01

# Preparação dos ambientes

## Transcrição

O script que o instrutor segue durante a aula é o seguinte:

```
# Instalar o plugin Config File Provider

# Configurar o Managed Files para Dev
    # Name : .env-dev
        [config]
        # Secret configuration
        SECRET_KEY = 'r*5ltfzw-61ksdm41fuul8+hxs$86yo9%k1%k=(!@=-wv4qtyv'
        # conf
        DEBUG=True
        # Database
        DB_NAME = "todo_dev"
        DB_USER = "devops_dev"
        DB_PASSWORD = "mestre"
        DB_HOST = "localhost"
        DB_PORT = "3306"

#Configurar o Managed Files para Prod
    # Name: .env.-prod
        [config]
        # Secret configuration
        SECRET_KEY = 'r*5ltfzw-61ksdm41fuul8+hxs$86yo9%k1%k=(!@=-wv4qtyv'
        # conf
        DEBUG=False
        # Database
        DB_NAME = "todo"
        DB_USER = "devops"
        DB_PASSWORD = "mestre"
        DB_HOST = "localhost"
        DB_PORT = "3306"

# No job: jenkins-todo-list-principal importar o env de dev para teste:

    Adicionar passo no build: Provide configuration Files
    File: .env-dev
    Target: ./to_do/.env

    Adicionar passo no build: Executar Shell

# Criando o Script para Subir o container com o arquivo de env e testar a app:
    #!/bin/sh

    # Subindo o container de teste
    docker run -d -p 82:8000 -v /var/run/mysqld/mysqld.sock:/var/run/mysqld/mysqld.sock -v /var/lib/

    # Testando a imagem
    docker exec -i todo-list-teste python manage.py test --keep
```

```
exit_code=$?

# Derrubando o container velho
docker rm -f todo-list-teste

if [ $exit_code -ne 0 ]; then
    exit 1
fi
```

[00:00] Oi pessoal, tudo bem? Nessa aula a gente vai entender como separar os ambientes da nossa aplicação. Nesse caso nós vamos utilizar dois ambientes que é o ambiente de desenvolvimento e o ambiente de produção.

[00:13] O que diferencia um ambiente do outro? É somente o arquivo .env. Se vocês se lembram, na aula que nós criamos o build manual, esse arquivo .env contém o endereço do banco, usuário, senha, algumas chaves de configuração de criptografia. Então a gente tem que separar esses arquivos porque ele não está versionado no nosso GitHub. Isso é muito importante porque é o que garante a segurança da nossa aplicação.

[00:41] E ele vai ser utilizado em tempo de execução. Como é que a gente faz isso dentro do Jenkins? Nós vamos precisar de um plugin chamado Config File Provider. Então a gente vai fazer a instalação dele agora. Dentro de gerenciar plugins, disponíveis, a gente vai digitar config file provider e vamos clicar pra instalar esse plugin. Ele vai instalar, a instalação dele é muito rápida.

[01:11] E agora a gente vai fazer a configuração desse plugin. Como que nós configuramos esse plugin? Dentro de "Gerenciar o Jenkins" nós vamos procurar uma entrada chamada "Managed files" dentro dessa configuração nós vamos criar dois arquivos.

[01:27] O primeiro deles vai ser um arquivo custom, se vocês olharem tem outros padrões aqui pré-determinados, aí vocês podem utilizar de acordo com a aplicação de vocês, nesse caso a gente vai criar um arquivo customizado. Ele vai me dar um ID, esse ID vai ser importante lá na frente quando a gente for configurar os nossos jobs. Mas, por enquanto, a gente vai clicar em submit, ele vai pedir um nome, vai ser .env-dev, e o conteúdo.

[01:58] No caso de desenvolvimento o conteúdo é esse aqui que a gente tá apontando pro banco de desenvolvimento. Vamos dar um submit.

[02:09] E agora a gente vai criar o arquivo de prod que é um arquivo customizado, um outro ID, isso aí é automático. O nome dele vai ser .env-prod e o conteúdo vai ser quase o mesmo, a diferença é que agora nós vamos apontar pro banco de produção.

[02:32] Então configurado, clicamos em submit. Vamos voltar lá no nosso job pra puxar esses arquivos quando necessário. Então, dentro desse job principal, qual vai ser a próxima etapa nossa dentro desse pipeline? Vai ser fazer o teste da aplicação. Nesse caso, pra testar, nós vamos utilizar o ambiente de Dev, o arquivo .env de dev. Nunca se deve apontar testes pra produção.

[03:05] Pra isso, dentro do job, a gente vai em Ambiente de build e vai ter uma opção chamada Provide Configuration files, clicando aqui ele vai mostrar um drop down com os dois arquivos criados, nesse caso a gente vai usar o arquivo de dev.

[03:24] Qual que é o target? Aonde ele vai salvar esse arquivo? Lembra que eu comentei que cada job tem um diretório dentro do Jenkins? Nós vamos salvar na raiz desse job com o nome .env, que é aquele arquivo que nós criamos lá quando fizemos o build manual. Só que nesse caso eu não preciso me preocupar em colocar ele dentro de "todo" na nossa aplicação, por quê? Nós, quando executarmos o container, ele vai receber como parâmetro de volume esse arquivo e aí ele vai subir a aplicação apontando.

[04:02] Então agora, que nós já temos o nosso arquivo de ambiente, tá na hora da gente rodar o teste da nossa aplicação. Pra isso, depois do linter ter rodado, ou seja, esse primeiro teste ter passado que o nosso Dockerfile é saudável e o build ter acontecido com esse nome de imagem, nós vamos adicionar um próximo step que vai ser um shell script, e esse shell vai ter esse conteúdo aqui. Vamos copiar lá e eu vou explicar pra vocês o que ele faz.

[04:43] Então basicamente, vamos expandir um pouquinho aqui pra ficar mais fácil de enxergar, eu vou rodar um container na porta 82. Porque que eu tô rodando na porta 82? Por que produção vai rodar na porta 80, que é a porta padrão; desenvolvimento vai rodar na porta 81, a gente vai chegar lá ainda, vai ser um job que vai criar só nossa aplicação rodando pra devs acessarem; e, pra teste, eu vou subir um container muito rápido na porta 82, passando dois parâmetros de volume.

[05:18] O primeiro é o volume do sock do MySQL, como o contêiner é imutável, ele não tem a capacidade de acessar o sock dele mesmo e, como eu tô utilizando um banco externo, eu tô mapeando pra minha aplicação que o sock, dentro de /var/run/mysqld/mysqld.sock, vai estar localmente localizado no mesmo lugar e não dentro do container. Sem essa opção nossa aplicação não vai funcionar.

[05:49] E tô passando um outro parâmetro pra ele, de volume, que é dentro de /var/lib/jenkins/workspace. Olha o nosso diretório aqui do job, lembra que eu falei que o Jenkins tem um diretório pra cada job? E eu tenho um arquivo de ambiente que a gente vai colocar aqui só o .env porque nós salvamos ele localmente e na raiz do nosso projeto.

[06:14] Agora passamos mais um último parâmetro que é o nome do nosso container, que seria todo-list-teste e a imagem. Reparem que eu tenho, como parâmetro último aqui, a imagem que eu vou fazer o build, o problema é que ainda nós não temos essa configuração definida. A gente vai entender daqui a pouquinho como funciona. De momento, que que nós vamos fazer? Copiar o nome da imagem e colocar a imagem dentro desse valor.

[07:03] Então agora temos todo o comando de run definido, ele vai subir esse container. Com esse container em execução, o que que a gente vai fazer agora? Nós vamos rodar o teste unitário da nossa aplicação, nesse caso é um teste bem simples que é para demonstrar pra vocês como funciona esse step. No caso de vocês, nas aplicações do dia-a-dia, os testes podem mudar, mas nesse caso é assim que funciona.

[07:33] Como é que ele executa o teste? Ele vai dar um exec nesse container todo-list-teste, que nós acabamos de subir, e passar um comando "python manage.py test". Eu tenho, dentro do meu código fonte, um arquivo de teste que vai testar se a minha aplicação tá saudável.

[07:54] Então aqui olha, se nós pegarmos o exit_code e ele não for igual a zero, qualquer coisa diferente de zero, então eu vou falhar esse step. Esse step falhando, o meu pipeline vai falhar. Feito isso, nós vamos salvar e agora nós vamos construir o nosso projeto.

[08:20] Vamos dar uma olhadinha aqui como é que tá os logs de execução. Olha, então enquanto ele tá construindo, só pra vocês entenderem o que tá acontecendo, ele puxou os arquivos lá do meu repositório e aqui ele tá copiando o arquivo de env, o env-dev, pra .env, por isso que a gente mapeou daquela maneira. Ele tá fazendo o build da minha imagem e, agora, assim que ele terminar o build a gente volta.

[08:53] Bom, terminou e, se a gente olhar os logs aqui, ele executou um teste e esse teste foi com sucesso. Então agora o nosso job já tá fazendo os testes puxando a configuração de ambiente.

[09:10] Na próxima aula a gente vai aprender como passar parâmetros para outros jobs e com isso a gente vai conseguir construir tanto nosso ambiente de desenvolvimento quanto nosso ambiente de produção. Beleza, a gente se vê lá.