

# Checkpoint DevOPS 2TDSPX

---

25 DE ABRIL

---

2TDSPX



---

# Integrantes

Beatriz Lucas - RM99104

Enzo Farias - RM98792

Ewerton Gonçalves - RM98571

Guilherme Tantulli - RM97890

Thiago Zupelli - RM99085

## GITHUB

**Java** - <https://github.com/tomgoncalvs/CP2-DevOPS/tree/main/Java>

**Python** - <https://github.com/tomgoncalvs/CP2-DevOPS/tree/main/Python>

**Node** - <https://github.com/tomgoncalvs/CP2-DevOPS/tree/main/Node>

---

# Java

**Construção da Imagem Docker:** Mostra a saída do terminal PowerShell após a execução do comando `docker build`. Este comando constrói uma imagem Docker para a aplicação Java, usando o Dockerfile fornecido. A saída mostra que a construção da imagem foi concluída com sucesso, indicada pela mensagem "Building 182.1s (9/9) FINISHED".

**Verificação da Imagem Docker:** Interface do Docker Desktop, mostrando que a imagem Docker `dimmoney-app` foi criada com sucesso. A imagem está listada como "In use", indicando que a imagem está pronta e disponível para ser utilizada.

**Conteúdo do Dockerfile:** Mostra o conteúdo do Dockerfile que foi usado para construir a imagem Docker. O arquivo inclui instruções para usar a imagem base do Tomcat 10, definir o diretório de trabalho, copiar o arquivo `.war` da aplicação, expor a porta 8080 e criar um volume apontando para o diretório de implantação do Tomcat.

**Implantação da Aplicação:** Exibe a aplicação sendo acessada por um navegador. A URL `localhost:8080/DimMoneyApp98571/` mostra a mensagem "Deploy efetuado com sucesso no Servidor Tomcat 10", confirmando que a aplicação Java foi implantada corretamente no servidor Tomcat via Docker e está servindo conteúdo conforme esperado.

**Dashboard do Docker Desktop:** Mostra a interface do Docker Desktop, mas desta vez com foco na seção "Images", onde a imagem `dimmoney-app` está destacada. Esta captura de tela serve como confirmação adicional de que a imagem Docker foi criada corretamente.

```

PS C:\Users\ewert\Desktop\Java> docker build -t dimmoney-app .
[+] Building 182.1s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.0s
=> => transferring dockerfile: 200B                                0.0s
=> [internal] load metadata for docker.io/library/tomcat:10       1.3s
=> [auth] library/tomcat:pull token for registry-1.docker.io      0.0s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 2B                                     0.0s
=> [1/3] FROM docker.io/library/tomcat:10@sha256:a768924119030b778276c46dc 46.8s
=> => resolve docker.io/library/tomcat:10@sha256:a768924119030b778276c46dc1 0.0s
=> => sha256:a768924119030b778276c46dc19293cc8e4dea63b6b1dd8c 979B / 979B 0.0s
=> => sha256:372381e3984a63c6df415bc3753ccf69f114edbcfdd4b4 2.00kB / 2.00kB 0.0s
=> => sha256:47f246a79344d0d339117f31ee7499efab2924a426fd 13.12kB / 13.12kB 0.0s
=> => sha256:7021d1b70935851c95c45ed18156980b5024eda29b99 30.44MB / 30.44MB 6.6s
=> => sha256:69b6c8970fafbd67b6e2704d6584bdfc6e2275755 158.51MB / 158.51MB 20.3s
=> => sha256:0f3320e4e2ae41973a279c466eba6f7af49750691a7d 17.46MB / 17.46MB 4.8s
=> => sha256:a95d7ac9605410015e4da369ce7b01bcee8f2ec9786143fb07 178B / 178B 5.0s
=> => sha256:bb7721b0553ddeb086a53f4c6198ae14407ecc6c369e1005cb 734B / 734B 5.2s
=> => sha256:2f18316bc2d97771c882ce315152626ea9d802d4b7f50d442b 172B / 172B 5.5s
=> => sha256:2eb8dea168744a559a05ac1dc4b571d56f2f83094388 18.72MB / 18.72MB 8.5s
=> => sha256:5cd2484b02ea15dc614a5ebe86a3c560e8dadb7aa9dc9aed11 131B / 131B 6.8s
=> => extracting sha256:7021d1b70935851c95c45ed18156980b5024eda29b995644290 7.5s
=> => extracting sha256:0f3320e4e2ae41973a279c466eba6f7af49750691a7d55baabd 5.3s
=> => extracting sha256:69b6c8970fafbd67b6e2704d6584bdfc6e2275755afdc04802a 7.3s
=> => extracting sha256:a95d7ac9605410015e4da369ce7b01bcee8f2ec9786143fb078 0.0s
=> => extracting sha256:bb7721b0553ddeb086a53f4c6198ae14407ecc6c369e1005cbf 0.0s
=> => extracting sha256:2f18316bc2d97771c882ce315152626ea9d802d4b7f50d442bd 0.0s
=> => extracting sha256:2eb8dea168744a559a05ac1dc4b571d56f2f83094388092b4cd 2.0s
=> => extracting sha256:5cd2484b02ea15dc614a5ebe86a3c560e8dadb7aa9dc9aed115 0.0s
=> [internal] load build context                                  2.5s
=> => transferring context: 19.77MB                                2.5s
=> [2/3] WORKDIR /usr/local/tomcat/webapps                      118.5s
=> exporting to image                                           2.6s
=> => writing image sha256:4e64fafa86013dba1e3ae5524814e3b6319f86e24f013783 0.1s
=> => naming to docker.io/library/dimmoney-app                    0.1s

```

View build details: [docker-desktop://dashboard/build/default/default/szv9takp3vhy2ldv6508f61rz](https://docker-desktop://dashboard/build/default/default/szv9takp3vhy2ldv6508f61rz)

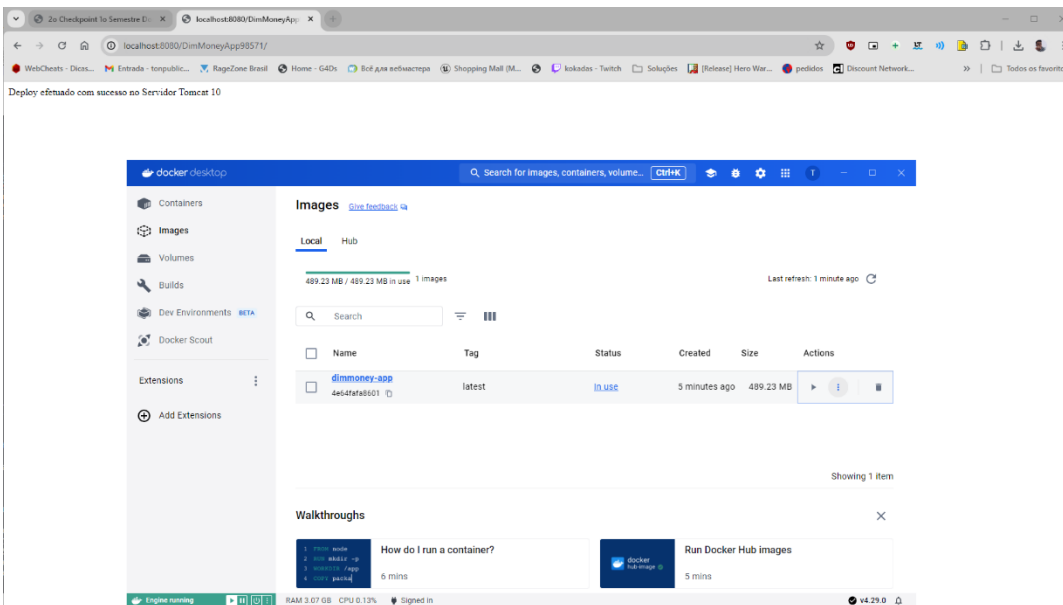
### What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

```

PS C:\Users\ewert\Desktop\Java> docker run -d --name dimmoney98571 -p 8080:8080 -v deploy-s-tomcat10:/usr/local/tomcat/webapps dimmoney-app
c43d22e19c29f256e6483a13b906684ac7e710bdbad7745de97895e4a141c4fc

```



SZV9TA

Java

desktop-linux

✓ Completed

2m 54s

43 minutes ago

N/A

Dockerfile > ...

```
1 FROM tomcat:10
2 WORKDIR /usr/local/tomcat/webapps
3 COPY ./target/DimMoneyApp98571.war /usr/local/tomcat/webapps/
4 EXPOSE 8080
5 VOLUME /usr/local/tomcat/webapps
6
```

localhost:8080/DimMoneyApp98571/

WebCheats - Dicas... Entrada - tonpublic... RageZone Brasil Home - G4Ds Boi da eefwactrepa Shopping Mall (M... kokadas - Twitch Soluções [Release] Hero War... pedidos Discount Network... Todos os favoritos

Deploy efetuado com sucesso no Servidor Tomcat 10

File Edit Selection View ... Java

EXPLORER

- JAVA
  - .mvn
  - .vscode
  - src
  - target
    - classes
    - generated-sources
    - generated-test-sources
    - java-0.0.1-SNAPSHOT
    - maven-archiver
    - maven-reports
    - maven-status
    - surefire-reports
    - test-classes
  - DimMoneyApp98571.war
  - java-0.0.1-SNAPSHOT.war
  - java-0.0.1-SNAPSHOT.war.ori...
  - .gitignore
  - Dockerfile
  - HELP.md
  - mvnw
  - mvnw.cmd
- > OUTLINE
- > TIMELINE
- > JAVA PROJECTS
- > MAVEN

Dockerfile

```
1 FROM tomcat:10
2 WORKDIR /usr/local/tomcat/webapps
3 COPY ./target/DimMoneyApp98571.war /usr/local/tomcat/webapps/
4 EXPOSE 8080
5 VOLUME /usr/local/tomcat/webapps
6
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powerShell

PS C:\Users\ewert\Desktop\Java> docker volume inspect deploys-tomcat10

[{"CreatedAt": "2024-04-25T00:40:30Z", "Driver": "local", "Labels": null, "Mountpoint": "/var/lib/docker/volumes/deploys-tomcat10/\_data", "Name": "deploys-tomcat10", "Options": null, "Scope": "local"}]

PS C:\Users\ewert\Desktop\Java> docker cp ./target/DimMoneyApp98571.war dimmoney98571:/usr/local/tomcat/webapps/

>>

Successfully copied 19.8MB to dimmoney98571:/usr/local/tomcat/webapps/

PS C:\Users\ewert\Desktop\Java> docker start dimmoney98571

dimmoney98571

---

# Python

**Criação do Dockerfile:** O Dockerfile define o ambiente necessário para executar nossa aplicação Python. Usando python:3.9-slim como base, estabelece /app como diretório de trabalho. Comandos subsequentes copiam o arquivo Python especificado para dentro do container e configuram o ambiente para executar a aplicação.

**Build da Imagem:** O comando docker build foi utilizado para construir a imagem Docker da aplicação Python, passando o nome do arquivo Python como argumento. O processo de build foi completado com sucesso, conforme indicado pela mensagem "Building 122.1s (9/9) FINISHED".

**Verificação da Imagem:** O print do Docker Desktop confirma que a imagem dimmoney-python foi construída com sucesso, como indicado pelo status 'Completed'. Isso verifica que a imagem está agora armazenada localmente e pronta para uso.

**Execução do Container:** O comando docker run foi empregado para executar a aplicação, onde o container Docker foi iniciado e, em seguida, automaticamente removido após a execução, graças à flag --rm. A saída do terminal exibe a mensagem esperada: "Implantação efetuada com sucesso".

**Confirmação da Execução:** Embora não haja um print do navegador para a aplicação Python, pois ela é executada no terminal, o texto no terminal do PowerShell confirma que a aplicação Python rodou com sucesso e exibiu a mensagem de implantação efetuada com sucesso, indicando que a aplicação foi executada como esperado.



```

PS C:\Users\ewert\Desktop\Python> docker build --build-arg APP_FILE=app98571 -t dim
money-python .
[+] Building 122.1s (9/9) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.2s
=> => transferring dockerfile: 181B                                0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim 2.2s
=> [auth] library/python:pull token for registry-1.docker.io      0.0s
=> [internal] load .dockerignore                                   0.1s
=> => transferring context: 2B                                      0.0s
=> [1/3] FROM docker.io/library/python:3.9-slim@sha256:44122e46edb1c3ae2a1 14.8s
=> => resolve docker.io/library/python:3.9-slim@sha256:44122e46edb1c3ae2a14 0.0s
=> => sha256:2e1f1541f093ecfd3bda3388cedeb5034e73cfe0d36a12 1.37kB / 1.37kB 0.0s
=> => sha256:a8260aee86e9399429c23095da23d7ab308426dc30e31 6.90kB / 6.90kB 0.0s
=> => sha256:b0a0cf830b12453b7e15359a804215a7bcccd3788e2b 29.15MB / 29.15MB 4.7s
=> => sha256:72914424168c8ebb0dbb3d0e08eb1d3b5b2a64cc51745b 3.51MB / 3.51MB 1.2s
=> => sha256:19611b60238f488b83fb4229632e28130ae9d76647eb 11.89MB / 11.89MB 3.4s
=> => sha256:44122e46edb1c3ae2a144778db3e01c78b6de3af20ddcc 1.86kB / 1.86kB 0.0s
=> => sha256:a1e1c999a12c5f9fef531eb908858fb2938f86a9cd98830bb8 243B / 243B 1.6s
=> => sha256:3d5cbb4d884e7912ff3bbbc08931cca207cebcb0daa15 3.13MB / 3.13MB 2.6s
=> => extracting sha256:b0a0cf830b12453b7e15359a804215a7bcccd3788e2bcecff2a 5.6s
=> => extracting sha256:72914424168c8ebb0dbb3d0e08eb1d3b5b2a64cc51745bd65ca 0.6s
=> => extracting sha256:19611b60238f488b83fb4229632e28130ae9d76647ebdc57f78 2.0s
=> => extracting sha256:a1e1c999a12c5f9fef531eb908858fb2938f86a9cd98830bb8a 0.0s
=> => extracting sha256:3d5cbb4d884e7912ff3bbbc08931cca207cebcb0daa15954c4 1.1s
=> [internal] load build context                                  0.1s
=> => transferring context: 83B                                      0.0s
=> [2/3] WORKDIR /app                                           104.3s
=> [3/3] COPY app98571.py .                                     0.1s
=> exporting to image                                           0.2s
=> => exporting layers                                           0.1s
=> => writing image sha256:875e0e54b5117b659259bbdad9cc0275e901f2680a9f6082 0.0s
=> => naming to docker.io/library/dimmoney-python              0.0s

```

View build details: [docker-desktop://dashboard/build/default/default/wg04aj4tyyrl1a1cluj00909p](#)

#### What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

```
PS C:\Users\ewert\Desktop\Python> 
```

#### What's Next?

View a summary of image vulnerabilities and recommendations → [docker scout quickview](#)

```
PS C:\Users\ewert\Desktop\Python> docker run --rm dimmoney-python
```

Implantação efetuada com sucesso

```
PS C:\Users\ewert\Desktop\Python> 
```

<input type="checkbox"/>	ID	Name	Builder	Status	Duration	↓ Created	Author
<input type="checkbox"/>	WG04AJ	<a href="#">Python</a>	 <a href="#">desktop-linux</a>	✓ Completed	2m 02s	4 minutes ago	N/A

```

🐳 Dockerfile > ...
1 FROM python:3.9-slim
2 WORKDIR /app
3 ARG APP_FILE
4 COPY ${APP_FILE}.py .
5 ENV APP_FILE=${APP_FILE}
6 CMD ["sh", "-c", "python ${APP_FILE}.py"]
7

```

---

# NODEJS

**Criação Docker File:** O Dockerfile foi configurado para criar um ambiente otimizado para a aplicação Node.js. A base escolhida foi a `node:lts-alpine3.19`, e o diretório de trabalho foi estabelecido como `/app-money`. A imagem, nomeada `dimmoney-node`, foi construída com sucesso, conforme indicado pela conclusão do processo de build no terminal PowerShell e verificação no Dashboard do Docker Desktop.

**Execução do Container:** O container `dimmoney-node98571` foi executado corretamente em segundo plano, com a porta 3000 exposta e mapeada, e o diretório atual montado como um volume dentro do container.

**Verificação da Aplicação em Execução:** A aplicação foi acessada através do navegador, onde a mensagem "Implantação efetuada com sucesso" foi exibida, confirmando que a aplicação Node.js estava rodando corretamente e acessível pela porta mapeada. O Dashboard do Docker Desktop também mostrou que a imagem `dimmoney-node` estava ativa e em uso, indicando que o container foi iniciado e está rodando conforme esperado.

**Redeploy da Aplicação:** Após uma alteração no código-fonte, um redeploy foi realizado com sucesso. A aplicação foi reconstruída, a imagem do Docker foi atualizada, e o container foi reiniciado. A nova mensagem "Implantação efetuada com sucesso. Bom trabalho!" foi validada no navegador, demonstrando que o processo de atualização da aplicação ocorreu sem problemas.

**Conclusão e Clean-Up:** Com a validação da execução da aplicação após o redeploy, o projeto Node.js pode ser considerado plenamente funcional dentro de um ambiente Dockerizado. O processo de clean-up não é documentado nos prints, mas é uma etapa importante para manter um ambiente de desenvolvimento limpo e organizado.





```
* History restored

>>
[+] Building 26.1s (12/12) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 263B
=> [internal] load metadata for docker.io/library/node:lts-alpine3.19
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/6] FROM docker.io/library/node:lts-alpine3.19@sha256:7a91aa397f2e2dfbfcad2e2d72599f374e0b0172be1d86eeb73fd33f36a4b2
=> [internal] load build context
=> => transferring context: 34.33kB
=> CACHED [3/6] COPY --chown=node:node package*.json ./
=> CACHED [4/6] RUN chown -R node:node /app-money
=> CACHED [5/6] RUN npm install
=> [6/6] COPY --chown=node:node . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:aa4da331c04c05842a2c2730d38aad0f0a407f010ccbef6abf01b099c7df490f
=> => naming to docker.io/library/dimmoney-node

View build details: docker-desktop://dashboard/build/default/default/i0hizl3o1txkvtrovy5rj8cdg

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\ewert\Desktop\nodejs> docker run -d -p 3000:3000 --name dimmoney-node98571 -v ${PWD}:/app-money dimmoney-node
docker: Error response from daemon: Conflict. The container name "/dimmoney-node98571" is already in use by container "88d32aa63eca6035e4a1cbdf187cae1b4b532e8a5ea93b7e6cc01f28982de7ea". You have to remove (or rename) that container to be able to reuse that name.
See 'docker run --help'.
PS C:\Users\ewert\Desktop\nodejs> docker run -d -p 3000:3000 --name dimmoney-node98571 -v ${PWD}:/app-money dimmoney-node
5436572f6c0882eaa14e5f5f7ad86f024bc7890b2212863785ad2c6cb9a8216f
PS C:\Users\ewert\Desktop\nodejs>
```

