**Install Docker and K8s cluster on on-premises lab VMs**

This document contains three procedures:

**Setup (Section 1):** This procedure needs to be once when we get the VMs from IT.

**Installation (Section 2)** : This procedure describes the process of installing Docker and k8s cluster on support lab VMs.

**Uninstall (Section 3):** This procedure uninstalls k8s and Docker and does necessary cleanup to prepare VMs for next install.

These three procedures are independent and not executed in certain order but when needed.

Typically, the sequence is: When we get the VMs from IT we execute the Setup.

When we provision a cluster for a project lab we run the Install, and when we tear down a lab we execute the uninstall.

At the beginning it will be done by ETAD team, the idea is to train a few support engineers who will lead and execute of this procedure in near future.

All procedures use Ansible scripts that are executed from dub-opnt-supp56.

**Note:** The procedures will be updated once we move to on premises VMs, changes are minimal.

**Note:** All ansible scripts and other supporting data is in Bitbucket:

<https://bitbucket.openet.com/projects/ETAD/repos/ansible_k8s_cluster_install/browse>

**Prerequisite (or pre Setup…):**

1) ETAD will prepare all VMs and Ansible VMs with needed packages, ssh keys and users

2) You will be given a username and login credentials on Ansible VM. This user will have already the needed ssh key set in order for Ansible to ssh as root to cluster VMs.

3) You have provisioned the VMs and have their IP addresses and their role in the cluster (master or worker).

**Note:** Typical cluster will have 5 nodes, but when needed, you may use 6 nodes.

**Section 1: Setup VMs Procedure**

Pre-setup: Done manually:

Login to vm provided by IT, sudo to root issue the following commands:

*mkdir .ssh*

*chmod 700 .ssh*

*vi .ssh/authorized\_keys* //Enter in file vi .ssh/authorized\_keys the key that we use for root to ssh

*chmod 400 .ssh/authorized\_keys*

Create partition sda3 if not created by IT (as root!), See Appendix A for commands session.

Following procedure needs to be run only one time when we get the lab VMs from IT, or when a VM is rebuilt by IT. It sets a few configurations in the VMs including proxy. This procedure will be run by ETAD!

**Note:** Waiting for IT on NTP and DNS.

1) Login to Ansible VM (currently dub-opnt-supp56) as user fworks.

2) cd to Ansible base directory:

*cd ansible*

3) Clone the cluster install repo, this will create a new directory ansible\_k8s\_cluster\_install:

*git clone ssh://git@bitbucket.openet.com:7999/etad/ansible\_k8s\_cluster\_install.git*

Note: this will create a directory ansible\_k8s\_cluster\_install

3a) cd into the cloned repo that contains all Ansible scripts:

*cd ansible\_k8s\_cluster\_install*

3b) Edit the inventory file to include ONLY the VMs you were provided for the cluster.

**Note:** Step 3b is **VERY IMPORTANT**, so you make sure you modify things ONLY on cluster VMs.

4) Disable selinux, firewalld and create several directories we will use:

*ansible-playbook -i inventory pre-setup-1-disable-selinux-firewalld\_and\_misc\_dirs.yaml*

**Note:** This will disable also firewalld

4a) Restart NOW the VM in order to pick the selinux change.

**Note:** Following steps 5 – 14 can be run by a single script: ZZZ

5) **Optional if needed:** Uninstall docker that comes with VM:

*ansible-playbook -i inventory setup-2-uninstall-docker-specific.yaml*

6) **Optional if needed:** Rename kubectl command that comes with VM:

*ansible-playbook -i inventory setup-3-kubectl\_docker\_usr\_local\_bin.yaml*

7) Update /etc/hosts file with all lab VMs:

*ansible-playbook -i inventory setup-4-update\_etc\_hosts\_file.yaml*

8) Replace default certificates, something is wrong with them:

*ansible-playbook -i inventory setup-5-replace\_crt\_and\_ca\_tls\_cert.yaml*

9) Set /etc/bashrc with all proxy stuff needed:

*aansible-playbook -i inventory setup-6-copy\_etc\_environment.yaml*

10) Create user fworks and config some basics for it:

*ansible-playbook -i inventory create-user-fworks.yaml*

**Note:** This script does the following:

Create user fworks

Add appropriate ~/.ssh/authorized\_keys file to fworks

Copy fworks file to /etc/sudoers.d, so user fworks can sudo.

11) Copy config.json to ~/docker for root and fworks users in cluster VMs:

*ansible-playbook -i inventory setup-8-copy-docker-config-to-home.docker.yaml*

12) Disable hugepage usage where voltdb will run (or on all nodes?):

*ansible-playbook -i inventory setup-9-install-disable-hugepage-service.yaml*

**Note:** I was not sure about this one, it is needed for voltdb, but I run it on all...

13) Disable vm over-commit:

*ansible-playbook -i inventory setup-10-set-vm-overcommit.yaml*

14) Install needed Python packages:

*ansible-playbook -i inventory 11-setup-install-pip-stuff.yaml*

15) Install ntp

*ansible-playbook -i inventory setup-12-install\_and\_enable\_ntp.yaml*

**Section 2: Install Docker, Kubernetes and create a K8S cluster**

**Note:** This install procedure assumes that all VMs in cluster are ready for installation (all setup listed in section 1 and cleanup needed were done).

**Note:** Steps 1 -3a (cloning the repo) are optional, execute as needed.

1) Login to Ansible VM ( dub-opnt-supp56 ) as user provided by ETAD.

2) cd to Ansible base directory:

*cd ansible*

3) Clone the cluster install repo, this will create a new directory ansible\_k8s\_cluster\_install:

*git clone ssh://git@bitbucket.openet.com:7999/etad/ansible\_k8s\_cluster\_install.git*

Note: this will create a directory ansible\_k8s\_cluster\_install

3a) cd into the cloned repo that contains all Ansible scripts:

*cd ansible\_k8s\_cluster\_install*

4) Edit the file proj\_vars and modify the following to match the versions you need:

docker\_version

kubernetes\_version

cni\_file\_version

Optionally you may want to modify also net\_pod\_cidr

5) Edit the inventory file and set the correct IP addresses for master and workers. app2 – app6 (or app2 – app5). The IPs of app2 – app5 are just semantic, you do not care which VM is whatever app.

6) Execute ansible script to install docker and k8s on all nodes:

*ansible-playbook -i inventory install-docker\_and-k8s-base-all-nodes.yaml*

**Note:** kubelet shows failed at this step

7) Init the k8s cluster:

*ansible-playbook -i inventory run-kubeadm-init-on-master-nonet.yaml*

**Notes:**

Before running this, verify in vars file versions and pods CIDR are what you want

No cni applied here!

Wait until you see that all pods in kube-system run (except coredns pods, that will be ready after cni install)

This creates also the /tmp file with kubeadm join cmmand.

8) Apply cni plugin only on master - after all pods except coredns are up!!!

*ansible-playbook -i inventory apply-cni-on-masetr.yaml*

9) Verify install was successful, **login to the master** and execute the following two commands:  
*kubectl get nodes*

You should see a response similar to the following:

NAME STATUS ROLES AGE VERSION

ip-100-66-23-38.eu-west-1.compute.internal Ready control-plane,master 2d4h v1.20.4

*kubectl get pods -n kubesystem*

You should see all pods in status Running and showing READY.

10) un-taint the master to enable running pods on it:

**Login to the master** and execute the following command on it:

kubectl taint node MASTER-NODE-NAME node-role.kubernetes.io/master:NoSchedule-

Note: MASTER-NODE-NAME is the name of the master node that you got on step 10 with command kubectl get nodes.

**Note:** We will add an Ansible script for this step 11…

11) Add remaining nodes to the cluster. Following command needs to be run for each “app” node (that you setup in step 6). Run the following command for all apps (app2 .. app5|6), and wait and verify node was added to the cluster before running it for next app.

*ansible-playbook -i inventory --limit 'app2' run-kubeadm-join-on-workers.yaml*

**Note:** Verify node was added and also supported pods with following commands:

*kubectl get nodes*

*kubectl get pods -n kube-system*

10) Install ceph-rook:

Based on: https://rook.io/docs/rook/v1.8/quickstart.html

Note: For now done manually on master, need to automate with ansible.

**Note:** You can install both rook-ceph and openEBS (steps 10 & 11), but typically we will use only rook-ceph.

10.1) clone/download rook-ceph, note that I use here v 1.8.10, it is not too new and not too old

and works fine with Optus project:

*git clone --single-branch --branch v1.8.10 https://github.com/rook/rook.git*

10.2) Deploy basic components of rook-ceph:

*cd rook/deploy/examples*

*kubectl create -f crds.yaml -f common.yaml -f operator.yaml*

Note: Before you contiune make sure operator pod is fully up in ns rook-ceph!

10.3) Optional: Modify the cluster.yaml to fit Support env.

Note: You can use for this step the deafult file (cluster.yaml), I prefer to

create a custom cluster-etad.yaml file. In the modified cluster-etad.yaml

I simply help rook, by guiding which device to use: deviceFilter: sda3.

Following shows the difference between cluster-etad.yaml and cluster.yaml:

*diff cluster-etad.yaml cluster.yaml*

218c218

< deviceFilter: sda3

---

> #deviceFilter:

10.4) Run the rook-ceph cluster:

*kubectl create -f cluster-etad.yaml*

Note: This step can take a few minutes, be patient and check with the following

that cluster is up:

*kubectl -n rook-ceph get pod*

Example of output for a 6 nodes cluster:

*kubectl -n rook-ceph get pod*

NAME READY STATUS RESTARTS AGE

csi-cephfsplugin-4b8nz 3/3 Running 0 14m

csi-cephfsplugin-84x42 3/3 Running 0 14m

csi-cephfsplugin-dz785 3/3 Running 0 14m

csi-cephfsplugin-hp278 3/3 Running 0 14m

csi-cephfsplugin-k9qtq 3/3 Running 0 14m

csi-cephfsplugin-mj6dt 3/3 Running 0 14m

csi-cephfsplugin-provisioner-6d4bd9b669-7f8lw 6/6 Running 0 14m

csi-cephfsplugin-provisioner-6d4bd9b669-fhsjw 6/6 Running 0 14m

csi-rbdplugin-88zlx 3/3 Running 0 14m

csi-rbdplugin-d4mhf 3/3 Running 0 14m

csi-rbdplugin-mnsrg 3/3 Running 0 14m

csi-rbdplugin-pd5q7 3/3 Running 0 14m

csi-rbdplugin-provisioner-6bcd78bb89-5vj97 6/6 Running 0 14m

csi-rbdplugin-provisioner-6bcd78bb89-8dtt7 6/6 Running 0 14m

csi-rbdplugin-tc6lr 3/3 Running 0 14m

csi-rbdplugin-xgqwd 3/3 Running 0 14m

rook-ceph-crashcollector-dub-opnt-supp12-66c5d6fcb4-rwm47 1/1 Running 0 9m37s

rook-ceph-crashcollector-dub-opnt-supp13-67fc4df7bc-kqfgf 1/1 Running 0 11m

rook-ceph-crashcollector-dub-opnt-supp14-5f459c68cd-cx96h 1/1 Running 0 10m

rook-ceph-crashcollector-dub-opnt-supp15-54884d8c85-x9s2z 1/1 Running 0 13m

rook-ceph-crashcollector-dub-opnt-supp16-6f8bfc754d-lp6hf 1/1 Running 0 9m46s

rook-ceph-crashcollector-dub-opnt-supp20-69cc56cd77-tpxb5 1/1 Running 0 13m

rook-ceph-mgr-a-5984dbc97c-kbnd2 1/1 Running 0 13m

rook-ceph-mon-a-6fd84b9b96-tdcr7 1/1 Running 0 15m

rook-ceph-mon-b-cc4668976-5ft49 1/1 Running 0 13m

rook-ceph-mon-c-6dcb4d468d-jkjcc 1/1 Running 0 13m

rook-ceph-operator-57c548c785-dhm5h 1/1 Running 0 19m

rook-ceph-osd-0-b9c578c5f-qb849 1/1 Running 0 11m

rook-ceph-osd-1-86f5444dbb-m744c 1/1 Running 0 11m

rook-ceph-osd-2-65459cf7cf-2btxn 1/1 Running 0 11m

rook-ceph-osd-3-8589c5c545-4lqhp 1/1 Running 0 10m

rook-ceph-osd-4-6856c79fb6-krjmv 1/1 Running 0 9m46s

rook-ceph-osd-5-867d995d6f-wnsbf 1/1 Running 0 9m37s

rook-ceph-osd-prepare-dub-opnt-supp12-gxbfs 0/1 Completed 0 12m

rook-ceph-osd-prepare-dub-opnt-supp13-d69sn 0/1 Completed 0 12m

rook-ceph-osd-prepare-dub-opnt-supp14-6zvct 0/1 Completed 0 12m

rook-ceph-osd-prepare-dub-opnt-supp15-7f87b 0/1 Completed 0 12m

rook-ceph-osd-prepare-dub-opnt-supp16-hhrc6 0/1 Completed 0 12m

rook-ceph-osd-prepare-dub-opnt-supp20-prxsb 0/1 Completed 0 12m

10.5) Optional (recommended): Start the rook-ceph toolbox pod, so you can troubleshoot

rook-ceph issues if/when needed:

*kubectl create -f toolbox.yaml*

**Note:** See https://rook.io/docs/rook/v1.8/ceph-toolbox.html on rook-ceph toolbox and commands it

uses.

Check status of rook-ceph (Replace name of toolbox pod with it name in your cluster...):

*k -n $rookceph exec -it rook-ceph-tools-7cd79f6fbf-kmf8n -- ceph status*

10.6) Create a storageclass:

*kubectl create -f csi/rbd/storageclass.yaml*

Note: Check that sc was created:

k get sc

NAME PROVISIONER RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE

rook-ceph-block rook-ceph.rbd.csi.ceph.com Delete Immediate true 8s

10.7) Make rook-ceph-block the default SC (it is a one-line command!):

*kubectl patch storageclass rook-ceph-block -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'*

Ready here to install project!

11) Install helm openEBS storage provider

**Note**: This step is optional as typically we work with rook-ceph.

11.1) Install openEBS helm chart

*ansible-playbook -i inventory install-openebs-storage-helm-chart.yaml*

Note: This also creates two storage classes: openebs-device and openebs-hostpath, we use the openebs-hostpath.

11.2) Make openEBS default SC:

*kubectl patch storageclass openebs-hostpath -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'*

Ready here to install project!

**Section 3: Uninstall k8s cluster and Docker (including cleanup):**

**Note:** Steps 1 – 3a (cloning the repo) are optional, execute as needed.

0) Uninstall project helm charts, for this you would usually have a script in project home in master node, typically project is under /home/fworks/lab.

*./uninstall\_optus5g\_lab.sh*

Note: Projects can have different names for uninstall, like *uninstallAllApps.bash* or uninstallproject.bash, we need to standardize this!

1) Login to Ansible VM (currently dub-opnt-supp56) as user provided by ETAD.

2) cd to Ansible base directory:

*cd ansible*

3) Clone the cluster install repo, this will create a new directory ansible\_k8s\_cluster\_install:

*git clone ssh://git@bitbucket.openet.com:7999/etad/ansible\_k8s\_cluster\_install.git*

Note: this will create a directory ansible\_k8s\_cluster\_install

3a) cd into the cloned repo that contains all Ansible scripts:

*cd ansible\_k8s\_cluster\_install*

4) Make sure no projects apps are running on cluster, this is typically done on master on the project directory with a script like *uninstallAllApps.bash* or uninstallproject.bash.

5) Uninstall k8s and Docker on master (this runs also kubeadm reset)

*ansible-playbook -i inventory uninstall-docker-and-k8s-master.yaml*

6) Uninstall k8s and docker on all workers:

*ansible-playbook -i inventory uninstall-docker-and-k8s-workers.yaml*

7) Flush IP tables on all cluster nodes:

*ansible-playbook -i inventory flush\_iptables\_all\_nodes.yaml*

8) Delete links and bridges created during Docker and k8s install:

*ansible-playbook -i inventory remove\_bridges\_and\_links.onprem.yaml*

9) Clean rook-ceph dir (/var/lib/rook) and device it uses (/dev/sda3):

*ansible-playbook -i inventory delete-rookceph-dirs-and-clean-sda3.yaml*

**Note: Be very careful** on this step. In On-premise support lab we use /dev/sda3 for rook.

If in the future this is changed then this ansible script and the bash script it uses need

to be changed!!

9) Optional: Restart the VMs here.

**Appendix A**

*fdisk /dev/sda* //Run fdisk on /dev/sda

Welcome to fdisk (util-linux 2.23.2).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Command (m for help): *p* //Print current partitions table

Disk /dev/sda: 268.4 GB, 268435456000 bytes, 524288000 sectors

Units = sectors of 1 \* 512 = 512 bytes

Sector size (logical/physical): 512 bytes / 512 bytes

I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk label type: dos

Disk identifier: 0x000be79a

Device Boot Start End Blocks Id System

/dev/sda1 \* 2048 411647 204800 83 Linux

/dev/sda2 411648 335972351 167780352 8e Linux LVM

Command (m for help): *n* //Create new partition

Partition type:

p primary (2 primary, 0 extended, 2 free)

e extended

Select (default p): *p* //New partition is of typePrimary

Partition number (3,4, default 3):

First sector (335972352-524287999, default 335972352):

Using default value 335972352

Last sector, +sectors or +size{K,M,G} (335972352-524287999, default 524287999):

Using default value 524287999

Partition 3 of type Linux and of size 89.8 GiB is set

Command (m for help): *w* //Write new partition table

The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.

The kernel still uses the old table. The new table will be used at

the next reboot or after you run partprobe(8) or kpartx(8)

Syncing disks.

**Note:** This will take effect only after reboot!