Quick Start Guide

Network and Edge Reference System Architectures - Video Production

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

Authors

Abhijit Sinha

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 and 5th Gen Intel® Xeon® Scalable processor**-based platforms for **Video Production**.

The Reference System has a variety of configuration profile settings for different network traffic workloads. **This quick start guide enables Video Production use case using Regional Data Center Configuration Profile**. For details on this and other Configuration Profiles or hardware options, refer to the User Guides listed in the Reference Documentation section.

Regional Data Center Configuration Profile Architecture

Figure 1 shows the architecture diagram of the Regional Data Center Configuration Profile (regional_dc), including additional features relevant for video production.

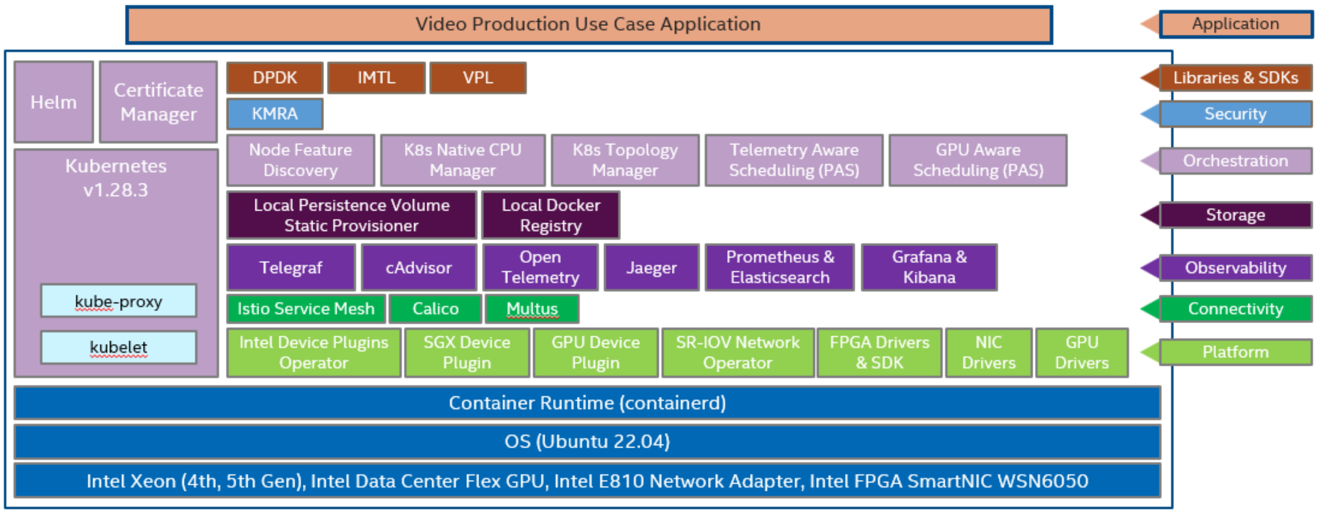

Figure 1: Architecture of the Regional Data Center Configuration Profile (regional_dc) with additions for Video Production

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

Hardware BOM

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

Ansible host	A laptop or server running a UNIX based distribution
Controller Node	Any 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
Ethernet Adapter	Intel® Ethernet Network Adapter E810-CQDA2 or E810-XXVDA2
Intel® FPGA	Intel® FPGA SmartNIC WSN6050 Platform (on worker node)
GPU	(Optional) Intel® Data Center GPU Flex 140 or Intel® Data Center GPU Flex 170
Recommended BIOS	Max Performance Turbo BIOS configuration is recommended (Refer Chapter 3.8 of BMRA User Guide)

For details of the hardware BOM for the **Regional Data Center Configuration Profile**, refer to the BMRA User Guide listed in the Reference Documentation section.

Software BOM

Following is the list of the software components that are utilized in this Reference System for Video Production:

BMRA Software	https://github.com/intel/container-experience-kits/
Orchestration	Kubernetes - v1.28.3
Scheduling	Node Feature Discovery, Platform Aware Scheduling
Observability & Telemetry	Telegraf, cAdvisor, Jaeger, OpenTelemetry, Elasticsearch, Prometheus, Kibana, Grafana
Networking & Connectivity	Multus CNI, Calico, Istio service mesh
Storage	Local Persistence Volume Static Provisioner (LPVSP)
Security	Key Management Reference Architecture (KMRA)
Media Libraries	Intel® Media Transport Library (IMTL) - v23.08 Video Processing Library (VPL) - v2023.3.0
Operators & Device Plugins	Intel® GPU device plugin Intel® SGX device plugin Intel® Device Plugins Operator SR-IOV Network Operator
Container Runtime	containerd
OS	Ubuntu (22.04.2 LTS)

Getting Started

Ansible playbooks are used to install the Bare Metal (BMRA), which set up the infrastructure for Video Production.

Deployment Setup

Figure 2 shows the deployment model for Regional Data Center Configuration Profile using BMRA.

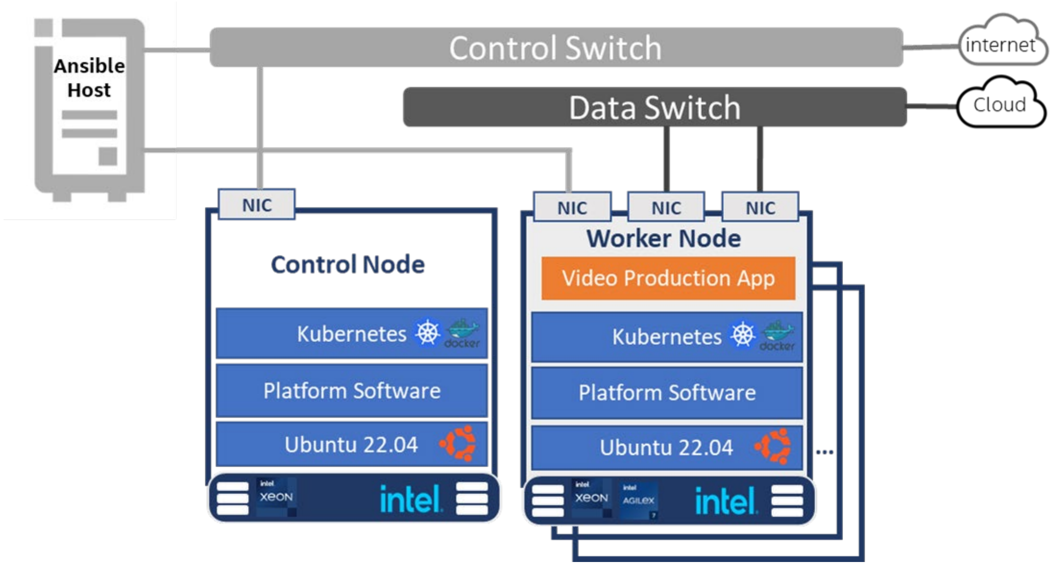

Figure 2: BMRA deployment setup for Video Production

Note: The K8s cluster deployed using BMRA is scalable to multiple controller and worker nodes.

Installation Flow for RA Deployment

Ansible playbooks are used to deploy the Bare Metal Reference Systems Architecture (BMRA). Before the playbooks can be run, there are a few steps to prepare the environment and change relevant configuration options.

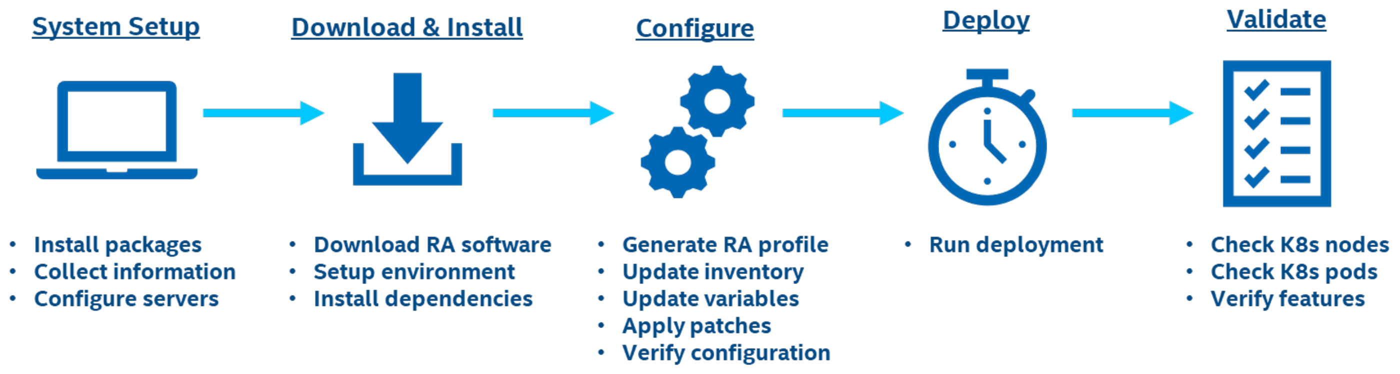

Figure 3: RA Deployment using Ansible playbooks

Step 1 - Set Up the System

System Setup

The below mentioned steps assume that both the Ansible host and target servers are running Ubuntu as the operating system.

Ansible Host

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 IP>
```
4. Verify passwordless connectivity to the target server:
```
# ssh root@<target IP>
```
5. For FPGA configuration, the installation script (e.g., fpga-ofs-2022-10-06-rc3-deb.sh) must be obtained and placed on the Ansible Host. The path and script name is needed in Step 3.

Target Servers

1. Install 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.
3. 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)
```

Step 2 - Download and Install

Download & 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 v24.01
```
2. Set up Python* virtual environment with necessary dependencies using pipenv:
```
# pip3 install pipenv
# pipenv install
# pipenv shell
```
Note: Alternative ways of installing dependencies can be found here: https://github.com/intel/container-experience-kits

3. Install Ansible dependencies for the Reference System:
```
# ansible-galaxy install -r collections/requirements.yml
```

Step 3 - Configure

Configure

The **Regional Data Center** configuration profile (regional_dc) is used to deploy the Reference System for Video Production.

Ansible Host

1. Generate the configuration files:
```
# make k8s-profile PROFILE=regional_dc ARCH=spr
```
2. Update the **inventory.ini** file with information about the target servers. The values for *<controller/worker hostname>* and *<controller/worker IP>* must be updated to match the target servers.
```
# 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. Copy the host_vars template for each worker node defined in inventory.ini:

```
# cp host_vars/node1.yml host_vars/<worker hostname>.yml
```

4. If the servers are behind a proxy, the following options in group_vars/all.yml can be uncommented and updated. If no proxy is used, the options should be left commented:

```
# 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"
```

5. Update group_vars/all.yml to enable features relevant for the Video Production use case:

```
# vim group_vars/all.yml

intel_media_transport_library_enabled: true # enables IMTL deployment
intel_media_transport_library:
  # Patch of ICE driver is needed, set to false if ICE driver is already patched
  # update_nic_drivers option in host_vars must be set to true in order to patch, build and
load ICE driver
  patch_nic_driver: true
```

6. By default, KMRA is enabled in the regional_dc configuration profile. To use this feature, an API key is required which can be requested through https://api.portal.trustedservices.intel.com/provisioning-certification. This key must be added in place of the api_key placeholder in group_vars/all.yml as shown below. Alternatively, KMRA can be disabled by setting all the Boolean options to false:

```
# vim group_vars/all.yml

kmra:
  sbx: false
  oran:
    enabled: false
    local_build: false
  oran_netopeer2_server:
    enabled: false
  oran_netopeer2_client:
    enabled: false
  pccs:
    enabled: true
    api_key: "ffffffffffffffffffffffffffffffff"
  apphsm:
    enabled: true
```

7. For SR-IOV functionality, there are a few configuration steps that must be performed. Start by enabling the SR-IOV Network Operator in group_vars/all.yml:

```
# vim group_vars/all.yml

sriov_network_operator_enabled: true
```

For each worker node, IOMMU and hugepages must be enabled and interface information, collected as part of Step 1, must be specified in host_vars/<worker hostname>.yml:

```
# vim host_vars/<worker hostname>.yml

iommu_enabled: true

hugepages_enabled: true
# Hugepage sizes available: 2M, 1G
```

```
default_hugepage_size: 1G
# Sets how many hugepages should be created
number_of_hugepages_1G: 4
number_of_hugepages_2M: 1024

#dataplane_interfaces: []
dataplane_interfaces:
  - bus_info: "18:00.0"                    # Use the SR-IOV PCI ID here
    pf_driver: ice
    default_vf_driver: "iavf"
    sriov_numvfs: 8
    sriov_vfs:                             # This is optional but can be used to change
the driver of specific VFs. Any VF not specified here will use the "default_vf_driver"
specified above.
      vf_00: "vfio-pci"
      vf_05: "vfio-pci"
```

Note: Additional details about the configuration options and values can be found as comments in the file.

8. For Intel® Media Transport Library, there are additional configuration options that must be specified for each worker node. The following options in host_vars/<worker hostname>.yml must be updated:

```
# vim host_vars/<worker hostname>.yml
```

DPDK must be enabled and version (optionally) changed to a known supported version:

```
install_dpdk: true
# The latest patches provided by Intel Media Transport Library are for DPDK v23.03
dpdk_version: "23.03"
```

The network adapter firmware must be at least version 4.20 for the SR-IOV PCI IDs that are added previously. This will be verified during the preflight check that happens prior to the start of the deployment. So, the below steps to enable "update_nic_firmware" can initially be skipped but complete if needed.

```
Update_nic_firmware: true
# The version can be specified to 4.20 as shown below, otherwise 4.30 is installed
nvmupdate:
  ice:
    required_fw_version: "4.20"
```

9. For FPGA support, the following options must be updated:

```
# vim host_vars/<worker hostname>.yml

iommu_enabled: true              # Should already be enabled due to SR-IOV

hugepages_enabled: true          # Should already be enabled due to SR-IOV

configure_fpga: true
# The below options must be uncommented and updated to match the path and filename of the
installation script located on the Ansible host.
Fpga_driver_staging_folder: /tmp/intel_fpga/
fpga_install_script: fpga-ofs-2022-10-06-rc3-deb.sh
```

10. For GPU support, including installation of Video Processing Library (VPL), the Intel® GPU Device Plugin for Kubernetes can be enabled:

```
# vim group_vars/all.yml
gpu_dp_enabled: true
```

Note: There are additional option flags for the device plugin available in group_vars/all.yml.

In addition, GPU configuration must be enabled for each worker node through host_vars/<worker hostname>.yml:

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

11. (Optional) FFmpeg libraries can be enabled:

```
# vim group_vars/all.yml
ffmpeg_install_enabled: true
```

Note: There are additional options for custom FFmpeg patches in group_vars/all.yml

12. (Required) Apply required patches for Kubespray:

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

13. (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
```

14. (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

Deploy

Ansible Host

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

```
# ansible-playbook -i inventory.ini playbooks/regional_dc.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

Validate

Ansible Host

1. To interact with the Kubernetes CLI (kubectl), start by connecting to a 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
```

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 Cloud Reference System Architecture User Guide* provides the means to develop and deploy cloud-native applications in a CSP environment and still experience Intel® technology benefits.

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

Other collaterals, including technical guides and solution briefs that explain in detail the technologies enabled in the Reference Systems 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 BMRA version to 23.10 and added profile architecture.
003	January 2024	Updated BMRA version to 24.01.

0124/DN/WIT/PDF 784340-003US