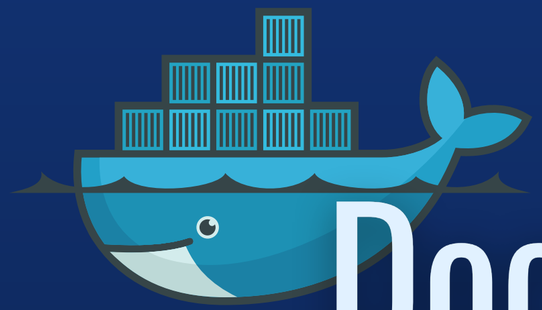


Docker & Kubernetes

- Installation et configuration
- Surveillance des instances d’une application

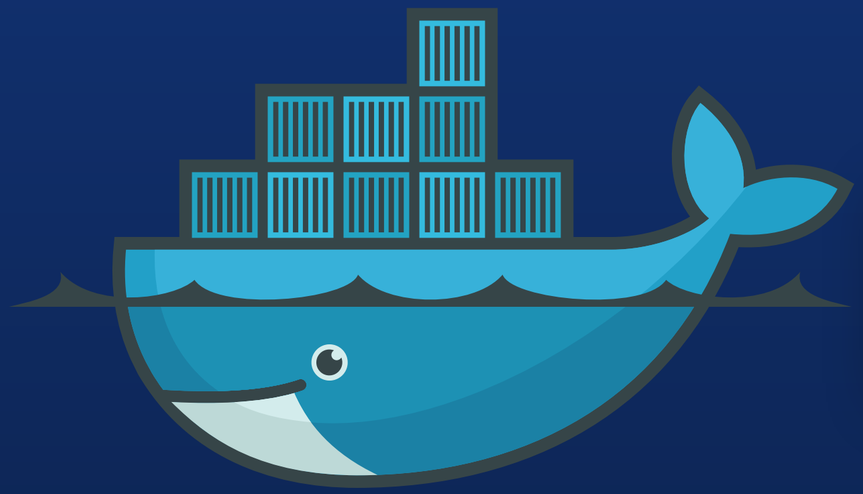
Préparé par :

- El Khoukh Marwane
- El Mourabet Nada
- El Hanoudi Maha
- El Hauari Mohamed

Supervisé par :

- Professeur El Bouhdidi Jaber

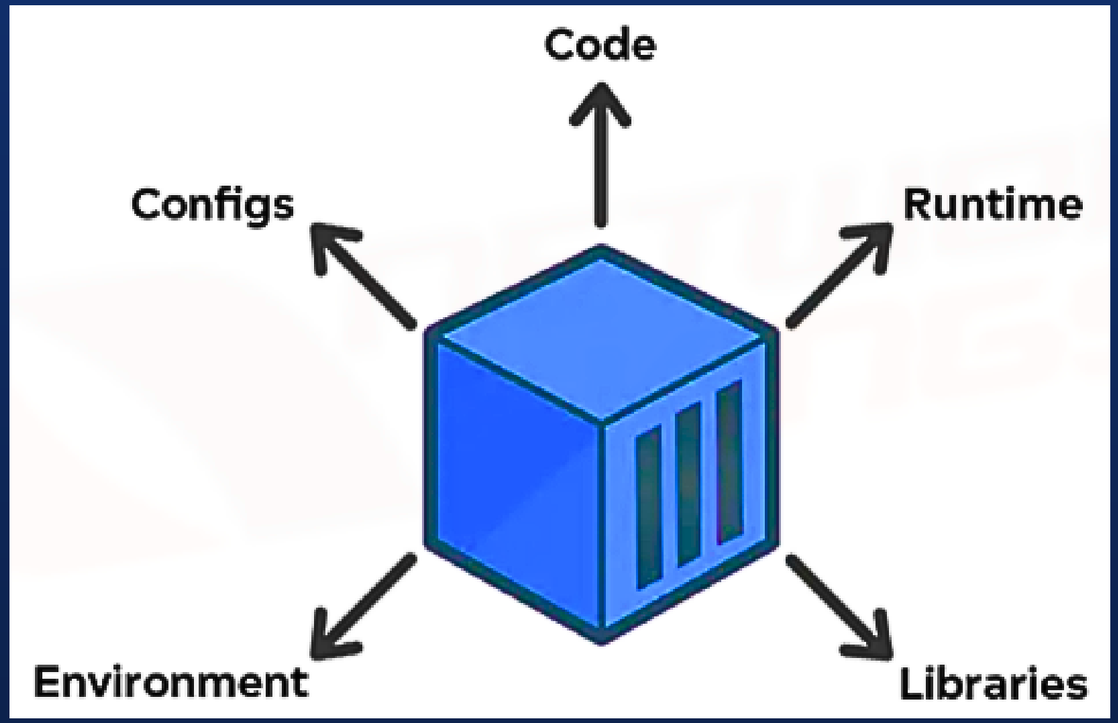
Table de matières



Docker

C’est quoi Docker?

- Docker est une plateforme open source qui facilite aux développeurs de construire, déployer, exécuter, mettre à jour et gérer des conteneurs.
- Cahaue conteneur peut contenir une ou plusieurs applications qu’il va étre isolée des autres conteneurs.



La conteneurisation

La conteneurisation est un concept de plus en plus populaire dans le monde du développement logiciel. Bien plus léger que les traditionnelles machines virtuelles, cette technique permet de regrouper et de gérer les applications et leurs dépendances dans des conteneurs séparés et isolés les uns des autres. Il faut s’imaginer la conteneurisation comme un système de couches. Avec en premier lieu, l’OS, puis les fonctionnalités nécessaires.

C’est quoi un conteneur ?

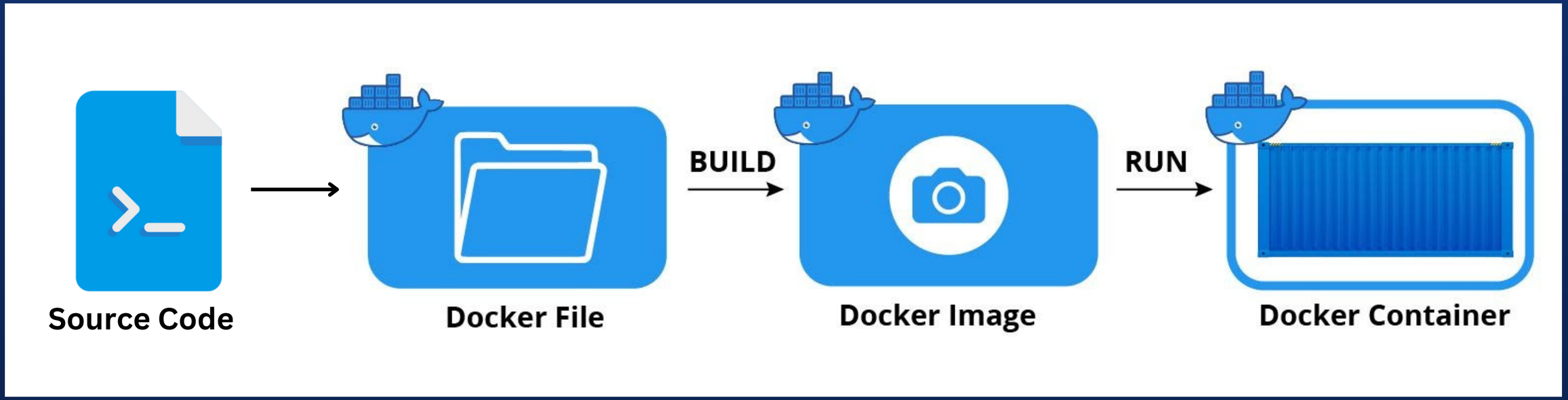
Un conteneur est une entité logicielle compacte qui inclut une application ainsi que tous les éléments indispensables à son fonctionnement. Ils offrent un environnement d'exécution isolé et facilement transportable, garantissant aux applications un fonctionnement uniforme et fiable, quel que soit le contexte de déploiement.

Les composants d’un conteneur

- Le code source de notre programme
- Les bibliothèques que l ’application nécessite pour bien fonctionner
- La distribution de l’os

Ces composants vont étre rassemblés dans un fichier s’appelle **IMAGE FILE**

Création d’image Docker



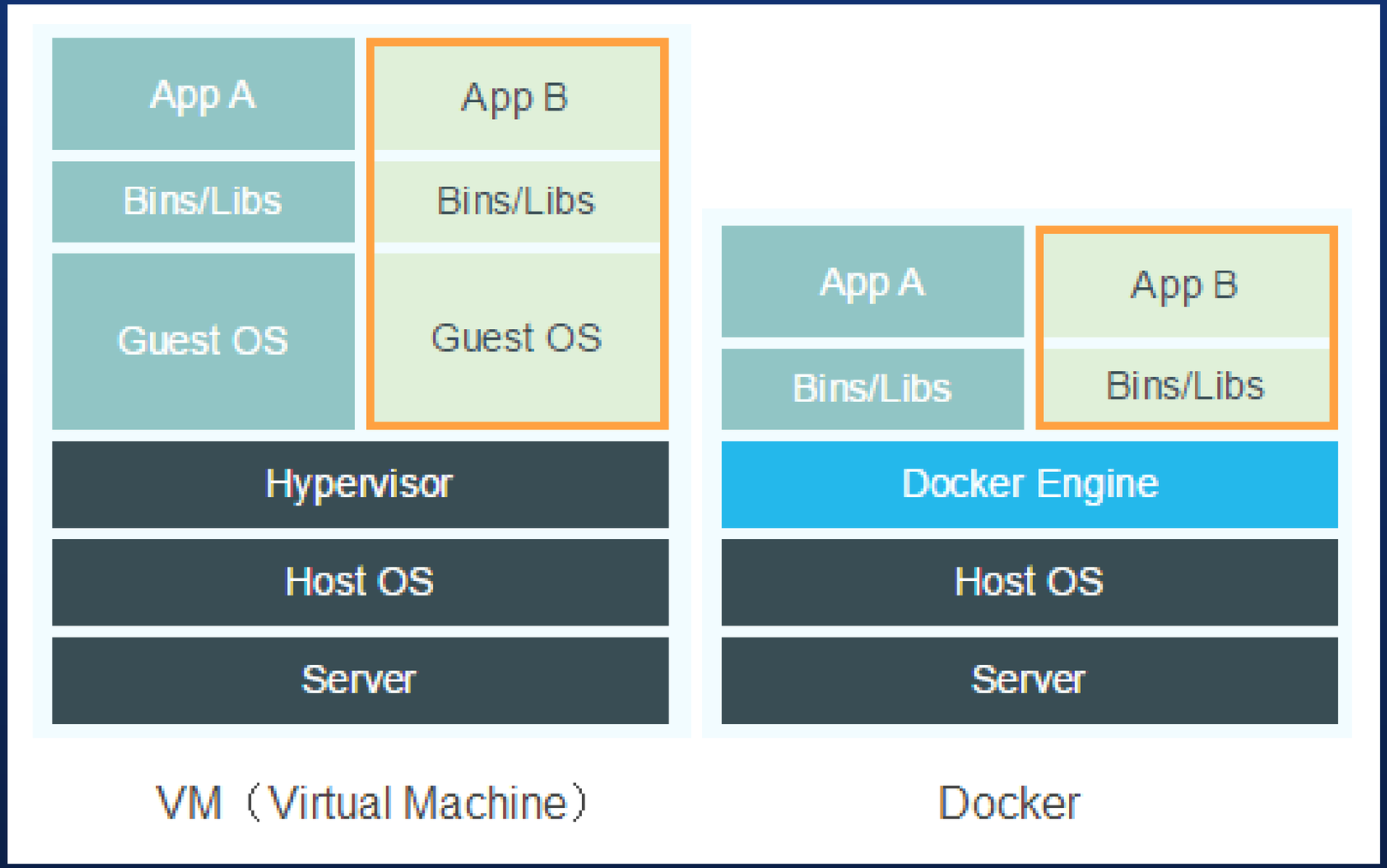
Dockerfile

```
FROM ubuntu:21.04

RUN install nodejs

COPY . ./app

RUN npm install
```

• Un Dockerfile est un fichier texte qui contient des instructions permettant de construire une image Docker.

• Ce fichier spécifie les étapes nécessaires pour assembler une image Docker qui contient tout ce dont une application a besoin pour s'exécuter, y compris le code source de l'application, les dépendances, les bibliothèques et les paramètres de configuration.

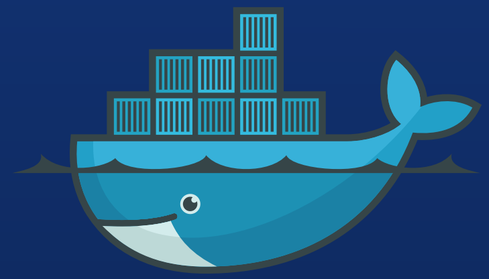
VM vs Docker


App A
App B
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Hypervisor
Host OS
Server
VM（Virtual Machine）
App A
App B
Bins/Libs
Bins/Libs
Docker Engine
Host OS
Server
Docker

VM vs Docker

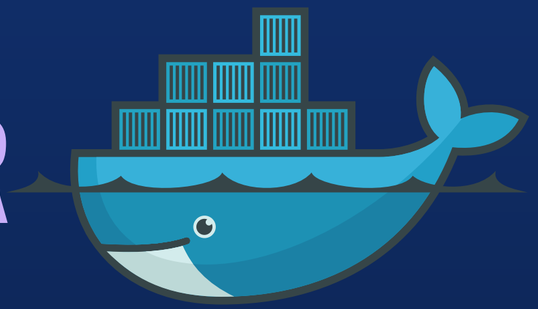
Caractéristique	Machines Virtuelles (VMs)	Conteneurs (Docker)
Type	Poids Lourd	Poids Léger
Performance	Performances limitées	Performances natives
Système d'Exploitation	VMs ont leur propre OS	Conteneurs partagent l'OS de l'hôte
Virtualisation	Virtualisation matérielle	Virtualisation de l'OS
Temps de Démarrage	Minutes	Millisecondes
Allocation Mémoire	Allocation fixe par VM	Allocation dynamique selon besoin
Isolation Sécuritaire	Entièrement isolé et plus sûr	Isolation au niveau du processus

On déduit que docker est :



- Plus léger
- Plus rapide dans l’éxécution
- Moins consommateur de ressources systéme

INSTALLATION DE DOCKER



Mettre à jour votre liste de packages existante :

```
nada@nada-virtual-machine:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.4 LTS
Release:        22.04
Codename:       jammy
nada@nada-virtual-machine:~$ sudo apt-get update
[sudo] password for nada:
Hit:1 http://ma.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://ma.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:4 http://ma.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://ma.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,458 kB]
Get:6 http://ma.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [585 kB]
Get:7 http://ma.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,054 kB]
Get:8 http://ma.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [693 kB]
Fetched 4,019 kB in 15s (273 kB/s)
Reading package lists... Done
```

Installer Docker:

sudo apt install docker.io

```
nada@nada-virtual-machine:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  linux-headers-6.2.0-26-generic linux-hwe-6.2-headers-6.2.0-26
  linux-image-6.2.0-26-generic linux-modules-6.2.0-26-generic
  linux-modules-extra-6.2.0-26-generic
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  bridge-utils containerd git git-man liberror-perl pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap
  docker-doc rinse zfs-fuse | zfsutils git-daemon-run | git-daemon-sysvinit
  git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  bridge-utils containerd docker.io git git-man liberror-perl pigz runc
  ubuntu-fan
0 upgraded, 9 newly installed, 0 to remove and 1 not upgraded.
Need to get 73.5 MB of archives.
After this operation, 287 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ma.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://ma.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://ma.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1~22.04.2 [4,267 kB]
Get:4 http://ma.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1~22.04.1 [36.0 MB]
35% [4 containerd 21.2 MB/36.0 MB 59%]                    35% [4 containerd 21.4 MB/36.0 MB 59%] 35% [4 containerd 21
36% [4 containerd 22.9 MB/36.0 MB 63%]                                                  40% [4 c40% Get:5 http://ma.archive.ubun
tu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.5-0ubuntu1~22.04.1 [28.9 MB]
```

Vérifier l’activation de Docker:

```
nada@nada-virtual-machine:~$ docker --version
Docker version 26.0.0, build 2ae903e
nada@nada-virtual-machine:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset>
     Active: active (running) since Sat 2024-03-30 16:51:15 +00; 2 days ago
TriggeredBy: ● docker.socke
       Docs: https://docs.d      .com
   Main PID: 9692 (dockerd)
      Tasks: 41
     Memory: 31.6M
        CPU: 2min 12.238s
     CGroup: /system.slice/docker.service
             ├─ 9692 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/con>
             ├─ 9825 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-po>
             ├─ 9836 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 55>
             ├─10046 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-po>
             └─10054 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 80>

11:12:39 02 أبريل nada-virtual-machine dockerd[9692]: time="2024-04-02T11:12:39>
11:12:39 02 أبريل nada-virtual-machine dockerd[9692]: time="2024-04-02T11:12:39>
11:12:41 02 أبريل nada-virtual-machine dockerd[9692]: time="2024-04-02T11:12:41>
```

Vérifier l’activation de Docker:

```
nada@nada-virtual-machine:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d000bc569937abbe195e20322a0bde6b2922d805332fd6d8a68b19f524b7d21d
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
```

Kubernetes

Kubernetes, qu’est-ce que c’est ?

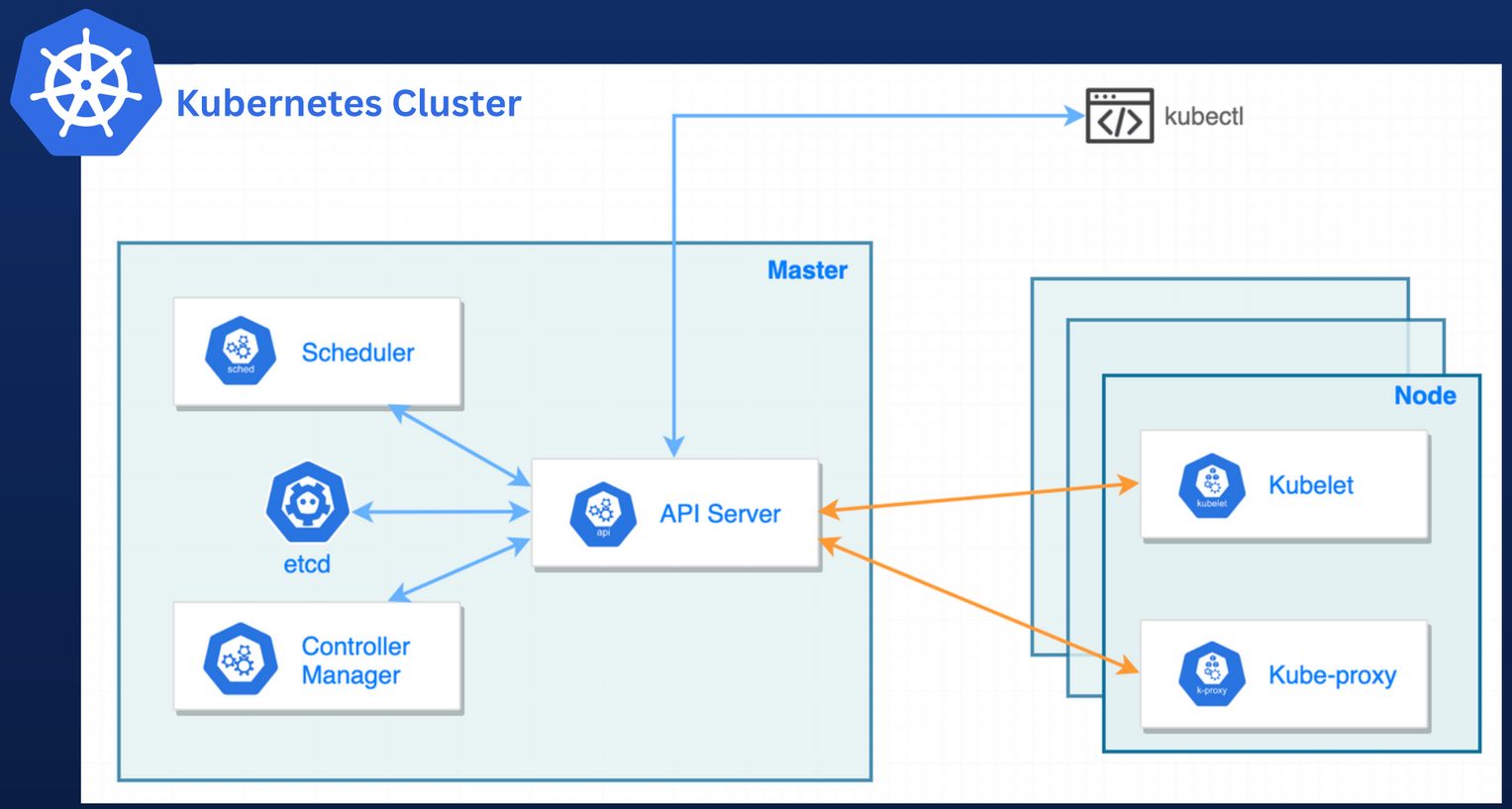
- Kubernetes “**k8s**”,vient du grec, signifiant "**gouvernail**" ou "**pilote**", est un système **Open Source** qui exécute et coordonne des applications conteneurisées dans des clusters. **Kubernetes** gère le cycle complet de vie des applications et des services permettant ainsi de limiter le nombre de processus nécessaire au déploiement et à la mise à l'échelle des applications conteneurisées.

Kubernetes et Docker

- **Docker** simplifie la virtualisation via des conteneurs, aide à gérer les variables d'environnement et à créer des images Docker. d’un autre coté **kubernetes** agit comme un système d'orchestration qui renforce la résilience des applications s'exécutant sur Docker Il permet de regrouper plusieurs serveurs Docker en un cluster, soutient des stratégies de déploiement et assure l'isolation des comptes de service pour une sécurité renforcée.

Architecture de Kubernetes

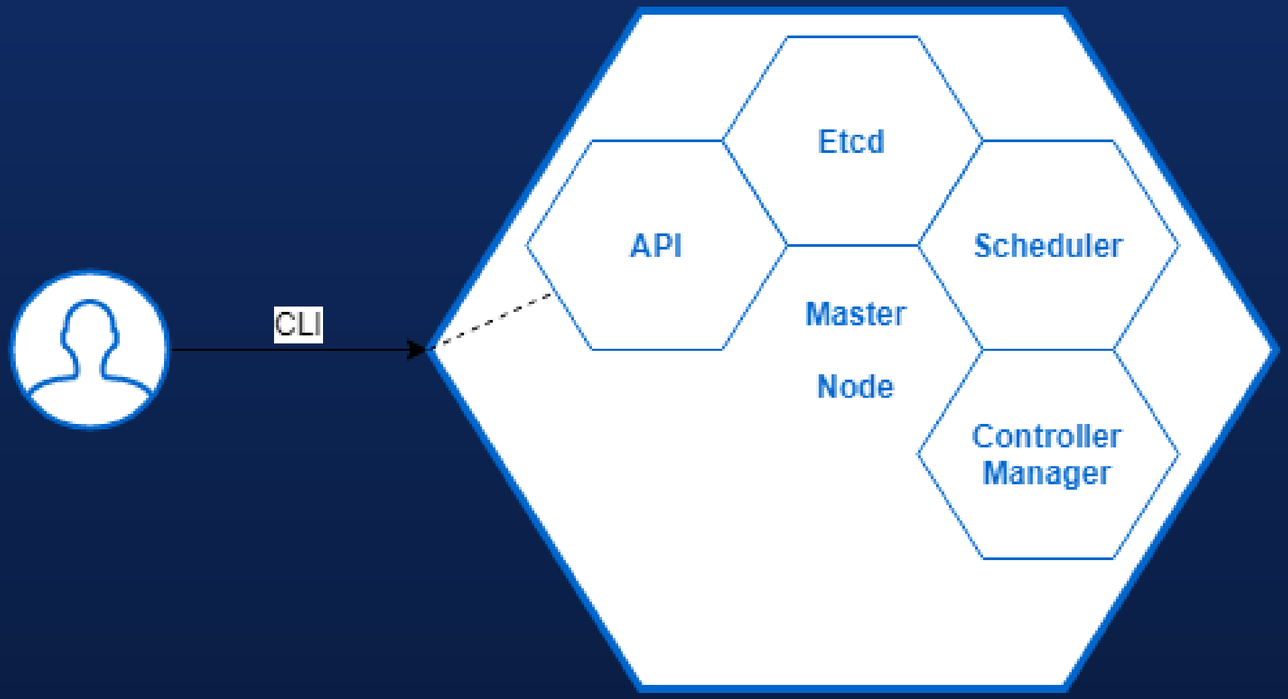
- **Kubernetes** fonctionne avec une architecture de cluster composée de **nœuds maîtres** et de **nœuds de travail**. Les nœuds maîtres sont responsables de la coordination et de la gestion des opérations dans le cluster, tandis que les nœuds de travail exécutent les charges de travail des applications.



Nœuds Maîtres

- Les **nœuds maîtres** sont le **cœur du cluster Kubernetes**. Ils prennent en charge la planification des tâches, la répartition des charges de travail sur les nœuds de travail, la gestion de l'état des applications et la prise de décisions pour garantir la disponibilité et la fiabilité du cluster.

- Les composants principaux du **Master Node** incluent : API - Etcd - Scheduler - Controller Manager

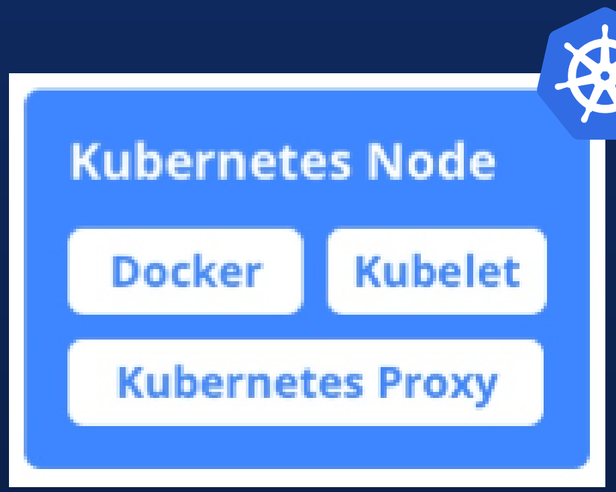


Nœuds de Travail

- Les **nœuds de travail** sont **les machines virtuelles** ou **physiques** qui exécutent les pods, qui sont les unités de base de Kubernetes. Chaque pod peut contenir un ou plusieurs conteneurs et est responsable de l'exécution d'une instance spécifique de l'application.

- Les composants principaux d'un worker node dans un cluster Kubernetes sont :

1. **Kubelet :** L'agent sur le worker node qui gère les pods.
2. **Kube-proxy :** Service réseau pour la connectivité des pods.
3. **Container Runtime :** Logiciel permettant d'exécuter les conteneurs (comme Docker).



Fonctionnement de Kubernetes

- Le fonctionnement de Kubernetes repose sur trois étapes essentielles pour la gestion efficace des applications conteneurisées. Ces étapes incluent la définition des ressources, la planification et le déploiement des applications, ainsi que la gestion du cluster.

1. Création des Ressources

- Utilisation de fichiers de configuration YAML pour décrire les ressources telles que les pods, les services, les déploiements, etc.
- Envoi de ces fichiers à l'API Server de Kubernetes, qui les valide et les enregistre dans le cluster.

Fichier YAML

Le YAML est un langage de sérialisation de données utilisé dans Kubernetes pour décrire la configuration des ressources telles que les pods, les services, les déploiements, etc.

```yaml

apiVersion: v1      # La version de l'API Kubernetes utilisée
kind: Pod           # Le type d'objet Kubernetes
metadata:           # Contient les métadonnées du Pod
  name: my-app            # Nom du Pod
  labels:                 # Étiquettes attribuées au Pod
    app: my-app           # Étiquette de l'application
spec:
  containers:             # Liste des conteneurs à exécuter dans le Pod
  - name: my-app-container      # Nom du conteneur
    image: nginx:latest         # Image Docker utilisée pour le conteneur
    ports:
    - containerPort: 80         # Port exposé par le conteneur

```

2. Planification et Déploiement

- Le Scheduler de Kubernetes assigne les pods à des nœuds de travail disponibles selon différents critères.
- Le Kubelet, présent sur chaque nœud, déploie les pods qui lui sont affectés en créant et exécutant les conteneurs correspondants.

3. Gestion et Surveillance

- Kubernetes surveille en permanence l'état des pods et des nœuds pour s'assurer qu'ils fonctionnent correctement.
- La mise à l'échelle automatique ajuste automatiquement le nombre de pods en fonction de la charge de travail.
- Kubernetes fournit également des fonctionnalités de surveillance et de journalisation pour aider à diagnostiquer les problèmes et à surveiller les performances du cluster.

Installation de Kubernetes

Installation kubernetes (microk8s) : “sudo snap install microk8s --classic”

```
elmourabet@elmourabet-virtual-machine:~$ sudo snap install microk8s --classic
microk8s (1.28/stable) v1.28.7 from Canonical✓ installed
```

- **MicroK8s** est une distribution **légère** de Kubernetes. Elle est conçue pour les développeurs et les utilisateurs qui veulent mettre en place rapidement et facilement un cluster Kubernetes à nœud unique pour des tests, du développement ou des déploiements de petite échelle.

Vérification de l’installation : “sudo microk8s status --wait-ready”

```
marwane@marwane-virtual-machine:~$ sudo microk8s status --wait-ready


microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
  enabled:
    dns                  # (core) CoreDNS
    ha-cluster           # (core) Configure high availability on the current node
    helm                 # (core) Helm - the package manager for Kubernetes
    helm3                # (core) Helm 3 - the package manager for Kubernetes
  disabled:
    cert-manager         # (core) Cloud native certificate management
    cis-hardening        # (core) Apply CIS K8s hardening
    community            # (core) The community addons repository
    dashboard            # (core) The Kubernetes dashboard
    gpu                  # (core) Alias to nvidia add-on
    host-access          # (core) Allow Pods connecting to Host services smoothly
```

Activer DNS : “sudo microk8s enable dns”

```
marwane@marwane-virtual-machine:~$ sudo microk8s enable dns
[sudo] password for marwane:
Infer repository core for addon dns
Addon core/dns is already enabled
```

- Activer la fonctionnalité DNS pour fournir la découverte de services et la communication entre les différents composants et applications s'exécutant au sein du cluster.

On va activer un registre Docker privé au sein d'un cluster MicroK8s
En utilisant **sudo microk8s enable registry**

```
marwane@marwane-virtual-machine:~$ sudo microk8s enable registry
Infer repository core for addon registry
Infer repository core for addon hostpath-storage
Enabling default storage class.
WARNING: Hostpath storage is not suitable for production environments.
         A hostpath volume can grow beyond the size limit set in the volume claim manifest.

deployment.apps/hostpath-provisioner created
storageclass.storage.k8s.io/microk8s-hostpath created
serviceaccount/microk8s-hostpath created
clusterrole.rbac.authorization.k8s.io/microk8s-hostpath created
clusterrolebinding.rbac.authorization.k8s.io/microk8s-hostpath created
Storage will be available soon.
The registry will be created with the size of 20Gi.
Default storage class will be used.
namespace/container-registry created
persistentvolumeclaim/registry-claim created
deployment.apps/registry created
service/registry created
configmap/local-registry-hosting configured
```

Activer Dashboear : “sudo microk8s enable dashdoard”

```
marwane@marwane-virtual-machine:~$ sudo microk8s enable dashboard
Infer repository core for addon dashboard
Enabling Kubernetes Dashboard
Infer repository core for addon metrics-server
Enabling Metrics-Server
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
clusterrolebinding.rbac.authorization.k8s.io/microk8s-admin created
Metrics-Server is enabled
```

On va activer et configurer le tableau de bord Kubernetes dans un cluster MicroK8s

on demande le cluster Kubernetes géré par MicroK8s d'obtenir des informations sur toutes les ressources dans tous les espaces de noms.

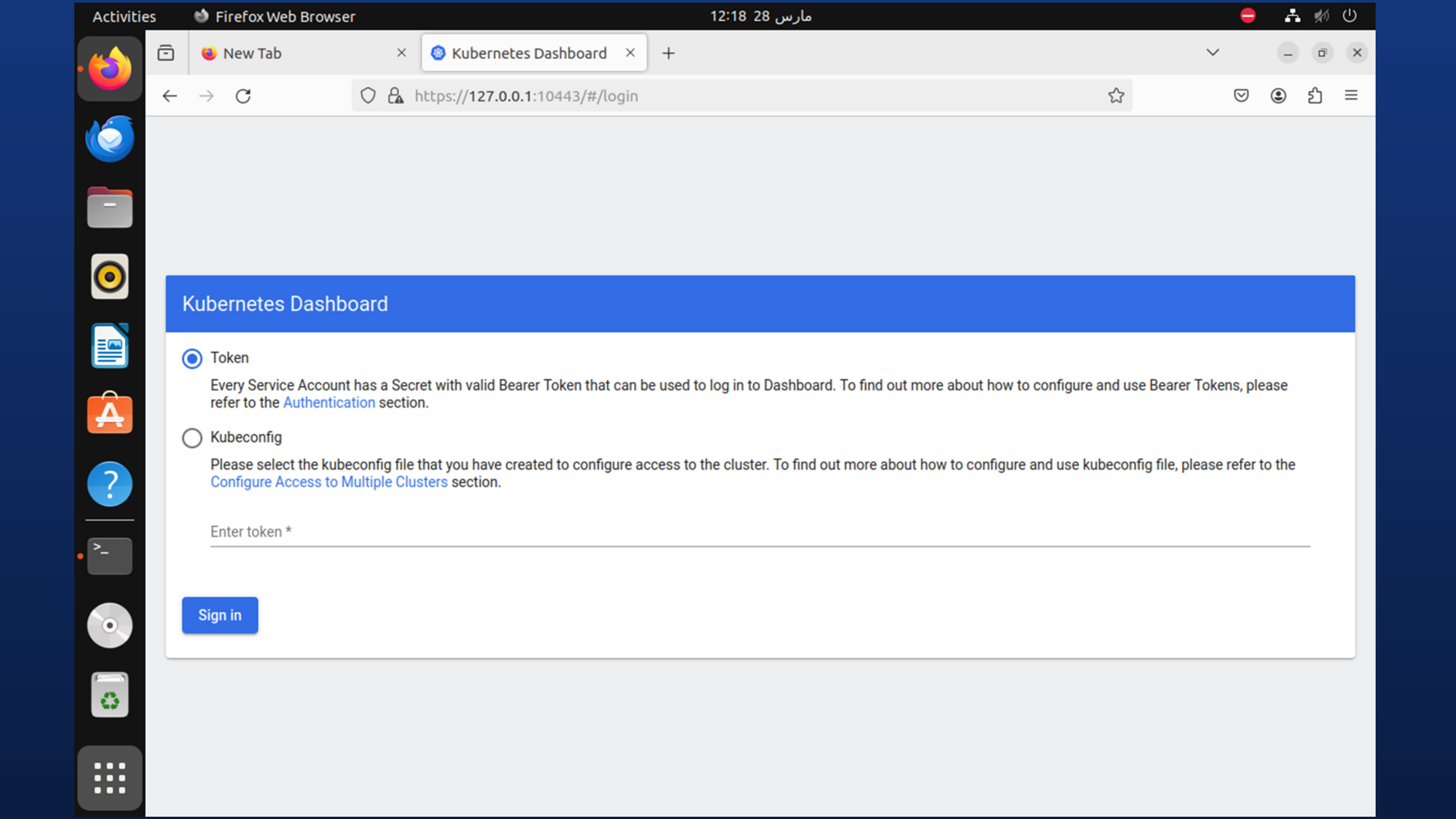
```
marwane@marwane-virtual-machine:~$ sudo microk8s kubectl get all --all-namespaces
NAMESPACE            NAME                                         READY   STATUS              RESTARTS        AGE
container-registry   pod/registry-6c9fcc695f-qpw8p                0/1     Pending             0               2m35s
kube-system          pod/calico-kube-controllers-77bd7c5b-sdpwv   1/1     Running             1               37m
kube-system          pod/calico-node-ct8kt                        1/1     Running             1 (6m49s ago)   37m
kube-system          pod/coredns-864597b5fd-ts8hl                 1/1     Running             1 (6m49s ago)   37m
kube-system          pod/dashboard-metrics-scraper-5657497c4c-86wjb   0/1     ContainerCreating   0               99s
kube-system          pod/hostpath-provisioner-756cd956bc-grq72    0/1     ContainerCreating   0               2m37s
kube-system          pod/kubernetes-dashboard-54b48fbf9-mlvp4     0/1     ContainerCreating   0               99s
kube-system          pod/metrics-server-848968bdcd-xdsbh          0/1     ContainerCreating   0               102s

NAMESPACE            NAME                                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
container-registry   service/registry                    NodePort    10.152.183.40    <none>        5000:32000/TCP           2m35s
default              service/kubernetes                  ClusterIP   10.152.183.1     <none>        443/TCP                  37m
kube-system          service/dashboard-metrics-scraper   ClusterIP   10.152.183.186   <none>        8000/TCP                 99s
kube-system          service/kube-dns                    ClusterIP   10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   37m
kube-system          service/kubernetes-dashboard        ClusterIP   10.152.183.233   <none>        443/TCP                  99s
kube-system          service/metrics-server              ClusterIP   10.152.183.225   <none>        443/TCP                  102s

NAMESPACE     NAME                             DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   daemonset.apps/calico-node       1         1         1       1            1           kubernetes.io/os=linux   37m

NAMESPACE            NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
container-registry   deployment.apps/registry              0/1     1            0           2m35s
kube-system          deployment.apps/calico-kube-controllers   1/1     1            1           37m
kube-system          deployment.apps/coredns               1/1     1            1           37m
kube-system          deployment.apps/dashboard-metrics-scraper   0/1     1            0           99s
kube-system          deployment.apps/hostpath-provisioner   0/1     1            0           2m37s
kube-system          deployment.apps/kubernetes-dashboard   0/1     1            0           99s
kube-system          deployment.apps/metrics-server        0/1     1            0           102s

NAMESPACE            NAME                                             DESIRED   CURRENT   READY   AGE
container-registry   replicaset.apps/registry-6c9fcc695f              1         1         0       2m35s
kube-system          replicaset.apps/calico-kube-controllers-77bd7c5b   1         1         1       37m
kube-system          replicaset.apps/coredns-864597b5fd               1         1         1       37m
kube-system          replicaset.apps/dashboard-metrics-scraper-5657497c4c   1         1         0       99s
kube-system          replicaset.apps/hostpath-provisioner-756cd956bc   1         1         0       2m37s
kube-system          replicaset.apps/kubernetes-dashboard-54b48fbf9   1         1         0       99s
kube-system          replicaset.apps/metrics-server-848968bdcd        1         1         0       102s
marwane@marwane-virtual-machine:~$
```

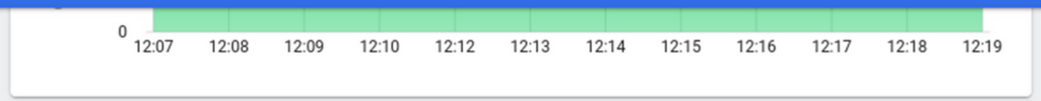
Pour accéder à l'interface utilisateur du tableau de bord Kubernetes lors du travail avec un cluster MicroK8s, on utilise: sudo microk8s dashboard-proxy

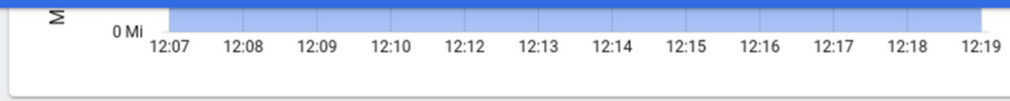
1- l'IP address

2-le token

```
vm1@vm1:~$ sudo microk8s dashboard-proxy
Checking if Dashboard is running.
Infer repository core for addon dashboard
Waiting for Dashboard to come up.
Trying to get token from microk8s-dashboard-token
Waiting for secret token (attempt 0)
Dashboard will be available at https://127.0.0.1:10443
Use the following token to login:
eyJhbGciOiJSUzI1NiIsImtpZCI6IkF2dzczaU1xYXNzRFRFX2xnUHJDWndsWTZyUFNocjVPVVBaeTVSOTllSjAifQ.eyJpc3
MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ
rdWJlLXN5c3RlbSIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJtaWNyb2s4cy1kYXNoYm9h
cmQtdG9rZW4iLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC5uYW1lIjoiZGVmYXVsdCIsI
mt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50LnVpZCI6ImZmOTE0MDE1LWViY2ItNDU0My04MD
kxLTk2M2QyZjI3MzQyNyIsInN1YiI6InN5c3RlbTpzZXJ2aWNlYWNjb3VudDprdWJlLXN5c3RlbTpkZWZhdWx0In0.eou-vYA
0WVOcx2jZMzWa_A5aJ0u873mBCz91McyocjOpXF9iFkP1qeGeSxJspYrWAkfqxVknME4HUVGthljgGVzZDxrjZAc6CB48IcP1
U8voLzk3RIN0jou_3iwNo-V0b0L9cFI4R2sj9kluQBdqNrkvuZFexKVLqUNRUiR1b10Bb__CC6hKBb13cyKSY_migspCwj6_m
aDTeNjPcpTUeMfMrJAIVzKI6gQZEUvbuLgVTpMnydfZ6yrMSCd44IbwGU5xkz-jJjDUCnliz966eH_aPlvvI0PDIraslG7-YV
D39x-hqTUKz2qEAcSTZawMeHL5g--7IXGbfIvDR4kpBg
```


Activities
Firefox Web Browser
12:18 28 مارس
New Tab
Kubernetes Dashboard
https://127.0.0.1:10443/#/login
Kubernetes Dashboard
Token
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.
Kubeconfig
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.
Enter token *
Sign in





Pods
Replica Sets
Replication Controllers
Stateful Sets

Service
Ingresses N
Ingress Classes
Services N

Config and Storage
Config Maps N
Persistent Volume Claims N
Secrets N
Storage Classes

Cluster
Cluster Role Bindings
Cluster Roles
Events N
Namespaces
Network Policies N
Nodes
Persistent Volumes
Role Bindings N
Roles N
Service Accounts N

Custom Resource Definitions

Pods

Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created
dashboard-metrics-scraper-5657497c4c-86wjb	kubernetesui/metrics-scraper:v1.0.8	k8s-app: dashboard-metrics-scraper pod-template-hash: 5657497c4c	marwane-virtual-machine	Running	0	4.00m	8.95Mi	26 minutes ago
kubernetes-dashboard-54b48fbf9-mlvp4	kubernetesui/dashboard:v2.7.0	k8s-app: kubernetes-dashboard pod-template-hash: 54b48fbf9	marwane-virtual-machine	Running	0	9.00m	29.09Mi	26 minutes ago
metrics-server-848968bdcd-xdsbh	registry.k8s.io/metrics-server/metrics-server:v0.6.3	k8s-app: metrics-server pod-template-hash: 848968bdcd	marwane-virtual-machine	Running	0	8.00m	16.76Mi	26 minutes ago
hostpath-provisioner-756cd956bc-grq72	cdkbot/hostpath-provisioner:1.5.0	k8s-app: hostpath-provisioner pod-template-hash: 756cd956bc	marwane-virtual-machine	Running	0	4.00m	10.05Mi	27 minutes ago
calico-kube-controllers-77bd7c5b-sdpwv	docker.io/calico/kube-controllers:v3.25.1	k8s-app: calico-kube-controllers pod-template-hash: 77bd7c5b	marwane-virtual-machine	Running	1	3.00m	16.48Mi	an hour ago
coredns-864597b5fd-ts8hl	coredns/coredns:1.10.1	k8s-app: kube-dns pod-template-hash: 864597b5fd	marwane-virtual-machine	Running	1	4.00m	17.23Mi	an hour ago
calico-node-ct8kt	docker.io/calico/node:v3.25.1	controller-revision-hash: 78595976d5 k8s-app: calico-node pod-template-generation: 1	marwane-virtual-machine	Running	1	64.00m	108.92Mi	an hour ago

La commande est utilisée on affiche la liste de tous les pods (conteneurs en cours d'exécution) dans tous les espaces de noms (toutes les zones logiques) d'un cluster Kubernetes géré par MicroK8s. en utilsant

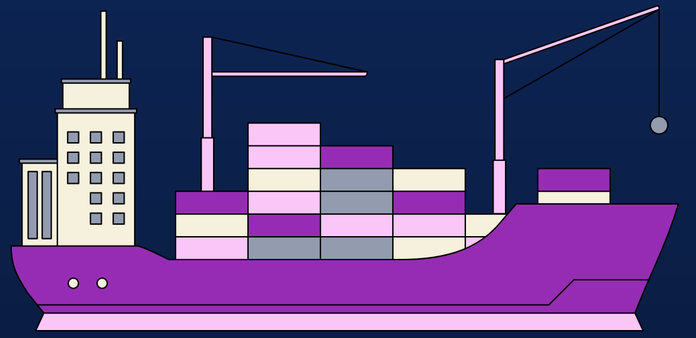
sudo microk8s kubectl get pods --all-namespaces

```
marwane@marwane-virtual-machine:~$ sudo microk8s kubectl get pods --all-namespaces
[sudo] password for marwane:
NAMESPACE           NAME                                         READY   STATUS    RESTARTS      AGE
container-registry  registry-6c9fcc695f-qpw8p                    1/1     Running   0             44m
kube-system         calico-kube-controllers-77bd7c5b-sdpwv       1/1     Running   1             78m
kube-system         calico-node-ct8kt                            1/1     Running   1 (48m ago)   78m
kube-system         coredns-864597b5fd-ts8hl                     1/1     Running   1 (48m ago)   78m
kube-system         dashboard-metrics-scraper-5657497c4c-86wjb   1/1     Running   0             43m
kube-system         hostpath-provisioner-756cd956bc-grq72        1/1     Running   0             44m
kube-system         kubernetes-dashboard-54b48fbf9-mlvp4         1/1     Running   0             43m
kube-system         metrics-server-848968bdcd-xdsbh              1/1     Running   0             43m
marwane@marwane-virtual-machine:~$
```

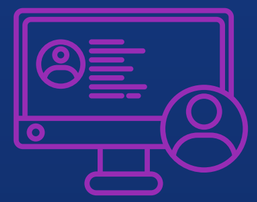
La commande "sudo microk8s kubectl get services --all-namespaces" est utilisée pour obtenir une liste de tous les services (services Kubernetes) déployés dans un cluster Kubernetes géré par MicroK8s.

```
marwane@marwane-virtual-machine:~$ sudo microk8s kubectl get services --all-namespaces
NAMESPACE           NAME                        TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
container-registry  registry                    NodePort    10.152.183.40    <none>        5000:32000/TCP           45m
default             kubernetes                  ClusterIP   10.152.183.1     <none>        443/TCP                  80m
kube-system         dashboard-metrics-scraper   ClusterIP   10.152.183.186   <none>        8000/TCP                 44m
kube-system         kube-dns                    ClusterIP   10.152.183.10    <none>        53/UDP,53/TCP,9153/TCP   80m
kube-system         kubernetes-dashboard        ClusterIP   10.152.183.233   <none>        443/TCP                  44m
kube-system         metrics-server              ClusterIP   10.152.183.225   <none>        443/TCP                  44m
```

La conteneurisation d’une application par docker



Application



```
nada@nada-virtual-machine:~$ mkdir docker
nada@nada-virtual-machine:~$ cd docker
nada@nada-virtual-machine:~/docker$ nano index.html
nada@nada-virtual-machine:~/docker$ cat index.html
!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Project Management System</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Project Management System</h1>
        <div id="projectForm">
            <input type="text" id="projectName" placeholder="Project Name">
            <button onclick="addProject()">Add Project</button>
        </div>
        <ul id="projectList"></ul>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Dockerfile

Instruction	Description
FROM	Définit l’image de conteneur qui sera utilisée pendant le processus de construction de l'image. Elle n'est utilisable qu'une seule fois dans un Dockerfile.
WORKDIR	Spécifie le répertoire dans lequel seront basées les instructions qui suivront son appel.
USER	Sert à déterminer l'utilisateur ou le groupe d'utilisateur pouvant interagir avec l'image qui sera créée.
COPY	Permet de copier des fichiers internes vers le conteneur Docker. Elle crée une nouvelle couche
EXPOSE	Redirige l'exécution du conteneur vers un port. Les ports doivent être exposés lors du lancement du conteneur.

Création de Dockerfile pour notre application

```
nada@nada-virtual-machine:~/docker$ nano Dockerfile
nada@nada-virtual-machine:~/docker$ cat Dockerfile
# Utilisez une image de Nginx légère
FROM nginx:alpine

# Copiez les fichiers de votre application dans le répertoire de travail de Nginx
COPY index.html /usr/share/nginx/html
COPY script.js /usr/share/nginx/html
COPY styles.css /usr/share/nginx/html

# Exposez le port 80 pour accéder à votre application via le navigateur
EXPOSE 80
```

Nginx est un puissant serveur web HTTP open-source qui peut également servir de proxy inverse, de proxy de messagerie électronique et de répartiteur de charge.

Image & Conteneur Docker

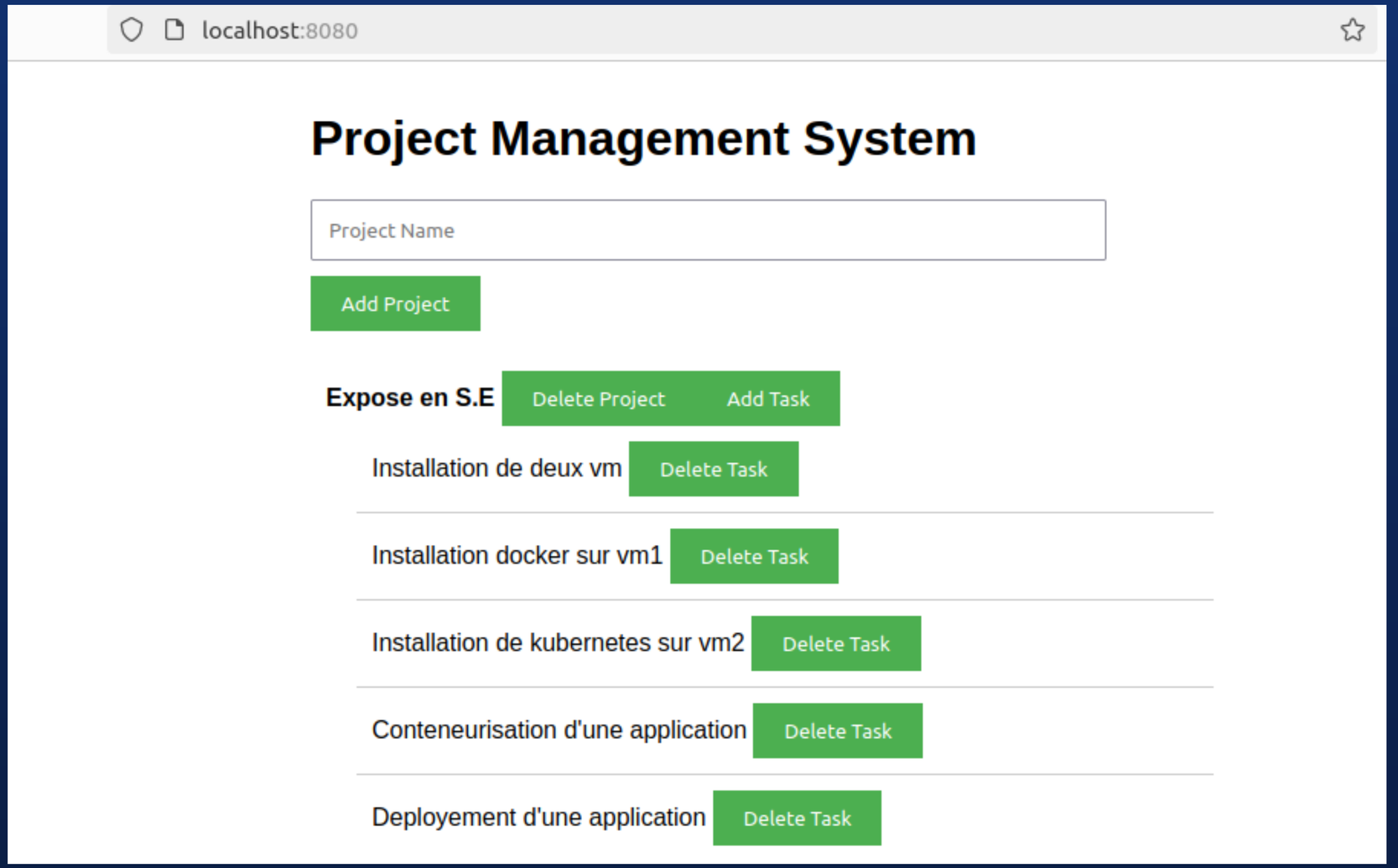
```
nada@nada-virtual-machine:~/docker$ docker build -t mon-application-web .
[+] Building 4.2s (9/9) FINISHED                                                       docker:default
 => [internal] load build definition from Dockerfile                                                0.1s
 => => transferring dockerfile: 378B                                                                0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                                     3.0s
 => [internal] load .dockerignore                                                                   0.1s
 => => transferring context: 2B                                                                     0.0s
 => CACHED [1/4] FROM docker.io/library/nginx:alpine@sha256:31bad00311cb5eeb8a6648beadcf67277a175da89989f14727420a80e2e76742  0.0s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 3.20kB                                                                 0.0s
 => [2/4] COPY index.html /usr/share/nginx/html                                                     0.2s
 => [3/4] COPY script.js /usr/share/nginx/html                                                      0.0s
 => [4/4] COPY styles.css /usr/share/nginx/html                                                     0.1s
 => exporting to image                                                                              0.2s
 => => exporting layers                                                                             0.2s
 => => writing image sha256:bc0e985c7353b0ac04f84c85967fd32fe266a1ce4f7d2fb5daa2c0fe43b01209        0.0s
 => => naming to docker.io/library/mon-application-web                                              0.0s
nada@nada-virtual-machine:~/docker$ docker run -d -p 8080:80 mon-application-web
16937a8fab92bca276b4356ca32e04a47f0912b59390dd3ff5ebc6aa82213fd6
```

Vérification du succés de notre conteneurisation

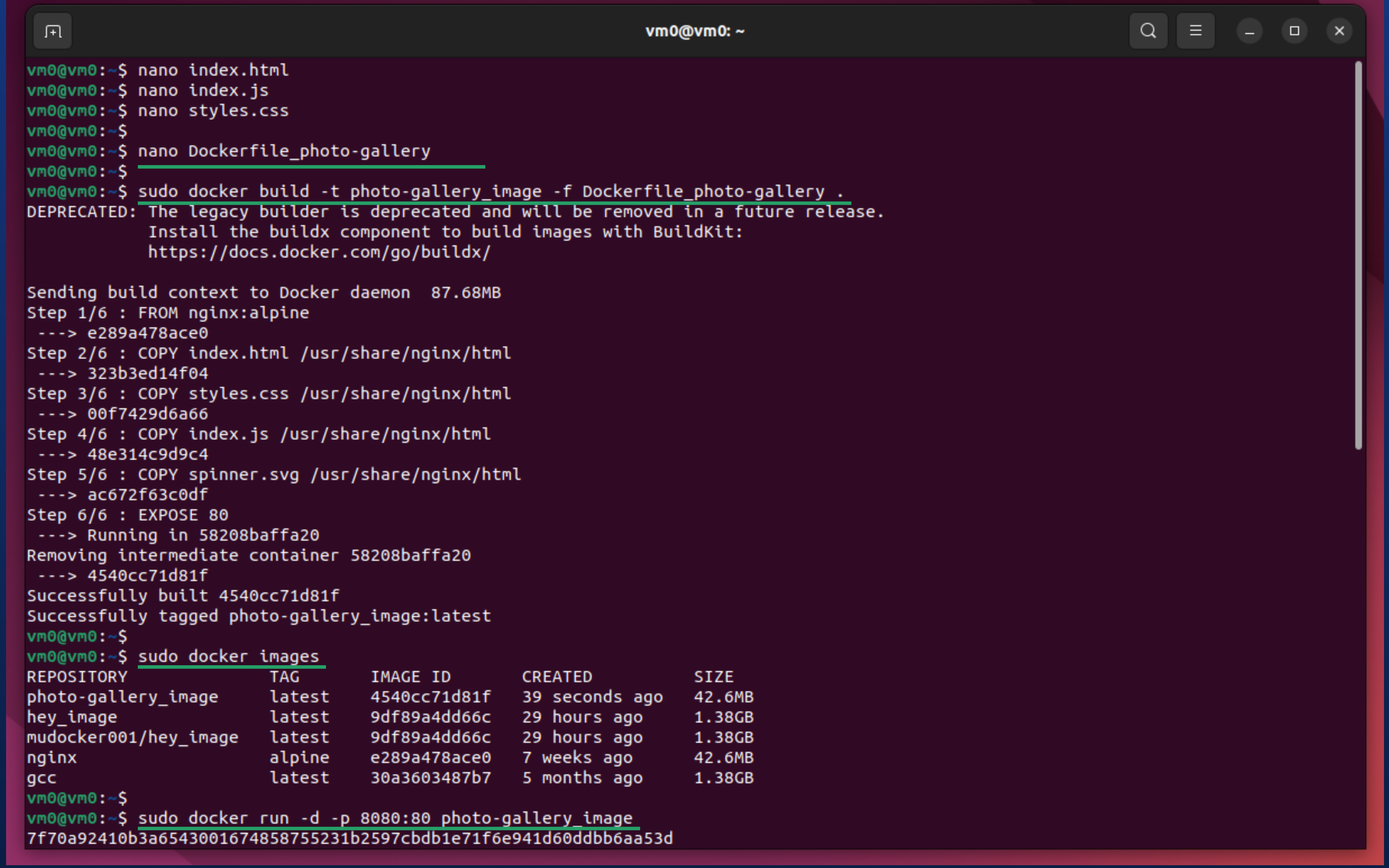
- **Construction de l'image Docker**
- **Exécution du conteneur**

```
nada@nada-virtual-machine:~/docker$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED        STATUS        PORTS                                       NAMES
0c47ec26b0a2   mon-application-web    "/docker-entrypoint.…"   22 hours ago   Up 22 hours   0.0.0.0:8080->80/tcp, :::8080->80/tcp       relaxed
_liskov
a628ab6abb7a   registry:2             "/entrypoint.sh /etc…"   23 hours ago   Up 23 hours   0.0.0.0:5500->5000/tcp, :::5500->5000/tcp   registr
y
nada@nada-virtual-machine:~/docker$ docker images
REPOSITORY                     TAG       IMAGE ID       CREATED         SIZE
application                    latest    5dd9fa5a80f5   25 hours ago    42.6MB
app                            latest    5dd9fa5a80f5   25 hours ago    42.6MB
mon-application-web            latest    bc0e985c7353   25 hours ago    42.6MB
nada818/mon-application-web    latest    bc0e985c7353   25 hours ago    42.6MB
applications                   latest    5e60d036921c   3 days ago      187MB
ourapp                         latest    5e60d036921c   3 days ago      187MB
calcul                         latest    9d5733137b12   6 days ago      167MB
calculate                      latest    9d5733137b12   6 days ago      167MB
calculator                     latest    9d5733137b12   6 days ago      167MB
mon_imagetest                  latest    77b8b4b78440   6 days ago      167MB
localhost:5000/mon_imagetest   latest    77b8b4b78440   6 days ago      167MB
registry                       2         9363667f8aec   2 weeks ago     25.4MB
```

Accés à notre application



Déploiement et gestion d'applications avec Kubernetes


vm0@vm0: ~
vm0@vm0:~$ nano index.html
vm0@vm0:~$ nano index.js
vm0@vm0:~$ nano styles.css
vm0@vm0:~$
vm0@vm0:~$ nano Dockerfile_photo-gallery
vm0@vm0:~$
vm0@vm0:~$ sudo docker build -t photo-gallery_image -f Dockerfile_photo-gallery .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 87.68MB
Step 1/6 : FROM nginx:alpine
---> e289a478ace0
Step 2/6 : COPY index.html /usr/share/nginx/html
---> 323b3ed14f04
Step 3/6 : COPY styles.css /usr/share/nginx/html
---> 00f7429d6a66
Step 4/6 : COPY index.js /usr/share/nginx/html
---> 48e314c9d9c4
Step 5/6 : COPY spinner.svg /usr/share/nginx/html
---> ac672f63c0df
Step 6/6 : EXPOSE 80
---> Running in 58208baffa20
Removing intermediate container 58208baffa20
---> 4540cc71d81f
Successfully built 4540cc71d81f
Successfully tagged photo-gallery_image:latest
vm0@vm0:~$
vm0@vm0:~$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
photo-gallery_image latest 4540cc71d81f 39 seconds ago 42.6MB
hey_image latest 9df89a4dd66c 29 hours ago 1.38GB
mudocker001/hey_image latest 9df89a4dd66c 29 hours ago 1.38GB
nginx alpine e289a478ace0 7 weeks ago 42.6MB
gcc latest 30a3603487b7 5 months ago 1.38GB
vm0@vm0:~$
vm0@vm0:~$ sudo docker run -d -p 8080:80 photo-gallery_image
7f70a92410b3a6543001674858755231b2597cbdb1e71f6e941d60ddbb6aa53d

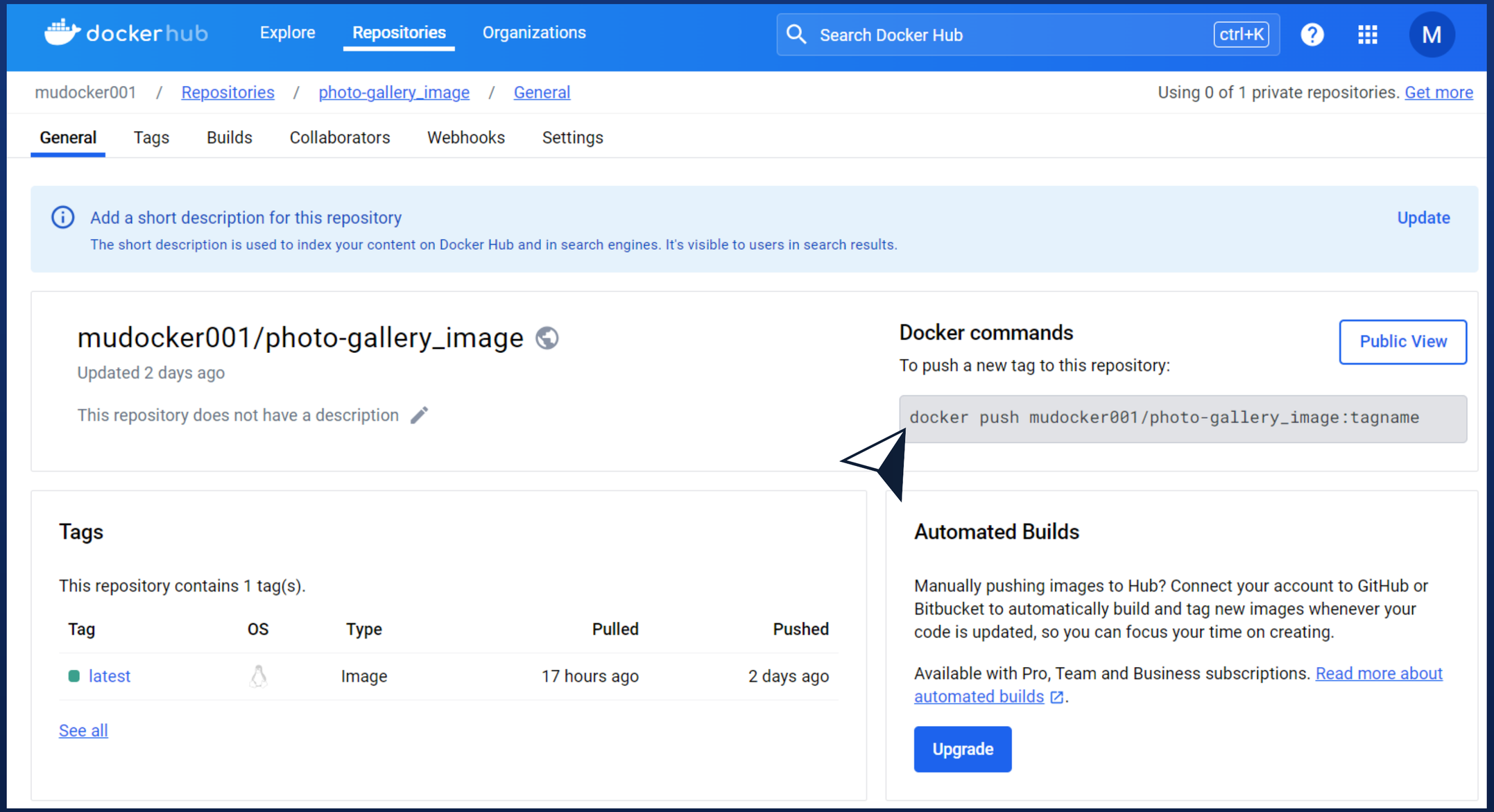
1. Pushing l'image Docker vers Docker Hub

- **Pousser** une image dans **Kubernetes** signifie la **mettre à disposition** dans un emplacement centralisé afin que Kubernetes puisse l'utiliser pour créer des instances de conteneurs en fonction de vos besoins de déploiement.

- Se connecter à Docker Hub
- Taguer l'image
- Pousser (push) l'image taguée vers Docker Hub

Les commandes pour pousser l'image

```
vm0@vm0:~$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED          STATUS          PORTS
   NAMES
7f70a92410b3   photo-gallery_image   "/docker-entrypoint.…"   12 seconds ago   Up 9 seconds    0.0.0.0:8080->80/tcp, :::8080->80/tcp
   lucid_zhukovsky
vm0@vm0:~$
vm0@vm0:~$
vm0@vm0:~$ sudo docker login -u mudocker001
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
vm0@vm0:~$
vm0@vm0:~$ sudo docker tag photo-gallery_image mudocker001/photo-gallery_image
vm0@vm0:~$
vm0@vm0:~$ sudo docker push mudocker001/photo-gallery_image
Using default tag: latest
The push refers to repository [docker.io/mudocker001/photo-gallery_image]
5c0690352932: Pushed
bcdd06338918: Pushed
794496852cb8: Pushed
77dca79ad07c: Pushed
13c52683b537: Mounted from mudocker001/my-web-app_image
0f73163669d4: Mounted from mudocker001/my-web-app_image

aedc3bda2944: Mounted from mudocker001/my-web-app_image

latest: digest: sha256:64f8ca6e1624f8c046f454057d172807cca8e3d824a8482a7840582c9d5e4138 size: 2817
```


docker hub
Explore
Repositories
Organizations
Search Docker Hub
ctrl+K
M
mudocker001 / Repositories / photo-gallery_image / General
Using 0 of 1 private repositories. Get more
General
Tags
Builds
Collaborators
Webhooks
Settings
Add a short description for this repository
The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results.
Update
mudocker001/photo-gallery_image
Updated 2 days ago
This repository does not have a description
Docker commands
Public View
To push a new tag to this repository:
docker push mudocker001/photo-gallery_image:tagname
Tags
This repository contains 1 tag(s).
Tag
OS
Type
Pulled
Pushed
latest
Image
17 hours ago
2 days ago
See all
Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.
Available with Pro, Team and Business subscriptions. Read more about automated builds.
Upgrade

2. Création des fichiers YAML pour Kubernetes

- Création des fichiers YAML nécessaires (deployment.yaml et service.yaml) à l'aide de l'éditeur Nano.

Fichier yaml pour le service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: photo-gallery-service
spec:
  selector:
    app: photo-gallery
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
  type: NodePort
```

- Le fichier de configuration Service permet d'exposer les pods de votre application à d'autres services ou utilisateurs en définissant un point d'entrée stable.

Fichier yaml pour le deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: photo-gallery-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: photo-gallery
  template:
    metadata:
      labels:
        app: photo-gallery
    spec:
      containers:
      - name: photo-gallery
        image: mudocker001/photo-gallery_image:latest
        ports:
        - containerPort: 8080
```

- Le fichier de configuration Deployment spécifie comment les pods doivent être déployés et gérés dans le cluster Kubernetes.

3. Appliquer les fichiers YAML sur Kubernetes

- En utilisant les fichiers YAML , on vérifie régulièrement l'état des déploiements, des services et des pods pour assurer le bon fonctionnement des applications dans le cluster.


vm1@vm1: ~
vm1@vm1:~$
vm1@vm1:~$ nano deployment_photo-gallery.yaml
vm1@vm1:~$
vm1@vm1:~$ nano service_photo-gallery.yaml
vm1@vm1:~$
/ /garantit que le nombre spécifié de pods fonctionne selon les spécifications
vm1@vm1:~$ sudo microk8s kubectl apply -f deployment_photo-gallery.yaml
deployment.apps/photo-gallery-deployment created
vm1@vm1:~$
.//le service fournit un point d'accès stable pour accéder à ces pods
vm1@vm1:~$ sudo microk8s kubectl apply -f service_photo-gallery.yaml
service/photo-gallery-service created
vm1@vm1:~$
vm1@vm1:~$ sudo microk8s kubectl get deployments
//d'obtenir une vue sur l'ensemble des déploiements dans notre cluster
NAME READY UP-TO-DATE AVAILABLE AGE
hey-deployment 1/1 1 1 16h
my-web-app-deployment 1/1 1 1 73m
photo-gallery-deployment 1/1 1 1 65s
vm1@vm1:~$
vm1@vm1:~$ sudo microk8s kubectl get services
//lister des services qui sont utiliser pour exposer l'application
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
kubernetes ClusterIP 10.152.183.1 <none> 443/TCP 3d14h
my-web-app-service NodePort 10.152.183.54 <none> 80:32723/TCP 73m
photo-gallery-service NodePort 10.152.183.65 <none> 80:31495/TCP 51s
vm1@vm1:~$
vm1@vm1:~$ sudo microk8s kubectl get pods
//utile pour surveiller l'état des pods et pour vérifier le fonctionemet de l'application
NAME READY STATUS RESTARTS AGE
hey-deployment-5bb496f987-wtz79 1/1 Running 1 (156m ago) 16h
my-web-app-deployment-5cbfffd65b-jqx9t 1/1 Running 0 71m
photo-gallery-deployment-5c6fd4c55d-srfg9 1/1 Running 0 98s
vm1@vm1:~$
vm1@vm1:~$
//fournit des informations détaillée sur chaque nœud ce qui est utile pour surveiller la sante de cluster
vm1@vm1:~$ sudo microk8s kubectl get nodes -o wide
NAME STATUS ROLES AGE VERSION INTERNAL-IP EXTERNAL-IP OS-IMAGE KERNEL-VERSION CONTAINER-RUNTIME
vm1 Ready <none> 7d v1.28.7 192.168.29.128 <none> Ubuntu 22.04.4 LTS 6.5.0-26-generic containerd://1.6.28
vm1@vm1:~$
vm1@vm1:~$

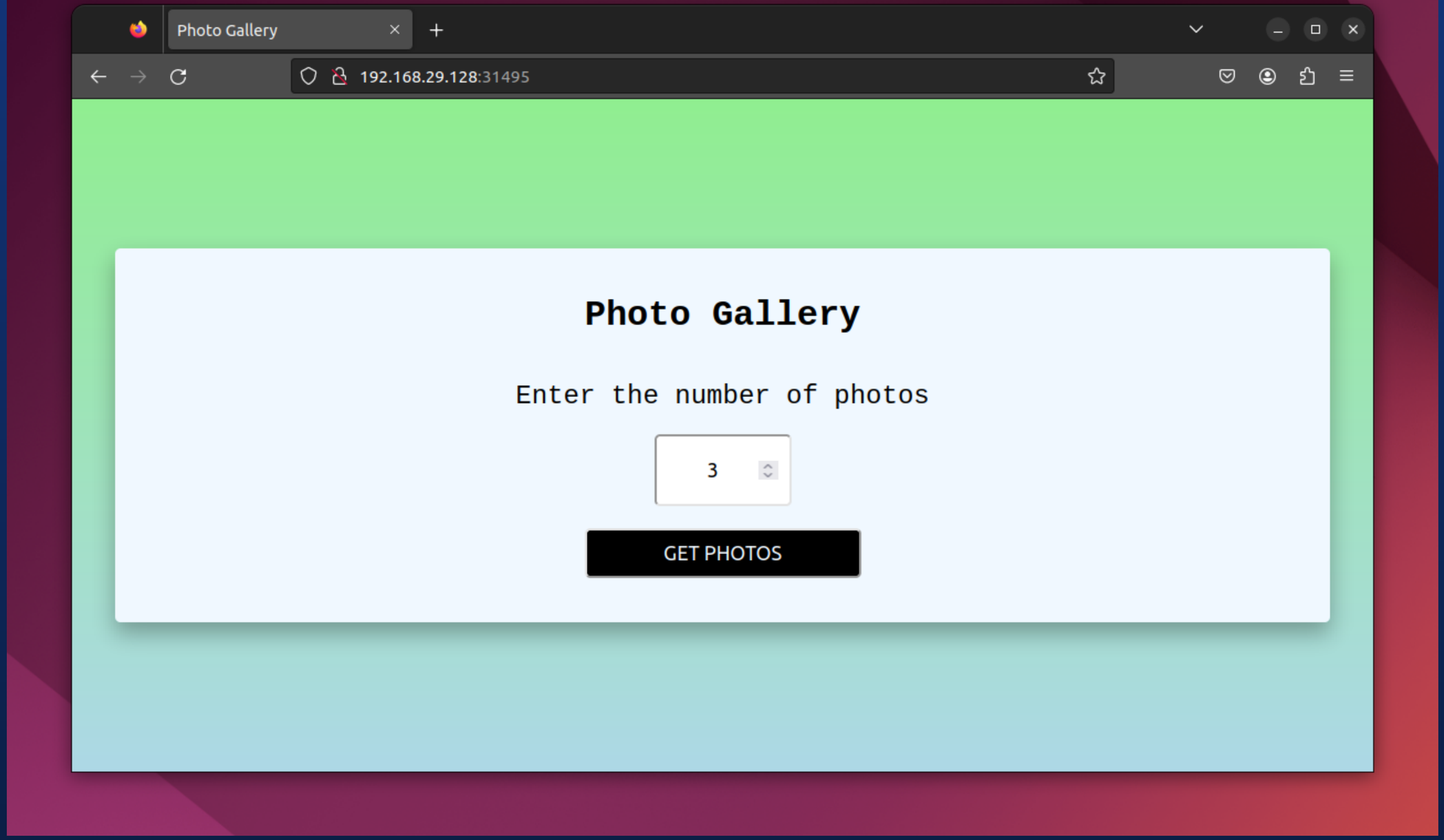


Photo Gallery
192.168.29.128:31495
Photo Gallery
Enter the number of photos
3
GET PHOTOS

4. Surveillance des instances de l'application avec Kubernetes

- Pour explorer les informations de surveillance des instances de l'application ; on se connecte au tableau de bord Kubernetes en utilisant ses fonctionnalités pour visualiser et analyser les métriques et les données de surveillance fournies par les instances déployées.

- **Requests** : C'est la quantité de ressources que le conteneur demande à Kubernetes pour s'exécuter
- **Limits** : C'est la quantité maximale de ressources qu'un conteneur est autorisé à utiliser.

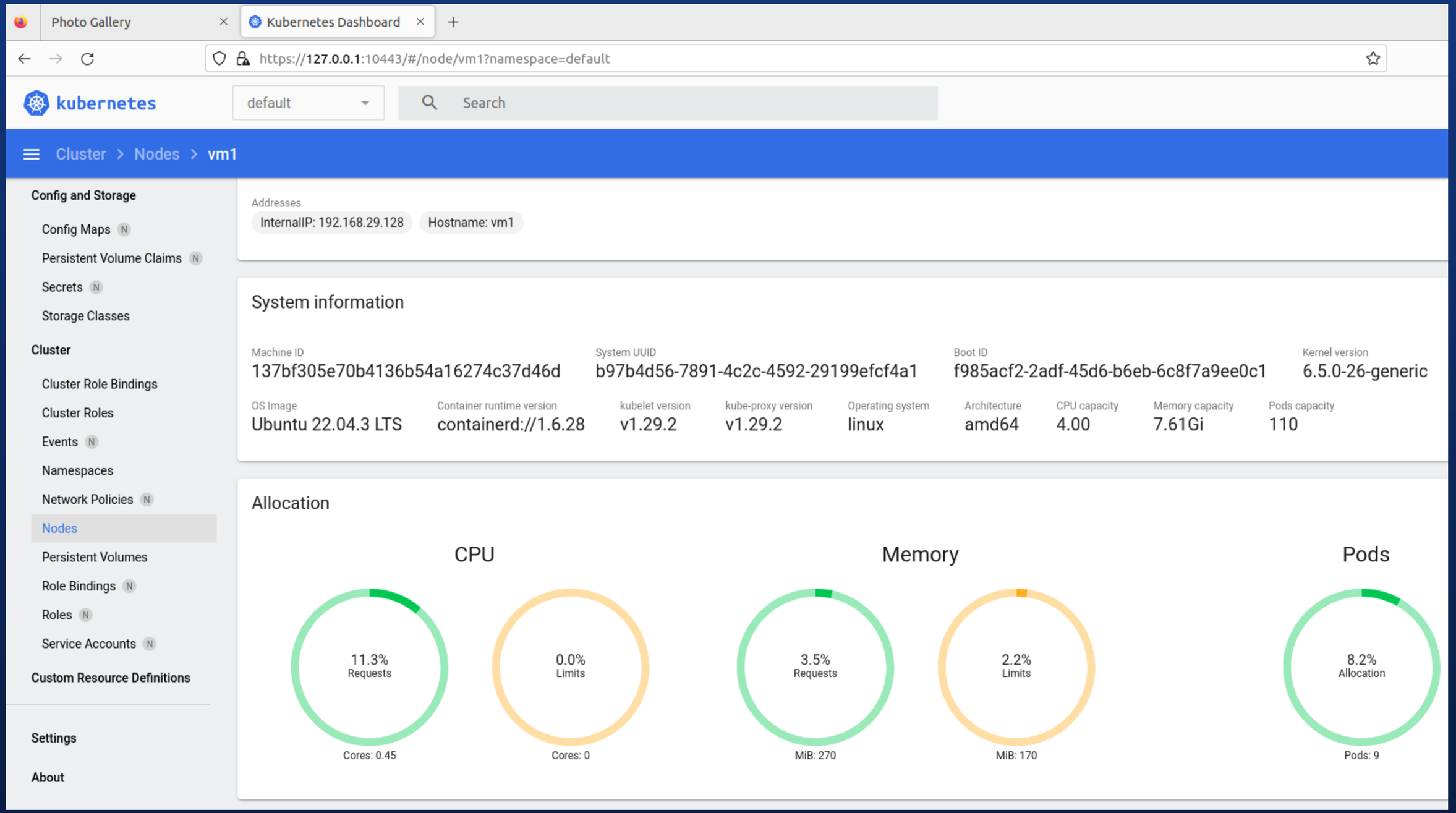


Photo Gallery
Kubernetes Dashboard
https://127.0.0.1:10443/#/node/vm1?namespace=default
kubernetes
default
Search
Cluster > Nodes > vm1
Config and Storage
Config Maps N
Persistent Volume Claims N
Secrets N
Storage Classes
Cluster
Cluster Role Bindings
Cluster Roles
Events N
Namespaces
Network Policies N
Nodes
Persistent Volumes
Role Bindings N
Roles N
Service Accounts N
Custom Resource Definitions
Settings
About
Addresses
InternalIP: 192.168.29.128
Hostname: vm1
System information
Machine ID
137bf305e70b4136b54a16274c37d46d
System UUID
b97b4d56-7891-4c2c-4592-29199efcf4a1
Boot ID
f985acf2-2adf-45d6-b6eb-6c8f7a9ee0c1
Kernel version
6.5.0-26-generic
OS Image
Ubuntu 22.04.3 LTS
Container runtime version
containerd://1.6.28
kubelet version
v1.29.2
kube-proxy version
v1.29.2
Operating system
linux
Architecture
amd64
CPU capacity
4.00
Memory capacity
7.61Gi
Pods capacity
110
Allocation
CPU
11.3%
Requests
Cores: 0.45
0.0%
Limits
Cores: 0
Memory
3.5%
Requests
MiB: 270
2.2%
Limits
MiB: 170
Pods
8.2%
Allocation
Pods: 9

Photo Gallery | Kubernetes Dashboard

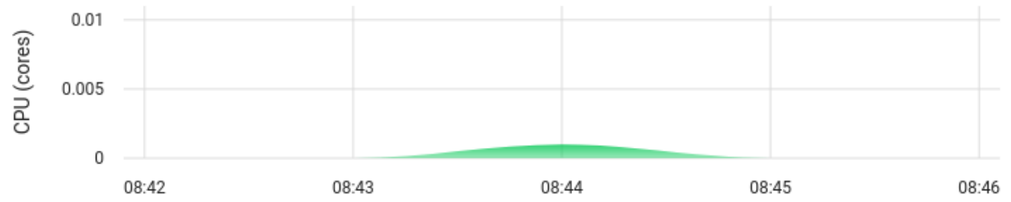
https://127.0.0.1:10443/#/pod/default/photo-gallery-deployment-5c6fd4c55d-qdzzf?namespace=default

kubernetes | default | Search

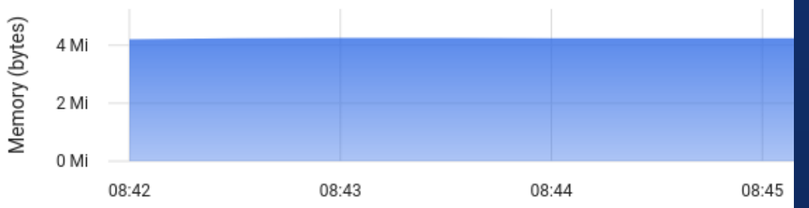
Workloads > Pods > photo-gallery-deployment-5c6fd4c55d-qdzzf

- Workloads
  - Cron Jobs
  - Daemon Sets
  - Deployments
  - Jobs
  - Pods
  - Replica Sets
  - Replication Controllers
  - Stateful Sets
- Service
  - Ingresses
  - Ingress Classes
  - Services
- Config and Storage
  - Config Maps
  - Persistent Volume Claims
  - Secrets
  - Storage Classes
- Cluster
  - Cluster Role Bindings

CPU Usage



Memory Usage



Metadata

Name	Namespace	Created	Age	UID
photo-gallery-deployment-5c6fd4c55d-qdzzf	default	Apr 5, 2024	8 minutes ago	489b5f2d-0e69-408a-bf05-9567dffc337b

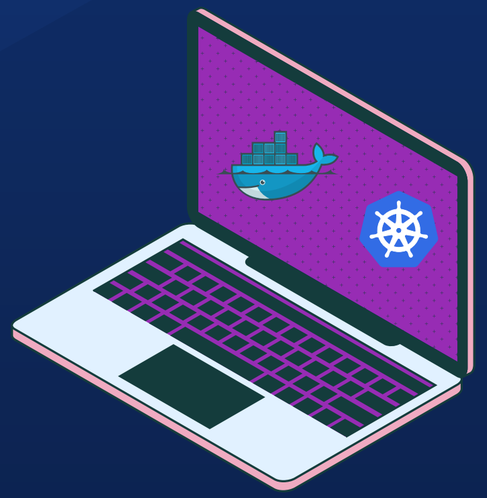
Labels: app: photo-gallery | pod-template-hash: 5c6fd4c55d

Annotations: cni.projectcalico.org/containerID | cni.projectcalico.org/podIP: 10.1.225.9/32 | cni.projectcalico.org/podIPs: 10.1.225.9/32

Resource information

Node	Status	IP	QoS Class	Restarts	Service Account
vm1	Running	10.1.225.9	BestEffort	0	default

TUTORIEL



MERCI POUR VOTRE ATTENTION!