Instalación Limpia de Docker y Kubernetes en Ubuntu 24.04

Este documento describe los pasos para instalar Docker Engine, Portainer (dashboard web), y Kubernetes (usando kind) en Ubuntu 24.04, dejando el entorno sincronizado y funcional.

1. Preparar el sistema

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg lsb-release apt-transport-https
sudo apt remove -y docker.io docker-doc docker-compose docker-compose-v2 podman-
docker
sudo apt autoremove -y
```

2. Instalar Docker Engine

```
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
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu noble stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER
```

3. Instalar Portainer (Dashboard de Docker)

```
docker volume create portainer_data
docker run -d -p 8000:8000 -p 9443:9443 --name portainer --restart=always -v
/var/run/docker.sock:/var/run/docker.sock -v portainer_data:/data portainer/portainer-
ce:latest
```

Accede en el navegador a: https://localhost:9443 o https://<IP_PUBLICA>:9443

4. Instalar kubectl y kind (Kubernetes en Docker)

```
# Instalar kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
chmod +x kubectl
sudo mv kubectl /usr/local/bin/kubectl

# Instalar kind
KIND_VERSION="v0.30.0"
curl -Lo ./kind "https://kind.sigs.k8s.io/dl/${KIND_VERSION}/kind-$(uname)-amd64"
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind

# Crear cluster
kind create cluster --name dev-cluster

kubectl cluster-info --context kind-dev-cluster
kubectl get nodes
```

5. Instalar Kubernetes Dashboard

```
# Instalar Helm
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Instalar Dashboard
helm repo add kubernetes-dashboard https://kubernetes.github.io/dashboard/
helm repo update
helm upgrade --install kubernetes-dashboard kubernetes-dashboard/kubernetes-
dashboard --create-namespace --namespace kubernetes-dashboard

# Crear usuario admin
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
EOF

# Obtener token
kubectl -n kubernetes-dashboard create token admin-user

# Acceso local
kubectl -n kubernetes-dashboard port-forward svc/kubernetes-dashboard-kong-proxy
8443:443
Accede a: https://localhost:8443
```

6. Ejemplo rápido con Docker

```
mkdir demo-docker && cd demo-docker
echo 'from flask import Flask, jsonify
app = Flask(__name__)
@app.route("/")
def home():
    return jsonify(message="Hola desde Docker!")
if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8000)' > app.py

echo "Flask==3.0.3" > requirements.txt

echo 'FROM python:3.12-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
EXPOSE 8000
CMD ["python", "app.py"]' > Dockerfile

docker build -t demo-web:1.0 .
docker run -d --name demo-web -p 8080:8000 demo-web:1.0
```

7. Ejemplo rápido con Kubernetes

```
# Cargar imagen en kind
kind load docker-image demo-web:1.0 --name dev-cluster

# Crear manifiesto
cat <<EOF > demo-k8s.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: demo-web
  template:
    metadata:
      labels:
        app: demo-web
    spec:
      containers:
      - name: demo-web
        image: demo-web:1.0
        ports:
        - containerPort: 8000
---
apiVersion: v1
kind: Service
metadata:
  name: demo-web
spec:
  type: NodePort
  selector:
    app: demo-web
  ports:
  - port: 8000
    targetPort: 8000
    nodePort: 30800
EOF
```

```
kubectl apply -f demo-k8s.yaml
kubectl get pods
kubectl get svc

# Acceso externo
http://<IP_PUBLICA>:30800
```