Quick Start Guide

Network and Edge Reference System Architectures - Edge Analytics Video Structuring Server (VSS)

Develop and verify cloud-native services for On-Prem Video Analytics using BMRA on 4th and 5th Gen Intel® Xeon® Scalable processor platform.

Authors

Abhijit Sinha

Zhifang Long

Yanping Wu

Introduction

The Reference System Architectures (Reference System[1]) are a cloud-native, forward-looking Kubernetes*-cluster template solution for network implementations. They provide Ansible* playbooks that define configuration profiles for fast, automatic deployment of needed cluster services and capabilities.

This document is a quick start guide to configure the **Container Bare Metal Reference System Architecture (BMRA)** on **4th or 5th Gen Intel® Xeon® Scalable processor**-based platform with **Intel® Data Center Flex GPU** for enabling **Video Analytics** workloads.

The Reference System is deployed using the **On-Prem VSS Configuration Profile** with **optimized configuration for Video AI Inference** to perform object detection and classification from a recorded video stream. The Video AI inference is enabled by OpenVINO-based libraries, GStreamer/DL-Streamer for Media Analytics. The platform is accelerated by Intel® GPU, Intel® DSA, and Intel® DLB and is secured by Intel® QAT and Intel® SGX as shown in Figure 1.

The Reference System brings the following benefits:

- Reduced deployment time – a dedicated "on_prem_vss" profile best suited for VSS workloads
- Optimized performance – the workload performance is validated and optimized by Intel engineering teams
- Predicted and repeatable results – the workload implemented according to the recipe is running on the reference platform

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

On-Prem VSS Configuration Profile Architecture

Figure 1 shows the architecture of the On-Prem VSS Configuration Profile, which deploys the infrastructure for the Video Analytics workload.

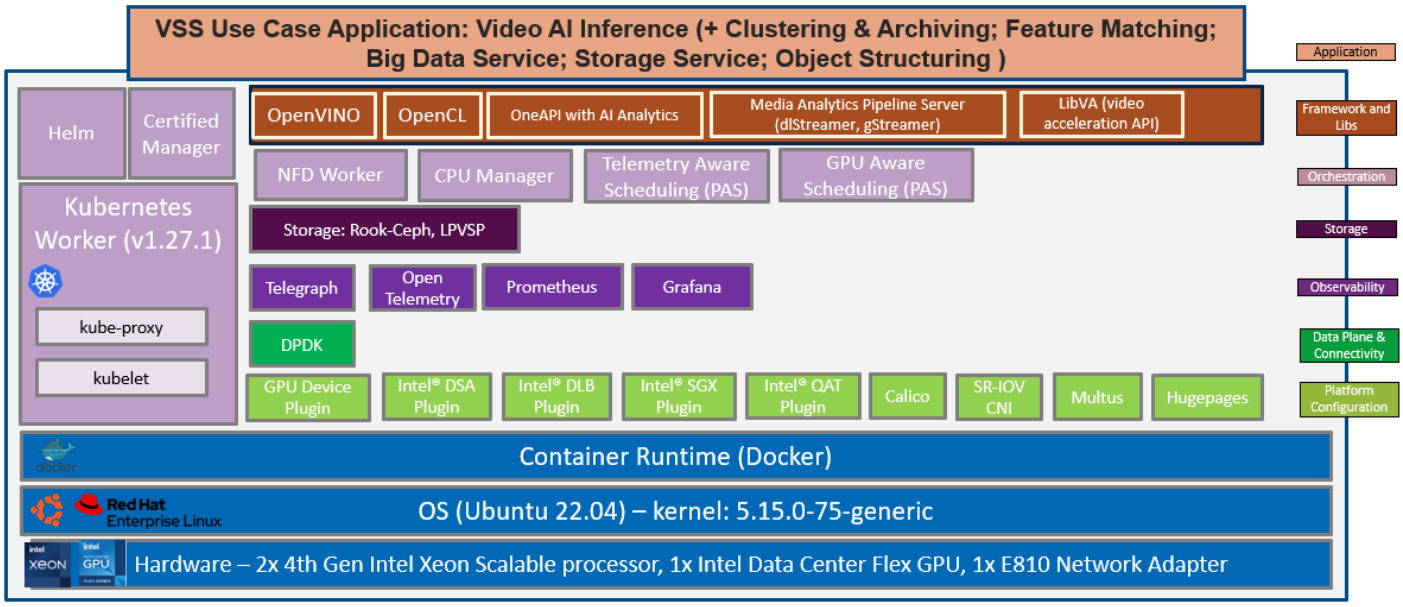

Figure 1: BMRA On-Prem-VSS Configuration Profile Architecture

Hardware BOM

Following is the list of the hardware components that are required for setting up reference systems:

Ansible host	Laptop or server running a UNIX base distribution
Controller Node	Any 3rd, 4th, or 5th Gen Intel® Xeon® Scalable processor-based server
Worker Node	4th Gen Intel® Xeon® Scalable processor-based server OR 5th Gen Intel® Xeon® Scalable processor-based server
GPU	Intel® Data Center GPU Flex 140 or Intel® Data Center GPU Flex 170 (On the worker node)
Ethernet Adapter	Intel® Ethernet Network Adapter E810-CQDA2 or Intel® Ethernet Controller XXV/XL710
Recommended BIOS	"Max Performance Turbo" BIOS configuration (refer to Chapter 3.8 of BMRA User Guide) - SGX needs to be enabled in BIOS

Note: The on_prem_vss profile is expected to work on COTS platform like Dell R660/R760 or HPE DL360/DL380 for VSS workload.

Software BOM

Following is the list of the software components that are required for setting up reference systems:

High Level Media Frameworks	DLStreamer, Gstreamer
Inference Frameworks	OpenVINO
Video Acceleration API	libva
Graphics Compute Runtime	OpenCL

OneAPI	Intel® oneAPI with AI analytics
Security	OpenSSL, SGX, QAT
Storage	rook-ceph, LPVSP
Observability	Telegraf, Open Telemetry, Prometheus, Grafana
Acceleration/ Data Plane	DPDK
Connectivity	istio_service_mesh
Operators & Device plugins	Intel® GPU device plugin, Intel® DSA, DLB, QAT, SGX plugins, Multus
Container Runtime	Docker
Orchestration	Kubernetes v1.26.3 Assisted Scheduling: Telemetry Aware Scheduling (TAS), GPU Aware Scheduling (GAS)
OS	Ubuntu 22.04.2 LTS (kernel 5.15.0-72 generic) and RHEL 9.2

Note: Only Specific Ubuntu and Kernel versions are supported to get the compatible GPU drivers.

For more details on software versions for the **On-Prem VSS Configuration Profile**, refer to Chapter 4 of BMRA User Guides listed in the Reference Documentation section.

Getting Started

Pre-Requisites

Before starting the deployment, perform the following steps:

- A fresh OS installation is expected on the controller and target nodes to avoid a conflict between the RA deployment process with the existing software packages. To deploy RA on the existing OS, ensure that there is no prior Docker or Kubernetes* (K8s) installations on the server(s).
- The controller and target server hostname(s) must be in lowercase, numerals, and hyphen '-'.
    - For example: wrk-8 is acceptable, wrk_8, WRK8, Wrk^8 are not accepted as hostnames.
- The servers in the cluster are Network Time Protocol (NTP) synced, i.e., they must have the same date and time.
- The BIOS on the target server is set as per the recommended settings and SGX is enabled.

Deployment Setup

Figure 2 shows the deployment model for VSS workload using BMRA. The Ansible host is used for configuring and deploying BMRA on a set of target servers.

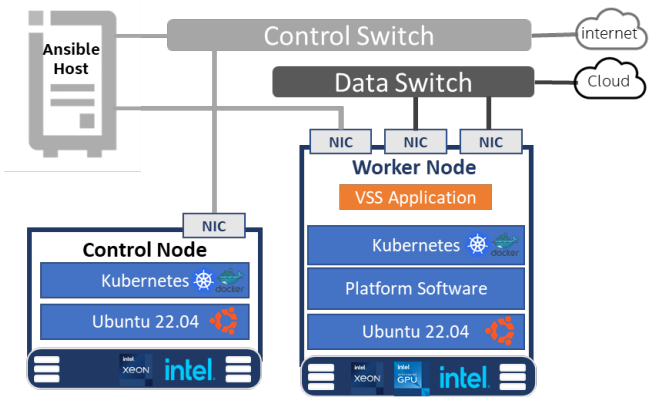

Figure 2: BMRA deployment setup for On Prem VSS

Note: The VSS can also be deployed on a single node K8s cluster (SNO) deployment where the controller and worker nodes are on the same server.

Installation Flow for RA Deployment

Ansible playbooks are used to install the Bare Metal Reference Systems Architecture (BMRA) for the On-Prem VSS profile. Before the playbooks can be run, there are a few steps to prepare the environment and change relevant configuration options.

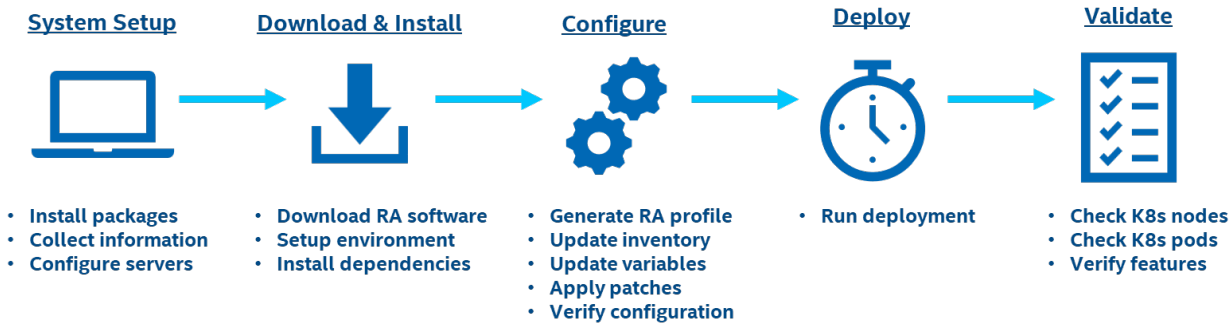

Figure 3: RA Deployment flow using Ansible Playbooks

Step 1 - Set Up the System

The below mentioned steps assume that both the Ansible host and target server are running Ubuntu as the operating system. For RHEL, use 'yum' or 'dnf' as the package manager instead of 'apt'.

Ansible Host

System Setup

1. Install necessary packages (some might already be installed):

```
# sudo apt update
# sudo apt install -y python3 python3-pip openssh-client git build-essential
# pip3 install --upgrade pip
```

2. Generate a SSH keypair if needed (check /root/.ssh/):

```
# ssh-keygen -t rsa -b 4096 -N "" -f ~/.ssh/id_rsa
```

3. Copy the public key to the target servers - controller and worker nodes:

```
# ssh-copy-id root@<target node IP>
```

4. Verify passwordless connectivity to the target servers:

```
# ssh root@<target node IP>
```

Target Server

1. Install the necessary packages (some might already be installed):

```
# sudo apt install -y python3 openssh-server lshw
```

2. As part of the configuration in Step 3, information about PCI devices for SR-IOV must be specified. Find the relevant PCI IDs (bus:device.function) using 'lspci', and note down the IDs for later when configuring host_vars on the Ansible host:

```
# lspci | grep Eth
18:00.0 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 01)
18:00.1 Ethernet controller: Intel Corporation Ethernet Controller E810-C for QSFP (rev 01)
```

3. Verify if the target server's CPU has acceleration devices like DSA, DLB and QAT visible in the OS.

```
DSA Device
# lspci -nnD | grep 0b25
0000:75:01.0 System peripheral [0880]: Intel Corporation Device [8086:0b25]
0000:f2:01.0 System peripheral [0880]: Intel Corporation Device [8086:0b25]

DLB Device
# lspci -nnD | grep 2710
0000:78:00.0 Co-processor [0b40]: Intel Corporation Device [8086:2710]
0000:7c:00.0 Co-processor [0b40]: Intel Corporation Device [8086:2710]
0000:f5:00.0 Co-processor [0b40]: Intel Corporation Device [8086:2710]
0000:f9:00.0 Co-processor [0b40]: Intel Corporation Device [8086:2710]
```

```
QAT Devices
# lspci -nnD | grep 494*
0000:76:00.0 Co-processor [0b40]: Intel Corporation Device [8086:4942] (rev 40)
0000:7a:00.0 Co-processor [0b40]: Intel Corporation Device [8086:4942] (rev 40)
0000:f3:00.0 Co-processor [0b40]: Intel Corporation Device [8086:4942] (rev 40)
0000:f7:00.0 Co-processor [0b40]: Intel Corporation Device [8086:4942] (rev 40)
```

4. Verify if the GPU device is visible in the OS:

```
# lspci | grep -i display
0000:30:00.0 Display controller [0380]: Intel Corporation Device [8086:56c1] (rev 05)
0000:33:00.0 Display controller [0380]: Intel Corporation Device [8086:56c1] (rev 05)
```

5. Verify if SGX is enabled:

```
# dmesg | grep -i sgx
[    4.554704] sgx: EPC section 0x1070180000-0x107f3fefff
[    4.555636] sgx: EPC section 0x2070180000-0x207fffffff
```

Note: In case any of the above is not available, the user needs to set them as false in group_vars and host_vars file in Step 3

Step 2 - Download and Install

Ansible Host

1. Download the source code from GitHub repository for the Reference System server:

```
# git clone https://github.com/intel/container-experience-kits/
# cd container-experience-kits
# git checkout v23.10
```

2. Set up Python* virtual environment and install dependencies:

```
# python3 -m venv venv
# source venv/bin/activate
# pip3 install -r requirements.txt
```

3. Install Ansible dependencies for the Reference System:

```
# ansible-galaxy install -r collections/requirements.yml
```

Download & Install

Step 3 – Configure

Below are the steps to configure the Reference Systems. The **On-Prem VSS** configuration profile (on_prem_vss) is used for this deployment.

Configure

Ansible Host

1. Generate the configuration files:

```
# make k8s-profile PROFILE=on_prem_vss ARCH=spr
```

2. Update the **inventory.ini** file to match the deployment set-up. The values for *<target hostname>* and *<target IP>* must be updated to match the target systems in the BMRA cluster as shown in Figure 1.

```
# vim inventory.ini
[all]
<controller-hostname>  ansible_host=<controller IP> ip=<controller IP> ansible_user=root
<worker-hostname>      ansible_host=<worker IP> ip=<worker IP> ansible_user=root
localhost              ansible_connection=local ansible_python_interpreter=/usr/bin/python3

[vm_host]

[kube_control_plane]
<controller-hostname>

[etcd]
<controller-hostname>

[kube_node]
<worker-hostname>
```

```
[k8s_cluster:children]
kube_control_plane
kube_node

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

3. Update the host_vars filename with the target machine's hostname:

```
# mv host_vars/node1.yml host_vars/<worker-hostname>.yml
```

4. Update *host_vars/<worker-hostname >.yml* with PCI device information specific to the worker node server:

```
# vim host_vars/<worker-hostname>.yml
dataplane_interfaces:
  - bus_info: "18:00.0"                          # Use the SR-IOV PCI ID here
```

Note: Be sure to remove the square brackets [ ] that follows the 'dataplane_interfaces' configuration option. Additional details about the configuration options and values can be found as comments in the file.

5. Verify and update the *host_vars/<worker_ hostname>.yml* with the below settings for VSS:

```
# vim host_vars/<worker-hostname>.yml
configure_gpu: true
```

6. Verify and update the *group_vars/all.yml* with the below settings to enable GPU device plugin and AI libraries for VSS:

```
# vim group_vars/all.yml yml
intel_media_analytics_enabled: true
gpu_dp_enabled: true # Intel GPU Device Plugin for Kubernetes
gas_enabled: true    # GPU Aware Scheduling (GAS)

intel_oneapi:
  # Set to true to deploy Intel oneAPI Base Kit
  basekit: false
  # Set to true to deploy Intel oneAPI AI Analytics Kit
  ai_analytics: true
```

7. If the server is behind a proxy, update *group_vars/all.yml* by updating and uncommenting the lines for `http_proxy`, `https_proxy`, and `additional_no_proxy`.

```
# vim group_vars/all.yml

## Proxy configuration ##
http_proxy: "http://proxy.example.com:port"
https_proxy: "http://proxy.example.com:port"
additional_no_proxy: ".example.com,mirror_ip"
```

8. (Required) Apply required patches for Kubespray:

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

9. (Optional, recommended) Verify that Ansible can connect to the target servers, by running the below command and checking the output generated in the **all_system_facts.txt** file:

```
# ansible -i inventory.ini -m setup all > all_system_facts.txt
```

10. (Optional, Recommended) Check dependencies of components enabled in group_vars and host_vars with the packaged dependency checker. This step is also run by default as part of the main playbook:

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

Step 4 – Deploy

Ansible Host

Deploy

Now the Reference System can be deployed by using the following command:

```
# ansible-playbook -i inventory.ini playbooks/on_prem_vss.yml --flush-cache
```

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

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

Step 5 – Validate

Ansible Host

Validate

1. To interact with the Kubernetes CLI (kubectl), start by connecting to the controller node in the cluster, which can be done using the below commands:
```
# ssh root@<controller ip>
```
2. Once connected, the status of the Kubernetes cluster can be checked:
```
# kubectl get nodes -o wide
# kubectl get pods --all-namespaces
```

The sample output of the on-prem_vss profile deployment using BMRA is shown in the Figure 4. You can verify that all the containers and plugins are deployed successfully.

```
NAMESPACE              NAME                                                READY   STATUS              RESTARTS      AGE
cert-manager           cert-manager-7b64ddcf45-66d6c                       1/1     Running             0             23m
cert-manager           cert-manager-cainjector-5f7875f996-5ztmd            1/1     Running             0             23m
cert-manager           cert-manager-webhook-5989877d6d-xc7wk               1/1     Running             0             23m
intel-media            intel-media                                         0/1     ContainerCreating   0             17s
istio-system           istio-ingressgateway-cbb4b8f4d-bb6wt                1/1     Running             0             15m
istio-system           istioctl-958c8dbb5-tmtjg                            1/1     Running             0             16m
istio-system           istiod-56977df46c-tpvrx                             1/1     Running             0             15m
kube-system            calico-kube-controllers-5c5b57ffb5-ghtwk            1/1     Running             0             23m
kube-system            calico-node-2vksp                                   1/1     Running             0             33m
kube-system            calico-node-sjjhx                                   1/1     Running             0             33m
kube-system            container-registry-7fdf455d6c-zshwb                 2/2     Running             0             27m
kube-system            coredns-5c469774b8-klshl                            1/1     Running             0             23m
kube-system            coredns-5c469774b8-th2xw                            1/1     Running             0             31m
kube-system            dns-autoscaler-f455cf558-5824n                      1/1     Running             0             31m
kube-system            gas-gpu-aware-scheduling-cdc7b5f5b-s2zjz            1/1     Running             0             17m
kube-system            intel-dlb-plugin-mv6qk                              1/1     Running             0             21m
kube-system            intel-dsa-plugin-5gsfd                              1/1     Running             0             21m
kube-system            intel-gpu-plugin-bwwc8                              1/1     Running             0             21m
kube-system            intel-qat-plugin-p85lh                              1/1     Running             0             21m
kube-system            intel-sgx-plugin-8mrdp                              1/1     Running             0             22m
kube-system            inteldeviceplugins-controller-manager-5554cbcdc5-lrzvj   2/2   Running             0             23m
kube-system            kube-apiserver-ao09-15-fcp                          1/1     Running             0             26m
kube-system            kube-controller-manager-ao09-15-fcp                 1/1     Running             2 (29m ago)   34m
kube-system            kube-multus-ds-amd64-j5lgq                          1/1     Running             0             32m
kube-system            kube-multus-ds-amd64-nzn9w                          1/1     Running             0             32m
kube-system            kube-proxy-85tdt                                    1/1     Running             0             33m
kube-system            kube-proxy-spp7n                                    1/1     Running             0             33m
kube-system            kube-scheduler-ao09-15-fcp                          1/1     Running             0             17m
kube-system            kubernetes-dashboard-5bf8857fc6-85bzf               1/1     Running             0             23m
kube-system            kubernetes-metrics-scraper-75d7f948-4dflx           1/1     Running             0             23m
kube-system            local-volume-provisioner-jg554                      1/1     Running             0             32m
kube-system            nginx-proxy-al09-07-quanta                          1/1     Running             0             33m
kube-system            node-feature-discovery-master-89d7bf858-xfd4j       1/1     Running             0             25m
kube-system            node-feature-discovery-worker-crzmk                 1/1     Running             1 (24m ago)   25m
kube-system            tas-telemetry-aware-scheduling-5fc8d76999-wxz48     1/1     Running             0             17m
rook-ceph              csi-cephfsplugin-b2g8d                              2/2     Running             0             12m
rook-ceph              csi-cephfsplugin-provisioner-f5dc9f54d-c2qlp        5/5     Running             0             12m
rook-ceph              csi-nfsplugin-972hd                                 2/2     Running             0             12m
rook-ceph              csi-nfsplugin-provisioner-85df9c758c-rgrxp          4/4     Running             0             12m
rook-ceph              csi-rbdplugin-kmm68                                 2/2     Running             0             12m
rook-ceph              csi-rbdplugin-provisioner-67f9f5865c-sspq9          5/5     Running             0             12m
rook-ceph              rook-ceph-crashcollector-al09-07-quanta-7648d44db8-ds6pn   1/1     Running             0             12m
rook-ceph              rook-ceph-mds-myfs-a-c95c4f96-m8xwr                 2/2     Running             0             11m
rook-ceph              rook-ceph-mds-myfs-b-657cb4f8b9-4tzw6               2/2     Running             0             11m
rook-ceph              rook-ceph-mgr-a-6c5c4b9644-jkhmw                    3/3     Running             0             12m
rook-ceph              rook-ceph-mgr-b-bc5467cff-kh2r9                     3/3     Running             0             12m
rook-ceph              rook-ceph-mon-a-85f6cddc89-6fqdl                    2/2     Running             0             12m
rook-ceph              rook-ceph-mon-b-b6cc5d5f6-qvmkj                     2/2     Running             0             12m
rook-ceph              rook-ceph-mon-c-7f75899475-zhdnw                    2/2     Running             0             12m
rook-ceph              rook-ceph-operator-5bf55f645f-6nhnt                 1/1     Running             0             15m
rook-ceph              rook-discover-fzkwh                                 1/1     Running             0             13m
sriov-network-operator sriov-device-plugin-mzrw7                           1/1     Running             0             23m
sriov-network-operator sriov-network-config-daemon-wstpn                   3/3     Running             0             24m
sriov-network-operator sriov-network-operator-5d8fcdb497-mvq4q             1/1     Running             0             24m
```

Figure 4: Post deployment verification for on_prem_vss profile

5.1. Validation of the VSS workload:

To test the VSS workload the user can refer to Chapter 4.4 of the Network and Edge Reference System Architecture Integration with Workload Service Framework User Guide.

Additional feature verification tests can be found here:
https://github.com/intel/container-experience-kits/tree/master/validation/verification-manual

Reference Documentation

The *Network and Edge Bare Metal Reference System Architecture User Guide* provides information and full set of installation instructions for a BMRA.

The *Network and Edge Reference System Architectures Portfolio User Manual* provides additional information for the Reference Architectures including a complete list of reference documents.

The Network and Edge Reference System Architecture Integration with Workload Service Framework User Guide provides information about validating the VSS workload.

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

Document Revision History

REVISION	DATE	DESCRIPTION
001	July 2023	Initial release.
002	October 2023	Updated the BMRA version to 23.10 and added reference to Workload Service Framework VSS workload.

1023/DN/WIT/PDF 784338-002US