06

Build da imagem

Transcrição

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

```
# Sugested Plugins
# Instalando os plugins
    Gerenciar Jenkins -> Gerenciar Plugins -> Disponíveis
        # docker
    Install without restart -> Depois reiniciar o jenkins
Gerenciar Jenkins -> Configurar o sistema -> Nuvem
    # Name: docker
    # URI: tcp://127.0.0.1:2376
    # Enabled
# This project is parameterized:
    DOCKER_HOST
    tcp://127.0.0.1:2376
# Voltar no job criado na aula anterior
    # Manter a mesma configuracao do GIT para desenvolvimento
    # Build step 1: Executar Shell
# Validando a sintaxe do Dockerfile
docker run --rm -i hadolint/hadolint < Dockerfile
    # Build step 2: Build/Publish Docker Image
        Directory for Dockerfile: ./
        Cloud: docker
        Image: rafaelvzago/django_todolist_image_build
```

[00:00] E aí pessoal, beleza? Agora chegou a hora tão esperada de configurar o nosso job pra fazer o build automático da aplicação, dentro de um container. Apesar da explicação ser grande, a aplicação é muito simples.

[00:16] Então vamos lá. Primeira coisa, a gente vai logar no nosso Jenkins e a gente vai instalar alguns plugins, na verdade a gente vai instalar um plugin agora que é necessário pra que isso funcione.

[00:32] Então a gente vem em gerenciar Jenkins, gerenciar plugins, aqui a gente vem em disponíveis, que é o marketplace de plugins que o Jenkins tem, eu escrevo docker. Tá vendo aqui, esse aqui é o plugin do Docker. A gente vai clicar nesse carinha aqui pra instalar, vai mandar instalar e depois reiniciar. Ele vai instalar algumas dependências e, assim que terminar a instalação das dependências, a gente volta.

[01:06] Legal, terminou a instalação, agora a gente vai mandar reiniciar o Jenkins pra que ele carregue esse plugin novo. Enquanto ele reinicia, uma explicação básica sobre esse plugin: nós vamos habilitar o daemon, que nós configuramos no servidor, dentro dessa configuração de plugin, pra que o meu Jenkins seja capaz de executar comandos Docker remotamente, nesse caso vai ser localmente mas poderia ser em qualquer servidor. Então assim que terminar de reiniciar a gente volta.

[01:41] Então pessoal, enquanto instala e reinicia, só pra explicar o que a gente acabou de fazer: a gente instalou um client, um plugin do Docker, pra que ele possa acessar um servidor Docker e executar os comandos. Nesse caso vai estar na mesma máquina, mas a gente tá habilitando isso pra instalações e intervenções futuras. Então a gente vai esperar reiniciar, quando reiniciar a gente volta.

[02:04] Pronto, reiniciou, então agora a gente loga novamente, vou marcar aqui pra ele me manter logado e eu tenho que fazer algumas configurações agora. Primeira coisa que eu vou fazer: eu vou vir aqui no Jenkins, Gerenciar Jenkins, Configurar o sistema, é a configuração básica do Jenkins, e aí lá no final da página eu tenho Nuvem, se você tiver usando em inglês vai estar escrito Cloud, e eu vou adicionar uma nova Nuvem.

[02:44] O que que seria essa nova nuvem? São daemons que serão chamados. Basicamente assim, a gente tá instalando um plugin que vai chamar um daemon, então nessa parte ele tá aparecendo Docker aqui por que o plugin tá habilitado. É como o Jenkins interage com a Nuvem, nesse caso vai ser localmente mas poderia ser em qualquer lugar.

[03:05] Então quando eu cliquei aqui ele falou o seguinte "Olha, qual que é o nome que você vai dar?", a gente vai dar de docker mesmo aí ele pede os detalhes, a gente vai fazer uma configuração muito simples que é, onde ele pede qual que é a URI de conexão, a gente vai colocar tcp, o IP do servidor que tem o daemon exposto e a porta. Nesse caso eu coloquei 127.0.0.1 porque tá rodando nessa máquina. E aí eu vou a testar conexão.

[03:36] Olha lá, ele encontrou a versão 1809 do meu Docker rodando e o daemon exposto ou seja, aquela configuração nossa funcionou e agora o nosso Jenkins tá habilitado pra executar comandos em Docker nesse servidor.

[03:51] A gente vai salvar e chegou a parte de terminar o nosso build, fazer que o nosso job construa a nossa imagem automaticamente. Como é que a gente vai fazer isso? A gente vai clicar aqui dentro do job, que nós já tínhamos criado, configurar, e nós vamos adicionar dois passos de build.

[04:16] Como é que o Jenkins funciona? Aqui embaixo eu tenho os meus build steps, que são os passos do meu build, eu posso ter quantos passos eu quiser. Primeiro passo que eu vou configurar é um passo muito simples que vai fazer uma checagem muito rápida de como é que o meu Dockerfile está, se ele está ou não está de acordo com as últimas convenções, ele vai fazer um linter do meu Docker.

[04:45] Pra isso, que que eu faço? Eu vou adicionar um Build step pra executar um shell e, nessa execução do shell, que que eu vou fazer? Eu vou colar só isso aqui. Vou tirar o espaço adicional aqui. O que ele tá fazendo? Ele tá usando uma imagem chamada hadolint que ele vai fazer o linter do meu Dockerfile.

[05:11] De novo, se você tá em dúvida do que o Dockerfile faz e exatamente como a gente constrói, tem o curso de Docker da Alura pra vocês assistirem.

[05:21] Agora, se esse step faz passar, ou seja se não falhar, eu vou executar um segundo Buildstep que é o build da minha imagem do Docker, que agora sim eu vou automatizar tudo.

[05:36] Primeiro ele me pergunta onde que o meu Dockerfile está. Só para relembrar, dentro do root aqui da minha aplicação eu tenho meu Dockerfile que, se eu olhar aqui o conteúdo dele rapidinho, basicamente ele tá copiando os meus arquivos pra dentro de um diretório, copiando o arquivo de requerimentos, rodando o pip install, que nós fizemos tudo isso na mão, expondo a porta 8000 e executando a aplicação na porta 8000.

[06:12] Lembrando que quando eu construo a imagem, não necessariamente eu estou rodando meu container. Se eu não pedir pra rodar eu só vou até a imagem lá.

[06:20] Então vamos lá. A primeira coisa, vamos ver quantas imagens eu tenho aqui. sudo docker images, eu tenho só o hadolint, então tá zerada a minha instalação. Então o meu Dockerfile tá dentro do próprio diretório, porque quando ele dá o clone pra este diretório ele vai jogar dentro todos os arquivos. E aí ele vai perguntar qual que é o Cloud que vai construir, vai ser o Cloud do Docker que a gente já montou.

[06:49] E agora eu preciso dar um nome pra essa imagem, a gente vai usar esse nome aqui: django_todolist_image_build, é um nome de uma imagem que a nossa aplicação vai ter. E agora eu vou dar um salvar.

[07:12] Tem um detalhe que a gente precisa voltar a fazer que é: dentro do meu Jenkins eu preciso habilitar o meu plugin do Docker porque eu só configurei o daemon. Então como é que eu faço isso? Eu vou lá embaixo em Cloud, Cloud details e marco o enable. Se eu não fizer isso, esse Cloud provider, que ele vai usar, não vai funcionar, eu preciso habilitar, ele vai exibir naquela lista mas ele não vai estar habilitado. Feito isso é simples agora, eu venho aqui no meu job e mando construir.

[07:52] Então, resumindo, ele vai clonar o meu repositório, vai ler o meu Dockerfile, vai construir a minha imagem e vai registrar ela localmente. Então, assim que terminar a gente volta.

[08:07] Bom, o build acabou, foi um build com sucesso. Se a gente chegar na máquina agora e digitar sudo docker images, nossa imagem foi criada. Então quais são os próximos passos agora? É aprimorar esse nosso build, fazer alguns tratamentos de erro pra ver se não tem nada que vai quebrar o nosso código, e colocar nossa aplicação pra rodar da mesma maneira, automatizada sem que a gente tenha que ficar configurando os jobs na mão.

[08:43] Eu vejo vocês na próxima.