	**Configuração de Ambiente Docker**
Software:	Sistema de Apoio ao PDI.
Scrum Team:	Leandro Souza, Vinícius Assunção e Vitor Augusto.

Este tutorial detalhado descreve o processo de criação de um ambiente de desenvolvimento utilizando Docker, especificamente com a utilização do servidor de aplicação Payara. O ambiente é composto por dois containers: um para o servidor Payara e outro para o banco de dados MySQL.

O Payara foi escolhido como servidor de aplicação em substituição ao Glassfish por conta de suas melhorias e aprimoramentos em relação à versão original do Glassfish. O Payara é uma plataforma Java EE de código aberto e oferece uma série de recursos adicionais, correções de bugs e aprimoramentos de desempenho em relação ao Glassfish. Com o Payara, você terá uma plataforma de aplicação robusta e confiável para desenvolver e implantar seus aplicativos Java.

Após seguir o tutorial e iniciar os containers, o servidor Payara estará disponível na porta 8080 do seu servidor. Para acessar o console de administração do Payara, é necessário acessar a porta 4848 do servidor. Abra um navegador da web e digite o seguinte URL:

```
http://seu_endereco_ip:4848/
```

Substitua "`seu_endereco_ip`" pelo endereço IP do servidor onde o Docker está sendo executado.

Ao acessar o console do Payara, você será solicitado a fazer login. Utilize as seguintes credenciais:

Usuário: `admin`

Senha: `admin`

Após o login bem-sucedido, você terá acesso ao console de administração do Payara, onde poderá configurar e gerenciar suas aplicações Java EE, configurar pools de conexão, implantar aplicativos, monitorar o desempenho do servidor e muito mais.

Lembre-se de que este processo de criação do ambiente de desenvolvimento é apenas uma base inicial. Você pode personalizá-lo de acordo com suas necessidades específicas, adicionando mais containers, volumes e serviços conforme necessário.

Passo 1: Verificar a instalação do Docker

Abra um terminal.

Execute o seguinte comando para verificar se o Docker está instalado e obter a versão instalada:

```
docker --version
```

Se o Docker estiver instalado, você verá a versão do Docker sendo exibida. Caso contrário, prossiga com as etapas a seguir.

Execute os seguintes comandos para instalar o Docker no Ubuntu 22.04:

```
sudo apt update
sudo apt install docker.io
```

Durante a instalação, você será solicitado a confirmar a instalação e fornecer a senha do usuário root. Insira a senha e aguarde a conclusão da instalação.

Verifique novamente a versão do Docker para confirmar que a instalação foi bem-sucedida:

```
docker --version
```

Passo 2: Criação do diretório do projeto

Escolha um diretório para o projeto. Por exemplo, você pode criar um diretório chamado "docker-project" em sua pasta pessoal.

Execute o seguinte comando para criar o diretório:

```
mkdir ~/docker-project
```

Navegue até o diretório do projeto usando o comando:

```
cd ~/docker-project
```

Passo 3: Criação do arquivo docker-compose.yml

Crie um arquivo chamado `docker-compose.yml` dentro do diretório do projeto.

```
nano docker-compose.yml
```

No editor de texto Nano, copie e cole a seguinte configuração no arquivo `docker-compose.yml`:

```
version: '3.3'
services:
  servidor-web:
    restart: always
    container_name: payara-sapdi
    image: payara/server-full:5.2022.5-jdk11
    ports:
      - "8080:8080"
      - "443:443"
      - "4848:4848"
    volumes:
      -
./autodeploy:/opt/payara/appserver/glassfish/domains/domain1/autodeploy
    depends_on:
      - mysql-db

  mysql-db:
    restart: always
    container_name: mysql-sapdi
    image: mysql/mysql-server:8.0.32
    env_file:
      - "variables.env"
    volumes:
      - ./db_data:/var/lib/mysql
```

Pressione "Ctrl + O" para salvar o arquivo e depois "Enter". Em seguida, saia do editor de texto pressionando "Ctrl + X".

Passo 4: Criação do arquivo variables.env

Crie um arquivo chamado `variables.env` dentro do diretório do projeto.

```
nano variables.env
```

No editor de texto Nano, copie e cole o seguinte conteúdo no arquivo `variables.env`:

```
MYSQL_DATABASE=recuperacao_paralela
MYSQL_HOST=mysql-db
MYSQL_PORT=3306
MYSQL_USER=root
MYSQL_PASSWORD=root
MYSQL_ROOT_PASSWORD=root
```

Pressione "Ctrl + O" para salvar o arquivo e depois "Enter". Em seguida, saia do editor de texto pressionando "Ctrl + X".

Passo 5: Inicialização dos containers

Execute o seguinte comando para iniciar os containers:

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

O parâmetro -d é usado para executar os containers em segundo plano.

Aguarde enquanto os containers são criados e iniciados com base nas informações fornecidas no arquivo `docker-compose.yml`.

Passo 6: Verificação dos containers iniciados

Execute o seguinte comando para verificar quais containers foram iniciados:

```
sudo docker ps
```

Isso exibirá uma lista dos containers em execução, incluindo os nomes e os IDs.

Relatório de Containers criados

Container: `payara-sapdi`

Descrição: Esse container é baseado na imagem `payara/server-full:5.2022.5-jdk11`.

Atributos relevantes:

`restart: always`: Garante que o container seja reiniciado automaticamente em caso de falha ou reinicialização do sistema.

`container_name: payara-sapdi`: Define o nome do container como "`payara-sapdi`".

`image: payara/server-full:5.2022.5-jdk11`: Indica a imagem a ser usada para criar o container.

`ports`: Mapeia as portas do host para as portas do container.

`volumes`: Mapeia um diretório do host para um diretório dentro do container.

`depends_on`: Especifica que esse container depende do container "`mysql-db`".

Container: `mysql-sapdi`

Descrição: Esse container é baseado na imagem `mysql/mysql-server:8.0.32`.

Atributos relevantes:

`restart: always`: Garante que o container seja reiniciado automaticamente em caso de falha ou reinicialização do sistema.

`container_name: mysql-sapdi`: Define o nome do container como "`mysql-sapdi`".

`image: mysql/mysql-server:8.0.32`: Indica a imagem a ser usada para criar o container.

`env_file`: Especifica o arquivo variables.env para definir as variáveis de ambiente do container.

`volumes`: Mapeia um diretório do host para um diretório dentro do container.