Documentação - Implantação com Kubernetes e Helm (Minikube)

1. Visão Geral da Aplicação

Este projeto tem como base uma aplicação web composta por três principais componentes:

- **Spring App**: serviço backend principal da aplicação, responsável pelas regras de negócio.
- **Email Service**: serviço auxiliar que envia e-mails via SMTP (Gmail).
- **MySQL**: banco de dados relacional utilizado pela aplicação.

Essa aplicação foi originalmente criada e conteinerizada com Docker, e neste trabalho foi adaptada para ser implantada em um cluster **Kubernetes local (Minikube)**, utilizando **Helm Charts**.

2. Estrutura do Projeto com Helm

Foi criado um Helm Chart chamado `devops-app` para gerenciar todos os recursos do Kubernetes necessários para o funcionamento da aplicação.

Estrutura:

- `Chart.yaml`: arquivo de metadados do Helm.
- `values.yaml`: arquivo de configuração central com variáveis parametrizadas.
- `templates/`: pasta contendo todos os artefatos Kubernetes (Deployment, Service, Secret, etc).

3. Componentes / Containers

a) Spring App

- Container com aplicação principal.
- Expõe a porta 8080.

- Conecta-se ao MySQL através de variáveis de ambiente.
- A imagem é gerada localmente e carregada no Minikube.

b) Email Service

- Serviço secundário, responsável por envio de e-mails com autenticação Gmail.
- Usa variáveis sensíveis (usuário e senha do e-mail) armazenadas em um `Secret`.

c) MySQL

- Container de banco de dados relacional.
- Implantado como `StatefulSet` para garantir persistência e identidade estável.
- Dados armazenados em volume persistente.

4. Artefatos Kubernetes Utilizados

Deployments

- Definem como os containers (Spring e Email Service) são gerenciados no cluster.
- Usam variáveis de ambiente para configuração.

StatefulSet

- Utilizado para o MySQL, pois esse tipo de workload requer volume persistente e nome fixo entre reinicializações.

Services

- Criam uma camada de rede interna para que os pods possam se comunicar:
    - `ClusterIP` para comunicação entre os serviços.
    - `Headless Service` no MySQL para que o StatefulSet funcione corretamente.

Secrets

- Armazenam informações sensíveis de forma segura, como:
    - Credenciais do banco (`MYSQL_ROOT_PASSWORD`)
    - Usuário e senha do Gmail (`SPRING_MAIL_USERNAME`, `SPRING_MAIL_PASSWORD`)

Ingress

- Usado para expor os serviços externamente via domínio `k8s.local`.
- Permite acessar os serviços:
    - `/` para a aplicação principal
    - `/email` para o serviço de e-mail

5. Helm Chart: Parametrização com `values.yaml`

O arquivo `values.yaml` centraliza as configurações dos serviços, como:

- Nome das imagens
- Portas dos serviços
- Senha do banco
- Nome do banco de dados

Isso permite reutilizar os templates YAML com diferentes configurações facilmente.

6. Configuração de Acesso via Ingress

Para acessar a aplicação via navegador, foi configurado um recurso de **Ingress**, mapeando:

- `http://k8s.local/` → Spring App
- `http://k8s.local/email` → Email Service

Foi necessário editar o arquivo do sistema:

`C:\Windows\System32\drivers\etc\hosts`

E adicionar a linha:

`127.0.0.1 k8s.local`

Isso redireciona o tráfego local para o cluster Minikube.

IMPORTANTE!!

Configuração Inicial Obrigatória

Antes de executar o script de instalação ou instalar com o Helm, **você deve editar os seguintes arquivos para inserir suas próprias credenciais pessoais**:

- **`email-secret.yaml`**
  Altere os campos:
  - `SPRING_MAIL_USERNAME`: informe o seu e-mail válido.
  - `SPRING_MAIL_PASSWORD`: informe a senha do seu aplicativo (gerada no Gmail com autenticação de 2 fatores).
- **`values.yaml`**
  Altere os valores de:
  - `mysql.auth.rootPassword`: defina a senha do root do MySQL.
  - `mysql.auth.database`: nome do banco de dados (se necessário).

Essas alterações são necessárias para que sua aplicação funcione corretamente no ambiente do Minikube.

EXECUÇÃO DA APLICAÇÃO

Opção 1: Script de Automação (`scripts/build.sh`)

Um script bash foi criado para automatizar:

1. O **build** das imagens Docker localmente.
2. O **load** dessas imagens no ambiente do Minikube.

```
#!/bin/bash

set -e  # Para o script se algum comando falhar

echo "🔧 Iniciando Minikube..."
minikube start

echo "🐳 Construindo imagens Docker..."

docker build -t email-service:latest ../email-service
docker build -t custom-mysql:latest ../mysql
docker build -t spring-app:latest ..

echo "📦 Carregando imagens no Minikube..."
minikube image load spring-app:latest
minikube image load email-service:latest
minikube image load mysql:custom

echo "🚀 Instalando aplicação com Helm..."
helm install devops-app ../devops-app --wait

echo "✅ Aplicação implantada! Verificando pods..."
kubectl get pods

echo "🌐 Para acessar os serviços (LoadBalancer), execute em um novo terminal:"
echo "👉 minikube tunnel"
```

Esse script garante que todas as imagens estejam disponíveis no cluster antes da instalação

Opção 2: Execução Manual (passo a passo)

Siga os passos abaixo caso queira realizar tudo manualmente:

a) Iniciar o Minikube

- minikube start

b) Fazer build das imagens Docker

Certifique-se de estar no diretório correto do projeto:

Exemplo(No meu caso): cd C:\Users\Nataly\Documents\Devops\Trabalho2>

Build do email-service (dentro da pasta email-service)

- docker build -t email-service:latest ./email-service

Build do mysql customizado (dentro da pasta mysql)

- docker build -t mysql:custom ./mysql

Build da aplicação Spring Boot (Dockerfile está na raiz)

- docker build -t spring-app:latest .

c) Carregar as imagens no Minikube

- minikube image load spring-app:latest
- inikube image load email-service:latest
- minikube image load mysql:custom

d) Instalar com Helm

- helm install devops-app ./devops-app --wait

e) Verificar os pods

- kubectl get pods

f) Executar o tunnel do Minikube

- minikube tunnel

🌐 Acesse: `http://k8s.local`