Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

SOA: Arquitectura Orientada a Servicios Aplicada a Sistemas Emergentes

Profesora: Ing. Alejandra Bolaños Murillo

PROYECTO I – SENTIMENT ANALYSIS INFRASTRUCTURE

Integrante: Calderón Yock, Sebastián

Alajuela 2022

Introducción

El proyecto de Sentiment Analysis Infrastructure consiste en un MVP para una aplicación de una empresa, que permita analizar los sentimientos de los empleados al recibir buenas noticias. La idea es, eventualmente, utilizar la herramienta para comunicar mejor los resultados mensuales de ganancias de la compañía.

Para el MVP se desarrolló la infraestructura básica del proyecto, se seleccionaron los componentes que van a formar parte de esta infraestructura y se definieron sus interacciones. El objetivo fundamental es proveer a los futuros desarrolladores del proyecto con una infraestructura estable, por medio de la automatización en las etapas del desarrollo de la aplicación. Las etapas automatizadas son la integración, la distribución y la implementación continuas.

Respecto a la aplicación, esta cuenta con un elemento serverless (Cloud Function), el cual ejecuta el código, y un Cloud Bucket, el cual es utilizado para almacenar la imagen a ser procesada. La aplicación toma la imagen desde el Cloud Bucket, y consulta el Cloud Vision API. Este API analiza la imagen y parametriza las emociones identificadas, para luego imprimir las emociones como resultado. Sumado a lo dicho, todo cambio a la aplicación que ha sido aprobado, debe inicializar el despliegue inmediatamente y la distribución de la aplicación al ambiente de producción.

Set de Tecnologías

En este apartado se listan las tecnologías utilizadas en el proyecto y su aplicación en el mismo. Los componentes que integran estas tecnologías, así como sus responsabilidades y conectores, se detallan en la siguiente sección del documento.

- **GitHub:** Versionador de código que permite a los desarrolladores interactuar con el proyecto y que contiene el código de la aplicación. Los desarrolladores crean ramas, de acuerdo con una branching strategy definida y explicada más adelante, para implementar features, bug-fixes y enhancements. Por medio de un Action, invoca el build de la aplicación una vez que se haya hecho un pull-request a la rama main.
- **Google Cloud:** Proveedor de servicios de computación en la nube; además del alojamiento de la aplicación. Se utiliza Google Cloud Platform para el alojamiento en la nube de la aplicación, Cloud Vision para análisis de rostros, Cloud Build para el build del código, Cloud Buckets para almacenamiento y Cloud Functions para ejecutar funciones con base en eventos.

- **HashiCorp Terraform:** Herramienta usada para gestionar la infraestructura y su configuración en Google Cloud, por medio de un proveedor o API. [1]
- **Python 3:** Lenguaje usado para desarrollar la funcionalidad del sistema.

Componentes y responsabilidades

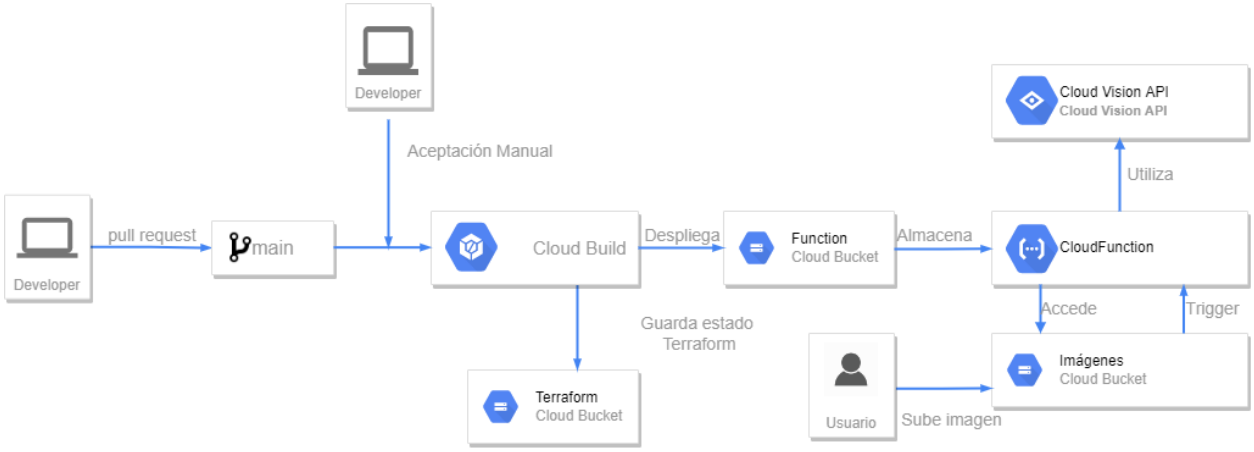
- **Cloud Build:** Es un servicio de Google Cloud que ejecuta compilaciones (builds) de los proyectos en una serie de pasos. Este servicio importa del repositorio del proyecto en GitHub el código fuente, prepara el ambiente e instala todas las dependencias según una configuración provista en un archivo YAML y compila el código, todo en un contenedor [2]. La conexión con GitHub se da por medio de la GitHub App Google Cloud Build, instalada en el repositorio del proyecto. El conector con el proyecto de Google Cloud Platform es Cloud Build API.
- **Cloud Storage Buckets:** Son contenedores que almacenan los datos. Todo lo que se almacena en Google Cloud Platform (GCP) se hace por medio de Buckets [3]. En el presente proyecto se utilizan Buckets para almacenar las imágenes que cargan los clientes, para almacenar el código de la Cloud Function y para almacenar una imagen de la aplicación en Terraform. Se conecta al proyecto en GCP por medio del servicio Cloud Storage y el API Cloud Storage.
- **Cloud Functions:** Son un entorno de ejecución serverless para compilar y conectar servicios de la nube. Permiten escribir funciones simples de propósito único que están ligadas a eventos de la infraestructura en la nube y servicios. Estas funciones pueden escribirse en distintos lenguajes, como Python 3 [4]. Las Cloud Functions son responsables del procesamiento de imágenes cuando un usuario carga una imagen de un rostro a un bucket y de retornar el análisis realizado por Cloud Vision API. El conector de Cloud Functions con el proyecto de GCP es por medio de Cloud Functions API.
- **Cloud Vision:** Es un servicio que provee la capacidad de detección y análisis visual a las aplicaciones, por medio de modelos de machine learning [5]. En el proyecto se utiliza para recibir una imagen de un rostro, analizarla y determinar, paramétricamente, qué emociones exhibe el rostro. Se conecta al proyecto por medio de Cloud Vision API. Su conexión con la funcionalidad de la aplicación es por medio de la Cloud Function.
- **GitHub:** Es un servicio web basado en la nube que permite el almacenamiento de código, su gestión y versionamiento. En el proyecto se utiliza para almacenar el código fuente, dar seguimiento a las versiones de este, gestionar la interacción de los desarrolladores por medio de una estrategia de branching definida (referirse a la sección de Estrategia de Branching) y para invocar el

Build en Google Cloud cuando se detecta un pull-request a la rama main. Se conecta al proyecto de GCP por medio de la GitHub App Google Cloud Build y el GitHub Action mencionado.

Diagrama de arquitectura

La arquitectura del sistema con todos sus componentes se muestra en la Figura 1. En esta ocurren dos flujos de interacción, el de los desarrolladores y los usuarios.

Figura 1 - Arquitectura del sistema



Respecto al flujo de los desarrolladores, cuando un desarrollador realiza un pull-request a la rama main, acciona inmediatamente el proceso de build en Google Cloud por medio de un GitHub Action. Cloud Build obtiene el código fuente de GitHub, por medio de Terraform hace el plan y almacena el estado en un bucket, luego despliega la aplicación a una cloud function.

En el caso de los usuarios, cuando cargan una imagen al bucket respectivo, se acciona la cloud function que llama a Cloud Vision API, le provee de la imagen cargada, recibe la respuesta de dicho API y retorna las emociones identificadas.

Decisiones del proyecto

En el siguiente apartado se desarrollará el fundamento de las diversas decisiones en términos de tecnologías , metodologías y procedimientos utilizados en este proyecto. Se procederá a explicar en qué consiste y la razón de su uso, así como detalles de su implementación.

Estrategia de Branching:

La estrategia de branching consiste en lo siguiente:

- main es la rama principal, sinónimo de prod, donde siempre debe estar una versión funcional del código. Esta es la rama que se despliega.
- develop es la rama basada en main con la que los desarrolladores interactúan.
- Los desarrolladores deben trabajar en branches generadas a partir de develop, con la siguiente sintaxis según sea el caso:
    - **feat/**<feature-to-implement>: ramas para implementar un nuevo feature a la aplicación.
    - **fix/**<bug-to-fix>: ramas para solucionar un bug o error en el código.
    - **improv/**<improvement-to-make>: ramas para realizar alguna mejora en el código.
- Para hacer merge de una rama en main, se deben seguir los siguientes pasos cuando se hayan realizado todos los cambios:
    1. **git rebase -i HEAD~x** (con x la cantidad de commits propios +1)
    2. **git squash**
    3. **git fetch**
    4. **git rebase origin/master**
    5. **git push -force**
    6. Luego en la página web de GitHub se dirige a la rama que acaba de subir y realiza un pull-request.

Pipeline de CI/CD:

Para este proyecto utilizamos Google Cloud como proveedor de servicios de compilación en la nube, en términos de beneficios y funcionalidades son semejantes y a pesar de que Jenkins es gratuito y es bastante más famoso que cloud [6]. Se decidió utilizar Google Cloud para evitar posibles incongruencias entre los servicios y demás complicaciones. Para este proyecto Cloud build asocia un trigger a la rama main de nuestro repositorio, por lo que cada vez que ocurre un push a main, GB ejecuta un archivo de configuración con las especificaciones de la compilación.

Pruebas de código:

Para este proyecto se decidió utilizar Pytest, primeramente por ser una opción completamente gratuita y segundo por la facilidad de instalación e implementación de pruebas unitarias, otras librerías pueden ser más complejas de comprender, o de instalar y ejecutar, mientras que pytest se resumen en dar la orden de instalación y de ejecución.

Referencias

[1] Hashicorp. Introduction to Terraform. [Online]. Available: https://www.terraform.io/intro

[2] Google Cloud. Cloud Build Documentation. [Online]. Available: https://cloud.google.com/build/docs

[3] Google Cloud. Cloud Storage Documentation. [Online]. Available: https://cloud.google.com/storage/docs/key-terms

[4] Google Cloud. Cloud Functions Overview. [Online]. Available: https://cloud.google.com/functions/docs/concepts/overview#:~:text=Google%20Cloud%20Functions%20is%20a,event%20being%20watched%20is%20fired.

[5] Google Cloud. Cloud Vision Documentation. [Online]. Available: https://cloud.google.com/vision/docs

[6] Stackshare(2022).*Google Cloud BuildvsJenkins.* https://stackshare.io/stackups/google-cloud-build-vs-jenkins