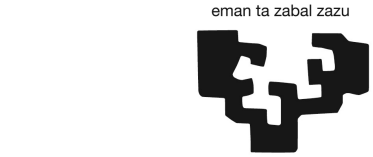

Administración de Sistemas

Ingeniería Informática de Gestión y Sistemas de Información

Coches.xyz

Autor:
Xabier Gabiña Barañano

1 de diciembre de 2023

Índice general

Introducción

Esta tarea tiene como objetivo explorar la implementación de una aplicación web con una base de datos incorporada, así como su despliegue utilizando Docker y Kubernetes. Estos dos entornos tecnológicos ofrecen soluciones fundamentales para abordar los desafíos actuales en el desarrollo de aplicaciones web al proporcionar un conjunto de herramientas y prácticas que simplifican la orquestación, escalabilidad y seguridad de dichas aplicaciones.

Docker, como plataforma de contenedores, permite a los desarrolladores encapsular aplicaciones y sus dependencias en entornos aislados. Esta metodología de encapsulamiento elimina problemas de compatibilidad, asegura la consistencia y facilita la migración de aplicaciones entre diferentes entornos. Además, Docker posibilita una gestión eficiente de recursos, contribuyendo así al despliegue efectivo de aplicaciones web.

Por otro lado, Kubernetes emerge como la solución definitiva para la orquestación de contenedores en clústeres. Desarrollada por Google, esta plataforma simplifica la administración y escalabilidad de aplicaciones web a gran escala. Kubernetes automatiza la distribución de contenedores, garantizando su disponibilidad, escalabilidad y equilibrio de carga, aspectos esenciales para aplicaciones que deben manejar un alto volumen de tráfico o que requieren una alta disponibilidad.

La combinación de Docker y Kubernetes se ha vuelto esencial para implementar aplicaciones web modernas, proporcionando una base sólida para el desarrollo, despliegue y gestión de sistemas en un entorno altamente dinámico y desafiante. Este enfoque promete optimizar los recursos, reducir los tiempos de despliegue y asegurar la confiabilidad y seguridad de las aplicaciones en un mundo cada vez más centrado en la nube.

Este informe se sumergirá en la creación de una aplicación web con una base de datos y su posterior despliegue mediante Docker y Kubernetes.

GitHub del proyecto

Descripción de la aplicación

El proyecto consiste en un entorno web enfocado a la compra y venta de coches de segunda mano. Una vez accedas a la pagina web podras ver un listado de vehiculos en venta, con los datos del vehiculo y del vendedor para poder comunicarte con el. Ademas, podras crearte una cuenta e iniciar sesion para poder publicar tu mismo un vehiculo y ponerlo a la venta.

Para almacenar tantos los usuarios como los anuncios que estos generan se ha implementado una base de datos PostgreSQL.

Dado que esto es un proyecto y por lo tanto no existen otros usuarios, se ha creado un programa en Python que cada cinco minutos genera un anuncio en la web. Para ello el programa se comunica con una API de modelos de coches y obtiene un modelo aleatorio con el que creara un anuncio.

Ademas, para facilitar el desarroyo y mantenimiento de la aplicacion se ha agregado un panel de administracion para la base de datos. En este caso se ha elegido Adminer, una herramienta web que permite gestionar diferentes sistemas de gestion de bases de datos entre los que se incluye PostgreSQL.

Para acabar, se ha implementado un gestor de claves llamado Vault. Este gestor nos permite almacenar de forma segura las claves de nuestro proyecto sin la necesidad de tenerlas escritas explicitamente en el codigo fuente de nuestra aplicacion.

Listado de las tareas realizadas

- Tareas obligatorias:
  - Desarroyo de una aplicación web. ✓
  - Creacion de una imagen Docker propia. ✓
  - Creacion de un entorno Docker Compose. ✓
- Tareas opcionales:
  - Creacion de un despliegue Kubernetes equivalente. ✓
  - Inclusion de imágenes adicionales. ✓
  - Utilizacion de funcionalidades Docker/Kubernetes no vistas en clase.

Explicaciones de las tareas realizadas

4.1. Desarroyo de una aplicación web.

El desarroyo de la aplicacion web se puede dividir en tres partes, el backend, el frontend y la base de datos. El frontend se ha desarroyado utilizando HTML y CSS. Para facilitar el desarroyo se ha utilizado Bootstrap, que nos permite crear paginas web responsive de forma sencilla. El backend se ha desarroyado utilizando PHP. PHP se utiliza para gestionar las conexiones de la base de datos como puede ser el registrar o inicio de sesion de un usuario, la creacion de anuncios y la obtencion y muestra de los datos de los anuncios.
Para almacenar la informacion de la web he usado PostgreSQL. PostgreSQL es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto.

Todo esto he creado dos imagenes Docker, una para el frontend y el backend y otra para la base de datos.

La primera imagen se ha creado a partir de la imagen oficial de PHP y Apache. Para ello he instalado las dependencias necesarias para el funcionamiento de la aplicacion y la comunicacion con la base de datos y tambien he copiado todos los arhivos necesarios para ello con los permisos correctos. Esta aplicacion esta disponible en el puerto 80.

La segunda imagen se ha creado a partir de la imagen oficial de PostgreSQL. En este caso lo unico que se ha hecho es copiar una archivo .sql con la estructura de la base de datos y con los datos de prueba. Esta base de datos esta disponible en el puerto 5432. Tambien se han definido las variables de entorno necesarias para la creacion de la base de datos y el usuario. Para acabar he desactivado la cuenta root como medida de seguridad.

4.2. Creacion de una imagen Docker propia.

Dado que la pagina web es un proyecto y por lo tanto no existen otros usuarios, he decidido crear un programa que cada minuto genera un anuncio en la web. Para la creacion de la imagen he utilizado la imagen oficial de Python Slim. He elegido esta imagen dado el reducido tamaño de la misma. Para la creacion de la imagen he copiado el archivo .py con el programa y he instalado las dependencias necesarias para su funcionamiento. El funcionamiento del programa es el siguiente:

1. El programa obtiene las claves necesarias para la comunicacion de la API de Vault.
2. Con las claves obtenidas se identifica con la API de carapi.
3. Se obtiene un modelo aleatorio de coche.
4. Se buscas la marca a la que pertenece el modelo.
5. Se escribe el anuncio directamente en la base de datos.
6. Se espera un minuto y se repite el proceso.

De esta forma se generan anuncion de forma automatica y se simula el funcionamiento de la web manteniendo ademas la seguridad gracias a no dejar implicitas las claves.

4.3. Creacion de un entorno Docker Compose.

Para orquestar el funcionamiento de la aplicacion con las diferentes imagenes que ya he nombrado y la que queda por nombrar he utilizado Docker Compose.

4.4. Creacion de un despliegue Kubernetes equivalente.

Aviso: Para la realizacion de esta tarea he utilizado Minikube. Minikube es una herramienta que nos permite crear un cluster de Kubernetes en local. El formato del .yaml del Ingress no es exactamente igual en Kubernetes de GCP que el de Minikube, ten esto en cuenta si vas a desplegarlo en GCP. Tambien tener en cuenta que Minikube no trae los Ingress activados por defectos y debe ser activados mediante `minikube addons enable ingress`

4.5. Inclusion de imágenes adicionales.

4.6. Utilizacion de funcionalidades Docker/Kubernetes no vistas en clase.

Declaración sobre uso de asistentes virtuales

Para la realizacion de este proyecto se ha utilizado dos asistentes virtuales, GPT-3.5 y GitHub Copilot. Ambos se han usado principalmente para la tarea de picado de codigo resolucion de errores.

Bibliografia

- GPT-3.5. (2023). Respuesta a una pregunta sobre PHP. OpenAI. https://www.openai.com/
- GitHub Copilot. (2022). Autocompletado. GitHub. https://github.com/features/copilot
- Docker. (2021). Documentación. Docker. https://docs.docker.com/
- Kubernetes. (2021). Documentación. Kubernetes. https://kubernetes.io/docs/home/
- PHP. (2021). Documentación. PHP. https://www.php.net/docs.php
- PostgreSQL. (2021). Documentación. PostgreSQL. https://www.postgresql.org/docs/
- Apache. (2021). Documentación. Apache. https://httpd.apache.org/docs/
- Minikube (2023). Documentación Minikube. https://minikube.sigs.k8s.io/docs/handbook/