Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Workflow managers

Alumno: Bryan De Anda Reyes

Código de alumno: 216195537

Materia: Computación Tolerante a Fallas (D06)

Horario: lunes y miércoles (11:00 – 13:00)

Carrera: INCO

Fecha de entrega: 07 de marzo de 2022

Instrucciones

El objetivo de esta actividad en realizar un ejemplo utilizando el manejo de Prefect con Python. Prefect es un nuevo sistema de gestión de flujos de trabajo, diseñado para infraestructuras modernas y alimentado por el motor de flujo de trabajo Prefect Core de código abierto. Los usuarios organizan Tasks en Flows, y Prefect se encarga del resto.

Desarrollo

El ejemplo que implementare para esta actividad es crear un flujo de trabajo de un conjunto de tareas implementadas en Python. Los datos los manejaremos como un mecanismo ETL.

ETL por sus siglas en inglés (Extract, Transform, Load) representa extracción, transformación y carga. Básicamente consiste en “Extraer” los datos crudos desde su origen (Source), “Transformarlos” según nuestras necesidades de analítica o la estructura que deseamos y “Cargarlos”.

En el caso de mi ejemplo, lo que hare es extraer datos de una lista de valores de un archivo que se encuentra guardado en mi computadora, estos datos de la lista se transformaran incrementado cada número en uno y después estos datos modificados se guardaran exactamente en el mismo archivo.

Flujo de trabajo

Para esto fue necesario crear tres funciones, una de ellas se encarga de extraer los datos del archivo, otra función incrementara los datos y la tercera función cargara los nuevos datos al mismo archivo.

La primer función llamada “extract()” recibirá como parámetro una dirección de archivo, la función “transform()” recibe los datos y la función “load()” recibe los datos y la dirección de archivo.

En la siguiente imagen se muestran las tres funciones creadas para el funcionamiento de ETL.

```python
def extract(path):
    with open(path, "r") as f:
        text = f.readline().strip()
    data = [int(i) for i in text.split(",")]
    return data

def transform(data):
    tdata = [i + 1 for i in data]
    return tdata

def load(data, path):
    with open(path, "w") as f:
        csv_writer = csv.writer(f)
        csv_writer.writerow(data)
    return
```

Ilustración 1. Funciones ETL

Hice la prueba de estas funciones mandándolas a llamas e imprimiendo en consola el resultado para verificar que estén funcionando correctamente.

```
[1, 2, 3, 4, 5, 6, 7, 8]
[2, 3, 4, 5, 6, 7, 8, 9]
```

Ilustración 3. Incremento de los valores

Ilustración 2. Captura en archivo

En lo mencionado anteriormente aun no se a trabajado con Prefect y podríamos decir que hasta este punto esto es un flujo de trabajo.

Implementación de Prefect

Lo que voy a hacer primero es convertir mis funciones en Prefect Task y despues vamos a encadenar nuestras tareas juntas en un Prefect Flow, escribiendo la palabra @task() sobre cada una de las funciones esto las convierte en Prefect Task.

Ahora para ejecutar el código necesitamos Prefect Flow, primero lo aremos como un administrador de contexto y corriendo el flujo con la función Flow.run().

```python
import csv
from prefect import task, Flow

@task
def extract(path):
    with open(path, "r") as f:
        text = f.readline().strip()
    data = [int(i) for i in text.split(",")]
    #print(data)
    return data

@task
def transform(data):
    tdata = [i + 1 for i in data]
    return tdata

@task
def load(data, path):
    with open(path, "w") as f:
        csv_writer = csv.writer(f)
        csv_writer.writerow(data)
    return

with Flow("my_etl") as flow:
    data = extract("values.csv")
    tdata = transform(data)
    result = load(tdata, "tvalues.csv")

flow.run()
```

Al ejecutar el código tal y como lo tenemos podemos observar que nos arroja diferente información, nos dice cuando comienza a correr una tarea, cuando finaliza la ejecución de esa tarea. Esta información es importante cuando se quiere saber más a detalla información sobre las tareas en un código, en este ejemplo podemos ver que todas las tareas se completaron con éxito.

```
"C:\Users\Bryan De Anda\PycharmProjects\pythonProject2\venv\Scripts\python.exe" "C:/Users/Bryan De Anda/PycharmProjects/pythonProject2/Workflow.py"
[2022-04-20 18:02:52-0500] INFO - prefect.FlowRunner | Beginning Flow run for 'my_etl'
[2022-04-20 18:02:52-0500] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-04-20 18:02:53-0500] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Success'
[2022-04-20 18:02:53-0500] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-04-20 18:02:53-0500] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Success'
[2022-04-20 18:02:53-0500] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-04-20 18:02:53-0500] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Success'
[2022-04-20 18:02:53-0500] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
```

Ahora voy a crear una función llamada “build_flow()” pera tener una propia función para el flujo de compilación y esta regresara un flujo.

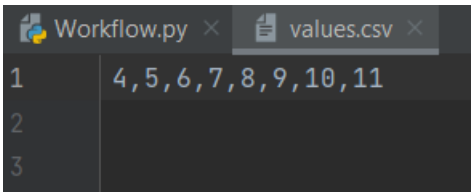
También se modifico la parte de la ruta para en lugar de mandar la ruta directamente se utilizaron “Parameter” de la librería “Prefect”.

```python
def build_flow():
    with Flow("my_etl") as flow:
        path = Parameter(name="path", required=True)
        data = extract(path)
        tdata = transform(data)
        result = load(tdata, path)

    return flow

flow = build_flow()
flow.run(parameters={
    "path": "values.csv"
})
```

Si ejecutamos el código nos muestra la información de cada tarea que se está ejecutando.

```
"C:\Users\Bryan De Anda\PycharmProjects\pythonProject2\venv\Scripts\python.exe" "C:/Users/Bryan De Anda/PycharmProjects/pythonProject2/Workflow.py"
[2022-04-20 18:35:58-0500] INFO - prefect.FlowRunner | Beginning Flow run for 'my_etl'
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'path': Starting task run...
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'path': Finished task run for task with final state: 'Success'
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Success'
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Success'
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-04-20 18:35:58-0500] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Success'
[2022-04-20 18:35:58-0500] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
```

También podemos observar que los valores al ejecutar el código tres veces incrementaron su número.



Ahora lo que hare a continuación es que el flujo de tareas se ejecute cada cierto tiempo para esto se implementó la siguiente función.

```
schedule = IntervalSchedule(
    start_date = datetime.datetime.now() + datetime.timedelta(seconds=1),
    interval = datetime.timedelta(seconds=5)
)
```

Ahora podemos observar que se ejecuta cada 5 minutos el flojo de prefect.

```
"C:\Users\Bryan De Anda\PycharmProjects\pythonProject2\venv\Scripts\python.exe" "C:/Users/Bryan De Anda/PycharmProjects/pythonProject2/Workflow.py"
[2022-04-20 20:03:31-0500] INFO - prefect.my_etl | Waiting for next scheduled run at 2022-04-21T01:03:32.477731+00:00
[2022-04-20 20:03:32-0500] INFO - prefect.FlowRunner | Beginning Flow run for 'my_etl'
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'path': Starting task run...
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'path': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-04-20 20:03:32-0500] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:32-0500] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
[2022-04-20 20:03:32-0500] INFO - prefect.my_etl | Waiting for next scheduled run at 2022-04-21T01:03:37.477731+00:00
[2022-04-20 20:03:37-0500] INFO - prefect.FlowRunner | Beginning Flow run for 'my_etl'
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'path': Starting task run...
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'path': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'extract': Starting task run...
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'extract': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'transform': Starting task run...
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'transform': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'load': Starting task run...
[2022-04-20 20:03:37-0500] INFO - prefect.TaskRunner | Task 'load': Finished task run for task with final state: 'Success'
[2022-04-20 20:03:37-0500] INFO - prefect.FlowRunner | Flow run SUCCESS: all reference tasks succeeded
[2022-04-20 20:03:37-0500] INFO - prefect.my_etl | Waiting for next scheduled run at 2022-04-21T01:03:42.477731+00:00
```

Link al código en el repositorio: https://github.com/BryanDeAnda/Workflow-managers.git

Conclusión

La realización de esta práctica me pareció muy interesante, ya que no conocía Prefect y me parece una gran herramienta de trabajo cuando se desea realizar flujos de tareas, algo que pude notar es que para que el flujo de trabajo funcione correctamente, las tareas deben estar conectadas entre si y algo que me ayudo a comprender de mejor manera esto fue el mecanismo ETL ya que este sus tareas necesitan estar entrelazadas para funcionar correctamente.

También pude notar que Prefect es bastante útil para analizar los tiempos en los que se inicia y termina de ejecutarse diferentes tareas.

Bibliografía

A. (2020, 2 junio). *¿Qué son los procesos ETL? Conócelos a fondo*. Platzi. Recuperado 2022, de https://platzi.com/blog/que-es-un-etl/

Fortier, K. (2020, 30 diciembre). *Prefect Tutorial* | *Indestructible Python Code*. YouTube. https://www.youtube.com/watch?v=0IcN117E4Xo