

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 - OpenBSD Secure Shellserver
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) :

```
kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-
flannel.yml
```

◆ Étape 4 : Vérification du bon fonctionnement du cluster

📋 Vérification :

```
kubectl get nodes
kubectl 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` :

```
version: '3.8'

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
```

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

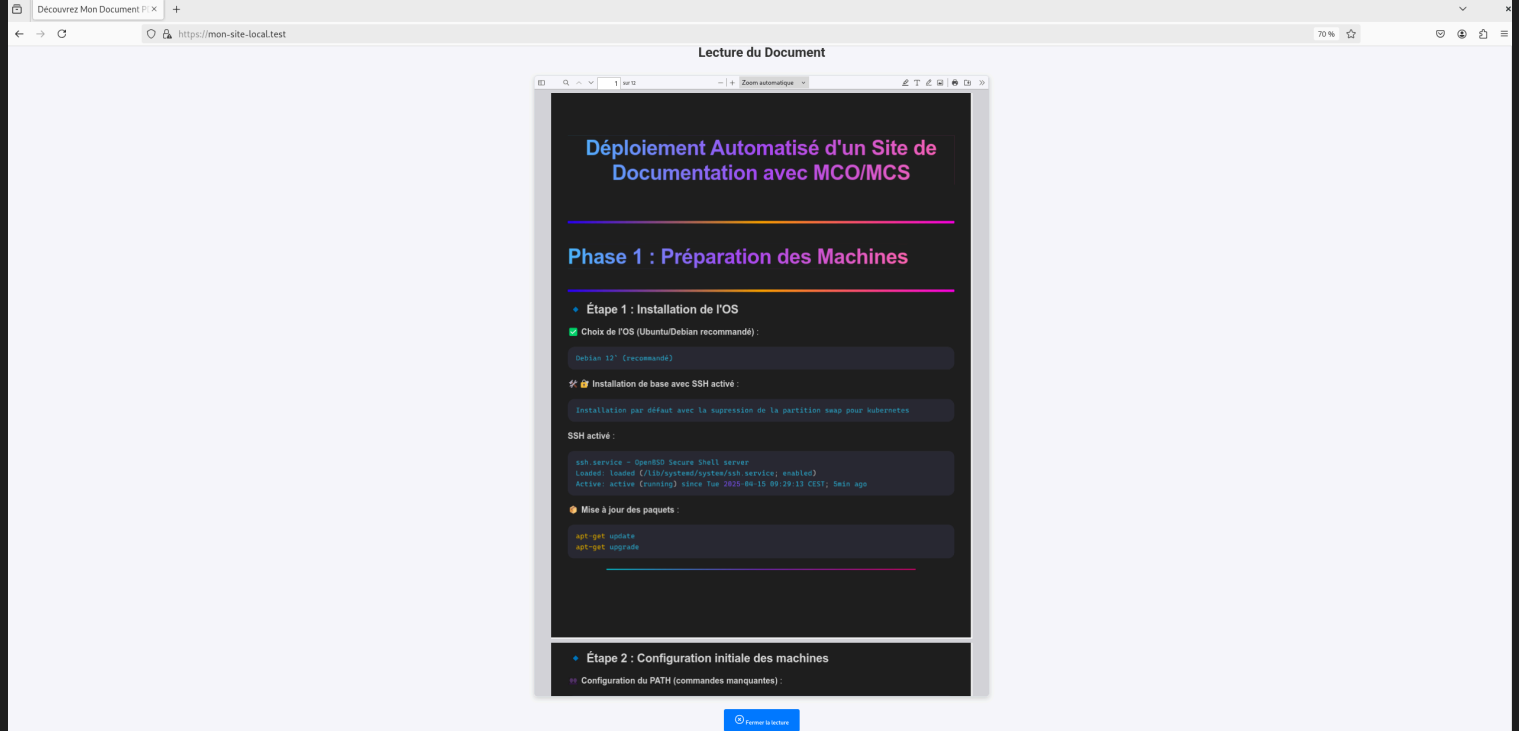
<https://mon-site-local.test:443>

The screenshot shows a web browser window with the address bar displaying `https://mon-site-local.test`. The page title is "Projet SMB 111". The main content area features a section titled "Rapport technique V1" with a brief description of the project's goal: to deploy a static website containing project documentation (Docker, Ansible, Kubernetes) on 3 machines. Below this, there are two buttons: "Lire le PDF" and "Télécharger le PDF".

Below the report section, there is a section titled "Phases de Déploiement du Projet" which lists six phases in a grid:

- Phase 1: Préparation des Machines** (Icon: Document with checkmark)
Installation de l'OS (Debian 12), configuration initiale des machines avec ajout d'utilisateurs et de logiciels, et mise en place de la connectivité SSH.
- Phase 2: Installation & Configuration de Docker** (Icon: Docker logo)
Installation de Docker sur les machines, ajout des utilisateurs au groupe Docker, et vérification du bon fonctionnement de Docker.
- Phase 3: Déploiement de Kubernetes** (Icon: Kubernetes logo)
Préparation de l'environnement (ovsppf), installation via kubeadm, installation du cluster, alignement de l'horod pour le temps des Pods, et configuration de kubelet.
- Phase 4: Déploiement du Site Web** (Icon: Web icon)
Conception et écriture du Dockerfile pour l'application web statique, création du document compose.yml, élaboration des manifests Kubernetes et déploiement final sur le cluster.
- Phase 5: Automatisation avec Ansible** (Icon: Ansible logo)
Création des playbooks Ansible pour le provisionnement des machines et l'intégration de la supervision du cluster et des conteneurs.
- Phase 6: Supervision & Monitoring** (Icon: Monitor icon)
Mise en place de la supervision complète du cluster Kubernetes et des conteneurs, incluant l'intégration des métriques et des logs.

At the bottom of the page, there is a footer: "© 2025 Valentin LEBLANC. Tous droits réservés."



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

Manifests Kubernetes (Deployment , Service , etc.) :

◆ Étapes 5 : Déploiement final sur Kubernetes

🚀 Déploiement final sur Kubernetes :

Phase 5 : Automatisation avec Ansible

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

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

📄 Création du fichier d'inventories.ini :

```
[all]
192.168.142.137 # VM Master
192.168.142.144 # VM Worker Node 1
192.168.142.143 # VM Worker Node 2

[masters]
192.168.142.137

[workers]
192.168.142.144
192.168.142.143
```

Création du fichier config.yaml :

A voir en Annexe (très long)

Récupération des playbooks, config docker, kubernetes etc via github :

◆ 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 :

Phase 6 : Maintenance et Supervision

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

🔧 Mise en place de la supervision du cluster & conteneurs :

```
kubectl create configmap prometheus-config --from-file=prometheus.yml
```

Install Cadvisor sur les workers

```
kubectl apply -f prometheus,gggraphana
```

5. Ajouter Prometheus comme source de données

1. Dans Grafana, clique sur **"Gear (⚙️) > Data Sources"**

2. Clique sur **"Add data source"**

3. Choisis **"Prometheus"**

4. Dans **URL**, mets :

pgsql

CopierModifier

``http://prometheus-service.default.svc.cluster.local:9090``

5. Clique sur **"Save & Test"** → ça doit être **"Data source is working"**

📺 6. Importer un Dashboard cAdvisor (facultatif mais recommandé)

1. Dans Grafana, clique sur le **"+" > Import**

2. Entre cet ID de dashboard public :

CopierModifier

``14282``

→ c'est un dashboard cAdvisor Prometheus prêt à l'emploi

3. Clique sur **"Load"**

4. Choisis la **Prometheus data source** créée, puis clique **"Import"**

🎉 Tu verras maintenant un dashboard avec CPU, mémoire, conteneurs, etc.

```
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
```

◆ Étapes 2 : Création des playbooks Ansible pour automatiser l'installation et la configuration

📁 Intégration des métriques & des logs :

◆ Étapes 3 : Création des playbooks Ansible pour automatiser l'installation et la configuration

📁 Automatisation des MAJ avec Ansible :

Annexe 1 : ARBORESCENCE

```
/
├── etc/
│   └── Ansible/
│       └── hosts.ini                # Fichier d'inventaire Ansible
├── var/
│   └── log/                        # Répertoire des logs (ex. collecte centralisée)
├── Project-docs/                  # Répertoire principal du projet DocHub
│   ├── Playbooks/                 # Playbooks Ansible (installation, config, gestion)
│   │   ├── setup_machines.yml     # Préparation initiale (OS, utilisateurs, SSH)
│   │   ├── install_docker.yml     # Installation et configuration de Docker
│   │   ├── deploy_kubernetes.yml  # Déploiement de Kubernetes
│   │   ├── deploy_website.yml     # Déploiement complet du site web
│   │   ├── update_management.yml   # MCO : automatisation des mises à jour
│   │   └── ...                     # Autres playbooks (supervision, durcissement, etc.)
│   ├── kubernetes_manifests/      # Manifests Kubernetes pour le site web
│   │   ├── deployment.yaml
│   │   ├── service.yaml
│   │   ├── ingress.yaml
│   │   └── ...
│   ├── Docker/                    # Dockerfile & docker-compose
│   │   ├── Dockerfile             # Pour l'app web statique
│   │   └── docker-compose.yml     # Orchestration locale
│   ├── Website_content/           # Fichiers du site web statique
│   │   ├── index.html
│   │   ├── style.css
│   │   ├── script.js
│   │   └── rapport.pdf            # Dossier pour le rapport en PDF
│   ├── Documentation/             # Docs Markdown pour le projet
│   │   └── rapport.md
│   ├── Scripts/                   # Scripts utilitaires (bash, python, etc.)
│   └── Config/                    # Fichiers de configuration divers
│       ├── nginx/                 # Config spécifique à Nginx
│       │   ├── certs
│       │   └── server.crt
```

```
|
|
|   |   |   | server.key
|   |   |   | conf
|   |   |   | default.conf
|   |   |   |
|   |   |   | ansible
|   |   |   |
|   |   |   | ansible.conf
|   |   |   |
|   |   |   | # Autres configs (monitoring, logs, etc.)
|   |   |   |
|
| home/
|   |
|   | val_worker/
|   |   |
|   |   | .ssh/
|   |   |   |
|   |   |   | id_rsa
|   |   |   |
|   |   |   | id_rsa.pub
|   |   |   |
|   |   |   | # Clé privée SSH (accès sans mot de passe)
|   |   |   |
|   |   |   | # Clé publique SSH
```