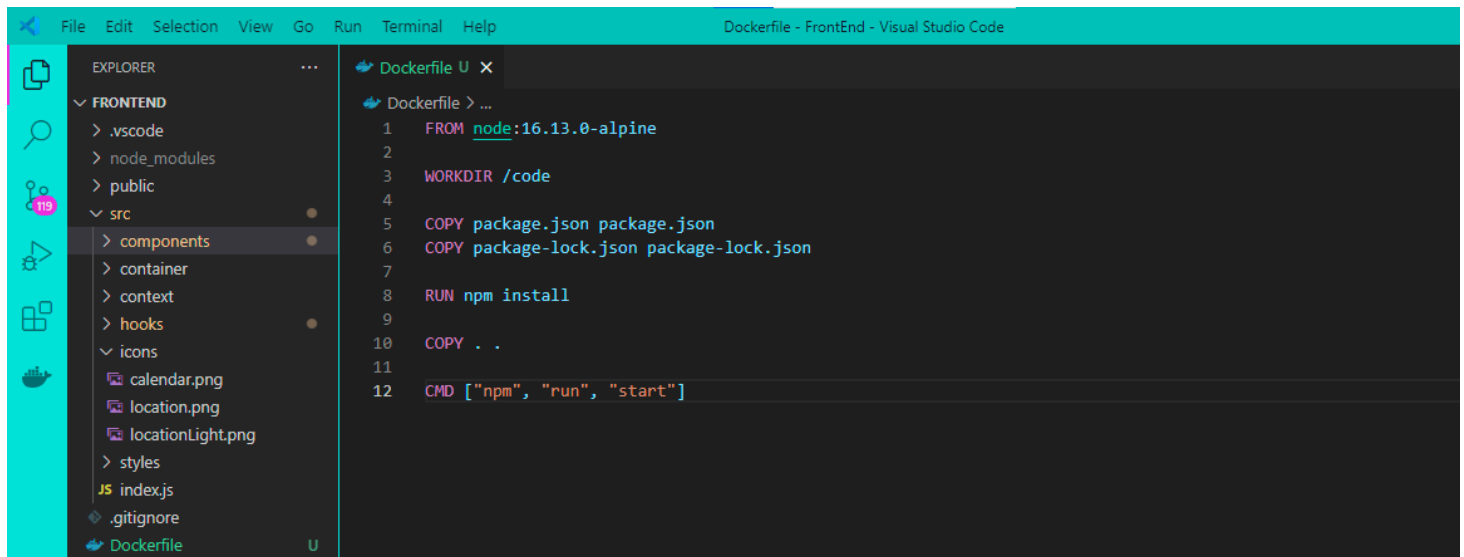


1. Deploy del código en el servidor web EC2 en AWS.

- (Opcional - Modo A) Hacer el deploy en una instancia Docker en el EC2.

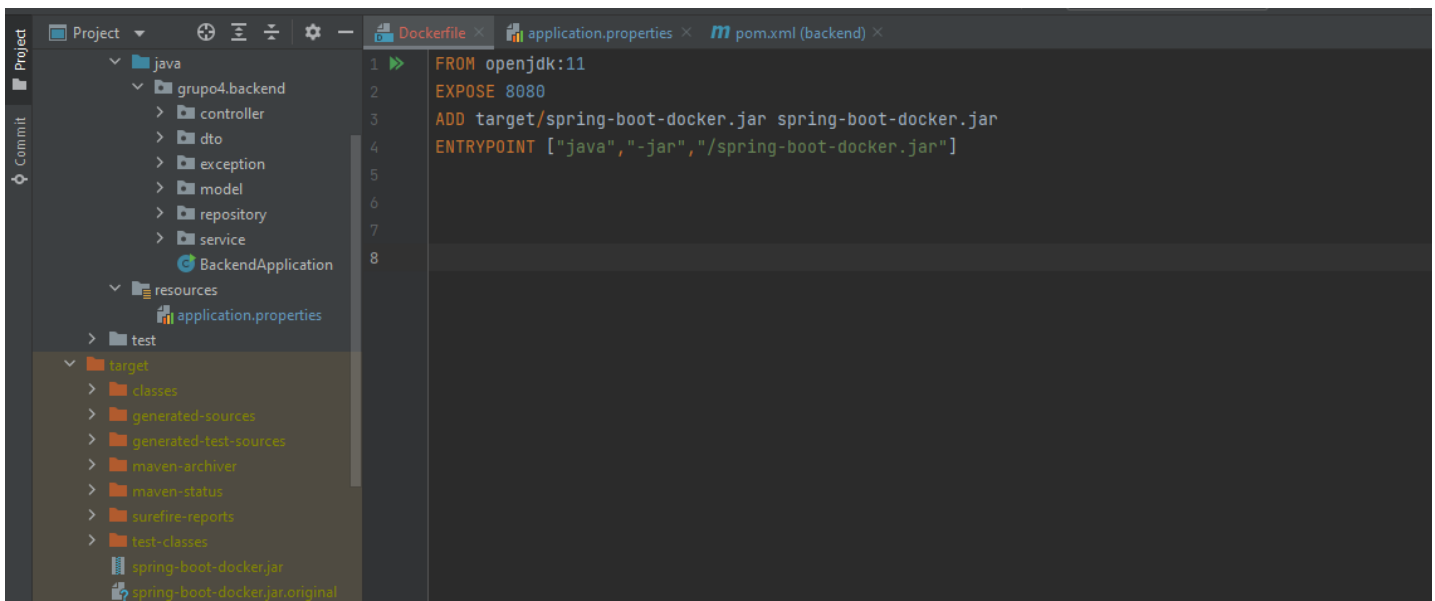
Primero creamos los Dockerfile, tanto del front como del back. Comenzamos con una imagen base y una versión específica (node:16.13.0 / openjdk:11); además, en el caso de node la construimos sobre *alpine* que es la más ligera de todas las distribuciones de linux), y esta imagen base la pone dentro de la imagen final encima de todo.

Luego seteamos el working directory, llamado code en nuestro caso. Nos moverá dentro de ese working directory y todos los comandos que se ejecuten después operarán dentro de él. Se copian los packages json dentro del working directory. El *run install* busca el package.json y el package-lock.json y luego instala el node_modules. Luego copiamos el código fuente donde estamos en la imagen. Finalmente el CMD se usa para indicarle el comando que docker engine debe ejecutar cuando se corre la imagen (docker run).



The screenshot shows the Visual Studio Code interface with a Dockerfile for the frontend. The Explorer on the left shows the project structure with folders like .vscode, node_modules, public, src, components, container, context, hooks, icons, and files like calendar.png, location.png, locationLight.png, styles, index.js, .gitignore, and Dockerfile. The Dockerfile content is as follows:

```
Dockerfile U
1 FROM node:16.13.0-alpine
2
3 WORKDIR /code
4
5 COPY package.json package.json
6 COPY package-lock.json package-lock.json
7
8 RUN npm install
9
10 COPY . .
11
12 CMD ["npm", "run", "start"]
```



The screenshot shows the Visual Studio Code interface with a Dockerfile for the backend. The Project Explorer on the left shows the project structure with folders like java, grupo4.backend, controller, dto, exception, model, repository, service, BackendApplication, resources, application.properties, test, target, classes, generated-sources, generated-test-sources, maven-archiver, maven-status, surefire-reports, test-classes, spring-boot-docker.jar, and spring-boot-docker.jar.original. The Dockerfile content is as follows:

```
Dockerfile
1 FROM openjdk:11
2
3 EXPOSE 8080
4
5 ADD target/spring-boot-docker.jar spring-boot-docker.jar
6
7 ENTRYPOINT ["java", "-jar", "/spring-boot-docker.jar"]
8
```

Docker build

Construimos las imágenes con el comando **docker build** de los Dockerfile. Los nombramos, y al no darle una versión específica, quedó como *latest*. También le dimos un contexto donde tiene que ejecutar el Dockerfile, en nuestro caso es en el mismo directorio donde ejecutamos el comando.

```
thairy@DESKTOP-B1VR69A MINGW64 ~/Desktop/repo/grupo-4/FrontEnd (development)
$ docker build -t react-docker-v2 .

[+] Building 6.2s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:16.13.0-alpine
=> [internal] load build context
=> => transferring context: 3.59MB
=> [1/6] FROM docker.io/library/node:16.13.0-alpine@sha256:60ef0bed1dc2ec835cfe3c4226d074fdfab571fd619c280474cc04e93f0ec5b
=> CACHED [2/6] WORKDIR /code
=> CACHED [3/6] COPY package.json package.json
=> CACHED [4/6] COPY package-lock.json package-lock.json
=> CACHED [5/6] RUN npm install
=> CACHED [6/6] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:ee734fdbaa06b9d82d2f70594d9f3d4b74af32097b32e1ca920c4c9c43e2688a
=> => naming to docker.io/library/react-docker-v2
```

```
thairy@DESKTOP-B1VR69A MINGW64 ~/Desktop/repo/grupo-4/BackEnd (development)
$ docker build -t spring-boot-docker.jar .

[+] Building 1.2s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 32B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:11
=> [internal] load build context
=> => transferring context: 79B
=> [1/2] FROM docker.io/library/openjdk:11@sha256:84539d4caf6f51c850978ee138458560f84c12e647ad78b8fd9f24854b27da1d
=> CACHED [2/2] ADD target/spring-boot-docker.jar spring-boot-docker.jar
=> exporting to image
=> => exporting layers
=> => writing image sha256:a24f4869a2d1ad3b3e5bb6ae059ca9fc69c748bcee68ecfc60954ffa7904c7a0
=> => naming to docker.io/library/spring-boot-docker.jar
```

Docker push

Taguemos y pusheamos las imágenes de docker host a dockerhub, para eso tuvimos que registrar nuestras credenciales de dockerhub.

```
thairy@DESKTOP-B1VR69A MINGW64 ~/Desktop/repo/grupo-4/FrontEnd (development)
$ docker tag react-docker-v2 dazathai/react-docker-v2


thairy@DESKTOP-B1VR69A MINGW64 ~/Desktop/repo/grupo-4/FrontEnd (development)
$ docker push dazathai/react-docker-v2
Using default tag: latest
The push refers to repository [docker.io/dazathai/react-docker-v2]
9f4b98f7ee59: Pushed
2ef09af3160e: Pushed
5b0f16c814d2: Pushed
c812601db344: Pushed
c095acf4f01e: Pushed
ff64ee97d76a: Pushed
480f61641fa1: Pushed
b3eae7a085d: Pushed
1a058d5342cc: Mounted from adoptopenjdk/openjdk11
latest: digest: sha256:130363cf805a0894138861145b57e72e21a9a42ec800731b6cda4bb093812fdd size: 2206
```

Si ya habíamos pusheado la imagen previamente, verá si hay diferencias entre ambas versiones, copiará lo que haya cambiado.

```
thairy@DESKTOP-B1VR69A MINGW64 ~/Desktop/repo/grupo-4/BackEnd (development)
$ docker tag spring-boot-docker dazathai/spring-boot-docker


thairy@DESKTOP-B1VR69A MINGW64 ~/Desktop/repo/grupo-4/BackEnd (development)
$ docker push dazathai/spring-boot-docker
Using default tag: latest
The push refers to repository [docker.io/dazathai/spring-boot-docker]
18d3b8bd7683: Layer already exists
a9e4c9343539: Layer already exists
```

Dockerhub

 Search for great content (e.g., mysql)

ExploreRepositoriesOrganizationsHelp

Upgrade

 dazathai

dazathai

Search by repository name

Create Repository

dazathai / react-docker-v2

Updated a few seconds ago

Not Scanned

0

1

Public

dazathai / spring-boot-docker


Updated 44 minutes ago

Not Scanned


0

12

Public



Tip: Not finding your repository? Try switching namespace via the top left dropdown.



dazathai/react-docker-v2:latest

DIGEST: sha256:130363cf805a0894138861145b57e72e21a9a42ec800731b6cda4bb093812fdd

Delete Tag

OS/ARCH

linux/amd64

COMPRESSED SIZE

174.26 MB

LAST PUSHED

26 minutes ago by dazathai

Image Layers

Vulnerabilities

IMAGE LAYERS

1	ADD file ... in /	2.69 MB
2	CMD ["/bin/sh"]	0 B
3	ENV NODE_VERSION=16.13.0	0 B
4	/bin/sh -c addgroup -g 1000	33.2 MB
5	ENV VARN_VERSION=1.7.2	0 B

Command

ADD file:762c899ec0505d1a32930ee804c5b008825f41611161be104076cba33b7e5b2b in /



dazathai/spring-boot-docker:latest

DIGEST: sha256:ecc10d0a793f4a42dd56cd3e279d2a8b2bd24f299e71677444363b4b81e1b883

OS/ARCH
linux/amd64

COMPRESSED SIZE ⓘ
363.9 MB

LAST PUSHED
15 minutes ago by dazathai

[Delete Tag](#)

Image Layers

Vulnerabilities

IMAGE LAYERS ⓘ

1	ADD file ... in /	52.39 MB
2	CMD ["bash"]	0 B
3	/bin/sh -c set -eux; apt-get	4.91 MB
4	/bin/sh -c set -ex; if	10.37 MB
5	/bin/sh -c apt-get update &&	52.04 MB
6	/bin/sh -c set -eux; apt-get	5.17 MB
7	ENV JAVA_HOME=/usr/local/openjdk-11	0 B
8	/bin/sh -c { echo '#/bin/sh';	209 B

Command

```
ADD file:5259fc086e8295ddb02e48abef38e9bf93a183079d3631aa7a59306b7f2f9df in /
```

Ec2 AWS

En la ec2 primero instalamos docker para poder usarlo

```
[ec2-user@ip-172-31-5-144 ~]$ sudo amazon-linux-extras install docker
Installing docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Cleaning repos: amzn2-core amzn2extra-docker
12 metadata files removed
4 sqlite files removed
0 metadata files removed
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core | 3.7 kB 00:00:00
amzn2extra-docker | 3.0 kB 00:00:00
(1/5): amzn2-core/2/x86_64/group_gz | 2.5 kB 00:00:00
(2/5): amzn2-core/2/x86_64/updateinfo | 423 kB 00:00:00
(3/5): amzn2extra-docker/2/x86_64/updateinfo | 76 B 00:00:00
(4/5): amzn2extra-docker/2/x86_64/primary_db | 84 kB 00:00:00
(5/5): amzn2-core/2/x86_64/primary_db | 58 MB 00:00:00
Resolving Dependencies
--> Running transaction check
---> Package docker.x86_64 0:20.10.7-3.amzn2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.7-3.amzn2.x86_64
--> Processing Dependency: libcgroup >= 0.40.rc1-5.15 for package: docker-20.10.7-3.amzn2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.7-3.amzn2.x86_64
```

Docker pull

Hicimos un *docker pull* de las imágenes de dockerhub: si pulleamos la misma imagen, nos dirá que ya está actualizada

```
[ec2-user@ip-172-31-5-144 ~]$ docker pull dazathai/react-docker-v2
Using default tag: latest
latest: Pulling from dazathai/react-docker-v2
Digest: sha256:130363cf805a0894138861145b57e72e21a9a42ec800731b6cda4bb093812fdd
Status: Image is up to date for dazathai/react-docker-v2:latest
docker.io/dazathai/react-docker-v2:latest
[ec2-user@ip-172-31-5-144 ~]$ |
```

```
[ec2-user@ip-172-31-5-144 ~]$ docker pull dazathai/spring-boot-docker
Using default tag: latest
latest: Pulling from dazathai/spring-boot-docker
Digest: sha256:ecc10d0a793f4a42dd56cd3e279d2a8b2bd24f299e71677444363b4b81e1b883
Status: Image is up to date for dazathai/spring-boot-docker:latest
docker.io/dazathai/spring-boot-docker:latest
[ec2-user@ip-172-31-5-144 ~]$ |
```

```
[ec2-user@ip-172-31-5-144 ~]$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED      SIZE
dazathai/react-docker  latest     09af8adf615f  2 days ago  573MB
dazathai/spring-boot-docker  latest     a24f4869a2d1  4 days ago  711MB
[ec2-user@ip-172-31-5-144 ~]$ |
```

Docker run

Corrimos las imágenes dentro de la ec2 con el comando *docker run* “*nombre de la imagen*”. Expusimos (publicamos) el puerto 3000, y mapeamos el puerto 3000 de la máquina virtual (que habilitamos previamente en su grupo de seguridad) con el puerto 3000 del contenedor. El proceso está aislado de la red por defecto por eso debemos exponer el puerto del contenedor y habilitar que el tráfico del puerto de la máquina virtual mapee con el del contenedor.

React

```
[ec2-user@ip-172-31-5-144 ~]$ docker run -it -p 3000:3000 dazathai/react-docker

> proyecto-integrador-grupo4@0.1.0 start
> react-scripts start
```

```
ec2-user@ip-172-31-5-144:~
Compiled with warnings.

src/hooks/useFetch.jsx
  Line 26:6:  React Hook useEffect has missing dependencies: 'URL' and 'options'. Either include them or remove the dependency array react-hooks/exhaustive-deps
Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

Java

```
[ec2-user@ip-172-31-5-144 ~]$ docker run -it -p 8080:8080 dazathai/spring-boot-docker

:: Spring Boot :: (v2.5.5)

2021-11-25 13:22:13.661 INFO 1 --- [main] grupo4.backend.BackendApplication : Starting BackendApplication v0.0.1
ing-boot-docker.jar started by root in /)
2021-11-25 13:22:13.668 INFO 1 --- [main] grupo4.backend.BackendApplication : No active profile set, falling back
2021-11-25 13:22:16.381 INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repository
2021-11-25 13:22:16.623 INFO 1 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository
```

Docker ps

Con el comando docker ps vemos los contenedores que están corriendo

```
[ec2-user@ip-172-31-5-144 ~]$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
dce328ea7ae6   dazathai/react-docker-v2           "docker-entrypoint.s..." 21 minutes ago Up 21 minutes 0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   priceless_cori
4c71788f4e62   dazathai/spring-boot-docker       "java -jar /spring-b..." 47 minutes ago Up 47 minutes  0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   boring_rhodes
```

Visualizamos el sitio

Abrimos el DNS público que nos da AWS y visualizamos el sitio en los puertos 3000 y 8080, respectivamente

digital booking

Iniciar sesión Crear cuenta

Busca ofertas en hoteles, hostales y mucho más

¿A dónde vamos? Check in - Check out Buscar

Buscar por tipo de alojamiento

Hotel
807.105 Hoteles

Hostal
807.105 Hoteles

Bed and Breakfast
807.105 Hoteles

Lodge
807.105 Hoteles

Recomendaciones

©2021 Digital Booking

Facebook LinkedIn Twitter Instagram

HOTEL
Garden Park Hotel

📍 San Miguel de Tucuman, Argentina

Muy bueno
★★★★★ 8



/v3/api-docs

Explore

OpenAPI definition v0 OAS3

/v3/api-docs

Servers

http://ec2-54-151-38-245.us-west-1.compute.amazonaws.com:8080 - Generated server url

product-controller

PUT /API/products/update

POST /API/products

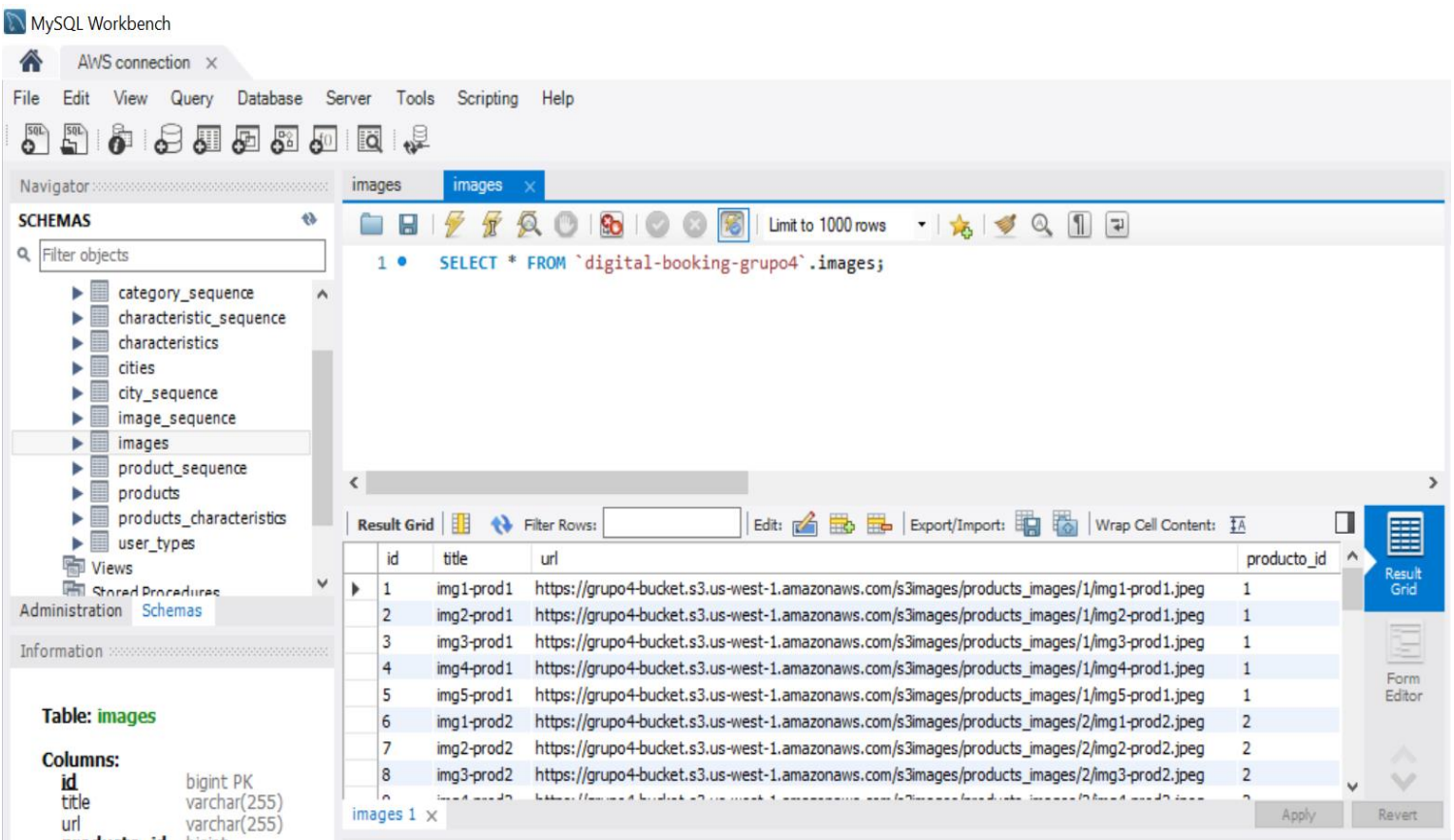
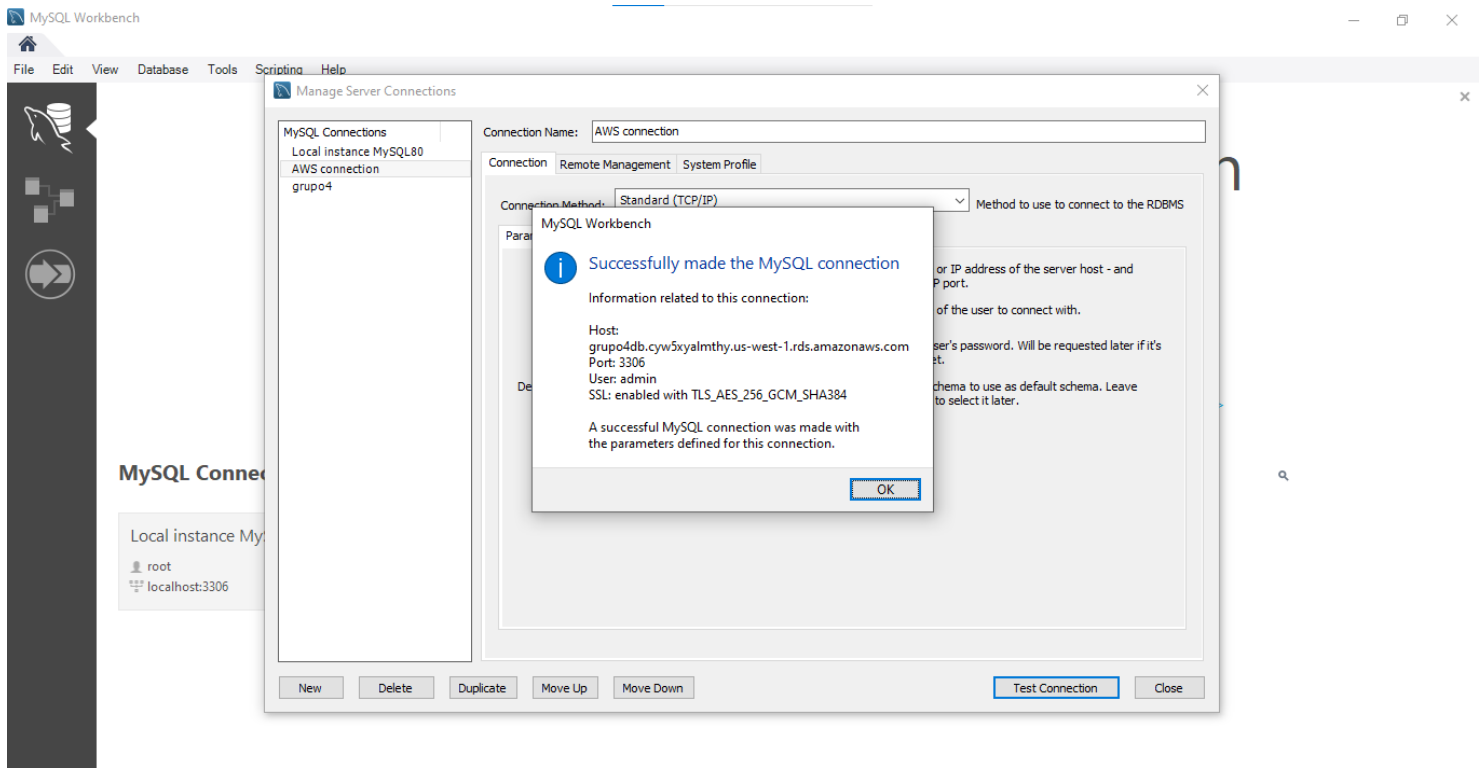
GET /API/products/{id}

DELETE /API/products/{id}

GET /API/products/getProductsByCity/{cityName}

2. Crear tablas en RDS en AWS

- Acceder a la base de datos en AWS a través de Workbench y crear las tablas y relaciones



3. Añadir imágenes en AWS Bucket para generar la URL

- Subir las imágenes al bucket, que generará una URL, añadir esas URLs a la base de datos.

Amazon S3

Account snapshot

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

Buckets (1) [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

< 1 > [Settings](#)

	Name ▲	AWS Region ▼	Access ▼	Creation date ▼
<input type="radio"/>	grupo4-bucket	US West (N. California) us-west-1	Public	November 10, 2021, 18:22:42 (UTC-03:00)

s3images/

[Copy S3 URI](#)

Objects



Properties

Objects (2)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

☐ Show versions

< 1 > [Settings](#)

<input type="checkbox"/>	Name ▲	Type ▼	Last modified ▼	Size ▼	Storage class ▼
<input type="checkbox"/>	 categories_images/	Folder	-	-	-
<input type="checkbox"/>	 products_images/	Folder	-	-	-

img1-prod1.jpeg [Info](#)[Copy S3 URI](#)[Download](#)[Open](#)[Object actions](#)[Properties](#)[Permissions](#)[Versions](#)

Object overview

Owner

thairydaza

AWS Region

US West (N. California) us-west-1

Last modified

November 11, 2021, 01:18:03 (UTC-03:00)

Size

146.5 KB

Type

jpeg

Key

[s3images/products_images/1/img1-prod1.jpeg](#)

S3 URI

[s3://grupo4-bucket/s3images/products_images/1/img1-prod1.jpeg](#)

Amazon Resource Name (ARN)

[arn:aws:s3:::grupo4-bucket/s3images/products_images/1/img1-prod1.jpeg](#)

Entity tag (Etag)

[1e3ab44707fffb10afc1911a51e7cd2e](#)

Object URL

https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/1/img1-prod1.jpeg

MySQL Workbench

AWS connection x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- category_sequence
- characteristic_sequence
- characteristics
- cities
- city_sequence
- image_sequence
- images
- product_sequence
- products
- products_characteristics
- user_types

Views

Administration Schemas

Information

Table: images

Columns:

- id bigint PK
- title varchar(255)
- url varchar(255)
- product_id bigint

images

Limit to 1000 rows

1 • SELECT * FROM `digital-booking-grupo4`.images;

Result Grid

	id	title	url	product_id
1	1	img1-prod1	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/1/img1-prod1.jpeg	1
2	2	img2-prod1	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/1/img2-prod1.jpeg	1
3	3	img3-prod1	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/1/img3-prod1.jpeg	1
4	4	img4-prod1	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/1/img4-prod1.jpeg	1
5	5	img5-prod1	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/1/img5-prod1.jpeg	1
6	6	img1-prod2	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/2/img1-prod2.jpeg	2
7	7	img2-prod2	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/2/img2-prod2.jpeg	2
8	8	img3-prod2	https://grupo4-bucket.s3.us-west-1.amazonaws.com/s3images/products_images/2/img3-prod2.jpeg	2

images 1 x

Apply Revert