodules to download Kubespray code.
   ```
   git submodule update --init
   ```
7. Decide which Configuration Profile you want to use and export the environmental variable.

For Kubernetes **Basic** Configuration Profile deployment:
```
export PROFILE=basic
```

For Kubernetes **Regional Data Center** Configuration Profile deployment:
```
export PROFILE=regional_dc
```

For Kubernetes **Remote Central Office-Forwarding** Configuration Profile deployment:
```
export PROFILE=remote_fp
```

For Kubernetes **On-Premises Edge** Configuration Profile deployment:
```
export PROFILE=on_prem
```

For Kubernetes **Full** Configuration Profile deployment:
```
export PROFILE=full_nfv
```

For Kubernetes **Storage** Configuration Profile deployment:
```
export PROFILE=storage
```

8. Install requirements needed by deployment scripts.
```
pip3 install -r requirements.txt
```
9. Generate example profiles. Make sure to aware of the machine's type before generating profiles such as 'spr', 'icx', 'clx', and 'skl'.
```
make k8s-profile PROFILE=$PROFILE ARCH=spr
```

2.5.3 Update Ansible Inventory File

Perform the following steps:

1. Edit the `inventory.ini` file generated in the previous steps.
2. In the section `[all]`, specify all your target servers. Use their actual hostnames and Management IP addresses. Also update `ansible_user` and `ansible_password` to match the SSH configuration of the target servers. If any of the servers are configured with passwordless SSH, the `ansible_password` host variable can be removed.

```
[all]
controller1 ansible_host=10.0.0.1 ip=10.0.0.1 ansible_user=USER ansible_password=XXXX
controller2 ansible_host=10.0.0.2 ip=10.0.0.2 ansible_user=USER ansible_password=XXXX
controller3 ansible_host=10.0.0.3 ip=10.0.0.3 ansible_user=USER ansible_password=XXXX
node1   ansible_host=10.0.0.4 ip=10.0.0.4 ansible_user=USER ansible_password=XXXX
node2   ansible_host=10.0.0.5 ip=10.0.0.5 ansible_user=USER ansible_password=XXXX
localhost ansible_connection=local ansible_python_interpreter=/usr/bin/python3

[vm_host]

[kube_control_plane]
controller1
controller2
controller3

[etcd]
controller1
controller2
controller3

[kube_node]
node1
node2

[k8s_cluster:children]
kube_control_plane
kube_node

[all:vars]
ansible_python_interpreter=/usr/bin/python3
```

2.5.4 Update Ansible Host and Group Variables

Perform the following steps.

1. Create `host_vars/<hostname>.yml` files for all worker nodes, matching their hostnames from the inventory file. The provided `host_vars/node1.yml` file can be used as a template.
1. Edit all `host_vars/<hostname>.yml` and `group_vars/all.yml` files to match your desired configuration. Each Configuration Profile uses its own set of variables. Refer to the specific Configuration Profile section for your profile to get a full list of variables and their documentation: Section 7, BMRA Basic Configuration Profile Setup, Section 8, BMRA Full

Configuration Profile Setup, Section 9, BMRA On-Premises Edge Configuration Profile Setup, Section 10, BMRA Remote Central Office-Forwarding Configuration Profile Setup, Section 11, BMRA Regional Data Center Configuration Profile Setup, and Section 12, Storage Configuration Profile Setup.

2.5.5 Run Ansible Cluster Deployment Playbook

After the inventory and vars are configured, you can run the provided playbooks from the root directory of the project.

It is recommended that you check dependencies of components enabled in `group_vars` and `host_vars` with the packaged dependency checker:

```
ansible-playbook -i inventory.ini playbooks/preflight.yml
```

If you are deploying an RHEL 8 cluster, you need to patch kubespray:

```
ansible-playbook -i inventory.ini playbooks/k8s/patch_kubespray.yml
```

Otherwise, you can skip directly to your chosen Configuration Profile playbook:

```
ansible-playbook -i inventory.ini playbooks/${PROFILE}.yml
```

Pay attention to logs and messages displayed on the screen. Depending on the selected Configuration Profile, network bandwidth, storage speed, and other similar factors, the execution may take up to 30-40 minutes.

After the playbook finishes without any "Failed" tasks, you can proceed with the deployment validation described in Section 5, Post Deployment Verification Guidelines.

Note: Additional information can be found in the Ansible Playbook readme.

2.5.6 Run Ansible Cluster Removal Playbook

If the playbook fails or if you want to clean up the environment to run a new deployment, you can optionally use the provided Cluster Removal Playbook (`redeploy_cleanup.yml`) to remove any previously installed Kubernetes and related plugins.

```
ansible-playbook -i inventory.ini playbooks/redeploy_cleanup.yml
```

After successful removal of Kubernetes components, you can repeat Section 2.5.5.

Note: Any OS and/or hardware configurations (for example, proxies, drivers, kernel parameters) are not reset by the cleanup playbook.

3 Reference Architecture Hardware Components and BIOS

For all BMRA Configuration Profiles, this section provides a menu of all possible hardware components for control node and worker node as well as the BIOS components available.

3.1 Hardware Components List for Control Node

The following tables list the hardware options for control nodes, which are responsible for managing the worker nodes in the cluster.

Table 2. Hardware Options for Control Node – 2nd Generation Intel Xeon Scalable Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
2nd Generation Intel® Xeon® Scalable processors	Intel® Xeon® Gold 5218 or 5218N processor at 2.3 GHz, 16 C/32 T, 125 W, or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	DRAM only configuration: 192 GB (12 x 16 GB DDR4 2666 MHz)	Required
Network Adapter	Option 1: Dual Port 25 GbE Intel® Ethernet Network Adapter XXV710-DA2 SFP28+, or	Required
	Option 2: Dual Port 10 GbE Intel® Ethernet Converged Network Adapter X710-DA2 SFP+, or	
	Option 3: Dual Port 10 GbE Intel® Ethernet Converged Network Adapter X520-DA2 SFP+	
Intel® QAT	Intel® QuickAssist Adapter 8970 (PCIe) AIC or equivalent third-party Intel® C620 Series Chipset Intel® QAT enabled PCIe AIC, with minimum 8 lanes of PCIe connectivity	Recommended
Storage (Boot Drive)	Intel® SATA Solid State Drive D3 S4510 at 480 GB or equivalent boot drive	Required
Storage (Capacity)	Intel® NVMe SSD DC P4510 Series at 2 TB or equivalent (Recommended NUMA Aligned)	Recommended

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10 Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

Table 3. Hardware Options for Control Node – 3rd Generation Intel Xeon Scalable Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
3rd Generation Intel Xeon Scalable processors	Intel® Xeon® Gold 5318N processor at 2.1 GHz, 20 C/40 T, 135W or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	256 GB DRAM (16x 16 GB DDR4, 2666 MHz)	Required
Network Adapter	Dual Port 100 GbE Intel® Ethernet Network Adapter E810-CQDA2 QSFP28	Required
Intel® QAT	Intel® QuickAssist Adapter 8960 or 8970 (PCIe*) AIC or equivalent third-party Intel® C620 Series Chipset	Recommended
Storage (Boot Drive)	Intel® SATA Solid State Drive D3 S4510 at 480 GB or equivalent boot drive	Required
Storage (Capacity)	Intel® SSD D7-P5510 Series at 3.84 TB or equivalent drive (recommended NUMA aligned)	Recommended
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10 Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

Table 4. Hardware Options for Control Node – Intel® Xeon® D Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
Intel® Xeon® D processors	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher	Required
Memory	DRAM only configuration: 16 GB DDR4 2933 MHz	Required
Network Adapter	2 x 10 GbE integrated Ethernet ports	Required
Intel® QAT	20 G Intel® QAT	Recommended
Storage (Boot Drive)	Intel® SSD 256 GB 2.5" internal SSD/M.2	Required
Additional Plug-in cards	N/A	

3.2 Hardware Components List for Worker Node Base

A Kubernetes cluster typically consists of multiple worker nodes managed by one or more Kubernetes control nodes.

The following tables list the hardware options for worker nodes in the “base” configuration. If your configuration needs improved processing, you may choose to use the “plus” configuration instead. See the next section for details.

Table 5. Hardware Components for Worker Node Base – 2nd Generation Intel Xeon Scalable Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
2nd Generation Intel® Xeon® Scalable processors	Intel® Xeon® Gold 6230 processor @ 2.1 GHz or 6230N CPU @ 2.3 GHz 20 C/40 T, 125 W, or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	Option 1: DRAM only configuration: 384 GB (12 x 32 GB DDR4 2666 MHz)	Required
	Option 2: DRAM only configuration: 384 GB (24 x 16 GB DDR4 2666 MHz)	

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
	Option 3: DRAM + Intel® Optane™ Persistent Memory DRAM: 192 GB (12x 16 GB DDR4, 2666 MHz)	
Intel® Optane™ Persistent Memory	512 GB (4x 128 GB Intel® Optane™ persistent memory in 2-1-1 Topology)	Recommended
Network Adapter	Option 1: Dual Port 100 GbE Intel® Ethernet Network Adapter E810-CQDA2 QSFP28	Required
	Option 2: Intel® Ethernet Network Adapter XXV710-DA2 QSFP28	
Intel® QAT	Intel® QuickAssist Adapter 8970 (PCIe) AIC or equivalent third-party Intel® C620 Series Chipset Intel® QAT enabled with minimum 8 lanes of PCIe connectivity	Required
Storage (Boot Drive)	Intel® SATA Solid State Drive D3 S4510 at 480 GB or equivalent boot drive	Required
Storage (Capacity)	Intel® NVMe SSD DC P4510 Series P4510 at 2 TB or equivalent (Recommended NUMA Aligned)	Required
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10 Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

Table 6. Hardware Components for Worker Node Base – 3rd Generation Intel Xeon Scalable Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
3rd Generation Intel Xeon Scalable processors	Intel® Xeon® Gold 5318N processor at 2.1 GHz, 24 C/48 T, 150 W, or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	Option 1: DRAM only configuration: 256 GB (8 x 32 GB DDR4, 2666 MHz)	Required
	Option 2: DRAM only configuration: 256 GB (16 x 16 GB DDR4, 2666 MHz)	
Intel® Optane™ Persistent Memory	512 GB (4x 128 GB Intel® Optane™ persistent memory in 2-1-1 Topology)	Recommended
Network Adapter	Option 1: Intel® Ethernet Network Adapter E810-CQDA2	Required
	Option 2: Intel® Ethernet Network Adapter E810-XXVDA-2	
Intel® QAT	Intel® QuickAssist Adapter 8960 or 8970 (PCIe*) AIC or equivalent third-party Intel® C620 Series Chipset	Required
Storage (Boot Drive)	Intel® SATA Solid State Drive D3 S4510 at 480 GB or equivalent boot drive	Required
Storage (Capacity)	Intel® SSD D7-P5510 Series at 3.84 TB or equivalent drive (recommended NUMA aligned)	Required
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10 Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

Table 7. Hardware Components for Worker Node Base – Intel® Xeon® D Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
Intel® Xeon® D processors	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or Intel® Xeon® D-1700 processor, 10 core LCC or Intel Xeon D-2733 NT processor, 8 cores HCC, 80 W	Required
Memory	DRAM only configuration: 32 GB DDR4 2667 MHz	Required
Network Adapter	2 x 10/25 GbE integrated Ethernet ports OR Intel® Ethernet Network Adapter E810-CQDA2	Required
Intel® QAT	Intel® QuickAssist Adapter 8960 or 8970 (PCIe*) AIC or equivalent third-party Intel® C620 Series Chipset	Recommended
Storage (Boot Drive)	Intel® SSD 256 GB 2.5" internal SSD/M.2	Required
Additional Plug-in cards	N/A	

3.3 Hardware Components List for Worker Node Plus

A Kubernetes cluster typically consists of multiple worker nodes managed by one or more Kubernetes control nodes.

The following tables list the hardware options for worker nodes in the "plus" configuration, which helps improve the processing capability due to more powerful CPU, more memory, more disk space, and an amazingly fast network.

Table 8. Hardware Components for Worker Node Plus – 2nd Generation Intel Xeon Scalable Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
2nd Generation Intel® Xeon® Scalable processors	Intel® Xeon® Gold 6252 processor @ 2.1 GHz or 6252N processor @ 2.3 GHz 24 C/48 T, 150 W, or higher number Intel® Xeon® Gold/Platinum CPU SKU	Required
Memory	Option 1: DRAM only configuration: 384 GB (12 x 32 GB DDR4 2666 MHz)	Required
	Option 2: DRAM only configuration: 384 GB (24 x 16 GB DDR4 2666 MHz)	
	Option 3: DRAM + Intel® Optane™ persistent memory DRAM: 192 GB (12 x 16 GB DDR4 2666 MHz)	
Intel® QAT	Intel® C620 Series Chipset integrated on base board Intel® C627/C628 Chipset, integrated with NUMA connectivity to each CPU or minimum 16 Peripheral Component Interconnect express (PCIe) lane connectivity to one CPU	Required
Intel® Optane™ Persistent Memory	Option 1: 1 TB (8x 128 GB Intel® Optane™ persistent memory in 2-2-1 Topology)	Recommended
	Option 2: 1.5 TB (12x 128 GB Intel® Optane™ persistent memory in 2-2-2 Topology)	
Network Adapter	Option 1: Dual Port 25 GbE Intel® Ethernet Network Adapter X710-DA4 SFP+, or	Required
	Option 2: Dual Port 100 GbE Intel® Ethernet Network Adapter E810-CQDA2 QSFP28	
Storage (Boot Drive)	Intel® SATA Solid State Drive D3 S4510 at 480 GB or equivalent boot drive	Required
Storage (Capacity)	Intel® NVMe SSD DC P4510 Series at 2 TB or equivalent (Recommended NUMA Aligned)	Required
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10 Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

Table 9. Hardware Components for Worker Node Plus – Intel® Xeon® D Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
Intel® Xeon® D processors	Intel® Xeon® D-2766NT processor 2.1GHz, 14 core HCC, 97 W, or higher	Required
Memory	DRAM only configuration: 64 GB DDR4 2667 MHz	Required
Network Adapter	4 x 10/25 GbE integrated Ethernet ports Intel® Ethernet Network Adapter E810-CQDA2	Required
Intel® QAT	Intel® QuickAssist Adapter 8960 or 8970 (PCIe*) AIC or equivalent third-party Intel® C620 Series Chipset	Recommended
Storage (Boot Drive)	Intel® SSD 512 GB 2.5" internal SSD/M.2	Required
Additional Plug-in cards	N/A	

3.4 Hardware Components List for Storage Node

Table 10. Hardware Components for Storage Node – 3rd Generation Intel Xeon Scalable Processor

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
3rd Generation Intel Xeon Scalable processors	Intel® Xeon® Gold 6338N CPU @ 2.2 GHz 32 C/64 T, 185 W, or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	Option 1: DRAM only configuration: 512 GB (16x 32 GB DDR4, 2666 MHz)	Required
	Option 2: DRAM only configuration: 512 GB (32x 16 GB DDR4, 2666 MHz)	

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
Intel® QAT	Intel® C620 Series Chipset integrated on base board Intel® C627/C628 Chipset, integrated with NUMA connectivity to each CPU or minimum 16 Peripheral Component Interconnect express (PCIe) lane connectivity to one CPU	Required
Intel® Optane™ Persistent Memory	512 GB (4x 128 GB Intel® Optane™ persistent memory in 2-1-1 topology)	Recommended
Network Adapter	Option 1: Intel® Ethernet Network Adapter E810-CQDA2	Required
	Option 2: Intel® Ethernet Network Adapter E810-2CQDA2	
Storage (Boot Drive)	Intel® SATA Solid State Drive D3 S4510 at 480 GB or equivalent boot drive	Required
Storage (Capacity)	Kioxia CM6 3.2TB NVMePCIe4x4 2.5"15mm SIE 3DWPD - KCM6XVUL3T20	Required
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10 Gbps port for Management Network Adapter	Required

3.5 Hardware BOMs Supporting all BMRA Configuration Profiles

The following tables list the hardware BOMs for Control Nodes, Worker Node Base, and Worker Node Plus.

Choose your controller profile from the three available profiles (Controller_xGen_1, Controller_xGen_2, or Controller_xGen_3) based on your BIOS profile (Energy Balance, Deterministic, or Max Performance, respectively).

The profiles for Worker Nodes vary with respect to network interface card, Intel® QuickAssist Technology, and BIOS profiles. You may choose based on the requirement for the workloads to be run on the worker nodes.

Table 11. Control Node Hardware Setup for all Configuration Profiles – 2nd Generation Intel Xeon Scalable Processor

NAME	Controller_2ndGen_1	Controller_2ndGen_2	Controller_2ndGen_3
Platform	S2600WFQ	S2600WFQ	S2600WFQ
CPU/node	2x 5218 or 2x 5218N	2x 5218 or 2x 5218N	2x 5218 or 2x 5218N
Mem	192 GB	192 GB	192 GB
Intel Optane Persistent Memory	Recommended	Recommended	Recommended
Network Adapter	2x XXV710-DA2 or 2x X710-DA2 or 2x X520-DA2	2x XXV710-DA2	2x XXV710-DA2
Storage (Boot Media)	Required - 2x	Required - 2x	Required - 2x
Storage (Capacity)	Recommended - 2x (1 per NUMA)	Recommended - 2x (1 per NUMA)	Recommended - 2x (1 per NUMA)
LOM	No	No	No
Intel® QAT	Recommended	N/A	N/A
BIOS Configuration			
Intel® HT Technology enabled	Yes	Yes	Yes
Intel® VT-x enabled	No	Yes	Yes
Intel® VT-d enabled	No	Yes	Yes
BIOS Profile	Energy Balance	Deterministic	Max Performance

Virtualization enabled	No	Yes	Yes

Table 12. Control Node Hardware Setup for all Configuration Profiles – 3rd Generation Intel Xeon Scalable Processor

NAME	Controller_3rdGen_1	Controller_3rdGen_2	Controller_3rdGen_3
Platform	M50CYP	M50CYP	M50CYP
CPU/node	2x 5318N 20c	2x 5318N 20c	2x 5318N 20c
Mem	256 GB	256 GB	256 GB
Intel Optane Persistent Memory	Recommended	Recommended	Recommended
Network Adapter	2x E810-CQDA2	2x E810-CQDA2	2x E810-CQDA2
Storage (Boot Media)	Required - 2x	Required - 2x	Required - 2x
Storage (Capacity)	Recommended - 2x (1 per NUMA)	Recommended - 2x (1 per NUMA)	Recommended - 2x (1 per NUMA)
LOM	No	No	No
Intel® QAT	Recommended	N/A	N/A
BIOS Configuration			
Intel® HT Technology enabled	Yes	Yes	Yes
Intel® VT-x enabled	No	Yes	Yes
Intel® VT-d enabled	No	Yes	Yes
BIOS Profile	Energy Balance	Deterministic	Max Performance
Virtualization enabled	No	Yes	Yes

Table 13. Control Node Hardware Setup for all Configuration Profiles – Intel® Xeon® D Processor

NAME	Controller_Xeon_D_1	Controller_Xeon_D_2	Controller_Xeon_D_3
Platform	Intel® SoC Based Server Reference Platform Board (Codenamed Brighton City) K97971-101	Intel® SoC Based Server Reference Platform Board (Codenamed Brighton City) K97971-101	Intel® SoC Based Server Reference Platform Board (Codenamed Brighton City) K97971-101
CPU/node	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher
Mem	16 GB DDR4 2933 MHz	16 GB DDR4 2933 MHz	16 GB DDR4 2933 MHz
Network Adapter	2 x 10 GbE integrated Ethernet ports	2 x 10 GbE integrated Ethernet ports	2 x 10 GbE integrated Ethernet ports

Storage (Boot Media)	Intel® SSD 256 GB 2.5" internal SSD/M.2	Intel® SSD 256 GB 2.5" internal SSD/M.2	Intel® SSD 256 GB 2.5" internal SSD/M.2
LOM	No	No	No
Intel® QAT AIC	Recommended	N/A	N/A
BIOS Configuration			
Intel® HT Technology enabled	Yes	Yes	Yes
Intel® VT-x enabled	No	Yes	Yes
Intel® VT-d enabled	No	Yes	Yes
BIOS Profile	Energy Balance	Deterministic	Max Performance
Virtualization enabled	No	Yes	Yes

Table 14. Worker Node Base Hardware Setup for all Configuration Profiles – 2nd Generation Intel Xeon Scalable Processor

NAME	Worker_2ndGen_Base_1	Worker_2ndGen_Base_2	Worker_2ndGen_Base_3
Platform	S2600WFQ	S2600WFQ	S2600WFQ
CPU/node	2x 6230 or 6230N	2x 6230 or 6230N	2x 6230 or 6230N
Mem	384 GB	384 GB	384 GB
Intel Optane Persistent Memory	Recommended – 512 GB	Recommended – 512 GB	Recommended – 512 GB
Network Adapter	2x XXV710-DA2 or 2x E810-CQDA2	2x XXV710-DA2	2x XXV710-DA2
Storage (Boot Media)	Required - 2x	Required - 2x	Required - 2x
Storage (Capacity)	Required- 2x (1 per NUMA)	Required- 2x (1 per NUMA)	Required- 2x (1 per NUMA)
LOM	No	No	Yes
Intel® QAT	No	Optional	Yes
Additional Plug-in cards	No	No	No
BIOS Configuration			
Intel® HT Technology enabled	Yes	Yes	Yes
Intel® VT-x enabled	Yes	Yes	Yes
Intel® VT-d enabled	Yes	Yes	Yes
BIOS Profile	Energy Balance	Deterministic	Max Performance
Virtualization enabled	No	Yes	Yes

Table 15. Worker Node Plus Hardware Setup for all Configuration Profiles – 2nd Generation Intel Xeon Scalable Processor

NAME	Worker_2ndGen_Plus_1	Worker_2ndGen_Plus_2
Platform	S2600WFQ	S2600WFQ
CPU/node	2x 6252 or 6252N	2x 6252 or 6252N
Mem	384 GB	384 GB
Intel Optane Persistent Memory	Recommended – 1 TB/1.5 TB	Recommended – 1 TB/1.5 TB
Network Adapter	2x E810-CQDA2	2x E810-CQDA2

NAME	Worker_2ndGen_Plus_1	Worker_2ndGen_Plus_2
Storage (Boot Media)	Required – 256 GB	Required – 256 GB
LOM	No	Yes
Intel® QAT	No	Yes
Additional Plug-in cards	No	No
BIOS Configuration		
Intel® HT Technology enabled	Yes	Yes
Intel® VT-x enabled	Yes	Yes
Intel® VT-d enabled	Yes	Yes
BIOS Profile	Deterministic	Max Performance
Virtualization enabled	Yes	Yes

Table 16. Worker Node Base Hardware Setup for all Configuration Profiles – 3rd Generation Intel Xeon Scalable Processor

NAME	Worker_3rdGen_Base_1	Worker_3rdGen_Base_2	Worker_3rdGen_Base_3
Platform	M50CYP	M50CYP	M50CYP
CPU/node	2x 5318N 24c	2x 5318N 24c	2x 5318N 24c
Mem	512 GB	512 GB	512 GB
Intel Optane Persistent Memory	Recommended – 512 GB	Recommended – 512 GB	Recommended – 512 GB
Network Adapter	2x E810-CQDA2	2x E810-CQDA2	2x E810-2CQDA2 or 4x E810-CQDA2
Storage (Boot Media)	Required - 2x	Required - 2x	Required - 2x
Storage (Capacity)	Required- 2x (1 per NUMA)	Required- 2x (1 per NUMA)	Required- 2x (1 per NUMA)
LOM	No	Yes	No
Intel® QAT	No	Yes	Optional
Additional Plug-in cards	No	No	No
BIOS Configuration			
Intel® HT Technology enabled	Yes	Yes	Yes
Intel® VT-x enabled	Yes	Yes	Yes
Intel® VT-d enabled	Yes	Yes	Yes
BIOS Profile	Energy Balance	Max Performance	Deterministic
Virtualization enabled	No	Yes	Yes

Table 17. Worker Node Plus and Storage Node Hardware Setup for all Configuration Profiles – 3rd Generation Intel Xeon Scalable Processor

NAME	Worker_3rdGen_Plus_1	Worker_3rdGen_Plus_2	Worker_3rdGen_Plus_3	Storage_3rdGen_1
Platform	M50CYP	M50CYP	M50CYP	M50CYP
CPU/node	2x 6338N 32c	2x 6338N 32c	2x 6338N 32c	2x 6338N 32c
Mem	512 GB	512 GB	512 GB	512 GB
Intel Optane Persistent Memory	Recommended – 512 GB	Recommended – 512 GB	Recommended – 512 GB	Recommended – 512 GB
Network Adapter	2x E810-2CQDA2 or 4x E810-CQDA2	2x E810-2CQDA2	2x E810-2CQDA2 or 4x E810-CQDA2	2x E810-2CQDA2 or 4x E810-CQDA2

NAME	Worker_3rdGen_Plus_1	Worker_3rdGen_Plus_2	Worker_3rdGen_Plus_3	Storage_3rdGen_1
Storage (Boot Media)	Required - 2x	Required - 2x	Required - 2x	Required - 2x
Storage (Capacity)	Required- 4x (2 per NUMA)	Required- 4x (2 per NUMA)	Required- 4x (2 per NUMA)	Required Kioxia 3.2TB - 8x
LOM	Yes	Yes	No	Yes
Intel® QAT	Yes	No	Optional	Yes
Additional Plug-in cards	No	Intel Server GPU	No	No
BIOS Configuration				
Intel® HT Technology enabled	Yes	Yes	Yes	Yes
Intel® VT-x enabled	Yes	Yes	Yes	Yes
Intel® VT-d enabled	Yes	Yes	Yes	Yes
BIOS Profile	Max Performance	Max Performance	Deterministic	Max Performance
Virtualization enabled	Yes	Yes	Yes	Yes

Table 18. Worker Node Base Hardware Setup for all Configuration Profiles – Intel® Xeon® D Processor

NAME	Worker_Xeon_D_Base_1	Worker_Xeon_D_Base_2	Worker_Xeon_D_Base_3
Platform	Intel® SoC Based Server Reference Platform Board (Codenamed Brighton City) K97971-101 or Taylors Falls Reference Design	Intel® SoC Based Server Reference Platform Board (Codenamed Brighton City) K97971-101 or Taylors Falls Reference Design	Intel® SoC Based Server Reference Platform Board (Codenamed Brighton City) K97971-101 or Taylors Falls Reference Design
CPU/node	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher	Intel® Xeon® D-1700 processor, 4 core LCC, 45 W, or higher
Mem	16 GB DDR4 2933 MHz	16 GB DDR4 2933 MHz	16 GB DDR4 2933 MHz
Network Adapter	Intel® Ethernet Network Adapter E810-CQDA2	Intel® Ethernet Network Adapter E810-CQDA2	Intel® Ethernet Network Adapter E810-CQDA2
Storage (Boot Media)	Required – 256 GB	Required – 256 GB	Required – 256 GB
LOM	No	Yes	No
Intel® QAT	No	Yes	Optional
Additional Plug-in cards	No	No	No
BIOS Configuration			
Intel® HT Technology enabled	Yes	Yes	Yes
Intel® VT-x enabled	Yes	Yes	Yes
Intel® VT-d enabled	Yes	Yes	Yes
BIOS Profile	Max Performance	Deterministic	Max Performance
Virtualization enabled	Yes	Yes	Yes

Table 19. Worker Node Plus Hardware Setup for all Configuration Profiles – Intel® Xeon® D Processor

NAME	Worker_Xeon_D_Plus_1	Worker_Xeon_D_Plus_2
Platform	Intel® SoC Based Server Reference Platform Board (Codenamed Moro City)	Intel® SoC Based Server Reference Platform Board(Codenamed Moro City)
CPU/node	Intel Xeon D-2700 processor, 16 core HCC, 105 W, or higher	Intel Xeon D-2700 processor, 16 core HCC, 105 W, or higher
Mem	64 GB DDR4 2933 MHz	64 GB DDR4 2933 MHz
Network Adapter	Intel® Ethernet Network Adapter E810-CQDA2	Intel® Ethernet Network Adapter E810-CQDA2
Storage (Boot Media)	Required – 512 GB	Required – 512 GB
LOM	Yes	No
Intel® QAT	Yes	Optional
Additional Plug-in cards	No	No
BIOS Configuration		
Intel® HT Technology enabled	Yes	Yes
Intel® VT-x enabled	Yes	Yes
Intel® VT-d enabled	Yes	Yes
BIOS Profile	Max Performance	Deterministic
Virtualization enabled	Yes	Yes

3.6 Platform BIOS

This section provides BIOS Configuration Profiles for each of the BMRA Configuration Profiles. For details on how the BIOS configuration should be set per each Configuration Profile, go to tables in Section 5.4.

For more information about BIOS settings, visit https://www.intel.com/content/dam/support/us/en/documents/server-products/Intel_Xeon_Processor_Scalable_Family_BIOS_User_Guide.pdf.

Table 20. Platform BIOS Settings for 2nd Generation Intel® Xeon® Scalable Processor

MENU (ADVANCED)	PATH TO BIOS SETTING	BIOS SETTINGS	ENERGY BALANCE	MAX PERFORMANCE	DETERMINISTIC
Advanced	Processor Configuration	Intel® Hyper-Threading Tech	Enabled	Enabled	Enabled
		Intel® Virtualization Technology	Enabled	Enabled	Enabled
	Integrated IO Configuration	Intel® VT for Directed I/O	Enabled	Enabled	Enabled
Advanced/Power Configuration	Power and Performance	CPU Power and Performance Policy	Balanced Performance	Performance	Performance
		Workload Configuration	I/O sensitive	I/O sensitive	I/O sensitive
	CPU P-state control	Enhanced Intel SpeedStep® Technology	Enabled	Enabled	Disabled* [Read footnote]
		Activate PBF	Disabled	Enabled	Enabled
		Configure PBF	Disabled	Disabled	Disabled
		Intel® Turbo Boost Technology	Enabled	Enabled	Disabled* [Read footnote]
		Energy Efficient Turbo	Enabled	Disabled	N/A
		Intel Configurable TDP	Disabled	Disabled	Disabled
	Hardware P-states	Hardware P-states	Native Mode with no legacy Support	Disabled** [Read footnote]	Disabled** [Read footnote]
		EPP Enable	Enabled	Enabled	Enabled
		RAPL Prioritization	Disabled	Enabled	Enabled
	CPU C-state Control	Package C-state	C6 Retention	C6 Retention	C0/C1 State
		C1E	Enabled	Enabled	Disabled
		Processor C6	Enabled	Enabled	Disabled
	Uncore Power Management	Uncore Frequency scaling	Enabled	Disabled	Disabled
		Performance P-limit	Enabled	Disabled	Disabled
Advanced	Memory Configuration	IMC Interleaving	2-way interleave	2-way Interleave	2-way Interleave
	System Acoustic and Performance Configuration	Set Fan Profile	Acoustic	Performance	Performance
GPU	GPU Fz	Lock 900Mhz	Optional	Optional	Optional

* Enabled in the case where Intel® SST-BF is enabled to allow for configuration of individual core speeds.

** "Native Mode with No Legacy Support" where Intel® SST-BF need to be enabled

Use the following table to configure the BIOS settings to use Intel SST-BF, Intel SST-TF, and Intel SST-PP in 3rd Generation Intel Xeon Scalable Processor systems.

Table 21. BIOS Settings to Enable Intel SST-BF, Intel SST-TF, and Intel SST-PP

BIOS SETTING	STATUS
Hardware PM State Control	
Scalability	Disable
Hardware PM Interrupt	Disable
CPU P-state	
Dynamic SST-PP	Enable
Speed Step (P-states)	Enable
Activate SST-BF	Enable
Configure SST-BF	Enable
EIST PSD Function	HW_All
Turbo	Enable
Energy Efficient Turbo	Enable
Boot Performance	Max
Freq: Prioritization AC	
SST-CP	Enable

In BIOS, the configuration paths might be slightly different, depending on platform, but the key settings are as follows and must be performed in order.

Table 22. BIOS Settings to Enable Intel SGX on 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

BIOS SETTING	STATUS
Socket Configuration > Processor Configuration > Total Memory Encryption (TME)	Enable
Socket Configuration > Common RefCode Configuration > UMA-Based Clustering	Disable (All2All)
Socket Configuration > Processor Configuration > SW Guard Extensions (SGX)	Enable

4 Reference Architecture Software Components

4.1 Software Components Supported

This section lists all the software components deployed by BMRA flavors, per Configuration Profiles definition.

Table 23. Software Components

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
	Ubuntu 20.04.3 Kernel version: 5.4.0-81-generic (20.04.3)	https://www.ubuntu.com
	Ubuntu 21.04 Kernel version: 5.11.0-16-generic	
	RHEL 8.5 Kernel version: 4.18.0-348.el8.x86_64	https://www.redhat.com/
Data Plane Development Kit	DPDK 21.11	https://core.dpdk.org/download/
Open vSwitch with DPDK	OVS-DPDK v2.16.2	https://github.com/openvswitch/ovs
Vector Packet Processing	VPP 19.04	https://docs.fd.io/vpp/

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
Telegraf	1.0	https://github.com/intel/observability-telegraf
collectd	a7cea43d9d2f67c38fbf0407786edbe660ee9072945f7bb272b55fd255e8eaca	https://www.collectd.org/
Grafana	8.3.4	https://www.grafana.com/
Prometheus	2.29.2	quay.io/prometheus/prometheus
Prometheus nginx image	1.21.5-alpine	docker.io/library/nginx:1.21.5-alpine
Ansible	3.4.0	https://www.ansible.com/
BMRA Ansible Playbook	v22.01	https://github.com/intel/container-experience-kits
Python	Python 3.6.x for RHEL 8 Python 3.8.x for Ubuntu 20.04 and Python 3.9.x for Ubuntu 21.04	https://www.python.org/
Kubespray	2.17	https://github.com/kubernetes-sigs/kubespray
Docker	20.10	https://www.docker.com/
containerd	1.4.9	Dependency of other software - not downloaded independently
CRI-O	1.21.3	Dependency of other software - not downloaded independently
Container orchestration engine	Kubernetes v1.22.3	https://github.com/kubernetes/kubernetes
	Kubernetes v1.21.5	
	Kubernetes v1.20.6	
CPU Manager (native to Kubernetes)	Available natively in Kubernetes	N/A
Platform Aware Scheduling (TAS)	TAS 0.1	https://github.com/intel/platform-aware-scheduling
Platform Aware Scheduling (GAS)	GAS 0.1	https://github.com/intel/platform-aware-scheduling
k8s-prometheus-adapter	0.8.4	Dependency of other software - not downloaded independently
k8s node-exporter	1.3.1	Dependency of other software - not downloaded independently
k8s prometheus-operator	0.50.0	https://github.com/prometheus-operator/kube-prometheus
k8s kube-rbac-proxy	0.11.0	Dependency of other software - not downloaded independently
Node Feature Discovery	0.10.0	https://github.com/kubernetes-sigs/node-feature-discovery
Multus CNI	3.8	https://github.com/intel/multus-cni
SR-IOV CNI	2.6.2	https://github.com/intel/sriov-cni
SR-IOV network device plugin	3.4.0	https://github.com/intel/sriov-network-device-plugin
SR-IOV Network Operator	1f7ecb88b32b669936264a75a3ae19f8c7f519eabb8b3bdc57fdffe3d5a26dbe	https://github.com/k8snetworkplumbingwg/sriov-network-operator
Device Plugins Operator	0.23	https://github.com/intel/intel-device-plugins-for-kubernetes

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
QAT device plugin	0.23	https://github.com/intel/intel-device-plugins-for-kubernetes
GPU device plugin	0.23	https://github.com/intel/intel-device-plugins-for-kubernetes
Intel® SGX device plugin	0.23	https://github.com/intel/intel-device-plugins-for-kubernetes
DLB device plugin	0.23	https://github.com/intel/intel-device-plugins-for-kubernetes
DSA device plugin	0.23	https://github.com/intel/intel-device-plugins-for-kubernetes
Userspace CNI	1.3	https://github.com/intel/userspace-cni-network-plugin
Bond CNI plugin	1.0	https://github.com/intel/bond-cni
Intel® Ethernet Drivers	i40e v2.17.15 ice v1.7.16 iavf v4.3.19	https://sourceforge.net/projects/e1000/files/i40e%20stable/2.17.15/ https://sourceforge.net/projects/e1000/files/ice%20stable/1.7.16/ https://sourceforge.net/projects/e1000/files/iavf%20stable/4.3.19/
Intel® Ethernet NVM Update Package 700 Series	8.50	https://www.intel.com/content/www/us/en/download/18190/non-volatile-memory-nvm-update-utility-for-intel-ethernet-network-adapter-700-series.html
Intel® Ethernet NVM Update Package 800 Series	3.00	https://downloadmirror.intel.com/29738/eng/e810_nvmupdatepackage_v3_00_linux.tar_.gz
DDP Profiles	Dynamic Device Personalization for Intel® Ethernet 700 Series Version 25.4	https://downloadmirror.intel.com/27587/eng/gtp.zip https://downloadmirror.intel.com/28940/eng/mplsogreudp.zip https://downloadmirror.intel.com/28040/eng/ppp-oe-ol2tpv2.zip https://downloadmirror.intel.com/29446/eng/esp-ah.zip https://downloadmirror.intel.com/29780/eng/ecpri.zip
	1.3.30.0	https://downloadcenter.intel.com/download/29889/Intel-Ethernet-800-Series-Telecommunication-Comms-Dynamic-Device-Personalization-DDP-Package
Intel® QAT Drivers	QAT.L.4.16.0-00017	Dependency of other software - not downloaded independently
Intel® QAT Drivers (internal)	QAT20.L.0.8.0-00071	Dependency of other software - not downloaded independently
OpenSSL	openssl-3.0.1	https://github.com/openssl/openssl https://www.openssl.org/source/
OpenSSL (internal)	openssl-1.1.1j	Dependency of other software - not downloaded independently
OpenSSL QAT Engine	0.6.7	https://github.com/intel/QAT_Engine
OpenSSL QAT Engine (internal)	0.6.10_INT	Dependency of other software - not downloaded independently
Intel ipsec-mb	1.1	https://github.com/intel/intel-ipsec-mb
Intel® SGX DCAP Drivers	1.41	https://download.01.org/intel-sgx/sgx-dcap/1.10.3/linux/
Intel® SGX SDK	2.15.101.1	https://download.01.org/intel-sgx/sgx-dcap/1.10.3/linux/
Intel® KMRA	1.4.0	https://01.org/key-management-reference-application-kmra
Intel® KMRA AppHSM	1.4.0	https://hub.docker.com/r/intel/apphsm
Intel® KMRA CTK	1.4.0	https://hub.docker.com/r/intel/ctk_loadkey
Intel® KMRA PCCS	1.4.0	https://hub.docker.com/r/intel/pccs
Istio operator	1.12.2	https://github.com/istio/istio/releases/download/

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
Istio operator (internal)	1.10.1	https://github.com/intel-innersource/applications.services.cloud.istio.deployment
istio-intel/pilot-cryptomb (internal)	21.06.2	https://github.com/intel-innersource/applications.services.cloud.istio.deployment
istio-intel/proxyv2-cryptomb (internal))	21.06.2	https://github.com/intel-innersource/applications.services.cloud.istio.deployment
istio-intel/proxyv2-openssl-qat (internal)	21.09.1	https://github.com/intel-innersource/applications.services.cloud.istio.deployment
istio-intel/proxyv2-openssl (internal)	1.9.4	https://github.com/intel-innersource/applications.services.cloud.istio.deployment
istio-intel/tcpip-bypass-ebpf (internal)	21.06.1	https://github.com/intel-innersource/applications.services.cloud.istio.deployment
istio-intel/trusted-certificate-issuer	1.0	https://github.com/intel/trusted-certificate-issuer
istio-intel/tcpip-bypass-ebpf	1.0	https://hub.docker.com/r/intel/istio-tcpip-bypass
MinIO operator	4.4.2	https://github.com/minio/operator
MinIO console	0.13.2	https://github.com/minio/operator
Power Manager Operator	1.0.0	https://hub.docker.com/r/intel/power-operator
Power Node Agent Operator	1.0.0	https://hub.docker.com/r/intel/power-node-agent
Intel® RDT	4.3.0	Dependency of other software - not downloaded independently

4.2 Software Components Compatibility Matrices

Legend for the tables in this section	
	Indicates that the combination is unsupported
	Indicates that the combination is supported and tested
	Indicates that the combination is expected to work but untested
	Indicates that the combination is not applicable

Note: Features that are not listed have been verified to work for all combinations

Feature/Platform Compatibility Limitations			
	2nd Generation Intel® Xeon® Scalable Processor	3rd Generation Intel® Xeon® Scalable Processor	Intel® Xeon® D Processor
dsa			
dlb			
sst-bf			
sst-cp			
sst-tf			
sst-pp			
sst-operator			
minio / nvme			
sgx			
kmra			
gpu			
vm			

Feature/Platform Validation Matrix			
	2nd Generation Intel® Xeon® Scalable Processor	3rd Generation Intel® Xeon® Scalable Processor	Intel® Xeon® D Processor
Basic			
Full			
On-Premises Edge			
Remote Central Office-Forwarding			
Regional Data Center			
Storage			

Feature / OS Compatibility Limitations			
	Ubuntu 20.04.3 5.4.0-81-generic	Ubuntu 21.04 5.11.0-16-generic	RHEL 8.5 4.18.0-348.el8.x86_64
dsa			
dlb			

Feature / Feature Compatibility Limitations											
	DPDK 21.11	DPDK 20.08	OVS 2.16.2	VPP	collectd	Telegraf	SST-BF	SST-CP	SST-TF	SST-PP	sst-operator
DPDK 21.11											
DPDK 20.08											
OVS 2.16.2											
VPP											
collectd											
Telegraf											
SST-BF											
SST-CP											
SST-TF											
SST-PP											
sst-operator											

5 Post Deployment Verification Guidelines

This section describes a set of processes that you can use to verify the components deployed by the scripts. The processes are not Configuration Profile-specific. They can be implemented for each of the Configuration Profiles described in the following sections:

- Section 7, BMRA Basic Configuration Profile Setup
- Section 8, BMRA Full Configuration Profile Setup
- Section 9, BMRA On-Premises Edge Configuration Profile Setup
- Section 10, BMRA Remote Central Office-Forwarding Configuration Profile Setup
- Section 11, BMRA Regional Data Center Configuration Profile Setup
- Section 12, Storage Configuration Profile Setup

5.1 Check the Kubernetes Cluster

Perform the following steps:

1. Check the post-deployment node status of the control nodes and worker nodes.

```
# kubectl get nodes -o wide
NAME          STATUS   ROLES                  AGE   VERSION   INTERNAL-IP    EXTERNAL-IP   OS-
IMAGE                              KERNEL-VERSION          CONTAINER-RUNTIME
node1   Ready    worker                 60m   v1.22.3   10.166.30.34   <none>        Red Hat
Enterprise Linux 8.5 (Ootpa)   4.18.0-348.el8.x86_64   docker://20.10.12
controller   Ready    control-plane,master   61m   v1.22.3   10.166.31.41   <none>        Red
Hat Enterprise Linux 8.5 (Ootpa)   4.18.0-348.el8.x86_64   docker://20.10.12
```

2. Check pod status of control nodes and worker nodes. All pods should be in Running or Completed status.

```
# kubectl get pods --all-namespaces
NAMESPACE               NAME                                                READY
STATUS            RESTARTS       AGE
cert-manager            cert-manager-56b686b465-5qxxt                       1/1
Running           0              21m
cert-manager            cert-manager-cainjector-75c94654d-tztfg             1/1
Running           0              21m
cert-manager            cert-manager-webhook-69bd5c9d75-m7rhf               1/1
Running           0              21m
intel-power             controller-manager-f584c9458-52wdq                  1/1
Running           0              11m
intel-power             power-node-agent-2pm84                              2/2
Running           0              11m
istio-system            istio-ingressgateway-88cc46fd6-n9pfn                1/1
Running           0              5m31s
istio-system            istiod-576d9f454-w9nvt                              1/1
Running           0              5m37s
```

```
kmra                 kmra-apphsm-85fb47f7f9-8rjb4                              2/2
Running            0             14m
kmra                 kmra-ctk-5558c9954b-zc6zv                                 0/1
Running            7 (2m7s ago)   14m
kmra                 kmra-pccs-675f458576-597lq                                2/2
Running            0             14m
kube-system          bypass-tcpip-5v2vh                                        1/1
Running            0             5m45s
kube-system          bypass-tcpip-nrjxj                                        1/1
Running            0             5m45s
kube-system          calico-kube-controllers-684bcfdc59-8sczm                  1/1
Running            2 (22m ago)   23m
kube-system          calico-node-865wh                                         1/1
Running            2 (21m ago)   23m
kube-system          calico-node-gb5wz                                         1/1
Running            1 (22m ago)   23m
kube-system          container-registry-55d455b586-z5shr                       2/2
Running            0             19m
kube-system          coredns-8474476ff8-p67xp                                  1/1
Running            1 (22m ago)   22m
kube-system          coredns-8474476ff8-pkvxs                                  1/1
Running            1 (22m ago)   22m
kube-system          dns-autoscaler-5ffdc7f89d-g9225                           1/1
Running            1 (22m ago)   22m
kube-system          intel-qat-plugin-flncg                                    1/1
Running            0             10m
kube-system          intel-sgx-aesmd-shhfk                                     1/1
Running            0             14m
kube-system          intel-sgx-plugin-r2jg9                                    1/1
Running            0             14m
kube-system          inteldeviceplugins-controller-manager-6475d97999-d5r8b    2/2
Running            0             17m
kube-system          kube-apiserver-ar09-09-cyp                                1/1
Running            0             18m
kube-system          kube-controller-manager-ar09-09-cyp                       1/1
Running            2 (22m ago)   29m
kube-system          kube-multus-ds-amd64-kq44m                                1/1
Running            1 (21m ago)   23m
kube-system          kube-multus-ds-amd64-xllwr                                1/1
Running            1 (21m ago)   23m
kube-system          kube-proxy-64dqd                                          1/1
Running            1 (22m ago)   29m
kube-system          kube-proxy-pxdzs                                          1/1
Running            2 (21m ago)   29m
kube-system          kube-scheduler-ar09-09-cyp                                1/1
Running            0             6m23s
kube-system          kubernetes-dashboard-548847967d-slhpt                     1/1
Running            1 (22m ago)   22m
kube-system          kubernetes-metrics-scraper-6d49f96c97-fn2rc                1/1
Running            1 (22m ago)   22m
kube-system          nginx-proxy-ar09-01-cyp                                   1/1
Running            2 (21m ago)   29m
kube-system          node-feature-discovery-controller-5bb85dfdbb-pb8m4        1/1
Running            0             18m
kube-system          node-feature-discovery-worker-7g5l2                       1/1
Running            0             18m
kube-system          tas-telemetry-aware-scheduling-688d74f657-dpr6c           1/1
Running            0             6m29s
minio-operator       console-6c9557b87d-tkf4s                                  1/1
Running            0             5m9s
minio-operator       minio-operator-7885bb8d4-2flpf                            1/1
Running            0             5m9s
minio-operator       minio-operator-7885bb8d4-ddccf                            1/1
Running            0             5m9s
minio-operator       minio-operator-7885bb8d4-hwqxz                            1/1
Running            0             5m9s
minio-operator       minio-operator-7885bb8d4-zwv6b                            1/1
Running            0             5m9s
```

```
minio-tenant              minio-tenant-ss-0-0                                  1/1
Running             0             4m28s
monitoring                node-exporter-7bc74                                  2/2
Running             0             7m23s
monitoring                node-exporter-wbxn2                                  2/2
Running             0             7m23s
monitoring                prometheus-k8s-0                                     4/4
Running             0             7m21s
monitoring                prometheus-operator-bf54b8f56-hcc72                  2/2
Running             0             7m25s
monitoring                telegraf-67nbn                                       2/2
Running             0             5m49s
sriov-network-operator    sriov-device-plugin-z8bjj                            1/1
Running             0             12m
sriov-network-operator    sriov-network-config-daemon-rkmg2                    3/3
Running             0             17m
sriov-network-operator    sriov-network-operator-98db5fcbf-6nzcg               1/1
Running             0             17m

sriov-network-operator    sriov-network-operator-bb8ff65d9-2td74               1/1
Running     0          40h
```

5.2 Check Intel SST-BF Configuration on 2nd Generation Intel Xeon Scalable Processor

The Intel SST-BF feature enables base frequency configuration, which allows some cores to run at a higher guaranteed base frequency than others, if such option is required. It provides three different configuration modes:

- sst_bf_mode: s - set high priority cores to 2700/2800 minimum and 2700/2800 maximum and set normal priority cores to 2100 minimum and 2100 maximum.
- sst_bf_mode: m - set P1 on all cores (2300 minimum and 2300 maximum).
- sst_bf_mode: r - revert cores to minimum/Turbo (set all cores to 800 minimum and 3900 maximum).

To verify, that Intel SST-BF was configured as expected, use the following command.

```
# sst-bf.py -i
```

The output should show the current Intel SST-BF frequency information, as shown below with mode `s`.

```
Name = 6252N
CPUs = 96
Base = 2300
     |------sysfs-------|
Core | base   max   min |
-----|------------------|
   0 | 2100  2100  2100 |
   1 | 2800  2800  2800 |
   2 | 2800  2800  2800 |
   3 | 2100  2100  2100 |
(...)
  94 | 2800  2800  2800 |
  95 | 2100  2100  2100 |
-----|------------------|
```

To learn more about Intel SST-BF, visit https://github.com/intel/CommsPowerManagement.

5.3 Check Intel SST on 3rd Generation Intel Xeon Scalable Processor

The steps in this section describe how to verify Intel SST.

5.3.1 Check Intel SST-BF Configuration

To display Intel SST-BF properties, use the following command.

```
root@sdp1146:~# intel-speed-select base-freq info -l 0
Intel® Speed Select Technology
Executing on CPU model:106[0x6a]
 package-0
  die-0
    cpu-0
      speed-select-base-freq-properties
        high-priority-base-frequency(MHz):2400
        high-priority-cpu-mask:00000000,000030cc,cc0c0330
        high-priority-cpu-list:4,5,8,9,18,19,26,27,30,31,34,35,38,39,44,45
        low-priority-base-frequency(MHz):1800
```

```
        tjunction-temperature(C):105
        thermal-design-power(W):185
 package-1
  die-0
    cpu-48
      speed-select-base-freq-properties
        high-priority-base-frequency(MHz):2400
        high-priority-cpu-mask:000c0f3c,c3c00000,00000000
        high-priority-cpu-list:54,55,56,57,62,63,66,67,68,69,72,73,74,75,82,83
        low-priority-base-frequency(MHz):1800
        tjunction-temperature(C):105
        thermal-design-power(W):185
```

To help ensure that Intel SST-BF is enabled, check the CPU base frequency.

```
root@sdp1146:~# cat /sys/devices/system/cpu/cpu0/cpufreq/base_frequency
1800000
root@sdp1146:~# cat /sys/devices/system/cpu/cpu4/cpufreq/base_frequency
2400000
root@sdp1146:~# cat /sys/devices/system/cpu/cpu5/cpufreq/base_frequency
2400000
```

CPUs 4 and 5 are marked as high priority CPUs and have base frequency of 2.4 GHz.

CPU 0 is marked as low priority CPU and has base frequency of 1.8 GHz.

5.3.2 Check Intel SST-CP

Intel SST-CP enables a user to set up to four classes of service (CLOS) and assign CPUs to certain CLOSes.

To verify the correctness of CLOS 0, use the following command.

```
root@sdp1146:~# intel-speed-select core-power get-config -c 0
Intel® Speed Select Technology
Executing on CPU model:106[0x6a]
 package-0
  die-0
    cpu-0
      core-power
        clos:0
        epp:0
        clos-proportional-priority:0
        clos-min:2400 MHz
        clos-max:Max Turbo frequency
        clos-desired:0 MHz
 package-1
  die-0
    cpu-48
      core-power
        clos:0
        epp:0
        clos-proportional-priority:0
        clos-min:2400 MHz
        clos-max:Max Turbo frequency
        clos-desired:0 MHz
root@sdp1146:~/linux/tools/power/x86/intel-speed-select#
```

To verify the CLOS assignment for CPUs 0, 4, and 5, use the following command.

```
root@sdp1146:~# intel-speed-select -c 0,4-5 core-power get-assoc
Intel® Speed Select Technology
Executing on CPU model:106[0x6a]
 package-0
  die-0
    cpu-0
      get-assoc
        clos:3
 package-0
  die-0
    cpu-4
      get-assoc
        clos:0
```

```
 package-0
  die-0
    cpu-5
      get-assoc
        clos:0
root@sdp1146:~/linux/tools/power/x86/intel-speed-select#
```

To learn more about Intel SST-CP or Intel SST-CP classes of service, refer to: https://www.kernel.org/doc/html/latest/admin-guide/pm/intel-speed-select.html#intel-r-speed-select-technology-core-power-intel-r-sst-cp

5.4 Check Intel SST-PP with Intel SST-TF on 3rd Generation Intel Xeon Scalable Processors

To verify the availability of Intel SST-PP and its features, use the following command.

```
root@sdp1146:~/linux/tools/power/x86/intel-speed-select#
[root@as09-35-ac ~]# intel-speed-select –info
Intel(R) Speed Select Technology
Executing on CPU model:143[0x8f]
Platform: API version : 1
Platform: Driver version : 1
Platform: mbox supported : 1
Platform: mmio supported : 1
Intel(R) SST-PP (feature perf-profile) is supported
TDP level change control is unlocked, max level: 3
Intel(R) SST-TF (feature turbo-freq) is supported
Intel(R) SST-BF (feature base-freq) is supported
Intel(R) SST-CP (feature core-power) is supported
```

To verify that Intel SST-PP is unlocked in the BIOS, use the following command.

```
[root@as09-35-ac ~]# intel-speed-select perf-profile get-lock-status
Intel(R) Speed Select Technology
Executing on CPU model:143[0x8f]
Caching topology information
 package-0
  die-0
    cpu-0
      get-lock-status:unlocked
```

To confirm that the statuses of Intel SST-BF, Intel SST-CP, and Intel SST-TF are `enabled` or `disabled`, which must match the value of SST-PP in `host_vars`, and to check the properties of `perf-level`, use the following commands.

```
[root@as09-35-ac ~]# intel-speed-select perf-profile info -l 0 2>&1 | grep 'speed-select'
        speed-select-turbo-freq:enabled
        speed-select-base-freq:enabled
        speed-select-core-power:enabled
        speed-select-base-freq-properties
        speed-select-turbo-freq-properties
          speed-select-turbo-freq-clip-frequencies

[root@as09-35-ac ~]# intel-speed-select perf-profile get-config-levels
Intel(R) Speed Select Technology
Executing on CPU model:143[0x8f]
 package-0
  die-0
    cpu-0
      get-config-levels:3
```

Note: `config-levels` must be `get-config-levels:3` (Intel SST-BF, Intel SST-CP, and Intel SST-TF). `get-config-levels:0` means misconfiguration of setup in software or BIOS.

Set turbo status `on` and `off` to verify busy workload frequency ranges dynamically when Intel SST-PP is configured. For example, if CPUs 0,40,1,41,2,42,3,43,5,45,8,48,9,49,10,50,11,51,13,53 are defined in `host_vars` for Intel SST-PP to get 100 MHz boost, use the following commands for verification.

```
[root@as09-35-ac ~]# echo 1 > /sys/devices/system/cpu/intel_pstate/no_turbo
[root@as09-35-ac ~]# turbostat -c 0,40,1,41,2,42,3,43,5,45,8,48,9,49,10,50,11,51,13,53 --show Package,Core,CPU,Bzy_MHz -i
1

Core        CPU     Bzy_MHz
-           -       798
0           0       800
```

```
0       40      803
1       1       800
1       41      799
2       2       800
2       42      803
3       3       800
3       43      803
5       5       800
5       45      801
8       8       800
8       48      803
9       9       800
9       49      802
10      10      800
10      50      798
11      11      800
11      51      802
13      13      800
13      53      800
```

```
[root@as09-35-ac ~]# echo 0 > /sys/devices/system/cpu/intel_pstate/no_turbo
[root@as09-35-ac ~]# turbostat -c 0,40,1,41,2,42,3,43,5,45,8,48,9,49,10,50,11,51,13,53 --show Package,Core,CPU,Bzy_MHz -i 1

Core    CPU     Bzy_MHz
-       -       2888
0       0       2896
0       40      2885
1       1       2884
1       41      2901
2       2       2899
2       42      2888
3       3       2900
3       43      2895
5       5       2889
5       45      2900
8       8       2901
8       48      2898
9       9       2857
9       49      2900
10      10      2900
10      50      2838
11      11      2900
11      51      2900
13      13      2900
13      53      2902
14      14      2901
```

Note that improved performance in a frequency range can be observed dynamically as it jumps from approximately 800 to approximately 2900 in CPUs when Intel SST-PP is configured with `turbo-freq`.[4][5]

5.5 Check DDP Profiles

DDP provides dynamic reconfiguration of the packet processing pipeline to meet specific use case needs on demand, adding new packet processing pipeline Configuration Profiles to a network adapter at runtime, without resetting or rebooting the server.

[4] Refer to https://software.intel.com/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

[5] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

5.5.1 Check DDP Profiles in Intel® Ethernet 700 Series Network Adapters

To verify that a correct DDP profile was loaded, use the command shown below.

```
~/ddp-tool# ./ddptool -a
Intel(R) Dynamic Device Personalization Tool
DDPTool version 2020.17.22.7
Copyright (C) 2019 - 2021 Intel Corporation.

NIC  DevId D:B:S.F      DevName          TrackId  Version      Name
==== ===== ============ ================ ======== ============ ==============================
001) 158B  0000:18:00.0 ens785f0         80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
002) 158B  0000:18:00.1 ens785f1         80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
003) 154C  0000:18:02.0 ens785f0v0       80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
004) 154C  0000:18:02.1 ens785f0v1       80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
005) 154C  0000:18:02.2 ens785f0v2       80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
006) 154C  0000:18:02.3 ens785f0v3       80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
007) 154C  0000:18:02.4 ens785f0v4       80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
008) 154C  0000:18:02.5 ens785f0v5       80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
009) 154C  0000:18:0A.0 N/A              80000008 1.0.3.0      GTPv1-C/U IPv4/IPv6 payload
010) 154C  0000:18:0A.1 N/A              -        -
011) 154C  0000:18:0A.2 N/A              -        -
012) 154C  0000:18:0A.3 N/A              -        -
013) 1592  0000:AF:00.0 ens801f0         -        -            -
014) 1592  0000:AF:00.1 ens801f1         -        -            -
```

Download `ddptool` at: https://github.com/intel/ddp-tool/tree/1.0.9.0.

5.5.2 Check DDP Profiles in Intel® Ethernet 800 Series Network Adapters

To verify that a correct DDP profile was loaded, use the command shown below.

```
~/ddp-tool# ./ddptool -a
Intel(R) Dynamic Device Personalization Tool
DDPTool version 2020.17.22.7
Copyright (C) 2019 - 2021 Intel Corporation.

NIC  DevId D:B:S.F      DevName          TrackId  Version      Name
==== ===== ============ ================ ======== ============ ==============================
001) 1592  0000:B1:00.0 ens801f0         C0000002 1.3.30.0     ICE COMMS Package
002) 1592  0000:B1:00.1 ens801f1         C0000002 1.3.30.0     ICE COMMS Package
003) 1889  0000:B1:01.0 ens801f0v0       C0000002 1.3.30.0     ICE COMMS Package
004) 1889  0000:B1:01.1 ens801f0v1       C0000002 1.3.30.0     ICE COMMS Package
005) 1889  0000:B1:01.2 ens801f0v2       C0000002 1.3.30.0     ICE COMMS Package
006) 1889  0000:B1:01.3 ens801f0v3       C0000002 1.3.30.0     ICE COMMS Package
007) 1889  0000:B1:01.4 ens801f0v4       C0000002 1.3.30.0     ICE COMMS Package
008) 1889  0000:B1:01.5 ens801f0v5       C0000002 1.3.30.0     ICE COMMS Package
009) 1889  0000:B1:11.0 N/A              C0000002 1.3.30.0     ICE COMMS Package
010) 1889  0000:B1:11.1 N/A              -        -
011) 1889  0000:B1:11.2 N/A              -        -
012) 1889  0000:B1:11.3 N/A              -        -
```

Download `ddptool` at: https://github.com/intel/ddp-tool/tree/1.0.9.0.

5.5.3 Check SR-IOV Resources

To verify that SR-IOV network resources are present, use the command shown below.

```
# kubectl get node <worker node name> -o json | jq '.status.allocatable'
{
  "cpu": "125",
  "ephemeral-storage": "353450007582",
  "hugepages-1Gi": "4Gi",
  "hugepages-2Mi": "256Mi",
  "intel.com/intel_sriov_dpdk_800_series": "6",
  "intel.com/intel_sriov_netdevice": "4",
  "memory": "518294736Ki",
  "pods": "110",
  "power.intel.com/balance-performance": "76",
  "power.intel.com/balance-performance-node1": "76",
  "power.intel.com/balance-power": "102",
```

```
  "power.intel.com/balance-power-node1": "102",
  "power.intel.com/performance": "51",
  "power.intel.com/performance-node1": "51",
  "qat.intel.com/generic": "32",
  "sgx.intel.com/enclave": "20",
  "sgx.intel.com/epc": "4261412864",
  "sgx.intel.com/provision": "20"
}
```

5.6 Check Node Feature Discovery

NFD is a Kubernetes add-on that detects and advertises hardware and software capabilities of a platform that can, in turn, be used to facilitate intelligent scheduling of a workload. NFD is one of the Intel technologies that supports targeting of intelligent configuration and capacity consumption of platform capabilities. NFD runs as a separate container on each individual node of the cluster, discovers capabilities of the node, and publishes these as node labels using the Kubernetes API. NFD only handles non-allocatable features.

To verify that NFD is running as expected, use the following command:

```
# kubectl get ds --all-namespaces | grep node-feature-discovery

kube-system node-feature-discovery-worker      1    1    1    1    1    <none>    61m
```

To check the labels created by NFD, use the following command:

```
# kubectl label node --list --all
Listing labels for Node./node1:
 feature.node.kubernetes.io/pci-0300_1a03.present=true
 feature.node.kubernetes.io/cpu-rdt.RDTCMT=true
 feature.node.kubernetes.io/cpu-pstate.status=active
 feature.node.kubernetes.io/kernel-config.NO_HZ=true
 feature.node.kubernetes.io/cpu-cpuid.FMA3=true
 feature.node.kubernetes.io/cpu-cpuid.AVX2=true
 feature.node.kubernetes.io/kernel-config.NO_HZ_FULL=true
 feature.node.kubernetes.io/cpu-cpuid.STIBP=true
 feature.node.kubernetes.io/cpu-cpuid.ADX=true
 feature.node.kubernetes.io/cpu-rdt.RDTMON=true
 kubernetes.io/arch=amd64
 feature.node.kubernetes.io/system-os_release.VERSION_ID=8.5
 feature.node.kubernetes.io/memory-numa=true
 feature.node.kubernetes.io/cpu-cpuid.SHA=true
 feature.node.kubernetes.io/intel.sgx=true
 feature.node.kubernetes.io/kernel-version.major=4
 feature.node.kubernetes.io/cpu-cpuid.AVX512VL=true
 feature.node.kubernetes.io/system-os_release.VERSION_ID.major=8
 sgx.configured=true
 feature.node.kubernetes.io/kernel-version.revision=0
 feature.node.kubernetes.io/cpu-cpuid.AVX=true
 feature.node.kubernetes.io/cpu-cpuid.AESNI=true
 feature.node.kubernetes.io/cpu-rdt.RDTL3CA=true
 feature.node.kubernetes.io/cpu-cpuid.VAES=true
 feature.node.kubernetes.io/cpu-cpuid.AVX512VNNI=true
 feature.node.kubernetes.io/cpu-sgx.enabled=true
 feature.node.kubernetes.io/cpu-cpuid.AVX512CD=true
 feature.node.kubernetes.io/cpu-cpuid.VPCLMULQDQ=true
 feature.node.kubernetes.io/network-sriov.capable=true
 feature.node.kubernetes.io/pci-1200_8086.sriov.capable=true
 feature.node.kubernetes.io/pci-1200_8086.present=true
 feature.node.kubernetes.io/cpu-cpuid.IBPB=true
 feature.node.kubernetes.io/cpu-cpuid.AVX512VPOPCNTDQ=true
 feature.node.kubernetes.io/network-sriov.configured=true
 storage=minio
 feature.node.kubernetes.io/cpu-cpuid.GFNI=true
 beta.kubernetes.io/arch=amd64
 feature.node.kubernetes.io/cpu-cpuid.WBNOINVD=true
 feature.node.kubernetes.io/system-os_release.ID=rhel
 feature.node.kubernetes.io/cpu-cpuid.AVX512BITALG=true
 feature.node.kubernetes.io/storage-nonrotationaldisk=true
 feature.node.kubernetes.io/cpu-pstate.scaling_governor=performance
```

```
 app=kmra
 feature.node.kubernetes.io/cpu-pstate.turbo=true
 feature.node.kubernetes.io/cpu-cpuid.AVX512DQ=true
 feature.node.kubernetes.io/cpu-cpuid.AVX512BW=true
 feature.node.kubernetes.io/kernel-version.full=4.18.0-348.el8.x86_64
 feature.node.kubernetes.io/cpu-cpuid.AVX512VBMI2=true
 kubernetes.io/hostname=node1
 feature.node.kubernetes.io/cpu-cpuid.AVX512VBMI=true
 feature.node.kubernetes.io/cpu-rdt.RDTMBA=true
 qat.configured=true
 feature.node.kubernetes.io/cpu-cpuid.VMX=true
 feature.node.kubernetes.io/pci-0b40_8086.sriov.capable=true
 feature.node.kubernetes.io/intel.qat=true
 feature.node.kubernetes.io/pci-0b40_8086.present=true
 feature.node.kubernetes.io/cpu-cstate.enabled=true
 feature.node.kubernetes.io/cpu-cpuid.AVX512F=true
 feature.node.kubernetes.io/cpu-hardware_multithreading=true
 kubernetes.io/os=linux
 beta.kubernetes.io/os=linux
 intel.power.node=true
 feature.node.kubernetes.io/kernel-version.minor=18
 feature.node.kubernetes.io/cpu-cpuid.AVX512IFMA=true
 feature.node.kubernetes.io/system-os_release.VERSION_ID.minor=5
 feature.node.kubernetes.io/cpu-rdt.RDTMBM=true
Listing labels for Node./controller:
 kubernetes.io/arch=amd64
 kubernetes.io/hostname=controller
 kubernetes.io/os=linux
 node-role.kubernetes.io/control-plane=
 node-role.kubernetes.io/master=
 node.kubernetes.io/exclude-from-external-load-balancers=
 beta.kubernetes.io/arch=amd64
 beta.kubernetes.io/os=linux
```

5.7 Check Topology Manager

An increasing number of systems use a combination of CPUs and hardware accelerators to support latency-critical execution and high-throughput parallel computation. These include workloads in fields such as telecommunications, scientific computing, machine learning, financial services, and data analytics. Such hybrid systems comprise a high-performance environment.

To help extract the best performance[6], optimizations related to CPU isolation, memory, and device locality are required. However, in Kubernetes these optimizations are handled by a disjoint set of components.[7]

Topology Manager is a Kubelet component that aims to coordinate the set of components that are responsible for these optimizations.

Topology Manager supports its allocation policies via a Kubelet flag, `--topology-manager-policy`. There are four supported policies:

- **none:** Kubelet does not perform any topology alignment.
- **best-effort (default in BMRA deployment script):** Using resource availability reported by hint providers for each container in a guaranteed pod, the Topology Manager stores the preferred NUMA node affinity for that container. If the affinity is not preferred, Topology Manager stores this and admits the pod to the node anyway. The hint providers can then use this information when making the resource allocation decision.
- **restricted:** Using resource availability reported by hint providers for each container in a guaranteed pod, the Topology Manager stores the preferred NUMA node affinity for that container. If the affinity is not preferred, Topology Manager rejects this pod from the node. This results in a pod in a terminated state with a pod admission failure.
  After the pod is in a terminated state, the Kubernetes scheduler will not attempt to reschedule the pod. We recommend you use a ReplicaSet or Deployment to trigger a redeploy of the pod. Alternatively, you could implement an external control loop to trigger a redeployment of pods that have the Topology Affinity error.
  If the pod is admitted, the hint providers can then use this information when making the resource allocation decision.

[6] Refer to https://software.intel.com/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

[7] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

- **single-numa-node:** Using resource availability reported by hint providers for each container in a guaranteed pod, the Topology Manager determines if a single NUMA node affinity is possible. If it is, Topology Manager stores this and the hint providers can then use this information when making the resource allocation decision. If this is not possible, however, then the Topology Manager rejects the pod from the node. This results in a pod in a terminated state with a pod admission failure.
  After the pod is in a terminated state, the Kubernetes scheduler will not attempt to reschedule the pod. It is recommended to use a deployment with replicas to trigger a redeploy of the pod. An external control loop could be also implemented to trigger a redeployment of pods that have the Topology Affinity error.

To verify that Topology Manager is running as expected, use the following command:

```
# journalctl | grep topology_manager

Feb 03 09:30:38 ar09-09-cyp kubelet[79709]: I0203 09:30:38.418222   79709
topology_manager.go:133] "Creating topology manager with policy per scope"
topologyPolicyName="best-effort" topologyScopeName="container"
Feb 03 09:30:38 ar09-09-cyp kubelet[79709]: I0203 09:30:38.620193   79709
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:30:38 ar09-09-cyp kubelet[79709]: I0203 09:30:38.636035   79709
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:30:38 ar09-09-cyp kubelet[79709]: I0203 09:30:38.652310   79709
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:30:44 ar09-09-cyp kubelet[80859]: I0203 09:30:44.589837   80859
topology_manager.go:133] "Creating topology manager with policy per scope"
topologyPolicyName="best-effort" topologyScopeName="container"
Feb 03 09:30:44 ar09-09-cyp kubelet[80859]: I0203 09:30:44.761846   80859
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:30:44 ar09-09-cyp kubelet[80859]: I0203 09:30:44.761979   80859
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:30:44 ar09-09-cyp kubelet[80859]: I0203 09:30:44.762071   80859
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:30:57 ar09-09-cyp kubelet[80859]: I0203 09:30:57.175938   80859
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:31:04 ar09-09-cyp kubelet[85026]: I0203 09:31:04.125414   85026
topology_manager.go:133] "Creating topology manager with policy per scope"
topologyPolicyName="best-effort" topologyScopeName="container"
Feb 03 09:31:04 ar09-09-cyp kubelet[85026]: I0203 09:31:04.300730   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:31:04 ar09-09-cyp kubelet[85026]: I0203 09:31:04.300891   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:31:04 ar09-09-cyp kubelet[85026]: I0203 09:31:04.301299   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:31:05 ar09-09-cyp kubelet[85026]: I0203 09:31:05.152963   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:31:45 ar09-09-cyp kubelet[85026]: I0203 09:31:45.782237   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:38:43 ar09-09-cyp kubelet[85026]: I0203 09:38:43.474942   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:39:08 ar09-09-cyp kubelet[85026]: I0203 09:39:08.233812   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:39:11 ar09-09-cyp kubelet[85026]: I0203 09:39:11.927488   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:39:26 ar09-09-cyp kubelet[85026]: I0203 09:39:26.554582   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:39:29 ar09-09-cyp kubelet[85026]: I0203 09:39:29.747469   85026
topology_manager.go:200] "Topology Admit Handler"
Feb 03 09:39:30 ar09-09-cyp kubelet[85026]: I0203 09:39:30.338807   85026
topology_manager.go:200] "Topology Admit Handler"
...
```

5.7.1 Change Topology Manager Policy: Redeploy Kubernetes Playbook

This section describes one of two ways to change Topology Manager policy configuration after cluster deployment, by redeploying the Kubernetes playbook.

1. Update the `group_vars/all.yml` file.

```
...
# Enable Kubernetes built-in Topology Manager
topology_manager_enabled: true
# There are four supported policies: none, best-effort, restricted, single-numa-node.
```

```
topology_manager_policy: "single-numa-node"
...
```

2. Execute the `ansible-playbook` command to apply the new configuration cluster-wide.

```
# ansible-playbook -i inventory.ini playbooks/k8s/k8s.yml
```

5.7.2 Change Topology Manager Policy: Manually Update Kubelet Flags

This section describes a method of changing Topology Manager policy configuration after cluster deployment, by manually updating Kubelet flags on a specific node.

1. Log in to the worker node via SSH, for example:

```
# ssh <worker node name>
```

2. Edit the kubelet configuration in the `/etc/kubernetes/kubelet-config.yaml` file.

```
(...)
topologyManagerPolicy: single-numa-node
(...)
```

3. Restart the Kubelet service.

```
# systemctl restart kubelet
```

5.8 Check Intel Device Plugins for Kubernetes

Like other vendors, Intel provides many hardware devices that help deliver efficient acceleration of graphics, computation, data processing, more security, and compression. Those devices optimize hardware for specific tasks, which saves CPU cycles for other workloads and typically results in performance gains.[8] The Kubernetes device plugin framework provides a vendor-independent solution for hardware devices. Intel has developed a set of device plugins that complies with the Kubernetes device plugin framework and allows users to request and consume hardware devices across Kubernetes clusters such as Intel® QuickAssist Technology, GPUs, DSAs, and FPGAs. The detailed documentation and code are available at:

- Documentation: https://builders.intel.com/docs/networkbuilders/intel-device-plugins-for-kubernetes-appnote.pdf
- Code: https://github.com/intel/intel-device-plugins-for-kubernetes

5.8.1 Check SR-IOV Network Device Plugin

The SR-IOV network device plugin discovers, filters, and exposes VFs on nodes in a cluster. The plugin creates consumable resources based on custom filters, which provides a flexible way of grouping VFs according to a set of selectors.

Using the default configuration, two resources are created. Both resources include VFs from common Intel network adapters but differentiate based on the driver used. Following a successful deployment, the resources are visible as shown below.

```
# kubectl get node <worker node name> -o json | jq ".status.allocatable"

  "cpu": "125",
  "ephemeral-storage": "353450007582",
  "hugepages-1Gi": "4Gi",
  "hugepages-2Mi": "256Mi",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_1": "1",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_2": "4",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_3": "1",
  "intel.com/ar09_01_cyp_ens801f1_intelnics_1": "4",
  "memory": "518294736Ki",
  "pods": "110",
  "power.intel.com/balance-performance": "76",
  "power.intel.com/balance-performance-node1": "76",
  "power.intel.com/balance-power": "102",
  "power.intel.com/balance-power-node1": "102",
  "power.intel.com/performance": "51",
  "power.intel.com/performance-node1": "51",
  "qat.intel.com/generic": "32",
  "sgx.intel.com/enclave": "20",
  "sgx.intel.com/epc": "4261412864",
  "sgx.intel.com/provision": "20"
}
```

In the above, there are three SR-IOV network device plugin resources: "`intel.com/intel_sriov_dpdk_700_series`", "`intel.com/intel_sriov_dpdk_800_series`" and "`intel.com/intel_sriov_netdevice`". The netdevice VFs are bound

[8] Refer to http://software.intel.com/en-us/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

to a kernel driver and can be configured using the SR-IOV CNI Plugin. The dpdk VFs are bound to a DPDK driver, which allows an application to operate in userspace, bypassing the kernel network stack for ultra-high performance.[9]

To check allocation of a DPDK resource, create the following pod.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-sriov-dpdk-1
spec:
  containers:
  - name: pod-sriov-dpdk-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
    resources:
      requests:
        intel.com/intel_sriov_dpdk_700_series: '1'
      limits:
        intel.com/intel_sriov_dpdk_700_series: '1'
```

After the pod is running, check that the resource is listed in the environment of the pod.

```
# kubectl exec pod-sriov-dpdk-1 -- env | grep PCIDEVICE
PCIDEVICE_INTEL_COM_INTEL_SRIOV_DPDK_700_SERIES=0000:86:02.1
```

The same approach can be used to check the netdevice resources. If you used the default configuration, the SR-IOV CNI and example network attachment definition were installed and can be used as well.

```
# kubectl get net-attach-def -all-namespaces
NAME               AGE
sriov-net          3d23h
```

```
# kubectl describe net-attach-def sriov-net | grep Annotations
Annotations:  k8s.v1.cni.cncf.io/resourceName: intel.com/intel_sriov_netdevice
```

Using this information, create a pod that assigns a `netdevice` VF from the SR-IOV network device plugin and creates an interface in the pod using the SR-IOV CNI plugin.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-sriov-netdevice-1
  annotations:
    k8s.v1.cni.cncf.io/networks: sriov-net
spec:
  containers:
  - name: pod-sriov-netdevice-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
    resources:
      requests:
        intel.com/intel_sriov_netdevice: '1'
      limits:
        intel.com/intel_sriov_netdevice: '1'
```

Start by checking that the VF has been added to the pod, and then check that the interface has been created through SR-IOV CNI.

```
# kubectl exec pod-sriov-netdevice-1 -- env | grep PCIDEVICE
PCIDEVICE_INTEL_COM_INTEL_SRIOV_NETDEVICE=0000:86:0a.1
```

```
# kubectl exec pod-sriov-netdevice-1 -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: eth0@if69: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP group default
    link/ether c6:0d:c3:a5:57:15 brd ff:ff:ff:ff:ff:ff link-netnsid 0
```

[9] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

```
    inet 10.244.1.26/24 brd 10.244.1.255 scope global eth0
       valid_lft forever preferred_lft forever
16: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 7e:ad:44:41:91:e5 brd ff:ff:ff:ff:ff:ff
    inet 10.56.217.172/24 brd 10.56.217.255 scope global net1
       valid_lft forever preferred_lft forever
```

5.8.2 Check QAT Device Plugin

The Intel® QuickAssist Technology (Intel® QAT) device plugin discovers and exposes QAT device VFs as consumable resources in Kubernetes[10]. It works like the SR-IOV network device plugin but provides access to accelerated cryptographic and compression features.

If enabled and supported, QAT resources show up as a node resource.

```
# kubectl get node <work node name> -o json | jq '.status.allocatable'

{
  "cpu": "125",
  "ephemeral-storage": "353450007582",
  "hugepages-1Gi": "4Gi",
  "hugepages-2Mi": "256Mi",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_1": "1",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_2": "4",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_3": "1",
  "intel.com/ar09_01_cyp_ens801f1_intelnics_1": "4",
  "memory": "518294736Ki",
  "pods": "110",
  "power.intel.com/balance-performance": "76",
  "power.intel.com/balance-performance-node1": "76",
  "power.intel.com/balance-power": "102",
  "power.intel.com/balance-power-node1": "102",
  "power.intel.com/performance": "51",
  "power.intel.com/performance-node1": "51",
  "qat.intel.com/generic": "32",
  "sgx.intel.com/enclave": "20",
  "sgx.intel.com/epc": "4261412864",
  "sgx.intel.com/provision": "20"
}
```

Now create a pod that requests a VF from the above resource.

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-qat-1
spec:
  containers:
  - name: pod-qat-dpdk-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
    resources:
      requests:
        qat.intel.com/generic: '1'
      limits:
        qat.intel.com/generic: '1'
```

After the pod is running, verify that the VF was correctly assigned to the pod.

```
kubectl exec pod-qat-1 -- env | grep QAT
QAT0=0000:3e:02.1
```

To further test the VFs, an application that supports offloading and acceleration is required, which can be done through DPDK. More information and examples can be found here: https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/demo

[10] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

5.8.3 Check SGX Device Plugin

The Intel® SGX device plugin discovers and exposes SGX device nodes to kubelet as consumable resources in Kubernetes[11].

If enabled and supported, SGX resources show up as a node resource.

```
# kubectl get node <worker node name> -o json | jq '.status.allocatable'

{
  "cpu": "125",
  "ephemeral-storage": "353450007582",
  "hugepages-1Gi": "4Gi",
  "hugepages-2Mi": "256Mi",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_1": "1",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_2": "4",
  "intel.com/ar09_01_cyp_ens801f0_intelnics_3": "1",
  "intel.com/ar09_01_cyp_ens801f1_intelnics_1": "4",
  "memory": "518294736Ki",
  "pods": "110",
  "power.intel.com/balance-performance": "76",
  "power.intel.com/balance-performance-node1": "76",
  "power.intel.com/balance-power": "102",
  "power.intel.com/balance-power-node1": "102",
  "power.intel.com/performance": "51",
  "power.intel.com/performance-node1": "51",
  "qat.intel.com/generic": "32",
  "sgx.intel.com/enclave": "20",
  "sgx.intel.com/epc": "4261412864",
  "sgx.intel.com/provision": "20"
}
```

5.8.4 Check DSA Device Plugin

The Intel® Data Streaming Accelerator (Intel® DSA) device plugin discovers and exposes DSA work queues as consumable resources in Kubernetes[12].

Configured DSA work queues should be visible on the system.

```
# ls /sys/bus/dsa/devices/ | grep "dsa\!wq"
dsa!wq0.0  dsa!wq0.1  dsa!wq0.2  dsa!wq0.3  dsa!wq0.4  dsa!wq0.5  dsa!wq0.6  dsa!wq0.7
dsa!wq1.0  dsa!wq1.1  dsa!wq1.2  dsa!wq1.3  dsa!wq1.4  dsa!wq1.5  dsa!wq1.6  dsa!wq1.7
dsa!wq2.0  dsa!wq2.1  dsa!wq2.2  dsa!wq2.3  dsa!wq2.4  dsa!wq2.5  dsa!wq2.6  dsa!wq2.7
dsa!wq3.0  dsa!wq3.1  dsa!wq3.2  dsa!wq3.3  dsa!wq3.4  dsa!wq3.5  dsa!wq3.6  dsa!wq3.7
```

If enabled and supported, DSA resources show up as node resources.

```
# kubectl get node <worker ndoe name> -o json | jq '.status.allocatable'

{
  "cpu": "77",
  "dsa.intel.com/wq-user-dedicated": "16",
  "dsa.intel.com/wq-user-shared": "160",
  "ephemeral-storage": "282566437625",
  "hugepages-1Gi": "4Gi",
  "hugepages-2Mi": "0",
  "memory": "125570920Ki",
  "pods": "110"
}
```

5.8.5 Check GPU Device Plugin

Intel® GPU Device Plugin facilitates Kubernetes workload offloading by providing access to Intel discrete (Xe) and integrated GPU hardware device files. It can be used for many applications including video transcoding, video analytics, online gaming, and AI acceleration, etc.

BMRA deploys GPU Device Plugin on the node with GPU capability. Refer to Section 5.13 for GPU capability details. You can set the value of `configure_gpu` in `host_vars` to enable the GPU Device Plugin on the node. If all the nodes are disabled, the

[11] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

[12] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

`gpu_dp_enabled` in `group_vars` must be disabled. GPU device plugin is supported by regional_dc profile and optionally supported by full_nfv and minimal profiles.

To check the installation of GPU Device Plugin, use the following command:

```
root@at09-24-wp:~# kubectl get node <work node name> -o json |jq .metadata.labels
{
  "beta.kubernetes.io/arch": "amd64",
  "beta.kubernetes.io/os": "linux",
  "feature.node.kubernetes.io/cpu-cpuid.ADX": "true",
  ……
  "feature.node.kubernetes.io/intel.qat": "true",
  "feature.node.kubernetes.io/kernel-version.full": "5.4.48",
  "feature.node.kubernetes.io/kernel-version.major": "5",
  "feature.node.kubernetes.io/kernel-version.minor": "4",
  "feature.node.kubernetes.io/kernel-version.revision": "48",
  ……
  "gpu.intel.com/cards": "card0.card1.card2.card3",
  "kubernetes.io/arch": "amd64",
  "kubernetes.io/hostname": "as09-16-wpr",
  "kubernetes.io/os": "linux",
  "node-role.kubernetes.io/worker": ""
}
```

5.9 Check Networking Features (After Installation)

This section describes how to verify certain CNI plugins.

5.9.1 Check Multus CNI Plugin

To verify that the Multus CNI Plugin is running, an additional network can be created using the basic CNI plugins installed as part of the playbooks. For this example, the `macvlan` CNI is used. Start by creating a NetworkAttachmentDefinition (net-attach-def) using the provided template and update the **`{{ interface }}`** value to match an interface name on the worker nodes of the system, for example, `ens786f1`.[13]

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: macvlan-multus-1
spec:
  config: '{
            "cniVersion": "0.3.0",
            "type": "macvlan",
            "master": "{{ interface }}",
            "mode": "bridge",
            "ipam": {
                "type": "host-local",
                "ranges": [
                    [ {
                         "subnet": "10.10.0.0/16",
                         "rangeStart": "10.10.1.20",
                         "rangeEnd": "10.10.3.50",
                         "gateway": "10.10.0.254"
                    } ]
                ]
            }
        }'
```

After applying the configuration, verify that it has been created and is available in the cluster:

```
# kubectl get net-attach-def
NAME               AGE
macvlan-multus-1   4d1h
```

Following this, create a pod that requests an interface from the newly created `net-attach-def`:

```
apiVersion: v1
kind: Pod
```

[13] Refer to https://software.intel.com/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products.

See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

```
metadata:
  name: pod-macvlan-1
  annotations:
    k8s.v1.cni.cncf.io/networks: macvlan-multus-1
spec:
  containers:
  - name: pod-macvlan-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
```

After the pod is running, verify that the additional interface is available in the pod:

```
# kubectl exec pod-macvlan-1 -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: eth0@if71: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP group default
    link/ether 7e:33:5e:5b:1b:4f brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.244.1.28/24 brd 10.244.1.255 scope global eth0
       valid_lft forever preferred_lft forever
4: net1@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 8e:c0:49:08:8a:ab brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.10.1.21/16 brd 10.10.255.255 scope global net1
       valid_lft forever preferred_lft forever
```

5.9.2 Check SR-IOV CNI Plugin

Intel introduced the SR-IOV CNI plugin to allow a Kubernetes pod to be attached directly to an SR-IOV VF using the standard SR-IOV VF driver in the container host's kernel. Details on the SR-IOV CNI plugin can be found at: https://github.com/intel/sriov-cni

Verify the network using the following command:

```
# kubectl get net-attach-def -all-namespaces
NAME            AGE
sriov-net       4d3h
```

An example using the SR-IOV CNI Plugin can be found in Section 5.8.1. To test it again, create a pod as shown below:

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-sriov-netdevice-1
  annotations:
    k8s.v1.cni.cncf.io/networks: sriov-net
spec:
  containers:
  - name: pod-sriov-netdevice-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
    resources:
      requests:
        intel.com/intel_sriov_netdevice: '1'
      limits:
        intel.com/intel_sriov_netdevice: '1'
```

After the pod is running, verify that the additional network interface has been added to the pod:

```
# kubectl exec pod-sriov-netdevice-1 -- ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
3: eth0@if72: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue state UP group default
    link/ether 32:b8:e3:04:c8:93 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.244.1.29/24 brd 10.244.1.255 scope global eth0
       valid_lft forever preferred_lft forever
14: net1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether de:05:9d:bc:62:f3 brd ff:ff:ff:ff:ff:ff
    inet 10.56.217.173/24 brd 10.56.217.255 scope global net1
       valid_lft forever preferred_lft forever
```

5.9.3 Check Userspace CNI Plugin

The Userspace CNI is a CNI plugin designed to implement userspace networking, such as DPDK-based applications. The current implementation supports DPDK enhanced Open vSwitch (OVS-DPDK) and Vector Packet Processing (VPP) along with the Multus CNI plugin in Kubernetes for the bare metal container deployment model. It enhances the high-performance container networking solution and data plane acceleration for NFV environments.[14]

By default, OVS is installed as the vSwitch, alongside a `net-attach-def` to expose vhostuser resources through Userspace CNI in Kubernetes.

Verify that the resource is available using the following command:

```
# kubectl get net-attach-def
NAME            AGE
userspace-ovs   7d
```

With the `userspace-ovs` resource available, create a pod requesting an interface:

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-userspace-1
  annotations:
    k8s.v1.cni.cncf.io/networks: userspace-ovs
spec:
  containers:
  - name: pod-userspace-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
    volumeMounts:
    - mountPath: /vhu/
      name: shared-dir
  volumes:
  - name: shared-dir
    hostPath:
      path: /var/lib/cni/vhostuser/
```

After the pod is running, verify that the `vhostuser` socket has been added to the container:

```
# kubectl exec pod-userspace-1 -- ls /vhu/
e4c7e6fb63ec737f7aee3f451e79f7fba2cac6d212b88fb0725da1b9afed1cfb
```

Before checking OVS, check the node that the pod is deployed on:

```
# kubectl describe pod pod-userspace-1 | grep Node:
Node:         node1/<node IP>
```

Connect to the node using SSH, and check that the `vhostuser` socket and interface has been added to OVS:

```
# ovs-vsctl show
fbb7d4e6-6f93-4bcd-a254-ff10f1bd5fd7
    Bridge br0
        datapath_type: netdev
        Port br0
            Interface br0
                type: internal
        Port e4c7e6fb63ec-net1
            Interface e4c7e6fb63ec-net1
                type: dpdkvhostuser
    ovs_version: "2.13.0"
```

At this point, the `vhostuser` socket is ready to use in the pod. The steps for using VPP as the vSwitch are similar, but instead of the userspace CNI resource name `userspace-ovs`, use `userspace-vpp`.

More details and examples can be found here: https://github.com/intel/userspace-cni-network-plugin

5.9.4 Check Bond CNI Plugin

Bond CNI provides a method for aggregating multiple network interfaces into a single bonded interface inside a container in a Kubernetes pod. Linux bonding drivers provide several modes for interface bonding, such as round robin and active aggregation.

[14] Refer to https://software.intel.com/articles/optimization-notice for more information regarding performance and optimization choices in Intel software products. See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

Bond CNI integrates with Multus and the network attachment definition policy declaration. It can be used alongside Multus and SR-IOV CNI to provide failover for network interfaces in a Kubernetes cluster, to support increasing the total bandwidth available to a single container interface, or for other cases in which interface bonding is used.

To verify that Bond CNI is installed on the host, look for its binary in the `/opt/cni/bin` directory:

```
# ll /opt/cni/bin/bond
-rwxr-xr-x. 1 root root 3836352 Feb 27 13:03 /opt/cni/bin/bond
```

Assuming the cluster was created using the provided configuration default, there should be SR-IOV network resources on the node as shown below:

```
# kubectl get node node1 -o json | jq '.status.allocatable'
  "cpu": "93",
  "ephemeral-storage": "452220352993",
  "hugepages-1Gi": "4Gi",
  "intel.com/intel_sriov_dpdk_700_series": "2",
  "intel.com/intel_sriov_dpdk_800_series": "0",
  "intel.com/intel_sriov_netdevice": "4",
  "memory": "191733164Ki",
  "pods": "110",
  "qat.intel.com/generic": "32"
}
```

If there are `netdevice` VFs configured, create a simple SR-IOV CNI resource as shown:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: sriov-bond-net
  annotations:
    k8s.v1.cni.cncf.io/resourceName: intel.com/intel_sriov_netdevice
spec:
  config: '{
  "type": "sriov",
  "name": "sriov-network",
  "spoofchk":"off"
}'
```

Now create a Bond CNI resource:

```
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
  name: bond-net
spec:
  config: '{
  "type": "bond",
  "cniVersion": "0.3.1",
  "name": "bond-net",
  "ifname": "bond0",
  "mode": "active-backup",
  "failOverMac": 1,
  "linksInContainer": true,
  "miimon": "100",
  "links": [
     {"name": "net1"},
     {"name": "net2"}
  ],
  "ipam": {
    "type": "host-local",
    "subnet": "10.56.217.0/24",
    "routes": [{
      "dst": "0.0.0.0/0"
    }],
    "gateway": "10.56.217.1"
  }
}'
```

Verify that both `bond-net` and `sriov-bond-net` were created:

```
# kubectl get net-attach-def
NAME             AGE
bond-net         15s
```

```
sriov-bond-net   36s
```

Now create a pod requesting two VFs from `sriov-bond-net`, which is used to create a bonded interface from `bond-net` through Bond CNI:

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-bond-cni-1
  annotations:
    k8s.v1.cni.cncf.io/networks: '[
      {"name": "sriov-bond-net",
        "interface": "net1"
      },
      {"name": "sriov-bond-net",
        "interface": "net2"
      },
      {"name": "bond-net",
        "interface": "bond0"
      }]'
spec:
  containers:
  - name: pod-bond-cni-1
    image: docker.io/ubuntu/tools:latest
    command:
    - /sbin/init
    resources:
      requests:
        intel.com/intel_sriov_netdevice: '2'
      limits:
        intel.com/intel_sriov_netdevice: '2'
```

After the pod is running, verify that the two interfaces and the bonded interface were added using the command `ip a` inside the container. The two VFs are shown as `net1` and `net2`, and the bonded interface configured with an IP address is shown as `bond0`.

For more information on how to use Bond CNI, refer to: https://github.com/intel/bond-cni

5.10 Check Grafana Telemetry Visualization

BMRA deploys Grafana for telemetry visualization. It is available on every cluster node on port 30000. Due to security reasons, this port is not exposed outside the cluster by default. Default credentials are admin/admin and you should change the default password after first login.

The Grafana TLS certificate is signed by the cluster CA and it is available in `/etc/kubernetes/ssl/ca.crt`

Visit Grafana at `https://<node-ip>:30000/`

BMRA comes with a set of dashboards from the kube-prometheus project (https://github.com/prometheus-operator/kube-prometheus). Dashboards are available in the Dashboards -> Manage menu as shown in Figure 3.

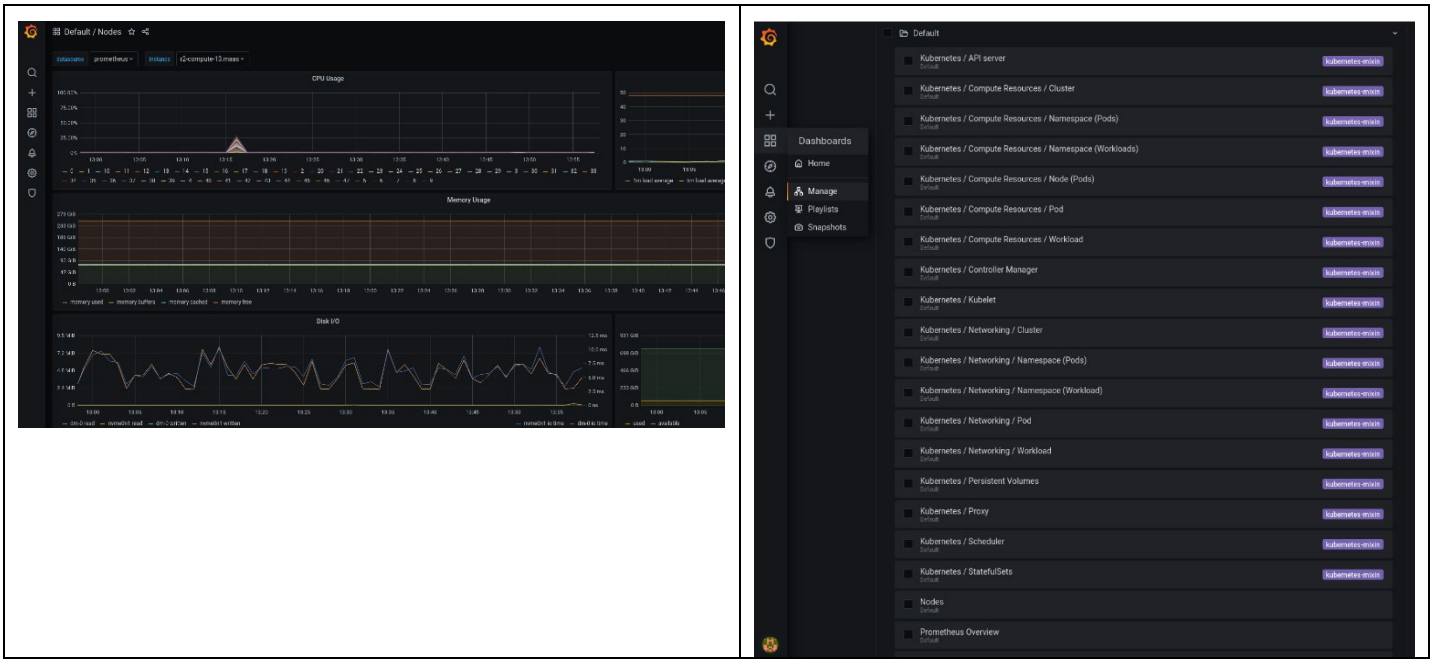

Figure 3. Grafana Dashboard Example

5.11 Check Telemetry Aware Scheduler

BMRA can deploy TAS Health Metric Demo Policy (https://github.com/intel/platform-aware-scheduling/blob/master/telemetry-aware-scheduling/docs/health-metric-example.md) when `tas_enable_demo_policy: true`, as shown below:

```
# Intel Telemetry Aware Scheduling
tas_enabled: true
tas_namespace: monitoring
# create and enable TAS demonstration policy: [true, false]
tas_enable_demo_policy: true
```

The Health Metric Demo Policy requires a Prometheus metric file to exist on the node and be read by Prometheus. For security reasons, BMRA does not deploy it in the `/tmp` directory, where every user has access. Instead, it is deployed in the `/opt/intel/tas-demo-policy/` directory with root-only access.

To verify that the policy has been deployed, use the command:

```
$ kubectl get taspolicies -n kube-system
NAME          AGE
demo-policy   4h
```

Details of this policy, including the rules and associated metrics, can be described with following command:

```
$ kubectl describe taspolicies demo-policy -n kube-system
```

To verify that the proper files exist on the worker node, use the following command:

```
$ cat /opt/intel/tas-demo-policy/test.prom
node_health_metric 0
```

The node health metric value indicates the following:

- When `node_health_metric = 0`, then it allows scheduling of pods on this node.
- When `node_health_metric = 2`, then the descheduler deschedules pods from this node.

5.11.1 Check Deschedule Policy

To see the impact of the descheduling policy, use a component called `descheduler`. For more details, visit https://github.com/intel/platform-aware-scheduling/blob/master/telemetry-aware-scheduling/docs/health-metric-example.md#seeing-the-impact

Start by checking the logs to verify that `node_health_metric` is `0` on worker nodes:

```
$ kubectl logs pod/tas-telemetry-aware-scheduling-xxxx-yyyy -n kube-system --tail=20
  "Evaluating demo-policy" component="controller"
  "controller1 health_metric = 0" component="controller"
  "worker1 health_metric = 0" component="controller"
```

Define and deploy a pod that is susceptible to the demo scheduling policy from a controller node:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: tas-test
  labels:
    app: demo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo
  template:
    metadata:
      labels:
        app: demo
        telemetry-policy: demo-policy
    spec:
      containers:
      - name: pod-tas-test
        image: ubuntu:focal
        imagePullPolicy: IfNotPresent
        command:
        - "/bin/bash"
        - "-c"
        args:
        - "tail -f /dev/null"
        resources:
          limits:
            telemetry/scheduling: 1
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: demo-policy
                    operator: NotIn
                    values:
                      - violating
```

The pod should deploy successfully and end up in state “running” as shown below:

```
$ kubectl get pods
  NAME                        READY   STATUS    RESTARTS   AGE
  tas-test-xxxx-yyyy          1/1     Running   0          90m
```

Set the `node_health_metric` to `2` (deschedule) on some or all worker nodes as follows:

```
$ echo 'node_health_metric 2' > /opt/intel/tas-demo-policy/test.prom
```

After a few seconds, check the logs to verify that the status has changed:

```
$ kubectl logs pod/tas-telemetry-aware-scheduling-xxxx-yyyy -n kube-system --tail=20
  "Evaluating demo-policy" component="controller"
  "controller1 health_metric = 0" component="controller"
  "worker1 health_metric = 2" component="controller"
  "worker1 violating demo-policy: health_metric Equals 2" component="controller"
  "Node worker1 violating demo-policy, " component="controller"
```

Create a policy file for descheduling pods on a controller node with the following content:

```
apiVersion: "descheduler/v1alpha1"
kind: "DeschedulerPolicy"
strategies:
  "RemovePodsViolatingNodeAffinity":
    enabled: true
    params:
      nodeAffinityType:
        - "requiredDuringSchedulingIgnoredDuringExecution"
```

Then run the descheduler from the same controller node with following command:

```
$ /opt/cek/sigs.k8s.io/descheduler/_output/bin/descheduler --policy-config-file
<path_to_policy_file> --kubeconfig /etc/kubernetes/admin.conf
```

Now check the status of the pod deployed previously:

```
$ kubectl get pods
  NAME                        READY   STATUS    RESTARTS   AGE
  tas-test-xxxx-yyyy          0/1     Pending   0          8m11s
```

The pod will be rescheduled onto a healthier node based on its TAS policy. If no other suitable nodes are available, the new pod fails to schedule as shown above.

5.12 Check Key Management Infrastructure with Intel SGX

To verify the Key Management infrastructure with SGX and use the private keys provisioned to Intel SGX enclaves, see Section and Section 13.1 for step-by-step instructions to set up and run the NGINX workload.

5.13 Check Intel® Server GPU Device and Driver

BMRA deploys the intel-gpu/kernel project from GitHub for the latest Intel® Server GPU kernel driver for media processing. To verify that the devices and drivers are present in the system with the correct configuration, perform the following actions.

After installation, both GPU device and kernel must be inspected for the function readiness. First confirm the i915 driver presence and that the ASPEED device is ignored. The kernel option `915.force_probe=*` helps ensure the i915 driver probes for the SG1 device and `i915.enable_guc=2` helps ensure that the device is enabled for SR-IOV.

```
# lsmod | grep i915
i915_spi               24576  0
mtd                    77824  17 i915_spi
i915                 2609152  0
video                  53248  1 i915
i2c_algo_bit           16384  1 i915
drm_kms_helper        217088  1 i915
drm                   610304  3 drm_kms_helper,i915

# cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.4.48+ root=/dev/sda1 ro crashkernel=auto rhgb quiet
i915.force_probe=* modprobe.blacklist=ast,snd_hda_intel i915.enable_guc=2
```

Verify the Intel Server GPU devices (4907) are present and using the i915 driver.

```
# lspci | grep -i VGA
02:00.0 VGA compatible controller: ASPEED Technology, Inc. ASPEED Graphics Family (rev 41)
1c:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
21:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
26:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
2b:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)

# lspci -n -v -s 1c:00.0
1c:00.0 0300: 8086:4907 (rev 01) (prog-if 00 [VGA controller])
        Subsystem: 8086:35cf
        Flags: bus master, fast devsel, latency 0, IRQ 423, NUMA node 0
        Memory at a8000000 (64-bit, non-prefetchable) [size=16M]
        Memory at 387e00000000 (64-bit, prefetchable) [size=8G]
        Expansion ROM at <ignored> [disabled]
        Capabilities: [40] Vendor Specific Information: Len=0c <?>
        Capabilities: [70] Express Endpoint, MSI 00
        Capabilities: [ac] MSI: Enable+ Count=1/1 Maskable+ 64bit+
        Capabilities: [d0] Power Management version 3
        Capabilities: [100] Latency Tolerance Reporting
        Kernel driver in use: i915
        Kernel modules: i915
```

Confirm that the Intel Server GPU kernel drivers are present on the system.

```
# ls /dev/dri/ -l
total 0
crw-rw----. 1 root video 226,   0 Apr  8 10:15 card0
crw-rw----. 1 root video 226,   1 Apr  8 10:15 card1
crw-rw----. 1 root video 226,   2 Apr  8 10:15 card2
crw-rw----. 1 root video 226,   3 Apr  8 10:15 card3
crw-rw----. 1 root video 226, 128 Apr  8 10:15 renderD128
crw-rw----. 1 root video 226, 129 Apr  8 10:15 renderD129
crw-rw----. 1 root video 226, 130 Apr  8 10:15 renderD130
crw-rw----. 1 root video 226, 131 Apr  8 10:15 renderD131
```

You are now ready to deploy transcode workloads to utilize the hardware components. For more information, see the Open Visual Cloud GitHub site: https://github.com/OpenVisualCloud/CDN-Transcode-Sample.

5.14 Check Intel QAT Engine with OpenSSL

Check the version of Intel QAT crypto engine present on the system.

```
# openssl engine -v qatengine
(qatengine) Reference implementation of QAT crypto engine(qat_sw) v0.6.10
     ENABLE_EXTERNAL_POLLING, POLL, ENABLE_HEURISTIC_POLLING,
     GET_NUM_REQUESTS_IN_FLIGHT, INIT_ENGINE
```

Run an OpenSSL speed test.

```
# openssl speed rsa2048
Doing 2048 bits private rsa's for 10s:
10364 2048 bits private RSA's in 9.97s
Doing 2048 bits public rsa's for 10s:
354887 2048 bits public RSA's in 9.98s
OpenSSL 1.1.1k  FIPS 25 Mar 2021
built on: Fri Nov 12 17:53:47 2021 UTC
options:bn(64,64) md2(char) rc4(16x,int) des(int) aes(partial) idea(int) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -O3 -O2 -g -pipe -Wall -
Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -fexceptions -fstack-
protector-strong -grecord-gcc-switches -specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -
specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -m64 -mtune=generic -fasynchronous-unwind-tables -
fstack-clash-protection -fcf-protection -Wa,--noexecstack -Wa,--generate-missing-build-
notes=yes -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -DOPENSSL_USE_NODELETE -DL_ENDIAN -
DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -
DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -
DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP_NISTZ256_ASM -
DX25519_ASM -DPOLY1305_ASM -DZLIB -DNDEBUG -DPURIFY -DDEVRANDOM="\"/dev/urandom\"" -
DSYSTEM_CIPHERS_FILE="/etc/crypto-policies/back-ends/openssl.config"
                  sign    verify    sign/s verify/s
rsa 2048 bits 0.000962s 0.000028s   1039.5  35559.8
```

Repeat the OpenSSL speed test with QAT engine for increased performance.

```
# openssl speed -engine qatengine -async_jobs 8 rsa2048
engine "qatengine" set.
Doing 2048 bits private rsa's for 10s:
42728 2048 bits private RSA's in 9.93s
Doing 2048 bits public rsa's for 10s:
797952 2048 bits public RSA's in 9.14s
OpenSSL 1.1.1k  FIPS 25 Mar 2021
built on: Fri Nov 12 17:53:47 2021 UTC
options:bn(64,64) md2(char) rc4(16x,int) des(int) aes(partial) idea(int) blowfish(ptr)
compiler: gcc -fPIC -pthread -m64 -Wa,--noexecstack -Wall -O3 -O2 -g -pipe -Wall -
Werror=format-security -Wp,-D_FORTIFY_SOURCE=2 -Wp,-D_GLIBCXX_ASSERTIONS -fexceptions -fstack-
protector-strong -grecord-gcc-switches -specs=/usr/lib/rpm/redhat/redhat-hardened-cc1 -
specs=/usr/lib/rpm/redhat/redhat-annobin-cc1 -m64 -mtune=generic -fasynchronous-unwind-tables -
fstack-clash-protection -fcf-protection -Wa,--noexecstack -Wa,--generate-missing-build-
notes=yes -specs=/usr/lib/rpm/redhat/redhat-hardened-ld -DOPENSSL_USE_NODELETE -DL_ENDIAN -
DOPENSSL_PIC -DOPENSSL_CPUID_OBJ -DOPENSSL_IA32_SSE2 -DOPENSSL_BN_ASM_MONT -
DOPENSSL_BN_ASM_MONT5 -DOPENSSL_BN_ASM_GF2m -DSHA1_ASM -DSHA256_ASM -DSHA512_ASM -
DKECCAK1600_ASM -DRC4_ASM -DMD5_ASM -DAESNI_ASM -DVPAES_ASM -DGHASH_ASM -DECP_NISTZ256_ASM -
DX25519_ASM -DPOLY1305_ASM -DZLIB -DNDEBUG -DPURIFY -DDEVRANDOM="\"/dev/urandom\"" -
DSYSTEM_CIPHERS_FILE="/etc/crypto-policies/back-ends/openssl.config"
                  sign    verify    sign/s verify/s
rsa 2048 bits 0.000232s 0.000011s   4302.9  87303.3
```

5.15 Check MinIO Operator/Console and Tenant

The Storage Configuration Profile deploys the MinIO operator/console and sample MinIO tenant via Helm charts. You can verify that the MinIO operator/console and the tenant are present on the system. As MinIO with distributed mode, it requires more than four nodes and one controller.

```
# kubectl get pods -n minio-operator
NAME                              READY   STATUS    RESTARTS   AGE
console-6c9557b87d-zzmgt          1/1     Running   0          12m
minio-operator-7885bb8d4-9l2sq    1/1     Running   0          12m
minio-operator-7885bb8d4-k4j8m    1/1     Running   0          12m
minio-operator-7885bb8d4-lcm56    1/1     Running   0          12m
```

```
minio-operator-7885bb8d4-xh5vp   1/1     Running   0          12m

# kubectl get pods -n minio-tenant
NAME                  READY   STATUS    RESTARTS   AGE
minio-tenant-ss-0-0   1/1     Running   0          12m
minio-tenant-ss-0-1   1/1     Running   0          12m
minio-tenant-ss-0-2   1/1     Running   0          12m
minio-tenant-ss-0-3   1/1     Running   0          12m
```

After the sample MinIO tenant is deployed successfully on four nodes, it requires four volumes per server. You must confirm that all volumes are properly bound.

```
# kubectl get pv
NAME                          CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLASS      REASON   AGE
…
mnt-data-1-am09-19-cyp        1Gi        RWO            Retain           Bound    minio-
tenant/data1-minio-tenant-ss-0-0   local-storage              19m
mnt-data-1-am09-22-cyp        1Gi        RWO            Retain           Bound    minio-
tenant/data2-minio-tenant-ss-0-3   local-storage              19m
mnt-data-1-am09-24-cyp        1Gi        RWO            Retain           Bound    minio-
tenant/data0-minio-tenant-ss-0-1   local-storage              19m
mnt-data-1-am09-26-cyp        1Gi        RWO            Retain           Bound    minio-
tenant/data0-minio-tenant-ss-0-2   local-storage              19m
…

kubectl get pvc -n minio-tenant
NAME                         STATUS   VOLUME                   CAPACITY   ACCESS MODES
STORAGECLASS    AGE
…
data0-minio-tenant-ss-0-0   Bound    mnt-data-2-am09-19-cyp   1Gi        RWO            local-
storage   20m
data1-minio-tenant-ss-0-0   Bound    mnt-data-1-am09-19-cyp   1Gi        RWO            local-
storage   20m
data2-minio-tenant-ss-0-0   Bound    mnt-data-4-am09-19-cyp   1Gi        RWO            local-
storage   20m
data3-minio-tenant-ss-0-0   Bound    mnt-data-3-am09-19-cyp   1Gi        RWO            local-
storage   20m
…
```

5.16 Check Intel Power Manager (Balance Performance Power-Profile & Sample Power-Pods)

Sample pods can be deployed by setting deploy_example_pods: true in group vars. Following are the results that can be obtained from Power Manager work

```
# kubectl get pods -n intel-power
NAME                                READY   STATUS    RESTARTS   AGE
balance-performance-power-pod       1/1     Running   0          33m
balance-power-power-pod             1/1     Running   0          33m
controller-manager-f584c9458-p5llp  1/1     Running   0          34m
performance-power-pod               1/1     Running   0          33m
power-node-agent-9dkch              2/2     Running   0          34m
```

(Note: each profile was deployed in a separate pod)

You can check the frequencies that will be set by balance-performance Power Profile

```
# kubectl get PowerProfiles -n intel-power balance-performance-ar09-01-cyp -o yaml
apiVersion: power.intel.com/v1alpha1
kind: PowerProfile
metadata:
  creationTimestamp: "2022-02-07T20:50:44Z"
  generation: 1
  name: balance-performance-node1
  namespace: intel-power
  resourceVersion: "4790"
  uid: 3bc5d223-f31e-4fdc-8c49-8a87148a014d
spec:
  epp: balance_performance
  max: 2700
```

```
  min: 2500
  name: balance-performance-node1
```

To obtain balance-performance cores, apply the Power Profile

```
# kubectl get PowerWorkloads -n intel-power balance-performance-ar09-01-cyp-workload -o yaml
apiVersion: power.intel.com/v1alpha1
kind: PowerWorkload
metadata:
  creationTimestamp: "2022-02-07T20:51:43Z"
  generation: 1
  name: balance-performance-node1-workload
  namespace: intel-power
  resourceVersion: "5090"
  uid: 19de2932-6ab6-4863-b664-764cc555e23d
spec:
  name: balance-performance-node1-workload
  nodeInfo:
    containers:
    - exclusiveCpus:
      - 4
      - 68
      id: 870e1d2eb4f971328d5030f97a647b8ee5fb7dae52daebec4714588e9a563667
      name: balance-performance-container
      pod: balance-performance-power-pod
      powerProfile: balance-performance-node1
    cpuIds:
    - 4
    - 68
    name: node1
  powerProfile: balance-performance-node1
```

If you want to check all the cores in your Power Nodes, you can use the following command

```
# kubectl get PowerNodes -A -o yaml
apiVersion: v1
items:
- apiVersion: power.intel.com/v1alpha1
  kind: PowerNode
  metadata:
    creationTimestamp: "2022-02-07T20:50:40Z"
    generation: 1018
    name: ar09-01-cyp
    namespace: intel-power
    resourceVersion: "44835"
    uid: 2aa0f908-2f18-473f-989e-12c46ad2811a
  spec:
    activeProfiles:
      balance-performance-ar09-01-cyp: true
      balance-power-ar09-01-cyp: true
      performance-ar09-01-cyp: true
    activeWorkloads:
    - cores:
      - 2
      - 66
      name: performance-ar09-01-cyp-workload
    - cores:
      - 4
      - 68
      name: balance-performance-ar09-01-cyp-workload
    - cores:
      - 3
      - 67
      name: balance-power-ar09-01-cyp-workload
    nodeName: ar09-01-cyp
    powerContainers:
    - exclusiveCpus:
      - 2
      - 66
      id: c152e29f49db457417beca958133e7d8d995ea7302f76073b96c5797fd20d770
```

```
      name: performance-container
      pod: performance-power-pod
      powerProfile: performance-ar09-01-cyp
      workload: performance-ar09-01-cyp-workload
    - exclusiveCpus:
      - 4
      - 68
      id: 870e1d2eb4f971328d5030f97a647b8ee5fb7dae52daebec4714588e9a563667
      name: balance-performance-container
      pod: balance-performance-power-pod
      powerProfile: balance-performance-ar09-01-cyp
      workload: balance-performance-ar09-01-cyp-workload
    - exclusiveCpus:
      - 3
      - 67
      id: 3ea83bf1369946fbe625e7fec4355de4760a1b8a1528959cd7eacb87c3e046a9
      name: balance-power-container
      pod: balance-power-power-pod
      powerProfile: balance-power-ar09-01-cyp
      workload: balance-power-ar09-01-cyp-workload
    sharedPools:
    - name: Default
      sharedPoolCpuIds:
      - 0
      - 1
      - 2
      - 3
      - 4
      - 5
...
```

Part 2:
Building a BMRA Step-by-Step

6 BMRA Setup – Applicable for All Configuration Profiles

This section is relevant for generating BMRA flavors based on their Configuration Profiles. It provides the prerequisites for a system setup and includes information that enables you to review BIOS prerequisites and software BOMs at a glance. The information is presented in multi-column tables to give an easy way to compare and assess the differences between the BMRA flavors that are available.

After setting up the Kubernetes system, refer to the specific section from the following list to build the BMRA flavor:

Section 7, BMRA Basic Configuration Profile Setup
Section 8, BMRA Full Configuration Profile Setup
Section 9, BMRA On-Premises Edge Configuration Profile Setup
Section 10, BMRA Remote Central Office-Forwarding Configuration Profile Setup
Section 11, BMRA Regional Data Center Configuration Profile Setup
Section 12, Storage Configuration Profile Setup

6.1 Set Up an Ansible Host

BMRA Kubernetes clusters require an Ansible Host that stores information about all remote nodes managed. In general, any machine running a recent Linux distribution can be used as Ansible Host for any of the supported BMRA deployments (regardless of target OS on the control and worker nodes), as long as it meets the following basic requirements:

- Network connectivity to the control and worker nodes, including SSH
- Internet connection (using Proxy if necessary)
- Git utility installed
- Python 3 installed
- Ansible version 3.4.0 installed (Ansible-base at 2.10.15)

Step-by-step instructions for building the Ansible Host are provided below for the same list of operating systems that are supported for the control and worker nodes (see Section 2.3.3):

6.1.1 RHEL Version 8 as Ansible Host

1. Install the Linux OS. If using the iso image, choose the Minimal iso version, or select the "Minimal Install" (Basic functionality) option under Software Selection.
2. Make the proper configuration during installation for the following key elements: Network (Ethernet) port IP Address, Host Name, Proxies (if necessary), and Network Time Protocol (NTP).
3. After the installation completes and the machine reboots, log in as root and confirm that it has a valid IP address and can connect (ping) to the control and worker nodes.
4. Make sure that the HTTP and HTTPS proxies are set, if necessary, for internet access. The configuration can be completed with the *export* command or by including the following lines in the `/etc/environment` file:

   ```
   http_proxy=http://proxy.example.com:1080
   https_proxy=http://proxy.example.com:1080
   ```

   Then, load the proxies configuration in the current environment:

   ```
   # source /etc/environment
   ```

5. Install Git:

   ```
   # yum install -y git
   ```

6. Install Python 3:

   ```
   # yum -y install python3
   ```

7. Install Ansible:

   ```
   # pip install ansible-base==2.10.15
   ```

The Ansible Host box is now ready to deploy the Container BMRA. Follow the instructions in Section 2.5.

6.1.2 Ubuntu 20.04 LTS as Ansible Host

1. Install the OS using any method supported by the vendor (Canonical Ltd.). Either the Desktop or Server distribution can be used. Select the “Minimal installation” option under “Updates and Other software”.
2. Follow steps 2, 3, and 4 as described above for RHEL.
3. Update the installation:

   ```
   # sudo apt update
   ```

4. Install SSH utilities:

   ```
   # sudo apt install openssh-server
   ```

5. Install Git:
```
# sudo apt install -y git
```
6. Install Python 3-pip:
```
# sudo apt install -y python3-pip
```
7. Install Ansible:
```
# sudo pip install ansible-base==2.10.15
```

The Ansible Host box is now ready to deploy the Container BMRA. Follow the instructions in Section 2.5.

6.2 Set Up the Control and Worker Nodes - BIOS Prerequisites

This section is applicable for all **BMRA Configuration Profiles.**

Enter the UEFI or BIOS menu and update the configuration as shown in Table 24 and Table 25.

Note: The method for accessing the UEFI or BIOS menu is vendor-specific, for example: https://www.dell.com/support/article/us/en/04/sln167315/how-to-boot-into-the-bios-or-the-lifecycle-controller-on-your-poweredge-server?lang=en

Table 24. BIOS Prerequisites for Control and Worker Nodes for Basic, Full, and Storage Configuration Profiles

PROFILES	BASIC CONFIGURATION PROFILE	FULL CONFIGURATION PROFILE	STORAGE CONFIGURATION PROFILE
Configuration			
BIOS Profile	Energy Balance	Max Performance	Max Performance
Grub Command Line (values are set by Ansible)			
Isolcpus	Optional	Yes	No
Hugepages	Optional	Yes	No
P-state=disable	Optional	Yes, No-SST-BF	No
Limit C-state	Optional	Yes	No

Table 25. BIOS Prerequisites for Control and Worker Nodes for On-Premises Edge, Remote Central Office-Forwarding, and Regional Data Center Configuration Profiles

PROFILES	ON-PREMISES EDGE CONFIGURATION PROFILE	REMOTE CENTRAL OFFICE-FORWARDING CONFIGURATION PROFILE	REGIONAL DATA CENTER CONFIGURATION PROFILE
Configuration			
BIOS Profile	Max Performance	Deterministic	Max Performance
Grub Command Line (values are set by Ansible)			
Isolcpus	Yes	Yes	Optional
Hugepages	Yes	Yes	Optional
P-state=disable	No	Yes, No-SST-BF	Optional
Limit C-state	No	Yes	Optional

The BIOS profile referenced in these tables consists of configurations in the power management, thermal management, and configuration for Intel® platform technologies such as Intel® Virtualization Technology, Intel® Hyper-Threading Technology, Intel SpeedStep® technology, and Intel® Turbo Boost Technology.

The table provides three different BIOS profiles.
1. Energy Balance
2. Max Performance
3. Deterministic

The configuration and values set per each BIOS profile are defined in Table 20.

Note: The above values are the recommended configuration options on the Intel® S2600WFQ and Intel® M50CYP server boards. Some server boards may not provide the same options that are documented in this table. Vendors typically provide options for max performance configuration with virtualization.

6.3 Configuration Dictionary - Group Variables

The following table lists the parameters available as group variables with their type (for example, Boolean, string, URL, list, integer), possible values, and descriptions. The variables in **bold** must be updated to match the target environment. The variables with blue highlight must be updated according to your infrastructure. Refer to the section that describes your Configuration Profile to see the parameters enabled for that Configuration Profile.

Table 26. Configuration Dictionary – Group Variables

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
Common Cluster Configuration				
Kubernetes		Boolean	true/false	Specifies whether to deploy Kubernetes
	kube_version	String	v1.21.1	Kubernetes version
	container_runtime	String	docker, crio, containerd	Container runtime to use as base engine for cluster deployment
	docker_version	String	19.03	Docker version
	containerd_version	String	1.4.6	Containerd version
	crio_version	String	1.21.3	CRI-O version
	update_all_packages	Boolean	false	Runs system-wide package update (apt dist-upgrade, yum update, ...). Tip: Can be set using host_vars for more granular control.
	http_proxy	**URL**	**http://proxy.example.com:1080**	**HTTP proxy address. Comment out if your cluster is not behind proxy.**
	https_proxy	**URL**	**http://proxy.example.com:1080**	**HTTPS proxy address. Comment out if your cluster is not behind proxy.**
	additional_no_proxy	**Comma-separated list of addresses**	**.example.com**	**Additional URLs that are not behind proxy, for example your corporate intra network DNS domain, e.g., ".intel.com". Note: Kubernetes nodes addresses, pod network, etc. are added to no_proxy automatically.**
	kube_network_plugin_multus	Boolean	True	Specifies whether to use the network plugin Multus
	multus_version	String	V3.7	Multus version
	kube_network_plugin	String	calico/flannel	Specifies networking CNI to use
	kube_pods_subnet	CIDR	10.244.0.0/16	Kubernetes pod subnet. Make sure that it matches your CNI plugin requirements (Calico by default) and doesn't overlap with your corporate LAN.
	kube_service_addresses	CIDR	10.233.0.0/18	Kubernetes service subnet. Make sure that it matches your CNI plugin requirements (Calico by default) and doesn't overlap with your corporate LAN.
	kube_proxy_mode	String	Iptables	Instructs kube_proxy how to set up NAT and load balancing functions
	kube_proxy_nodeport_addresses_cidr	CIDR	127.0.0.0/8	Kubernetes service subnet
	cluster_name	DNS domain	cluster.local	Name of the cluster
	registry_local_address	String	"localhost:30500"	Container registry address IP and port
	psp_enabled	Boolean	true/false	Enable pod security policy admission controller and create minimal set of rules
	always_pull_enabled	Boolean	true/false	Set image pull policy to Always. Pulls images before starting containers. Valid credentials must be configured.
Node Feature Discovery				
nfd_enabled		Boolean	true/false	Specifies whether to deploy Node Feature Discovery
	nfd_version	String	0.9	NFD version

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
	nfd_build_image_locally	Boolean	false	Builds NFD image locally instead of using the one from public registry.
	nfd_namespace	String	kube-system	Kubernetes namespace used for NFD deployment
	nfd_sleep_interval	String	60s	Defines how often NFD queries node status and update node labels
Native Built-in Kubernetes CPU Manager				
native_cpu_manager_enabled		Boolean	true/false	Kubernetes CPU manager controls CPU management policies on the nodes. Setting this option as "true" enables the "static" policy; otherwise the default "none" policy is used.
	native_cpu_manager_system_reserved_cpus	Kubernetes millicores	2000m	Number of CPU cores to be reserved for housekeeping (2000m = 2000 millicores = 2 cores)
	native_cpu_manager_kube_reserved_cpus	Kubernetes millicores	1000m	Number of CPU cores to be reserved for Kubelet
	native_cpu_manager_reserved_cpus	Comma-separated list of integers or integer ranges	0,1,2	Explicit list of the CPUs reserved from pods scheduling. ***Note:*** Supported only with kube_version 1.17 and newer, overrides 2 previous options.
Topology Manager (Kubernetes Built-in)[15]				
topology_manager_enabled		Boolean	true/false	Enables Kubernetes built-in Topology Manager
	topology_manager_policy	String, options: none, best-effort, restricted, single-numa-node	best-effort	Topology Manager policy
Intel SR-IOV Network Device Plugin				
sriov_network_operator_enabled		Boolean	true/false	Enables SR-IOV Network Operator
	sriov_network_operator_namespace	String	sriov-network-operator	Kubernetes namespace used to deploy SR-IOV network operator
sriov_net_dp_enabled		Boolean	true/false	Enables SR-IOV network device plugin
	sriov_net_dp_namespace	String	kube-system	Kubernetes namespace used to deploy SR-IOV network device plugin
	sriov_net_dp_build_image_locally	Boolean	true/false	Build and store image locally or use one from public external registry
	sriovdp_config_data	Multi-line string in JSON format	Two resource pools for kernel stack and DPDK-based networking respectively	SR-IOV network device plugin configuration. For more information on supported configurations, refer to https://github.com/intel/sriov-network-device-plugin#configurations
Intel Device Plugins for Kubernetes				
Intel_dp_namespace		String	kube-system	Kubernetes namespace used to deploy Intel device plugin operator
dsa_dp_enabled		Boolean	true/false	Enables Intel DSA device plugin
	configure_dsa_devices	Boolean	true/false	Specifies whether to configure DSA devices
	dsa_devices	List	[ ]	DSA devices to configure worker queues for

[15] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
qat_dp_enabled		Boolean	true/false	Enables Intel QAT device plugin
	qat_dp_namespace	String	kube-system	Namespace used for Intel QAT device plugin
sgx_dp_enabled		Boolean	true/false	Enables Intel SGX device plugin
	sgx_dp_build_image_locally	Boolean	true/false	Build and store image locally or use one from public external registry
	sgx_aesmd_namespace	String	kube-system	Kubernetes namespace used to deploy SGX device plugin
	sgx_dp_provision_limit	Integer	20	
	sgx_dp_enclave_limit	Integer	20	
gpu_dp_enabled		Boolean	true	Enables Intel GPU device plugin
	gpu_dp_namespace	String	kube-system	Namespace used for Intel GPU device plugin
Intel Key Management Reference Application				
kmra_enabled		Boolean	true/false	Enables Intel Key Management Reference Application
	kmra_pccs_api_key	String	"ffffff……"	API Key obtained from Intel's Provisioning Certificate Service
	kmra_deploy_demo_workload	Boolean	true/false	Enable to deploy a KMRA demo workload (NGINX Server)
Service Mesh				
Istio_enabled		Boolean	true/false	Enables Istio service mesh for Kubernetes
Intel Telemetry Aware Scheduling				
tas_enabled		Boolean	true/false	Enables Intel Telemetry Aware Scheduling
	tas_namespace	String	monitoring	Kubernetes namespace used for TAS deployment
	tas_enable_demo_policy	Boolean	false	Creates demo TAS policy
Telemetry Configuration				
collectd_enabled		Boolean	true/false	Gather platform metrics with collectd
	collectd_scrap_interval	Integer	30	Duration to gather metrics using collectd
telegraf_enabled		Boolean	true/false	Gather platform metrics with telegraf
	telegraf_scrap_interval	Integer	30	Duration to gather metrics using Telegraf
Example Network Attachment Definitions (Ready to Use Examples of Custom CNI Plugin Configuration)				
example_net_attach_defs		List of dictionaries	[]	Example network attached definition objects to create
	userspace_ovs_dpdk	Boolean	true/false	Example net-attach-def for Userspace CNI with OVS-DPDK
	userspace_vpp	Boolean	true/false	Example net-attach-def for Userspace CNI with VPP
	sriov_net_dp	Boolean	true/false	Example net-attach-def for SR-IOV Net DP and SR-IOV CNI
MinIO Configuration				
minio_enabled		Boolean	true/false	Enables MinIO Operator/Console
	minio_tenant_enabled	Boolean	true/false	Specifies whether to install sample MinIO Tenant
	minio_tenant_servers	Integer	4	The number of MinIO Tenant nodes
	minio_tenant_volumes_per_server	Integer	4	The number of volumes per server

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
	minio_deploy_test_mode	Boolean	true/false	true (Test Mode) – use a file as a loop device when creating storage called "virtual block device", which is useful for test or automation purpose false (Performance Mode) – use an actual NVME or SSD device when creating storage

6.4 Configuration Dictionary - Host Variables

The following table lists the parameters available as host variables with their type (for example, Boolean, string, URL, list, integer), possible values, and descriptions. The variables in **bold** must be updated to match the target environment. The variables with blue highlight must be updated according to your infrastructure. Refer to the section that describes your Configuration Profile to see the parameters enabled for that Configuration Profile.

Table 27. Configuration Dictionary – Host Variables

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
SR-IOV and Network Devices Configuration				
iommu_enabled		Boolean	true/false	Sets up SR-IOV related kernel parameters and enables further SR-IOV configuration
dataplane_interfaces		**List of dictionaries**	**n/a**	**SR-IOV related NIC configuration using per-port approach**
	dataplane_interfaces[*].name	**String**	**enp24s0f0, enp24s0f1**	**Name of the interface representing PF port**
	dataplane_interfaces[*].bus_info	**String (PCI address)**	**18:00.0, 18:00.1**	**PCI address of the PF port**
	dataplane_interfaces[*].pf_driver	**String**	**ice**	**PF driver, "i40e", "ice"**
	dataplane_interfaces[*].sriov_numvfs	**Integer**	**6, 4**	**Number of VFs to be created, associated with the PF**
	dataplane_interfaces[*].default_vf_driver	**String, options: "i40evf", "iavf", "vfio-pci", "igb_uio"**	**vfio-pci for DPDK, iavf for kernel network stack**	**Default driver module name that the VFs are bound to**
	dataplane_interfaces[*].sriov_vfs[*]	**List of dictionaries**	**n/a**	**List of vfs to create with specific driver (non-default)**
	dataplane_interfaces[*].ddp_profile	**String, optional**	**gtp.pkgo**	**Name of the DDP package to be loaded onto the Network Adapter. Note: Use only for the port 0 of the Network Adapter (PCI address ending with :00.0)**
update_nic_drivers		Boolean	true/false	Set to 'true' to update Linux kernel drivers for Intel Network Adapters
update_nic_firmware		Boolean	true/false	Set 'true' to update Network Adapter firmware
install_ddp_packages		Boolean	true/false	Install Intel X700 and X800 series Network Adapters DDP packages. Required if DDP packages configured in dataplane_interfaces.
install_dpdk		Boolean	true/false	DPDK installation is required for sriov_cni_enabled:true
	dpdk_version	String	21.11	DPDK version to install
	dpdk_local_patches_dir	String	Empty	Path to user-supplied patches to apply against the specified version of DPDK
SR-IOV and Bond CNI Plugins				
sriov_cni_enabled		Boolean	true/false	Installs SR-IOV CNI plugin binary on the node
bond_cni_enabled		Boolean	true/false	Installs Bond CNI plugin binary on the node
Userspace Networking Plugins and Accelerated Virtual Switches				

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
userspace_cni_enabled		Boolean	true/false	Installs userspace CNI plugin binary on the node
ovs_dpdk_enabled		Boolean	true/false	Installs OVS-DPDK on the node
	ovs_dpdk_lcore_mask	Hex integer	0x1	CPU mask for OVS-DPDK PMD threads
	ovs_dpdk_socket_mem	Integer or comma-separated list of integers	256,0	Amount of memory per NUMA node allocated to OVS-DPDK PMD threads
vpp_enabled		Boolean	true/false	Installs FD.io VPP
Hugepages/Memory Configuration				
hugepages_enabled		Boolean	true/false	Enables hugepages support
	default_hugepage_size	String, options: 2M, 1G	1G	Default hugepages size
	number_of_hugepages	Integer	4	Sets how many hugepages should be created
CPU Configuration				
isolcpus_enabled		Boolean	true/false	Enables CPU cores isolation from Linux scheduler
	isolcpus	Comma-separated list of CPU cores/ranges	4-11	CPU cores isolated from Linux scheduler
intel_pstate		String	hwp_only	Enables Intel P-state scaling driver. Available parameters: disable, passive, force, no_hwp, hwp_only, support_aci_pcc, per_cpu_perf_limites
	turbo_boost_enabled	Boolean	true/false	Enables Turbo Boost for P-state attribute
sst_pp_configuration_enabled		Boolean	true/false	Enables Intel SST Performance Profiles for flexible configuration of SST-BF, SST-CP, and SST-TF
	sst_pp_config_list	List of dictionaries	sst_bf: enable/disable sst_cp: enable/disable sst_tf: enable/disable	Enables configuration of SST features through SST-PP
	online_cpus_range	String	auto	Specifies automatic configuration of online CPUs versus manual configuration of each SST feature
sst_bf_configuration_enabled		Boolean	true/false	Enables Intel SST Base Frequency technology. Support of SST-BF requires 'intel_pstate' to be 'enabled'
	clx_sst_bf_mode	Character, options: s, m, r	s	Configure SST-BF mode for 2nd Generation Intel® Xeon® [s] Set SST-BF config (set min/max to 2700/2700 and 2100/2100) [m] Set P1 on all cores (set min/max to 2300/2300) [r] Revert cores to min/Turbo (set min/max to 800/3900)
	icx_sst_bf_enabled	Boolean	true/false	Enables Intel SST Base Frequency technology. 3rd Generation Intel® Xeon® support of SST-BF requires 'intel_pstate' to be 'enabled'.
	icx_sst_bf_with_core_priority	Boolean	true/false	Prioritize (SST-CP) power flow to high frequency cores
sst_cp_configuration_enabled		Boolean	true/false	Enables Intel SST Core Power technology on 3rd Generation Intel® Xeon®. SST-CP overrides any 'SST-BF configuration'.
	sst_cp_priority_type	Integer	1	0 – proportional 1 - ordered

COMPONENT	COMPONENT PARAMETER	TYPE	VALUE	DESCRIPTION/COMMENT
	sst_cp_clos_groups	List of dictionaries	[ ]	Allows for configuration of up to 4 CLOS groups including id, frequency_weight, min_MHz, max_MHz
	sst_cp_cpu_clos	List of dictionaries	[ ]	Allows for definition of CPU cores per close group
sst_tf_configuration_enabled		Boolean	true/false	Enables Intel SST Turbo Frequency
Miscellaneous				
dns_disable_stub_listener	dns_disable_stub_listener	Boolean	true/false	(Ubuntu only) Disables DNS stub listener from the systemd-resolved service, which is known to cause problems with DNS and Docker containers on Ubuntu
install_real_time_package	install_real_time_package	Boolean	true/false	Installs real-time Linux kernel packages.
QAT Configuration				
update_qat_drivers		Boolean	true/false	Install QAT drivers and services
qat_devices		List of dictionaries	[ ]	SR-IOV related QAT configuration using per-port approach
	qat_devices[*].qat_id	String (PCI address)	0000:ab:00.0, 0000:xy:00.0, 0000:yz:00.0	PCI address of the PF port
	qat_devices[*].qat_sriov_numvfs	Integer	10	Number of VFs to be created per QAT device physical function
openssl_install		Boolean	true/false	Install OpenSSL for use with QAT engine
MinIO Configuration				
minio_pv		List of dictionaries	[]	PV related MinIO configuration
	minio_pv[*].name	String	“mnt-data-1”	PV identifier followed by node name for creating PVs
	minio_pv[*].storageClassName	String	“local-storage”	Storage class name to match with PVC
	minio_pv[*].accessMode	String	“ReadWriteOnce”	Access mode when mounting a volume: ReadWriteOnce, ReadOnlyMany, ReadWriteMany, ReadWriteOncePod
	minio_pv{*}.persistentVolumeReclaimPolicy	String	“Retain”	Reclaim policy when a volume is released once it’s bound: Retain, Recycle, Delete
	minio_pv(*).mountPath	String	/mnt/data0	Mount path of a volume
	minio_pv(*).device	String	/dev/nvme0n1	Target storage device name when creating a volume. When group_var: minio_deploy_test_mode == true, use files (/tmp/diskimage[*]) as a loop device (/dev/loop[*]) for storage. Otherwise, use an actual NVME or SSD device for storage on the device name for storage.
	minio_pv(*).capacity	String	1GiB	Volume capacity when creating a partition on the target device. Supports units: GiB, TiB

7 BMRA Basic Configuration Profile Setup

This section contains a step-by-step description of how to set up your BMRA Basic Configuration Profile Flavor.

To use the BMRA Basic Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to Section 7.1 for details.
   You also need to build your Kubernetes cluster.
2. Download the Ansible playbook for your Configuration Profile. Refer to Section 7.2 for details.

3. Configure the optional Ansible parameters using the information in the Configuration Profile tables. Refer to Section 7.3 for details.
4. Deploy the platform. Refer to Section 7.4 for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

Be aware of the definitions of terminology used in tables in this section.

TERM	DESCRIPTION
Hardware Taxonomy	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, or similar value)
DISABLED	Setting must be disabled in the BIOS (configured as Disabled, No, False, or any other value with this meaning.)
OPTIONAL	Setting can be either disabled or enabled, depending on user's workload. Setting does not affect the Configuration Profile or platform deployment.
Software Taxonomy	
TRUE	Feature is included and enabled by default.
FALSE	Feature is included but disabled by default - can be enabled and configured by user.
N/A	Feature is not included and cannot be enabled or configured.

7.1 Step 1 - Set Up Basic Configuration Profile Hardware

The tables in this section list the hardware BOM for the Basic Configuration Profile, including Control Node, Worker Node Base, and Worker Node Plus. We recommend that you set up at least one control node and one worker node.

Table 28. Hardware Setup for Basic Configuration Profile – 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	Controller_2ndGen_1	Controller_3rdGen_1
Worker Node Options	Worker_2ndGen_Base_1	Worker_3rdGen_Base_1

Table 29. Hardware Setup for Basic Configuration Profile –Intel Xeon D Processor

NODE OPTIONS	INTEL XEON D PROCESSOR
Control Node Options	Controller_Xeon_D_1
Worker Node Options	Worker_Xeon_D_Base_1

7.2 Step 2 - Download Basic Configuration Profile Ansible Playbook

This section contains details for downloading the Basic Configuration Profile Ansible playbook. It also provides an overview of the Ansible playbook and lists the software that is automatically installed when the playbook is deployed.

Download the Basic Configuration Profile Ansible playbook using the steps described in Section 2.5.

7.2.1 Basic Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Basic Configuration Profile allows you to provision a production-ready Kubernetes cluster. Every capability included in the Basic Configuration Profile playbook can be disabled or enabled. Refer to the diagram and group and host variables tables below to see which Ansible roles are included and executed by default.

The diagram shows the architecture of the Ansible playbooks and roles that are included in the Basic Configuration Profile.

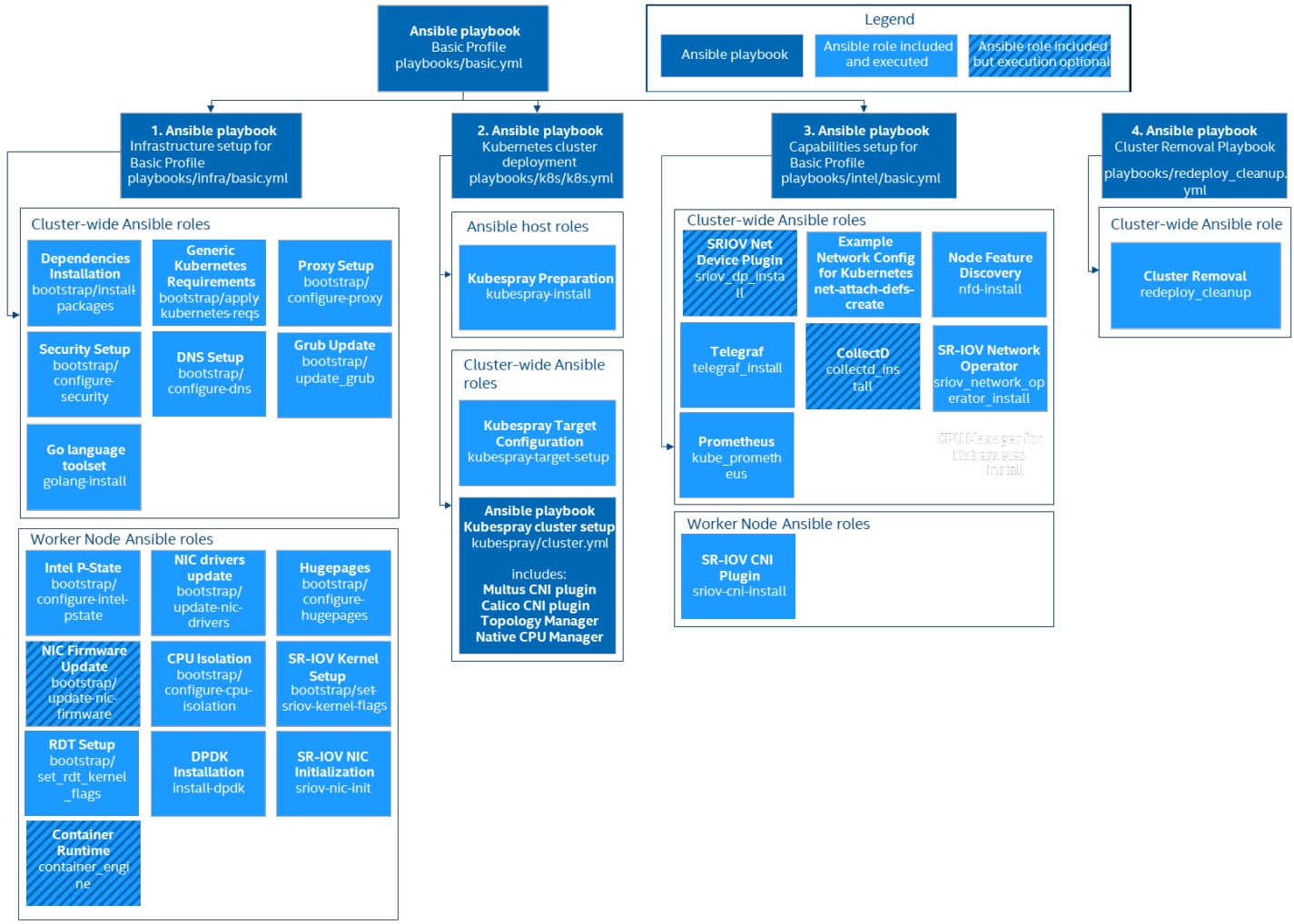

Figure 4. Basic Configuration Profile Ansible Playbook

7.3 Step 3 - Set Up Basic Configuration Profile

Review the optional Ansible group and host variables in this section and select options that match your desired configuration.

1. Update the `inventory.ini` file with your environment details as described in Section 2.5.3.
2. Create `host_vars` files for all worker nodes as specified in Section 2.3.4.
3. Update group and host variables to match your desired configuration as specified in Section 2.3.4. Refer to the tables in Section 7.3.1 and Section 7.3.2.

Variables are grouped into two main categories:

1. Group variables – apply to both control and worker nodes and have cluster-wide impact.
2. Host variables – scope is limited to a single worker node.

The tables below are a summary of group and host variables. For lists showing all configurable properties, see Section 6.3 and Section 6.4. All of the variables are important but pay special attention to variables in **bold** as they almost always need to be updated to match the target environment.

7.3.1 Basic Configuration Profile Group Variables

Table 30. Basic Configuration Profile – Group Variables

COMPONENT	VALUE	
Kubernetes	true	For the list of all configurable properties, see Section 6.3
nfd_enabled	true	
topology_manager_enabled	true	
sriov_network_operator_enabled	false	

COMPONENT	VALUE
sriov_net_dp_enabled	false
example_net_attach_defs	false
collectd_enabled	false
telegraf_enabled	true

7.3.2 Basic Configuration Profile Host Variables[16]

Table 31. Basic Configuration Profile – Host Variables

COMPONENT	VALUE	
iommu_enabled	false	For the list of all configurable properties, see Section 6.4
sriov_cni_enabled	false	
install_dpdk	false	
isolcpus_enabled	false	
dataplane_interfaces	[]	

7.4 Step 4 – Deploy and Validate Basic Configuration Profile Platform

Deploy the Basic Configuration Profile Ansible playbook using the steps described in Section 2.3.5.

Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

8 BMRA Full Configuration Profile Setup

This section contains a step-by-step description of how to set up your BMRA Full Configuration Profile Flavor.

To use the BMRA Full Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to Section 8.1 for details.
   You also need to build your Kubernetes cluster.
2. Download the Ansible playbook for your Configuration Profile. Refer to Section 8.2 for details.
3. Configure the optional Ansible parameters using the information in the Configuration Profile tables. Refer to Section 8.3 for details.
4. Deploy the platform. Refer to Section 8.4 for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

Be aware of the definitions of terminology used in tables in this section.

TERM	DESCRIPTION
Hardware Taxonomy	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, or similar value.)
DISABLED	Setting must be disabled in the BIOS (configured as Disabled, No, False, or any other value with this meaning.)
OPTIONAL	Setting can be either disabled or enabled, depending on user's workload. Setting does not affect the Configuration Profile or platform deployment.
Software Taxonomy	
TRUE	Feature is included and enabled by default.
FALSE	Feature is included but disabled by default - can be enabled and configured by user.
N/A	Feature is not included and cannot be enabled or configured.

[16] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

8.1 Step 1 - Set Up Full Configuration Profile Hardware

The tables in this section list the hardware BOM for the Full Configuration Profile, including Control Node, Worker Node Base, and Worker Node Plus. We recommend that you set up at least three control nodes and two worker nodes.

Table 32. Hardware Setup for Full Configuration Profile – 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	Controller_2ndGen_3	Controller_3rdGen_3
Worker Node Options	Worker_2ndGen_Plus_1	Worker_3rdGen_Plus_1

Table 33. Hardware Setup for Full Configuration Profile –Intel Xeon D Processor

NODE OPTIONS	INTEL XEON D PROCESSOR
Control Node Options	Controller_Xeon_D_3
Worker Node Options	Worker_Xeon_D_Plus_1

8.2 Step 2 - Download Full Configuration Profile Ansible Playbook

This section contains details for downloading the Full Configuration Profile Ansible playbook. It also provides an overview of the Ansible playbook and lists the software that is automatically installed when the playbook is deployed.

Download the Full Configuration Profile Ansible playbook using the steps described in Section 2.5.

8.2.1 Full Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Full Configuration Profile allows you to provision a production-ready Kubernetes. It also applies any additional requirements, such as host OS configuration or Network Adapter drivers and firmware updates. Full Configuration Profile playbook includes all features available through BMRA Ansible Playbook and provides one of the highest degrees of configurability. Every capability included in the Full Configuration Profile playbook can be disabled or enabled. Refer to the diagram and group and host variables tables below to see which Ansible roles are included and executed by default.

The diagram shows the architecture of the Ansible playbooks and roles that are included in the Full Configuration Profile.

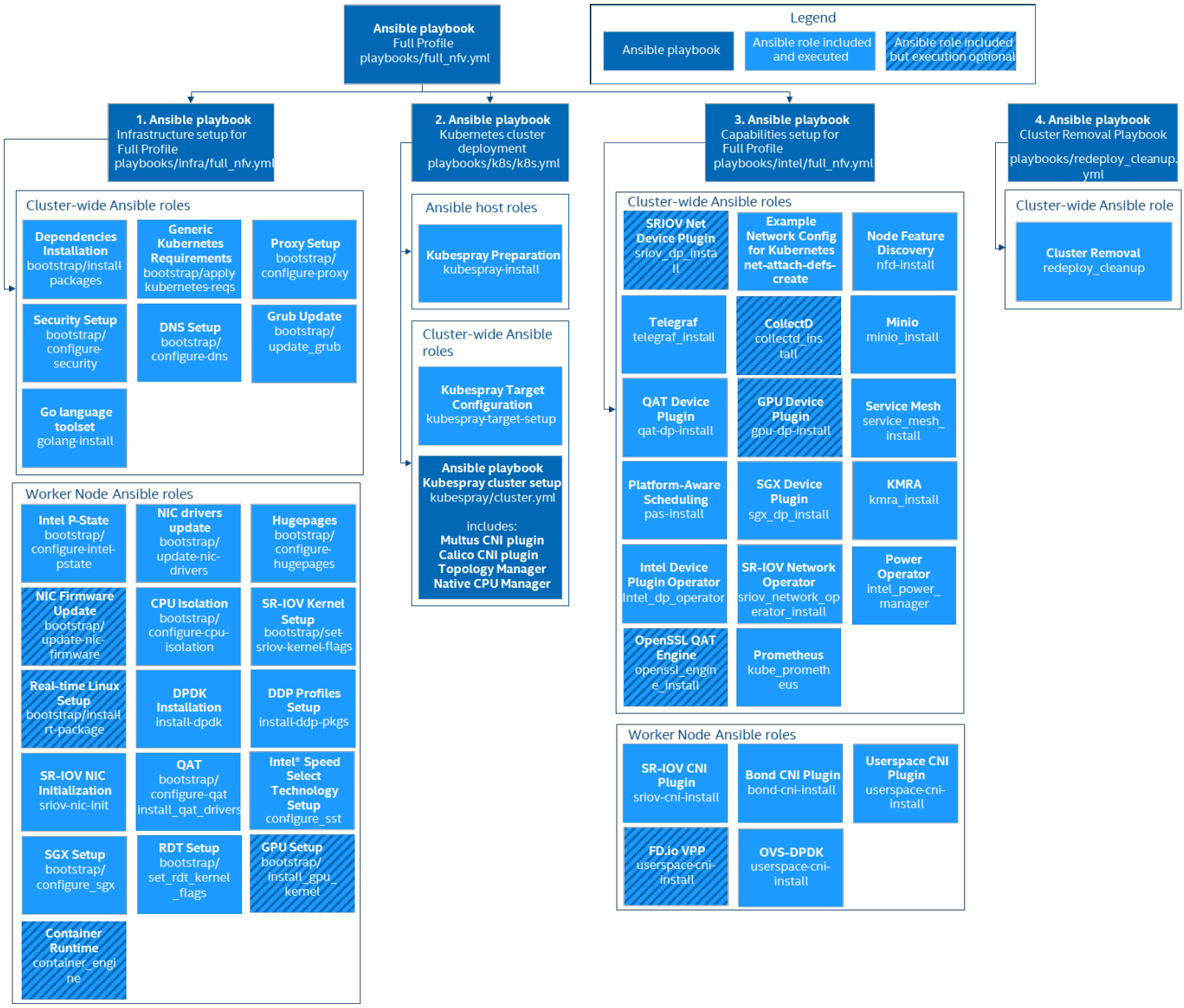

Figure 5. Full Configuration Profile Ansible Playbook

8.3 Step 3 - Set Up Full Configuration Profile

Review the optional Ansible group and host variables in this section and select options that match your desired configuration.

1. Update the `inventory.ini` file with your environment details as described in Section 2.3.3.
2. Create `host_vars` files for all worker nodes as specified in Section 2.3.4
3. Update group and host variables to match your desired configuration as specified in Section 2.3.4. Refer to the tables in Section 8.3.1 and Section 8.3.2.

Variables are grouped into two main categories:

1. Group variables – apply to both control and worker nodes and have cluster-wide impact.
2. Host variables – scope is limited to a single worker node.

The tables below are a summary of group and host variables. For lists showing all configurable properties, see Section 6.3 and Section 6.4. All of the variables are important but pay special attention to variables in **bold** as they almost always need to be updated to match the target environment.

8.3.1 Full Configuration Profile Group Variables

Table 34. Full Configuration Profile – Group Variables

COMPONENT	VALUE	
Kubernetes	true	For the list of all configurable properties, see Section 6.3
nfd_enabled	true	
native_cpu_manager_enabled	true	
topology_manager_enabled	true	
sriov_network_operator_enabled	true	
sriov_net_dp_enabled	false	
sgx_dp_enabled	true	
gpu_dp_enabled	false	
qat_dp_enabled	true	
openssl_engine_enabled	true	
kmra_enabled	true	
tas_enabled	true	
gas_enabled	false	
example_net_attach_defs	false	
collectd_enabled	false	
telegraf_enabled	true	
service_mesh	true	
power_manager	false	
minio_enabled	false	

8.3.2 Full Configuration Profile Host Variables[17]

Table 35. Full Configuration Profile – Host Variables

COMPONENT	VALUE	
iommu_enabled	true	For the list of all configurable properties, see Section 6.4
sriov_cni_enabled	false	
bond_cni_enabled	true	
ddp_enabled	true	
sst_pp_configuration_enabled	false	
userspace_cni_enabled	false	
hugepages_enabled	true	
isolcpus_enabled	false	
install_dpdk	true	
install_ddp_packages	true	
qat_devices	[]	
dataplane_interfaces	[]	
minio_pv	[]	

8.4 Step 4 - Deploy and Validate Full Configuration Profile Platform

Deploy the Full Configuration Profile Ansible playbook using the steps described in Section 2.5.5.

Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

[17] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

9 BMRA On-Premises Edge Configuration Profile Setup

This section contains a step-by-step description of how to set up your BMRA On-Premises Edge Configuration Profile Flavor.

To use the BMRA On-Premises Edge Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to Section 9.1 for details.
   You also need to build your Kubernetes cluster.
2. Download the Ansible playbook for your Configuration Profile. Refer to Section 9.2 for details.
3. Configure the optional Ansible parameters using the information in the Configuration Profile tables. Refer to Section 9.3 for details.
4. Deploy the platform. Refer to Section 9.4 for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

Be aware of the definitions of terminology used in tables in this section.

TERM	DESCRIPTION
Hardware Taxonomy	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, or similar value.)
DISABLED	Setting must be disabled in the BIOS (configured as Disabled, No, False, or any other value with this meaning.)
OPTIONAL	Setting can be either disabled or enabled, depending on user's workload. Setting does not affect the Configuration Profile or platform deployment.
Software Taxonomy	
TRUE	Feature is included and enabled by default.
FALSE	Feature is included but disabled by default - can be enabled and configured by user.
N/A	Feature is not included and cannot be enabled or configured.

9.1 Step 1 - Set Up On-Premises Edge Configuration Profile Hardware

The tables in this section list the hardware BOM for the On-Premises Edge Configuration Profile, including Control Node, Worker Node Base, and Worker Node Plus. We recommend that you set up at least one control node and one worker node.

Table 36. Hardware Setup for On-Premises Edge Configuration Profile – 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	Controller_2ndGen_1	Controller_3rdGen_1
Worker Node Options	Worker_2ndGen_Base_2 or Worker_2ndGen_Plus_1	Worker_3rdGen_Base_2 or Worker_3rdGen_Plus_1

Table 37. Hardware Setup for On-Premises Edge Configuration Profile – Intel Xeon D Processor

NODE OPTIONS	INTEL XEON D PROCESSOR
Control Node Options	Controller_Xeon_D_1
Worker Node Options	Worker_Xeon_D_Base_2 or Worker_Xeon_D_Plus_1

9.2 Step 2 - Download On-Premises Edge Configuration Profile Ansible Playbook

This section contains details for downloading the On-Premises Edge Configuration Profile Ansible playbook. It also provides an overview of the Ansible playbook and lists the software that is automatically installed when the playbook is deployed.

Download the On-Premises Edge Configuration Profile Ansible playbook using the steps described in Section 2.5.

9.2.1 On-Premises Edge Configuration Profile Ansible Playbook Overview

The Ansible playbook for the On-Premises Edge Configuration Profile allows you to provision a production-ready Kubernetes cluster. It also applies any additional requirements, such as host OS configuration or Network Adapter drivers and firmware updates. Every capability included in the On-Premises Edge Configuration Profile playbook can be disabled or enabled. Refer to the diagram and group and host variables tables below to see which Ansible roles are included and executed by default.

The diagram shows the architecture of the Ansible playbooks and roles that are included in the On-Premises Edge Configuration Profile.

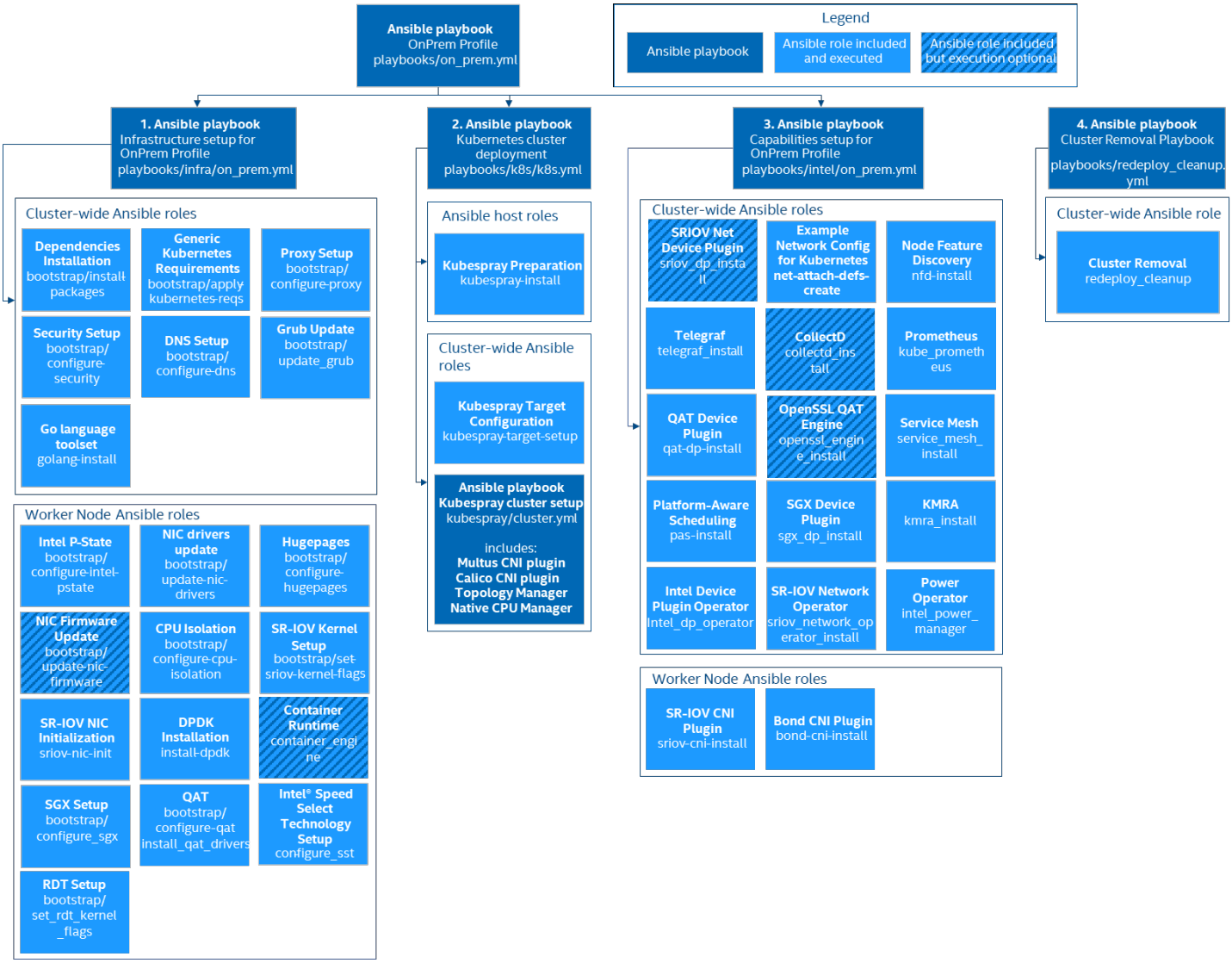

Figure 6. On-Premises Edge Configuration Profile Ansible Playbook

9.3 Step 3 - Set Up On-Premises Edge Configuration Profile

Review the optional Ansible group and host variables in this section and select options that match your desired configuration.

1. Update the `inventory.ini` file with your environment details as described in Section 3.3.3.
2. Create `host_vars` files for all worker nodes specified in the inventory. For example, if you have worker1, worker2 and worker3 in the kube-node group, execute:

```
mv host_vars/node1.yml host_vars/worker1.yml
cp host_vars/worker1.yml host_vars/worker2.yml
cp host_vars/worker1.yml host_vars/worker3.yml
```

3. Update group and host variables to match your desired configuration. Refer to the tables in Section 9.3.1 and Section 9.3.2.

   Note: Pay special attention to the variables in **bold** as these almost always need to be updated individually to match your environment details. Make sure that <worker_node_name>.yml files have been created for all worker nodes specified in your inventory file.

```
vim group_vars/all.yml
```

```
vim host_vars/<worker_node_name>.yml
```

The complete set of configuration variables for the On-Premises Edge Configuration Profile along with their default values can be found in the examples/on_prem directory.

Variables are grouped into two main categories:
1. Group variables – apply to both control and worker nodes and have cluster-wide impact.
2. Host variables – scope is limited to a single worker node.

The tables below are a summary of group and host variables. For lists showing all configurable properties, see Section 6.3 and Section 6.4. All of the variables are important but pay special attention to variables in **bold** as they almost always need to be updated to match the target environment.

9.3.1 On-Premises Edge Configuration Profile Group Variables

Table 38. On-Premises Edge Configuration Profile – Group Variables

COMPONENT	VALUE	
Kubernetes	true	For the list of all configurable properties, see Section 6.3
nfd_enabled	true	
native_cpu_manager_enabled	true	
topology_manager_enabled	true	
sriov_network_operator_enabled	true	
sriov_net_dp_enabled	false	
sgx_dp_enabled	true	
qat_dp_enabled	true	
openssl_engine_enabled	true	
kmra_enabled	true	
tas_enabled	true	
example_net_attach_defs	false	
collectd_enabled	false	
telegraf_enabled	true	
service_mesh	true	
power_manager	false	

9.3.2 On-Premises Edge Configuration Profile Host Variables[18]

Table 39. On-Premises Edge Configuration Profile – Host Variables

COMPONENT	VALUE	
iommu_enabled	true	For the list of all configurable properties, see Section 6.4
sriov_cni_enabled	false	
bond_cni_enabled	false	
hugepages_enabled	true	
isolcpus_enabled	false	
sst_pp_configuration_enabled	false	
install_dpdk	true	
qat_devices	[]	
dataplane_interfaces	[]	

[18] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

9.4 Step 4 - Deploy and Validate On-Premises Edge Configuration Profile Platform

Deploy the On-Premises Edge Configuration Profile Ansible playbook using the steps described in Section 2.5.5.

Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

10 BMRA Remote Central Office-Forwarding Configuration Profile Setup

This section contains a step-by-step description of how to set up your BMRA Remote Central Office-Forwarding Configuration Profile Flavor.

To use the BMRA Remote Central Office-Forwarding Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to Section 10.1 for details.
   You also need to build your Kubernetes cluster.
2. Download the Ansible playbook for your Configuration Profile. Refer to Section 10.2 for details.
3. Configure the optional Ansible parameters using the information in the Configuration Profile tables. Refer to Section 10.3 for details.
4. Deploy the platform. Refer to Section 10.4 for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

Be aware of the definitions of terminology used in tables in this section.

TERM	DESCRIPTION
Hardware Taxonomy	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, or similar value.)
DISABLED	Setting must be disabled in the BIOS (configured as Disabled, No, False, or any other value with this meaning.)
OPTIONAL	Setting can be either disabled or enabled, depending on user's workload. Setting does not affect the Configuration Profile or platform deployment.
Software Taxonomy	
TRUE	Feature is included and enabled by default.
FALSE	Feature is included but disabled by default - can be enabled and configured by user.
N/A	Feature is not included and cannot be enabled or configured.

10.1 Step 1 - Set Up Remote Central Office-Forwarding Configuration Profile Hardware

The tables in this section list the hardware BOM for the Remote Central Office-Forwarding Configuration Profile, including Control Node, Worker Node Base, and Worker Node Plus. We recommend that you set up at least one control node and one worker node.

Table 40. Hardware Setup for Remote Central Office-Forwarding Configuration Profile – 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	Controller_2ndGen_2	Controller_3rdGen_2
Worker Node Options	Worker_2ndGen_Base_3 or Worker_2ndGen_Plus_2	Worker_3rdGen_Base_3 or Worker_3rdGen_Plus_2

Table 41. Hardware Setup for Remote Central Office-Forwarding Configuration Profile – Intel Xeon D Processor

NODE OPTIONS	INTEL XEON D PROCESSOR
Control Node Options	Controller_Xeon_D_2
Worker Node Options	Worker_Xeon_D_Base_3 or Worker_Xeon_D_Plus_2

10.2 Step 2 - Download Remote Central Office-Forwarding Configuration Profile Ansible Playbook

This section contains details for downloading the Remote Central Office-Forwarding Configuration Profile Ansible playbook. It also provides an overview of the Ansible playbook and lists the software that is automatically installed when the playbook is deployed.

Download the Remote Central Office-Forwarding Configuration Profile Ansible playbook using the steps described in Section 2.5.

10.2.1 Remote Central Office-Forwarding Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Remote Central Office-Forwarding Configuration Profile allows you to provision a production-ready Kubernetes cluster. It also applies any additional requirements, such as host OS configuration or Network Adapter drivers and firmware updates. Every capability included in the Remote Central Office-Forwarding Configuration Profile playbook can be disabled or enabled. Refer to the diagram and group and host variables tables below to see which Ansible roles are included and executed by default.

The diagram shows the architecture of the Ansible playbooks and roles that are included in the Remote Central Office-Forwarding Configuration Profile.

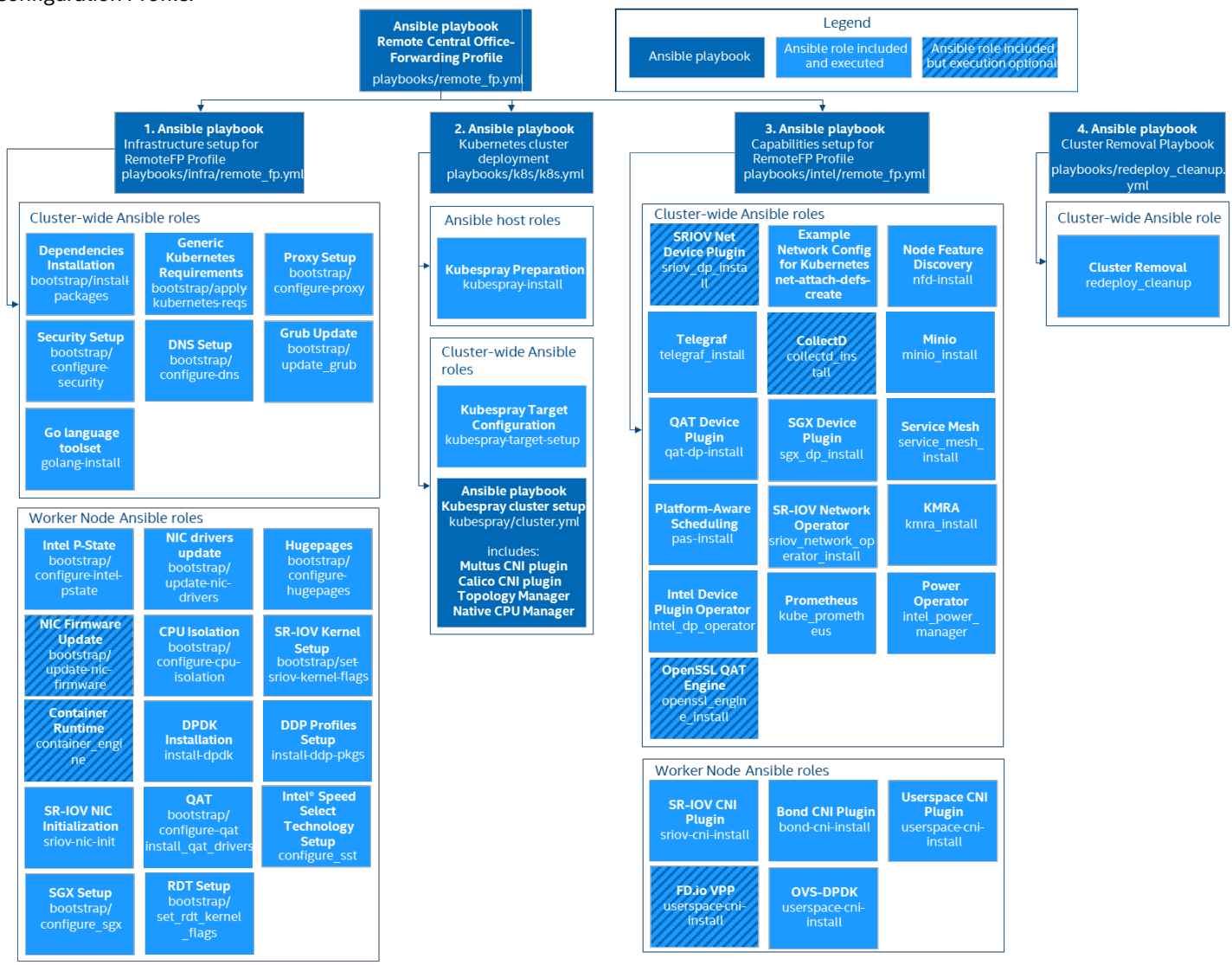

Figure 7. Remote Central Office-Forwarding Configuration Profile Ansible Playbook

10.3 Step 3 - Set Up Remote Central Office-Forwarding Configuration Profile

Review the optional Ansible group and host variables in this section and select options that match your desired configuration.

1. Update the `inventory.ini` file with your environment details as described in Section 2.3.3.
2. Create `host_vars` files for all worker nodes as specified in Section 2.3.4.
3. Update group and host variables to match your desired configuration as specified in Section 2.3.4. Refer to the tables in Section 10.3.1 and Section 10.3.2.

Variables are grouped into two main categories:
1. Group variables – apply to both control and worker nodes and have cluster-wide impact.
2. Host variables – scope is limited to a single worker node.

The tables below are a summary of group and host variables. For lists showing all configurable properties, see Section 6.3 and Section 6.4. All of the variables are important but pay special attention to variables in **bold** as they almost always need to be updated to match the target environment.

10.3.1 Remote Central Office-Forwarding Configuration Profile Group Variables

Table 42. Remote Central Office-Forwarding Configuration Profile – Group Variables

COMPONENT	VALUE	
Kubernetes	true	For the list of all configurable properties, see Section 6.3
nfd_enabled	true	
native_cpu_manager_enabled	true	
topology_manager_enabled	true	
sriov_network_operator_enabled	true	
sriov_net_dp_enabled	false	
sgx_dp_enabled	true	
qat_dp_enabled	false	
openssl_engine_enabled	true	
kmra_enabled	true	
tas_enabled	true	
example_net_attach_defs	false	
collectd_enabled	false	
telegraf_enabled	true	
service_mesh	true	
power_manager	false	

10.3.2 Remote Central Office-Forwarding Configuration Profile Host Variables[19]

Table 43. Remote Central Office-Forwarding Configuration Profile – Host Variables

COMPONENT	VALUE	
iommu_enabled	true	For the list of all configurable properties, see Section 6.4
sriov_cni_enabled	false	
bond_cni_enabled	false	
ddp_enabled	true	
userspace_cni_enabled	false	
hugepages_enabled	true	
isolcpus_enabled	false	
sst_pp_configuration_enabled	false	
install_dpdk	true	
install_ddp_packages	true	
qat_devices	[]	
dataplane_interfaces	[]	

[19] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

10.4 Step 4 - Deploy and Validate Remote Central Office-Forwarding Configuration Profile Platform

Deploy the Remote Central Office-Forwarding Configuration Profile Ansible playbook using the steps described in Section 2.5.5.

Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

11 BMRA Regional Data Center Configuration Profile Setup

This section contains a step-by-step description of how to set up your BMRA Regional Data Center Configuration Profile Flavor.

To use the Regional Data Center Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to Section 11.1 for details.
   You also need to build your Kubernetes cluster.
2. Download the Ansible playbook for your Configuration Profile. Refer to Section 11.2 for details.
3. Configure the optional Ansible parameters using the information in the Configuration Profile tables. Refer to Section 11.3 for details.
4. Deploy the platform. Refer to Section 11.4 for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

Be aware of the definitions of terminology used in tables in this section.

TERM	DESCRIPTION
Hardware Taxonomy	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, or similar value.)
DISABLED	Setting must be disabled in the BIOS (configured as Disabled, No, False, or any other value with this meaning.)
OPTIONAL	Setting can be either disabled or enabled, depending on user's workload. Setting does not affect the Configuration Profile or platform deployment.
Software Taxonomy	
TRUE	Feature is included and enabled by default.
FALSE	Feature is included but disabled by default - can be enabled and configured by user.
N/A	Feature is not included and cannot be enabled or configured.

11.1 Step 1 - Set Up Regional Data Center Configuration Profile Hardware

The tables in this section list the hardware BOM for the Regional Data Center Configuration Profile, including Control Node, Worker Node Base, and Worker Node Plus. We recommend that you set up at least one control node and one worker node.

Table 44. Hardware Setup for Regional Data Center Configuration Profile – 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	N/A*	Controller_3rdGen_3
Worker Node Options	N/A*	Worker_3rdGen_Plus_3
*Configuration Profile only tested with 3rd Generation Intel Xeon Scalable processor		

Table 45. Hardware Setup for Regional Data Center Configuration Profile –and Intel Xeon D Processor

NODE OPTIONS	INTEL XEON D PROCESSOR
Control Node Options	N/A*
Worker Node Options	N/A*

11.2 Step 2 - Download Regional Data Center Configuration Profile Ansible Playbook

This section contains details for downloading the Regional Data Center Configuration Profile Ansible playbook. It also provides an overview of the Ansible playbook and lists the software that is automatically installed when the playbook is deployed.

Download the Regional Data Center Configuration Profile Ansible playbook using the steps described in Section 2.5.

11.2.1 Regional Data Center Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Regional Data Center Configuration Profile allows you to provision a production-ready Kubernetes cluster. It also applies any additional requirements, such as host OS configuration or Network Adapter drivers and firmware updates. Every capability included in the Regional Data Center Configuration Profile playbook can be disabled or enabled. Refer to the diagram and group and host vars tables below to see which Ansible roles are included and executed by default.

The diagram shows the architecture of the Ansible playbooks and roles that are included in the Regional Data Center Configuration Profile.

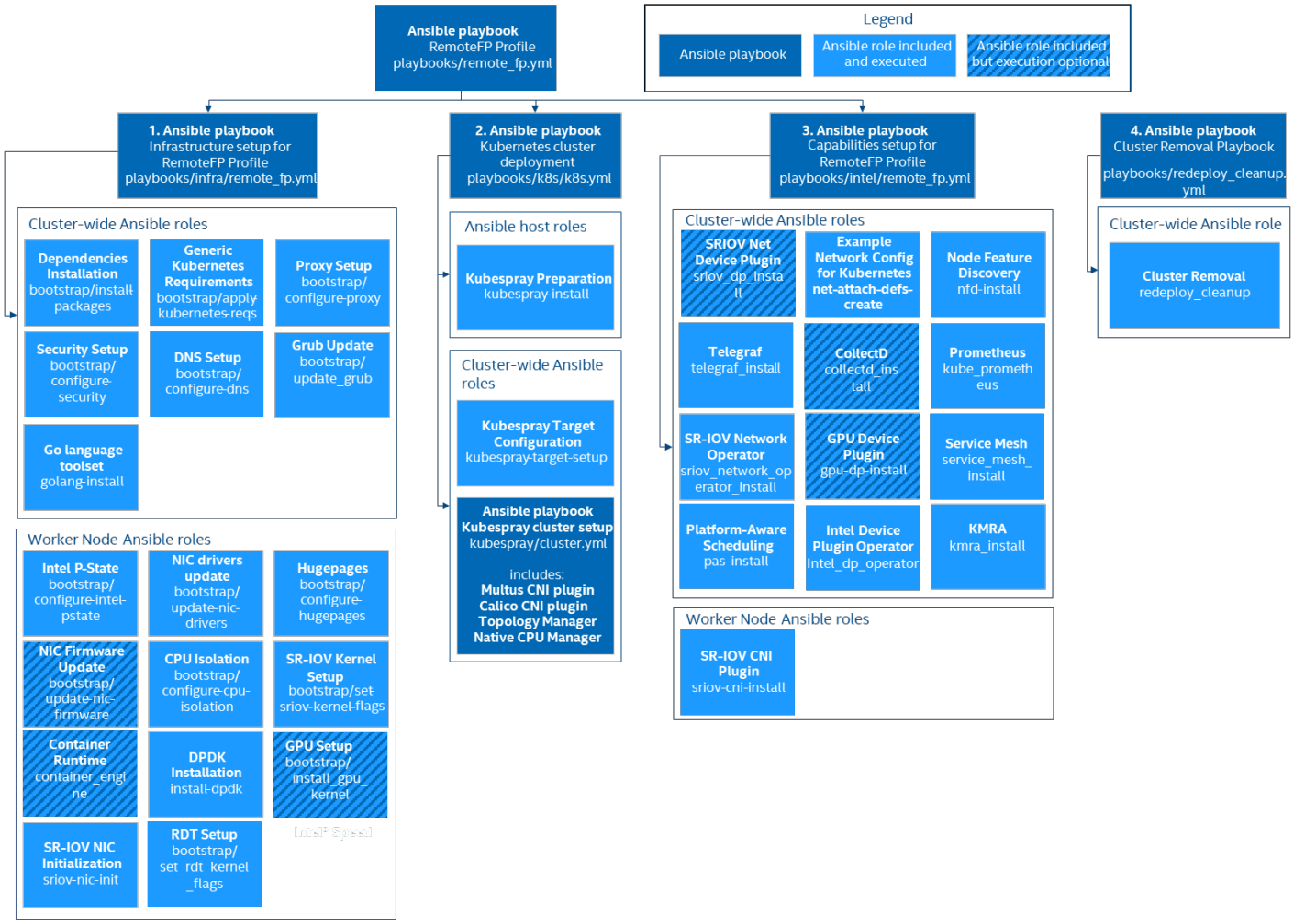

Figure 8. Regional Data Center Configuration Profile Ansible Playbook

11.3 Step 3 - Set Up Regional Data Center Configuration Profile

Review the optional Ansible group and host variables in this section and select options that match your desired configuration.

1. Update the `inventory.ini` file with your environment details as described in Section 2.3.3.
2. Create `host_vars` files for all worker nodes as specified in Section 2.3.4.
3. Update group and host variables to match your desired configuration as specified in Section 2.3.4. Refer to the tables in Section 11.3.1 and Section 11.3.2.

Variables are grouped into two main categories:

1. Group variables – apply to both control and worker nodes and have cluster-wide impact.
2. Host variables – scope is limited to a single worker node.

The tables below are a summary of group and host variables. For lists showing all configurable properties, see Section 6.3 and Section 6.4. All of the variables are important but pay special attention to variables in **bold** as they almost always need to be updated to match the target environment.

11.3.1 Regional Data Center Configuration Profile Group Variables

Table 46. Regional Data Center Configuration Profile – Group Variables

COMPONENT	VALUE	
Kubernetes	true	For the list of all configurable properties, see Section 6.3
nfd_enabled	true	
native_cpu_manager_enabled	true	
topology_manager_enabled	true	
sriov_network_operator_enabled	false	
sriov_net_dp_enabled	false	
gpu_dp_enabled	true	
tas_enabled	true	
gas_enabled	true	
example_net_attach_defs	false	
collectd_enabled	false	
telegraf_enabled	true	
service_mesh	true	

11.3.2 Regional Data Center Configuration Profile Host Variables[20]

Table 47. Regional Data Center Configuration Profile – Host Variables

COMPONENT	VALUE	
iommu_enabled	false	For the list of all configurable properties, see Section 6.4
sriov_cni_enabled	false	
hugepages_enabled	false	
isolcpus_enabled	false	
install_dpdk	false	
dataplane_interfaces	[]	

11.4 Step 4 – Deploy and Validate Regional Data Center Configuration Profile Platform

Deploy the Regional Data Center Configuration Profile Ansible playbook using the steps described in Section 2.5.5.

Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

12 Storage Configuration Profile Setup

The BMRA for Storage focuses exclusively on object storage with the use of MinIO, where the singular workload for the storage solution is the MinIO application, as shown in Figure 9. MinIO has no requirements for telco-specific infrastructure such as DPDK. Hence MinIO's only architectural requirement is the basic OS support for it to service clients and utilize connected storage devices such as HDDs and SSDs.

[20] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

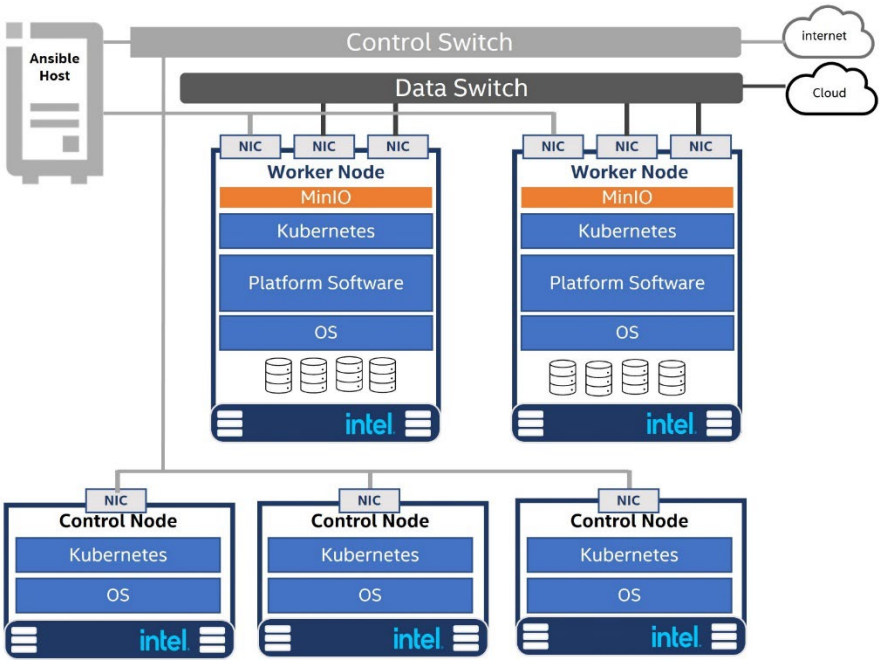

Figure 9. BMRA Storage for MinIO Object Storage

This section contains a step-by-step description of how to set up your BMRA Storage Configuration Profile flavor.

To use the Storage Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to Section 12.1 for details. You also need to build your Kubernetes cluster.
2. Download the Ansible playbook for your Configuration Profile. Refer to Section 12.2 for details.
3. Configure the optional Ansible parameters using the information in the Configuration Profile tables. Refer to Section 12.3 for details.
4. Deploy the platform. Refer to Section 12.4 for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

Be aware of the definitions of terminology used in tables in this section.

TERM	DESCRIPTION
Hardware Taxonomy	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, or similar value.)
DISABLED	Setting must be disabled in the BIOS (configured as Disabled, No, False, or any other value with this meaning.)
OPTIONAL	Setting can be either disabled or enabled, depending on user's workload. Setting does not affect the Configuration Profile or platform deployment.
Software Taxonomy	
TRUE	Feature is included and enabled by default.
FALSE	Feature is included but disabled by default - can be enabled and configured by user.
N/A	Feature is not included and cannot be enabled or configured.

12.1 Step 1 - Set Up Storage Configuration Profile Hardware

The tables in this section list the hardware BOM for the Storage Configuration Profile, including Control Node, Worker Node Base, and Worker Node Plus. We recommend that you set up at least one control node and one worker node.

Table 48. Hardware Setup for Storage Configuration Profile – 2nd Generation and 3rd Generation Intel Xeon Scalable Processors

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	N/A*	Controller_3rdGen_3
Worker Node Options	N/A*	Worker_3rdGen_Plus_4
*Configuration Profile only tested with 3rd Generation Intel Xeon Scalable processor		

Table 49. Hardware Setup for Storage Configuration Profile – Intel Xeon D Processor

NODE OPTIONS	INTEL XEON D PROCESSOR
Control Node Options	N/A*
Worker Node Options	N/A*

12.2 Step 2 - Download Storage Configuration Profile Ansible Playbook

This section contains details for downloading the Storage Configuration Profile Ansible playbook. It also provides an overview of the Ansible playbook and lists the software that is automatically installed when the playbook is deployed.

Download the Storage Configuration Profile Ansible playbook using the steps described in Section 2.5.

12.2.1 Storage Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Storage Configuration Profile allows you to provision a production-ready Kubernetes cluster. It also applies any additional requirements, such as host OS configuration or Network Adapter drivers and firmware updates. Every capability included in the Storage Configuration Profile playbook can be disabled or enabled. Refer to the diagram and group and host vars tables below to see which Ansible roles are included and executed by default.

The diagram shows the architecture of the Ansible playbooks and roles that are included in the Storage Configuration Profile.

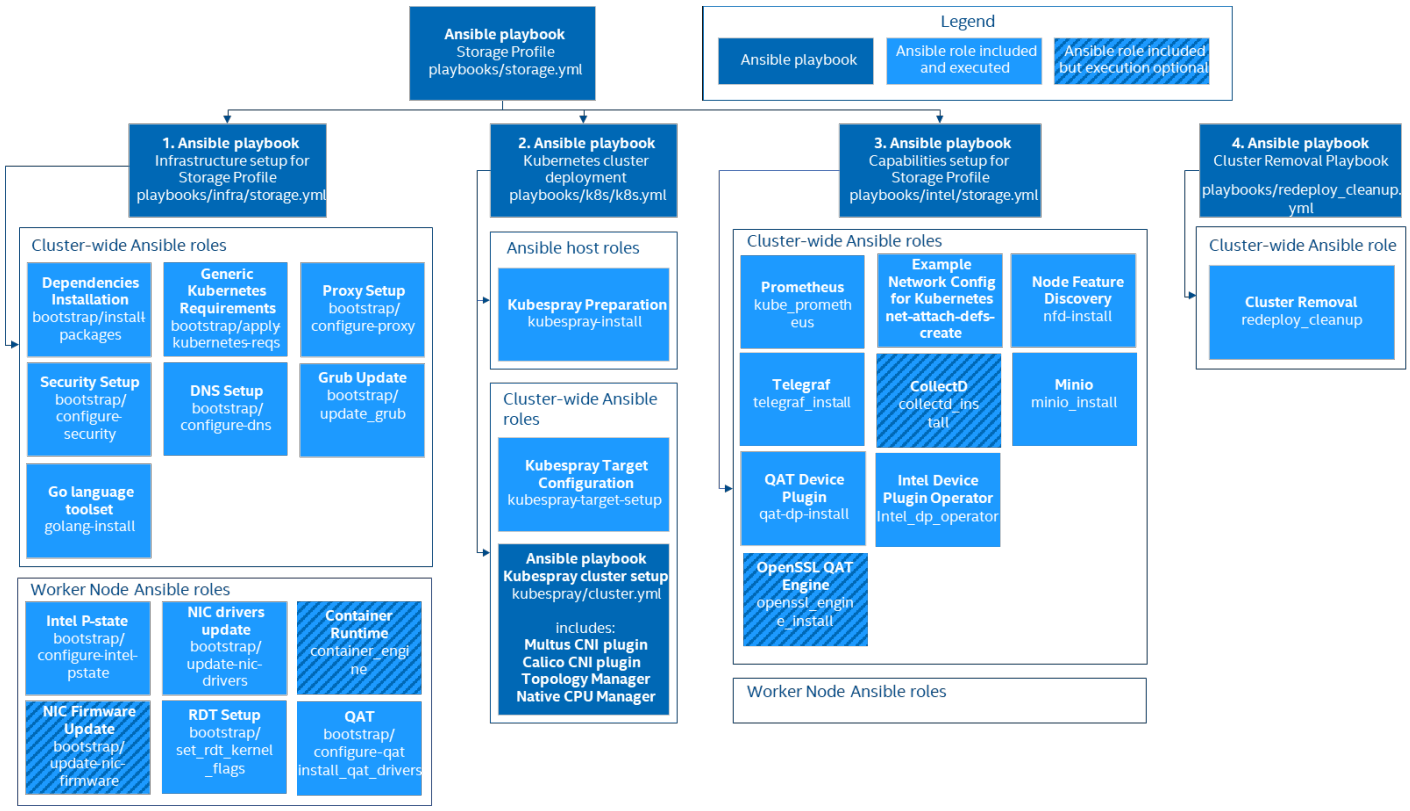

Figure 10. Storage Configuration Profile Ansible Playbook

12.3 Step 3 - Set Up Storage Configuration Profile

Review the optional Ansible group and host variables in this section and select options that match your desired configuration.

1. Update the `inventory.ini` file with your environment details as described in Section 2.3.3.
2. Create `host_vars` files for all worker nodes as specified in Section 2.3.4.
3. Update group and host variables to match your desired configuration as specified in Section 2.3.4. Refer to the tables in Section 12.3.1 and Section 12.3.2.

Variables are grouped into two main categories:

1. Group variables – apply to both control and worker nodes and have cluster-wide impact.
2. Host variables – scope is limited to a single worker node.

The tables below are a summary of group and host variables. For lists showing all configurable properties, see Section 6.3 and Section 6.4. All of the variables are important but pay special attention to variables in **bold** as they almost always need to be updated to match the target environment.

12.3.1 Storage Configuration Profile Group Variables

Table 50. Storage Configuration Profile – Group Variables

COMPONENT	VALUE	
Kubernetes	true	For the list of all configurable properties, see Section 6.3
nfd_enabled	true	
native_cpu_manager_enabled	true	
topology_manager_enabled	true	
qat_dp_enabled	false	
tas_enabled	true	
minio_enabled	true	
collectd_enabled	false	
telegraf_enabled	true	

12.3.2 Storage Configuration Profile Host Variables[21]

Table 51. Storage Configuration Profile – Host Variables

COMPONENT	VALUE	
iommu_enabled	false	For the list of all configurable properties, see Section 6.4
sriov_cni_enabled	false	
hugepages_enabled	false	
isolcpus_enabled	false	
qat_devices	[]	
dataplane_interfaces	[]	
minio_pv	[]	

12.4 Step 4 - Deploy and Validate Storage Configuration Profile Platform

Deploy the Storage Configuration Profile Ansible playbook using the steps described in Section 2.5.5.

Validate the setup of your Kubernetes cluster. Refer to the tasks in Section 5 and run the validation processes according to the hardware and software components that you have installed.

[21] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

Part 3:
BMRA Applications

13 Workloads and Application Examples

This section provides examples of how to provision and deploy example applications or workloads.

13.1 Enabling Key Management NGINX Applications

KMRA source code and Dockerfiles: https://01.org/key-management-reference-application-kmra

KMRA docker images on Docker hub:

- Apphsm: https: //hub.docker.com/r/intel/apphsm
- Ctk_loadkey: https://hub.docker.com/r/intel/ctk_loadkey
- PCCS: https://hub.docker.com/r/intel/pccs

KMRA Helm charts are located in `/roles/kmra_install/charts`.

Steps to deploy the full KMRA NGINX demo:

1. Generate a new PCCS primary API key and update the `kmra_pccs_api_key` variable in `group_vars/all.yml` (go to https://api.portal.trustedservices.intel.com/provisioning-certification and Subscribe).
2. Ensure that the `kmra_deploy_demo_workload` variable in the `group_vars/all.yml` is set to `true`.
3. Deploy the `full_nfv`, `on_prem`, or `remote_fp` profile to set up KMRA demo with NGINX. The `kmra` variable must be set to `on` in `profiles/profiles.yml`.

13.2 Enabling Trusted Certificate Service

Trusted Certificate Service (TCS) is a Kubernetes certificate signing solution that uses the security capabilities provided by the Intel® SGX. The signing key is stored and used inside the SGX enclaves and is never stored in clear anywhere in the system. TCS is implemented as a cert-manager external issuer by supporting both cert-manager and Kubernetes certificate signing APIs.

To enable TCS on BMRA, follow the guide available at https://github.com/intel/trusted-certificate-issuer

13.2.1 Istio Custom CA Integration using Kubernetes CSR

Istio supports integrating custom certificate authority(CA) using Kubernetes CSR as an experimental feature.

Detailed example steps described in the https://github.com/intel/trusted-certificate-issuer/blob/main/docs/istio-custom-ca-with-csr.md document show how to provision Istio workload certificates using an Issuer provided by the Trusted Certificate Service (TCS).

Note: Due to misconfiguration of the Istio Demo application, you might need to disable `hugepages` temporarily to avoid demo app stuck in the `CrashLoopBackOff` state. To disable hugepages, execute following command on the worker node: `echo 0 > /proc/sys/vm/nr_hugepages`

13.2.2 Remote Attestation and Manual Key Management

Trusted Certificate Service (TCS) supports SGX remote attestation and sample key management reference application.

All required steps are described in the https://github.com/intel/trusted-certificate-issuer/blob/main/docs/integrate-key-server.md document.

Part 4:
BMRA Release Notes

Appendix A BMRA Release Notes

This section lists the notable changes from the previous releases, including new features, bug fixes, and known issues.[22]

A.1 BMRA 22.01 Notable Facts

New Components:

- Intel Service Mesh
- MinIO Object Storage
- Intel Power Manager (Power Operator)
- Platform Aware Scheduling (Telemetry Aware Scheduling + GPU Aware Scheduling)
- Intel DLB (Dynamic Load Balancer)

New Platforms:

- Taylors Falls Reference Design (Intel Xeon-D)

Updates/Changes:

- Playbooks and profile config files are generated automatically
- Profiles list expanded with 'Storage'
- RHEL bumped to version 8.5 as base operating system
- Version upgrades for key components:
  - DPDK = 21.11
  - Kubernetes = 1.22
  - Kubespray = 2.17
  - KMRA = 1.4

Removed support:

- Intel CPU Manager for Kubernetes (CMK)
- CentOS (all distro versions) as base operating system

Known Limitations (Errata):

- Settings related to Intel P-state and Intel Turbo Boost Technology are not enforced until the target is rebooted, leading certain tasks to not execute properly (see: https://github.com/intel/container-experience-kits/issues/74)

A.2 BMRA 22.01 Bug Fixes

The following bug fixes were completed in the BMRA 22.01 release:

- Missing makefile: https://github.com/intel/container-experience-kits/issues/72
- Installing dependencies on localhost: https://github.com/intel/container-experience-kits/issues/73
- Duplicate code within the Profile specific playbooks: https://github.com/intel/container-experience-kits/issues/78
- Using role/vars/main.yml to set defaults makes changing almost impossible: https://github.com/intel/container-experience-kits/issues/79
- SRIOV-CNI build fails on Ubuntu20: https://github.com/intel/container-experience-kits/issues/80

A.3 BMRA 21.09 New Features

The following new features were updated or added in the BMRA 21.09 release:

- Support for Istio service mesh operator, Envoy, and control plane
- Support for Telegraf telemetry collection
- Support Intel Telemetry Insight Reports
- Support for additional container runtime: CRI-O
- Updated default network plugin: Calico
- Support Intel® Speed Select Technology - Performance Profile (Intel® SST-PP)
- Support for rendering profile config files from template
- Updated Intel® Ethernet 700 and 800 Network Adapter drivers
- Updated Intel® Software Guard Extensions (Intel® SGX) Software Development Kit (SDK)
- Updated Data Plane Development Kit (DPDK) and Open vSwitch (OVS) DPDK for use of AVX-512 instruction sets
- Updated Prometheus, Grafana, and Node Exporter telemetry packages
- Updated Node Feature Discovery (NFD)
- Updated Multus container network interface (CNI)
- Updated OpenSSL toolkit

[22] See backup for workloads and configurations or visit www.Intel.com/PerformanceIndex. Results may vary.

- Updated Intel® QuickAssist Technology Engine for OpenSSL (Intel® QAT Engine for OpenSSL)
- Updated Intel® Multi-Buffer Crypto for IPSec (intel-ipsec-mb)

A.4 BMRA 21.09 Bug Fixes

The following bug fixes were completed in the BMRA 21.09 release:

- Fixed inventory groups for inclusive terminology
- Fixed kubelet –cpu-cfs-quota to eliminate performance throttling
- Fixed QAT driver VF binding issue on RHEL 8.4
- Fixed inadvertent Intel SST-CP frequency throttling with proportional settings

A.5 BMRA 21.08 New Features

The following new features were updated or added in the BMRA 21.08 release:

- Updated Intel® Ethernet 700 and 800 Network Adapter drivers
- Updated Intel® Ethernet 800 Dynamic Device Personalization (DDP) profiles
- Updated Intel device plugins (Intel QAT, Intel® Software Guard Extensions (Intel® SGX), Intel® Server GPU)
- Support additional operating system versions: RHEL 8.4 and Ubuntu 21.04
- Support additional container runtime: containerd
- Support Kubernetes version 1.21
- Updated Kubernetes features: Node Feature Discovery (NFD), Telemetry Aware Scheduling (TAS), and SR-IOV device plugin (DP)
- Support Kubernetes Operators for: Intel® device plugin operator (Intel SGX, Intel Server GPU) and SR-IOV network
- Support Intel® QuickAssist Technology Engine for OpenSSL (Intel® QAT Engine for OpenSSL)
- Support containerized Intel® SGX Key Management Service (KMS) including integration of Key Management Reference Application (KMRA) version 1.2.1
- Updated collectd, Prometheus, and Grafana components
- Support for DPDK 21.05 and OVS 2.15
- Support Ansible Cluster Removal Playbook for cluster teardown and redeployment

A.6 BMRA 21.08 Bug Fixes

The following bug fixes were completed in the BMRA 21.08 release:

- Fixed cluster recovery errors on reboot
- Fixed TAS demo policy failure
- Fixed deployment failure with Intel® Turbo Boost Technology disabled
- Fixed SGX DP pod crashes
- Fixed mismatch in available Intel QAT resources
- Fixed deployment failure with missing Intel QAT configuration
- Fixed kernel header mismatch for compiled kernel modules
- Fixed package installation dependencies
- Fixed isolcpu generation for configurations with HT disabled
- Fixed DPDK installation failures
- Fixed DNS file permission issues
- Fixed IP dependency in inventory file

A.7 Known Issues

Issue:

Occasionally the sriov-network-device-plugin does not detect new or updated VF resources.

Detail:

There is a known issue with sriov-network-device-plugin where the service fails to detect new or updated VF resources if not available when the service creates its ConfigMap and loads the daemonset. See https://github.com/k8snetworkplumbingwg/sriov-network-device-plugin/issues/276.

Workaround:

Delete the sriov-device-plugin-pod and resources will be present when pod is automatically restarted.

Issue:

Intel® Speed Select tool errors on 3rd Generation Intel® Xeon® Scalable processor servers with RHEL 8.2.

Detail:

The currently distributed RHEL 8.2 kernels are not compiled with CONFIG_INTEL_SPEED_SELECT_INTERFACE enabled.

Workaround:

Upgrade to RHEL 8.3.

The RHEL 8.2 default kernel can be recompiled with this setting (check your vendor support before proceeding). Alternatively, the Intel SST-BF feature is available on selected 2nd Generation Intel® Xeon® Scalable processor SKUs (https://access.redhat.com/articles/4481221) and both the Intel SST-BF and Intel SST-CP features are available on select 3rd Generation Intel® Xeon® Scalable processor SKUs with other supported OSes.

Issue:

KMRA PCCS pod fails to load on Ubuntu 20.04.

Detail:

The PCCS container requires kernel 5.4.0-65 or greater.

Workaround:

Update your Ubuntu 20.04 release to Ubuntu 20.04.3 or newer.

Issue:

collectd pod fails to start on Ubuntu 18.04 inbox kernel.

Detail:

```
intel_rapl: driver does not support CPU family 6 model 106
```

The Intel RAPL power capping driver in the 18.04 inbox kernel does not support 3rd Generation Intel® Xeon® Scalable processors.

Workaround:

Use update_kernel in Ansible group_vars or install a more recent OS with a newer kernel.

Issue:

collectd plugin fails to start.

Detail:

On some platforms, the collectd pod fails to start due to various plugin incompatibilities.

Workaround:
Disable problematic collectd plugins by adding to the exclude_collectd_plugins list in the Ansible host_vars configuration file.

Part 5: Abbreviations

Appendix B Abbreviations

The following abbreviations are used in this document.

ABBREVIATION	DESCRIPTION
AGF	Access Gateway Function
AIA	Accelerator Interfacing Architecture
AMX	Advance Matrix Multiply
BIOS	Basic Input/Output System
BMRA	Bare Metal Reference Architecture
CA	Certificate Authority
CDN	Content Delivery Network
CLOS	Class of Service
CMK	CPU Manager for Kubernetes
CMTS	Cable Modem Termination System
CNI	Container Network Interface
CO	Central Office
CTK	Crypto-API Toolkit
CXL	Compute Express Link
DDP	Dynamic Device Personalization
DHCP	Dynamic Host Configuration Protocol
DLB	Intel® Dynamic Load Balancer (Intel® DLB)
DNS	Domain Name Service
DPDK	Data Plane Development Kit
DRAM	Dynamic Random Access Memory
DSA	Intel® Data Streaming Accelerator (Intel® DSA)
EIST	Enhanced Intel Speedstep® Technology
FPGA	Field-Programmable Gate Array
FW	Firmware
GAS	GPU Aware Scheduling
GPU	Graphics Processor Unit
HA	High Availability
HCC	High Core Count
HSM	Hardware Security Model
HT	Hyper Threading
IAX	In-Memory Analytics
IMC	Integrated Memory Controller
Intel® AVX	Intel® Advanced Vector Extensions (Intel® AVX)
Intel® AVX-512	Intel® Advanced Vector Extension 512 (Intel® AVX-512)
Intel® DCAP	Intel® Software Guard Extensions Data Center Attestation Primitives (Intel® SGX DCAP)
Intel® DLB	Intel® Dynamic Load Balancer (Intel® DLB)
Intel® DSA	Intel® Data Streaming Accelerator (Intel® DSA)
Intel® HT Technology	Intel® Hyper-Threading Technology (Intel® HT Technology)
Intel® QAT	Intel® QuickAssist Technology (Intel® QAT)
Intel® RDT	Intel® Resource Director Technology (Intel® RDT)
Intel® SGX	Intel® Software Guard Extensions (Intel® SGX)
Intel® SST-BF	Intel® Speed Select Technology – Base Frequency (Intel® SST-BF)

ABBREVIATION	DESCRIPTION
Intel® SST-CP	Intel® Speed Select Technology – Core Power (Intel® SST-CP)
Intel® SST-PP	Intel® Speed Select Technology – Performance Profile (Intel® SST-PP)
Intel® SST-TF	Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF)
Intel® VT-d	Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d)
Intel® VT-x	Intel® Virtualization Technology (Intel® VT) for IA-32, Intel® 64 and Intel® Architecture (Intel® VT-x)
IOMMU	Input/Output Memory Management Unit
ISA	Instruction Set Architecture
I/O	Input/Output
K8s	Kubernetes
KMRA	Key Management Reference Application (KMRA)
KMS	Key Management Service (KMS)
LCC	Low Core Count
LLC	Last Level Cache
LOM	LAN on Motherboard
NFD	Node Feature Discovery
NFV	Network Function Virtualization
NIC	Network Interface Card
NTP	Network Time Protocol
NUMA	Non-Uniform Memory Access
NVM	Non-Volatile Memory
NVMe	Non-Volatile Memory
OAM	Operation, Administration, and Management
OCI	Open Container Initiative
OS	Operating System
OVS	Open vSwitch
OVS DPDK	Open vSwitch with DPDK
PBF	Priority Based Frequency
PCCS	Provisioning Certification Caching Service
PCI	Physical Network Interface
PCIe	Peripheral Component Interconnect express
PMD	Poll Mode Driver
PXE	Preboot Execution Environment
QAT	Intel® QuickAssist Technology
QoS	Quality of Service
RA	Reference Architecture
RAS	Reliability, Availability, and Serviceability
RDT	Intel® Resource Director Technology
RHEL	Red Hat Enterprise Linux
S3	Amazon Web Services Simple Storage Service
S-IOV	Intel® Scalable I/O Virtualization (Intel® Scalable IOV)
SA	Service Assurance
SGX	Intel® Software Guard Extensions (Intel® SGX)
SR-IOV	Single Root Input/Output Virtualization
SSD	Solid State Drive
SSH	Secure Shell Protocol

ABBREVIATION	DESCRIPTION
SVM	Shared Virtual Memory
TAS	Telemetry Aware Scheduling
TCS	Intel® Trusted Certificate Service
TDP	Thermal Design Power
TLS	Transport Layer Security
TME	Total Memory Encryption
TMUL	Tile Multiply
UEFI	Unified Extensible Firmware Interface
UPF	User Plane Function
vBNG	Virtual Broadband Network Gateway
vCMTS	Virtual Cable Modem Termination System
VF	Virtual Function
VPP	Vector Packet Processing

0322/DN/WIPRO/PDF 721796-002US