

BEATRIZ COELHO VIEIRA

MAT: 212454015 | POLO: ROLANDIA/PR

LARYSSA STEPHANIE ANDRADE DA COSTA SILVA

MAT: 211454050 | POLO: LONDRINA/PR

MURILLO AUGUSTO FERMINO

MAT: 212454014 | POLO: ROLANDIA/PR

AVA3 – COMPUTAÇÃO EM NUVEM
STARTUP 09 – APLICATIVO FIXIN

SUMÁRIO

| | |
|--|----|
| SUMÁRIO | 2 |
| 1 INTRODUÇÃO..... | 3 |
| 2 OBJETIVOS | 5 |
| 2.1 Descrição da atividade | 5 |
| 2.2 Proposta | 5 |
| 3 DESENVOLVIMENTO DA API | 6 |
| 3.1 Dependências..... | 6 |
| 3.2 Criação da aplicação..... | 6 |
| 3.3 Definindo os <i>endpoints</i> do <i>REST</i> | 8 |
| 3.4 Criando o <i>Dockerfile</i> | 9 |
| 3.5 Criando o Docker Compose..... | 10 |
| 4 IMPLANTAÇÃO DA API NA AWS | 12 |
| 5 REFERÊNCIAS | 24 |

1 INTRODUÇÃO

Muitas vezes, quando desenvolvemos uma solução tecnológica, estamos interessados em possibilitar que outros desenvolvedores manipulem as funções de nosso sistema programaticamente. Para isso, podemos disponibilizar o acesso ao nosso sistema por meio de uma API REST. Nesse caso, podemos realizar operações com GET e POST via requisição HTTP ou HTTPS para executar funções em um sistema remotamente.

Antes de entendermos a definição de API Rest, é preciso, primeiro, entender o que é API. O termo API é um acrônimo para *Application Programming Interface*, ou, em português, Interface de Programação de Aplicativos. De forma resumida, a API é um conjunto de rotinas e requisições que possibilita a comunicação e troca de dados entre aplicações diferentes. O objetivo da interface é proporcionar um desenvolvimento consistente e implementação de aplicações e serviços. Através dela, os desenvolvedores têm a garantia de um sistema imutável para integrar sua aplicação a um serviço terceiro.

Já o termo Rest significa *Representational State Transfer* que, em português, significa Transferência de Estado Representacional. Diferente da API – que pode ser definida como um conjunto de padrões –, o Rest é um conjunto de restrições e princípios de arquitetura, utilizadas para que as requisições HTTP atendam às normas previamente definidas.

E essa arquitetura possui alguns princípios, como:

- **Cliente-servidor:** garante que as responsabilidades das aplicações existentes no servidor e no usuário devem ser mantidas separadas;
- **Stateless:** toda requisição é entendida como uma transação independente e sem qualquer relação a requisições anteriores;
- **Cache:** melhora a comunicação entre diferentes aplicações, o que torna a comunicação cliente-servidor mais ágil;
- **Interface uniforme:** o cliente e o servidor compartilham a mesma interface. Dessa forma, é preciso criar um contrato para que essa comunicação interoperacional aconteça de forma segura.

Dito isso, a API Rest é, sem dúvidas, crucial para o desenvolvimento de qualquer aplicação web. Afinal, é ela que estabelece a comunicação entre diferentes aplicações e possibilita a troca de informações de forma rápida e segura.

Logo, ela pode ser utilizada de diferentes formas. As redes sociais, por exemplo, fornecem APIs que possibilitam o acesso às informações de uma outra página, sem a necessidade de sair da URL original.

2 OBJETIVOS

2.1 Descrição da atividade

Para esta atividade, devemos definir uma API REST que implemente, pelo menos, duas funções do seu projeto. Para disponibilizar a API para terceiros, você deverá escolher uma provedora de serviço em nuvem analisada em aula (AWS, GCP ou Azure), criar uma máquina virtual para o serviço, realizar toda a configuração necessária e disponibilizar o serviço.

2.2 Proposta

Com objetivo de atender as demandas da atividade, temos como proposta a criação de uma Prova de Conceito (POC) de um sistema CRUD (Create, Read, Update e Delete) que implementará as 4 funções básicas para administração de um banco de dados de serviços, visto que o projeto da startup foca no gerenciamento de anúncios de prestadores de serviços. Dito isso, o desenvolvimento será realizado no framework Flask, pois visa uma fácil prototipagem das funções e, também, com o auxílio do Docker e PostgreSQL. Por fim, a API será disponibilizada por meio de uma máquina virtual fornecida pela AWS (EC2).

3 DESENVOLVIMENTO DA API

3.1 Dependências

Em primeiro lugar, precisamos definir as bibliotecas Python dependentes para nosso aplicativo. O método padrão em Python é criar um arquivo `requirements.txt` e listar nossas dependências. Portanto, criamos um arquivo chamado “`requirements.txt`” onde podemos digitar as seguintes dependências:

- **Flask:** o micro framework em Python onde a api será desenvolvida;
- **Psycopg2-Binary:** necessário para criar a conexão com o *Postgres Database*;
- **Flask-SQLAlchemy:** necessário para gerar as queries do SQL sem precisar programar manualmente.
-

```
≡ requirements.txt
1 flask
2 Flask-SQLAlchemy
3 psycopg2-binary
```

3.2 Criação da aplicação

- Primeiramente, devemos especificar as bibliotecas que usaremos:

```
from flask import Flask, request, jsonify
from flask_sqlalchemy import SQLAlchemy
import os
```

- Em seguida, definir o aplicativo Flask e como executá-lo:

```
app = Flask(__name__)  
  
if __name__ == "__main__":  
    app.run(debug=True)
```

- Definimos uma variável de ambiente como uma *string* e inicializamos a instância do *SQLAlchemy* para lidar com o banco de dados *Postgres*:

```
app.config["SQLALCHEMY_DATABASE_URI"] = os.environ.get("DATABASE_URL")  
db = SQLAlchemy(app)
```

- Em seguida, definimos nosso modelo de dados. Criaremos uma classe chamada *Anuncio* com apenas “título” e “descrição” como propriedades. Também adicionaremos um inteiro que se incrementará automaticamente chamado “id”. Isso atuará como a chave primária para nossa tabela.

```
class Anuncio(db.Model):  
    id = db.Column(db.Integer, primary_key=True)  
    titulo = db.Column(db.String(80), unique=True, nullable=False)  
    descricao = db.Column(db.String(240), unique=True, nullable=False)  
  
    def __init__(self, titulo, descricao):  
        self.titulo = titulo  
        self.descricao = descricao
```

- Por fim, com esta linha deixamos o *SQLAlchemy* sincronizar com o banco de dados *Postgres*. Isso criará nossa tabela de banco de dados automaticamente.

```
db.create_all()
```

3.3 Definindo os *endpoints* do *REST*

Dando sequência, agora precisamos implementar nossos *endpoints* para a aplicação funcionar como um CRUD, ou seja:

- **CREATE:** criar um novo anúncio;
- **READ:** ler um anúncio existente;
- **UPDATE:** atualizar um anúncio existente;
- **DELETE:** deletar um anúncio existente.

Estas serão as quatro funções básicas do nosso aplicativo.

- Para recuperar um único anúncio, definimos esta função:

```
@app.route("/anuncios/<id>", methods=["GET"])
def get_anuncio(id):
    item = Anuncio.query.get(id)
    del item.__dict__["_sa_instance_state"]
    return jsonify(item.__dict__)
```

- Para obter todos os anúncios no banco de dados, definimos esta função:

```
@app.route("/anuncios", methods=["GET"])
def get_anuncios():
    anuncios = []
    for anuncio in db.session.query(Anuncio).all():
        del anuncio.__dict__["_sa_instance_state"]
        anuncios.append(anuncio.__dict__)
    return jsonify(anuncios)
```


- Para criar um novo anúncio:

```
@app.route("/anuncios", methods=["POST"])
def create_anuncio():
    body = request.get_json()
    db.session.add(Anuncio(body["titulo"], body["descricao"]))
    db.session.commit()
    return "Anúncio criado!"
```

- Para atualizar um anúncio existente:

```
@app.route("/anuncios/<id>", methods=["PUT"])
def update_anuncio(id):
    body = request.get_json()
    db.session.query(Anuncio).filter_by(id=id).update(
        dict(title=body["titulo"], content=body["descricao"])
    )
    db.session.commit()
    return "Anúncio atualizado!"
```

- Para excluir um anúncio existente:

```
@app.route("/anuncios/<id>", methods=["DELETE"])
def delete_anuncio(id):
    db.session.query(Anuncio).filter_by(id=id).delete()
    db.session.commit()
    return "Anúncio deletado!"
```

3.4 Criando o *Dockerfile*

Um *Dockerfile* é um arquivo de texto para definir um conjunto de comandos para criar uma imagem. A partir desta imagem, executaremos nossos containers Python, facilitando a futura implantação da aplicação na AWS.

Temos como arquivo final:

```
FROM python:3.6-slim-buster

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

# EXPOSE 80

CMD ["flask", "run", "--host=0.0.0.0", "--port=80"]
```

Onde,

- **FROM:** Defina a imagem base a ser usada para instruções subsequentes. FROM deve ser a primeira instrução em um Dockerfile;
- **COPY:** Copia arquivos ou pastas da origem para o caminho de destino no sistema de arquivos da imagem. A primeira COPY copia o arquivo requirements.txt dentro do sistema de arquivos da imagem; o segundo copia todo o resto;
- **RUN:** Executa qualquer comando no topo da imagem atual como uma nova camada e confirma os resultados. Nesse caso, estamos executando o *pip* para instalar as bibliotecas Python listadas nas dependências (arquivo “requirements.txt”);
- **EXPOSE:** Informa ao Docker a porta que usaremos na execução;
- **CMD:** forneça padrões para um container em execução. Se um executável não for especificado, ENTRYPOINT também deverá ser especificado. Só pode haver uma instrução CMD em um Dockerfile.

3.5 Criando o Docker Compose

Para facilitar a execução e o gerenciamento dos containers e imagens, criamos um arquivo *docker-compose.yml* contendo as seguintes informações:

```
version: '3.9'
```

```
services:
```

```
  pythonapp:
```

```
    container_name: pythonapp
```

```
    image: pythonapp
```

```
    build: .
```

```
    ports:
```

```
      - "80:80"
```

```
    environment:
```

```
      - DATABASE_URL=postgresql://postgres:postgres@db:5432/postgres
```

```
    depends_on:
```

```
      - db
```

```
  db:
```

```
    container_name: db
```

```
    image: postgres:12
```

```
    ports:
```

```
      - "5432:5432"
```

```
    environment:
```

```
      - POSTGRES_PASSWORD=postgres
```

```
      - POSTGRES_USER=postgres
```

```
      - POSTGRES_DB=postgres
```

```
    volumes:
```

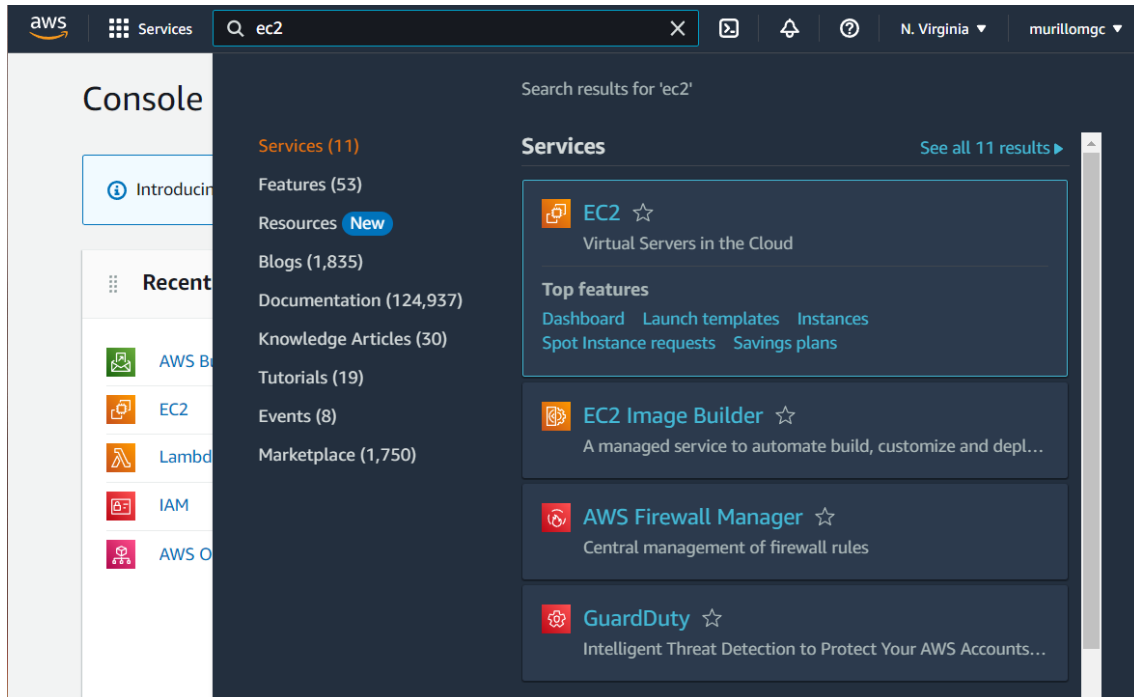
```
      - pgdata:/var/lib/postgresql/data
```

```
volumes:
```

```
  pgdata: {}
```

4 IMPLANTAÇÃO DA API NA AWS

- Após a criação da conta na AWS, o primeiro passo é procurar pelo serviço do EC2 (Elastic Compute Cloud):



- Em seguida, selecionar a opção “*Launch instance*” para dar início a configuração da máquina virtual:

Resources

EC2 Global view ↗ ↻ ⚙

You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:

| | | | |
|---------------------|---|-----------------|---|
| Instances (running) | 0 | Dedicated Hosts | 0 |
| Elastic IPs | 0 | Instances | 0 |
| Key pairs | 0 | Load balancers | 0 |
| Placement groups | 0 | Security groups | 1 |
| Snapshots | 0 | Volumes | 0 |

i Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. [Learn more](#) ✕

Launch instance

To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.

Launch instance ▼

Migrate a server ↗

Note: Your instances will launch in the US East (N. Virginia) Region

Service health

↻

AWS Health Dashboard ↗

Region
US East (N. Virginia)

Status
✔ This service is operating normally

- Adicionar um nome para a instância da máquina virtual:

Launch an instance [Info](#)

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags [Info](#)

Name

ava3_cloud

[Add additional tags](#)

- Selecionar a imagem base para o OS (Amazon Linux 2):

▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

🔍 Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux
aws


macOS
Mac

Ubuntu
ubuntu

Windows
Microsoft

Red Hat
Red Hat

SL
>


Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type
ami-0b0dcb5067f052a63 (64-bit (x86)) / ami-01b5ec3ed8678d8b7 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20221103.3 x86_64 HVM gp2

Architecture

64-bit (x86) ▼

AMI ID

ami-0b0dcb5067f052a63

Verified provider

- Selecionar o tipo t2.micro:

▼ Instance type [Info](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory

On-Demand Linux pricing: 0.0116 USD per Hour

On-Demand Windows pricing: 0.0162 USD per Hour

Free tier eligible ▼

[Compare instance types](#)

- Criar um par de chaves para a futura autenticação por SSH:

Create key pair

×

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn more](#) ↗

Key pair name

ava3_cloud_key

The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type

☒ RSA
RSA encrypted private and public key pair

☐ ED25519
ED25519 encrypted private and public key pair (Not supported for Windows instances)

Private key file format

☒ .pem
For use with OpenSSH

☐ .ppk
For use with PuTTY

Cancel

Create key pair


- Habilitar as chaves criadas para a instância sendo configurada:

▼ **Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

ava3_cloud_key ▼

 [Create new key pair](#)

- Configurar as informações de rede para permitir o acesso da aplicação por SSH, HTTP e HTTPS:

▼ **Network settings** [Info](#)

[Edit](#)

Network [Info](#)

vpc-02eaaad26bf0a1269

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

☒ Allow SSH traffic from

Helps you connect to your instance

Anywhere


0.0.0.0/0 ▼


☒ Allow HTTPS traffic from the internet

To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet

To set up an endpoint, for example when creating a web server

 Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.



- Configurar o armazenamento:

▼ **Configure storage** [Info](#) [Advanced](#)

1x GiB ▼ Root volume (Not encrypted)

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage

×

Add new volume

0 x File systems [Edit](#)

- Resumo das configurações da instância:

▼ **Summary**

Number of instances [Info](#)

Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...[read more](#)
ami-0b0dcb5067f052a63

Virtual server type (instance type)
t2.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

ⓘ **Free tier:** In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier AMIs per month, 30 GiB of EBS storage, 2 million IOs, 1 GB of snapshots, and 100 GB of bandwidth to the internet.

×

Cancel

Launch instance

- Em seguida, a conexão na instância pelo próprio console da AWS:

EC2 > Instances > i-0b0d77eac4503f33a > Connect to instance

Connect to instance [Info](#)

Connect to your instance i-0b0d77eac4503f33a (ava3_cloud) using any of these options

EC2 Instance Connect

Session Manager

SSH client

EC2 serial console

Instance ID
i-0b0d77eac4503f33a (ava3_cloud)

Public IP address
3.94.101.139

User name

Connect using a custom user name, or use the default user name ec2-user for the AMI used to launch the instance.

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

Connect

- Atualização dos pacotes:

```
aws Services Search [Alt+S] N. Virginia murillomgc
[ec2-user@ip-172-31-83-3 ~]$ sudo yum update
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package kernel.x86_64 0:5.10.149-133.644.amzn2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
kernel x86_64 5.10.149-133.644.amzn2 amzn2extra-kernel-5.10 32 M
Transaction Summary
=====
Install 1 Package

Total download size: 32 M
Installed size: 136 M
Is this ok [y/d/N]: y
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
kernel-5.10.149-133.644.amzn2.x86_64.rpm | 32 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : kernel-5.10.149-133.644.amzn2.x86_64 1/1
  Verifying : kernel-5.10.149-133.644.amzn2.x86_64 1/1

Installed:
kernel.x86_64 0:5.10.149-133.644.amzn2

Complete!
[ec2-user@ip-172-31-83-3 ~]$
```

- Instalação do *Docker*:

```
aws Services Search [Alt+S] N. Virginia murillomgc
[ec2-user@ip-172-31-83-3 ~]$ sudo yum install docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.17-1.amzn2.0.1 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: libcgroupp >= 0.40.rc1-5.15 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Processing Dependency: pigz for package: docker-20.10.17-1.amzn2.0.1.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.6.6-1.amzn2.0.2 will be installed
--> Package libcgroupp.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.1.3-1.amzn2.0.2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
docker x86_64 20.10.17-1.amzn2.0.1 amzn2extra-docker 39 M
Installing for dependencies:
containerd x86_64 1.6.6-1.amzn2.0.2 amzn2extra-docker 27 M
libcgroupp x86_64 0.41-21.amzn2 amzn2-core 66 k
pigz x86_64 2.3.4-1.amzn2.0.1 amzn2-core 81 k
runc x86_64 1.1.3-1.amzn2.0.2 amzn2extra-docker 2.9 M
=====

Transaction Summary
=====
Install 1 Package (+4 Dependent packages)

Total download size: 69 M
Installed size: 260 M
Is this ok [y/d/N]: y
Downloading packages:
(1/5): libcgroupp-0.41-21.amzn2.x86_64.rpm | 66 kB 00:00:00
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm | 81 kB 00:00:00
(3/5): containerd-1.6.6-1.amzn2.0.2.x86_64.rpm | 27 MB 00:00:00
(4/5): docker-20.10.17-1.amzn2.0.1.x86_64.rpm | 39 MB 00:00:01
(5/5): runc-1.1.3-1.amzn2.0.2.x86_64.rpm | 2.9 MB 00:00:00
=====
Total 65 MB/s | 69 MB 00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : runc-1.1.3-1.amzn2.0.2.x86_64 1/5
Installing : containerd-1.6.6-1.amzn2.0.2.x86_64 2/5
Installing : libcgroupp-0.41-21.amzn2.x86_64 3/5
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
Installing : docker-20.10.17-1.amzn2.0.1.x86_64 5/5
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 1/5
Verifying : libcgroupp-0.41-21.amzn2.x86_64 2/5
Verifying : docker-20.10.17-1.amzn2.0.1.x86_64 3/5
Verifying : containerd-1.6.6-1.amzn2.0.2.x86_64 4/5
Verifying : runc-1.1.3-1.amzn2.0.2.x86_64 5/5

Installed:
docker.x86_64 0:20.10.17-1.amzn2.0.1

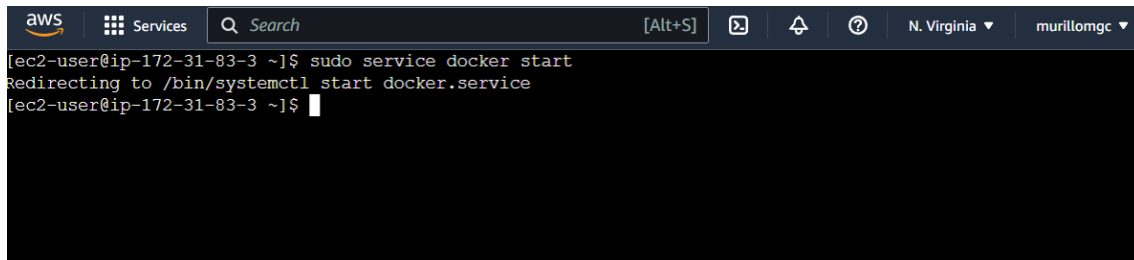
Dependency Installed:
containerd.x86_64 0:1.6.6-1.amzn2.0.2 libcgroupp.x86_64 0:0.41-21.amzn2 pigz.x86_64 0:2.3.4-1.amzn2.0.1
runc.x86_64 0:1.1.3-1.amzn2.0.2

Complete!
[ec2-user@ip-172-31-83-3 ~]$
```

- Instalação do *pip* e *docker-compose*:

```
aws Services Search [Alt+S] N. Virginia murillomgc
[ec2-user@ip-172-31-83-3 ~]$ sudo yum install python3-pip
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package python3-pip-20.2.2-1.amzn2.0.3.noarch already installed and latest version
Nothing to do
[ec2-user@ip-172-31-83-3 ~]$ pip3 install --user docker-compose # without root access for security reasons
Requirement already satisfied: docker-compose in /usr/local/lib/python3.7/site-packages (1.29.2)
Requirement already satisfied: texttable<2,>=0.9.0 in /usr/local/lib/python3.7/site-packages (from docker-c
ompose) (1.6.5)
Requirement already satisfied: cached-property<2,>=1.2.0; python_version < "3.8" in /usr/local/lib/python3.
7/site-packages (from docker-compose) (1.5.2)
Requirement already satisfied: jsonschema<4,>=2.5.1 in /usr/local/lib/python3.7/site-packages (from docker-c
ompose) (3.2.0)
Requirement already satisfied: distro<2,>=1.5.0 in /usr/local/lib/python3.7/site-packages (from docker-comp
ose) (1.8.0)
Requirement already satisfied: python-dotenv<1,>=0.13.0 in /usr/local/lib/python3.7/site-packages (from doc
ker-compose) (0.21.0)
Requirement already satisfied: websocket-client<1,>=0.32.0 in /usr/local/lib/python3.7/site-packages (from
docker-compose) (0.59.0)
Requirement already satisfied: docker[ssh]>=5 in /usr/local/lib/python3.7/site-packages (from docker-compos
e) (6.0.1)
Requirement already satisfied: requests<3,>=2.20.0 in /usr/local/lib/python3.7/site-packages (from docker-c
ompose) (2.28.1)
Requirement already satisfied: PyYAML<6,>=3.10 in /usr/local/lib64/python3.7/site-packages (from docker-com
pose) (5.4.1)
Requirement already satisfied: dockerpty<1,>=0.4.1 in /usr/local/lib/python3.7/site-packages (from docker-c
ompose) (0.4.1)
Requirement already satisfied: docopt<1,>=0.6.1 in /usr/local/lib/python3.7/site-packages (from docker-comp
ose) (0.6.2)
Requirement already satisfied: setuptools in /usr/lib/python3.7/site-packages (from jsonschema<4,>=2.5.1->d
ocker-compose) (49.1.3)
Requirement already satisfied: six>=1.11.0 in /usr/local/lib/python3.7/site-packages (from jsonschema<4,>=2
.5.1->docker-compose) (1.16.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /usr/local/lib/python3.7/site-packages (from jsonschem
a<4,>=2.5.1->docker-compose) (0.19.2)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/local/lib/python3.7/site-
packages (from jsonschema<4,>=2.5.1->docker-compose) (5.0.0)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/site-packages (from jsonschema<4,>
=2.5.1->docker-compose) (22.1.0)
Requirement already satisfied: packaging>=14.0 in /usr/local/lib/python3.7/site-packages (from docker[ssh]>
=5->docker-compose) (21.3)
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.7/site-packages (from docker[ssh]>
=5->docker-compose) (1.26.12)
Requirement already satisfied: paramiko>=2.4.3; extra == "ssh" in /usr/local/lib/python3.7/site-packages (f
rom docker[ssh]>=5->docker-compose) (2.12.0)
Requirement already satisfied: charset-normalizer<3,>=2 in /usr/local/lib/python3.7/site-packages (from req
uests<3,>=2.20.0->docker-compose) (2.1.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.7/site-packages (from requests<3,>=2.
20.0->docker-compose) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/site-packages (from requests<
3,>=2.20.0->docker-compose) (2022.9.24)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/site-packages (from importlib-metadata
; python_version < "3.8"->jsonschema<4,>=2.5.1->docker-compose) (3.10.0)
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in /usr/local/lib/python3.7
/site-packages (from importlib-metadata; python_version < "3.8"->jsonschema<4,>=2.5.1->docker-compose) (4.4
.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/site-packages (from pac
kaging>=14.0->docker[ssh]>=5->docker-compose) (3.0.9)
Requirement already satisfied: cryptography>=2.5 in /usr/local/lib64/python3.7/site-packages (from paramiko
>=2.4.3; extra == "ssh"->docker[ssh]>=5->docker-compose) (38.0.3)
Requirement already satisfied: bcrypt>=3.1.3 in /usr/local/lib64/python3.7/site-packages (from paramiko>=2.
4.3; extra == "ssh"->docker[ssh]>=5->docker-compose) (4.0.1)
Requirement already satisfied: pynacl>=1.0.1 in /usr/local/lib64/python3.7/site-packages (from paramiko>=2.
4.3; extra == "ssh"->docker[ssh]>=5->docker-compose) (1.5.0)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib64/python3.7/site-packages (from cryptography>=2
.5->paramiko>=2.4.3; extra == "ssh"->docker[ssh]>=5->docker-compose) (1.15.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/site-packages (from cffi>=1.12->crypto
graphy>=2.5->paramiko>=2.4.3; extra == "ssh"->docker[ssh]>=5->docker-compose) (2.21)
[ec2-user@ip-172-31-83-3 ~]$
```

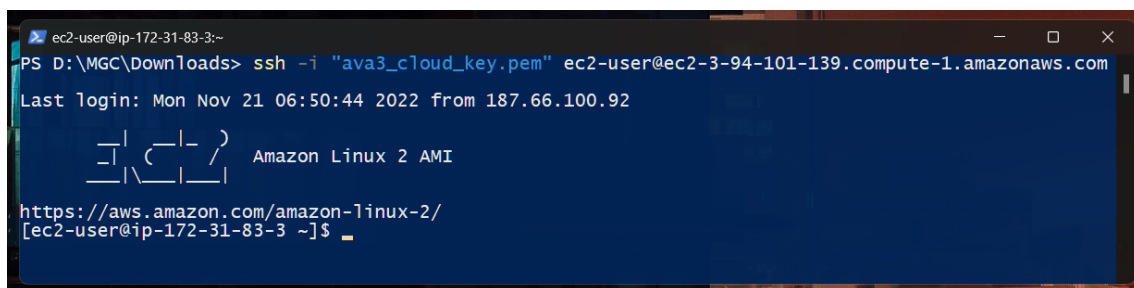
- Inicialização do serviço do *Docker*:



A terminal window with an AWS header bar. The header bar includes the AWS logo, 'Services', a search bar, '[Alt+S]', and icons for help, notifications, and a dropdown menu showing 'N. Virginia' and 'murillomgc'. The terminal text shows the command 'sudo service docker start' being executed, followed by a message 'Redirecting to /bin/systemctl start docker.service' and a prompt for the user 'ec2-user@ip-172-31-83-3 ~]'.

```
aws Services Search [Alt+S] N. Virginia murillomgc
[ec2-user@ip-172-31-83-3 ~]$ sudo service docker start
Redirecting to /bin/systemctl start docker.service
[ec2-user@ip-172-31-83-3 ~]$
```

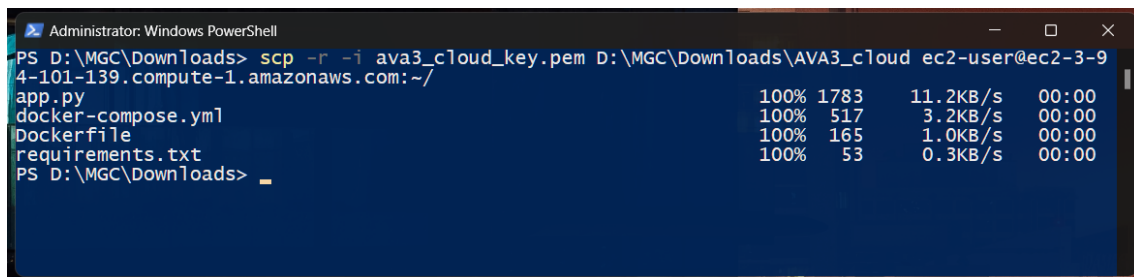
- Teste de conexão da máquina local com os arquivos do projeto:



A terminal window showing an SSH connection from a Windows machine to an Amazon Linux 2 instance. The prompt is 'PS D:\MGC\Downloads>'. The command is 'ssh -i "ava3_cloud_key.pem" ec2-user@ec2-3-94-101-139.compute-1.amazonaws.com'. The output shows the last login time and a welcome message from the Amazon Linux 2 AMI, including the URL 'https://aws.amazon.com/amazon-linux-2/'. The prompt changes to '[ec2-user@ip-172-31-83-3 ~]\$'.

```
ec2-user@ip-172-31-83-3:~
PS D:\MGC\Downloads> ssh -i "ava3_cloud_key.pem" ec2-user@ec2-3-94-101-139.compute-1.amazonaws.com
Last login: Mon Nov 21 06:50:44 2022 from 187.66.100.92
 _ _ _ _ _
| | | | |
|_|_|_|_|_| Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-83-3 ~]$
```

- Upload via SSH dos arquivos do projeto para a instância na AWS:



A Windows PowerShell terminal window showing the upload of project files to an AWS instance using SCP. The prompt is 'PS D:\MGC\Downloads>'. The command is 'scp -r -i ava3_cloud_key.pem D:\MGC\Downloads\AVA3_cloud ec2-user@ec2-3-94-101-139.compute-1.amazonaws.com:~/'. The output shows a progress bar for each file: 'app.py' (100% 1783 11.2KB/s 00:00), 'docker-compose.yml' (100% 517 3.2KB/s 00:00), 'Dockerfile' (100% 165 1.0KB/s 00:00), and 'requirements.txt' (100% 53 0.3KB/s 00:00). The prompt returns to 'PS D:\MGC\Downloads>'.

```
Administrator: Windows PowerShell
PS D:\MGC\Downloads> scp -r -i ava3_cloud_key.pem D:\MGC\Downloads\AVA3_cloud ec2-user@ec2-3-94-101-139.compute-1.amazonaws.com:~/
app.py 100% 1783 11.2KB/s 00:00
docker-compose.yml 100% 517 3.2KB/s 00:00
Dockerfile 100% 165 1.0KB/s 00:00
requirements.txt 100% 53 0.3KB/s 00:00
PS D:\MGC\Downloads>
```

- Por fim, a execução do *docker-compose*, criando as imagens necessárias e executando os containers do banco de dados (*db*) e do CRUD em API REST desenvolvido (*pythonapp*):

```
aws Services Search [Alt+S] N. Virginia murillomgc
[ec2-user@ip-172-31-83-3 AWA3_cloud]$ sudo docker-compose up
Creating db ... done
Creating pythonapp ... done
Attaching to db, pythonapp
db
| PostgreSQL Database directory appears to contain a database; Skipping initialization
db
| 2022-11-21 07:17:20.990 UTC [1] LOG: starting PostgreSQL 12.13 (Debian 12.13-1.pgdg110+1) o
n x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
db
| 2022-11-21 07:17:20.992 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db
| 2022-11-21 07:17:20.992 UTC [1] LOG: listening on IPv6 address ":::", port 5432
db
| 2022-11-21 07:17:20.995 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL
.5432"
db
| 2022-11-21 07:17:21.024 UTC [25] LOG: database system was shut down at 2022-11-21 07:07:12
UTC
db
| 2022-11-21 07:17:21.052 UTC [1] LOG: database system is ready to accept connections
pythonapp
| * Environment: production
pythonapp
| WARNING: This is a development server. Do not use it in a production deployment.
pythonapp
| Use a production WSGI server instead.
pythonapp
| * Debug mode: off
pythonapp
| /usr/local/lib/python3.6/site-packages/flask_sqlalchemy/__init__.py:873: FSADeprecationWarni
ng: SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and will be disabled by default in the future.
Set it to True or False to suppress this warning.
pythonapp
| 'SQLALCHEMY_TRACK_MODIFICATIONS adds significant overhead and '
pythonapp
| /usr/local/lib/python3.6/site-packages/psycopg2/__init__.py:144: UserWarning: The psycopg2 w
heel package will be renamed from release 2.8; in order to keep installing from binary please use "pip inst
all psycopg2-binary" instead. For details see: <http://initd.org/psycopg/docs/install.html#binary-install-f
rom-pypi>.
pythonapp
| """)
pythonapp
| * Running on all addresses.
pythonapp
| WARNING: This is a development server. Do not use it in a production deployment.
pythonapp
| * Running on http://172.18.0.3:80/ (Press CTRL+C to quit)
```

5 REFERÊNCIAS

DHOLAKIA, J. Creating RESTful Web APIs using Flask and Python. **Towards Data Science**, 2020. Disponível em: <https://towardsdatascience.com/creating-restful-apis-using-flask-and-python-655bad51b24> . Acesso em: 18 nov. 2022.