AWS - Flask API CI/CD

En este taller vas a poner en marcha una API Flask de Python dentro de un entorno Cloud como es el de Amazon. Desarrollarás tu API en local y la desplegarás en la nube, pudiendo acceder a la misma desde cualquier lugar.

Todos los recursos de AWS que se van a crear en este taller son gratuitos en la Free Tier de AWS durante los dos primeros meses. No obstante, se recomienda eliminarlos después del taller para evitar futuros cargos.

Seguiremos los siguientes pasos:

1. Configurar un entorno virtual de Python en local, con los paquetes necesarios para desarrollar una API Flask.
2. Crearemos una API Flask
3. Implementación de la API en AWS mediante Elastic Beanstalk.
4. Eliminación de los recursos.

Requisitos

Para el desarrollo de este taller necesitaremos:
- Cuenta gratuita de AWS
- Tener Python instalado en local
NOTA: Para el taller se recomienda crear una carpeta nueva, que llamaremos eb-flask, en
la ruta que desees del ordenador.

Crear un entorno virtual de Python

Cuando se realiza un despliegue en un entorno productivo no es necesario que vaya con
todos los paquetes que has utilizado en tus analíticas, por lo que habrá que crear un entorno de Python desde 0, con los paquetes necesarios. Para ello, partiremos del intérprete de Python 3 que tengamos instalado, y añadiremos las librerías que consideremos.

Windows

Abre el cmd y utiliza el siguiente comando en tu carpeta de trabajo donde crearemos nuestro entorno virtual:

```
python -m venv virt
```

La ruta es dónde queremos que se cree el nuevo entorno virtual. En este caso “virt” sería su
nombre. Accede al entorno virtual:

```
virt\Scripts\activate
```

Debería aparecer delante de la línea de comandos el nombre del entorno virtual.

MAC

Abrimos el terminal y ejecutamos:

```
python -m venv virt
```

La ruta es dónde queremos que se cree el nuevo entorno virtual. En este caso “virt” sería su nombre. Accede al entorno virtual:

```
source virt\Scripts\activate
```

Los siguientes pasos ya son comunes a Windows y MAC.

Después, mediante “pip freeze” veremos los paquetes que tenga instalados. De momento, ninguno.

Ahora instalamos flask:

```
pip install flask
```

Comprueba de nuevo mediante “pip freeze” que flask está instalado. Ten en cuenta que flask tiene otras dependencias, por lo que no será el único paquete instalado.

Guarda la salida de los paquetes en un archivo llamado “requirements.txt”, este archivo le servirá a Elastic Beanstalk para instalar los paquetes en su entorno virtual. El entorno virtual que acabamos de crear no lo vamos a subir a la nube. Es el propio recurso de Amazon el que lo despliega a partir del “requirements.txt” que hemos obtenido.

```
pip freeze > requirements.txt
```

Aplicación en Flask

Lo siguiente que haremos será ejecutar el script para comprobar que funciona correctamente.

¿Qué tenemos hecho hasta ahora?
Hemos creado un nuevo entorno virtual únicamente con los paquetes necesarios para este proyecto, así como sus versiones. Esta información la hemos exportado a un .txt que será todo lo que necesitará Elastic Beanstalk para desplegar la aplicación. Además, hemos creado una sencilla app y desplegada en local para comprobar que funciona correctamente.

Despliegue en Elastic Beanstalk

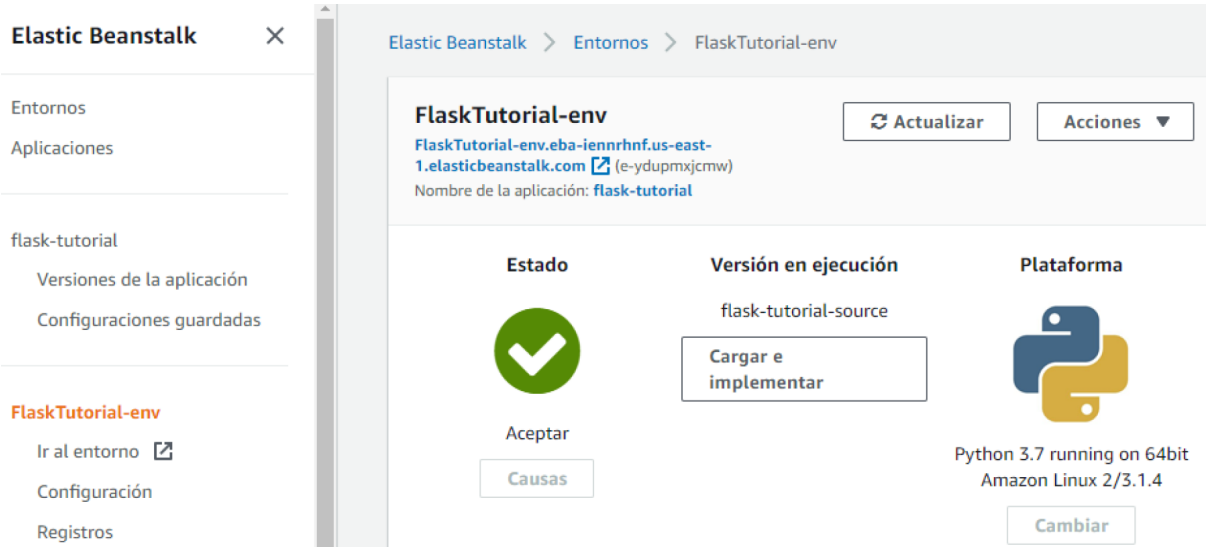
Ya tenemos todo lo necesario para realizar el despliegue. Lo primero que haremos será montar el paquete con el software. Simplemente hay que montar un zip con "application.py" y "requirements.txt".
Ahora vamos a la consola de AWS -> Servicios -> Elastic Beanstalk. Desde aquí podremos desplegar, configurar y monitorizar cualquier despliegue web que hagamos. Ya que lo que queremos subir es una API que esté accesible desde la web, este es el sitio.
Seguiremos los siguientes pasos (todo lo que no se indique en estos pasos, déjalo por defecto):

1. Create Application
2. Introducir nombre. Cuidado con espacios y caracteres raros.
3. Etiquetas. No es necesario tocar nada
4. Plataforma -> Python
5. Ramificación de la plataforma -> La versión de Python tiene que coincidir con la del
entorno virtual que hayamos creado. Selecciona la 3.7.
6. Código de aplicación -> Cargar el código -> Elegir archivo. Sube el zip creado con el
código y los requirements.
7. Crear una aplicación

Automáticamente va a realizar el despliegue de la APP, con todos los recursos necesarios.

Esto puede tardar unos minutos. Realizarlo manualmente es algo tedioso, ya que necesitamos una máquina, un grupo de seguridad, un balanceador de carga, entre otros recursos… Por suerte EC tiene esto bastante automatizado y nos ahorra mucho tiempo.

Una vez haya acabado el despliegue, debería aparecer una pantalla similar a la siguiente:



Despliegue en CI/CD con Code pipeline

Con un servicio llamado code pipeline, podemos hacer CI/CD, que significa que podemos realizar desarrollos del código de la aplicación de tal forma que se despliegue automáticamente.

Deberás seguir una serie de pasos:

1. Crear una canalización
2. Seleccionar proveedor de origen (Github 2)
3. Conectarnos con nuestro Github, dando un nombre a la conexión
4. Seleccionar el repositorio
5. Seleccionar la rama del repositorio a desplegar
6. Omitir compilación
7. Implementar con AWS Elastic Beanstalk
8. Elegir la aplicación

De esta forma cada vez que se actualice el código del repositorio y la rama seleccionada, automáticamente se desplegará la app en AWS.

Eliminación

El acceso a estos recursos no es ilimitado, por lo que deberías eliminar todo lo que hayas creado en este taller para evitar futuros cobros en AWS.
Elimina la aplicación. Suele tardar unos minutos en eliminarse. También se producirá la
eliminación del entorno.
Por último, elimina el bucket de S3 que crea EC. Ve a Servicios -> Almacenamiento -> S3 -> Buckets, y eliminalo.