Desenvolva usando Docker

Quem somos?

- Robinho, vulgo Robson Peixoto
- Gomex, vulgo Rafael Gomes

O que é Docker?

- Padroniza a entrega de software através das imagens
- Padroniza a execução de software através dos containers
- Funciona em Windows, Linux e Mac
- Permite que tudo seja facilmente versionável

Instalando Docker

- Docker Nativo
- Docker for Windows ou Mac

Criando uma Docker ID

- https://hub.docker.com/

Play with Docker

- https://labs.play-with-docker.com/
- http://training.play-with-docker.com/

Hello world

```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
ca4f61b1923c: Pull complete
Digest: sha256:be0cd392e45be79ffeffa6b05338b98ebb16c87b255f48e297ec7f98e123905c
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

...
```

```
$ docker run -it ubuntu bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
660c48dd555d: Pull complete
4c7380416e78: Pull complete
421e436b5f80: Pull complete
e4ce6c3651b3: Pull complete
be588e74bd34: Pull complete
Digest: sha256:7c67a2206d3c04703e5c23518707bdd4916c057562dd51c74b99b2ba26af0f79
Status: Downloaded newer image for ubuntu:latest

root@da72ee310354:/# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  18240  3272 pts/0    Ss   21:17   0:00 bash
root         9  0.0  0.0  34424  2920 pts/0    R+   21:20   0:00 ps aux

root@da72ee310354:/# exit
exit
```

Primeira Imagem

Crie o arquivo `Dockerfile` com o conteúdo:

`Dockerfile`

```
FROM ubuntu
CMD ["printf", "FÓRUM BAIANO DE TECNOLOGIAS ABERTAS\n"]
```

E execute o comando:

```
$ docker build -t  uefs/fbta:001 .
```

E agora vamos criar um container

```
$ docker run  uefs/fbta:001
FÓRUM BAIANO DE TECNOLOGIAS ABERTAS
```

Cadê esse container?

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
```

Morreu?

```
$ docker ps -a
CONTAINER ID  IMAGE          COMMAND                  CREATED        STATUS                      PORTS  NAMES
096e732d91b2  uefs/fbta:001 "printf 'FÓRUM BAI..."  6 seconds ago  Exited (0) 4 seconds ago           modest_golick
```

Loop infinito!

Vamos criar o script `loop.sh`

```
while true ; do
    AGORA="$(date)"
    echo "${AGORA} => TOU VIVO"
    sleep 1s
done
```

Loop infinito - Dockerfile

```
FROM ubuntu

WORKDIR /app
COPY loop.sh .

CMD ["sh", "loop.sh"]
```

Loop infinito - criando a imagem

```
$ docker build -t  uefs/fbta:002 .
Sending build context to Docker daemon  3.072kB
Step 1/4 : FROM ubuntu
 ---> 20c44cd7596f
Step 2/4 : WORKDIR /app
 ---> f516fc326ed1
Removing intermediate container 4fa1f011c4be
Step 3/4 : COPY loop.sh .
 ---> b1c2579d9689
Step 4/4 : CMD sh loop.sh
 ---> Running in f7f6cff8695d
 ---> 894ace62e2eb
Removing intermediate container f7f6cff8695d
Successfully built 894ace62e2eb
Successfully tagged  uefs/fbta:002
```

Loop infinito - rodando...

```
$ docker run  uefs/fbta:002
Mon Nov 27 02:21:56 UTC 2017 => TOU VIVO
Mon Nov 27 02:21:57 UTC 2017 => TOU VIVO
Mon Nov 27 02:21:58 UTC 2017 => TOU VIVO
Mon Nov 27 02:21:59 UTC 2017 => TOU VIVO
Mon Nov 27 02:22:00 UTC 2017 => TOU VIVO
...
```

Loop infinito - em background ...

```
$ docker run -d  uefs/fbta:002
44d1c36f7faa2ade4317430bad0ae544ad20248ffc2a57f8797cb80a1a4aa6ea

$ docker ps
CONTAINER ID   IMAGE                    COMMAND             CREATED              STATUS
44d1c36f7faa    uefs/fbta:002   "sh loop.sh"        18 seconds ago      Up 16 seconds

$ docker logs -f --tail 2 44d1c36f7faa
Mon Nov 27 02:23:44 UTC 2017 => TOU VIVO
Mon Nov 27 02:23:45 UTC 2017 => TOU VIVO
Mon Nov 27 02:23:46 UTC 2017 => TOU VIVO
...
```

Loop infinito - mata logo!

```
$ docker kill 44d1c36f7faa
44d1c36f7faa

$ docker ps -a
CONTAINER ID  IMAGE          COMMAND                  CREATED          STATUS                          PORTS   NAMES
4a53c0deae0b  uefs/fbta:002  "sh loop.sh"             9 seconds ago    Exited (137) 2 seconds ago              nostalgic_mirzakhani
096e732d91b2  uefs/fbta:001  "printf 'FÓRUM BAI..."   15 seconds ago   Exited (0) 13 seconds ago               modest_golick
```

Limpando a casa

```
$ docker rm 4a53c0deae0b
$ docker rm 096e732d91b2
```

Me mostra alguma coisa útil ...

Agora vamos criar uma simples aplicação em Python que:

- cria e lista usuários
- manda email quando o usuário é criado

Python 101

```
$ docker run -it python:3
Python 3.6.3 (default, Nov  4 2017, 22:17:09)
[GCC 4.9.2] on linux
Type "help", "copyright", "credits" or "license" for more information.

>>> print('oi')
oi

>>> print("oi")
oi
```

Python 101 - parte 2

```
>>> def soma(x, y):
...     return x + y
...

>>> soma(1,2)
3
```

Python 101 - parte 3

```
>>> lista = [1,2,3,4]
>>> for item in lista:
...     print(item, item + 1)
...
1 2
2 3
3 4
4 5
```

Python 101 - parte 4

```
>>> i = 0
>>> while i < 5:
...     print(i)
...     i = i + 1
...
0
1
2
3
4
```

Python 101 - parte 5

```
>>> import datetime

>>> datetime.datetime.now()
datetime.datetime(2017, 11, 27, 3, 11, 58, 637834)

>>> str(datetime.datetime.now())
'2017-11-27 03:12:03.517113'
```

Desafio 1

Fazer o hello world em Python

Python - Hello world

Crie o arquivo `hello.py`

```
print("FÓRUM BAIANO DE TECNOLOGIAS ABERTAS")
```

Crie o arquivo `Dockerfile`

```
FROM python:3
WORKDIR /app
COPY hello.py .
CMD ["python3", "hello.py"]
```

Desafio 2

Fazer o loop infinito em Python

Loop infinito em Python

Crie o arquivo `loop.py`

```
import datetime
import time

while True:
    print(datetime.datetime.now(), "TOU VIVO")
    time.sleep(1)
```

Loop infinito em Python - parte 2

Crie o arquivo `Dockerfile`

```
FROM python:3
WORKDIR /app
COPY loop.py .
CMD ["python3", "loop.py"]
```

Conhecendo o Flask

Crie o arquivo `requirements.txt`

```
flask
```

Crie `api.py`

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

Desafio 3

Rodar essa aplicação escrita em Flask

Conhecendo o Flask - parte 2

Crie o `Dockerfile`

```
FROM python:3
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "api.py"]
```

Depois execute `docker build -t uefs/fbta:005 .` para criar a imagem

Desafio 3.1

Acessar a aplicação

Conhecendo o Flask - parte 3

```
$ docker run -d -p 5000:5000 uefs/fbta:005
b690a907759db366ab9b8745830d84ced35e01f071fd90bddd6259d7a8a07c45

$ curl localhost:5000
Hello World!

$ docker ps
CONTAINER ID  IMAGE           COMMAND           PORTS                     NAMES
efaadfad53bc  uefs/fbta:005   "python api.py"   0.0.0.0:5000->5000/tcp    jolly_pasteur

$ docker rm --force b690a907759db366ab9b8745830d84ced35e01f071fd90bddd6259d7a8a07c45
b690a907759db366ab9b8745830d84ced35e01f071fd90bddd6259d7a8a07c45
```

Desafio 3.2

Mude a mensagem e veja a nova mensagem

O que aconteceu?

Desafio 3.3

Deixe o build mais rápido

Deixe o build mais rápido

Mude o `Dockerfile` para:

```
FROM python:3
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "api.py"]
```

Rode o build(`docker build -t uefs/fbta:006 .`) e veja o que acontece

Docker cache

```
$ docker build -t uefs/fbta:006 .
Sending build context to Docker daemon  4.096kB
Step 1/6 : FROM python:3
 ---> 79e1dc9af1c1
Step 2/6 : WORKDIR /app
 ---> Using cache
 ---> 8ae6ae199c9c
Step 3/6 : COPY requirements.txt .
 ---> Using cache
 ---> 7c35f7de84d3
Step 4/6 : RUN pip install -r requirements.txt
 ---> Using cache
 ---> 311e80a05ee3
Step 5/6 : COPY . .
 ---> 0c012bdbd591
Step 6/6 : CMD python api.py
 ---> Running in 06ffb420c1f8
 ---> 4a1d298cbdaf
Removing intermediate container 06ffb420c1f8
Successfully built 4a1d298cbdaf
Successfully tagged uefs/fbta:006
```

Desafio 4

Queremos uma API REST, retorne um JSON

Mude a `api.py` para:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000)
```

Isso já tá ficando chato ...

Docker Compose

Crie o arquivo `docker-compose.yml`

```
version: '3'
services:
  api:
    build: .
    ports:
    - "5000:5000"
```

E rode o comando `docker-compose up --build`.

Desafio 5

Mude a mensagem que está no JSON

Não tem como deixar menos chato?

Depente!!!

Volumes!

Modifique o arquivo `docker-compose.yml`

```
version: '3'
services:
  api:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - ".:/app"
```

Modifique o `app.run` do `api.py` para:

```
app.run(host='0.0.0.0', port=5000, debug=True)
```

Desafio 6

Modifique o código e veja como o Docker é lindo

API Rest - Cadastrando livros

- O que precisamos para cadastrar um livro?
- Como criar uma simples API Rest?
- Como levantar um banco de dados ?

Desafio 7

Suba um mongodb usando o docker compose

Desafio 8

Se conecte ao banco mongo

Desafio 9

Repita, bem rápido, 10 vezes a frase abaixo:

O padre pouca capa tem, porque pouca capa compra.

Cadastrando um novo livro

Adicione o mongo no `requirements.txt`

```
flask
Flask-PyMongo
```

Cadastrando um novo livro - parte 2

E o mongo no `docker-compose.yml`

```
version: '3'
services:
  api:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - ".:/app"
  mongo:
    image: mongo
```

Cadastrando um novo livro - parte 3

Modifique o código do `api.py` para

```
from flask import Flask, jsonify, request
from flask_pymongo import PyMongo

app = Flask(__name__)
app.config['MONGO_HOST'] = 'mongo'
app.config['MONGO_DBNAME'] = 'fbta'
mongo = PyMongo(app)

@app.route("/", methods=['POST'])
def cadastrar():
    livro = request.json
    resultado = mongo.db.livros.insert_one(livro)
    livro['id'] = str(livro['_id'])
    del livro['_id']
    return jsonify(livro)

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Cadastrando um novo livro - parte 4

Agora teste rodando o comando abaixo em outro terminal:

```
curl 'http://localhost:5000' \
  -H 'Content-Type: application/json' \
  -X POST \
  -d '{"titulo": "docker para desenvolvedores", "autor": "Rafael Gomes"}'
```

Desafio 10

Recupere o livro quando o usuário acessar a URL `/<codigo-do-livro>`

Recuperando o livro

Adicione ao `api.py` a função abaixo:

```
from flask_pymongo import PyMongo, ObjectId

@app.route("/<codigo>", methods=['GET'])
def recuperar(codigo):
    livro = mongo.db.livros.find_one({'_id': ObjectId(codigo)})
    livro['id'] = str(livro['_id'])
    del livro['_id']
    return jsonify(livro)
```

E test com o comando `curl localhost:5000/5a1cc89736e48d005e4626e1`

Desafio 11

Crie uma simples rota `/sorteio` para sortear o livro Docker para Desenvolvedores

Sorteio

Adicione ao `api.py` a função abaixo:

```
import random

@app.route("/sorteio")
def sorteio():
    alunos = ['A', 'B', 'C']
    escolhido = random.choice(alunos)
    return jsonify(escolhido)
```

Perguntas?