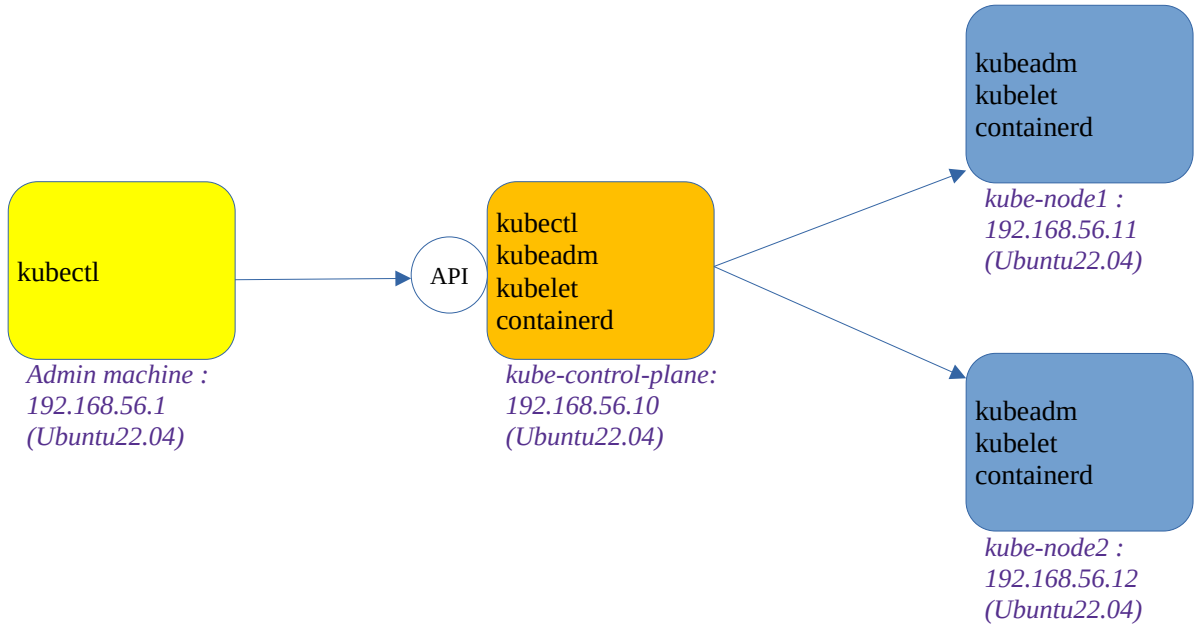
Lab01 – Creating a Cluster with Kubeadm

In this exercise, you will learn how to create a cluster using kubeadm. The cluster will contain of a single control plane node named `kube-control-plane`, and two worker nodes named `kube-node1` and `kube-node2`.



You can find a full description of the (https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/) in the official Kubernetes documentation.

Initializing the Control Plane Node

1. Git clone the https://github.com/brahimhamdi/k8s-lab repo, and then cd to k8s-lab directory.

```
brahim@Training:~$ git clone https://github.com/brahimhamdi/k8s-lab
Clonage dans 'k8s-lab'...
remote: Enumerating objects: 29, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 29 (delta 8), reused 29 (delta 8), pack-reused 0
Réception d'objets: 100% (29/29), 10.64 Kio | 265.00 Kio/s, fait.
Résolution des deltas: 100% (8/8), fait.
brahim@Training:~$ cd k
bash: cd: k: Aucun fichier ou dossier de ce type
brahim@Training:~$ cd k8s-lab
brahim@Training:~/k8s-lab$
```

2. Deploy vagrant environment using the command `vagrant up`.

```
brahim@Training:~/k8s-lab$ vagrant up
Bringing machine 'kube-control-plane' up with 'virtualbox' provider...
Bringing machine 'kube-node1' up with 'virtualbox' provider...
Bringing machine 'kube-node2' up with 'virtualbox' provider...
==> kube-control-plane: Importing base box 'generic/ubuntu2004'...
==> kube-control-plane: Matching MAC address for NAT networking...
==> kube-control-plane: Checking if box 'generic/ubuntu2004' version '4.2.14' is up to date...
==> kube-control-plane: A newer version of the box 'generic/ubuntu2004' for provider 'virtualbox' is
==> kube-control-plane: available! You currently have version '4.2.14'. The latest is version
==> kube-control-plane: '4.2.16'. Run `vagrant box update` to update.
==> kube-control-plane: Setting the name of the VM: k8s-lab_kube-control-plane_1689795061604_38153
==> kube-control-plane: Clearing any previously set network interfaces...
==> kube-control-plane: Preparing network interfaces based on configuration...
    kube-control-plane: Adapter 1: nat
    kube-control-plane: Adapter 2: hostonly
==> kube-control-plane: Forwarding ports...
    kube-control-plane: 22 (guest) => 2222 (host) (adapter 1)
==> kube-control-plane: Running 'pre-boot' VM customizations...
==> kube-control-plane: Booting VM...
==> kube-control-plane: Waiting for machine to boot. This may take a few minutes...
    kube-control-plane: SSH address: 127.0.0.1:2222
    kube-control-plane: SSH username: vagrant
    kube-control-plane: SSH auth method: private key
    kube-control-plane: Warning: Connection reset. Retrying...
```

3. Shell into control plane node using the command `vagrant ssh kube-control-plane`. Initializing the control plane using the `kubeadm init` command. Provide `10.32.0.0/12` as the IP addresses for the Pod network. Use `192.168.56.10` for the IP address the API Server will advertise it's listening on.

```
vagrant@kube-control-plane:~$ sudo kubeadm init --apiserver-advertise-address 192.168.56.10 --pod-network-cidr 10.32.0.0/12
I0219 08:55:59.541886    8913 version.go:256] remote version is much newer: v1.29.2; falling back to: stable-1.26
[init] Using Kubernetes version: v1.26.14
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

...

```
Your Kubernetes control-plane has initialized successfully!

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

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

kubeadm join 192.168.56.10:6443 --token floio4.l9dfu49dxj6gelpc \
        --discovery-token-ca-cert-hash sha256:7dd95037b5a673775a9db7dac69fab04fad6a811c2f35db70c47835301dd5859
vagrant@kube-control-plane:~$
```

4. After the `init` command finished, run the necessary commands to use *kubectl* tool as non-root user.

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

5. Install CNI plugin Calico using the command:

   kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml

```
vagrant@kube-control-plane:~$ kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
```

   - Set Calico to use eth1, with command :

   kubectl set env daemonset/calico-node -n kube-system IP_AUTODETECTION_METHOD=interface=eth1

```
vagrant@kube-control-plane:~$ kubectl set env daemonset/calico-node  -n kube-system IP_AUTODETECTION_METHOD=interface=eth1
vagrant@kube-control-plane:~$ _
```

6. Verify that the control plabe node indicates the "Ready" status with the command `kubectl get nodes -o wide`.

```
vagrant@kube-control-plane:~$ kubectl get nodes -o wide
NAME                 STATUS   ROLES           AGE   VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER
-RUNTIME
kube-control-plane   Ready    control-plane   11m   v1.27.4   192.168.56.10   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   container
d://1.6.21
vagrant@kube-control-plane:~$ _
```

7. Copy `kubeadm join` command generated by `sudo kubeadm token create --print-join-command` command, and then exit out of the VM using the command `exit`.

```
vagrant@kube-control-plane:~$ sudo kubeadm token create --print-join-command
kubeadm join 192.168.56.10:6443 --token 871ibd.wc0x7dqpbtrjj77a --discovery-token-ca-cert-hash sha256:8f97c9c18c7e0ad440d4a7e07536be4197ce19b7
ab604e9bdf461a51ab7e56c9
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$ exit
logout
brahim@Training:~/k8s-lab$
```

Joining the Worker Nodes

8. Shell into first worker node using the command `vagrant ssh kube-node1`. Join the worker node to cluster using the `kubeadm join` command. Provide the join token and CA cert hash.

```
brahim@Training:~/k8s-lab$ vagrant ssh kube-node1
vagrant@kube-node1:~$ sudo kubeadm join 192.168.56.10:6443 --token 871ibd.wc0x7dqpbtrjj77a --discovery-token-ca-cert-hash sha256:8f97c9c18c7e0
ad440d4a7e07536be4197ce19b7ab604e9bdf461a51ab7e56c9
[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 yaml'
[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.

vagrant@kube-node1:~$
```

9. Verify that the worker node indicates the "Ready" status with the command `kubectl get nodes`.

```
vagrant@kube-node1:~$ exit
logout
brahim@Training:~/k8s-lab$ vagrant ssh kube-control-plane
Last login: Wed Jul 19 20:25:43 2023 from 10.0.2.2
vagrant@kube-control-plane:~$ kubectl get nodes -o wide
NAME                STATUS   ROLES           AGE    VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINE
R-RUNTIME
kube-control-plane   Ready    control-plane   26m    v1.27.4   192.168.56.10   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   containe
rd://1.6.21
kube-node1           Ready    <none>          2m2s   v1.27.4   192.168.56.11   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   containe
rd://1.6.21
vagrant@kube-control-plane:~$
```

10. Exit out of the VM using the command `exit`. Repeat the steps for worker node `kube-node2`. Exit out of the VM using the command `exit`.

```
vagrant@kube-control-plane:~$ exit
logout
brahim@Training:~/k8s-lab$ vagrant ssh kube-node2
vagrant@kube-node2:~$ sudo kubeadm join 192.168.56.10:6443 --token 871ibd.wc0x7dqpbtrjj77a --discovery-token-ca-cert-hash sha256:8f97c9c18c7e0
ad440d4a7e07536be4197ce19b7ab604e9bdf461a51ab7e56c9
[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 yaml'
[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.

vagrant@kube-node2:~$
vagrant@kube-node2:~$ exit
logout
brahim@Training:~/k8s-lab$
brahim@Training:~/k8s-lab$ vagrant ssh kube-control-plane
Last login: Wed Jul 19 20:26:32 2023 from 10.0.2.2
vagrant@kube-control-plane:~$ kubectl get nodes -o wide
NAME                STATUS     ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTA
INER-RUNTIME
kube-control-plane   Ready      control-plane   29m     v1.27.4   192.168.56.10   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   conta
inerd://1.6.21
kube-node1           Ready      <none>          5m10s   v1.27.4   192.168.56.11   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   conta
inerd://1.6.21
kube-node2           NotReady   <none>          100s    v1.27.4   192.168.56.12   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   conta
inerd://1.6.21
vagrant@kube-control-plane:~$
```

Verifying the Installation

11. Check that all nodes have been correctly registered and are in the "Ready" status.

```
vagrant@kube-control-plane:~$ kubectl get nodes -o wide
NAME                STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAIN
ER-RUNTIME
kube-control-plane   Ready    control-plane   30m     v1.27.4   192.168.56.10   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contain
erd://1.6.21
kube-node1           Ready    <none>          6m39s   v1.27.4   192.168.56.11   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contain
erd://1.6.21
kube-node2           Ready    <none>          3m9s    v1.27.4   192.168.56.12   <none>        Ubuntu 20.04.5 LTS   5.4.0-139-generic   contain
erd://1.6.21
vagrant@kube-control-plane:~$
```

12. Check that cluster is ok, and all system pods are running in the kube-system namespace.

```
vagrant@kube-control-plane:~$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.56.10:6443
CoreDNS is running at https://192.168.56.10:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$ kubectl  get pod -A
NAMESPACE     NAME                                          READY   STATUS     RESTARTS      AGE
default       web                                           1/1     Running    0             6m16s
kube-system   calico-kube-controllers-658d97c59c-92mmq      1/1     Running    0             161m
kube-system   calico-node-ccxm7                             1/1     Running    0             43m
kube-system   calico-node-kbswc                             1/1     Running    0             68m
kube-system   calico-node-nldfr                             0/1     Init:2/3   1 (11m ago)   21m
kube-system   coredns-5dd5756b68-9n2d4                      1/1     Running    0             4h3m
kube-system   coredns-5dd5756b68-wv77k                      1/1     Running    0             4h3m
kube-system   etcd-kube-control-plane                       1/1     Running    0             4h3m
kube-system   kube-apiserver-kube-control-plane             1/1     Running    0             4h3m
kube-system   kube-controller-manager-kube-control-plane    1/1     Running    0             4h3m
kube-system   kube-proxy-r4v62                              1/1     Running    0             43m
kube-system   kube-proxy-ssxxb                              1/1     Running    0             21m
kube-system   kube-proxy-w8fds                              1/1     Running    0             4h3m
kube-system   kube-scheduler-kube-control-plane             1/1     Running    0             4h3m
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$
```

13. Create a new Pod named `web` with the image `nginx`. Check the node the Pod has been scheduled on.

```
vagrant@kube-control-plane:~$ kubectl get pod -owide
No resources found in default namespace.
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$ kubectl run web --image=nginx
pod/web created
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$ kubectl get pod -owide
NAME   READY   STATUS              RESTARTS   AGE   IP       NODE         NOMINATED NODE   READINESS GATES
web    0/1     ContainerCreating   0          3s    <none>   kube-node1   <none>           <none>
vagrant@kube-control-plane:~$
vagrant@kube-control-plane:~$ kubectl get pod -owide
NAME   READY   STATUS    RESTARTS   AGE   IP          NODE         NOMINATED NODE   READINESS GATES
web    1/1     Running   0          84s   10.44.0.1   kube-node1   <none>           <none>
vagrant@kube-control-plane:~$
```

14. Install kubectl and copy kubeconfig from master to Admin PC.

sudo mkdir /etc/apt/keyrings

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubectl

scp -o StrictHostKeyChecking=no -i ../.vagrant/machines/kube-control-plane/virtualbox/private_key -r \ vagrant@192.168.56.10:/home/vagrant/.kube $HOME/

```
brahim@Training:~/k8s-lab$ sudo mkdir /etc/apt/keyrings
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/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:/v1.28/deb/ /' | sudo tee /etc/apt/sources
.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
scp -o StrictHostKeyChecking=no -i ../.vagrant/machines/kube-control-plane/virtualbox/private_key -r vagrant@192.168.56.10:/home/vagrant/.kube
 $HOME/
[sudo] Mot de passe de brahim :
mkdir: impossible de créer le répertoire «/etc/apt/keyrings»: Le fichier existe
Atteint :1 http://fr.archive.ubuntu.com/ubuntu jammy InRelease
Réception de :2 http://fr.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Réception de :3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Atteint :4 https://packages.microsoft.com/repos/ms-teams stable InRelease
Atteint :5 https://download.docker.com/linux/ubuntu jammy InRelease
Atteint :6 http://download.virtualbox.org/virtualbox/debian jammy InRelease
Réception de :7 https://dl.google.com/linux/chrome/deb stable InRelease [1 825 B]
```

...

```
brahim@Training:~/k8s-lab$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.56.10:6443
CoreDNS is running at https://192.168.56.10:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
brahim@Training:~/k8s-lab$
brahim@Training:~/k8s-lab$ kubectl get pod -A
NAMESPACE     NAME                                       READY   STATUS     RESTARTS      AGE
default       web                                        1/1     Running    0             5m11s
kube-system   calico-kube-controllers-658d97c59c-92mmq   1/1     Running    0             160m
kube-system   calico-node-ccxm7                          1/1     Running    0             42m
kube-system   calico-node-kbswc                          1/1     Running    0             66m
kube-system   calico-node-nldfr                          0/1     Init:2/3   1 (10m ago)   20m
kube-system   coredns-5dd5756b68-9n2d4                   1/1     Running    0             4h2m
kube-system   coredns-5dd5756b68-wv77k                   1/1     Running    0             4h2m
kube-system   etcd-kube-control-plane                    1/1     Running    0             4h2m
kube-system   kube-apiserver-kube-control-plane          1/1     Running    0             4h2m
kube-system   kube-controller-manager-kube-control-plane 1/1     Running    0             4h2m
kube-system   kube-proxy-r4v62                           1/1     Running    0             42m
kube-system   kube-proxy-ssxxb                           1/1     Running    0             20m
kube-system   kube-proxy-w8fds                           1/1     Running    0             4h2m
kube-system   kube-scheduler-kube-control-plane          1/1     Running    0             4h2m
brahim@Training:~/k8s-lab$
brahim@Training:~/k8s-lab$
```