# Configuración entorno para uso local Dagster

Pre-requisitos:

- Python 3.6+
- Motor de base de datos (postgres, mysql)
- Virtualenv(recomendado) o conda
- OS Linux cualquier distribución (Recomendado Ubuntu, Debian)

El uso de la herramienta Dagster nos permite orquestar flujo de datos, automatizar tareas de tal forma que se puedan visualizar y monitorear.

**Instalar Dasgter - Dagit**

1. (Opcional) Crea un entorno virtual en la carpeta en donde ejecutará Dagster y ejecuta el siguiente comando:}

   En virtualenv: `python3 -m venv dagster-env`
   En conda(el argumento final “anaconda” es opcional si desea paquetes como pandas): `conda create -n dagster-env python=3.6 anaconda`
2. Activar el entorno virtual:

   ```
   source dagster-env/bin/activate
   ```

3. Instala Dagster y Dagit en el ambiente virtual en donde se alojará la aplicación.

   ```
   pip3 install dagster dagit
   ```

Apartir de aqui ya puedes utilizar dagster, pero sin las funcionalidades de schedule (programación de tareas), sensors (Permite iniciar ejecuciones en funcion de cualquier cambio de estado externo)

**Configuración storage**

1. Instalación API de base de datos, dependiendo de la base de datos en donde se van a guardar los logs, eventos, entre otros:
   `pip3 install dagster-mysql`
   o
   `pip3 install dagster-postgres`
2. Configura la variable de entorno `DAGSTER_HOME`, en un carpeta (*dasgter_home*) donde se alojan los archivos de configuración de la aplicación
   `DAGSTER_HOME=path/dagster_home`
3. Para tener un entorno persistente de implementación en Dagster, es necesario configurar componentes del entorno, con el fin de guardar los logs de las ejecuciones, historial de ejecuciones. Para ello crea dentro de la carpeta `dagster_home` un archivo `dagster.yaml`, con la siguiente configuración (el siguiente ejemplo muestra el uso de una base de datos de postgres).
   Pero puedes seguir la siguiente documentación en caso de utilizar otro motor de base de datos u otras configuraciones adicionales (https://docs.dagster.io/deployment/dagster-instance )

```
run_storage:
        module: dagster_postgres.run_storage
        class: PostgresRunStorage
        config:
                postgres_db:
                        username: 'DAGSTER_PG_USERNAME'
                        password: 'DAGSTER_PG_PASSWORD'
                        hostname: 'DAGSTER_PG_HOST'
                        db_name: 'DAGSTER_PG_DB'
                        port: 'DAGSTER_PG_PORT'
event_log_storage:
        module: dagster_postgres.event_log
        class: PostgresEventLogStorage
        config:
                postgres_db:
                        username: 'DAGSTER_PG_USERNAME'
                        password: 'DAGSTER_PG_PASSWORD'
                        hostname: 'DAGSTER_PG_HOST'
                        db_name: 'DAGSTER_PG_DB'
                        port: 'DAGSTER_PG_PORT'
schedule_storage:
        module: dagster_postgres.schedule_storage
        class: PostgresScheduleStorage
        config:
                postgres_db:
                        username: 'DAGSTER_PG_USERNAME'
                        password: 'DAGSTER_PG_PASSWORD'
                        hostname: 'DAGSTER_PG_HOST'
                        db_name: 'DAGSTER_PG_DB'
                        port: 'DAGSTER_PG_PORT'
```

**Verificaciones**

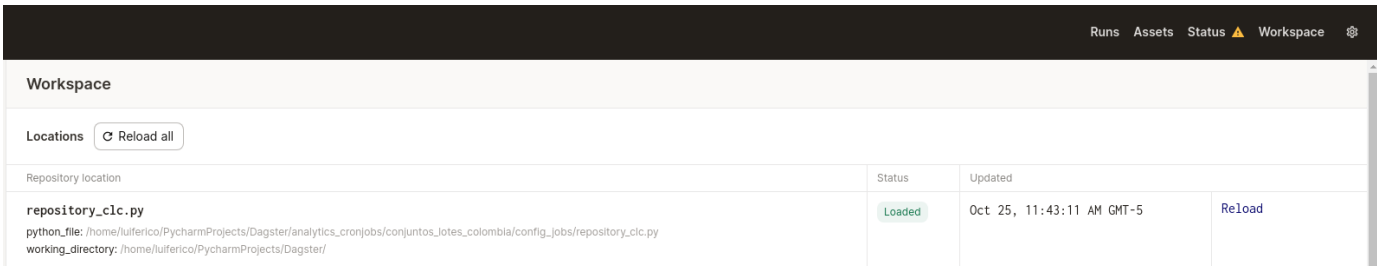
1. Para activar la funcionalidad del daemon que nos ofrece dagster, lo activaremos desde consola con el siguiente comando.

```
nohup dagster-daemon run >/dev/null 2>&1 & disown
```

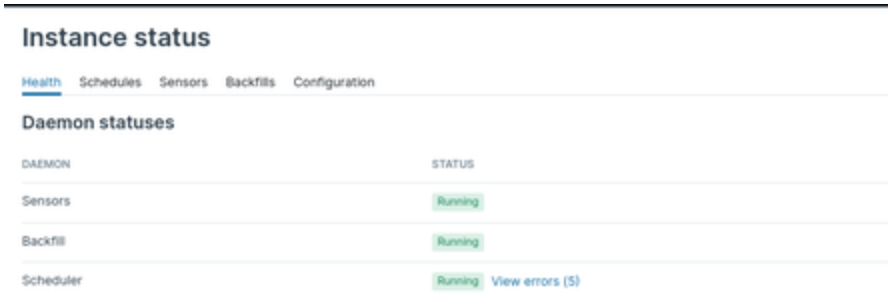
2. Configura un trabajo de acuerdo al siguiente tutorial: https://docs.dagster.io/tutorial/intro-tutorial/single-solid-pipeline

3. Verifica en el entorno dagit la ejecucion del trabajo de ejemplo. Ejecuta:

```
dagit
```




4. Verifica el status de la instancia.


Instance status
Health
Schedules
Sensors
Backfills
Configuration
Daemon statuses
DAEMON
STATUS
Sensors
Running
Backfill
Running
Scheduler
Running
View errors (5)