User Guide

Network and Edge Reference System Architecture - Integration with Intel® Workload Services Framework

Authors

Abhijit Sinha

Zhifang Long

Yanping Wu

Syed Faraz Ali Shah

Hao Hu

1 Introduction

1.1 Purpose and Scope

The Network and Edge Reference System Architecture (Reference System[1]) provides a portfolio of cloud-native software that enables deployments based on Intel® hardware platforms and accelerators, combined with the latest software innovations. The Reference System enables different deployment models (bare metal, virtual machine, and cloud) addressing Native, Private, and Public-Cloud (AWS*, Microsoft Azure*) deployments.

The Intel® Workload Services Framework repository provides a set of workloads that can be used to exercise multiple infrastructure platforms. The Intel® Workload Services Framework enables you to run a proxy workload (or proxy application) on top of the Reference System provisioned system. Each workload is a complete software benchmark implemented as an Open Container Initiative (OCI) compatible container. Workloads can be built and run collectively or individually.

This document describes how the integration of the Reference System with the Intel® Workload Services Framework provides a complete proxy system that can be used for system benchmark purposes, technology POC, or as a reference for the solution developed.

See the Network and Edge Reference System Architectures Portfolio User Manual for an overview of the Network and Edge Reference System Architecture. For details on the Reference System releases, see the Container Experience Kits GitHub page.

The Intel® Workload Services Framework provides a set of workloads that are available in the following location: https://github.com/intel/workload-services-framework.

1.2 Version 24.01 Release Information

Experience Kits, the collaterals that explain in detail the technologies enabled in release 24.01, including benchmark information, are available in the following location: Network & Edge Platform Experience Kits.

For NDA material, contact your regional Intel representative.

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

Table of Contents

Figures

Tables

Document Revision History

Revision	Date	Description
001	December 2022	Initial release.
002	July 2023	Updated for public distribution to Intel Network Builders.
003	October 2023	Added new workload-specific setup steps for Video Structure and CDN.
004	January 2024	Added new workload-specific setup steps for Istio Envoy.

2 Overview

2.1 General Overview

Several workloads in the Intel® Workload Services Framework must be deployed on specified hardware and software configurations. The Reference System provides this base platform. As described in Figure 1, the Reference System enables the IA-based cloud-native system capabilities, thus acts as a natural foundational layer, and supports consistent and seamless execution of workloads in the Intel® Workload Services Framework across different infrastructure setups. It supports a Cloud Native approach by providing orchestration layers through Kubernetes* or can allow vertical and monolithic workloads to run on top of it.

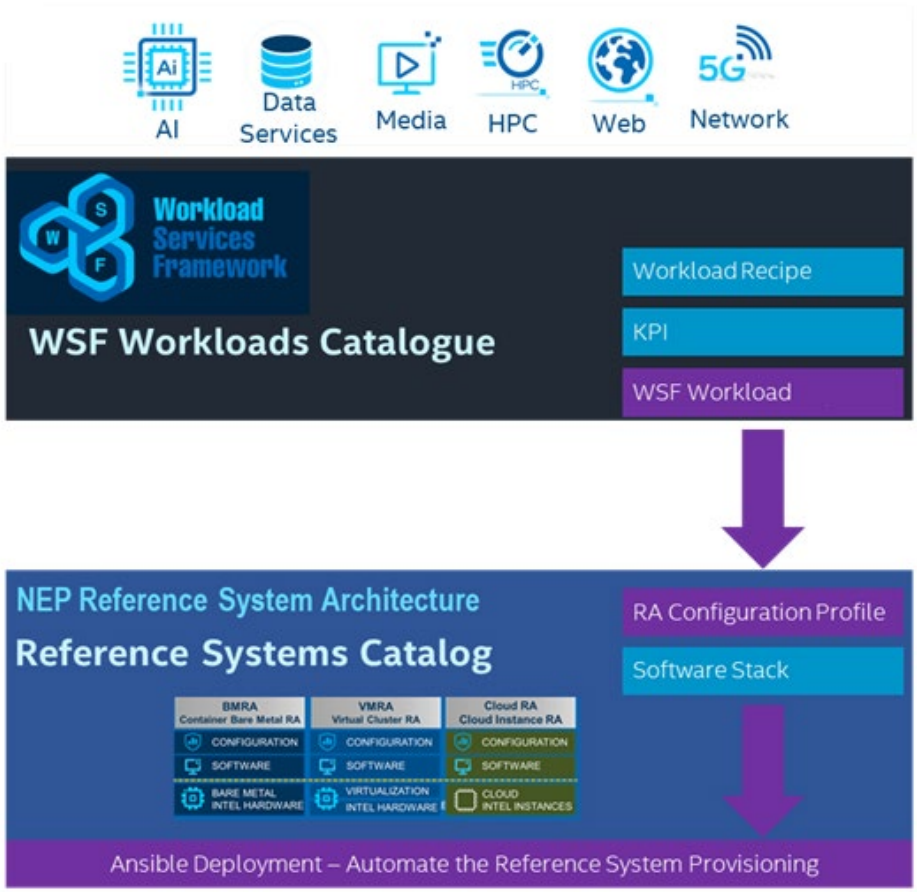

Figure 1. Intel® Workload Services Framework workloads deployed on top of the Reference System Architecture

3 General Setup Steps for Reference System Architecture and Intel® Workload Services Framework

Below is the overall deployment diagram to run the Intel® Workload Services Framework workloads on RA.

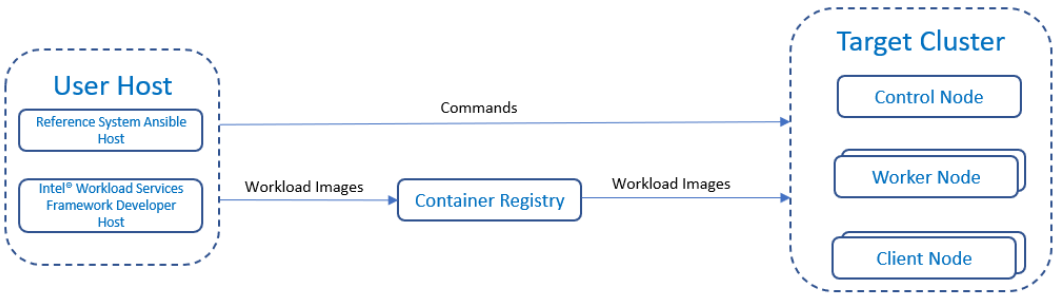

Figure 2. Intel® Workload Services Framework workloads on top of the Reference System Architecture deployment diagram

There are several servers or server roles in the deployment diagram:

- **User Host**:
  - **Reference System Ansible* Host** - The Server or PC that the user used to execute the Reference System Ansible scripts
  - **Intel® Workload Services Framework Developer Host** - The Server or PC that the user used to build the Intel® Workload Services Framework workload and execute the Intel® Workload Services Framework scripts

  The Reference System Ansible Host and the Intel® Workload Services Framework Developer Host can be the same server or different servers.

- **Target Cluster:**
  The servers to be deployed by the Reference System and used to run the Intel® Workload Services Framework workload to get performance data. It includes:
  - **Control Node** - The Server or PC that the user used as K8s control node
  - **Worker Nodes** - The Servers or PCs that the user used as K8s worker node
  - **Client Nodes** - The Servers or PCs that the user used as workload client

  The control node and worker node can be the same server or different servers, depending on the workload requirement. Reference System can deploy the client node like a worker node.

- **Container Registry:**

The Server or PC that the user used to store the workload container images. The Intel® Workload Services Framework Developer Host will build workload images and push them to this registry, and then the Target Cluster will pull these images to run. The Container Registry can be located in the User Host or Target Cluster.

NOTE: Do not put User Host and Target Cluster on the same server; it may result in user account access right issue.

3.1 Reference System Architecture - General Setup Steps

For detailed steps on setting up the Reference System, please refer to *Network and Edge Container Bare Metal Reference System Architecture User Guide*. Here is a summary of the main steps:

1. Clone the Reference System repo from GitHub to Reference System Ansible Host.

```
# git clone https://github.com/intel/container-experience-kits.git {cek_dir}
```

2. Install dependencies, specify Reference System profile value, and generate host_vars, group_vars, and inventory files.

```
# cd {cek_dir}
# export PROFILE={profile_value}
# pip3 install -r requirements.txt
# ansible-galaxy install -r collections/requirements.yml
# make examples ARCH=<atom,core,**icx**,spr> NIC=<fvl,**cvl**>
```

3. Edit host_vars, group_vars and inventory files to specify the configuration and target machines.

```
# cp -r examples/k8s/${PROFILE}/group_vars examples/k8s/${PROFILE}/host_vars .
```

   - Edit inventory.ini file to specify the name, IP address, and username of each machine for the target cluster.
   - Edit host_vars and groups_vars files to specify common configurations for the target clusters.
   - Edit host_vars and groups_vars files to specify workload-specific options. Refer to Section 4 - Workload Specific Setup Steps.

4. Execute Reference System playbooks to deploy on the target cluster.

```
# ansible-playbook -i inventory.ini playbooks/k8s/patch_kubespray.yml
# ansible-playbook -i inventory.ini playbooks/preflight.yml
# ansible-playbook -i inventory.ini playbooks/${PROFILE}.yml
```

After completing the above-mentioned profile setup, you can perform additional workload-specific setup steps. Refer to Section 4 - Workload Specific Setup Steps.

3.2 Container Registry Setup Steps for Workload Image Transfer

The Intel® Workload Services Framework workload building process will generate workload container images and push them to a container registry. When running the Intel® Workload Services Framework test, those workload container images will be pulled from the container registry to the Target Cluster. There are two ways to set up the container registry for such usage.

3.2.1 Setup an insecure container registry

1. Start the container registry service on any of your servers.

```
# docker run --name registry -d  -p 5000:5000 --restart=always  -v /opt/data/registry:/var/lib/registry registry
```

2. Modify the Intel® Workload Services Framework developer machine Docker* option to allow push images to the registry.

```
# vim /etc/docker/daemon.json
    Add the following lines.
{
  "insecure-registries": ["registry_ip:port"]
}
# systemctl daemon-reload
# systemctl restart docker
```

3. Modify the target cluster Docker option to allow pulling image from the insecure registry.

```
# vim /etc/systemd/system/docker.service.d/docker-options.conf
   add below lines:
   --insecure-registry {registry_ip:port}
# systemctl daemon-reload
# systemctl restart docker
```

3.2.2 Use the container registry Reference System deployed on Target Cluster

By default, the Reference System will deploy a container registry in the Target Cluster. To enable the User Host to push to or pull from this registry, the user must copy its certificate files to the User Host.

1. Change the value of "registry_local_address" in the Reference System group_vars before running the Reference System playbook.

```
group_vars:
registry_local_address: "control_node_hostname:{{ registry_nodeport }}" 8
```

2. Copy Target Cluster /docker/certs.d/{local_resgiry_name:port}/ca.crt to the same User Host folder.
3. Copy authorization token from target cluster `/root/.docker/config.json` to User Host `/home/{user}/.docker/config.json`

```
$ vim ~/etc/.docker/config.json
Add below lines :
    "auths": {
        "registry_hostname:30500": {
            "auth": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
        }
    },
$ sudo systemctl daemon-reload
$ sudo systemctl restart docker
```

3.3 Intel® Workload Services Framework - General Setup Steps

1. Clone the Intel® Workload Services Framework repo from GitHub to the Intel® Workload Services Framework Developer Host.

```
# git clone https://github.com/intel/workload-services-framework {wsf_dir}
```

2. Set up the Intel® Workload Services Framework Developer Host and specify the target cluster.

```
# cd {wsf_dir}/script/setup
# ./setup-dev.sh
# vim {wsf_dir} /script/terraform/terraform-config.static.if
```

Based on the workload needed, modify "worker_profile", "controller_profile", and "client_profile" sections to specify IP addresses and user names for the target cluster. The "worker_profile" and "controller_profile" should point to the cluster deployed by RA.

1. Build the Intel® Workload Services Framework workload and push it to the container registry.

```
# cd {wsf_dir}
# mkdir buid; cd build
# cmake -DPLATFORM=SPR -DREGISTRY={registry_ip:port} -DTIMEOUT=60000,3600 -DBENCHMARK= -
DRELEASE=:wsfonra -DBACKEND=terraform -DTERRAFORM_OPTIONS='--svrinfo --intel_publish --
intel_publisher_mongodb_name=services-framework --owner={username} ' -DACCEPT_LICENSE=ALL -
DTERRAFORM_SUT=static ../
```

2. Change the {registry_ip:port} and {username} values in above cmd line according to your environment.

```
# cd workload/{workload_name}
# make
```

3. Run the Intel® Workload Services Framework workload and obtain the KPI result.

```
# cd {wsf_dir}/build/workload/{workload_name}
```

Use ctest -N to check how many test cases can be run:

```
# ./ctest.sh -N
```

Use ctest -R to run the test case:

```
# ./ctest.sh -R {test_name} -V
```

Check test result and KPI value:

```
# ./list-kip.sh  {test_log_name}
```

4 Intel® Workload Services Framework - Workload Specific Setup Steps

For each Intel® Workload Services Framework workload, the user needs to specify the Reference System profile this workload requires, modify some Reference System group_vars and host_vars options before running the Reference System playbook, and may even need to run additional Reference System commands after completion of the Reference System playbook.

Different Intel® Workload Services Framework workloads should specify different test cases to run. This section describes the workload-specific settings or steps that the user should combine with the General Setup Steps described in Section 3.

4.1 Malconv

1. Deploy Reference System with profile: basic
2. Modify Reference System host_vars and group_vars files to specify the below values before running the Reference System playbook. For this workload, the default configuration is enough; no change is needed.
3. Deploy cluster with selected Reference System profile:

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

4. Build the Intel® Workload Services Framework workload and run it:

```
# cd {wsf_dir}/build/workload/Malconv
# make
# ./ctest.sh -V
# ./list-kpi.sh log*
```

4.2 NGINX*

1. Deploy Reference System with profile: remote_fp
2. Modify Reference System host_vars and group_vars files to specify the below values before running the Reference System playbook:

```
host_vars:
        update_qat_drivers: false
        number_of_hugepages_1G: 8
        number_of_hugepages_2M: 4096
```

3. Deploy cluster with selected profile:

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

4. Build the Intel® Workload Services Framework workload and run it:

```
# cd {wsf_dir}/build/workload/Nginx
# make
# ./ctest.sh -R qathw -E 3node -V
# ./list-kpi.sh log*
```

4.3 OpenSSL* 3.0 -RSAMB

1. Deploy Reference System with profile: remote_fp
2. Modify Reference System host_vars and group_vars files to specify the below values before running the Reference System playbook:

```
host_vars:
        update_qat_drivers: false
        number_of_hugepages_1G: 8
        number_of_hugepages_2M: 4096
```

3. Deploy cluster with selected profile:

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

4. Build the Intel® Workload Services Framework workload and run it:

```
# cd {wsf_dir}/build/workload/OpenSSL3-RSAMB
# make
# ./ctest.sh -V
# ./list-kpi.sh log*
```

4.4 Video Structure

1. Deploy Reference System with profile: on_prem_vss
2. Modify Reference System host_vars and group_vars files to specify the below values before running the Reference System playbook:

```
host_vars:
        configure_gpu: true
        isolcpus_enabled: false
group_vars:
        native_cpu_manager_enabled: false
```

3. Deploy cluster with selected profile:

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

4. Build the Intel® Workload Services Framework workload and run it:

```
# cd {wsf_dir}/build/workload/Video-Structure
# make
# ./ctest.sh -R GPU -V
# ./list-kpi.sh log*
```

4.5 CDN

1. Make sure the server configuration meets the below requirements:
   - Network:
     - Dedicate network interfaces with IP addresses for CDN traffic: <node1_cdn_nic_ip> and <node2_cdn_nic_ip>
     - Separate network interface with IP addresses to communicate with Ansible Host: <node1_ansible_nic_ip> and <node2_ansible_nic_ip>
   - Storage:
     - Two NVME SSD disks installed on the worker node
2. Deploy Reference System with profile: on_prem
3. Modify Reference System host_vars and group_vars files to specify the below values before running the Reference System playbook:

```
host_vars:
        update_qat_driver : false
        default_hugepage_size: 2M
        number_of_hugepages_2M: 4096

        persistent_volumes:
        - name: "mnt-data-1"
          storageClassName: "local-storage"
          accessMode: "ReadWriteOnce
          persistentVolumeReclaimPolicy: "Retain"
          mountPath: /mnt/disks/disk0
          device: /dev/nvme1n1
          fsType: ext4
        - name: "mnt-data-2"
          storageClassName: "local-storage"
          accessMode: "ReadWriteOnce
          persistentVolumeReclaimPolicy: "Retain"
          mountPath: /mnt/disks/disk1
          device: /dev/nvme2n1
          fsType: ext4
group_vars：
        # Add CDN IP addresses to no_proxy setting
        additional_no_proxy: xxx, <node1_cdn_nic_ip>, <node2_cdn_nic_ip>
```

4. Deploy cluster with selected profile:

```
# ansible-playbook -i inventory.ini playbook/on_prem.yml
```

5. Build the Intel® Workload Services Framework workload and run it:

```
#  cd {wsf_dir}/build/workload/CDN--NGINX
# make
# -R qathw -E gated --options=--k8s_abort_on_failure=true --set NICIP_W1={node1_cdn_nic_ip} --set
QAT_RESOURCE_TYPE=qat.intel.com/generic -V
# ./list-kpi.sh log*
```

4.6 Istio Envoy

1. Deploy Reference System with profile: on_prem
2. Modify Reference System host_vars and group_vars files to specify the below values before running the Reference System playbook. This workload requires only the default configuration; no change is needed.
3. Deploy cluster with selected Reference System profile:

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

4. Check the version of Istio installed by Reference System (current 1.20) and WSF (current 1.16). If the versions are different, refer to the command below to uninstall the Reference System Istio. Use "--start-at -task" to specify the starting task, and "--step" to run Ansible step by step. Ensure to stop Ansible at the task "create istio-system namespace".

```
#  ansible-playbook -i inventory.ini playbooks/on_prem.yml --start-at-task=" remove existing istio service mesh
resources" --step
```

5. Build the Intel® Workload Services Framework workload and run it:

```
#  cd {wsf_dir}/build/workload/Istio-Envoy
# make
# ./ctest.sh  -N
# ./list-kpi.sh log*
```

5 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, 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 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 of a cluster on a public Cloud Service Provider. The cluster can be Kubernetes with containers based.
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. In addition, generic Configuration Profiles are available to developers for flexible deployments.
Reference System Architecture Flavor	An instance of Reference System 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	Automates quick, repeatable, and predictive deployments using Ansible playbooks. Various Configuration Profiles and Ansible scripts allow 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.
Intel® Platforms	Prescribes Intel platforms for optimized operations. The platforms are based on 4th Gen and 5th Intel® Xeon® Scalable processors. The platforms integrate Intel® Ethernet 700 and 800 Series, Intel® QuickAssist Technology (Intel® QAT), Intel® Server GPU (Graphic Processor Unit), Intel® Optane™ technology, and more.

6 Reference Documentation

The *Network and Edge Reference System Architectures Portfolio User Manual* contains a complete list of reference documents. Additionally, a bare metal-based reference system architecture (BMRA) deployment allows creation of a Kubernetes cluster on multiple nodes. The *Network and Edge Container Bare Metal Reference System Architecture User Guide* provides information and installation instructions for a BMRA. A virtual machine-based reference architecture (VMRA) deployment allows creation of a Kubernetes cluster for a Configuration Profile on a virtualized infrastructure. The *Network and Edge Virtual Machine Reference System Architecture User Guide* provides information and installation instructions for a VMRA. The Cloud Reference System Architecture (Cloud RA) provides the means to develop and deploy cloud-native applications in a CSP environment and still experience Intel® technology benefits. Find more details in the *Network and Edge Cloud Reference System Architecture User Guide*.

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

Appendix A Abbreviations

The following abbreviations are used in this document.

Table 2. Abbreviations

TERM	DESCRIPTION
AWS	Amazon Web Services
BMRA	Bare Metal Reference Architecture
CEK	Container Experience Kits
CLI	Command Line Interface
GPU	Graphic Processor Unit
I/O	Input/Output
K8s	Kubernetes
OCI	Open Container Initiative
OS	Operating System
PC	Personal Computer
QAT	Intel® QuickAssist Technology (Intel® QAT)
VMRA	Virtual Machine Reference Architecture
WSF	Workload Service Framework

0124/DN/WIT/PDF 757797-004US