Assets

Past, Present, Future

3.0

TP Chung

About me

- Astronomer
- Airflow PMC member
- Python packaging
- Community organiser in Taiwan
- Timetables
- Dynamic task mapping
- Un-execution date
- Assets

What are assets really?

Dataset

- Airflow 2.4
- OpenLineage (since ??)
- Some place data live
- The data themselves

DatasetEvent

- “There’s an update”
- Runtime vs declared context
- Not from OpenLineage?

Airflow needs its own terminology

“It’s more than data…”

- Machine learning
- Vaguer!
- “Thing”

Where we are now

3.0

Concepts

- Asset
- AssetEvent
- AssetAlias

Eventing

- Producing task
- Consuming dag
- Metadata
- AssetWatcher

```
example = Asset(name="example", uri="s3://asset-bucket/example.csv")
```

```
example = Asset(name="example", uri="s3://asset-bucket/example.csv")

with DAG(dag_id="output_to_example", schedule="@daily"):
    create_object = S3CreateObjectOperator(
        task_id="create_object",
        s3_bucket="asset-bucket",
        s3_key="example.csv",
        data=YOUR_DATA_HERE,
        replace=True,
        outlets=[example],
    )
```

```
example = Asset(name="example", uri="s3://asset-bucket/example.csv")

with DAG(dag_id="output_to_example", schedule="@daily"):
    create_object = S3CreateObjectOperator(
        task_id="create_object",
        s3_bucket="asset-bucket",
        s3_key="example.csv",
        data=YOUR_DATA_HERE,
        replace=True,
        outlets=[example],
    )

with DAG(dag_id="trigger_on_example_write", schedule=example):
    @task
    def do_something():
        context = get_current_context()
        last_event = context["triggering_asset_events"][example][-1]
        print("Triggered by", last_event.source_dag_run)
```

```
s3_bucket = AssetAlias(name="s3_bucket")

per_day_assets = {
    i: Asset(name="first_day", uri=f"s3://asset_bucket/{i}.csv")
    for i in range(1, 8)
}
```

```
s3_bucket = AssetAlias(name="s3_bucket")

per_day_assets = {
    i: Asset(name="first_day", uri=f"s3://asset_bucket/{i}.csv")
    for i in range(1, 8)
}

with DAG(dag_id="output_to_example", schedule="@daily"):
    @task(outlets=[s3_bucket])
    def emit_conditionally():
        context = get_current_context()
        asset = per_day_assets[context["logical_date"].isoweekday()]
        yield Metadata(asset, alias=s3_bucket)
```

```
s3_bucket = AssetAlias(name="s3_bucket")

per_day_assets = {
    i: Asset(name="first_day", uri=f"s3://asset_bucket/{i}.csv")
    for i in range(1, 8)
}

with DAG(dag_id="output_to_example", schedule="@daily"):
    @task(outlets=[s3_bucket])
    def emit_conditionally():
        context = get_current_context()
        asset = per_day_assets[context["logical_date"].isoweekday()]
        yield Metadata(asset, alias=s3_bucket)

with DAG(dag_id="prcess_monday", schedule=per_day_assets[1]):
    ...

with DAG(dag_id="prcess_everything", schedule=s3_bucket):
    ...
```

```
example = Asset(
    name="example",
    uri="s3://asset-bucket/example.csv",
    watchers=[AssetWatcher(...)],
)
```

```
example = Asset(
    name="example",
    uri="s3://asset-bucket/example.csv",
    watchers=[AssetWatcher(...)],
)

with DAG(dag_id="trigger_on_example_write", schedule=example):
    @task
    def do_something():
        context = get_current_context()
        last_event = context["triggering_asset_events"][example][-1]
        if last_event.extra.get("from_trigger"):
            print("Triggered by watcher!")
```

```
example = Asset(name="example", uri="s3://asset-bucket/example.csv")


with DAG(dag_id="output_to_example", schedule="@daily"):
    create_object = S3CreateObjectOperator(
        ..., outlets=[example],
    )



# Different file...

with DAG(dag_id="trigger_on_example", schedule=Asset.ref("example")):
    ...
```

Where we are going

3.0

Better integration

- Object Storage
- Dataframes!
- I/O manager

```
example_asset = Asset(
    name="example",
    uri="s3://asset-bucket/example.csv",
)

example_path = ObjectStoragePath(
    example_asset.uri,
    conn_id="aws_default",
)

@task(outlets=[example_asset])
def write_data():
    with example_path.open("wb") as f:
        f.write(YOUR_DATA_HERE)

@task(inlets=[example_asset])
def read_data():
    with example_path.open("rb") as f:
        data = f.read()
```

```
example_asset = Asset(
    name="example",
    uri="mysql://myhost/example",
)
```

```
example_asset = Asset(
    name="example",
    uri="mysql://myhost/example",
)

example_client = DatabaseClient(
    example_asset.uri,
    conn_id="mysql_default",
)
```

```
example_asset = Asset(
    name="example",
    uri="mysql://myhost/mydb/mytable",
)

example_client = DatabaseClient(
    example_asset.uri,
    conn_id="mysql_default",
)

@task(outlets=[example_asset])
def write_data():
    with example_client.connect() as t:
        t.insert(YOUR_DATA_HERE)

@task(inlets=[example_asset])
def read_data():
    with example_client.connect() as t:
        data = t.to_pandas()
```

```
example_asset = Asset(
    name="example",
    uri="mysql://myhost/mydb/mytable",
)

example_client = DatabaseClient(
    example_asset.uri,
    conn_id="mysql_default",
)

@task(outlets=[example_asset])
def write_data():
    with example_client.connect() as t:
        t.insert(YOUR_DATA_HERE)

@task(inlets=[example_asset])
def read_data():
    with example_client.connect() as t:
        data = t.to_pandas()
```

```
example_asset = Asset(
    name="example",
    uri="mysql://myhost/mydb/mytable",
)

@task(outlets=[example_asset])
def write_data(io_manager):
    with io_manager.connect(example_asset) as t:
        t.insert(YOUR_DATA_HERE)

@task(inlets=[example_asset])
def read_data(io_manager):
    with io_manager.connect(example_asset) as t:
        data = t.to_pandas()
```

Better metadata

- Validation
- Partition
- Watermarking
- Structured extra

Validation

- Not to replace e.g. GX
- Fail task on critical errors
- Prevent writes

Partitions

- Granular freshness
- Downstream selectivity
- UI integration

Watermarking

- Different enough
- Focus on latest state
- State after run

```
example = Asset(
    name="example",
    uri="s3://asset-bucket/example.csv",
    watchers=[AssetWatcher(...)],
)

with DAG(dag_id="trigger_on_example_write", schedule=example):
    @task
    def do_something():
        context = get_current_context()
        last_event = context["triggering_asset_events"][example][-1]
        if last_event.extra.get("from_trigger"):
            print("Triggered by watcher!")
```

Words at the end

- I love this
- For the crowd
- As a crowd

AIRFLOW SUMMIT

The 2025 Apache Airflow® Survey is here!

Fill it out to for a free Airflow 3 fundamentals or dag authoring in Airflow 3 certification code

Questions?

tp@astronomer.io
Community Slack: Tzu-ping Chung