Data Engineering with Dagster — Week 3

Assets

CODE LOCATION

`week_3/content/etl.py`

Metadata in Dagster

To close out this week, we are going to revisit a pipeline from last week and make one small addition. Our pipeline `etl_docker` wrote a random number of lines to a table in a Postgres database. We are going to attach an `asset` to the op while keeping everything else the same. At the end of the `insert_into_table()` op, we will add the following.

```
@op(
    required_resource_keys={"database"},
    tags={"kind": "postgres"},
)
def insert_into_table(context, table_name: String):
    sql = f"INSERT INTO {table_name} (column_1) VALUES (1);"

    number_of_rows = randint(1, 10)
    for _ in range(number_of_rows):
        context.resources.database.execute_query(sql)
        context.log.info("Inserted a row")

    context.log.info("Batch inserted")

    # New this week
    context.log_event(
        AssetMaterialization(
            asset_key="my_micro_batch",
            description="Inserting a random batch of records",
            metadata={"table_name": table_name, "number_of_rows":
number_of_rows},
        )
    )
```

This is a very simple `asset`. An asset is anything produced by a pipeline. We set a specific key to identity our asset and give it a description. We have also included some metadata specific to this asset for the table name and the number of rows that have been produced.

Asset Catalog

After we run our pipeline a few times, we can go into Dagit and view our materialized assets in the "Assets" tab.

We get a full history of every time our asset has been materialized. In the Dagster, an asset is materialized when it has been created or updated. Listed with every asset materialization is the job, op and run ID associated with that particular asset materialization. We can also see the custom metadata that we tied to our asset. Our table name has remained consistent, but our number of rows has changed on different runs of the pipeline.

Software Defined Assets

This was a very brief overview of assets. However, they are one of the most important abstractions in Dagster. In fact, assets have always been one of the defining concepts in Dagster's philosophy and something that has gone through a lot of iterations as the framework has matured. Next week we will talk about software-defined assets and how they differ from simple assets.

Summary

Assets help us make sense of what our pipelines actually produce. This can sometimes be obfuscated by our desire to have our pipelines succeed. However, we should not ignore what is produced by our pipelines. We run our pipelines to produce data assets that can be leveraged for analytics or machine learning. Really, it should be assets that connect our data platform together rather than pipelines or jobs themselves. This is the promise fulfilled by software-defined assets, which we will discuss next week.

Object	Relationship	Description
Asset	An op can produce one or more assets. One or more jobs can rely on the same asset.	An asset is the materialization of work done by Dagster.