Aplicação "Amizade" em Kubernetes

Autor: Luiz Gustavo da Silva Barros
Data: 17 de julho de 2025
Disciplina: DevOps

---

1. Visão Geral da Aplicação

Este documento apresenta a arquitetura e o processo de implantação da **Aplicação Amizade**, um sistema web completo baseado no conceito de microsserviços. A aplicação foi desenvolvida para rodar em um ambiente orquestrado com **Kubernetes**, garantindo escalabilidade, resiliência e uma separação clara de responsabilidades entre os seus componentes.

A arquitetura segue o modelo de **três camadas** (frontend, backend e banco de dados), todas containerizadas com **Docker**. A implantação no cluster local (Minikube) é totalmente automatizada e gerenciada por meio de um **Helm Chart**.

---

2. Arquitetura e Componentes (Containers)

A aplicação é dividida em três containers independentes, cada um com uma função específica:

2.1. Frontend (`lgstavo/frontend`)

- **Tecnologia:** Aplicação de página única (SPA) desenvolvida com **Vue.js**, servida por um servidor **NGINX** leve e eficiente.
- **Responsabilidade:** Responsável pela interface com o usuário. Exibe a UI, gerencia o estado da aplicação e faz requisições REST para o backend, realizando operações como cadastro, login, entre outras.

2.2. Backend (`lgstavo/backend`)

- **Tecnologia:** API RESTful desenvolvida em **Java** com o framework **Spring Boot**. Utiliza **Spring Security** para autenticação baseada em tokens JWT.
- **Responsabilidade:** É a camada lógica da aplicação. Faz validação de dados, gerenciamento de usuários e comunicação com o banco de dados. Expõe endpoints seguros consumidos pelo frontend.

2.3. Banco de Dados (`lgstavo/db`)

- **Tecnologia:** Imagem Docker customizada baseada no **MySQL**.

- **Responsabilidade:** Armazena todos os dados da aplicação (usuários, credenciais, relacionamentos, etc.). A imagem já inclui um script `init.sql` para criar automaticamente as tabelas e estruturas na primeira execução.

---

3. Implantação no Kubernetes (Minikube)

A implantação no cluster local Minikube é feita de forma **declarativa**, com todos os manifestos Kubernetes organizados e parametrizados via Helm Chart. Isso torna o processo mais previsível, fácil de manter e reaplicável.

3.1. Artefatos Kubernetes Utilizados

- **Deployment:** Usado para os serviços stateless (frontend e backend). Garante a execução do número desejado de réplicas e permite atualizações controladas (rolling updates).
- **StatefulSet:** Usado para o serviço stateful (MySQL). Fornece nomes de rede e volumes persistentes estáveis, essenciais para o banco de dados.
- **Service (ClusterIP):**
  - `amizade-app-frontend-service`: Acesso interno para os pods do frontend.
  - `amizade-app-backend-service`: Acesso interno para os pods do backend.
  - `amizade-app-mysql-service`: Permite ao backend se conectar ao banco de dados via DNS interno.
    Obs.: A escolha por ClusterIP evita expor os serviços diretamente ao exterior, aumentando a segurança.
- **Secret:** Armazena informações sensíveis como credenciais do banco de dados e segredo do JWT. Esses dados são injetados como variáveis de ambiente nos containers, mantendo o código limpo e seguro.
- **PersistentVolumeClaim (PVC):** Garante persistência dos dados do MySQL, mesmo que o pod seja reiniciado ou substituído.
- **Ingress:** É a porta de entrada da aplicação, gerenciada pelo **NGINX Ingress Controller**.
  Exposição da aplicação via: `http://amizade.k8s.local`
  - Rota `/` direciona para o frontend.
  - Rota `/api/` direciona para o backend.

3.2. Helm Chart (`app-chart`)

Todos os manifestos Kubernetes são organizados em um **Helm Chart**, o que traz várias vantagens:

- **Instalação com um único comando:**
  `helm install amizade-app ./app-chart`
- **Configuração centralizada:**
  O arquivo `values.yaml` permite ajustar imagens, réplicas e hosts sem editar os arquivos YAML diretamente.

- **Gerenciamento fácil:**
  Atualizações (`helm upgrade`) e remoções (`helm uninstall`) são simples e seguras.

---

4. Como Executar a Aplicação

Para rodar a aplicação no seu ambiente local, siga os passos abaixo:

1. **Pré-requisitos:**
   Ter o Docker, Minikube, kubectl e Helm instalados.

2. **Iniciar o cluster Minikube:**
   `bash minikube start`

3. **Ativar o Ingress Controller:**
   `bash minikube addons enable ingress`

4. **Construir e enviar as imagens para o Docker Hub:**
   ```bash docker build -t lgstavo/frontend ./frontend
   docker build -t lgstavo/backend ./backend
   docker build -t lgstavo/db ./db

   docker push lgstavo/frontend
   docker push lgstavo/backend
   docker push lgstavo/db ```

5. **Instalar a aplicação com Helm:**
   `bash helm install amizade-app ./app-chart`

6. **Configurar o acesso local:**
   Adicionar ao `/etc/hosts` (ou equivalente no Windows/Mac):
   `bash <ip-do-minikube> amizade.k8s.local`
   Descubra o IP com:
   `bash minikube ip`

7. **Acessar a aplicação:**
   Abra o navegador e vá até:
   http://amizade.k8s.local

---
   ```