Version - **2025.01**

Last update : 2025/01/17 16:37

DOE301 - Creating Kubernetes clusters

Curriculum

VirtualBoxes

- Kubemaster
- Kubenode1
- Kubenode2

Container orchestration

The main containerization solutions are :

- **Docker**,
- **containerd**,
- **CRI-O**.

The main container orchestration solutions are:

- **Docker Swarm**,
- **Kubernetes**,
- **Mesos**.

Container orchestration provides:

- High availability,
- Load balancing,
- Scale up and scale down services.

Introduction to Kubernetes (k8s)

Kubernetes is also known as **K8s** because there are 8 characters between the letter **K** and the letter **S** ⇒ **K** ubernete **S**.

Control Plane

The Control Plane is a collection of services responsible for managing the K8s cluster:

- **kube-api-server**,
  - Main interface to the Control Plane,
- **Etcd**,
  - Key-value store, which stores all data used to manage the cluster and manage locks,
- **kube-controller-manager**,
  - Monitors the status of containers, nodes and end-points. Responsible for setting up new containers in the event of failures.
- **kube-scheduler**,
  - Distributes existing containers to nodes and searches for new containers and allocates them to nodes.
- **cloud-controller-manager**,
  - Provides an interface between K8s and the infrastructures of public cloud providers such as AWS and Azure.

Theoretically, the above services can be distributed across several servers in the Control Plane. In practice, they are often on a single server called the Controller.

Controller

Certain ports must be open on the Controller:

Protocol	Direction	Port(s)	Executable
TCP	Incoming	6443	Kubernetes API server
TCP	Inbound	2379-2380	etcd server client API
TCP	Inbound	10250	Kubelet API
TCP	Input	10251	kube-scheduler
TCP	Inbound	10252	kube-controller-manager

Nodes (Minions)

- Physical or virtual machine on which Kubernetes is installed,

- A worker on which Kubernetes launches containers,

The Node contains :

- **kubelet**,
  - agent running on each node,
  - responsible for monitoring containers.
  - It obtains its instructions from the Control Plane and communicates the status of the node and containers to the Control Plane,
- **Container runtime**,
  - Docker,
  - Containerd,
  - CRI-O (crio),
- **kube-proxy**,
  - a network proxy that handles networking tasks between containers and services in the cluster.cloud_user

Certain ports must be open on each worker node:

Protocol	Direction	Port(s)	Executable
TCP	Inbound	10250	Kubelet API
TCP	Inbound	30000-32767	Services NodePort

LAB #1 - Creating a Kubernetes cluster with Virtual Machines

1.1 - Overview

Note that the virtual machines used with Kubernetes must be running one of the following distributions:

- Ubuntu 16.04+,
- Debian 9+,
- CentOS 7,
- RHEL 7,
- Fedora 25+,
- HypriotOS v1.0.1+,

- Flatcar Container Linux (tested with 2512.3.0).

Each machine must have :

- A minimum of 2 GB RAM,
- A minimum of 2 CPUs.

Machines must :

- be on the same network,
- have a unique host name, a unique MAC address and a unique product_uuid,
- have swap **disabled**,
- use of **dnsmasq** by NetworkManager under Systemd **disabled**.

Three **Debian 9** virtual machines were configured according to the table below:

Virtual Machine	**Host name**	**Interface 1**	**Interface 2**
kubemaster	kubemaster.ittraining.loc	10.0.2.65	192.168.56.2
kubenode1	kubenode1.ittraining.loc	10.0.2.66	192.168.56.3
kubenode2	kubenode2.ittraining.loc	10.0.2.67	192.168.56.4

User names and passwords are :

User	**Password**
trainee	trainee
root	fenestros

Important: Each virtual machine has been pre-installed with **Docker**, **kubeadm**, **kubelet** and **kubectl**.

1.2 - Connecting to the kubemaster

Type the following command to connect to the **kubemaster** machine:

```
trainee@gateway:~$ ssh -l trainee 192.168.56.2
```

1.3 - Testing the network

Check the connectivity of each virtual machine:

```
trainee@kubemaster:~$ ping -c 4 192.168.56.3
PING 192.168.56.3 (192.168.56.3) 56(84) bytes of data.
64 bytes from 192.168.56.3: icmp_seq=1 ttl=64 time=0.762 ms
64 bytes from 192.168.56.3: icmp_seq=2 ttl=64 time=0.765 ms
64 bytes from 192.168.56.3: icmp_seq=3 ttl=64 time=0.819 ms
64 bytes from 192.168.56.3: icmp_seq=4 ttl=64 time=0.682 ms

--- 192.168.56.3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3006ms
rtt min/avg/max/mdev = 0.682/0.757/0.819/0.048 ms
trainee@kubemaster:~$ ping -c 4 192.168.56.4
PING 192.168.56.4 (192.168.56.4) 56(84) bytes of data.
64 bytes from 192.168.56.4: icmp_seq=1 ttl=64 time=1.26 ms
64 bytes from 192.168.56.4: icmp_seq=2 ttl=64 time=0.710 ms
64 bytes from 192.168.56.4: icmp_seq=3 ttl=64 time=0.684 ms
64 bytes from 192.168.56.4: icmp_seq=4 ttl=64 time=0.710 ms

--- 192.168.56.4 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3002ms
rtt min/avg/max/mdev = 0.684/0.841/1.260/0.242 ms
trainee@kubemaster:~$ ping -c 4 www.free.fr
PING www.free.fr (212.27.48.10) 56(84) bytes of data.
```

```
64 bytes from www.free.fr (212.27.48.10): icmp_seq=1 ttl=53 time=64.6 ms
64 bytes from www.free.fr (212.27.48.10): icmp_seq=2 ttl=53 time=76.3 ms
64 bytes from www.free.fr (212.27.48.10): icmp_seq=3 ttl=53 time=75.3 ms
64 bytes from www.free.fr (212.27.48.10): icmp_seq=4 ttl=53 time=87.2 ms

--- www.free.fr ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 64.674/75.894/87.200/7.975 ms
```

Become a the **root** user:

```
trainee@kubemaster:~$ su -
Password: fenestros
```

1.4 - Initializing the Cluster Controller

Remove any previous Kubernetes configuration:

```
root@kubemaster:~# kubeadm reset
```

Check the kubelet version:

```
root@kubemaster:~# kubelet --version
Kubernetes v1.24.2
```

When initializing the Kubernetes cluster, it will need certain images. View the list with the following command:

```
root@kubemaster:~# kubeadm config images list --kubernetes-version 1.24.2
k8s.gcr.io/kube-apiserver:v1.24.2
k8s.gcr.io/kube-controller-manager:v1.24.2
k8s.gcr.io/kube-scheduler:v1.24.2
k8s.gcr.io/kube-proxy:v1.24.2
```

```
k8s.gcr.io/pause:3.7
k8s.gcr.io/etcd:3.5.3-0
k8s.gcr.io/coredns/coredns:v1.8.6
```

Edit the **disabled_plugins** line in the **/etc/containerd/config.toml** file installed by Docker and restart the **containerd** service:

```
root@kubemaster:~# vi /etc/containerd/config.toml
root@kubemaster:~# cat /etc/containerd/config.toml
# Copyright 2018-2022 Docker Inc.

# Licensed under the Apache License, Version 2.0 (the “License”);
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at

# http://www.apache.org/licenses/LICENSE-2.0

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an “AS IS” BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#disabled_plugins = [“cri”]
disabled_plugins = [“”]

#root = “/var/lib/containerd”
#state = “/run/containerd”
#subreaper = true
#oom_score = 0

#[grpc]
# address = “/run/containerd/containerd.sock”
# uid = 0
# gid = 0
```

```
#[debug]
# address = “/run/containerd/debug.sock”
# uid = 0
# gid = 0
# level = “info”

root@kubemaster:~# systemctl restart containerd
```

Download the images :

```
root@kubemaster:~# kubeadm config images pull --kubernetes-version 1.24.2
[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.24.2
[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.24.2
[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.24.2
[config/images] Pulled k8s.gcr.io/kube-proxy:v1.24.2
[config/images] Pulled k8s.gcr.io/pause:3.7
[config/images] Pulled k8s.gcr.io/etcd:3.5.3-0
[config/images] Pulled k8s.gcr.io/coredns/coredns:v1.8.6
```

Enable local routing :

```
root@kubemaster:~# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Activate the **br_netfilter** module:

```
root@kubemaster:~# modprobe br_netfilter
```

Initialize the **kubemaster** cluster controller by specifying the CIDR of the **Calico** network extension and the contoller's IP address:

```
root@kubemaster:~# kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=192.168.56.2 --
kubernetes-version 1.24.2
[init] Using Kubernetes version: v1.24.2
[preflight] Running pre-flight checks
[WARNING SystemVerification]: missing optional cgroups: hugetlb
```

```
[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'
[certs] Using certificateDir folder “/etc/kubernetes/pki”
[certs] Generating “ca” certificate and key
[certs] Generating “apiserver” certificate and key
[certs] apiserver serving cert is signed for DNS names [kubemaster.ittraining.loc kubernetes kubernetes.default
kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.56.2]
[certs] Generating “apiserver-kubelet-client” certificate and key
[certs] Generating “front-proxy-ca” certificate and key
[certs] Generating “front-proxy-client” certificate and key
[certs] Generating “etcd/ca” certificate and key
[certs] Generating “etcd/server” certificate and key
[certs] etcd/server serving cert is signed for DNS names [kubemaster.ittraining.loc localhost] and IPs
[192.168.56.2 127.0.0.1 ::1]
[certs] Generating “etcd/peer” certificate and key
[certs] etcd/peer serving cert is signed for DNS names [kubemaster.ittraining.loc localhost] and IPs
[192.168.56.2 127.0.0.1 ::1]
[certs] Generating “etcd/healthcheck-client” certificate and key
[certs] Generating “apiserver-etcd-client” certificate and key
[certs] Generating “sa” key and public key
[kubeconfig] Using kubeconfig folder “/etc/kubernetes”
[kubeconfig] Writing “admin.conf” kubeconfig file
[kubeconfig] Writing “kubelet.conf” kubeconfig file
[kubeconfig] Writing “controller-manager.conf” kubeconfig file
[kubeconfig] Writing “scheduler.conf” kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file “/var/lib/kubelet/kubeadm-flags.env”
[kubelet-start] Writing kubelet configuration to file “/var/lib/kubelet/config.yaml”
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder “/etc/kubernetes/manifests”
[control-plane] Creating static Pod manifest for “kube-apiserver”
[control-plane] Creating static Pod manifest for “kube-controller-manager”
[control-plane] Creating static Pod manifest for “kube-scheduler”
[etcd] Creating static Pod manifest for local etcd in “/etc/kubernetes/manifests”
```

```
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory
“/etc/kubernetes/manifests”. This can take up to 4m0s
[kubelet-check] Initial timeout of 40s passed.
[apiclient] All control plane components are healthy after 160.003672 seconds
[upload-config] Storing the configuration used in ConfigMap “kubeadm-config” in the “kube-system” Namespace
[kubelet] Creating a ConfigMap “kubelet-config” in namespace kube-system with the configuration for the kubelets
in the cluster
[upload-certs] Skipping phase. Please see --upload-certs
[mark-control-plane] Marking the node kubemaster.ittraining.loc as control-plane by adding the labels: [node-
role.kubernetes.io/control-plane node.kubernetes.io/exclude-from-external-load-balancers]
[mark-control-plane] Marking the node kubemaster.ittraining.loc as control-plane by adding the taints [node-
role.kubernetes.io/master:NoSchedule node-role.kubernetes.io/control-plane:NoSchedule]
[bootstrap-token] Using token: 1uzpm6.urchj75w8vshk2sv
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to get nodes
[bootstrap-token] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get
long term certificate credentials
[bootstrap-token] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a
Node Bootstrap Token
[bootstrap-token] Configured RBAC rules to allow certificate rotation for all node client certificates in the
cluster
[bootstrap-token] Creating the “cluster-info” ConfigMap in the “kube-public” namespace
[kubelet-finalize] Updating “/etc/kubernetes/kubelet.conf” to point to a rotatable kubelet client certificate and
key
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 192.168.56.2:6443 --token 1uzpm6.urchj75w8vshk2sv \
--discovery-token-ca-cert-hash sha256:5814a04ca7f75a186a8b91731b596505401f99d4857e158cfbdd76929fec425b
```

Important: Note the message **Your Kubernetes control-plane has initialized successfully**.

Now create the **KUBECONFIG** variable:

```
root@kubemaster:~# export KUBECONFIG=/etc/kubernetes/admin.conf
```

Insert the following two lines at the end of the **/root/.bashrc** file:

```
root@kubemaster:~# vi .bashrc
root@kubemaster:~# tail .bashrc
...
KUBECONFIG=/etc/kubernetes/admin.conf
export KUBECONFIG
```

Important: By default, the kubemaster will not be used to host containers. To use the kubemaster as well as the 2 minions to host containers, use the **kubectl taint nodes –all**

node-role.kubernetes.io/master- command.

If you encounter a problem, such as a timeout, before running the kubeadm init command again, run the following command:

```
# kubeadm reset -f
```

1.5 - Installing a Network Extension for communication between PODs

Use the **kubectl** command to view the **pods** currently running:

```
root@kubemaster:~# kubectl get pods --namespace kube-system
NAME                                          READY   STATUS    RESTARTS   AGE
coredns-6d4b75cb6d-dw4ph                      0/1     Pending   0          7m28s
coredns-6d4b75cb6d-ms2jm                      0/1     Pending   0          7m28s
etcd-kubemaster.ittraining.loc                1/1     Running   0          7m42s
kube-apiserver-kubemaster.ittraining.loc      1/1     Running   0          7m42s
kube-controller-manager-kubemaster.ittraining.loc   1/1     Running   0          7m42s
kube-proxy-jx76x                              1/1     Running   0          7m29s
kube-scheduler-kubemaster.ittraining.loc      1/1     Running   0          7m42s
```

Important: A **namespace** is a virtual cluster that allows resources to be isolated from the physical cluster.

Note that both **coredns** pods are in a **pending** state. To switch the coredns state to running and enable the pods to communicate with each other, you need to install a network extension. There are several extensions, which we'll come back to later in this course:

- **Calico**,
- **Cilium**,
- **Flannel**,

- **Kube-router**,
- **Romana**,
- **WeaveNet**,
- **Antrea**,
- **kube-ovn**,
- Channel (uses Flannel for the network and Calico for the firewall).

To get a working cluster, we'll use the first extension on the list, namely **Calico** :

```
root@kubemaster:~# kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/calico.yaml
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
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
daemonset.apps/calico-node created
```

```
serviceaccount/calico-node created
deployment.apps/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
poddisruptionbudget.policy/calico-kube-controllers created
```

Important: If your unable to download the **calico.yaml** file above, download it from **here** to the current directory and modify the above command to **kubectl apply -f calico.yaml**.

1.6 - Connecting workers to the Controller

Run the command **kubeadm token create –print-join-command** and copy the command output :

```
root@kubemaster:~# kubeadm token create --print-join-command
kubeadm join 192.168.56.2:6443 --token vt0z6w.7vxmnnj6kinpu4it --discovery-token-ca-cert-hash
sha256:5814a04ca7f75a186a8b91731b596505401f99d4857e158cfbdd76929fec425b
```

Log in to **kubenode1** :

```
root@kubemaster:~# ssh -l trainee kubenode1
The authenticity of host 'kubenode1 (192.168.56.3)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjqF/aJgUc9jg56slNaZQdAUcvBOvE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'kubenode1,192.168.56.3' (ECDSA) to the list of known hosts.
trainee@kubenode1's password:
Linux kubenode1.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jul 12 11:25:40 2022 from 10.0.2.40
trainee@kubenode1:~$
```

Become **root** :

```
trainee@kubenode1:~$ su -
Password: fenestros
```

Edit the **disabled_plugins** line in the **/etc/containerd/config.toml** file installed by Docker and restart the **containerd** service:

```
root@kubenode1:~# vi /etc/containerd/config.toml
root@kubenode1:~# cat /etc/containerd/config.toml
# Copyright 2018-2020 Docker Inc.

# Licensed under the Apache License, Version 2.0 (the “License”);
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at

# http://www.apache.org/licenses/LICENSE-2.0

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an “AS IS” BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#disabled_plugins = [“cri”]
disabled_plugins = [“”]

#root = “/var/lib/containerd”
#state = “/run/containerd”
#subreaper = true
```

```
#oom_score = 0

#[grpc]
# address = “/run/containerd/containerd.sock”
# uid = 0
# gid = 0

#[debug]
# address = “/run/containerd/debug.sock”
# uid = 0
# gid = 0
# level = “info”

root@kubenode1:~# systemctl restart containerd
```

Remove any previous Kubernetes configuration :

```
root@kubenode1:~# kubeadm reset
```

Now use the copied command to join the node to the cluster:

```
root@kubenode1:~# kubeadm join 192.168.56.2:6443 --token vt0z6w.7vxmnnj6kinpu4it --discovery-token-ca-cert-hash
sha256:5814a04ca7f75a186a8b91731b596505401f99d4857e158cfbdd76929fec425b
[preflight] Running pre-flight checks
[WARNING SystemVerification]: missing optional cgroups: hugetlb
[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.
```

Disconnect from kubenode1 and connect to **kubenode2** :

```
root@kubenode1:~# exit
logout
trainee@kubenode1:~$ exit
logout
Connection to kubenode1 closed.
root@kubemaster:~# ssh -l trainee kubenode2
The authenticity of host 'kubenode2 (192.168.56.4)' can't be established.
ECDSA key fingerprint is SHA256:sEfHBv9azmK60cjqF/aJgUc9jg56slNaZQdAUcvBOvE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'kubenode2,192.168.56.4' (ECDSA) to the list of known hosts.
trainee@kubenode2's password:
Linux kubenode2.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jul 12 11:30:42 2022 from 10.0.2.40
trainee@kubenode2:~$
```

Edit the **disabled_plugins** line in the **/etc/containerd/config.toml** file installed by Docker and restart the **containerd** service:

```
root@kubenode2:~# vi /etc/containerd/config.toml
root@kubenode2:~# cat /etc/containerd/config.toml
# Copyright 2018-2020 Docker Inc.
```

```
# Licensed under the Apache License, Version 2.0 (the “License”);
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at

# http://www.apache.org/licenses/LICENSE-2.0

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an “AS IS” BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

#disabled_plugins = [“cri”]
disabled_plugins = [“”]

#root = “/var/lib/containerd”
#state = “/run/containerd”
#subreaper = true
#oom_score = 0

#[grpc]
# address = “/run/containerd/containerd.sock”
# uid = 0
# gid = 0

#[debug]
# address = “/run/containerd/debug.sock”
# uid = 0
# gid = 0
# level = “info”

root@kubenode2:~# systemctl restart containerd
```

Delete any previous Kubernetes configuration :

```
root@kubenode2:~# kubeadm reset
```

Now use the copied command to join the node to the cluster:

```
root@kubenode2:~# kubeadm join 192.168.56.2:6443 --token vt0z6w.7vxmnnj6kinpu4it --discovery-token-ca-cert-hash
sha256:5814a04ca7f75a186a8b91731b596505401f99d4857e158cfbdd76929fec425b
[preflight] Running pre-flight checks
[WARNING SystemVerification]: missing optional cgroups: hugetlb
[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.
```

The configuration file created by this process on each node is **/var/lib/kubelet/config.yaml** :

```
root@kubenode2:~# cat /var/lib/kubelet/config.yaml
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    cacheTTL: 0s
    enabled: true
  x509:
    clientCAFile: /etc/kubernetes/pki/ca.crt
authorization:
```

```
  mode: Webhook
  webhook:
    cacheAuthorizedTTL: 0s
    cacheUnauthorizedTTL: 0s
cgroupDriver: systemd
clusterDNS:
- 10.96.0.10
clusterDomain: cluster.local
cpuManagerReconcilePeriod: 0s
evictionPressureTransitionPeriod: 0s
fileCheckFrequency: 0s
healthzBindAddress: 127.0.0.1
healthzPort: 10248
httpCheckFrequency: 0s
imageMinimumGCAge: 0s
kind: KubeletConfiguration
logging:
  flushFrequency: 0
  options:
    json:
      infoBufferSize: "0"
  verbosity: 0
memorySwap: {}
nodeStatusReportFrequency: 0s
nodeStatusUpdateFrequency: 0s
rotateCertificates: true
runtimeRequestTimeout: 0s
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
```

A second file contains the kubelet environment:

```
root@kubenode2:~# cat /var/lib/kubelet/kubeadm-flags.env
KUBELET_KUBEADM_ARGS="--container-runtime=remote --container-runtime-
endpoint=unix:///var/run/containerd/containerd.sock --pod-infra-container-image=k8s.gcr.io/pause:3.7”
```

Finally, run the **kubectl get nodes** command in kubemaster :

```
root@kubenode2:~# exit
déconnexion
trainee@kubenode2:~$ exit
déconnexion
Connection to kubenode2 closed.
root@kubemaster:~# kubectl get nodes
NAME                        STATUS     ROLES           AGE    VERSION
kubemaster.ittraining.loc   Ready      control-plane   3h8m   v1.24.2
kubenode1.ittraining.loc    NotReady   <none>          5m3s   v1.24.2
kubenode2.ittraining.loc    NotReady   <none>          89s    v1.24.2
```

1.7 - K8s and High Availability

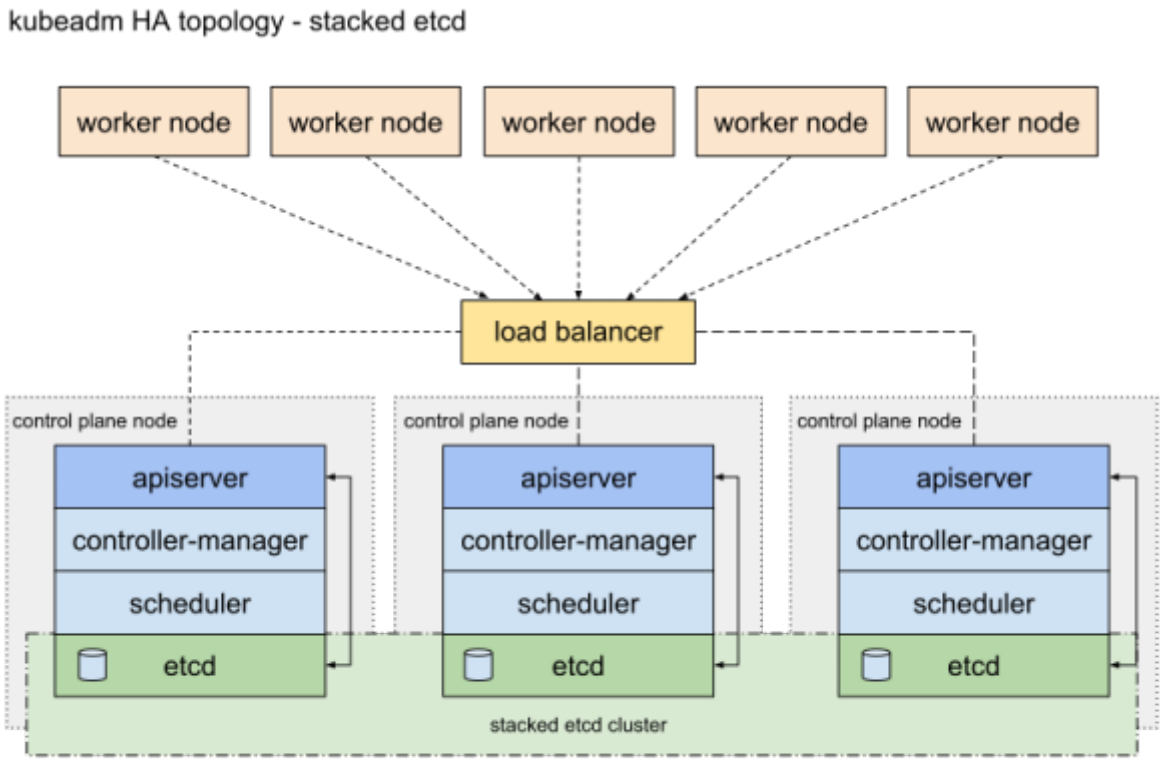
In order to implement K8s in a high-availability environment, it is necessary to have at least two **Controllers** in the cluster.

There are two methods for managing Etcd instances in the case of high availability:

- **Stacked** Etcd
- **External** Etcd

Stacked Etcd

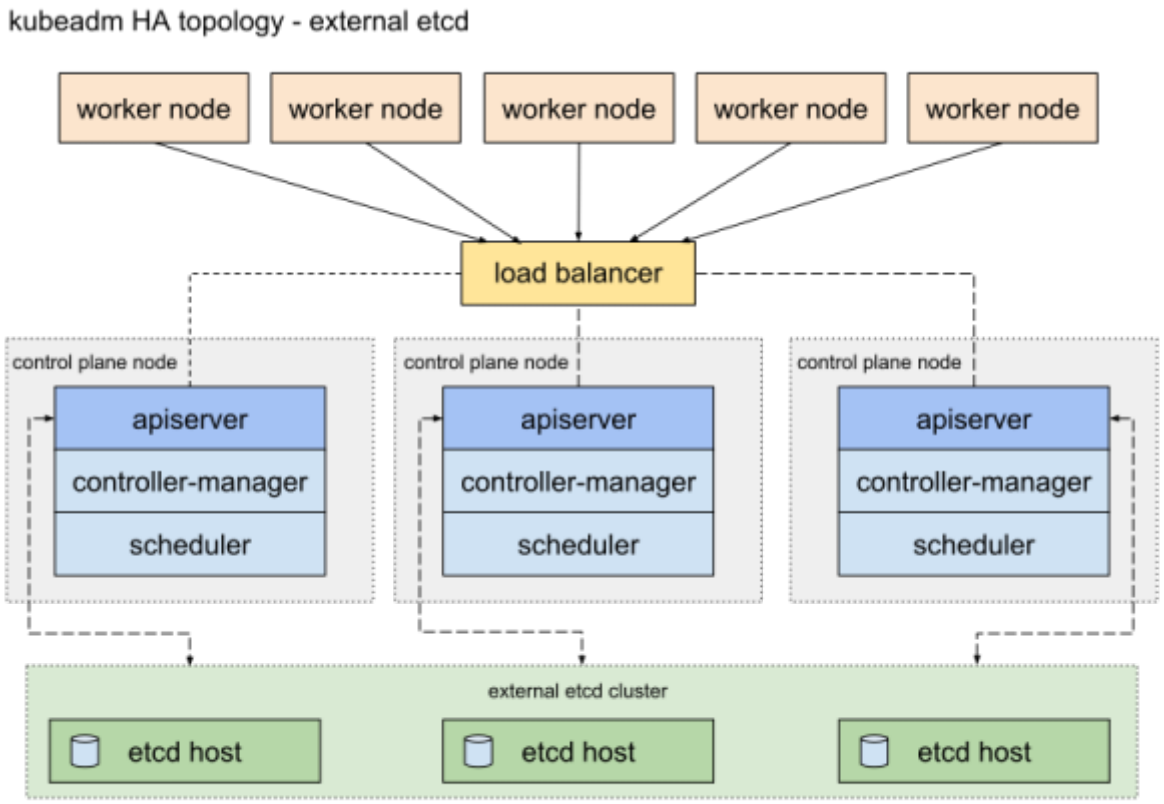
In this case, each Controller has its own Etcd instance, which communicates with each of the other instances:



Important: This architecture is the one used by default by the **kubeadm** command when creating a cluster.

External Etcd

In this case, each Etcd instance is placed on a different server that is not a Controller:



Important: For more information about setting up a cluster using Stacked Etcd or External Etcd, see **this page** in the official K8s documentation.

LAB #2 - Creating a Kubernetes cluster with Minikube

2.1 - Introducing Minikube

To install Kubernetes quickly and easily, you need to use Minikube. Minikube lets you create a cluster with a single node.

2.2 - Minikube installation

Return to your Gateway and download Minikube :

```
trainee@gateway:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 71.4M  100 71.4M    0     0  58.0M      0  0:00:01  0:00:01 --:--:-- 58.0M
```

Rename the downloaded binary and make it executable:

```
trainee@gateway:~$ mv minikube-linux-amd64 minikube
trainee@gateway:~$ chmod u+x minikube
```

Then move the minikube binary to the **/usr/local/bin/** directory:

```
trainee@gateway:~$ su -
Password:
root@gateway:~# mv /home/trainee/minikube /usr/local/bin/
root@gateway:~# exit
logout
trainee@gateway:~$
```

Important: Ask your trainer for the **root** user password.

Then test the installation with the **minikube version** command:

```
trainee@gateway:~$ minikube version
minikube version: v1.34.0
commit: f4b412861bb746be73053c9f6d2895f12cf78565
```

2.3 - Minikube configuration

Now configure minikube's default hypervisor:

```
trainee@gateway:~$ minikube config set vm-driver virtualbox
❗ These changes will take effect upon a minikube delete and then a minikube start
```

Check that the last command was successful:

```
trainee@gateway:~$ minikube config get vm-driver
virtualbox
```

By default, when Minikube starts up, it will allocate 2 vCPUs and 2GB of RAM to its virtual machine. Increase the amount of memory allocated with the following command:

```
trainee@gateway:~$ minikube config set memory 4000
❗ These changes will take effect upon a minikube delete and then a minikube start
```

Check that the last command was successful:

```
trainee@gateway:~$ minikube config get memory
4000
```

2.4 - Installing Docker

Docker is not included in the Debian repository. To install it, you need to add the docker repository. First, you need to install the packages that enable Debian to use an https repository:

```
root@gateway:~# apt-get update
...
root@gateway:~# apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common
```

Download the official docker GPG key:

```
root@gateway:~# curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
```

Add the **stable** docker repository :

```
root@gateway:~# add-apt-repository “deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release -cs)
stable”
```

Important - Note that the **lsb_release -cs** command returns the name of the Debian distribution, in this case **stretch**.

Now install the **docker-ce** package:

```
root@gatewaydocker:~# apt-get update
...
root@gateway:~# apt-get install docker-ce
```

Lastly, check the version of Docker client and server :

```
root@gateway:~# docker version
Client: Docker Engine - Community
```

```
 Version:           27.3.1
 API version:       1.47
 Go version:        go1.22.7
 Git commit:        ce12230
 Built:             Fri Sep 20 11:41:19 2024
 OS/Arch:           linux/amd64
 Context:           default

Server: Docker Engine - Community
 Engine:
  Version:          27.3.1
  API version:      1.47 (minimum version 1.24)
  Go version:       go1.22.7
  Git commit:       41ca978
  Built:            Fri Sep 20 11:41:19 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.24
  GitCommit:        88bf19b2105c8b17560993bee28a01ddc2f97182
 runc:
  Version:          1.2.2
  GitCommit:        v1.2.2-0-g7cb3632
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Important - Note that the docker-ce package needs the **containerd.io** and **docker-ce-cli** packages. Also note that the above procedure installs the most recent version of Docker.

2.5 - Installing kubectl

Now start Minikube:

```
trainee@gateway:~$ minikube start
😄  minikube v1.34.0 on Debian 11.8 (kvm/amd64)
✨  Using the virtualbox driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🤷  virtualbox "minikube" VM is missing, will recreate.
🔥  Creating virtualbox VM (CPUs=2, Memory=4000MB, Disk=20000MB) ...
🐳  Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
┌─────────────────────────────────────────────────────────────────────────────────────────────┐
│                                                                                             │
│    You have selected "virtualbox" driver, but there are better options !                    │
│    For better performance and support consider using a different driver:                    │
│            - kvm2                                                                           │
│            - qemu2                                                                          │
│                                                                                             │
│    To turn off this warning run:                                                            │
│                                                                                             │
│            $ minikube config set WantVirtualBoxDriverWarning false                         │
│                                                                                             │
│                                                                                             │
│    To learn more about on minikube drivers checkout https://minikube.sigs.k8s.io/docs/drivers/ │
│    To see benchmarks checkout https://minikube.sigs.k8s.io/docs/benchmarks/cpuusage/        │
│                                                                                             │
└─────────────────────────────────────────────────────────────────────────────────────────────┘
🔎  Verifying Kubernetes components...
```

```
  Enabled addons: default-storageclass, storage-provisioner
  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Note the error **kubectl not found.**. Run **minikube kubectl – get pods -A** to install kubectl :

```
trainee@gateway:~$ minikube kubectl -- get pods -A
    > kubectl.sha256:  64 B / 64 B [-------------------------] 100.00% ? p/s 0s
    > kubectl:  53.77 MiB / 53.77 MiB [----------] 100.00% 227.90 MiB p/s 400ms
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE
kube-system   coredns-6f6b679f8f-9ql4g           1/1     Running   0          3m48s
kube-system   coredns-6f6b679f8f-n5b86           1/1     Running   0          3m48s
kube-system   etcd-minikube                      1/1     Running   0          4m
kube-system   kube-apiserver-minikube            1/1     Running   0          3m52s
kube-system   kube-controller-manager-minikube   1/1     Running   0          3m51s
kube-system   kube-proxy-5rnt6                   1/1     Running   0          3m49s
kube-system   kube-scheduler-minikube            1/1     Running   0          3m58s
kube-system   storage-provisioner                1/1     Running   0          3m40s
```

See the list of running virtual machines:

```
trainee@gateway:~$ VBoxManage list runningvms
“minikube” {6c0e23dd-c7ab-45ce-b859-7821051f5bac}
```

Stop Minikube now:

```
trainee@gateway:~$ minikube stop
 Stopping node “minikube” ...
 1 node stopped.
```

Note that, although stopped, the **minikube** virtual machine is still present:

```
trainee@gateway:~$ VBoxManage list runningvms
trainee@gateway:~$ VBoxManage list vms
```

```
“minikube” {6c0e23dd-c7ab-45ce-b859-7821051f5bac}
```

Start minikube again:

```
rainee@gateway:~$ minikube start
😄  minikube v1.34.0 on Debian 11.8 (kvm/amd64)
✨  Using the virtualbox driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🔄  Restarting existing virtualbox VM for "minikube" ...
🐳  Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
╭─────────────────────────────────────────────────────────────────────────────────────────────╮
│                                                                                             │
│    You have selected "virtualbox" driver, but there are better options !                    │
│    For better performance and support consider using a different driver:                    │
│            - kvm2                                                                           │
│            - qemu2                                                                          │
│                                                                                             │
│    To turn off this warning run:                                                            │
│                                                                                             │
│            $ minikube config set WantVirtualBoxDriverWarning false                         │
│                                                                                             │
│                                                                                             │
│    To learn more about on minikube drivers checkout https://minikube.sigs.k8s.io/docs/drivers/     │
│    To see benchmarks checkout https://minikube.sigs.k8s.io/docs/benchmarks/cpuusage/        │
│                                                                                             │
╰─────────────────────────────────────────────────────────────────────────────────────────────╯
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🔎  Verifying Kubernetes components...
🌟  Enabled addons: default-storageclass, storage-provisioner
❗  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Check which version of kubectl has been installed:

```
trainee@gateway:~$ minikube kubectl version
Client Version: v1.31.0
Kustomize Version: v5.4.2
Server Version: v1.31.0
```

Important : The output of this command indicates Kubernetes version 1.31.0.

The version of kubectl installed by minikube can be found in the directory **/home/trainee/.minikube/cache/linux/amd64/v1.31.0/** :

```
trainee@gateway:~$ ls -l /home/trainee/.minikube/cache/linux/amd64/v1.31.0/kubectl
-rwxr-xr-x 1 trainee trainee 56381592 Dec 4 13:58 /home/trainee/.minikube/cache/linux/amd64/v1.31.0/kubectl
```

For ease of use, copy the command to the **/usr/local/bin/** directory:

```
trainee@gateway:~$ su -
Password:
root@gateway:~# cp /home/trainee/.minikube/cache/linux/amd64/v1.31.0/kubectl /usr/local/bin
root@gateway:~# exit
logout
```

Then check that the command is available:

```
trainee@gateway:~$ which kubectl
/usr/local/bin/kubectl
```

2.6 - The minikube addons command

Minikube uses modules. These modules are called **addons**. To view installed addons and their status, use the **minikube addons list** command:

```
trainee@gateway:~$ minikube addons list
|-----------------------------|----------|--------------|---------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER            |
|-----------------------------|----------|--------------|---------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)          |
| auto-pause                  | minikube | disabled     | minikube                        |
| cloud-spanner               | minikube | disabled     | Google                          |
| csi-hostpath-driver         | minikube | disabled     | Kubernetes                      |
| dashboard                   | minikube | disabled     | Kubernetes                      |
| default-storageclass        | minikube | enabled □    | Kubernetes                      |
| efk                         | minikube | disabled     | 3rd party (Elastic)             |
| freshpod                    | minikube | disabled     | Google                          |
| gcp-auth                    | minikube | disabled     | Google                          |
| gvisor                      | minikube | disabled     | minikube                        |
| headlamp                    | minikube | disabled     | 3rd party (kinvolk.io)          |
| helm-tiller                 | minikube | disabled     | 3rd party (Helm)                |
| inaccel                     | minikube | disabled     | 3rd party (InAccel              |
|                             |          |              | [info@inaccel.com])             |
| ingress                     | minikube | disabled     | Kubernetes                      |
| ingress-dns                 | minikube | disabled     | minikube                        |
| inspektor-gadget            | minikube | disabled     | 3rd party                       |
|                             |          |              | (inspektor-gadget.io)           |
| istio                       | minikube | disabled     | 3rd party (Istio)               |
| istio-provisioner           | minikube | disabled     | 3rd party (Istio)               |
| kong                        | minikube | disabled     | 3rd party (Kong HQ)             |
| kubeflow                    | minikube | disabled     | 3rd party                       |
| kubevirt                    | minikube | disabled     | 3rd party (KubeVirt)            |
| logviewer                   | minikube | disabled     | 3rd party (unknown)             |
| metallb                     | minikube | disabled     | 3rd party (MetalLB)             |
| metrics-server              | minikube | disabled     | Kubernetes                      |
| nvidia-device-plugin        | minikube | disabled     | 3rd party (NVIDIA)              |
| nvidia-driver-installer     | minikube | disabled     | 3rd party (NVIDIA)              |
| nvidia-gpu-device-plugin    | minikube | disabled     | 3rd party (NVIDIA)              |
| olm                         | minikube | disabled     | 3rd party (Operator Framework)  |
```

```
| pod-security-policy         | minikube | disabled     | 3rd party (unknown)            |
| portainer                   | minikube | disabled     | 3rd party (Portainer.io)       |
| registry                    | minikube | disabled     | minikube                       |
| registry-aliases            | minikube | disabled     | 3rd party (unknown)            |
| registry-creds              | minikube | disabled     | 3rd party (UPMC Enterprises)   |
| storage-provisioner         | minikube | enabled      | minikube                       |
| storage-provisioner-gluster | minikube | disabled     | 3rd party (Gluster)            |
| storage-provisioner-rancher | minikube | disabled     | 3rd party (Rancher)            |
| volcano                     | minikube | disabled     | third-party (volcano)          |
| volumesnapshots             | minikube | disabled     | Kubernetes                     |
| yakd                        | minikube | disabled     | 3rd party (marcnuri.com)       |
|-----------------------------|----------|--------------|--------------------------------|
```

To activate the **metrics-server** module, use the **minikube addons enable** command:

```
trainee@gateway:~$ minikube addons enable metrics-server
    metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/metrics-server/metrics-server:v0.7.2
    The 'metrics-server' addon is enabled
```

Now check that the previous command has been taken into account:

```
trainee@gateway:~$ minikube addons list
|-----------------------------|----------|--------------|--------------------------------|
|         ADDON NAME          | PROFILE  |    STATUS    |           MAINTAINER           |
|-----------------------------|----------|--------------|--------------------------------|
| ambassador                  | minikube | disabled     | 3rd party (Ambassador)         |
| auto-pause                  | minikube | disabled     | minikube                       |
| cloud-spanner               | minikube | disabled     | Google                         |
| csi-hostpath-driver         | minikube | disabled     | Kubernetes                     |
| dashboard                   | minikube | disabled     | Kubernetes                     |
| default-storageclass        | minikube | enabled      | Kubernetes                     |
| efk                         | minikube | disabled     | 3rd party (Elastic)            |
```

```
| freshpod                     | minikube | disabled     | Google                          |
| gcp-auth                     | minikube | disabled     | Google                          |
| gvisor                       | minikube | disabled     | minikube                        |
| headlamp                     | minikube | disabled     | 3rd party (kinvolk.io)          |
| helm-tiller                  | minikube | disabled     | 3rd party (Helm)                |
| inaccel                      | minikube | disabled     | 3rd party (InAccel              |
|                              |          |              | [info@inaccel.com])             |
| ingress                      | minikube | disabled     | Kubernetes                      |
| ingress-dns                  | minikube | disabled     | minikube                        |
| inspektor-gadget             | minikube | disabled     | 3rd party                       |
|                              |          |              | (inspektor-gadget.io)           |
| istio                        | minikube | disabled     | 3rd party (Istio)               |
| istio-provisioner            | minikube | disabled     | 3rd party (Istio)               |
| kong                         | minikube | disabled     | 3rd party (Kong HQ)             |
| kubeflow                     | minikube | disabled     | 3rd party                       |
| kubevirt                     | minikube | disabled     | 3rd party (KubeVirt)            |
| logviewer                    | minikube | disabled     | 3rd party (unknown)             |
| metallb                      | minikube | disabled     | 3rd party (MetalLB)             |
| metrics-server               | minikube | enabled      | Kubernetes                      |
| nvidia-device-plugin         | minikube | disabled     | 3rd party (NVIDIA)              |
| nvidia-driver-installer      | minikube | disabled     | 3rd party (NVIDIA)              |
| nvidia-gpu-device-plugin     | minikube | disabled     | 3rd party (NVIDIA)              |
| olm                          | minikube | disabled     | 3rd party (Operator Framework)  |
| pod-security-policy          | minikube | disabled     | 3rd party (unknown)             |
| portainer                    | minikube | disabled     | 3rd party (Portainer.io)        |
| registry                     | minikube | disabled     | minikube                        |
| registry-aliases             | minikube | disabled     | 3rd party (unknown)             |
| registry-creds               | minikube | disabled     | 3rd party (UPMC Enterprises)    |
| storage-provisioner          | minikube | enabled      | minikube                        |
| storage-provisioner-gluster  | minikube | disabled     | 3rd party (Gluster)             |
| storage-provisioner-rancher  | minikube | disabled     | 3rd party (Rancher)             |
| volcano                      | minikube | disabled     | third-party (volcano)           |
| volumesnapshots              | minikube | disabled     | Kubernetes                      |
| yakd                         | minikube | disabled     | 3rd party (marcnuri.com)        |
```

```
|-----------------------------|----------|--------------|---------------------------------|
```

2.7 - The minikube dashboard addon

Activate the **dashboard** module:

```
trainee@gateway:~$ minikube addons enable dashboard
    dashboard is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
    Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server

    The 'dashboard' addon is enabled
```

Connect to your Gateway GUI.

Run the **minikube dashboard** command in a graphical terminal. The Firefox browser will be launched, giving you access to Kubernetes Dashboard.

Return to the SSH connection window.

Now return to the **kubemaster** VM:

```
trainee@gateway:~$ ssh -l trainee 10.0.2.65
trainee@10.0.2.65's password:
Linux kubemaster.ittraining.loc 4.9.0-19-amd64 #1 SMP Debian 4.9.320-2 (2022-06-30) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jul 12 16:14:47 2022 from 10.0.2.40
trainee@kubemaster:~$
```

Become root and check the status of the cluster nodes:

```
trainee@kubemaster:~$ su -
Mot de passe :
root@kubemaster:~# kubectl get nodes
NAME                        STATUS   ROLES           AGE    VERSION
kubemaster.ittraining.loc   Ready    control-plane   4h1m   v1.24.2
kubenode1.ittraining.loc    Ready    <none>          57m    v1.24.2
kubenode2.ittraining.loc    Ready    <none>          54m    v1.24.2
```

Then check the status of the pods:

```
root@kubemaster:~# kubectl get pods --all-namespaces
NAMESPACE     NAME                                                READY   STATUS    RESTARTS        AGE
kube-system   calico-kube-controllers-bc5cbc89f-rpbsc             1/1     Running   0               65m
kube-system   calico-node-9qwnr                                   1/1     Running   0               65m
kube-system   calico-node-rrkkk                                   1/1     Running   0               51m
kube-system   calico-node-szmcq                                   1/1     Running   0               53m
kube-system   coredns-6d4b75cb6d-btmcw                            1/1     Running   0               67m
kube-system   coredns-6d4b75cb6d-m6mpc                            1/1     Running   0               67m
kube-system   etcd-kubemaster.ittraining.loc                      1/1     Running   3 (63m ago)     67m
kube-system   kube-apiserver-kubemaster.ittraining.loc            1/1     Running   5 (62m ago)     67m
kube-system   kube-controller-manager-kubemaster.ittraining.loc   1/1     Running   12 (57m ago)    67m
kube-system   kube-proxy-7z9hs                                    1/1     Running   0               53m
kube-system   kube-proxy-k2q55                                    1/1     Running   0               67m
kube-system   kube-proxy-pcdj9                                    1/1     Running   0               51m
kube-system   kube-scheduler-kubemaster.ittraining.loc            1/1     Running   13 (57m ago)    67m
```