Preparação do Ambiente Virtual Conteinerizado

Tutorial de como realizar o deploy dos contêineres necessários para iniciar o desenvolvimento do **Guavira.**

Deploy dos contêineres

1. Estruturação Inicial e Banco de Dados

Como definido previamente, decidiu-se iniciar a implementação do Guavira pela estruturação e criação do seu banco de dados. Para isso, é possível consultar a documentação disponibilizada pela UFVJM, que apresenta um guia para a criação de um contêiner com o serviço de banco de dados PostgreSQL, responsável por hospedar a estrutura do sistema.

2. Acesso ao Repositório e Clonagem do Projeto

Após obter acesso ao GitLab por meio do LDAP, que é o sistema de login institucional, e realizar a criação de uma chave SSH para autorizar operações a partir do seu computador, é possível prosseguir para a próxima etapa. Nessa fase, será necessário implementar no projeto da API o contêiner responsável por hospedar o banco de dados do Guavira.

Para isso, deve-se realizar a clonagem do projeto para o ambiente local, permitindo sua manipulação de forma segura. É importante destacar que os projetos clonados do GitLab devem estar localizados no diretório ***home*** do usuário, pois alguns comandos e operações dependem dessa localização. Após a clonagem, recomenda-se executar o comando de configuração de codificação, a fim de evitar que diferenças de encoding sejam interpretadas como alterações pelo Git.

```
git clone git@git.dds.ufvjm.edu.br:conta-institucional/api.git api


# entrar na pasta baixada
cd api


# Instrução para o Git ignorar alterações de permissão de arquivo (as permissões não são versionadas)
git config core.fileMode false
```

3. Configuração do LDAP e Variáveis de Ambiente

Para que o projeto da API funcione corretamente, é necessário que o serviço de LDAP esteja ativo, visto que grande parte das funcionalidades depende desse mecanismo de autenticação. **Para realizar o deploy dos contêineres, será preciso obter as credenciais do LDAP, que são confidenciais e devem ser solicitadas via RocketChat a um responsável do DSI, não sendo, portanto, incluídas nesta documentação.**

Com as credenciais em mãos, deve-se criar um arquivo `.env`, contendo as variáveis de ambiente necessárias para o funcionamento dos contêineres. Esse arquivo pode ser gerado a partir de uma cópia do `.env.example`, substituindo-se os valores pelas credenciais corretas.

```
cp .env.example .env
```

4. Deploy dos Contêineres

O processo de deploy será realizado com o auxílio de um ***Makefile*****,** que contém scripts responsáveis por executar comandos Docker. Inicialmente, é necessário clonar o projeto de automação, uma vez que ele contém o serviço de LDAP. Em seguida, deve-se realizar o login no HUB da UFVJM utilizando o comando apropriado.

```
docker login -u nome.sobrenome hub.dds.ufvjm.edu.br
```

Caso esteja utilizando o Docker Desktop, pode ocorrer um erro durante esse processo. Para solucioná-lo, é necessário acessar o arquivo `~/.docker/config.json` e remover o par chave-valor `"credsStore"`.

Após isso, pode-se prosseguir com a clonagem do projeto de automação e realizar o deploy apenas do contêiner de LDAP, sendo recomendado comentar os demais serviços para evitar consumo desnecessário de recursos da máquina.

```
# ir pra home do usuario
cd ~


# baixar o repositório
git clone git@git.dds.ufvjm.edu.br:dds/automacao.git
```

5. Execução dos Serviços

Por fim, com todas as configurações concluídas, pode-se utilizar o script `start` presente no Makefile para realizar o deploy dos contêineres da API. Paralelamente, deve-se utilizar o comando `docker compose up` para iniciar o contêiner de automação, garantindo o funcionamento completo do ambiente necessário para o desenvolvimento do sistema.

```
# cd ~/api
make start
```

```
# cd ~/automacao
docker compose up
```

Como etapa final de organização do ambiente de desenvolvimento, recomenda-se realizar o *attach* (ou abertura) dos dois projetos — API e automação — em uma IDE de sua preferência, como o PhpStorm. Isso facilita a navegação entre os arquivos, execução de comandos, edição de código e integração com ferramentas de versionamento. Em IDEs como o PhpStorm, é possível abrir ambos os projetos na mesma janela, utilizando a opção de *Attach Project*, ou simplesmente abrindo os diretórios simultaneamente no workspace.

Instalação do PHP, Composer e preparando ambiente Laravel no Linux

Tutorial: Preparação do Ambiente Laravel no Linux (Ubuntu)

Este tutorial descreve o processo de preparação de um ambiente Laravel em uma máquina Linux Ubuntu, considerando que:

- O projeto já foi clonado do GitLab;
- O arquivo `.env` já existe;
- O objetivo é preparar o ambiente para executar migrations e desenvolver a aplicação.

---

1. Verificando a versão do PHP exigida pelo projeto

Antes de instalar ou atualizar o PHP, é importante verificar qual versão é exigida pelo projeto.

Na raiz do projeto, abra o arquivo `composer.json` e procure pela seção `require`:

```
{
   "require": {
      "php": "^8.2",
      "laravel/framework": "^12.0"
   }
}
```

O parâmetro `"php"` define a versão mínima necessária.

Exemplos:

Configuração	Significado
`"php": "^8.1"`	PHP 8.1 ou superior
`"php": "^8.2"`	PHP 8.2 ou superior
`"php": "^8.2|^8.3"`	PHP 8.2 ou 8.3

Também é possível verificar utilizando o terminal:

```
cat composer.json | grep php
```

---

2. Verificando a versão atual do PHP

Para verificar a versão instalada:

```
php -v
```

Exemplo de saída:

```
PHP 8.1.2 (cli)
```

Caso a versão instalada seja inferior à exigida pelo projeto, será necessário instalar uma versão mais recente.

---

3. Instalando o PHP 8.2 no Ubuntu

Primeiramente, atualize os repositórios:

```
sudo apt update
```

Caso o PHP 8.2 não esteja disponível nos repositórios padrão, adicione o repositório mantido por Ondřej Surý:

```
sudo add-apt-repository ppa:ondrej/php
sudo apt update
```

Instale o PHP 8.2 juntamente com as extensões mais utilizadas pelo Laravel:

```
sudo apt install php8.2 \
php8.2-cli \
php8.2-common \
php8.2-mysql \
php8.2-mbstring \
php8.2-xml \
php8.2-curl \
php8.2-zip \
php8.2-bcmath \
```

```
php8.2-intl
```

---

4. Definindo o PHP 8.2 como versão padrão

Após a instalação, o sistema pode continuar utilizando a versão antiga.

Liste as versões disponíveis:

```
sudo update-alternatives --config php
```

Exemplo:

```
There are 2 choices for the alternative php:

Selection    Path
-----------------------------------------
0            /usr/bin/php8.2
1            /usr/bin/php8.1
2            /usr/bin/php8.2
```

Digite o número correspondente ao PHP 8.2.

Verifique novamente:

```
php -v
```

Saída esperada:

```
PHP 8.2.x
```

---

5. Instalando o Composer

O Composer é o gerenciador de dependências utilizado pelo Laravel.

Verifique se ele já está instalado:

```
composer --version
```

Caso o comando não seja encontrado, instale-o:

```
sudo apt install composer
```

Alternativamente, pode-se utilizar o instalador oficial disponível em:

Composer

---

6. Verificando a versão do Composer

Após a instalação:

```
composer --version
```

Exemplo:

```
Composer version 2.8.5
```

---

7. Instalando as dependências do projeto

Com o PHP e o Composer configurados, navegue até a raiz do projeto e execute:

```
composer install
```

Este comando:

- Lê o arquivo `composer.lock`;
- Baixa todas as dependências necessárias;
- Cria a pasta `vendor`;
- Gera o autoloader da aplicação.

Ao final, deverá existir a seguinte estrutura:

```
projeto/
├── app/
├── database/
├── routes/
├── vendor/
├── composer.json
├── composer.lock
└── .env
```

---

8. Verificando se as dependências foram instaladas corretamente

Após o término da instalação, execute:

```
php artisan
```

Se o ambiente estiver configurado corretamente, será exibida a lista de comandos do Artisan.

Exemplo:

```
Laravel Framework 12.x

Available commands:
  about
  migrate
  route:list
  serve
  ...
```

---

9. Verificando se o ambiente está pronto

Execute os seguintes comandos:

```
php -v
composer --version
php artisan
```

Se todos responderem corretamente, o ambiente está preparado para iniciar o desenvolvimento e executar as migrations.

Resumo dos comandos principais

```
# Verificar versão do PHP
php -v

# Instalar PHP 8.2
sudo add-apt-repository ppa:ondrej/php
sudo apt update

sudo apt install php8.2 \
php8.2-cli \
php8.2-common \
php8.2-mysql \
php8.2-mbstring \
php8.2-xml \
php8.2-curl \
php8.2-zip \
php8.2-bcmath \
php8.2-intl

# Definir PHP padrão
sudo update-alternatives --config php

# Verificar Composer
```

```
composer --version

# Instalar Composer
sudo apt install composer

# Instalar dependências do projeto
composer install

# Gerar APP_KEY
php artisan key:generate

# Verificar Laravel
php artisan
```

Após esses passos, o ambiente estará pronto para a criação e execução das migrations do projeto.