Quick Start Guide

Network and Edge Reference System Architectures - CDN

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

Authors

Abhijit Sinha

Renzhi Jiang

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 for **Content Delivery Network (CDN)**.

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

Hardware BOM

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

Ansible host	Laptop or server running a UNIX base distribution with an internet connection
Controller Node	Any 4th or 5th Gen Intel® Xeon® Scalable processor-based server [Minimum 1 controller node]
Target Server	2x 4th or 5th Gen Intel® Xeon® Scalable processors on Intel SDP S2EG4SEQ5Q [Minimum 2 worker nodes]
Ethernet Adapter	Intel® Ethernet Network Adapter E810-CQDA2
Storage	6x Kioxia CM6 3.2 TB NVMe PCIe 4x4 2.5"15mm SIE 3DWPD - KCM6XVUL3T20 [on worker nodes]
Recommended BIOS	"Max Performance Turbo" BIOS configuration (refer to Chapter 3.8 of the BMRA User Guide) - Intel® SGX needs to be enabled in the BIOS

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

Software BOM

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

Security	OpenSSL, Intel® SGX, Intel® QAT
Storage	LPVSP
Observability	Prometheus, Telegraf, Jaeger, Open Telemetry, Elastic Search, Kibana, Cadvisor, Grafana
Acceleration/ Data Plane	DPDK
Operators & Device plugins	Intel® QAT and Intel® SGX plugins, Multus, SRIOV network operator, Intel® Ethernet Operator
Container Runtime	containerd
Orchestration	Kubernetes v1.28.3, Telemetry Aware Scheduling (TAS), CPU Manager
OS	Ubuntu 22.04.2 LTS (kernel 5.15.0-72 generic) and RHEL 9.2

For details on the software versions for the **On-Premises Edge Profile**, refer to the BMRA User Guides listed in the Reference Documentation section.

The **On-Premises Edge Configuration Profile** enables storage-related features such as local block file storage LPVSP (Local Persistence Volume Static Provisioner). The configuration profile also enables security features like Intel® QAT and Intel® SGX.

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 with Intel® SGX enabled.

Deployment Setup

Figure 1 shows the deployment model for Remote Central Office-Forwarding Configuration using BMRA. The Ansible host is used for configuring and deploying BMRA on a set of target servers. The K8s cluster deployed using BMRA is scalable to multiple controller and worker nodes.

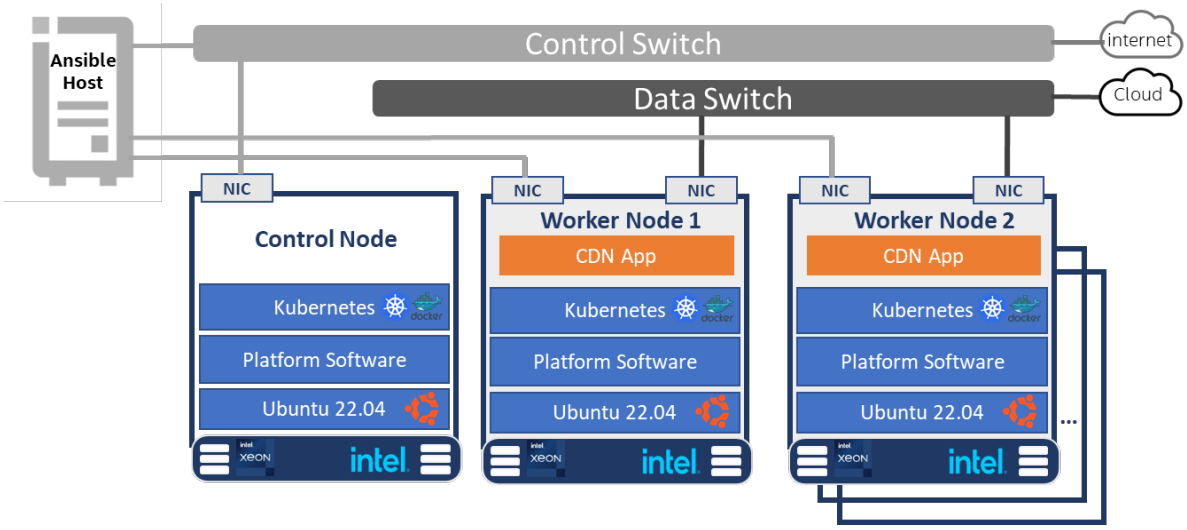

Figure 1: BMRA deployment setup for CDN

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.

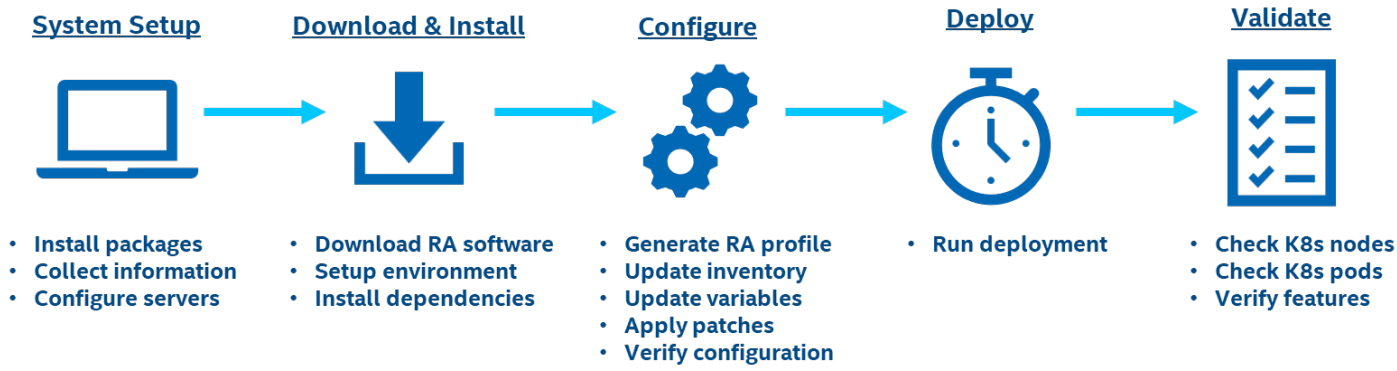

Figure 2: RA Deployment using Ansible Playbooks

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 IP>
```
4. Verify passwordless connectivity to the target servers:
```
# ssh root@<target IP>
```

Target Server

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 the dataplane_interfaces in the 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)
```

4. Find the relevant Intel® QAT device IDs (bus:device.function) using 'lspci', and note down the IDs for later when configuring the qat_devices in host_vars on the Ansible host:

```
# lspci -nn | egrep -e '8086:37c8|8086:19e2|8086:0435|8086:6f54|8086:4940|8086:4942'
e8:00.0 Co-processor: Intel Corporation Device 4940 (rev 40)
```

Step 2 – Download and Install

Ansible Host

Download & Install

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 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
```

Step 3 – Configure

Configure

The **On-Premises Edge** configuration profile (on_prem) is used for this deployment.

Ansible Host

1. Generate the configuration files:

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

2. Update the **inventory.ini** file to match the cluster deployment. The values for *<target hostnames>* and *<target IPs>* must be updated to match the Reference System cluster.

```
# vim inventory.ini
[all]
<controller hostname>  ansible_host=<controller IP> ip=<controller IP> ansible_user=root
<worker1 hostname>     ansible_host=<worker IP> ip=<worker IP> ansible_user=root
<worker2 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]
<worker1 hostname>
<worker2 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 worker node server machine's hostname:

```
# cp host_vars/node1.yml host_vars/<worker1 hostname>.yml
# cp host_vars/node1.yml host_vars/<worker2 hostname>.yml
```

4. Update host_vars on all worker nodes with PCI device information of the data plane interfaces specific to that server:

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

5. Update host_vars on all worker nodes with Intel® QAT device information specific to that server:

```
# vim host_vars/<worker hostname>.yml
qat_devices:
  - qat_id: "0000:e8:00.0"            # QAT device id one using DPDK compatible driver for
                                        VF devices to be used by vfio-pci kernel driver.
```

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

6. Change the hugepages settings in host_vars on all worker nodes:

```
# vim host_vars/<worker hostname>.yml
default_hugepage_size: 2M
number_of_hugepages_2M: 4096
```

7. To use in-tree drivers, disable the Intel® QAT driver update in host_vars on all worker nodes:

```
# vim host_vars/<worker hostname>.yml
update_qat_drivers: false
```

Local Static Volume Provisioner Configuration

1. Configure the disk information in host_vars for Local Storage:

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

#persistent_volumes: []
persistent_volumes:
  - name: "mnt-data-1"
    storageClassName: "local-storage"
    accessMode: "ReadWriteOnce"
    persistentVolumeReclaimPolicy: "Retain"
    mountPath: /mnt/disks/disk0
    device: /dev/nvme0n1
    fsType: ext4

  - name: "mnt-data-2"
    storageClassName: "local-storage"
    accessMode: "ReadWriteOnce"
    persistentVolumeReclaimPolicy: "Retain"
    mountPath: /mnt/disks/disk1
    device: /dev/nvme1n1
    fsType: ext4
```

Note: Additional configuration options and values for the persistent volumes can be found as comments in the file.

Remote Storage Configuration

The remote storage only supports containerd and crio as the container runtime. Below are the steps to configure the remote storage functionality:

1. Configure the disk/storage node information in group_vars for Remote Storage:

```
# vim group_vars/all.yml

  storage_nodes: []    #if there is no setting, all kubenode will be used as storage node.
  # storage_nodes:
  #   - node0
  #   - node1
```

2. Configure the disk information in host_vars for Remote Storage:

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

persistent_volumes:
  - name: "mnt-data-1"
    storageClassName: "local-storage"
    accessMode: "ReadWriteOnce"
    persistentVolumeReclaimPolicy: "Retain"
    mountPath: /mnt/disks/disk0
```

```
    device: /dev/nvme0n1
    fsType: ext4
```

Once the cluster is successfully deployed, the user can start using the remote storage function in the cluster.

3. Change the container runtime to containerd in the group_vars:

```
# vim group_vars/all.yml
container_runtime: containerd
```

4. Update the Intel device plugin operator namespace name the group_vars:

```
# vim group_vars/all.yml
intel_dp_namespace: inteldeviceplugins-system
```

5. Verify that the local persistence volume static provisioner (LPVSP) is enabled in the group_vars:

```
# vim group_vars/all.yml
storage_deploy_test_mode: true
local_volume_provisioner_enabled: true
```

6. 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. Add the CDN IP address to no_proxy setting.

```
# 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,<Add the IP address used for CDN>"
```

7. (Required) Apply required patches for Kubespray:

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

8. (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
```

9. (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.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 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

Validation of the CDN workload

To test the CDN workload, the user can refer to Chapter 4.5 of the Network and Edge Reference System Architecture Integration with Workload Service Framework User Guide.

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 Systems 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, added installation flow for deployment and added reference to WSF-based CDN workload validation.
003	January 2024	Updated BMRA version to 24.01 and added installation flow for remote storage.

0124/DN/WIT/PDF 784335-003US