DevOps - Trabalho 2

Alunos
Enzo Youji Murayama - 813606
Lucas Sciarra Gonçalves - 811948

Este documento detalha a nova abordagem para executar e gerenciar nosso projeto em ambiente de desenvolvimento local. Estamos migrando do uso direto do Docker (e docker-compose) para uma orquestração com Kubernetes, utilizando Minikube para simular um cluster local e Helm para gerenciar nossos pacotes de aplicação.

Por que essa mudança?

Paridade entre Ambientes: Nosso ambiente de desenvolvimento agora se assemelha muito mais aos ambientes de produção (Staging, Produção), que também rodam em Kubernetes. Isso reduz drasticamente os problemas do tipo "funciona na minha máquina".

Padronização: O Kubernetes é o padrão de mercado para orquestração de contêineres. Adotar essa tecnologia nos alinha com as melhores práticas da indústria.

Gerenciamento Simplificado: Com o Helm, podemos versionar, instalar e atualizar toda a nossa aplicação (composta por múltiplos serviços, bancos de dados, etc.) com um único comando.

Escalabilidade: Embora o foco aqui seja o desenvolvimento local, essa estrutura nos prepara para escalar nossos serviços de forma simples e declarativa no futuro.

Pré-requisitos

Antes de começar, garanta que você tenha as seguintes ferramentas instaladas e configuradas em sua máquina:

1. **Docker:** Continua sendo essencial para construir as imagens dos nossos contêineres. Instalação do Docker
2. **Minikube:** A ferramenta que cria um cluster Kubernetes de um único nó na sua máquina. Instalação do Minikube
3. **kubectl:** A interface de linha de comando para interagir com o cluster Kubernetes. Geralmente é instalado junto com o Docker Desktop ou pode ser instalado separadamente: Instalação do kubectl

4. **Helm:** O gerenciador de pacotes para o Kubernetes. Instalação do Helm

Arquitetura Kubernetes da Nossa Aplicação

1. **deployments**
   Um Deployment é a peça central. Ele é a "planta" ou o "projeto" que descreve como executar sua aplicação. Sua principal responsabilidade é garantir que um número específico de cópias idênticas do seu contêiner (chamadas de Pods) esteja sempre rodando.

2. **services**
   Pods são "mortais" e podem ser substituídos a qualquer momento, mudando de endereço IP. Um Service fornece um ponto de acesso estável e único (um endereço IP interno e um nome DNS) para um grupo de Pods.

3. **ingress**
   Enquanto os Services expõem sua aplicação dentro do cluster, o Ingress a expõe para o mundo exterior (a internet, ou no seu caso, sua máquina local). Ele atua como um roteador inteligente.

4. **secrets**
   Um Secret é um objeto feito especificamente para armazenar dados sensíveis, como senhas de banco de dados, chaves de API e tokens.

5. **pvc**
   Contêineres são efêmeros, e seus sistemas de arquivos são apagados quando eles morrem. Para dados que precisam persistir (como um banco de dados), usamos Volumes Persistentes. A PVC (PersistentVolumeClaim) é um "pedido" de armazenamento.

6. **jobs**
   Diferente de um Deployment que deve rodar para sempre, um Job cria um ou mais Pods para executar uma tarefa específica e, uma vez que a tarefa termina com sucesso, os Pods são encerrados.

Como rodar o projeto?

Esta seção descreve como configurar e executar o ambiente de desenvolvimento local completo.

O Caminho Fácil (Via Script Automatizado)

Para inicializar o ambiente, execute os seguintes comandos no terminal a partir da raiz do projeto:

```shell
# Concede permissão de execução ao script
chmod +x start.sh

# Executa o script
./start.sh
```

Este script realiza automaticamente as seguintes tarefas:

- **Verificação de Dependências:** Garante que minikube, kubectl e helm estejam instalados.
- **Inicialização do Minikube:** Inicia o Minikube com um perfil dedicado (trabalho1) para isolar este ambiente.
- **Habilitação de Addons:** Ativa o ingress, que é essencial para o roteamento de tráfego externo.
- **Carregamento de Imagens Docker:** Constrói as imagens Docker da aplicação e as carrega diretamente no registro interno do Minikube.
- **Criação de Secrets:** Cria os segredos Kubernetes (Secrets) necessários para senhas e certificados.
- **Configuração de Domínio Local:** Adiciona a entrada k8s.local ao seu arquivo /etc/hosts, apontando para o IP do Minikube.
- **Deploy com Helm:** Implanta toda a pilha de aplicações (API, frontend, etc.) usando nosso Helm Chart customizado.

Observação: Será solicitada sua senha de administrador (sudo) durante a execução. Isso é necessário para que o script possa editar o arquivo /etc/hosts e mapear o domínio k8s.local para o seu cluster Minikube.

Após a execução bem-sucedida, a aplicação estará acessível em: https://k8s.local

Ciclo de Vida do Ambiente e Comandos Úteis

I. **Verificando o Status**
   Após a inicialização, verifique se todos os componentes estão rodando:

```shell
# Veja os Pods, Services e o Ingress
kubectl get pods,svc,ingress
```

II. **Parando e Limpando o Ambiente**
Quando terminar de trabalhar, você pode parar ou deletar o ambiente para liberar recursos.

```shell
# Para desinstalar a aplicação, mas manter o cluster Minikube:
helm uninstall trabalho1

# Para parar o cluster Minikube (pode ser reiniciado depois):
minikube stop -p trabalho1

# Para deletar completamente o cluster e liberar todo o espaço em disco:
minikube delete -p trabalho1
```