Office Hours

May 20, 2020

Claus Herther

calogica.com

Calogica

How do I keep bad data from being published in my dashboards?


Speed &
Agility

vs.


Trust &
Adoption

Using the WAP Pattern on Google BigQuery with dbt

→ **Blue/Green** Deployments

→ **WAP**

→ Data Warehouse **SLAs**

→ **Layering** your DAG

→ **Custom Schema** based on --target

→ Implementation via **Airflow** and **dbt Cloud**

Blue-Green Deployments

Via **Martin Fowler**:

1. Deploy new code to a copy (green) of the production environment (blue)
2. Test our code there, and then once we're satisfied
3. Flip a switch (router in Fowler's example) to environment with the new code [1]

[1] https://martinfowler.com/bliki/BlueGreenDeployment.html

WAP

Write-Audit-Publish

~~Stolen~~ Borrowed from the nice folks at **Netflix** [2]:

1. **Write** to partition in audit table
2. **Test** audit table
3. **Swap** partition from audit table with prod table

[2] Scaling Data Quality at Netflix

Data Warehouse SLAs

The worst thing that can happen with WAP is that your data warehouse becomes (slightly) stale!

Escalators are never broken, they just become stairs. Sorry for the convenience!

— Mitch Hedberg

Data Warehouse SLAs

Data warehouse temporarily has data from the launch of our business through two days ago, sorry for the convenience

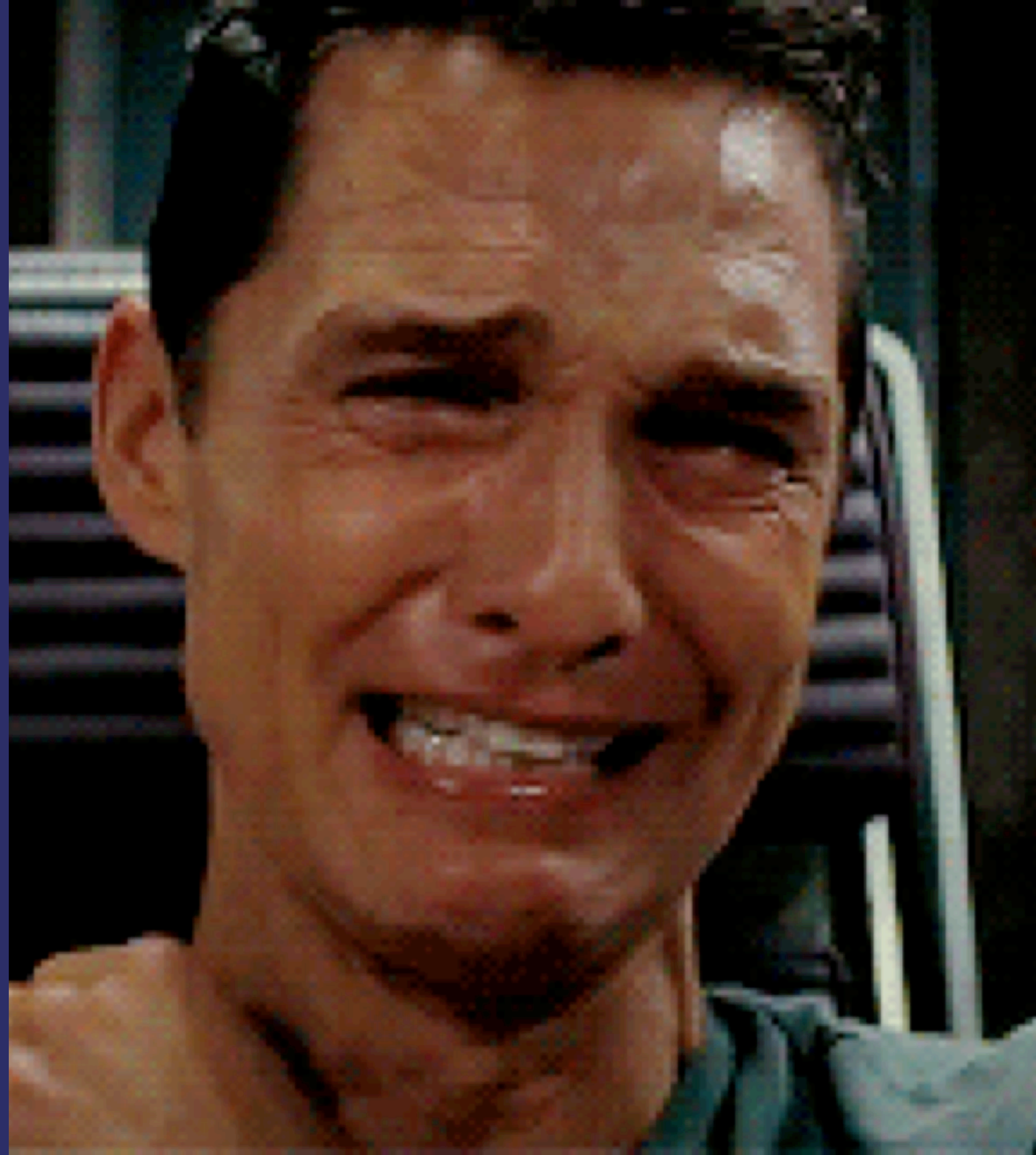
— Scott Breitenother [3]

[3] Should Your Data Warehouse Have an SLA? (Part 2)

WAP on BigQuery

→ BigQuery doesn't have partition swapping

→ BigQuery doesn't have zero-copy clones




Fake It!

WAP on BigQuery (with dbt!)

Write

`dbt run` your DAG into an audit schema/database

Audit

`dbt test` your DAG in the audit schema/database

Publish

`dbt run` **part** of your DAG *again* into the prod schema/database


Layer your
DAG

Layer your DAG

Break up your DAG so you can *selectively* publish

→ **Stage** >> **Transform** >> **DW** >> **XA**

→ Private >> Private >> **Public** >> **Public**

Layer your DAG

Stage

→ rename columns, fix data types

Transform

→ all the complicated business logic

→ *heavy* transforms

Layer your DAG

DW

→ model transforms as **Fact** or **Dimension** tables

→ *lightweight* transforms

XA = eXtended Aggregates

→ combine fact and dimension tables into denormalized reporting tables (Looker)

→ *lightweight* joins/aggregations

Publishing to the Audit Schema

Custom Schema based on --target

	audit	prod
Stage	<ephemeral>	<ephemeral>
Transform	transform	transform
DW	unaudited	<subject area>
XA	unaudited	xa

Custom Schema Macro

```
{% macro generate_schema_name_for_env(custom_schema_name=none) -%}
    {%- set default_schema = target.schema -%}
    {%- if custom_schema_name is not none -%}
        {%- if custom_schema_name not in ("stage", "transform") and
                "audit" in target.name -%}
        unaudited
        {%- else -%}
        {{ custom_schema_name | trim }}
        {%- endif -%}
    {%- else -%}
        {{ default_schema }}
    {%- endif -%}
{%- endmacro %}

{% macro generate_schema_name(schema_name, node) -%}
    {{ generate_schema_name_for_env(schema_name) }}
{%- endmacro %}
```

How do you test this locally?

Set up targets:

```
dev:
  type: bigquery
  method: service-account
  keyfile: key.json
  project: my_dev_project
  dataset: dw
  timeout_seconds: 300
  priority: interactive
  threads: 16
```

How do you test this locally?

Set up targets:

```
dev_audit:
  type: bigquery
  method: service-account
  keyfile: key.json
  project: my_dev_project
  dataset: dw
  timeout_seconds: 300
  priority: interactive
  threads: 16
```

How do you test this locally?

Set up targets:

```
prod:
  type: bigquery
  method: service-account
  keyfile: key.json
  project: prod_project
  dataset: dw
  timeout_seconds: 300
  priority: interactive
  threads: 16
```

How do you test this locally?

Set up targets:

```
prod_audit:
  type: bigquery
  method: service-account
  keyfile: key.json
  project: prod_project
  dataset: dw
  timeout_seconds: 300
  priority: interactive
  threads: 16
```

Putting it all together

```
dbt run --target prod_audit
```

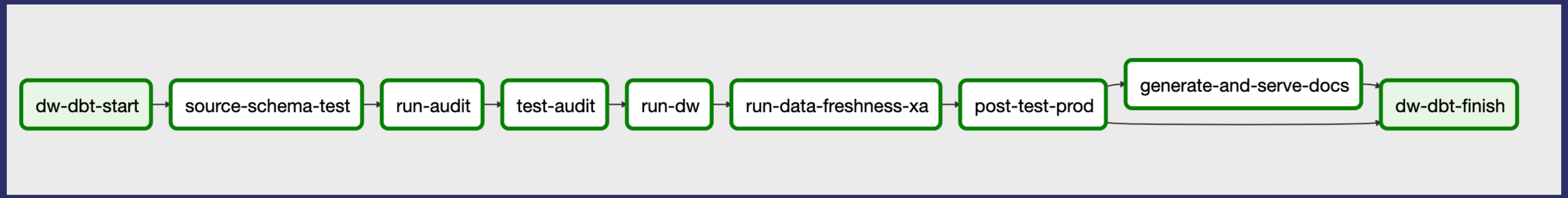
```
dbt test --target prod_audit
```

```
dbt run --target prod --models dw+
```

for the paranoid:

```
dbt test --target prod
```

Airflow



(thanks @josh !)

dbt Cloud

Instead of `target` use `var`!

```
{% macro generate_schema_name_for_env(custom_schema_name=none) -%}
    {%- set default_schema = target.schema -%}
    {%- if custom_schema_name is not none -%}
        {%- if custom_schema_name not in ("stage", "transform")  and
                ("audit" in target.name or var("audit") == true) -%}
        unaudited
        {%- else -%}
        {{ custom_schema_name | trim }}
        {%- endif -%}
    {%- else -%}
        {{ default_schema }}
    {%- endif -%}
{%- endmacro %}

{% macro generate_schema_name(schema_name, node) -%}
    {{ generate_schema_name_for_env(schema_name) }}
{%- endmacro %}
```

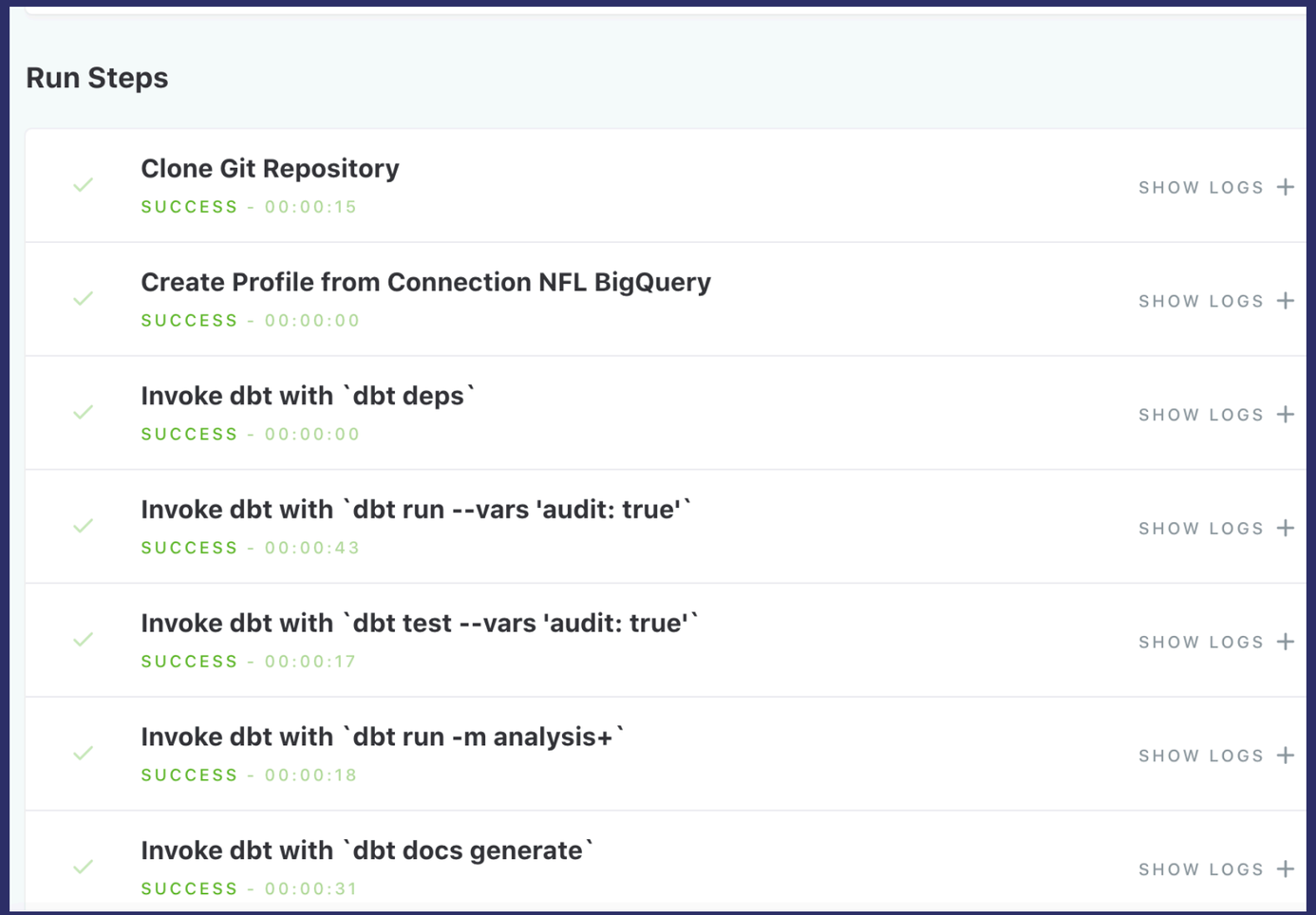
dbt Cloud

Commands

Specify which `dbt` commands this job should execute.

```
1. dbt run --vars 'audit: true'
2. dbt test --vars 'audit: true'
3. dbt run -m analysis+
```

dbt Cloud


Run Steps
Clone Git Repository
SUCCESS - 00:00:15
SHOW LOGS
Create Profile from Connection NFL BigQuery
SUCCESS - 00:00:00
SHOW LOGS
Invoke dbt with `dbt deps`
SUCCESS - 00:00:00
SHOW LOGS
Invoke dbt with `dbt run --vars 'audit: true'`
SUCCESS - 00:00:43
SHOW LOGS
Invoke dbt with `dbt test --vars 'audit: true'`
SUCCESS - 00:00:17
SHOW LOGS
Invoke dbt with `dbt run -m analysis+`
SUCCESS - 00:00:18
SHOW LOGS
Invoke dbt with `dbt docs generate`
SUCCESS - 00:00:31
SHOW LOGS

dbt Cloud

```
23:58:34 | 7 of 13 OK created incremental model audit.plays..................... [MERGE (0) in 4.92s]
23:58:35 | 9 of 13 OK created table model audit.teams........................... [CREATE TABLE (43) in 2.60s]
23:58:35 | 10 of 13 START incremental model audit.xa_field_goals................ [RUN]
23:58:35 | 11 of 13 START incremental model audit.xa_fourth_downs............... [RUN]
23:58:39 | 11 of 13 OK created incremental model audit.xa_fourth_downs.......... [MERGE (0) in 4.45s]
23:58:40 | 10 of 13 OK created incremental model audit.xa_field_goals........... [MERGE (0) in 4.76s]
23:58:40 | 8 of 13 OK created table model staging.stg_rosters................... [CREATE TABLE (13625) in 10.81s]
23:58:40 | 12 of 13 START table model audit.players............................. [RUN]
23:58:40 | 13 of 13 START table model audit.teams_players....................... [RUN]
23:58:42 | 12 of 13 OK created table model audit.players........................ [CREATE TABLE (2871) in 1.92s]
23:58:42 | 13 of 13 OK created table model audit.teams_players.................. [CREATE TABLE (13625) in 1.92s]
```

dbt Cloud

```
23:59:12 | 3 of 8 OK created table model analysis.teams..................... [CREATE TABLE (43) in 2.12s]
23:59:12 | 5 of 8 OK created table model analysis.players.................... [CREATE TABLE (2871) in 2.13s]
23:59:12 | 1 of 8 OK created table model analysis.dates...................... [CREATE TABLE (767) in 2.17s]
23:59:12 | 6 of 8 OK created table model analysis.teams_players.............. [CREATE TABLE (13625) in 2.35s]
23:59:14 | 4 of 8 OK created incremental model analysis.games................ [MERGE (0) in 3.88s]
23:59:15 | 2 of 8 OK created incremental model analysis.plays................ [MERGE (0) in 4.88s]
23:59:15 | 7 of 8 START incremental model analysis.xa_field_goals............ [RUN]
23:59:15 | 8 of 8 START incremental model analysis.xa_fourth_downs........... [RUN]
23:59:18 | 8 of 8 OK created incremental model analysis.xa_fourth_downs...... [MERGE (0) in 3.65s]
23:59:18 | 7 of 8 OK created incremental model analysis.xa_field_goals....... [MERGE (0) in 3.70s]
23:59:18 |
23:59:18 | Finished running 4 table models, 4 incremental models in 9.44s.
```

W(R)AP

→ Layer your DAG into private and public layers

→ Conditional Custom Schema macro

→ Run full DAG with `audit` flag

→ Test full DAG with `audit` flag

→ Run public layers of DAG again into `prod` schemas

Questions?

claus@calogica.com