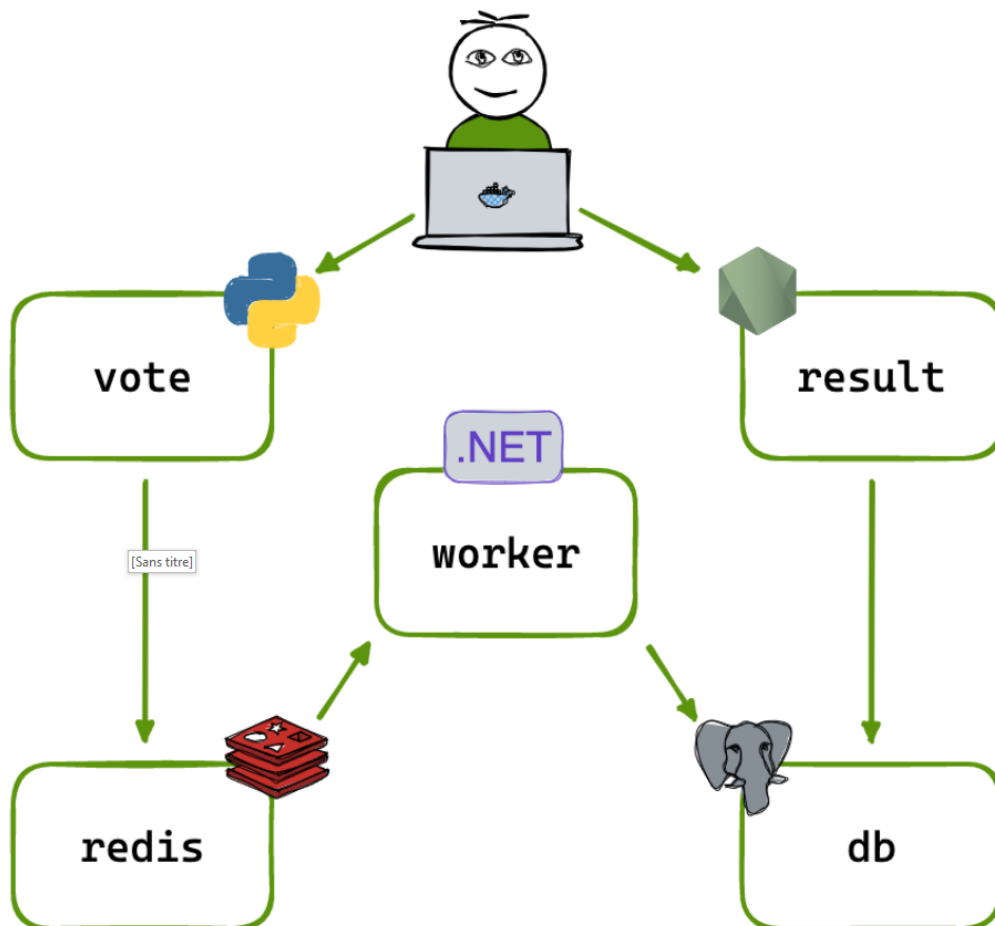


HumansBestFriend app?

CATs or DOGs ?

Virtualisation et conteneurisation



Sommaire

Deploy the app first using docker compose.....	2
1. Préparation de l'environnement :.....	2
2. Gestion des Conteneurs Docker et Déploiement de Services.....	3
Etape 1: build les images sur docker:.....	3
Etape 2 :Publication d'Images Docker : Gestion des Registres.....	3
Etape 3 : téléchargement des images + lancement du fichier compose.yaml:	7
3. Optimisation de l'Environnement Docker :.....	8
Deploy the app inside a kubernetes cluster.....	10
1. Configuration de l'Environnement: (Vu en cours).....	10
2. Déploiement des Machines Virtuelles.....	23
Configuration Post-Installation : Mises à Jour, Visudo et Clé SSH :.....	23
Création de Machines Virtuelles pour kubernetes:.....	24
Préparation des vm de kubernets:.....	25
3. Mise en Place de Kubernetes avec Kubespray:.....	26
Choix du mode d'installation: Utilisation de Kubespray.....	26
Vérification de l'Opérationnalité des Composants Kubernetes.....	28
4. Lancement de l'Infrastructure Kubernetes :.....	29
Conversion des Fichiers Docker Compose en Manifestes Kubernetes :.....	29
Lancement de l'infrastructure:.....	30
Configuration de l'accès au site Web:.....	30
5. Lien github : https://github.com/Merimiam/esiea-ressources.....	33

Meriam Elkhia
Leon Nadot

Julien Taveau
Ruben Woliner

Deploy the app first using docker compose

1. Préparation de l'environnement :

Add Docker's official GPG key:

```
sudo apt-get update
```

```
sudo apt-get install ca-certificates curl gnupg
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor  
-o /etc/apt/keyrings/docker.gpg
```

```
sudo chmod a+r /etc/apt/keyrings/docker.gpg
```

Add the repository to Apt sources:

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture)  
signed-by=/etc/apt/keyrings/docker.gpg]  
https://download.docker.com/linux/ubuntu \
```

```
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update
```

Install Docker and related packages:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin  
docker-compose-plugin
```

Check Docker installation:

```
docker ps
```

Install Git:

apt install git

Clone the repository:

git clone <https://github.com/pascalito007/esiea-ressources.git>

2. Gestion des Conteneurs Docker et Déploiement de Services

Etape 1: build les images sur docker:

Pour créer les images nécessaires sur Docker, nous avons créé un fichier appelé docker-compose.build.yml (sans exécution des conteneurs). Ce fichier sera chargé de construire les images d'application à partir du contenu du fichier Dockerfile fourni.

commande:

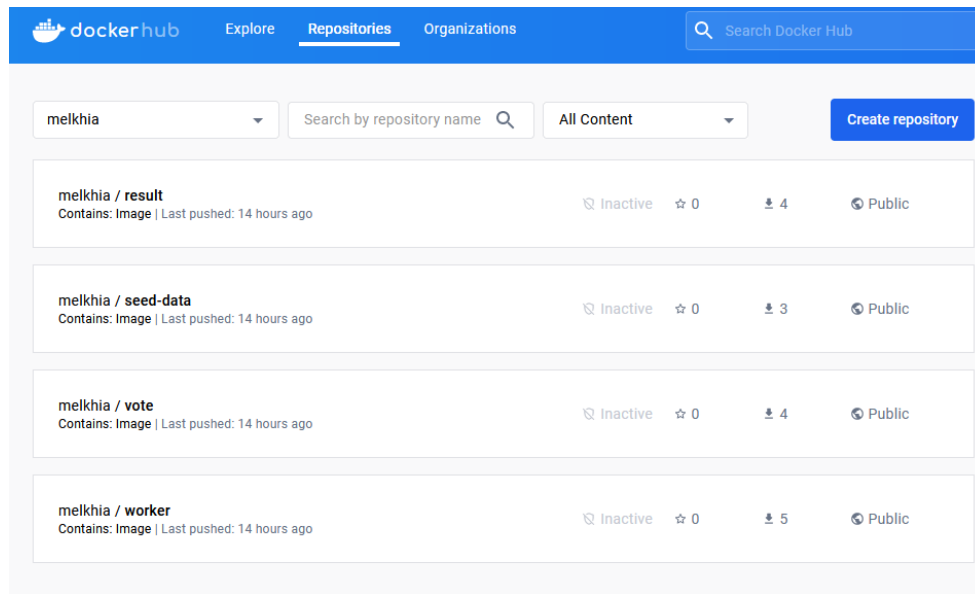
docker-compose -f docker-compose.build.yml build

```
meriam@meriam:~/HumansBestFriend/esiea-ressources$ sudo docker-compose -f docker-compose.build.yml build
db uses an image, skipping
redis uses an image, skipping
Building worker
[+] Building 14.9s (S/15)
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for mcr.microsoft.com/dotnet/runtime:7.0
=> [internal] load metadata for mcr.microsoft.com/dotnet/sdk:7.0
=> [build 1/7] FROM mcr.microsoft.com/dotnet/sdk:7.0@sha256:da14150ee9883b44b163b44edc4d0fb73aa5fc2a8f442002e195b8344afe19f
=> => resolve mcr.microsoft.com/dotnet/sdk:7.0@sha256:da14150ee9883b44b163b44edc4d0fb73aa5fc2a8f442002e195b8344afe19f
=> => sha256:4927d9f3828c8b5d951739eeefc3ea38f779871dc609c0b5278bd6f32b155 5.31kB / 5.31kB
=> => sha256:b7f91549542cca4b35a34cdee5364339f17468971ea730bb072864d3e78c8b94 31.42MB / 31.42MB
=> => sha256:a976e6e7a5a13f3b88541bec8be0171b688df446365c3d24140a2892b92eb34 5.24MB / 14.97MB
=> => sha256:da14150ee9883b44b163b44edc4d0fb73aa5fc2a8f442002e195b8344afe19f 1.79kB / 1.79kB
=> => sha256:1408928979c9f667aa3fbbe49130421800d451f32749ad31e5e2025d17935765 2.01kB / 2.01kB
=> => sha256:01f22fa85c0ae740e3132b4e518b3dcf1ca9fe8bf540c5cd072d5047295742bf 6.29MB / 32.46MB
=> => sha256:e77250e794662a50cc3b6a01908cf71cd7940bcff7f85f1b885751ecad2888 155B / 155B
=> => extracting sha256:b7f91549542cca4b35a34cdee5364339f17468971ea730bb072864d3e78c8b94 1.9s
=> => sha256:f16f1888acf5e94c7831885068d3371c10b2df2a3dc3774f8ab04ae03413eeb 10.12MB / 10.12MB
=> => sha256:24d6e94ca71aac0384ac7df8bb8ed3c463f5cb4db215b90129790bcd744737dc 8.39MB / 25.38MB
=> [stage-1 1/3] FROM mcr.microsoft.com/dotnet/runtime:7.0@sha256:3790a506af2f994e338ff96a9bcb1c8f723a4721964ce741e8b08798ac809b8
=> => resolve mcr.microsoft.com/dotnet/runtime:7.0@sha256:3790a506af2f994e338ff96a9bcb1c8f723a4721964ce741e8b08798ac809b8
=> => sha256:4136fcfcf722901af4b07056d5ff44aca480f018c22c5e10926948e5134d30f6 1.16kB / 1.16kB
=> => sha256:ea93e1e267081c24164669014ce3638c60325b8f01f2d6131f506cc3a9603fba 1.93kB / 1.93kB
```

Etape 2 :Publication d'Images Docker : Gestion des Registres

Les images que nous avons construites doivent d'abord être publiées dans notre registre Docker public, puis dans un registre privé. Nous nous sommes assurés d'avoir une interface utilisateur pour le registre privé, comme nous l'avons vu dans le cours.

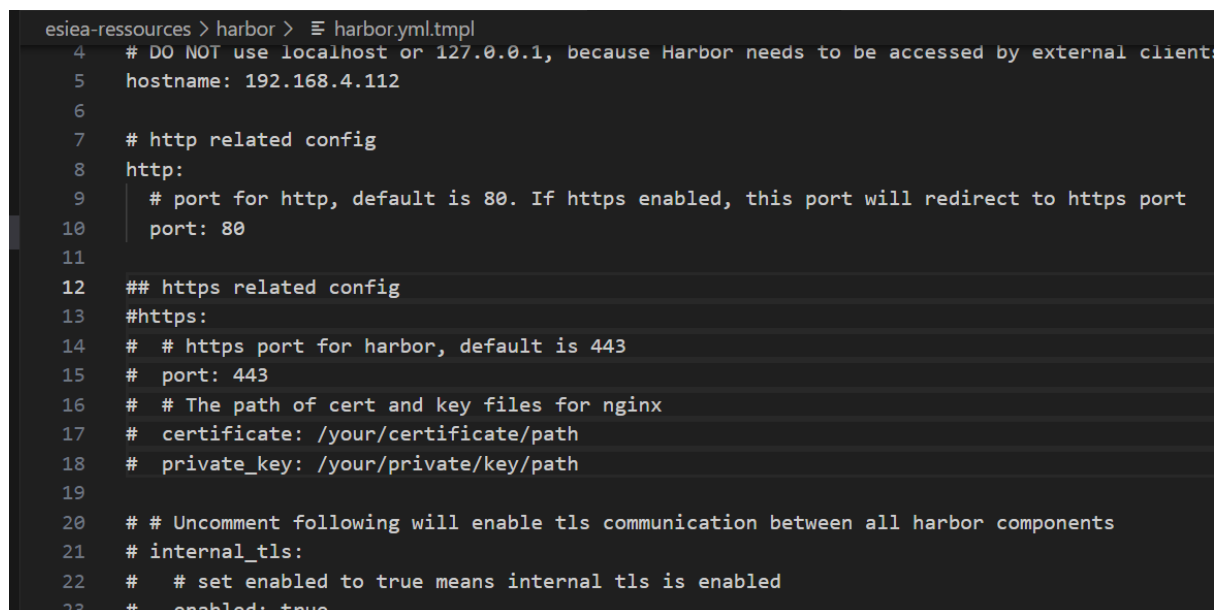
Registre docker public:



Registre docker en local:

On peut aussi mettre les images dans un registry en local, pour ce faire nous utilisons harbor :

Il faut d'abord le configurer puis lancer l'installation :



```
● meriam@meriam:~/HumansBestFriend/esiea-ressources/harbor$ cp harbor.yml.tpl harbor.yml
○ meriam@meriam:~/HumansBestFriend/esiea-ressources/harbor$ ./install.sh
```

[Step 0]: checking if docker is installed ...

Note: docker version: 24.0.7

[Step 1]: checking docker-compose is installed ...

Note: Docker Compose version v2.21.0

[Step 2]: loading Harbor images ...

[Step 5]: starting Harbor ...

[+] Running 10/10

✓ Network harbor_harbor Created

✓ Container harbor-log Started

✓ Container registryctl Started

✓ Container redis Started

✓ Container registry Started

✓ Container harbor-db Started

✓ Container harbor-portal Started

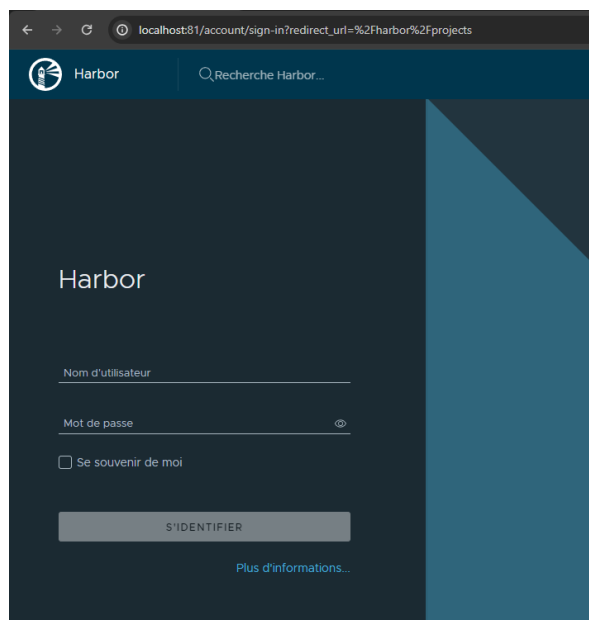
✓ Container harbor-core Started

✓ Container harbor-jobservice Started

✓ Container nginx Started

✓ ----Harbor has been installed and started successfully.----

```
○ meriam@meriam:~/HumansBestFriend/esiea-ressources/harbor$
```



Nous créons un nouveau projet où publier les images:

The screenshot shows a 'Nouveau Projet' (New Project) form with the following fields and options:

- Nom du Projet ***: buildforkube
- Niveau d'accès ⓘ**: ☒ Public
- Project quota limits ⓘ ***: -1 GiB ▾
- Proxy Cache ⓘ**: ☐

At the bottom right, there are two buttons: 'ANNULER' (Cancel) and 'OK'.

Nous publions les images:

```
# Boucle sur chaque service
for SERVICE in "${SERVICES[@]}"; do
    # Construire l'image
    docker-compose build $SERVICE

    # Récupérer le nom de l'image construite
    # Assurez-vous que cette commande reflète le nom correct de vos images
    IMAGE_NAME=$(docker-compose images -q $SERVICE)

    # Vérifier si l'image existe
    if [ -z "$IMAGE_NAME" ]; then
        echo "Image pour le service $SERVICE introuvable. Vérifiez vos configurations Docker Compose."
        continue
    fi

    # Tagger l'image pour Harbor
    docker tag $IMAGE_NAME "${HARBOR_REGISTRY}/${PROJECT_NAME}/${SERVICE}:latest"

    # Pousser l'image sur Harbor
    docker push "${HARBOR_REGISTRY}/${PROJECT_NAME}/${SERVICE}:latest"
done
```



```
=> exporting to image
=> => exporting layers
=> => writing image sha256:c819cc3b8f30dcde1c8381420f6bb381dd3425bc221bea4909ddca756ac7e1e0
=> => naming to docker.io/library/esiea-ressources_worker
/usr/lib/python3/dist-packages/paramiko/transport.py:236: CryptographyDeprecationWarning: Blowfish has been deprecated
"__class__": algorithms.Blowfish,
The push refers to repository [127.0.0.1/buildforkube/worker]
f17b3c09ccc5: Pushed
a9e9dc456082: Pushed
9165c11d69c3: Pushed
db3084a64bb8: Pushed
d3a323cd3227: Pushed
1b6fd3ad4ce6: Pushing [=====> ] 75.45MB/80.55MB
```

Etape 3 : téléchargement des images + lancement du fichier compose.yaml:

Nous avons créé un autre fichier appelé compose.yml, qui sera chargé de déployer l'application et tous les conteneurs nécessaires. Nous nous sommes assurés que les images référencent celles de notre registre privé

```
meriam@meriam:~/HumansBestFriend/esiea-ressources$ sudo docker compose up -d
[sudo] password for meriam:
[+] Running 14/19
  db 9 layers [#####] 4.311MB/91.4MB Pulling
  ✓ 661ff4d9561e Pull complete
  ✓ e4a3f96ea8e5 Pull complete
  ✓ 0c1e2e159ea1 Pull complete
  " 26c071a8426e Downloading [==> ] 4.311MB/91.4MB
  ✓ e9a1ba05d22c Download complete
  ✓ efc39a79d7dc Download complete
  ✓ 72124e665f9e Download complete
  " aa569f3e770e Waiting
  " 86d5fe07cb37 Waiting
  redis 8 layers [#####] 0B/0B Pulling
  ✓ 1f7ce2fa46ab Already exists
  ✓ 4827e9d1e197 Pull complete
  ✓ 5845062cfda9 Pull complete
  ✓ 44d659adcf8b Pull complete
  ✓ b6962d83313d Download complete
  ✓ 5d29cf86ecab Download complete
  ✓ 4f4fb700ef54 Download complete
  ✓ 3a2d9f90268c Download complete
```

3. Optimisation de l'Environnement Docker :

Correction du healthcheck du container Redis en remplaçant le sh par "bash"

Le healthcheck du container redis n'est pas passé à cause d'une erreur de syntax nous avons changé le sh pour "bash":

```
healthchecks > $ redis.sh
1 |#!/bin/bash
2 |set -eo pipefail
3 |
4 |host="$(hostname -i || echo '127.0.0.1')"
5 |
6 |if ping="$(redis-cli -h "$host" ping)" && [ "$ping" = 'PONG' ]; then
7 |    exit 0
8 |fi
9 |
10|exit 1
11|
```

Initialisation de la table "vote" dans la base de données avec un entryptpoint

Sur la base de données il manquait une table vote qui permet de stocker les votes on l'initialise de la manière suivante avec un entryptpoint :

```
init.sql U X  docker-compose-build.yaml U
init > init.sql
1 |CREATE TABLE votes (
2 |    id SERIAL PRIMARY KEY,
3 |    vote VARCHAR(255) NOT NULL
4 |);
5 |
```

Lancement avec des données préexistantes (seed data)

Utilisation de la commande *docker-compose --profile seed up -d* pour démarrer les conteneurs avec des données préexistantes, facilitant les tests et le

développement.

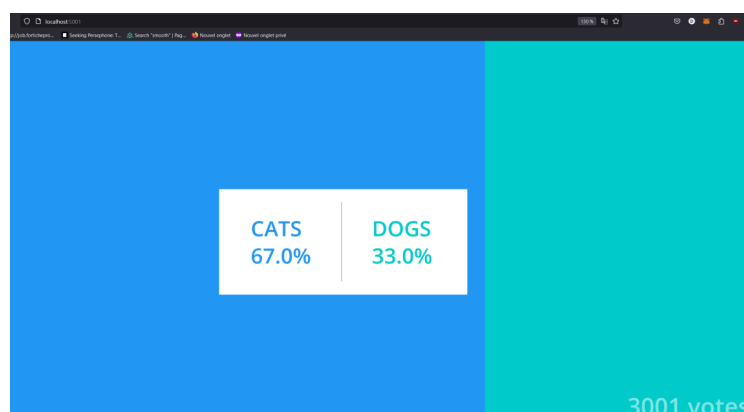
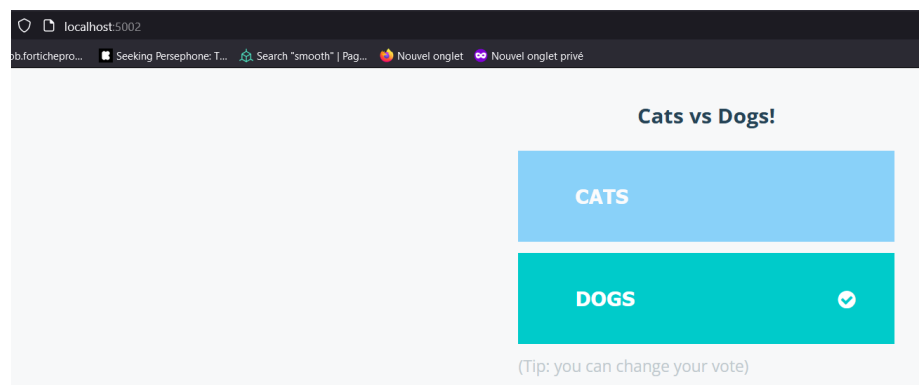
Vérification de l'état des containers

Présentation de l'état opérationnel des containers avec une confirmation que tous les containers sont "up" et fonctionnent correctement.

```
root@meriam:/home/meriam/HumansBestFriend/esiea-ressources# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS
d7f09e21f44d   esiea-ressources-worker            "dotnet Worker.dll"     11 seconds ago Up 4 seconds
1809eca994fa   esiea-ressources-result            "/usr/bin/tini -- no..." 11 seconds ago Up 4 seconds        127.0.0.1:9229->9229/tcp, 0.0.0.0:5001->80/tcp
80/tcp        esiea-ressources-result-1
fc81e351b81a   esiea-ressources-vote              "gunicorn app:app -b..." 11 seconds ago Up 10 seconds (health: starting) 0.0.0.0:5002->80/tcp, :::5002->80/tcp
28f5da8df24f   postgres:15-alpine                "docker-entrypoint.s..." 11 seconds ago Up 10 seconds (healthy) 5432/tcp
398664c4a56f   redis                              "docker-entrypoint.s..." 11 seconds ago Up 10 seconds (healthy) 6379/tcp
25700e76d54c   postgres:15-alpine                "docker-entrypoint.s..." 12 minutes ago Up 12 minutes (healthy) 5432/tcp
esiea-ressources-db-1
root@meriam:/home/meriam/HumansBestFriend/esiea-ressources# docker exec -it esiea-ressources-redis-1 bash
```

Captures des sites web une fois que les conteneurs sont opérationnels

Illustration visuelle des sites web capturés après le déploiement réussi des conteneurs, mettant en avant le processus de vote pour les chiens.



Deploy the app inside a kubernetes cluster

1. Configuration de l'Environnement: (Vu en cours)

Nous commençons par se créer un compte et télécharger le esxi

Your downloads are available below

VMware vSphere Hypervisor - Binaries

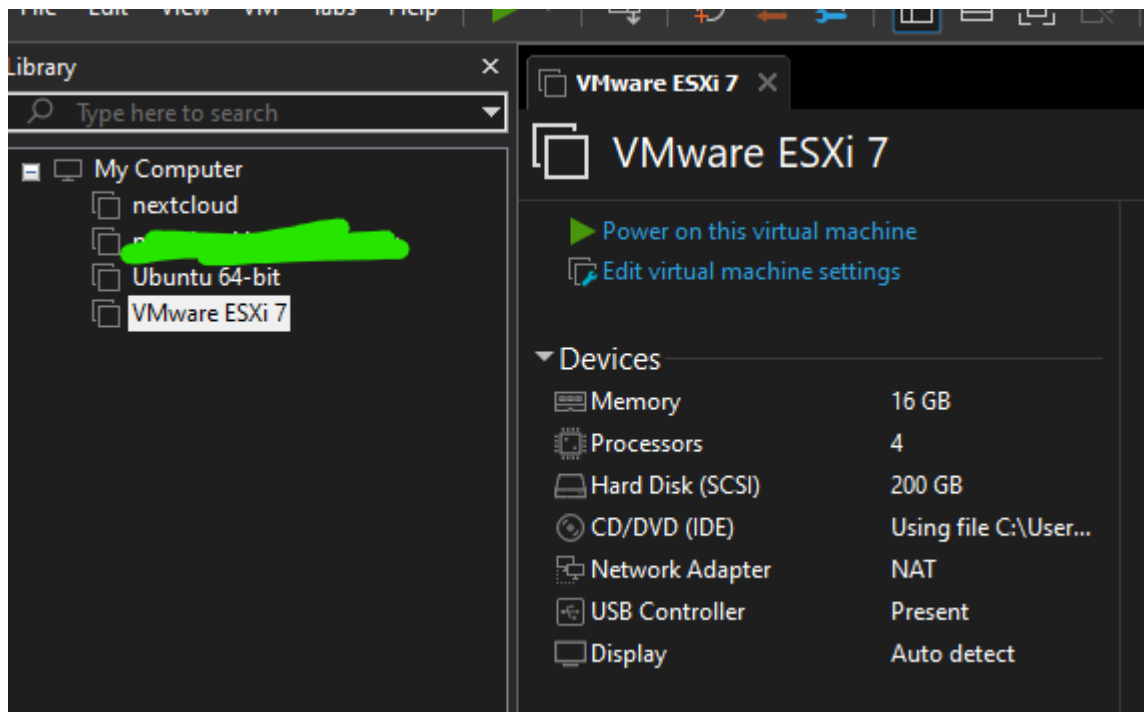
VMware vSphere Hypervisor (ESXi ISO) image
2023-07-06 | 7.0U3n | 382.05 MB | iso

[Manually Download](#)

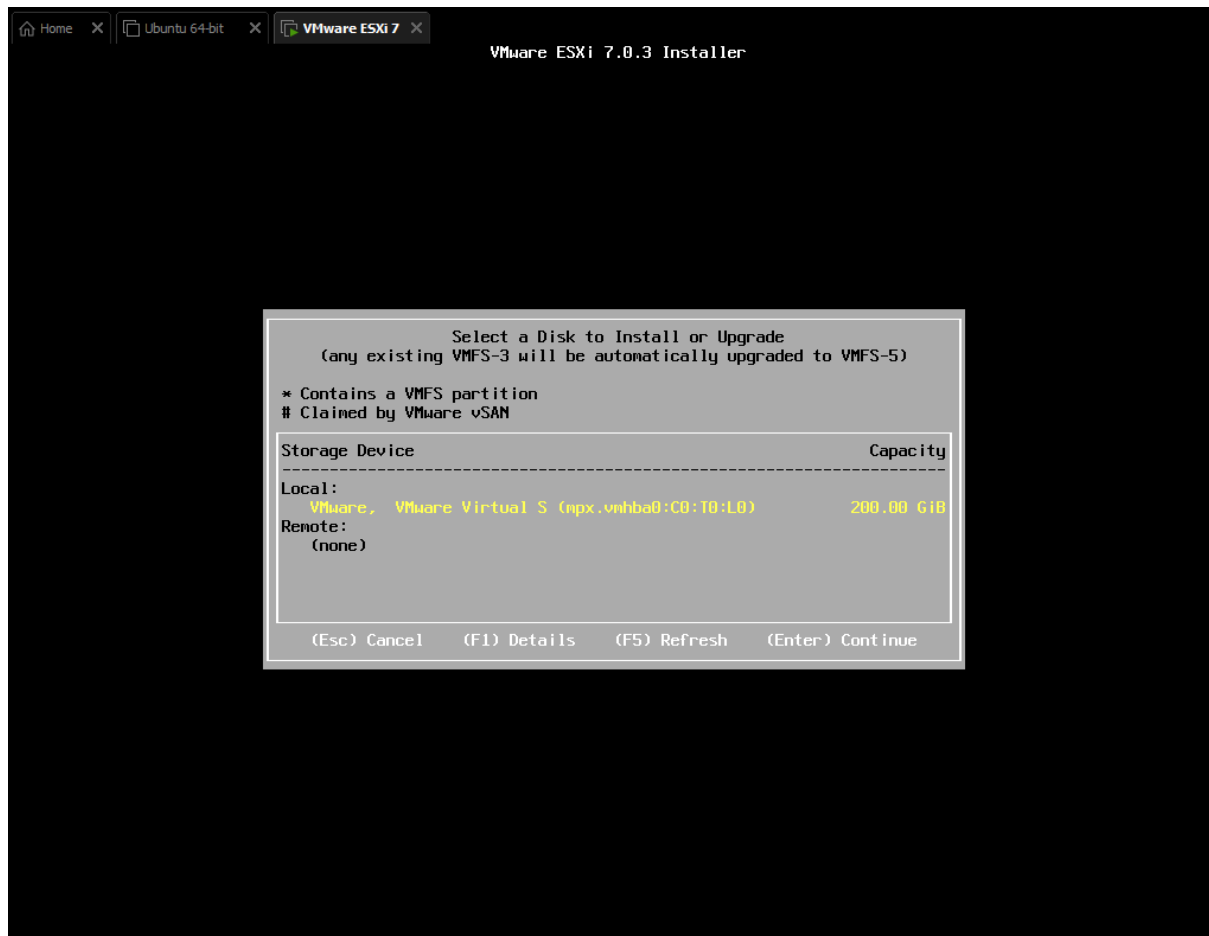
Boot your server with this image in order to install or upgrade to ESXi (ESXi requires 64-bit capable servers).
This ESXi image includes VMware Tools.

MD5SUM(*): 0cacd44568b7f6112c6d003ef210dcfe
SHA1SUM(*): 07b2f7677b274fca656bac564445c0882b540835
SHA256SUM(*): b5e65fc738900296dd70d1938d76ebb72dab19b9b722dedad445669fc77f1272

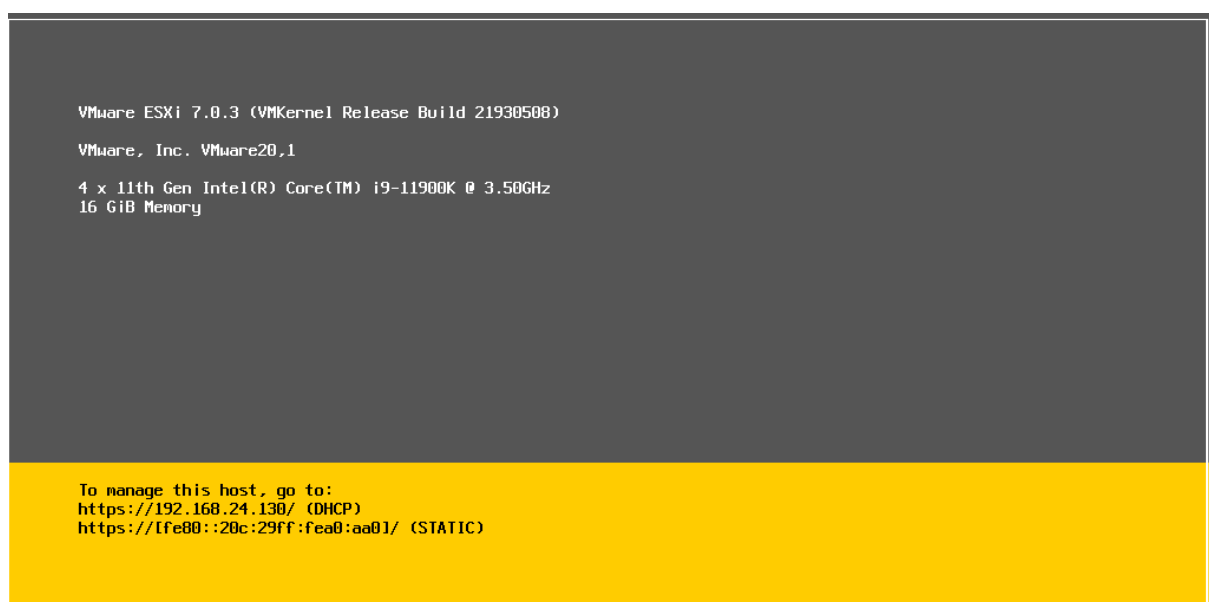
Ensuite nous importons la machine, et nous configurons l'interface réseau en bridge



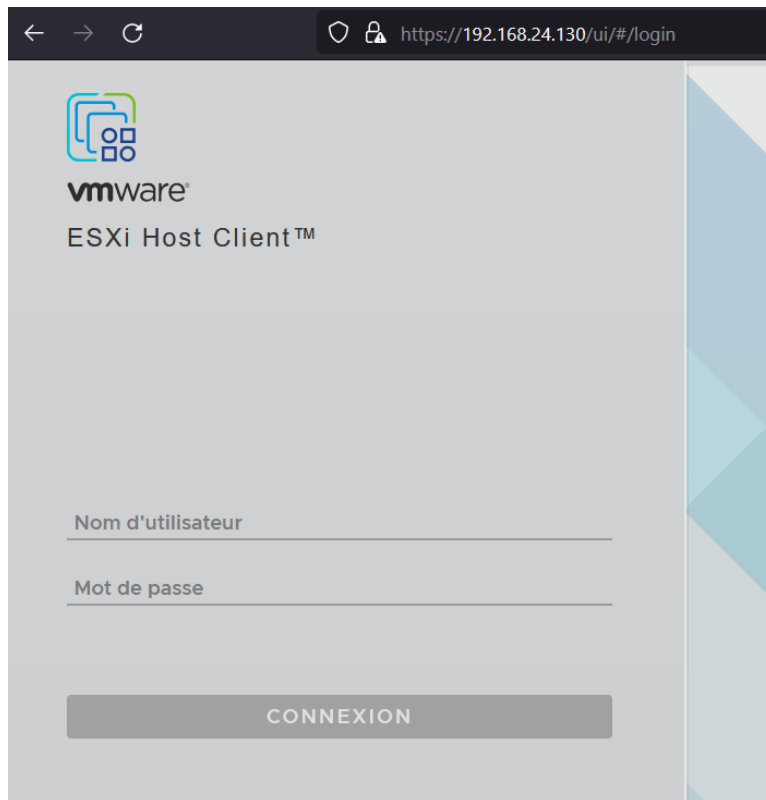
Installation d'ESXi et Vérification de la Connexion Bridge



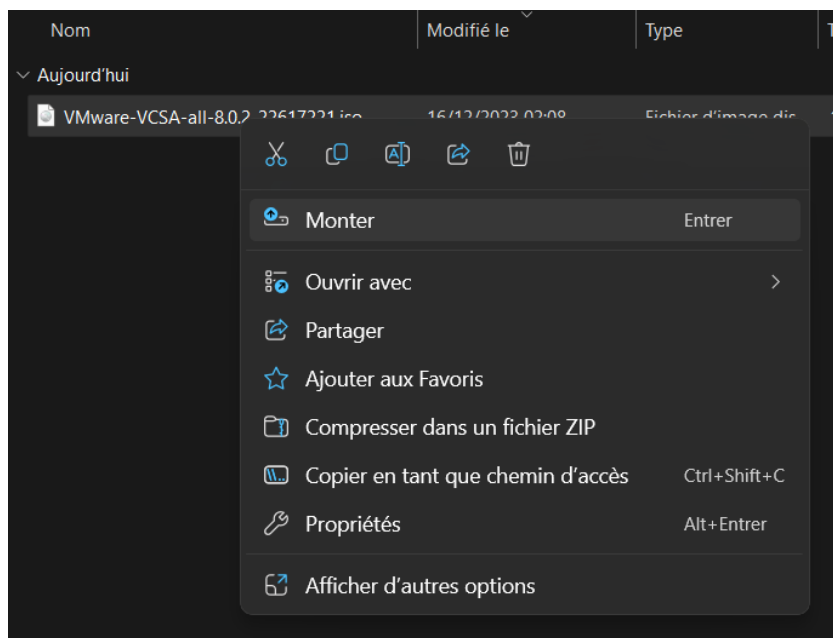
Esxi est up



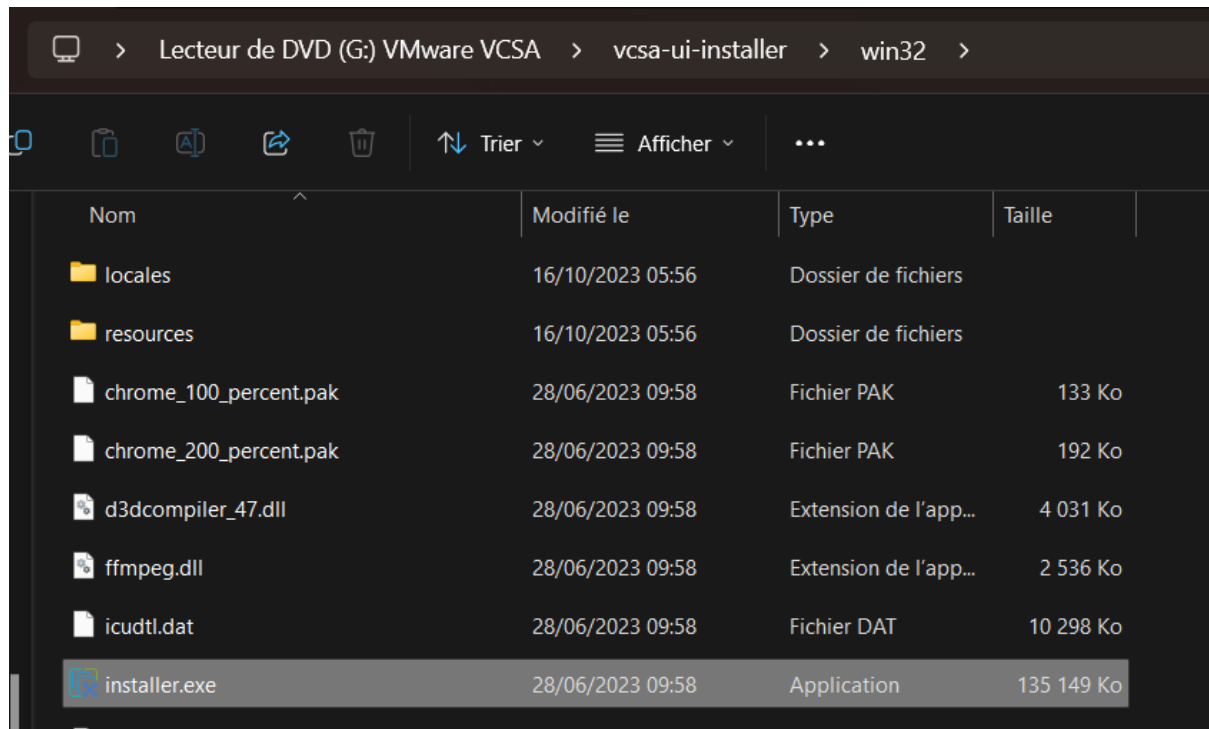
Tester la connexion BRIDGE:



Téléchargement et Installation de vCenter (Optionnel)



Choisir le mode d'installation



Faire l'install et compléter toute les étapes

vCenter Server deployment target

Specify the vCenter Server deployment target settings. The target is the ESXi host or vCenter Server instance on which the vCenter Server will be deployed.

ESXi host or vCenter Server name	192.168.24.130	
HTTPS port	443	
User name	root	
Password	

CANCEL


BACK



NEXT

Select deployment size

Select the deployment size for this vCenter Server.

For more information on deployment sizes, refer to the vSphere 8.0 documentation.


Deployment size Tiny 

Storage size Default  

Resources required for different deployment sizes

Deployment Size	vCPUs	Memory (GB)	Storage (GB)	Hosts (up to)	VMs (up to)
Tiny	2	14	579	10	100
Small	4	21	694	100	1000
Medium	8	30	908	400	4000
Large	16	39	1358	1000	10000
X-Large	24	58	2283	2000	35000

Select datastore

 Insufficient disk space for thick provisioning. It requires 579 GB 

Select the storage location for this vCenter Server

☒ Install on an existing datastore accessible from the target host

☒ Show only compatible datastores

Name	Type	Capacity	Free	Provisioned	Thin Provisioning
datastore1	VMFS-6	71.75 GB	70.34 GB	1.41 GB	Supported

1 item

☒ Enable Thin Disk Mode 

☐ Install on a new vSAN cluster containing the target host 

Configure network settings

Configure network settings for this vCenter Server

Network	VM Network	ⓘ
IP version	IPv4	
IP assignment	static	
FQDN	FQDN (optional)	ⓘ
IP address	192.168.24.200	
Subnet mask or prefix length	24	ⓘ
Default gateway	192.168.24.1	
DNS servers	8.8.8.8	
Common Ports		
HTTP	80	
HTTPS	443	

CANCEL

BACK

NEXT

Ready to complete stage 1

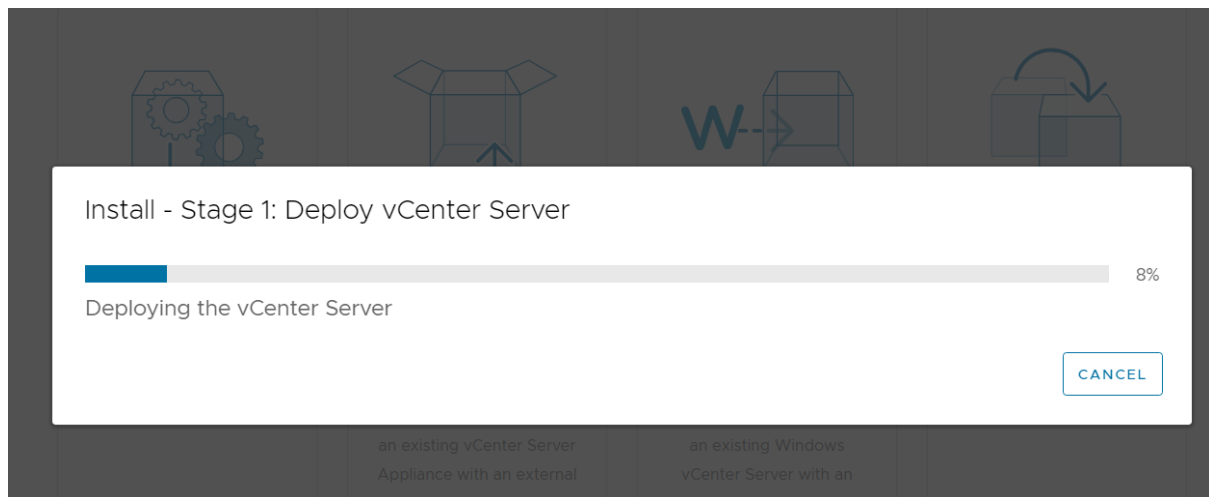
Review your settings before starting the vCenter Server deployment.

Deployment Details	
Target ESXi host	192.168.24.130
VM name	VMware vCenter Server
Deployment size	Tiny
Storage size	Default
Datastore Details	
Datastore , Disk mode	datastore1 , thin
Network Details	
Network	VM Network
IP settings	IPv4 , static
IP address	192.168.24.200
Subnet mask or prefix length	24
Default gateway	192.168.24.1
DNS servers	8.8.8.8
HTTP Port	80
HTTPS Port	443

CANCEL

BACK

FINISH



Configuration de vCenter :

Setup Wizard

- 1 Introduction
- 2 vCenter Server Configuration
- 3 SSO Configuration**
- 4 Configure CEIP
- 5 Ready to complete

SSO Configuration

☒ Create a new SSO domain

Single Sign-On domain name ⓘ	vsphere.local
Single Sign-On username	administrator
Single Sign-On password ⓘ ⓘ
Confirm password ⓘ

☐ Join an existing SSO domain

vCenter Server

Ready to complete

Review your settings before finishing the wizard.

Network Details	
Network configuration	Assign static IP address
IP version	IPv4
IP address	192.168.24.200
Subnet mask	24
Host name	localhost
Gateway	192.168.24.1
DNS servers	8.8.8.8
vCenter Server Details	
Time synchronization mode	Synchronize time with the ESXi host
SSH access	Activated
SSO Details	
SSO Details	vsphere.local
Username	administrator
Customer Experience Improvement Program	
CEIP setting	Opted out

CANCEL BACK FINISH

Vérification du bon fonctionnement du vcenter:

← → ↺ 🔒 🔍 https://192.168.4.200/webso/SAML2/SSO/vsphere.local?SAML...

VMware® vSphere

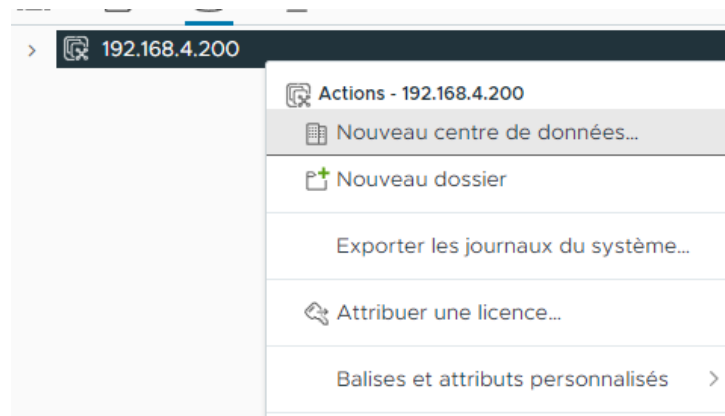
administrator@vsphere.local

.....

☐ Utiliser l'authentification de session Windows

CONNEXION

Gestion des Clusters et Ajout d'ESXi en tant qu'Hôte :



Nouveau centre de données

Nom





Emplacement  192.168.4.200


ANNULER

OK

Création un nouveau cluster pour ce datacenter :


Informations de base

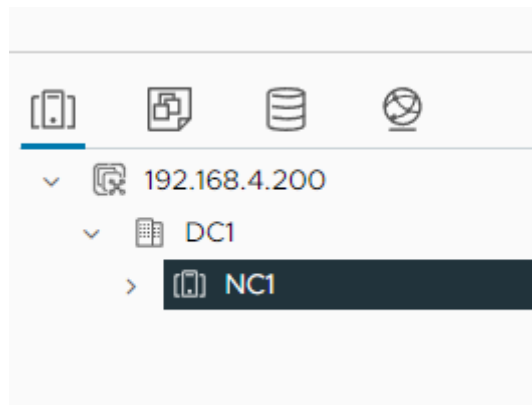
Nom	<input type="text" value="NC1"/>
Emplacement	 DC1
 vSphere DRS	<input checked="" type="checkbox"/>
 vSphere HA	<input checked="" type="checkbox"/>
vSAN	<input checked="" type="checkbox"/> Activer vSAN ESA 

☒ Gérer tous les hôtes du cluster avec une seule image 

Choisir comment configurer l'image du cluster

- ☒ Composer une nouvelle image
- ☐ Importer une image à partir d'un hôte existant dans l'inventaire vCenter
- ☐ Importer une image à partir d'un nouvel hôte

☐ Gérer la configuration au niveau du cluster 



Ajouter l'esxi en host:

Ajouter un hôte

- 1 Nom et emplacement**
- 2 Paramètres de connexion
- 3 Résumé de l'hôte

Nom et emplacement

Saisissez le nom ou l'adresse IP de l'hôte à ajouter à vCenter Server.

Nom d'hôte ou adresse IP : 192.168.4.113

Emplacement : DC1

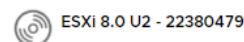
Ajouter un hôte

- 1 Nom et emplacement
- 2 Paramètres de connexion
- 3 Résumé de l'hôte
- 4 Host lifecycle
- 5 Image
- 6 Attribuer une licence
- 7 Mode de verrouillage
- 8 Emplacement de la VM
- 9 Prêt à terminer**

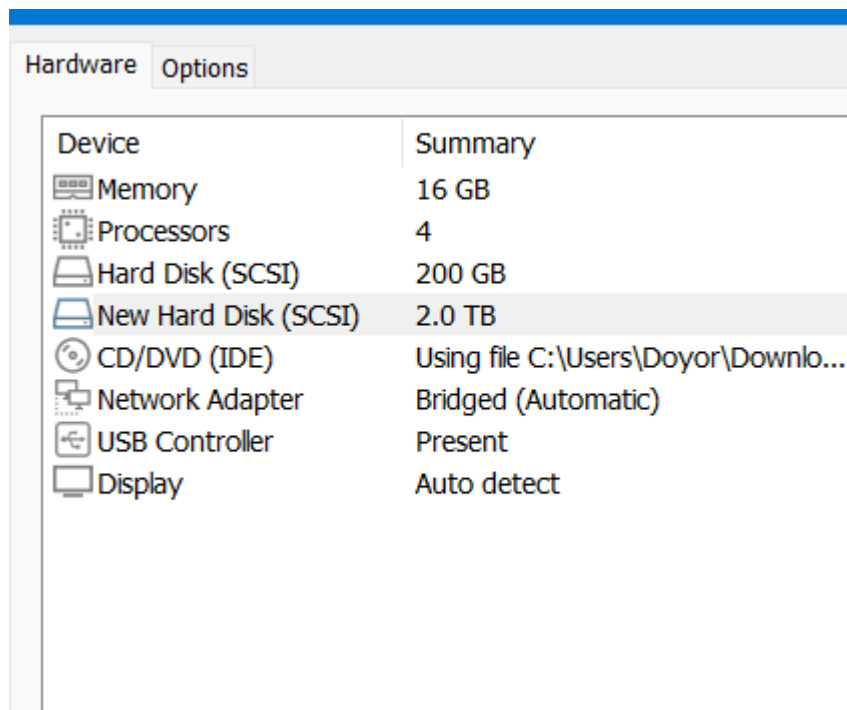
Prêt à terminer

Cliquez sur Terminé pour ajouter l'hôte

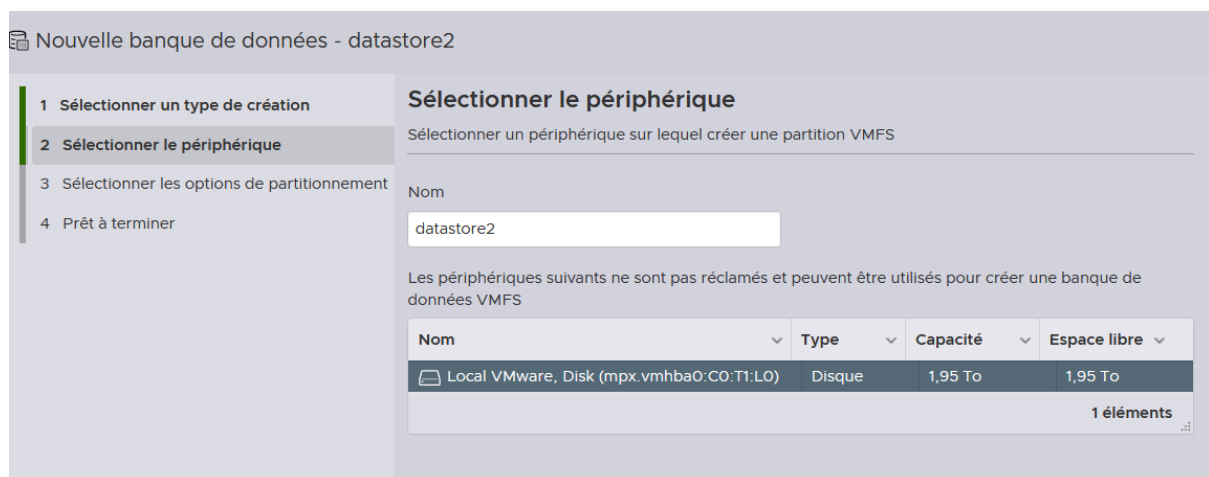
Nom	192.168.4.113
Emplacement	DC1
Version	VMware ESXi 8.0.2 build-22380479
Licence	Licence d'évaluation
Réseaux	VM Network
Banques de données	datastore1
Mode de verrouillage	Désactivé
Emplacement de la VM	DC1
Image for host	



Ajouter un disque sur la machine virtuelle :



Configuration d'un nouveau datastore:



Création d'une Template Ubuntu et Ajout de l'ISO Ubuntu :

Nouvelle machine virtuelle

1 Sélectionner un type de création

2 Sélectionner un nom et un dossier

3 Sélectionner une ressource de calcul

4 Sélectionner un stockage

5 Sélectionner une compatibilité

Sélectionner un système

Sélectionner un nom et un dossier

Spécifiez un nom unique et un emplacement cible

Nom de la machine virtuelle :

Sélectionnez un emplacement pour la machine virtuelle.

192.168.4.200

DC1

vCLS

Nouvelle machine virtuelle

1 Sélectionner un type de création

2 Sélectionner un nom et un dossier

3 Sélectionner une ressource de calcul

4 Sélectionner un stockage

5 Sélectionner une compatibilité

6 Sélectionner un système d'exploitation invité

7 Personnaliser le matériel

8 Prêt à terminer

Sélectionner un stockage

Sélectionner le stockage pour les fichiers de configuration et de disque

☐ Chiffrer la machine virtuelle ⓘ

Stratégie de stockage VM

☐ Désactiver Storage DRS pour cette machine virtuelle

	Nom	Compatibilité de stockage	Capacité	Provisionné	Libre	T ₁
<input type="radio"/>	datastore1	--	71,75 Go	604,33 Go	20,08 Go	V
<input checked="" type="radio"/>	datastore2	--	1,95 To	1,44 Go	1,95 To	V

Gérer Les Colonnes

Éléments par page 10 2 élément(s)

Ajout de l'iso ubuntu à la VM:

Modifier les paramètres | ubuntu_template

Matériel virtuel

Options VM

Paramètres avancés

AJOUTER UN PI

> CPU

2 ⓘ

> Mémoire

2 Go

> Disque dur 1

25 Go

> Contrôleur SCSI 0

Paravirtuel VMware

> Adaptateur réseau 1

VM Network ☒ Connecté

> Lecteur CD/DVD 1

Fichier ISO de la bibliothèque de contenu ☒ Connecté

Statut

☒ Connecter lors de la mise sous tension

Support CD/DVD

[contentLib] /images/ubi

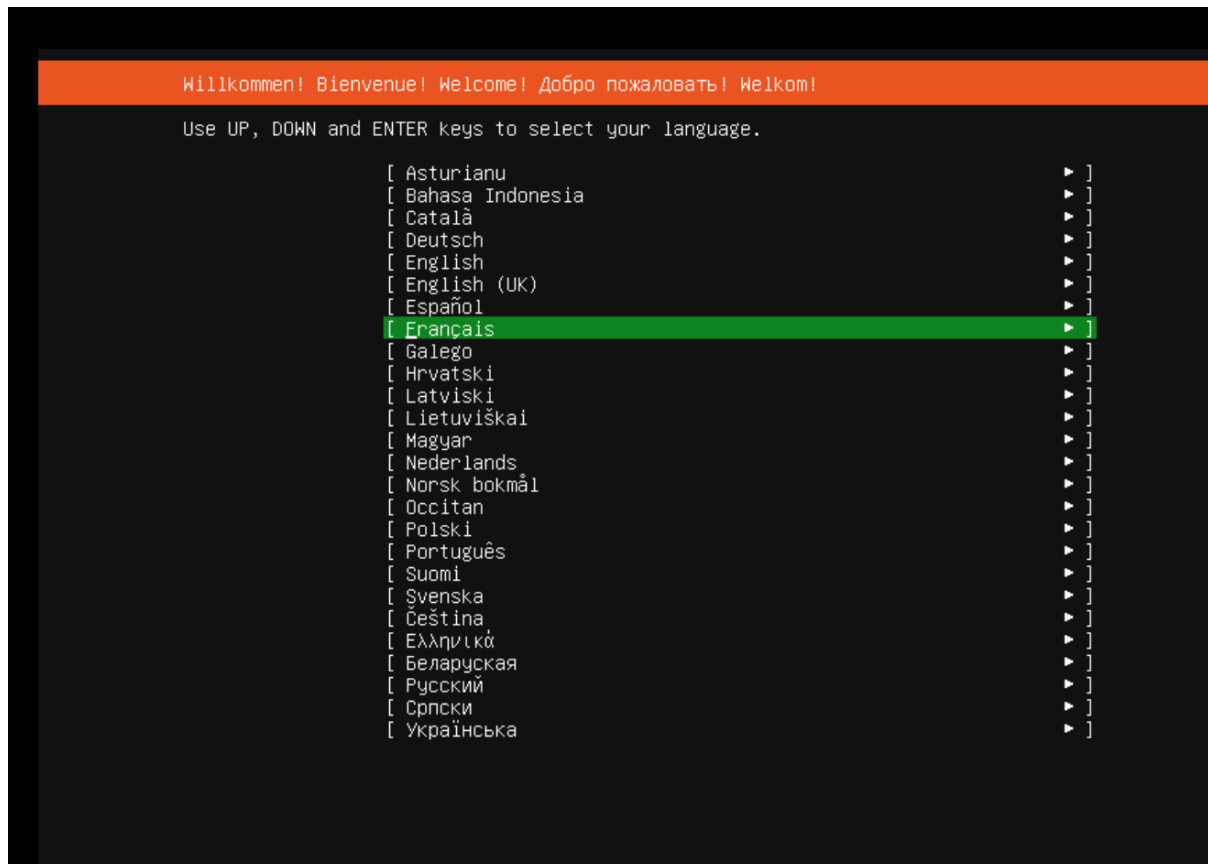
Mode Périphérique

Émuler CD-ROM

Nœud de périphérique virtuel

Contrôleur SATA 0 SATA(0:0) Lecteur CD/DVD 1

Install et configuration de la template Unbuntu:



2. Déploiement des Machines Virtuelles

Configuration Post-Installation : Mises à Jour, Visudo et Clé SSH :

Nous commençons par nous connecter:

```
meriam@meriam:~/HumansBestFriend/esiea-ressources/terraform/terraform-vsphere-kubespray$ ssh kubenode@192.168.4.150
The authenticity of host '192.168.4.150 (192.168.4.150)' can't be established.
ED25519 key fingerprint is SHA256:hwFQm7jq8muwVoGFsMKopEJkOfqTUaYk2Ej1rStHzAY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.4.150' (ED25519) to the list of known hosts.
kubenode@192.168.4.150's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-91-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kubenode@kubenode:~$
```

Nous configurons visudo pour ajouter notre user

\$ sudo visudo

...

kubenode ALL=(ALL) NOPASSWD: ALL

Nous Générons une clé sur machine admin:

```
meriam@meriam:~/HumansBestFriend/esiea-ressources$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/meriam/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/meriam/.ssh/id_rsa
Your public key has been saved in /home/meriam/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:dLYzqDBafj6qK8m5//NjqnlznsKXl3dPnFMlWgQeAmw meriam@meriam
The key's randomart image is:
+---[RSA 3072]-----+
|      . . . . O . . |
|      E   o   o     |
|      . . O . O .   |
|      . + . O . .   |
|      +   S + . .   |
|      + o .   o . o |
| ..O o O . .   =   |
| oo .O.*.o . . . |
| o=*+=+%=o . . . |
+---[SHA256]-----+
```

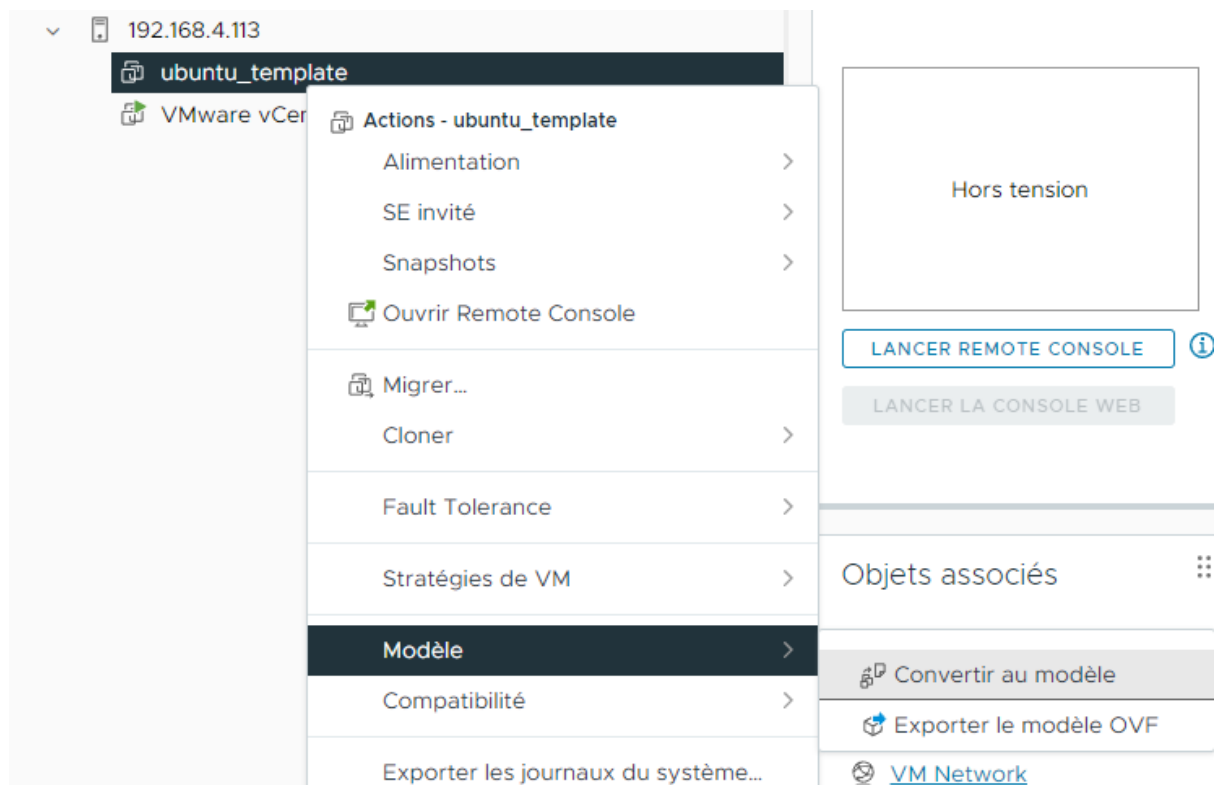
Nous ajoutons ensuite la clé à la template:

```
meriam@meriam:~/HumansBestFriend/esiea-ressources$ ssh-copy-id kubenode@192.168.4.150
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/meriam/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
kubenode@192.168.4.150's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'kubenode@192.168.4.150'"
and check to make sure that only the key(s) you wanted were added.
```

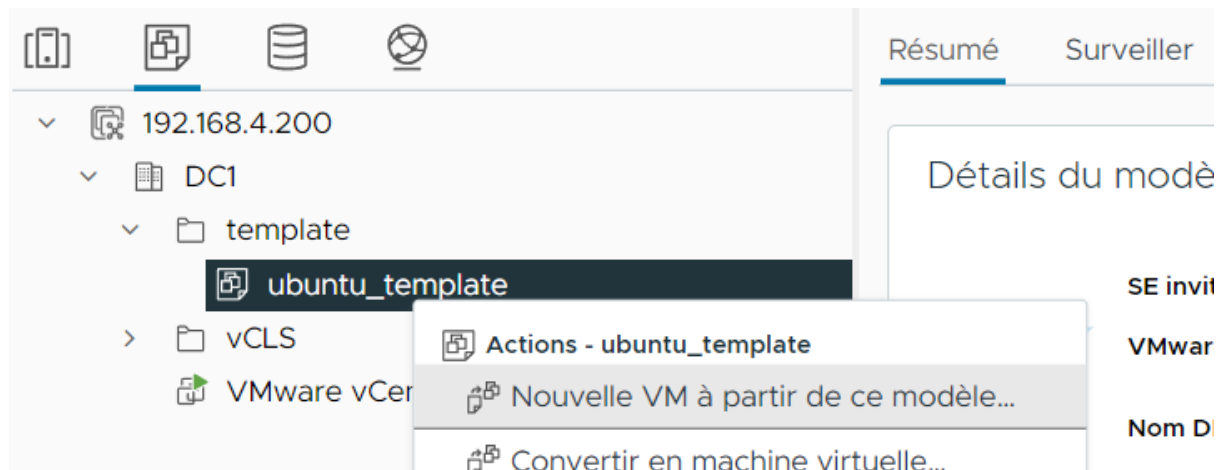
Nous arrêtons la machine et nous la convertissons en template :



Création de Machines Virtuelles pour kubernetes:

Nous aurions pu utiliser terraform ou Tenzu pour provisionner le vcenter mais nous préférons créer directement les machines:

Nous utilisons la méthode manuel :



Préparation des vm de kubernets:

Nous donnons une ip différente à chaque machines et on la nomme en fonction

```
GNU nano 6.2 /etc/hosts *
127.0.0.1 localhost
127.0.1.1 kubenode01

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0  ip6-localnet
ff00::0  ip6-mcastprefix
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
```

```
GNU nano 6.2 /etc
# This is the network config written by 'subiquity'
network:
  ethernets:
    ens33:
      addresses:
        - 192.168.4.151/24
      nameservers:
        addresses:
          - 8.8.8.8
        search: []
      routes:
        - to: default
          via: 192.168.4.1
      version: 2
```

```
GNU nano 6.2 /etc/hostname *
kubenode01
```

3. Mise en Place de Kubernetes avec Kubespray:

Choix du mode d'installation: Utilisation de Kubespray

Nous aurions pu utiliser Tanzu mais nous choisissons d'utiliser kubespray.

#Nous donnons les droits admin sur chaque noeuds pour l'utilisateur kubenode:

```
echo 'kubenode ALL=(ALL) NOPASSWD:ALL' | sudo tee /etc/sudoers.d/kubenode
```

Sur machine admin

Cloner kubespray :

```
$ sudo apt update
```

```
$ sudo apt install git python3 python3-pip -y
```

```
$ git clone https://github.com/kubernetes-incubator/kubespray.git
```

```
$ cd kubespray
```

```
$ pip install -r requirements.txt
```

#Nous installons ansible afin de plus tard pouvoir installer kubernetes sur les machines

```
apt install ansible
```

```
ansible --version
```

```
$ cp -rfp inventory/sample inventory/mycluster
```

```
$ declare -a IPS=(192.168.4.151 192.168.4.152 192.168.4.153)
```

```
$ CONFIG_FILE=inventory/mycluster/hosts.yaml python3  
contrib/inventory_builder/inventory.py ${IPS[@]}
```

#vérification que le fichier d'inventaire a bien été généré :

```
cat inventory/mycluster/hosts.yaml
```

```
ansible all -i inventory/mycluster/hosts.yaml -m shell -a "sudo systemctl stop  
firewalld && sudo systemctl disable firewalld"
```

```
$ ansible all -i inventory/mycluster/hosts.yaml -m shell -a "echo  
'net.ipv4.ip_forward=1' | sudo tee -a /etc/sysctl.conf"
```

```
$ ansible all -i inventory/mycluster/hosts.yaml -m shell -a "sudo sed -i 's/  
swap /  
s/^\(.*\)$/#\1/g' /etc/fstab && sudo swapoff -a"
```

#Lancement du déploiement de kubernetes :

```
ansible-playbook -i inventory/mycluster/hosts.yaml --become --become-user=root  
cluster.yml
```

Nous vérifions l'état des noeuds :

```
East login: sat Dec 18 20:00:20 2021 from 192.168.4.151
kubenode@node1:~$ sudo kubectl get nodes
NAME        STATUS    ROLES    AGE     VERSION
node1       Ready     control-plane  5m5s    v1.28.4
node2       Ready     <none>     4m19s   v1.28.4
node3       Ready     <none>     4m21s   v1.28.4
kubenode@node1:~$
```

Vérification de l'Opérationnalité des Composants Kubernetes

Nous nous assurons que les composants de Kubernetes sont opérationnels (api-server, kubelet, kubeproxy, et ainsi de suite...)

```

root@node1:/home/kubenode/kubemanifest# kubectl get all -n kube-system
NAME                                READY    STATUS    RESTARTS    AGE
pod/calico-kube-controllers-648dff99-pr7sk    1/1     Running   0            124m
pod/calico-node-4zkjh                        1/1     Running   0            124m
pod/calico-node-1k7xn                        1/1     Running   0            124m
pod/calico-node-m5qbl                        1/1     Running   0            124m
pod/coredns-77f7cc69db-fngt7                1/1     Running   0            123m
pod/coredns-77f7cc69db-pdt1z                1/1     Running   0            123m
pod/dns-autoscaler-8576bb9f5b-zgdvw          1/1     Running   0            123m
pod/kube-apiserver-node1                     1/1     Running   1            125m
pod/kube-controller-manager-node1            1/1     Running   2            125m
pod/kube-proxy-bblhr                         1/1     Running   0            124m
pod/kube-proxy-fpl8p                         1/1     Running   0            124m
pod/kube-proxy-jgc9f                         1/1     Running   0            124m
pod/kube-scheduler-node1                     1/1     Running   1            125m
pod/nginx-proxy-node2                       1/1     Running   0            125m
pod/nginx-proxy-node3                       1/1     Running   0            125m
pod/nodelocaldns-c9j7f                       1/1     Running   0            123m
pod/nodelocaldns-frvkr                       1/1     Running   0            123m
pod/nodelocaldns-lcq5l                       1/1     Running   0            123m

NAME                                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)                                AGE
service/coredns                    ClusterIP           10.233.0.3    <none>         53/UDP,53/TCP,9153/TCP               123m

NAME                                DESIRED    CURRENT    READY    UP-TO-DATE    AVAILABLE    NODE
selector                            AGE
daemonset.apps/calico-node          3          3          3        3              3
kubernetes.io/os=linux              124m
daemonset.apps/kube-proxy            3          3          3        3              3
kubernetes.io/os=linux              125m
daemonset.apps/nodelocaldns          3          3          3        3              3
kubernetes.io/os=linux              123m

NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/calico-kube-controllers    1/1     1              1            124m
deployment.apps/coredns                    2/2     2              2            123m
deployment.apps/dns-autoscaler             1/1     1              1            123m

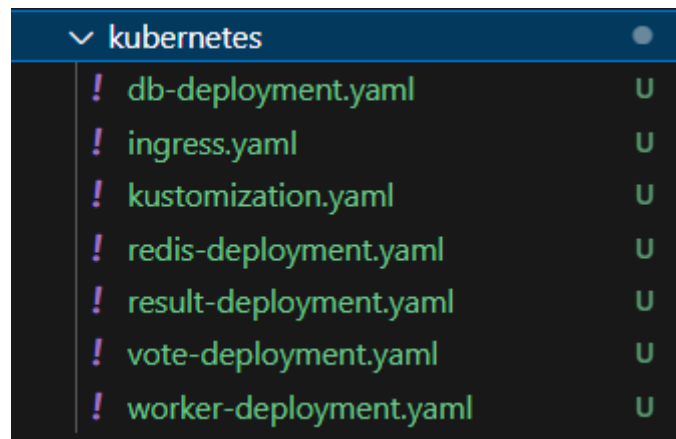
NAME                                DESIRED    CURRENT    READY    AGE
replicaset.apps/calico-kube-controllers-648dff99    1          1          1            124m
replicaset.apps/coredns-77f7cc69db                2          2          2            123m
replicaset.apps/dns-autoscaler-8576bb9f5b          1          1          1            123m

```

4. Lancement de l'Infrastructure Kubernetes :

Conversion des Fichiers Docker Compose en Manifestes Kubernetes :

Nous devons convertir manuellement les fichiers docker compose en fichiers manifests compatible avec kube. Pour cela nous créons pour chaque service un fichier manifest:



Exemple:

```
esiea-ressources > kubernetes > ! db-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: db
5  spec:
6    selector:
7      matchLabels:
8        app: db
9    template:
10     metadata:
11       labels:
12         app: db
13     spec:
14       containers:
15         - name: db
16           image: postgres:15-alpine
17           env:
18             - name: POSTGRES_PASSWORD
19               value: postgres
20 ---
21 apiVersion: v1
22 kind: Service
23 metadata:
24   name: db
25 spec:
26   selector:
27     app: db
28   ports:
29     - protocol: TCP
30       port: 5432
31       targetPort: 5432
32
```

Lancement de l'infrastructure:

Nous lançons l'infrastructure avec la commande : *kubectl apply -k* . Nous pouvons ensuite vérifier que l'infrastructure s'est bien lancée

État de l'infrastructure:

```
worker-69974bbcd5-7jg4d 0/1 ContainerCreating 0 9s
kubenode@node1:~/kubemanimfest$ sudo kubectl get pod
NAME READY STATUS RESTARTS AGE
db-6cfd88fbf4-bztmm 0/1 ContainerCreating 0 12s
redis-7c888f4788-4j9pf 0/1 ContainerCreating 0 12s
result-5488dd99d5-dv6xz 0/1 ContainerCreating 0 12s
vote-5455668bd4-gh2n6 0/1 ContainerCreating 0 11s
worker-69974bbcd5-7jg4d 0/1 ContainerCreating 0 11s
kubenode@node1:~/kubemanimfest$
```

Nous faisons bien attention à appeler les services kube comme sur docker pour que les images docker puissent fonctionner sans erreur de noms.

```
19 | | | | value: postgres
20 | ---
21 | apiVersion: v1
22 | kind: Service
23 | metadata:
24 |   name: db
25 | spec:
26 |   selector:
27 |     app: db
28 |   ports:
29 |     - protocol: TCP
30 |       port: 5432
31 |       targetPort: 5432
32
```

Configuration de l'accès au site Web:

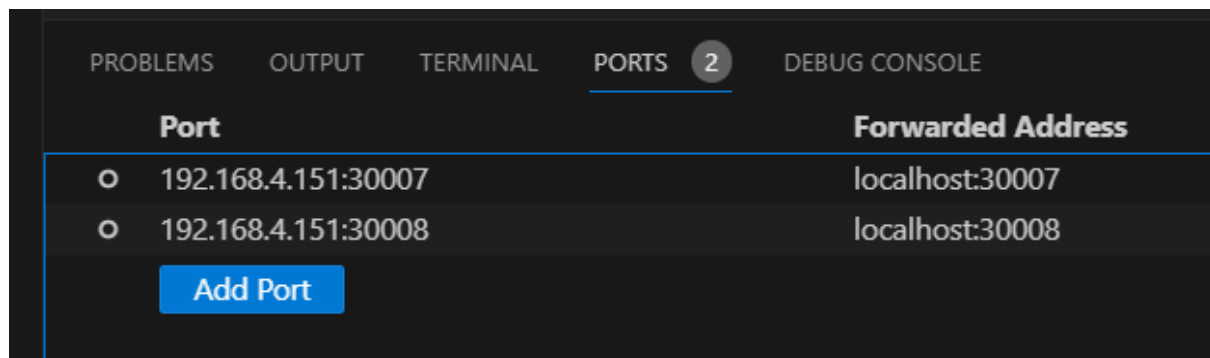
Pour accéder aux applications depuis l'extérieur on peut soit utiliser une ingress avec un load balancer ou bien utiliser un service NodePort kube.

Pour des raison de simplicité nous allons utiliser NodePort:

```
19 ---
20 apiVersion: v1
21 kind: Service
22 metadata:
23   name: vote
24 spec:
25   type: NodePort
26   selector:
27     app: vote
28   ports:
29     - protocol: TCP
30       port: 80
31       targetPort: 80
32       nodePort: 30008
```

Le stockage persistant a aussi été enlevé pour des raisons de simplicité

Nous établissons un simple tunnel ssh vers n'importe qu'elle noeuds kubernetes afin de faciliter l'accès :



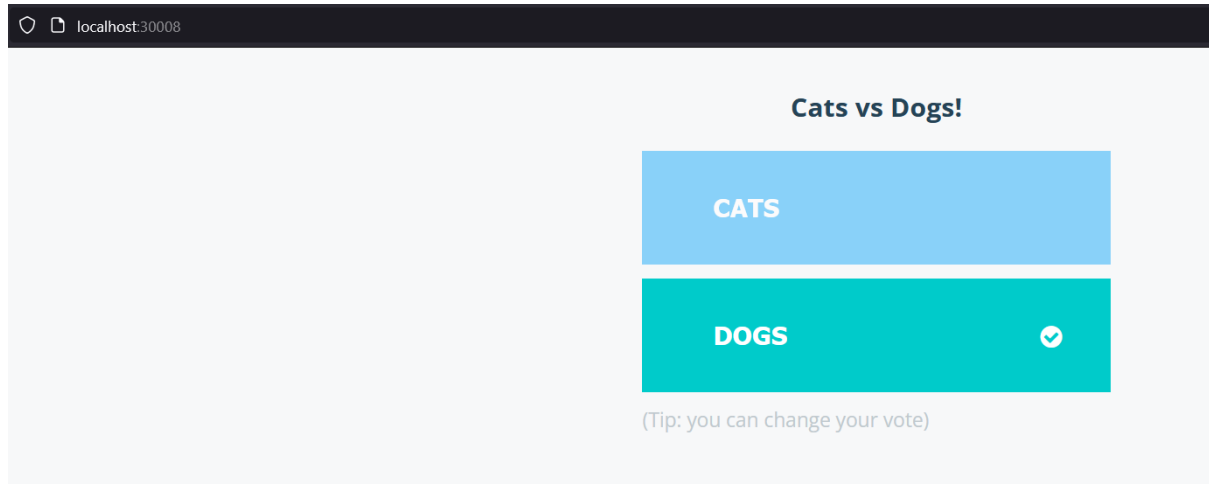
The screenshot shows the 'PORTS' tab with two entries:

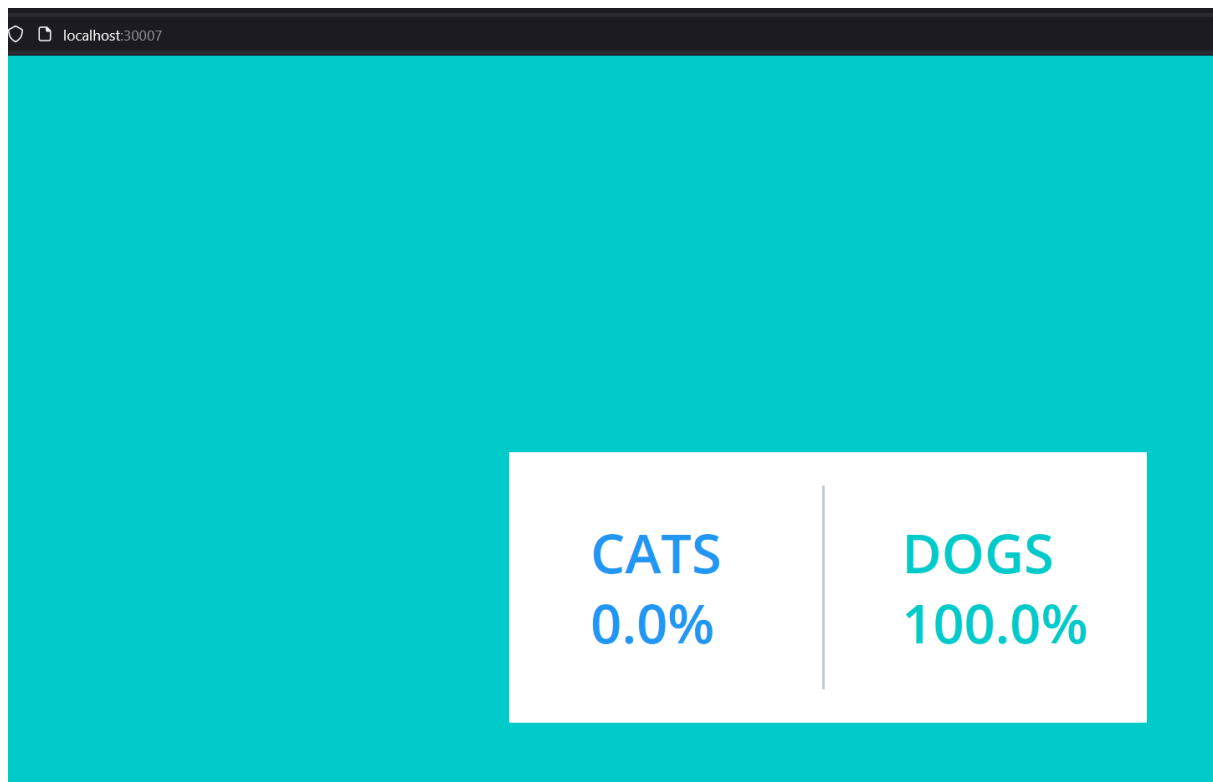
Port	Forwarded Address
192.168.4.151:30007	localhost:30007
192.168.4.151:30008	localhost:30008

Below the table is an 'Add Port' button.

Captures des sites web une fois que les conteneurs sont opérationnels

Illustration visuelle des sites web capturés après le déploiement réussi des conteneurs, mettant en avant le processus de vote pour les chiens.





5. Lien github : <https://github.com/Merimiam/esiea-ressources>

Meriam Elkhia
Leon Nadot

Julien Taveau
Ruben Woliner