ICN R6 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. While the CD jobs run in our private Jenkins cluster.

We have the following Jenkins slave nodes joined Akraino Jenkins. ICN CI jobs are supposed to be scheduled to our slave nodes by label icn-dev.

Slave Information			Server Information
Slave Name	Labels	Slave Root	Server Info
prd-ubuntu-dev-44c-64g	icn-dev	/home/jenkins/akraino/slave_root	pod14-node1

To add more Jenkins slave nodes, please follow the Akriano Jenkins guide

To setup private Jenkins, please refer to the README.md under icn/ci/

The private Jenkins cluster is setup on pod14-node2. We can visit the Jenkins with the node IP address: http://10.10.140.22:8080/

Currently we support only AIO private Jenkins.

Akraino Test Group Information

not applicable

Overall Test Architecture

Test Architecture

We support the following jobs

CI job

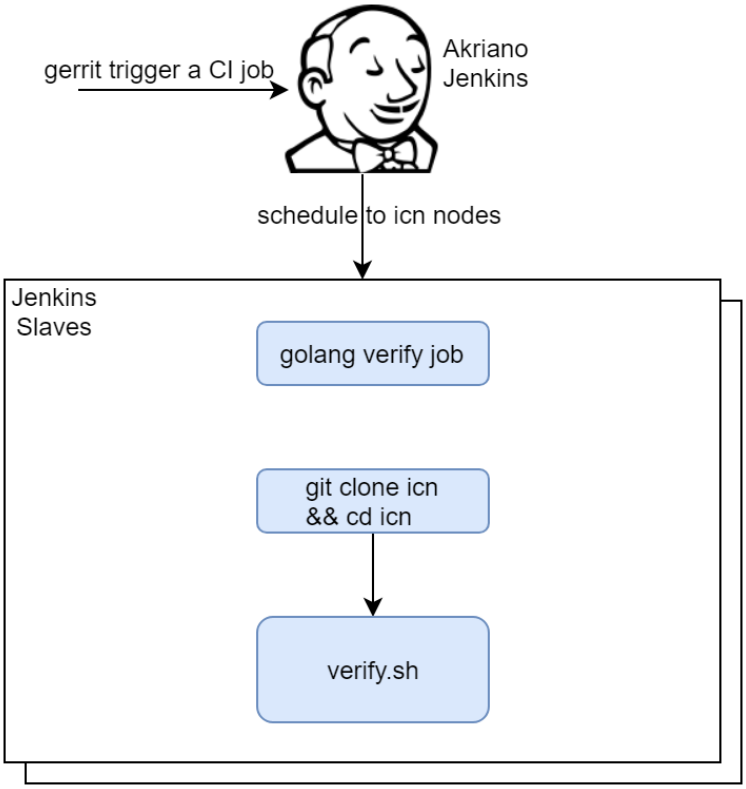
- Triggered by gerrit patch creation/update.
- The job runs verify.sh under ICN project. The verify.sh currently has integrated the bashate test.
- Post +1/-1 for gerrit patch if the build succeeds/fails
- Upload the job log to Nexus server in post-build actions

CD job for test

- Triggered daily automatically (We can also trigger it manually)
- Run a make command, which creates VM(s) and deploys ICN components on the VM(s)
- Upload the job log to Nexus server in post-build actions

CI job detail

Update the verify.sh can update the CI job content.



CD job detail

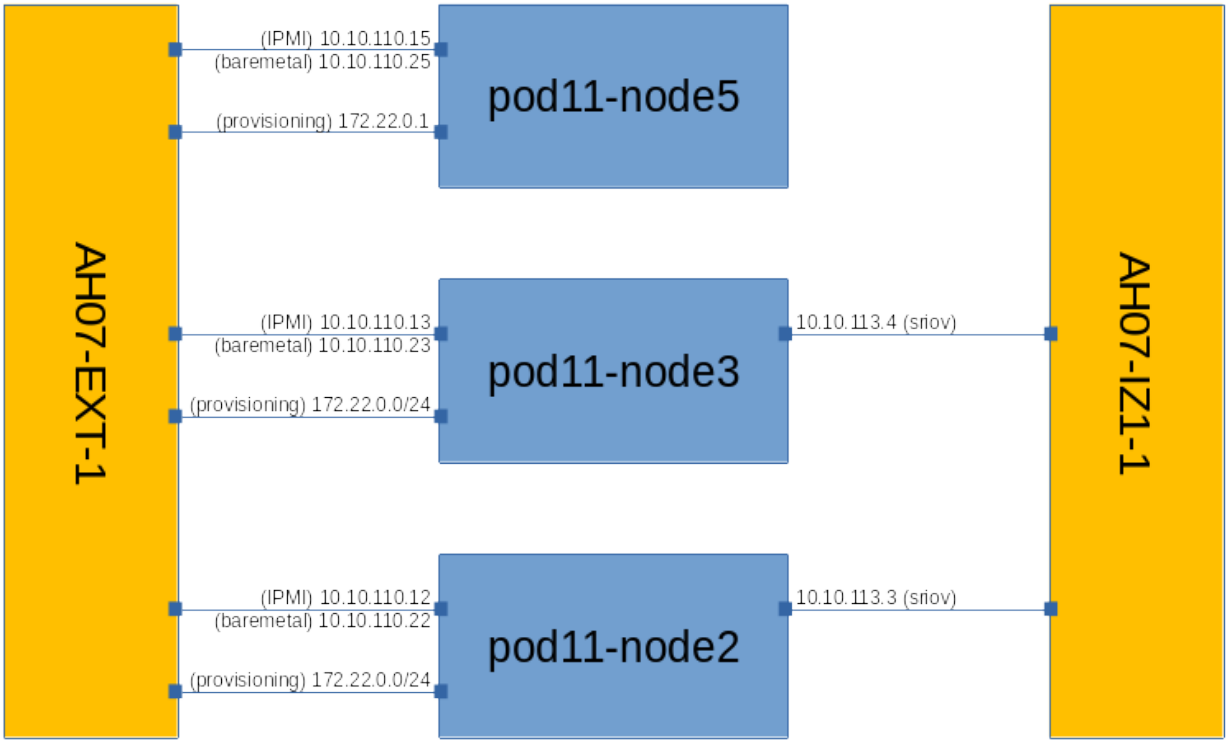
We have the following steps for CD job:

1. On our private Jenkins node, we provision a VM by vagrant. A Vagrantfile which defines the VMs properties is needed. We can define many VM properties in the Vagrantfile:
    - VM hostname
    - VM memory 40G, CPU 32, disk 400GB
2. Login to the VM and run 'make verifier' which installs the components in the VM
3. We destroy the VM as the last step of the job

Test Bed

Pod Topology

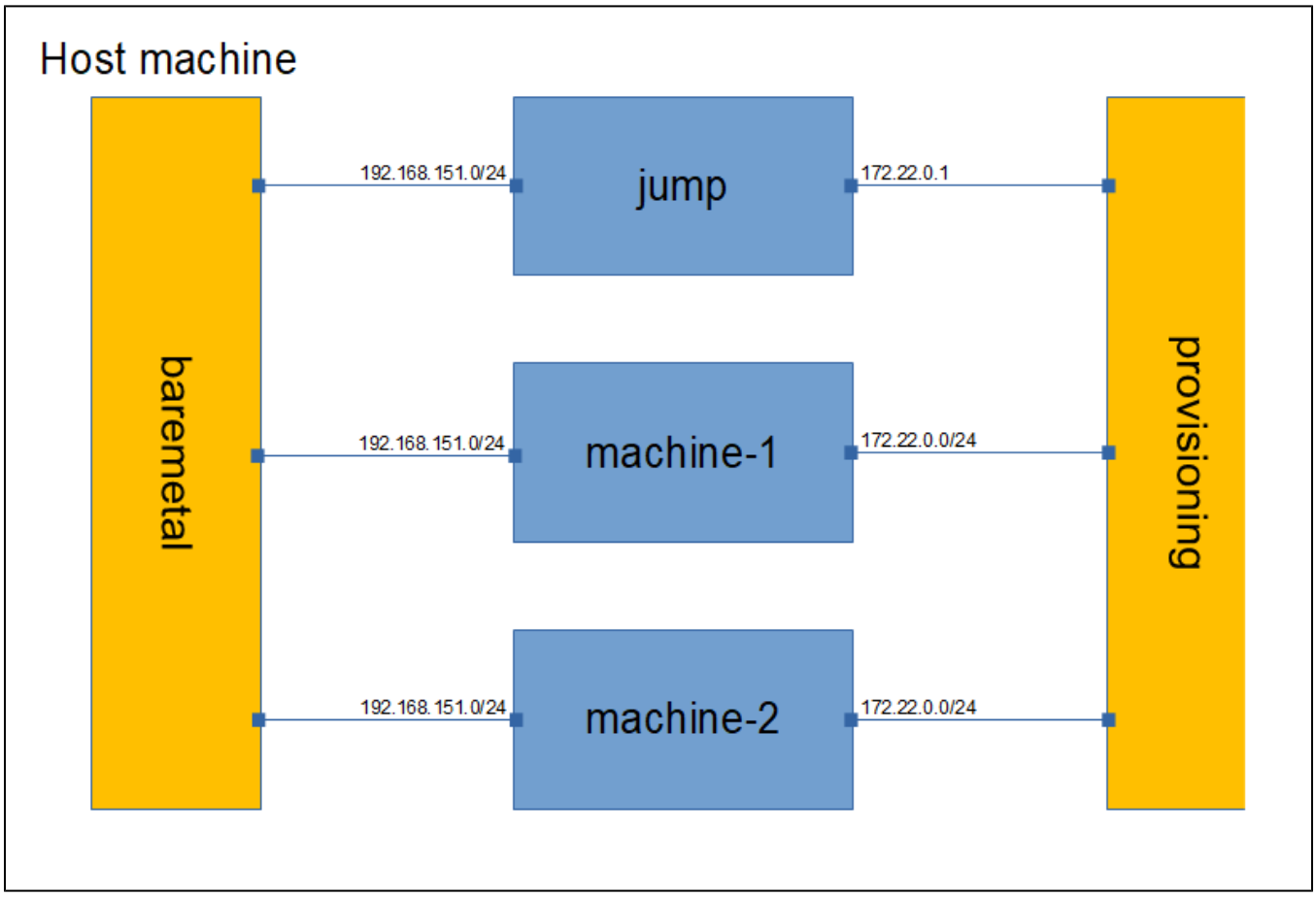
ICN Master Bare Metal Deployment Verifier



(image source: https://gerrit.akraino.org/r/gitweb?p=icn.git;a=blob;f=doc/pod11-topology.png)

- Baremetal network, lab networking for SSH, it is used as the control plane for K8s, used by OVN and Calico for the overlay networking with Internet access
- Provisioning network used by Ironic to do inspection and serve OS images (Will be having DHCP server)
- IPMI LAN to do IPMI protocols for the OS provisioning

ICN Master Virtual Deployment Verifier



(image source: https://gerrit.akraino.org/r/gitweb?p=icn.git;a=blob;f=doc/vm-topology.png)

- Baremetal network is used as control plane for K8s, used by OVN and Calico for overlay network with NAT's Internet access
- Provisioning network used by Ironic to do inspection and server OS images
- Redfish protocol is executed over baremetal network using sushy-emulator

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#
pod11-node5 (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)	
pod11-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)	
pod11-node3	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

Virtual deployment

Hostname	CPU Model	Memory	Storage	1GbE: NIC#, VLAN, (Connected extreme 480 switch)	10GbE: NIC# VLAN, Network (Connected with IZ1 switch)

pod14-node2	Intel 2xE5-2699	64GB	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)

Test Framework

All components are tested with end-to-end testing

Test description

Testing

CI Testing:

bashate:

> The bashate test is to check the shell scripts code style. i.e. trailing white space. We find all files with suffix '.sh' and run bashate against the files.

CD Verifier (end-to-end testing):

All the test case are tested as follows:

Cluster API (infrastructure and bootstrap provisioning):

Verifier will check that servers are provisioned with OS and K8s control plane is ready. The provisioning status is checked every 60 seconds.

Addons:

Test cases verify if the addons are running correctly. Test cases can be found in the ICN deploy/addons, deploy/istio and deploy/kata directories.

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

Nodus:

- Nodus provide 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.
- Nodus networking is setup and created

Nodus Validation and test case results

Tools	Logs
Synk	synk logs data
BDBA	BDBA log data
CheckMarx	Nodus Checkmarx data
Fuzzing tool(Radamsa)	Fuzzing testing shell script Fuzzing test with 100 iterations
Kube-hunter	Kube hunter logs
kube-bench	Kube bench logs

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.
- Create 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 up. Otherwise, the Pod will be in a pending state in case there are no nodes with matching labels that are requested by the Pod

SRIOV

- The SRIOV network device plugin is Kubernetes device plugin for discovering and advertising SRIOV network virtual functions (VFs) in a Kubernetes host.
- We first determine which hosts are SRIOV capable and install the drivers on them and run the DaemonSet and register Network Attachment Definition
- On an SRIOV capable hosts, we can get the resources for the node before we run the pod. When we run the test case, there is a request for a VF from the pod, therefore the number of resources for the node is increased.

QAT

- KUD identify if there are QAT devices in the host and decide whether to deploy QAT device plugin into Kubernetes cluster.
- The QAT device plugin discovers and advertises QAT virtual functions (VFs) to Kubernetes cluster.
- KUD assign 1 QAT VF to the Kernel workloads. After the assignment finished, the allocated resources in node description will increase.

CMK

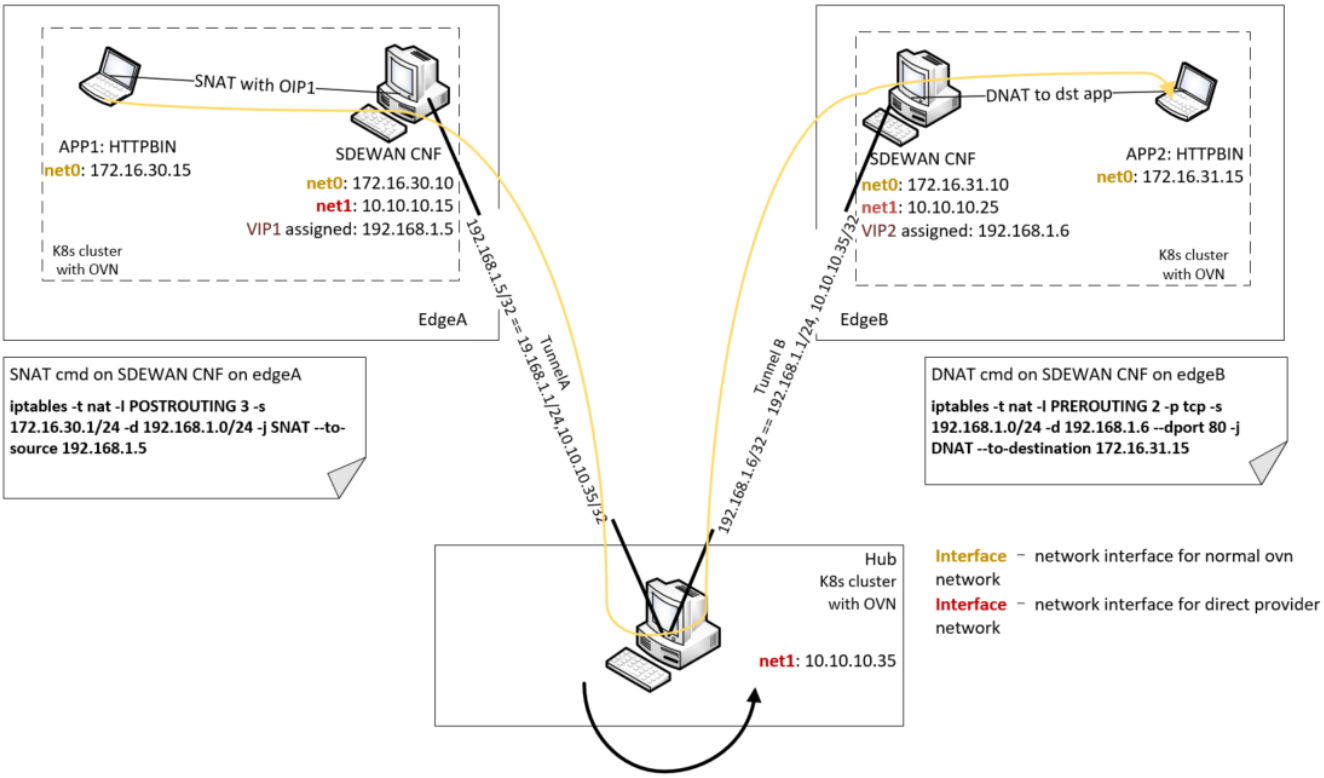
- CPU Manager for Kubernetes provides CPU pinning for K8s workloads. In KUD, there are two test cases for the exclusive and shared CPU pools testing.

EMCO:

- EMCO sanity testing check the health connectivity to EMCO micro services, once it is installed

SDEWAN:

- Use KUD to setup 3 clusters (sdewan-hub, edge-a, edge-b)
- Run the SDEWAN CRD Controller in each clusters.
- Create SDEWAN CNF instance and dummy pod (using httpbin instead) in edge-a, SDEWAN CNF instance and httpbin pod in edge-b
- Create IPSec CR to configure sdewan-hub as responder to provide virtual IP addresses to any authenticated party requesting for IP addresses through SDEWAN CRD Controller.
- Create IPSec CR to configure edge-a and edge-b IPSec configuration to get the IP addresses through SDEWAN CRD Controller.
- Establish edge-a tunnel to sdewan-hub, edge-b tunnel to sdewan-hub, and hub XFRM policies will automatically route traffic between edge-a and edge-b
- Create SNAT CR to establish SNAT rule in edge-a and DNAT CR to establish DNAT rule in edge-b which will enable TCP connection from edge-a to edge-b's httpbin service.
- Verify curl command is successful from edge-a dummy pod (using httpbin instead) to edge-b's httpbin service. The function of the curl command is to return back the ip address of the requester.



Connection test: Returning back the ip address where it is calling from
root@simple-http-service-84b4b4ccc9-6xtxt:/# curl -X GET "http://192.168.1.6/ip" -H "accept: application/json"
{
"origin": "192.168.1.5"
}

BluVal Testing

Release 6 Blueprint Scanning Status

OS Vuls Scan • Pass/Fail • Exceptions	**OS Lynis Scan** • Pass/Fail • Exceptions	**Kube-Hunter Scan** • Pass /Fail • Exceptions
See results here Exceptions requested for the following: • CVE-2021-33574 • CVE-2019-19814 • CVE-2021-35942 Exception requests	See results here Exceptions requested for the following: • BOOT-5122: GRUB boot password interferes with the unattended reboot during OS provisioning. • USB-2000: USB hubs and HID device must be enabled for BMC Console Redirection. • SSH-7408: MaxSessions of 2 prevents lynis from running under Bluval. Lynis, etc. robot files need to be updated to handle a different port. • KRNL-6000: Kernel module loading required by accelerator drivers. Forwarding required by k8s.	See results here Pass

Akraino CVE Vulnerability Exception Request

Akraino BluVal Exception Request

CD logs

ICN Master Bare Metal Deployment Verifier

ICN Master Virtual Deployment Verifier

ICN SDEWAN Master End2End Testing

Test Dashboards

All the testing results are in logs

Additional Testing

not applicable

Bottlenecks/Errata

not applicable