Informe del Proyecto Final: Infraestructura de Dulcería

Manuel Alexander Serna Jaraba - 202259345

Adrián Felipe Velázquez Arias- 202259456

Edgar Fabián Rueda Colonia - 202259606

Universidad del valle – Tuluá

Ingeniería en Sistemas

INFRAESTRUCTURAS PARALELAS Y DISTRIBUIDAS-51

Profesor: Carlos Andres Delgado

26 de Diciembre de 2024

Entrega

Informe del Proyecto Final: Infraestructura de Dulcería

1. Introducción

Descripción del proyecto.

Este proyecto consiste en desarrollar una aplicación web para gestionar una dulcería. Incluye tres componentes principales: una base de datos PostgreSQL, un backend desarrollado con Flask y un frontend diseñado para la interacción del usuario. La infraestructura se implementa localmente con Docker Compose y se despliega en la nube para pruebas de rendimiento y disponibilidad.

Objetivos del proyecto

- Diseñar una arquitectura modular que incluya base de datos, backend y frontend.

- Implementar la infraestructura localmente con Docker Compose.

- Desplegar la aplicación en la nube, asegurando su escalabilidad y accesibilidad.

- Evaluar el rendimiento en un entorno local y en la nube.

2. Solución Local

Descripción de la arquitectura local

La infraestructura local se construye utilizando Docker Compose con tres servicios:

1. PostgreSQL (Base de datos): Administra la persistencia de datos de la dulcería.

2. Flask (Backend): Proporciona la lógica del negocio y se comunica con la base de datos.

3. Frontend: Permite a los usuarios interactuar con la aplicación.

Configuración del archivo docker-compose.yml

yaml

```
version: '3.9'

services:

  postgres:

    image: basedatos

    environment:

      POSTGRES_USER: administrador

      POSTGRES_PASSWORD: admin
```

```
    POSTGRES_DB: Dulceria
  ports:
    - "5432:5432"
  volumes:
    - postgres_data:/var/lib/postgresql/data
  networks:
    - dulceria_network
  flask-api:
  image: backend
  container_name: flask-api
  environment:
    DATABASE_URL: postgresql://administrador:admin@postgres:5432/Dulceria
  ports:
    - "8080:8080"
  depends_on:
    - postgres
  volumes:
    - uploads_data:/app/uploads
  networks:
    - dulceria_network
frontend:
  image: frontend
  container_name: frontend
  environment:
    API_URL: 'http://flask-api:8080'
  ports:
    - "80:80"
```

```
    depends_on:
      - flask-api
    networks:
      - dulceria_network
networks:
  dulceria_network:
volumes:
  postgres_data:
  uploads_data:
```

3. Solución en la Nube

Despliegue en la nube

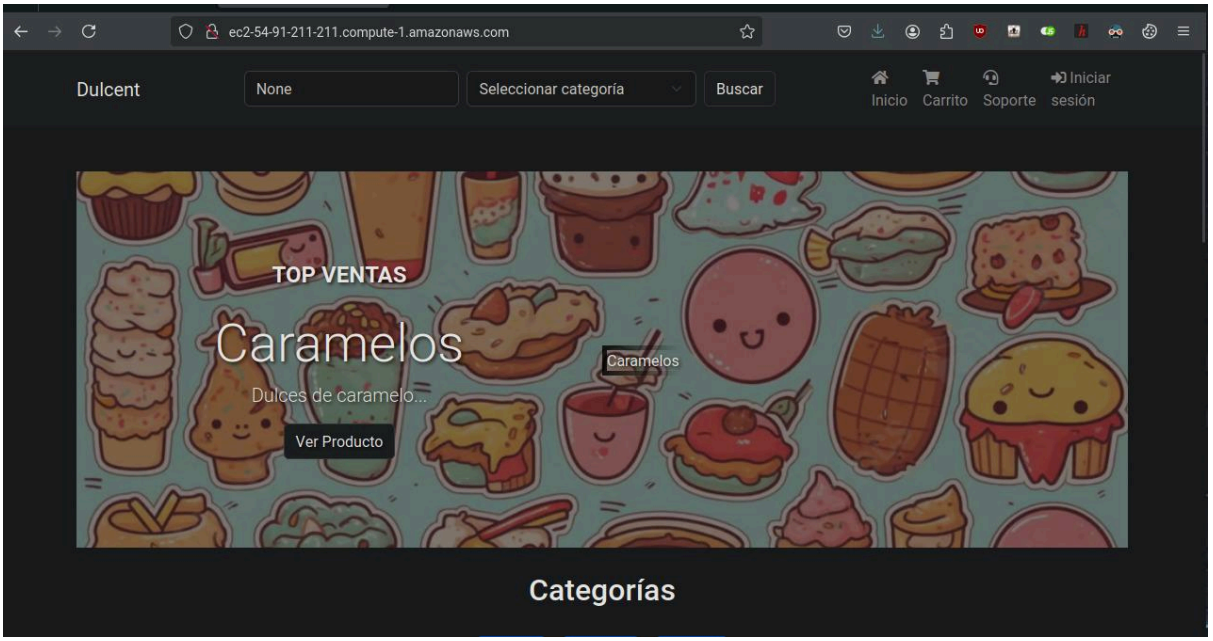
La aplicación se despliega en la nube utilizando un servicio de contenedores, como Azure Kubernetes Service (AKS). La configuración asegura alta disponibilidad y balanceo de carga para los servicios.

Pasos para el despliegue

1. Crear un grupo de recursos y un clúster de Kubernetes en Azure:

```
 docker build -t backend ./backend
docker build -t frontend ./frontend
docker build -t basedatos ./base_datos
docker swarm init
docker stack deploy --compose-file docker-compose.yml dulceria-stack
docker service scale subasta-stack_back=3
```

2. Subir las imágenes de los servicios a un registro de contenedores.



3. Implementar la infraestructura en Kubernetes mediante manifiestos YAML.

Enlace a la aplicación en la nube:

http://ec2-54-91-211-211.compute-1.amazonaws.com/login

4. Análisis y Conclusiones

Análisis de rendimiento

- Local: La aplicación muestra tiempos de respuesta rápidos debido a la proximidad de los servicios.

- En la nube: Incremento en la latencia, pero con mayor capacidad para manejar múltiples usuarios simultáneamente.

Retos enfrentados

- Configuración de redes seguras en la nube.

- Ajustes en las variables de entorno para diferentes entornos.

Reflexiones finales

- La separación de componentes con Docker Compose simplifica el desarrollo y las pruebas.

- Kubernetes en la nube proporciona escalabilidad y resiliencia a la aplicación.