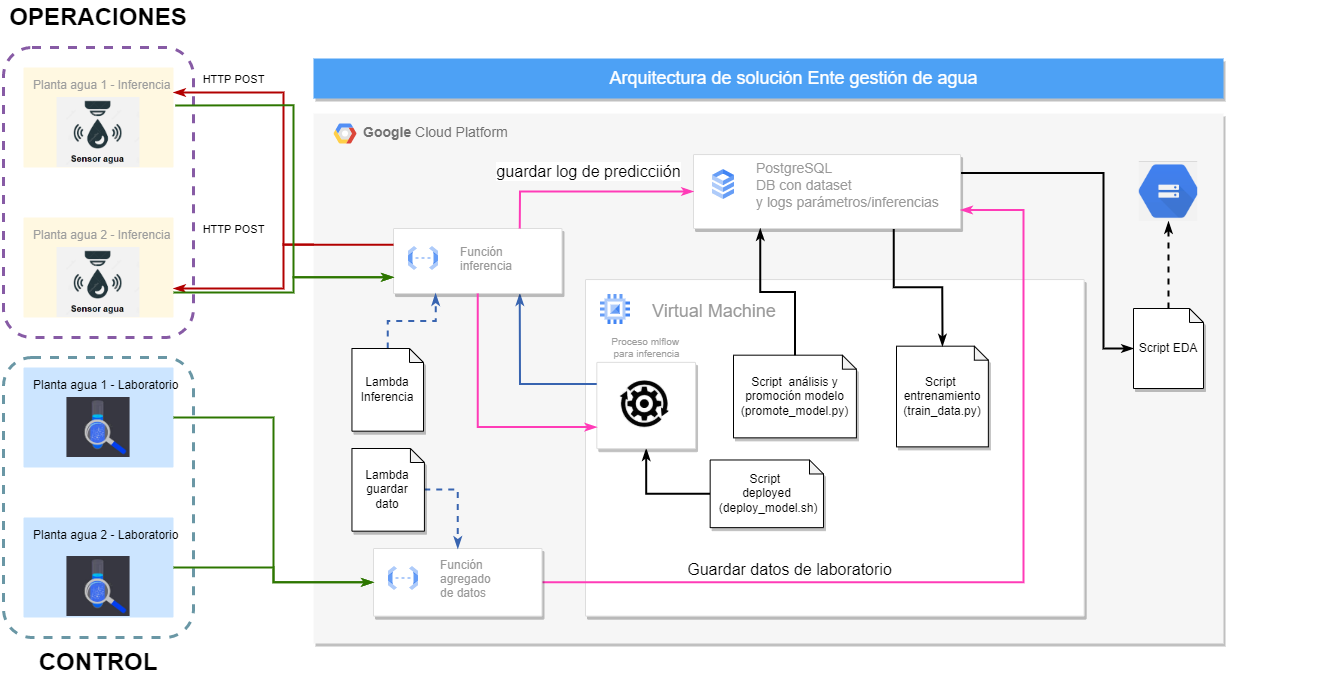
# Esquema de arquitectura

A continuación, se muestra el diagrama de arquitectura propuesta:



La misma está implementada en la nube de Google Cloud y cuenta con estos elementos:

1. Un bucket: donde se guarda la jupyter notebook que se utiliza para hacer el análisis de datos del modelo de agua y ver si es necesario modificar el algoritmo de entrenamiento.
2. Una base de datos PostreSQL: que se utiliza para guardar el dataset con todas las features de agua y los nuevos datos que van ingresando al sistema a medida que se certifican los nuevos ensayos de laboratorio. Además se guarda en la misma un log de datos de inferencias y toda la configuración (metadata) utilizada por mlflow para el control y registro de métricas de las ejecuciones.

La misma cuenta con 2 esquemas:

* Esquema public: utilizado para gestionar la metadata utilizada por mlflow. La misma cuenta internamente con varias tablas.
* Esquema wáter: para gestionar los datos del negocio. Cuenta con 2 tablas:
  + wáter\_potability: que tiene las muestras analizadas y la indicación si la misma es potable o no según sus features asociadas.
  + Wáter\_inferences: que tiene un log de las predicciones del modelo, junto a los valores de las features analizadas, el resultado inferido, la fecha de inferencia y el sensor que hizo la solicitud.

1. Una virtual machine: que es la encargada de servir el modelo y hacer el reentrenamiento del mismo, para que en caso que otro modelo performe mejor, cambiarlo y dejar en producción el más adecuado.

Internamente cuenta con con los siguientes scripts:

* Train\_data.py: este script es el encargado de realizar el entrenamiento de todos los modelos utilizando un gridSerachCV. Toma el dataset de una carpeta local en formato csv y guarda cada ejecución de los modelos utilizando mlflow y la base de datos mencionada anteriormente.

Dentro de los modelos que se entrena se tiene:

* + Random Forest
  + Decission Tree
  + SVM
  + Regresion logística
* reTrainModel.sh: este script es el encargado de hacer el reentrenamiento. Está configurado para ejecutarse todos los días. El mismo copia los datos de la base de datos y ejecuta el script de train\_data.py para reentrenar el modelo complete
* promote\_model.py: este script se ejecuta a continuación de reTrainModel.sh. Es el encargado de analizar todas las corridas realizadas, analizando la métrica de precisión. El modelo que mejor performa lo promueve a producción.
* Deploy\_model.sh: es el encargado de servir el modelo productivo. Para ello se corre el comando mlfow serve en el puerto 5000.

1. Dos lambda functions: utilizadas para la interacción con el modelo productivo.

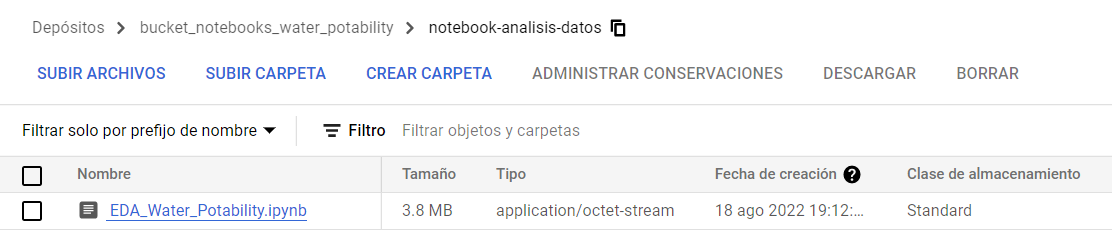
* Lambda\_inferencia: es la encargada de hacer inferencias contra el modelo. Recibe los datos de solicitud de los sensores, hace la inferencia comunicándose con la máquina virtual, guarda un log en la tabla wáter\_inferences de la base de datos y retorna el valor predicho al sensor.
* Lambda\_agregado: permite el agregado de nuevos datos a la tabla wáter\_portability de la base de datos, que corresponde a las muestras de laboratorio realizadas por el personal de planta. Esta lambda es accedida solo por personal habilitado por el ente regulador.

# Bucket

A continuación, se muestra la configuración del bucket en Google Cloud:

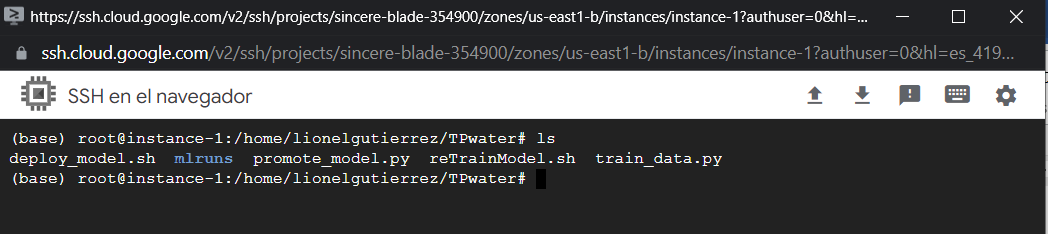


Y la notebook de análisis de datos:



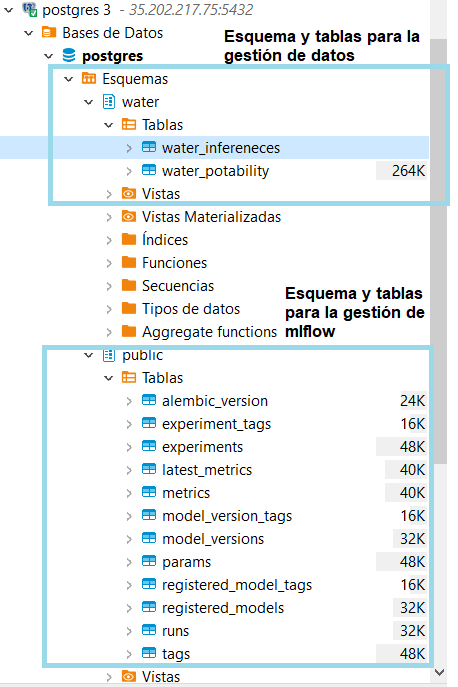
# Virtual Machine

A continuación, se muestra la carpeta con los scripts de la virtual machine. El contenido de cada archivo se puede consultar directamente sobre el repositorio adjunto.



# Base de datos

A continuación, se muestra la estructura de tablas de la base de datos desde dBeaver. Adjunto en el archivo “databases.txt” se cuenta con los scripts utilizados para la generación de las tablas.



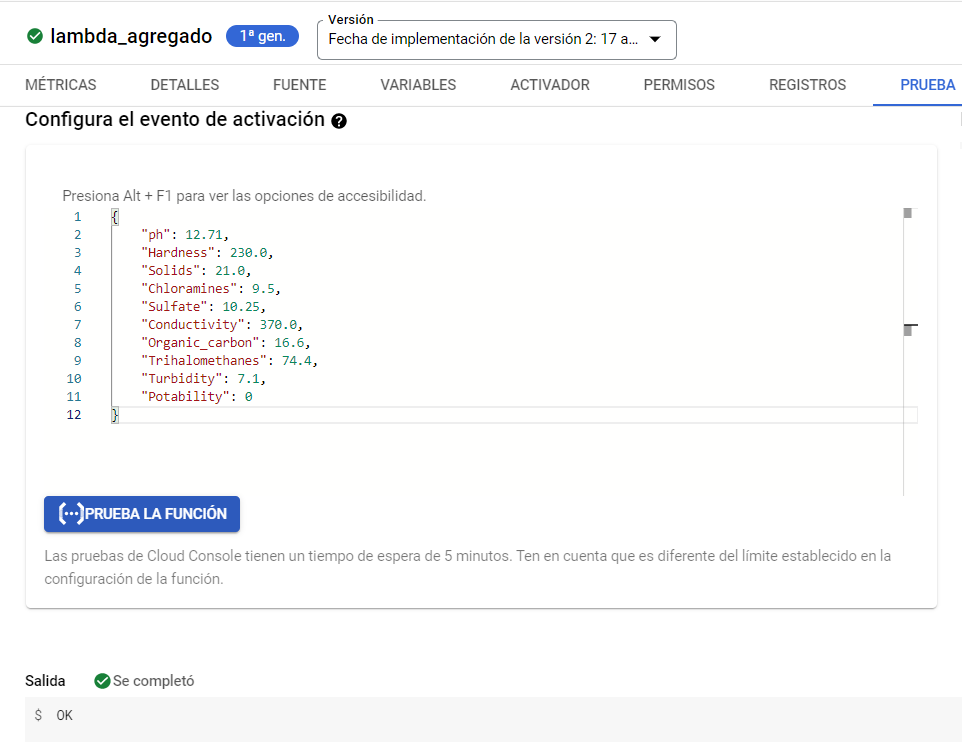
# Lambda functions

A continuación, se muestra las pruebas realizadas sobre las lambda functions desde la interfaz de Google Cloud primero y desde el aplicativo Postman luego.

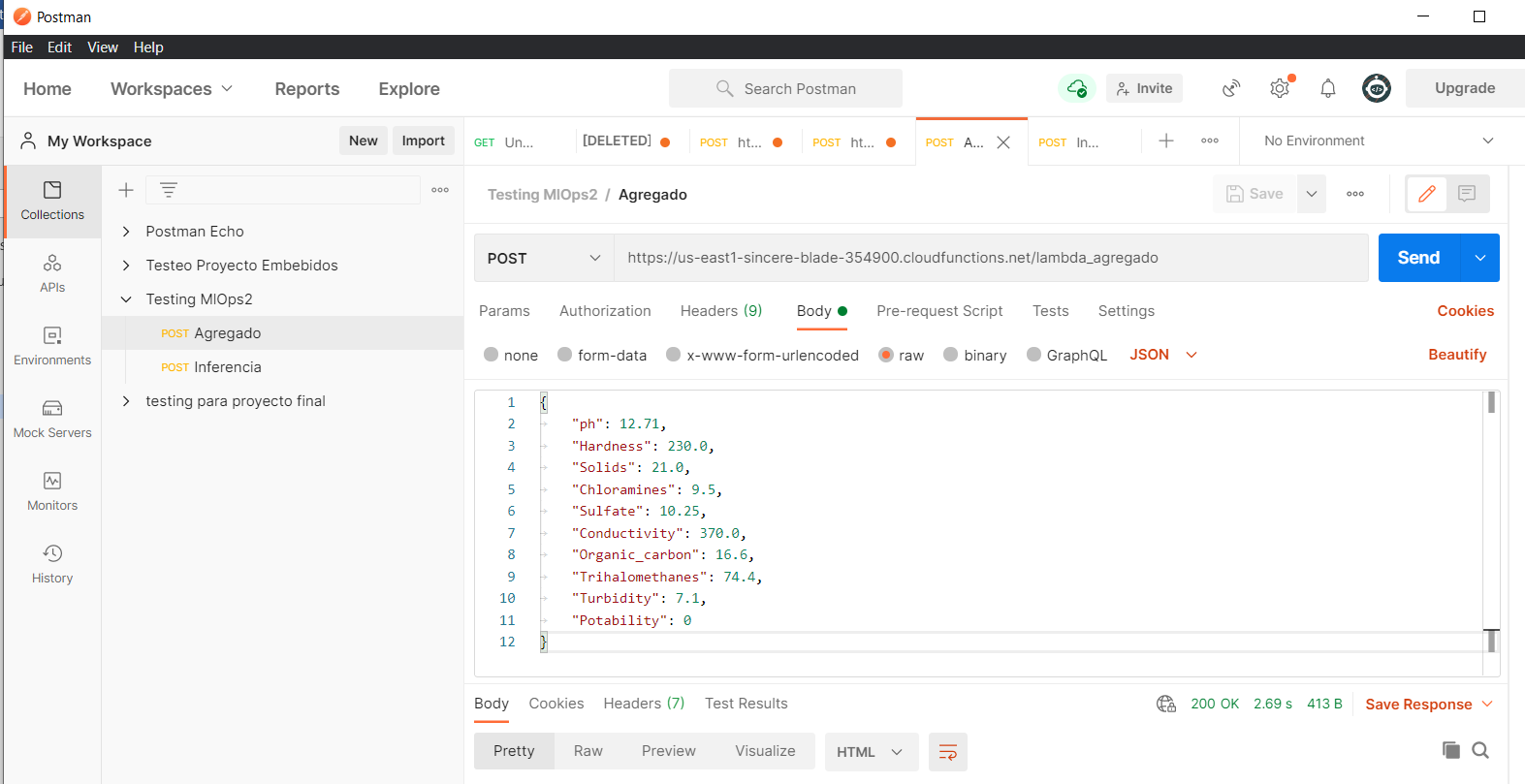
El código de las funciones se adjunta en los archivos .zip “lambda\_inferencia.zip” y “lambda-agregado.zip”

## Lambda agregado

Prueba desde interfaz de Google Cloud

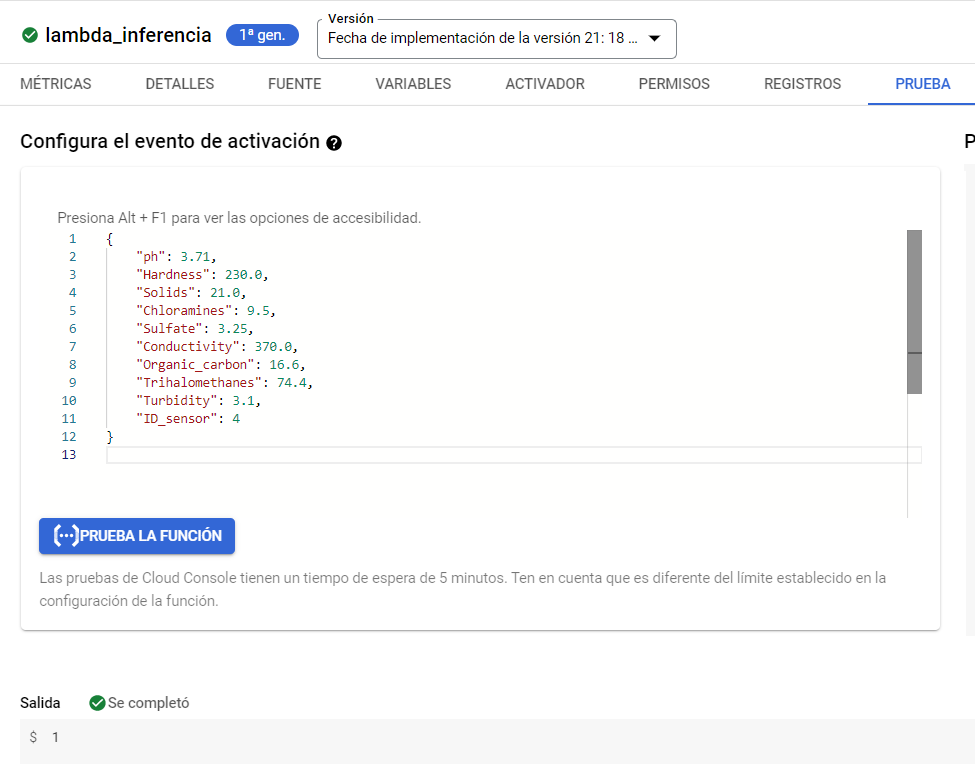


Prueba desde Postman



## Lambda inferencia

Prueba desde interfaz de Google Cloud



Prueba desde Postman

