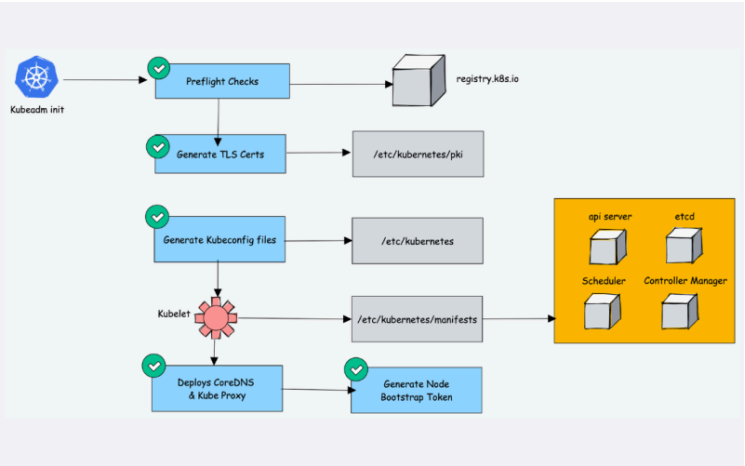
Install k8s cluster with vagrant and kubeadm

Introduction:

Kubeadm is a tool to set up a minimum viable Kubernetes cluster without much complex configuration. Also, Kubeadm makes the whole process easy by running a series of pre checks to ensure that the server has all the essential components and configs to run Kubernetes.



- Kubeadm Setup Prerequisites:

Following are the prerequisites for Kubeadm Kubernetes cluster setup.

- Minimum two **Ubuntu nodes** [One master and one worker node]. You can have more worker nodes as per your requirement.
- The master node should have a minimum of **2 vCPU and 2GB RAM**.
- For the worker nodes, a minimum of 1vCPU and 2 GB RAM is recommended.
- **10.X.X.X/X** network range with static IPs for master and worker nodes. We will be using the **192.x.x.x** series as the pod network range that will be used by the Calico network plugin. Make sure the Node IP range and pod IP range don't overlap.

Please refer to the following image and make sure all the ports are allowed for the control plane (master) and the worker nodes.

Control-plane node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services**	All

Since we are using vagrant-based Ubuntu VMs, the firewall will be disabled by default. So you don't have to do any firewall configurations.

Below are the high-level steps involved in setting up a kubeadm-based Kubernetes cluster.

1. Install container runtime on all nodes- We will be using cri-o.
2. Install Kubeadm, Kubelet, and kubectl on all the nodes.
3. Initiate Kubeadm control plane configuration on the master node.
4. Save the node join command with the token.
5. Install the Calico network plugin (operator).
6. Join the worker node to the master node (control plane) using the join command.
7. Validate all cluster components and nodes.
8. Install Kubernetes Metrics Server
9. Deploy a sample app and validate the app

All the steps given in this guide are referred from the official Kubernetes documentation.

1- Provision the VMs using Vagrant:

We will use Vagrant to set up the Kubernetes cluster, you will use of the Vagrantfile present in the training Github repo here: https://github.com/ghazelatech/cka-preparation/blob/main/labs/6-%20Cluster%20install/Vagrantfile

It launches 3 VMs, 1 Control plan and kubernetes workers.

2- Enable iptables Bridged Traffic on all the Nodes

Execute the following commands on **all the nodes** for IPtables to see bridged traffic. Here we are tweaking some kernel parameters and setting them using *sysctl*

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

sudo modprobe overlay
sudo modprobe br_netfilter

# sysctl params required by setup, params persist across reboots
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-iptables  = 1
net.bridge.bridge-nf-call-ip6tables = 1
net.ipv4.ip_forward                 = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

3- Disable swap on all the NodesDisable swap on all the Nodes

For kubeadm to work properly, you need to disable swap on all the nodes using the following command.

```
sudo swapoff -a
(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true
```

The fstab entry will make sure the swap is off on system reboots.
You can also control swap errors using the kubeadm parameter `--ignore-preflight-errors` Swap we will look at it in the latter part.

4- Install CRI-O Runtime On All The Nodes

We will be using CRI-O instead of Docker for this setup as Kubernetes deprecated the Docker engine.
Execute the following commands **on all the nodes** to install required dependencies and the latest version of CRIO.

```
sudo apt-get update -y
sudo apt-get install -y software-properties-common curl apt-transport-https
ca-certificates

curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key
|
    gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg
```

```
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg]
https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" |
    tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y
sudo apt-get install -y cri-o

sudo systemctl daemon-reload
sudo systemctl enable crio --now
sudo systemctl start crio.service
```

Install crictl:

```
VERSION="v1.28.0"
wget
https://github.com/kubernetes-sigs/cri-tools/releases/download/$VERSION/crictl-
$VERSION-linux-amd64.tar.gz
sudo tar zxvf crictl-$VERSION-linux-amd64.tar.gz -C /usr/local/bin
rm -f crictl-$VERSION-linux-amd64.tar.gz
```

crictl, a CLI utility to interact with the containers created by the container runtime.

5- Install Kubeadm & Kubelet & Kubectl on all Nodes

Download the GPG key for the Kubernetes APT repository on all the nodes.

```
KUBERNETES_VERSION=1.29

sudo mkdir -p /etc/apt/keyrings
curl -fsSL
https://pkgs.k8s.io/core:/stable:/v$KUBERNETES_VERSION/deb/Release.key | sudo
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v$KUBERNETES_VERSION/deb/ /" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
```

Update apt repo: `sudo apt-get update -y`

You can use the following commands to find the latest versions. Install the first version in 1.29 so that you can practice cluster upgrade task:

```
apt-cache madison kubeadm | tac
```

Specify the version as shown below. Here I am using 1.29.0-1.1

```
sudo apt-get install -y kubelet=1.29.0-1.1 kubectl=1.29.0-1.1
kubeadm=1.29.0-1.1
```

Add hold to the packages to prevent upgrades.

```
sudo apt-mark hold kubelet kubeadm kubectl
```

Now we have all the required utilities and tools for configuring Kubernetes components using kubeadm.
Add the node IP to **KUBELET_EXTRA_ARGS**.

```
sudo apt-get install -y jq
local_ip="$(ip --json addr show eth0 | jq -r '.[0].addr_info[] |
select(.family == "inet") | .local')"
cat > /etc/default/kubelet << EOF
KUBELET_EXTRA_ARGS=--node-ip=$local_ip
EOF
```

6- Initialize Kubeadm On Master Node To Setup Control Plane

Execute the commands in this section only on the master node.

We are using a Private IP for the master Node, Set the following environment variables. Replace 10.0.0.10 with the IP of your master node.

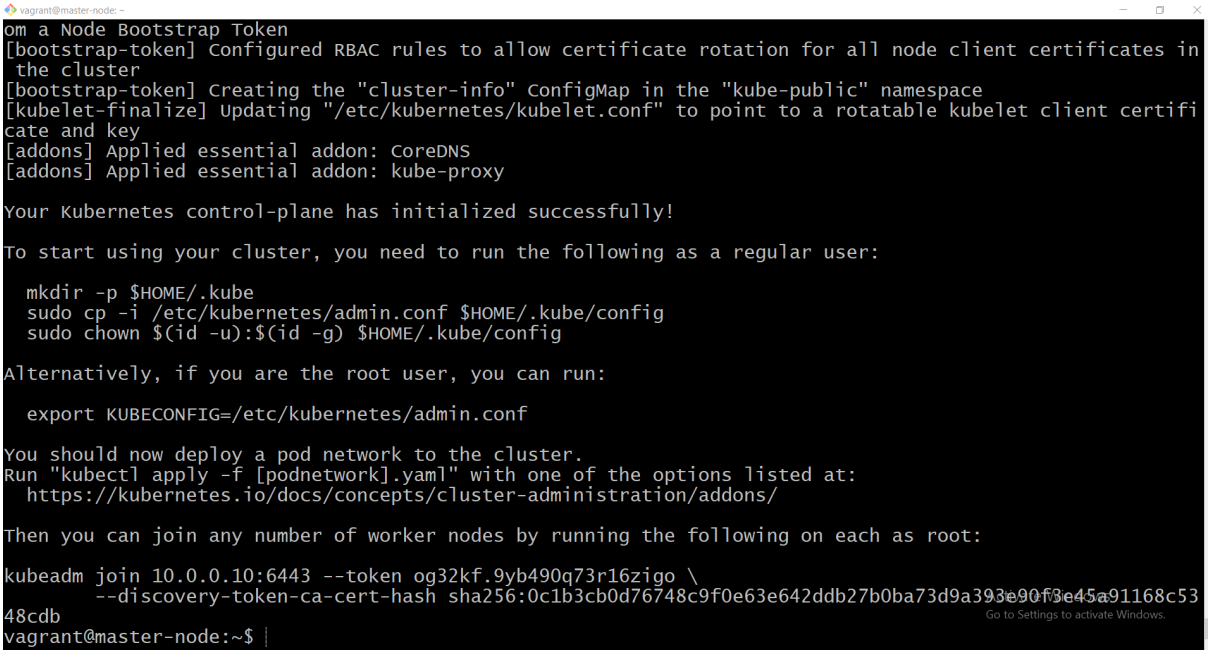
```
IPADDR="10.0.0.10"
NODENAME=$(hostname -s)
POD_CIDR="192.168.0.0/16"
```

Now, initialize the master node control plane configurations using the kubeadm command.
For a Private IP address-based setup use the following init command.

```
sudo kubeadm init --apiserver-advertise-address=$IPADDR
--apiserver-cert-extra-sans=$IPADDR  --pod-network-cidr=$POD_CIDR --node-name
$NODENAME --ignore-preflight-errors Swap
```

`--ignore-preflight-errors Swap` is actually not required as we disabled the swap initially.
On a successful kubeadm initialization, you should get an output with kubeconfig file location and the join command with the token as shown below. Copy that and save it to the file. we will need it for joining the worker node to the master.

On a successful kubeadm initialization, you should get an output with kubeconfig file location and the **join command with the token** as shown below. Copy that and save it to the file. we will need it for **joining the worker node to the master**.



Use the following commands from the output to create the kubeconfig in master so that you can use kubectl to interact with cluster API

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Now, verify the kubeconfig by executing the following kubectl command to list all the pods in the kube-system namespace.

```
kubectl get po -n kube-system
```

You should see the following output. You will see the **two Coredns pods** in a **pending** state. It is the expected behavior. Once we install the network plugin, it will be in a running state.

You verify all the cluster component health statuses using the following command.

```
kubectl get --raw='/readyz?verbose'
```

You can get the cluster info using the following command.

```
kubectl cluster-info
```

By default, apps won't get scheduled on the master node. If you want to use the master node for scheduling apps, taint the master node.

```
kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

7- Join Worker Nodes To Kubernetes Master Node

We have set up cri-o, kubelet, and kubeadm utilities on the worker nodes as well.

Now, let's join the worker node to the master node using the Kubeadm join command you have got in the output while setting up the master node.

If you missed copying the join command, execute the following command in the master node to recreate the token with the join command.

```
kubeadm token create --print-join-command
```

Here is what the command looks like. Use sudo if you are running as a normal user. This command performs the TLS bootstrapping for the nodes.

```
sudo kubeadm join 10.128.0.37:6443 --token j4eice.33vgvgyf5cxw4u8i \
    --discovery-token-ca-cert-hash
sha256:37f94469b58bcc8f26a4aa44441fb17196a585b37288f85e22475b00c36f1c61
```

On successful execution, you will see the output saying, "This node has joined the cluster".

```
root@worker-node01:~# kubeadm join 10.0.0.10:6443 --token og32kf.9yb490q73r16zigo \
        --discovery-token-ca-cert-hash sha256:0c1b3cb0d76748c9f0e63e642ddb27b0ba73d9a393e90f3e45a91168c53
48cdb
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o y
aml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Now execute the kubectl command from the master node to check if the node is added to the master.

```
kubectl get nodes
```

```
vagrant@master-node:~$ kubectl get po -n kube-system
NAME                                    READY   STATUS    RESTARTS   AGE
calico-kube-controllers-658d97c59c-b4q9j   1/1     Running   0          65s
calico-node-pj2n9                       1/1     Running   0          65s
calico-node-vhddk                       1/1     Running   0          65s
coredns-76f75df574-2dp8h                1/1     Running   0          12m
coredns-76f75df574-plqmz                1/1     Running   0          12m
etcd-master-node                        1/1     Running   0          13m
kube-apiserver-master-node              1/1     Running   0          13m
kube-controller-manager-master-node     1/1     Running   0          13m
kube-proxy-tcn2x                        1/1     Running   0          12m
kube-proxy-zlf76                        1/1     Running   0          2m56s
kube-scheduler-master-node              1/1     Running   0          13m
vagrant@master-node:~$ kubectl get nodes
NAME           STATUS   ROLES           AGE     VERSION
master-node    Ready    control-plane   13m     v1.29.0
worker-node01  Ready    <none>          2m58s   v1.29.5
```

In the above command, the ROLE is <none> for the worker nodes. You can add a label to the worker node using the following command. Replace worker-node01 with the hostname of the worker node you want to label.

```
kubectl label node node01  node-role.kubernetes.io/worker=worker
```

You can further add more nodes with the same join command.

8- Install Calico Network Plugin for Pod Networking

Kubeadm does not configure any network plugin. You need to install a network plugin of your choice for *kubernetes pod* networking and enable network policy.
We'll use the Calico network plugin for this setup.

on the master node (or where you have configured the *kubeconfig* file.), Execute the following commands to install the Calico network plugin operator on the cluster.

Execute the following commands to install the **Calico network plugin** operator on the cluster.

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

After a couple of minutes, if you check the pods in kube-system namespace, you will see calico pods and running CoreDNS pods.

```
kubectl get po -n kube-system
```

```
vagrant@master-node:~$ kubectl get po -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-658d97c59c-b4q9j   1/1     Running   0          65s
calico-node-pj2n9                          1/1     Running   0          65s
calico-node-vhddk                          1/1     Running   0          65s
coredns-76f75df574-2dp8h                   1/1     Running   0          12m
coredns-76f75df574-plqmz                   1/1     Running   0          12m
etcd-master-node                           1/1     Running   0          13m
kube-apiserver-master-node                 1/1     Running   0          13m
kube-controller-manager-master-node        1/1     Running   0          13m
kube-proxy-tcn2x                           1/1     Running   0          12m
kube-proxy-zlf76                           1/1     Running   0          2m56s
kube-scheduler-master-node                 1/1     Running   0          13m
```

9- Setup Kubernetes Metrics Server

Kubeadm doesn't install metrics server component during its initialization. We have to install it separately.

To verify this, if you run the top command, you will see the Metrics API not available error.

```
root@controlplane:~# kubectl top nodes
error: Metrics API not available
```

To install the metrics server, execute the following metric server manifest file. It deploys metrics server version v0.6.2

```
kubectl apply -f
https://raw.githubusercontent.com/techiescamp/kubeadm-scripts/main/manifest
s/metrics-server.yaml
```

Once the metrics server objects are deployed, it takes a minute for you to see the node and pod metrics using the top command.

```
kubectl top nodes
```

You should be able to view the node metrics as shown below.

```
root@controlplane:~# kubectl top nodes
NAME           CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
controlplane   142m         7%     1317Mi          34%
node01         36m          1%     915Mi           23%
```

You can also view the pod CPU and memory metrics using the following command.

```
kubectl top pod -n kube-system
```