

# Déploiement Automatisé d'un Site de Documentation avec MCO/MCS

Valentin LEGRAND, devoir rendu le 20 juin 2025

## Phase 1 : Préparation des Machines

### ◆ Étape 1 : Installation de l'OS

✅ Choix de l'OS (Ubuntu/Debian recommandé) :

Debian 12` (recommandé)

🔧🔒 Installation de base avec SSH activé :

Installation par défaut avec la suppression de la partition swap pour kubernetes

SSH activé :

```
ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
Active: active (running) since Tue 2025-04-15 09:29:13 CEST; 5min ago
```

📦 Mise à jour des paquets :

```
apt-get update
apt-get upgrade
```

## ◆ Étape 2 : Configuration initiale des machines

### 🔧 Configuration du PATH (commandes manquantes) :

```
export PATH=$PATH:/usr/sbin #Ajout de ce dossier au path pour avoir des commandes  
nécessaire à la suite du projet
```

### 🔧 Ajout des utilisateurs :

```
adduser val_master      # VM Master  
adduser val_worker1    # VM Worker Node 1  
adduser val_worker2    # VM Worker Node 2
```

### 🔧 Ajout des utilisateurs au groupe sudo :

```
usermod -aG sudo val_master      # VM Master  
usermod -aG sudo val_worker1    # VM Worker Node 1  
usermod -aG sudo val_worker2    # VM Worker Node 2
```

### 🌐 Configuration réseau & hostnames :

```
hostnamectl set-hostname Master    #VM Master  
hostnamectl set-hostname Worker1   #VM Worker Node 1  
hostnamectl set-hostname Worker2   #VM Worker Node 2
```

### 🔑 Génération de clé SSH + copie :

```
ssh-keygen -t rsa -b 4096  # Avec passphrase "valentin"  
  
ssh-copy-id val_worker1@192.168.142.144  
ssh-copy-id val_worker2@192.168.142.143  
ssh-copy-id val_master@192.168.142.137
```

### 📄 Résolution de nom ( /etc/hosts ) :

```
192.168.142.137 Master  
192.168.142.144 Worker1  
192.168.142.143 Worker2
```

 **Fichier de configuration SSH ( /etc/ssh/sshd\_config ) :**

```
PubKeyAuthentication yes
PermitRootLogin no
PasswordAuthentication no
```


 **Vérification SSH sans mot de passe via des clés publiques :**

```
ssh val_worker1@192.168.142.142
# → Enter passphrase...

ssh root@192.168.142.142
# → Permission denied (clé requise)
```

---

## ◆ Étape 3 : Installation des prérequis

 **Installation de python :**

```
sudo apt install python3 python3-pip
```


 **Vérification de la connectivité entre les machines :**

```
val_master# ping 192.168.142.144
val_master# ping 192.168.142.143
```

---

# Phase 2 : Installation & Configuration de Docker

## ◆ Étape 1 : Installation de Docker

 [Guide IT-Connect Docker Debian](#)

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common
sudo curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
sudo echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
sudo systemctl enable docker
sudo systemctl status docker
```

## ◆ Étape 2 : Ajout des utilisateurs à Docker (pour éviter l'utilisation de sudo)

 Ajouter l'utilisateur au groupe `docker` :

```
usermod -aG docker ${USER}
```

## ◆ Étape 3 : Vérification du bon fonctionnement de Docker

✓ Tester le fonctionnement :

```
docker run hello-world
```

# Phase 3 : Déploiement de Kubernetes

## ◆ Étape 1 : Installation de Kubernetes (kubeadm, kubelet, kubectl)

🔧 Préparation :

```
sudo modprobe br_netfilter
```

📦 [Installer Kubernetes via kubeadm](#)

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl gpg
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.33/deb/Release.key | sudo gpg --dearmor
-o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.33/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
sudo systemctl enable --now kubelet
```

📦 [Télécharger cri-dockerd](#)

```
wget https://github.com/Mirantis/cri-dockerd/releases/download/v0.3.17/cri-
dockerd_0.3.17.3-0.debian-bookworm_amd64.deb
sudo dpkg -i cri-dockerd_0.3.17.3-0.debian-bookworm_amd64.deb
```

## ◆ Étape 2 : Initialisation du cluster

🚀 Initialiser le cluster :

```
sudo kubeadm init --cri-socket unix:///var/run/cri-dockerd.sock
```

🔧 Configurer kubectl :

```
# Si on est en root
export KUBECONFIG=/etc/kubernetes/admin.conf
```

```
# Si on est pas root
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

---

## ◆ Étape 3 : Ajout des nœuds au cluster

Connexion des workers au noeud master :

```
kubeadm join 192.168.142.137:6443 --token lyu32e.828538t417gn9omw \
--discovery-token-ca-cert-hash
sha256:d49e4ea40e6f390e95ad78ce10441809a5a8fb56eccbbac2f33f89fb2ea81137 --cri-socket
unix:///var/run/cri-dockerd.sock
```

🌐 Déployer Flannel (réseau Pods) :

```
kubectll apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-
flannel.yml # Sur le master
```

---

## ◆ Étape 4 : Vérification du bon fonctionnement du cluster

📋 Vérification :

```
kubectll get nodes
kubectll get pods
```

# Phase 4 : Déploiement du Site Web

## ◆ Étapes 1 : Écriture d'un Dockerfile (pour contenir l'application web statique)

📄 Écriture du `Dockerfile` pour app web statique :

```
# Utiliser la dernière image officielle de Nginx
FROM nginx:latest

# Copier le répertoire contenant le html,css,js dans le répertoire de service de Nginx
COPY Website_content /usr/share/nginx/html

# Copier notre default.conf dans le répertoire de service de Nginx
COPY Config/nginx/conf/default.conf /etc/nginx/conf.d/default.conf

# Copier le fichier PDF dans le répertoire de service de Nginx
COPY Website_content/rapport.pdf /usr/share/nginx/html/mon_fichier.pdf

# Copier les certificats SSL dans le répertoire de service de Nginx
COPY Config/nginx/certs/server.crt /etc/nginx/certs/server.crt
COPY Config/nginx/certs/server.key /etc/nginx/certs/server.key
```

## ◆ Étapes 2 : Création d'un docker-compose.yml pour l'orchestration locale

📄 Création du `docker-compose.yml` :

```
services:
  static-app:
    build: .
    ports:
      - "80:80"
      - "443:443"
    restart: always
```

## ◆ Étapes 3 : Déploiement initial avec Docker Compose pour validation

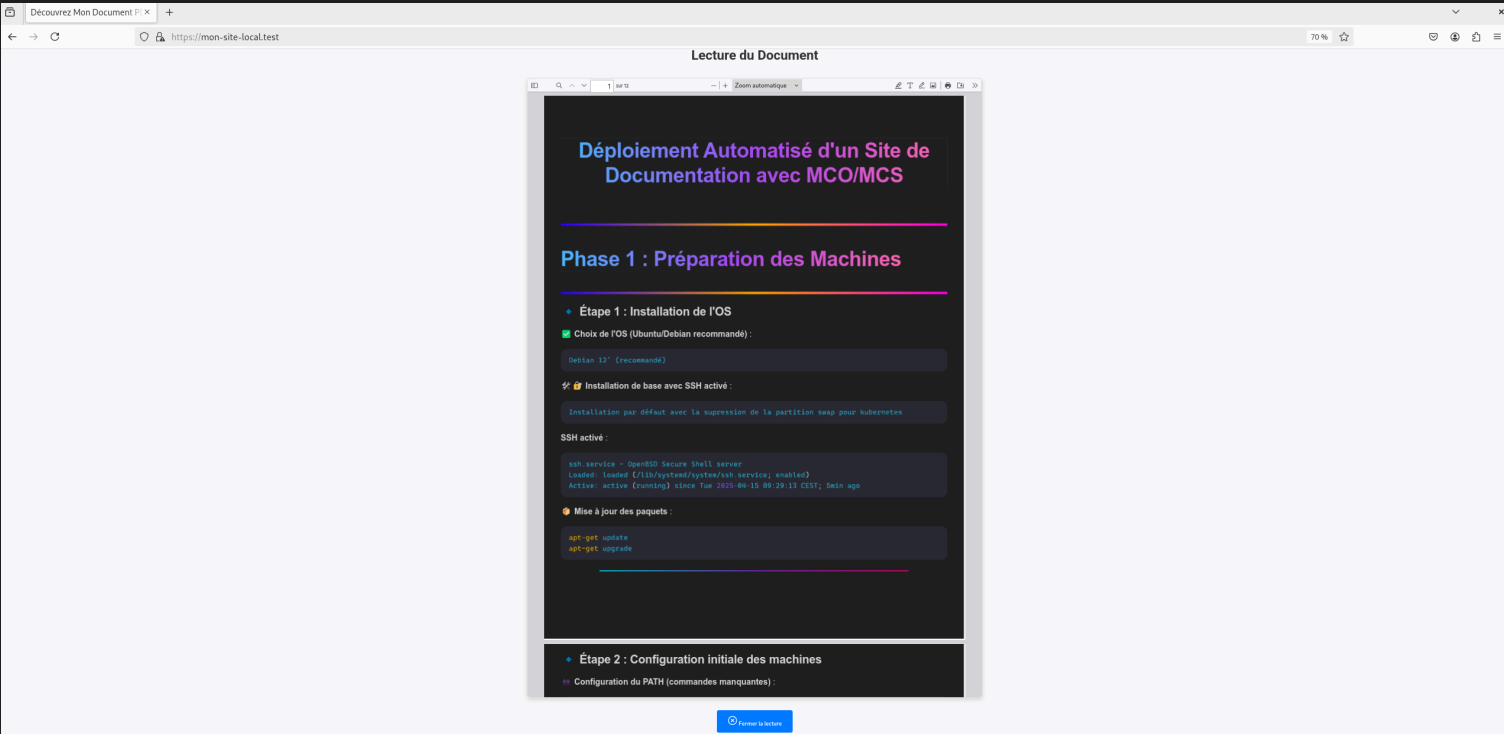
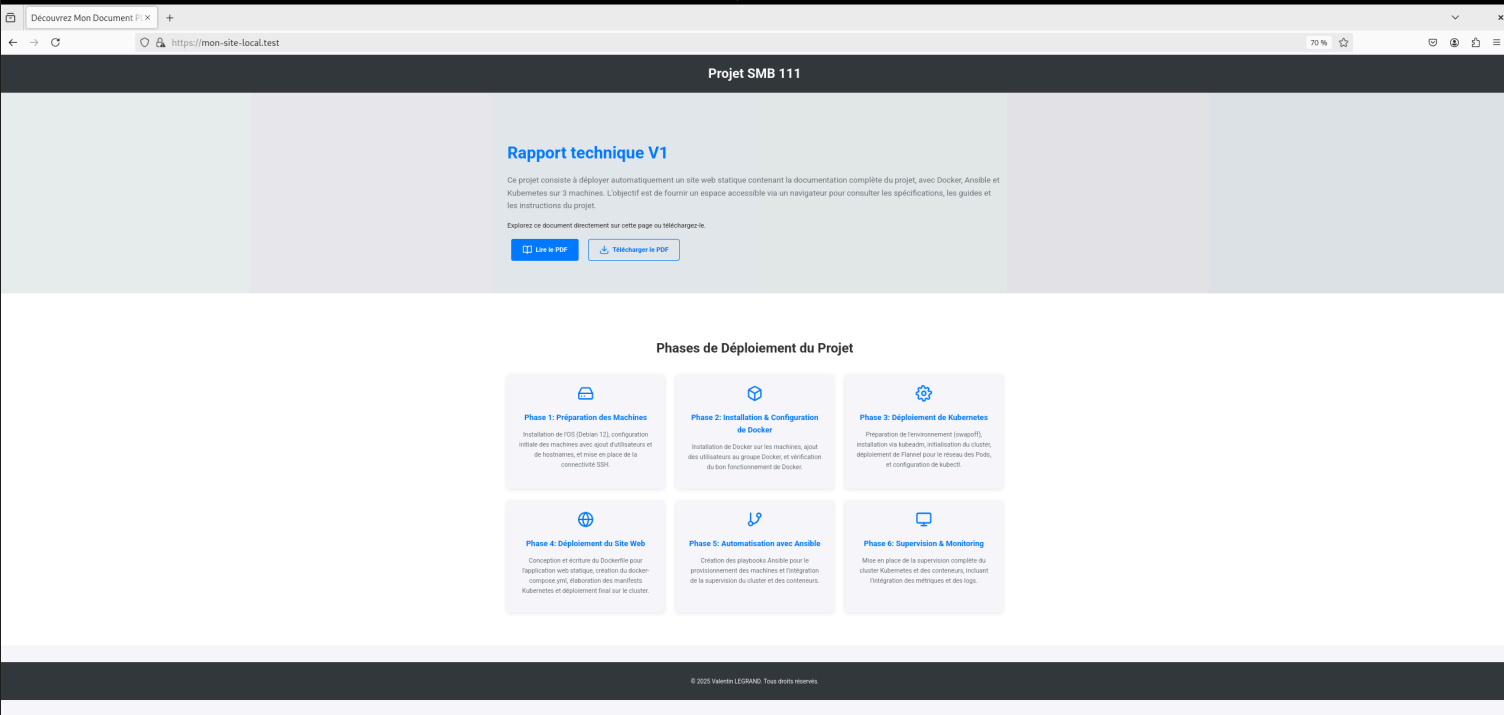
🔧 Déploiement local avec Docker Compose :

docker compose up --build

✓ \*\* Test des accès au site :

<https://mon-site-local.test:443>

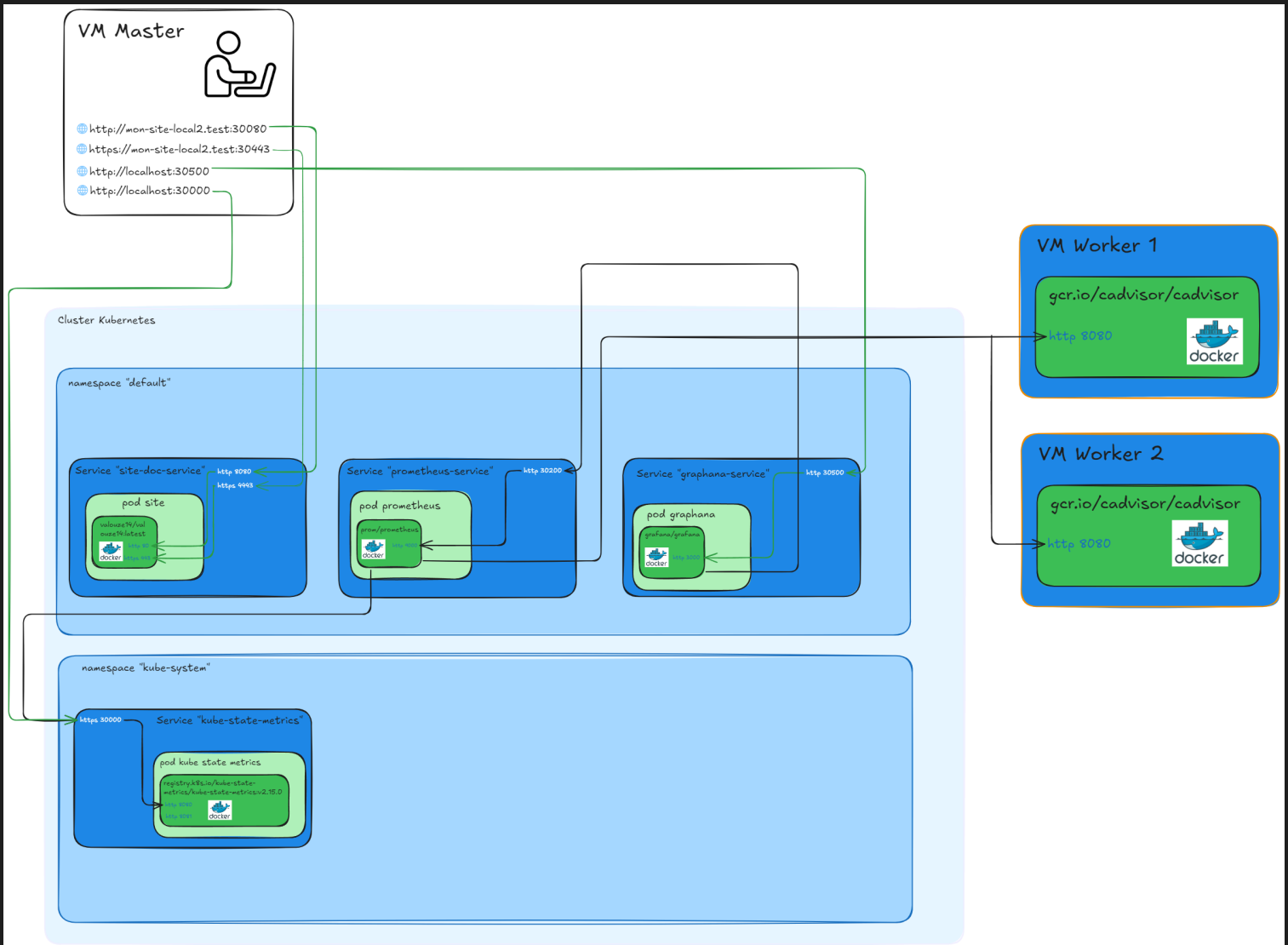
<http://mon-site-local.test:80>



◆ Étapes 4 : Création des manifests Kubernetes pour le déploiement sur le cluster



## 🚀🚀🚀 Schéma de l'architecture logiciel :



🌐 \*\*Push de l'image docker issue du Dockerfile de l'étape 1 sur docker hub afin de pouvoir l'utiliser directement dans les pods.

### 📄 Création du deployment `site-deployment.yaml` :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: site-doc-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: site-doc
  template:
    metadata:
      labels:
        app: site-doc
    spec:
      containers:
```

- name: site
- image: valouze14/valouze14:latest
- ports:
  - containerPort: 80
  - containerPort: 443

### Création du service `site-service.yaml` :

```
apiVersion: v1
kind: Service
metadata:
  name: site-doc-service
spec:
  selector:
    app: site-doc # Cible tous les pods qui ont le labels site-doc
  type: NodePort
  ports:
    - name: http
      protocol: TCP
      port: 8080 # Le port sur lequel le service écoutera à l'intérieur du cluster
      targetPort: 80 # Le port sur lequel votre application écoute à l'intérieur du pod
      # (port 80 de votre conteneur)
      nodePort: 30080 # Un port entre 30000 et 32767. Choisissez-en un disponible.
    - name: https
      protocol: TCP
      port: 4443
      targetPort: 443
      nodePort: 30443
```

---

## ◆ Étapes 5 : Déploiement final sur Kubernetes

### Déploiement final sur Kubernetes :

```
kubectl apply -f /home/val_master/Documents/Projet/Project-docs/Kubernetes/App/Site
```

### Test des accès au site :

<https://mon-site-local.test2:30443>

<http://mon-site-local.test2:30080>

## Rapport technique V1

Ce projet consiste à déployer automatiquement un site web statique contenant la documentation complète du projet, avec Docker, Ansible et Kubernetes sur 3 machines. L'objectif est de fournir un espace accessible via un navigateur pour consulter les spécifications, les guides et les instructions du projet.

Explorez ce document directement sur cette page ou téléchargez-le.

Lire le PDF

Télécharger le PDF

### Phases de Déploiement du Projet



#### Phase 1: Préparation des Machines

Installation de l'OS (Debian 12), configuration initiale des machines avec ajout d'utilisateurs et de hostnames, et mise en place de la connectivité SSH.



#### Phase 2: Installation & Configuration de Docker

Installation de Docker sur les machines, ajout des utilisateurs au groupe Docker, et vérification du bon fonctionnement de Docker.



#### Phase 3: Déploiement de Kubernetes

Préparation de l'environnement (swapoff), installation via kubeadm, initialisation du cluster, déploiement de Flannel pour le réseau des Pods, et configuration de kubect.



#### Phase 4: Déploiement du Site Web

Conception et écriture du Dockerfile pour l'application web statique, création du docker-compose.yml, élaboration des manifests Kubernetes et déploiement final sur le cluster.



#### Phase 5: Automatisation avec Ansible

Création des playbooks Ansible pour le provisionnement des machines et l'intégration de la supervision du cluster et des conteneurs.



#### Phase 6: Supervision & Monitoring

Mise en place de la supervision complète du cluster Kubernetes et des conteneurs, incluant l'intégration des métriques et des logs.

## Rapport technique V1

Ce projet consiste à déployer automatiquement un site web statique contenant la documentation complète du projet, avec Docker, Ansible et Kubernetes sur 3 machines. L'objectif est de fournir un espace accessible via un navigateur pour consulter les spécifications, les guides et les instructions du projet.

Explorez ce document directement sur cette page ou téléchargez-le.

Lire le PDF

Télécharger le PDF

### Phases de Déploiement du Projet



#### Phase 1: Préparation des Machines

Installation de l'OS (Debian 12), configuration initiale des machines avec ajout d'utilisateurs et de hostnames, et mise en place de la connectivité SSH.



#### Phase 2: Installation & Configuration de Docker

Installation de Docker sur les machines, ajout des utilisateurs au groupe Docker, et vérification du bon fonctionnement de Docker.



#### Phase 3: Déploiement de Kubernetes

Préparation de l'environnement (swapoff), installation via kubeadm, initialisation du cluster, déploiement de Flannel pour le réseau des Pods, et configuration de kubect.



#### Phase 4: Déploiement du Site Web

Conception et écriture du Dockerfile pour l'application web statique, création du docker-compose.yml, élaboration des manifests Kubernetes et déploiement final sur le cluster.



#### Phase 5: Automatisation avec Ansible

Création des playbooks Ansible pour le provisionnement des machines et l'intégration de la supervision du cluster et des conteneurs.



#### Phase 6: Supervision & Monitoring

Mise en place de la supervision complète du cluster Kubernetes et des conteneurs, incluant l'intégration des métriques et des logs.

✓ Vérification load balancer : \*\*

Par défaut, le service distribue sans affinité aux pods associés ce qui fait un load-balancer naturel, je peux vérifier en regardant en direct les logs des trois pods qui héberge mon site. A chaque fois, il faut supprimer les données dans le cache à pour établir une nouvelle session. Je constate qu'à chaque nouvelle session, les requêtes n'arrivent jamais sur le même pod.

```
kubectl logs -f site-doc-deployment-7887cccc5b-mp5vt
```

```
val_master@Master:~/Documents/Projet/Project-docs/Kubernetes/App$ kubectl logs -f site-doc-deployment-7887cccc5b-mp5vt
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/06/18 09:21:43 [notice] 1#1: using the "epoll" event method
2025/06/18 09:21:43 [notice] 1#1: nginx/1.27.5
2025/06/18 09:21:43 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2025/06/18 09:21:43 [notice] 1#1: OS: Linux 6.1.0-37-amd64
2025/06/18 09:21:43 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/06/18 09:21:43 [notice] 1#1: start worker processes
2025/06/18 09:21:43 [notice] 1#1: start worker process 29
2025/06/18 09:21:43 [notice] 1#1: start worker process 30
2025/06/18 09:21:43 [notice] 1#1: start worker process 31
2025/06/18 09:21:43 [notice] 1#1: start worker process 32
10.244.0.0 - - [18/Jun/2025:13:17:11 +0000] "GET / HTTP/1.1" 200 8938 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0" "-"
```

---

# Phase 5 : Automatisation avec Ansible

## Dépôt des ressources sur Github

```
git clone https://github.com/Valouze14/Projet.git
```

## Création du fichier d'inventories.ini :

```
[all]
192.168.142.137 ansible_user=val_master # VM Master
192.168.142.144 ansible_user=val_worker1 # VM Worker Node 1
192.168.142.143 ansible_user=val_worker2 # VM Worker Node 2

[masters]
192.168.142.137

[workers]
192.168.142.144
192.168.142.143
```

## ◆ Étapes 1 : Création des playbooks Ansible pour automatiser l'installation et la configuration

## Création du fichier setup-hosts.yml :

```
- name: Setup Hosts
  hosts: all
  become: true
  tasks:

#####
# Ajouter les entrées des hôtes et le nom de domaine dans le fichier /etc/hosts
#####
- name: Définir les entrées des hôtes + nom de domaine du site web
  blockinfile:
    path: /etc/hosts
    block: |
      192.168.142.137 Master
      192.168.142.144 Worker1
      192.168.142.143 Worker2
      192.168.142.137 mon-site-local.test
```

backup: true

```
#####
```

```
# Définir le nom d'hôte pour la VM Master
```

```
#####
```

```
- name: Définir le nom d hôte pour la VM Master
```

```
hostname:
```

```
  name: Master
```

```
when: inventory_hostname == '192.168.142.137'
```

```
#####
```

```
# Définir le nom d'hôte pour la VM Worker1
```

```
#####
```

```
- name: Définir le nom d hôte pour la VM Worker 1
```

```
hostname:
```

```
  name: Worker1
```

```
when: inventory_hostname == '192.168.142.144'
```

```
#####
```

```
# Définir le nom d'hôte pour la VM Worker2
```

```
#####
```

```
- name: Définir le nom d hôte pour la VM Worker 2
```

```
hostname:
```

```
  name: Worker2
```

```
when: inventory_hostname == '192.168.142.143'
```

```
#####
```

```
# Installer Python 3 et pip (prérequis pour d autres tâches Ansible)
```

```
#####
```

```
- name: Installer python3 et python3-pip
```

```
apt:
```

```
  name:
```

```
    - python3
```

```
    - python3-pip
```

```
#####
```

```
# Charger le module noyau br_netfilter (nécessaire pour Kubernetes)
```

```
#####
```

```
- name: Charger le module br_netfilter
```

```
modprobe:
```

```
  name: br_netfilter
```

```
  state: present
```

```
#####
```

```
# Mettre à jour la liste des paquets et effectuer une mise à jour complète
```

```
#####
```

```
- name: Mise à jour de la liste des paquets
```

```
apt:
```

```
  update_cache: true
```

```
  upgrade: dist
```

## Création du fichier install-docker.yml :

```
- name: Install Docker
  hosts: all
  become: true
  tasks:

#####
# Installer les paquets nécessaires au fonctionnement de Docker
#####
- name: Installer les paquets nécessaires pour Docker
  apt:
    name:
      - apt-transport-https
      - ca-certificates
      - curl
      - gnupg2
      - software-properties-common
    state: present
    update_cache: true

#####
# Ajouter la clé GPG officielle de Docker pour sécuriser le dépôt
#####
- name: Ajouter la clé GPG officielle de Docker
  shell: curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
  args:
    creates: /usr/share/keyrings/docker-archive-keyring.gpg

#####
# Ajouter le dépôt officiel Docker à la liste des sources APT
#####
- name: Ajouter le dépôt Docker à la liste des sources APT
  lineinfile:
    path: /etc/apt/sources.list.d/docker.list
    line: deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/debian {{ ansible_distribution_release }} stable
    create: true
    mode: '0644'

#####
# Mettre à jour l'index des paquets après ajout du dépôt Docker
#####
- name: Mettre à jour l'index des paquets après l'ajout du dépôt Docker
  apt:
    update_cache: true
```

```
#####
# Installer Docker Engine, l'interface en ligne de commande et containerd
#####
- name: Installer Docker Engine, CLI et Containerd
  apt:
    name:
      - docker-ce
      - docker-ce-cli
      - containerd.io
    state: present

#####
# Activer et démarrer le service Docker au démarrage de la machine
#####
- name: Vérifier que le service Docker est démarré et activé au démarrage
  systemd:
    name: docker
    state: started
    enabled: true

#####
# Vérifier le statut du service Docker (diagnostic uniquement)
#####
- name: Vérifier le statut du service Docker (pour information, ne modifie rien)
  command: systemctl status docker
  register: docker_status

#####
# Afficher le statut du service Docker dans la sortie Ansible
#####
- name: Afficher le statut du service Docker
  debug:
    var: docker_status.stdout_lines

#####
# Ajouter le user val_master au groupe docker (accès sans sudo)
#####
- name: Ajouter val_master au groupe docker
  user:
    name: val_master
    groups: docker
    append: true
  when: inventory_hostname == '192.168.142.137'

#####
# Ajouter le user utilisateur val_worker1 au groupe docker
#####
- name: Ajouter val_worker1 au groupe docker
  user:
    name: val_worker1
```



```

    groups: docker
    append: true
    when: inventory_hostname == '192.168.142.144'

#####
# Ajouter l'utilisateur val_worker2 au groupe docker
#####
- name: Ajouter val_worker2 au groupe docker
  user:
    name: val_worker2
    groups: docker
    append: true
  when: inventory_hostname == '192.168.142.143'

#####
# Vérifier que Docker fonctionne avec un conteneur de test
#####
- name: Exécuter docker run hello-world pour vérifier la bonne installation
  command: docker run hello-world
  become: false
  register: docker_hello_world_result

#####
# Afficher le résultat du test docker run hello-world
#####
- name: Afficher le résultat de docker run hello-world
  debug:
    var: docker_hello_world_result.stdout_lines

#####
# Télécharger le binaire cri-dockerd depuis GitHub
#####
- name: Télécharger le paquet cri-dockerd
  get_url:
    url: https://github.com/Mirantis/cri-dockerd/releases/download/v0.3.17/cri-
dockerd_0.3.17.3-0.debian-bookworm_amd64.deb
    dest: /tmp/cri-dockerd_0.3.17.3-0.debian-bookworm_amd64.deb
    mode: '0644'

#####
# Installer le paquet cri-dockerd téléchargé localement
#####
- name: Installer le paquet cri-dockerd
  ansible.builtin.apt:
    deb: /tmp/cri-dockerd_0.3.17.3-0.debian-bookworm_amd64.deb

#####
# Lancer Docker Compose pour déployer les conteneurs du projet
#####
- name: Lancer docker compose up avec le répertoire de projet spécifié

```

```

ansible.builtin.command: docker compose up --build -d
args:
  chdir: /home/val_master/Documents/Projet/Project-docs/    # Spécifie le répertoire
de travail
  when: inventory_hostname == '192.168.142.137' # Exécuter uniquement sur le nœud
maître

```

## Création du fichier install-k8s.yml :

```

#####
# 1. Installation des prérequis Kubernetes
#####
- name: Installer les paquets et composants Kubernetes
  hosts: all
  become: true
  tasks:
    - name: Installer les paquets nécessaires
      apt:
        name:
          - apt-transport-https
          - ca-certificates
          - curl
          - gpg
        state: present

    - name: Ajouter la clé GPG officielle de Kubernetes
      shell: >
        curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.33/deb/Release.key |
        gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
      args:
        creates: /etc/apt/keyrings/kubernetes-apt-keyring.gpg

    - name: Ajouter le dépôt Kubernetes
      lineinfile:
        path: /etc/apt/sources.list.d/kubernetes.list
        line: deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.33/deb/ /
        create: true
        mode: '0644'

    - name: Mise à jour des paquets
      apt:
        update_cache: true

    - name: Installer kubelet, kubeadm et kubectl
      apt:
        name:
          - kubelet
          - kubeadm

```

```

    - kubectl
    state: present

- name: Marquer kubelet, kubeadm et kubectl en hold
  dpkg_selections:
    name: '{{ item }}'
    selection: hold
  loop:
    - kubelet
    - kubeadm
    - kubectl

- name: Activer et démarrer le service kubelet
  systemd:
    name: kubelet
    enabled: true
    state: started

#####
# 2. Initialisation du master Kubernetes
#####
- name: Initialiser le master Kubernetes
  hosts: 192.168.142.137
  become: true
  tasks:
    - name: Initialiser le cluster avec kubeadm
      command: kubeadm init --pod-network-cidr=10.244.0.0/16 --cri-socket
unix:///var/run/cri-dockerd.sock
      tags: [kubeadm_init]

- name: Créer .kube/config pour le user val_master
  shell: |
    mkdir -p /home/val_master/.kube
    cp -i /etc/kubernetes/admin.conf /home/val_master/.kube/config
    chown val_master:val_master /home/val_master/.kube/config

- name: Appliquer le manifeste Flannel CNI
  ansible.builtin.shell:
    kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
  become: false

#####
# 3. Récupération de la commande de jointure
#####
- name: Obtenir la commande de jointure
  hosts: 192.168.142.137
  become: true
  tasks:

```

```

- name: Générer la commande kubeadm join
  command: kubeadm token create --print-join-command
  register: join_command
  run_once: true

- name: Sauvegarder la commande dans un fact partagé
  set_fact:
    kube_join_cmd: "{{ join_command.stdout }} --cri-socket unix:///var/run/cri-
dockerd.sock"
  run_once: true

#####
# 4. Faire joindre les workers au cluster
#####
- name: Faire rejoindre les workers au cluster Kubernetes
  hosts: all
  become: true
  tasks:
    - name: Joindre les workers (sauf master)
      command: "{{ hostvars['192.168.142.137'].kube_join_cmd }}"
      when: inventory_hostname != '192.168.142.137'
      register: kubeadm_join_output_workers
      tags: [kubeadm_join_workers]

#####
# (Facultatif) Copier admin.conf depuis le master si nécessaire
#####
- name: Copier admin.conf sur les workers depuis localhost (optionnel)
  hosts: workers
  become: true
  tasks:
    - name: Rendre admin.conf lisible en local
      file:
        path: /etc/kubernetes/admin.conf
        mode: '0644'
        owner: root
        group: root
        delegate_to: localhost
        run_once: true

    - name: Copier admin.conf sur les workers
      copy:
        src: /etc/kubernetes/admin.conf
        dest: /etc/kubernetes/admin.conf
        owner: root
        group: root
        mode: '0644'

```

```
#####
# 5. Configuration de .kube/config sur les workers (facultatif)
#####
- name: Configurer kubectl sur les workers (facultatif)
  hosts: workers
  become: true
  tasks:
    - name: Créer le répertoire .kube et copier le fichier admin.conf (Master
      uniquement)
      ansible.builtin.shell: |
        mkdir -p /home/val_worker1/.kube
        sudo cp -i /etc/kubernetes/admin.conf /home/val_worker1/.kube/config
        sudo chown val_worker1:val_worker1 /home/val_worker1/.kube/config
      when: inventory_hostname == '192.168.142.144'

    - name: Créer le répertoire .kube et copier le fichier admin.conf (Master
      uniquement)
      ansible.builtin.shell: |
        mkdir -p /home/val_worker2/.kube
        sudo cp -i /etc/kubernetes/admin.conf /home/val_worker2/.kube/config
        sudo chown val_worker2:val_worker2 /home/val_worker2/.kube/config
      when: inventory_hostname == '192.168.142.143'
```

#### Création du fichier install-k8s-manifests.yml (inclus la phase 6 pour la supervision):

```
---
- name: Déploiement des outils de monitoring et des applications Kubernetes (sans
  collections externes)
  hosts: masters

  tasks:

#####
# Créer une ConfigMap Prometheus à partir d'un fichier de configuration local
#####
    - name: Créer la ConfigMap Prometheus
      ansible.builtin.command:
        kubectl create configmap prometheus-config --from-
file=/home/val_master/Documents/Projet/Project-docs/Kubernetes/Config-
map/prometheus.yml

#####
# Déployer les manifests Kubernetes de l'application web (site)
#####
    - name: Appliquer les manifestes Kubernetes pour site
      ansible.builtin.command: kubectl apply -f
/home/val_master/Documents/Projet/Project-docs/Kubernetes/App/site

#####
```

```

# Déployer les manifests Kubernetes de Prometheus (monitoring)
#####
- name: Appliquer les manifests Kubernetes pour prometheus
  ansible.builtin.command: kubectl apply -f
/home/val_master/Documents/Projet/Project-docs/Kubernetes/App/prometheus

#####
# Déployer les manifests Kubernetes de Grafana (dashboard)
#####
- name: Appliquer les manifests Kubernetes pour grafana
  ansible.builtin.command: kubectl apply -f
/home/val_master/Documents/Projet/Project-docs/Kubernetes/App/grafana

#####
# Déployer les manifests Kubernetes de kube-metrics (exporter de métriques)
#####
- name: Appliquer les manifests Kubernetes pour kube-metrics
  ansible.builtin.command: kubectl apply -k
/home/val_master/Documents/Projet/Project-docs/Kubernetes/App/kube-metrics

- name: Configurer kubectl sur les workers (facultatif)
  hosts: workers
  become: true

  tasks:

#####
# Démarrer le conteneur cAdvisor pour surveiller les performances du nœud
#####
- name: Démarrer le conteneur cAdvisor
  ansible.builtin.shell: |
    docker run -d \
      --name=cadvisor \
      --restart=unless-stopped \
      --volume=/:/rootfs:ro \
      --volume=/var/run:/var/run:ro \
      --volume=/sys:/sys:ro \
      --volume=/var/lib/docker/:/var/lib/docker:ro \
      -p 8080:8080 \
      gcr.io/cadvisor/cadvisor
    # Ajout d'une vérification pour l'idempotence (pas strictement nécessaire pour
'docker run -d' car il ne relancera pas si le conteneur existe déjà)
  args:
    creates: /var/run/docker/cadvisor.pid

```

## ◆ Étapes 2 : Exécution des playbooks pour provisionner l'ensemble des machines

▶ Exécution des playbooks sur le parc :

```
ansible-playbook Projet/Project-docs/Playbooks/All.yml -i Projet/Ansible/hosts.ini -K
```

## ◆ Étapes 3 : Validation du bon fonctionnement

✓ Validation de l'installation automatisée :

```
PLAY RECAP *****
192.168.142.137      : ok=37  changed=18  unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
192.168.142.143      : ok=34  changed=7   unreachable=0    failed=0    skipped=6    rescued=0    ignored=0
192.168.142.144      : ok=35  changed=8   unreachable=0    failed=0    skipped=6    rescued=0    ignored=0
```

# Phase 6 : Maintenance et Supervision

## ◆ Étapes 1 : Mise en place de la supervision des conteneurs et du cluster Kubernetes

Déploiement de métriques des conteneurs avec `prometheus` \*\* :

### 📄 Création du config-map `prometheus.yml`

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'cadvisor-worker1'
    static_configs:
      - targets: ['192.168.142.144:8080']

  - job_name: 'cadvisor-worker2'
    static_configs:
      - targets: ['192.168.142.143:8080']

  - job_name: k8s-kube-state-metrics-cluster
    honor_timestamps: true
    metrics_path: /metrics
    scheme: http
    static_configs:
      - targets: ['192.168.142.137:30000']
    metric_relabel_configs:
      - target_label: cluster
        replacement: YourDefinedK8scluster
```

### 📄 Création du deployment `prometheus-deployment.yaml`

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
```



```

metadata:
  labels:
    app: prometheus
spec:
  containers:
  - name: prometheus
    image: prom/prometheus
    args:
      - "--config.file=/etc/prometheus/prometheus.yml"
    ports:
      - containerPort: 9090
    volumeMounts:
      - name: config-volume
        mountPath: /etc/prometheus/
  volumes:
  - name: config-volume
    configMap:
      name: prometheus-config

```

## Création du service prometheus-service.yaml

```

apiVersion: v1
kind: Service
metadata:
  name: prometheus-service
spec:
  selector:
    app: prometheus
  ports:
    - protocol: TCP
      port: 9090
      targetPort: 9090
      nodePort: 30200
  type: NodePort

```

Mise en place de la supervision avec `grafana` \*\* :

## Création du deployment grafana-deployment.yaml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grafana

```

```

template:
  metadata:
    labels:
      app: grafana
  spec:
    containers:
      - name: grafana
        image: grafana/grafana
        ports:
          - containerPort: 3000

```

## Création du service grafana-service.yaml

```

apiVersion: v1
kind: Service
metadata:
  name: grafana-service
spec:
  selector:
    app: grafana
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
      nodePort: 30500
  type: NodePort

```

## ◆ Étapes 2 : Déploiement de métriques et logs pour surveiller l'infrastructure

### Déploiement de métriques des conteneurs avec cAdvisor :

```

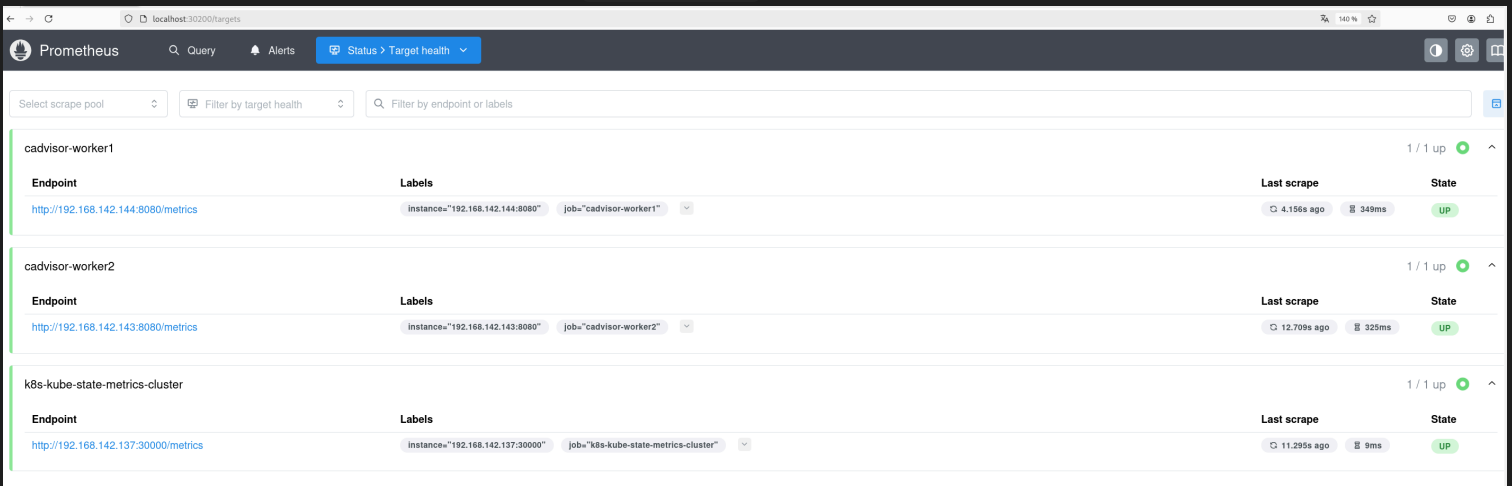
docker run -d \
  --name=cadvisor \
  --restart=unless-stopped \
  --volume=/:/rootfs:ro \
  --volume=/var/run:/var/run:ro \
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  -p 8080:8080 \
  gcr.io/cadvisor/cadvisor

```

### Déploiement de métriques du cluster Kubernetes avec le service kube-state-metric :

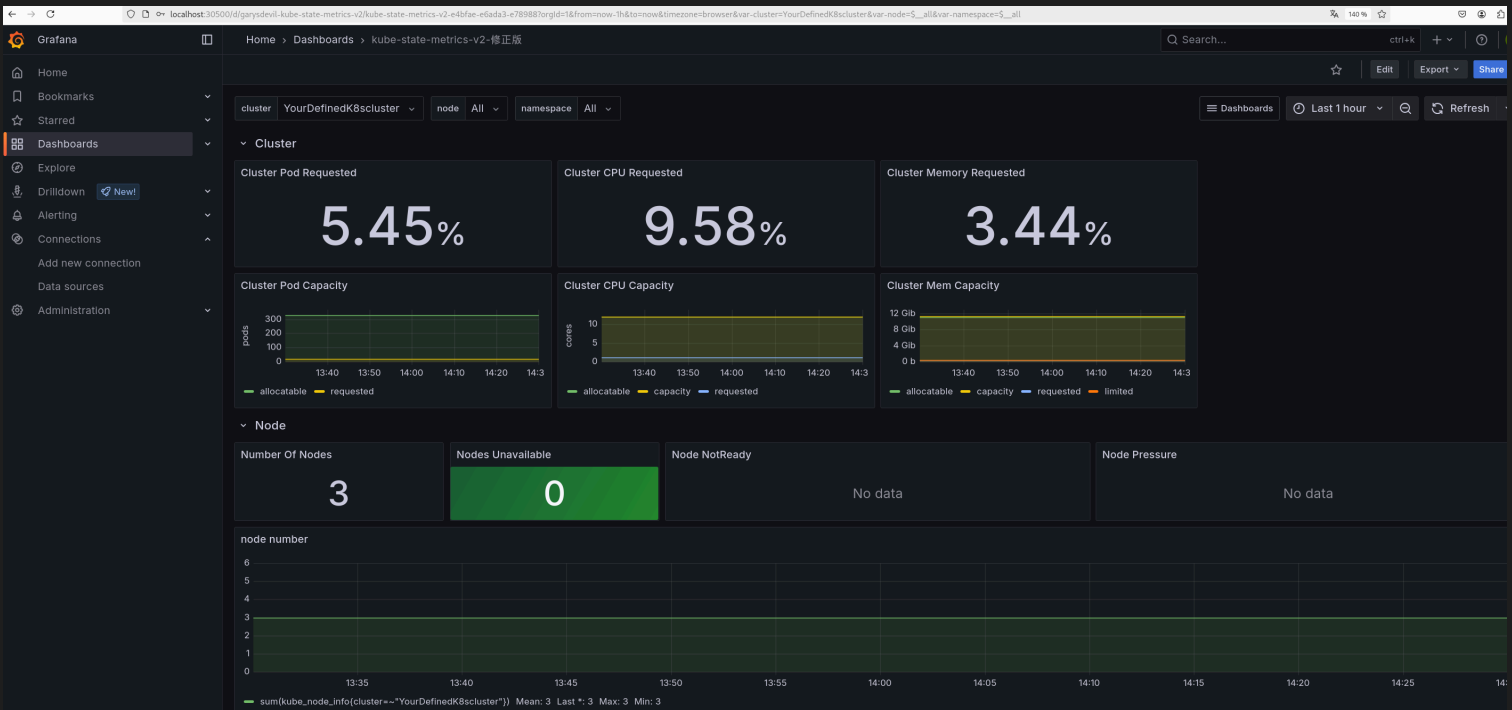
 <https://github.com/kubernetes/kube-state-metrics/tree/main/examples/standard>

## ✓ Vérification de la configuration de prometheus :

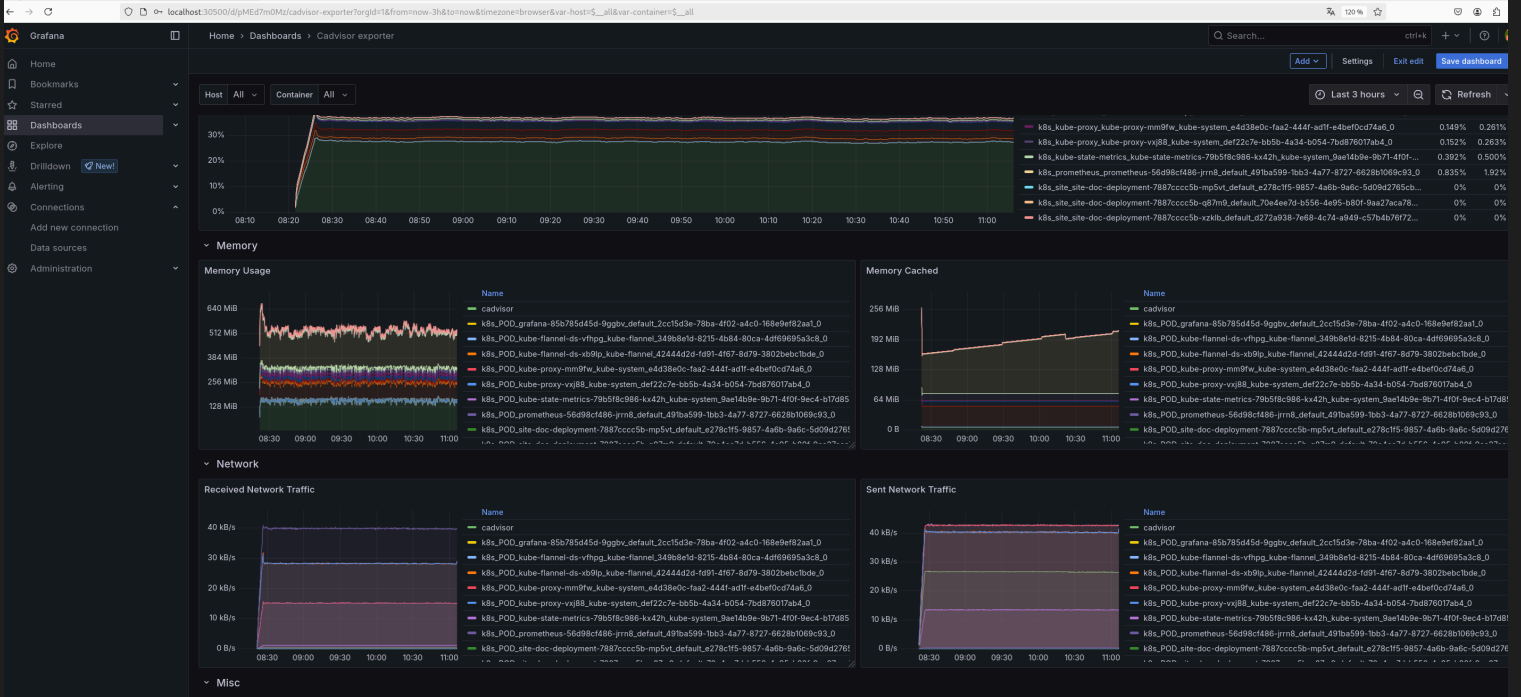


## ✓ Vérification de la supervision avec grafana :

ID template utilisé pour la supervision du cluster Kubernetes : 13332



ID template utilisé pour la supervision les conteneurs : 14282



## ◆ Étapes 3 : Automatisation de la gestion des mises à jour avec Ansible

### 📁 Automatisation des MAJ avec Ansible :

### 📄 Création du playbook mco.yaml

```
---
- name: MCO
  hosts: all
  become: true

  tasks:
    - name: Mettre à jour les paquets et nettoyer le cache (Debian/Ubuntu)
      ansible.builtin.apt:
        upgrade: dist
        update_cache: yes
        autoclean: yes
        autoremove: yes
      register: apt_update_result

    - name: Redémarrer si le noyau a été mis à jour et si nécessaire (Debian/Ubuntu)
      ansible.builtin.reboot:
        reboot_timeout: 600
      when:
        - apt_update_result.reboot_required is defined and
          apt_update_result.reboot_required

    - name: Nettoyer les fichiers temporaires dans /tmp (tous les OS)
```



# Axes d'améliorations

---

- Gestion de log avec promtail et loki (pas réussi à faire fonctionner loki)
- Segmentation réseau des vm pour avoir un réseau dédié pour l'*administration*.
- Configuration des namespaces pour éviter de mettre dans le default.
- Affiner les différents éléments configurés tout au long du projet.

🚧 🚧 🚧 Ajout d'une page HTML affichant directement le rapport directement (fait à la toute fin)

Non sécurisémon-site-local.test

ChNAMAdministratifCyberAnglaisIAFreeThaïlandeMalaisieGoogle KeepGoogle DriveSolitaire Femme Cie...

Projet SMB 111

# Rapport technique V1

Ce projet consiste à déployer automatiquement un site web statique contenant la documentation complète du projet, avec Docker, Ansible et Kubernetes sur 3 machines. L'objectif est de fournir un espace accessible via un navigateur pour consulter les spécifications, les guides et les instructions du projet.

Explorez ce document directement sur cette page ou téléchargez-le.

Lire le PDF

Télécharger le PDF

## Découvrez plus sur notre projet

Visitez notre page dédiée pour en savoir plus sur les détails techniques et les aspects avancés du projet.

Accéder à la Page Détails

## Phases de Déploiement du Projet

Phase 1 : Préparation des Machines

Installation de l'OS (Debian 12), configuration

Phase 2: Installation & Configuration de Docker

Phase 3: Déploiement de Kubernetes

Préparation de l'environnement (swapoff).

Non sécurisémon-site-local.test/page.html

ChNAMAdministratifCyberAnglaisIAFreeThaïlandeMalaisieGoogle KeepGoogle DriveSolitaire Femme Cie...

# Déploiement Automatisé d'un Site de Documentation avec MCO/MCS

## Phase 1 : Préparation des Machines

Étape 1 : Installation de l'OS

Choix de l'OS (Ubuntu/Debian recommandé):

Debian 12\* (recommandé)

Installation de base avec SSH activé:

Installation par défaut avec la suppression de la partition swap pour Kubernetes

SSH activé:

ssh.service - OpenSSH Secure Shell server.  
loaded: loaded (/lib/systemd/system/ssh.service; enabled)  
Active: active (running) since Tue 2025-04-15 09:29:13 CEST; 5min ago

Mise à jour des paquets:

apt-get update  
apt-get upgrade

Étape 2 : Configuration initiale des machines

Configuration du PATH (commandes manquantes):

export PATH=\$PATH:/usr/sbin #Ajout de ce dossier au path pour avoir des commandes nécessaire à la suite du projet

Ajout des utilisateurs:

adduser val\_master # VM Master  
adduser val\_worker1 # VM Worker Node 1  
adduser val\_worker2 # VM Worker Node 2

Ajout des utilisateurs au groupe sudo:

usermod -sG sudo val\_master # VM Master  
usermod -sG sudo val\_worker1 # VM Worker Node 1  
usermod -sG sudo val\_worker2 # VM Worker Node 2

Configuration réseau & hostnames:

hostnamectl set-hostname Master #VM Master  
hostnamectl set-hostname Worker1 #VM Worker Node 1  
hostnamectl set-hostname Worker2 #VM Worker Node 2

Génération de clé SSH + copie:

ssh-keygen -t rsa -b 4096 -C "root@master" -f /root/.ssh/id\_rsa  
ssh-copy-id val\_master  
ssh-copy-id val\_worker1  
ssh-copy-id val\_worker2

# Annexe 1 : ARBORESCENCE

```
.
└─ Projct
    └─ Ansible
        └─ hosts.ini
    └─ Project-docs
        └─ Config
            └─ nginx
                └─ certs
                    └─ server.crt
                    └─ server.key
                └─ conf
                    └─ default.conf
        └─ docker-compose.yml
        └─ Dockerfile
        └─ Documentation
            └─ Excalidraw
                └─ Drawing 2025-06-11 14.26.16.excalidraw.md
            └─ Pasted image 20250604214223.png
            └─ Pasted image 20250604214320.png
            └─ Pasted image 20250604214528.png
            └─ Pasted Image 20250611143455_879.png
            └─ Pasted Image 20250611144601_143.png
            └─ Pasted image 20250618151743.png
            └─ Pasted image 20250618152106.png
            └─ Pasted image 20250618155137.png
            └─ Pasted image 20250618155144.png
            └─ Pasted image 20250619110512.png
            └─ Pasted image 20250619110538.png
            └─ Pasted image 20250619110645.png
            └─ Pasted image 20250619112740.png
            └─ Rapport.md
            └─ Rapport.pdf
        └─ Kubernetes
            └─ App
                └─ grafana
                    └─ grafana-deployment.yaml
                    └─ grafana-service.yaml
                └─ kube-metrics
                    └─ cluster-role-binding.yaml
                    └─ cluster-role.yaml
                    └─ deployment.yaml
                    └─ kustomization.yaml
                    └─ metric-service.yaml
                    └─ service-account.yaml
                └─ prometheus
                    └─ prometheus-deployment.yaml
                    └─ prometheus-service.yaml
            └─ site
                └─ site-deployment.yaml
```



```
├── site-service.yaml
├── Config-map
│   └── prometheus.yml
├── Playbooks
│   ├── All.yml
│   ├── install-docker.yml
│   ├── install-k8s-manifests.yml
│   ├── install-k8s.yml
│   ├── mco.yaml
│   ├── old
│   │   └── All.yml
│   └── setup-hosts.yml
├── Website_content
│   ├── index.html
│   ├── page
│   │   ├── Dark PDF export.css
│   │   ├── page.html
│   │   ├── Pasted image 20250604214223.png
│   │   ├── Pasted image 20250604214320.png
│   │   ├── Pasted image 20250604214528.png
│   │   ├── Pasted Image 20250611143455_879.png
│   │   ├── Pasted Image 20250611144601_143.png
│   │   ├── Pasted image 20250618151743.png
│   │   ├── Pasted image 20250618152106.png
│   │   ├── Pasted image 20250618155137.png
│   │   ├── Pasted image 20250618155144.png
│   │   ├── Pasted image 20250619110512.png
│   │   ├── Pasted image 20250619110538.png
│   │   ├── Pasted image 20250619110645.png
│   │   ├── Pasted image 20250619112740.png
│   │   └── supercharged-links-gen.css
│   ├── rapport.pdf
│   ├── script.js
│   └── style.css
└── SMB111- Projet - 2024 - 2025.docx.pdf
```