# 📘 Annexe — Déploiement complet Docker & K3s

Infrastructure applicative conteneurisée (FOOT_Store)

---

## 🎯 Objectif de cette annexe

Cette annexe a pour objectif de **documenter intégralement** le déploiement de l'infrastructure applicative du projet *Mini-Hébergeur Associatif*.
Elle est rédigée comme un **tutoriel technique reproductible**, permettant à toute personne extérieure au projet de :

- comprendre l'architecture retenue,
- exécuter les commandes **dans le bon ordre**,
- déployer l'application **sans ambiguïté**.

Toutes les commandes sont présentées de manière **linéaire** et doivent être exécutées **dans l'ordre indiqué**.

---

## 🧱 Architecture générale

L'infrastructure applicative repose sur un **cluster K3s** (Kubernetes léger), déployé sur des **machines virtuelles Proxmox** et utilisant **Docker** comme moteur de conteneurisation.

### 🔷 Répartition des rôles (VLAN 12 – Réseau VM)

| Rôle | Adresse IP |
|---|---|
| Manager K3s | `10.0.12.11` |
| Worker K3s 1 | `10.0.12.12` |
| Worker K3s 2 | `10.0.12.13` |
| Passerelle (pfSense) | `10.0.12.14` |

Le **manager** assure les rôles suivants :

- installation et gestion de Docker,
- hébergement d'un **registry Docker local**,
- pilotage du cluster K3s,
- déploiement de l'application FOOT_Store.

---

## 🐋 1. Installation de Docker (Manager uniquement)

Docker est utilisé pour construire les images applicatives et héberger un registry local.

### 1.1 Nettoyage préalable

On supprime toute version existante afin d'éviter les conflits.

```
sudo apt-get remove -y docker docker-engine docker.io containerd runc ||
true
sudo apt-get update
```

### 1.2 Installation des dépendances système

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

### 1.3 Ajout du dépôt Docker officiel

```
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

### 1.4 Installation de Docker Engine

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

### 1.5 Activation et démarrage du service

```
sudo systemctl enable docker
sudo systemctl start docker
sudo systemctl status docker
```

### 1.6 Autoriser Docker sans sudo (recommandé)

```
sudo usermod -aG docker $USER
newgrp docker
```

### 1.7 Test de fonctionnement

```
docker run --rm hello-world
```

---

## 📦 2. Mise en place du registry Docker local

Le registry local permet au cluster K3s de récupérer les images sans dépendre d'Internet.

### 2.1 Lancement du registry

```
docker run -d -p 5000:5000 --restart=always --name registry registry:2
```

### 2.2 Autorisation du registry HTTP

Docker bloque par défaut les registries non HTTPS.

```
sudo mkdir -p /etc/docker
sudo nano /etc/docker/daemon.json
```

Contenu du fichier :

```
{
  "insecure-registries": ["10.0.12.11:5000"]
}
```

Redémarrage de Docker :

```
sudo systemctl restart docker
```

Test :

```
curl http://10.0.12.11:5000/v2/_catalog
```

---

## ☸️ 3. Installation de K3s — Manager

K3s est une version allégée de Kubernetes, adaptée à des environnements à ressources limitées.

```
curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="server --node-
ip=10.0.12.11 --advertise-address=10.0.12.11" sh -
```

Vérification :

```
sudo kubectl get nodes
```

Récupération du token du cluster :

```
sudo cat /var/lib/rancher/k3s/server/node-token
```

---

## 🧩 4. Installation de K3s — Workers

Worker 1 (`10.0.12.12`)

```
curl -sfL https://get.k3s.io | K3S_URL="https://10.0.12.11:6443"
K3S_TOKEN="TOKEN" INSTALL_K3S_EXEC="agent --node-ip=10.0.12.12" sh -
```

Worker 2 (`10.0.12.13`)

```
curl -sfL https://get.k3s.io | K3S_URL="https://10.0.12.11:6443"
K3S_TOKEN="TOKEN" INSTALL_K3S_EXEC="agent --node-ip=10.0.12.13" sh -
```

Vérification depuis le manager :

```
sudo kubectl get nodes
```

---

## 🔗 5. Déclaration du registry dans K3s (tous les nœuds)

```
sudo mkdir -p /etc/rancher/k3s
sudo nano /etc/rancher/k3s/registries.yaml
```

Contenu :

```
mirrors:
  "10.0.12.11:5000":
    endpoint:
      - "http://10.0.12.11:5000"
```

Redémarrage :

```
sudo systemctl restart k3s
sudo systemctl restart k3s-agent
```

---

## 🚀 6. Déploiement de l'application FOOT_Store

### 6.1 Build et push de l'image

```
cd ~/foot_store
docker build -t 10.0.12.11:5000/footstore-app:latest .
docker push 10.0.12.11:5000/footstore-app:latest
```

### 6.2 Initialisation de la base MySQL

```
sudo kubectl create configmap footstore-init-sql --from-
file=init.sql=aitbaha4u_footstore.sql
```

### 6.3 Déploiement MySQL

```
sudo kubectl apply -f mysql-k8s.yaml
sudo kubectl get pods
```

### 6.4 Déploiement de l'application

```
sudo kubectl apply -f k3s-footstore.yaml
sudo kubectl get pods
```

---

## 🌐 7. Reverse proxy & accès applicatif (Traefik)

Traefik est intégré nativement à K3s.

```
sudo kubectl get pods -n kube-system | grep traefik
sudo kubectl apply -f footstore-ingress.yaml
```

Accès final :

```
http://10.0.12.11
```

---

## ✅ Conclusion technique

L'infrastructure applicative est désormais :

- conteneurisée avec Docker,
- orchestrée par K3s,
- distribuée sur plusieurs nœuds,
- accessible via un reverse proxy,
- reproductible et documentée.

Cette architecture répond aux objectifs de **résilience**, **maintenabilité** et **simplicité opérationnelle** du projet.