Proyecto 1: Feelings Analyzer

Luis Daniel Prieto Sibaja
Email: prieto.luisdaniel@gmail.com
Área Académica de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

***Abstract*—The following is a report for the Computer Architecture course of the Computer Engineering career, detailing the development of a simple project which consists of a Google Cloud service, the one which receives an image locally in a bucket, analyzes the image and gives back an analysis over the emotional response of the faces in said image. For this task, the project employs several cloud computing technologies, including the likes of Google Cloud Storage, Cloud Build, Cloud Functions and Terraform.**

***Palabras clave*—Cloud, pipeline, CI/CD, function, Terraform.**

I. Introducción

La presente es un documento correspondiente a un ensayo requerido para la tercera investigación del curso CE5508: SOA41D: Arquitectura Orientada a Sistemas Emergentes. El motivo de la investigación es detallar la documentación requerida para los

El documento se divide en las siguientes secciones: comienza con la presente introducción, el desarrollo del mismo detalla en 3 subsecciones la documentación del mismo, sean estos componentes, conectores y los diagramas pertinentes en la arquitectura de sistemas.

II. Desarrollo

El proyecto consiste en el desarrollo de un software empresarial que permita analizar los sentimientos de los empleados al recibir buenas noticias mediante un algoritmo de reconocimiento facial, el cual pueda realizar análisis sentimental a las imágenes de los empleados que recibe.

El software empresarial, como se detalló previamente, va a recibir la fotografía de algunos empleados de la empresa, determinando mediante un script y un elemento serverless (cloud computing), la emoción que el empleado refleja, mediante consultas a la API de GCP Vision. El sistema retorna esta emoción como resultado de las llamadas. El sistema aplica los conceptos de CI/CD para proveer a los futuros desarrolladores del proyecto, una infraestructura estable, con el uso de la automatización en sus etapas.

A. Componentes y conectores

Para desarrollar el presente proyecto se tomarán en cuenta para su elaboración los siguientes componentes:

- Python: Se utiliza Python para realizar el script del algoritmo que va a consultar a la GCP Vision API, el cual recibe una foto del empleado a consultar, y realiza las peticiones correspondientes para que el API retorne los resultados.
- GCP Vision API: Es la API de Google Cloud que recibe y ejecuta las peticiones del Cloud Function (script de Python) para realizar el análisis de la imagen. El API contiene todas las funciones que son importadas al script de Python para realizar las llamadas correspondientes y ejecutar los algoritmos de reconocimiento facial.
- Terraform: Es el componente que se encarga de realizar toda la configuración y ejecución del proyecto. Cumple con los requisitos del proyecto de desarrollar un ambiente IaC (Infrastructure as Code). Contiene los archivos que realizan la configuración y ejecución del backend, buckets necesarios, así como la Cloud Function a utilizar. De esta forma es el componente esencial que se encarga de definir, como su nombre lo dice, la infraestructura del proyecto.
- Cloud Function: Es la función en la nube que integra los conceptos de cloud computing con el desarrollo de software ya conocidos. Es una función alojada en un servicio de la nube, dentro del proyecto remoto el cual maneja el desarrollador, que puede recibir entradas y salidas, mediante los buckets definidos por el desarrollador. Estas entradas son entonces tomadas como entradas del script que aloja y puede ejecutar, para que el desarrollador pueda realizar pruebas y obtener resultados. De esta forma integra el Google Cloud, el Vision API y Python.
- Cloud Build: Es el servicio que provee Google Cloud Platform para realizar integración continua y despliegue continuo. Se utiliza para configurar el pipeline que el proyecto utilizará como conector principal.
- Github: Es el repositorio de código del proyecto. Aloja los archivos de código del proyecto en una instancia local de la máquina del desarrollador para su continuo desarrollo, siendo el responsable de llevar el control de versiones de los archivos de código de Terraform, Google Cloud e incluso el pipeline.

Para integrar satisfactoriamente todo el proyecto, este cuenta con los siguientes conectores, los cuales integran las diferentes partes del proyecto.

- Pipeline: Es el conector principal del proyecto, siendo la encarnación de los principios de CI/CD (integración continua y despliegue continuo). El pipeline es configurado desde un archivo YAML, el cual contiene la configuración de la infraestructura del proyecto. Este archivo YAMl es capaz de apuntar a los directorios en los que se encuentra la infraestructura de Terraform, y puede realizar tanto el plan de Terraform como su ejecución. Adicionalmente,

el pipeline es el encargado de conectar el repositorio en Github con Cloud Build, de esta manera todos los recursos que se encuentran en Github son enviados al Cloud Build para su consiguiente compilación.

- Activadores: En el proyecto se manejan varios activadores, estos son servicios que provee GCP para cuando ocurre cierta acción, entonces ejecuten otra acción en el proyecto de forma automática. El activador de Cloud Build es el conector entre el pipeline y el proyecto alojado en Cloud Build, puesto que este determina si la compilación es correcta. De lo contrario, el proyecto no puede ejecutarse en Google Cloud Platform.

B. Diagrama de arquitectura

Para el diagrama de arquitectura, este se maneja a tres niveles diferentes, basándose en la arquitectura de modelado C4. El primero es el diagrama de contexto del sistema, el cual simplemente muestra la interacción del usuario con el sistema. El segundo es el diagrama de contenedores, el cual muestra esta misma interacción pero a nivel interno con los distintos componentes del sistema. Finalmente, se tiene un diagrama de componentes que describe de forma específica los componentes de los contenedores y la interacción entre ellos.

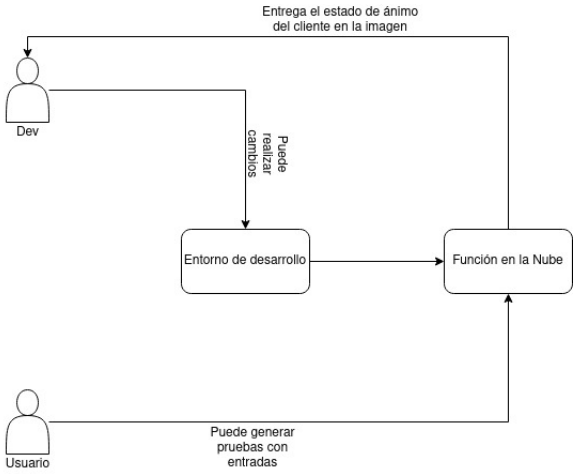

Fig. 1. Diagrama de contexto del sistema

C. Documentación de decisiones

A lo largo del proyecto se realizaron distintas decisiones sobre como debía de realizarse el mismo, incluyendo varias que se terminaron descartando. En primera instancia, los requerimientos para el proyecto son los siguientes, por lo que su integración era requerida. De esta misma forma, se tomaron en cuenta los siguientes aspectos para su elección:

- Terraform: Es el encargado de realizar la infraestructura del proyecto y de configurar y ejecutar todos los componentes del mismo. Fue escogido por su fácil integración con el pipeline de Cloud Build, siendo capaz de ser llamado simplemente mediante un archivo YAML, para posteriormente configurar todos los recursos en la nube.

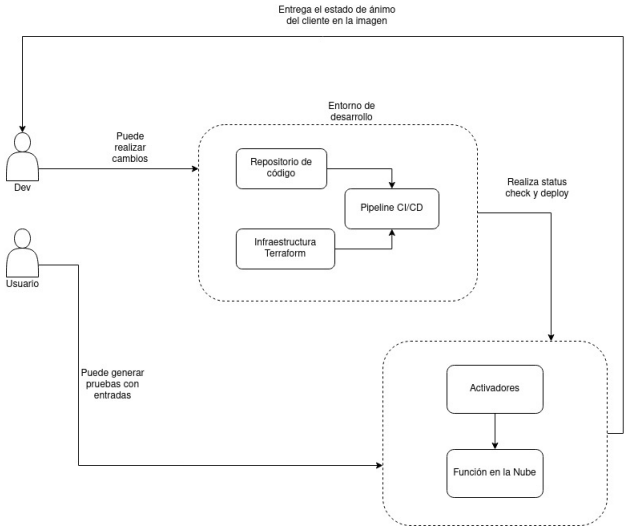

Fig. 2. Diagrama de contenedores del sistema

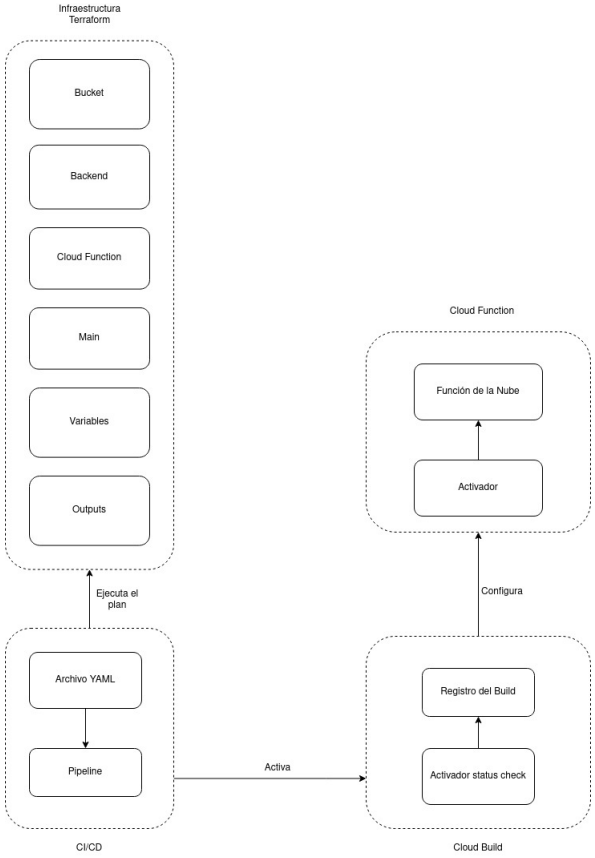

Fig. 3. Diagrama de componentes del sistema

- Github: Es el repositorio que va a realizar el control de versiones de todo el proyecto. Fue escogido por su fácil y conocida integración con los diferentes servicios que debe integrar, siendo estos el pipeline que ejecuta Terraform, los activadores que realizan la compilación del código en la nube y la consola del sistema operativo. Además es una herramienta conocida en la comunidad de desarrolladores y de fácil uso, costo y acceso. En buena teoría debería

tener una rama de producción como de desarrollo, sin embargo por razones de tiempo y facilidad de uso se trabajó sobre una única rama.
- Vision API: Es el encargado de realizar el análisis de las imágenes que se le envían mediante los buckets. Fue escogido por su facilidad de uso, puesto que este ya tiene la mayoría de funciones implementadas con amplia documentación al respecto.

De esta misma forma, algunas decisiones fueron de libre escogencia para el desarrollador del proyecto y futuros desarrolladores. Las mismas se detallan a continuación:

- Cloud Build: Es el encargado de realizar la configuración del pipeline CI/CD del proyecto y de garantizar su uso. Fue escogido sobre la VM de Jenkins por su fácil integración con el resto de servicios de Google Cloud Platform, así como su facilidad de uso e integración con Github, con el cual tiene muy buena cooperación como ambiente de desarrollo con amplia documentación al respecto.
- Cloud Function: Es el componente funcional del proyecto alojado en Google Cloud Platform. Para su ejecución simplemente se le deben entregar las entradas y salidas necesarias mediante buckets o alguna entrada manual. Fue escogido sobre App Engine por las mismas razones que Terraform, su fácil integración con el sistema y servicios de GCP y la facilidad de uso que le dan los desarrolladores.
- Python: Es el encargado de realizar el análisis de las imágenes que se le envían mediante un script alojado en el Cloud Function con llamadas a la Vision API. Fue escogido por su facilidad de uso, puesto que este ya tiene la mayoría de funciones implementadas con amplia documentación al respecto.

En el set de tecnologías detallado en el Readme.MD se puede leer más brevemente por las decisiones tomadas.

IX. Referencias

[1] Google Cloud. (2022). Vision API (y sus tutoriales derivados). Recuperado de: https://cloud.google.com/vision?hl=es
[2] Google Cloud. (2022). Vision API (y sus tutoriales derivados). Recuperado de: https://cloud.google.com/docs/ci-cd
[3] Beranek, L. (2021). CI/CD with Google Cloud Build, Terraform, and Github. Recuperado de: https://xbery.medium.com/continuous-integration-with-google-cloud-build-terraform-and-github-8e636d7d0df4
[4] Bisbé, L. R. (2019). Creando diagramas de manera efectiva con el modelo C4. Recuperado de: https://rlbisbe.net/2019/03/29/creando-diagramas-de-manera-efectiva-con-el-modelo-c4/