

Sistemas Distribuídos — Lab K3S

Instâncias

t3.micro com Ubuntu 24.04

Crie 3 instâncias, cada uma com seu respectivo Security Group e portas expostas de acordo com o esquema a seguir:

- K3S-Server:

portas:

SSH 22

TCP 6443

TCP 10250

UDP 8472

TCP 30080

elastic ip: não

- K3S-DB:

portas:

SSH 22

TCP 5432

elastic ip: não

- K3S-Nginx:

portas:

SSH 22

HTTP 80

HTTPS 443

elastic ip: sim

IPs Privados

Anote conforme necessário

K3s-server:

K3s-DB:

K3s-Nginx:

Instalação

Preparação das instâncias e tecnologias

K3s-server

Configuração inicial do k3s-server

[Instalar k3s]

```
sudo apt update && sudo apt upgrade -y

curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="server
--write-kubeconfig-mode=644" sh -
```

[Criar usuário e configurar kubectl sem sudo]

```
mkdir -p ~/.kube
sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config
sudo chown $USER:$USER ~/.kube/config

sudo systemctl restart k3s
kubectl get nodes
```

K3s-DB

Configuração inicial do database

[Instalar o docker]

```
sudo apt update && sudo apt upgrade -y

curl -fsSL https://get.docker.com | bash
```

[Configurar docker-compose]

```
mkdir database
cd database/

nano docker-compose.yml
```

[docker-compose.yml]

```
version: '3.9'

services:
  postgres:
    image: postgres:17
    container_name: postgres_aws
    restart: always
    environment:
      POSTGRES_USER: appuser
      POSTGRES_PASSWORD: appSenha123
      POSTGRES_DB: app_db
    ports:
      - "5432:5432"
    volumes:
      - ./data:/var/lib/postgresql/data
```

A URL para conexão com o database

será: postgresql://<POSTGRES_USER>:<POSTGRES_PASSWORD>@DB_PRIV:5432/<POSTGRES_DB>

[Subir a imagem]

```
sudo docker compose up -d
sudo docker ps
```

K3s-Nginx

Configuração do nginx

[Instalação do docker]

```
sudo apt update && sudo apt upgrade -y

curl -fsSL https://get.docker.com | bash
```

[Configuração da imagem nginx]

```
mkdir ~/nginx && cd ~/nginx
mkdir conf.d
nano docker-compose.yml
```

[docker-compose.yml]

```
version: '3.9'

services:
  nginx:
    image: nginx:latest
    container_name: nginx_gateway
    restart: always
    ports:
      - "80:80"
    volumes:
      - ./conf.d:/etc/nginx/conf.d
```

[Editando o .conf do nginx]

```
nano conf.d/app.conf
```

[app.conf]

```
upstream app_backend {
    server K3S_SERVER_PRIV:30080; # troca pelo IP privado do k3s-server
}

server {
    listen 80;
    server_name _;

    location / {
        proxy_pass http://app_backend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Quando subir um Service NodePort 30080 no k3s, qualquer acesso ao IP do NGINX vai redirecionar para o app

[subindo a imagem]

```
sudo docker compose up -d
```

Expondo app no k3s

SSH no **k3s-server**

[Deploy simples + service nodeport]

```
nano app-test.yaml
```

[app-test.yaml]

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: demo-nginx
  template:
    metadata:
      labels:
        app: demo-nginx
    spec:
      containers:
        - name: nginx
          image: nginx:alpine
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: demo-nginx
spec:
  type: NodePort
  selector:
    app: demo-nginx
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30080
```

[Aplicar yaml]

```
kubectl apply -f app-test.yaml
kubectl get pods -o wide
kubectl get svc
```

Testando aplicação

Avaliando se o k3s está ativo

[Na instância k3s-server]

```
curl http://127.0.0.1:30080
```

Se o resultado for um html do nginx, funcionou.

[Na instância k3s-nginx]

```
curl http://<IP Privado do k3s-server>:30080
```

Se funcionar, o proxy também vai funcionar

[No computador pessoal]

```
curl http://<IP Elástico do k3s-nginx>
```

Ou abre no navegador

Se aparecer a página default do nginx, a infra está correta.

Escalabilidade Horizontal

Subindo mais um agente

1. Conecte-se ao k3s-server e guarde o token

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

2. Crie mais uma instância na EC2 com o Security Group do k3s-Server

3. No novo node, rode:

```
curl -sfL https://get.k3s.io | K3S_URL="https://K3S_SERVER_PRIV:6443"  
K3S_TOKEN=" < NODE TOKEN >" sh -
```

4. Execute:

```
kubectl get nodes
```

Agora temos 2 nodes balanceando a aplicação.

Subindo qualquer app docker

Com a infraestrutura montada, é fácil subir qualquer aplicação

Criar uma imagem Docker

exemplo: app python

[Iniciando uma aplicação flask]

```
mkdir meu_app
cd meu_app

nano app.py
```

[app.py]

```
from flask import Flask

app = Flask(__name__)

@app.get("/")
def home():
    return "deploy simples do app. Valeu k3s!"
```

[Criando Dockerfile]

```
nano Dockerfile
```

[Dockerfile]

```
FROM python:3.11-slim
WORKDIR /app
COPY . .
RUN pip install flask
CMD ["python", "app.py"]
```

Pode se usar qualquer imagem python disponível no dockerhub

[Buildar a imagem e subir pro docker hub]

```
docker build -t meuapp:v1 .

docker tag meuapp:v1 usuario/meuapp:v1
docker push usuario/meuapp:v1
```

Lembrando que é necessário uma conta no docker hub, o usuário será o username que você definir.

[Criar o Deployment no k3s-server]

```
nano meuapp.yaml
```

[meuapp.yaml]

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: meuapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: meuapp
  template:
    metadata:
      labels:
        app: meuapp
    spec:
      containers:
        - name: meuapp
          image: marcioferreira/meuapp:v1
          ports:
            - containerPort: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: meuapp-svc
spec:
  type: NodePort
  selector:
    app: meuapp
  ports:
    - port: 5000
      targetPort: 5000
      nodePort: 30081 # abra outras portas no security group se
necessário
```

[Aplicar deployment]

```
kubectl apply -f meuapp.yaml
```

Adicionando Gateway no Nginx

Para o nginx reconhecer sua nova aplicação

[Na instância k3s-nginx]

```
nano nginx/conf.d/meuapp.conf
```

[meuapp.conf]

```
location /demo {  
    proxy_pass http://< IP Privado do k3s-server >:30081;  
    proxy_set_header Host $host;  
}
```

[Reiniciar]

```
sudo docker compose restart
```

Agora acesse o endereço IP elástico do nginx, a saída deve ser o que foi feito usando Flask