En este archivo se recopilan todos los pasos y modificaciones necesarias para poner en marcha un GD desde cero.

****Nota: Leer el archivo completo antes de comenzar a implementar****

Configuración del Servidor Ubuntu (si es necesario)

1. **Instalación de Programas:**
    - Se deben instalar los programas necesarios en el servidor Ubuntu según los requisitos del proyecto.
2. **Localización de la Tarjeta de Red:**
    - Se debe identificar y localizar la tarjeta de red del servidor para configurar la conexión de red correctamente.
3. **Asignación de IP (DHCP True):**
    - La asignación de una dirección IP al servidor debe configurarse con DHCP activado, si corresponde según la infraestructura de red.
4. **Instalación de Docker y Configuración de Permisos (chmod):**
    - Docker debe ser instalado en el servidor y se deben configurar los permisos adecuados, utilizando el comando **chmod**, para garantizar su correcto funcionamiento.
5. **Instalación de Clave de Git:**
    - Sigue las instrucciones proporcionadas en la documentación oficial de GitHub para generar una nueva clave SSH y añadirla al agente SSH del servidor https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Como acceder a GDDataPlataform:

- Entrar 172.30.3.5 y login con credenciales de usuario normal. Ahí se encontrar como entrar a gddataplatform.
- Usar DNS y entrar por ssh sampolidi@gddataplatform

Puesta en Producción del Gemelo

Parte de EDGE

****Nota: Entrar al servidor de la planta por ssh****

1. **En local, añadir los tags JSON al MongoDB de Producción:**
    - Se deben crear las colecciones NombreGD y NombreGD_models y añadir los Tags JSON al MongoDB en el servidor de producción en "centrales". mongodb://sampolidi:L0cal2021@ubuntuweb:27017/?authMechanism=DEFAULT
2. **Subir al servidor del GD el repositorio TweenStreaming:**
    - Cargar el repositorio TweenStreaming en el servidor de la planta. Añadir **docker.yml** y cambiar la IP del kafka en **Alice.**
3. **Subir al servidor el repositorio TweenStreamingAPI y Realizar Modificaciones:**
    - Subir el repositorio TweenStreamingAPI.
    - En **mongo-seed** cambiar import.sh: collection y collectionmodels poniendo los nombres de la planta como en mongoDB.
    - Añadir DB.cfg a **mongo-seed** con el contenido: plant=NombrePlanta, ej: plant=Hyatt
    - Añadir DB.cfg de la API (donde esta DB_template.cfg)
    - Añadir en **scheduler**, mongo.cfg y añadir carpeta taglists en scheduler de la API
    - Modificar la IP del backfill en **producer/**backfill_stream.py
    - **Arrancar el productor Kafka**
    - **Arrancar la API docker tween-streaming-api -d en TweenStremingApi**
    - **Arrancar docker completo de TweenStremingApi (para ver errores)**
    - **Abrir portainer para ver que todo ha ido bien: *172.30.10.152:9443* para ver logs y reiniciar contenedores etc.**
    - **Abrir Kafka para ver si llegan mensajes 172.30.10.152:8080/**
4. **Modificar en GDDataPlataform, el repositorio TweenDataLake:**
    - Crear la base de datos en Dbeaver (Timescale).
        1. Host 172.30.10.249
        2. port 5432
        3. usr y pw: root
        4. Nombre de la DB: el nombre de la central igual que en mongo.

- Añadir archivo db/init.sql. Editarlo y añadir las 3 filas que hay para cada GD.
- En TweenDataLake/consumer/config.cfg añadir el nuevo GD con la IP y el nombre de la Planta. Mas abajo añadir la central en [BDmongo].

5. Otros:
    - Comando matar/eliminar topics creados erroneamente: kafka-topics --bootstrap-server localhost:9092 --delete --topic Hyatt.Chillers

Parte de Cloud

1. **Clonar el repositorio TweenOfflineDataOrchestrator:**
2. **Configuración de Nuevos Modelos:**
    - Si es subir nuevos modelos, añadirlos en el **dag** existente.
    - Si es nuevo gemelo, en **custom_dags** crear un nuevo archivo **.py** para la central, ej: **models_hyatt.py** (copiando el que ya existe). En este archivo:
        1. Primero crear backfill
        2. Crear el train_model para cada modelo
        3. Crear evaluate_model para cada modelo
        4. Añadir a la chain tanto el evaluate como el train
    - Crear credenciales para planta en **dags\common_package\credentials.ini**
    - Modificar **strat_backfill.py** añadiendo el GD en la configuración
    - Modificar **train_flow_model.py** añadiendo el GD en la configuración.
    - Probar el orquestador en local. Para ello hacer un *dokcer compose up* únicamente, el resto de las actualizaciones del código se ejecutarán simultáneamente.
    - Una vez funcione, subirlo a Git y hacer un pull en **GDdataplatform**.
    - Subir los nuevos modelos a **GDdataplatform** y su api (equivalente a ParcBit_Models)