

18_Introducción a orquestadores y planificación (Airflow, Prefect)

June 18, 2025

Creado por:

Isabel Maniega

```
[1]: from IPython import display
```

1 Introducción a orquestadores y planificación (Airflow, Prefect)

Prefect

<https://www.prefect.io/>

Prefect es una herramienta de orquestación de flujos de trabajo en Python de código abierto creada para los ingenieros modernos de datos y aprendizaje automático. Ofrece una API sencilla que te permite construir una canalización de datos rápidamente y gestionarla a través de un panel interactivo.

Prefect ofrece un modelo de ejecución híbrido, lo que significa que puedes desplegar el flujo de trabajo en la nube y ejecutarlo allí o utilizar el repositorio local.

Comparado con Airflow, Prefect viene con funciones avanzadas como dependencias automatizadas de tareas, activadores basados en eventos, notificaciones integradas, infraestructura específica de flujos de trabajo e intercambio de datos entre tareas. Estas capacidades la convierten en una potente solución para gestionar flujos de trabajo complejos de forma eficiente y eficaz.

Lo primero será realizar la instalación del paquete mediante:

```
pip install -U prefect
```

```
[2]: # pip install -U prefect
```

Después, crearemos un script de Python llamado `prefect_etl.py` en la carpeta de files y escribiremos el siguiente código.

```
from prefect import task, flow
import pandas as pd

# Extract data
@task
def extract_data():
    # Simulating data extraction
    data = {
```

```

        "name": ["Alice", "Bob", "Charlie"],
        "age": [25, 30, 35],
        "city": ["New York", "Los Angeles", "Chicago"]
    }
    df = pd.DataFrame(data)
    return df

# Transform data
@task
def transform_data(df: pd.DataFrame):
    # Example transformation: adding a new column
    df["age_plus_ten"] = df["age"] + 10
    return df

# Load data
@task
def load_data(df: pd.DataFrame):
    # Simulating data load
    print("Loading data to target destination:")
    print(df)

# Defining the flow
@flow(log_prints=True)
def etl():
    raw_data = extract_data()
    transformed_data = transform_data(raw_data)
    load_data(transformed_data)

# Running the flow
if __name__ == "__main__":
    etl()

```

El código anterior define las funciones de tarea `extract_data()`, `transform_data()` y `load_data()` y las ejecuta en serie en una función de flujo llamada `etl()`. Estas funciones se crean utilizando decoradores Prefect Python.

En resumen, estamos creando un DataFrame de pandas, transformándolo y mostrando el resultado final mediante `print`. Esta es una forma sencilla de simular un canal ETL.

Para ejecutar el flujo de trabajo, basta con ejecutar el script de Python mediante el siguiente comando.

```
python prefect_etl.py
```

```
[3]: display.Image("./images/prefect_1.png")
```

```
[3]:
```

```

(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL$ cd files/
(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$ ls
iris.csv iris_versicolor.json prefect_etl.py SampleExcel.xlsx simple.csv yellow_tripdata_2018-01.parquet
(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$ python prefect_etl.py
16:09:53.278 | INFO | prefect - Starting temporary server on http://127.0.0.1:8667
See https://docs.prefect.io/3.0/manage/self-host#self-host-a-prefect-server for more information on running a dedicated Prefect server.
16:10:05.847 | WARNING | prefect.server.utilities.postgres_listener - Cannot create Postgres LISTEN connection: PREFECT_API_DATABASE_CONNECTION_URL is not a PostgreSQL
connection URL (driver: sqlite+aiosqlite).
16:10:06.021 | INFO | Flow run 'gabby-bull' - Beginning flow run 'gabby-bull' for flow 'etl'
16:10:06.107 | INFO | Task run 'extract_data-056' - Finished in state Completed()
16:10:06.152 | INFO | Task run 'transform_data-b1e' - Finished in state Completed()
16:10:06.192 | INFO | Task run 'load_data-eld' - Loading data to target destination:
16:10:06.197 | INFO | Task run 'load_data-eld' -
name age city age_plus_ten
0 Alice 25 New York 35
1 Bob 30 Los Angeles 40
2 Charlie 35 Chicago 45
16:10:06.200 | INFO | Task run 'load_data-eld' - Finished in state Completed()
16:10:06.263 | INFO | Flow run 'gabby-bull' - Finished in state Completed()
16:10:06.290 | INFO | prefect - Stopping temporary server on http://127.0.0.1:8667
(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$

```

Como podemos ver, la ejecución de nuestro flujo de trabajo se ha completado con éxito

Desplegar el flujo

Ahora desplegaremos nuestro flujo de trabajo para que podamos ejecutarlo según una programación o activarlo en función de un evento. Desplegar el flujo también nos permite supervisar y gestionar varios flujos de trabajo de forma centralizada.

Para desplegar el flujo, utilizaremos la CLI de Prefect. La función `deploy` requiere el nombre del archivo Python, el nombre de la función de flujo en el archivo y el nombre de la implantación. En este caso, llamaremos a esta implantación “`simple_etl`”.

```
prefect deploy prefect_etl.py:etl -n 'simple_etl'
```

Tras ejecutar el script anterior en el terminal, puede que recibamos los siguientes mensajes:

- * La infraestructura que queremos escoger para ejecutar el pool: `process`, `ecs`, `docker`, `kubernetes`, etc, en nuestro caso escogeremos la primera ejecutarlo en nuestro sistema.
- * Nombre del pool: `test`
- * Si quieres que ejecuten un código en remoto: le diremos que no `n`.
- * Puedes implementar un horario para la ejecución, en este caso le diremos que no `n`.
- * Nos pedirá si queremos guardar esta configuración para usarla en un futuro le diremos que si `y`

```
[4]: display.Image("./images/prefect_2.png")
```

```
[4]:
```

```
(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$ prefect deploy prefect_etl.py:etl -n 'simple_etl2'
```

The following deployment(s) could not be found and will not be deployed: simple_etl2
 Could not find any deployment configurations with the given name(s): simple_etl2. Your flow will be deployed with a new deployment configuration.

```
16:29:26.326 | INFO    | prefect - Starting temporary server on http://127.0.0.1:8857
See https://docs.prefect.io/3.0/manage/self-host#self-host-a-prefect-server for more information on running a dedicated Prefect server.
16:29:29.013 | WARNING | prefect.server.utilities.postgres.listener - Cannot create Postgres LISTEN connection: PREFECT_API_DATABASE_CONNECTION_URL is not a PostgreSQL connection URL (driver: sqlliteaiosqlite).
? Looks like you don't have any work pools this flow can be deployed to. Would you like to create one? [y/n] (y): y
? What infrastructure type would you like to use for your new work pool? [Use arrows to move; enter to select]
```

	Type	Description
>	process	Execute flow runs as subprocesses on a worker. Works well for local execution when first getting started.
	ecs	Execute flow runs within containers on AWS ECS. Works with EC2 and Fargate clusters. Requires an AWS account.
	azure-container-instance	Execute flow runs within containers on Azure's Container Instances service. Requires an Azure account.
	docker	Execute flow runs within Docker containers. Works well for managing flow execution environments via Docker images. Requires access to a running Docker daemon.
	cloud-run	Execute flow runs within containers on Google Cloud Run. Requires a Google Cloud Platform account.
	cloud-run-v2	Execute flow runs within containers on Google Cloud Run (V2 API). Requires a Google Cloud Platform account.
	vertex-ai	Execute flow runs within containers on Google Vertex AI. Requires a Google Cloud Platform account.
	kubernetes	Execute flow runs within jobs scheduled on a Kubernetes cluster. Requires a Kubernetes cluster.

```
? Work pool name: test2
Your work pool 'test2' has been created!
? Would you like to configure schedules for this deployment? [y/n] (y): n
```

Deployment 'etl/simple_etl2' successfully created with id '85ba6a0a-57c9-4c1d-9e58-ef66d75d378'.

To execute flow runs from these deployments, start a worker in a separate terminal that pulls work from the None work pool:

```
$ prefect worker start --pool None
```

To schedule a run for this deployment, use the following command:

```
$ prefect deployment run 'etl/simple_etl2'
```

```
16:29:51.882 | INFO    | prefect - Stopping temporary server on http://127.0.0.1:8857
(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$
```

Añadimos la configuración para poder ejecutar el pool:

```
prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
```

```
[5]: display.Image("./images/prefect_3.png")
```

```
[5]: isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$ prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
Set 'PREFECT_API_URL' to 'http://127.0.0.1:4200/api'.
Updated profile 'local'.
isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$
```

Tras ejecutar el script anterior en el terminal, puede que recibamos el mensaje de que no tenemos un grupo de trabajadores para ejecutar el despliegue. Para crear el pool de trabajadores, utiliza el siguiente comando.

```
prefect worker start --pool "test"
```

```
[6]: display.Image("./images/prefect_4.png")
```

```
[6]:
```

```

isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$ prefect worker start --pool "test2"
Discovered type 'process' for work pool 'test2'.
Worker 'ProcessWorker e40721ab-f6e3-489a-a8cd-1426c755c8ba' started!
17:14:47.658 | INFO | prefect.flow_runs.worker - Worker 'ProcessWorker e40721ab-f6e3-489a-a8cd-1426c755c8ba' submitting flow run '9ffdba8d-8b99-40db-89c7-bff43c616949'
17:14:47.661 | INFO | prefect.flow_runs.worker - Worker 'ProcessWorker e40721ab-f6e3-489a-a8cd-1426c755c8ba' submitting flow run '368db525-552d-43cb-8381-278bc924e583'
17:14:47.762 | INFO | prefect.flow_runs.runner - Opening process...
17:14:47.779 | INFO | prefect.flow_runs.worker - Completed submission of flow run '368db525-552d-43cb-8381-278bc924e583'
17:14:48.673 | INFO | Flow run 'daring-vulture' - > Running set working directory step...
17:14:48.761 | INFO | Flow run 'daring-vulture' - Beginning flow run 'daring-vulture' for flow 'etl'
17:14:48.762 | INFO | Flow run 'daring-vulture' - View at http://127.0.0.1:4200/runs/flow-run/368db525-552d-43cb-8381-278bc924e583
17:14:48.791 | INFO | Task run 'extract_data-1fs' - Finished in state Completed()
17:14:48.819 | INFO | Task run 'transform_data-7db' - Finished in state Completed()
17:14:48.838 | INFO | Task run 'load_data-f0f' - Loading data to target destination:
17:14:48.841 | INFO | Task run 'load_data-f0f' -
0 Alice 25 New York 35
1 Bob 30 Los Angeles 40
2 Charlie 35 Chicago 45
17:14:48.843 | INFO | Task run 'load_data-f0f' - Finished in state Completed()
17:14:48.893 | INFO | Flow run 'daring-vulture' - Finished in state Completed()
17:14:49.833 | INFO | prefect.flow_runs.runner - Process for flow run 'daring-vulture' exited cleanly.
17:14:49.575 | INFO | prefect.flow_runs.runner - Opening process...
17:14:49.590 | INFO | prefect.flow_runs.worker - Completed submission of flow run '9ffdba8d-8b99-40db-89c7-bff43c616949'
17:14:50.542 | INFO | Flow run 'happy-locust' - > Running set working directory step...
17:14:50.625 | INFO | Flow run 'happy-locust' - Beginning flow run 'happy-locust' for flow 'etl'
17:14:50.625 | INFO | Flow run 'happy-locust' - View at http://127.0.0.1:4200/runs/flow-run/9ffdba8d-8b99-40db-89c7-bff43c616949
17:14:50.655 | INFO | Task run 'extract_data-644' - Finished in state Completed()
17:14:50.686 | INFO | Task run 'transform_data-ff7' - Finished in state Completed()
17:14:50.705 | INFO | Task run 'load_data-bef' - Loading data to target destination:
17:14:50.709 | INFO | Task run 'load_data-bef' -
0 Alice 25 New York 35
1 Bob 30 Los Angeles 40
2 Charlie 35 Chicago 45
17:14:50.710 | INFO | Task run 'load_data-bef' - Finished in state Completed()
17:14:50.772 | INFO | Flow run 'happy-locust' - Finished in state Completed()
17:14:50.915 | INFO | prefect.flow_runs.runner - Process for flow run 'happy-locust' exited cleanly.
[]

```

Ahora que tenemos un grupo de trabajadores, lanzaremos otra ventana de terminal y ejecutaremos el despliegue. El comando `prefect deployment run` requiere “/” como argumento, como se muestra en el comando siguiente.

```
prefect deployment run 'etl/simple_etl'
```

```
[7]: display.Image("./images/prefect_5.png")
```

```

[7]:
isabelmaniega@isabelmaniega:~/Documentos/Python_ETL$ prefect deployment run 'etl/simple_etl'
Creating flow run for deployment 'etl/simple_etl'...
Created flow run 'authentic-loon'.
  __ UUID: 1a7765b0-f39b-432f-892b-a773713f02fc
  __ Parameters: {}
  __ Job Variables: {}
  __ Scheduled start time: 2025-06-18 17:16:54 CEST (now)
  __ URL: http://127.0.0.1:4200/runs/flow-run/1a7765b0-f39b-432f-892b-a773713f02fc
isabelmaniega@isabelmaniega:~/Documentos/Python_ETL$

```

Como resultado de la ejecución del despliegue, recibirás el mensaje de que el flujo de trabajo se está ejecutando. Normalmente, al flujo que se crea se le asigna un nombre aleatorio, en mi caso `authentic-loon`.

Debes iniciar el servidor web de Prefect para visualizar la ejecución del flujo de forma más sencilla y gestionar otros flujos de trabajo.

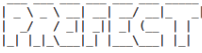
```
$ prefect server start
```

Debemos abrir el dashboard en la siguiente url: `http://127.0.0.1:4200/dashboard`

```
[8]: display.Image("./images/prefect_6.png")
```

```
[8]:
```

```
(env) isabelmaniega@isabelmaniega:~/Documentos/Python_ETL/files$ prefect server start
```



```

Configure Prefect to communicate with the server with:
    prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api
View the API reference documentation at http://127.0.0.1:4200/docs
Check out the dashboard at http://127.0.0.1:4200

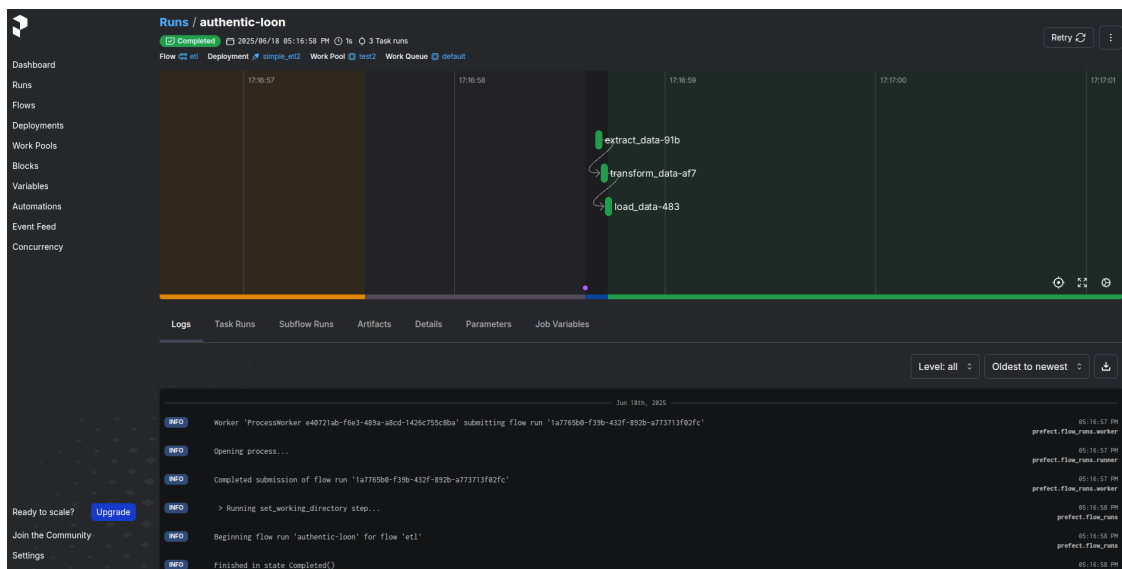
16:35:25.044 | WARNING | prefect.server.utilities.postgres_listener - Cannot create Postgres LISTEN connection: PREFECT_API_DATABASE_CONNECTION_URL is not a PostgreSQL connection URL (driver: sqlite+aiosqlite).

```

Tras ejecutar el comando anterior, se te redirigirá al panel de Prefect. También puedes ir directamente a <http://127.0.0.1:4200> en tu navegador.

```
[9]: display.Image("./images/prefect_7.png")
```

[9]:



El panel de control te permite volver a ejecutar el flujo de trabajo, ver los registros, comprobar los pools de trabajo, establecer notificaciones y seleccionar otras opciones avanzadas. Es una solución completa para tus necesidades modernas de orquestación de datos.

Creado por:

Isabel Maniega