Building a robust data pipeline with dbt, Airflow, and Great Expectations

Sam Bail, Superconductive / Great Expectations

Coalesce 2020

1: You should test your data.

(No, really.)

Don’t believe me?

"Our stakeholders would notice data issues before we did… which really eroded trust in the data and our team."

(A Great Expectations user)

SUPERCONDUCTIVE

"Re-running our pipelines after finding a data quality issue would incur actual costs for the compute environment."

(A Great Expectations user)

SUPERCONDUCTIVE

"Remember that one Thanksgiving where we worked all weekend to fix those data issues we only noticed at the last minute? Never again."

(That was me. True #datahorrorstory.)

But… where do we start?

(Which tool should we use? How do we know we're testing the right thing? What do we do when tests fail? Who owns this? How do we keep them up to date? How do our stakeholders find out about the state of the data?)

2: Data testing is kinda hard.

(But I can show you how to get started...)

SUPERCONDUCTIVE

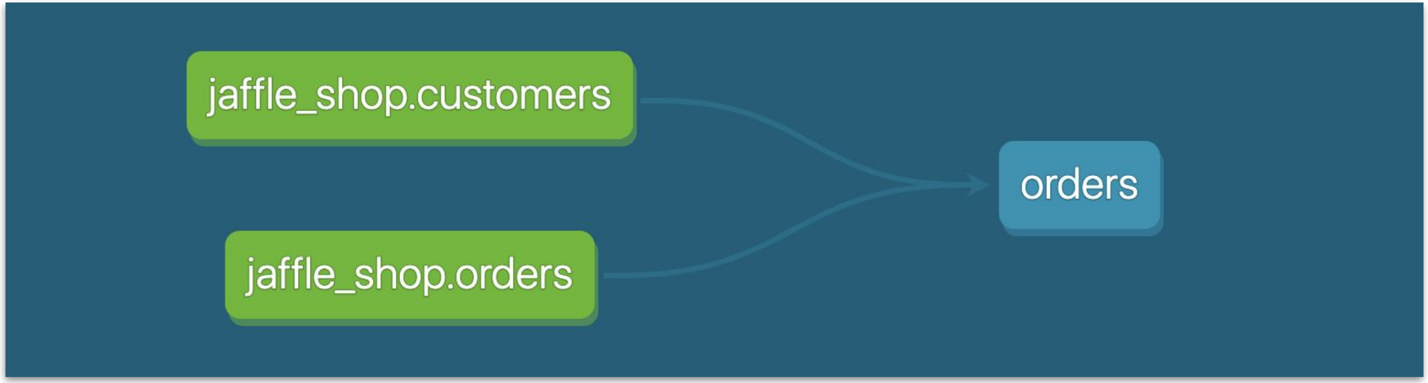
Data testing in the "dAG" Stack: dbt, Airflow*, Great Expectations

* This could be any other workflow orchestration tool, in fact, basically all of them work with dbt and Great Expectations! Hi @ dagster, Prefect, Kedro...

dbt

"The T in ELT"

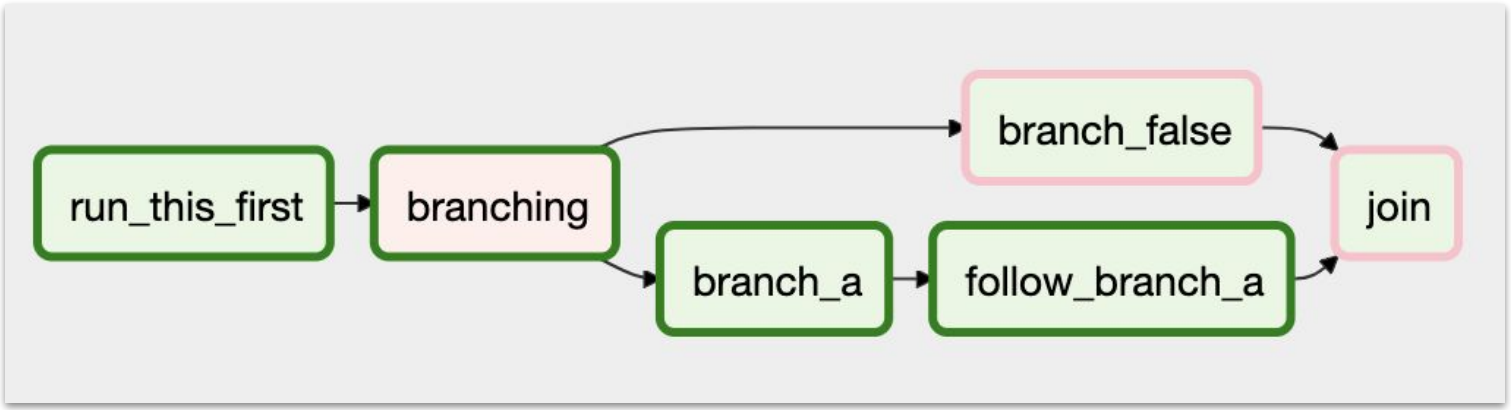
… but you know this already :)



Apache Airflow

Workflow orchestration tool

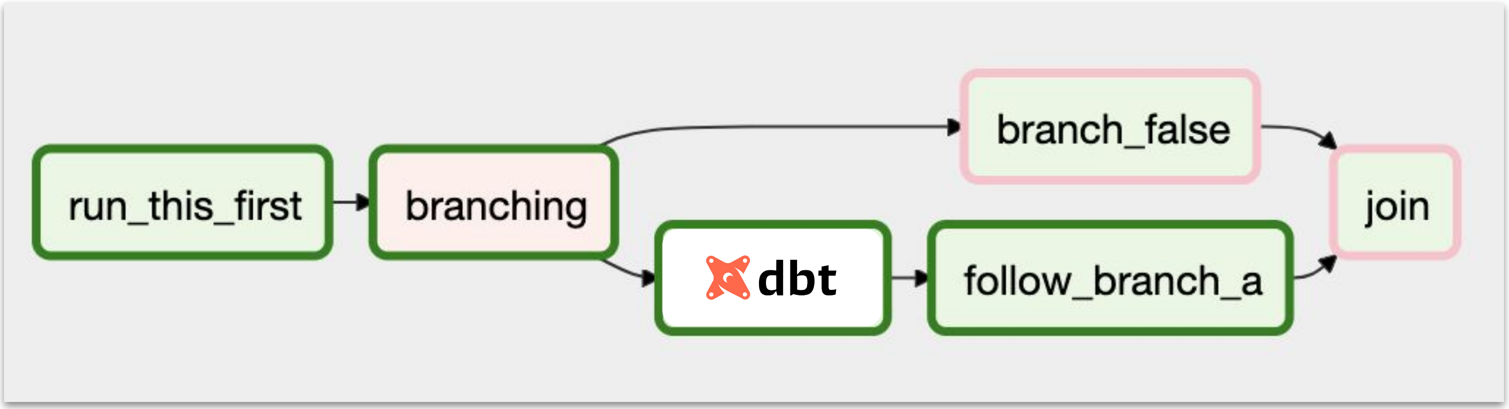
“Cron on steroids” - scheduling and more



Apache Airflow

Workflow orchestration tool

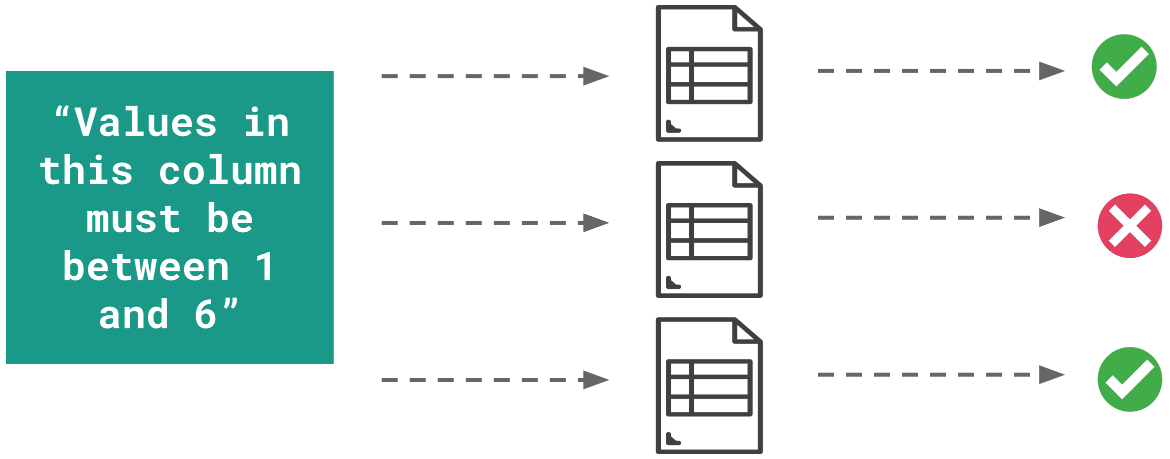
"Cron on steroids" - scheduling and more



Great Expectations

Open source data validation and documentation tool

Let's you express what you *expect* from your data (ha!)



What is an Expectation?

```
 expect_column_values_to_be_between(
      column='passenger_count',
      min_value=1,
      max_value=6
 )
```

A statement about what we expect from our data, that can be expressed in code

```
{
  "expectation_type": "expect_column_values_to_be_between",
      "kwargs": {
        "column": "passenger_count",
        "min_value": 1,
        "max_value": 6
      },
}
```

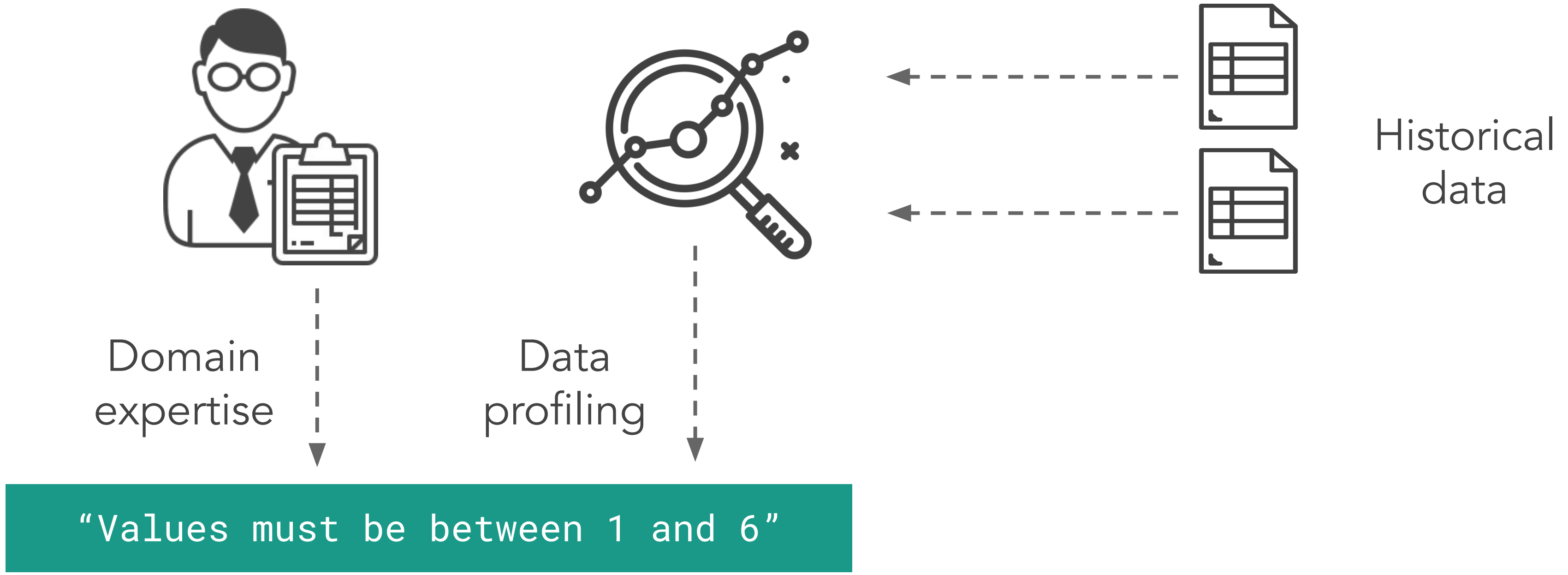
That is stored in JSON

```
"Values in this column must be between 1 and 6"
```

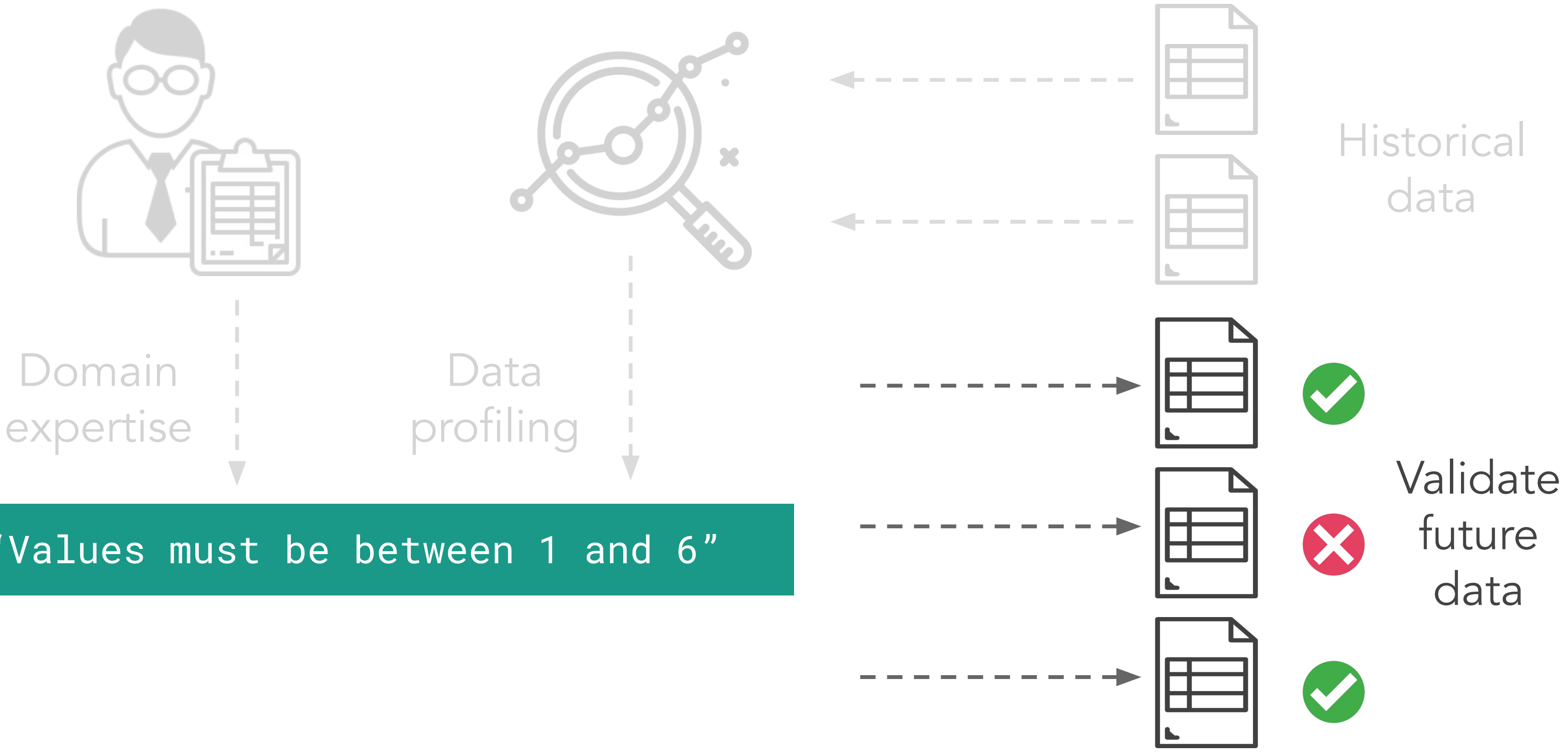
And can be translated into a human-readable format

SUPERCONDUCTIVE

Automated profiling to “scaffold” Expectations



Validating your data



SUPERCONDUCTIVE

Data Docs renders validation results into "reports"

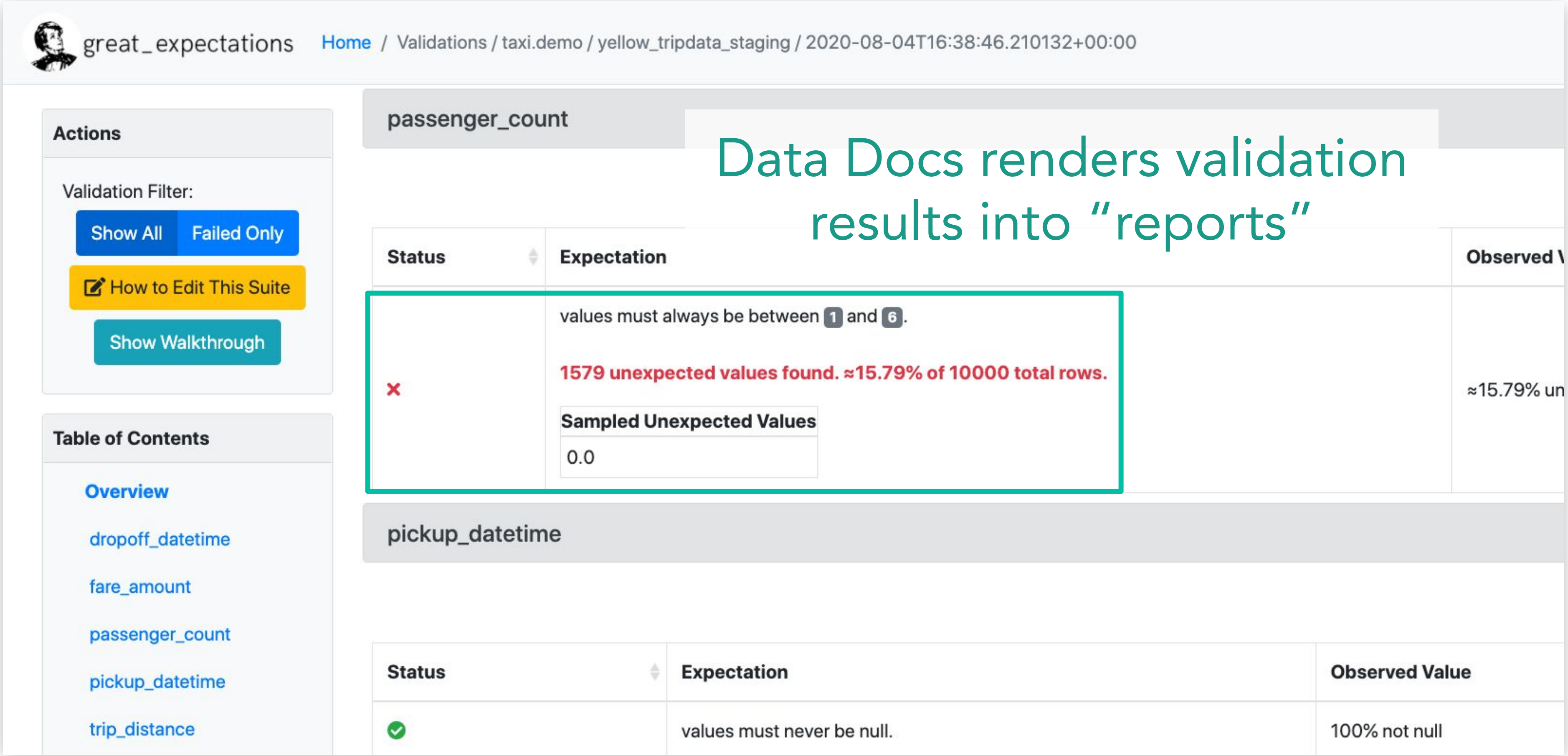

great_expectations Home / Validations / taxi.demo / yellow_tripdata_staging / 2020-08-04T16:38:46.210132+00:00

Table of Contents

passenger_count

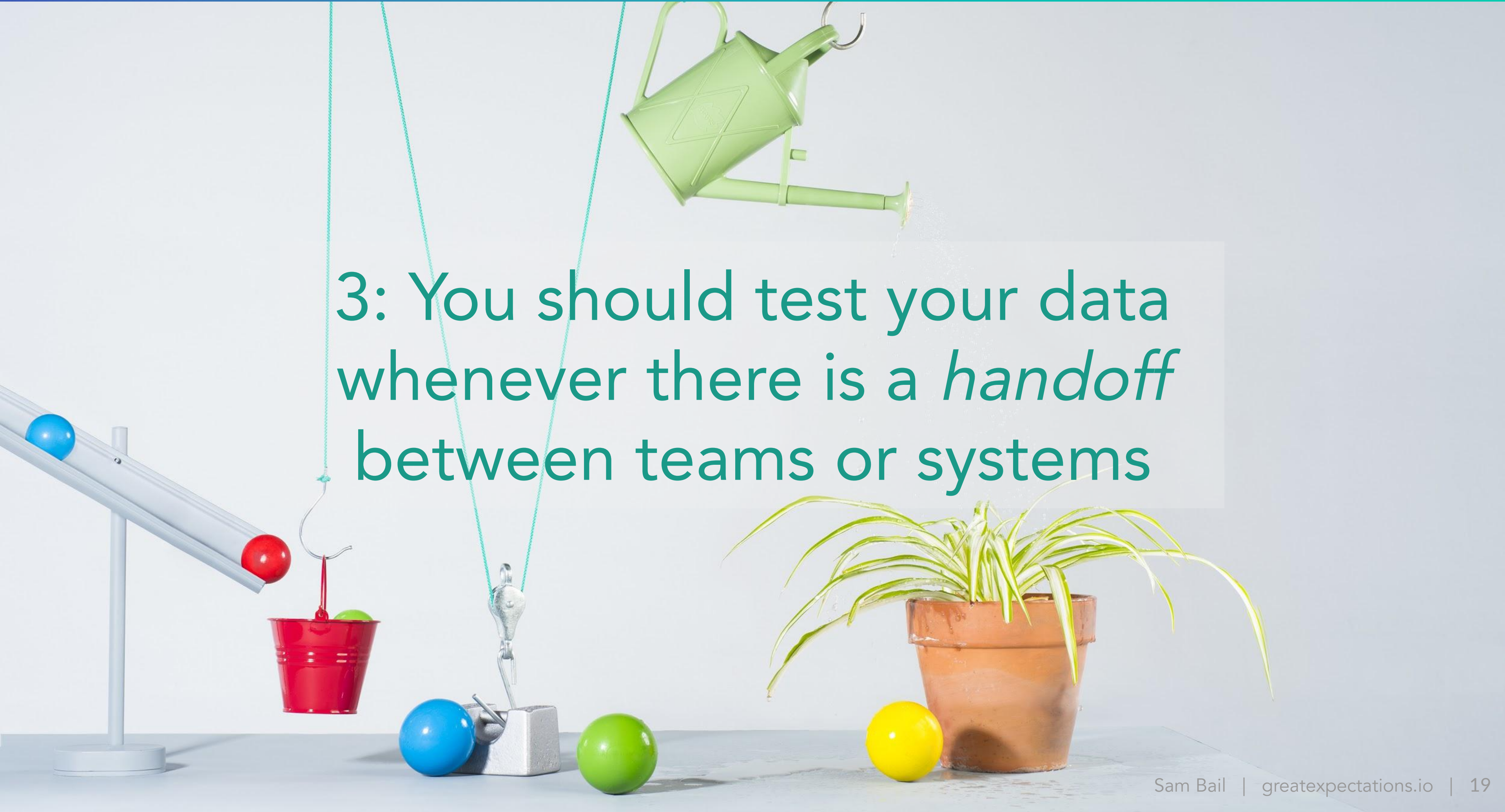
Status	Expectation	Observed V
×	values must always be between 1 and 6. 1579 unexpected values found. ≈15.79% of 10000 total rows. Sampled Unexpected Values: 0.0	≈15.79% un

pickup_datetime

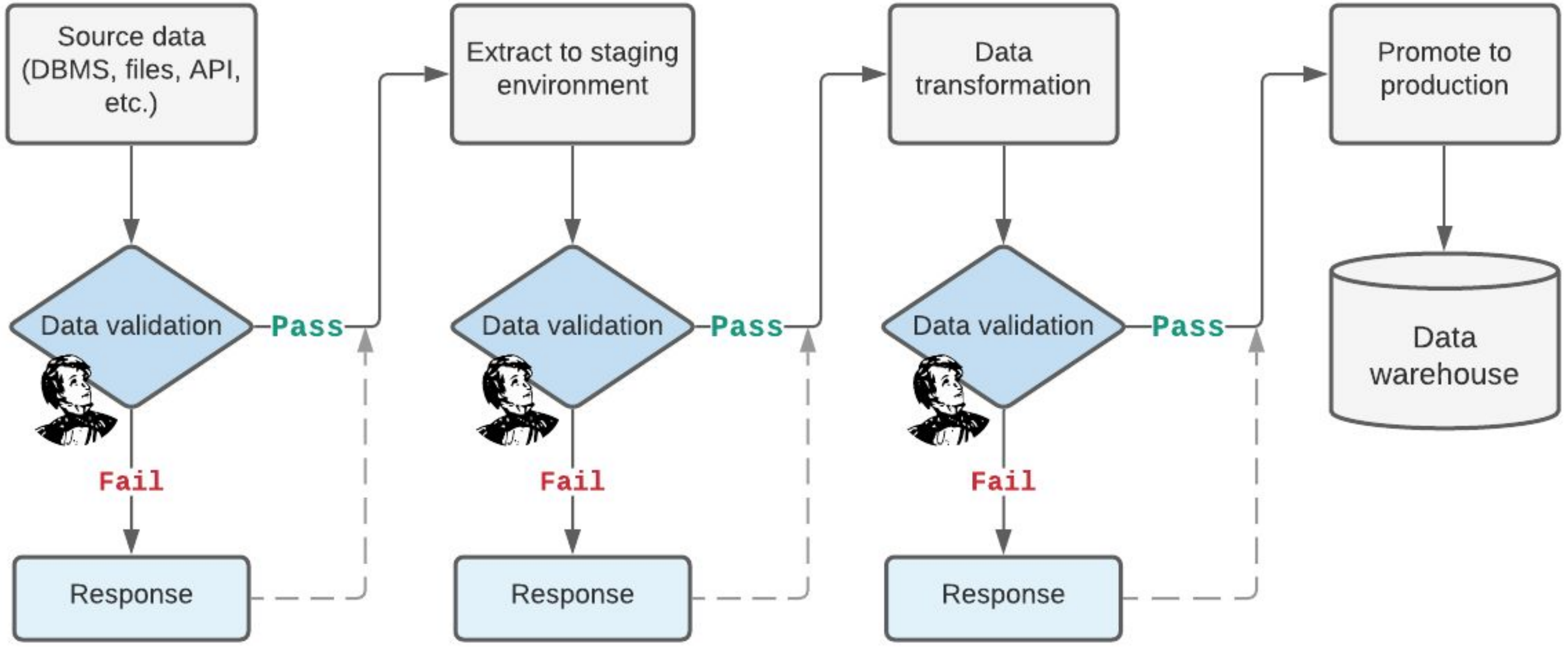
Status	Expectation	Observed Value
✓	values must never be null.	100% not null

Ok, now back to our stack: How does this all fit together?

SUPERCONDUCTIVE

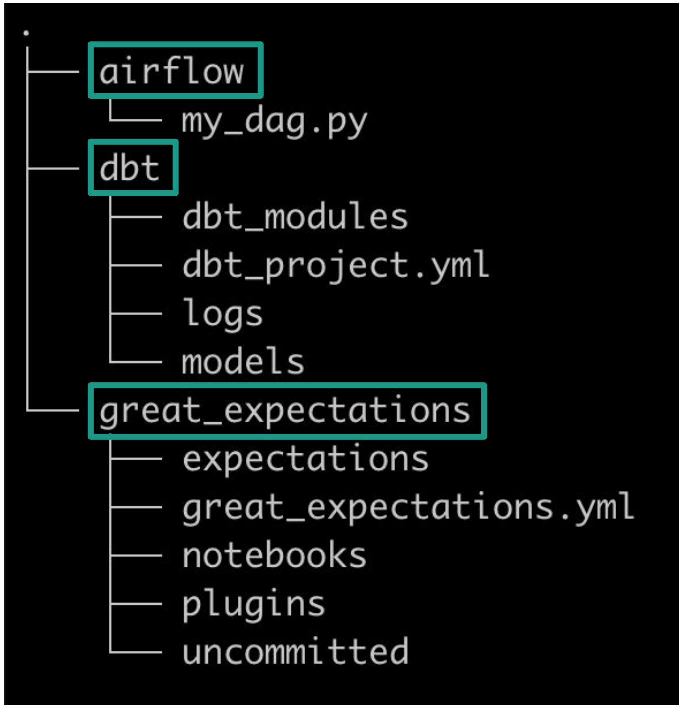


3: You should test your data whenever there is a *handoff* between teams or systems

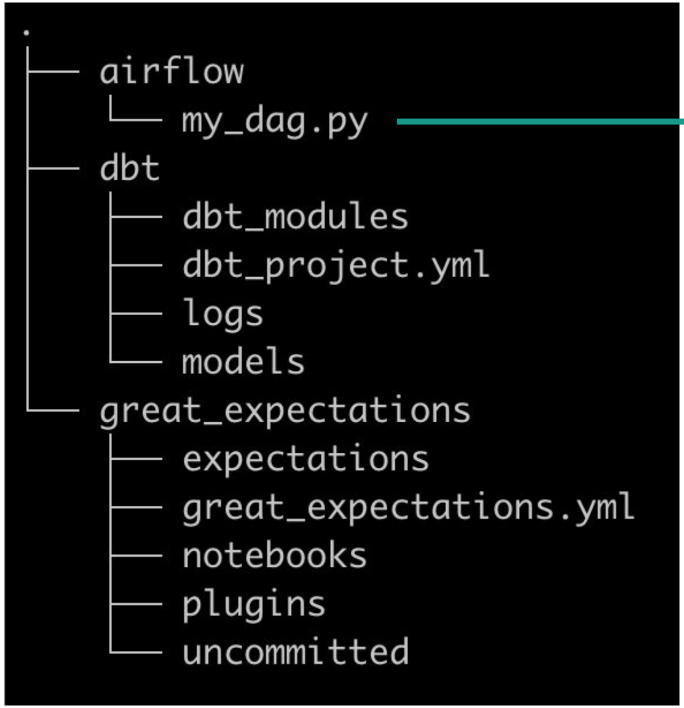

Source data (DBMS, files, API, etc.)
Data validation
Pass
Fail
Response
Extract to staging environment
Data validation
Pass
Fail
Response
Data transformation
Data validation
Pass
Fail
Response
Promote to production
Data warehouse

Ok, *now* we're back to the stack.

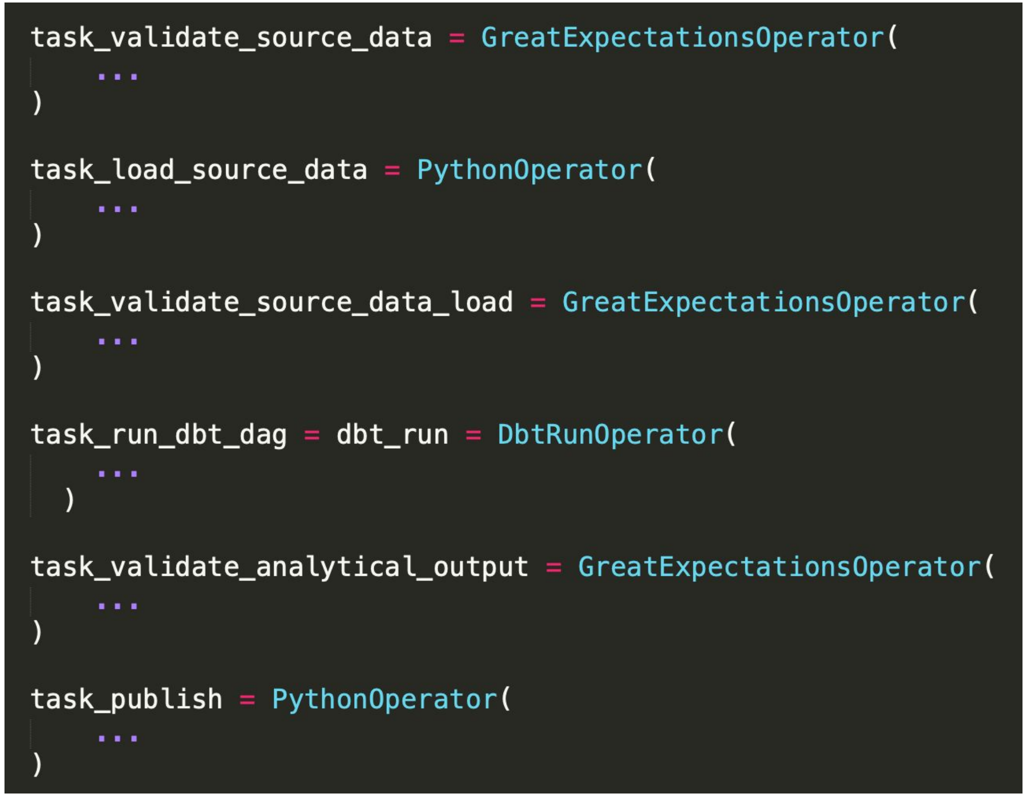
SUPERCONDUCTIVE



```
.
├── airflow
│   └── my_dag.py
├── dbt
│   ├── dbt_modules
│   ├── dbt_project.yml
│   ├── logs
│   └── models
└── great_expectations
    ├── expectations
    ├── great_expectations.yml
    ├── notebooks
    ├── plugins
    └── uncommitted
```

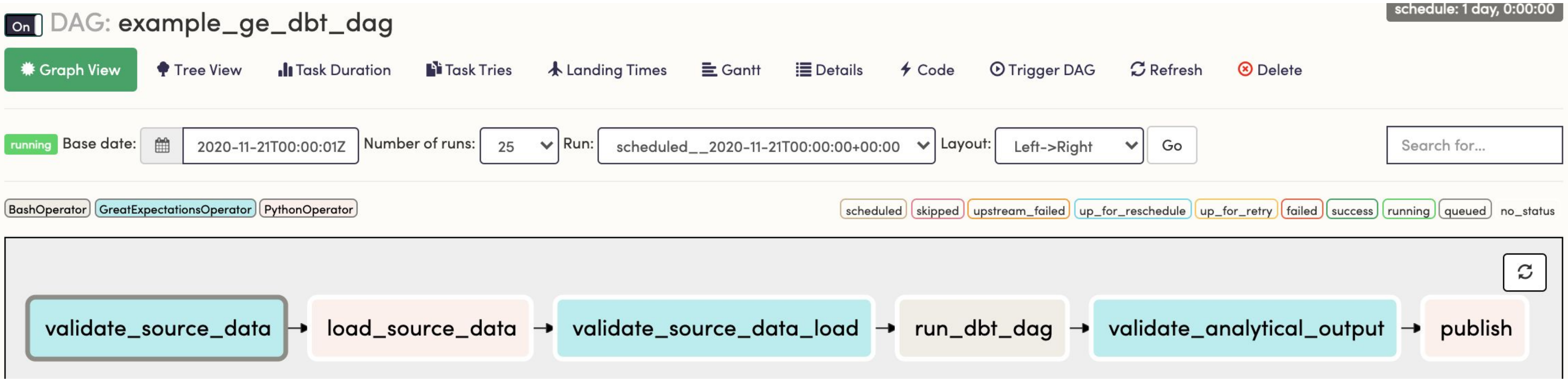


```
.
├── airflow
│   └── my_dag.py
├── dbt
│   ├── dbt_modules
│   ├── dbt_project.yml
│   ├── logs
│   └── models
└── great_expectations
    ├── expectations
    ├── great_expectations.yml
    ├── notebooks
    ├── plugins
    └── uncommitted
```



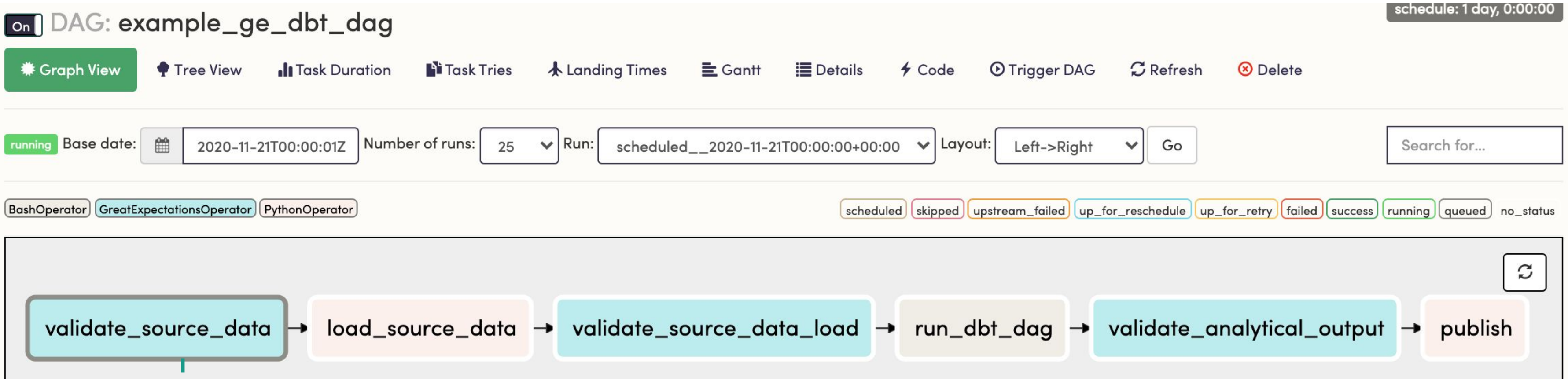
```
task_validate_source_data = GreatExpectationsOperator(
    ...
)

task_load_source_data = PythonOperator(
    ...
)

task_validate_source_data_load = GreatExpectationsOperator(
    ...
)

task_run_dbt_dag = dbt_run = DbtRunOperator(
    ...
  )

task_validate_analytical_output = GreatExpectationsOperator(
    ...
)

task_publish = PythonOperator(
    ...
)
```

SUPERCONDUCTIVE


schedule: 1 day, 0:00:00

On DAG: example_ge_dbt_dag

running Base date: 2020-11-21T00:00:01Z Number of runs: 25 Run: scheduled__2020-11-21T00:00:00+00:00 Layout: Left->Right Go



On DAG: example_ge_dbt_dag

schedule: 1 day, 0:00:00

Graph View | Tree View | Task Duration | Task Tries | Landing Times | Gantt | Details | Code | Trigger DAG | Refresh | Delete

running Base date: 2020-11-21T00:00:01Z Number of runs: 25 Run: scheduled__2020-11-21T00:00:00+00:00 Layout: Left->Right Go

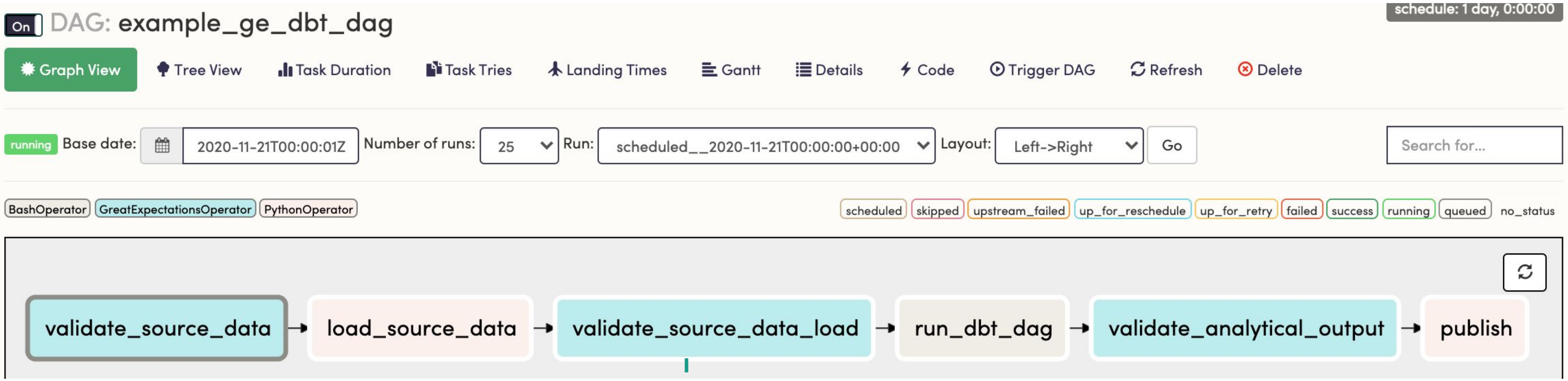
Search for...

BashOperator GreatExpectationsOperator PythonOperator

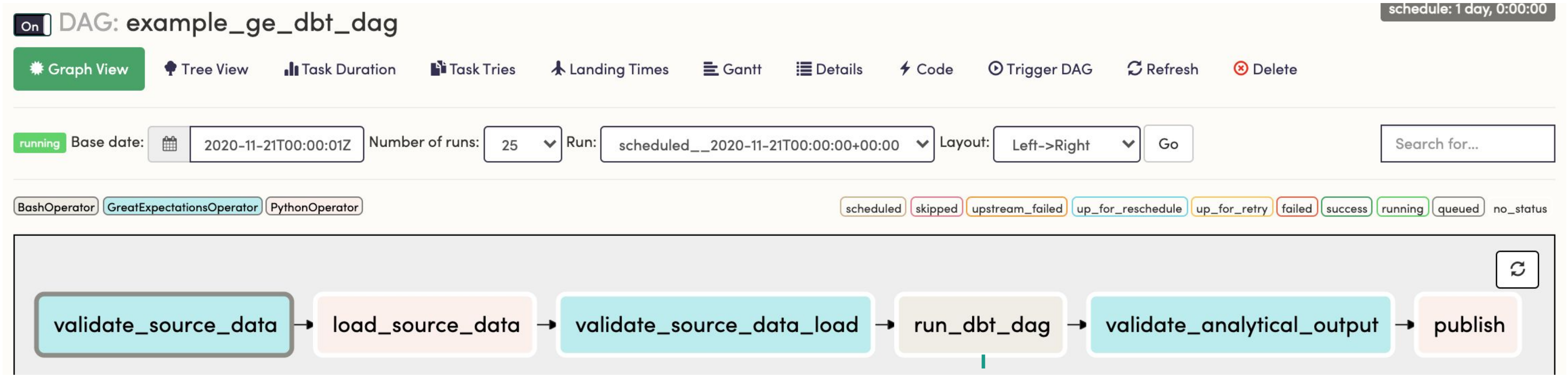
scheduled skipped upstream_failed up_for_reschedule up_for_retry failed success running queued no_status

validate_source_data → load_source_data → validate_source_data_load → run_dbt_dag → validate_analytical_output → publish

Test that source data matches expected format, e.g. correct number of columns, data types, row count “similar” to last month’s, etc.



Test that source data load was successful, e.g. no rows lost compared to source

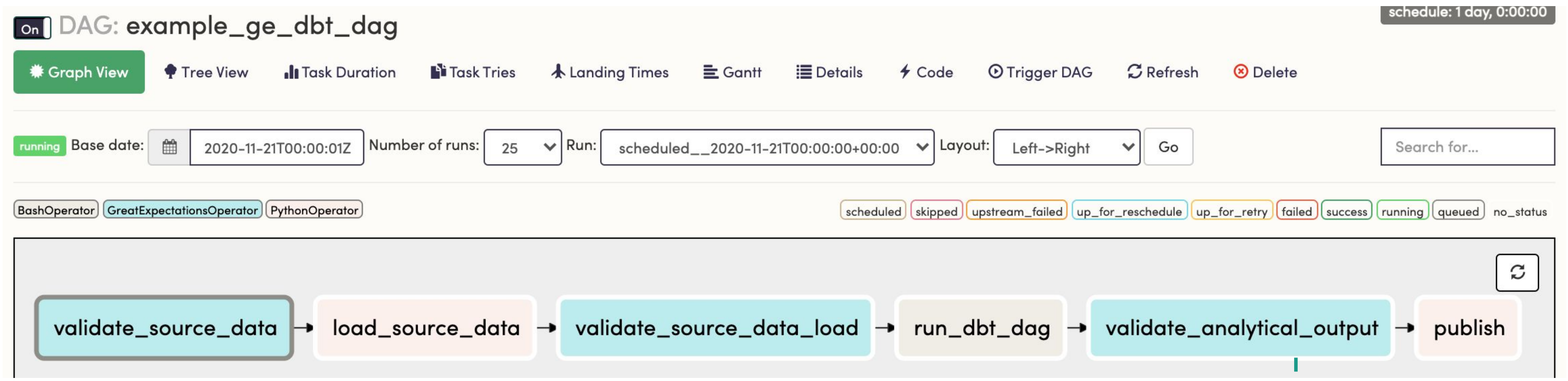


On DAG: example_ge_dbt_dag

schedule: 1 day, 0:00:00

running Base date: 2020-11-21T00:00:01Z Number of runs: 25 Run: scheduled__2020-11-21T00:00:00+00:00 Layout: Left->Right Go

dbt Run tests during DAG *development* to check for integrity of transformations



On DAG: example_ge_dbt_dag

schedule: 1 day, 0:00:00

running Base date: 2020-11-21T00:00:01Z Number of runs: 25 Run: scheduled__2020-11-21T00:00:00+00:00 Layout: Left->Right Go

validate_source_data → load_source_data → validate_source_data_load → run_dbt_dag → validate_analytical_output → publish

dbt Test integrity of transformations, e.g. no fan-out joins, no NULL columns, etc.

Use off-the-shelf methods for complex tests, e.g. distributions of values - and generate Data Docs

SUPERCONDUCTIVE

Test your data 👏

In multiple places 👏

With different types of tests 👏

Thank you!

Looking forward to chatting \o/

SUPERCONDUCTIVE