dagster

Dagster Deep Dives: Automation

Pedram Navid

Head of Data Engineer & DevRel - Dagster Labs

Overview

dagster

Dagster is a Python data orchestrator that allows you to build a world-class data platform.

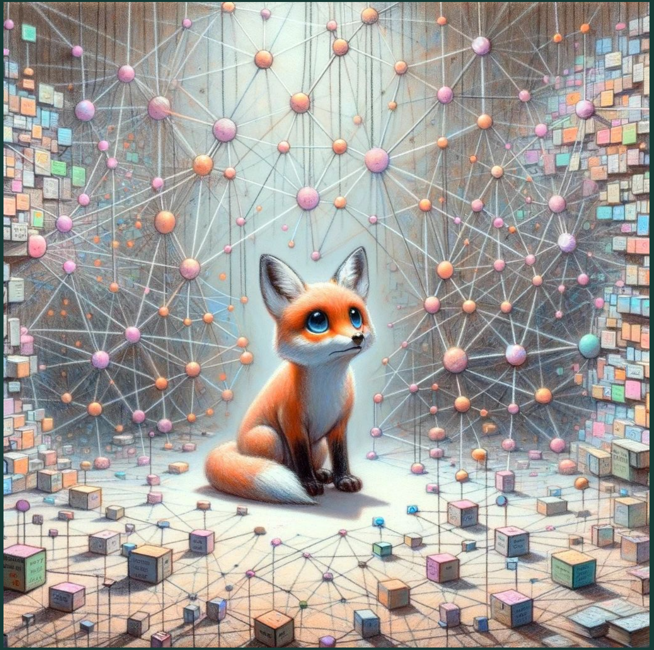
Run and monitor your pipelines, using Python code as your source of truth

A data-centric orchestrator

dagster

Why Automation?

- Running things manually is not fun!
- Complex platforms have differing needs
- Declarative approaches simplifies pipelines



Materialize not too often, but not too infrequently

With Dagster, you have state-of-the-art flexibility in how you automate your data assets.

Schedules, **Sensors,** and **Auto-Materialize Policies** are the core foundations of automation in Dagster.

Schedules

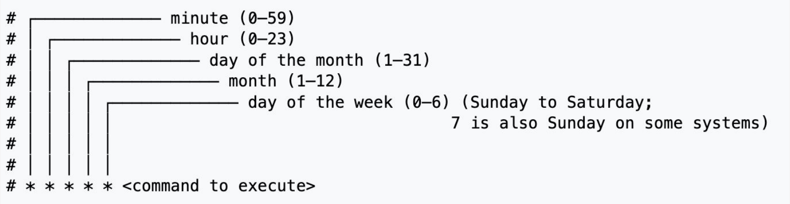
dagster

Schedules

- A schedule consists of a **job** and a **cron-expression**
- Runs at some regular interval
- Against a subset/selection of your assets

```
bird_feeders_table_job = define_asset_job(
    name="bird_feeders_table_job",
selection="bird_feeders_table"
)

bird_feeder_schedule = ScheduleDefinition(
    job=bird_feeder_notebook_job,
    cron_schedule="0 0 * * *",  # daily at
midnight
)
```



```
# ┌───────────── minute (0–59)
# │ ┌───────────── hour (0–23)
# │ │ ┌───────────── day of the month (1–31)
# │ │ │ ┌───────────── month (1–12)
# │ │ │ │ ┌───────────── day of the week (0–6) (Sunday to Saturday;
# │ │ │ │ │                                   7 is also Sunday on some systems)
# │ │ │ │ │
# │ │ │ │ │
# * * * * * <command to execute>
```

dagster

Sensors

Sensors

- Creates runs based off events
- Useful for polling for external situations (new file, system resource) or asset materialization
- Returns a **RunRequest** that targets a **Job**
- Uses a **cursor** or a **run_key** for idempotence

```python
@sensor(job=bird_feeders_table_job)
def bird_observation(context: SensorEvaluationContext):
    previous_state = json.loads(context.cursor) if context.cursor else {}
    current_state = {}
    files_to_process = []
    for ... :
       current_state[filename] = ...
        if file not in previous_state ...
       files_to_process.append(
         RunRequest(run_key=f"bird_observation_{filename}_{last_modified}",
                        run_config=...some config…))
    return SensorResult(run_requests=files_to_process,
                        cursor=json.dumps(current_state))
```

dagster

Auto-materialize policies

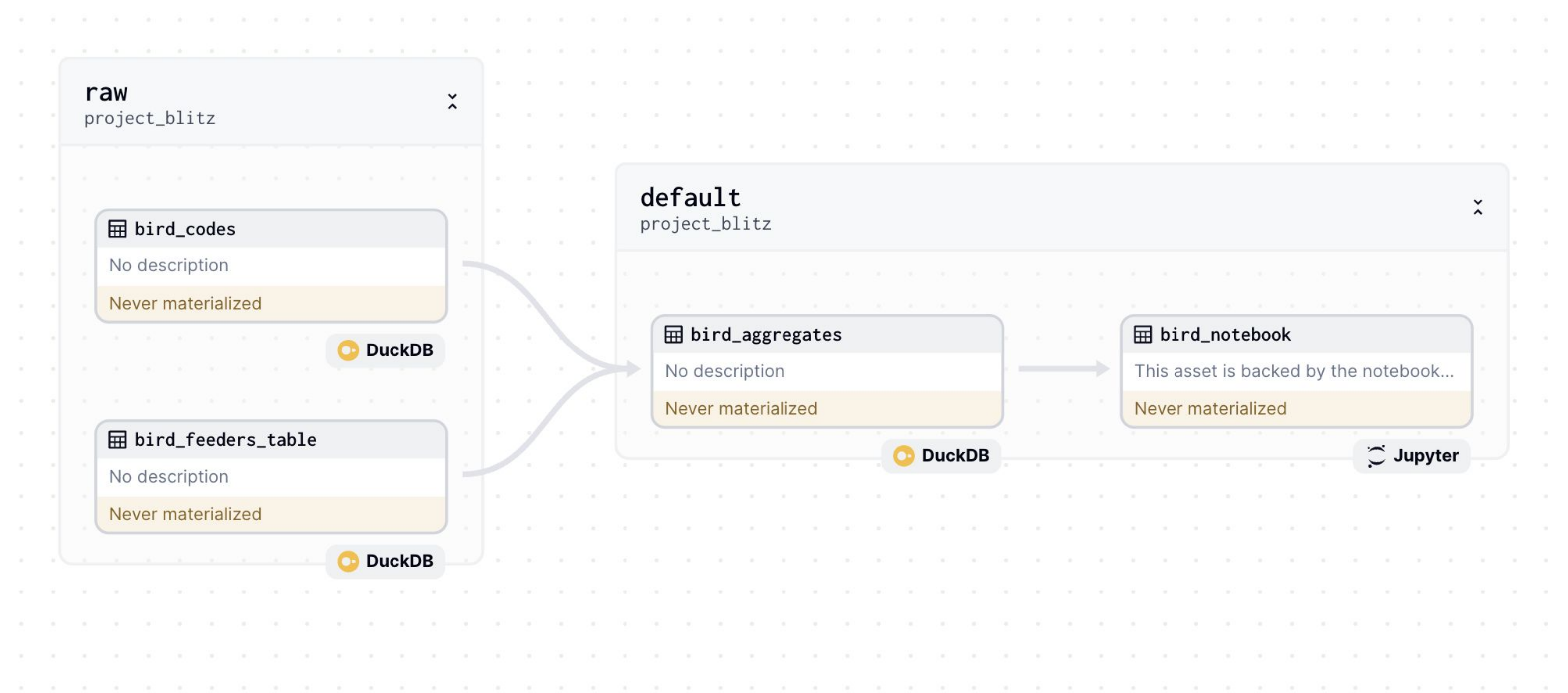
dagster

Auto-materialize policies

- Declarative approach to scheduling
- Can auto-materialize if parents are updated or if partitions are missing

```python
@asset(
    compute_kind="duckdb",
    deps=[bird_feeders_table, bird_codes],
    auto_materialize_policy=AutoMaterializePolicy.eager(),
)
def bird_aggregates(context: AssetExecutionContext, duckdb: DuckDBResource):
   ...
```

What we’ll build



raw
project_blitz

bird_codes
No description
Never materialized
DuckDB

bird_feeders_table
No description
Never materialized
DuckDB

default
project_blitz

bird_aggregates
No description
Never materialized
DuckDB

bird_notebook
This asset is backed by the notebook...
Never materialized
Jupyter

Demo Time!

Star Us on Github!

https://bit.ly/DagDemo

Next steps & resources

Join us in Slack

Connect with other data practitioners. Share knowledge or find help

bit.ly/DagSlack

Sign up for Dagster Cloud

Sign up for Dagster Cloud and get started with a free 30 day trial

bit.ly/DagCloud

Get the Newsletter

Stay up-to-date on the latest events and news

bit.ly/DagNews

Star the Repo

Let us know you enjoyed this presentation!

bit.ly/DagDemo