



Workflow Managers

Nombre: Diana Fernanda Castillo Rebolledo

Materia: Computación Tolerante a Fallas

Horario: L-I 11:00 am-1:00 pm

Profesor: Dr. Michel Emanuel López Franco

Sección: D06

Mi ejemplo generado con Prefect consta de un proceso ETL (Extract, transform and load) que consiste en la extracción de datos del link “<https://jsonplaceholder.cypress.io/todos>”, el cual simula información sobre varios usuarios, esto en una lista de objetos json, cada objeto es un usuario. La petición de la información se realizó con ayuda de la librería requests que es para realizar peticiones HTTP. Si por alguna razón no se puede obtener la información se lanza una excepción, de lo contrario se retorna la información obtenida.

En la función transform, se pasará como parámetro la información obtenida de la extracción, la cual es una lista de diccionarios. Por cada objeto dentro de la lista, se crea un diccionario usando los valores de cada objeto, pero modificando las keys, y cada uno de estos nuevos diccionarios se agregan a una lista nueva y esta se retorna.

En la función load, se pasará como parámetro la lista retornada por la función anterior y el archivo json en el que se guardará la información, se abrirá el archivo, se guardará la lista de diccionarios con ayuda de json.dump y se cerrará el archivo.

Una vez completadas estas funciones, para convertirlas en un flujo de trabajo se necesita importar task y Flow de prefect, colocar los decoradores @task en la parte superior de cada función para convertirlas en una tarea, y después crear un flujo, en el que estarán todas las tareas a realizar.

```
schedule = IntervalSchedule(interval=datetime.timedelta(minutes=2))

def prefect_flow(schedule):
    with Flow('etl_flow', schedule=schedule) as flow:
        url = Parameter(name='parametro_url', required=True)
        usuarios = extract(url)
        transformados = transform(usuarios)
        load(transformados, 'datos.json')

    return flow

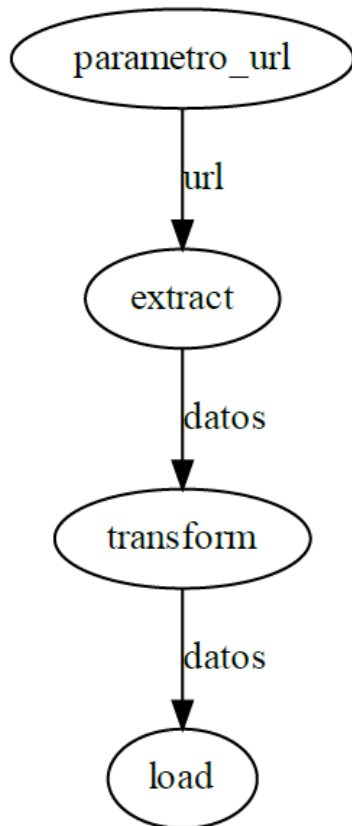
flow = prefect_flow(schedule)
flow.visualize()
flow.run(parameters={
    'parametro_url': 'https://jsonplaceholder.cypress.io/todos'
})
```

También agregué un cronograma para que se ejecute el flujo cada determinado tiempo, en este caso dos minutos, esto con la ayuda de IntervalSchedule de prefect.schedules. Este objeto schedule lo pasé como parámetro a la función donde se encuentra el flujo.

Para ejecutar el flujo que nombré “flow” se realiza con flow.run. Para pasarle parámetros al flujo se necesita importar Parameter de prefect, en este caso pasé el link del cual

obtendrá la información, y adentro del flujo se crea el parámetro con el mismo nombre que se le otorgó al pasar el parámetro a la hora de ejecutar el flujo.

Para poder visualizar el diagrama del flujo generado utilicé visualize:



Aquí se puede notar que cada nodo es una tarea y estas están conectadas entre sí hacia una misma dirección. Hay una dependencia entre parametro_url y extract, esto se debe a que extract utiliza el url obtenido de parametro_url. De la misma manera, la tarea transform depende de extract, ya que utiliza los datos retornados por extract, y load depende de transform ya que utiliza sus datos retornados.

La tarea extract que es la que extrae los datos del url, podría fallar por alguna razón de la red. Podría darse el caso de que la API no esté disponible en ese preciso momento pero que su disponibilidad regrese en poco tiempo. Para solucionar esto, se puede ampliar el decorador @task de esa tarea de la siguiente manera:

```
@task(max_retries=10, retry_delay=datetime.timedelta(seconds=10))
def extract(url):
    resultado = requests.get(url)

    if not resultado:
        raise Exception('Datos no obtenidos.')

    return json.loads(resultado.content)
```

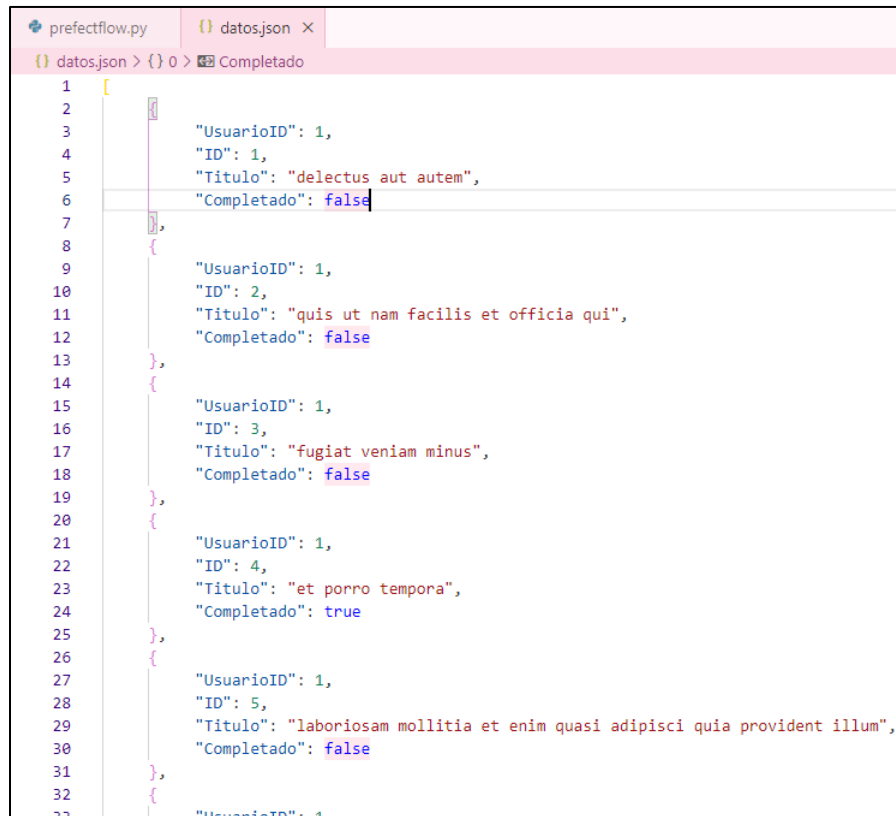
Incluyendo los parámetros max_retries y retry_delay, en los cuales se definen cuántos intentos se le darán a esa tarea para volver a ejecutarse en caso de un error y cada cuánto tiempo se volverá a ejecutar, en este caso puse 10 intentos cada 10 segundos.

Si de pronto no pueden encontrarse los datos en la API por algún motivo (en este caso introducir un link que no existe) se intentará volver a ejecutar la tarea de extract 10 veces esperando que se puedan extraer los datos:

```
PS C:\Users\user\Desktop\prefect> python -u "c:\Users\user\Desktop\prefect\prefectflow.py"
[2022-03-06 20:43:23-0600] INFO - prefect.etl_flow | Waiting for next scheduled run at 2022-03-07T02:44:00+00:00
[2022-03-06 20:44:00-0600] INFO - prefect.FlowRunner | Beginning Flow run for 'etl_flow'
[2022-03-06 20:44:00-0600] INFO - prefect.TaskRunner | Task 'parametro_url': Starting task run...
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'parametro_url': Finished task run for task with final state: 'Success'
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-03-06 20:44:01-0600] ERROR - prefect.TaskRunner | Task 'extract': Exception encountered during task execution!
Traceback (most recent call last):
  File "C:\Users\user\AppData\Local\Programs\Python\Python39\lib\site-packages\prefect\engine\task_runner.py", line 876, in get_task_run_state
    value = prefect.utilities.executors.run_task_with_timeout(
  File "C:\Users\user\AppData\Local\Programs\Python\Python39\lib\site-packages\prefect\utilities\executors.py", line 468, in run_task_with_timeout
    return task.run(*args, **kwargs) # type: ignore
  File "c:\Users\user\Desktop\prefect\prefectflow.py", line 14, in extract
    raise Exception('Datos no obtenidos.')
Exception: Datos no obtenidos.
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Retrying'
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Pending'
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-03-06 20:44:01-0600] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Pending'
[2022-03-06 20:44:01-0600] INFO - prefect.FlowRunner | Flow run RUNNING: terminal tasks are incomplete.
[2022-03-06 20:44:01-0600] INFO - prefect.etl_flow | Waiting for next available Task run at 2022-03-07T02:44:11.781364+00:00
[2022-03-06 20:44:11-0600] INFO - prefect.FlowRunner | Beginning Flow run for 'etl_flow'
[2022-03-06 20:44:11-0600] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-03-06 20:44:12-0600] ERROR - prefect.TaskRunner | Task 'extract': Exception encountered during task execution!
Traceback (most recent call last):
  File "C:\Users\user\AppData\Local\Programs\Python\Python39\lib\site-packages\prefect\engine\task_runner.py", line 876, in get_task_run_state
    value = prefect.utilities.executors.run_task_with_timeout(
```

Por el contrario, si no hay ningún error, al ejecutar el flujo se muestra el momento en el que comienza y termina cada tarea (el flujo se vuelve a ejecutar cada dos minutos) y se carga la información obtenida en el archivo json:

```
PS C:\Users\user\Desktop\prefect> python -u "c:\Users\user\Desktop\prefect\prefectflow.py"
[2022-03-06 20:33:23-0600] INFO - prefect.etl_flow | Waiting for next scheduled run at 2022-03-07T02:34:00+00:00
[2022-03-06 20:34:00-0600] INFO - prefect.FlowRunner | Beginning Flow run for 'etl_flow'
[2022-03-06 20:34:00-0600] INFO - prefect.TaskRunner | Task 'parametro_url': Starting task run...
[2022-03-06 20:34:01-0600] INFO - prefect.TaskRunner | Task 'parametro_url': Finished task run for task with final state: 'Success'
[2022-03-06 20:34:01-0600] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-03-06 20:34:01-0600] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Success'
[2022-03-06 20:34:01-0600] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-03-06 20:34:02-0600] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Success'
[2022-03-06 20:34:02-0600] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-03-06 20:34:02-0600] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Success'
[2022-03-06 20:34:02-0600] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
[2022-03-06 20:34:02-0600] INFO - prefect.etl_flow | Waiting for next scheduled run at 2022-03-07T02:36:00+00:00
[2022-03-06 20:36:00-0600] INFO - prefect.FlowRunner | Beginning Flow run for 'etl_flow'
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'parametro_url': Starting task run...
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'parametro_url': Finished task run for task with final state: 'Success'
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Success'
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Success'
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-03-06 20:36:00-0600] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Success'
[2022-03-06 20:36:00-0600] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
```



```
1  [
2      {
3          "UsuarioID": 1,
4          "ID": 1,
5          "Titulo": "delectus aut autem",
6          "Completado": false
7      },
8      {
9          "UsuarioID": 1,
10         "ID": 2,
11         "Titulo": "quis ut nam facilis et officia qui",
12         "Completado": false
13     },
14     {
15         "UsuarioID": 1,
16         "ID": 3,
17         "Titulo": "fugiat veniam minus",
18         "Completado": false
19     },
20     {
21         "UsuarioID": 1,
22         "ID": 4,
23         "Titulo": "et porro tempora",
24         "Completado": true
25     },
26     {
27         "UsuarioID": 1,
28         "ID": 5,
29         "Titulo": "laboriosam mollitia et enim quasi adipisci quia provident illum",
30         "Completado": false
31     },
32     {
33         "UsuarioID": 1
```

Código en GitHub: <https://github.com/diana-castillo/Workflow-Managers.git>

Conclusiones:

El uso de prefect me pareció muy interesante ya que es una manera sencilla de crear y ejecutar un flujo de trabajo con varias tareas que tengan dependencia entre sí, es decir, que necesiten una de la otra para funcionar correctamente. Un buen ejemplo de uso de prefect es en los procesos ETL, pues en estos procesos se manejan datos y se modifican para guardarse en otro lado, ya sea en una base de datos o diferentes tipos de archivos, en esta integración y manejo de datos puede haber varios errores por distintas razones ya que los datos están en constante movimiento, por lo que un flujo de trabajo con prefect ayuda a asegurar que las tareas se lleven a cabo correctamente y a poder distinguir las dependencias que existen entre tareas para poder tratar cada una con lo necesario para que el flujo se ejecute con éxito.

Referencias:

Radečić, D. (2021, 22 julio). *Prefect: How to Write and Schedule Your First ETL Pipeline with Python. Towards Data Science*. Recuperado 5 de marzo de 2022, de <https://towardsdatascience.com/prefect-how-to-write-and-schedule-your-first-etl-pipeline-with-python-54005a34f10b>

Fortier, K. [Kevin Fortier]. (2020, 30 diciembre). *Prefect Tutorial | Indestructible Python Code* [Video]. YouTube. <https://www.youtube.com/watch?v=0lcN117E4Xo>

ETL with Prefect. (s. f.). Prefect Docs. Recuperado 5 de marzo de 2022, de <https://docs.prefect.io/core/tutorial/02-etl-flow.html>