Proyecto Final de Cloud Computing

Integrantes:

José Miguel Guzmán Chauca
Fabrizio Paul Rosado Málaga

1 Introducción

El proyecto consiste en utilizar las herramientas de Cloud Computing para crear un servicio de ejecución de código. Un servicio parecido al provisto por Replit que tiene como entrada el código que escribe el usuario para luego ser evaluado y ejecutado. A partir del resultado que devuelva la evaluación, se imprimirá la salida del código.

Con lo aprendido en el curso se recrea un servicio parecido al de Replit que toma como parámetro el código en el lenguaje de programación Python que el usuario introduce. Se procede a crear nuestra versión con las siguientes herramientas: Google Cloud Processing (GCP), HTML, CSS, Python, NGINX, JavaScript, Kubernetes, Docker.

2 Estructura del Proyecto

Partimos de crear un clúster en Google Kubernetes Engine (GKE), dentro del cual se crean las imágenes para la interfaz de usuario, y otra para el servidor según las indicaciones iniciales. A continuación se muestra un gráfico con los componentes de nuestro proyecto.

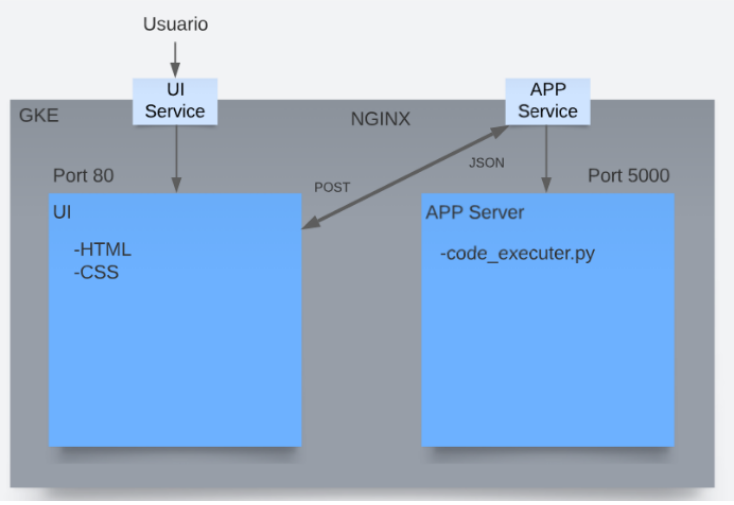

Figure 1: Componentes del proyecto.

El usuario se comunica a través del servicio de la interfaz. Esta conexión se logra con el load balancer que se encarga de volver el servicio elástico y disponible al público. Dentro del clúster se encuentran los contenedores y los pods necesarios para la virtualización de los archivos del proyecto.

El contenedor UI escucha por el puerto 80 y se comunica con el load balancer para recibir el código del Usuario. Luego, a través del método POST y el uso de NGINX se comunica con el servicio de la aplicación.

El contenedor de la aplicación ejecuta el código y devuelve la salida al servicio UI a través de NGINX en forma de JSON.

El clúster GKE cuenta con un Horizontal Pods Autoscalling que se encarga de realizar el autoescalado de los pods que se van a poner a disposición de los usuarios. Se determina un mínimo, máximo de instancias, y métricas de autoescalado, en este caso utilizamos el porcentaje de uso del CPU de las instancias para inicializar otras instancias. Este procedimiento se realiza para cada despliegue.

Y finalmente el servicio UI muestra la salida del código, con un mensaje si se ejecutó correctamente o si hubo algún problema con la ejecución.

3 Resultados Obtenidos

El proyecto realiza su cometido. Para utilizar la funcionalidad de introducir código debemos conectarnos a la dirección del hostname que provee GCP: http://34.139.106.232/ y nos redireccionará a la interfaz como se muestra en la siguiente imagen:

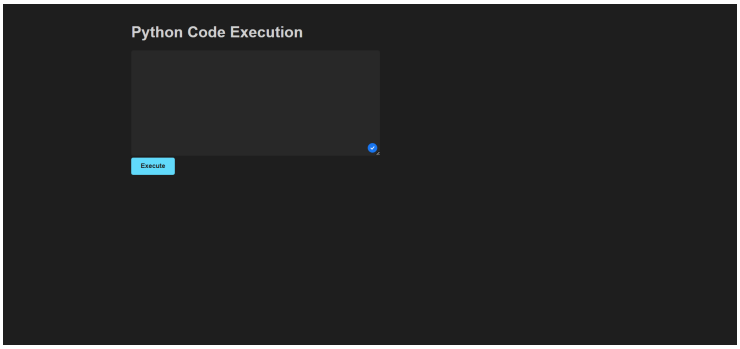

Figure 2: Interfaz de entrada

Luego de introducir el código en el formulario y presionar el botón "Execute" nos redireccionará a la siguiente interfaz mostrando la salida y el resultado de la ejecución.

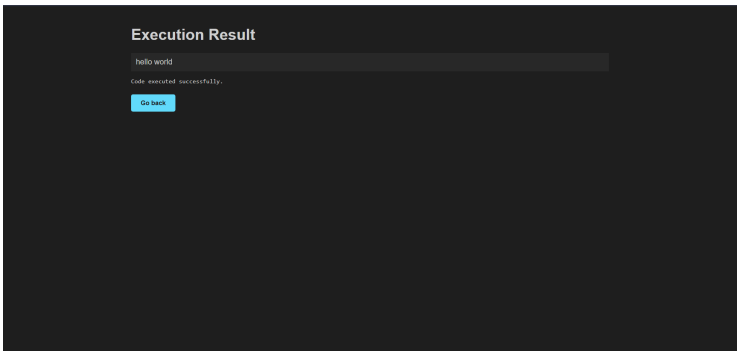

Figure 3: Interfaz de salida

4 Prueba de estrés del Proyecto

ApacheBench proporciona estadísticas sobre las solicitudes, incluyendo el número de solicitudes por segundo, el tiempo de respuesta promedio y más. Estos resultados nos ayudan a evaluar el rendimiento y el comportamiento de escalado automático del endpoint.

Server Software: nginx/1.21.6
Server Hostname: 34.139.106.232
Server Port: 80

Document Path: /
Document Length: 490 bytes

Concurrency Level: 1000
Time taken for tests: 95.948 seconds
Complete requests: 10000
Failed requests: 21
(Connect: 0, Receive: 0, Length: 21, Exceptions: 0)
Total transferred: 7218432 bytes
HTML transferred: 4892160 bytes
Requests per second: 104.22 [#/sec] (mean)
Time per request: 9594.801 [ms] (mean)
Time per request: 9.595 [ms] (mean, across all concurrent requests)
Transfer rate: 73.47 [Kbytes/sec] received

Connection Times (ms)
min mean[+/-sd] median max
Connect: 0 1683 2349.0 1096 32769
Processing: 123 1745 3264.2 914 63091
Waiting: 123 1072 1471.6 696 25659
Total: 267 3427 4053.4 2174 63091

Percentage of the requests served within a certain time (ms)
50% 2174
66% 2684
75% 4019
80% 4804
90% 6785
95% 9449
98% 15510
99% 17988
100% 63091 (longest request)

Estas estadísticas nos permiten evaluar la capacidad de respuesta de nuestro endpoint bajo una carga de 1000 solicitudes concurrentes. Podemos observar que durante la prueba se realizaron un total de 10000 solicitudes, de las cuales 21 fallaron. El tiempo promedio por solicitud fue de 9.595 ms, y el servidor fue capaz de atender aproximadamente 104.22 solicitudes por segundo.

Además, los tiempos de conexión y procesamiento se encuentran dentro de un rango aceptable, con una mediana de 1096 ms para la conexión y 914 ms para el procesamiento. También podemos observar que el porcentaje de solicitudes atendidas en un tiempo determinado sigue una distribución gradual, siendo la solicitud más larga de 63091 ms.

Estos resultados nos ayudan a evaluar la capacidad de nuestro proyecto para manejar una carga de solicitudes y nos permiten tomar decisiones sobre el escalado automático del endpoint en función del rendimiento obtenido durante la prueba de estrés.

5 Instrucciones para Replicar el Proyecto

- Descargar los archivos del repositorio.
- Crear un clúster GKE en GCP.
- Descargar e instalar Docker.
- Descargar la API de Google Cloud.
- Configurar Docker y conectarlo al clúster de GKE.
- Realizar las configuraciones necesarias en los archivos.
- Subir los archivos modificados a Docker.
- Realizar el despliegue del proyecto utilizando los comandos de Kubernetes.

6 Repositorio:

En el siguiente enlace se puede acceder al repositorio que contiene los archivos necesarios para poder ejecutar el proyecto.

https://github.com/FPRM2021/Cloud-Python-App