FIAP GRADUAÇÃO

FIAP

TECNOLOGIA EM DESENVOLVIMENTO DE SISTEMAS

DevOps Tools & Cloud Computing

2o Checkpoint 1o Semestre – Dockerfile

PROF. JOÃO MENK profjoao.menk@fiap.com.br

Projeto da disciplina: Dimdim

Uma POC no caminho,
no caminho uma POC...

Projeto da Disciplina: Dimdim

Desafios

A consultoria do seu grupo irá entrar novamente em ação para ajudar Steves Jobs e a DimDim

Sua equipe de DevOps foi encarregado de implantar várias tecnologias para realizar testes de migração do ambiente de desenvolvimento para Containers Docker

Serão três tecnologias utilizadas: **Java**, **Python** e **NodeJS**

Sua equipe precisa realizar a implantação dessas tecnologias criando Dockerfiles e realizando a execução em Containers

Colha as evidências desses testes para a equipe de Arquitetura da DimDim

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

Você é um membro da equipe de DevOps da DimDim e foi encarregado de criar um Dockerfile para implantar uma aplicação **Java** no servidor **Tomcat 10**. A aplicação Java simplesmente imprime a mensagem "Deploy efetuado com sucesso no Servidor Tomcat 10" na tela quando é executada no servidor Tomcat

01) Escreva um programa Java que imprima na tela Web a mensagem desejada

02) Crie um Dockerfile que:

2.1) Use o Tomcat 10 como imagem

2.2) Copie o arquivo WAR da aplicação para o diretório padrão de implantação do Tomcat

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

2.3) Exponha a porta adequada para que a aplicação seja acessível pelo nosso Host

2.4) Defina o diretório de trabalho como:

/usr/local/tomcat/webapps

Obs: Certifique-se de que o programa Java esteja empacotado em um arquivo WAR com a estrutura correta para realizar o Deploy no Tomcat 10

03) Crie um Volume com o nome: **deploys-tomcat10**

04) O arquivo WAR deve ter o seguinte nome:

DimMoneyApp<*seuRM*>.war (Exemplo: DimMoneyAppPF0841.war)

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

05) O nome da Imagem a ser criada deve ser: **dimmoney-app**

06) Rode o Container com base nessa imagem em Segundo Plano com o nome **dimmoney<seuRM>**, Volume apontando para o diretório padrão do Container e as outras propriedades ncessárias

07) Verifique os LOGs do Container (aplicação rodando no Tomcat)

08) Acesse a Aplicação em seu Host via Web Browser

09) Realize o *undeploy* dessa aplicação pelo Volume do seu Host

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

10) No código fonte, altere a mensagem para:

Deploy efetuado com sucesso no Servidor Tomcat 10. Bom trabalho!

Agora realize o Deploy novamente, **sem recriar a imagem**, por comando do Docker no Terminal

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

11) A entrega serão os seguintes Prints em um arquivo PDF com o nome: **CP2_<seugrupo>.PDF** e o link no Github com o projeto completo (incluindo o Dockerfile)

11.1) Print do Dockerfile criado

11.2) Print da tela da criação da imagem com sucesso (um print do terminal)

11.3) Print da execução do *docker container run* (um print do terminal com o comando executado com sucesso)

11.4) Print do diretório: */usr/local/tomcat/webapps/* (print do terminal do Container com *ls -l Não acessar pelo terminal do Docker Desktop, entrar pelo comando do Docker no terminal*)

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

11.5) Print do Log do Container (print do terminal. *Não acessar pelo log do Docker Desktop*). Pode ser somente as últimas linhas, o que couber no seu terminal

11.6) Print do Web Browser com a aplicação rodando (pegar a tela completa com o link da aplicação)

11.7) Print do Volume em seu Host (print da tela do Windows Explorer ou Console para Mac/Linux) com o **Deploy efetuado**

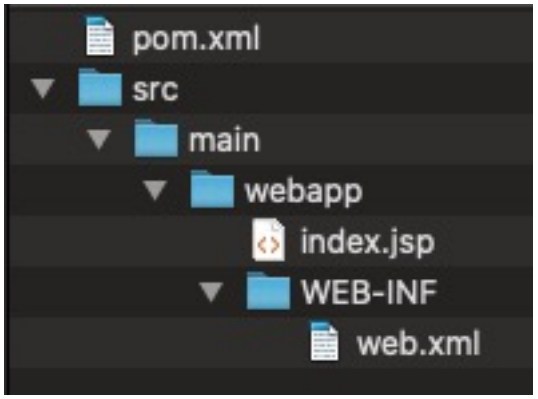
11.8) Print do Volume em seu Host (print da tela do Windows Explorer ou Console para Mac/Linux) com o **Undeploy efetuado**

Projeto da Disciplina: Dimdim

Primeiro desafio - Tecnologia Java

11.9) Print do Volume em seu Host (print da tela do Windows Explorer ou Console para Mac/Linux) com o **Redeploy efetuado**

Gabarito



web.xml

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>dimmoneyPF0841</display-name>
</web-app>
```

index.jsp

```
<html>
  <body>
    <h2>Deploy efetuado com sucesso no Servidor Tomcat 10. Bom trabalho!</h2>
  </body>
</html>
```

Gabarito

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
  <groupId>com.hello</groupId>
  <artifactId>dimmoneyPF0841</artifactId>
  <packaging>war</packaging>
  <version>1.0-SNAPSHOT</version>

  <name>dimmoneyPF0841 Maven Webapp</name>
  <url>http://maven.apache.org</url>

  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.2.3</version>
          <configuration>
              <failOnMissingWebXml>false</failOnMissingWebXml>
              <warName>dimmoneyPF0841</warName>
          </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

Gabarito

Empacotar o código

```
mvn clean package
```

Dockerfile

```
FROM tomcat:10
WORKDIR /usr/local/tomcat/webapps
COPY target/dimmoneyPF0841.war .
EXPOSE 8080
```

Volume

```
docker volume create deploys-tomcat10
```

Imagem

```
docker build -t dimmoney-app .
```

Container

```
docker container run --name dimmoneyPF0841 -d -p 8080:8080 -v deploys-tomcat10:/usr/local/tomcat/webapps dimmoney-app
```

log da aplicação

```
docker logs dimmoneyPF0841
```

Terminal do Container

```
docker container exec -it dimmoneyPF0841 bash
```

Redeploy

```
docker cp target/dimmoneyPF0841.war dimmoneyPF0841:/usr/local/tomcat/webapps/
```

Projeto da Disciplina: Dimdim

Segundo desafio - Tecnologia Python

Você é um membro da equipe de DevOps da DimDim e foi encarregado de criar um Dockerfile para implantar uma aplicação **Python**. A aplicação Python simplesmente imprime a mensagem "Implantação efetuada com sucesso" **na saída do terminal** quando é executada

01) Escreva um programa Python que imprime a mensagem desejada. Nome do arquivo: **app<seuRM>.py**

02) Crie um Dockerfile que:

2.1) Use a imagem Python na versão *3.9-slim* como base

2.2) Defina o diretório de trabalho como: */app*

Projeto da Disciplina: Dimdim

Segundo desafio - Tecnologia Python

2.3) Utilize um **Argumento** para receber o nome da Aplicação no Build e uma **Variável de Ambiente** para ser executada no CMD

2.4) Copie o arquivo Python da aplicação para o Container

2.5) Configure o comando padrão para executar seu App no Python. Utilizar o CMD

03) O nome da Imagem a ser criada deve ser: **dimmoney-python**

04) Rode o Container com base nessa imagem. Não precisa rodar em modo interativo ou em Segundo Plano nem possuir nome. Somente a execução do Container. Inclua no parâmetro a remoção do Container assim que for executado

Projeto da Disciplina: Dimdim

Segundo desafio - Tecnologia Python

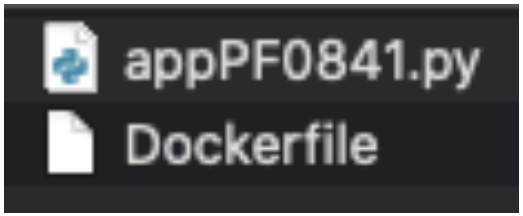
05) A entrega serão os seguintes Prints em um arquivo PDF com o nome: **CP2_<seugrupo>.PDF** e o link no Github com o projeto completo (incluindo o Dockerfile)

5.1) Print do Dockerfile criado

5.2) Print da tela da criação da imagem com sucesso (um print do terminal)

5.3) Print da execução do *docker container run* (um print do terminal com o comando executado com sucesso)

Gabarito



appPF0841.py

```
echo 'print("Implantação efetuada com sucesso")' > appPF0841.py
```

Dockerfile

```
FROM python:3.9-slim
ARG nomeApp
ENV nomeAppRun=${nomeApp}
WORKDIR /app
ADD ${nomeApp}.py .
CMD python ${nomeAppRun}.py
```

Imagem

```
docker build -t dimmoney-python --build-arg nomeApp="appPF0841" .
```

Container

```
docker run --rm dimmoney-python
```

Projeto da Disciplina: Dimdim

Terceiro desafio - Tecnologia NodeJS

Como último desafio crie um Dockerfile para implantar uma aplicação em **NodeJS**. A aplicação mostra a mensagem "Implantação efetuada com sucesso" na página Web quando executada

01) Escreva um programa Node.js que imprime a mensagem desejada em uma página Web (Tem que ser Web, não console)

02) Crie um Dockerfile que:

2.1) Use uma imagem Node na versão *lts-alpine3.19* como base

2.2) Utilize o usuário **node** para executar a aplicação

2.3) Defina o diretório de trabalho como: */app-money*

2.4) Copie todos os diretórios e arquivos da sua aplicação para o Container

Projeto da Disciplina: Dimdim

Terceiro desafio - Tecnologia NodeJS

2.5) Exponha a porta adequada para que a aplicação seja acessível

2.6) Configure o comando padrão para executar seu App NodeJS. Pode utilizar CMD ou ENTRYPOINT

03) Realize o empacotamento do seu App no seu host primeiramente

04) Somente depois crie a imagem. O nome da Imagem a ser criada deve ser: **dimmoney-node**

05) Rode o Container com base nessa imagem em segundo plano com o nome **dimmoney-node<seuRM>**, utilize um volume para mapear seu diretório do App no Host e apontando para o diretório de trabalho do Container (não criar Volume no Dockerfile, comando direto na linha de parâmetros) e inclua os outros parâmetros necessários

Projeto da Disciplina: Dimdim

Terceiro desafio - Tecnologia NodeJS

06) Após realizar os testes, altere a mensagem para: Implantação efetuada com sucesso. Bom trabalho!. Agora realize os procedimentos:

6.1) Empacote novamente seu App no Host

6.2) Crie novamente a imagem

6.3) Rode o container novamente

6.4) Realize os testes

Projeto da Disciplina: Dimdim

Terceiro desafio - Tecnologia NodeJS

07) A entrega serão os seguintes Prints em um arquivo PDF com o nome: **CP2_<seugrupo>.PDF** e o *link no Github* com o projeto completo (incluindo o Dockerfile)

7.1) Print do Dockerfile criado

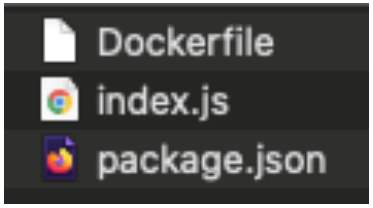
7.2) Print da tela da criação da imagem com sucesso (um print do terminal)

7.3) Print da execução do *docker container run* (um print do terminal)

7.4) Print do Web Browser com a aplicação rodando (pegar a tela completa com o link da aplicação)

7.5) Print dos procedimentos para realizar o Redeploy. Seguir o item 6 e printar cada passo no Terminal do Host

Gabarito



index.js

```
const express = require('express')
const app = express()

const port = process.env.PORT || 3030;

app.get('/', (req, res) => res.send('Implantação efetuada com sucesso'))

app.listen(port, (err) => {
   if (err) {
   console.log('Error::', err);
   }
   console.log(`App listening on port ${port}`);
});
```

Gabarito

package.json

```
{
  "name": "nodejs-hello",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.3"
  }
}
```

Gabarito

Dockerfile

```
# Define a imagem base
FROM node:lts-alpine3.19

# Define um usuário não privilegiado para executar a aplicação
USER node

# Define um diretório padrão para a aplicação
WORKDIR /app-money

ADD ./ .

# Exposição da porta para acesso à aplicação
EXPOSE 3030

# Define o comando padrão para iniciar a aplicação Node.js
CMD ["npm", "start"]
```

Gabarito

npm install (Empacotar o App)

```
# Na raiz do seu projeto
npm install
```

Imagem

```
docker build -t dimmoney-node  .
```

Container

```
docker container run -d --name dimmoney-nodePF0841 -p 3030:3030 -v /Users/Menk/nodejs-hello:/app-money dimmoney-node
```

Para Redeploy

```
docker rm -f dimmoney-nodePF0841
npm install
docker build -t dimmoney-node .
docker container run -d --name dimmoney-nodePF0841 -p 3030:3030 -v /Users/Menk/nodejs-hello:/app-money dimmoney-node
```

A melhor prática é incluir o comando "npm install" no Dockerfile

Projeto da Disciplina: Dimdim

Informações Gerais

- ✓ **Entregas fora do padrão** irão sofrer 25% de desconto na nota
- ✓ A documentação deve ser entregue com **Prints de qualidade**
- ✓ Subir o arquivo pdf no **Teams**. Somente o **representante** do grupo deve subir
- ✓ O número do **RM** a ser utilizado nos **exercícios** deve ser o do representante do grupo

Projeto da disciplina: Dimdim