Minikube est une solution permettant de faire tourner un cluster Kubernetes en local, soit dans des machines virtuelles, soit dans des containers Docker.

1. Récupération du binaire

Depuis le lien suivant https://github.com/kubernetes/minikube/releases, vous trouverez la dernière release de Minikube et la procédure d'installation en fonction de votre environnement.

- si vous êtes sur macOS:

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-d
$ chmod +x minikube
$ sudo mv minikube /usr/local/bin/
```

- si vous êtes sur Linux:

```
$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-l
$ chmod +x minikube
$ sudo mv minikube /usr/local/bin/
```

- si vous êtes sur Windows:

```
$ curl -Lo minikube.exe https://storage.googleapis.com/minikube/releases/latest/miniku
```

Il faudra ensuite ajouter minikube.exe dans votre PATH.

:fire: sur Windows, je vous recommande d'utiliser Git Bash afin d'avoir accès à un shell proche du shell Linux et de pouvoir notamment utiliser *curl* et d'autres commandes bien pratiques.

2. Choix du driver

Selon ce qui est installé sur votre machine locale, Minikube détectera automatiquement le driver à utiliser parmis ceux disponibles:

- virtualbox
- parallels
- vmwarefusion
- hyperkit
- vmware
- docker
- podman (expérimental)

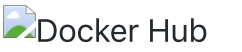
Minikube vous permettra également de spécifier explicitement le driver que vous souhaitez utiliser.

Utilisation du driver *docker*

Si vous souhaitez que Minikube lance un cluster dans des containers, il est nécessaire d'installer Docker sur votre machine.

- Si vous êtes sur MacOS ou Windows, vous pouvez utiliser *Docker Desktop* une solution très bien intégrée à ces environnements.

Depuis le DockerHub, vous trouverez les instructions nécessaires pour son installation.

 Docker Hub

- Si vous êtes sur Linux, vous pouvez simplement lancer le script suivant:

```
$ curl -sSL https://get.docker.com | sh
```

Utilisation d'un hypervieur

Si vous souhaitez que Minikube lance un cluster dans des machines virtuelles il est nécessaire d'installer un hyperviseur sur votre machine locale. En fonction de l'OS, différents hyperviseurs sont supportés:

- si vous êtes sur macOS, vous pouvez utiliser l'un des hyperviseurs suivants:
  - VirtualBox (https://www.virtualbox.org/wiki/Downloads)
  - VMware Fusion (https://www.vmware.com/products/fusion)
  - HyperKit (https://github.com/moby/hyperkit)
- si vous êtes sur Linux, vous pouvez utiliser l'un des hyperviseurs suivants:
  - VirtualBox (https://www.virtualbox.org/wiki/Downloads)
  - KVM (http://www.linux-kvm.org/)

  Note: Minikube supporte également une option --vm-driver=none qui exécute les composants Kubernetes sur la machine hôte et non dans une VM. L'utilisation de ce pilote nécessite Docker et un environnement Linux mais pas d'hyperviseur.
- si vous êtes sur Windows, vous pouvez utiliser l'un des hyperviseurs suivants:
  - VirtualBox (https://www.virtualbox.org/wiki/Downloads)
  - Hyper-V (https://msdn.microsoft.com/en-us/virtualization/hyperv_on_windows/quick_start/walkthrough_install)

3. Lancement

:fire: de nombreuses options de lancement sont disponible (vous pouvez lister celles-ci avec la commande `minikube start --help`. Par exemple, si vous souhaitez spécifier le driver à utiliser, vous pouvez le faire grace à l'option *--driver* lors du lancement.

Pour lancer Minikube:

- si vous êtes sur macOS ou Linux

```
$ minikube start [options]
```

- si vous êtes sur Windows

```
$ ./minikube.exe start [options]
```

Exemple: cluster avec un seul node

La commande suivante permet de lancer Minikube en utilisant le driver détecté automatiquement (*docker* dans cet exemple). Le cluster kubernetes ainsi mis en place tournera dans un seul container.

```
$ minikube start
😄  minikube v1.16.0 on Darwin 11.0.1
✨  Automatically selected the docker driver. Other choices: hyperkit, virtualbox
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
💾  Downloading Kubernetes v1.20.0 preload ...
    > preloaded-images-k8s-v8-v1....: 491.00 MiB / 491.00 MiB  100.00% 17.88 Mi
🔥  Creating docker container (CPUs=2, Memory=2947MB) ...
🐳  Preparing Kubernetes v1.20.0 on Docker 20.10.0 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace
```

Assurez-vous d'avoir installé le binaire *kubectl* (cela a normalement été effectué dans un exercice précédent). *kubectl* est indispensable pour communiquer avec Kubernetes depuis la ligne de commande. Lancez ensuite la commande suivante afin de lister les nodes du cluster:

```
$ kubectl get node
NAME       STATUS   ROLES                  AGE   VERSION
```

```
minikube   Ready    control-plane,master   40m   v1.20.0
```

Exemple: cluster avec 3 nodes

La commande suivante permet de lancer un cluster de 3 nodes en spécifiant le driver *docker*.

```
$ minikube start --driver docker --nodes 3
😄  minikube v1.16.0 on Darwin 11.0.1
✨  Using the docker driver based on user configuration
👍  Starting control plane node minikube in cluster minikube
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.20.0 on Docker 20.10.0 .../ 🔗  Configuring CNI (Container

    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, default-storageclass

👍  Starting node minikube-m02 in cluster minikube
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🌐  Found network options:
    ▪ NO_PROXY=192.168.49.2
🐳  Preparing Kubernetes v1.20.0 on Docker 20.10.0 ...
    ▪ env NO_PROXY=192.168.49.2
🔎  Verifying Kubernetes components...

👍  Starting node minikube-m03 in cluster minikube
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🌐  Found network options:
    ▪ NO_PROXY=192.168.49.2,192.168.49.3
🐳  Preparing Kubernetes v1.20.0 on Docker 20.10.0 ...
    ▪ env NO_PROXY=192.168.49.2
    ▪ env NO_PROXY=192.168.49.2,192.168.49.3
🔎  Verifying Kubernetes components...
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace
```

Comme précédemement, les nodes peuvent être listés avec la commande suivante:

```
$ kubectl get no
NAME           STATUS   ROLES                  AGE   VERSION
minikube       Ready    control-plane,master   99s   v1.20.0
minikube-m02   Ready    <none>                 56s   v1.20.0
minikube-m03   Ready    <none>                 21s   v1.20.0
```

4. Suppression

Afin de détruire un cluster mis en place avec Minikube, il suffit de lancer la commande suivante:

```
$ minikube delete
🔥  Deleting "minikube" in docker ...
🔥  Deleting container "minikube" ...
🔥  Deleting container "minikube-m02" ...
🔥  Removing /Users/[user]/.minikube/machines/minikube ...
💀  Removed all traces of the "minikube" cluster.
```