Version - **2024.01**

Dernière mise-à-jour : 2024/12/15 07:03

DOF301 - Création de Clusters Kubernetes

Contenu du Module

- **DOF301 - Création de Clusters Kubernetes**
  - Contenu du Module
  - L'Orchestration de Conteneurs
  - Présentation de Kubernetes (k8s)
    - Master
    - Nœuds (Minions)
  - LAB #1 - Création du Cluster Kubernetes avec des Machines Virtuelles
    - 1.1 - Présentation
    - 1.2 - Connexion à la Machine Virtuelle kubemaster
    - 1.3 - Tester le Réseau
    - 1.4 - Initialisation du Maître du Cluster
    - 1.5 - Installation d'une Extension Réseau pour la Communication entre des PODs
    - 1.6 - Connexion des Travailleurs au Maître
    - 1.7 - K8s et la Haute Disponibilité
  - LAB #2 - Création du Cluster Kubernetes avec Minikube
    - 2.1 - Présentation de Minikube
    - 2.2 - Installation de Minikube
    - 2.3 - Configuration de Minikube
    - 2.4 - Installation de Docker
    - 2.5 - Installation de kubectl
    - 2.6 - La Commande minikube addons
    - 2.7 - La Commande minikube dashboard

Machines Virtuelles VirtualBox

- Kubemaster
- Kubenode1
- Kubenode2

Ressources

Lab #1

- calico.yaml

L'Orchestration de Conteneurs

Les principales solutions de la containérisation sont :

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

Les principales solutions d'orchestration de conteneurs sont :

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

L'orchestration de conteneurs apporte :

- La haute disponibilité,
- L'équilibrage de charge,

- L'augmentation et la réduction du Services (Scale up / scale down).

Présentation de Kubernetes (k8s)

Kubernetes est aussi connu sous le nom **K8s** car il y a 8 caractères entre la letter **K** et la lettre **S** ⇒ **K** ubernete **S**.

Control Plane

Le Control Plane est une collection de services qui sont responsables de la gestion du cluster K8s :

- **kube-api-server**,
  - Interface principale au Control Plane,
- **Etcd**,
  - Key-value store qui stocke toutes les données utilisées pour gérer le cluster et gérer les verrous,
- **kube-controller-manager**,
  - Surveille l'état des conteneurs, nœuds et end-points. Responsable de la mise en place de nouveaux conteneurs en cas de défaillances.
- **kube-scheduler**,
  - Distribue les conteneurs existants aux nœuds et cherche des nouveaux conteneurs et les attribue aux nœuds.
- **cloud-controller-manager**,
  - Fournit une interface entre K8s et les infrastructures des for,nisseurs de cloud publics tels AWS et Azure.

Théoriquement les services ci-dessus peuvent être distribués sur plusieurs serveurs dans le Control Plane. En pratique ils sont souvent sur un seul et unique serveur appelé le Contrôleur.

Contrôleur

Certains ports doivent être ouverts sur le Contrôleur :

Protocole	Direction	Port(s)	Exécutable
TCP	Entrante	6443	Kubernetes API server

Protocole	Direction	Port(s)	Exécutable
TCP	Entrante	2379-2380	etcd server client API
TCP	Entrante	10250	Kubelet API
TCP	Entrante	10251	kube-scheduler
TCP	Entrante	10252	kube-controller-manager

Noeuds (Minions)

- Machine physique ou virtuelle sur laquelle est installé Kubernetes,
- Un travailleur sur lequel Kubernetes lance des conteneurs,

Le Nœud contient :

- **kubelet**,
  - agent qui s'exécute sur chaque noeud,
  - responsable de la surveillance des conteneurs.
  - obtient ses instructions du Control Plane et communique le statut du neoud et les containers au Control Plane,
- **Container runtime**,
  - Docker,
  - Containerd,
  - CRI-O (crio),
- **kube-proxy**,
  - un proxy réseau qui s'occupe des tâches en relation avec le networking entre conteneurs et services dans le cluster.cloud_user

Certains ports doivent être ouverts sur chaque noeud travailleur :

Protocole	Direction	Port(s)	Exécutable
TCP	Entrante	10250	Kubelet API
TCP	Entrante	30000-32767	Services NodePort

LAB #1 - Création du Cluster Kubernetes avec des Machines Virtuelles

1.1 - Présentation

Notez que les machines virtuelles utilisées avec Kubernetes doivent être sous une des distributions suivantes :

- Ubuntu 16.04+,
- Debian 9+,
- CentOS 7,
- RHEL 7,
- Fedora 25+,
- HypriotOS v1.0.1+,
- Flatcar Container Linux (tested with 2512.3.0).

Chaque machine doit avoir :

- Un minimum de 2 GO de RAM,
- Un minimum de 2 CPU.

Les machines doivent :

- être dans le même réseau,
- posséder un nom d'hôte unique, une adresse MAC unique ainsi qu'un product_uuid unique,
- avoir le swap **désactivé**,
- avoir l'utilisation de **dnsmasq** par NetworkManager sous Systemd **désactivée**.

Trois machines virtuelles **Debian 9** ont été configurées selon le tableau ci-dessous :

Machine Virtuelle	**Nom d'hôte**	**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

Les noms d'utilisateurs et les mots de passe sont :

Utilisateur	Mot de Passe
trainee	trainee
root	fenestros

Important : Chaque machine virtuelle a été pré-installée avec **Docker**, **kubeadm**, **kubelet** et **kubectl**.

1.2 - Connexion à la Machine Virtuelle kubemaster

Tapez la commande suivante pour vous connecter à la machine **kubemaster** :

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

1.3 - Tester le Réseau

Vérifiez la connectivité de chaque machine virtuelle :

```
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
```

Devenez maintenant **root** :

```
trainee@kubemaster:~$ su -
Mot de passe : fenestros
```

1.4 - Initialisation du Maître du Cluster

Supprimez toute configuration Kubernetes antérieure :

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

Vérifiez la version de kubelet :

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

Lors de l'initialisation du cluster de Kubernetes, celle-ci aura besoin de certaines images. Visualisez la liste avec la commande suivante :

```
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
```

Editez la ligne **disabled_plugins** du fichier **/etc/containerd/config.toml** installé par Docker et re-démarrez le service **containerd** :

```
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
```

Téléchargez ces 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
```

Activez le routage local :

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

Activez le module **br_netfilter** :

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

Initialisez le maître du cluster **kubemaster** en spécifiant le CIDR de l'extension réseau **Calico** ainsi que l'adresse IP du maître :

```
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 : Notez le message **Your Kubernetes control-plane has initialized successfully**.

Créez maintenant la variable **KUBECONFIG** :

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

Insérez les deux lignes suivantes à la fin du fichier **/root/.bashrc** :

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

Important : Par défaut, le kubemaster ne sera pas utilisé pour héberger de conteneurs. Pour utiliser le kubemaster aussi bien que les 2 minions pour héberger des conteneurs, il convient d'utiliser la commande **kubectl taint nodes –all node-role.kubernetes.io/master-**.

Si vous rencontrez un problème, par exemple un timeout, avant de relancer la commande kubeadm init, lancez la commande suivante :

```
# kubeadm reset -f
```

1.5 - Installation d'une Extension Réseau pour la Communication entre des PODs

Utilisez la commande **kubectl** pour visualiser les **pods** en cours d'exécution :

```
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 : Un **namespace** est un cluster virtuel qui permet l'isolation des ressources dans le cluster physique.

Notez que les deux pods **coredns** sont dans un état de **pending**. De façon passer l'état des coredns en running et permettre les Pods de se communiquer entre-eux, il faut installer une extension pour le réseau . Il existe plusieurs extensions sur lesquelles nous reviendrons plus tard dans ce cours :

- **Calico**,
- **Cilium**,
- **Flannel**,
- **Kube-router**,
- **Romana**,
- **WeaveNet**,
- **Antrea**,
- **kube-ovn**,
- Canal (utilise Flannel pour le réseau et Calico pour le pare-feu).

Afin d'obtenir un cluster fonctionnel, nous allons utiliser la première extension de la liste, à savoir **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
```

1.6 - Connexion des Travailleurs au Maître

Exécutez la commande **kubeadm token create –print-join-command** et copiez la sortie de la commande :

```
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
```

Connectez-vous à **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:~$
```

Devenez **root** :

```
trainee@kubenode1:~$ su -
Mot de passe : fenestros
```

Editez la ligne **disabled_plugins** du fichier **/etc/containerd/config.toml** installé par Docker et re-démarrez le service **containerd** :

```
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
```

Supprimez toute configuration Kubernetes antérieure :

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

Utilisez maintenant la commande copiée pour joindre le nœud au 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.
```

Déconnectez-vous de kubenode1 et connectez-vous à **kubenode2** :

```
root@kubenode1:~# exit
déconnexion
trainee@kubenode1:~$ exit
déconnexion
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:~$
```

Éditez la ligne **disabled_plugins** du fichier **/etc/containerd/config.toml** installé par Docker et re-démarrez le service **containerd** :

```
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
```

Supprimez toute configuration Kubernetes antérieure :

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

Utilisez maintenant la commande copiée pour joindre le nœud au 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.
```

Le fichier de configuration créé par ce processus sur chaque noeud est **/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
```

Un deuxième fichier contient l'environnement de kubelet :

```
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"
```

Dernièrement, exécutez la commande **kubectl get nodes** dans 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 et la Haute Disponibilité

Afin de mettre en place K8s dans un environnement de haute disponibilité, il est necéssaire d'avoir au moins deux **Contrôleurs** dans le cluster.

Il existe deux méthodes pour gérer les instances d'Etcd dans le cas de la haute disponiblité :

- **Stacked** Etcd

- **External** Etcd

Stacked Etcd

Dans ce cas, chaque Contrôleur possède sa propre instance d'Etcd qui communiquent entre elles :

> **Important** : Cette architecture est celle utilisée par défaut par la commande **kubeadm** lors de la création d'un cluster.

External Etcd

Dans ce cas, chaque instance d'Etcd est placée sur un serveur différent qui n'est pas un Contrôleur :

> **Important** : Pour plus d'information concernant la mise en place d'un cluster en utilisant Stacked Etcd ou External Etcd, consultez **cette page** de la documentation officielle de K8s.

LAB #2 - Création du Cluster Kubernetes avec Minikube

2.1 - Présentation de Minikube

Pour installer Kubernetes rapidement et facilement il convient d'utiliser Minikube. Minikube permet de créer un cluster avec un seul nœud.

2.2 - Installation de Minikube

Revenez dans votre Gateway et téléchargez 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
```

Renommez le binaire téléchargé et rendez-le exécutable :

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

Déplacez ensuite le binaire minikube dans le répertoire **/usr/local/bin/** :

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

Important : Demandez le mot de passe de l'utilisateur **root** à votre formateur.

Testez ensuite l'installation avec la commande **minikube version** :

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

2.3 - Configuration de Minikube

Configurez maintenant l'hyperviseur par défaut de minikube :

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

Vérifiez la prise en charge de la dernière commande :

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

Par défaut, lors de démarrage de Minikube, celui-ci va allouer 2 vCPUs et 2Go de RAM à sa machine virtuelle. Augmentez la quantité de mémoire qui sera allouée avec la commande suivante :

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

Vérifiez la prise en charge de la dernière commande :

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

2.4 - Installation de Docker

Minikube a besoin de Docker pour fonctionner.

Présentation de Docker

Docker est une application de virtualisation légère qui utilise des **images** et des **conteneurs**.

Une **image** est un paquet exécutable contenant tout ce qu'il est nécessaire afin d'exécuter un logiciel donné, incluant :

- le code
- un runtime
- des bibliothèques,
- des variables d'environnement
- des fichiers de configuration

Un **conteneur** est une instance de l'image en cours d'exécution en mémoire. Elle est isolée de l'environnement de l'hôte par défaut mais peut accéder à des fichiers et de ports de l'hôte selon la configuration.

Les conteneurs exécutent des applications nativement en utilisant le noyau de la machine hôte. De ce fait les performances d'un conteneur sont supérieures à celles d'une machine virtuelle qui doit passer par un hyperviseur pour accéder aux ressources de la machine hôte :

Docker existe en deux versions **Docker-CE** (Docker Community Edition) et **Docker-EE** (Docker Enterprise Edition). Pour consulter les différences entre les deux versions, consultez le lien **https://docs.docker.com/engine/installation/**.

Installer docker

Docker n'est pas dans le dépôts de Debian. Afin de l'installer il convient d'ajouter le dépôt de docker. Premièrement, il est nécessaire d'installer les paquets permettant à Debian d'utiliser un dépôt en https :

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

Téléchargez la clef GPG officielle de docker :

```
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
```

Ajoutez le dépôt **stable** de docker :

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

> **Important** - Notez que la commande **lsb_release -cs** retourne le nom de la distribution Debian, à savoir dans ce cas **stretch**.

Installez maintenant le paquet **docker-ce** :

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

Dernièrement, vérifiez la version de Docker client et serveur :

```
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 - Notez que le paquet docker-ce a besoin des paquets **containerd.io** et **docker-ce-cli**. Notez aussi que la procédure ci-dessus installe la version la plus récente de Docker.

2.5 - Installation de kubectl

Démarrez maintenant 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
```

Notez l'erreur **kubectl not found.**. Exécutez donc la commande **minikube kubectl – get pods -A** pour installer 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
```

Consultez la liste des machines virtuelles en cours d'exécution :

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

Arrêtez maintenant Minikube :

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

Notez que, bien qu'arrêtée, la machine virtuelle **minikube** est toujours présente :

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

Démarrez de nouveau 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
```

```
  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
```

Contrôlez la version de kubectl qui a été installée :

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

Important : La sortie de cette commande indique une version 1.31.0 de Kubernetes.

La version de kubectl installée par minikube se trouve dans le répertoire **/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
```

Pour une utilisation plus facile, copiez la commande vers le répertoire **/usr/local/bin/** :

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

Vérifiez ensuite que la commande est disponible :

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

2.6 - La Commande minikube addons

Minikube utilise des modules. Ces modules sont appelés des **addons**. Pour consulter les addons installés ainsi que leurs statuts, utilisez la commande **minikube addons list** :

```
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)       |
|-----------------------------|----------|--------------|--------------------------------|
```

Pour activer le module **metrics-server**, utilisez la commande **minikube addons enable** :

```
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
```

Vérifiez maintenant la prise en compte de la commande précédente :

```
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 - La Commande minikube dashboard

Activez le module **dashboard** :

```
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
```

Connectez-vous à l'interface graphique de votre Gateway.

Minikube embarque l'application **Kubernetes Dashboard**. Lancez la commande **minikube dashboard** dans un terminal graphique. Le navigateur Firefox sera lancé vous donnant accès à Kubernetes Dashboard.

Revenez à la fenêtre de connexion en SSH.

Retournez maintenant dans la VM **kubemaster** :

```
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:~$
```

Devenez root et vérifiez l'état des noeuds du cluster :

```
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
```

Vérifiez ensuite l'état des 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
```
