

Proyecto Parcial Cloud Computing

Integrantes:

- Luis Méndez
- Jeffry Hilario
- Mauricio Alvarez





Sistema de administración de empleados y tareas TAMBO

Hoy en día, muchas empresas están buscando herramientas que mejoren la productividad de su personal administrativo y empleados. Es por ello, que se requiere de software que organice sus recursos y se ajuste a sus necesidades. En ese marco, nace este proyecto que pretende solucionar estos problemas en aquellas organizaciones que buscan máxima eficiencia. Entre estas se encuentra la cadena de tiendas Tambo+, siendo nuestro proyecto a quien va dirigido.



Base de datos

- Para la implementación de la base de datos creamos una instancia de Amazon RDS para MySQL.
- Las RESTful Apis tiene acceso a la base de datos mediante sus credenciales.

```
database_path = 'mysql://{username}:{password}@{host}:{port}/{database}'.format(  
    'admin', # Username  
    'pt9vfW7tYQJpXI5U5o6C', # Password  
    'database-1.czmlp3xton0a.us-east-1.rds.amazonaws.com', # Host  
    '3306', # Port  
    'tambo' # Database  
)
```

Database: tambo

<input type="checkbox"/>	Table	Engine?	Collation?
<input type="checkbox"/>	administrador	InnoDB	utf8mb4_0900_ai_ci
<input type="checkbox"/>	empleado	InnoDB	utf8mb4_0900_ai_ci
<input type="checkbox"/>	tarea	InnoDB	utf8mb4_0900_ai_ci
	3 in total	InnoDB	utf8mb4_0900_ai_ci



amazon
RDS

Databases							
		<input checked="" type="checkbox"/> Group resources		Modify	Actions ▼	Restore from S3	Create database
<input type="text" value="Filter by databases"/>		<div>< 1 > </div>					
DB identifier	▲	Role ▼	Engine ▼	Region & AZ ▼	Size ▼	Status ▼	Actions ▼
database-1		Instance	MySQL Community	us-east-1c	db.t3.micro	Available	2 Actions

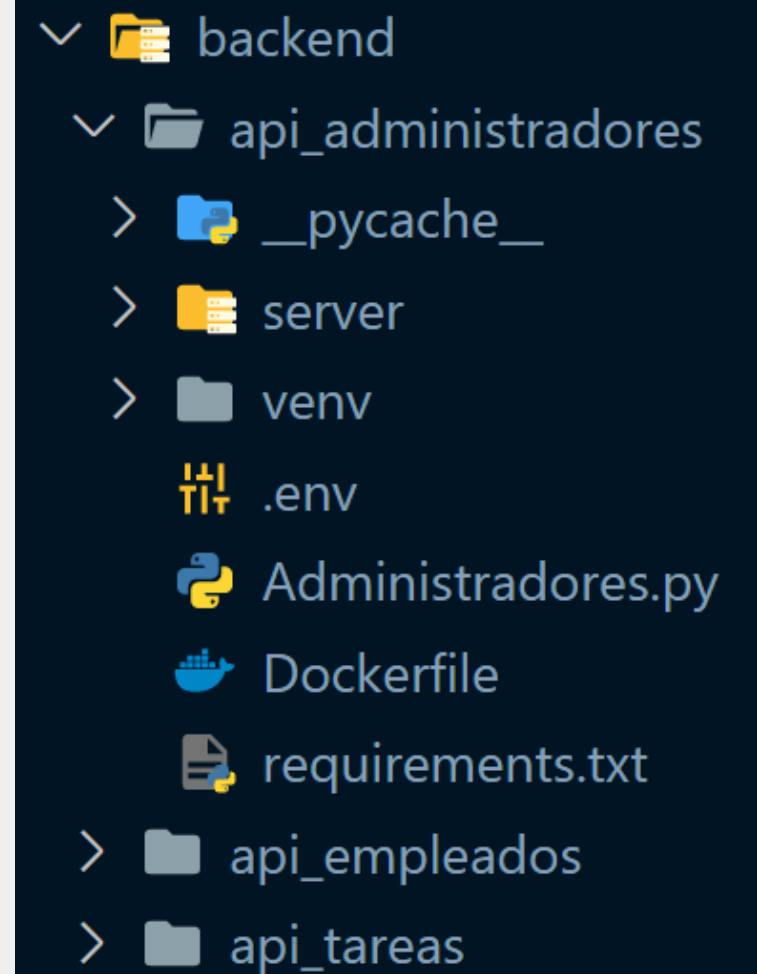
Backend

Microservicios (3)

- Api administradores (port: 5001): registro, login y consulta de administradores.
- Api empleados (port: 5002): registro, modificación, eliminación y consulta de empleados.
- Api tareas (port: 5003): asignación (creación), actualización y consulta de tareas.



Flask



```
@app.route('/login/log_admin', methods = ['POST'])
def log_admin():
    response = {}
    error = False

    try:
        dni_admin = request.get_json()['dni']
        password = request.get_json()['password']
        admin = Administrador.query.filter_by(dni_admin = dni_admin).first()

        if admin is not None and check_password_hash(admin.password, password):
            response['success'] = True
            response['admin'] = admin.format()
            response['token'] = jwt.encode({
                'dni_admin': dni_admin
            }, app.config['SECRET_KEY'])
        else:
```


Frontend



Microservicios (1)

- Frontend (port: 8080): despliegue de la UI y consumo de las Apis.

```
methods: {
  async handleSubmit(event) {
    event.preventDefault();

    const path = `${process.env.VUE_APP_ROOT_API}:5001/Login/Log_admin`;

    const response = await fetch(path, {
      method: "POST",
      body: JSON.stringify({
        dni: this.dni,
        password: this.password,
      }),
      headers: {
        "Content-Type": "application/json",
      },
    });

    let data = await response.json();
    console.log("Response: ", response);
    console.log("Data: ", data);

    if (data["success"]) {
```

```
✓ frontend
  > node_modules
  > public
  > src
  .gitignore
  babel.config.js
  Dockerfile
  jsconfig.json
  package-lock.json
  package.json
  README.md
  vue.config.js
  yarn-error.log
  yarn.lock
```

Deployment


```
Dockerfile
frontend > Dockerfile > ...
1 FROM node:18.16-alpine
2 WORKDIR /frontend
3 RUN yarn install
4 COPY . /frontend
5 EXPOSE 8080
6 CMD ["yarn", "serve"]
7
```

```
Dockerfile
backend > api_administradores > Dockerfile > ...
1 FROM python:3.11.3-alpine
2 RUN apk update && apk add --no-cache gcc
3 RUN apk add --no-cache python3-dev mariadb-conne
4 WORKDIR /api_administradores
5 COPY requirements.txt requirements.txt
6 RUN pip install -r requirements.txt
7 COPY . /api_administradores
8 EXPOSE 5001
9 CMD flask run --host=0.0.0.0 --port=5001
10
```

docker-compose build
docker-compose push


```
docker-compose.yml
docker-compose.yml
1 version: "3.3"
2
3 services:
4   api-administradores:
5     build: ./backend/api_administradores
6     image: luisfmendezl/api-administradores
7     ports:
8       - 5001:5001
9   api-empleados:
10    build: ./backend/api_empleados
11    image: luisfmendezl/api-empleados
12    ports:
13      - 5002:5002
14   api-tareas:
15    build: ./backend/api_tareas
16    image: luisfmendezl/api-tareas
17    ports:
18      - 5003:5003
19   frontend:
20    build: ./frontend
21    image: luisfmendezl/frontend
22    ports:
23      - 8080:8080
24
```

docker-compose up -d




luisfmendezl/frontend • 📄 4 • ☆ 0
By [luisfmendezl](#) • Updated 2 hours ago

Image




luisfmendezl/api-tareas • 📄 4 • ☆ 0
By [luisfmendezl](#) • Updated 2 hours ago

Image



luisfmendezl/api-administradores • 📄 4 • ☆ 0
By [luisfmendezl](#) • Updated 2 hours ago

Image



luisfmendezl/api-empleados • 📄 4 • ☆ 0
By [luisfmendezl](#) • Updated 2 hours ago

Image

```
Command Prompt - ssh -i lat x + v
~/app-tambo $ cat docker-compose.yml
version: "3.3"

services:
  api-administradores:
    image: luisfmendezl/api-administradores
    ports:
      - 5001:5001
  api-empleados:
    image: luisfmendezl/api-empleados
    ports:
      - 5002:5002
  api-tareas:
    image: luisfmendezl/api-tareas
    ports:
      - 5003:5003
  frontend:
    image: luisfmendezl/frontend
    ports:
      - 8080:8080
    env_file:
      - frontend.env
~/app-tambo $ cat frontend.env
VUE_APP_ROOT_API=http://54.221.252.38
~/app-tambo $ docker-compose up -d
Starting app-tambo_api-tareas_1 ... done
Starting app-tambo_api-administradores_1 ... done
Starting app-tambo_api-empleados_1 ... done
Recreating app-tambo_frontend_1 ... done
~/app-tambo $ |
```

Load balancer

EC2 > Load balancers

Load balancers (1)

Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter by property or value

< 1 >

⚙

Name

DNS name

State

VPC ID

Availability Zones

Type

lb-prod

lb-prod-735881176.us-east-1.elb.amazonaws.com

Active

vpc-0a1833df18f9c45ff

2 Availability Zones

app

Security Groups

Target groups

HTTP:8080

Forward to target group

- TG-Prod-8080: 1 (100%)
- Group-level stickiness: Off

HTTP:5001

Forward to target group

- TG-Prod-5001: 1 (100%)
- Group-level stickiness: Off

HTTP:5002

Forward to target group

- TG-Prod-5002: 1 (100%)
- Group-level stickiness: Off

HTTP:5003

Forward to target group

- TG-Prod-5003: 1 (100%)
- Group-level stickiness: Off

Manage tags

Edit inbound rules

Filter by property or value

< 1 >

⚙

Security group rule...

IP version

Type

Protocol

Port range

sgr-0b524696c27162...

IPv4

Custom TCP

TCP

8080

sgr-0293b5626b30e4...

IPv4

Custom TCP

TCP

5001

sgr-025a7a952693d7c...

IPv4

SSH

TCP

22

sgr-0284d2d124aeeb2...

IPv4

HTTP

TCP

80

sgr-0c9f2c2432bba7ced

IPv4

Custom TCP

TCP

5003

sgr-036f8dc2f3df7d410

IPv4

Custom TCP

TCP

8001

sgr-0780f38ac9b576bdb

IPv4

Custom TCP

TCP

8000

sgr-08ea1424bf8ff373b

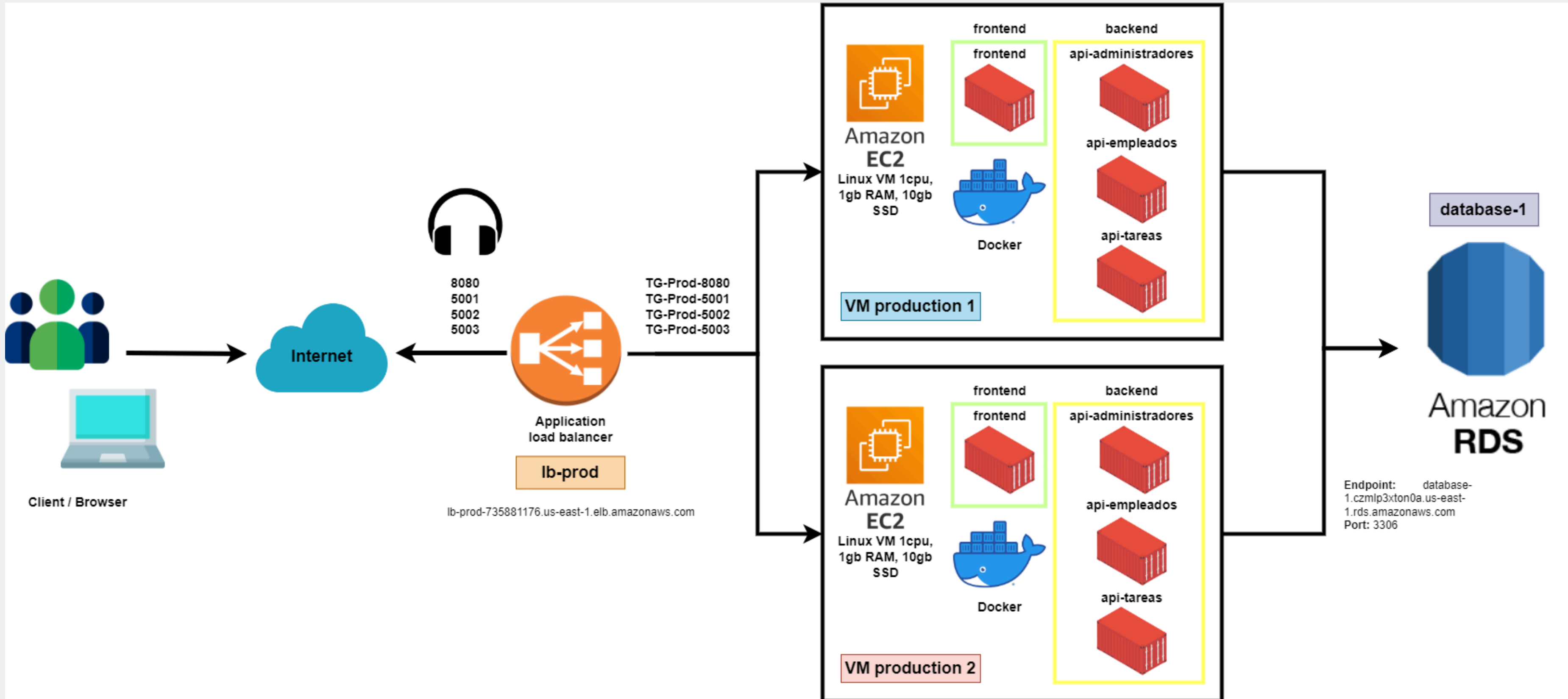
IPv4

Custom TCP

TCP

5002

Solution Architecture Diagram



Demostración en vivo con el load balancer...

<http://lb-prod-735881176.us-east-1.elb.amazonaws.com:8080/>

<input type="checkbox"/>	VM production 2	i-09e89ab318a031556	Running		t2.micro	2/2 checks passed	No alarms		us-east-1b
<input type="checkbox"/>	VM production 1	i-0479f17565ec0c8ce	Running		t2.micro	2/2 checks passed	No alarms		us-east-1a