

# Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

## Authors

Joel A. Gibson

Octavia Carotti

Shivapriya Hiremath

Dave Cremins

Dana Nehama

Michael Pedersen

Veronika Karpenko

## 1 Introduction

The **Container Bare Metal Reference Architecture (BMRA)** is a cloud-native, forward-looking Kubernetes-cluster template solution for network implementations. This guide provides instructions on how to set, install, provision, and run a Container Bare Metal Kubernetes-cluster setup based on Intel® platforms and open-source software. In addition, the guide provides access to a set of open-source Ansible scripts that enable automatic and easy provisioning of the BMRA using optimized configurations, thus, decreasing the installation time from days to a few hours.

The Container BMRA Release 21.03 is a major release that supports the recently launched 3rd Generation Intel® Xeon® Scalable processors and the value-add capabilities it introduces.

Additionally, this release provides technology guidelines for two major use cases:

- **Media** – enabling optimized visual-cloud services based on both Intel Xeon and the Intel® Server GPU card.
- **Security** – enabling network access and security-key-management deployments with Intel® Software Guard Extensions (Intel® SGX). The guide also provides a proof-of-concept software created to demonstrate the integration of Intel SGX asymmetric key capability with a hardware security model (HSM) on a centralized key server.

This guide describes the set of hardware and open-source software components forming the BMRA. The hardware platforms supported are based on 2nd Generation Intel® Xeon® Scalable processors, 3rd Generation Intel® Xeon® Scalable processors, Intel accelerators, and other advanced Intel platform technologies. **Although the provisioning of BMRA Kubernetes clusters is completely automated, using open-source Ansible scripts**, the guide goes into great detail to describe the software capabilities and the optimized hardware and software configurations that are implemented by the Ansible scripts.

If you wish to learn about the BMRA, Part 1 of this document will provide you with an overview (sections 1 – 2) and Part 2 (sections 3 – 7) will provide you with more profound technology and implementation details. If you wish to start building your BMRA Flavor right away, you may go directly to Part 3 of this document (Appendixes A – H). Part 4 is the BMRA Release Notes. Part 5 contains examples of applications and workloads.

Network deployments vary by location; each location imposes different hardware and software specifications and configurations due to varying workloads, cost, density, and performance requirements. To address these matters, the Container BMRA supports the concept of **Reference Architecture Configuration Profiles**.

The following Reference Architecture Configuration Profiles are defined for BMRA 21.03 and more will be introduced in upcoming releases to meet specific workload needs. Three of the Reference Architecture 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), Virtual Broadband Network Gateway (vBNG), and User Plane Function (UPF).
- **Regional Data Center Configuration Profile** (newly introduced in Release 21.03) – Central-office location typical Configuration Profile. Currently tailored exclusively for Media Visual Processing workloads such as CDN Transcoding.

Two 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.

A set of Ansible scripts is used to automatically configure the Reference Architecture Configuration Profile. These Ansible scripts generate a **Reference Architecture Flavor** (or **BMRA Flavor**). The BMRA 21.03 provides Ansible scripts to generate a BMRA Flavor for each of the current five Reference Architecture Configuration Profiles.

BMRA Release 21.03 includes a significant portfolio of technologies that enable a variety of On-Premises to Core network use cases addressing: Security, Media, Cable (vCMTS), Central office (AGF, vBNG, 5G-UPF), and Smart City deployments. Following are the main new hardware and software technologies and upgrades. For the full list, refer to the [BMRA Release Notes](#).

- 3rd Generation Intel® Xeon® Scalable processor
- Intel® Ethernet 800 Series Network Adapters with Dynamic Device Personalization (DDP) profiles (update)
- Intel® Server GPU
- Intel® Quick Assist Technology (Intel® QAT) drivers and services
- Intel® Optane™
- Intel® Advanced Vector Extensions 512 (Intel® AVX-512) enhancements
- Security with Intel® SGX Key Management and NGINX reference application (KMRA)
- Power Management including support for Intel® Speed Select Technology – Core Power (Intel® SST-CP) and Intel® Speed Select Technology – Base Frequency (Intel® SST-BF) (update)
- Media support on Intel® Xeon® processor or with Intel® Server GPU
- Kubernetes version update to 1.19.x (1.20 not automated)
- Kubernetes Device Plugin for Intel® Software Guard Extensions (Intel® SGX)
- Telemetry with new Collectd plugins (unixsock, network)
- Grafana dashboards (cpu, disk, intel, ipmi, netlink, ovs, power, numa, hugepages, ethstats)
- Multiple Data Plane Development Kit (DPDK) version options with custom patch support
- Multiple SR-IOV driver assignments per physical function (PF)

Experience Kits, the collaterals that explain in great detail the technologies enabled in BMRA release 21.03, including benchmark information, are available in the following locations:

<https://networkbuilders.intel.com/intel-technologies/network-transformation-exp-kits>,  
<https://networkbuilders.intel.com/intel-technologies/container-experience-kits>, and  
<https://networkbuilders.intel.com/intel-technologies/3rd-gen-intel-xeon-scalable-processors-experience-kits>.

## Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>1</b>
1.1	About this Document .....	6
1.2	Terminology .....	6
1.3	Taxonomy.....	8
1.4	Reference Documents.....	8
<b>2</b>	<b>Reference Architecture Overview .....</b>	<b>9</b>
2.1	Architecture Delivered.....	9
2.2	Reference Architecture Configuration Profiles.....	11
2.3	Use Cases by Network Location .....	12
2.4	Configuration Profile Installation Playbooks.....	13
2.5	Hardware Components .....	13
2.6	Software Capabilities.....	14
2.6.1	Kubernetes Features .....	14
2.6.2	Platform System Features.....	14
2.6.3	Observability .....	15
<b>Part 2:</b>	<b>.....</b>	<b>16</b>
<b>3</b>	<b>Reference Architecture Deployment – Ansible Playbooks .....</b>	<b>17</b>
3.1	Reference Architecture Installation Prerequisites.....	17
3.1.1	Hardware BOM Selection and Setup for Control and Worker Nodes .....	17
3.1.2	BIOS Selection for Control and Worker Nodes .....	17
3.1.3	Operating System (OS) Selection for Control and Worker Nodes.....	17
3.1.4	Network Interface Requirements for Control and Worker Nodes.....	18
3.1.5	Software Prerequisites for Ansible Host, Control Nodes, and Worker Nodes .....	18
3.2	Ansible Playbook Review.....	18
3.2.1	Ansible Playbooks Building Blocks .....	18
3.2.2	Ansible Playbook Phases.....	19
3.3	Deployment using Ansible Playbook .....	20
3.3.1	Prepare Target Servers.....	20
3.3.2	Get Ansible Playbook and Prepare Configuration Templates .....	20
3.3.3	Update Ansible Inventory File.....	21
3.3.4	Update Ansible Host and Group Variables.....	22
3.3.5	Run Ansible Playbook.....	22
<b>4</b>	<b>Software Capabilities Review .....</b>	<b>22</b>
4.1	Kubernetes Plugins.....	22
4.1.1	Multus CNI.....	22
4.1.2	SR-IOV Network Device Plugin.....	23
4.1.3	SR-IOV CNI.....	23
4.1.4	Userspace CNI.....	23
4.1.5	Bond CNI.....	23
4.1.6	Intel® QuickAssist Device Plugin .....	23
4.1.7	Intel® Software Guard Extensions (SGX) Device Plugin .....	23
4.2	Kubernetes Features .....	23
4.2.1	Node Feature Discovery.....	24
4.2.2	Topology Manager.....	25
4.2.3	Kubernetes Native CPU Manager.....	25
4.2.4	CPU Manager for Kubernetes (CMK).....	25
4.2.5	Telemetry Aware Scheduling .....	25
4.3	Real-Time System Support.....	26
4.4	Dynamic Device Personalization (DDP) .....	26
4.4.1	DDP on Intel Ethernet 700 Series Network Adapters .....	26
4.4.2	DDP on Intel® Ethernet 800 Series Network Adapters.....	27
4.5	Intel® Speed Select Technologies.....	29
4.5.1	Intel® Speed Select – Base Frequency.....	29
4.5.2	Intel® Speed Select – Core Power .....	30
4.6	Security.....	30
4.6.1	Cluster Security .....	30
4.6.2	Intel® Security Libraries for Data Center (Intel® SecL – DC).....	30
4.6.3	Intel® Software Guard Extensions.....	31
4.7	Intel® Server GPU.....	31
4.8	Security - Key Management Reference Application with Intel® SGX .....	32
4.9	Observability .....	32
4.9.1	Observability Components Overview .....	33

# Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

4.9.2	Platform Telemetry Security .....	34
<b>5</b>	<b>Reference Architecture Hardware Components and BIOS .....</b>	<b>34</b>
5.1	Hardware Component List for Control Node .....	34
5.2	Hardware Component List for Worker Node Base .....	35
5.3	Hardware Component List for Worker Node Plus.....	36
5.4	Hardware BOMs for all Configuration Profiles.....	37
5.5	Platform BIOS .....	40
<b>6</b>	<b>Reference Architecture Software Components.....</b>	<b>43</b>
<b>7</b>	<b>Post Deployment Verification Guidelines.....</b>	<b>44</b>
7.1	Check the Kubernetes Cluster .....	45
7.2	Check Intel® Speed Select Technology – Base Frequency (Intel® SST-BF) Configuration on 2nd Generation Intel® Xeon® Scalable Processor.....	46
7.3	Check Intel® Speed Select Technology on 3rd Generation Intel® Xeon® Scalable Processor .....	46
7.3.1	Check Intel® Speed Select Technology - Base Frequency (Intel® SST-BF) Configuration.....	46
7.3.2	Check Intel® Speed Select Technology – Core Power (Intel® SST-CP).....	47
7.4	Check DDP Profiles .....	48
7.4.1	Check DDP Profiles in Intel® Ethernet 700 Series Network Adapters .....	48
7.4.2	Check DDP Profiles in Intel® Ethernet 800 Series Network Adapters .....	48
7.4.3	Check SR-IOV Resources .....	48
7.5	Check Node Feature Discovery (NFD) .....	48
7.6	Check CPU Manager for Kubernetes .....	50
7.7	Check Topology Manager.....	52
7.7.1	Change Topology Manager Policy: Redeploy Kubernetes Playbook.....	53
7.7.2	Change Topology Manager Policy: Manually Update Kubelet Flags .....	53
7.8	Check Intel Device Plugins for Kubernetes.....	53
7.8.1	Check SR-IOV Network Device Plugin.....	53
7.8.2	Check QAT Device Plugin .....	55
7.8.3	Check SGX Device Plugin.....	55
7.9	Check Networking Features (after Installation) .....	56
7.9.1	Check Multus CNI Plugin.....	56
7.9.2	Check SR-IOV CNI Plugin.....	57
7.9.3	Check Userspace CNI Plugin.....	57
7.9.4	Check Bond CNI Plugin .....	58
7.10	Check Grafana Telemetry Visualization.....	60
7.11	Check Telemetry Aware Scheduler .....	60
7.11.1	Check Dontschedule Policy .....	61
7.11.2	Check Deschedule Policy.....	61
7.12	Check Key Management infrastructure with Intel® SGX.....	61
7.13	Check Intel® Server GPU device and driver.....	61
<b>8</b>	<b>Conclusion – Automation Eases Reference Application Deployment.....</b>	<b>62</b>
<b>Part 3:</b>	<b>.....</b>	<b>63</b>
<b>Appendix A</b>	<b>BMRA Setup for All Flavors and Configuration Profile Options .....</b>	<b>65</b>
A.1	Set Up an Ansible Host.....	65
A.2	Set Up the Control and Worker Nodes - BIOS Prerequisites.....	65
A.3	Software BOMs for all BMRA Configuration Profiles .....	66
<b>Appendix B</b>	<b>BMRA Basic Configuration Profile Setup.....</b>	<b>70</b>
B.1	Step 1 - Set Up Basic Configuration Profile Hardware.....	70
B.2	Step 2 - Download Basic Configuration Profile Ansible Playbook.....	70
B.2.1	Basic Configuration Profile Ansible Playbook Overview .....	71
B.2.2	Basic Configuration Profile Software BOM.....	71
B.3	Step 3 - Set Up Basic Configuration Profile.....	73
B.3.1	Basic Configuration Profile Group Variables .....	73
B.3.2	Basic Configuration Profile Host Variables .....	74
B.4	Step 4 - Deploy Basic Configuration Profile Platform.....	75
B.5	Step 5 - Validate Basic Configuration Profile.....	75
<b>Appendix C</b>	<b>BMRA Full Configuration Profile Setup .....</b>	<b>77</b>
C.1	Step 1 - Set Up Full Configuration Profile Hardware.....	77
C.2	Step 2 - Download Full Configuration Profile Ansible Playbook.....	77
C.2.1	Full Configuration Profile Ansible Playbook Overview.....	78
C.2.2	Full Configuration Profile Software BOM.....	78
C.3	Step 3 - Set Up Full Configuration Profile .....	80

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

C.3.1	Full Configuration Profile Group Variables.....	80
C.3.2	Full Configuration Profile Host Variables.....	83
C.4	Step 4 - Deploy Full Configuration Profile Platform.....	85
C.5	Step 5 - Validate Full Configuration Profile.....	85
<b>Appendix D</b>	<b>BMRA On-Premises Edge Configuration Profile Setup.....</b>	<b>87</b>
D.1	Step 1 - Set Up On-Premises Edge Configuration Profile Hardware.....	87
D.2	Step 2 - Download On-Premises Edge Configuration Profile Ansible Playbook.....	87
D.2.1	On-Premises Edge Configuration Profile Ansible Playbook Overview.....	88
D.2.2	On-Premises Edge Configuration Profile Software BOM.....	88
D.3	Step 3 - Set Up On-Premises Edge Configuration Profile.....	90
D.3.1	On-Premises Edge Configuration Profile Group Variables.....	90
D.3.2	On-Premises Edge Configuration Profile Host Variables.....	92
D.4	Step 4 - Deploy On-Premises Edge Configuration Profile Platform.....	94
D.5	Step 5 - Validate On-Premises Edge Configuration Profile.....	95
<b>Appendix E</b>	<b>BMRA Remote CO-Forwarding Configuration Profile Setup.....</b>	<b>97</b>
E.1	Step 1 - Set Up Remote CO-Forwarding Configuration Profile Hardware.....	97
E.2	Step 2 - Download Remote CO-Forwarding Configuration Profile Ansible Playbook.....	97
E.2.1	Remote CO-Forwarding Configuration Profile Ansible Playbook Overview.....	98
E.2.2	Remote CO-Forwarding Configuration Profile Software BOM.....	98
E.3	Step 3 - Set Up Remote CO-Forwarding Configuration Profile.....	100
E.3.1	Remote CO-Forwarding Configuration Profile Group Variables.....	100
E.3.2	Remote CO-Forwarding Configuration Profile Host Variables.....	102
E.4	Step 4 - Deploy Remote CO-Forwarding Configuration Profile Platform.....	105
E.5	Step 5 - Validate Remote-CO Forwarding Configuration Profile.....	105
<b>Appendix F</b>	<b>BMRA Regional Data Center Configuration Profile Setup.....</b>	<b>107</b>
F.1	Step 1 - Set Up Regional Data Center Configuration Profile Hardware.....	107
F.2	Step 2 - Download Regional Data Center Configuration Profile Ansible Playbook.....	107
F.2.1	Regional Data Center Configuration Profile Ansible Playbook Overview.....	108
F.2.2	Regional Data Center Configuration Profile Software BOM.....	108
F.3	Step 3 - Set Up Regional Data Center Configuration Profile.....	110
F.3.1	Regional Data Center Configuration Profile Group Variables.....	110
F.3.2	Regional Data Center Configuration Profile Host Variables.....	112
F.4	Step 4 - Deploy Regional Data Center Configuration Profile Platform.....	114
F.5	Step 5 - Validate Regional Data Center Configuration Profile.....	114
<b>Appendix G</b>	<b>Ansible Host Setup.....</b>	<b>116</b>
<b>Part 4:</b>	.....	<b>117</b>
<b>Appendix H</b>	<b>BMRA Release Notes.....</b>	<b>118</b>
H.1	BMRA 21.03 New Features.....	118
H.2	BMRA 21.03 Bug Fixes.....	118
H.3	BMRA 2.1 New Features.....	118
H.4	BMRA 2.1 Bug Fixes.....	118
H.5	BMRA 2.0 New Features.....	119
H.6	BMRA 2.0 Bug Fixes.....	119
H.7	Known Issues.....	119
<b>Part 5:</b>	.....	<b>121</b>
<b>Appendix I</b>	<b>Workloads and Application Examples.....</b>	<b>122</b>
I.1	Enabling Key Management NGINX Applications.....	122

## Figures

Figure 1.	Example of Container Bare Metal Reference Architecture Kubernetes Cluster Setup.....	10
Figure 2.	Key Components forming the Container Bare Metal Reference Architecture Kubernetes Cluster.....	10
Figure 3.	Identified Key Network Locations.....	12
Figure 4.	Example of Container Bare Metal Configuration Profiles Per Location.....	12
Figure 5.	Ansible Playbooks Overview.....	13
Figure 6.	Platform Telemetry Available with Collectd.....	15
Figure 7.	High Level BMRA Ansible Playbooks Architecture.....	19
Figure 8.	Features Detected by NFD.....	24
Figure 9.	CPU Core Frequency Deployment Methods.....	29
Figure 10.	Key Management Reference Application Infrastructure with Intel® SGX.....	32
Figure 11.	Platform Telemetry Available with Collectd.....	33

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

Figure 12.	Grafana Dashboard Example.....	60
Figure 13.	Basic Configuration Profile Ansible Playbook.....	71
Figure 14.	Full Configuration Profile Ansible Playbook.....	78
Figure 15.	On-Premises Edge Configuration Profile Ansible Playbook.....	88
Figure 16.	Remote CO-Forwarding Configuration Profile Ansible Playbook.....	98
Figure 17.	Regional Data Center Configuration Profile Ansible Playbook.....	108

## Tables

Table 1.	Abbreviations.....	6
Table 2.	Hardware and Software Taxonomy .....	8
Table 3.	Reference Documents.....	8
Table 4.	Collectd Plugins .....	33
Table 5.	Hardware Options for Control Node – 2nd Generation Intel Xeon Scalable Processor .....	34
Table 6.	Hardware Options for Control Node – 3rd Generation Intel Xeon Scalable Processor.....	35
Table 7.	Hardware Components for Worker Node Base – 2nd Generation Intel Xeon Scalable Processor.....	35
Table 8.	Hardware Components for Worker Node Base – 3rd Generation Intel Xeon Scalable Processor.....	36
Table 9.	Hardware Components for Worker Node Plus – 2nd Generation Intel Xeon Scalable Processor.....	36
Table 10.	Hardware Components for Worker Node Plus – 3rd Generation Intel Xeon Scalable Processor.....	37
Table 11.	Control Node Hardware Setup for all Configuration Profiles – 2nd Generation Intel Xeon Scalable Processor .....	37
Table 12.	Control Node Hardware Setup for all Configuration Profiles – 3rd Generation Intel Xeon Scalable Processor .....	38
Table 13.	Worker Node Base Hardware Setup for all Configuration Profiles – 2nd Generation Intel Xeon Scalable Processor.....	39
Table 14.	Worker Node Plus Hardware Setup for all Configuration Profiles – 2nd Generation Intel Xeon Scalable Processor.....	39
Table 15.	Worker Node Base Hardware Setup for all Configuration Profiles – 3rd Generation Intel Xeon Scalable Processor.....	39
Table 16.	Worker Node Plus Hardware Setup for all Configuration Profiles – 3rd Generation Intel Xeon Scalable Processor.....	40
Table 17.	Platform BIOS Settings for 2 <sup>nd</sup> Generation Intel® Xeon® Scalable Processor.....	41
Table 18.	Platform BIOS Settings for 3rd Generation Intel® Xeon® Scalable Processor.....	42
Table 19.	Software Components.....	43
Table 20.	BIOS Prerequisites for Control and Worker Nodes for Basic and Full Configuration Profiles.....	65
Table 21.	BIOS Prerequisites for Control and Worker Nodes for On-Premises Edge, Remote Co-Forwarding, and Regional Data Center Configuration Profiles.....	65
Table 22.	Software BOMs for Basic and Full Configuration Profiles .....	66
Table 23.	Software BOMs for On-Premises Edge, Remote Co-Forwarding, and Regional Data Center Configuration Profiles.....	67
Table 24.	Hardware Setup for Basic Configuration Profile .....	70
Table 25.	Software BOM for Basic Configuration Profile.....	71
Table 26.	Hardware Setup for Full Configuration Profile.....	77
Table 27.	Software BOM for Full Configuration Profile.....	78
Table 28.	Hardware Setup for On-Premises Edge Configuration Profile .....	87
Table 29.	Software BOM for On-Premises Edge Configuration Profile.....	88
Table 30.	Hardware Setup for Remote CO-Forwarding Configuration Profile .....	97
Table 31.	Software BOM for Remote CO-Forwarding Configuration Profile.....	98
Table 32.	Hardware Setup for Regional Data Center Configuration Profile .....	107
Table 33.	Software BOM for Remote Regional Data Center Configuration Profile .....	108

## Document Revision History

There are two previous editions of the BMRA document. One only covered 2nd Generation Intel® Xeon® Scalable processors, and the other only covered 2nd Generation and 3rd Generation Intel® Xeon® Scalable processors. The editions were released starting from April 2019.

REVISION	DATE	DESCRIPTION
001	November 2020	Initial release of the BMRA document for 2nd Generation and 3rd Generation Intel® Xeon® Scalable processors.
002	February 2021	Added Appendix for Ansible Host setup. Updated software versions. Updated the release notes to include details about bug fixes for Release 2.1 of the Container Bare Metal Reference Architecture (BMRA).
003	April 2021	Added Appendix for BMRA Regional Data Center Configuration Profile Setup. Added Appendix for Workloads and Application Examples. Introduced new hardware and software technologies and upgrades, including Intel® Server Graphics 1 support. Updated the release notes to include details about bug fixes for BMRA Release 21.03. Revised the document for public release to Intel® Network Builders.
004	April 2021	Added <a href="#">Section 7.13 Check Intel® Server GPU device and driver.</a>

## 1.1 About this Document

The BMRA document is composed of five parts, which are described below.

**Note:** This document goes into great detail to describe the hardware and software ingredients and the configuration options for each BMRA Configuration Profile. Be aware that most of these data points are informational because the setup of each resulting BMRA Flavor is executed automatically by an Ansible playbook.

**Part 1 (Chapters 1 and 2 in this guide):** The guide begins with an overview of the container Bare Metal Reference Architecture (BMRA), including: an introduction to the new concept of Reference Architecture Configuration Profiles, use cases supported, a description of software capabilities, and an overview of the Ansible scripts that enable automatic deployment.

**Part 2 (Chapters 3 – 7 in this guide):** This is the main part of the guide. It provides a complete and detailed description of the BMRA components and configuration values, and the processes used for deployment and verification.

- The Ansible Playbooks that generate the BMRA Flavors by implementing the BMRA Configuration Profiles are explained. The basic building blocks of Ansible Playbooks are described, along with details about how to use them for deployment.
- The next chapter explains all the common software capabilities, including Kubernetes features, packet-processing, telemetry, and others, so that you can evaluate which ones are appropriate for your scenario.
- The following chapters provide a complete description of the BIOS setting options, hardware and software components consumed by the BMRA, and the hardware and software configuration values set by the Ansible playbooks.
- The Post Deployment Verification Guidelines chapter describes a set of processes that you can use to verify all the components deployed by the scripts.

**Part 3 (Appendixes A – G):** Build your BMRA Flavor using customized instructions for each BMRA Configuration Profile.

Those wanting to start right away building a BMRA Flavor can start with Part 3 of the document. The appendixes provide step-by-step instructions on how to generate and deploy a BMRA Flavor according to a specific Configuration Profile. Each appendix contains the hardware and software BOMs and complete details for configuring and executing the appropriate playbook.

**Part 4 (Appendix H):** Release Notes for BMRA V21.03

The release notes provide an overview of the new capabilities, bug fixes, and known issues for this BMRA release.

**Part 5 (Appendix I):** Workloads and Application Examples

This part provides examples on how to provision and deploy example applications or workloads. For BMRA 21.03, instructions are provided for a Key Management reference application.

## 1.2 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 containers 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 defined 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.
- A **Kubernetes cluster** contains at least one *worker node* that runs containerized applications. *Pods* are the components of the application workload that are hosted on worker nodes. *Control nodes* manage the pods and worker nodes.

**Table 1. Abbreviations**

ABBREVIATION	DESCRIPTION
ADI	Ad Insertion
AV1	AOMedia Video 1 video coding format
AVC	Advanced Video Coding video compression standard

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

ABBREVIATION	DESCRIPTION
BIOS	Basic Input / Output System
BMRA	Bare Metal Reference Architecture
CDN	Content Delivery Network
CMK	Intel CPU Manager for Kubernetes
CNI	Container Networking Interface
CO	Central Office
DDP	Dynamic Device Personalization
DHCP	Dynamic Host Configuration Protocol
DPDK	Data Plane Development Kit
GPU	Graphics Processor Unit
HA	High Availability
HEVC	High Efficiency Video Coding video compression
IA	Intel® Architecture
Intel® HT Technology	Intel® Hyper-Threading Technology
Intel® QAT	Intel® QuickAssist Technology
Intel® SST-BF	Intel® Speed Select Technology – Base Frequency
Intel® VT-d	Intel® Virtualization Technology (Intel® VT) for Directed I/O
Intel® VT-x	Intel® Virtualization Technology (Intel® VT) for IA-32, Intel® 64 and Intel® Architecture
K8s	Kubernetes
MPEG-2	Moving Picture Experts Group standard for digital television and DVD video
NIC	Network Interface Card
NFD	Node Feature Discovery
NFV	Network Functions Virtualization
OS	Operating System
OVS DPDK	Open vSwitch with DPDK
QAT	Intel® QuickAssist Technology
RA	Reference Architecture
SA	Service Assurance
SDN	Software-Defined Networking
SGX	Intel® Software Guard Extensions (Intel® SGX)
SHVS	Standard High-Volume Servers
SIMD	Single Instruction, Multiple Data
SMTC	Smart City
SOCKS	Socket Secure
SR-IOV	Single Root Input/Output Virtualization
TAS	Telemetry Aware Scheduling
vBNG	Virtual Broadband Network Gateway
vCMTS	Virtual Cable Modem Termination System
VLAN	Virtual LAN
VNF	Virtual Network Function
VP9	Video coding format for streaming over the internet
VPP	Vector Packet Processing
VXLAN	Virtual Extensible LAN

### 1.3 Taxonomy

The following table describes the terminology used in the other tables in this document.

**Table 2. Hardware and Software Taxonomy**

TERM	DESCRIPTION
<b>Hardware Taxonomy</b>	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, etc.)
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.
<b>Software Taxonomy</b>	
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.

### 1.4 Reference Documents

Collaterals, including technical guides and solution briefs that explain in detail the technologies enabled in BMRA release 21.03 are available in the following locations: <https://networkbuilders.intel.com/intel-technologies/network-transformation-exp-kits>, <https://networkbuilders.intel.com/intel-technologies/container-experience-kits>, and <https://networkbuilders.intel.com/intel-technologies/3rd-gen-intel-xeon-scalable-processors-experience-kits>

**Table 3. Reference Documents**

REFERENCE	SOURCE
Advanced Networking Features in Kubernetes and Container Bare Metal Application Note	<a href="https://builders.intel.com/docs/networkbuilders/adv-network-features-in-kubernetes-app-note.pdf">https://builders.intel.com/docs/networkbuilders/adv-network-features-in-kubernetes-app-note.pdf</a>
CPU Management - CPU Pinning and Isolation in Kubernetes Technology Guide	<a href="https://builders.intel.com/docs/networkbuilders/cpu-pin-and-isolation-in-kubernetes-app-note.pdf">https://builders.intel.com/docs/networkbuilders/cpu-pin-and-isolation-in-kubernetes-app-note.pdf</a>
Enhanced Utilization Telemetry for Polling Workloads with collectd and the Data Plane Development Kit (DPDK) User Guide	<a href="https://networkbuilders.intel.com/solutionslibrary/enhanced-utilization-telemetry-for-polling-workloads-with-collectd-and-the-data-plane-development-kit-dpdk-user-guide">https://networkbuilders.intel.com/solutionslibrary/enhanced-utilization-telemetry-for-polling-workloads-with-collectd-and-the-data-plane-development-kit-dpdk-user-guide</a>
Intel Device Plugins for Kubernetes Application Note	<a href="https://builders.intel.com/docs/networkbuilders/intel-device-plugins-for-kubernetes-appnote.pdf">https://builders.intel.com/docs/networkbuilders/intel-device-plugins-for-kubernetes-appnote.pdf</a>
Intel® Ethernet Controller 700 Series - Dynamic Device Personalization Support for CNF with Kubernetes Technology Guide	<a href="https://builders.intel.com/docs/networkbuilders/intel-ethernet-controller-700-series-dynamic-device-personalization-support-for-cnf-with-kubernetes-technology-guide.pdf">https://builders.intel.com/docs/networkbuilders/intel-ethernet-controller-700-series-dynamic-device-personalization-support-for-cnf-with-kubernetes-technology-guide.pdf</a>
Intel® Ethernet Controller 700 Series GTPv1 - Dynamic Device Personalization Application Note	<a href="https://networkbuilders.intel.com/solutionslibrary/intel-ethernet-controller-700-series-gtpv1-dynamic-device-personalization">https://networkbuilders.intel.com/solutionslibrary/intel-ethernet-controller-700-series-gtpv1-dynamic-device-personalization</a>
Intel® Ethernet Controller E810 Dynamic Device Personalization Package (DDP) for Telecommunications Technology Guide	<a href="https://www.intel.com/content/www/us/en/design/products-and-solutions/networking-and-io/ethernet-controller-e810/technical-library.html">https://www.intel.com/content/www/us/en/design/products-and-solutions/networking-and-io/ethernet-controller-e810/technical-library.html</a>
Intel® Speed Select Technology – Base Frequency - Enhancing Performance Application Note	<a href="https://builders.intel.com/docs/networkbuilders/intel-speed-select-technology-base-frequency-enhancing-performance.pdf">https://builders.intel.com/docs/networkbuilders/intel-speed-select-technology-base-frequency-enhancing-performance.pdf</a>
Node Feature Discovery Application Note	<a href="https://builders.intel.com/docs/networkbuilders/node-feature-discovery-application-note.pdf">https://builders.intel.com/docs/networkbuilders/node-feature-discovery-application-note.pdf</a>
QCT Validated Kubernetes Platform with Enhanced Platform Awareness Technical Brief	<a href="https://networkbuilders.intel.com/solutionslibrary/qct-validated-kubernetes-platform-with-enhanced-platform-awareness">https://networkbuilders.intel.com/solutionslibrary/qct-validated-kubernetes-platform-with-enhanced-platform-awareness</a>
Telemetry Aware Scheduling - Automated Workload Optimization with Kubernetes (K8s) Technology Guide	<a href="https://builders.intel.com/docs/networkbuilders/telemetry-aware-scheduling-automated-workload-optimization-with-kubernetes-k8s-technology-guide.pdf">https://builders.intel.com/docs/networkbuilders/telemetry-aware-scheduling-automated-workload-optimization-with-kubernetes-k8s-technology-guide.pdf</a>
Topology Management - Implementation in Kubernetes Technology Guide	<a href="https://builders.intel.com/docs/networkbuilders/topology-management-implementation-in-kubernetes-technology-guide.pdf">https://builders.intel.com/docs/networkbuilders/topology-management-implementation-in-kubernetes-technology-guide.pdf</a>
Intel® Speed Select Technology – Base Frequency (Intel® SST-BF) with Kubernetes Application Note	<a href="https://networkbuilders.intel.com/solutionslibrary/intel-speed-select-technology-base-frequency-with-kubernetes-application-note">https://networkbuilders.intel.com/solutionslibrary/intel-speed-select-technology-base-frequency-with-kubernetes-application-note</a>
Secure the Network Infrastructure - Secure Cloud Native Network Platforms User Guide	<a href="https://networkbuilders.intel.com/solutionslibrary/secure-the-network-infrastructure-secure-cloud-native-network-platforms-user-guide">https://networkbuilders.intel.com/solutionslibrary/secure-the-network-infrastructure-secure-cloud-native-network-platforms-user-guide</a>

REFERENCE	SOURCE
Closed Loop Automation - Telemetry Aware Scheduler for Service Healing and Platform Resilience Demo	<a href="https://networkbuilders.intel.com/closed-loop-automation-telemetry-aware-scheduler-for-service-healing-and-platform-resilience-demo">https://networkbuilders.intel.com/closed-loop-automation-telemetry-aware-scheduler-for-service-healing-and-platform-resilience-demo</a>
Closed Loop Automation - Telemetry Aware Scheduler for Service Healing and Platform Resilience White Paper	<a href="https://builders.intel.com/docs/networkbuilders/closed-loop-platform-automation-service-healing-and-platform-resilience.pdf">https://builders.intel.com/docs/networkbuilders/closed-loop-platform-automation-service-healing-and-platform-resilience.pdf</a>
Intel® RDT	<a href="https://wiki.opnfv.org/display/fastpath/Intel_RDT">https://wiki.opnfv.org/display/fastpath/Intel_RDT</a>
Intel® Server GPU Features	<a href="https://www.intel.com/content/www/us/en/products/discrete-gpus/server-graphics-card.html">https://www.intel.com/content/www/us/en/products/discrete-gpus/server-graphics-card.html</a>
Intel® Server GPU for High-Density Cloud Gaming and Media Streaming Performance Summary	<a href="https://www.intel.com/content/www/us/en/benchmarks/server/graphics/intelservergpu.html">https://www.intel.com/content/www/us/en/benchmarks/server/graphics/intelservergpu.html</a>
Intel® Server GPU Data Center Graphics for High-Density Cloud Gaming and Media Streaming	<a href="https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/server-graphics-card-product-brief.html">https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/server-graphics-card-product-brief.html</a>
Intel® Server GPUs for Cloud Gaming and Media Delivery	<a href="https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/server-graphics-solution-brief.html">https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/server-graphics-solution-brief.html</a>
Intel® Server GPU Specifications	<a href="https://ark.intel.com/content/www/us/en/ark/products/210576/intel-server-gpu.html">https://ark.intel.com/content/www/us/en/ark/products/210576/intel-server-gpu.html</a>
Intel® AVX-512 - Packet Processing with Intel® AVX-512 Instruction Set Solution Brief	<a href="https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-packet-processing-with-intel-avx-512-instruction-set-solution-brief">https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-packet-processing-with-intel-avx-512-instruction-set-solution-brief</a>
Intel® AVX-512 - Instruction Set for Packet Processing Technology Guide	<a href="https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-instruction-set-for-packet-processing-technology-guide">https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-instruction-set-for-packet-processing-technology-guide</a>
Intel® AVX-512 - Writing Packet Processing Software with Intel® AVX-512 Instruction Set Technology Guide	<a href="https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-writing-packet-processing-software-with-intel-avx-512-instruction-set-technology-guide">https://networkbuilders.intel.com/solutionslibrary/intel-avx-512-writing-packet-processing-software-with-intel-avx-512-instruction-set-technology-guide</a>

## 2 Reference Architecture Overview

This chapter provides an overview of the Reference Architecture itself, including high-level descriptions of the Configuration Profiles, use cases, hardware components, software capabilities, and Ansible playbooks.

Review [Section 1.1](#), which introduces key concepts and terminology used in this guide.

### 2.1 Architecture Delivered

The Container Bare Metal Reference Architecture (BMRA) is an open Kubernetes cluster architecture designed for the convergence of key applications and services, control plane, and high-performance packet processing functions. It enables adoption of Intel platforms (starting with Early Access hardware platforms) and open-source platform software capabilities.

This complete, flexible, and scalable solution template allows deployment of Kubernetes clusters that can be based on multiple worker nodes managed by one or more Kubernetes control nodes. All servers are connected with one or two switches that provide connectivity within the cluster and to the cloud. The Ansible playbook allows auto-provisioning and auto-configuration of the BMRA and can be installed on any connected host server. In addition, this Reference Architecture uses testpmd and pktgen pods and sample CNF workloads (vCMTS and vBNG) for integration validation.

[Figure 1](#) below provides an example of a typical Kubernetes cluster composed of three control nodes and two worker nodes.

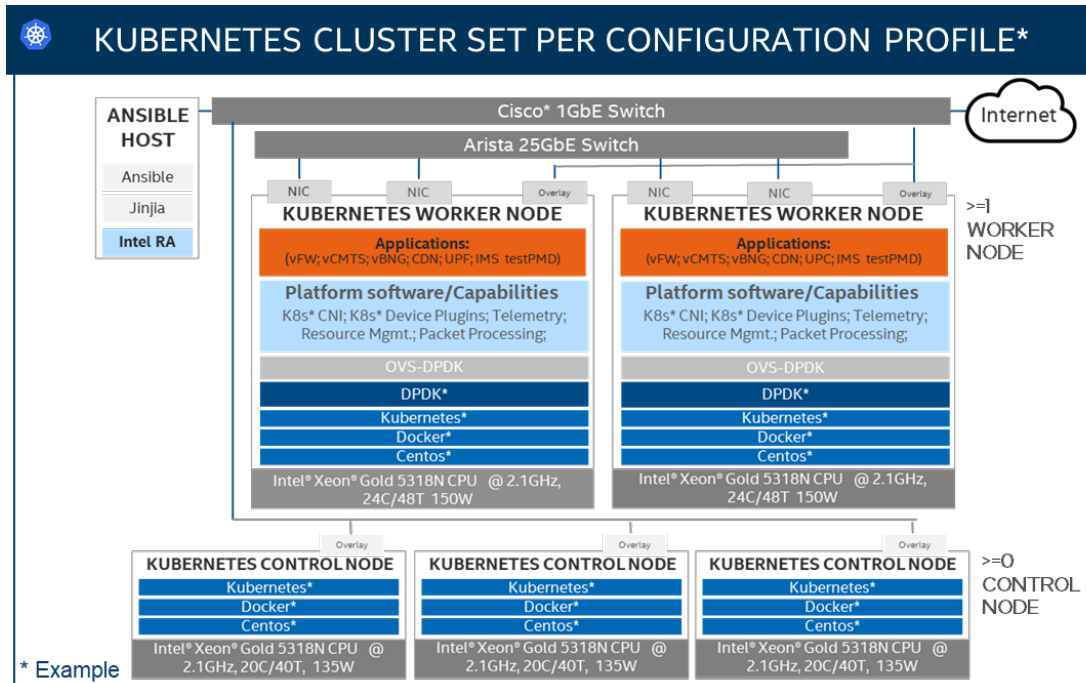


Figure 1. Example of Container Bare Metal Reference Architecture Kubernetes Cluster Setup<sup>1</sup>

The main elements forming the reference architecture are:

- **Hardware Components:** Multiple platform hardware options are available, including a variety of 2nd Generation Intel® Xeon® Scalable Processor SKUs, 3rd Generation Intel® Xeon® Scalable Processor SKUs, and Intel® Ethernet Network Adapters. For details see [Section 5, Reference Architecture Hardware Components and BIOS](#). BIOS options are provided, and you are expected to select and deploy the most optimal BIOS values before the cluster provisioning. For details, see [Section 5.5, Platform BIOS](#).
- **Software Capabilities:** This container environment uses a Docker containers runtime. The software capabilities are based on open-source software enabled in communities such as DPDK, FD.io, OVS, Kubernetes, and through Intel GitHub. Three Linux-based operating system options are available: CentOS, RHEL, and Ubuntu. For details, see [Section 6, Reference Architecture Software Components](#).
- **Configuration:** Specific hardware and software configurations are provided based on Intel assessment and verification. The hardware configuration addresses two modes of operation: “base” and “plus” (for high performance).
- **Installation Playbook:** Ansible playbooks implement the best practice configuration setup per each BMRA Flavor. This guide provides great details about the hardware and software configuration options available, however, the Ansible playbooks are ready to use, automate, and shorten the setup of your BMRA Flavor of choice.

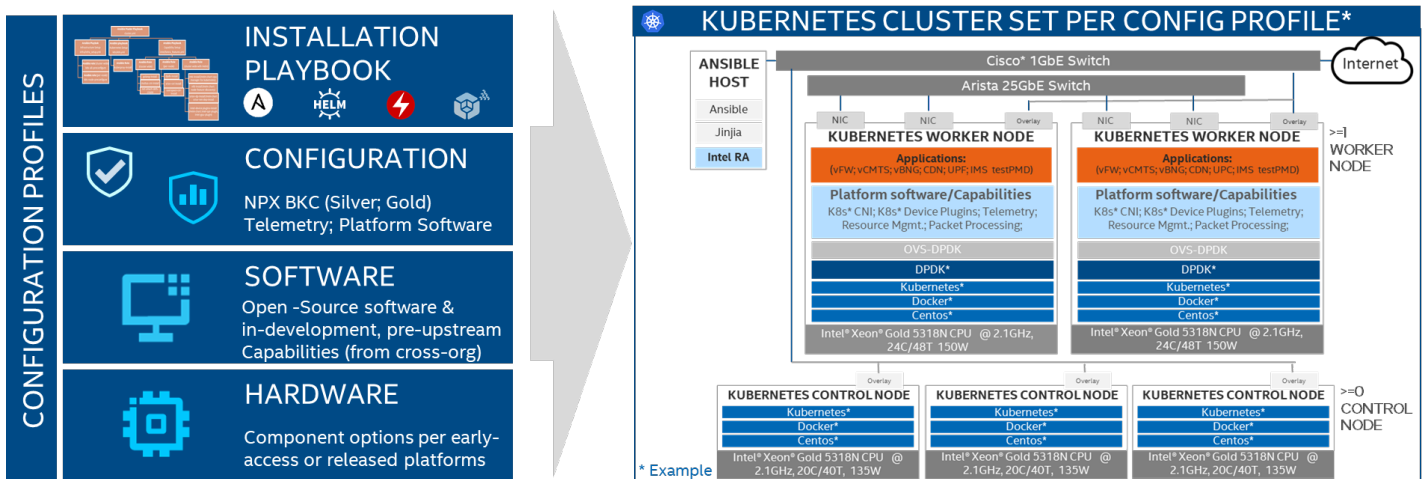


Figure 2. Key Components forming the Container Bare Metal Reference Architecture Kubernetes Cluster

<sup>1</sup> 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](http://www.Intel.com/PerformanceIndex). Results may vary.

## 2.2 Reference Architecture Configuration Profiles

Deployment requirements across the network are different depending on the location in the network and on the typical applications supported at each location. Each location can be characterized by the sites' density, distance from the user, and performance (latency and throughput). We identified the following main “network locations” as shown in [Figure 3: On-Premises Edge](#) (also called Customer Premise), **Central Office** (also called Near Edge), **Regional Data Center** (also called Regional POP), and **Telco Data Center Core Network**.

The BMRA introduces the concept of **Reference Architecture Configuration Profiles**. Multiple Configuration Profiles are assigned to the Reference Architecture in order to address the specific configuration, performance, and functionality required per workload type and network location. The following Configuration Profiles are supported and are shown in [Figure 4](#).

- **On-Premises Edge Configuration Profile:** This Configuration Profile recommends a K8s 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. Refer to [Appendix D, BMRA On-Premises Edge Configuration Profile Setup](#) for details.
- **Remote CO-Forwarding Configuration Profile:** This Configuration Profile addresses a K8s cluster hardware, software capabilities, and configurations that enable high performance for packet forwarding packets. In this category, we can find workloads such as UPF, vBNG, vCMTS, vCDN, FW, and more. The corresponding Configuration Profile is described in [Appendix E, BMRA Remote CO-Forwarding Configuration Profile Setup](#).
- **Regional Data Center Configuration Profile** – This Configuration Profile is tailored exclusively and defined for Media Visual Processing workloads such as CDN Transcoding. The corresponding Configuration Profile is described in [Appendix F BMRA Regional Data Center Configuration Profile Setup](#).

In addition, we have identified the following Configuration Profiles that are not specific to network location or workload:

- **Basic Configuration Profile:** Minimum hardware and software capabilities required for supporting a BMRA Kubernetes cluster setup, described in [Appendix B, BMRA Basic Configuration Profile Setup](#).
- **Full Configuration Profile:** Complete hardware and software capabilities set offered in BMRA, described in [Appendix C, BMRA Full Configuration Profile Setup](#).

Key elements of each Reference Architecture Configuration Profile are:

1. Example Use Cases: Scenarios that support the specific workloads enabled by the Configuration Profile.
2. Installation Playbook: An Ansible script enabling the deployment of all hardware and software components with optimized configurations.
3. Hardware Components ([Section 5, Reference Architecture Hardware Components and BIOS](#)): a set of optimized hardware elements for the location. For example, to support UPF workload at the Remote CO, the BMRA is populated with the maximum available network interface cards (NICs).
4. Software Capabilities ([Section 6, Reference Architecture Software Components](#)): The software components and the configuration per feature.

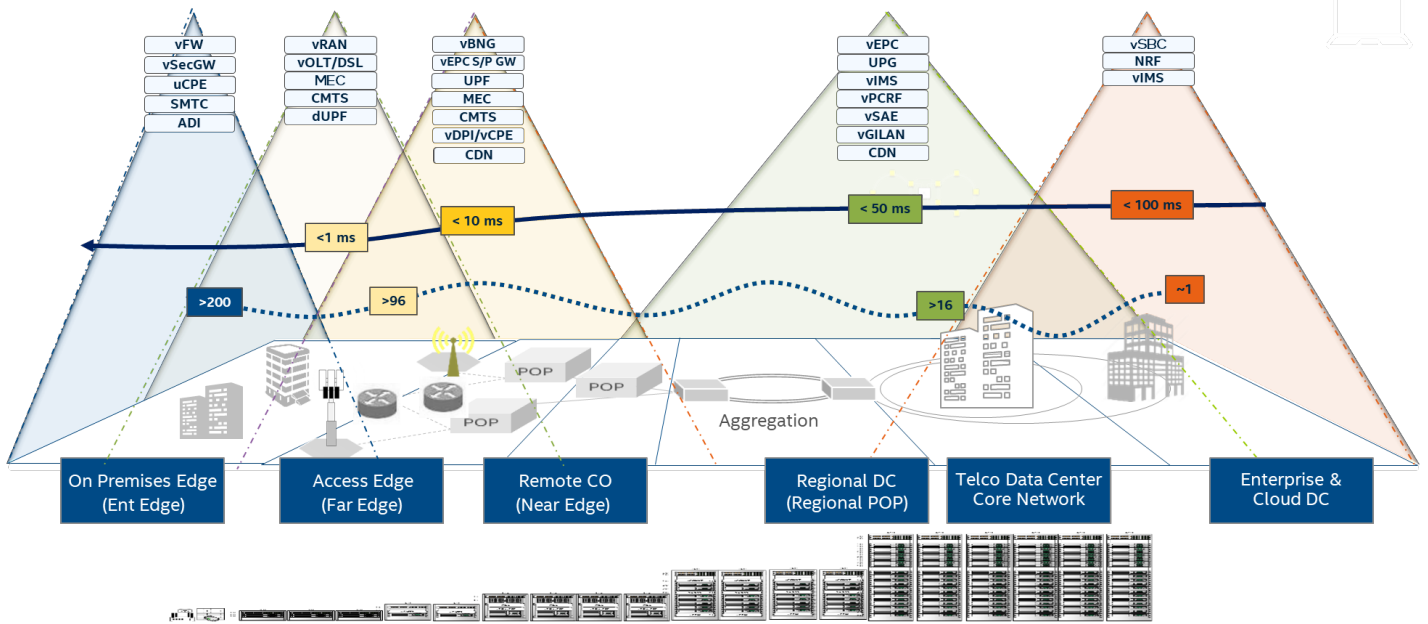


Figure 3. Identified Key Network Locations

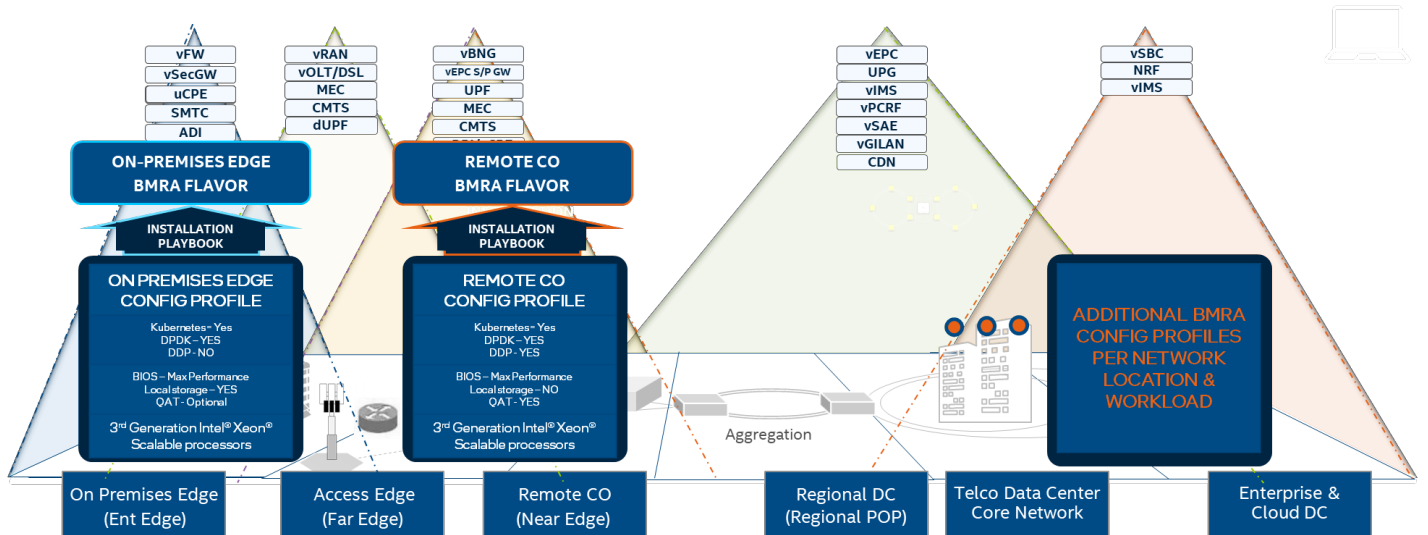


Figure 4. Example of Container Bare Metal Configuration Profiles Per Location

### 2.3 Use Cases by Network Location

This section explains the corresponding use cases per network location (see Figure 3). BMRA release 21.03 supports three of the Network Locations shown in Figure 3 (see Section 2.2).

#### On-Premises Edge (Enterprise Edge)

Small cluster of stationary or mobile server platforms, ranging between one and 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.

- Optimized data processing (to gain insights and reduce upstream transmission volumes) is priority.
- Power efficiency is important in most use cases.
- Automated infrastructure orchestration capabilities are mandatory.

### Remote Central Office (Near Edge)

The near edge consists of 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 (e.g., real-time gaming, stock trading, video conferencing).

- Power efficiency is a requirement.
- Resource usage efficiency and sharing are a priority.
- Scaling and redundancy/reliability through scale-out; performance is defined at the cluster level.
- Multitenancy support may be required.
- Operational automation is important (site is unmanned).

### Regional Data Center (Regional POP)

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.

- Automation is mandatory due to scale.
- High connectivity (in the aggregate- not individual data plane performance) is important.
- Scaling and redundancy/reliability through scale-out.
- Multitenancy support may be required.

### Telco Data Center Core Network

The Telco Data Center Core Network shares many of the characteristics and usage scenarios of the CO Regional Cloud, although at an even larger scale. The core network runs many of the same workloads as the CO Regional Cloud, with the addition of the centralized corporate services (e.g., finance, OSS/BSS, multi-cloud service orchestration).

## 2.4 Configuration Profile Installation Playbooks

Intel provides a set of Ansible scripts and Helm charts that enable easy and fast automatic installation<sup>2</sup> on a container bare metal NFV platform. Ansible is an agentless configuration management tool that uses playbooks to perform actions on many machines and Helm is a package manager tool that runs on top of Kubernetes to automate the installation process of plugins and K8s capabilities. BMRA Ansible playbooks enable users to customize multiple parameters to fit their installation requirements.

The BMRA Ansible playbooks take into consideration the Intel BKC (Best Known Configuration) for optimized performance.

Installation is done using one of the top-level BMRA Ansible playbooks and includes three phases:

1. Infrastructure setup, which addresses the initial system configuration, including telemetry setup.
2. Kubernetes setup, which deploys Kubernetes capabilities and its add-ons via Kubespray.
3. Installation and configuration of the system capabilities.

The following figure shows an overview of the BMRA Ansible playbooks. For more details, refer to [Section 3](#). In addition, each Configuration Profile has its own playbook and set of scripts, which are described in the Configuration Profile-specific Appendixes.

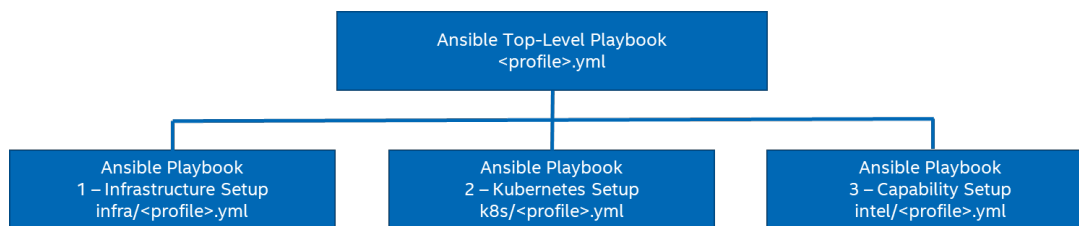


Figure 5. Ansible Playbooks Overview

## 2.5 Hardware Components

The BMRA supports a range of hardware that enables the different deployment models as explained earlier. Multiple platform hardware options are available, including a variety of 2nd Generation Intel® Xeon® Scalable Processor SKUs, 3rd Generation Intel® Xeon® Scalable Processor SKUs, Intel® Ethernet Network Adapters, Intel® QAT, and Intel® Server GPU. For details, see [Section 5, Reference Architecture Hardware Components and BIOS](#).

<sup>2</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

## 2.6 Software Capabilities

The technology described in this document consists of capabilities implemented across the software stack, which enables a Kubernetes environment that uses Intel technologies. It targets intelligent platform capability, configuration, and capacity data consumption. Intel and partners have worked together to advance the discovery, scheduling, and isolation of server hardware features using the following technologies.

### 2.6.1 Kubernetes Features

Containers are increasingly important in cloud computing and fundamental to cloud native adoption. A container is lightweight, agile, and portable, and it can be quickly created, updated, and removed. Kubernetes (K8s) is the leading open-source system for automating deployment, scaling, and management of containerized applications. To enhance Kubernetes for network functions virtualization (NFV) and networking usage, Intel and its partners are developing a suite of capabilities and methodologies that exposes Intel® Architecture platform features for increased and deterministic application and network performance.

- **Multus** enables support of multiple network interfaces per pod to expand the networking capability of Kubernetes. Supporting multiple network interfaces is a key requirement for many virtual network functions (VNFs), as they require separation of control, management, and data planes. Multiple network interfaces are also used to support different protocols or software stacks and different tuning and configuration requirements.
- **Node Feature Discovery (NFD)** enables generic hardware capability discovery in Kubernetes, including Intel® Xeon® processor-based hardware, for example.
- **Bond CNI** allows for aggregating multiple network interfaces into one logical interface.
- **Telemetry Aware Scheduling (TAS)** is a Kubernetes add-on that consumes cluster metrics and makes intelligent policy-based decisions. TAS enables automated actions driven by informed decisions based on up-to-date platform telemetry.
- **Native CPU Manager and Intel CPU Manager for Kubernetes** provide mechanisms for CPU core pinning and isolation of containerized workloads.
- **Topology Manager**, a native component of Kubernetes v1.16, enables other Kubelet components to make resource allocation decisions using topology-based information.
- **Hugepages** support, added to Kubernetes v1.8, enables the discovery, scheduling, and allocation of hugepages as a native first-class resource. This support addresses low latency and deterministic memory access requirements.
- **Device plugins**, including Intel® QuickAssist Technology and SR-IOV device plugins, boost performance and platform efficiency.

### 2.6.2 Platform System Features

In conjunction with the Kubernetes capabilities mentioned above, Intel is constantly developing new Intel® Architecture platform-level system capabilities for enhanced and deterministic application and network performance.

- **Dynamic Device Personalization (DDP)** is one of the key technologies of the Intel® Ethernet 700 and 800 Series. It enables workload-specific optimizations using the programmable packet processing pipeline to support a broader range of traffic types.
- **Single Root Input/Output Virtualization (SR-IOV)** provides I/O virtualization that makes a single PCIe device (typically a NIC) appear as many network devices in the Linux kernel. In Kubernetes, this results in network connections that can be separately managed and assigned to different pods.
- **Intel® Advanced Vector Extensions (Intel® AVX-512)** promotes 512-bit SIMD instruction extensions. They include extensions of the Intel AVX family of SIMD instructions but are encoded using a new encoding scheme. This scheme supports 512-bit vector registers (up to 32 vector registers in 64-bit mode) and conditional processing using opmask registers.
- **Intel® Speed Select Technology – Core Power (Intel® SST-CP)** allows OS/VMM to control which cores can take advantage of any power headroom that's available in the system to enable greater system performance.
- **Intel® Speed Select Technology – Base Frequency (Intel® SST-BF)** offers a deterministic higher-frequency pool of processing cores to execute high priority workloads and a pool of cores running at lower frequency for non-critical workloads.
- **Secure Cloud Native Network Platforms** – Using Intel® Security Libraries for Data Center (Intel® SecL – DC), BMRA can help you to build end-to-end platform security. Intel® SecL-DC integrates platform attestation into the cloud native architecture and uses Kubernetes to orchestrate and run workloads only on trusted pods.
- **Key Management Reference Application with Intel® Software Guard Extensions (SGX)** demonstrates the integration of the asymmetric key capability of Intel® SGX with a third-party hardware security model (HSM) on a centralized key server.
- **Media** adds support for media processing on the new Intel® Server GPU card. The Intel® Server GPU is the first discrete graphics processing unit for data centers based on the new Intel X<sup>e</sup> architecture, with support for MPEG-2, AVC, HEVC, and VP9 transcoding plus AV1 decode.

### 2.6.3 Observability

This solution deploys a broad range of telemetry collectors. Platform collectors capture metrics from CPU performance monitoring, which includes Intel hardware features such as RAS, power, PMU, RDT, NVME storage, network interfaces, disks, platform power, memory, and thermal. In addition to hardware telemetry, BMRA captures DPDK and OVS metrics.

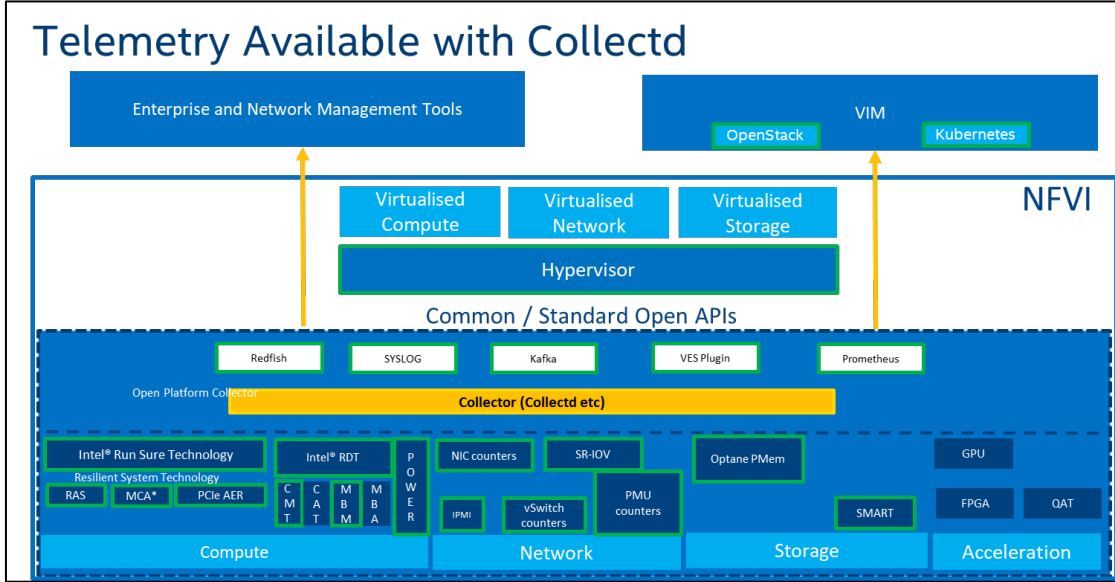


Figure 6. Platform Telemetry Available with Collectd<sup>3</sup>

Metrics can be published to Prometheus and other interfaces such as SNMP, Kafka, and VES.

The telemetry software stack consists of the following components:

- Collectd – UNIX daemon that collects a wide range of platform telemetry, including RDT and PMU metrics.
- Node Exporter – platform telemetry monitoring agent.
- Prometheus – telemetry monitoring system with time series database.
- Grafana – telemetry visualization application.
- Prometheus Adapter - implementation of the Kubernetes resource metrics API and custom metrics API. Enables Telemetry Aware Scheduler.
- Telemetry Aware Scheduler - makes telemetry data available to scheduling and descheduling decisions in Kubernetes.

<sup>3</sup> 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](http://www.Intel.com/PerformanceIndex). Results may vary.

**Part 2:**  
**Reference Architecture Deployment:**  
**Ansible Playbooks**  
**Common Hardware Components**  
**Software Ingredients**  
**Recommended Configurations**

## 3 Reference Architecture Deployment – Ansible Playbooks

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. The chapter contains the following sections:

- [Reference Architecture Installation Prerequisites](#)
- [Ansible Playbook Review](#), including description of building blocks and phases
- [Deployment using Ansible Playbook](#)

**Note:** Ansible playbooks for 2nd Generation Intel Xeon Scalable processors and 3rd Generation Intel Xeon Scalable processors are open source and available [here](#).

### 3.1 Reference Architecture Installation Prerequisites

Before the Ansible playbook can begin, you must identify the required hardware components, 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 - list of necessary network connections between the nodes
- Installation of software dependencies needed to execute Ansible playbooks
- Generation and distribution of SSH keys that will be used for authentication between Ansible host and Kubernetes cluster target servers

After satisfying these prerequisites, Ansible playbooks for 2nd Generation Intel Xeon Scalable processors and 3rd Generation Intel Xeon Scalable processors can be downloaded directly from the dedicated GitHub page (<https://github.com/intel/container-experience-kits/releases>) or cloned using the Git. Command-line tool and executed after applying desired the configuration as described in [Section 3](#) in this document.

#### 3.1.1 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 Node(s) - Review [Section 5.1](#) for recommended Control node assembly.
- Worker Node(s) – Refer to the following sections for recommended Worker node assembly:
  - Base Worker Node – Review [Section 5.2](#) to satisfy base performance characteristics.
  - Plus Worker Node – Review [Section 5.3](#) to satisfy plus performance characteristics.

[Appendix B.1](#), [Appendix C.1](#), [Appendix D.1](#), [Appendix E.1](#), and [Appendix F.1](#) contain details about the hardware BOM selection and setup.

#### 3.1.2 BIOS Selection for Control and Worker Nodes

Enter the UEFI or BIOS menu and update the configuration as listed in [Section A.3](#) and [Table 17](#), which describe the BIOS selection in detail.

#### 3.1.3 Operating System (OS) Selection for Control and Worker Nodes

The following Linux Operating Systems are supported for the Control and Worker Nodes:

- CentOS Linux Version 8 (8.3)
- CentOS Linux Version 7 (7.9)
- RHEL for x86\_64 Version 8 (8.3)
- Ubuntu 20.04 LTS (20.04.1)
- Ubuntu 18.04 LTS (18.04.5)

For all supported distributions, the base images are sufficient to be built using the "Minimal" option during installation. In addition, the following shall 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
```

### 3.1.4 Network Interface Requirements for Control and Worker Nodes

The following list provides a brief description of different networks and network interfaces needed for deployment. (see [Figure 1](#))

- 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 flannel pod network interface (This can be a shared interface with the internet network)
  - Kubernetes control and worker node inter-node communications
  - Flannel pod network will run over this network
  - Configured to use a private static address
- Tenant data network(s)
  - Dedicated networks for traffic
  - SR-IOV enabled
  - VF can be DPDK bound in pod

### 3.1.5 Software Prerequisites for Ansible Host, Control Nodes, and Worker Nodes

Before deployment of the BMRA Ansible playbooks, the Ansible Host must be prepared. In order to successfully run the deployment, perform the following tasks before you download the BMRA Ansible code.

Perform the following steps:

1. Login to the Ansible host machine using SSH or your preferred method to access the shell on that machine.
2. Install packages on Ansible Host. The following example assumes that the host is running CentOS 8.2. Other operating systems may have slightly different installation steps:

```
$ yum install python3
$ pip3 install ansible==2.9.17
$ pip3 install jinja2 --upgrade
$ yum install libselinux-python3
```

3. Enable passwordless login between all nodes in the cluster.
  - a. Create Authentication SSH-Keygen Keys on Ansible Host:
 

```
$ ssh-keygen
```
  - b. Upload Generated Public Keys to all the nodes from Ansible Host:
 

```
$ ssh-copy-id root@<target_server_address>
```

## 3.2 Ansible Playbook Review

The reference architecture is configured and provisioned automatically using Ansible scripts<sup>4</sup>. 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 3.3](#).

### 3.2.1 Ansible Playbooks Building Blocks

The following components make up the BMRA Ansible playbooks.

**Configuration Files** provide examples of cluster-wide and host-specific configuration options for each of the Configuration Profiles. With minimal changes they can be used directly with their corresponding playbooks. For default values refer to the Configuration Profile-specific sections for configuration options for your desired Configuration Profile: : [Appendix B BMRA Basic Configuration Profile Setup](#), [Appendix C BMRA Full Configuration Profile Setup](#), [Appendix D BMRA On-Premises Edge Configuration Profile Setup](#), [Appendix E BMRA Remote CO-Forwarding Configuration Profile Setup](#), [Appendix F BMRA Regional Data Center Configuration Profile Setup](#).

- inventory.ini
- group\_vars
- host\_vars

**Ansible Playbooks** act as a user entry point and include all relevant Ansible roles and Helm charts. There are four top-level Ansible playbooks available, one per each Configuration Profile, which allows for 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 7](#).

- basic.yml
- full\_nfv.yml

<sup>4</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

- on\_prem.yml
- remote\_fp.yml
- regional\_dc.yml

Each of these playbooks encompasses **Ansible Roles** grouped into three main execution phases, which are depicted in [Figure 7](#) and further explained in the next section:

- Infrastructure Setup
- Kubernetes Deployment
- Capabilities Setup

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.

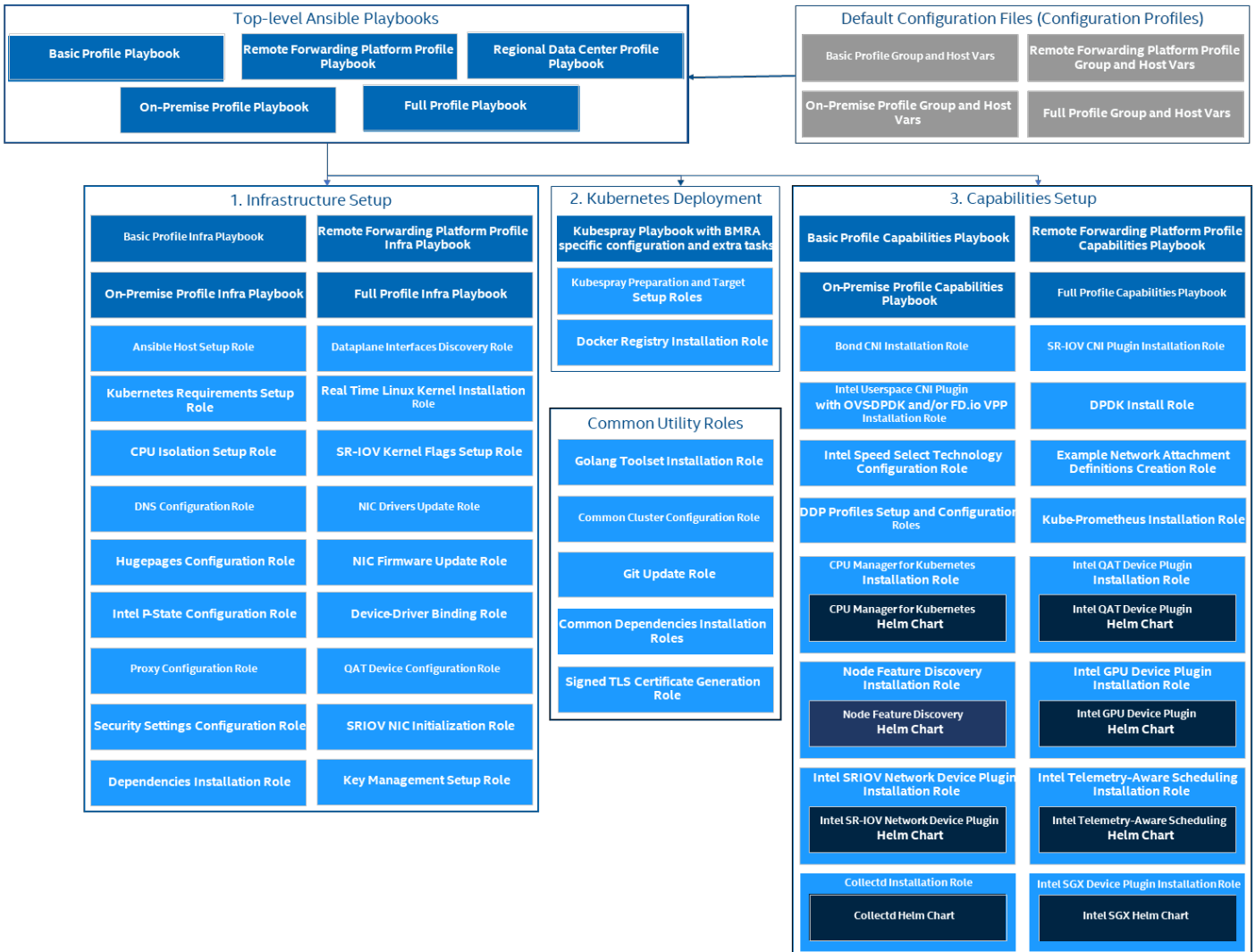


Figure 7. High Level BMRA Ansible Playbooks Architecture<sup>5</sup>

### 3.2.2 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.

<sup>5</sup> Refer to <https://software.intel.com/articles/optimization-notice> for more information regarding performance and optimization choices in Intel software products.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

- 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 ensure that all latest capabilities such as DDP profiles are enabled.
  - Intel® Speed Select Technology configuration, which provides control over base frequency.
  - Installation of Dynamic Device Personalization profiles, which can increase packet throughput, 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) K8s cluster using Kubespray, which is a project under the Kubernetes community that deploys production-ready Kubernetes clusters. The Multus CNI plugin, which is specifically designed to provide support for multiple networking interfaces in a K8s environment, is deployed by Kubespray along with Flannel and Helm. Preferred security practices are used in the default configuration. On top of Kubespray, there's also a Docker registry instance deployed to store Docker images of various control-plane Kubernetes applications such as TAS, CMK, or Device Plugins.
3. **BMRA System Capabilities Setup** (sub-playbooks located in playbooks/intel directory):  
Advanced networking technologies, Enhanced Platform Awareness, and device plugin features are deployed by this playbook using Helm Charts as part of the BM RA. The following capabilities are deployed:
- Device Plugins that allow using 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.
  - CPU Manager for Kubernetes, 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 K8s 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.

### 3.3 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.

#### 3.3.1 Prepare Target Servers

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

- Python is installed. The version depends on the target distribution. Python 3 is highly recommended, but Python 2 is still supported for CentOS 7.
- SSH keys are exchanged between the Ansible host and each target node. The same SSH keys need to be configured on each of the machines. You can achieve that by executing below command on the Ansible host to copy to each target server in the cluster:

```
ssh-copy-id root@<target_server_address>
```

- Internet access on all target servers is mandatory. Proxies are supported and can be configured in the Ansible vars.
- Ensure that BIOS configuration matching the desired state is applied. For details, refer to the specific Configuration Profile Appendix for your profile: [Appendix B BMRA Basic Configuration Profile Setup](#), [Appendix C BMRA Full Configuration Profile Setup](#), [Appendix D BMRA On-Premises Edge Configuration Profile Setup](#), [Appendix E BMRA Remote CO-Forwarding Configuration Profile Setup](#), [Appendix F BMRA Regional Data Center Configuration Profile Setup](#).

For detailed steps on how to build the Ansible host, refer to [Appendix G](#).

#### 3.3.2 Get Ansible Playbook and Prepare Configuration Templates

Perform the following steps:

1. Login to your Ansible host (the one that you will run these Ansible playbooks from).
2. 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 19](#), for example:

```
git checkout v21.03
```

**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/v21.03.tar.gz  
tar xf v21.03.tar.gz  
cd container-experience-kits-21.03
```

- Decide which Configuration Profile you want to use and export the environmental variable.

For Kubernetes **Basic** Configuration Profile deployment:

```
export PROFILE=basic
```

For Kubernetes **Full** Configuration Profile deployment:

```
export PROFILE=full_nfv
```

For Kubernetes **On-Premises Edge** Configuration Profile deployment:

```
export PROFILE=on_prem
```

For Kubernetes **Remote Central Office-Packet Forwarding** Configuration Profile deployment:

```
export PROFILE=remote_fp
```

For Kubernetes **Regional Data Center** Configuration Profile deployment:

```
export PROFILE=regional_dc
```

- Copy example inventory file to the project root dir.

```
cp examples/${PROFILE}/inventory.ini .
```

- Copy example configuration files to the project root dir:

```
cp -r examples/${PROFILE}/group_vars examples/${PROFILE}/host_vars .
```

### 3.3.3 Update Ansible Inventory File

Perform the following steps:

- Edit the inventory.ini file copied in the previous steps.
- In the section [all], specify all your target servers. Use their actual hostnames and Management IP addresses. Also list your Ansible host as "localhost" with the Python version packaged with your OS distribution.

Example: There is an Ansible host and three servers - one controller and two worker nodes. Their hostnames are controller1, node1, and node2 respectively. They are all located in the 10.100.200.0/24 subnet and are assigned static IP addresses (10.100.200.10x). All three cluster servers have CentOS 8 installed with Python 3, and the Ansible host has CentOS 7 with Python 2. In this case the [all] section should be configured like this:

```
[all]
localhost ansible_python_interpreter=/usr/bin/python2
controller1 ansible_host=10.100.200.101 ip=10.100.200.101
node1      ansible_host=10.100.200.102 ip=10.100.200.102
node2      ansible_host=10.100.200.103 ip=10.100.200.103
...

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

Note: In this example, Python 3 is used as a default Python interpreter for all servers except for the Ansible host, which we configured to use Python 2 as it uses CentOS 7.

- Assign servers to Ansible groups.

Example: As mentioned above, we want to have one controller and two worker nodes. It's recommended to place "etcd" (cluster database) on the same node as the remaining control-plane components. So, in that case, the final inventory.ini file would look like this:

```
[all]
localhost ansible_python_interpreter=/usr/bin/python2
controller1 ansible_host=10.100.200.101 ip=10.100.200.101
node1      ansible_host=10.100.200.102 ip=10.100.200.102
node2      ansible_host=10.100.200.103 ip=10.100.200.103

[kube-master]
controller1

[etcd]
controller1

[kube-node]
node1
node2

[k8s-cluster:children]
kube-master
kube-node
```

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

### 3.3.4 Update Ansible Host and Group Variables

Perform the following steps:

1. Create `host_vars` files for all nodes, matching their hostnames from the inventory file.

Example: Following the setup from previous steps, we need a file for each of the nodes. We don't need a host vars file for the controller:

```
cp host_vars/node1.yml host_vars/node2.yml
```

2. Edit `host_vars/<node_name>.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 Appendix for your profile to get a full list of variables and their documentation: [Appendix B BMRA Basic Configuration Profile Setup](#), [Appendix C BMRA Full Configuration Profile Setup](#), [Appendix D BMRA On-Premises Edge Configuration Profile Setup](#), [Appendix E BMRA Remote CO-Forwarding Configuration Profile Setup](#), [Appendix F BMRA Regional Data Center Configuration Profile Setup](#).

### 3.3.5 Run Ansible 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 a CentOS/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 7, Post Deployment Verification Guidelines](#).

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

## 4 Software Capabilities Review

Intel, in collaboration with industry partners, continues to work to bring Intel® Architecture advancements to the cloud native ecosystem through support of **Kubernetes features**, extension of **Kubernetes plugins**, and development of **system hardware resources** as described below<sup>6</sup>.

### 4.1 Kubernetes Plugins

The following device plugins are used to advertise Intel® Architecture system hardware resources to Kubernetes. Several of the plugins are Container Network Interface (CNI), which is explained here: <https://github.com/containernetworking/cni>

#### 4.1.1 Multus CNI

Kubernetes natively supports only a single network interface. Multus is a CNI plugin specifically designed to provide support for multiple networking interfaces in a Kubernetes environment. Operationally, Multus behaves as a broker and arbiter of other CNI plugins, meaning it invokes other CNI plugins (such as Flannel, Calico, SR-IOV, or Userspace CNI) to do the actual work of creating the network interfaces. Multus v3.3 has recently been integrated with KubeVirt, officially recognized as a CNCF project, and officially released with Kubespray v2.12.

Supporting multiple network interfaces is a key requirement for many Network Functions (NFs), as they require separation of control, management, and data planes. Multiple network interfaces are also used to support different protocols or software stacks and different tuning and configuration requirements. A set of basic plugins is provided through [CNI Plugins](#) from the container networking team, and the below plugins, SR-IOV, Userspace and Bond CNIs, can be installed as part of the deployment using Ansible playbook.

<sup>6</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

For more details see: <https://github.com/intel/multus-cni>

### 4.1.2 SR-IOV Network Device Plugin

Single Root Input/Output Virtualization (SR-IOV) provides I/O virtualization that makes a single PCIe device (typically a network adapter) appear as many network devices, also known as virtual functions (VFs) in the Linux kernel. In Kubernetes, this results in network connections that can be separately managed and assigned to different pods.

The Intel SR-IOV Network device plugin discovers and exposes SR-IOV network resources as consumable extended resources in Kubernetes. It works with SR-IOV VFs with both kernel drivers and DPDK drivers. When a VF is attached with a kernel driver, then the SR-IOV CNI plugin can be used to configure this VF in the pod. When using the DPDK driver, a VNF application configures this VF as required.

The SR-IOV network device plugin provides a way to filter the available VFs and make them available as endpoints in Kubernetes. Using a list of selectors, a subset of VFs can be associated with a resource name, that can be used when assigning resources to pods. An example of a resource is shown below:

```
"resourceName": "intel_sriov_netdevice",
"selectors": {
  "vendors": ["8086"],
  "devices": ["154c", "10ed", "1889"],
  "drivers": ["iavf", "i40evf", "ixgbevf"]
}
```

For more details, see: <https://github.com/intel/sriov-network-device-plugin>

### 4.1.3 SR-IOV CNI

The Single Root I/O Virtualization SR-IOV device plugin enables partition of a single physical network adapter (PCI) resource into virtual PCI functions (VFs) that can be attached to Kubernetes pods. To attach an SR-IOV device resource to the Kubernetes POD network, we use the SR-IOV CNI. The SR-IOV CNI plugin enables the Kubernetes pod to be attached directly to an SR-IOV virtual function (VF) using the standard SR-IOV VF driver in the container host's kernel.

For more details, see: <https://github.com/intel/sriov-cni>

### 4.1.4 Userspace CNI

The Userspace CNI is a Container Network Interface (CNI) designed to implement userspace networking (as opposed to kernel space networking), such as DPDK-based applications. It is designed to run with either OVS-DPDK or VPP along with the Multus CNI plugin in Kubernetes deployments. Userspace CNI provides a high-performance container networking solution and Data Plane Acceleration for containers.

### 4.1.5 Bond CNI

Bond CNI allows for aggregation of multiple network interfaces into one logical interface. Interface bonding is used to provide additional network capacity and/or redundancy. When used as standalone plugin, interfaces are obtained from the host's network namespace. Bonded interface is created in the container network namespace. When used with Multus you can bond interfaces that were previously passed to the container.

### 4.1.6 Intel® QuickAssist Device Plugin

Intel® QuickAssist adapters integrate hardware acceleration of compute-intensive workloads such as bulk cryptography, public key exchange, and compression on Intel® Architecture platforms. The Intel® QAT device plugin for Kubernetes supports Intel® QuickAssist Adapters and includes an example scenario that uses the Data Plane Development Kit (DPDK) drivers.

For more details, see: <https://github.com/intel/intel-device-plugins-for-kubernetes>

### 4.1.7 Intel® Software Guard Extensions (SGX) Device Plugin

Intel® SGX device plugin allows Kubernetes workloads to use Intel® SGX on 3rd Generation Intel® Xeon® Scalable processors. The plugin is used in conjunction with an SGX Admission webhook and EPC memory registration to isolate specific application code and data in memory via enclaves that are designed to be protected from processes running at higher privilege levels. An additional remote attestation server is required to verify software is running inside an SGX enclave on a trusted computing node.

For more details, see: <https://github.com/intel/intel-device-plugins-for-kubernetes>

## 4.2 Kubernetes Features

Kubernetes (K8s) is the leading open-source orchestration platform for automating deployment, scaling, and management of containerized applications. To enhance Kubernetes for network functions virtualization (NFV) and networking usage, Intel and its partners are developing a suite of capabilities and methodologies that exposes Intel® Architecture platform features for increased

and deterministic application and network performance<sup>7</sup>. This section outlines K8s capabilities common to all profiles within the BMRA 21.03.

### 4.2.1 Node Feature Discovery

In a standard deployment, Kubernetes reveals very few details about the underlying platform to the user. This may be a good strategy for general data center use, but, in many cases a workload behavior or its performance may improve by leveraging the platform (hardware and/or software) features. Node Feature Discovery (NFD) is a Kubernetes add-on that detects and advertises hardware and software capabilities of a platform that can be used to facilitate intelligent scheduling of a workload. NFD currently detects following features:

- **CPUID:** NFD advertises CPU features such as Intel® AVX. Certain workloads, such as machine learning, may gain a significant performance improvement from these extensions.
- **SR-IOV networking:** NFD detects the presence of SR-IOV-enabled NICs, allowing optimized scheduling of network-intensive workloads.
- **Intel® RDT:** Intel® Resource Director Technology (Intel® RDT) allows visibility and control over the use of last-level cache (LLC) and memory bandwidth between co-running workloads. By allowing allocation and isolation of these shared resources, and thus reducing contention, RDT helps in mitigating the effects of noisy neighbors. NFD detects the different RDT technologies supported by the underlying hardware platform.
- **Intel® Turbo Boost Technology:** NFD detects the state of Intel® Turbo Boost Technology, allowing optimal scheduling of workloads that have a well-understood dependency on this technology.
- **IOMMU:** An input/output memory management unit (IOMMU), such as Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d) technology, allows isolation and restriction of device accesses. This enables direct hardware access in virtualized environments, highly accelerating I/O performance by removing the need for device emulation and bounce buffers.
- **SSD storage:** NFD detects the presence of non-rotational block storage on the node, making it possible to accelerate workloads requiring fast local disk access.
- **NUMA topology:** NFD detects the presence of NUMA topology, making it possible to optimize scheduling of applications based on their NUMA-awareness.
- **Linux kernel:** NFD detects the kernel version and advertises it through multiple labels, allowing the deployment of workloads with different granularity of kernel version dependency.
- **PCI:** NFD detects PCI devices, allowing optimized scheduling of workloads dependent on certain PCI devices.

NFD currently detects the features shown below.

CPU: The cpu feature source supports the following labels			Kernel: The kernel feature source supports the following labels			Memory: The memory feature source supports the following labels						
Feature name	Attribute	Description	Feature	Attribute	Description	Feature	Attribute	Description				
cpuid	<cpuidflag>	CPU capability is supported	config	<option name>	Kernel config option is enabled (set 'y' or 'm'). Default options are NO_HZ, NO_HZ_IDLE, NO_HZ_FULL and PREEMPT	numa		Multiple memory nodes i.e., NUMA architecture detected				
hardware_multithreading		Hardware multithreading, such as Intel HTT, enabled (number of logical CPUs is greater than physical CPUs)				nv	present	NVDIMM device(s) are present				
power	sst_bf.enabled	Intel SST-BF (Intel Speed Select Technology - Base frequency) enabled	selinux	enabled	Selinux is enabled on the node	nv	dax	NVDIMM region(s) configured in DAX mode are present				
pstate	status	The status of the Intel pstate driver when in use and enabled, either 'active' or 'passive'.	version	full	Full kernel version as reported by /proc/sys/kernel/osrelease (e.g., '4.5.6-7-g123abcde')	<b>IOMMU: The iommu feature source supports the following labels</b>						
turbo	turbo	Set to 'true' if turbo frequencies are enabled in Intel pstate driver, set to 'false' if they have been disabled.		major	First component of the kernel version (e.g., '4')	<table border="1"> <thead> <tr> <th>Feature name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>enabled</td> <td>IOMMU is present and enabled in the kernel</td> </tr> </tbody> </table>			Feature name	Description	enabled	IOMMU is present and enabled in the kernel
Feature name	Description											
enabled	IOMMU is present and enabled in the kernel											
scaling_governor	scaling_governor	The value of the Intel pstate scaling_governor when in use, either 'powersave' or 'performance'.		minor	Second component of the kernel version (e.g., '5')	<b>Storage: The storage feature source supports the following labels</b>						
cstate	enabled	Set to 'true' if cstates are set in the intel_idle driver, otherwise set to 'false'.		revision	Third component of the kernel version (e.g., '6')	<table border="1"> <thead> <tr> <th>Feature name</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>nonrotationaldisk</td> <td>Non-rotational disk, like SSD, is present in the node</td> </tr> </tbody> </table>			Feature name	Description	nonrotationaldisk	Non-rotational disk, like SSD, is present in the node
Feature name	Description											
nonrotationaldisk	Non-rotational disk, like SSD, is present in the node											
rdt	RDTMON	Intel RDT Monitoring Technology	<b>PCI: The pci feature source supports the following labels</b>			<b>System: The system feature source supports the following labels</b>						
	RDTMNT	Intel Cache Monitoring (CMT)	Feature	Attribute	Description	os_release	ID	Operating system identifier				
	RDTMBM	Intel Memory Bandwidth Monitoring (MBM)	<device label>	present	PCI device is detected		VERSION_ID	Operating system version identifier (e.g., '6.7')				
	RDTL3CA	Intel L3 Cache Allocation Technology	<device label>	sriov.capable	Single Root Input/Output Virtualization (SR-IOV) enabled PCI device present		VERSION_ID.major	First component of the OS version id (e.g., '6')				
	RDTL2CA	Intel L2 Cache Allocation Technology	<b>USB: The usb feature source supports the following labels</b>				VERSION_ID.minor	Second component of the OS version id (e.g., '7')				
	RDTMBA	Intel Memory Bandwidth Allocation (MBA) Technology	Feature	Attribute	Description	<b>Network: The network feature source supports the following labels</b>						
			<device label>	present	USB device is detected	Feature	Attribute	Description				
						sriov	capable	Single Root Input/Output Virtualization (SR-IOV) enabled Network Interface Card(s) present				
							configured	SR-IOV virtual functions have been configured				

Figure 8. Features Detected by NFD

<sup>7</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

For more details see: <https://github.com/kubernetes-sigs/node-feature-discovery>

### 4.2.2 Topology Manager

Today's systems use a combination of CPUs and hardware accelerators to support latency-critical execution and high-throughput parallel computation. To help extract the optimal performance out of such systems, the required optimizations related to CPU isolation, memory, and device locality must be made. In Kubernetes however, these optimizations are handled by a disjointed set of components. Topology Manager is a beta feature of Kubernetes in release 1.19.8. It is a kubelet component that coordinates the set of components that are responsible for making topology aligned resource allocations.

For details see: <https://builders.intel.com/docs/networkbuilders/topology-management-implementation-in-kubernetes-technology-guide.pdf>

### 4.2.3 Kubernetes Native CPU Manager

Native CPU Manager for Kubernetes provides mechanisms for allocation of CPU cores to workloads in situation where pods contend for resources of the CPU. When contention happens workloads may get moved to other CPUs and workload performance may be impacted. To avoid this, CPU manager offers an option to allocate exclusive cores to a workload (POD) by specifying "guaranteed QoS" and integer CPU requests.

See: <https://kubernetes.io/blog/2018/07/24/feature-highlight-cpu-manager/>

### 4.2.4 CPU Manager for Kubernetes (CMK)

Intel CPU Manager for Kubernetes provides mechanisms for CPU core pinning and isolation of containerized workloads. Intel CMK divides the CPUs on a system into three pools (by default) with an additional optional pool. The CPUs are divided into pools according to exclusivity: exclusive, shared, and infra. The CMK maintains lists of CPUs in each of the lists from which user containers can acquire CPUs. Intel CMK delivers predictable performance for high-priority applications due to reduced or eliminated thread preemption and maximizing CPU cache utilization. CMK provides a single multiuse command-line program to perform various functions for host configuration, managing groups of CPUs, and constraining workloads to specific CPUs.

See: <https://builders.intel.com/docs/networkbuilders/cpu-pin-and-isolation-in-kubernetes-app-note.pdf>

### 4.2.5 Telemetry Aware Scheduling

Telemetry Aware Scheduling (TAS) makes telemetry data available for scheduling and descheduling decisions in Kubernetes. Through a user-defined policy, TAS enables rule-based decisions on pod placement powered by up-to-date platform metrics. Policies can be applied on a workload by workload basis - allowing the right indicators to be used to place the right pod.

For example, a pod that requires certain cache characteristics can be scheduled based on output from Intel® RDT metrics. Likewise, a combination of RDT, RAS, and other platform metrics could be used to provide a signal for the overall health of a node and could be used to proactively ensure workload resiliency.

Telemetry Aware Scheduling is made up of two components deployed in a single pod on a Kubernetes cluster:

- **Telemetry Aware Scheduler Extender** is contacted by the generic Kubernetes scheduler every time it needs to make a scheduling decision on a pod calling for telemetry scheduling. The extender checks if there is a telemetry policy associated with the workload. If so, it inspects the strategies associated with the policy and returns opinions on pod placement to the generic scheduler. The scheduler extender has two strategies it acts on - `scheduleonmetric` and `dontschedule`. This is implemented and configured as a Kubernetes Scheduler Extender.
- **Telemetry Policy Controller** consumes TAS Policies - a custom resource. The controller parses these policies and places them in a cache to make them locally available to all TAS components. It consumes new telemetry policies as they are created, removes them when deleted, and updates them as they are changed. The policy controller also monitors the current state of policies to see if they are violated

TAS acts on three strategy types:

- **scheduleonmetric** has only one rule. It is consumed by the Telemetry Aware Scheduling Extender and prioritizes nodes based on a comparator and an up-to-date metric value. For example:  
`scheduleonmetric when health_metric is LessThan`
- **dontschedule** has multiple rules, each with a metric name and operator and a target. A pod with this policy is never scheduled on a node that breaks any one of the rules. For example:  
`dontschedule if health_metric Equals 1`
- **deschedule** is consumed by the Telemetry Policy Controller and can have multiple rules. If a pod is running on a node that violates this policy, it can be descheduled with the Kubernetes descheduler. For example:  
`deschedule if health_metric Equals 2`

TAS allows arbitrary, user-defined rules to be put in place in order to impact scheduling in a K8s cluster. Using the K8s descheduler, it can evict workloads that are breaking some rules in order to have it replaced on a more suitable node.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

In a modern cloud computing cluster, there is a torrent of data that only certain subject matter experts know how to interpret and act upon. In scheduling workloads, operators know on which compute nodes a workload may perform better based on up-to-date utilization metrics. Likewise, certain telemetry values, or combinations of values, can be recognized as signs that a node has some serious problem that is interfering with workload operation. The TAS policy system allows these insights to influence the scheduling and lifecycle placement process – turning implicit personal knowledge into formal, actionable information.

For details, refer to: <https://networkbuilders.intel.com/solutionslibrary/telemetry-aware-scheduling-automated-workload-optimization-with-kubernetes-k8s-technology-guide>

### 4.3 Real-Time System Support

For use cases that require handling of time sensitive workloads (i.e., PODs), exposure of this support ensures that any time-critical events PODs may depend on are processed and responded to as efficiently as possible. The real-time system provides a 'best effort' mechanism allowing for user tasks to run at higher priority than kernel interrupts. This enables deterministic and repeatable results and quantifiable latency resulting in execution of what you want when you want it.

**Note:** Support is currently limited to CentOS 7.7, based on the CentOS-RT.repo and associated RT Kernel and tuned system packages.

### 4.4 Dynamic Device Personalization (DDP)

One of the key technologies of the Intel® Ethernet 700 and 800 Series Network Adapters is Dynamic Device Personalization (DDP). This technology enables workload-specific optimizations using the programmable packet-processing pipeline. Additional protocols in the default set improve packet processing efficiency that results in better performance in specific use cases.<sup>8</sup>

Additional information about DDP can be found in the Technology Brief <https://www.intel.com/content/www/us/en/architecture-and-technology/ethernet/dynamic-device-personalization-brief.html>

#### 4.4.1 DDP on Intel Ethernet 700 Series Network Adapters

Intel Ethernet 700 Series Network Adapters support only one DDP image loaded on a network card. An image must be loaded to the first physical function of the device (PFO), but the configuration is applied to all ports of the adapter.

For Intel Ethernet 700 Series Network Adapters, BMRA provides the following list of DDP images:

- ecpri.pkg
- esp-ah.pkg
- ppp-oe-ol2tpv2.pkg
- mplsogreudp.pkg
- gtp.pkg

To use specific DDP-enabled Virtual Functions in BMRA, create a new resource with `ddpImage` selector, by extending the `sriovdp_config_data` variable before deployment.

```
PATH: examples/full_nfv/group_vars/all.yml
sriovdp_config_data: |
  {
    "resourceList": [
      {
        "resourceName": " intel_sriov_dpdk_700_series_gtpgo",
        "selectors": {
          "vendors": ["8086"],
          "devices": ["154c"],
          "drivers": ["vfio-pci"]
          "ddpProfiles": ["GTPv1-C/U IPv4/IPv6 payload"]
        }
      },
      { (...) }
    ]
  }
```

For more details about SR-IOV Device Plugin configuration, refer to <https://github.com/intel/sriov-network-device-plugin#configurations>

After BMRA Deployment, request the `intel_sriov_dpdk_700_series_gtpgo` resource in the workload's Pod manifest:

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

<sup>8</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

```

metadata:
  name: testpod1
  annotations:
    k8s.v1.cni.cncf.io/networks: sriov-net1
spec:
  containers:
  - name: appcntrl
    image: centos/tools
    imagePullPolicy: IfNotPresent
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "while true; do sleep 300000; done;" ]
    resources:
      requests:
        intel.com/intel_sriov_dpdk_700_series_gtpgo: '1'
      limits:
        intel.com/intel_sriov_dpdk_700_series_gtpgo: '1'

```

Keep the following points in mind when working with DDP images on Intel® Ethernet 700 Series Network Adapters:

- DDP profiles are NOT automatically unloaded when the driver is unbound/unloaded. Note that subsequent driver reload may corrupt the profile configuration during its initialization and is NOT recommended.
- DDP profiles should be manually rolled-back before driver unload/unbind if the intention is to start with a clean hardware configuration.
- Exercise caution while loading DDP profiles. Attempting to load files other than DDP profiles provided by Intel may cause system instability, system crashes, or system hangs.

#### Additional References

- <https://software.intel.com/content/www/us/en/develop/articles/dynamic-device-personalization-for-intel-ethernet-700-series.html>
- <https://builders.intel.com/docs/networkbuilders/intel-ethernet-controller-700-series-gtpv1-dynamic-device-personalization.pdf>
- <https://github.com/intel/sriov-network-device-plugin>

### 4.4.2 DDP on Intel® Ethernet 800 Series Network Adapters

Intel® Ethernet 800 Series Network Adapters support a broad range of protocols in DDP profile images<sup>9</sup>. You can choose the default image or the *telecommunications* (comms) image that supports certain market-specific protocols in addition to protocols in the OS-default package. The OS-default DDP package supports the profiles listed in the following table.

#### DEFAULT PACKAGE

---

MAC

---

EtherType

---

VLAN

---

IPv4

---

IPv6

---

TCP

---

ARP

---

UDP

---

SCTP

---

ICMP

---

ICMPV6

---

CTRL

---

LLDP

---

VXLAN-GPE

---

VxLAN (non-GPE)

---

Geneve

---

GRE

---

NVGRE

<sup>9</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

**DEFAULT PACKAGE**


---

RoCEv2

---

**COMMS PACKAGE**


---

Extends default profile package with the following protocols:

---

GTP

---

PPPOE

---

L2TPv3

---

IPsec

---



---

PFCP

---

To deploy communication profile on the network adapter, the host variable `dataplane_interfaces` must contain a proper `ddp_profile` value. For example:

```
dataplane_interfaces:
- name: enp24s0f0           # PF interface name
  bus_info: "18:00.0"      # pci bus info
  pf_driver: i40e          # PF driver, "i40e", "ice"
  default_vf_driver: "iavf" # default driver to be used with VFs
  sriov_numvfs: 6         # total number of VFs to create
  sriov_vfs:              # list of VFs to create with specific driver
    vf_00: "vfio-pci"     # VF driver to be attached to this VF under this PF.
    vf_05: "vfio-pci"     # VF driver to be attached to this VF under this PF.
```

`ddp_profile: "ice_comms-1.3.24.0"` # DDP package name to be loaded into the NETWORK ADAPTER. The image must be loaded to first physical function of the device (PFO), but the configuration is applied to all ports of the adapter.

To deploy a workload with a DDP SR-IOV VF, the `intel_sriov_dpdk_800_series` resource must be requested in the workload's Pod manifest, after BMRA Deployment:

```
apiVersion: v1
kind: Pod
metadata:
  name: testpod1
  annotations:
    k8s.v1.cni.cncf.io/networks: sriov-net1
spec:
  containers:
  - name: appcntrl
    image: centos/tools
    imagePullPolicy: IfNotPresent
    command: [ "/bin/bash", "-c", "--" ]
    args: [ "while true; do sleep 300000; done;" ]
    resources:
      requests:
        intel.com/intel_sriov_dpdk_800_series: '1'
      limits:
        intel.com/intel_sriov_dpdk_800_series: '1'
```

#### 4.4.2.1 Kubernetes Support for DDP in Intel Ethernet 800-Series Network Adapters

SR-IOV devices (VFs included) are exposed to the Kubernetes pods via SR-IOV Device Plugin (SR-IOV DP). In BMRA, the plugin cannot discover loaded DDP plugins on Intel Ethernet 800-Series Network Adapters (<https://github.com/intel/sriov-network-device-plugin#configurations>), so that one cannot orchestrate a workload that requires a DDP profile on such network adapters.

However, you can work around this issue in the following ways:

1. You can deploy *comms* profile on every node with Intel Ethernet 800 Series Network Adapter, or
2. You can deploy *comms* profile on a subset of Intel Ethernet 800 Series Network Adapters and label the nodes accordingly using Kubernetes labels. Then deploy the workload that requires specific DDP profile with specific Node Selector.

## 4.5 Intel® Speed Select Technologies

### 4.5.1 Intel® Speed Select – Base Frequency

Select SKUs of 2nd Generation Intel® Xeon® Scalable processors (5218N, 6230N, and 6252N) and 3rd Generation Intel® Xeon® Scalable processors (6318N and 6338N) offer a new capability called Intel® Speed Select Technology – Base Frequency (Intel® SST-BF). Intel® SST-BF controls the core frequency model. When enabled, some cores are at higher frequency and the remaining cores run at lower frequency<sup>10</sup>.

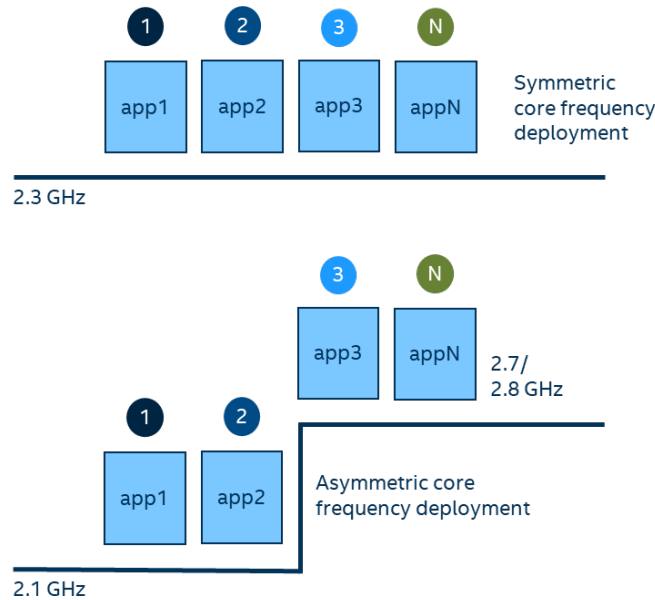


Figure 9. CPU Core Frequency Deployment Methods

Applications that consist of virtualized switches or load distribution functions in front of worker threads can benefit from the Intel® SST-BF feature. With high frequencies, load distribution threads pass data extremely fast to workers.

High frequency cores are advertised as *exclusive cores* by the CPU Manager for Kubernetes. Pod manifest must request for `cmk.intel.com/exclusive-cores` to obtain one.

Intel® SST-BF enabled on node is advertised by the Node Feature Discovery with the `sst_bf.enabled` label.

Example pod manifest with exclusive cores request:

```
apiVersion: v1
kind: Pod
metadata:
  name: high-priority-pod
  annotations:
    cmk.intel.com/mutate: "true"
spec:
  restartPolicy: Never
  containers:
  - name: nginx
    image: nginx:1.19.3
    ports:
    - containerPort: 80
  resources:
    requests:
      cmk.intel.com/exclusive-cores: '1'
    limits:
      cmk.intel.com/exclusive-cores: '1'
  nodeSelector:
    feature.node.kubernetes.io/cpu-power.sst_bf.enabled: true
```

For more information about scheduling workloads on Kubernetes with Intel® SST-BF support, visit <https://builders.intel.com/docs/networkbuilders/intel-speed-select-technology-base-frequency-with-kubernetes-application-note.pdf>

<sup>10</sup> See backup for workloads and configurations or visit [www.intel.com/PerformanceIndex](http://www.intel.com/PerformanceIndex). Results may vary.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

For more information about Intel® SST-BF technology, refer to <https://www.kernel.org/doc/html/latest/admin-guide/pm/intel-speed-select.html#intel-r-speed-select-technology-base-frequency-intel-r-sst-bf>

On the 3rd Generation Intel® Xeon® Scalable processor platform it is possible to prioritize power flow to the high priority cores in case power constraint scenario – for example, TDP ceiling hit or power supply malfunction. Power prioritization helps high frequency cores to maintain their frequency.

In BMRA deployment, Core Priority of high frequency cores is enabled by default.

### 4.5.2 Intel® Speed Select – Core Power

Intel® Speed Select Technology – Core Power (Intel® SST-CP) available on 3rd Generation Intel® Xeon® Scalable processor allows users to define priorities in core power flow in case power constraint scenario – for example TDP ceiling hit or power supply malfunction. Power prioritization helps cores to maintain their frequency, while power is drawn from cores with lesser priority.

BMRA allows you to define four Intel® SST-CP Classes of Service that contain:

1. Priority
2. Affected Cores
3. Minimum CPU Frequency
4. Maximum CPU Frequency

Intel® SST-CP configuration can be found in `example/profile/host_vars/node1.yml`.

Intel® SST-CP enabled on a node is advertised by Node Feature Discovery with `sst_cp.enabled` label.

For more information about Intel® SST-CP, 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>

## 4.6 Security

### 4.6.1 Cluster Security

Extra effort was placed on securing the Kubernetes cluster with this release<sup>11</sup>. The following security additions were made as reference only:

- Enabled Kubernetes audit trail and log backups by default
- Enabled certs for API server access and timeouts
- Added checksums for all downloaded artifacts
- Enabled RBAC by default
- Help secure cluster communication via TLS cipher suites
- Help secure the Docker registry
  - Limited access with TLS and basic authentication
  - Required server keys and certs signed with Kubernetes CA
- Limited etcd permissions
- Limited open firewall ports and subnets
- Enabled the Pod Security Policies (PSP) admission controller with minimal set of rules (defines a set of conditions pods must use to run within the system):
  - EventRateLimit – mitigates DoS attacks against API server
  - AlwaysPullImages – forces credential checks every time a pod image is accessed
  - NodeRestriction – limits objects a kubelet can modify
  - PodSecurityPolicy
- Added security policies to each Helm chart for deployment (collectd, nfd, tas, etc.)

For additional security considerations, refer to:

- <https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/>
- <https://kubernetes.io/docs/concepts/policy/pod-security-policy/>
- <https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#noderestriction>

### 4.6.2 Intel® Security Libraries for Data Center (Intel® SecL – DC)

Security for cloud native applications is an increasing challenge for developers. Hardware and software suppliers in the industry are cooperatively developing the cloud architecture to deliver 5G network and edge infrastructure, which makes security an important requirement as workloads and suppliers converge.

---

<sup>11</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

By using Intel® SecL – DC, Hardware Root of Trust, and Secure Boot, we can create an end-to-end platform security solution for network and edge platforms. Key components of Intel® SecL – DC include the following:

- Verification service: Installed in the central control node, it gathers the secure boot signatures and maintains the platform's "trusted" or "untrusted" evaluation results. It maintains the platform trust database with established "known good" values or expected measurements. If a certain firmware or Linux kernel is found to be compromised, the policy here can change the platform trust status.
- Trust agent: Installed in every physical node that needs to be monitored by the platform security. It takes the TPM ownership on the platform and reports the platform security status to the remote management module.
- Integration hub: Connects with orchestration software such as Kubernetes and contains an extension to work with K8s. It retrieves the information from the verification service and shares with the orchestration software at a specified interval.

For details on how to deploy, refer to: <https://builders.intel.com/docs/networkbuilders/secure-the-network-infrastructure-secure-cloud-native-network-platforms-user-guide.pdf>

### 4.6.3 Intel® Software Guard Extensions

Intel® Software Guard Extensions (Intel® SGX) is an Intel technology to protect select code and data from disclosure or modification. It operates by allocating hardware-protected memory where code and data reside. The protected memory area is called an *enclave*. Data within the enclave memory can only be invoked via special instructions.

#### System Requirements

Hardware requirement: 6<sup>th</sup> Generation Intel® Core™ processor (or later) based platform with SGX Enabled BIOS support

Supported Linux OS distributions:

- Ubuntu 16.04 LTS 64-bit Desktop and Server version
- Ubuntu 18.04 LTS 64-bit Desktop and Server version
- Red Hat Enterprise Linux Server 8.2 (for x86\_64)
- Red Hat Enterprise Linux Server 7.6 (for x86\_64)
- CentOS 8.1
- Fedora 31
- SUSE 15

You must install the following components in order to use Intel® SGX.

- Intel® SGX driver
- Intel® SGX PSW (platform software)
- Intel® SGX SDK

For details, refer to: [https://download.01.org/intel-sgx/sgx-linux/2.11/docs/Intel\\_SGX\\_Installation\\_Guide\\_Linux\\_2.11\\_Open\\_Source.pdf](https://download.01.org/intel-sgx/sgx-linux/2.11/docs/Intel_SGX_Installation_Guide_Linux_2.11_Open_Source.pdf)

## 4.7 Intel® Server GPU

The Intel® Server GPU is the first discrete graphics processing unit for data centers based on the new Intel X<sup>e</sup> architecture with support for MPEG2, AVC, HEVC, and VP9 transcoding plus AV1 decode. The Intel® Server GPU is a high-density media processing accelerator that enables users, via configurable presets, to choose the video quality/density setting that meets their requirements. The Intel® Server GPU programming is supported via open-source software including drivers, APIs, and developer tools such as the Intel® Media SDK and FFmpeg plugin. The Intel® Server GPU supports low level programmability including frame and sub-frame level controls.

The Intel® Server GPU is based on a low-power discrete system-on-chip (SoC) design, with a 128-bit wide pipeline and 8 GB of dedicated onboard low-power DDR4 memory. Four GPU SoCs are packaged together in a three-quarter length, full height x16 PCIe Gen3 add-in card from H3C, with a target configuration of up to four cards per server. Kontron offers a low-profile alternative with two SoCs on a GEN4 x16 PCIe half width/half height card with high bandwidth per SoC.

The Open Visual Cloud is a set of open-source software stacks (with full end-to-end sample pipelines) for media, analytics, graphics, and immersive media, optimized for cloud native deployment on commercial-off-the-shelf x86 CPU architecture. An Open Visual Cloud reference application for OTT Transcoding and delivery via CDN, on both Intel® Xeon® and Intel® Server GPU, is available in the Open Visual Cloud GitHub site <https://github.com/OpenVisualCloud/CDN-Transcode-Samplev>.

For more details on supported operating systems and required firmware for the Intel® Server GPU, work with your local Intel sales representative. For more details on the product, see the following links:

<https://www.intel.com/content/www/us/en/products/discrete-gpus/server-graphics-card.html>

<https://www.intel.com/content/www/us/en/benchmarks/server/graphics/intelservergpu.html>

<https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/server-graphics-card-product-brief.html>

<https://www.intel.com/content/www/us/en/products/docs/discrete-gpus/server-graphics-solution-brief.html>

<https://ark.intel.com/content/www/us/en/ark/products/210576/intel-server-gpu.html>

### 4.8 Security - Key Management Reference Application with Intel® SGX

Key Management Reference Application (KMRA) is a proof-of-concept software created to demonstrate the integration of Intel® Software Guard Extensions (Intel® SGX) asymmetric key capability with a hardware security model (HSM) on a centralized key server. The goal of this section is to outline BMRA setup of KMRA infrastructure with Intel SGX for private key provisioning to an Intel® SGX enclave on a 3rd Generation Intel® Xeon® Scalable processor, using the Public-Key Cryptography Standard (PKCS) #11 interface and OpenSSL.

BMRA contains Ansible automation scripts to set up the KMRA infrastructure. This includes the setup of a centralized key server and multiple compute nodes enabled with Intel SGX. The compute nodes are provisioned with private keys into the Intel SGX enclaves to be used by workloads. This general solution can work with any application or workload. For this release, a sample NGINX workload/application Ansible script is provided as an example of how the KMRA infrastructure can be used. The NGINX workload uses the private keys from Intel SGX enclave to establish TLS connections. The private key is never in the clear and the certificate signing happens more securely inside the enclave.

Step by step instructions for deploying KMRA with BMRA are detailed in [Workloads and Application Examples](#). The instructions also provide information on how to deploy the NGINX workload and configure it to use the private key more securely from inside the enclave to establish TLS connections.

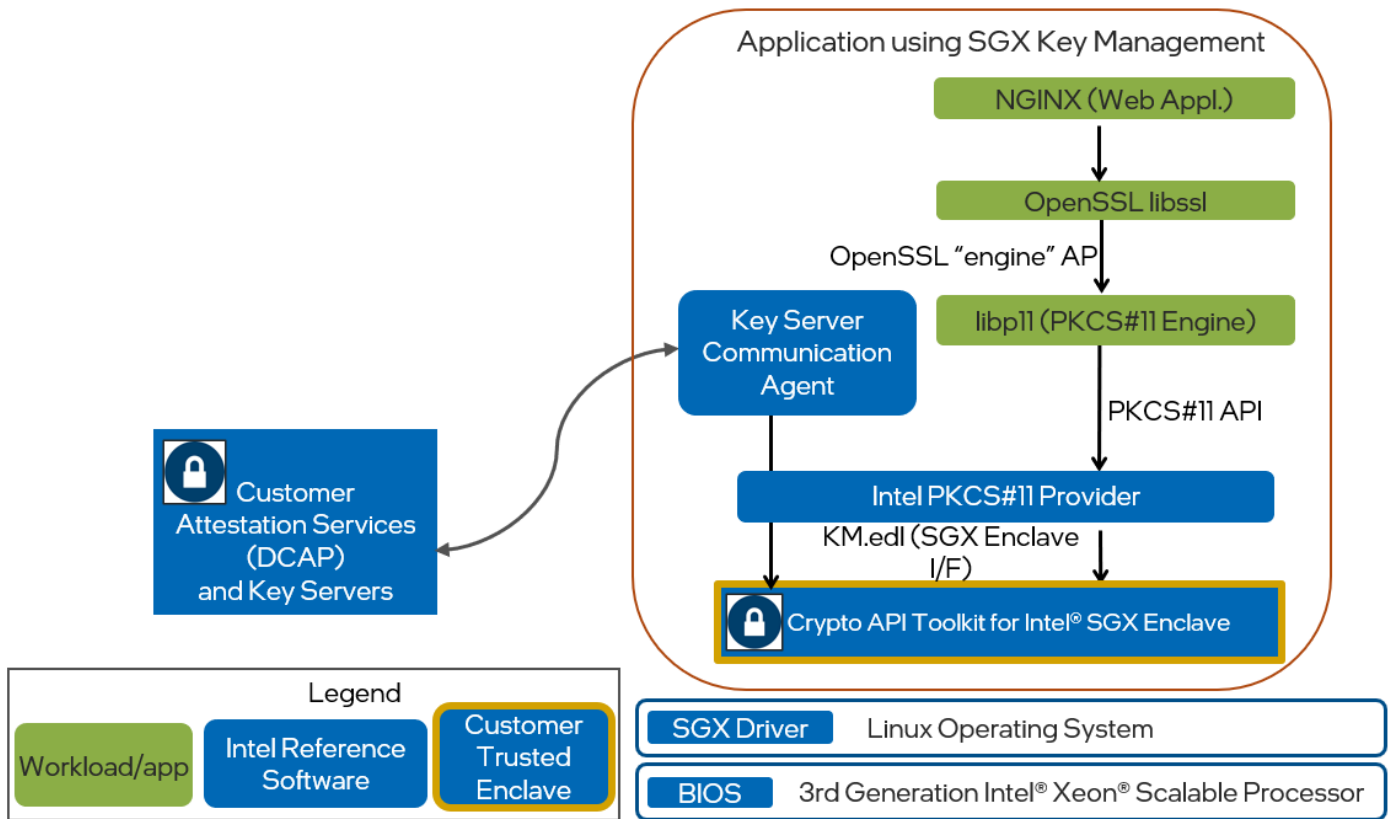


Figure 10. Key Management Reference Application Infrastructure with Intel® SGX

### 4.9 Observability

The BMRA Telemetry SW stack consists of telemetry collectors deployed on every node, a telemetry time-series database that pulls the metrics from collectors, telemetry visualization software, and a policy agent that influences cluster scheduling decisions.

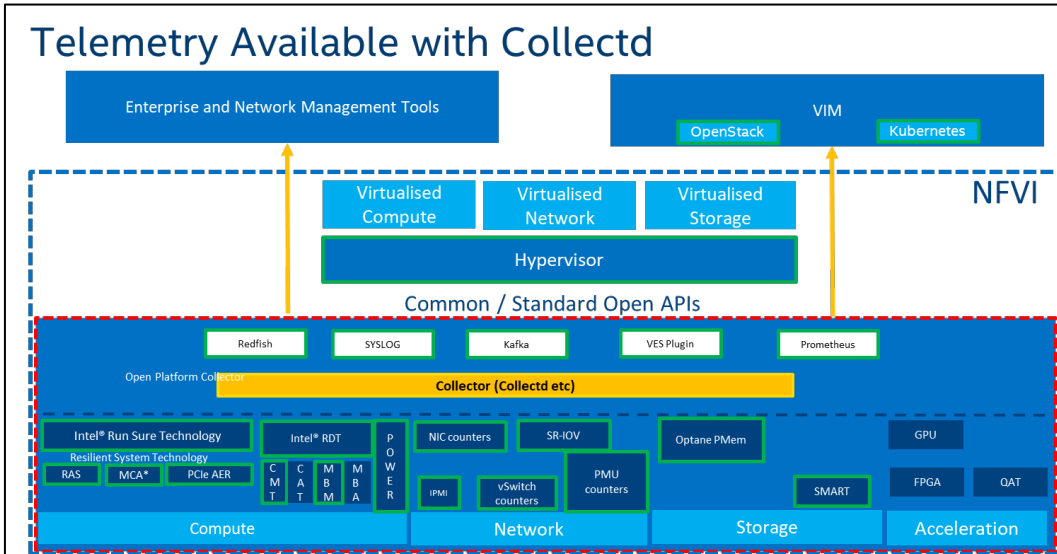


Figure 11. Platform Telemetry Available with Collectd

### 4.9.1 Observability Components Overview

This section describes the components shown in the previous diagram, including Collectd, Node Exporter, Prometheus, Grafana, and Prometheus adapter.

#### 4.9.1.1 Collectd

Collectd is a UNIX daemon responsible for collecting a wide spectrum of platform telemetry. It has a modular architecture and data acquisition depends on loaded plugins. Plugins that are loaded with collectd depend on an installed profile.

Table 4. Collectd Plugins

PLUGIN	PROFILE
cpu	All
cpufreq	All
disk	All
ipmi	All
numa	All
smart.conf	All
ethstat	All
netlink	All
intel_pmu	All
intel_rdt	All
pkgpower.py	All
dpdkevents	Full
dpdkstat	Full
hugepages	Full
ovs_events	Full
ovs_pmd_stats	Full
ovs_stats	Full

For detailed descriptions of the plugins and metrics available, visit <https://collectd.org/> and <https://wiki.opnfv.org/display/fastpath/Barometer+Home>.

For more information about pkgpower.py plugin, visit <https://github.com/intel/CommsPowerManagement/blob/master/power.md>

**Note:** To enable intel\_pmu plugin, set the enable\_intel\_pmu\_plugin: true configuration variable.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

**Note:** When intel\_rdt and intel\_pmu plugins run on a node at the same time, the following metrics are measured incorrectly:

- Instructions
- Cycles
- Instructions per cycle (IPC)
- Cycles per instruction (CPI)

These metrics are measured incorrectly because both plugins count the same metric using different means (MSR write vs. perf). You can choose to disable one plugin or the other by commenting the lines with `- intel_pmu.conf` or `- rdt.conf` in the `roles/collectd-install/default/main.yml` file.

### 4.9.1.2 Node Exporter

Node Exporter is a Prometheus exporter for hardware and OS metrics exposed by NIX kernels, written in Go with pluggable metric collectors.

Node Exporter is run with its default configuration on every node.

### 4.9.1.3 Prometheus

Prometheus is an open-source systems-monitoring server that scrapes and stores time series data.

### 4.9.1.4 Grafana

Grafana is an open-source telemetry visualization tool, which is available under the HTTPS endpoint: `https://localhost:30000` on any of the cluster nodes. Due to security reasons, this port is not exposed outside the cluster by default.

**Note:** Grafana is deployed with default credentials of admin/admin. After first log-in, the user is asked to change the password. We recommend that you change this as soon as possible.

### 4.9.1.5 Prometheus Adapter

Prometheus Adapter is an implementation of the Kubernetes resource metrics API and custom metrics API. It provides Prometheus metrics to the Telemetry Aware Scheduler.

## 4.9.2 Platform Telemetry Security

Platform Telemetry is an asset that must be protected. This section provides an overview of telemetry security.

### 4.9.2.1 Data at Rest Security

**Note:** BMRA does not provide any data at rest security for telemetry. We recommend that you provide proper means of data at rest security, such as self-encrypting drives.

### 4.9.2.2 Data in Transit Security

BMRA helps secure the telemetry traffic between nodes using TLS. Prometheus verifies a collector's certificate on connection using an in-cluster CA certificate while the collector ensures that the service account token provided with HTTP connection allows it to read `/metrics` endpoint. Service Account Token verification is provided by the Kubernetes API Server.

## 5 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.

### 5.1 Hardware Component List for Control Node

This table lists the hardware options for control nodes, which are responsible for managing the worker nodes in the cluster.

**Table 5. 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, 125W 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

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
Intel® Optane™ Persistent Memory	512 GB (4 x 128 GB Intel® Optane™ Persistent Memory in 2-1-1 Topology)	Recommended
Network Adapter	Option 1: Dual Port 25GbE Intel® Ethernet Network Adapter XXV710-DA2 SFP28+, or	Required
	Option 2: Dual Port 10GbE Intel® Ethernet Converged Network Adapter X710-DA2 SFP+, or	
	Option 3: Dual Port 10GbE 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
LAN on Motherboard (LOM)	10 Gbps or 25 Gbps port for Preboot Execution Environment (PXE) and Operation, Administration, and Management (OAM)	Required
	1/10Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

**Table 6. 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
Intel® Optane™ Persistent Memory	512 GB (4x 128 GB Intel® Optane™ persistent memory in 2-1-1 topology)	Recommended
Network Adapter	Dual Port 100GbE 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 3.84TB 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
	1/10Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

## 5.2 Hardware Component List for Worker Node Base

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

This table lists 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 7. 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 20C/40T, 125W or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	Option 1: DRAM only configuration: 384GB (12 x 32 GB DDR4 2666 MHz)	Required
	Option 2: DRAM only configuration: 384GB (24 x 16 GB DDR4 2666 MHz)	
	Option 3: DRAM + Intel® Optane™ Persistent Memory DRAM: 192 GB (12x 16 GB DDR4, 2666 MHz)	

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
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 100GbE 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
	1/10Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

Table 8. 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, 24C/48T, 150W or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	DRAM only configuration: 512 GB DRAM (32x 16 GB DDR4, 2666 MHz)	Required
Intel® Optane™ Persistent Memory	512 GB (4x 128 GB Intel® Optane™ persistent memory in 2-1-1 Topology)	Recommended
Network Adapter	Option 2: Dual Port 100GbE Intel® Ethernet Network Adapter E810-CQDA2 QSFP28, or	Required
	Option 3: Dual Port 100GbE Intel® Ethernet Network Adapter E810-2CQDA2 QSFP28	
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 3.84TB 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
	1/10Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

### 5.3 Hardware Component List for Worker Node Plus

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

This table lists 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 9. 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 24C/48T, 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

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
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 25GbE Intel® Ethernet Network Adapter X710-DA4 SFP+, or	Required
	Option 2: Dual Port 100GbE 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
	1/10Gbps port for Management Network Adapter	Required
Additional Plug-in cards	N/A	

**Table 10. Hardware Components for Worker Node Plus – 3rd Generation Intel Xeon Scalable Processor**

INGREDIENT	REQUIREMENT	REQUIRED/ RECOMMENDED
3rd Generation Intel Xeon Scalable Processors	Intel® Xeon® Gold 6338N CPU @ 2.2 GHz 32C/64T, 185W or higher number Intel® Xeon® Gold or Platinum CPU SKU	Required
Memory	DRAM only configuration: 512 GB DRAM (32x 16 GB DDR4, 2666 MHz)	Required
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: 512 GB (4x 128 GB Intel® Optane™ persistent memory in 2-1-1 topology)	Recommended
	Option 2: 1 TB (8x 128 GB Intel® Optane™ persistent memory in 2-2-1 Topology)	
	Option 3: 2 TB (16x 128 GB Intel® Optane™ persistent memory in 2-2-2 Topology)	
Network Adapter	Option 1: Dual Port 100GbE Intel® Ethernet Network Adapter E810-CQDA2 QSFP28	Recommended
	Option 2: Dual Port 100GbE Intel® Ethernet Network Adapter E810-2CQDA2 QSFP28	
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 3.84TB 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
	1/10Gbps port for Management Network Adapter	Required
Additional Plug-in cards	Intel® Server Graphics 1 card	Optional

## 5.4 Hardware BOMs for all Configuration Profiles

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

**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

**Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor**

Mem	192GB	192GB	192GB
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 AIC/Lewisburg-NS	Recommended	N/A	N/A
<b>BIOS Configuration</b>			
Intel® HT 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 Enable	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	256GB	256GB	256GB
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 AIC/Lewisburg-NS	Recommended	N/A	N/A
<b>BIOS Configuration</b>			
Intel® HT 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 Enable	No	Yes	Yes

**Table 13. 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	384GB	384GB	384GB
Intel Optane Persistent Memory	Recommended - 512GB	Recommended - 512GB	Recommended - 512GB
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	Yes	No
Intel® QAT AIC/Lewisburg-NS	No	Yes	Optional
Additional Plug-in cards	No	No	No
<b>BIOS Configuration</b>			
Intel® HT 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 Enable	No	Yes	Yes

**Table 14. 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	384GB	384GB
Intel Optane Persistent Memory	Recommended - 1TB/1.5TB	Recommended - 1TB/1.5TB
Network Adapter	2x E810-CQDA2	2x E810-CQDA2
Storage (Boot Media)	Required – 256GB	Required – 256GB
LOM	No	Yes
Intel® QAT AIC/Lewisburg-NS	No	Yes
Additional Plug-in cards	No	No
<b>BIOS Configuration</b>		
Intel® HT enabled	Yes	Yes
Intel® VT-x enabled	Yes	Yes
Intel® VT-d enabled	Yes	Yes
BIOS Profile	Deterministic	Max Performance
Virtualization Enable	No	Yes

**Table 15. 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	512GB	512GB	512GB
Intel Optane Persistent Memory	Recommended – 512GB	Recommended – 512GB	Recommended – 512GB

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

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 AIC/Lewisburg-NS	No	Yes	Optional
Additional Plug-in cards	No	No	No
<b>BIOS Configuration</b>			
Intel® HT 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 Enable	No	Yes	Yes

**Table 16. Worker Node Plus 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
Platform	M50CYP	M50CYP	M50CYP
CPU/node	2x 6338N 32c	2x 6338N 32c	2x 6338N 32c
Mem	512GB	512GB	512GB
Intel Optane Persistent Memory	Recommended – 512GB	Recommended – 512GB	Recommended – 512GB
Network Adapter	2x E810-2CQDA2 or 4x E810-CQDA2	2x E810-2CQDA2 or 4x E810-CQDA2	2x E810-2CQDA2
Storage (Boot Media)	Required - 2x	Required - 2x	Required - 2x
Storage (Capacity)	Required- 4x (2 per NUMA)	Required- 4x (2 per NUMA)	Required- 4x (2 per NUMA)
LOM	Yes	No	Yes
Intel® QAT AIC/Lewisburg-NS	Yes	Optional	No
Additional Plug-in cards	No	No	Intel Server GPU
<b>BIOS Configuration</b>			
Intel® HT 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 Enable	Yes	Yes	Yes

## 5.5 Platform BIOS

This chapter 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 chapter 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](https://www.intel.com/content/dam/support/us/en/documents/server-products/Intel_Xeon_Processor_Scalable_Family_BIOS_User_Guide.pdf).

Table 17. Platform BIOS Settings for 2<sup>nd</sup> 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 is enabled

Table 18. Platform BIOS Settings for 3rd Generation Intel® Xeon® Scalable Processor

MENU (Advanced)	Path to BIOS Setting	BIOS Setting	Energy Balance	Max Performance with Turbo	Deterministic
Socket Configuration	Processor Configuration	Hyper-Threading	Enable	Enable	Enable
		XAPIC	Enable	Enable	Enable
		VMX	Enable	Enable	Enable
		Uncore frequency scaling	Enable	Enable	Disable
		Uncore frequency	800-2400	800-2400	2400
Power Configuration	Power and Performance	CPU Power and Performance Policy	Balance Performance	Performance	Performance
		Workload Configuration	I/O sensitive	I/O sensitive	I/O sensitive
	CPU P State Control	EIST PSD Function	HW_ALL	HW_ALL	HW_ALL
		Boot Performance Mode	Max. Performance	Max. Performance	Max. Performance
		AVX License Pre-Grant	Disable	Disable	Disable
		AVX ICCP Pre Grant Level	NA	NA	NA
		AVX P1	Nominal	Nominal	Nominal
		Energy Efficient Turbo	Enable	Enable	Disable
		WFR Uncore GV rate Reduction	Enable	Enable	Enable
		GPSS timer	500us	0us	0us
		Turbo	Disable	Enable	Disable
		Intel SpeedStep® Technology (P-states)	Enable	Enable	Disable
	Frequency Prioritization	RAPL Prioritization	Enable	Disable	Disable
	Hardware PM State Control	Hardware P-States	Native Mode with no legacy Support	Disabled	Disable
		EPP enable	Enable	Disabled	Disable
	CPU C State Control	Enable Monitor Mwait	Enable	Enable	Enable
		CPU C1 Auto Demotion	Enable	Disable	Disable
		CPU C1 Auto unDemotion	Enable	Disable	Disable
		CPU C6 Report	Enable	Enable	Disable
		Processor C6	Enable	Enable	Disable
		Enhanced Halt State (C1E)	Enable	Enable	Disable
		OS ACPI Cx	ACPI C2	ACPI C2	ACPI C2
	Energy Performance Bias	Power Performance Tuning	OS Controls EPB	OS Controls EPB	OS Controls EPB
		ENERGY_PERF_BIAS_CFG mode	Performance	Performance	Performance

	Package C State Control	Workload Configuration	I/O Sensitive	I/O Sensitive	I/O Sensitive
		Package C State	C6 Retention	C6 Retention	C0/C1 State
		Dynamic L1	Enable	Disable	Disable
		Package C-state Latency Negotiation	Disable	Disable	Disable
		PKG_CSA_PS_CRITERIA	Disable	Disable	Disable
Memory Configuration		Memory Configuration	2-way interleave	2-way interleave	2-way interleave
		Enforce POR	Enable	Enable	Enable
Platform Configuration	Miscellaneous Configuration	Serial Debug Message Level	Minimum	Minimum	Minimum
	PCI Express* Configuration	PCIe* ASPM Support	Per Port	Per Port	Per Port
	PCI Express* Configuration	PCIe* ASPM	Enable	Disable	Disable
	PCI Express* Configuration	ECRC generation and checking	Enable	Enable	Enable
Server Management		Resume on AC Power Loss	Power On	Power On	Power On
System Acoustic and Performance Configuration		Set Fan Profile	Acoustic	Performance	Performance

## 6 Reference Architecture Software Components

This section describes the software version details.

**Table 19. Software Components**

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
Host OS	CentOS 7.9 Kernel version: 3.10.0-1160.el7.x86_64	<a href="https://www.centos.org/">https://www.centos.org/</a>
	CentOS 8.3 Kernel version: 4.18.0-240.el8.x86_64	
	Ubuntu 18.04 Kernel version: 4.15.0-96-generic	<a href="https://ubuntu.com/">https://ubuntu.com/</a>
	Ubuntu 20.04 Kernel version: 5.4.0-26-generic	
	RHEL 8.3 Kernel version: 4.18.0-240	
Ansible	Ansible v2.7.16	<a href="https://www.ansible.com/">https://www.ansible.com/</a>
BMRA Ansible Playbook	v21.03	<a href="https://github.com/intel/container-experience-kits">https://github.com/intel/container-experience-kits</a>
Python	Python 3.6 for RHEL 8/CentOS 8 and Ubuntu 18.04 Python 3.8 for Ubuntu 20.04 Python 2.7 for CentOS 7	<a href="https://www.python.org/">https://www.python.org/</a>
Kubespray	Kubespray v2.14+	<a href="https://github.com/kubernetes-sigs/kubespray">https://github.com/kubernetes-sigs/kubespray</a>
Docker	Docker 19.03	<a href="https://www.docker.com/">https://www.docker.com/</a>
Container orchestration engine	Kubernetes v1.20.4	<a href="https://github.com/kubernetes/kubernetes">https://github.com/kubernetes/kubernetes</a>
	Kubernetes v1.19.8	
	Kubernetes v1.18.16	
CPU Manager (native to Kubernetes)	Available natively in K8s	N/A

SOFTWARE FUNCTION	SOFTWARE COMPONENT	LOCATION
CPU Manager for Kubernetes	CPU Manager for Kubernetes v1.5.1	<a href="https://github.com/intel/CPU-Manager-for-Kubernetes">https://github.com/intel/CPU-Manager-for-Kubernetes</a> Container image requires GNU libc 2.29 or newer to run CPU Manager for Kubernetes.
Telemetry Aware Scheduling	TAS 0.2	<a href="https://github.com/intel/telemetry-aware-scheduling">https://github.com/intel/telemetry-aware-scheduling</a>
CollectD	v5.12	<a href="https://www.collectd.org/">https://www.collectd.org/</a>
Grafana	v7.2.0	<a href="https://www.grafana.com/">https://www.grafana.com/</a>
Kube Prometheus	v0.6.0	<a href="https://github.com/prometheus-operator/kube-prometheus">https://github.com/prometheus-operator/kube-prometheus</a>
Node Feature Discovery	NFD v0.7.0	<a href="https://github.com/kubernetes-sigs/node-feature-discovery">https://github.com/kubernetes-sigs/node-feature-discovery</a>
Data Plane Development Kit	DPDK 19.11.6	<a href="https://core.dpdk.org/download/">https://core.dpdk.org/download/</a>
Open vSwitch with DPDK	OVS-DPDK v2.13.0	<a href="http://docs.openvswitch.org/en/latest/intro/install/dpdk/">http://docs.openvswitch.org/en/latest/intro/install/dpdk/</a>
Vector Packet Processing	VPP 19.04	<a href="https://docs.fd.io/vpp/19.04/index.html">https://docs.fd.io/vpp/19.04/index.html</a>
Multus CNI	Multus CNI v3.4.2	<a href="https://github.com/intel/multus-cni">https://github.com/intel/multus-cni</a>
SR-IOV CNI	SR-IOV CNI v2.6	<a href="https://github.com/intel/sriov-cni">https://github.com/intel/sriov-cni</a>
SR-IOV network device plugin	SR-IOV network device plugin v3.3.1	<a href="https://github.com/intel/sriov-network-device-plugin">https://github.com/intel/sriov-network-device-plugin</a>
DDP Profiles	Dynamic Device Personalization for Intel® Ethernet 700 Series Version 25.4	<a href="https://downloadmirror.intel.com/27587/eng/gtp.zip">https://downloadmirror.intel.com/27587/eng/gtp.zip</a> <a href="https://downloadmirror.intel.com/28940/eng/mplsogreudp.zip">https://downloadmirror.intel.com/28940/eng/mplsogreudp.zip</a> <a href="https://downloadmirror.intel.com/28040/eng/ppp-oe-ol2tpv2.zip">https://downloadmirror.intel.com/28040/eng/ppp-oe-ol2tpv2.zip</a> <a href="https://downloadmirror.intel.com/29446/eng/esp-ah.zip">https://downloadmirror.intel.com/29446/eng/esp-ah.zip</a> <a href="https://downloadmirror.intel.com/29780/eng/ecpri.zip">https://downloadmirror.intel.com/29780/eng/ecpri.zip</a>
	1.3.24.0	<a href="https://downloadcenter.intel.com/download/29889/Intel-Ethernet-800-Series-Telecommunication-Comms-Dynamic-Device-Personalization-DDP-Package">https://downloadcenter.intel.com/download/29889/Intel-Ethernet-800-Series-Telecommunication-Comms-Dynamic-Device-Personalization-DDP-Package</a>
QAT device plugin	0.19.0	<a href="https://github.com/intel/intel-device-plugins-for-kubernetes.git">https://github.com/intel/intel-device-plugins-for-kubernetes.git</a>
GPU device plugin	0.19.0	<a href="https://github.com/intel/intel-device-plugins-for-kubernetes.git">https://github.com/intel/intel-device-plugins-for-kubernetes.git</a>
SGX device plugin	0.19.0	<a href="https://github.com/intel/intel-device-plugins-for-kubernetes.git">https://github.com/intel/intel-device-plugins-for-kubernetes.git</a>
Userspace CNI	Userspace CNI v1.3	<a href="https://github.com/intel/userspace-cni-network-plugin">https://github.com/intel/userspace-cni-network-plugin</a>
Bond CNI plugin	Bond CNI plugin v1.0	<a href="https://github.com/intel/bond-cni">https://github.com/intel/bond-cni</a>
Intel® SecL – DC	v1.6	<a href="https://01.org/intel-secl">https://01.org/intel-secl</a>
Intel® Ethernet Drivers	Multiple	<a href="https://sourceforge.net/projects/e1000/files/i40e%20stable/2.15.9/">https://sourceforge.net/projects/e1000/files/i40e%20stable/2.15.9/</a> <a href="https://sourceforge.net/projects/e1000/files/ice%20stable/1.4.11/">https://sourceforge.net/projects/e1000/files/ice%20stable/1.4.11/</a> <a href="https://sourceforge.net/projects/e1000/files/iavf%20stable/4.1.1/">https://sourceforge.net/projects/e1000/files/iavf%20stable/4.1.1/</a>
Intel® QAT Drivers	1.7.L.4.13.0-00009	<a href="https://downloadmirror.intel.com/30178/eng/QAT1.7.L.4.13.0-00009.tar.gz">https://downloadmirror.intel.com/30178/eng/QAT1.7.L.4.13.0-00009.tar.gz</a>
Intel® SGX DCAP Drivers	1.41	<a href="https://download.01.org/intel-sgx/sgx-dcap/1.10/linux/distro/">https://download.01.org/intel-sgx/sgx-dcap/1.10/linux/distro/</a>
	1.36.2 (KMRA Only)	<a href="https://download.01.org/intel-sgx/sgx-dcap/1.9/linux/distro/">https://download.01.org/intel-sgx/sgx-dcap/1.9/linux/distro/</a>
Intel® SGX SDK	2.13.100.4	<a href="https://download.01.org/intel-sgx/sgx-dcap/1.10/linux/distro/">https://download.01.org/intel-sgx/sgx-dcap/1.10/linux/distro/</a>
	2.12.100.3 (KMRA Only)	<a href="https://download.01.org/intel-sgx/sgx-dcap/1.9/linux/distro/">https://download.01.org/intel-sgx/sgx-dcap/1.9/linux/distro/</a>

12

## 7 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 appendices:

- [Appendix B, BMRA Basic Configuration Profile Setup](#)
- [Appendix C, BMRA Full Configuration Profile Setup](#)

<sup>12</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

- [Appendix D, BMRA On-Premises Edge Configuration Profile Setup](#)
- [Appendix E, BMRA Remote CO-Forwarding Configuration Profile Setup](#)
- [Appendix F, BMRA Regional Data Center Configuration Profile Setup](#)

## 7.1 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
controller1      Ready    master   4d1h   v1.19.8   10.250.192.155 <none>        Ubuntu 20.04.2
LTS 5.4.0-66-generic docker://19.3.12
node1            Ready    <none>   4d1h   v1.19.8   10.250.190.112 <none>        Ubuntu 20.04.2
LTS 5.4.0-66-generic docker://19.3.12
```

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
kube-system    cmk-qwskg                                              2/2     Running  0
4d1h
kube-system    cmk-webhook-6697d4fcf8-xjfk8                          1/1     Running  0
4d1h
kube-system    coredns-dff8fc7d-1kc5p                                1/1     Running  1
4d1h
kube-system    coredns-dff8fc7d-w6hvx                                1/1     Running  2
4d1h
kube-system    dns-autoscaler-66498f5c5f-msc2v                       1/1     Running  1
4d1h
kube-system    docker-registry-5fff99bb6-ptzh8                       1/1     Running  0
4d1h
kube-system    intel-qat-plugin-intel-qat-plugin-2cq1n              1/1     Running  0
4d1h
kube-system    kube-apiserver-controller1                             1/1     Running  2
4d1h
kube-system    kube-controller-manager-controller1                   1/1     Running  1
4d1h
kube-system    kube-flannel-7dtzh                                    1/1     Running  1
4d1h
kube-system    kube-flannel-ljnvj                                    1/1     Running  1
4d1h
kube-system    kube-multus-ds-amd64-jn558                            1/1     Running  1
4d1h
kube-system    kube-multus-ds-amd64-lg622                            1/1     Running  1
4d1h
kube-system    kube-proxy-g5vvr                                       1/1     Running  1
4d1h
kube-system    kube-proxy-q7h25                                       1/1     Running  2
4d1h
kube-system    kube-scheduler-controller1                             1/1     Running  0
4d1h
kube-system    kubernetes-dashboard-57777fbdc6-6mtbk                1/1     Running  2
4d1h
kube-system    kubernetes-metrics-scraper-54fbb4d595-vg24r          1/1     Running  2
4d1h
kube-system    nginx-proxy-node1                                     1/1     Running  1
4d1h
kube-system    node-feature-discovery-controller-59f6c87899-ngsk5    1/1     Running  0
4d1h
kube-system    node-feature-discovery-worker-r9v2t                  1/1     Running  6
4d1h
kube-system    sriov-net-dp-kube-sriov-device-plugin-amd64-vvjzf     1/1     Running  0
4d1h
monitoring     collectd-txnjn                                         3/3     Running  0
4d1h
monitoring     node-exporter-7zvd1                                    2/2     Running  0
4d1h
```

monitoring 4dlh	prometheus-k8s-0	5/5	Running	1
monitoring 4dlh	prometheus-operator-56b749d7c8-xhndm	2/2	Running	0
monitoring 4dlh	tas-telemetry-aware-scheduling-7d5d67ccbc-mtsdf	2/2	Running	0

## 7.2 Check Intel® Speed Select Technology – Base Frequency (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>.

**Note:** The Intel® SST-BF feature is supported only on Ubuntu 20.04 and CentOS 8.2.

## 7.3 Check Intel® Speed Select Technology on 3rd Generation Intel® Xeon® Scalable Processor

The steps in this section describe how to verify Intel® Speed Select Technology.

### 7.3.1 Check Intel® Speed Select Technology - Base Frequency (Intel® SST-BF) Configuration

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

```
root@sdpl1146:~# intel-speed-select base-freq info -l 0
Intel(R) 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 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.

### 7.3.2 Check Intel® Speed Select Technology – Core Power (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(R) 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(R) 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 SST-CP or 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>

## 7.4 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.

### 7.4.1 Check DDP Profiles in Intel® Ethernet 700 Series Network Adapters

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

```
# ddptool -a
Intel(R) Dynamic Device Personalization Tool
DDPTool version 1.0.0.0
Copyright (C) 2019 Intel Corporation.
```

NIC	DevId	D:B:S:F	DevName	TrackId	Version	Name
001)	1572	0000:02:00.0	enp2s0f0	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
002)	1572	0000:02:00.1	enp2s0f1	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
003)	1572	0000:02:00.2	enp2s0f2	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
004)	1572	0000:02:00.3	enp2s0f3	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
005)	154C	0000:03:02.0	N/A	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
006)	154C	0000:03:02.1	enp3s2f1	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
007)	154C	0000:03:02.2	enp3s2f2	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload
008)	154C	0000:03:02.3	N/A	80000008	1.0.3.0	GTPv1-C/U IPv4/IPv6 payload

Download ddptool at: <https://downloads.sourceforge.net/project/e1000/ddptool%20stable/ddptool-1.0.0.0/ddptool-1.0.0.0.tar.gz>

### 7.4.2 Check DDP Profiles in Intel® Ethernet 800 Series Network Adapters

To verify the loaded DDP profile, use the dmesg tool.

```
[root@silpixa00385224 ~]# dmesg | grep DDP
[5505869.937683] ice 0000:0a:00.0: DDP package already present on device: ICE OS Default
Package version 1.3.4.0
```

The output contains the following information:

OUTPUT STRING	DESCRIPTION
ice	OS Driver for the Network Adapters
0000:0a:00.0	PCI Address of the Network Adapters
ICE OS Default Package version 1.3.4.0	DDP Package name and version

### 7.4.3 Check SR-IOV Resources

When everything is installed and set up correctly, you see that the device plugin is able to discover VFs with names given in the resource pool selector.

```
# kubectl get node node1 -o json | jq ".status.allocatable"
{
  "cpu": "8",
  "ephemeral-storage": "169986638772",
  "hugepages-1Gi": "0",
  "hugepages-2Mi": "8Gi",
  "intel.com/intel_sriov_dpdk_700_series": "2",
  "intel.com/intel_sriov_dpdk_800_series": "0",
  "memory": "7880620Ki",
  "pods": "100"
}
```

BMRA does not create SR-IOV resources with DDP by default. These must be created by the user, as described in the [Dynamic Device Personalization](#) section.

## 7.5 Check Node Feature Discovery (NFD)

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 finally, 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>    3d2h
```

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

```
# kubectl label node --list --all
Listing labels for Node./controller1:
kubernetes.io/arch=amd64
kubernetes.io/hostname=controller1
kubernetes.io/os=linux
node-role.kubernetes.io/master=
beta.kubernetes.io/arch=amd64
beta.kubernetes.io/os=linux
Listing labels for Node./node1:
beta.kubernetes.io/arch=amd64
feature.node.kubernetes.io/cpu-pstate.turbo=true
feature.node.kubernetes.io/cpu-cpuid.VMX=true
feature.node.kubernetes.io/kernel-version.minor=4
feature.node.kubernetes.io/cpu-rdt.RDTMBM=true
feature.node.kubernetes.io/cpu-cpuid.AESNI=true
feature.node.kubernetes.io/cpu-cpuid.AVX512F=true
feature.node.kubernetes.io/pci-0b40_8086.present=true
feature.node.kubernetes.io/memory-numa=true
feature.node.kubernetes.io/cpu-cpuid.MPX=true
kubernetes.io/os=linux
feature.node.kubernetes.io/network-sriov.configured=true
feature.node.kubernetes.io/cpu-power.sst_bf.enabled=true
feature.node.kubernetes.io/system-os_release.VERSION_ID.minor=04
feature.node.kubernetes.io/cpu-cpuid.ADX=true
feature.node.kubernetes.io/cpu-cpuid.AVX512VL=true
feature.node.kubernetes.io/cpu-rdt.RDTMON=true
feature.node.kubernetes.io/system-os_release.VERSION_ID.major=20
feature.node.kubernetes.io/cpu-cpuid.AVX512VNNI=true
feature.node.kubernetes.io/system-os_release.VERSION_ID=20.04
feature.node.kubernetes.io/kernel-version.revision=0
feature.node.kubernetes.io/pci-0300_1a03.present=true
kubernetes.io/arch=amd64
feature.node.kubernetes.io/cpu-cpuid.AVX512CD=true
feature.node.kubernetes.io/kernel-version.major=5
kubernetes.io/hostname=node1
feature.node.kubernetes.io/network-sriov.capable=true
feature.node.kubernetes.io/cpu-cpuid.AVX512DQ=true
feature.node.kubernetes.io/cpu-cpuid.IBPB=true
feature.node.kubernetes.io/system-os_release.ID=ubuntu
feature.node.kubernetes.io/cpu-cpuid.STIBP=true
feature.node.kubernetes.io/cpu-rdt.RDTL3CA=true
feature.node.kubernetes.io/kernel-config.NO_HZ_IDLE=true
feature.node.kubernetes.io/cpu-rdt.RDTMBA=true
feature.node.kubernetes.io/storage-nonrotationaldisk=true
feature.node.kubernetes.io/iommu-enabled=true
feature.node.kubernetes.io/cpu-cpuid.AVX2=true
feature.node.kubernetes.io/cpu-cpuid.FMA3=true
feature.node.kubernetes.io/cpu-cpuid.AVX=true
feature.node.kubernetes.io/kernel-config.NO_HZ=true
beta.kubernetes.io/os=linux
feature.node.kubernetes.io/cpu-hardware_multithreading=true
feature.node.kubernetes.io/cpu-rdt.RDTCMT=true
feature.node.kubernetes.io/kernel-version.full=5.4.0-66-generic
cmk.intel.com/cmknode=true
feature.node.kubernetes.io/cpu-cpuid.AVX512BW=true
```

The node labels can be used when provisioning a pod. In the following example, the pod is scheduled only if there is a node with CPU Manager for Kubernetes (CMK) available:

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nfd-1
spec:
```

```
nodeSelector:
  cmk.intel.com/cmk-node: "true"
containers:
- name: pod-nfd-1
  image: ubuntu:focal
  command:
  - "/bin/bash"
  - "-c"
  args:
  - "tail -f /dev/null"
```

In the above node labels, this is true for `node1`, and the pod is scheduled there. If no node is available that matches the `nodeSelectors`, the pod remains in `pending` status.

## 7.6 Check CPU Manager for Kubernetes

Kubernetes supports CPU and memory first class resources, while also providing basic support for CPU pinning and isolation through the native CPU Manager. To aid commercial adoption, Intel has created CPU Manager for Kubernetes, an open-source project that introduces additional CPU optimization capabilities. Without CPU Manager for Kubernetes, the kernel task scheduler treats all CPUs as available for scheduling process threads and regularly preempts executing process threads to give CPU time to other threads. This non-deterministic behavior makes it unsuitable for latency sensitive workloads.

Using the preconfigured `isolcpus` boot parameter, CPU Manager for Kubernetes can ensure that a CPU (or set of CPUs) is isolated from the kernel scheduler. Then the latency-sensitive workload process thread(s) can be pinned to execute on that isolated CPU set only, providing them exclusive access to that CPU set. While beginning to guarantee the deterministic behavior of priority workloads, isolating CPUs also addresses the need to manage resources, which allows multiple VNFs to coexist on the same physical server. The exclusive pool within CPU Manager for Kubernetes assigns entire physical cores exclusively to the requesting container, meaning no other container has access to the core.

CPU Manager for Kubernetes performs a variety of operations to enable core pinning and isolation on a container or a thread level. These include:

- Discovering the CPU topology of the machine.
- Advertising the resources available via Kubernetes constructs.
- Placing workloads according to their requests.
- Keeping track of the current CPU allocations of the pods, ensuring that an application receives the requested resources provided they are available.

CPU Manager for Kubernetes creates three distinct pools: exclusive, shared and infra. The exclusive pool is restricted, meaning only a single task may be allocated to a CPU at a time, whereas the shared and infra pools are shared such that multiple processes may be allocated to a CPU. Following is an output for successful CPU Manager for Kubernetes deployment and CPU initialization. In the example setup, Intel® HT is enabled. Therefore both physical and associated logical processors are isolated. Using the default values in `group_vars/all.yml`, CPU Manager for Kubernetes will allocate 2 cores to the exclusive pool, and 2 cores to the shared pool. The default values for `isolcpus` for the node are "4-11", which only partially isolates the physical cores as the system is configured with Intel® HT. Inspecting the logs for CPU Manager for Kubernetes will show output similar to the following:

```
root@av09-07-wp:~# kubectl get pods -A
NAMESPACE      NAME                                     READY   STATUS    RESTARTS   AGE
kube-system    cmk-init-discover-av09-06-wp-pl5tv     0/3     Completed 0           26m
kube-system    cmk-rnq9q                               2/2     Running   0           25m
kube-system    cmk-webhook-6c9d5f8578-jkfc9          1/1     Running   0           25m
```

```
root@av09-07-wp:~# kubectl get cm
NAME          DATA   AGE
cmk-config-av09-06-wp  1       30m

root@av09-07-wp:~# kubectl describe cm cmk-config-av09-06-wp
Name:         cmk-config-av09-06-wp
Namespace:    default
Labels:       <none>
Annotations:  Owner:

Data
====
config:
----
exclusive:
```

```

0:
  0,44: []
  1,45: []
1: {}
infra:
  0:

4,48,5,49,6,50,7,51,8,52,9,53,10,54,11,55,12,56,13,57,14,58,15,59,16,60,17,61,18,62,19,63,20,64,21,65:
  - '40218'
  1:
  ?
22,66,23,67,24,68,25,69,26,70,27,71,28,72,29,73,30,74,31,75,32,76,33,77,34,78,35,79,36,80,37,81,38,82,39,83,40,84,41,85,42,86,43,87
  : - '40167'
shared:
  0:
  2,46,3,47: []
  1: {}

Events: <none>

```

CPU Manager for Kubernetes prioritizes using the high priority SST-BF cores for the exclusive core list when possible. If there are fully isolated physical cores (through `isolcpus`) these are also prioritized for the exclusive pool. In the above output, there are no physical cores that are fully isolated, in which case CPU Manager for Kubernetes only looks at high priority SST-BF cores.

On successful run, the allocatable resource list for the node should be updated with resource discovered by the plugin as shown below. Note that the resource name is displayed in the format `cmk.intel.com/exclusive-cores`.

```

# kubectl get node node1 -o json | jq '.status.allocatable'
{
  "cmk.intel.com/exclusive-cores": "2",
  "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"
}

```

CPU Manager for Kubernetes ensures exclusivity, therefore the performance of latency sensitive workloads is not impacted by having a noisy neighbor on the system. CPU Manager for Kubernetes can be used along with the other Intel technology capabilities to achieve the improved network I/O, deterministic compute performance, and server platform sharing benefits offered by Intel® Xeon® Processor-based platforms.

An example pod requesting one exclusive core can be seen below:

```

apiVersion: v1
kind: Pod
metadata:
  name: pod-cmk-1
  annotations:
    cmk.intel.com/mutate: "true"
  namespace: kube-system
spec:
  serviceAccountName: cmk
  containers:
  - name: pod-cmk-1
    image: ubuntu:focal
    command:
    - "/bin/bash"
    - "-c"
    args:
    - "tail -f /dev/null"
  resources:
    requests:
      cmk.intel.com/exclusive-cores: '1'
    limits:
      cmk.intel.com/exclusive-cores: '1'

```

```

volumeMounts:
- mountPath: /opt/bin
  name: cmk-install-dir
volumes:
- hostPath:
  path: /opt/bin
  name: cmk-install-dir

```

After creating the pod, the core allocation from CPU Manager for Kubernetes can be seen in the pod:

```

# kubectl exec pod-cmk-1 -n kube-system -- /opt/bin/cmk isolate --pool=exclusive env | grep CMK
CMK_CPUS_ASSIGNED_MASK=200000000002
CMK_CPUS_ASSIGNED=1,49
CMK_PROC_FS=/host/proc
CMK_NUM_CORES=1
CMK_CPUS_INFRA=0,48,3,51,4,52,5,53,6,54,7,55,8,56,11,59,12,60,13,61,14,62,15,63,16,64,17,65,18,
66,19,67,20,68,21,69,22,70,23,71,24,72,25,73,26,74,27,75,28,76,29,77,30,78,31,79,32,80,33,81,34
,82,35,83,36,84,37,85,38,86,39,87,40,88,41,89,42,90,43,91,44,92,45,93,46,94,47,95

```

By default, CPU Manager for Kubernetes sets the affinity of the application. Some applications, e.g., DPDK-based ones, usually take the core allocation as an input parameter. These can be run by adding the '--no-affinity' option to CPU Manager for Kubernetes, and then reading the list of 'CMK\_CPUS\_ASSIGNED' and using these for core pinning.

For more details on usage, see: <https://github.com/intel/CPU-Manager-for-Kubernetes>

## 7.7 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 optimal performance<sup>13</sup>, required optimizations related to CPU isolation, memory, and device locality must be made. It is enabled by default starting with Kubernetes 1.19.8. Topology Manager is beta feature and is enabled by default.

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.
- **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 topologymanager
Dec 04 10:39:27 silpixa00390843 kubelet[9247]: I1204 10:39:27.305994 9247
topology_manager.go:92] [topologymanager] Creating topology manager with best-effort policy
Dec 04 10:39:27 silpixa00390843 kubelet[9247]: I1204 10:39:27.306005 9247
container_manager_linux.go:300] [topologymanager] Initializing Topology Manager with best-effort
policy

```

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

```
Dec 04 10:39:48 silpixa00390843 kubelet[9247]: I1204 10:39:48.050934 9247
topology_manager.go:308] [topologymanager] Topology Admit Handler
Dec 04 10:39:48 silpixa00390843 kubelet[9247]: I1204 10:39:48.050942 9247
topology_manager.go:317] [topologymanager] Pod QoS Level:
Dec 04 10:39:48 silpixa00390843 kubelet[9247]: I1204 10:39:48.050950 9247
topology_manager.go:332] [topologymanager] Topology Manager only affinitises Guaranteed pods.
Dec 04 10:39:48 silpixa00390843 kubelet[9247]: I1204 10:39:48.084236 9247
topology_manager.go:308] [topologymanager] Topology Admit Handler
Dec 04 10:39:48 silpixa00390843 kubelet[9247]: I1204 10:39:48.084251 9247
topology_manager.go:317] [topologymanager] Pod QoS Level: BestEffort
Dec 04 10:39:48 silpixa00390843 kubelet[9247]: I1204 10:39:48.084263 9247
topology_manager.go:332] [topologymanager] Topology Manager only affinitises Guaranteed pods.
```

### 7.7.1 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
```

### 7.7.2 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 node1
```

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
```

## 7.8 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.<sup>14</sup> The Kubernetes device plugin framework provides a vendor-independent solution for hardware devices. Intel has developed a set of device plugins that comply with the Kubernetes device plugin framework and allow users to request and consume hardware devices across Kubernetes clusters such as Intel® QuickAssist Technology, GPUs, 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>

### 7.8.1 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. These resources both 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 node1 -o json | jq '.status.allocatable'
{
  "cmk.intel.com/exclusive-cores": "2",
  "cpu": "93",
  "ephemeral-storage": "452220352993",
  "hugepages-1Gi": "4Gi",
```

<sup>14</sup> 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](http://www.Intel.com/PerformanceIndex). Results may vary.

```

"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"
}

```

In the above there are two SR-IOV Network Device Plugin resources: “intel.com/intel\_sriov\_dpdk\_710\_series” and “intel.com/intel\_sriov\_netdevice”. The “netdevice” VFs are bound 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.<sup>15</sup>

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/centos/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, but if the default configuration has been used, the SR-IOV CNI and example network attachment definition were installed and can be used as well.

```

# kubectl get net-attach-def
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 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/centos/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

```

<sup>15</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

```
# 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
    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
```

## 7.8.2 Check QAT Device Plugin

The Intel® QuickAssist Technology (QAT) Device Plugin discovers and exposes QAT device VFs as a consumable resource in Kubernetes<sup>16</sup>. 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 node1 -o json | jq '.status.allocatable'
{
  "cmk.intel.com/exclusive-cores": "2",
  "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"
}
```

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/centos/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 info and examples can be found here: <https://github.com/intel/intel-device-plugins-for-kubernetes/tree/master/demo>

## 7.8.3 Check SGX Device Plugin

The Intel® SGX plugin deployment and hardware support can be verified by inspecting the node resource allocations:

```
# kubectl describe node node1 | grep sgx.intel.com
nfd.node.kubernetes.io/extended-resources: sgx.intel.com/epc
sgx.intel.com/enclave: 20
sgx.intel.com/epc: 98566144
sgx.intel.com/provision: 20
```

<sup>16</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

```
sgx.intel.com/enclave: 20
sgx.intel.com/epc: 98566144
sgx.intel.com/provision: 20
sgx.intel.com/enclave 1 1
sgx.intel.com/epc 400 400
sgx.intel.com/provision 1 1
```

## 7.9 Check Networking Features (after Installation)

This section describes how to verify certain CNI plugins.

### 7.9.1 Check Multus CNI Plugin

To verify that 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 make sure to update the **{{ interface }}** value to match an interface name on the worker node(s) of the system, e.g., “ens786f1”.<sup>17</sup>

```
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
metadata:
  name: pod-macvlan-1
  annotations:
    k8s.v1.cni.cncf.io/networks: macvlan-multus-1
spec:
  containers:
  - name: pod-macvlan-1
    image: docker.io/centos/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
```

<sup>17</sup> Refer to <https://software.intel.com/articles/optimization-notice> for more information regarding performance and optimization choices in Intel software products.

```

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

```

## 7.9.2 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 virtual function (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 networks using the following command:

```

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

```

An example using the SR-IOV CNI Plugin can be found in [Section 7.8.1](#). 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/centos/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

```

## 7.9.3 Check Userspace CNI Plugin

The Userspace CNI is a Container Network Interface plugin designed to implement userspace networking such as DPDK-based applications. The current implementation supports the 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 environment.<sup>18</sup>

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

Verify the resource is available using the following command:

```

# kubectl get net-attach-def
NAME          AGE

```

<sup>18</sup> 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](http://www.Intel.com/PerformanceIndex). Results may vary.

```
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/centos/tools:latest
    command:
    - /sbin/init
    volumeMounts:
    - mountPath: /vhu/
      name: socket
  volumes:
  - name: socket
    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/
e4c7e6fb63ec737f7ae3f451e79f7fba2cac6d212b88fb0725da1b9afed1cfb
```

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`, it is `userspace-vpp`.

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

## 7.9.4 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.

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 increase 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'
"cmk.intel.com/exclusive-cores": "2",
"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 below:

```

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/centos/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>

## 7.10 Check Grafana Telemetry Visualization

BMRA deploys Grafana for telemetry visualization. It is available on every cluster node on port 30000. Default credentials are `admin/admin` and user is requested to change 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 kube-prometheus (<https://github.com/prometheus-operator/kube-prometheus>) project. Dashboards are available in the Dashboards -> Manage menu as shown in [Figure 12](#).

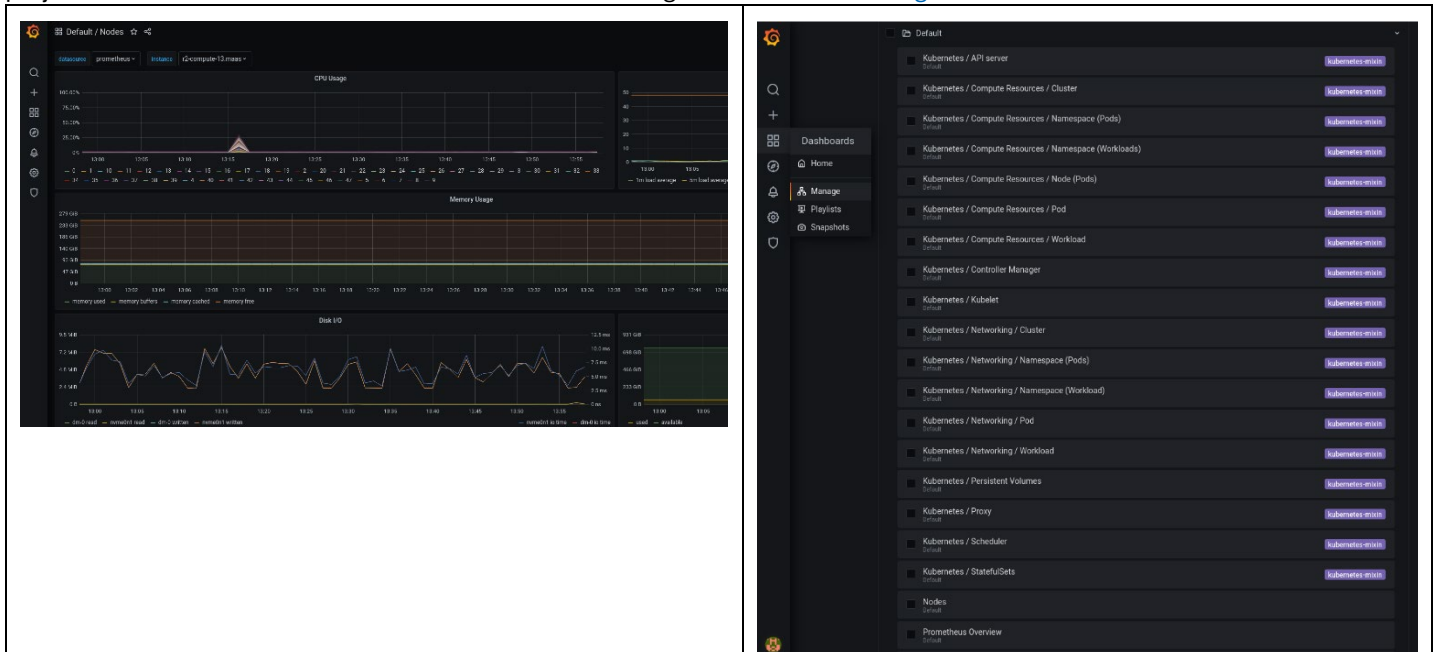


Figure 12. Grafana Dashboard Example

## 7.11 Check Telemetry Aware Scheduler

BMRA can deploy TAS Health Metric Demo Policy (<https://github.com/intel/telemetry-aware-scheduling/blob/master/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

```

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

The Health Metric Demo Policy requires a Prometheus metric file to be existent on node and 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 monitoring
```

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

```
kubectl describe taspolicies demo-policy -n monitoring
```

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

### 7.11.1 Check Dontschedule Policy

To change the value of the `node_health_metric` to `dontschedule`, use the following command:

```
echo 'node_health_metric 2' | ssh <user@worker> -T "cat /opt/intel/tas-demo-policy/test.prom"
```

Then, deploy a pod susceptible to the `dontschedule` policy, run the following command on the first controller node:

```
kubectl apply -f /usr/src/telemetry-aware-scheduling/deploy/health-metric-demo/demo-pod.yaml
```

The pod should not be deployed on node with `node_health_metric == 2`.

You can verify that TAS has been called to schedule the pod by looking at the logs:

```
kubectl logs pod/tas-telemetry-aware-scheduling-xxxx-yyyy -c tas-controller -n monitoring
```

Example output:

```
2019/08/19 15:30:59 NODE_B health_metric = 2
2019/08/19 15:30:59 NODE_A health_metric = 0
2019/08/19 15:30:59 NODE_A violating : health_metric Equals 2
2019/08/19 15:30:59 NODE_C health_metric = 0]
2019/08/19 15:30:59 Filtered nodes available for demo-policy : NODE_A NODE_C
```

### 7.11.2 Check Deschedule Policy

To see the impact of the descheduling policy, use a component called `descheduler`. For more details, visit

<https://github.com/intel/telemetry-aware-scheduling/blob/master/docs/health-metric-example.md#seeing-the-impact>

Set the `node_health_metric` to `deschedule` as follows:

```
echo 'node_health_metric 2' | ssh <user@worker> -T "cat /opt/intel/tas-demo-policy/test.prom"
```

Then run the descheduler with following command:

```
/usr/src/sigs.k8s.io/descheduler/_output/bin/descheduler --policy-config-file
/usr/src/telemetry-aware-scheduling/deploy/health-metric-demo/descheduler-policy.yaml --
kubeconfig /etc/kubernetes/admin.conf
```

The pod should be rescheduled onto a healthier node based on its TAS Policy. If no other suitable node is available, the new pod fails to schedule.

## 7.12 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 [Enabling Key Management NGINX Applications](#) for step-by-step instructions to set up and run the NGINX workload.

## 7.13 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, confirm the i915 driver presence and that the ASPEED device is blacklisted. The kernel option `915.force_probe=*` ensures the i915 driver probes for the SG1 device and `i915.enable_guc=2` ensures 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 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 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>.

## 8 Conclusion – Automation Eases Reference Application Deployment

This document contains notes on installation, configuration, and use of networking and device plug-in features for Kubernetes. By following this document, it is possible to set up a Kubernetes cluster and add simple configurations for some of the features provided by Intel. The playbook enables users to perform automated deployments, which decrease installation time from days to hours. The included example use cases show how the features can be consumed to provide additional functionality in both Kubernetes and the deployed pods, including but not limited to, flexible network configurations, Node Feature Discovery, and CPU pinning for exclusive access to host cores.

Intel and its partners have been working with open-source communities to add new techniques and address key barriers to networking adoption in Kubernetes for containers by harnessing the power of Intel® architecture-based servers to improve configuration, manageability, deterministic performance, network throughput, service-assurance, and resilience of container deployments.

We highly recommend that you take advantage of these advanced network features and device plug-ins in container-based NFV deployments.

# **Part 3:**

## **Build Your Reference Architecture**

# Appendix A

## BMRA Setup for All Configuration Profile Options

## Appendix A BMRA Setup for All Flavors and Configuration Profile Options

This appendix is relevant for all BMRA Flavors. 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 appendix from the following list in order to build the BMRA flavor:

- [Appendix B, BMRA Basic Configuration Profile Setup](#)
- [Appendix C, BMRA Full Configuration Profile Setup](#)
- [Appendix D, BMRA On-Premises Edge Configuration Profile Setup](#)
- [Appendix E, BMRA Remote CO-Forwarding Configuration Profile Setup](#)
- [Appendix F, BMRA Regional Data Center Configuration Profile Setup](#)

**Note:** The taxonomy for the BMRA Configuration Profile settings is defined in [Section 1.3](#).

### A.1 Set Up an Ansible Host

BMRA Kubernetes clusters require an Ansible Host that will store information about all the remote nodes managed. Ansible Host installation instructions are available in [Appendix G](#).

### A.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 20](#) and [Table 21](#).

**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 20. BIOS Prerequisites for Control and Worker Nodes for Basic and Full Configuration Profiles**

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

**Table 21. BIOS Prerequisites for Control and Worker Nodes for On-Premises Edge, Remote Co-Forwarding, and Regional Data Center Configuration Profiles**

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

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

The BIOS profile referenced in these tables consists of a number 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 four different BIOS profiles.

1. Energy Balance,
2. Max Performance,
3. Deterministic.
4. Max Performance with Low Latency

The configuration and values set per each BIOS profile are defined starting on [Table 17](#).

**Note:** The above values are the recommended configuration options on the Intel® S2600WFQ server board. 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.

### A.3 Software BOMs for all BMRA Configuration Profiles

The tables below list the software components and versions for all Configuration Profiles. Software settings are also listed, where applicable.

**Note:** The taxonomy for the BMRA Configuration Profile settings is defined in [Section 1.3](#).

**Table 22. Software BOMs for Basic and Full Configuration Profiles**

COMPONENT	BASIC CONFIGURATION PROFILE	FULL CONFIGURATION PROFILE
Kubernetes	1.19.8	1.19.8
Docker	19.03	19.03
Kubespray	2.14+	2.14+
cluster_name:	cluster.local	cluster.local
docker_registry	True	True
Multus	V3.4.2	V3.4.2
Topology Manager	1.18	1.18
K8s Native CPU Manager	N/A	N/A
Node Feature Discovery	0.7.0	0.7.0
CMK	N/A	1.5.1
GPU device plugin	N/A	False
SR-IOV device plugin	N/A	3.3.1
SR-IOV CNI	N/A	2.6
Bond CNI	N/A	1.0
QAT-DP	N/A	0.19.0
GPU-DP	N/A	False
SGX-DP	N/A	True
TAS	N/A	0.2
Userspace CNI	N/A	1.3
DDP	N/A	True
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True	True
SST BF	N/A	True (Available only on Ubuntu 20.04 and CentOS 8.3)
SST CP (3rd Generation Intel Xeon only)	N/A	True
OVS-DPDK	N/A	2.13.0
FD.IO VPP	N/A	False
collectd (*-with required plugins in this doc)	Telemetry (True)	True
NodeExporter	True	True
cAdvisor	True	True
Prometheus	False	True
Custom collector 1	N/A	N/A

COMPONENT	BASIC CONFIGURATION PROFILE	FULL CONFIGURATION PROFILE
Grafana	True	True
<b>General Telemetry Plugins</b>		
RAS memory plugin	N/A	N/A
Plugin:SMART	True	True
Plugin:numa	True	True
QAT plugin (experimental)	N/A	N/A
OVS plugin	False	N/A
Netlink Plugin	N/A	N/A
ethstats plugin	True	True
Intel PMU plugin	True	True
Intel RDT plugin	N/A	N/A
Hugepage plugin	False	N/A
Disk plugin	True	True
CPU Plugin	True	True
virt plugin	N/A	N/A
<b>Power Telemetry Plugins</b>		
dpdk telemetry plugin	False	N/A
Turbostat/pkgpower.py	True	True
IPMI	True	True
cpufreq	True	True
<b>General Worker Node Settings</b>		
isolcpus_enabled	False	True
isolcpus:	ex: "4-11"	ex: "4-11"
hugepages_enabled	N/A	True
default_huge_page_size	ex: 1G	ex: 1G
hugepages_1G	ex: 4	ex: 4
hugepages_2M	ex: 0	ex: 0
<b>SR-IOV NICs</b>		
iommu_enabled	False	True
update_nic_drivers	True	True
dataplane_interfaces	Empty	Empty
<b>QAT</b>		
update_qat_drivers	True	True
qat_devices	Empty	Empty
<b>DPDK</b>		
install_dpdk:	False	TRUE
DPDK version	19.11.6	19.11.6
dpdk_local_patches	Empty	Empty

Table 23. Software BOMs for On-Premises Edge, Remote Co-Forwarding, and Regional Data Center Configuration Profiles

COMPONENT	ON-PREMISES EDGE CONFIGURATION PROFILE	REMOTE CO-FORWARDING CONFIGURATION PROFILE	REGIONAL DATA CENTER CONFIGURATION PROFILE
Kubernetes	1.19.8	1.19.8	1.19.8
Docker	19.03	19.03	19.03
Kubespray	2.14+	2.14+	2.14+
cluster_name:	cluster.local	cluster.local	cluster.local
docker_registry	True	True	True
Multus	V3.4.2	V3.4.2	V3.4.2
Topology Manager	1.18	1.18	1.18
K8s Native CPU Manager	N/A	N/A	TRUE
Node Feature Discovery	0.7.0	0.7.0	0.7.0
CMK	1.5.1	1.5.1	N/A
GPU device plugin	N/A	N/A	TRUE
SR-IOV device plugin	3.3.1	3.3.1	TRUE
SR-IOV CNI	2.6	2.6	TRUE
Bond CNI	False	False	N/A
QAT-DP	0.19.0	False	N/A
TAS	0.2	0.2	FALSE
Userspace CNI	N/A	False	N/A

**Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor**

COMPONENT	ON-PREMISES EDGE CONFIGURATION PROFILE	REMOTE CO-FORWARDING CONFIGURATION PROFILE	REGIONAL DATA CENTER CONFIGURATION PROFILE
DDP	N/A	True	N/A
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True	True	N/A
SST BF	False	False	N/A
SST CP (3rd Generation Intel Xeon only)	0	0	N/A
OVS-DPDK	False	False	N/A
FD.IO VPP	False	False	N/A
collectd (*-with required plugins in this doc)	True	True	True
NodeExporter	True	True	True
cAdvisor	True	True	True
Prometheus	False	False	True
Custom collector 1	N/A	N/A	True
Grafana	True	True	False
<b>General Telemetry Plugins</b>			
RAS memory plugin	N/A	N/A	True
Plugin:SMART	True	True	True
Plugin:numa	True	True	True
QAT plugin (experimental)	N/A	N/A	N/A
OVS plugin	False	False	False
Netlink Plugin	N/A	N/A	N/A
ethstats plugin	True	True	True
Intel PMU plugin	True	True	N/A
Intel RDT plugin	N/A	N/A	N/A
Hugepage plugin	False	False	True
Disk plugin	True	True	True
CPU Plugin	True	True	True
virt plugin	N/A	N/A	N/A
<b>Power Telemetry Plugins</b>			
dpdk telemetry plugin	False	False	False
Turbostat/pkgpower.py	True	True	True
IPMI	True	True	True
cpufreq	True	True	True
<b>General Worker Node Settings</b>			
isolcpus_enabled	False	True	False
isolcpus:	ex: "4-11"	ex: "4-11"	ex: "4-11"
hugepages_enabled	N/A	True	True
default_huge_page_size	ex: 1G	ex: 1G	ex: 1G
hugepages_1G	ex: 4	ex: 4	ex: 4
hugepages_2M	ex: 0	ex: 0	ex: 0
<b>SR-IOV NICs</b>			
iommu_enabled	True	True	True
update_nic_drivers	True	True	True
dataplane_interfaces	Empty	Empty	Empty
<b>QAT</b>			
update_qat_drivers	True	True	True
qat_devices	Empty	Empty	Empty
<b>DPDK</b>			
install_dpdk	TRUE	TRUE	TRUE
DPDK version	19.11.6	19.11.6	19.11.6
dpdk_local_patches	Empty	Empty	Empty

# **Appendix B**

## **BMRA Basic**

### **Configuration Profile Setup**

## Appendix B BMRA Basic Configuration Profile Setup

This appendix 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 [B.1](#) for details. You also need to build your Kubernetes cluster. [Figure 1](#) is an example.
2. Download the Ansible playbook for your Configuration Profile. Refer to [B.2](#) for details.
3. Set up the optional Ansible parameters using the info in the Configuration Profile tables. Refer to [B.3](#) for details.
4. Deploy the platform. Refer to [B.4](#) for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) 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 appendix.

TERM	DESCRIPTION
<b>Hardware Taxonomy</b>	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, etc.)
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.
<b>Software Taxonomy</b>	
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.

### B.1 Step 1 - Set Up Basic Configuration Profile Hardware

This section describes the hardware BOM and the BIOS configuration recommendation for using the BMRA Basic Configuration Profile Flavor.

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 24. Hardware Setup for Basic Configuration Profile**

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	<a href="#">Controller_2ndGen_1</a>	<a href="#">Controller_3rdGen_1</a>
Worker Node Options	<a href="#">Worker_2ndGen_Base_1</a>	<a href="#">Worker_3rdGen_Base_1</a>

### B.2 Step 2 - Download Basic Configuration Profile Ansible Playbook

This section contains step-by-step 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 following steps:

1. Login to your Ansible host (the one that you will run these Ansible playbooks from).
2. 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 19](#). For example:

```
git checkout v21.03
```

3. Export the environmental variable for Kubernetes **Basic** Configuration Profile deployment:

```
export PROFILE=basic
```

4. Copy the example inventory file to the project root dir.

```
cp examples/${PROFILE}/inventory.ini .
```

5. Copy the example configuration files to the project root dir:

```
cp -r examples/${PROFILE}/group_vars examples/${PROFILE}/host_vars .
```

### B.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 with all components listed in [Section B.2.2](#). Every capability included in the Basic 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 Basic Configuration Profile.

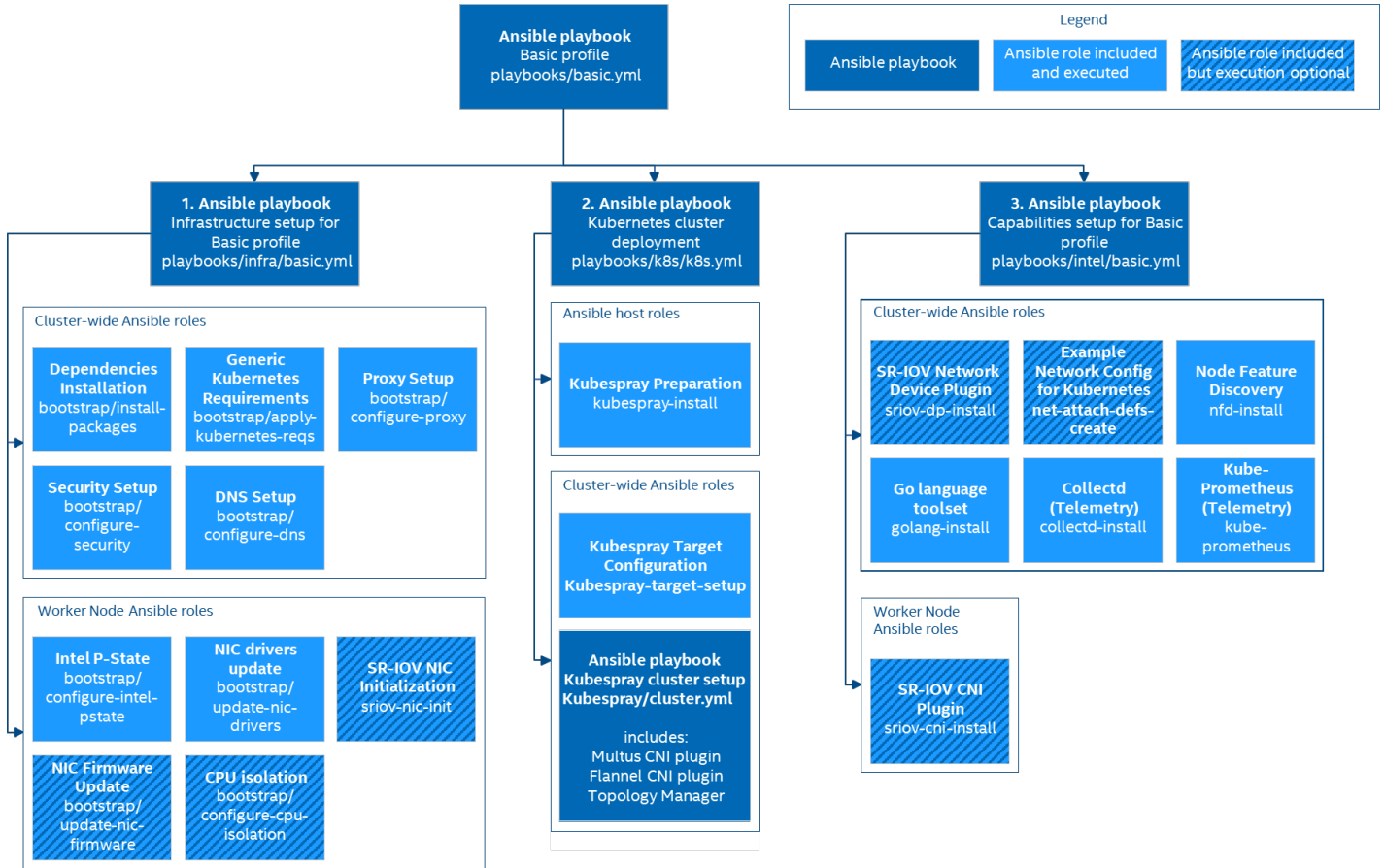


Figure 13. Basic Configuration Profile Ansible Playbook

### B.2.2 Basic Configuration Profile Software BOM

The table below lists the software components and versions that are automatically installed when the Basic Configuration Profile playbook is deployed. Software settings are also listed, where applicable.

Table 25. Software BOM for Basic Configuration Profile

COMPONENT	BASIC CONFIGURATION PROFILE
Kubernetes	1.19.8
Docker	19.03
Kubespray	2.14+
cluster_name:	cluster.local
docker_registry	True
Multus	V3.4.2
Topology Manager	1.18
K8s Native CPU Manager	N/A
Node Feature Discovery	0.7.0
CMK	N/A
GPU device plugin	N/A
SR-IOV device plugin	N/A
SR-IOV CNI	N/A
Bond CNI	N/A

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

COMPONENT	BASIC CONFIGURATION PROFILE
QAT-DP	N/A
TAS	N/A
Userspace CNI	N/A
DDP	N/A
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True
SST BF	N/A
SST CP (3rd Generation Intel Xeon only)	0
OVS-DPDK	N/A
FD.IO VPP	N/A
collectd (*-with required plugins in this doc)	Telemetry (True)
NodeExporter	True
cAdvisor	True
Prometheus	False
Custom collector 1	N/A
Grafana	True
<b>General Telemetry Plugins</b>	
RAS memory plugin	N/A
Plugin:SMART	True
Plugin:numa	True
QAT plugin (experimental)	N/A
OVS plugin	False
Netlink Plugin	N/A
ethstats plugin	True
Intel PMU plugin	True
Intel RDT plugin	N/A
Hugepage plugin	False
Disk plugin	True
CPU Plugin	True
virt plugin	N/A
<b>Power Telemetry Plugins</b>	
dpdk telemetry plugin	False
Turbostat/pkgpower.py	True
IPMI	True
cpufreq	True
pkgpower.py (*- see above)	True
<b>GRUB Settings</b>	
isolcpus_enabled	False
hugepages_enabled	N/A
<b>Default Configuration</b>	
isolcpus:	ex: "4-11"
default_huge_page_size	ex: 1G
hugepages_1G	ex: 4
hugepages_2M	ex: 0
<b>SR-IOV NICs</b>	
iommu_enabled	False
<b>Default Configuration</b>	
"- name:"	enp24s0f0
sriov_numvfs	2
vf_driver	vfio-pci
ddp_profile	"gtp.pkgo"
"- name:"	enp24s0f1
sriov_numvfs	4
vf_driver:	iavf
<b>QAT</b>	
Dependency: qat-dp Enabled	N/A
<b>Default Configuration</b>	
qat_sriov_numvfs	n/a
<b>DPDK</b>	

COMPONENT	BASIC CONFIGURATION PROFILE
DPDK version	19.11.6
Dependency: sriov Enabled	False
<b>Default Configuration</b>	
install_dpdk	TRUE

### B.3 Step 3 - Set Up Basic Configuration Profile

Review the optional Ansible group and host variables in this section and select the 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 [B.3.1](#) and [B.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 full set of configuration variables for the Basic Configuration Profile along with their default values can be found in `examples/basic` 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 – their scope is limited to a single worker node.

The tables below contain a full list of group and host variables. All these variables are important, but pay special attention to the variables in **bold** as they almost always need to be updated to match the target environment.

#### B.3.1 Basic Configuration Profile Group Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Common cluster configuration</b>			
Kubernetes	Boolean	true	Specifies whether to deploy Kubernetes or not.
kube_version	String (semver)	v1.19.8	Kubernetes version.
update_all_packages	Boolean	false	Runs system-wide package update (apt dist-upgrade, yum update, ...). Tip: Can be set using <code>host_vars</code> for more granular control.
<b>http_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTP proxy address. Comment out if your cluster is not behind proxy.</b>
<b>https_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTPS proxy address. Comment out if your cluster is not behind proxy.</b>
<b>additional_no_proxy</b>	<b>Comma-separated list of addresses</b>	<b>.example.com</b>	<b>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 <code>no_proxy</code> automatically.</b>
kube_pods_subnet	CIDR	10.244.0.0/16	Kubernetes pod subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overall with your corporate LAN.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
kube_service_addresses	CIDR	10.233.0.0/18	Kubernetes service subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
cluster_name	DNS domain	cluster.local	Name of the cluster
always_pull_enabled	Boolean	true	Set image pull policy to Always. Pulls images before starting containers. Valid credentials must be configured.
<b>Node Feature Discovery</b>			
nfd_enabled	Boolean	true	Specifies whether to deploy Node Feature Discovery.
nfd_build_image_locally	Boolean	false	Builds NFD Docker 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 will query node status and update node labels.
<b>Topology Manager (Kubernetes built-in)</b>			
topology_manager_enabled	Boolean	true	Enables Kubernetes built-in Topology Manager
topology_manager_policy	String, options: none, best-effort, restricted, single-numa-node	best-effort	Topology Manager policy.
<b>Intel SR-IOV Network Device Plugin</b>			
sriov_net_dp_enabled	Boolean	false	Enables SR-IOV Network Device Plugin
sriov_net_dp_namespace	String	kube-system	Kubernetes namespace that will be used to deploy SR-IOV Network Device Plugin
sriov_net_dp_build_image_locally	Boolean	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 configuration refer to: <a href="https://github.com/intel/sriov-network-device-plugin#configurations">https://github.com/intel/sriov-network-device-plugin#configurations</a>
<b>Example Network Attachment Definitions (ready to use examples of custom CNI plugin configuration)</b>			
example_net_attach_defs. userspace_ovs_dpdk	Boolean	false	Example net-attach-def for Userspace CNI with OVS-DPDK

### B.3.2 Basic Configuration Profile Host Variables<sup>19</sup>

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>SR-IOV and network devices configuration</b>			
iommu_enabled	Boolean	false	Sets up SR-IOV related kernel parameters and enables further SR-IOV configuration.
dataplane_interfaces	List of dictionaries	n/a	<b>SR-IOV related NIC configuration using per-port approach</b>

<sup>19</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<code>dataplane_interfaces[*].name</code>	String	<code>enp24s0f0, enp24s0f1</code>	Name of the interface representing PF port
<code>dataplane_interfaces[*].bus_info</code>	String (PCI address)	<code>18:00.0, 18:00.1</code>	PCI address of the PF port
<code>dataplane_interfaces[*].pf_driver</code>	String	<code>i40e</code>	PF driver, "i40e", "ice"
<code>dataplane_interfaces[*].sriov_numvfs</code>	Integer	<code>6, 4</code>	Number of VFs to be created, associated with the PF
<code>dataplane_interfaces[*].default_vf_driver</code>	String, options: "i40evf", "iavf", "vfiopci", "igb_uio"	<code>vfio-pci for DPDK, iavf for kernel network stack</code>	Default driver module name that the VFs will be bound to
<code>dataplane_interfaces[*].sriov_vfs[*]</code>	List of dictionaries	n/a	List of vfs to create with specific driver (non-default)
<code>update_nic_drivers</code>	Boolean	<code>true</code>	Set to 'true' to update Linux kernel drivers for Intel Network Adapters
<code>update_nic_firmware</code>	Boolean	<code>false</code>	# Set 'true' to update Network Adapter firmware
<code>firmware_update_nics</code>	List of strings	<code>[enp24s0f0, enp24s0f1]</code>	Additional list of Network Adapters that the FW update will be executed on. Note: FW update will be also executed on all Network Adapters listed in "dataplane_interfaces[*].name"
<code>install_dpdk</code>	Boolean	<code>false</code>	DPDK installation is required for <code>sriov_cni_enabled:true</code>
<code>dpdk_version</code>	String	<code>19.11.6</code>	DPDK version to install
<code>dpdk_local_patches_dir</code>	String	Empty	Path to user supplied patches to apply against the specified version of DPDK
<b>SR-IOV CNI plugin</b>			
<code>sriov_cni_enabled</code>	Boolean	<code>false</code>	Installs SR-IOV CNI plugin binary on the node.
<b>CPU configuration</b>			
<code>isolcpus_enabled</code>	Boolean	<code>false</code>	Enables CPU cores isolation from Linux scheduler
<code>isolcpus</code>	Comma-separated list of CPU cores/ranges	<code>4-11</code>	CPU cores isolated from Linux scheduler, if CMK is enabled it's a good practice to match it with total number of shared and exclusive cores
<code>intel_pstate</code>	String, options: enabled, disabled	<code>enabled</code>	Enables Intel P-state scaling driver
<b>Miscellaneous</b>			
<code>dns_disable_stub_listener</code>	Boolean	<code>true</code>	(Ubuntu only) Disables DNS stub listener from the systemd-resolved service, which is known to cause problems with DNS and Docker containers on Ubuntu

## B.4 Step 4 - Deploy Basic Configuration Profile Platform

**Note:** You must download the Configuration Profile playbook as described in [B.2](#) and set it up as described in [B.3](#) before you complete this step.

In order to deploy the Basic Configuration Profile playbook, change the working directory to where you have cloned or unarchived the BMRA Ansible Playbook source code (as described in [Section 3.3.2](#)) and execute the command below:

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

## B.5 Step 5 - Validate Basic Configuration Profile

Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) and run the validation processes according to the hardware and software components that you have installed.

# Appendix C

## BMRA Full

### Configuration Profile Setup

## Appendix C BMRA Full Configuration Profile Setup

This appendix 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 [C.1](#) for details. You also need to build your Kubernetes cluster. [Figure 1](#) is an example.
2. Download the Ansible playbook for your Configuration Profile. Refer to [C.2](#) for details.
3. Configure the optional Ansible parameters using the info in the Configuration Profile tables. Refer to [C.3](#) for details.
4. Deploy the platform. Refer to [C.4](#) for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) 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 appendix.

TERM	DESCRIPTION
<b>Hardware Taxonomy</b>	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, etc.)
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.
<b>Software Taxonomy</b>	
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.

### C.1 Step 1 - Set Up Full Configuration Profile Hardware

This section describes the hardware BOM and the BIOS configuration recommendation for using the BMRA Full Configuration Profile Flavor.

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 26. Hardware Setup for Full Configuration Profile**

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	<a href="#">Controller_2ndGen_3</a>	<a href="#">Controller_3rdGen_3</a>
Worker Node Options	<a href="#">Worker_2ndGen_Plus_1</a>	<a href="#">Worker_3rdGen_Plus_1</a>

### C.2 Step 2 - Download Full Configuration Profile Ansible Playbook

This section contains step-by-step 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 following steps:

1. Login to your Ansible host (the one that you will run these Ansible playbooks from).
2. 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 19](#). For example:

```
git checkout v21.03
```

3. Export the environmental variable for Kubernetes **Full** Configuration Profile deployment:

```
export PROFILE=full_nfv
```

4. Copy the example inventory file to the project root dir.

```
cp examples/${PROFILE}/inventory.ini .
```

5. Copy the example configuration files to the project root dir:

```
cp -r examples/${PROFILE}/group_vars examples/${PROFILE}/host_vars .
```

### C.2.1 Full Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Full Configuration Profile allows you to provision a production-ready Kubernetes cluster with all components listed in [Section C.2.2](#). 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 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 Full Configuration Profile.

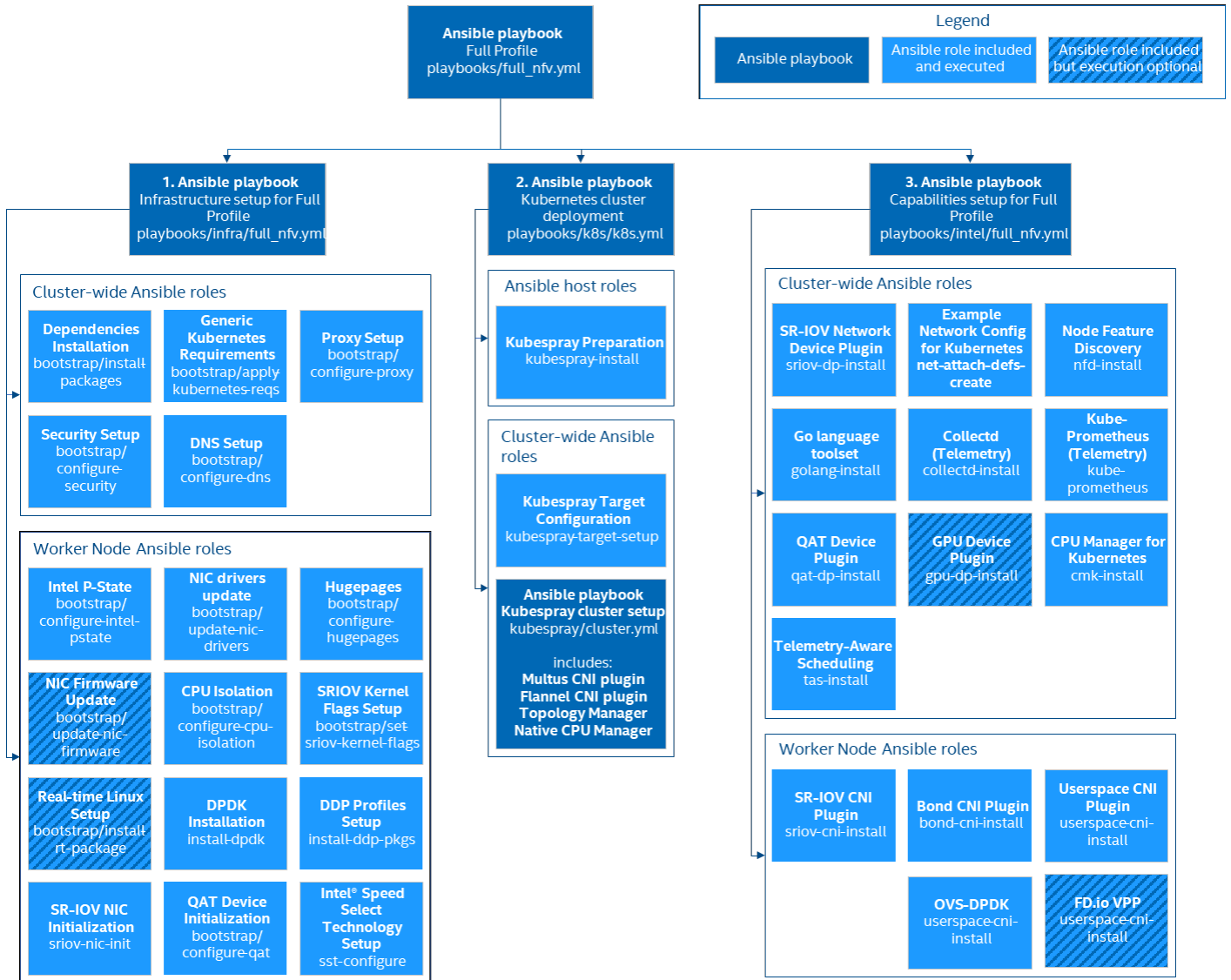


Figure 14. Full Configuration Profile Ansible Playbook

### C.2.2 Full Configuration Profile Software BOM

The table below lists the software components and versions that are automatically installed when the Full Configuration Profile playbook is deployed. Software settings are also listed, where applicable.

Table 27. Software BOM for Full Configuration Profile

COMPONENT	FULL CONFIGURATION PROFILE
Kubernetes	1.19.8
Docker	19.03
Kubespray	2.14+
cluster_name:	cluster.local

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

COMPONENT	FULL CONFIGURATION PROFILE
docker_registry	True
Multus	V3.4.2
Topology Manager	1.18
K8s Native CPU Manager	N/A
Node Feature Discovery	0.7.0
CMK	1.5.1
GPU device plugin	False
SR-IOV device plugin	3.3.1
SR-IOV CNI	2.6
Bond CNI	1.0
QAT-DP	0.19.0
TAS	0.2
Userspace CNI	1.3
DDP	True
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True
SST BF	True (Available only on Ubuntu 20.04 and CentOS 8.2)
SST CP (3rd Generation Intel Xeon only)	0
OVS-DPDK	2.13.0
FD.IO VPP	False
collectd (*-with required plugins in this doc)	True
NodeExporter	True
cAdvisor	True
Prometheus	True
Custom collector 1	N/A
Grafana	True
<b>General Telemetry Plugins</b>	
RAS memory plugin	N/A
Plugin:SMART	True
Plugin:numa	True
QAT plugin (experimental)	N/A
OVS plugin	N/A
Netlink Plugin	N/A
ethstats plugin	True
Intel PMU plugin	True
Intel RDT plugin	N/A
Hugepage plugin	N/A
Disk plugin	True
CPU Plugin	True
virt plugin	N/A
<b>Power Telemetry Plugins</b>	
dpdk telemetry plugin	N/A
Turbostat/pkgpower.py	True
IPMI	True
cpufreq	True
pkgpower.py (*- see above)	True
<b>GRUB Settings</b>	
isolcpus_enabled	True
hugepages_enabled	True
<b>Default Configuration</b>	
isolcpus:	ex: "4-11"
default_huge_page_size	ex: 1G
hugepages_1G	ex: 4
hugepages_2M	ex: 0
<b>SR-IOV NICs</b>	
iommu_enabled	True
<b>Default Configuration</b>	
"- name:"	enp24s0f0
sriov_numvfs	2

COMPONENT	FULL CONFIGURATION PROFILE
vf_driver	vfio-pci
ddp_profile	"gtp.pkgo"
"- name:"	enp24s0f1
sriov_numvfs	4
vf_driver	iavf
<b>QAT</b>	
Dependency: qat-dp Enabled	True
<b>Default Configuration</b>	
qat_sriov_numvfs	16
<b>DPDK</b>	
DPDK version	19.11.6
Dependency: sriov Enabled	True
<b>Default Configuration</b>	
install_dpdk:	TRUE

### C.3 Step 3 - Set Up Full Configuration Profile

Review the optional Ansible group and host variables in this section and select the 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 [C.3.1](#) and [C.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 Full Configuration Profile along with their default values can be found in the `examples/full_nfv` directory.

Variables are grouped into two main categories:

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

Tables below contain a full list of group and host variables. All these variables are important, but pay special attention to the variables in **bold** as they almost always need to be updated to match the target environment.

#### C.3.1 Full Configuration Profile Group Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Common cluster configuration</b>			
Kubernetes	Boolean	true	Specifies whether to deploy Kubernetes or not.
kube_version	String (semver)	v1.19.8	Kubernetes version.
update_all_packages	Boolean	false	Runs system-wide package update (apt dist-upgrade, yum update, ...). Tip: Can be set using <code>host_vars</code> for more granular control.
<b>http_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTP proxy address. Comment out if your cluster is not behind proxy.</b>
<b>https_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTPS proxy address. Comment out if your cluster is not behind proxy.</b>

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
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_pods_subnet	CIDR	10.244.0.0/16	Kubernetes pod subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
kube_service_addresses	CIDR	10.233.0.0/18	Kubernetes service subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
cluster_name	DNS domain	cluster.local	Name of the cluster
always_pull_enabled	Boolean	true	Set image pull policy to Always. Pulls images before starting containers. Valid credentials must be configured.
<b>Node Feature Discovery</b>			
nfd_enabled	Boolean	true	Specifies whether to deploy Node Feature Discovery.
nfd_build_image_locally	Boolean	false	Builds NFD Docker 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 will query node status and update node labels.
<b>Intel CPU Manager for Kubernetes<sup>20</sup></b>			
cmk_enabled	Boolean	true	Enables Intel CPU Manager for Kubernetes.
cmk_namespace	String	kube-system	Kubernetes namespace used for CMK deployment.
cmk_use_all_hosts	Boolean	false	Whether to deploy CMK on all nodes in the cluster (including control nodes).
cmk_hosts_list	Comma-separated strings	node1,node2	Comma-separated list of K8s worker node names that the CMK will run on.
cmk_shared_num_cores	Integer	2	Number of CPU cores to be assigned to the "shared" pool on each of the nodes
cmk_exclusive_num_cores	Integer	2	Number of CPU cores to be assigned to the "exclusive" pool on each of the nodes
cmk_shared_mode	String, options: packed, spread	packed	Shared pool allocation mode
cmk_exclusive_mode	String, options: packed, spread	packed	Exclusive pool allocation mode
<b>Native built-in Kubernetes CPU manager</b>			
native_cpu_manager_enabled	Boolean	false	Enabling CMK and built-in CPU Manager is not recommended. Setting this option as "true" enables the "static" policy, otherwise the default "none" policy is used.

<sup>20</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
native_cpu_manager_system_reserved_cpus	Kubernetes millicores	2000m	Number of CPU cores that will be reserved for housekeeping (2000m = 2000 millicores = 2 cores)
native_cpu_manager_kube_reserved_cpus	Kubernetes millicores	1000m	Number of CPU cores that will 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.
<b>Topology Manager (Kubernetes built-in)<sup>21</sup></b>			
topology_manager_enabled	Boolean	true	Enables Kubernetes built-in Topology Manager
topology_manager_policy	String, options: none, best-effort, restricted, single-numa-node	best-effort	Topology Manager policy.
<b>Intel SR-IOV Network Device Plugin</b>			
sriov_net_dp_enabled	Boolean	true	Enables SR-IOV Network Device Plugin
sriov_net_dp_namespace	String	kube-system	Kubernetes namespace that will be used to deploy SR-IOV Network Device Plugin
sriov_net_dp_build_image_locally	Boolean	true	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 configuration refer to: <a href="https://github.com/intel/sriov-network-device-plugin#configurations">https://github.com/intel/sriov-network-device-plugin#configurations</a>
<b>Intel Device Plugins for Kubernetes</b>			
qat_dp_enabled	Boolean	true	Enables Intel QAT Device Plugin
qat_dp_namespace	String	kube-system	Namespace used for Intel QAT Device Plugin
gpu_dp_enabled	Boolean	true	Enables Intel GPU Device Plugin
gpu_dp_namespace	String	kube-system	Namespace used for Intel GPU Device Plugin
<b>Intel Telemetry Aware Scheduling</b>			
tas_enabled	Boolean	true	Enables Intel Telemetry Aware Scheduling
tas_namespace	String	default	Kubernetes namespace used for TAS deployment
tas_create_policy	Boolean	false	Creates demo TAS policy
<b>Example Network Attachment Definitions (ready to use examples of custom CNI plugin configuration)</b>			
example_net_attach_defs.userspace_ovs_dpdk	Boolean	true	Example net-attach-def for Userspace CNI with OVS-DPDK
example_net_attach_defs.userspace_vpp	Boolean	false	Example net-attach-def for Userspace CNI with VPP
example_net_attach_defs.sriov_net_dp	Boolean	true	Example net-attach-def for SR-IOV Net DP and SR-IOV CNI

<sup>21</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

## C.3.2 Full Configuration Profile Host Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>SR-IOV and network devices configuration</b>			
iommu_enabled	Boolean	true	Sets up SR-IOV related kernel parameters and enables further SR-IOV configuration.
dataplane_interfaces	List of dictionaries	n/a	<b>SR-IOV related NIC configuration using per-port approach</b>
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	i40e	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 will be 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. <b>Note: Use only for the port 0 of the Network Adapter (PCI address ending with :00.0)</b>
update_nic_drivers	Boolean	true	Set to 'true' to update Linux kernel drivers for Intel Network Adapters
update_nic_firmware	Boolean	false	# Set 'true' to update Network Adapter firmware
firmware_update_nics	List of strings	[enp24s0f0, enp24s0f1]	<b>Additional list of Network Adapter interfaces that the FW update will be executed on. Note: FW update will be also executed on all Network Adapters listed in "dataplane_interfaces[*].name"</b>
install_ddp_packages	Boolean	true	Install Intel X700 & X800 series Network Adapters DDP packages. Required if DDP packages configured in dataplane_interfaces.
install_dpdk	Boolean	true	DPDK installation is required for sriov_cni_enabled:true
dpdk_version	String	19.11.6	DPDK version to install
dpdk_local_patches_dir	String	Empty	Path to user supplied patches to apply against the specified version of DPDK
<b>SR-IOV and Bond CNI plugins</b>			
sriov_cni_enabled	Boolean	true	Installs SR-IOV CNI plugin binary on the node.
bond_cni_enabled	Boolean	true	Installs Bond CNI plugin binary on the node.
<b>Userspace networking plugins and accelerated virtual switches</b>			
userspace_cni_enabled	Boolean	true	Installs Userspace CNI plugin binary on the node.
ovs_dpdk_enabled	Boolean	true	Installs OVS-DPDK on the node.

**Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor**

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
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	false	Installs FD.io VPP (CentOS 7 and Ubuntu 18.04 only)
<b>Hugepages/memory configuration</b>			
hugepages_enabled	Boolean	true	Enables hugepages support
default_hugepage_size	String, options: 2M, 1G	1G	Default hugepage size
hugepages_1G	Integer	4	Sets how many hugepages of 1G size should be created
hugepages_2M	Integer	0	Sets how many hugepages of 2M size should be created
<b>CPU configuration</b>			
isolcpus_enabled	Boolean	true	Enables CPU cores isolation from Linux scheduler
isolcpus	Comma-separated list of CPU cores/ranges	411	CPU cores isolated from Linux scheduler, if CMK is enabled it's a good practice to match it with total number of shared and exclusive cores
intel_pstate	String, options: enabled, disabled	enabled	Enables Intel P-state scaling driver
sst_bf_configuration_enabled	Boolean	true	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	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	False	Prioritize (SST-CP) power flow to high frequency cores
sst_cp_configuration_enabled	Boolean	False	Enables Intel SST Core Priority technology on 3rd Generation Intel® Xeon®. SST-CP overrides any 'SST-BF configuration'
sst_cp_priority_type	Integer	0	0 – proportional 1 - ordered
sst_cp_clos_groups	List of dictionaries	n/a	Allows for configuration of up to 4 CLOS groups including id, frequency_weight, min_MHz, max_MHz
sst_cp_cpu_clos	List of dictionaries	n/a	Allows for definition of cpu cores per close group
<b>Miscellaneous</b>			
dns_disable_stub_listener	Boolean	true	(Ubuntu only) Disables DNS stub listener from the systemd-resolved service, which is known to cause problems with DNS and Docker containers on Ubuntu

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
install_real_time_package	Boolean	false	(CentOS 7 only) Installs real-time Linux kernel packages.
<b>QAT configuration</b>			
qat_devices	List of dictionaries	n/a	SR-IOV related QAT configuration using per-port approach
qat_devices[*].qat_dev	String	Crypto01, Crypto02, Crypto03	Name of the interface representing PF port
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[*].module_type	String	qat_c62x	QAT hardware identifier, qat_c62x, qat_dh895xcc, qat_c3xxx, etc...
qat_devices[*].pci_type	String	c6xx	PF driver, "c6xx", "c3xx", "d15xx", etc...
qat_devices[*].qat_sriov_numvfs	Integer	10	Number of VFs to be created per QAT device physical function

#### C.4 Step 4 - Deploy Full Configuration Profile Platform

**Note:** You must download the Configuration Profile playbook as described in [C.2](#) and configure it as described in [C.3](#) before you complete this step.

In order to deploy the Full Configuration Profile playbook, change the working directory to where you have cloned or unarchived the BMRA Ansible Playbook source code (as described in [Section 3.3.2](#)) and execute the command below:

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

#### C.5 Step 5 - Validate Full Configuration Profile

Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) and run the validation processes according to the hardware and software components that you have installed.

# Appendix D

## BMRA On-Premises Edge Configuration Profile Setup

## Appendix D BMRA On-Premises Edge Configuration Profile Setup

This appendix 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 [D.1](#) for details.  
You also need to build your Kubernetes cluster. [Figure 1](#) is an example.
2. Download the Ansible playbook for your Configuration Profile. Refer to [D.2](#) for details.
3. Configure the optional Ansible parameters using the info in the Configuration Profile tables. Refer to [D.3](#) for details.
4. Deploy the platform. Refer to [D.4](#) for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) 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 appendix.

TERM	DESCRIPTION
<b>Hardware Taxonomy</b>	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, etc.)
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.
<b>Software Taxonomy</b>	
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.

### D.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 28. Hardware Setup for On-Premises Edge Configuration Profile**

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	<a href="#">Controller_2ndGen_1</a>	<a href="#">Controller_3rdGen_1</a>
Worker Node Options	<a href="#">Worker_2ndGen_Base_2</a>	<a href="#">Worker_3rdGen_Base_2</a>
	or <a href="#">Worker_2ndGen_Plus_1</a>	or <a href="#">Worker_3rdGen_Plus_1</a>

### D.2 Step 2 - Download On-Premises Edge Configuration Profile Ansible Playbook

This section contains step-by-step 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 following steps:

1. Login to your Ansible host (the one that you will run these Ansible playbooks from).
2. 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 19](#). For example:

```
git checkout v21.03
```

3. Export the environmental variable for Kubernetes **On-Premises Edge** Configuration Profile deployment:

```
export PROFILE=on_prem
```

4. Copy the example inventory file to the project root dir.

```
cp examples/${PROFILE}/inventory.ini .
```

5. Copy the example configuration files to the project root dir:

```
cp -r examples/${PROFILE}/group_vars examples/${PROFILE}/host_vars .
```

### D.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 with all components listed in [Section D.2.2](#). 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 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 On-Premises Edge Configuration Profile.

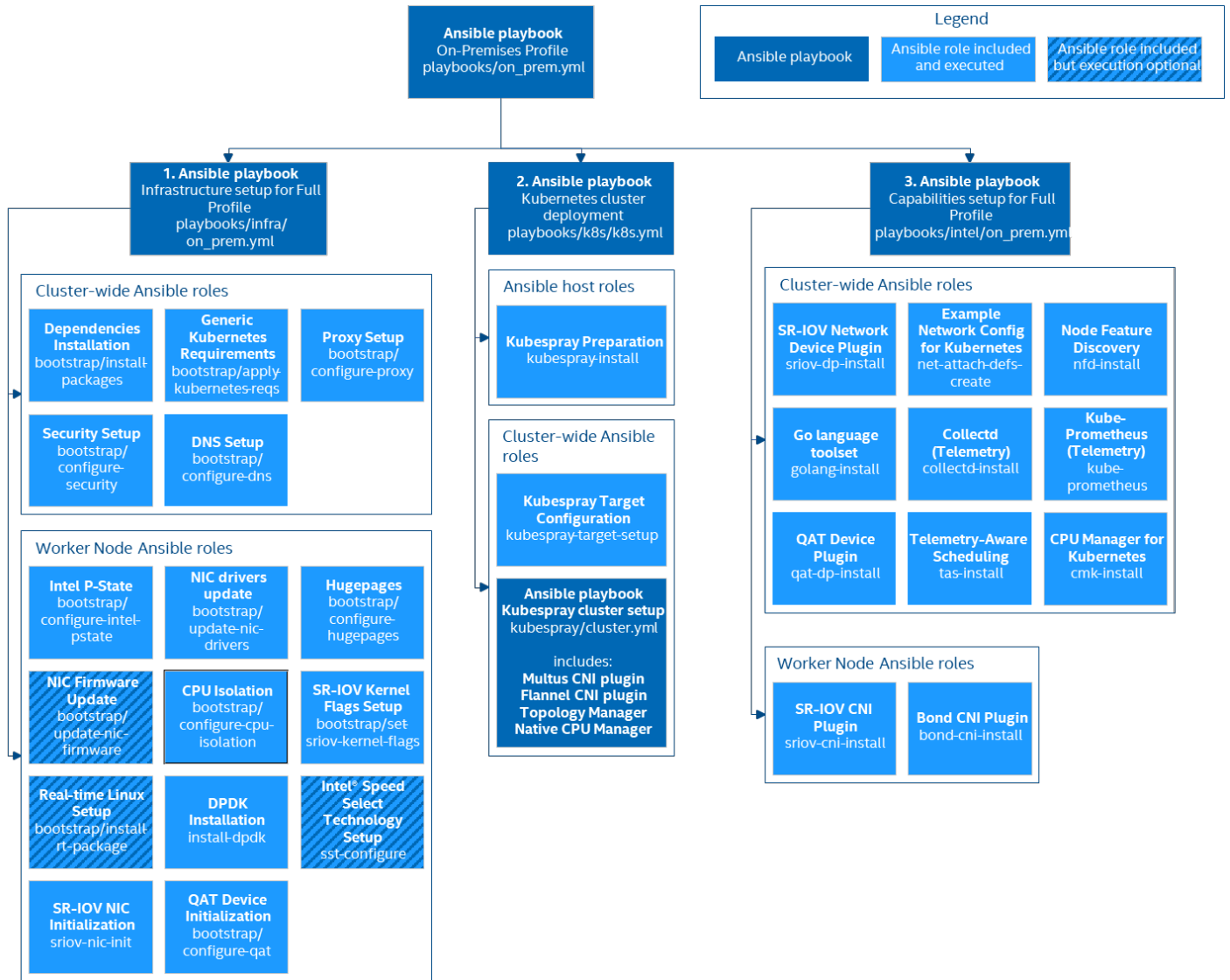


Figure 15. On-Premises Edge Configuration Profile Ansible Playbook

### D.2.2 On-Premises Edge Configuration Profile Software BOM

The table below lists the software components and versions that are automatically installed when the On-Premises Edge Configuration Profile playbook is deployed. Software settings are also listed, where applicable.

Table 29. Software BOM for On-Premises Edge Configuration Profile

COMPONENT	ON-PREMISES EDGE CONFIGURATION PROFILE
Kubernetes	1.19.8
Docker	19.03
Kubespray	2.14+
cluster_name:	cluster.local

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

COMPONENT	ON-PREMISES EDGE CONFIGURATION PROFILE
docker_registry	True
Multus	V3.4.2
Topology Manager	1.18
K8s Native CPU Manager	N/A
Node Feature Discovery	0.7.0
CMK	1.5.1
GPU device plugin	N/A
SR-IOV device plugin	3.3.1
SR-IOV CNI	2.6
Bond CNI	False
QAT-DP	0.19.0
TAS	0.2
Userspace CNI	N/A
DDP	N/A
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True
SST BF	False
SST CP (3rd Generation Intel Xeon only)	0
OVS-DPDK	False
FD.IO VPP	False
collectd (*-with required plugins in this doc)	True
NodeExporter	True
cAdvisor	True
Prometheus	False
Custom collector 1	N/A
Grafana	True
<b>General Telemetry Plugins</b>	
RAS memory plugin	N/A
Plugin:SMART	True
Plugin:numa	True
QAT plugin (experimental)	N/A
OVS plugin	False
Netlink Plugin	N/A
ethstats plugin	True
Intel PMU plugin	True
Intel RDT plugin	N/A
Hugepage plugin	False
Disk plugin	True
CPU Plugin	True
virt plugin	N/A
<b>Power Telemetry Plugins</b>	
dpdk telemetry plugin	False
Turbostat/pkgpower.py	True
IPMI	True
cpufreq	True
pkgpower.py (*- see above)	True
<b>GRUB Settings</b>	
isolcpus_enabled	True
hugepages_enabled	True
<b>Default Configuration</b>	
isolcpus:	ex: "4-11"
default_huge_page_size	ex: 1G
hugepages_1G	ex: 4
hugepages_2M	ex: 0
<b>SR-IOV NICs</b>	
iommu_enabled	True
<b>Default Configuration</b>	
"- name:"	enp24s0f0
sriov_numvfs	2
vf_driver	vfio-pci
ddp_profile	"gtp.pkgo"

COMPONENT	ON-PREMISES EDGE CONFIGURATION PROFILE
"- name:"	enp24s0f1
sriov_numvfs	4
vf_driver	iavf
<b>QAT</b>	
Dependency: qat-dp Enabled	True
<b>Default Configuration</b>	
qat_sriov_numvfs	16
<b>DPDK</b>	
DPDK version	19.11.6
Dependency: sriov Enabled	True
<b>Default Configuration</b>	
install_dpdk:	TRUE

### D.3 Step 3 - Set Up On-Premises Edge Configuration Profile

Review the optional Ansible group and host variables in this section and select the 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 [D.3.1](#) and [D.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 full 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 vars – they apply to both control and worker nodes and have cluster-wide impact.
2. Host vars – their scope is limited to a single worker node.

Tables below contain a full list of group and host variables. All these variables are important, but pay special attention to the variables in **bold** as they almost always need to be updated to match the target environment.

#### D.3.1 On-Premises Edge Configuration Profile Group Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Common cluster configuration</b>			
Kubernetes	Boolean	true	Specifies whether to deploy Kubernetes or not.
kube_version	String (semver)	v1.19.8	Kubernetes version.
update_all_packages	Boolean	false	Runs system-wide package update (apt dist-upgrade, yum update, ...). Tip: Can be set using <code>host_vars</code> for more granular control.
<b>http_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTP proxy address. Comment out if your cluster is not behind proxy.</b>
<b>https_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTPS proxy address. Comment out if your cluster is not behind proxy.</b>

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
additional_no_proxy	Comma-separated list of addresses	.example.com	<b>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.</b>
kube_pods_subnet	CIDR	10.244.0.0/16	Kubernetes pod subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
kube_service_addresses	CIDR	10.233.0.0/18	Kubernetes service subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
cluster_name	DNS domain	cluster.local	Name of the cluster
always_pull_enabled	Boolean	true	Set image pull policy to Always. Pulls images before starting containers. Valid credentials must be configured.
<b>Node Feature Discovery</b>			
nfd_enabled	Boolean	true	Specifies whether to deploy Node Feature Discovery.
nfd_build_image_locally	Boolean	false	Builds NFD Docker 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 will query node status and update node labels.
<b>Intel CPU Manager for Kubernetes</b>			
cmk_enabled	Boolean	true	Enables Intel CPU Manager for Kubernetes.
cmk_namespace	String	kube-system	Kubernetes namespace used for CMK deployment.
cmk_use_all_hosts	Boolean	false	Whether to deploy CMK on all nodes in the cluster (including control nodes).
cmk_hosts_list	Comma-separated strings	node1,node2	<b>Comma-separated list of K8s worker node names that the CMK will run on.</b>
cmk_shared_num_cores	Integer	2	<b>Number of CPU cores to be assigned to the "shared" pool on each of the nodes</b>
cmk_exclusive_num_cores	Integer	2	<b>Number of CPU cores to be assigned to the "exclusive" pool on each of the nodes</b>
cmk_shared_mode	String, options: packed, spread	packed	Shared pool allocation mode
cmk_exclusive_mode	String, options: packed, spread	packed	Exclusive pool allocation mode
<b>Native built-in Kubernetes CPU manager</b>			
native_cpu_manager_enabled	Boolean	false	Enabling CMK and built-in CPU Manager is not recommended. 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 that will be reserved for housekeeping (2000m = 2000 millicores = 2 cores)

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
native_cpu_manager_kube_reserved_cpus	Kubernetes millicores	1000m	Number of CPU cores that will 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.
<b>Topology Manager (Kubernetes built-in)<sup>22</sup></b>			
topology_manager_enabled	Boolean	true	Enables Kubernetes built-in Topology Manager
topology_manager_policy	String, options: none, best-effort, restricted, single-numa-node	best-effort	Topology Manager policy.
<b>Intel SR-IOV Network Device Plugin</b>			
sriov_net_dp_enabled	Boolean	true	Enables SR-IOV Network Device Plugin
sriov_net_dp_namespace	String	kube-system	Kubernetes namespace that will be used to deploy SR-IOV Network Device Plugin
sriov_net_dp_build_image_locally	Boolean	true	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 configuration refer to: <a href="https://github.com/intel/sriov-network-device-plugin#configurations">https://github.com/intel/sriov-network-device-plugin#configurations</a>
<b>Intel Device Plugins for Kubernetes</b>			
qat_dp_enabled	Boolean	true	Enables Intel QAT Device Plugin
qat_dp_namespace	String	kube-system	Namespace used for Intel QAT Device Plugin
<b>Intel Telemetry Aware Scheduling</b>			
tas_enabled	Boolean	true	Enables Intel Telemetry Aware Scheduling
tas_namespace	String	default	Kubernetes namespace used for TAS deployment
tas_create_policy	Boolean	true	Creates demo TAS policy
<b>Example Network Attachment Definitions (ready to use examples of custom CNI plugin configuration)</b>			
example_net_attach_defs. userspace_ovs_dpdk	Boolean	false	Example net-attach-def for Userspace CNI with OVS-DPDK

### D.3.2 On-Premises Edge Configuration Profile Host Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>SR-IOV and network devices configuration</b>			
iommu_enabled	Boolean	true	Sets up SR-IOV related kernel parameters and enables further SR-IOV configuration.
dataplane_interfaces	List of dictionaries	n/a	<b>SR-IOV related NIC configuration using per-port approach</b>
dataplane_interfaces[*].name	String	enp24s0f0, enp24s0f1	<b>Name of the interface representing PF port</b>
dataplane_interfaces[*].bus_info	String (PCI address)	18:00.0, 18:00.1	<b>PCI address of the PF port</b>

<sup>22</sup> See backup for workloads and configurations or visit [www.intel.com/PerformanceIndex](http://www.intel.com/PerformanceIndex). Results may vary.

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
dataplane_interfaces[*].pf_driver	String	i40e	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 will be bound to
dataplane_interfaces[*].sriov_vfs[*]	List of dictionaries	n/a	List of vifs 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. <b>Note: Use only for the port 0 of the Network Adapter (PCI address ending with :00.0)</b>
update_nic_drivers	Boolean	true	Set to 'true' to update Linux kernel drivers for Intel Network Adapters
update_nic_firmware	Boolean	false	# Set 'true' to update Network Adapter firmware
firmware_update_nics	List of strings	[enp24s0f0, enp24s0f1]	Additional list of Network Adapters that the FW update will be executed on. <b>Note: FW update will be also executed on all Network Adapters listed in "dataplane_interfaces[*].name"</b>
install_ddp_packages	Boolean	true	Install Intel Ethernet 700 and 800 Series Network Adapters DDP packages. Required if DDP packages configured in dataplane_interfaces.
install_dpdk	Boolean	true	DPDK installation is required for sriov_cni_enabled:true
dpdk_version	String	19.11.6	DPDK version to install
dpdk_local_patches_dir	String	Empty	Path to user supplied patches to apply against the specified version of DPDK
<b>SR-IOV and Bond CNI plugins</b>			
sriov_cni_enabled	Boolean	false	Installs SR-IOV CNI plugin binary on the node.
bond_cni_enabled	Boolean	false	Installs Bond CNI plugin binary on the node.
<b>Hugepages/memory configuration<sup>23</sup></b>			
hugepages_enabled	Boolean	true	Enables hugepages support
default_hugepage_size	String, options: 2M, 1G	1G	Default hugepage size
hugepages_1G	Integer	4	Sets how many hugepages of 1G size should be created
hugepages_2M	Integer	0	Sets how many hugepages of 2M size should be created
<b>CPU configuration</b>			
isolcpus_enabled	Boolean	true	Enables CPU cores isolation from Linux scheduler

<sup>23</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
isolcpus	Comma-separated list of CPU cores/ranges	4-11	CPU cores isolated from Linux scheduler, if CMK is enabled it's a good practice to match it with total number of shared and exclusive cores
intel_pstate	String, options: enabled, disabled	disabled	Enables Intel P-state scaling driver
sst_bf_configuration_enabled	Boolean	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	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	False	Prioritize (SST-CP) power flow to high frequency cores
sst_cp_configuration_enabled	Boolean	False	Enables Intel SST Core Priority technology on 3rd Generation Intel® Xeon®. SST-CP overrides any 'SST-BF configuration'
sst_cp_priority_type	Integer	0	0 – proportional 1 – ordered
sst_cp_clos_groups	List of dictionaries	n/a	Allows for configuration of up to 4 CLOS groups including id, frequency_weight, min_MHz, max_MHz
sst_cp_cpu_clos	List of dictionaries	n/a	Allows for definition of cpu cores per close group
<b>Miscellaneous</b>			
dns_disable_stub_listener	Boolean	true	(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	Boolean	false	(CentOS 7 only) Installs real-time Linux kernel packages.
<b>QAT configuration</b>			
qat_devices	List of dictionaries	n/a	SR-IOV related QAT configuration using per-port approach
qat_devices[*].qat_dev	String	Crypto01, Crypto02, Crypto03	Name of the interface representing PF port
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[*].module_type	String	qat_c62x	QAT hardware identifier, qat_c62x, qat_dh895xcc, qat_c3xxx, etc...
qat_devices[*].pci_type	String	c6xx	PF driver, "c6xx", "c3xx", "d15xx", etc...
qat_devices[*].qat_sriov_numvfs	Integer	10	Number of VFs to be created per QAT device physical function

#### D.4 Step 4 - Deploy On-Premises Edge Configuration Profile Platform

**Note:** You must download the Configuration Profile playbook as described in [D.2](#) and configure it as described in [D.3](#) before you complete this step.

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

In order to deploy the On-Premises Edge Configuration Profile playbook, change the working directory to where you have cloned or unarchived the BMRA Ansible Playbook source code (as described in [Section 3.3.2](#)) and execute the command below:

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

### D.5 Step 5 - Validate On-Premises Edge Configuration Profile

Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) and run the validation processes according to the hardware and software components that you have installed.

# Appendix E

## BMRA Remote Central Office-Forwarding Configuration Profile Setup

## Appendix E BMRA Remote CO-Forwarding Configuration Profile Setup

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

To use the Remote CO-Forwarding Configuration Profile, perform the following steps:

1. Choose your hardware, set it up, and configure the BIOS. Refer to [E.1](#) for details.  
You also need to build your Kubernetes cluster. [Figure 1](#) is an example.
2. Download the Ansible playbook for your Configuration Profile. Refer to [E.2](#) for details.
3. Configure the optional Ansible parameters using the info in the Configuration Profile tables. Refer to [E.3](#) for details.
4. Deploy the platform. Refer to [E.4](#) for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) 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 appendix.

TERM	DESCRIPTION
<b>Hardware Taxonomy</b>	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, etc.)
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.
<b>Software Taxonomy</b>	
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.

### E.1 Step 1 - Set Up Remote CO-Forwarding Configuration Profile Hardware

The tables in this section list the Hardware BOM for the Remote CO-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 30. Hardware Setup for Remote CO-Forwarding Configuration Profile**

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	<a href="#">Controller_2ndGen_2</a>	<a href="#">Controller_3rdGen_2</a>
Worker Node Options	<a href="#">Worker_2ndGen_Base_3</a>	<a href="#">Worker_3rdGen_Base_3</a>
	or <a href="#">Worker_2ndGen_Plus_2</a>	or <a href="#">Worker_3rdGen_Plus_2</a>

### E.2 Step 2 - Download Remote CO-Forwarding Configuration Profile Ansible Playbook

This section contains step-by-step details for downloading the Remote CO-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 CO-Forwarding Configuration Profile Ansible playbook using the following steps:

1. Login to your Ansible host (the one that you will run these Ansible playbooks from).
2. 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 19](#). For example:
 

```
git checkout v21.03
```
3. Export the environmental variable for Kubernetes **Remote CO-Forwarding** Configuration Profile deployment:
 

```
export PROFILE=remote_fp
```
4. Copy the example inventory file to the project root dir.
 

```
cp examples/${PROFILE}/inventory.ini .
```

5. Copy the example configuration files to the project root dir:

```
cp -r examples/${PROFILE}/group_vars examples/${PROFILE}/host_vars .
```

### E.2.1 Remote CO-Forwarding Configuration Profile Ansible Playbook Overview

The Ansible playbook for the Remote CO-Forwarding Configuration Profile allows you to provision a production-ready Kubernetes cluster with all components listed in [Section E.2.2](#). It also applies any additional requirements, such as host OS configuration or Network Adapter drivers and firmware updates. Every capability included in the Remote CO-Forwarding 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 Remote CO-Forwarding Configuration Profile.

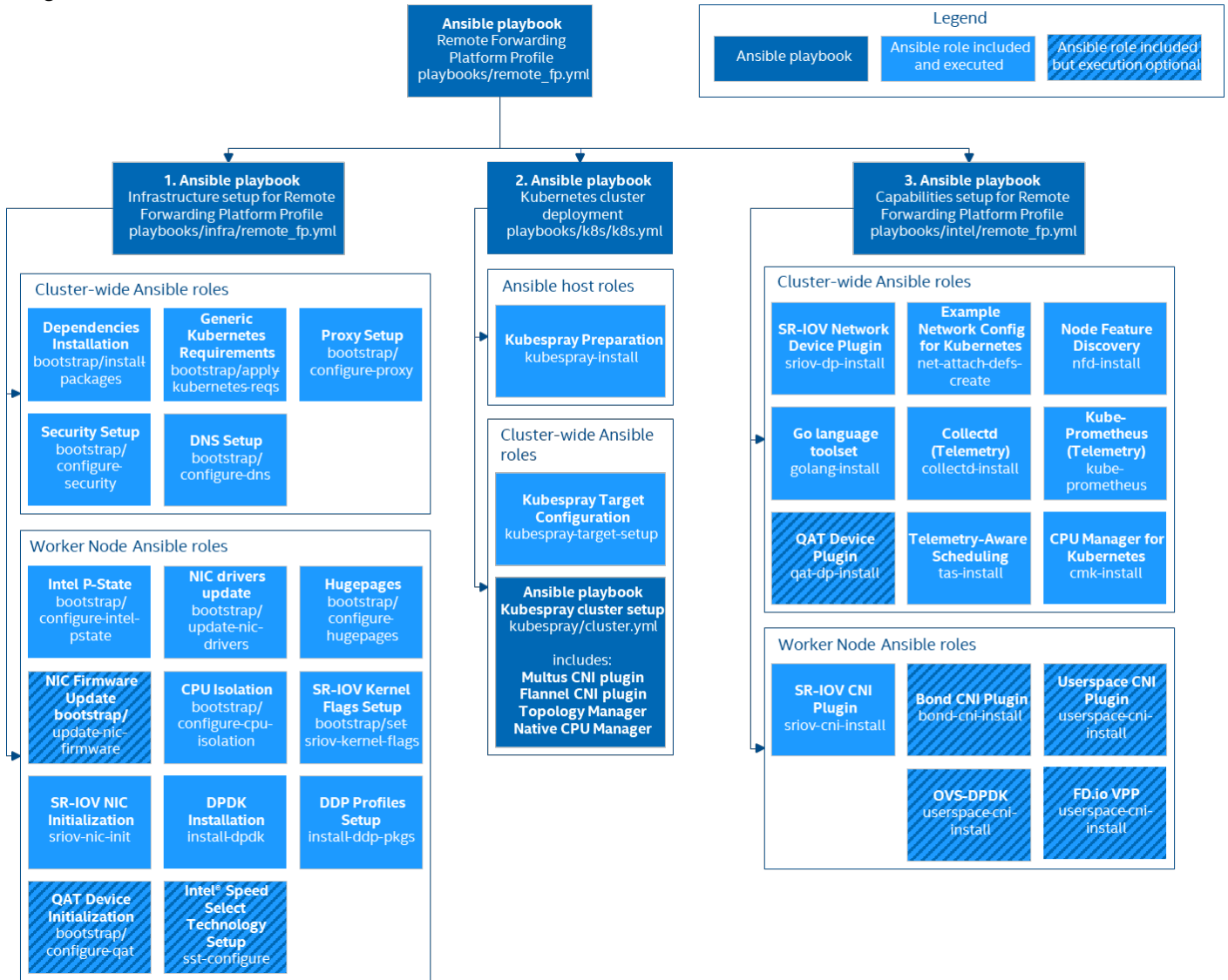


Figure 16. Remote CO-Forwarding Configuration Profile Ansible Playbook

### E.2.2 Remote CO-Forwarding Configuration Profile Software BOM

The table below lists the software components and versions that are automatically installed when the Remote CO-Forwarding Configuration Profile playbook is deployed. Software settings are also listed, where applicable.

Table 31. Software BOM for Remote CO-Forwarding Configuration Profile

COMPONENT	REMOTE CO-FORWARDING CONFIGURATION PROFILE
Kubernetes	1.19.8
Docker	19.03
Kubespray	2.14+

**Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor**

COMPONENT	REMOTE CO-FORWARDING CONFIGURATION PROFILE
cluster_name:	cluster.local
docker_registry	True
Multus	V3.4.2
Topology Manager	1.18
K8s Native CPU Manager	N/A
Node Feature Discovery	0.7.0
CMK	1.5.1
GPU device plugin	N/A
SR-IOV device plugin	3.3.1
SR-IOV CNI	2.6
Bond CNI	False
QAT-DP	False
TAS	0.2
Userspace CNI	False
DDP	True
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True
SST BF	False
SST CP (3rd Generation Intel Xeon only)	0
OVS-DPDK	False
FD.IO VPP	False
collectd (*-with required plugins in this doc)	True
NodeExporter	True
cAdvisor	True
Prometheus	False
Custom collector 1	N/A
Grafana	True
<b>General Telemetry Plugins</b>	
RAS memory plugin	N/A
Plugin:SMART	True
Plugin:numa	True
QAT plugin (experimental)	N/A
OVS plugin	False
Netlink Plugin	N/A
ethstats plugin	True
Intel PMU plugin	True
Intel RDT plugin	N/A
Hugepage plugin	False
Disk plugin	True
CPU Plugin	True
virt plugin	N/A
<b>Power Telemetry Plugins</b>	
dpdk telemetry plugin	False
Turbostat/pkgpower.py	True
IPMI	True
cpufreq	True
pkgpower.py (*- see above)	True
<b>GRUB Settings</b>	
isolcpus_enabled	True
hugepages_enabled	True
<b>Default Configuration</b>	
isolcpus:	ex: "4-11"
default_huge_page_size	ex: 1G
hugepages_1G	ex: 4
hugepages_2M	ex: 0
<b>SR-IOV NICs</b>	
iommu_enabled	True
<b>Default Configuration</b>	
"- name:"	enp24s0f0
sriov_numvfs	2
vf_driver	vfio-pci
ddp_profile	"gtp.pkgo"

COMPONENT	REMOTE CO-FORWARDING CONFIGURATION PROFILE
"- name:"	enp24s0f1
sriov_numvfs	4
vf_driver	iavf
<b>QAT</b>	
Dependency: qat-dp Enabled	False
<b>Default Configuration</b>	
qat_sriov_numvfs	16
<b>DPDK</b>	
DPDK version	19.11.6
Dependency: sriov Enabled	True
<b>Default Configuration</b>	
install_dpdk	TRUE

### E.3 Step 3 - Set Up Remote CO-Forwarding Configuration Profile

Review the optional Ansible group and host variables in this section and select the 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/nodel.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 [E.3.1](#) and [E.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 full set of configuration variables for the Remote CO-Forwarding Configuration Profile along with their default values can be found in `examples/remote_fp` directory.

Variables are grouped into two main categories:

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

Tables below contain a full list of group and host variables. All these variables are important, but pay special attention to the variables in **bold** as they almost always need to be updated to match the target environment.

#### E.3.1 Remote CO-Forwarding Configuration Profile Group Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Common cluster configuration</b>			
Kubernetes	Boolean	true	Specifies whether to deploy Kubernetes or not.
kube_version	String (semver)	v1.19.8	Kubernetes version.
update_all_packages	Boolean	false	Runs system-wide package update (apt dist-upgrade, yum update, ...). Tip: Can be set using <code>host_vars</code> for more granular control.
<b>http_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTP proxy address. Comment out if your cluster is not behind proxy.</b>
<b>https_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTPS proxy address. Comment out if your cluster is not behind proxy.</b>

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
additional_no_proxy	Comma-separated list of addresses	.example.com	<b>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.</b>
kube_pods_subnet	CIDR	10.244.0.0/16	Kubernetes pod subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
kube_service_addresses	CIDR	10.233.0.0/18	Kubernetes service subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
cluster_name	DNS domain	cluster.local	Name of the cluster
always_pull_enabled	Boolean	true	Set image pull policy to Always. Pulls images before starting containers. Valid credentials must be configured.
<b>Node Feature Discovery</b>			
nfd_enabled	Boolean	true	Specifies whether to deploy Node Feature Discovery.
nfd_build_image_locally	Boolean	false	Builds NFD Docker 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 will query node status and update node labels.
<b>Intel CPU Manager for Kubernetes</b>			
cmk_enabled	Boolean	true	Enables Intel CPU Manager for Kubernetes.
cmk_namespace	String	kube-system	Kubernetes namespace used for CMK deployment.
cmk_use_all_hosts	Boolean	false	Whether to deploy CMK on all nodes in the cluster (including control nodes).
cmk_hosts_list	Comma-separated strings	node1,node2	<b>Comma-separated list of K8s worker node names that the CMK will run on.</b>
cmk_shared_num_cores	Integer	2	<b>Number of CPU cores to be assigned to the "shared" pool on each of the nodes</b>
cmk_exclusive_num_cores	Integer	2	<b>Number of CPU cores to be assigned to the "exclusive" pool on each of the nodes</b>
cmk_shared_mode	String, options: packed, spread	packed	Shared pool allocation mode
cmk_exclusive_mode	String, options: packed, spread	packed	Exclusive pool allocation mode
<b>Native built-in Kubernetes CPU manager</b>			
native_cpu_manager_enabled	Boolean	false	Enabling CMK and built-in CPU Manager is not recommended. 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 that will be reserved for housekeeping (2000m = 2000 millicores = 2 cores)

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
native_cpu_manager_kube_reserved_cpus	Kubernetes millicores	1000m	Number of CPU cores that will 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.
<b>Topology Manager (Kubernetes built-in)</b>			
topology_manager_enabled	Boolean	true	Enables Kubernetes built-in Topology Manager
topology_manager_policy	String, options: none, best-effort, restricted, single-numa-node	best-effort	Topology Manager policy.
<b>Intel SR-IOV Network Device Plugin</b>			
sriov_net_dp_enabled	Boolean	true	Enables SR-IOV Network Device Plugin
sriov_net_dp_namespace	String	kube-system	Kubernetes namespace that will be used to deploy SR-IOV Network Device Plugin
sriov_net_dp_build_image_locally	Boolean	true	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 configuration refer to: <a href="https://github.com/intel/sriov-network-device-plugin#configurations">https://github.com/intel/sriov-network-device-plugin#configurations</a>
<b>Intel Device Plugins for Kubernetes<sup>24</sup></b>			
qat_dp_enabled	Boolean	false	Enables Intel QAT Device Plugin
qat_dp_namespace	String	kube-system	Namespace used for Intel QAT Device Plugin
<b>Intel Telemetry Aware Scheduling</b>			
tas_enabled	Boolean	true	Enables Intel Telemetry Aware Scheduling
tas_namespace	String	default	Kubernetes namespace used for TAS deployment
tas_create_policy	Boolean	false	Creates demo TAS policy
<b>Example Network Attachment Definitions (ready to use examples of custom CNI plugin configuration)</b>			
example_net_attach_defs.userspace_ovs_dpdk	Boolean	false	Example net-attach-def for Userspace CNI with OVS-DPDK
example_net_attach_defs.userspace_vpp	Boolean	false	Example net-attach-def for Userspace CNI with VPP
example_net_attach_defs.sriov_net_dp	Boolean	true	Example net-attach-def for SR-IOV Net DP and SR-IOV CNI

### E.3.2 Remote CO-Forwarding Configuration Profile Host Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>SR-IOV and network devices configuration<sup>25</sup></b>			
iommu_enabled	Boolean	true	Sets up SR-IOV related kernel parameters and enables further SR-IOV configuration.

<sup>24</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

<sup>25</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
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	i40e	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 will be 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. <b>Note: Use only for the port 0 of the Network Adapter (PCI address ending with :00.0)</b>
update_nic_drivers	Boolean	true	Set to 'true' to update Linux kernel drivers for Intel Network Adapters
update_nic_firmware	Boolean	false	# Set 'true' to update Network Adapter firmware
firmware_update_nics	List of strings	[enp24s0f0, enp24s0f1]	Additional list of Network Adapters that the FW update will be executed on. <b>Note: FW update will be also executed on all Network Adapters listed in "dataplane_interfaces[*].name"</b>
install_ddp_packages	Boolean	true	Install Intel Ethernet 700 and 800 Series Network Adapters DDP packages. Required if DDP packages configured in dataplane_interfaces.
install_dpdk	Boolean	true	DPDK installation is required for sriov_cni_enabled:true
dpdk_version	String	19.11.6	DPDK version to install
dpdk_local_patches_dir	String	Empty	Path to user supplied patches to apply against the specified version of DPDK
<b>SR-IOV and Bond CNI plugins</b>			
sriov_cni_enabled	Boolean	false	Installs SR-IOV CNI plugin binary on the node.
bond_cni_enabled	Boolean	false	Installs Bond CNI plugin binary on the node.
<b>Userspace networking plugins and accelerated virtual switches</b>			
userspace_cni_enabled	Boolean	false	Installs Userspace CNI plugin binary on the node.
ovs_dpdk_enabled	Boolean	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

**Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor**

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
vpp_enabled	Boolean	false	Installs FD.io VPP (CentOS 7 and Ubuntu 18.04 only)
<b>Hugepages/memory configuration</b>			
hugepages_enabled	Boolean	true	Enables hugepages support
default_hugepage_size	String, options: 2M, 1G	1G	Default hugepage size
hugepages_1G	Integer	4	Sets how many hugepages of 1G size should be created
hugepages_2M	Integer	0	Sets how many hugepages of 2M size should be created
<b>CPU configuration</b>			
isolcpus_enabled	Boolean	true	Enables CPU cores isolation from Linux scheduler
isolcpus	Comma-separated list of CPU cores/ranges	4-11	CPU cores isolated from Linux scheduler, if CMK is enabled it's a good practice to match it with total number of shared and exclusive cores
intel_pstate	String, options: enabled, disabled	disabled	Enables Intel P-state scaling driver
sst_bf_configuration_enabled	Boolean	true	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	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	False	Prioritize (SST-CP) power flow to high frequency cores
sst_cp_configuration_enabled	Boolean	False	Enables Intel SST Core Priority technology on 3rd Generation Intel® Xeon®. SST-CP overrides any 'SST-BF configuration'
sst_cp_priority_type	Integer	0	0 – proportional 1 - ordered
sst_cp_clos_groups	List of dictionaries	n/a	Allows for configuration of up to 4 CLOS groups including id, frequency_weight, min_MHz, max_MHz
sst_cp_cpu_clos	List of dictionaries	n/a	Allows for definition of cpu cores per close group
<b>Miscellaneous</b>			
dns_disable_stub_listener	Boolean	true	(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	Boolean	false	(CentOS 7 only) Installs real-time Linux kernel packages.
<b>QAT configuration</b>			
qat_devices	List of dictionaries	n/a	SR-IOV related QAT configuration using per-port approach

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
qat_devices[*].qat_dev	String	Crypto01, Crypto02, Crypto03	Name of the interface representing PF port
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[*].module_type	String	qat_c62x	QAT hardware identifier, qat_c62x, qat_dh895xcc, qat_c3xxx, etc...
qat_devices[*].pci_type	String	c6xx	PF driver, "c6xx", "c3xx", "d15xx", etc...
qat_devices[*].qat_sriov_numvfs	Integer	10	Number of VFs to be created per QAT device physical function

#### E.4 Step 4 - Deploy Remote CO-Forwarding Configuration Profile Platform

**Note:** You must download the Configuration Profile playbook as described in [E.2](#) and configure it as described in [E.3](#) before you complete this step.

In order to deploy the Remote CO-Forwarding Configuration Profile playbook, change the working directory to where you have cloned or unarchived the BMRA Ansible Playbook source code (as described in [Section 3.3.2](#)) and execute the command below:

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

#### E.5 Step 5 - Validate Remote-CO Forwarding Configuration Profile

Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) and run the validation processes according to the hardware and software components that you have installed.

# Appendix F

## BMRA Regional Data Center Configuration Profile Setup

## Appendix F BMRA Regional Data Center Configuration Profile Setup

This appendix 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 [F.1](#) for details.  
You also need to build your Kubernetes cluster. [Figure 1](#) is an example.
2. Download the Ansible playbook for your Configuration Profile. Refer to [F.2](#) for details.
3. Configure the optional Ansible parameters using the info in the Configuration Profile tables. Refer to [F.3](#) for details.
4. Deploy the platform. Refer to [F.4](#) for details.
5. Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) 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 appendix.

TERM	DESCRIPTION
<b>Hardware Taxonomy</b>	
ENABLED	Setting must be enabled in the BIOS (configured as Enabled, Yes, True, etc.)
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.
<b>Software Taxonomy</b>	
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.

### F.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 32. Hardware Setup for Regional Data Center Configuration Profile**

NODE OPTIONS	2ND GENERATION INTEL XEON SCALABLE PROCESSOR	3RD GENERATION INTEL XEON SCALABLE PROCESSOR
Control Node Options	N/A*	<a href="#">Controller_3rdGen_3</a>
Worker Node Options	N/A*	<a href="#">Worker_3rdGen_Plus_3</a>

### F.2 Step 2 - Download Regional Data Center Configuration Profile Ansible Playbook

This section contains step-by-step 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 following steps:

1. Login to your Ansible host (the one that you will run these Ansible playbooks from).
2. 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 19](#). For example:

```
git checkout v21.03
```

3. Export the environmental variable for Kubernetes **BMRA Regional Data Center** deployment:

```
export PROFILE=remote_fp
```

4. Copy the example inventory file to the project root dir.

```
cp examples/${PROFILE}/inventory.ini .
```

5. Copy the example configuration files to the project root dir:

```
cp -r examples/${PROFILE}/group_vars examples/${PROFILE}/host_vars .
```

### F.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 with all components listed in [Section F.2.2](#). 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.

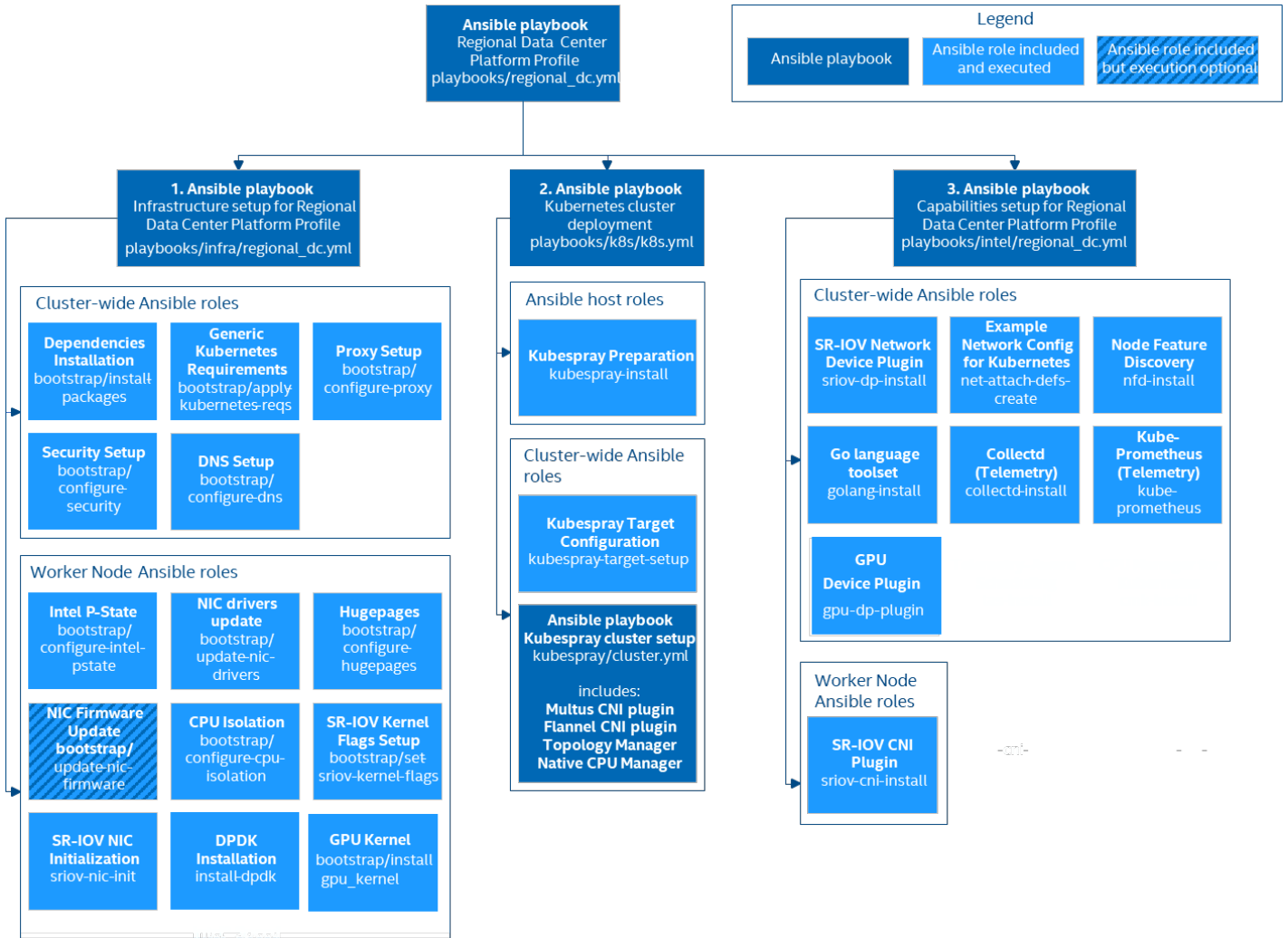


Figure 17. Regional Data Center Configuration Profile Ansible Playbook

### F.2.2 Regional Data Center Configuration Profile Software BOM

The table below lists the software components and versions that are automatically installed when the Remote CO-Forwarding Configuration Profile playbook is deployed. Software settings are also listed, where applicable.

Table 33. Software BOM for Remote Regional Data Center Configuration Profile

COMPONENT	REGIONAL DATA CENTER CONFIGURATION PROFILE
Kubernetes	1.19.8
Docker	19.03
Kubespray	2.14+
cluster_name:	cluster.local
docker_registry	True
Multus	V3.4.2

**Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor**

COMPONENT	REGIONAL DATA CENTER CONFIGURATION PROFILE
Topology Manager	1.18
K8s Native CPU Manager	TRUE
Node Feature Discovery	0.7.0
CMK	N/A
GPU device plugin	3.3.1
SR-IOV device plugin	3.3.1
SR-IOV CNI	TRUE
Bond CNI	N/A
QAT-DP	N/A
TAS	FALSE
Userspace CNI	N/A
DDP	N/A
Intel Ethernet 700 and 800 Series Network Adapters Drivers	True
SST BF	N/A
SST CP (3rd Generation Intel Xeon only)	N/A
OVS-DPDK	N/A
FD.IO VPP	N/A
collectd (*-with required plugins in this doc)	True
NodeExporter	True
cAdvisor	True
Prometheus	True
Custom collector 1	False
Grafana	True
<b>General Telemetry Plugins</b>	
RAS memory plugin	True
Plugin:SMART	True
Plugin:numa	True
QAT plugin (experimental)	N/A
OVS plugin	N/A
Netlink Plugin	N/A
ethstats plugin	True
Intel PMU plugin	N/A
Intel RDT plugin	N/A
Hugepage plugin	True
Disk plugin	True
CPU Plugin	True
virt plugin	N/A
<b>Power Telemetry Plugins</b>	
dppk telemetry plugin	False
Turbostat/pkgpower.py	True
IPMI	True
cpufreq	True
<b>GRUB Settings</b>	
isolcpus_enabled	True
hugepages_enabled	True
<b>Default Configuration</b>	
isolcpus:	ex: "4-11"
default_huge_page_size	ex: 1G
hugepages_1G	ex: 4
hugepages_2M	ex: 0
<b>SR-IOV NICs</b>	
iommu_enabled	True
<b>Default Configuration</b>	
"- name:"	enp24s0f0
sriov_numvfs	2
vf_driver	vfio-pci
ddp_profile	"gtp.pkgo"
"- name:"	enp24s0f1
sriov_numvfs	4
vf_driver	iavf

**QAT**

COMPONENT	REGIONAL DATA CENTER CONFIGURATION PROFILE
Dependency: qat-dp Enabled	False
<b>Default Configuration</b>	
qat_sriov_numvfs	16
<b>DPDK</b>	
DPDK version	19.11.6
Dependency: sriov Enabled	True
<b>Default Configuration</b>	
install_dpdk:	TRUE

### F.3 Step 3 - Set Up Regional Data Center Configuration Profile

Review the optional Ansible group and host variables in this section and select the 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 [F.3.1](#) and [F.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 full set of configuration variables for the Regional Data Center Configuration Profile along with their default values can be found in `examples/remote_fp` directory.

Variables are grouped into two main categories:

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

Tables below contain a full list of group and host variables. All these variables are important, but pay special attention to the variables in **bold** as they almost always need to be updated to match the target environment.

#### F.3.1 Regional Data Center Configuration Profile Group Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Common cluster configuration</b>			
Kubernetes	Boolean	True	Specifies whether to deploy Kubernetes or not.
kube_version	String (semver)	v1.19.8	Kubernetes version.
update_all_packages	Boolean	False	Runs system-wide package update (apt dist-upgrade, yum update, ...). Tip: Can be set using <code>host_vars</code> for more granular control.
<b>http_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTP proxy address. Comment out if your cluster is not behind proxy.</b>
<b>https_proxy</b>	URL	<b>http://proxy.example.com:1080</b>	<b>HTTPS proxy address. Comment out if your cluster is not behind proxy.</b>
<b>additional_no_proxy</b>	Comma-separated list of addresses	<b>.example.com</b>	<b>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 <code>no_proxy</code> automatically.</b>

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
kube_pods_subnet	CIDR	10.244.0.0/16	Kubernetes pod subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
kube_service_addresses	CIDR	10.233.0.0/18	Kubernetes service subnet. Ensure that it matches your CNI plugin requirements (Flannel by default) and doesn't overlap with your corporate LAN.
cluster_name	DNS domain	cluster.local	Name of the cluster
always_pull_enabled	Boolean	True	Set image pull policy to Always. Pulls images before starting containers. Valid credentials must be configured.
<b>Node Feature Discovery</b>			
nfd_enabled	Boolean	True	Specifies whether to deploy Node Feature Discovery.
nfd_build_image_locally	Boolean	False	Builds NFD Docker 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 will query node status and update node labels.
<b>Intel CPU Manager for Kubernetes</b>			
cmk_enabled	Boolean	True	Enables Intel CPU Manager for Kubernetes.
cmk_namespace	String	kube-system	Kubernetes namespace used for CMK deployment.
cmk_use_all_hosts	Boolean	False	Whether to deploy CMK on all nodes in the cluster (including control nodes).
cmk_hosts_list	Comma-separated strings	node1,node2	<b>Comma-separated list of K8s worker node names that the CMK will run on.</b>
cmk_shared_num_cores	Integer	2	<b>Number of CPU cores to be assigned to the "shared" pool on each of the nodes</b>
cmk_exclusive_num_cores	Integer	2	<b>Number of CPU cores to be assigned to the "exclusive" pool on each of the nodes</b>
cmk_shared_mode	String, options: packed, spread	packed	Shared pool allocation mode
cmk_exclusive_mode	String, options: packed, spread	packed	Exclusive pool allocation mode
<b>Native built-in Kubernetes CPU manager</b>			
native_cpu_manager_enabled	Boolean	False	Enabling CMK and built-in CPU Manager is not recommended. 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 that will be reserved for housekeeping (2000m = 2000 millicores = 2 cores)
native_cpu_manager_kube_reserved_cpus	Kubernetes millicores	1000m	Number of CPU cores that will 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.

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Topology Manager (Kubernetes built-in)</b>			
topology_manager_enabled	Boolean	True	Enables Kubernetes built-in Topology Manager
topology_manager_policy	String, options: none, best-effort, restricted, single-numa-node	best-effort	Topology Manager policy.
<b>Intel SR-IOV Network Device Plugin</b>			
sriov_net_dp_enabled	Boolean	True	Enables SR-IOV Network Device Plugin
sriov_net_dp_namespace	String	kube-system	Kubernetes namespace that will be used to deploy SR-IOV Network Device Plugin
sriov_net_dp_build_image_locally	Boolean	True	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 configuration refer to: <a href="https://github.com/intel/sriov-network-device-plugin#configurations">https://github.com/intel/sriov-network-device-plugin#configurations</a>
<b>Intel Device Plugins for Kubernetes<sup>26</sup></b>			
qat_dp_enabled	Boolean	False	Enables Intel QAT Device Plugin
qat_dp_namespace	String	kube-system	Namespace used for Intel QAT Device Plugin
<b>Intel Telemetry Aware Scheduling</b>			
tas_enabled	Boolean	True	Enables Intel Telemetry Aware Scheduling
tas_namespace	String	Default	Kubernetes namespace used for TAS deployment
tas_create_policy	Boolean	False	Creates demo TAS policy
<b>Example Network Attachment Definitions (ready to use examples of custom CNI plugin configuration)</b>			
example_net_attach_defs.userspace_ovs_dpdk	Boolean	False	Example net-attach-def for Userspace CNI with OVS-DPDK
example_net_attach_defs.userspace_vpp	Boolean	False	Example net-attach-def for Userspace CNI with VPP
example_net_attach_defs.sriov_net_dp	Boolean	True	Example net-attach-def for SR-IOV Net DP and SR-IOV CNI

### F.3.2 Regional Data Center Configuration Profile Host Variables

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>SR-IOV and network devices configuration<sup>27</sup></b>			
iommu_enabled	Boolean	True	Sets up SR-IOV related kernel parameters and enables further SR-IOV configuration.
dataplane_interfaces	List of dictionaries	N/A	<b>SR-IOV related NIC configuration using per-port approach</b>
dataplane_interfaces[*].name	String	enp24s0f0, enp24s0f1	<b>Name of the interface representing PF port</b>
dataplane_interfaces[*].bus_info	String (PCI address)	18:00.0, 18:00.1	<b>PCI address of the PF port</b>

<sup>26</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

<sup>27</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
dataplane_interfaces[*].pf_driver	String	i40e	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 will be 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	Set to 'true' to update Linux kernel drivers for Intel Network Adapters
update_nic_firmware	Boolean	False	# Set 'true' to update Network Adapter firmware
firmware_update_nics	List of strings	[enp24s0f0, enp24s0f1]	Additional list of Network Adapters that the FW update will be executed on. Note: FW update will be also executed on all Network Adapters listed in "dataplane_interfaces[*].name"
install_dpdk	Boolean	False	DPDK installation is required for sriov_cni_enabled:true
dpdk_version	String	19.11.6	DPDK version to install
dpdk_local_patches_dir	String	Empty	Path to user supplied patches to apply against the specified version of DPDK
<b>SR-IOV and Bond CNI plugins</b>			
sriov_cni_enabled	Boolean	False	Installs SR-IOV CNI plugin binary on the node.
<b>Hugepages/memory configuration</b>			
hugepages_enabled	Boolean	True	Enables hugepages support
default_hugepage_size	String, options: 2M, 1G	1G	Default hugepage size
hugepages_1G	Integer	4	Sets how many hugepages of 1G size should be created
hugepages_2M	Integer	0	Sets how many hugepages of 2M size should be created
<b>CPU configuration</b>			
isolcpus_enabled	Boolean	True	Enables CPU cores isolation from Linux scheduler
isolcpus	Comma-separated list of CPU cores/ranges	4-11	CPU cores isolated from Linux scheduler, if CMK is enabled it's a good practice to match it with total number of shared and exclusive cores

NAME	TYPE	DEFAULT VALUE	DESCRIPTION/COMMENTS
<b>Miscellaneous</b>			
dns_disable_stub_listener	Boolean	True	(Ubuntu only) Disables DNS stub listener from the systemd-resolved service, which is known to cause problems with DNS and Docker containers on Ubuntu

#### F.4 Step 4 - Deploy Regional Data Center Configuration Profile Platform

**Note:** You must download the Configuration Profile playbook as described in [F.2](#) and configure it as described in [F.3](#) before you complete this step.

In order to deploy the Regional Data Center Configuration Profile playbook, change the working directory to where you have cloned or unarchived the BMRA Ansible Playbook source code (as described in [Section 3.3.2](#)) and execute the command below:

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

#### F.5 Step 5 - Validate Regional Data Center Configuration Profile

Validate the setup of your Kubernetes cluster. Refer to the tasks in [Section 7](#) and run the validation processes according to the hardware and software components that you have installed.

# Appendix G

## Ansible Host Setup

## Appendix G Ansible Host Setup

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 2.9.17 installed

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 3.1.3](#)):

- **Using CentOS Linux or RHEL Version 8 or Version 7 as Ansible Host**

1. Install the Linux OS using any method supported by the vendor (CentOS Community, respectively Red Hat, Inc.). 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(s) IP Address; Host Name, Proxies (if necessary) and NTP (Network Time Protocol).
3. After the installation completes and the machine reboots, login as root and confirm that it has a valid IP address and can connect (ping) to the control and worker nodes.
4. Make sure 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:

```
# pip3 install ansible==2.9.17
```

The Ansible Host box is now ready to deploy the Container BMRA. Follow the instructions in [Section 3.3](#).

- **Using Ubuntu 20.04 LTS or 18.04 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 CentOS or 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 pip3 install ansible==2.9.17
```

The Ansible Host box is now ready to deploy the Container BMRA. Follow the instructions in [Section 3.3](#).

**Part 4:**  
**Appendix H**  
**BMRA 21.03 Release Notes**

## Appendix H BMRA Release Notes

This appendix lists the notable changes from the previous releases, including new features, bug fixes, and known issues.<sup>28</sup>

### H.1 BMRA 21.03 New Features

The following new features were updated or added in this release:

- Kubernetes version update to 1.19.x
- Kubernetes feature/plugin updates (NFD, TAS, SRIOV-DP)
- CentOS 8.3 support
- RHEL 8.3 support
- CentOS 7.9 support
- 3rd Generation Intel® Xeon® Scalable processor support
- Intel® Software Guard Extensions Device Plugin
- Intel® SGX Key Management Services
- Intel® Quick Assist Technology Drivers and Services
- Intel® Speed Select Technology – Core Power (Intel® SST-CP)
- Intel® Server Graphics 1 card support
- Updated Intel® Ethernet 700 & 800 Network Adapter Drivers and DDP profiles
- New Regional Data Center Configuration Profile for Visual Compute Media workloads using Intel® Server Graphics 1
- Additional Collectd plugins (unixsock, network)
- Additional Grafana dashboards (cpu, disk, intel, ipmi, netlink, ovs, power, numa, hugepages, ethstats)
- Multiple DPDK version options with custom patch support
- Multiple SR-IOV driver assignments per PF

### H.2 BMRA 21.03 Bug Fixes

The following bug fixes were completed in the BMRA 21.03 release:

- Fixed Intel® Network Adapter driver compilation on RHEL
- Fixed Comms DDP profiles not loading at OS boot time
- Cleaned up Ansible warnings occurring during playbook runtime
- Forced DPDK bindings for active devices defined in host\_vars
- Fixed CMK installation failure across multiple worker nodes
- Fixed certificate handling causing Prometheus pod failures

### H.3 BMRA 2.1 New Features

No new features were added.

### H.4 BMRA 2.1 Bug Fixes

The following bug fixes were completed in the BMRA 2.1 release:

- Intel® Ice driver fails to load on Ubuntu 18.04
- Updated Intel Network Adapter drivers to resolve driver compilation issues
- Increased driver download timeouts
- Removed duplicate collectd install in remote\_fp profile
- Added missing NFD role to remote\_fp profile
- Fixed CMK container template to include cmk binary
- Cleaned up data plane interface and IOMMU terminology
- Defaulted data plane interface examples as empty lists
- Updated Intel Network Adapter DDP profile URL
- Updated Intel Network Adapter FW package URL
- Fixed data plane interface selection on DDP profile loading
- Updated Helm stable repo URL
- Fixed DPDK vfio-pci binding for Ubuntu 18.04 and 20.04
- Updated default DPDK version to fix compilation on CentOS/RHEL 8
- Fixed PowerTools repository names for CentOS 8

---

<sup>28</sup> See backup for workloads and configurations or visit [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex). Results may vary.

## H.5 BMRA 2.0 New Features

The following new features were added in this release:

- Ubuntu 20.04 support
- CentOS 8.2 support
- Red Hat Enterprise Linux 8.2 support
- Intel® Speed Select Technology (SST) Base Frequency (BF) & Core Power (CP) support
- Intel® Software Guard Extensions (SGX) support
- Intel® Ethernet Controller E810 Series Adapter Support
- FW Update Support for Intel® E710 and E810 Series Adapters
- Location-based Configuration Profiles
- Kubernetes version update
- Kubernetes feature/plugin updates (CMK, Multus, QAT, TAS, SR-IOV, etc.)
- Intel® Dynamic Device Personalization (DDP) support for E710 and E810 Series Adapters
- CentOS 7.6 RT kernel installation
- Introduction of Telemetry components (collectd, Prometheus, node-exporter, Grafana)
- Kubernetes Pod Security Policies and a more secure Docker registry

## H.6 BMRA 2.0 Bug Fixes

The following bug fixes were completed in the BMRA 2.0 release:

- Fixed deployment issues with Ubuntu 18.04 nodes
- Updated CMK to v1.5.1 to address pod restarting issues
- Fixed SR-IOV VF bindings with 710 Network Adapters
- General network driver installation fixes
- Fixed connection issues between pods on different nodes

## H.7 Known Issues

### 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 SST-BF feature is available on selected 2nd Generation Intel® Xeon® Scalable Processor SKUs (<https://access.redhat.com/articles/4481221>) and both the SST-BF and SST-CP features are available on selected 3rd Generation Intel® Xeon® Scalable Processor SKUs with other supported OSes.

### Issue:

CPU Manager for Kubernetes (CMK) v1.5.1 is not tested with both SST-BF and SST-CP enabled on a node.

### Detail:

CMK was not tested with enabled SST-CP and SST-BF enabled at the same time on a node. The integration tests for this scenario will be included in the next release.

### Workaround:

N/A

### Issue:

## Reference Architecture | Container Bare Metal for 2nd Generation and 3rd Generation Intel® Xeon® Scalable Processor

Kubespray does not fully support Python 3.

### Detail:

The Kubespray project uses Ansible modules that still have dependencies on Python 2. Deployment of BMRA on recent distributions (CentOS/RHEL 8+) that natively support Python 3 will fail.

### Workaround:

A playbook to patch Kubespray (playbooks/k8s/patch\_kubespray.yml) is provided, which should be run before installing your desired profile (i.e., `ansible-playbook -i inventory.ini playbooks/k8s/patch_kubespray.yml`).

### 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:**

## **Appendix I**

### **Workloads and Application Examples**

## Appendix I Workloads and Application Examples

This part provides examples of how to provision and deploy example applications or workloads. For BMRA 21.03, instructions are provided for a Key Management reference application.

### I.1 Enabling Key Management NGINX Applications

To set up KMRA infrastructure and run the Key Management workload, follow the steps below. KMRA currently supports Ubuntu 18.04/20.04, SGX version 2.12, and DCAP 1.9.

The BMRA infrastructure sets up KMRA distributed HSM key server and compute nodes with SGX. A private key is securely provisioned to Intel Crypto-api-toolkit on the compute node and imported into a token called `kmra_token`. The provisioned key is called `nginx_hsm_priv` and pin is 1234. When configuring a workload to use a private key from Intel Crypto-api-toolkit, the workload (in this demo the workload is NGINX) is configured with a URI for the private key.

The URI used in this demo: `"engine:pkcs11:pkcs11:token=kmra_token;object=nginx_hsm_priv;pin-value=1234"`

The Key Management NGINX application below sets up NGINX workload and configures it with OpenSSL. A custom version of OpenSSL is installed on the compute node and configured with the `libp11` interface and `pkcs11` engine. The `pkcs11` engine is an interface to Intel Crypto-api-Toolkit with SGX and it uses the URI configured in the NGINX workload to use the private key. A certificate is generated and signed inside the Intel Crypto-api-Toolkit enclave and stored on the compute node. A link to the certificate is added to the NGINX configuration. The last step of the workload setup is a test using `opens s_time` to verify the number of TLS connections that NGINX is able to establish using the private keys from Intel SGX enclave.

1. Download KMRA source code.
  - a. Go to <https://01.org/key-management-reference-application-kmra> and download Key Management Reference Application (KMRA) v1.1 source code package.
  - b. Create KMRA folder in root directory of BMRA.
  - c. Untar and place contents into KMRA directory.
2. Run BMRA Configuration Profile `full_nfv` with some variable changes in `group_vars/all`.
  - a. Kubernetes option must be set to false.
  - b. Proxy settings must be changed accordingly.
  - c. Set `kmra_use_custom_package_versions` to true.
3. Go to KMRA folder created in step 1 and run the NGINX workload Ansible script.
  - a. Go to `KMRA/ansible/sgx_infra_setup`.
  - b. Update inventory file with correct key server and compute node hosts.
  - c. Run Ansible scripts using the following command:  
`ansible-playbook -i inventory provision_ctk_token_and_start_nginx.yml`



Performance varies by use, configuration and other factors. Learn more at [www.Intel.com/PerformanceIndex](http://www.Intel.com/PerformanceIndex).

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

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

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.