Kubernetes Baremetal deployment setup instructions

1. Hardware Requirements
2. Software Requirements
3. Instructions to run KUD on Baremetal environment
4. Aio.sh explained
5. Enabling Nested-Virtualization
6. Deploying KUD Services
7. Running test cases

Bare-Metal Provisioning

The Kubernetes Deployment, aka KUD, has been designed to be consumed by Virtual Machines as well as Bare-Metal servers. The kud/hosting_providers /baremetal/aio.sh script contains the bash instructions for provisioning an All-in-One Kubernetes deployment on a Bare-Metal server.

This document lists the Hardware & Software requirements and provides a walkthrough to set up all-in-one deployment (a.i.o) using *aio.sh*.

Hardware Requirements

Concept	Amount
CPUs	8
Memory	32GB
Hard Disk	150GB

Software Requirements

- Ubuntu Server 18.04 LTS

Instructions to run KUD on Baremetal environment

Prepare the environment and clone the repo

```
$ sudo apt-get update -y
$ sudo apt-get upgrade -y
$ sudo apt-get install -y python-pip

$ git clone https://git.onap.org/multicloud/k8s/

#Run script to setup KUD

$k8s/kud/hosting_providers/baremetal/aio.sh
```

Aio.sh explained

This bash script provides an automated process for deploying an All-in-One Kubernetes cluster. Given that the ansible inventory file created by this script doesn't specify any information about user and password, it's necessary to execute this script as the root user.

Overall, this script can be summarized in three general phases:

1. Cloning and configuring the KUD project.
2. Enabling Nested-Virtualization.
3. Deploying KUD services.
4. Cloning and configuring the KUD project

KUD requires multiple files(bash scripts and ansible playbooks) to operate. Therefore, it's necessary to clone the *ONAP multicloud/k8s* project to get access to the *vagrant* folder.

Ansible works with multiple systems, the way for selecting them is through the usage of the inventory. The inventory file is a static source for determining the target servers used for the execution of ansible tasks.

The *aio.sh* script creates an inventory file for addressing those tasks to localhost. The inventory file needs to be explicitly updated with the ansible_ssh_host=*with the IP address of the machine or host-IP* along with *ansible_ssh_port*. This is necessary to have some of the test cases run.

Create the host.ini file for Kubespray and Ansible

```
cat <<EOL > inventory/hosts.ini
[all]
Localhost ansible_ssh_host=10.10.110.21 ansible_ssh_port=22
# The ansible_ssh_host IP is an example here. Please update the ansible_ssh_host IP accordingly

[kube-master]
localhost

[kube-node]
localhost

[etcd]
localhost

[ovn-central]
localhost

[ovn-controller]
localhost

[virtlet]
localhost

[k8s-cluster:children]
kube-node
kube-master
EOL
```

KUD consumes kubespray for provisioning a Kubernetes base deployment. As part of the deployment process, this tool downloads and configures *kubectl* binary.

Ansible uses SSH protocol for executing remote instructions. The following instructions create and register ssh keys which avoid the usage of passwords.

Generate ssh-keys

```
$ echo -e "\n\n\n" | ssh-keygen -t rsa -N ""
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod og-wx ~/.ssh/authorized_keys
```

Enabling Nested-Virtualization

KUD installs Virtlet Kubernetes CRI for running Virtual Machine workloads. Nested-virtualization gives the ability to run a Virtual Machine within another. The *node.sh* bash script contains the instructions for enabling Nested-Virtualization.

Enable nested virtualization

```
$sudo ./node.sh
```

Deploying KUD Services

Finally, the KUD provisioning process can be started through the use of the *installer.sh* bash script. The output of this script is collected in the *kud_installer.log* file for future reference.

Bring the cluster up by running the following

```
$ ./installer.sh | tee kud_installer.log
```

Running test cases

The *kud/tests* folder contain the health check scripts that guarantee the proper installation/configuration of Kubernetes add-ons. Some of the examples for test scripts are *virtlet.sh, multus.sh*, *ovn4nfv.sh* etc.