S5.B.01 Phase 4
Déploiement de services

Maxence Lagourgue

7 janvier 2026

Table des matières

I Introduction

I.A Outils

Dans cette partie, les outils utilisés seront :

- Rancher pour la gestion des clusters
- RKE2 pour la mise en œuvre Kubernetes des nœuds de travail
- k3s pour le cluster Rancher
- kubectl pour la gestion des ressources
- Helm pour la gestion des applications

Plus tard, si nous avons le temps, nous utiliserons Ansible pour automiser la chaîne de production Rancher.

I.B Machines

Les machines utilisées au cours de ce projet seront :

- applicatif ⇒ k3s cluster + Rancher Server
- K8SA2 (k8s1) ⇒ RKE2 cluster + Master, Etcd, Worker Nodes
- K8SB2 (k8s2) ⇒ RKE2 cluster + Worker nodes + Backup
- K8SC2 (k8s3) ⇒ ???

I.C Configuration générale

Dans toutes les VMs impliquées dans le cluster Kubernetes, la configuration suivante sera définie.

Listing I.1 – /etc/hosts

```
10.0.1.3        rancher.rancher
10.0.1.4        master.rancher
10.0.1.5        worker.rancher
```

Pour monitorer les ressources des VMs, nous ne pouvons pas nous servir des indications données par Proxmox VM car le Guest Agent est désactivé. Nous aurons donc de mauvaises indications pour la RAM par exemple. Je conseille donc :

```
wget https://github.com/fastfetch-cli/fastfetch/releases/download/2.56.1/fastfetch-linux-
    amd64.deb && dpkg -i fastfetch-linux-amd64.deb
```

II Installation de Rancher dans un cluster k3s

Pour utiliser Rancher, plusieurs méthodes d'installation s'offrent à nous. L'une avec docker, l'autre en tant que noeud Kubernetes. Les autres installations reposent sur l'utilisation d'un Cloud Provider ainsi que Terraform donc inutile dans notre cas.
Exemple de tutorial : Tutorial Rancher 2025
Faire le gitlab en tant qu'application kubernetes/rancher.

II.A Installation de k3s

```
curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION="v1.24.10+k3s1" sh -s - server
--cluster-init --bind-address 10.0.1.3
```

II.A.1 Accès distant au cluster (Optionnel)

<IP_OF_LINUX_MACHINE> est l'IP de la machine distante sur laquelle se trouve le cluster.

```
scp root@<IP_OF_LINUX_MACHINE>:/etc/rancher/k3s/k3s.yaml ~/.kube/config
```

```
nano ~/.kube/config
```

II.A.2 Installation d'helm

```
sudo apt-get install curl gpg apt-transport-https --yes

curl -fsSL https://packages.buildkite.com/helm-linux/helm-debian/gpgkey | gpg --dearmor |
sudo tee /usr/share/keyrings/helm.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/helm.gpg]
https://packages.buildkite.com/helm-linux/helm-debian/any/ any main" | sudo tee
/etc/apt/sources.list.d/helm-stable-debian.list

sudo apt-get update
sudo apt-get install helm
```

II.A.3 Installation de kubectl

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

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.34/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

```
sudo chmod 644 /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]
https://pkgs.k8s.io/core:/stable:/v1.34/deb/ /' | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo chmod 644 /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update
sudo apt-get install -y kubectl
```

II.A.4 Création d'un Déploiement Rancher

<Hostname> correspond au nom de domaine utilisé pour contacter le pod rancher.

```
export KUBECONFIG=/etc/rancher/k3s/k3s.yaml



kubectl create namespace cattle-system
kubectl config set-context --current --namespace=cattle-system

kubectl apply -f
https://github.com/cert-manager/cert-manager/releases/download/v1.19.2/cert-manager.crds.yaml

helm repo add rancher-latest https://releases.rancher.com/server-charts/latest

helm repo add jetstack https://charts.jetstack.io

helm repo update

helm install cert-manager jetstack/cert-manager \
  --namespace cert-manager \
  --create-namespace

helm install rancher rancher-latest/rancher \
  --namespace cattle-system \
  --set hostname=rancher.rancher \
  --set replicas=1 \
  --set bootstrapPassword=testpassword
```

Il faut maintenant attendre car l'installation nécessite quelques minutes. On peut vérifier avec `kubectl␣get␣pods␣-n␣cattle-system`
Une fois fait, on se connecte à la page et on récupère le mot de passe :

```
kubectl get secret --namespace cattle-system bootstrap-secret -o
go-template='{{.data.bootstrapPassword|base64decode}}{{"\n"}}'
```

On définit un nouveau mot de passe qui est `qJHiA@wwaagi46U`.

II.A.5 Pour arrêter/pauser Rancher

```
kubectl scale --replicas=0 deployment/rancher -n cattle-system
```

Cela permet d'arrêter temporairement le pod Rancher.

II.B Réinitialisation du cluster

Pour revenir à l'état 0 du cluster, il est possible de :

```
rm -rf /var/lib/rancher/k3s/server/db/etcd
/usr/local/bin/k3s-killall.sh
systemctl restart k3s.service
```

III Création d'un cluster

III.A Enrôlement de machines

Pour cela il faut aller dans la section **Clusters** ⇒ **<Cluster>** ⇒ **Registration**

Listing III.1 – Machine master

```
export CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
export CONTAINERD_ADDRESS=unix:///run/k3s/containerd/containerd.sock
export PATH=$PATH:/var/lib/rancher/rke2/bin
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml

curl --insecure -fL https://rancher.rancher/system-agent-install.sh | sudo  sh -s - --
    server https://rancher.rancher --label 'cattle.io/os=linux' --token
    tvg69x5vkm9szzlzsj6qqkx7wggzrgvt2grc755nth29h2ncjbgthz --ca-checksum
    fdc9c50ea58442994213e96883b6a5ca39227fc7d4116e60fa1026c123f56583 --etcd --controlplane
    --worker --address 10.0.1.4 --internal-address 10.0.1.4
```

Listing III.2 – Machine worker

```
export CRI_CONFIG_FILE=/var/lib/rancher/rke2/agent/etc/crictl.yaml
export CONTAINERD_ADDRESS=unix:///run/k3s/containerd/containerd.sock
export PATH=$PATH:/var/lib/rancher/rke2/bin
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml

curl --insecure -fL https://rancher.rancher/system-agent-install.sh | sudo  sh -s - --
    server https://rancher.rancher --label 'cattle.io/os=linux' --token
    tvg69x5vkm9szzlzsj6qqkx7wggzrgvt2grc755nth29h2ncjbgthz --ca-checksum
    fdc9c50ea58442994213e96883b6a5ca39227fc7d4116e60fa1026c123f56583 --worker --address
    10.0.1.5 --internal-address 10.0.1.5
```

Si DNS issue :

```
export KUBE_EDITOR="nano"
kubectl edit configmap rke2-coredns-rke2-coredns -n kube-system

hosts {
    10.0.1.3 rancher.rancher
    fallthrough
}

kubectl rollout restart deployment rke2-coredns-rke2-coredns -n kube-system
```

IV Creation d'un Workload

Tout d'abord, avant de déployer quoi que ce soit il faut convertir le docker-compose.yml en fichier de déploiement Kubernetes, un manifest.

Listing IV.1 – Docker-compose

```
services:
  nginx:
    image: nginx:alpine
    container_name: nailloux-club-nginx
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/conf.d/default.conf:ro
      - ./nginx/certs:/etc/nginx/certs:ro
    networks:
      - nailloux-network
    extra_hosts:
      - "www.nailloux.lan:10.0.1.3"
    depends_on:
      - app

  app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: nailloux-club-app
    restart: unless-stopped
    expose:
      - "80"
    environment:
      - DB_HOST=db
      - DB_NAME=nailloux
      - DB_USER=root
      - DB_PASSWORD=rootpassword
    volumes:
      - ./:/var/www/html
      - ./upload:/var/www/html/upload
    networks:
      - nailloux-network
    depends_on:
      db:
```

```
        condition: service_healthy

  db:
    image: mysql:8.0
    container_name: nailloux-club-db
    restart: unless-stopped
    ports:
      - "33060:3306" # Changed port to avoid conflicts
    environment:
      MYSQL_DATABASE: nailloux
      MYSQL_ROOT_PASSWORD: rootpassword
    command: --default-authentication-plugin=mysql_native_password --character-set-server=
        utf8mb4 --collation-server=utf8mb4_unicode_ci
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      timeout: 5s
      retries: 10
    volumes:
      - dbdata:/var/lib/mysql
      - ./nailloux.sql:/docker-entrypoint-initdb.d/nailloux.sql
    networks:
      - nailloux-network

networks:
  nailloux-network:
    driver: bridge
    enable_ipv6: true
volumes:
  dbdata:
    driver: local
```