Quick Start Guide

Network and Edge Reference System Architectures - Single Server

Develop and verify cloud-native services on a single server setup based on 4th or 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)** or the **Virtual Machine Reference System Architecture (VMRA)** on a single **4th or 5th Gen Intel® Xeon® Scalable processor-based** platform.

The Reference System has a variety of configuration profile settings for different network traffic workloads. **This quick start guide has selected the Basic Configuration Profile** but can also be used with the **Build-Your-Own Configuration Profile**. For details on these and other Configuration Profiles, and other hardware options, refer to the User Guides listed in the Reference Documentation section.

Release 24.01 also includes other Workload-specific Quick Start Guides as listed in the Reference Documentation section.

Hardware BOM

Following is the list of the hardware components that are required for setting up the system on a single server:

Ansible host	Laptop or server running a UNIX base distribution
Target Server	4th or 5th Gen Intel® Xeon® Scalable processor-based server – Quanta S6Q (S2EG3SEQ8B)
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)

Software BOM

Following is the list of the software components that are required for setting up the system on a single server:

Security	OpenSSL
Storage	LPVSP (Local Persistent Volume Static Provisioner)
Observability	Prometheus, Telegraph, Jaeger, Open Telemetry, Elastic Search, Kibana, Cadvisor, Grafana

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

Operators & Device plugins	Multus, Node feature discovery
Container Runtime	containerd
Orchestration	Kubernetes v1.28.3
OS	Ubuntu 22.04.2 LTS (kernel 5.15.0-72 generic) and RHEL 9.2

For more details of software versions for the **Basic** and **Build-Your-Own Configuration Profiles**, refer to Chapter 4 of BMRA User Guides listed in the Reference Documentation section.

Getting Started

Ansible playbooks are used to install the Bare Metal (BMRA) and Virtual Machine (VMRA).

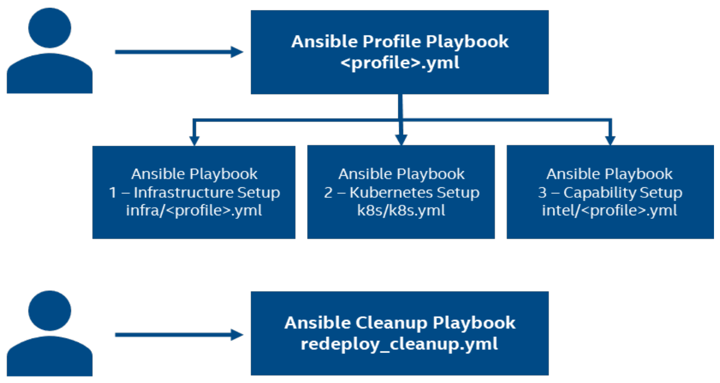
Pre-Requisites

Before starting the deployment, perform the following steps:

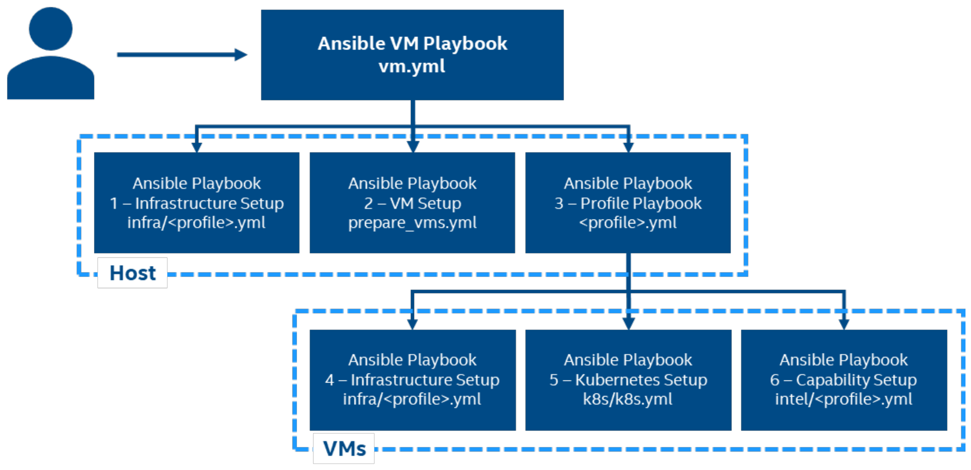
- A fresh OS installation is expected on the controller and target nodes to avoid a conflict between the RA deployment process with the existing software packages. To deploy RA on the existing OS, ensure that there is no prior Docker or Kubernetes* (K8s) installations on the server(s).
- The hostname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

BMRA - provisioning is split into three steps:

1. **Infrastructure**
2. **Kubernetes**
3. **Capabilities**

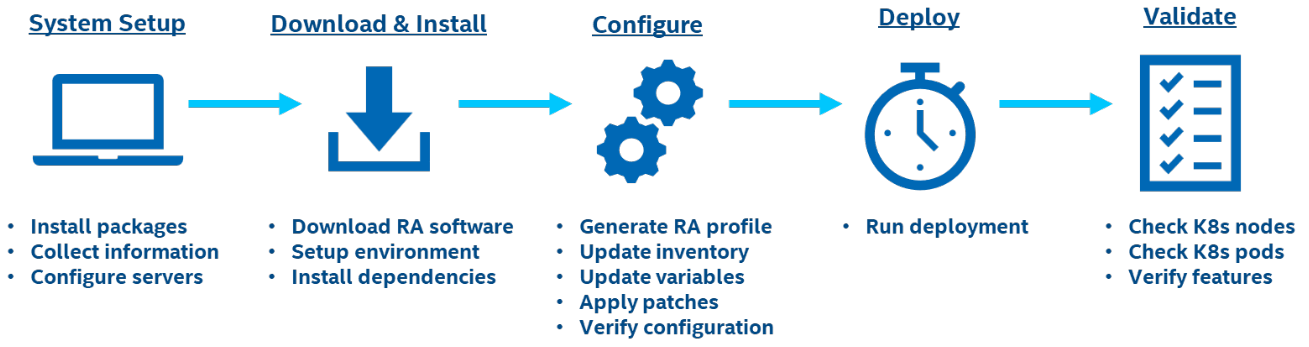

Ansible Profile Playbook <profile>.yml
Ansible Playbook 1 – Infrastructure Setup infra/<profile>.yml
Ansible Playbook 2 – Kubernetes Setup k8s/k8s.yml
Ansible Playbook 3 – Capability Setup intel/<profile>.yml
Ansible Cleanup Playbook redeploy_cleanup.yml

VMRA installation is split between the host and the VMs. On the host, the steps are: **Infrastructure** and **VM Setup**. Once VMs have been created, the Ansible inventory is updated, and the same three steps used for BMRA are repeated for the virtual machines (VMs).


Ansible VM Playbook vm.yml
Ansible Playbook 1 – Infrastructure Setup infra/<profile>.yml
Ansible Playbook 2 – VM Setup prepare_vms.yml
Ansible Playbook 3 – Profile Playbook <profile>.yml
Host
Ansible Playbook 4 – Infrastructure Setup infra/<profile>.yml
Ansible Playbook 5 – Kubernetes Setup k8s/k8s.yml
Ansible Playbook 6 – Capability Setup intel/<profile>.yml
VMs

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.



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 server:
```
# ssh-copy-id root@<target IP>
```
4. Verify passwordless connectivity to the target server:
```
# 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.

   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)
```

Target Server - Additional steps (only needed for VMRA)

For VMRA information about PCI, IDs of Virtual Functions (VFs) must be specified for the configuration of VMs.

1. Using device `18:00.0` as an example, create VFs for the device:
```
# echo "2" > /sys/bus/pci/devices/0000\:18\:00.0/sriov_numvfs
```
2. Find the relevant PCI IDs for the SR-IOV VFs:
```
# lspci | grep Eth | grep Virtual
18:01.0 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (rev 01)
18:01.1 Ethernet controller: Intel Corporation Ethernet Adaptive Virtual Function (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 dependencies using pipenv:

```
# pip3 install pipenv
# pipenv install
# pipenv shell
```

3. Install Ansible dependencies for the Reference System:

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

Step 3 - Configure

Configure

The **Basic** configuration profile (basic) is used for this deployment. The BMRA and VMRA configuration steps are different.

Configure BMRA

Ansible Host

1. Generate the configuration files:

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

Note: The fully customizable **Build-Your-Own Configuration Profile** can be selected by replacing `PROFILE=basic` with `PROFILE=build_your_own` in the above command.

2. Update the **inventory.ini** file to match a one server deployment. The values for *<target hostname>* and *<target IP>* must be updated to match the target system.

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

[kube_control_plane]
<target hostname>

[etcd]
<target hostname>

[kube_node]
<target 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/<target hostname>.yml
```

To utilize features depending on SR-IOV, host_vars must be updated with information about the PCI devices on the target server. The example below can be used as a reference for the configuration but should be updated to match the correct PCI IDs of the target server.

4. Update host_vars/<target_hostname>.yml with PCI device information specific to the target server:

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

5. 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"
```

6. (Required) Apply required patches for Kubespray:

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

7. (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
```

8. (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
```

Configure VMRA

Ansible Host

1. Generate the configuration files:

```
# make vm-profile PROFILE=basic ARCH=spr
```

Note: The fully customizable **Build-Your-Own Configuration Profile** can be selected by replacing `PROFILE=basic` with `PROFILE=build_your_own` in the above command.

2. Update the **inventory.ini** file to match a one server deployment. The values for *<target hostname>* and *<target IP>* must be updated to match the target system. Do not remove or modify the commented lines with “vm-ctrl-1” and “vm-work-1”.

```
# vim inventory.ini

[all]
<taget hostname>    ansible_host=<target IP> ip=<target IP> ansible_user=root
localhost           ansible_connection=local ansible_python_interpreter=/usr/bin/python3

[vm_host]
<target hostname>

[kube_control_plane]
#vm-ctrl-1

[etcd]
#vm-ctrl-1

[kube_node]
#vm-work-1

[k8s_cluster:children]
kube_control_plane
kube_node

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

3. Update the filename of the host_vars file for the target machine:

```
# mv host_vars/host-for-vms-1.yml host_vars/<target hostname>.yml
```

4. To utilize features depending on SR-IOV, host_vars must be updated with information about the PCI devices on the server. The below example can be used as a reference for the configuration but should be updated to match the correct PCI IDs of the target server.

   Due to the additional layer of virtualization through the VMs, both host_vars/<target hostname>.yml and host_vars/vm-work-1.yml will need to be updated with PCI information:

```
# vim host_vars/<target hostname>.yml
dataplane_interfaces:
  - bus_info: "18:00.0"                      # Use the SR-IOV PCI ID here
    pf_driver: ice
    flow_configuration: false
    default_vf_driver: "vfio-pci"            # The driver here should always be "vfio-pci"
    sriov_numvfs: 8
vms:
  - type: "ctrl"
    (...)
  - type: "work"
    (...)
    pci:
      - "18:01.0"          # Use the SR-IOV VF PCI ID(s) here
      - "18:01.1"          # Use the SR-IOV VF PCI ID(s) here
```

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

   Note: Be sure to remove the square brackets [ ] that follows the 'dataplane_interfaces' configuration option by default.

5. After updating the PCI assignment to the worker VM, the configuration in host_vars/vm-work-1.yml must be updated to use these devices. Take note of the PCI IDs used here, as they follow the order in which the devices are assigned to the VM above. Using this example, the VF with ID "`18:01.0`" above is the first in the list, which will correspond to "`06:00.0`" inside of the VM as shown below. Similarly, "`18:01.1`" will be "`07:00.0`" in the VM.

```
# vim host_vars/vm-work-1.yml
dataplane_interfaces:
  - bus_info: "06:00.0"
    pf_driver: iavf                      # Do not modify the value of "pf_driver"
    sriov_numvfs: 0                      # Do not modify the value of "sriov_numvfs"
    default_vf_driver: "igb_uio"         # Do not modify the value of "default_vf_driver"
  - bus_info: "07:00.0"
    pf_driver: iavf
    sriov_numvfs: 0
    default_vf_driver: "igb_uio"
```

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.

```
# 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"
```

7. (Required) Apply required patches for Kubespray:

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

8. (Optional) Verify that Ansible can connect to the target server, 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. (Recommended) It is recommended that you check dependencies of components enabled in group_vars and host_vars with the package dependency checker:

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

Step 4 - Deploy

Deploy

Ansible Host

Now the Reference System can be deployed. The command used depends on the choice between [BMRA] and [VMRA] above:

```
[BMRA] # ansible-playbook -i inventory.ini playbooks/basic.yml
[VMRA] # ansible-playbook -i inventory.ini playbooks/vm.yml
```

Note: If you are using the **Build-Your-Own Configuration Profile**, for BMRA in the commands above you will need to replace `basic.yml` with `build_your_own.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 for either BMRA or VMRA:

```
(BMRA) # ssh root@<target ip>
(VMRA) # ssh vm-ctrl-1           # Uses an alias specified in
/root/.ssh/config
```

2. Once connected, the status of the Kubernetes cluster can be checked:

```
# kubectl get nodes -o wide
# kubectl get pods --all-namespaces
```

Note: For single node Kubernetes deployment using BMRA, both controller and worker node are the same.

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 Virtual Machine Reference System Architecture User Guide* provides information and full set of installation instructions for a VMRA.

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 Systems including a complete list of reference documents.

Other available Quick Start Guides:

- Network and Edge Reference System Architectures - vRAN Setup with FlexRAN™ Software Quick Start Guide
- Network and Edge Reference System Architectures - 5G Core UPF 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 - Video Production Quick Start Guide
- Network and Edge Reference System Architectures - 5G vRAN Security Quick Start Guide
- Network and Edge Reference System Architectures - On Premises Edge AI Box Quick Start Guide

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

Document Revision History

REVISION	DATE	DESCRIPTION
001	May 2022	Initial release.
002	June 2022	Added more references in the Getting Started section.
003	July 2022	Updated for BMRA Release 22.06.
004	October 2022	Updated for BMRA Release 22.08.
005	December 2022	Updated for BMRA Release 22.11.
006	March 2023	Updated for BMRA Release 23.02.
007	July 2023	Updated for BMRA Release 23.07.
008	October 2023	Updated for BMRA Release 23.10.
009	January 2024	Updated for BMRA Release 24.01.

0124/DN/WIT/PDF 731862-009US