ICN-MTSCN R5 Test Document

Introduction

ICN Pod Topology

Jenkins Information

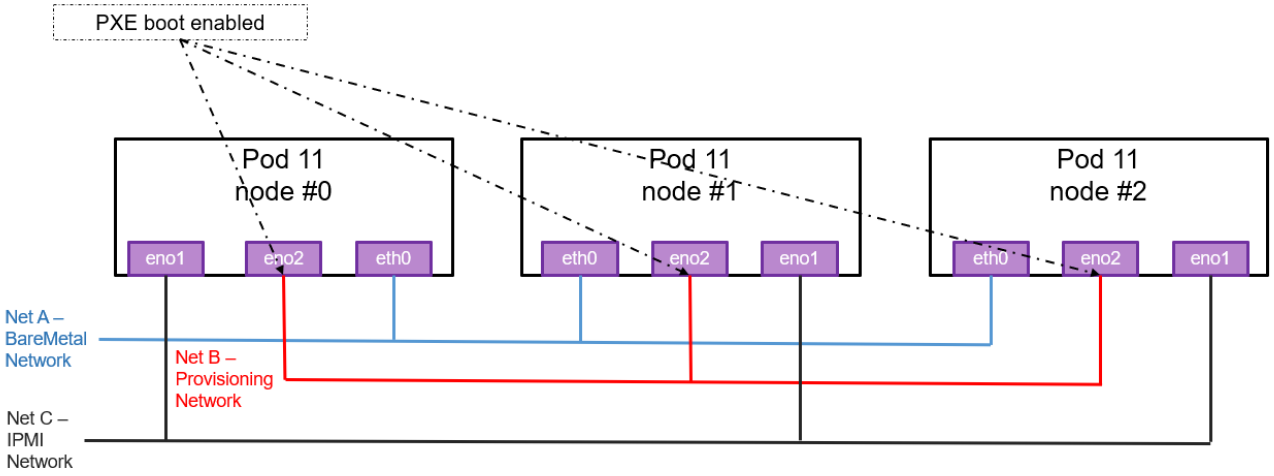
Akraino community has a public Jenkins cluster. ICN leverages the Akraino public Jenkins to run CI jobs.

Overall Test Architecture

Test Bed

Pod Topology

CD Topology – ICN Master BareMetal Deployment Verifier



Net A – BareMetal Network, lab networking for SSH. Used as control plane for Kubernetes, by OVN and flannel for the overlay networking with Internet Access
Net B (Internal networking) – Provisioning networking used by Ironic to do inspection (will have DHCP server)
Net C (Internal networking) – IPMI Lan to run IPMI protocol calls for OS provisioning

Bare-metal deployment

Hostname	CPU Model	Memory	BMC Firmware	Storage	1GbE: NIC#, VLAN, (Connected extreme 480 switch)	10GbE: NIC# VLAN, Network (Connected with IZ1 switch)	40GbE: NIC#
Jump	Intel 2xE5-2699	64GB	1.46.9995	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)	
node1	Intel 2xE5-2699	64GB	1.46.9995	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)	
node2	Intel 2xE5-2699	64GB	1.46.9995	3TB (Sata) 180 (SSD)	IF0: VLAN 110 (DMZ) IF1: VLAN 111 (Admin)	IF2: VLAN 112 (Private) **VLAN 114** (Management) IF3: VLAN 113 (Storage) **VLAN 1115** (Public)	IF4: SRIOV

Note: virtualization must already be enabled on the worker nodes that will be part of the Kubernetes cluster.

Test description

The 'Multitenant Secure Cloud Native Platform' provides the possibility to launch pods using Kata Containers. To use Kata Containers, Containerd is used in Kubernetes instead of the default docker-shim.

We use the next values for the `kud-installer.yaml` to properly run tests with Containerd and Kata Containers.

```
CONTAINER_RUNTIME: "containerd"

KUD_ENABLE_TESTS: "true"

ENABLE_KATA_WEBHOOK: "false"

KATA_WEBHOOK_RUNTIMECLASS: "kata-clh"
```

If ENABLE_KATA_WEBHOOK is set to true, then every pod that could run as a Kata container (e.g. infrastructure pods) will mutate to run as a Kata container. This could lead to some pods to be stuck in pending. If KUD_ENABLE_TESTS is set to true, then the webhook will be started before the verification tests are run to force Kata eligible pods to run as a Kata container. The webhook will be uninstalled after the tests run if ENABLE_KATA_WEBHOO is set to false.

Notes:

> We recommend to only enable the webhook provided by the Kata project for testing purposes as it may not meet production needs.
>
> For this blueprint, we are only running bare-metal testing as we have hit timeouts when double-nesting Kata Containers.

Testing

CI Testing:

Bashate:

> `bashate` test is used to check the shell scripts coding style. i.e. Trailing Whitespace. We find all files with suffix `.sh` and run `bashate` against the files. All vendor directories are excluded.

Golang testing:

BPA Operator:

A Configmap coming from the `kud-installer.yaml` file is what the BPA operator reads from to determine what CRI to use and whether to run the testing.

The BPA operator has unit tests using the go framework. The unit tests check the following:

- Job is created with the right job name for KUD installation.
- The job metadata has the right cluster name.
- Expected error is produced when a host with the specified MAC address is not found.
- Expected error is produced when no dhcp lease is found for the specified host.

BPA Rest Agent:

- Currently, automated unit tests are implemented using the Go testing framework.

CD Verifier(end-to-end testing):

All the test cases are tested as follows:

Metal3:

Metal3 verifier will check all the servers are provisioned, Metal3 verifier checks the status of the bare-metal servers for every 60 second for the provisioning status.

BPA Operator:

Bare-Metal host Provisioning

- The `bpa_verifier.sh` script get the MAC addresses and IP addresses of the 2 nodes provisioned by metal3, then creates a fake DHCP lease file using the IP address and MAC address information. It also creates a provisioning CR using the MAC address information.
- The script creates an ssh secret key using the ssh keys of the test host, applies the provisioning CR.
- The script busy loops until the KUD installation job completes or fails. If it completes successfully, it runs a curl command using the authentication info of the new cluster to confirm if it was successful or not. On completing all the steps, it runs a teardown where it deletes everything it created.

BPA Rest Agent

- Test script, `e2e_test.sh`, creates dummy image file, creates test JSON file, checks bpa rest agent status, issues POST, GET, and PATCH requests sequentially.
- Next, `e2e_test.sh` checks uploaded MinIO image object size, and calls DELETE.
- If the script fails at any point, then verification would be taken as unsuccessful.

Kubernetes Deployment (KuD)

KuD contains test cases to verify if the add-ons are running correctly. All the test cases can be found in tests directory in the multicloud-k8s project. For each of these, we bring up the deployment that is specific to the addon and perform add-on specific actions on the pod related to the deployment.

Multus:

- Multus CNI is a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods. This is accomplished by Multus acting as a "meta-plugin", a CNI plugin that can call multiple other CNI plugins.
- A 'NetworkAttachmentDefinition' is used to set up the network attachment, i.e. secondary interface for the pod.
- A pod is created with requesting specific network annotations with bridge CNI to create multiple interfaces. When the pod is up and running, we can attach to it to check the network interfaces on it by running *ip a* command.

OVN4NFV:

- OVN4NFV provides Provider networks using VLAN networking and Service Function Chaining.
- After the pod is up and running, we will be able to attach to the pod and check for multiple interfaces created inside the container.
- OVN4NFV networking is setup and created along the EMCO composite vFW testing.

Node Feature Discovery

- Node Feature Discovery for Kubernetes detects hardware features available on each node in a Kubernetes cluster and advertises those features using node labels.
- Creates a pod with specific label information in the case the pods are scheduled only on nodes whose major kernel version is 3 and above. Since the NFD master and worker daemonset is already running, the master has all the label information about the nodes which is collected by the worker.
- If the OS version matches, the Pod will be scheduled and launched. Otherwise, the Pod will be in a pending state in case there are no nodes with matching labels that are requested by the Pod.

CMK

- CPU Manager for Kubernetes provides CPU pinning for K8s workloads. In KUD, there are two test cases for the exclusive and shared CPU pools testing.

EMCO:

- EMCO Sanity testing checks the health connectivity EMCO Micro service once it is installed.

BluVal Testing

Status as of June 25th, 2021:

Layer	Result	Comments	Nexus
os/lynis	PASS with exceptions	Exceptions: • USB-2000 • SSH-7408: Checking MaxSessions, Checking Port • KRNL-6000: net.ipv4.conf.all.forwarding	Logs
os/vuls	PASS with exceptions	Exceptions: • CVE-2016-1585 • CVE-2017-18342 • CVE-2017-8283 • CVE-2018-20839 • CVE-2019-17041 • CVE-2019-17042 • CVE-2019-19814	Logs
k8s/conformance	PASS with exceptions	Exceptions: • Sonobuoy v0.16.1 does not support Kubernetes v1.18.9	Logs
k8s/kube-hunter	PASS	With aquasec/kube-hunter:edge image	Logs

Release 5 Blueprint Scanning Status

Akraino CVE Vulnerability Exception Request

Akraino BluVal Exception Request

CD Logs

ICN Master bare-metal Deployment Verifier

Test Dashboards

All the testing results are in logs.