Fundamentos de DevOps

ADO 02 – Imagem Docker Personalizada

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

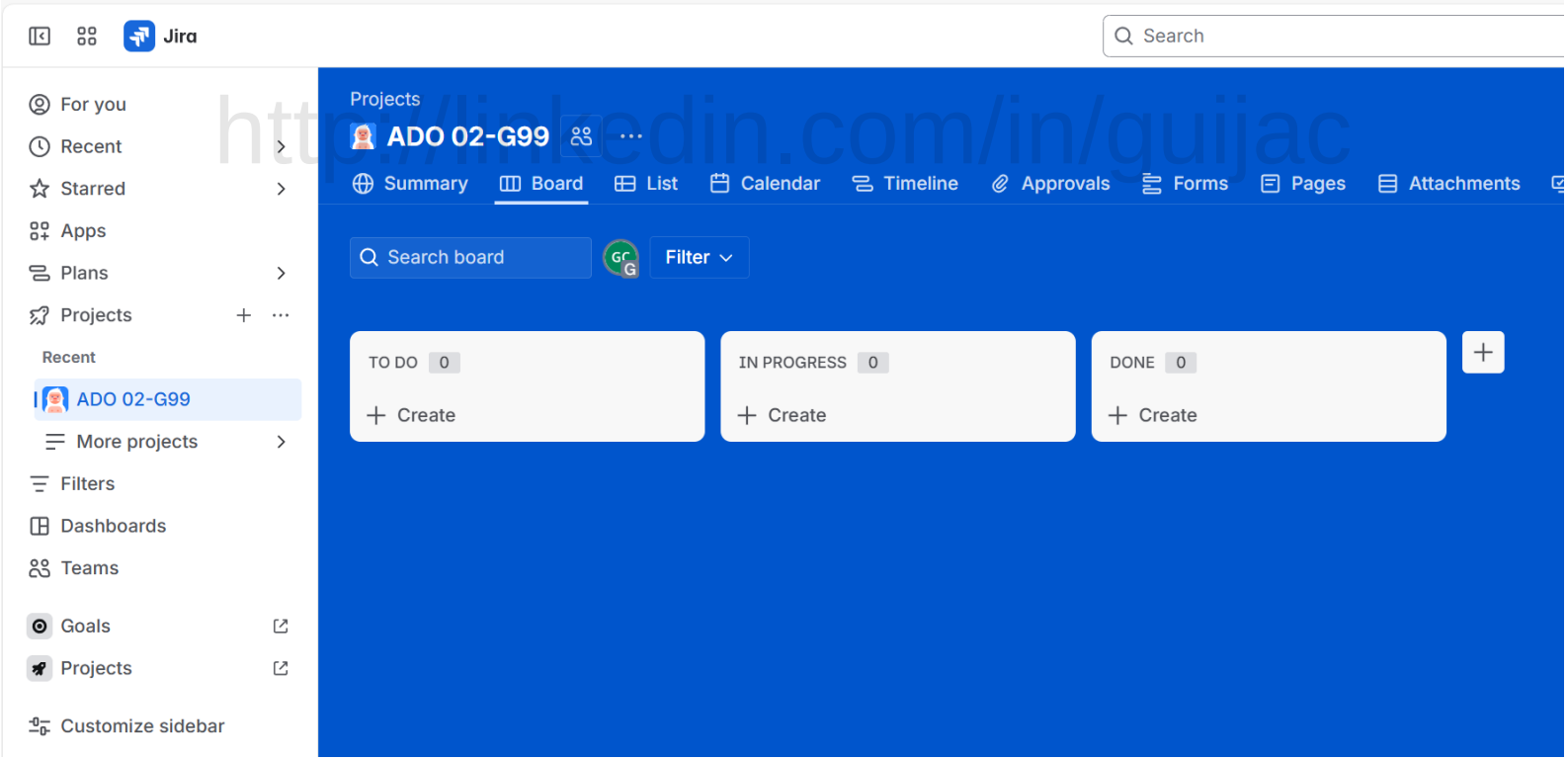
linkedin.com/in/guijac

ADO 02 – Imagem Docker Personalizada

- **Em grupos,** criar a sua própria imagem Docker (Dockerfile) para execução de uma aplicação Web Java + Spring Boot;
- Entregar o projeto no Gitlab (README.md, código-fonte e Dockerfile);
- Incluir meu usuário como "**developer**" em seu Gitlab e como "Administrator" em seu board do Jira (**guilherme.jacruz@sp.senac.br**);
- **Dinâmica sugerida**: dividir o grupo em desenvolvimento (Dev) e Operações (Ops), definir também uma pessoa para gestão das tarefas no Jira;
- Postar no Blackboard os *links* para acesso ao repositório e ao *board*;
- Critérios de Aceite:
  - Quadro Jira com controle e divisão das atividades executadas (1.0);
  - Documentação da aplicação no README.md (2.0);
  - Breve descrição da Imagem Base, com sua justificativa para uso (2.0);
  - Execução com sucesso do comando "docker build" (2.50);
  - Execução com sucesso do comando "docker run" (2.50);
- **Entrega até 21/03/2026 às 23h59;**
- Qualquer dúvida podem me chamar!

ADO 02 – Imagem Docker Personalizada

- Dentro do Jira, crie um novo projeto, para controle das atividades desta ADO;
- Nomeie este quadro com o seguinte critério:
  - ADO 02-G[Número do seu grupo].



ADO 02 – Imagem Docker Personalizada

- Documente a sua aplicação no arquivo “README.md”, é esperado que o documento possua a seguinte **estrutura mínima**:

````
# Nome do Projeto

Breve descrição da aplicação (o que ela faz, principais endpoints, etc).

## Imagem Base

Informe aqui qual imagem base Docker foi escolhida e **justifique** essa escolha
(compatibilidade, tamanho, segurança, etc).

## Build da Imagem Docker

```sh
$ docker build --tag <usuario>/<nome-da-imagem>:<versao> .

## Testando

```sh
$ docker run -d -p 8080:8080 <usuario>/<nome-da-imagem>:<versao>

Explique como acessar a aplicação (porta, endpoint, resposta esperada).
````

Template de arquivo README.md com estrutura mínima esperada.

ADO 02 – Imagem Docker Personalizada

- Crie um novo projeto em seu Gitlab e faça o clone do mesmo;
- Crie um arquivo chamado “Dockerfile” na raiz deste projeto e realize as adaptações conforme a necessidade da sua aplicação Java:
  - Por exemplo, pesquise uma imagem base no Docker Hub Container Image Library ou use uma IA Generativa de apoio.


```
# Especifica a imagem base a ser utilizada:
FROM <nome-da-imagem:<versão-da-imagem>
# Especifica um diretório de trabalho do contêiner, para execução dos comandos seguintes:
WORKDIR </nome-do-seu-diretório-de-trabalho>
# Permite a cópia de um arquivo do diretório local para o workdir do contêiner:
COPY <nome-do-arquivo-origem.txt> <nome-do-arquivo-destino.txt>
# Permite a execução de comandos de SO, como a geração de um JAR, através do Maven:
RUN <comando-mvn>
# Documenta a porta de exposição da aplicação (opcional, mas útil para documentar):
EXPOSE <porta-http>
# Instrução para a execução da aplicação do contêiner:
CMD ["<seu>", "<-comando>" , "<sua-aplicação.jar>"]
```

Exemplo parcial de um arquivo Dockerfile para execução de uma aplicação Java + Spring Boot

ADO 02 – Imagem Docker Personalizada

- Após a conclusão do arquivo Dockerfile, realize o seu “build” da imagem a partir do comando “docker build”, escolhendo um nome para sua imagem:

```
PS C:\Users\Guilherme\workspace\java-spring-docker> docker build --tag guijac/my-spring-app-docker .
ERROR: error during connect: this error may indicate that the docker daemon is not running: Get "http://%2F%2F.%2Fpipe%2Fdocker_engine/_ping": open
//./pipe/docker_engine: The system cannot find the file specified.
PS C:\Users\Guilherme\workspace\java-spring-docker> docker build --tag seu-usuario/my-spring-app-docker .
[+] Building 43.4s (10/10) FINISHED
docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                                    0.0s
 => => transferring dockerfile: 569B                                                                                                    0.0s
 => [internal] load metadata for docker.io/library/maven:3.9.6-eclipse-temurin-21                                                       2.4s
 => [auth] library/maven:pull token for registry-1.docker.io                                                                            0.0s
 => [internal] load .dockerignore                                                                                                       0.0s
 => => transferring context: 2B                                                                                                         0.0s
 => [1/4] FROM docker.io/library/maven:3.9.6-eclipse-temurin-21@sha256:8d63d4c1902cb12d9e79a70671b18ebe26358cb592561af33ca1808f00d935cb  24.3s
 => [3/4] COPY . .                                                                                                                      0.1s
 => [4/4] RUN mvn package -DskipTests                                                                                                  15.0s
 => exporting to image                                                                                                                  0.9s
 => => exporting layers                                                                                                                 0.9s
 => => writing image sha256:1c743fa7ae0a4dd5c4d154411d851cc7856010710340dff24f35a3c397aa3d3b                                            0.0s
 => => naming to docker.io/seu-usuario/my-spring-app-docker                                                                             0.0s
What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

Exemplo de saída do console após a execução do comando “docker build --tag guijac/my-spring-app-docker .

ADO 02 – Imagem Docker Personalizada

- Certifique-se que a imagem foi gerada corretamente através do comando “docker images”:

```
PS C:\Users\Guilherme\workspace\java-spring-docker> docker images
REPOSITORY                      TAG           IMAGE ID           CREATED           SIZE
guijac/my-spring-app-docker     latest        1c743fa7ae0a       47 hours ago      597MB
```


- Execute um contêiner a partir da imagem criada com o comando “docker run” (neste exemplo, por ser uma aplicação Java Spring Boot é necessário o mapeamento da porta 8000):

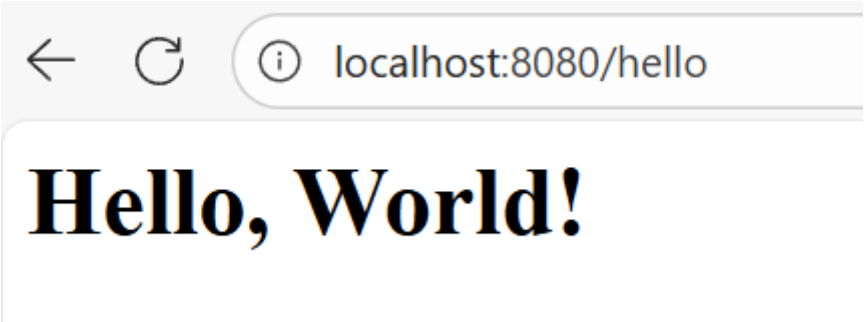
```
PS C:\Users\Guilherme\workspace\flask-docker> docker run -d -p 8000:8000 my-spring-app-docker
4f3c157818633d79ea3860d9155c3ed2cca3d3467486bc210230cb6eeba7c308
```

ADO 02 – Imagem Docker Personalizada

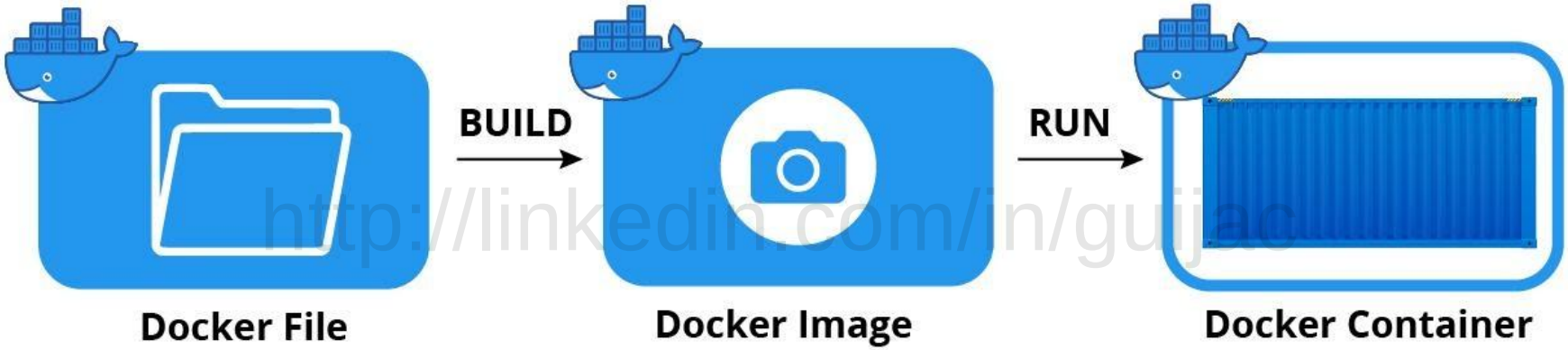
- Certifique-se que o contêiner está em execução, através do comando “docker ps”:

```
PS C:\Users\Guilherme\workspace\java-spring-docker> docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS                                       NAMES
52460f5a60bd   guijac/my-spring-app-docker     "/usr/local/bin/mvn-…"   5 seconds ago   Up 3 seconds   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   determined_joliot
```

- Teste o acesso da aplicação (neste exemplo, através da porta 8000):



ADO entregue? Bora explorar os laboratórios pendentes!



Fonte: Understanding and Building Docker Images (jfrog.com)

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.jacruz@sp.senac.br

linkedin.com/in/guijac

Licença

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac