Let's talk orchestration.

An Intro to scheduling dbt with Dagster.

Sydney dbt meetup October 2021
Benoit Perigaud

aginic

Agenda

An intro to dagster and dbt

1 What are we going to talk about

2 Why we might need orchestration tools

3 Dagster, Prefect and Airflow

4 Dagster core concepts

5 Demo time!

Benoit Perigaud

Me, myself and I

- Principal Analytics Engineer at Aginic
- Data consulting company, part of the list of dbt Preferred Consulting Providers

What this talk will and won't be about

Especially won't

👍

- An intro to why you might need an orchestration tool
- A brief overview of different options in the data space
- An overview of key concepts in Dagster
- A repo that you can clone and play with

👎

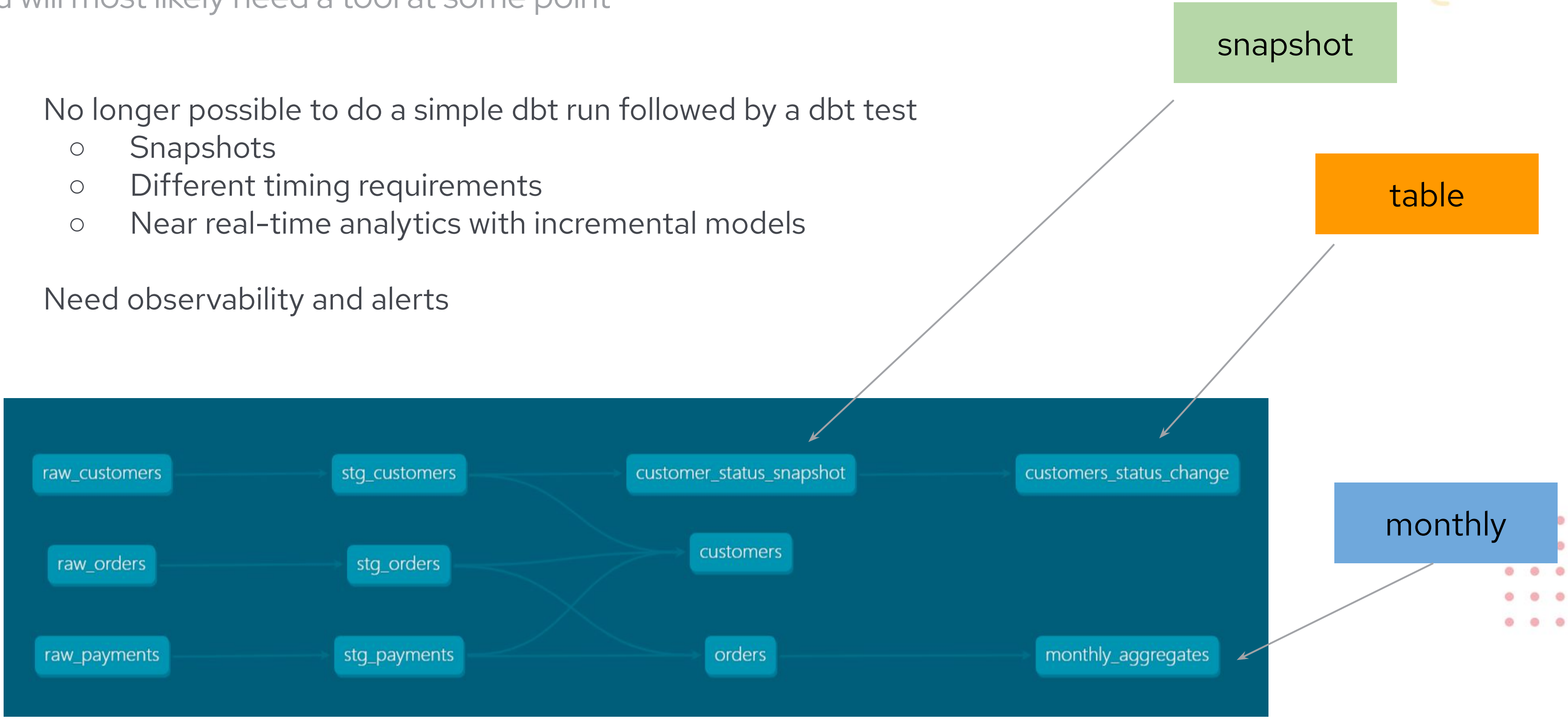
- CI/CD orchestration of dbt
- An in-depth view of which orchestration tool is the best for you
- Ins and Outs of Dagster
- How to build a Dagster project using best practices
- How to deploy Dagster

A story about dbt scheduling

You will most likely need a tool at some point

- No longer possible to do a simple dbt run followed by a dbt test
  - Snapshots
  - Different timing requirements
  - Near real-time analytics with incremental models

- Need observability and alerts



Different options

But where to start

- dbt Cloud
- automation servers
  - As part of CI/CD
- cloud capabilities
  - Azure example: Azure Function App / Azure Data Factory
  - ...
- cron 😭
- data orchestration tools
  - Airflow
  - Dasgter
  - Prefect

The 3 main players for data orchestration

Elevator pitch

Apache Airflow

"Airflow is a platform created by the community to programmatically author, schedule and monitor workflows."

- The OG
- Big community and numerous packages
- More difficult to host/manage (managed services exist)
- Local development not as easy
- Existing dbt examples

DAGSTER

"Dagster is a data orchestrator for machine learning, analytics, and ETL"

- In the process of releasing a commercial hosted version, on top of the open source package
- Cares about the "what" on top of the "how"
- Configurable IO Managers
- Concept of data assets
- Easy local testing
- Integrates with Airflow

PREFECT

"Orchestrate the modern data stack"

- Commercial cloud offering as well as open source
- Talks to "negative/positive engineering"
- More modern, like Dagster

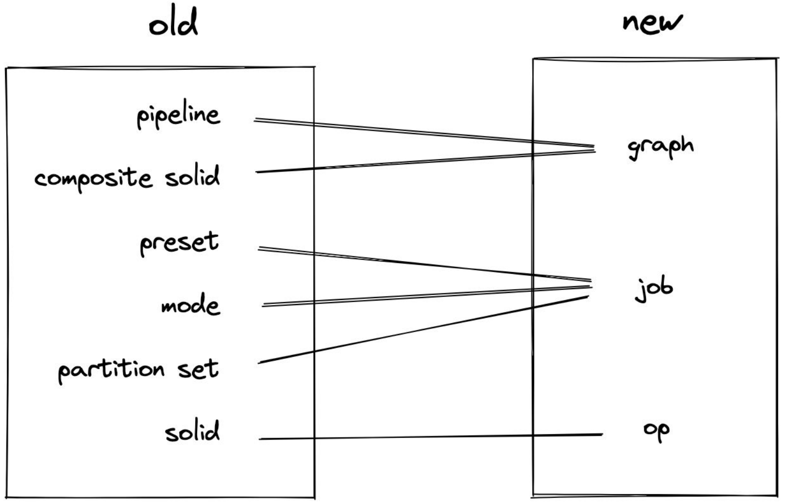
Dagster 101

Main concepts

The current way of working (as of oct 2021)

- solid
  - functional unit of work
  - read inputs, do an action, emit outputs
  - can materialise assets
- pipeline
  - links solids together
  - e.g.: solid_2(solid_1())
- preset/mode
  - run pipelines with specific configuration

This is changing



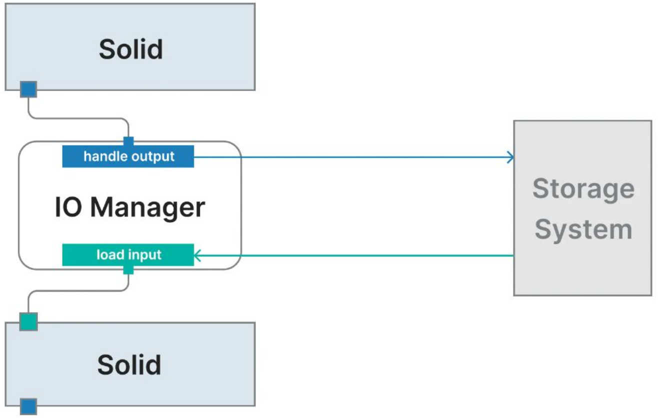
https://dagster.io/blog/core-apis-moving-towards-1-0

Other dagster concepts

Differentiators

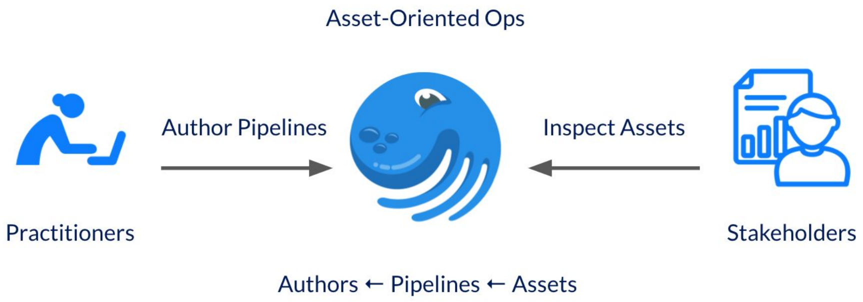
IO Managers

Ability to switch between In Memory, Local, Blob or create your own. Allow partial re-runs



Asset Materialisations

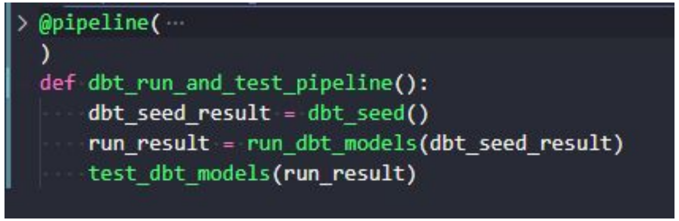
"Asset" is Dagster's word for an entity, external to solids, that is mutated or created by a solid.
Accessible through the Asset Catalog



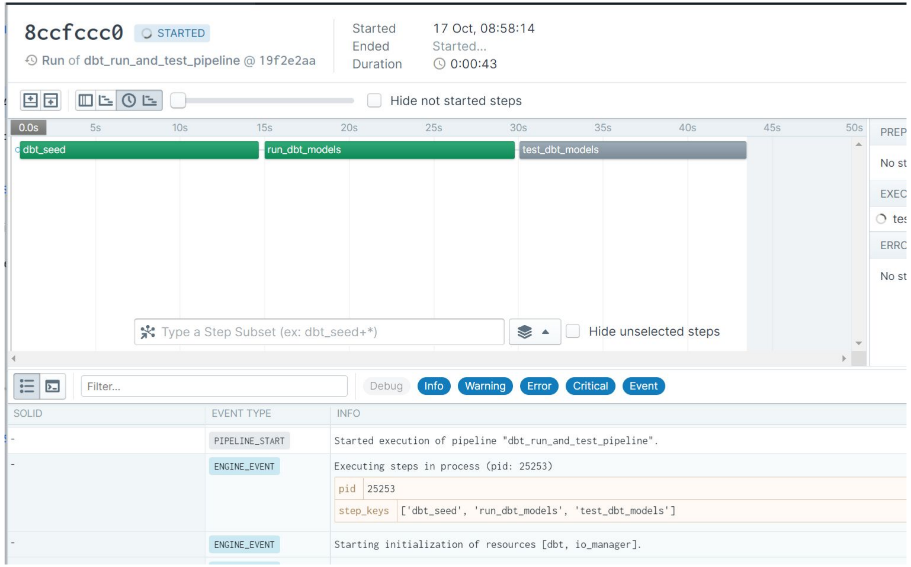
Demo!

dbt deps | dbt run | dbt test

- dagit shows the results and progress of the different dbt commands
- Ability to look at past runs and re-run them



```
> @pipeline(···
)
def dbt_run_and_test_pipeline():
    dbt_seed_result = dbt_seed()
    run_result = run_dbt_models(dbt_seed_result)
    test_dbt_models(run_result)
```

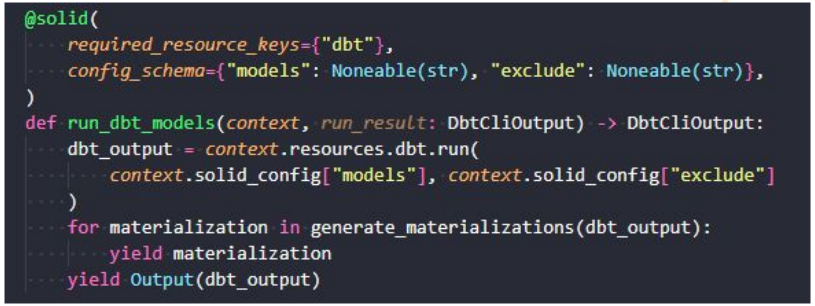


```
8ccfccc0  STARTED                                   Started   17 Oct, 08:58:14
Run of dbt_run_and_test_pipeline @ 19f2e2aa         Ended     Started...
                                                    Duration  0:00:43

Hide not started steps

0.0s  5s  10s  15s  20s  25s  30s  35s  40s  45s  50s
dbt_seed | run_dbt_models | test_dbt_models

Type a Step Subset (ex: dbt_seed+*)     Hide unselected steps

Filter...   Debug  Info  Warning  Error  Critical  Event

SOLID   EVENT TYPE       INFO
-       PIPELINE_START   Started execution of pipeline "dbt_run_and_test_pipeline".
-       ENGINE_EVENT     Executing steps in process (pid: 25253)
                         pid        25253
                         step_keys  ['dbt_seed', 'run_dbt_models', 'test_dbt_models']
-       ENGINE_EVENT     Starting initialization of resources [dbt, io_manager].
```

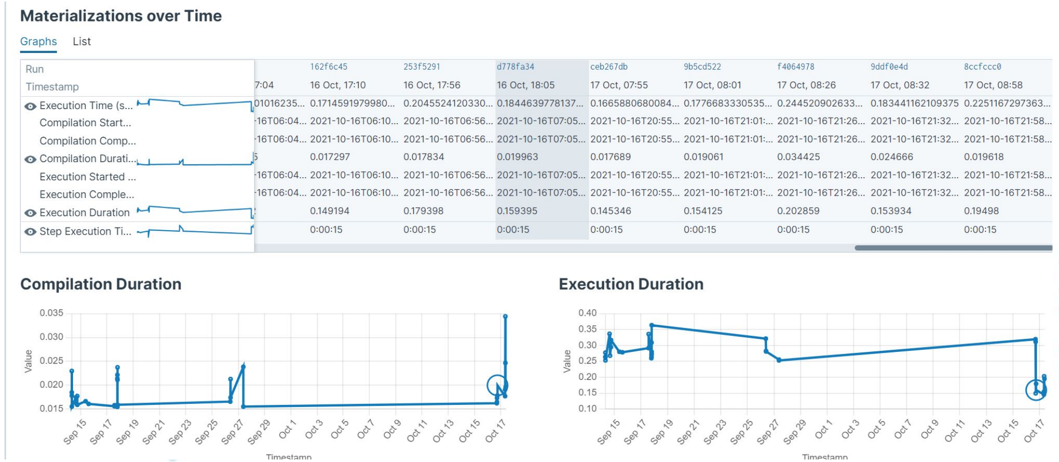
Demo!

dbt assets materialisation

- Out of the box support for Asset Materialisation of dbt models
- Reads and Parses the “run_results.json” artefact
- By default, no support for tests, but very easy to implement (see function _generate_materializations_dbt_test_)



```
@solid(
    required_resource_keys={"dbt"},
    config_schema={"models": Noneable(str), "exclude": Noneable(str)},
)
def run_dbt_models(context, run_result: DbtCliOutput) -> DbtCliOutput:
    dbt_output = context.resources.dbt.run(
        context.solid_config["models"], context.solid_config["exclude"]
    )
    for materialization in generate_materializations(dbt_output):
        yield materialization
    yield Output(dbt_output)
```



Materializations over Time

Graphs List

Run		162f6c45	253f5291	d778fa34	ceb267db	9b5cd522	f4064978	9ddf0e4d	8ccfccc0
Timestamp	7:04	16 Oct, 17:10	16 Oct, 17:56	16 Oct, 18:05	17 Oct, 07:55	17 Oct, 08:01	17 Oct, 08:26	17 Oct, 08:32	17 Oct, 08:58
Execution Time (s...	01016235...	0.1714591979980...	0.2045524120330...	0.1844639778137...	0.1665880680084...	0.1776683330535...	0.244520902633...	0.183441162109375	0.2251167297363...
Compilation Start...	-16T06:04...	2021-10-16T06:10...	2021-10-16T06:56...	2021-10-16T07:05...	2021-10-16T20:55...	2021-10-16T21:01:...	2021-10-16T21:26...	2021-10-16T21:32...	2021-10-16T21:58...
Compilation Comp...	-16T06:04...	2021-10-16T06:10...	2021-10-16T06:56...	2021-10-16T07:05...	2021-10-16T20:55...	2021-10-16T21:01:...	2021-10-16T21:26...	2021-10-16T21:32...	2021-10-16T21:58...
Compilation Durati...	5	0.017297	0.017834	0.019963	0.017689	0.019061	0.034425	0.024666	0.019618
Execution Started ...	-16T06:04...	2021-10-16T06:10...	2021-10-16T06:56...	2021-10-16T07:05...	2021-10-16T20:55...	2021-10-16T21:01:...	2021-10-16T21:26...	2021-10-16T21:32...	2021-10-16T21:58...
Execution Comple...	-16T06:04...	2021-10-16T06:10...	2021-10-16T06:56...	2021-10-16T07:05...	2021-10-16T20:55...	2021-10-16T21:01:...	2021-10-16T21:26...	2021-10-16T21:32...	2021-10-16T21:58...
Execution Duration		0.149194	0.179398	0.159395	0.145346	0.154125	0.202859	0.153934	0.19498
Step Execution Ti...		0:00:15	0:00:15	0:00:15	0:00:15	0:00:15	0:00:15	0:00:15	0:00:15

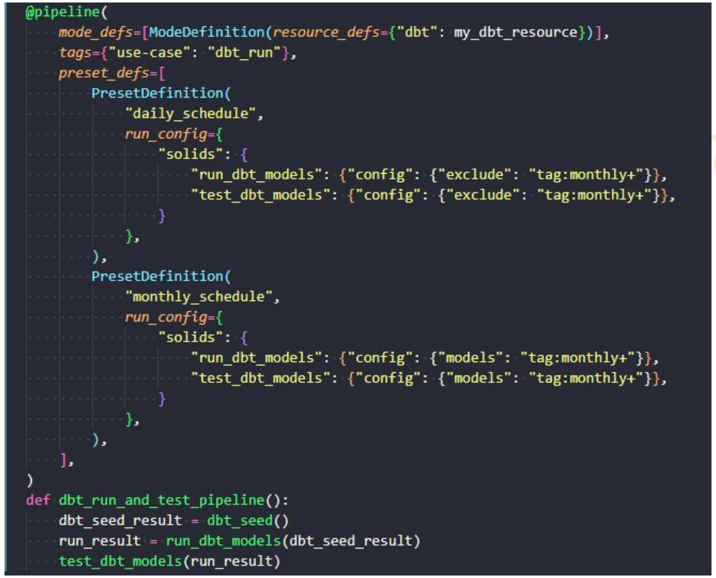
Compilation Duration

Execution Duration

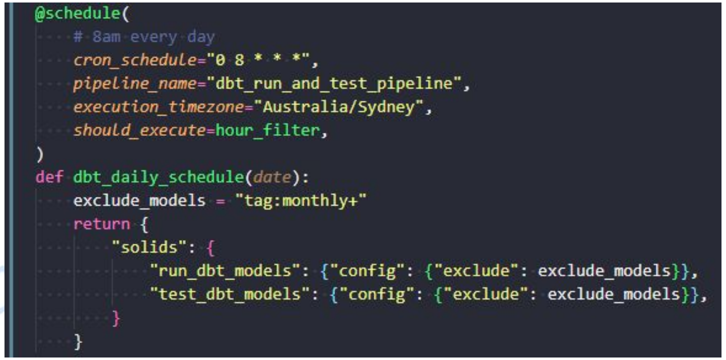
Demo!

setup presets and schedules

- presets
  - used to execute ad-hoc pipelines
  - can be loaded as YAML files
- schedules
  - cron-based
  - able to “skip” runs based on filters



```
@pipeline(
    mode_defs=[ModeDefinition(resource_defs={"dbt": my_dbt_resource})],
    tags={"use-case": "dbt_run"},
    preset_defs=[
        PresetDefinition(
            "daily_schedule",
            run_config={
                "solids": {
                    "run_dbt_models": {"config": {"exclude": "tag:monthly+"}},
                    "test_dbt_models": {"config": {"exclude": "tag:monthly+"}},
                }
            },
        ),
        PresetDefinition(
            "monthly_schedule",
            run_config={
                "solids": {
                    "run_dbt_models": {"config": {"models": "tag:monthly+"}},
                    "test_dbt_models": {"config": {"models": "tag:monthly+"}},
                }
            },
        ),
    ],
)
def dbt_run_and_test_pipeline():
    dbt_seed_result = dbt_seed()
    run_result = run_dbt_models(dbt_seed_result)
    test_dbt_models(run_result)
```

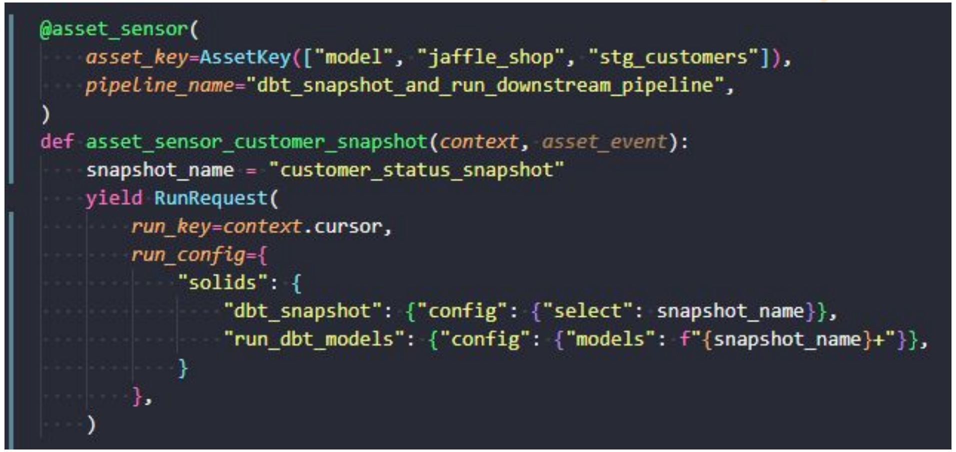


```
@schedule(
    # 8am every day
    cron_schedule="0 8 * * *",
    pipeline_name="dbt_run_and_test_pipeline",
    execution_timezone="Australia/Sydney",
    should_execute=hour_filter,
)
def dbt_daily_schedule(date):
    exclude_models = "tag:monthly+"
    return {
        "solids": {
            "run_dbt_models": {"config": {"exclude": exclude_models}},
            "test_dbt_models": {"config": {"exclude": exclude_models}},
        }
    }
```

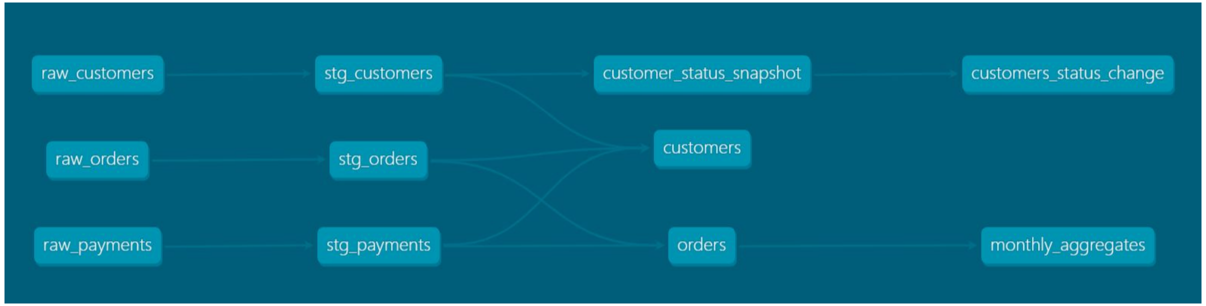
Demo!

asset sensors

- trigger actions when an asset has been updated
- useful for dbt snapshots
- allows easier maintenance and less dependencies of cross-teams data pipelines



```
@asset_sensor(
    asset_key=AssetKey(["model", "jaffle_shop", "stg_customers"]),
    pipeline_name="dbt_snapshot_and_run_downstream_pipeline",
)
def asset_sensor_customer_snapshot(context, asset_event):
    snapshot_name = "customer_status_snapshot"
    yield RunRequest(
        run_key=context.cursor,
        run_config={
            "solids": {
                "dbt_snapshot": {"config": {"select": snapshot_name}},
                "run_dbt_models": {"config": {"models": f"{snapshot_name}+"}},
            }
        },
    )
```



What next?

More configuration and going beyond dbt

- Leverage “dbt build” 🎉
- Prevent concurrent runs of dbt
    - https://docs.dagster.io/deployment/run-coordinator
- Setup alerts and/or retries when the pipeline fails
- Maybe try to generate the full dbt DAG in dagster
    - Trying with Dynamic Mapping & Collect: https://docs.dagster.io/concepts/solids-pipelines/pipelines#dynamic-mapping--collect
- Make our DAG more than dbt
    - Pause/resume EL pipeline
    - Refresh downstream applications (e.g. Power BI dataset)
    - Send notifications in slack
- Use the GraphQL API
    - Query information
    - Trigger pipelines

Thanks everyone!

That’s all folks

- This talk (slides and code)
    - https://github.com/Aginic/syd-dbt-meetup-dagster

- Dagster
    - dagster: https://dagster.io/
    - dagster and dbt: https://docs.dagster.io/integrations/dbt
    - dagster slack: https://dagster-slackin.herokuapp.com/

- dbt
    - dbt build: https://docs.getdbt.com/reference/commands/build

Questions?