**To start: this was done on 3 vms running CentOS (CentOS-7-x86\_64-Minimal-2009) using Vmware. It can be one VM and then cloned to as many as you want. For me I cloned it twice as I am going to work on 1 master node and 2 worker nodes.**

**A screenshot of a computer

Description automatically generated**

**Now for the internet connection it was set to NAT.**

**Note for ease of life and speed: as we are working without GUI I used remote SSH tool MobaXterm\_Portable as I can copy paste commands functionality in it and have some features that make it easier.**

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**In this case since I cave to run a lot of commands in all the nudes, so I used the MultiExec option to run the same command in all the nodes in the same time:**

**A picture containing text, screenshot, symbol, font

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**To do this with ease will need to edit the the keyboard shortcuts settings:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**The paste to terminal I edit it to Ctrl + V**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Also we are using Kubernetes Cluster Setup.**

**Now lets start:**

**For our project we will need:**

sudo yum install python3 ##as centos have python 2 defualt. We used (python 3.6)

sudo yum install pip3 ## we need it to install the container dependencies.

Sudo yum install nano

**Optional: to change the host name to know which is which like who is master and who is worker and map the ip and the host name for each one**

$ hostnamectl set-hostname new\_hostname

$ sudo nano /etc/hosts

192.168.116.156 master

192.168.116.155 worker1

192.168.116.154 worker2

192.168.116.161 worker3

**Step 1: Pre-requisites**

1.a.. Check the OS, Hardware Configurations & Network connectivity

1.b.. Turn off the swap & firewall

$ sudo swapoff -a

$ sudo nano /etc/fstab

Will need comment out the swap by add “#” at the start of swap line



$ sudo systemctl stop firewalld

$ sudo systemctl disable firewalld

Step 2. Configure the local IP tables to see the Bridged Traffic

2.a.. Enable the bridged traffic

$ lsmod | grep br\_netfilter

$ sudo modprobe br\_netfilter

$ lsmod | grep br\_netfilter

2.b.. Copy the below contents in this file.. /etc/modules-load.d/k8s.conf

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

br\_netfilter

EOF

2.c.. Copy the below contents in this file.. /etc/sysctl.d/k8s.conf

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-iptables = 1

net.bridge.bridge-nf-call-ip6tables = 1

EOF

$ sudo sysctl --system

**Step 3. Install Docker as a Container RUNTIME**

3.a.. Uninstall any Older versions

$ sudo yum remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine

**3.b.. Install Yum Utilities | Config Manager**

$ sudo yum install -y yum-utils

**3.c.. Setup the Docker Repository**

$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

**3.d.. Install Docker Engine, Docker CLI, Docker RUNTIME**

$ sudo yum install -y docker-ce docker-ce-cli containerd.io

**Step 4. Configure Docker Daemon for cgroups management & Start Docker**

**4.a.. Create directory**

$ sudo mkdir /etc/docker

**4.b.. Copy the below contents in this file.. /etc/docker/daemon.json**

cat <<EOF | sudo tee /etc/docker/daemon.json

{

"exec-opts": ["native.cgroupdriver=systemd"],

"log-driver": "json-file",

"log-opts": {

"max-size": "100m"

},

"storage-driver": "overlay2"

}

EOF

$ sudo systemctl daemon-reload

$ sudo systemctl restart docker

$ sudo systemctl enable docker

$ sudo systemctl status docker

**Step 5. Install kubeadm, kubectl, kubelet**

**5.a.. Copy the below contents in this file.. /etc/yum.repos.d/kubernetes.repo**

cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kubelet kubeadm kubectl

EOF

**5.b.. Set SELinux in permissive mode (effectively disabling it)**

$ sudo setenforce 0

$ sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

$ sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

$ sudo systemctl enable --now kubelet

**####nomrally if u can take a snapshot of all the nodes here###**

**Step 6. Configuring a cgroup driver**

**Ignore if docker is used as a CRI**

**Step 7. Check containerd status for all the nodes:**

**$ systemctl status containerd**

**\*\*if u face an error like this or even if it you normally just do it but not the If needed part :**

**DO this to fix:**

```

The link: <https://forum.linuxfoundation.org/discussion/862825/kubeadm-init-error-cri-v1-runtime-api-is-not-implemented>:

**####this done for all the nodes###**

Remove the installed default config file: **rm /etc/containerd/config.toml**

Restart containerd:$ **systemctl restart containerd**

Check status **:$ systemctl status containerd**

Now have to reboot: **$ reboot**

check the status again **$ systemctl status containerd**

**If you get any error with the status like this:**

**A screen shot of a computer screen

Description automatically generated with low confidence**

**Then have to do these commands in this order:**

Remove the old containerd:$ **sudo yum remove containerd**

Update repository data and install the new containerd**:$ sudo yum install containerd.io**

Remove the installed default config file: **rm /etc/containerd/config.toml**

Restart containerd:$ **systemctl restart containerd**

Check status **:$ systemctl status containerd**

Now have to reboot: **$ reboot**

check the status again **$ systemctl status containerd**

**Step 7. Deploy a kubernetes cluster using kubeadm**

**# Run only in Master node**

$ sudo su –

**Note: Will need to but the actual master vm ip address for the below command and copy the output:**

$ kubeadm init --pod-network-cidr=10.10.0.0/16 --apiserver-advertise-address=**192.168.116.156**

**Copy the outpout:**

A screen shot of a computer program

Description automatically generated with low confidence

**To start using your cluster, you need to run the following as a regular user:**

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

**then if you are the root user, you can run:**

$ export KUBECONFIG=/etc/kubernetes/admin.conf

**Then you can join any number of worker nodes by running the following on each as root:**

**###dont join now do after step 8###**

kubeadm join 192.168.116.156:6443 --token orp862.qlvw15p948vjv7p9 \

--discovery-token-ca-cert-hash sha256:9ce7949ae76d98e45950c40ba22188f691ff60e826dbed0b869b80c67856c2aa

**Step 8. Install CNI for POD Networking**

**Now we can check the nodes:**

kubectl get nodes

A black background with white text

Description automatically generated with low confidence

**# Run only in Master node**

**Weave Networks:**

$ kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

**###not needed as we copied and it didn’t work###**

**If u didn’t copy the kubeadm join command u can use this to view it:**

$ sudo kubeadm token list

$ sudo kubeadm token create gdk7g9.3a9sa55hpcriyqhk --print-join-command --ttl 0

**Step 9. Join the worker nodes to the master**

**###if u didn’t store the commad to join you can get it by running on the master node:**

**kubeadm token create --print-join-command**

# Run in Worker Nodes as "Root"

sudo kubeadm join 192.168.116.156:6443 --token orp862.qlvw15p948vjv7p9 \

--discovery-token-ca-cert-hash sha256:9ce7949ae76d98e45950c40ba22188f691ff60e826dbed0b869b80c67856c2aa

A screen shot of a computer code

Description automatically generated with low confidence

**Now that nodes are ready:**

**Step 10. Access the K8s Cluster & Deploy a POD to test**

$ kubectl run vsparkz --image nginx

A screen shot of a computer code

Description automatically generated with low confidence

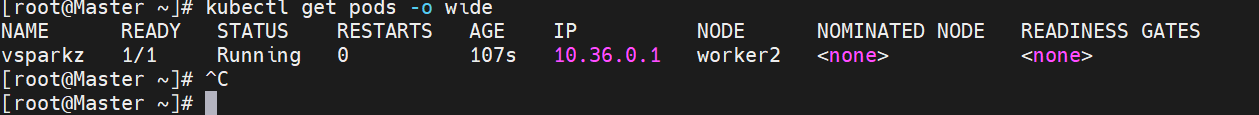
Now wer check the pods:

$ kubectl get pods

A screen shot of a computer

Description automatically generated with low confidence

$ kubectl get pods -o wide



**Some further commands:**

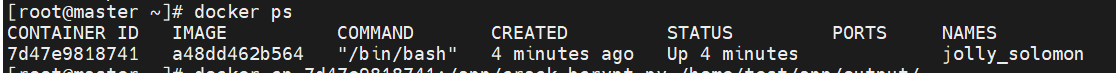
Need to run the **image** container (need the pull the image 1s or if its already pulled):

$ **docker run -it a48dd462b564 /bin/**bash #access the terminal of the running container with bash interface can be other things.

copy shit from running container:

**docker cp <container\_id\_or\_name>:<path\_to\_file\_inside\_container> <path\_on\_local\_file\_system>**

example: need the container id not the image id . need to run it the do $ **docker ps**



**docker cp 7d47e9818741:/app/crack\_bcrypt.py /home/test/app/output/**

**build docker image container:**

docker build -t test:56 .

**tags docker image container:**

docker tag test:56 mahmoodmattar/test:56

**push docker image container:**

docker push mahmoodmattar/test:56

**deploy yaml**

kubectl apply -f bycrpt.yaml

**to check all pods running:**

kubectl get pods -o wide

kubectl get pods --all-namespaces

**check a pod logs:**

kubectl logs <pod name>

kubectl describe pod <my-pod>

**to kill deployment :**

kubectl delete deployment cracking-deployment

**to delete pods:**

kubectl delete pod –all

**dealing with jobs:**

**apply jobs:**

kubectl apply -f 1job\_2chunk.yaml

**checking logs:**

kubectl logs -l job-name=cracking-job

**terminate:**

kubectl delete job cracking-job