User Manual

Network and Edge Reference System Architectures Portfolio Release v24.01

Guided Deployments of Validated Systems at Specific Network Locations using 4th and 5th Gen Intel® Xeon® Scalable processor, Intel® Core™ processor, Intel Atom® processor, and Intel® Xeon® D processor Platforms

Authors

Aparna Balachandran

Octavia Carotti

Alek Du

Veronika Karpenko

Zhifang Long

Dana Nehama

Abhijit Sinha

Mathieu Sobrero

Daniel Ugarte

1 Introduction

1.1 Purpose

The Network and Edge Reference System Architectures (Reference System[1]) provide a common platform that addresses the growing need for a simple and accelerated cloud-native platform designed to support diverse use cases across network locations, such as on-premises edge, access edge, and 5G core. This user manual provides system definitions (hardware, software, and configurations) and installation instructions for the reference architectures to support key workloads in edge to 5G core and private/public cloud deployments.

The delivered reference systems integrate and validate Intel® platforms and open-source software with Intel® best-known configurations and practices using a cloud-native approach. As a result, forward-looking, cloud-native applications and implementations can be achieved quicker and with confidence.

1.2 Audience and Scope

This Network and Edge Reference System Architectures portfolio enables developers and deployment engineers to implement cloud-native solutions over a bare metal, virtual machine (VM), or cloud service provider (CSP) infrastructure. The documentation for the Network and Edge Reference System Architectures portfolio includes this user manual and individual deployment user guides for the available deployment models. Current deployment models include the following:

- **Container Bare Metal Reference System Architecture (BMRA)** for containers on bare metal. Kubernetes manages the clusters. This model allows an application-ready cluster to be built on single or multiple servers for cloud-native applications.
- **Virtual Machine Reference System Architecture (VMRA)** for virtual clusters implemented with or without Kubernetes. This model allows a virtual cluster to be built on a single server or on multiple servers based on VMs running Kubernetes containers, ready for cloud-native applications.
- **Cloud Reference System Architecture (Cloud RA)** for CSP Intel hardware-based instances defined by the Configuration Profile implemented for running cloud-native applications in the cloud.

This user manual describes concepts and components for the Network and Edge Reference System Architectures, including BMRA and VMRA hardware components, and incorporated cloud native and Kubernetes software technologies. Reading this user manual gives you the background to understand the deployment concepts, components available, and processes required for deployments based on different network locations.

User guides with detailed deployment instructions are available for BMRA, VMRA, and Cloud RA. The quick start guides provide step-by-step instructions for automated deployment procedures using Ansible* scripts. Examples of installation validation are also available.

Note: Some capabilities supported by the reference architectures are only available under NDA. Contact your Intel representative for access to the NDA material.

[1] In this document, "Reference System" refers to the Network and Edge Reference System Architecture.

Table of Contents

Figures

Tables

Document Revision History

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

1. Covered 2nd Gen Intel® Xeon® Scalable processors
2. Covered 2nd and 3rd Gen Intel® Xeon® Scalable processors
3. Covered 2nd and 3rd Gen Intel® Xeon® Scalable processors and Intel® Xeon® D processor

Revision	Date	Description
001	February 2022	Initial release.
002	March 2022	Updated a few URLs.
003	May 2022	Covers the 4th Gen Intel® Xeon® Scalable processor.
004	June 2022	The changes include updates to the discussion of the BMRA for Storage Deployment Model and Figure 2.
005	June 2022	Updated a few URLs.
006	July 2022	Added NDA support for FlexRAN™ software, updated Istio* and service mesh features.

Revision	Date	Description
007	July 2022	The changes include updates to the discussion of the Access Edge Configuration Profile and to Figures 1 and 3.
008	October 2022	Added support for Cloud Reference System Architecture (Cloud RA).
009	December 2022	Includes improvements and updates on RA in alignment with the launch of the 4th Gen Intel® Xeon® Scalable processor.
010	March 2023	Updated for BMRA Release 23.02; includes improvements to run FlexRAN™ software in a container and addition of Media Analytics Libraries.
011	July 2023	Updated for BMRA Release 23.07; includes support for 5th Gen Intel® Xeon® Scalable processors, Intel® Core™ processors, Intel Atom® processors, and other hardware/software updates.
012	October 2023	Updated for BMRA Release 23.10; includes Intel® Trust Domain Extensions (Intel® TDX), Intent-Driven Orchestration, Edge AI Box, and Intel® Media Transport Library.
013	January 2024	Updated for BMRA Release 24.01; includes Intel® In-Band Manageability framework software, KubeVirt, and ingress-nginx.

1.3 Portfolio Architecture

Throughout an infrastructure—whether a private enterprise or public cloud—various optimized systems are needed based on the workloads being run and the sites serviced (Figure 1). The Network and Edge Reference System Architectures Portfolio prescribes deployment solutions optimized to support services based on the location in the infrastructure and the typical applications/workloads served at that location. These solutions can be deployed quickly using validated, best-known configurations (BKCs) and automated deployment tools provided in the Reference System Architectures. The configurations and tools were designed for the network location and a chosen deployment model (containerized or virtual) to deliver a predictive solution for performance-, security-, usability-, and cost-optimized operations.

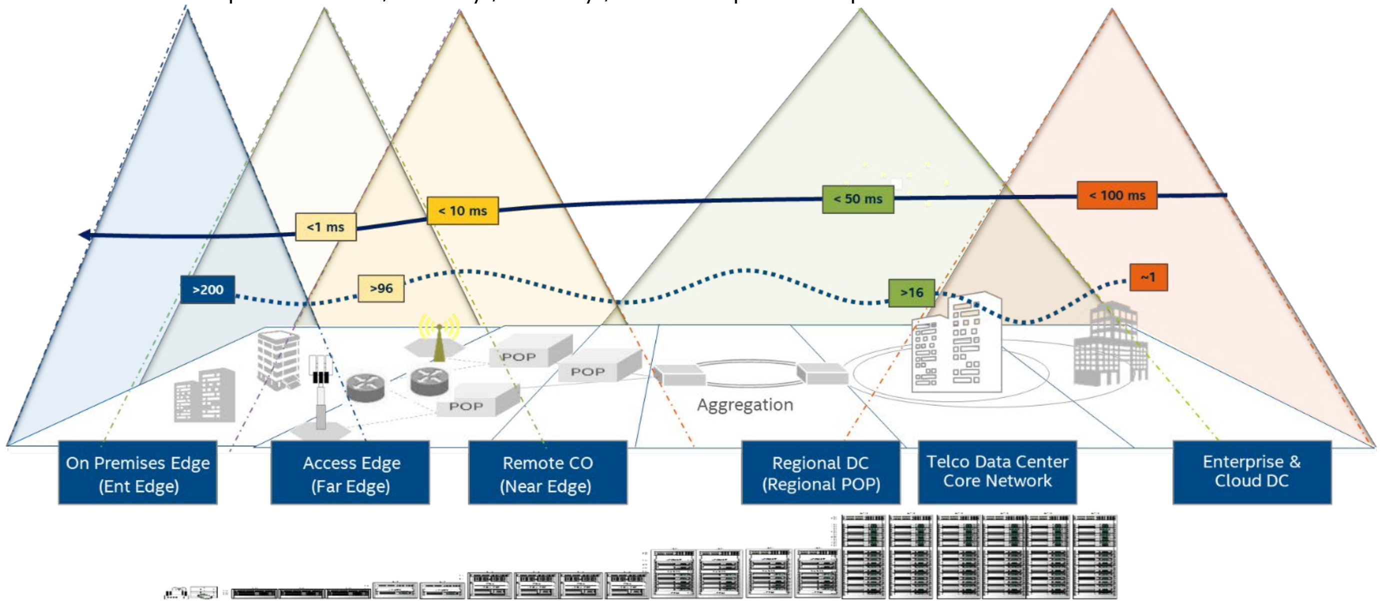
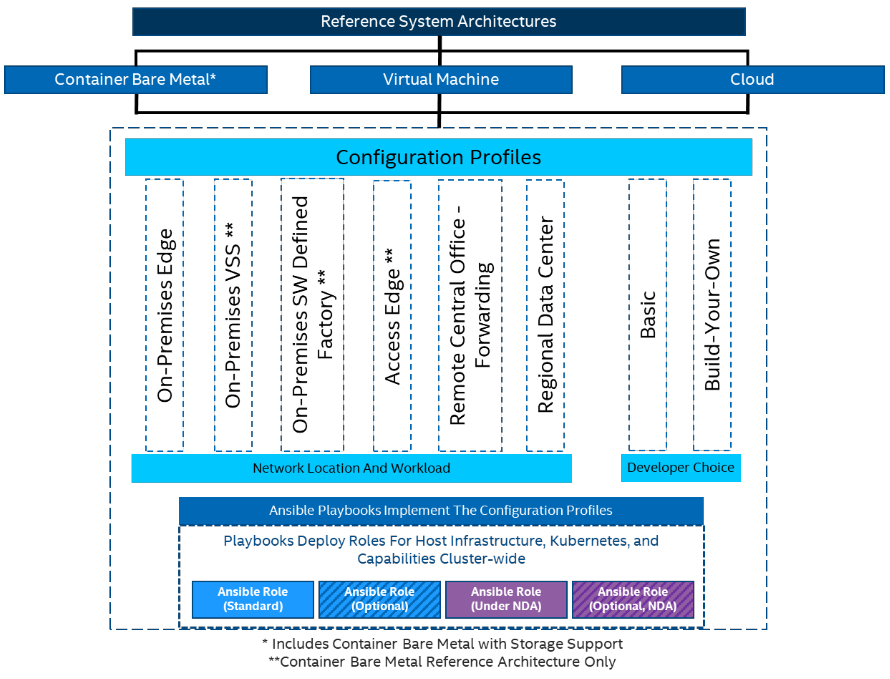


Figure 1. Landscape for Network and Edge Reference System Architectures Portfolio

The architectural hierarchy for the Network and Edge Reference System Architectures Portfolio is shown in Figure 2. The VMRA and BMRA implement Configuration Profiles that provide a recipe for implementing hardware, software, and optimized configuration per the network location. Figure 3 illustrates an example of Configuration Profiles designed for network locations.



Reference System Architectures
Container Bare Metal*
Virtual Machine
Cloud
Configuration Profiles
On-Premises Edge
On-Premises VSS **
On-Premises SW Defined Factory **
Access Edge **
Remote Central Office - Forwarding
Regional Data Center
Basic
Build-Your-Own
Network Location And Workload
Developer Choice
Ansible Playbooks Implement The Configuration Profiles
Playbooks Deploy Roles For Host Infrastructure, Kubernetes, and Capabilities Cluster-wide
Ansible Role (Standard)
Ansible Role (Optional)
Ansible Role (Under NDA)
Ansible Role (Optional, NDA)
* Includes Container Bare Metal with Storage Support
**Container Bare Metal Reference Architecture Only

Figure 2. Network and Edge Reference System Architectures Portfolio Hierarchy

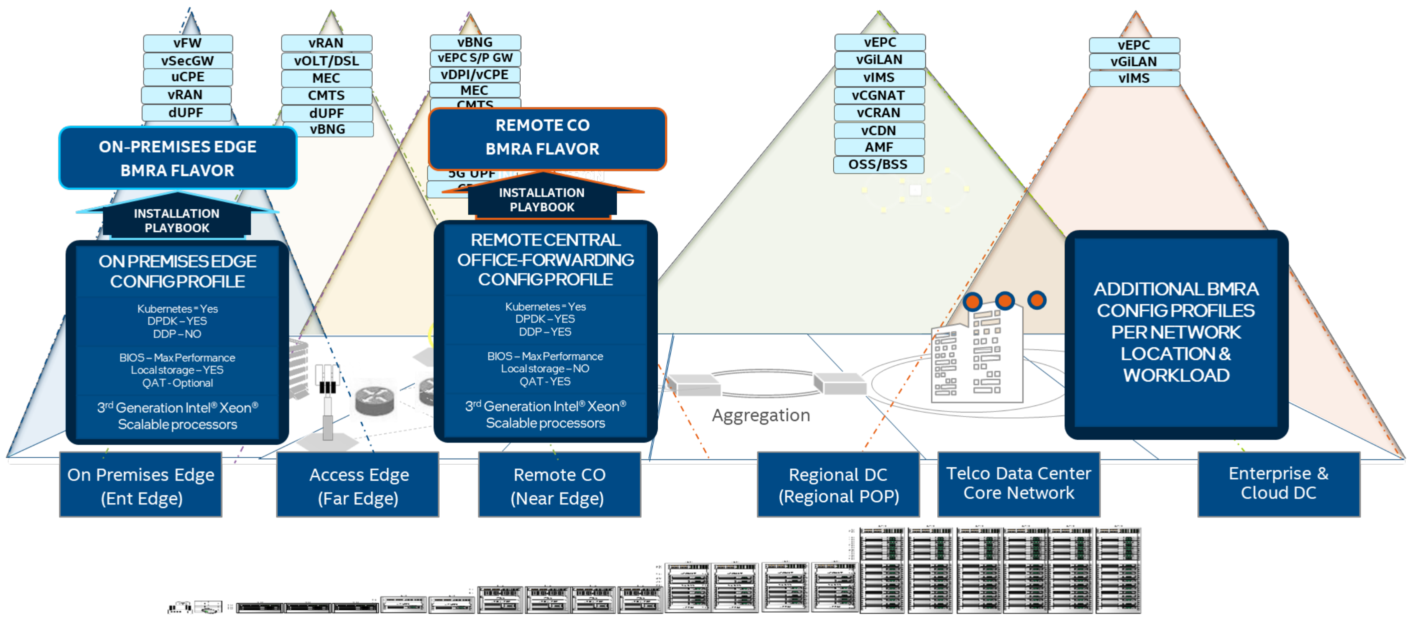


Figure 3. Configuration Profile Implementation Examples

1.3.1 Key Terms

Table 1 lists the key terms used throughout the portfolio. These terms are specific to Network and Edge Reference System Architectures Portfolio deployments.

Table 1. Terms Used

Term	Description
Experience Kits	Guidelines delivered in the form of manuals, user guides, application notes, solution briefs, and training videos for best-practice implementation of cloud native and Kubernetes technologies to ease developments and deployments.
Network and Edge Reference System Architectures Portfolio	A templated system-level blueprint for a range of locations in enterprise and cloud infrastructure with automated deployment tools. The portfolio integrates the latest Intel platforms and cloud-native technologies for multiple deployment models to simplify and accelerate deployments of key workloads across a service infrastructure.
Deployment Model	Provides flexibility to deploy solutions according to business and IT needs. The portfolio offers three deployment models: ▪ **Container Bare Metal Reference System Architecture (BMRA)** – A deployment model of a Kubernetes cluster with containers on a bare metal platform. ▪ **Virtual Machine Reference System Architecture (VMRA)** – A deployment model of a virtual cluster on a physical node. The virtual cluster can be a Kubernetes containers-based cluster. ▪ **Cloud Reference System Architecture (Cloud RA)** – A deployment model that uses a CSP's Intel-based instances for running cloud-native applications in the cloud. The worker instances are provided based on the Configuration Profile that workload demands.
Configuration Profiles	A prescribed set of components—hardware, software modules, hardware/software configuration specifications—designed for a deployment for specific workloads at a network location (such as Access Edge). Configuration Profiles define the components for optimized performance, usability, and cost per network location and workload needs[2]. In addition, generic Configuration Profiles are available for developers' flexible deployments.
Reference System Architecture Flavor	An instance of reference architecture generated by implementing a Configuration Profile specification.
Ansible Playbook	A set of validated scripts that prepare, configure, and deploy a Reference System Architecture Flavor per Configuration Profile specification.
Configuration Profile Ansible Scripts	Automate quick, repeatable, and predictive deployments using Ansible playbooks. Various Configuration Profiles and Ansible scripts enable automated installations that are application-ready, depending on the workload and network location.
Kubernetes Cluster	A deployment that installs at least one worker node running containerized applications. Pods are the components of the application workload that are hosted on worker nodes. Control nodes manage the pods and worker nodes.

[2] Workloads and configurations. Results may vary.

Term	Description
Intel® Platforms	Prescribes Intel platforms for optimized operations. The platforms are based on 4th and 5th Gen Intel® Xeon® Scalable processors, Intel® Core™ Ultra processors, Intel® Core™ processors, Intel Atom® processors, Intel® Xeon® D processors, and 4th Gen Intel® Xeon® Scalable processor with Intel® vRAN Boost. The platforms integrate Intel® Ethernet Controller 700 Series and 800 Series, Intel® QuickAssist Technology (Intel® QAT), Intel® Data Center GPU Flex Series, Intel® Arc™ Series GPUs, Intel® Optane™ technology, Intel® Infrastructure Processing Unit (Intel® IPU), and more. ***Note:*** This release of VMRA does not support the Intel® Xeon® D processor.

In addition to key terms, portfolio deployment procedures follow a hardware and software configuration taxonomy. Table 2 describes the taxonomy used throughout this document.

Table 2. Hardware and Software Configuration Taxonomy

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

1.3.2 Intel Investments of Capabilities

Intel investments in networking solutions are designed to help IT centers accelerate deployments, improve operational efficiencies, and lower costs. Table 3 highlights Intel investments in the portfolio and their benefits.

Table 3. Intel Capabilities Investments and Benefits

Capability	Benefit
Performance	Intel® platform innovation and accelerators, combined with packet processing innovation for cloud-native environments, deliver superior and predictive application and network performance.[3]
Orchestration and Automation	Implementing Kubernetes containers orchestration, including Kubernetes Operators, simplifies and manages deployments and removes barriers in Kubernetes to support networking functionality.
Observability	Collecting platform metrics by using, as an example, the collectd daemon and Telegraf server agent, publishing the data, and generating reports, enables high visibility of platform status and health.
Power Management	Leveraging Intel platform innovation, such as Intel® Speed Select Technology (Intel® SST) and Kubernetes Power Manager plug-in, supports optimized platform power utilization.
Security	Intel security technologies help ensure platform and transport security. These technologies include the following: ▪ Intel® Security Libraries for Data Center (Intel® SecL - DC) ▪ Intel® QuickAssist Technology Engine for OpenSSL*(Intel® QAT Engine for OpenSSL*) ▪ Intel® Software Guard Extensions (Intel® SGX) ▪ Intel® Trust Domain Extensions (Intel® TDX) on 5th Gen Intel® Xeon® Scalable processors ▪ Key Management Reference Application (KMRA) implementation
Storage	The following storage configurations are supported: ▪ Object storage - a disaggregated, high-performance, scalable storage platform using MinIO Object Storage supports data-intensive applications, such as media streaming, big data analytics, AI, and machine learning. ▪ Local block storage - for containerized workloads that implement Local Persistent Volume Static Provisioner (LPVSP) for reliable container-attached storage by turning any storage available on Kubernetes worker nodes into local or distributed persistent volumes (PVs). ▪ Rook/Ceph – Ceph is a scalable distributed storage system that provides a unified storage service with object, block, and file interfaces from a single cluster. Rook turns distributed storage systems into self-managing, self-scaling, self-healing storage services and automates the tasks of a storage

[3] Workloads and configurations. Results may vary.

Capability	Benefit
	administrator: deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management.
Service Mesh	Implementing a service mesh architecture using Istio allows application services that can be added, connected, monitored, more secure, and load-balanced with few or no code changes. Service mesh is integrated with Trusted Certificate Service for Kubernetes* platform, providing secure key management.

1.4 Deployment Process

This user manual provides the foundational information for deployments. After understanding the concepts, refer to one of the following, depending on the intended deployment scenario.

- Kubernetes containers deployments: Network and Edge Container Bare Metal Reference System Architecture User Guide
- Virtualized system deployments: Network and Edge Virtual Machine Reference System Architecture User Guide
- Cloud deployments: Network and Edge Cloud Reference System Architecture User Guide

Intel also provides Experience Kits that detail the technologies enabled in the Reference System Architectures Portfolio, including benchmark information. Experience Kits are available on Intel® Network Builders at Network & Edge Platform Experience Kits.

For NDA material, contact your local Intel representative.

2 Reference System Architectures Overview

2.1 Deployment Model Options

The Network and Edge Reference System Architectures Portfolio offers three deployment options: BMRA, VMRA, and Cloud RA. These options are designed for the convergence of key applications and services, control plane, and high-performance packet processing functions in an infrastructure. Tools such as testpmd and pktgen pods and sample cloud-native network function (CNF) workloads (vCMTS and vBNG) are used to validate the Reference System Architectures delivered.

The BMRA is a scalable, open Kubernetes cluster composed of physical worker and control nodes networked through switches and connected, as depicted in Figure 4.

The VMRA is a scalable virtual cluster supporting both a virtual Kubernetes cluster and a VMRA cluster with a scalable number of VMs. The VMs are connected as a virtual cluster of worker and control VMs within the single node, as depicted in Figure 5, and on multiple nodes, as depicted in Figure 6.

The Cloud RA is a scalable cluster on a public cloud. Its architecture is shown in Figure 7 and Figure 8. The Cloud RA provides the means to develop and deploy cloud-native applications in a CSP environment (public cloud) and still experience Intel technology benefits. This release supports Amazon Web Services (AWS) EKS and the Microsoft Azure* Kubernetes Service (AKS) using either HashiCorp* Terraform or their native CLI.

The BMRA and VMRA are tuned to support diverse deployment scenarios. Based on the application and network location, a BMRA or VMRA can be configured for optimal operation and performance using one of the provided Configuration Profiles[4]. Ansible playbooks are defined based on those Configuration Profiles, allowing fast and easy auto-configuration and auto-provisioning of the Reference System Architectures.

[4] Workloads and configurations. Results may vary.

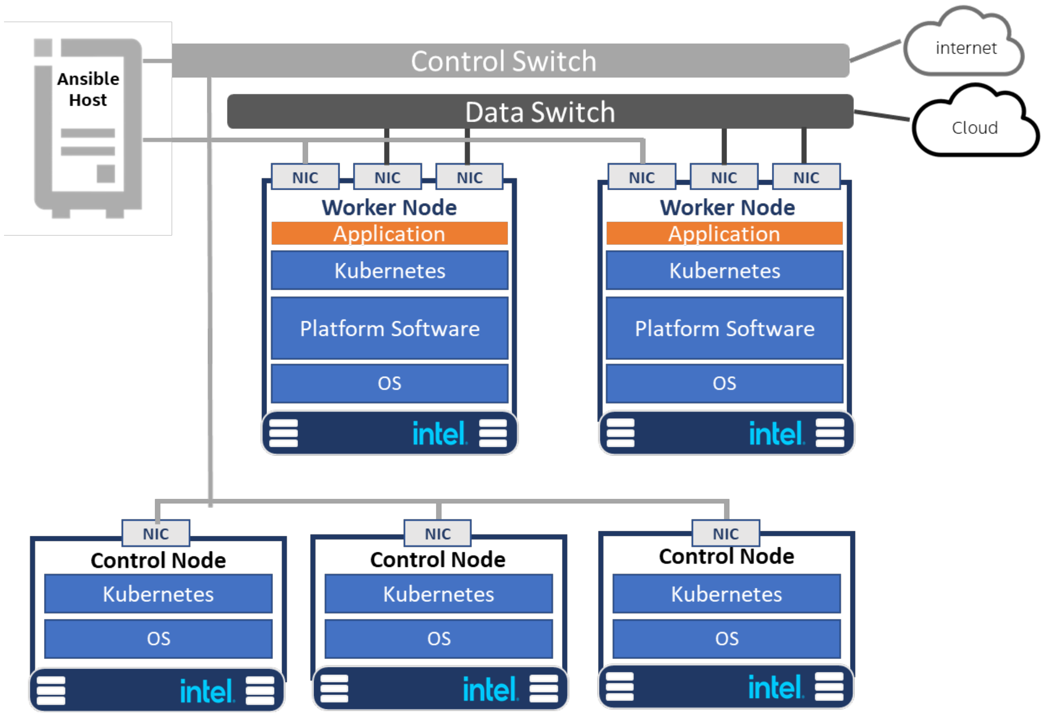


Figure 4. Container Bare Metal Reference System Architecture (BMRA)

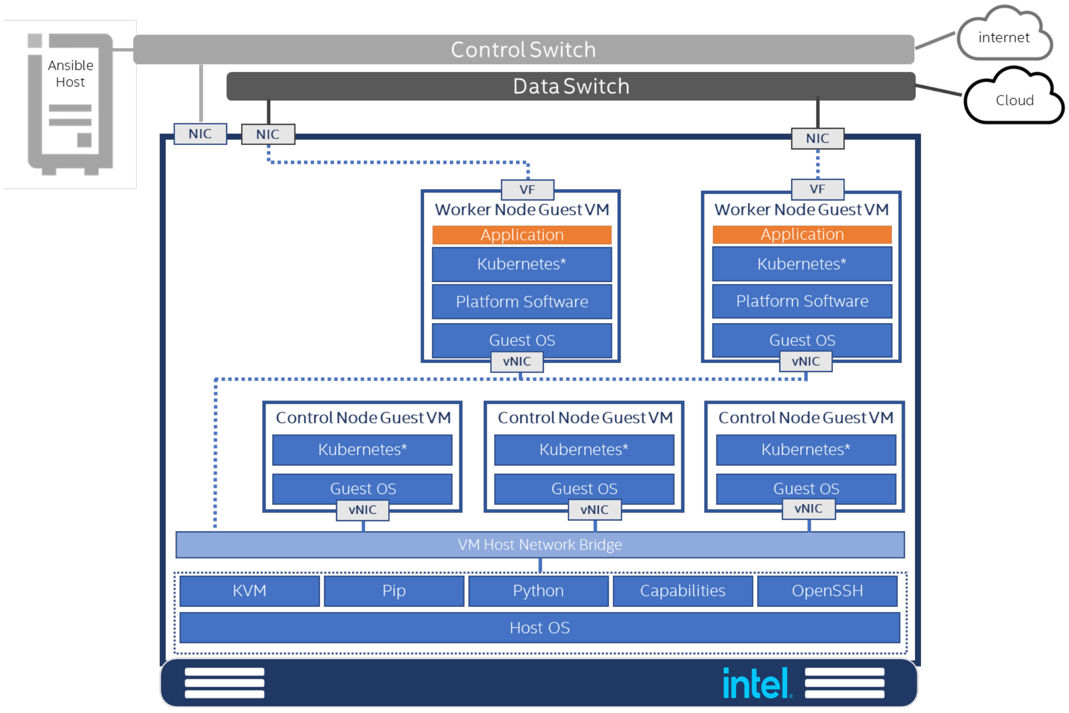


Figure 5. VMRA – Virtual Cluster on a Single Node

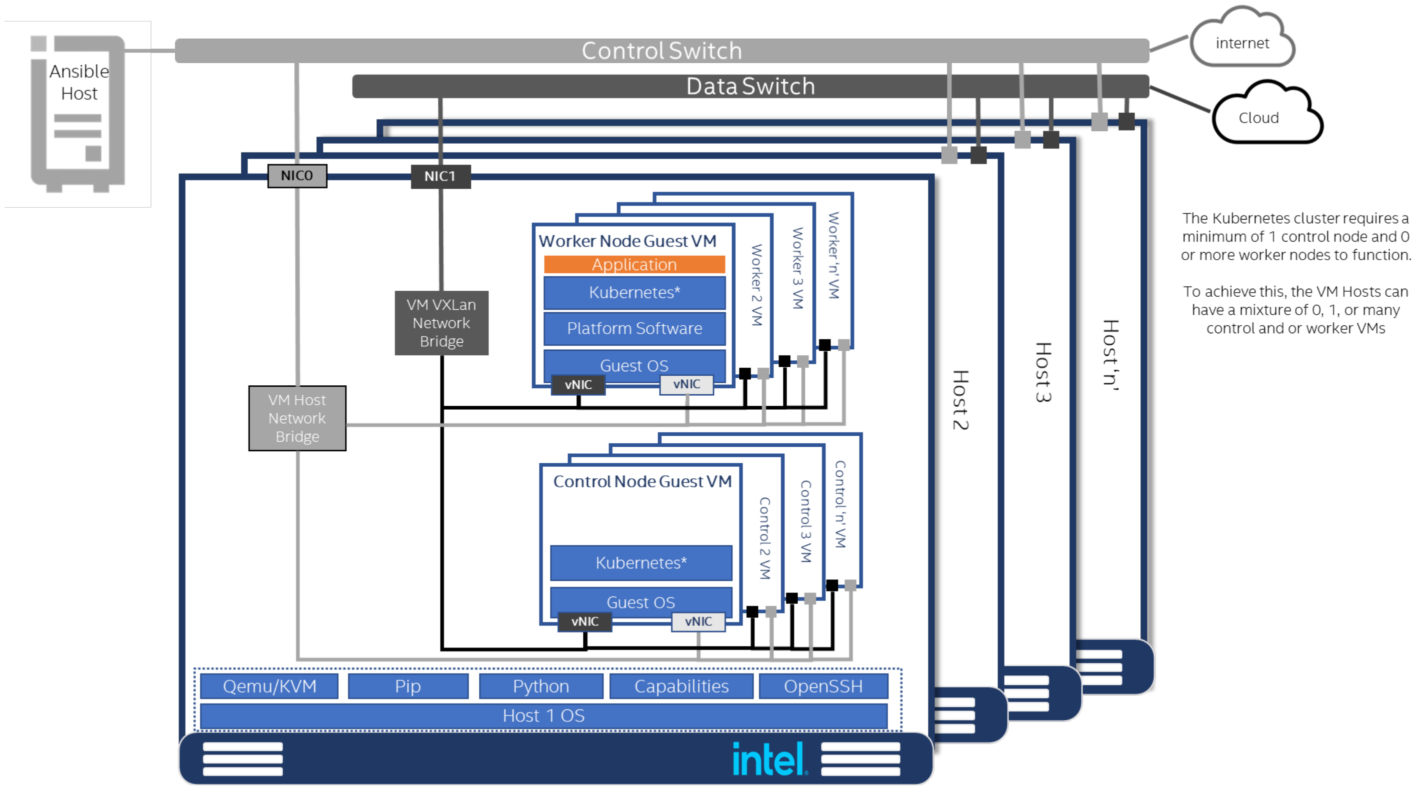


Figure 6. VMRA – Virtual Cluster on Multiple Nodes

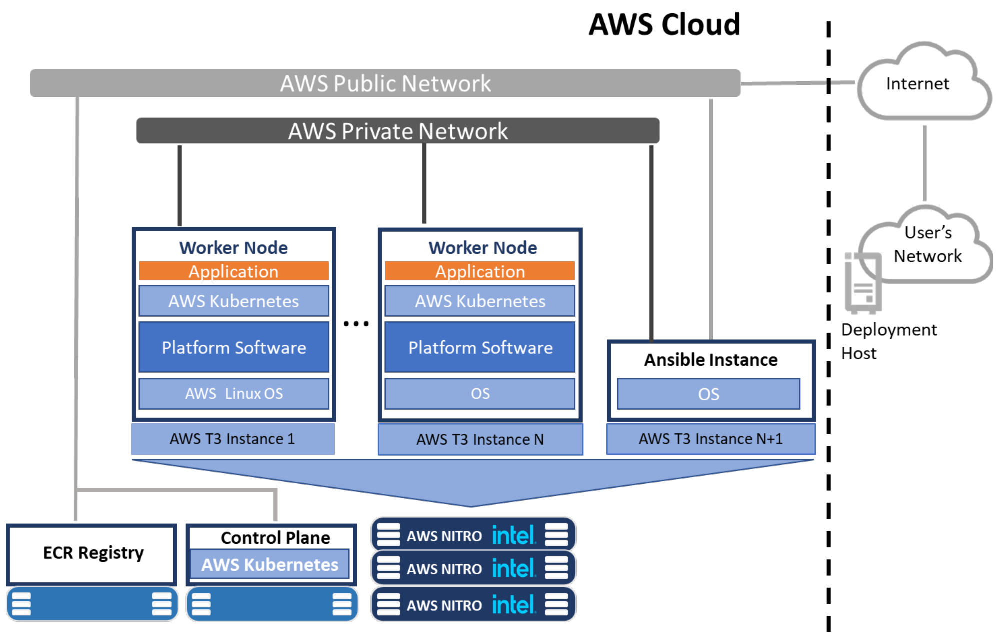


Figure 7. Cloud RA for Amazon EKS

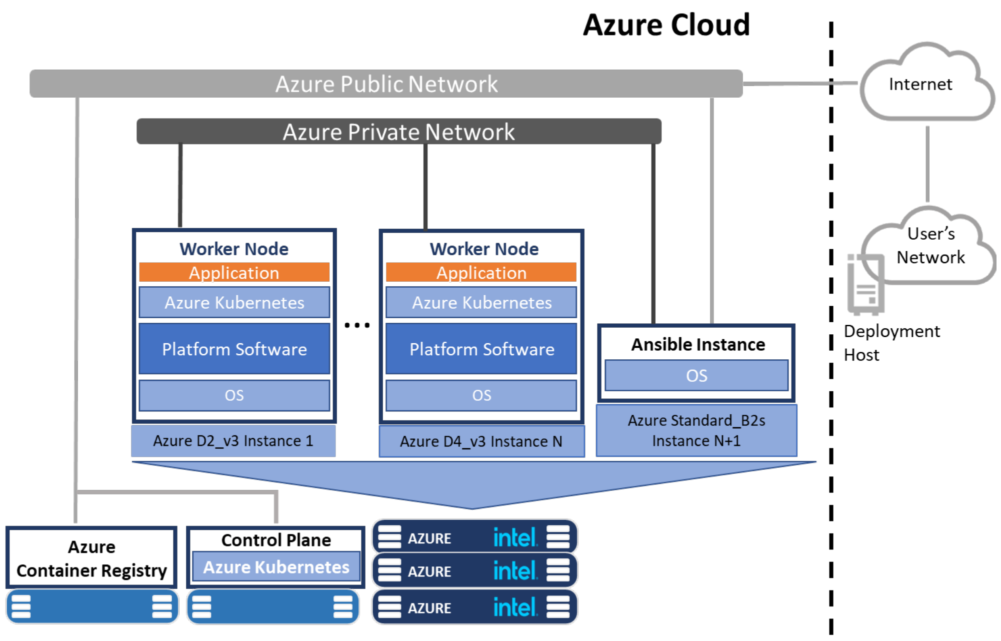


Figure 8. Cloud RA for Microsoft Azure Kubernetes Service (AWS)

2.2 Key Elements

The main elements of the Reference System Architectures include the following (refer to Figure 9 for a BMRA example):

- **Hardware Components:** Multiple platform hardware options are available, including a variety of 5th and 4th Gen Intel® Xeon® Scalable processor SKUs, Intel® Xeon® D processor SKUs, Intel® Core™ Ultra processors, Intel® Core™ processors, Intel Atom® processors, Intel® Ethernet Network Adapters, Intel® QAT, Intel® Data Center GPU Flex Series, and Intel® Arc™ Series GPUs. BIOS options are listed in each Reference System Architecture user guide. Deployment engineers should refer to the appropriate user guide during deployment to select and configure optimal BIOS values before cluster provisioning. For details, see Section 3 - Hardware Components.
- **Software Capabilities:** The software capabilities are based on open-source software delivered by cloud-native and CNCF communities driving Kubernetes, Istio, observability, DPDK, FD.io. OVS, OVS-DPDK, and through Intel GitHub. Options for Red Hat * Enterprise Linux* (RHEL) and Ubuntu Linux operating systems are available. Container environments use Docker container runtime, containerd, and CRI O. For details, see Section 4 – Software Components.
- **Configuration Profiles:** Specific hardware and software configurations are provided in the Configuration Profiles based on Intel assessment and verification.
- **Installation Playbooks:** Ansible playbooks implement the best-practice configuration and setup for each Configuration Profile to ease and accelerate installation. For more details about the hardware and software configuration options available and ready to use, see Section 5 – Provisioning a Reference Architecture.

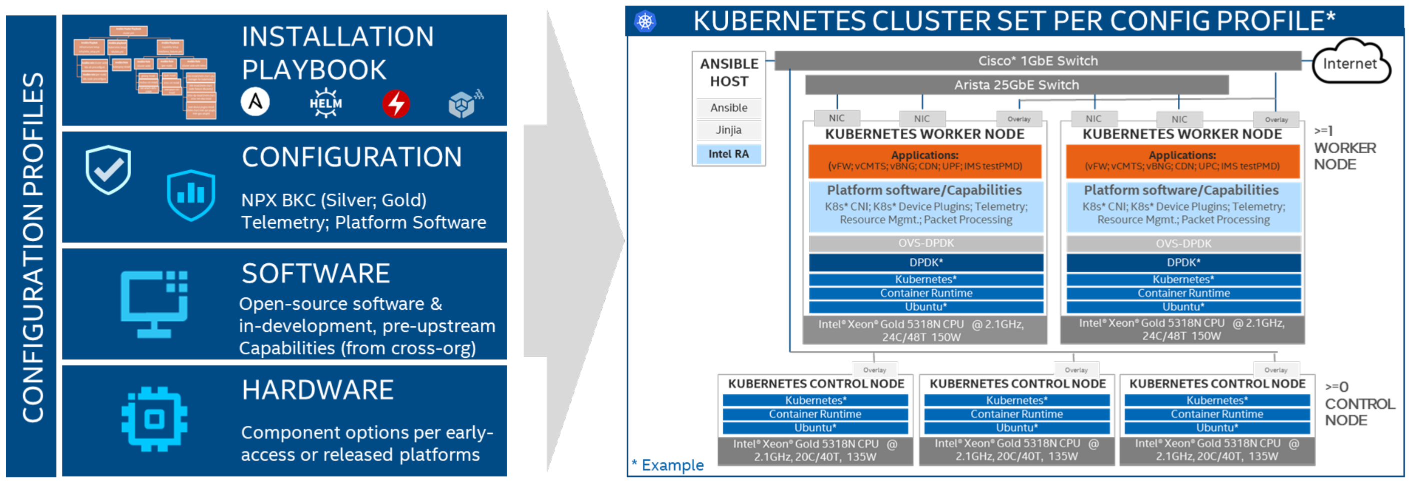

CONFIGURATION PROFILES
INSTALLATION PLAYBOOK
CONFIGURATION
NPX BKC (Silver; Gold)
Telemetry; Platform Software
SOFTWARE
Open-source software & in-development, pre-upstream Capabilities (from cross-org)
HARDWARE
Component options per early-access or released platforms
KUBERNETES CLUSTER SET PER CONFIG PROFILE*
ANSIBLE HOST
Ansible
Jinja
Intel RA
Cisco* 1GbE Switch
Arista 25GbE Switch
Internet
KUBERNETES WORKER NODE
Applications: (vFW; vCMTS; vBNG; CDN; UPF; IMS testPMD)
Platform software/Capabilities
K8s* CNI; K8s* Device Plugins; Telemetry; Resource Mgmt.; Packet Processing
OVS-DPDK
DPDK*
Kubernetes*
Container Runtime
Ubuntu*
Intel® Xeon® Gold 5318N CPU @ 2.1GHz, 24C/48T 150W
>=1 WORKER NODE
KUBERNETES CONTROL NODE
Intel® Xeon® Gold 5318N CPU @ 2.1GHz, 20C/40T, 135W
>=0 CONTROL NODE
* Example

Figure 9. Reference Architecture Key Elements – BMRA Example

2.3 Configuration Profiles

2.3.1 Configuration Profiles Overview

A Configuration Profile describes a specific hardware and software BOM and configurations applicable for a specific deployment (Figure 3). Configuration Profiles take into consideration the best-known configuration (BKC) validated by Intel for optimized performance.[5]

Each Configuration Profile includes installation scripts to deploy the required components for a Reference Architecture Flavor.

Installation scripts are available to deploy the required components for a Reference Architecture Flavor. Each Reference Architecture is built on the following:

- **Intel Platform foundation** with Intel processors and technologies.
- **Hardware BOM** optimized for delivering an application at a specific location using a deployment model. For example, to support a UPF workload at the Remote CO, the BMRA deployment is populated with the maximum available network interface cards.
- **Software BOM** leverages the Intel platform and enables cloud-native adoption.
- **Installation (Ansible) Playbook** automates the installation of a Reference Architecture Flavor per a Configuration Profile specification.

This version of the Network and Edge Reference System Architectures Portfolio includes the following Configuration Profiles that support per network locations.

- On-Premises Edge Configuration Profile
- On-Premises VSS Configuration Profile
- On-Premises AI Box Configuration Profile
- On-Premises SW Defined Factory Configuration Profile
- Access Edge Configuration Profile
- Remote Central Office-Forwarding Configuration Profile
- Regional Data Center Configuration Profile

Additional Configuration Profiles support non-location-specific deployments: Basic and Build-Your-Own.

2.3.2 On-Premises Edge Configuration Profile (Customer Premises)

This Configuration Profile is designed for a small cluster of stationary or mobile server platforms, ranging from one to four servers. Usage scenarios include data collection from sensors, local (edge) processing, and upstream data transmission. Sample locations are hospitals, factory floors, law enforcement, media, cargo transportation, and power utilities.

This Configuration Profile recommends a Kubernetes cluster hardware configuration, software capabilities, and specific hardware and software configurations that typically support enterprise edge workloads used in Smart City deployments, CDN, media analytics, and ad insertion. This Configuration Profile is designed to meet the following:

- Optimized data processing (to gain insights and reduce upstream transmission volumes) is priority

[5] Workloads and configurations. Results may vary.

- Power efficiency is important in most use cases
- Automated infrastructure orchestration capabilities are mandatory
- Complex media analytics pipelines at the Edge are required

2.3.3 On-Premises VSS Configuration Profile (Customer Premises)

This Configuration Profile is designed for a small cluster of stationary or mobile server platforms, ranging from one to four servers. Usage scenario is for customer premises deployment supporting Video Structuring Server (VSS).

This Configuration Profile recommends a Kubernetes cluster hardware configuration, software capabilities, and specific hardware and software configurations that typically support enterprise edge workloads used by media analytics. This Configuration Profile is designed to meet the following:

- 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
- Complex media analytics pipelines at the Edge are required

2.3.4 On-Premises AI Box Configuration Profile (Customer Premises)

This Configuration Profile is designed for a small cluster of stationary or mobile server platforms, ranging from one to four servers. Usage scenario is for customer premises deployment supporting Intel® Edge Ai Box.

This Configuration Profile recommends a Kubernetes cluster hardware configuration, software capabilities, and specific hardware and software configurations that typically support enterprise edge workloads used by media analytics. This Configuration Profile is designed to meet the following:

- 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
- Complex media analytics pipelines at the Edge are required

2.3.5 On-Premises SW Defined Factory Configuration Profile (Customer Premises)

This Configuration Profile is designed for a small cluster of stationary or mobile server platforms, ranging from one to four servers. Usage scenarios include data collection from sensors, local (edge) processing, and upstream data transmission. Sample location tuned for factory floors.

This Configuration Profile recommends a Kubernetes cluster hardware configuration, software capabilities, and specific hardware and software configurations that typically support industrial workloads used in factory deployments. This Configuration Profile is designed to meet the following:

- 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
- Complex media analytics pipelines at the Edge are required

2.3.6 Access Edge Configuration Profile (Far Edge)

The Access Edge Configuration Profile sets an edge computing infrastructure primarily targeting cellular access networks. This Configuration Profile is designed for a small, secure data processing cluster for high-speed and low latency radio access service. These nodes represent a large portion of the total number of network elements for an operator with equipment possibly being in an outside plant in harsh, minimally controlled temperature cabinets, making power management a necessity. The reference architecture uses the Access Edge configuration to set up the infrastructure for virtualized radio access networks (vRAN) and FlexRAN™ software solutions.

This RA Configuration Profile is designed to meet the following:

- Optimal balance of power and packet performance for the given application
- Platform and network security
- Low latency with high-speed packet processing
- Automated infrastructure orchestration capabilities are mandatory

2.3.7 Remote Central Office-Forwarding Configuration Profile (Near Edge)

This Configuration Profile is designed for clusters ranging from a half rack to a few racks of servers, typically in a pre-existing, repurposed, unmanned structure. Usage scenarios include running latency-sensitive applications near the user (for example, real-time gaming, stock trading, video conferencing).

This Configuration Profile addresses an installation using Kubernetes cluster hardware, software capabilities, and configurations that help enable high performance for packet forwarding. This category could contain workloads, such as UPF, vBNG, and vCMTS. This Configuration Profile is designed to meet the following:

- 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 might be required
- Operational automation is important (site is unmanned)

2.3.8 Regional Data Center Configuration Profile (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 5G core (5GC) and media visual transcoding. It is designed to meet the following:

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

2.3.9 Non-Location-Specific Configuration Profiles

In addition, the following Configuration Profiles are not specific to network location. They are provided to allow maximum configuration flexibility:

- **Basic Configuration Profile:** Minimum hardware and software capabilities required to support a BMRA Kubernetes cluster setup.
- **Build-Your-Own Configuration Profile:** A complete set of all available software features targeted at developers and deployers who are looking to evaluate, control, and configure all the software and hardware ingredients and dependencies individually.

2.4 Ansible Playbooks

Intel provides a set of Ansible Playbooks using Ansible scripts and Helm charts that enable easy and fast automatic installation[6] of a BMRA and a VMRA. The Ansible playbooks enable users to customize multiple parameters to fit their installation requirements. Each of the Ansible playbooks executes the configuration defined by a Configuration Profile.

- Ansible is an agentless configuration management tool that uses playbooks to perform actions on many machines.
- Helm is a package manager tool that runs on top of Kubernetes to automate the installation process of plugins and Kubernetes capabilities.

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

1. **Infrastructure Setup:** Addresses the initial system configuration, including telemetry setup.
   When deploying a VMRA, the infrastructure setup is run on both the host and in the virtual machines.
2. **Kubernetes Setup:** Deploys Kubernetes capabilities and Kubernetes add-ons via Kubespray or Rancher* RKE2.
3. **Capability Setup:** Installation and configuration of the system capabilities, with some using Helm charts.

Figure 10 provides a high-level view of the **BMRA Ansible Playbook**.

[6] Workloads and configurations. Results may vary.

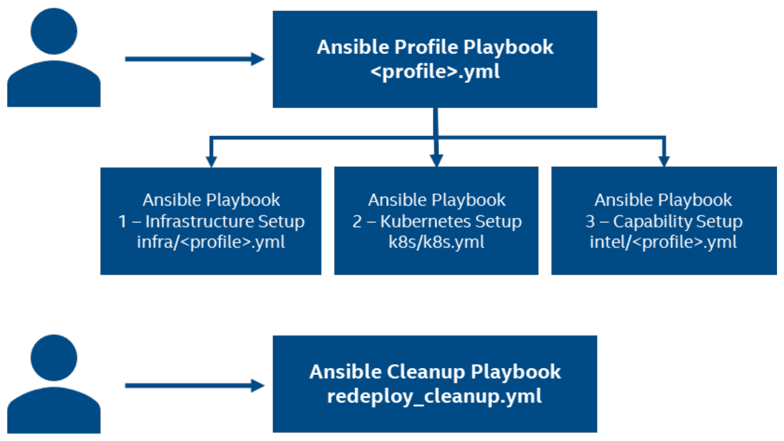


Figure 10. BMRA Ansible Playbook Overview

Figure 5 illustrates the VMRA Ansible Playbook. This playbook leverages some of the functionality from BMRA but includes an additional overlay Ansible playbook for setting up the VM infrastructure. Deployment includes the steps outlined in Figure 11. Note that a VMRA can be created without Kubernetes.

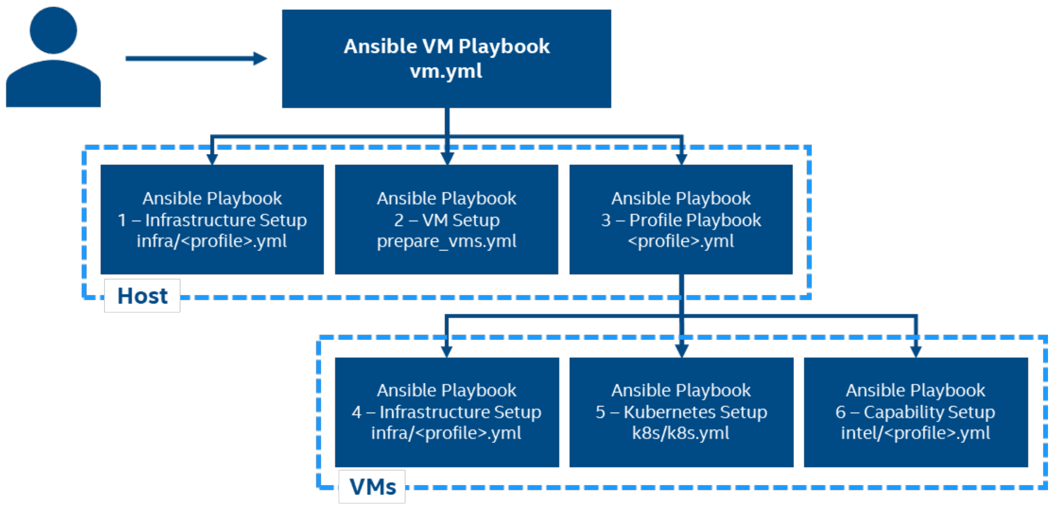


Figure 11. VMRA Ansible Playbook Overview

For more details about a playbook, refer to Section 3 of the appropriate user guide: Network and Edge Container Bare Metal Reference System Architecture User Guide and Network and Edge Virtual Machine Reference System Architecture User Guide.

3 Hardware Components

The BMRA and VMRA support a range of hardware that enables the different Configuration Profiles and deployment models. For all Configuration Profiles, possible hardware components for different nodes as well as the BIOS components available are listed in the appropriate user guide.

When implementing a Kubernetes cluster, the cluster typically consists of multiple worker nodes managed by one or more Kubernetes control nodes. The following tables show example listings of the hardware options for control and worker nodes. Worker nodes can be in the "base" configuration, or, if your configuration needs improved processing, you might choose to upgrade the CPU, network adapter, and memory.

This release of VMRA does not support the Intel Xeon D processor, 5th Gen Intel Xeon Scalable processor, Intel® Core™ processor, or Intel Atom® processor.

3.1 4th Gen Intel® Xeon® Scalable Processor Capabilities

Table 4 provides a list of the silicon capabilities of the 4th Gen Intel® Xeon® Scalable processor. For more information about the features enabled in the Reference System Architectures, see the appropriate user guide.

Table 4. Silicon Capabilities of 4th Gen Intel® Xeon® Scalable Processor

	Feature	Description
CPU/Accelerator	IAX	In-Memory Analytics accelerator provides effective bandwidth and fast offload by enhancing deeper compression than software-only techniques. IAX performs computationally demanding scan and filter operations in place of cores.
	QAT	Intel QuickAssist Technology is a set of accelerators for acceleration of bulk crypto, compression, and public key cryptography. Intel QAT provides acceleration for network security protocols such as IPSec and TLS, web and storage compression, and a more secure key protection for encrypted customer keys.
	DLB	Intel® Dynamic Load Balancer (Intel® DLB) offloads queue management from IA cores and provides dynamic, flow-aware balancing and reordering. It is primarily designed to help ensure balanced workload distribution, provide ultra-fast throughput and resource allocation, and reduce I/O and memory footprint.
	DSA	Intel® Data Streaming Accelerator (Intel® DSA) helps to enhance performance for workloads reliant on data movement by offloading data movement instructions to dedicated engines, allowing core cycles to be prioritized for other operations.
	CLDEMOTE	In Intel® architecture, L3 cache is shared across all cores and is non-exclusive to L2 and L1 cache. The CLDEMOTE instruction allows cache to be demoted from the L1 and L2 cache structure into the L3 shared cache. This allows data to be accessible by other cores and reduces the latency in snooping lower-level caches.
	MOVDIRI	MOVDIRI is an instruction for higher-throughput writes to memory. It is a more proficient mechanism for software to issue 4B/8B writes to MMIO-mapped register devices and is designed to improve streaming accelerator usages.
Power Management	Intel SST-PP/ Intel SST-TF	The 4th Gen Intel Xeon Scalable processor introduces an enhanced set of Intel Speed Select technologies. Intel® Speed Select Technology – Performance Profile (Intel® SST-PP) provides a set of performance profiles, while Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF) looks to optimize opportunistic excursions into turbo frequencies.
	Power Instructions – UMONITOR, UMWAIT, TPAUSE	These are a set of instructions designed to transition into a power optimized state, helping enable fast entry and exit latency from power states. UMONITOR instruction is ordered as a load operation with respect to other memory transactions, while UMWAIT instructs the processor to enter an implementation-dependent optimized state while monitoring a range of addresses. The TPAUSE functionality is similar to monitor-less UMWAIT.
Security	SGX	Intel® Software Guard Extensions (Intel® SGX) provides fine-grain data protection via application isolation in memory. Integrity provides more resistance against physical attacks.
	CryptoDev and CryptoNI	The cryptodev library provides a crypto device framework for management and provisioning of hardware and software crypto poll mode drivers, defining generic APIs that support several different crypto operations. The framework currently only supports cipher, authentication, chained cipher/authentication, and AEAD symmetric and asymmetric crypto operations.
RAS	RAS	Reliability, Availability, and Serviceability is a cornerstone of a highly performant network. The 4th Gen Intel Xeon Scalable processor provides a set of features to enhance platform resiliency, debuggability, manageability, and health monitoring. Telemetry further builds on the set of features to provide reporting from multiple layers of hardware and software.
ISA	FP-16	FP-16 provides floating-point instructions to enhance the existing range of Intel® Advanced Vector Extensions (Intel® AVX) instructions. These instructions are specific to network workloads that require greater precision for RAN-type algorithms and allow for simplified instructions calls.
	AMX (TMUL)	Advance Matrix Multiply is a scalable grid of computes designed to increase the volume of work done per instruction. Tile Multiply (TMUL) also provides a set of multiply instructions specific to Intel AVX enhancements.
	VP2INTERSECT	VP2INTERSECT is an instruction for Intel® Advanced Matrix Extensions 512 (Intel® AVX-512) for acceleration of sparse matrices multiplication. It generates controls from SIMD indices that Intel AVX-512 data can efficiently process using the controls.
I/O	CXL 1.1	Compute Express Link is an enhanced IO interface that provides a high-bandwidth bus for inter-device transfer. In addition, it also unlocks sharing of resources between devices.
	PCIe Gen5	PCIe Gen5, introduced in the 4th Gen Intel Xeon Scalable processor spec, provides around double the bandwidth from the previous generation.
Virtualization	S-IOV	Intel® Scalable I/O Virtualization (Intel® Scalable IOV) is an accelerator for communication between VMs and containers and I/O devices such as PCIe. Building on the SR-IOV architecture, it provides incremental scalability and higher performance and allows multiple

	Feature	Description
		guest OSes running simultaneously within a single platform to natively share PCIe devices like the DSA and network adapters.
	SVM	Shared Virtual Memory enables accelerator devices to operate in CPU virtual address space. It provides a consistent view of memory across host application and offloaded tasks. Offload operations can support data structures with pointers/references to other data structures. SVM avoids memory pinning and copying (marshalling) overheads. It supports I/O page fault through the IOMMU to enable memory over-commit via demand paging for both CPU and device access to memory.

3.2 5th Gen Intel® Xeon Scalable Processor Capabilities

The 5th Gen Intel Xeon Scalable processors feature an increase in processor cores and are pin compatible with the 4th Gen Intel Xeon Scalable processors, providing an easy migration path to take advantage of the processor's built-in workload accelerators, enhanced security features, and increased performance within the same power envelope. Customers who upgrade to 5th Gen Intel Xeon Scalable processors require minimal validation, speeding up their time to deployment.

The 5th Gen Intel Xeon Scalable processors also introduce Intel® Trust Domain Extensions (Intel® TDX). Intel TDX extends Virtual Machine Extensions (VMX) and Intel® Total Memory Encryption – Multi-Key (Intel® TME-MK) with a new kind of virtual machine guest called a trust domain (TD). A TD runs in a CPU mode that protects the confidentiality of its memory contents and its CPU state from any other software, including the hosting Virtual Machine Monitor (VMM).

3.3 Intel® Core™ Ultra Processor Capabilities

- Multiple compute engines in one SoC: P-cores, E-cores, Intel® Arc™ GPU, and Intel® AI Boost—an integrated neural processing unit (NPU) dedicated to AI.
- Built-in Intel Arc GPU with up to eight Xe-cores (up to 128 graphics execution units).
- Hardware-accelerated AV1 encode, integrated DisplayPort 2.1 (USB-C), and HDMI 2.1, new graphics system controller.

4 Software Components

Intel, with its partners and the hardware and developer communities, continues to enable Intel® architecture advancements for the cloud native ecosystem through support of **Kubernetes features**, extension of **Kubernetes plugins**, and development of **system hardware resources**.

Note: This release of VMRA does not support the Intel Xeon D processor.

4.1 Kubernetes Features

To enhance Kubernetes for network functions virtualization (NFV) and networking usage, Intel and its partners are developing capabilities and methodologies that expose Intel® architecture platform features for increased and predictive application and network performance[7]. Such features include the following:

- **Cilium** is an open-source software solution for providing, securing, and observing network connectivity between container workloads. Cilium is cloud native, using eBPF Kernel technology.
- **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.
- **Bond CNI** allows for aggregating multiple network interfaces into one logical interface.
- **Platform Aware Scheduling (PAS)** contains a group of related projects designed to expose platform-specific attributes to the Kubernetes scheduler.
- **Native CPU Manager** provides mechanisms for CPU core pinning and isolation of containerized workloads.
- **Topology Manager**, a native component of Kubernetes (starting with v1.16), enables other kubelet components to make resource allocation decisions using topology-based information.

[7] Workloads and configurations. Results may vary.

- **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, DSA, and SR-IOV device plugins, help boost performance and platform efficiency.
- **Kubernetes Operators** are software extensions to Kubernetes that use custom resources to manage applications and their components. Operators follow Kubernetes principles for resource automation, enabling automated installation.
- **CPU Control Plane Plugin** enables fine granular control of CPU resources in terms of CPU and/or memory pinning. A privileged daemonset is responsible for control of the cgroups for a given set of pods and containers whereas an agent is responsible for watching pods CRUD events.

4.2 Platform System Features

With the Kubernetes capabilities mentioned earlier, Intel is constantly developing new platform-level capabilities for enhanced and predictive 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 network adapter) 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 512 (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 – 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.
- **Intel® Speed Select Technology – Core Power (Intel® SST-CP)** allows an OS/VMM to control which cores can take advantage of any power headroom that is available in the system to help enable greater system performance.
- **Intel® Speed Select Technology – Performance Profile (Intel® SST-PP)** unlocks greater flexibility for defining core throughput by allowing greater granularity of core performance and a more effective balance between core count, thermals, and frequency.
- **Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF)** provides an ability to select a pool of cores to opportunistically access additional Turbo Mode frequencies (P0) for high throughput workloads.
- **Secure Cloud Native Network Platforms – Using Intel® Security Libraries for Data Center (Intel® SecL – DC)**, Reference System Architectures can help 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 (KMRA) with Intel® Software Guard Extensions (Intel® 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.
- **Intel QAT Engine for OpenSSL** is an option in the libcrypto general purpose cryptographic library. This implementation supports secure applications by directing the requested cryptographic operations to the hardware or software capability present on the underlying platform.
- **Intel® Data Center GPU Flex Series and Intel® Arc™ Series GPUs** – This is a discrete graphics procession unit based on Intel Xe architecture, which can accelerate media, graphics, and general compute usages. Media 21 supports MPEG-2, AVC, HEVC, and VP9 transcoding, plus AV1 transcoding. Graphics supports rendering acceleration with OpenGL*, OpenGL ES, and Vulkan* APIs. General compute usage supports computation acceleration with Intel® oneAPI Level Zero (Level Zero) and OpenCL™ APIs.

4.3 Container Runtimes

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 is a leading open-source system for automating deployment, scaling, and management of containerized applications.

A container runtime is software that allows Kubernetes to create and manage containers on a node (see Figure 12). The runtime usually consists of two parts: a high-level runtime that implements the container runtime interface (CRI) specification used by Kubernetes, and a low-level runtime that creates and runs the container processes on request through the Open Container Initiative (OCI) specification.

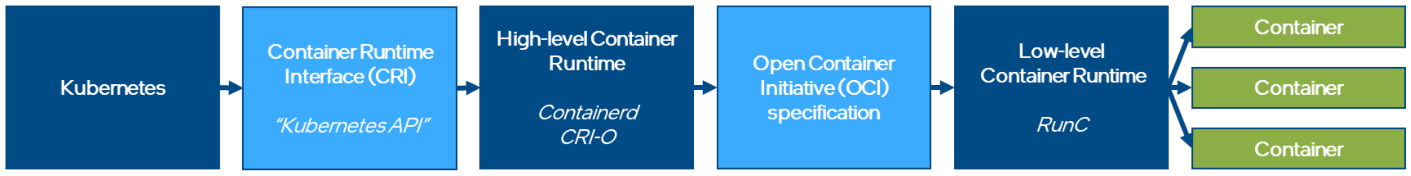

Kubernetes
Container Runtime Interface (CRI)
"Kubernetes API"
High-level Container Runtime
Containerd
CRI-O
Open Container Initiative (OCI) specification
Low-level Container Runtime
RunC
Container
Container
Container

Figure 12. Generic Kubernetes CRI Stack

One exception is Docker, which is more of a container platform or engine that also works as a stand-alone tool for running and managing containers (see Figure 13).

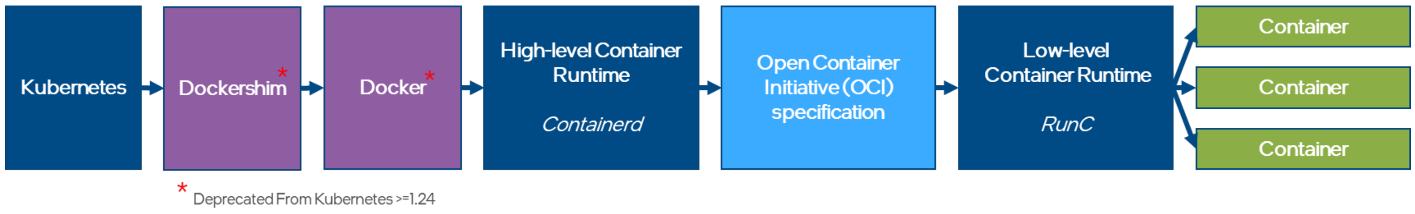

Kubernetes
Dockershim*
Docker*
High-level Container Runtime
Containerd
Open Container Initiative (OCI) specification
Low-level Container Runtime
RunC
Container
Container
Container
* Deprecated From Kubernetes >=1.24

Figure 13. Deprecated Kubernetes Runtime Stack Using Docker

Note: Docker was deprecated as a runtime for Kubernetes. For more information, see Check whether dockershim removal affects you.

4.4 Kubernetes Plugins

The following device plugins are used to advertise Intel® architecture system hardware resources to Kubernetes. Several of the plugins are container network interfaces (CNIs), which are explained here: CNI.

4.4.1 Multus CNI

Kubernetes natively supports only a single network interface. Multus is a CNI plugin specifically designed to support multiple networking interfaces in a Kubernetes environment. 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. 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. Note that the plugins SR-IOV and Userspace and Bond CNIs can be installed as part of the reference architecture deployment using an Ansible playbook. Multus v3.3 has been integrated with KubeVirt, officially recognized as a CNCF project, and officially released starting with Kubespray v2.12.

For more details, see Multus CNI.

4.4.2 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, the containerized 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.

For more details, see SR-IOV Network Device Plugin.

4.4.3 SR-IOV CNI

The SR-IOV device plugin enables partition of a single physical network adapter (PCI) resource into virtual PCI functions that can be attached to Kubernetes pods. To attach an SR-IOV device resource to the Kubernetes pod network, the SR-IOV CNI can be used. 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. The SR-IOV CNI can be omitted if the VF is attached to a containerized application through a userspace VF driver.

For more details, see SR-IOV CNI.

4.4.4 Userspace CNI

Userspace CNI provides a high-performance container networking solution and data plane acceleration for containers. It is 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.

4.4.5 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 a stand-alone plugin, interfaces are obtained from the host's network namespace. A bonded interface is created in the container network namespace. When used with Multus, interfaces that were previously passed to the container can be bonded.

4.4.6 Intel® QuickAssist Device Plugin

Intel® QuickAssist Adapters integrate hardware acceleration for 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.

For more details, see Intel Device Plugins for Kubernetes.

4.4.7 Intel® Software Guard Extensions (Intel® SGX) Device Plugin

The Intel SGX device plugin allows Kubernetes workloads to use Intel SGX on 4th and 5th Gen Intel® Xeon® Scalable processors. The plugin is used 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 that software is running inside an SGX enclave on a trusted computing node.

For more details, see Intel Device Plugins for Kubernetes.

4.4.8 Node Feature Discovery

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

- **CPUID:** NFD advertises CPU features such as Intel® Advanced Vector Extensions (Intel® AVX). Certain workloads, such as machine learning, might gain a significant performance improvement from these extensions.
- **SR-IOV networking:** NFD detects the presence of SR-IOV-enabled network adapters, allowing optimized scheduling of network-intensive workloads.
- **Intel® Resource Director Technology (Intel® RDT):** 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, Intel RDT helps to mitigate the effects of noisy neighbors. NFD detects the different Intel 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), 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.

For more details, see Node Feature Discovery Feature sources and Node Feature Discovery.

4.4.9 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 from such systems, 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 feature of Kubernetes distribution. It is a kubelet component that coordinates the set of components that are responsible for making topology-aligned resource allocations.

For more details, see Topology Management – Implementation in Kubernetes Technology Guide.

4.4.10 Kubernetes Native CPU Manager

Native CPU Manager for Kubernetes provides mechanisms for allocation of CPU cores to workloads in situations where pods contend for resources of the CPU. When contention happens, workloads might be moved to other CPUs and workload performance might 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.

Note: The CPU Manager for Kubernetes was deprecated in Kubernetes 1.22. Use the Kubernetes Native CPU Manager.

For more details, see Feature Highlight: CPU Manager.

4.4.11 CPU Control Plane Management

The CPU Control Plane Plugin is a K8s component that enables fine granular NUMA-aware control of CPU resources in terms of CPU and/or memory pinning. The component consists of two parts:

- A privileged daemonset responsible for control of the cgroups for a given set of pods and containers
- An agent responsible for watching pods CRUD events

For more details, see CPU Control Plane.

4.4.12 Platform Aware Scheduling

Platform Aware Scheduling (PAS) contains a group of related projects designed to expose platform-specific attributes to the Kubernetes scheduler using a modular policy-driven approach. PAS contains a core library and information for building custom scheduler extensions as well as specific implementations that can be used in a working cluster or leveraged as a reference for creating new Kubernetes scheduler extensions.

Telemetry Aware Scheduling (TAS) is the initial reference implementation of PAS. It can expose any platform-level metric to the Kubernetes Scheduler for policy-driven filtering and prioritization of workloads. You can read more about TAS here: Telemetry Aware Scheduling.

GPU Aware Scheduling is the implementation of the GPU resource-aware Kubernetes scheduler extension.

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 that calls 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 that it acts on – `scheduleonmetric` and `dontschedule` – and is implemented and configured as a Kubernetes Scheduler Extender. See Extending Kubernetes.
- **Telemetry Policy Controller** consumes TAS policies. The Telemetry Policy Controller parses TAS policies and places them in a cache to make them locally available to all TAS components. The controller 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 two strategy types.

- **`scheduleonmetric`** has only one rule. It is consumed by the Telemetry Aware Scheduler Extender and prioritizes nodes based on a comparator and an up-to-date metric value. For example:
```
scheduleonmetric when health_metric is LessThan
```
- **`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 to impact scheduling in a Kubernetes cluster. Using the Kubernetes descheduler, it can evict a workload that is breaking rules to have the workload placed on a more suitable node.

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 might 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 a 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 more details, refer to Telemetry Aware Scheduling (TAS) – Automated Workload Optimization with Kubernetes (K8s) Technology Guide.

4.4.13 Application Device Queues (ADQ) Plugin

The ADQ plugin is a software system that integrates with Kubernetes and allows dedicated hardware packet queues on Intel® Ethernet 800 Series Network Adapters to be assigned to containerized applications to provide user-configurable quality-of-service in terms of throughput, latency, and jitter to traffic flows. See the ADQ Resource Center for more information. The plugin for integrating ADQ into Kubernetes is available on GitHub.

4.4.14 Intent-Driven Orchestration

Intent-Driven Orchestration enables management of applications through their service level objectives, while minimizing developer and administrator overhead.

Intent-Driven Orchestration is new way to do orchestration – moving from an imperative model to an intent driven model in which the user expresses their intents in form of objectives (e.g., as required latency, throughput, or reliability targets) and the orchestration stack itself determines what resources in the infrastructure are required to fulfill the objectives. This approach benefits from community investments in scheduling (determining when and where to place workloads) and is augmented with a continuously running planning loop that determines what and how to configure the system.

4.4.15 KubeVirt*

KubeVirt* enables the orchestration and management of virtual machines (VMs) in an existing Kubernetes cluster, alongside containerized workloads. KubeVirt utilizes KVM and libvirtd to set up VMs, and the virtctl CLI to perform operations on the VMs that are not possible directly through the Kubernetes API.

Using the custom resource "VirtualMachine", the VMs can be configured in a similar way to pods and containers. This includes the metadata and specification details that are usually configured for pods, but also resource requests such as CPU, memory, and additional devices (e.g., SR-IOV) that are provided by device plugins. KubeVirt supports Multus for adding additional interfaces to VMs.

The KubeVirt feature is available in most of the Configuration Profiles and can be enabled through the `group_vars` configuration file by setting `kubevirt_enabled: true`. Additional information can be found on GitHub.

4.4.16 NGINX* Ingress Controller (ingress-nginx)

The NGINX* Ingress controller is an Ingress Controller implementation for NGINX that can load balance WebSocket, gRPC, TCP, and UDP applications. It supports standard Ingress features such as content-based routing and TLS/SSL termination. Several NGINX features are available as extensions to Ingress resources through Annotations and the ConfigMap resource.

4.5 Istio* Service Mesh

Istio is an open-source service mesh that layers transparently onto existing distributed applications.

"A service mesh is a dedicated infrastructure layer that you can add to your applications. It allows you to transparently add capabilities like observability, traffic management, and security, without adding them to your own code. The term 'service mesh' describes both the type of software that you use to implement this pattern, and the security or network domain that is created when you use that software."[8]

Istio's features provide a uniform and efficient way to help secure, connect, and monitor services. Istio is the path to load balancing, service-to-service authentication, and monitoring – with few or no service code changes. Its powerful control plane enables vital features, including:

[8] "The Istio Service Mesh." Istio. Accessed October 8, 2021. https://istio.io/latest/about/service-mesh/.

- More secure service-to-service communication in a cluster with TLS encryption, strong identity-based authentication, and authorization
- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection
- A pluggable policy layer and configuration API supporting access controls, rate limits, and quotas
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress

For more details, see The Istio service mesh.

The Reference System Architectures are shipped with a predefined set of Istio configurations as defined by the Istio community. These Istio configurations are implemented as part of the RA Configuration Profiles. The following Istio configuration profiles are available: default, minimal, external, empty, none. If no Istio configuration is defined by the variable `istio_service_mesh.profile` in the *group_vars/all.yml* file, then the default Istio configuration profile is applied. If no Istio configuration profile is set, then no specific Istio configuration profile is deployed.

It is also possible to deploy a custom Istio profile by placing a profile configuration YAML file to the service mesh role directory: `roles/istio_service_mesh/files/profiles/`. The name of the file must correspond to the profile name.

For more details on Istio configuration profiles, see Istio Installation Configuration Profiles.

4.5.1 Bypass TCP/IP Stack in Istio Service Mesh Using eBPF

This option provides acceleration of the Istio TCP/IP traffic in the following scenarios:

- Service to service communication when the services are in the same pod
- Service to service communication when the services are in the same node

Acceleration is accomplished using an eBPF program loaded on every node participating in the service mesh communication. To enable this feature, set *group_vars/all.yml* variable `istio_service_mesh.tcpip_bypass_ebpf` to `true`.

For more details, see Istio TCP/IP Bypass.

4.5.2 TLS Splicing

In TLS splicing, after a TCP connection is established between the client and Envoy, any subsequent TLS traffic is forwarded to an upstream server as raw TCP data. This is required for the client to access some external services where connection data should not be decoded by Envoy.

4.5.3 Istio Acceleration with Intel AVX-512 Crypto

Intel acceleration features enabled:

- Intel AVX-512 crypto acceleration for TLS connections
- Intel AVX-512 vector AES for symmetric data encryption

4.5.4 Istio Acceleration and Compression with Intel QAT 2.0

Intel acceleration features enabled:

- Intel QAT 2.0 crypto acceleration for TLS handshakes
- Intel QAT 2.0 compression acceleration for HTTPs connections

4.5.5 Istio mTLS Private Key Protection with SGX

The security of the private keys is ensured by the mTLS communication between service mesh workloads by the Intel® SGX. The private keys are stored and used inside the SGX enclaves and are never stored anywhere else in the system. Istio proxies can use the private key in the enclave by the key handle provided by SGX to communicate with other counterparts.

4.5.6 Istio Multi-tenancy

The Kubernetes cert-manager add-on automates the management and issuance of TLS certificates from various issuing sources. A user can use cert-manager to approve and sign certificate signing requests (CSRs) for workloads that are deployed in service mesh. This means that different workloads owned by a particular tenant can hold different certificate authorities (CAs).

4.6 Linkerd Service Mesh

Linkerd is provided as an optional replacement for the Istio service mesh. Rather than using Envoy as a proxy, Linkerd uses its own proxy written from the ground up in Rust.

While this release of the Reference Architecture enables a vanilla Linkerd installation, future releases plan to expose additional configuration options and telemetry.

4.7 Kubernetes Operators

Kubernetes operators are subject-specific controllers that extend the functionality of the Kubernetes API to create, configure, and manage complex entities on behalf of a Kubernetes user. Kubernetes is designed from the ground up for automation as it includes native mechanisms for deploying, running, and scaling workloads. However, some workloads and services require deeper knowledge of how the system should behave outside the bounds of core Kubernetes. Operators are purpose-built with operational intelligence to address the individuality of such constructs. Tools for operator creation help developers build an automation experience for cluster administrators and end users. By extending a common set of Kubernetes APIs and tools, Kubernetes operators can help provide ease of deployment and streamlined Day 1 and Day 2 operations. The Kubernetes operator pattern has emerged as a solution to help ensure that the automation of these function-specific applications is possible in a Kubernetes cluster.

4.7.1 SR-IOV Network Operator

The SR-IOV Network Operator is designed to help the user provision and configure the SR-IOV CNI plugin and device plugin in the Kubernetes cluster.

When the SR-IOV Network Operator is enabled by setting `sriov_network_operator_enabled: true` in the *group_vars/all.yml* file, it is mutually exclusive with SR-IOV CNI and device plugins. The plugins must be disabled by setting the respective options in *group_vars/all.yml* and *host_vars/<hostname>.yml*. The SR-IOV Network Operator is enabled by default, and SR-IOV CNI and device plugins are disabled by default.

To configure SR-IOV device custom resource definition (CRD), provide the `SriovNetworkNodePolicy`. The Reference System Architecture handles creation of such CRDs and creates and populates them automatically according to the provided configuration in the *host_vars/node1.yml* file, namely in the section `dataplane_interfaces`.

For more details, see SR-IOV Network Operator.

4.7.2 Intel Device Plugins Operator

The Intel device plugins operator is a Kubernetes custom controller. Its goal is to serve the installation and lifecycle management of Intel device plugins for Kubernetes. It provides cluster administrators with a single point of control for GPU, QAT, SGX, DSA, and FPGA devices. Device plugins are deployed by applying custom resources, and each device plugin has its own custom resource definition (CRD). The corresponding controller watches CRUD operations to those custom resources. Currently, the Reference System Architecture uses the Intel device plugins operator to deploy the Intel SGX device plugin and Intel GPU device plugin.

For more details, see Intel Device Plugins for Kubernetes.

4.7.3 MinIO Operator

MinIO provides the central process for servicing storage. However, to meet redundancy and scalability requirements, a storage node never operates in isolation. Rather, it is deployed in concert with a number of peers. Figure 14 represents this figuratively.

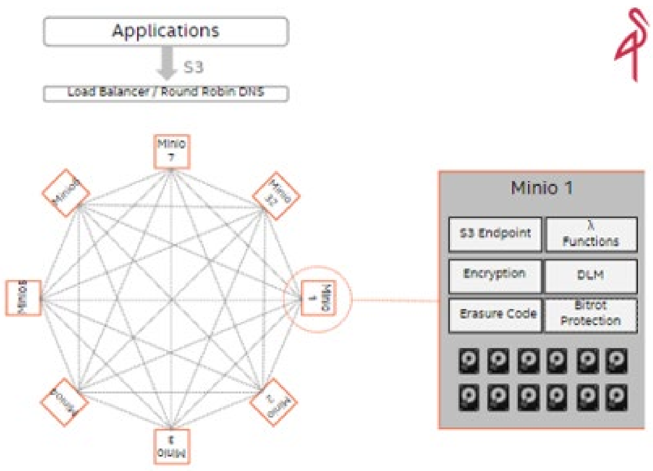


Figure 14. Storage Node Deployment Example

In the Reference System Architecture, four MinIO instances are selected for the cluster – the minimum requirement to ensure data integrity. However, this does not limit the user to expand beyond this number to address specific deployment and capacity needs.

MinIO itself is configured into a mesh – each knowing and operating in concert with its peers. It is the MinIO operator that in a Kubernetes environment orchestrates the rollout.

For more details, see MinIO Operator.

4.7.4 Intel® Ethernet Operator

The role of the Intel® Ethernet Operator is to orchestrate and manage the configuration of the capabilities exposed by Intel Ethernet E810 Series network adapters. The operator is a state machine that configures certain functions, monitors the status, and acts autonomously based on user interaction. The Intel Ethernet Operator provides functionality for:

- Update of the devices' FW via NVM Update tool
- Update of the devices' DDP profile
- Node Flow configuration of traffic handling for the devices, based on supported DDP profile
- Cluster Flow configuration supports cluster wide configuration of flow rules using a single CRD, as opposed to Node Flow Config, which supports node specific configuration of flow rules using a CRD per node

Figure 15 shows an overview of the architecture of the Intel Ethernet Operator, which includes the following:

- The operator provides APIs, controllers, and daemons for the management and execution of supported features.
- You interact with the operator by providing custom resources (CR) to Kubernetes that are constantly monitored to detect changes and act based on what is detected.
- Separate CRs are provided for the FW/DDP update functionality and the flow configuration functionality.
- After a CR is applied or updated, the operator checks if the configuration is already applied, and if it is not, it applies the configuration.

For more details, see Intel Ethernet Operator.

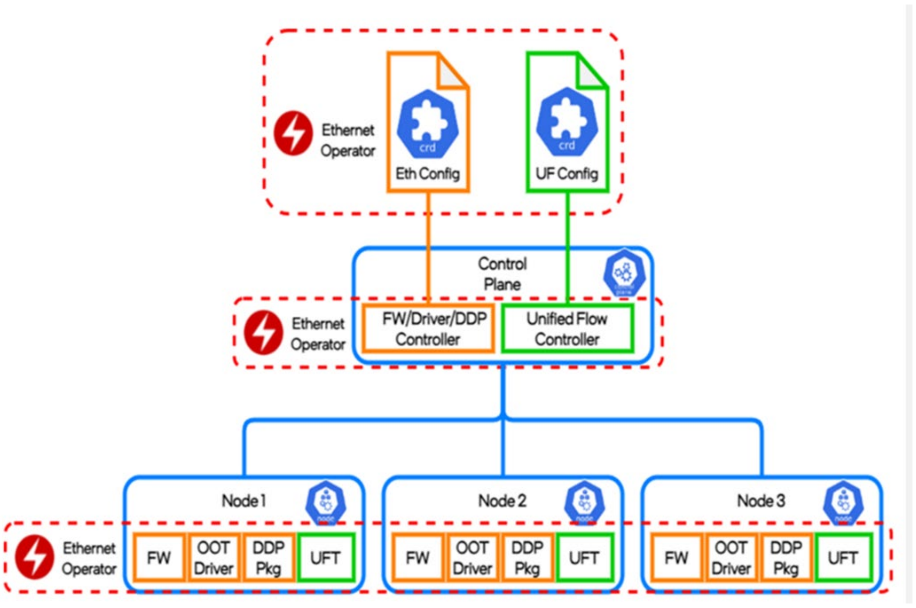


Figure 15. Intel® Ethernet Operator Architecture Overview

4.7.5 Forward Error Correction (FEC) Operator

The role of the FEC Operator for Intel Wireless FEC Accelerator is to orchestrate and manage the resources and devices exposed by a range of Intel's vRAN FEC acceleration devices and hardware within the Kubernetes cluster. The operator is a state machine that configures the resources and then monitors them; it acts autonomously based on user interaction.

This operator handles the management of the FEC devices used to accelerate the FEC process in vRAN L1 applications – the FEC devices are provided by a designated hardware (for example, FPGA or eASIC PCI extension cards). It provides functionality to create desired VFs (virtual functions) for the FEC device, binds them to appropriate drivers, and configures the VF's queues for desired functionality in 4G or 5G deployments. It also deploys an instance of the SR-IOV network device plugin that manages the FEC VFs as a Kubernetes cluster resource and configures this device plugin to detect the resources. The user interacts with the operator by providing a CR (Custom Resource). The operator constantly monitors the state of the CR to detect any changes and acts based on the changes detected. The CR is provided per cluster configuration. The components for individual nodes can be configured by specifying appropriate values for each component per "nodeName." After the CR is applied or updated, the operator/daemon checks if the configuration is applied already, and, if not it binds the PFs to the driver, creates the desired number of VFs, binds them to the driver, and runs the `pf-bb-config` utility to configure the VF queues to the desired configuration.

Additional information about the FEC Operator can be found at SR-IOV FEC Operator.

4.7.6 Kubernetes Power Manager

The Kubernetes Power Manager is a Kubernetes operator that was developed to provide cluster users with a mechanism to dynamically request adjustment of worker node power management settings applied to cores allocated to the pods. The Kubernetes Power Manager bridges the gap between the container orchestration layer and hardware features enablement, specifically Intel SST.

For more details, refer to Kubernetes Power Manager.

4.8 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. Protocols also help improve packet processing efficiency, which can result in enhanced performance in specific use cases.[9]

Additional information about DDP can be found at Improve Packet Processing Efficiency with Dynamic Device Personalization (DDP) Feature Brief.

[9] Workloads and configurations. Results may vary.

4.8.1 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 (PF0), but the configuration is applied to all ports of the adapter.

For Intel Ethernet 700 Series Network Adapters, the Reference System Architectures provide the following DDP images:

- `ecpri.pkg (eCPRI)`
- `esp-ah.pkg (ESP, AH)`
- `ppp-oe-ol2tpv2.pkg (PPPoE)`
- `mplsogreudp.pkg (MPLSoGRE/MPLSoUDP)`
- `gtp.pkgo (GTPv1)`

For more information, see the following: Dynamic Device Personalization for Intel® Ethernet 700 Series.

4.8.2 DDP on Intel® Ethernet 800 Series Network Adapters

Intel® Ethernet 800 Series Network Adapters support a broad range of protocols in DDP profile images. 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 Table 5.

Table 5. DDP Default Package Profiles

Default Packages
MAC
EtherType
VLAN
Ipv4
Ipv6
TCP
ARP
UDP
SCTP
ICMP
ICMPV6
CTRL
LLDP
VXLAN-GPE
VxLAN (non-GPE)
Geneve
GRE
NVGRE
RoCEv2
Comms Package
Extends default profile package with the following protocols:
GTP
PPPOE
L2TPv3
Ipsec
PFCP

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

SR-IOV devices (VFs included) are exposed to the Kubernetes pods via the SR-IOV device plugin (SR-IOV DP). In the Reference System Architectures, the plugin cannot discover loaded DDP plugins on Intel Ethernet 800-Series Network Adapters (SR-IOV Network Device Plugin Configurations).

As a result, a workload that requires a DDP profile cannot be orchestrated on such network adapters. A workaround can be accomplished in the following ways:

- Deploy a comms profile on every node with Intel® Ethernet 800 Series Network Adapter, or
- Deploy a 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 a specific DDP profile with specific Node Selector.

4.9 Intel® Speed Select Technology

Intel® Speed Select Technology (Intel® SST) is a collection of features that give more granular control over CPU performance.

4.9.1 Intel® Speed Select Technology – Base Frequency

Intel® Speed Select Technology – Base Frequency (Intel® SST-BF) is offered in 4th and 5th Gen Intel Xeon Scalable processors. Intel SST-BF controls the core frequency model (Figure 16). When enabled, some cores run at higher frequency and the remaining cores run at lower frequency.

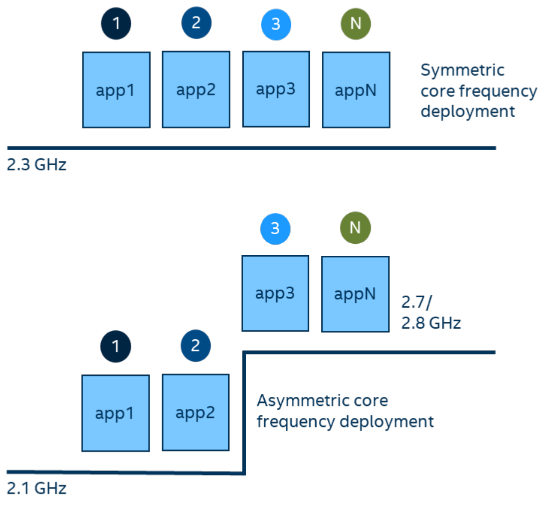

1
2
3
N
app1
app2
app3
appN
Symmetric core frequency deployment
2.3 GHz
3
N
app3
appN
2.7/ 2.8 GHz
1
2
app1
app2
Asymmetric core frequency deployment
2.1 GHz

Figure 16. 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 can pass data extremely fast to workers.[10]

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

For more information about scheduling workloads on Kubernetes with Intel SST-BF support, visit Intel® Speed Select Technology – Base Frequency (Intel® SST-BF) with Kubernetes Application Note.

In BMRA deployments, the core priority of high frequency cores is enabled by default.

For more information about Intel SST-BF technology, refer to Intel® Speed Select Technology – Base Frequency (Intel® SST-BF).

4.9.2 Intel® Speed Select Technology – Core Power

Intel® Speed Select Technology – Core Power (Intel® SST-CP), available on 4th and 5th Gen Intel Xeon Scalable processors, allows users to define priorities in core power flow in the event there is a power constraint scenario, for example TDP ceiling hit or power supply malfunction. Power prioritization helps cores maintain their frequency, while power is drawn from cores with lower priority.

Reference System Architectures allow users to define four Intel SST-CP Classes of Service that contain:

1. Priority
4. Affected Cores
5. Minimum CPU Frequency
6. Maximum CPU Frequency

Intel SST-CP configuration can be found in `host_vars`. Intel SST-CP enabled on a node is advertised by the Node Feature Discovery with the `sst_cp.enabled` label.

[10] Workloads and configurations. Results may vary

For more information about Intel® SST-CP technology, refer to Intel® Speed Select Technology – Core Power (Intel® SST-CP).

4.9.3 Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF)

Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF), available on 4th and 5th Gen Intel Xeon Scalable processors, allows users to assign specific cores to achieve a higher turbo frequency. This feature enables the ability to set different "All core turbo ratio limits" to cores based on priority. By using this feature, some cores can be configured for a higher turbo frequency by designating them as high priority at the cost of lower or no turbo frequency on the low priority cores. For this reason, this feature is only useful when the system is busy using all CPUs, but the user wants a configurable option to get high performance on some CPUs.

The support of Intel SST-TF depends on Intel SST-PP performance level configuration. It is possible that only a certain performance level supports Intel SST-TF. It is also possible that only the base performance level (level = 0) has the support of Intel SST-TF. Hence, first select the desired performance level to enable this feature.

For more information about Intel SST-TF technology, refer to Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF) and Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF) Overview User Guide.

4.9.4 Intel® Speed Select Technology – Performance Profile (Intel® SST-PP)

Intel® Speed Select Technology – Performance Profile (Intel® SST-PP), available on 4th and 5th Gen Intel Xeon Scalable processors, permits dynamic configuration of a server based on workload performance necessities. The working nature of Intel SST-PP is to set up CPUs that need to be online and others that need to remain offline to sustain a guaranteed frequency performance level. Intel SST-PP provides the ability to monitor changes in frequency levels dynamically.

Reference System Architectures provide two options for Intel SST-PP deployment:

- Configure Intel SST-PP with all features (Intel SST-BF, Intel SST-CP, and Intel SST-TF) in auto mode, which enables turbo frequency for all available online CPUs automatically.
- Configure Intel SST-PP with all features (Intel SST-BF, Intel SST-CP, and Intel SST-TF) with user-defined specific CPU ranges such as "2,3,5" to prioritize those CPUs over others.

Intel SST-PP configuration can be found in `host_vars`.

For more information about Intel SST-PP technology, refer to Intel® Speed Select Technology – Performance Profile (Intel® SST-PP).

4.10 Security

The following sections describe some of the security features in the Network and Edge Reference System Architectures Portfolio.

4.10.1 Native Kubernetes Cluster Security Features

Security features for the native Kubernetes cluster include the following:

- ubernetes audit trail and log backups by default
- Certs for API server access and timeouts
- Checksums for all downloaded artifacts
- RBAC enabled by default
- TLS cipher suites to help secure cluster communication
- Help secure the container registry by:
  - Limited access with TLS and basic authentication
  - Required server keys and certs signed with Kubernetes Certificate Authority (CA)
- Limited etcd permissions
- Restricted open firewall ports and subnets
- Pod Security Policies (PSP) admission controller enabled 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 that a kubelet can modify
  - PodSecurityPolicy
- Security policies on each Helm chart for deployment (collectd, NFD, TAS, and the like)

For additional security considerations, refer to:

- Securing a Cluster
- Pod Security Policies
- NodeRestriction

4.10.2 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.

By using Intel SecL – DC, Hardware Root of Trust, and Secure Boot, you 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 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 Kubernetes. 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 Secure the Network Infrastructure – Secure Cloud Native Network Platforms User Guide.

4.10.3 Intel® Software Guard Extensions (Intel® SGX)

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. This feature is only available on 4th and 5th Gen Intel Xeon Scalable processors.

4.10.4 Intel® Trust Domain Extensions (Intel® TDX)

Intel TDX is an Intel technology that extends Virtual Machine Extensions (VMX) and Intel® Total Memory Encryption – Multi-Key (Intel® TME-MK) with a new kind of virtual machine guest called a trust domain (TD). A TD runs in a CPU mode that protects the confidentiality of its memory contents and its CPU state from any other software, including the hosting Virtual Machine Monitor (VMM). This feature is only available on 5th Gen Intel Xeon Scalable processors.

4.10.5 Key Management Reference Application (KMRA) with Intel® SGX

Key Management Reference Application (KMRA) is 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 (Figure 17).

The following description outlines the setup of KMRA infrastructure with Intel SGX for private key provisioning to an Intel SGX enclave on a 4th or 5th Gen Intel Xeon Scalable processor. This method uses the Public-Key Cryptography Standard (PKCS) #11 interface and OpenSSL.

The BMRA Ansible scripts automatically set up the KMRA infrastructure. The process 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 Intel SGX enclaves used by workloads. This general solution can work with any application or workload.

In the Network and Edge Reference System Architectures Portfolio, a sample NGINX workload/application is provided as an example of how the KMRA infrastructure can be used. The NGINX workload uses the private keys from the Intel SGX enclave to establish TLS connections. The private key is never in the clear; thus, certificate signing happens more securely inside the enclave.

Step-by-step instructions for deploying KMRA with BMRA are detailed in the Network and Edge Container Bare Metal Reference System Architecture User Guide. 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.

The Network and Edge Reference System Architectures – 5G vRAN Security Quick Start Guide provides another example of how client/server communication implementation via NETCONF is secured by Intel's KMRA using Intel SGX enclaves.

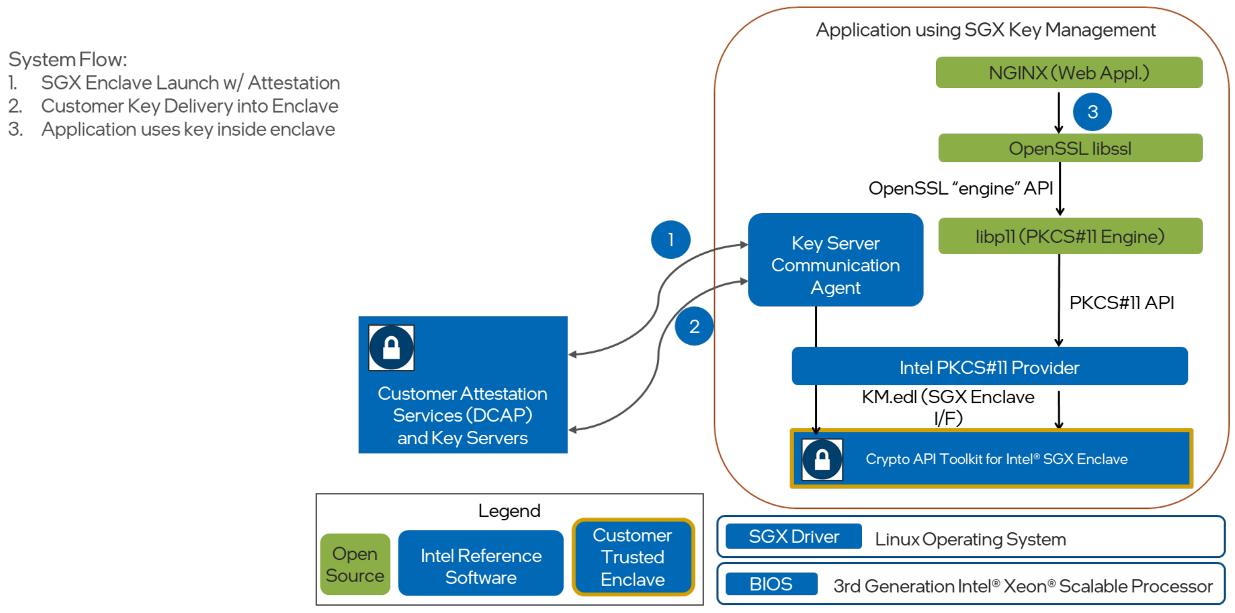

System Flow:
1. SGX Enclave Launch w/ Attestation
2. Customer Key Delivery into Enclave
3. Application uses key inside enclave
Application using SGX Key Management
NGINX (Web Appl.)
3
OpenSSL libssl
OpenSSL "engine" API
Key Server Communication Agent
libp11 (PKCS#11 Engine)
1
2
PKCS#11 API
Customer Attestation Services (DCAP) and Key Servers
Intel PKCS#11 Provider
KM.edl (SGX Enclave I/F)
Crypto API Toolkit for Intel® SGX Enclave
Legend
Open Source
Intel Reference Software
Customer Trusted Enclave
SGX Driver
Linux Operating System
BIOS
3rd Generation Intel® Xeon® Scalable Processor

Figure 17. KMRA Infrastructure with Intel® SGX

For more information, see Intel® Software Guard Extensions (Intel® SGX) – Key Management Reference Application (KMRA) on Intel® Xeon® Processors Technology Guide and Intel® Software Guard Extensions (Intel® SGX) – Key Management Reference Application (KMRA) on Intel® Xeon® Processors User Guide.

4.10.6 OpenSSL and QAT Engine

Network security vendors can rely on the OpenSSL project, and specifically the `libcrypto` general purpose cryptographic library, when executing in cloud compute environments. To address the need to have both portability and performance, Intel offers the Intel QAT Engine for OpenSSL as an additional option to the default library.

The Intel QAT Engine for OpenSSL provides support for secure applications by directing the requested computation of cryptographic operations to the available hardware acceleration or instruction acceleration present on the platform. The engine supports both the traditional synchronous mode for compatibility with existing applications and the asynchronous mode introduced in OpenSSL 1.1.0 to help achieve maximum performance.

4.10.7 Trusted Certificate Service

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

For detailed information and use cases, see Trusted Certificate Issuer.

4.10.8 Trusted Attestation Controller

The Trusted Attestation Controller is a Kubernetes controller for reconciling the Quote Attestation requests initiated by the Trusted Certificate Service (TCS). It mediates between the TCS and the key servers that support attestation services. The key servers can plug into the controller by implementing the API provided by the controller.

For detailed information and use cases, see Trusted Attestation Controller.

4.10.9 sigstore

sigstore combines several different technologies that focus on automatic key management and transparency logs. The technologies can be used independently or as one single process, and together they can create a safer chain of custody tracing software back to the source.

sigstore policy controller is a Kubernetes-based admission hook to enforce image integrity and security check before loading a container from the registry (Figure 18). Two kinds of policies are supported: Key based and keyless. sigstore policy controller enforcement of image verification is used in the vRAN security use case detailed in the Network and Edge Reference System Architectures – 5G vRAN Security Quick Start Guide.

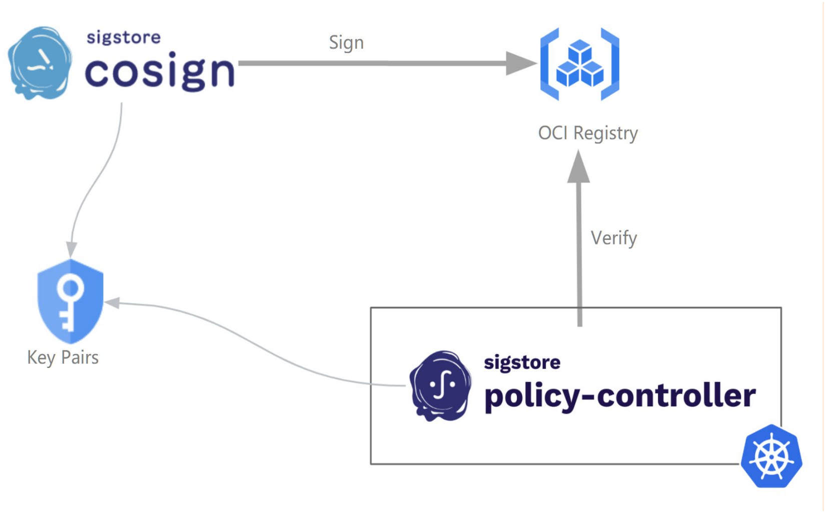


Figure 18. sigstore Policy-Controller Enforcing Image Verification

4.10.10 Secure NETCONF Communications

NETCONF is an IETF-defined protocol for network management. It is extensively used in vRAN systems for configuration of network elements. NETCONF uses SSH or TLS as a transport layer, with TLS the desired protocol for vRAN. The Network and Edge Reference System Architectures – 5G vRAN Security Quick Start Guide details how the TLS keys for NETCONF client/server communication (either RSA or ECDSA) can be managed by Intel's KMRA using Intel SGX enclaves. See Section 4.10.5 for more information on KMRA.

4.11 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, the Reference System Architectures capture DPDK and OVS metrics.

Metrics can be published to Prometheus and other interfaces such as SNMP 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. It has a modular architecture, and data acquisition depends on loaded plugins. Plugins that are loaded with collectd depend on an installed profile. For detailed descriptions of the plugins and metrics available, visit collectd and Barometer.
- **Telegraf** – Cloud native server telemetry collection agent that collects a wide array of platform and application telemetry, including RDT, PMU, and DPDK metrics. Telegraf is an open-source plugin-based server agent that you can use to collect custom application metrics, logs, network performance data, platform metrics, and more. Developed by InfluxData, Telegraf's pluggable architecture uses input plugins for collecting metrics and output plugins for sending metrics to various endpoints, such as time series databases like Prometheus and InfluxDB. For more information on Telegraf and its available plugins, visit Telegraf.
- **OpenTelemetry** – OpenTelemetry provides a standardized method of collecting and transforming metrics across various platforms and languages. It can be seen as either an alternative or complementary to Prometheus. For a detailed overview, visit the Observability Primer. For a comparison between Prometheus and OpenTelemetry, visit Timescale's comparison.
- **Jaeger** – Jaeger is a distributed tracing option that allows developers and users to pinpoint bottlenecks in their microservice applications. For a more detailed overview, visit the Jaeger blog post. The version of Jaeger integrated into the Reference Architecture supports OpenTelemetry trace data.
- **Node Exporter** – Platform telemetry monitoring agent; 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.
- **Prometheus** – Prometheus is an open-source telemetry systems monitoring server that scrapes and stores time series data.
- **Grafana** – Grafana is an open-source telemetry visualization tool that is available under the HTTPS endpoint https://localhost:30000 on any of the cluster nodes.
- **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.
- **Intel® XPU Manager (Intel® XPUM)** – Intel® XPUM is an in-band node-level tool that provides local/remote GPU management. It is easily integrated into the cluster management solutions and cluster scheduler.
- **Telemetry Aware Scheduler** – Makes telemetry data available to scheduling and descheduling decisions in Kubernetes.

- **Intel® Platform Telemetry Insights** – Intel® Platform Telemetry Insights software is available under NDA at the following link: Intel Platform Telemetry Insights. Contact your Intel representative for information.
- **Intel® In-Band Manageability Framework (INBM)** - Software that enables an administrator to perform critical device management operations over-the-air remotely from the cloud.

4.11.1 Intel Platform Telemetry Insights

Intel Platform Telemetry Insights uses fine grained telemetry and provides meaningful information and actionable data across single or multiple servers. These insights distil IA metrics into networking and operational metrics covering platform health, resource utilization, and congestion, and can be used in automated control and orchestration systems to trigger remediation actions.

- **Telemetry Insights Provider:** Provides insights into platform health, resource utilization and congestion, and platform configuration checks.
- **Streaming Telemetry Insights Provider:** Built as a Kafka Streams application, this is where telemetry is collected from the platform and insights are calculated (similar to Telemetry Insights Provider) and made available to the end user in one continuous stream. The insights are published back to the Kafka cluster.
- **Power Insights Kit:** The Power Insights Kit is a stand-alone deliverable that includes tools to collect, manage, and visualize your power usage at a node and cluster level. Intel Platform Telemetry Insights supports the installation of the Power Insights Kit on top of BMRA.

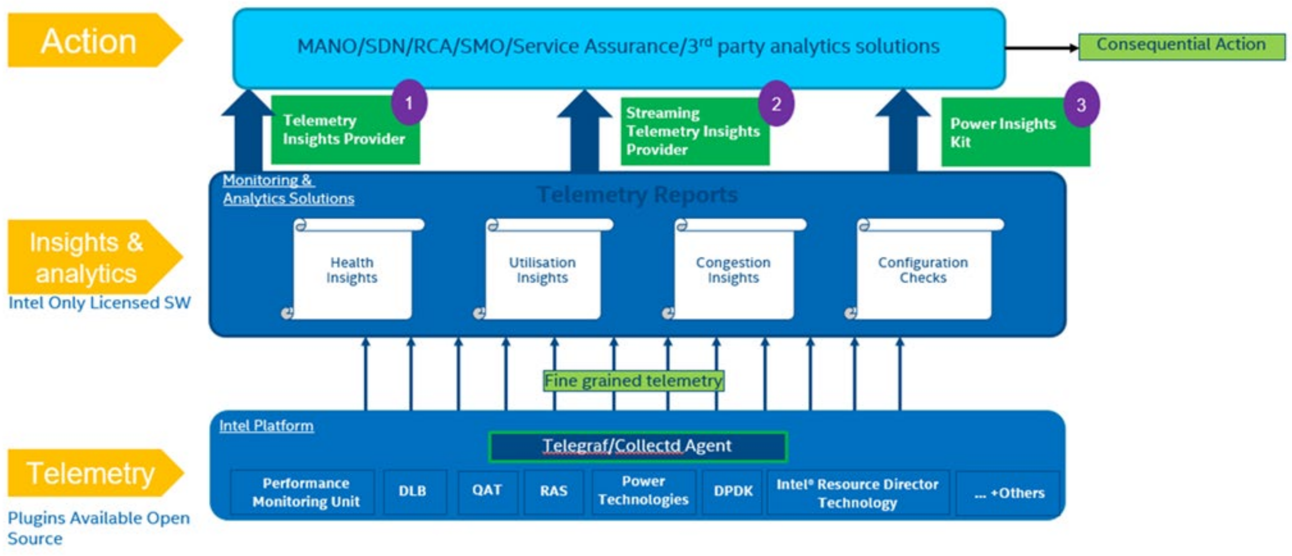


Figure 19. Intel Platform Telemetry Insights High-Level Architecture

4.11.2 Intel® In-Band Manageability Framework (INBM)

The Intel® In-Band Manageability Framework is software that enables an administrator to perform critical device management operations over-the-air remotely from the cloud. It also facilitates the publishing of telemetry and critical events and logs from an IoT device to the cloud, enabling the administrator to take corrective actions if and when necessary. The framework is designed to be modular and flexible ensuring scalability of the solution across preferred cloud service providers (for example, Azure* IoT Central, ThingsBoard.io).

Key advantages of the Intel® In-Band Manageability solution include:

- Out-of-box cloud support: Azure* IoT Central and THINGSBOARD*.
- Single interface to handle OS, firmware, and application (Docker container) updates.

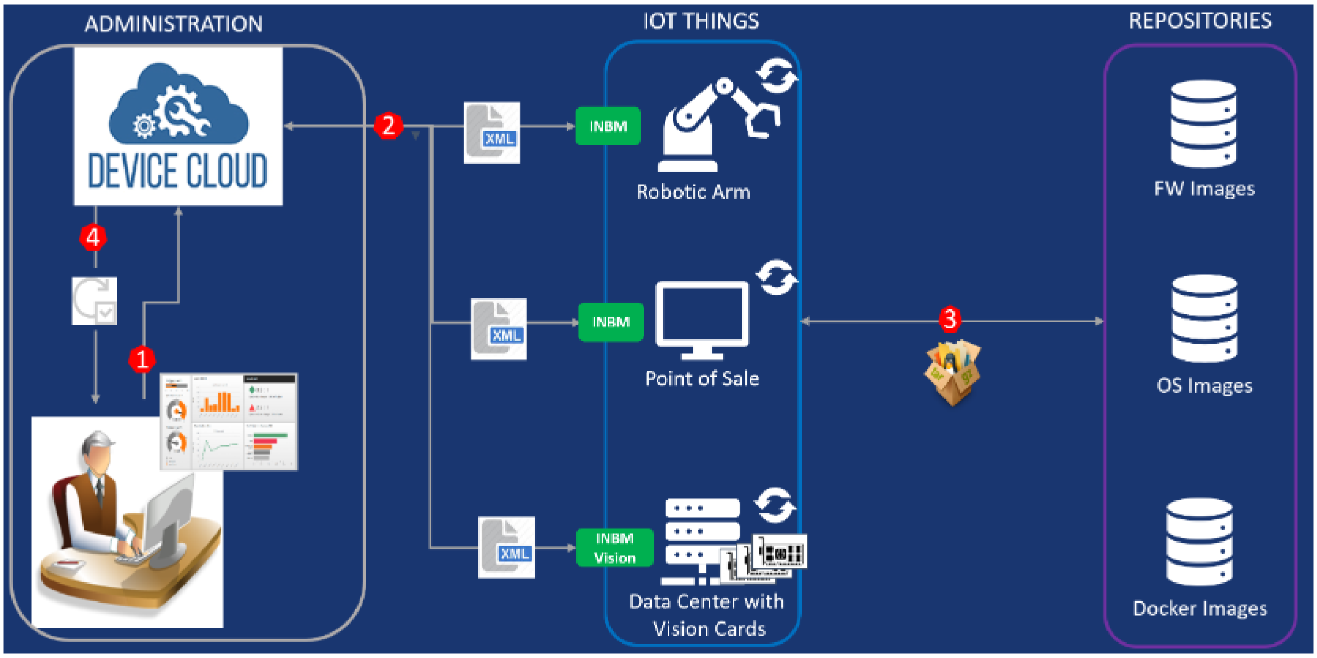


Figure 20. INBM OTA Use Cases and Flow

The flow is as follows:

1. An administrator sends a request via the cloud.
2. A manifest is created based on the cloud request and sent to INBM.
3. Any required packages are downloaded from the remote repository as specified in the manifest.
4. INBM performs the update and sends the result back to the administrator.

4.12 Storage

Data can be broken into three generic types of storage: block, file, and object.

4.12.1 MinIO

The BMRA uses MinIO for object storage. MinIO offers high-performance, S3-compatible object storage and is native to Kubernetes. MinIO has no requirement for telco-specific infrastructure such as DPDK. Hence, MinIO's only architectural requirement is the basic operating system (OS) support for it to service clients and use connected storage devices, such as HDDs and SSDs.

MinIO is a high-throughput, distributed-object storage server, designed for large-scale cloud infrastructure. MinIO's primary targeted uses cases include:

- High throughput media streaming
- Machine learning and analytics
- Apache* Hadoop*/Big Data storage disaggregation

MinIO delivers the following features:

- Amazon Simple Storage Service (Amazon S3) compatible
- Erasure codes and bit rot protection
- AWS Lambda Compute – event driven notification
- Encryption and tamper proofing
- Pluggable storage backbends
- Distributed

MinIO does not use a proxy layer. All MinIO nodes run the MinIO S3 endpoint. A load balancer uses DNS round robin to distribute client connections across all nodes, servicing those client's S3 API requests.

For a more detailed description of MinIO, see Multi-Cloud Object Storage.

4.12.2 Local Persistent Volume Static Provisioner

Local persistent volumes allow users to access local storage through the standard PVC interface in a simple and portable way. The PersistentVolume (PV) contains node affinity information that the system uses to schedule pods to the correct nodes.

The local volume static provisioner manages PersistentVolume lifecycle for pre-allocated disks by detecting and creating PVs for each local disk on the host and cleaning up the disks when released. It does not support dynamic provisioning.

4.12.3 Rook/Ceph

Ceph is a scalable distributed storage system that provides a unified storage service with object, block, and file interfaces from a single cluster. Rook turns distributed storage systems into self-managing, self-scaling, self-healing storage services and automates the tasks of a storage administrator: deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management.

4.13 Traffic Analytics Development Kit (TADK)

Traffic Analytics Development Kit (TADK) is a set of tools and libraries that provide a starting point for developing network applications. These applications can use acceleration instructions, targeting AI inferencing performance, available in Intel® Xeon® Scalable processors.

Included with the Reference Architecture is a workload example of a web application firewall that uses a trained machine-learning (ML) model and the acceleration instructions to predict if an attack on the network is occurring. This example is based on the open source ModSecurity Web Application Firewall (WAF), but with changes to the detection libraries from using traditional logic to ML and AI-based prediction.

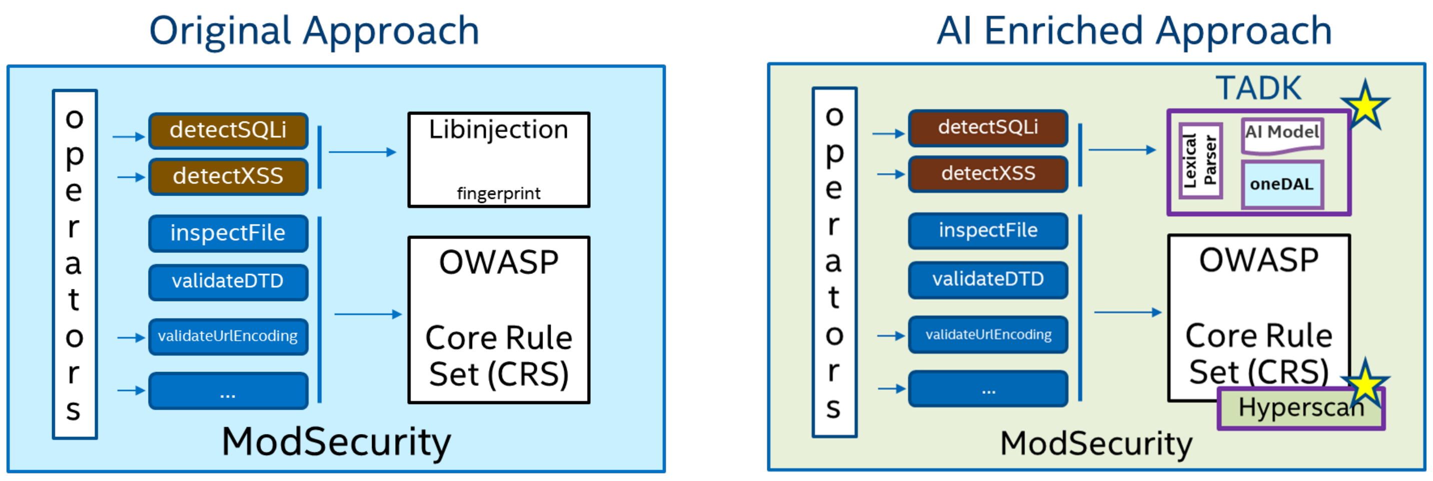


Figure 21. TADK Enhanced Implementation of ModSecurity WAF

TADK is part of a broader suite of AI Technologies that Intel has provided for existing AI frameworks and libraries. More information about these technologies can be found on Intel® Network Builders.

4.14 Access Edge – vRAN

The next generation of 5G network architectures is rapidly evolving to process high-bandwidth real-time data. This means that the core telecom network is transforming from a fixed-function, hardwired appliance architecture to a software-based, open hybrid cloud platform.

The virtualization of the radio access network (vRAN) is an essential step into the 5G future because it simplifies the deployment of new features and algorithms. vRAN also separates network functions from the underlying hardware, making it possible for a flexible and dynamic RAN environment.

4.14.1 Generic vRAN

Using Kubernetes and cloud native capabilities in an Ubuntu environment, a Distributed Unit (DU) can be configured and deployed easily using the 4th Gen Intel Xeon Scalable processor with Intel® vRAN Boost, making use of the latest hardware and software features to accelerate encoding/decoding of physical layer frames and helping to ensure low latency and power consumption.

4.14.2 FlexRAN™ Software Deployment

Intel has created a vRAN reference implementation for virtualized cloud-enabled radio access networks that is named FlexRAN™ software.

Reference System Architectures enable FlexRAN™ software to be configured and deployed easily. Furthermore, power management and the FEC accelerator operators allow for high-resolution configurability and management, helping to result in high throughput and low latency while minimizing power consumption.

Deployment of the "Timer mode" test mode and "XRAN test mode" is enabled to provide an experience that showcases FlexRAN™ software's stand-alone functionality and benchmarking of the FlexRAN™ software and the underlying platform combination. The FlexRAN™ software can be deployed in timer and xRAN mode as containers in POD for 3rd Gen and 4th Gen Intel Xeon Scalable processor server or in bare metal for both 3rd and 4th Gen Intel Xeon Scalable processors. For more information on FlexRAN™ software deployment, refer the Network and Edge Reference System Architectures - vRAN Setup with FlexRAN™ Software Quick Start Guide.

4.15 Libraries

4.15.1 FFmpeg*

FFmpeg is a collection of libraries and tools to process multimedia content such as audio, video, subtitles, and related metadata. The libraries and tools available are as follows:

Libraries:

- librarieslibavcodec provides implementation of a wider range of codecs
- libavformat implements streaming protocols, container formats and basic I/O access
- libavutil includes hashers, decompressors, and miscellaneous utility functions
- libavfilter provides means to alter decoded audio and video through a directed graph of connected filters
- libavdevice provides an abstraction to access capture and playback devices
- libswresample implements audio mixing and resampling routines
- libswscale implements color conversion and scaling routines

Tools:

- FFmpeg is a command line toolbox to manipulate, convert, and stream multimedia content.
- fplay is a minimalistic multimedia player.
- ffprobe is a simple analysis tool to inspect multimedia content.

FFmpeg with Intel-optimized patches is available on GitHub. In addition to CPU-based acceleration for video encode, decode, and post processing, it also supports Intel GPU-based acceleration via VAAPI and Intel® oneAPI Video Processing Library (oneVPL) plugins. A mechanism to automatically build FFmpeg with your own patch set is also provided.

For more information about FFmpeg, see FFmpeg.

4.15.2 oneAPI

oneAPI is an open, cross-industry, standards-based, unified, multiarchitecture, multi-vendor programming model that delivers a common developer experience across accelerator architectures – for faster application performance, more productivity, and greater innovation.

oneAPI toolkits are available to build, analyze, and optimize high-performance, cross-architecture applications on CPUs and XPUs with best-in-class compilers, performance libraries, frameworks, and analysis and debug tools. Two toolkits are available: Intel® oneAPI Base Toolkit (Base Kit) and Intel® AI Analytics Toolkit.

The Base Kit is a core set of tools and libraries for developing high-performance, data-centric applications across diverse architectures. It features an industry-leading C++ compiler that implements SYCL*, an evolution of C++ for heterogeneous computing.

Domain-specific libraries and the Intel® Distribution for Python* provide drop-in acceleration across relevant architectures. Enhanced profiling, design assistance, and debug tools complete the kit.

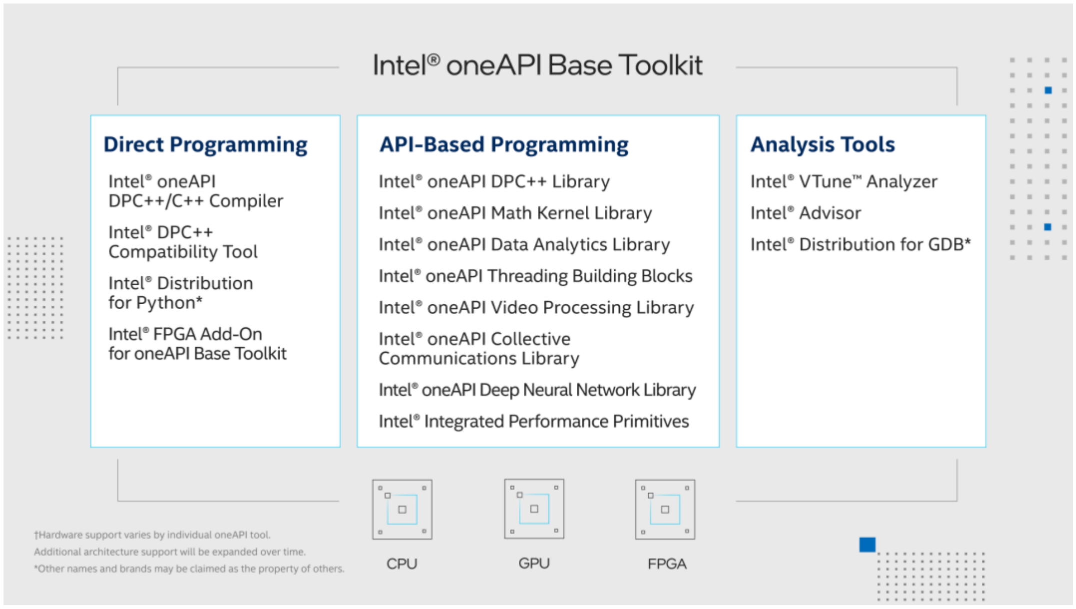

Intel® oneAPI Base Toolkit
Direct Programming
Intel® oneAPI DPC++/C++ Compiler
Intel® DPC++ Compatibility Tool
Intel® Distribution for Python*
Intel® FPGA Add-On for oneAPI Base Toolkit
API-Based Programming
Intel® oneAPI DPC++ Library
Intel® oneAPI Math Kernel Library
Intel® oneAPI Data Analytics Library
Intel® oneAPI Threading Building Blocks
Intel® oneAPI Video Processing Library
Intel® oneAPI Collective Communications Library
Intel® oneAPI Deep Neural Network Library
Intel® Integrated Performance Primitives
Analysis Tools
Intel® VTune™ Analyzer
Intel® Advisor
Intel® Distribution for GDB*
CPU
GPU
FPGA
†Hardware support varies by individual oneAPI tool.
Additional architecture support will be expanded over time.
*Other names and brands may be claimed as the property of others.

Figure 22. Intel® oneAPI Base Toolkit

The Intel® AI Analytics Toolkit targeted for data scientists and AI developers helps with accelerating end-to-end data science and machine learning pipelines using Python* tools and frameworks.

For more information about oneAPI, see oneAPI.

4.15.3 Edge AI Box

The Edge AI Box Dockerfiles and associated sample pod has the necessary libraries made available along with support for GPU.

The libraries included are as follows:

Table 6. Edge AI Box Libraries

Library Names	Repo URL
DLStreamer	https://github.com/dlstreamer/dlstreamer.git
OpenVINO™	https://www.intel.com/content/www/us/en/developer/articles/release-notes/openvino-lts/2022-3.html
OpenCV	https://github.com/opencv/opencv
Intel® XPU Manager	https://hub.docker.com/r/intel/xpumanager
Ffmpeg - cartwheel 2023q2	https://github.com/intel/cartwheel-ffmpeg
OpenCL	https://dgpu-docs.intel.com/releases/stable_555_20230124.htm
Level Zero GPU	
DPC++	
VA-API	

Refer to the Network and Edge Reference System Architectures - On Premises Edge AI Box Quick Start Guide for details on Intel® Edge AI Box workload.

4.15.4 Media Analytics Libraries

The media analytics-related Dockerfiles and associated sample pod has the necessary libraries made available along with support for GPU. This sample pod can be leveraged to run any workload requiring AI-based analysis performed on video streams. The libraries included are as follows:

Table 7. Media Analytics Libraries

Library Names	Repo URL
DLStreamer	https://github.com/dlstreamer/dlstreamer.git
OpenVINO™	https://www.intel.com/content/www/us/en/developer/articles/release-notes/openvino-lts/2022-3.html

Library Names	Repo URL
OpenCL	https://dgpu-docs.intel.com/releases/stable_555_20230124.htm
Level Zero GPU	
DPC++	
VA-API	

Refer to the Network and Edge Reference System Architectures - Edge Analytics Video Structuring Server (VSS) Quick Start Guide for details on VSS workload.

4.15.5 Video Production Libraries

Video production requires the following additional libraries:

- The Intel® Media Transport Library is a software-based solution designed for high-throughput, low-latency transmission and reception of media data.
- The Intel® oneAPI Video Processing Library (oneVPL) supports AI visual inference, media delivery, cloud gaming, and virtual desktop infrastructure use cases by providing access to hardware accelerated video decode, encode, and frame processing capabilities on Intel® GPUs.

Refer to the Network and Edge Reference System Architectures - Video Production Quick Start Guide for details on Video Production workload.

5 Provisioning a Reference Architecture

5.1 Introduction

This section provides an overview of how a Reference Architecture Flavor is generated and deployed. The process includes selection of a deployment model (e.g., BMRA, VMRA) and Configuration Profile, followed by installation and setup of hardware, and then system provisioning using Ansible playbooks. Intel provides a set of Ansible Playbooks to install and configure the Reference System Architectures.[11]

Note: Ansible playbooks for 4th and 5th Gen Intel Xeon Scalable processors and the Intel® Xeon® D processor are open source and available through GitHub. The VMRA does not currently support deployments using the Intel Xeon D processor.

The following information describes the general installation process. For a complete guide on deployment, including the required hardware, BIOS configurations, network topology requirements, and more, refer to the appropriate user guide (BMRA or VMRA).

5.2 Installation Overview

Configuration of Reference System Architecture deployments is based on the following:

- Deployment Model: Bare metal cluster (BMRA) or virtual cluster (VMRA).
- The Configuration Profile (a specific network location or generic): Remote Central Office-Forwarding, Regional Data Center, On-Premises Edge, On-Premises VSS, On-Premises AI Box, On-Premises SW Defined Factory Configuration Profile – Industrial deployment, Access Edge, or a generic Basic or Build-Your-Own configuration. The choice impacts the hardware required and the Ansible scripts that provision the hardware and software specified.

The Reference System Architectures were designed with Kubernetes in mind for flexibility and simplicity. However, installing the Kubernetes distro is not mandatory. The Reference Systems and the installations were validated with Kubernetes on both bare metal and virtualized clusters.

These choices help drive configuration and setup for the following prerequisites needed for the Ansible host to run the scripts and for other nodes. Details for the prerequisites are listed in the appropriate user guide:

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

[11] Refer to Optimization Notices for more information regarding performance and optimization choices in Intel software products.

After you complete these prerequisites, you can download Ansible playbooks for 4th and 5th Gen Intel Xeon Scalable processors and Intel Xeon D processors directly from the dedicated GitHub releases page or cloned using Git.

5.2.1 Operating System Selection

The Reference System Architectures Portfolio was validated for the following operating systems for Kubernetes control and worker nodes, VM host, and VMs and their containers:

- RHEL9.2
- Rocky Linux 9.1
- Ubuntu 22.04
- [BMRA only] Ubuntu 22.04 RT
- [BMRA only] RHEL9.2 RT

5.2.2 Network Interface Requirements

The following network requirements must be met before deployment:

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

5.2.3 Software Prerequisites for Ansible Host, Control Nodes, and Worker Nodes

Before deployment of the Ansible playbooks, the Ansible host must be prepared by logging into it using SSH or your preferred method to access the shell. Install packages on the Ansible host according to the appropriate user guide.

5.3 Ansible Playbook Review

The Reference System Architecture is configured and provisioned automatically using Ansible scripts. Each Configuration Profile has a dedicated Ansible playbook. This section describes how the Ansible playbooks allow for an automated deployment of a fully functional Reference System Architecture, including initial system configuration, Kubernetes or VMs deployment, and set up of capabilities.

Ansible Playbooks act as a user entry point and include all relevant Ansible roles and Helm charts. Top-level Ansible playbooks exist for each Configuration Profile to allow use case-oriented cluster deployments (Reference Architecture Flavors). Additionally, a special Cluster Removal playbook exists to simplify removal of a deployment. This playbook is useful for restarting an installation from the beginning or removing one completely.

Ansible Playbooks use configuration files that define cluster-wide and host-specific configuration options for each of the Reference Architecture Flavors. With minimal changes, they can be used directly with their corresponding Ansible playbooks. Default values for the configuration files of each Reference Architecture Flavor are shown in the appropriate user guide.

Each playbook executes multiple phases – Infrastructure Setup, Kubernetes, VM Deployment (VMRA only), and Capabilities Setup – and assigns Ansible Roles during each phase. For a VMRA deployment, there are individual phases for the Host and the VMs. Figure 23 illustrates the phases assigned in each Ansible playbook phase. The phases are described in more detail in Section 5.3.1.

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.

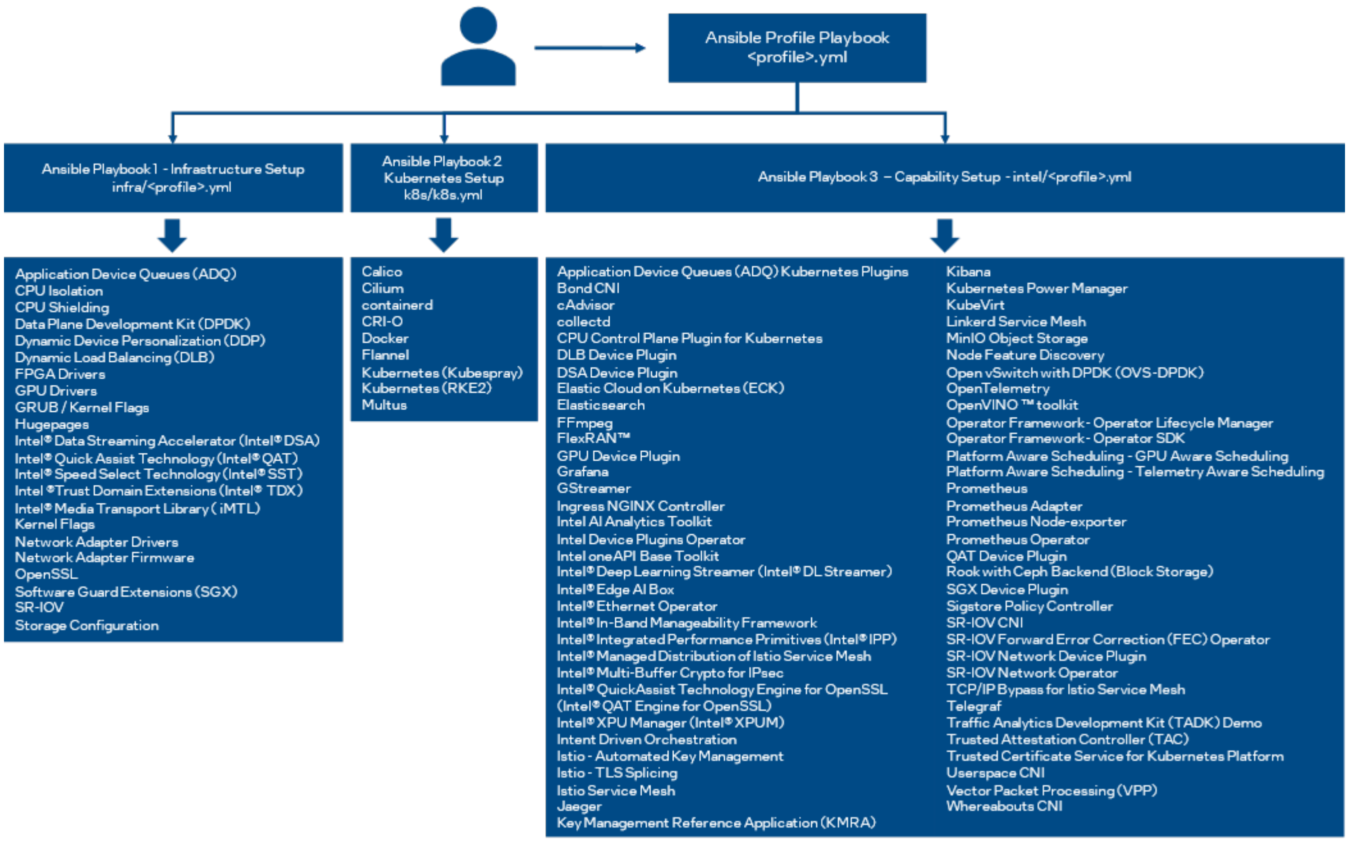

Ansible Profile Playbook
<profile>.yml
Ansible Playbook 1 - Infrastructure Setup
infra/<profile>.yml
Ansible Playbook 2
Kubernetes Setup
k8s/k8s.yml
Ansible Playbook 3 – Capability Setup - intel/<profile>.yml
Application Device Queues (ADQ)
CPU Isolation
CPU Shielding
Data Plane Development Kit (DPDK)
Dynamic Device Personalization (DDP)
Dynamic Load Balancing (DLB)
FPGA Drivers
GPU Drivers
GRUB / Kernel Flags
Hugepages
Intel® Data Streaming Accelerator (Intel® DSA)
Intel® Quick Assist Technology (Intel® QAT)
Intel® Speed Select Technology (Intel® SST)
Intel ®Trust Domain Extensions (Intel® TDX)
Intel® Media Transport Library (iMTL)
Kernel Flags
Network Adapter Drivers
Network Adapter Firmware
OpenSSL
Software Guard Extensions (SGX)
SR-IOV
Storage Configuration
Calico
Cilium
containerd
CRI-O
Docker
Flannel
Kubernetes (Kubespray)
Kubernetes (RKE2)
Multus
Application Device Queues (ADQ) Kubernetes Plugins
Bond CNI
cAdvisor
collectd
CPU Control Plane Plugin for Kubernetes
DLB Device Plugin
DSA Device Plugin
Elastic Cloud on Kubernetes (ECK)
Elasticsearch
FFmpeg
FlexRAN™
GPU Device Plugin
Grafana
GStreamer
Ingress NGINX Controller
Intel AI Analytics Toolkit
Intel Device Plugins Operator
Intel oneAPI Base Toolkit
Intel® Deep Learning Streamer (Intel® DL Streamer)
Intel® Edge AI Box
Intel® Ethernet Operator
Intel® In-Band Manageability Framework
Intel® Integrated Performance Primitives (Intel® IPP)
Intel® Managed Distribution of Istio Service Mesh
Intel® Multi-Buffer Crypto for IPsec
Intel® QuickAssist Technology Engine for OpenSSL
(Intel® QAT Engine for OpenSSL)
Intel® XPU Manager (Intel® XPUM)
Intent Driven Orchestration
Istio - Automated Key Management
Istio - TLS Splicing
Istio Service Mesh
Jaeger
Key Management Reference Application (KMRA)
Kibana
Kubernetes Power Manager
KubeVirt
Linkerd Service Mesh
MinIO Object Storage
Node Feature Discovery
Open vSwitch with DPDK (OVS-DPDK)
OpenTelemetry
OpenVINO ™ toolkit
Operator Framework - Operator Lifecycle Manager
Operator Framework - Operator SDK
Platform Aware Scheduling - GPU Aware Scheduling
Platform Aware Scheduling - Telemetry Aware Scheduling
Prometheus
Prometheus Adapter
Prometheus Node-exporter
Prometheus Operator
QAT Device Plugin
Rook with Ceph Backend (Block Storage)
SGX Device Plugin
Sigstore Policy Controller
SR-IOV CNI
SR-IOV Forward Error Correction (FEC) Operator
SR-IOV Network Device Plugin
SR-IOV Network Operator
TCP/IP Bypass for Istio Service Mesh
Telegraf
Traffic Analytics Development Kit (TADK) Demo
Trusted Attestation Controller (TAC)
Trusted Certificate Service for Kubernetes Platform
Userspace CNI
Vector Packet Processing (VPP)
Whereabouts CNI

Figure 23. Features Included in Each Ansible Playbook Phase

5.3.1 Ansible Playbook Phases

Regardless of the selected Configuration Profile, the installation process always consists of three main phases. For VMRA, individual setups occur for VM Host and VMs as shown in Figure 5.

1. **Infrastructure Setup** (sub-playbooks in playbooks/infra/ directory)

   These playbooks modify kernel boot parameters and apply the initial system configuration for the nodes. Depending on the selected Configuration Profile, this includes:
   - Generic host OS preparation, for example, installation of required packages, Linux kernel configuration, proxy and DNS configuration, and modification of SELinux policies and firewall rules.
   - Configuration of the kernel boot parameters according to the user-provided configuration 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 VFs and binding to appropriate Linux kernel modules.
   - Network adapter drivers and firmware updates, which help ensure that all latest capabilities such as DDP profiles are enabled.
   - Intel® Speed Select Technology (Intel® SST) configuration, which provides control over base frequency.
   - Installation of Dynamic Device Personalization (DDP) profiles, which can increase packet throughput, help reduce latency, and lower CPU usage by offloading packet classification and load balancing to the network adapter.
   - Configuration and provisioning of the virtualization layer when deploying the VMRA. Separate infrastructure setup procedures are executed for Host and VMs.

2. **Kubernetes Setup** (in playbooks/k8s/ directory)

   Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. The Reference System supports two Kubernetes distributions and deployment modes. The first is the CNCF open-source Kubernetes deployed using Kubespray. The second is Rancher RKE2 Kubernetes distribution.

   When a CNCF open-source Kubernetes cluster is deployed, this playbook deploys a high availability (HA) Kubernetes cluster using Kubespray. Kubespray is a project under the Kubernetes community that deploys production-ready Kubernetes clusters. The Multus CNI plugin, which is specifically designed to support multiple networking interfaces in a Kubernetes environment, is deployed by Kubespray along with Calico and Helm. Preferred security practices are used in the default configuration. On top of Kubespray, there is also a container registry instance deployed to store images of various control-plane Kubernetes applications, such as TAS or device plugins.

Rancher next generation distribution RKE2 is supported. RKE2 is the next iteration of the Rancher Kubernetes Engine (RKE), known as RKE1. RKE2 is a fully conformant Kubernetes distribution that focuses on security and compliance within the U.S. Federal Government sector. For additional information on RKE, see Rancher Kubernetes Engine (RKE) and Launching Kubernetes with Rancher.

3. **System Capabilities Setup** (sub-playbooks in playbooks/intel directory)

   Advanced networking technologies, enhanced platform awareness, and device plugin features are deployed by this playbook using operators or Helm charts. For a VMRA deployment, system capabilities are installed on the VMs, not on the host. The following capabilities are deployed:
   - Device plugins that allow using, for example, SR-IOV, QAT, and GPU devices in workloads running on cluster.
   - 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.
   - Node Feature Discovery (NFD), which is a Kubernetes add-on to detect and advertise hardware and software capabilities of a platform that can, in turn, be used to facilitate intelligent scheduling of a workload.
   - Telemetry Aware Scheduling, which allows scheduling workloads based on telemetry data.
   - Full Telemetry Stack consisting of collectd, Kube-Prometheus, and Grafana, which give cluster and workload monitoring capabilities and act as a source of metrics that can be used in TAS to orchestrate scheduling decisions.

5.4 Deployment Using Ansible Playbooks

The following steps are required to obtain the Ansible playbook source code, prepare target servers, configure inventory and variable files, and deploy the Kubernetes cluster.

5.4.1 Prepare Target Servers

For each target server that will act as a Kubernetes control or worker node, make sure that it meets the following requirements:
- Python 3 is installed. The version depends on the target distribution.
- SSH is configured and the server is reachable from the Ansible host.
- Internet access on all target servers is mandatory.
- BIOS configuration matching the desired state is applied.

For detailed steps on how to build the Ansible host, refer to the appropriate user guide.

5.4.2 Get Ansible Playbook and Prepare Configuration Templates

On the Ansible host, select the Configuration Profile for deployment and export the environmental variable. For example, for Kubernetes **Basic** Configuration Profile deployment:

```
export PROFILE=basic
```

Install requirements needed by render.py script:

```
pip3 install -r requirements.txt
```

Generate example profiles for either BMRA or VMRA:

```
[BMRA] make k8s-profile PROFILE=$PROFILE ARCH=<spr,icx,clx> NIC=<cvl,fvl>
[VMRA] make vm-profile PROFILE=$PROFILE ARCH=<spr,icx,clx> NIC=<cvl,fvl>
```

5.4.3 Update Ansible Inventory File

Edit the *inventory.ini* file generated in the previous steps.
1. For a BMRA, specify all target servers by using actual hostnames and management IP addresses and update the `ansible_password` to match the SSH configuration of the target servers.
2. For a VMRA, specify the target VM host server by using actual hostname and management IP address and update the `ansible_user` and `ansible_password` to match the SSH configuration of the target server. In the `[vm_host]` section, update the hostname to match that defined in `[all]`. Do not uncomment any of the hostnames defined under `[kube_control_plane]`, `[etcd]`, and `[kube_node]`, as these are dynamically updated based on the number of virtual machines defined for the target VM host server in `host_vars`.

5.4.4 Update Ansible Host and Group Variables

Each Configuration Profile uses its own set of variables. Refer to the specific Configuration Profile section in the appropriate user guide. The following outlines the procedure:
1. Create *host_vars/<hostname>.yml* files for all worker nodes, matching their hostnames from the inventory file.
2. Edit *host_vars/<hostname>.yml* and *group_vars/all.yml* files to match your desired configuration.

5.4.5 Run Ansible Cluster Deployment Playbook

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

It is recommended to first install a set of OS-specific patches:

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

After completing the patches, the deployment can be started:

```
[BMRA] ansible-playbook -i inventory.ini playbooks/${PROFILE}.yml
[VMRA] ansible-playbook -i inventory.ini playbooks/vm.yml
```

Depending on the selected Configuration Profile, network bandwidth, storage speed, and other similar factors, the execution might take up to 30-40 minutes.

After the playbook finishes without any “Failed” tasks, you can proceed with the deployment validation described in the user guides.

5.4.6 Ansible Cluster Removal Playbook

If you want to clean up the environment to run a new deployment or remove a completed, failed, or incomplete deployment, run the provided Cluster Removal playbook (*redeploy_cleanup.yml*). It removes any previously installed Kubernetes and related plugins.

Appendix A Terminology

The following terminology is used in this document.

Table 8. Terminology

Abbreviation	Description
5GC	5G Core
ADQ	Application Device Queries (ADQ)
AEAD	Authentication Encryption with Associated Data
AF_XDP	Address Family eXpress Data Path
AKS	Azure Kubernetes Service
AMX	Advance Matrix Multiply
API	Application Programming Interface
AV1	AOMedia Video 1 video coding format
AVC	Advanced Video Coding video compression standard
AWS	Amazon Web Services
BIOS	Basic Input/Output System
BKC	Best Known Configuration
BMRA	Bare Metal Reference Architecture
BOM	Bill of Materials
CA	Certificate Authority
CDN	Content Delivery Network
Cloud RA	Cloud Reference Architecture
CMTS	Cable Modem Termination System
CNCF	Cloud Native Computing Foundation
CNF	Cloud Native Network Function
CNI	Container Network Interface
CO	Central Office
CoSP	Communications Service Provider
CPI	Cycles Per Instruction
CPU	Central Processing Unit
CR	Custom Resource
CRD	Custom Resource Definition
CRI	Container Runtime Interface
CRS	Core Rule Set
CRUD	Create, Read, Update, and Delete
CSP	Cloud Service Provider
CSR	Certificate Signing Request
CU	Central Unit
CXL	Compute Express Link
DCAP	Datacenter Attestation Primitives
DDP	Dynamic Device Personalization (DDP)
DHCP	Dynamic Host Configuration Protocol
DLB	Intel® Dynamic Load Balancer (Intel® DLB)
DNS	Domain Name Service
DoS	Denial of Service
DP	Device Plugin
DPDK	Data Plane Development Kit (see DPDK)

Abbreviation	Description
DRAM	Dynamic Random Access Memory
DSA	Intel® Data Streaming Accelerator (Intel® DSA)
DU	Distributed/Distribution Unit
ECDSA	Elliptic curve digital signature algorithm
EKS	Elastic Kubernetes Service
EPC	Electronic Product Code
FEC	Forward Error Correction
FPGA	Field-Programmable Gate Array
FW	Firmware
GPU	Graphics Processor Unit
HA	High Availability
HDD	Hard Disk Drive
HEVC	High Efficiency Video Coding video compression
HQM	Hardware Queue Manager
HSM	Hardware Security Model
HT	Hyper-Threading
IA	Intel® architecture
IAX	In-Memory Analytics
Intel® AVX	Intel® Advanced Vector Extensions (Intel® AVX)
Intel® AVX-512	Intel® Advanced Vector Extension 512 (Intel® AVX-512)
Intel® DLB	Intel® Dynamic Load Balancer (Intel® DLB)
Intel® DSA	Intel® Data Streaming Accelerator (Intel® DSA)
Intel® HT Technology	Intel® Hyper-Threading Technology (Intel® HT Technology)
Intel® QAT	Intel® QuickAssist Technology (Intel® QAT)
Intel® RDT	Intel® Resource Director Technology (Intel® RDT)
Intel® Scalable IOV	Intel® Scalable I/O Virtualization (Intel® Scalable IOV)
Intel® SecL – DC	Intel® Security Libraries for Data Center (Intel® SecL – DC)
Intel® SGX	Intel® Software Guard Extensions (Intel® SGX)
Intel® SST	Intel® Speed Select Technology (Intel® SST)
Intel® SST-BF	Intel® Speed Select Technology – Base Frequency (Intel® SST-BF)
Intel® SST-CP	Intel® Speed Select Technology – Core Power (Intel® SST-CP)
Intel® SST-PP	Intel® Speed Select Technology – Performance Profile (Intel® SST-PP)
Intel® SST-TF	Intel® Speed Select Technology – Turbo Frequency (Intel® SST-TF)
Intel® TDX	Intel® Trust Domain Extension (Intel® TDX)
Intel® VT-d	Intel® Virtualization Technology (Intel® VT) for Directed I/O (Intel® VT-d)
IOMMU	Input/Output Memory Management Unit
IoT	Internet of Things
IPC	Instructions per Cycle
ISA	Instruction Set Architecture
I/O	Input/Output
K8s	Kubernetes
KMRA	Key Management Reference Application (KMRA)
LLC	Last Level Cache
LOM	LAN on Motherboard
LPVSP	Local Persistent Volume Static Provisioner
LTS	Long Term Support

Abbreviation	Description
MANO	Management and Orchestration
MBM	Memory Bandwidth Monitoring (MBM)
MEC	Multi-Access Edge Control
ML	Machine Learning
MMIO	Memory-Mapped Input/Output
MPEG-2	Moving Picture Experts Group standard for digital television and DVD video
MSR	Model-Specific Register
mTLS	Mutual Transport Layer Security
NIC	Network Interface Card
NF	Network Function
NFD	Node Feature Discovery
NFV	Network Functions Virtualization
NPX	Network Platform Xeon
NTB	Non-Transparent Bridge
NUMA	Non-Uniform Memory Access
NVDIMM	Non-Volatile DIMM
NVM/NVMe	Non-Volatile Memory
OAM	Operation, Administration, and Management
OCI	Open Container Initiative
OS	Operating System
OVS	Open vSwitch
OVS-DPDK	Open vSwitch with DPDK
OWASP	Open Web Application Security Project
PGW	Packet Gateway
PAS	Platform Aware Scheduling
PCI	Physical Network Interface
PCIe	Peripheral Component Interconnect Express
PF0	First physical function of the device
PKCS	Public-Key Cryptography Standard
PMD	Poll Mode Driver
PMU	Power Management Unit
POP	Post Office Protocol
PSP	Pod Security Policy
PXE	Preboot Execution Environment
QAT	Intel® QuickAssist Technology
QCT	Quanta Cloud Technology
QoS	Quality of Service
RA	Reference Architecture
RAN	Radio Access Network
RAS	Reliability, Availability, and Serviceability
RBAC	Role-Based Access Control
RDT	Intel® Resource Director Technology (Intel® RDT)
RSA	Rivest–Shamir–Adleman (public-key cryptosystem)
S3	Amazon Web Services Simple Storage Service
S-IOV	Intel® Scalable I/O Virtualization (Intel® Scalable IOV)
SATA	Serial Advanced Technology Attachment

Abbreviation	Description
SDN	Software-Defined Networking
SGX	Intel® Software Guard Extensions (Intel® SGX)
SIMD	Single Instruction, Multiple Data
SMF	Session Management Function
SNMP	Simple Network Management Protocol
SR-IOV	Single Root Input/Output Virtualization
SSD	Solid State Drive
SSH	Secure Shell Protocol
SSL	Secure Sockets Layer
SVM	Shared Virtual Memory
TADK	Traffic Analytics Development Kit
TAS	Telemetry Aware Scheduling
TCP	Transmission Control Protocol
TCS	Trusted Certificate Service
TDP	Thermal Design Power
TLS	Transport Layer Security
TMUL	Tile Multiply
TPM	Trusted Platform Module
UDS	UNIX Domain Socket
UEFI	Unified Extensible Firmware Interface
UPF	User Plane Function
UPG	User Profile Gateway
v5GC	Virtual 5G Core
vBNG	Virtual Broadband Network Gateway
vCDN	Virtualized Content Delivery Network
vCMTS	Virtual Cable Modem Termination System
vCPE	Virtual Customer Premises Equipment
VF	Virtual Function
VLAN	Virtual LAN
VM	Virtual Machine
VMM	Virtual Machine Manager
VMRA	Virtual Machine Reference Architecture
VNF	Virtual Network Function
VNFM	Virtual Network Functions Manager
VP9	Video coding format for streaming over the internet
vPCRF	Virtual Policy and Changing Rules Function
VPP	Vector Packet Processing
VXLAN	Virtual Extensible LAN

Appendix B Reference Documentation

Experience Kits collateral provides a collection of best-practice architecture and development guidelines that address industry needs in automation, interfaces standardization, security, resources management, and more. These Experience Kits offer developers, technical leads, and other audiences a variety of materials required to enable adoption of the new technologies and service-enabling capabilities needed for next-generation, open, agile, and efficient networks.

Collaterals, including technical guides and solution briefs that explain in detail the technologies enabled in the Network and Edge Reference System Architectures Portfolio, are available in the following location: Network & Edge Platform Experience Kits.

Table 9. Reference Documents

Reference
Network & Edge Platform Experience Kits
Network and Edge Container Bare Metal Reference System Architecture User Guide
Network and Edge Virtual Machine Reference System Architecture User Guide
Network and Edge Cloud Reference System Architecture User Guide
Network and Edge Reference System Architectures Release Overview Solution Brief
Network and Edge Reference System Architectures - Single Server Quick Start Guide
Network and Edge Reference System Architectures - CDN Quick Start Guide
Network and Edge Reference System Architectures - Edge Analytics Video Structuring Server (VSS) Quick Start Guide
Network and Edge Reference System Architectures - On Premises Edge AI Box Quick Start Guide
Network and Edge Reference System Architectures - Industrial Controller Quick Start Guide
Network and Edge Reference System Architectures - vRAN Setup with FlexRAN™ Software Quick Start Guide
Network and Edge Reference System Architectures - 5G vRAN Security Quick Start Guide
Network and Edge Reference System Architectures - 5G Core UPF Quick Start Guide
Network and Edge Reference System Architectures - Video Production Quick Start Guide

0124/DN/WIT/PDF 721797-0130US