Data's Inferno

- ~~7~~ 7 layers of data testing hell

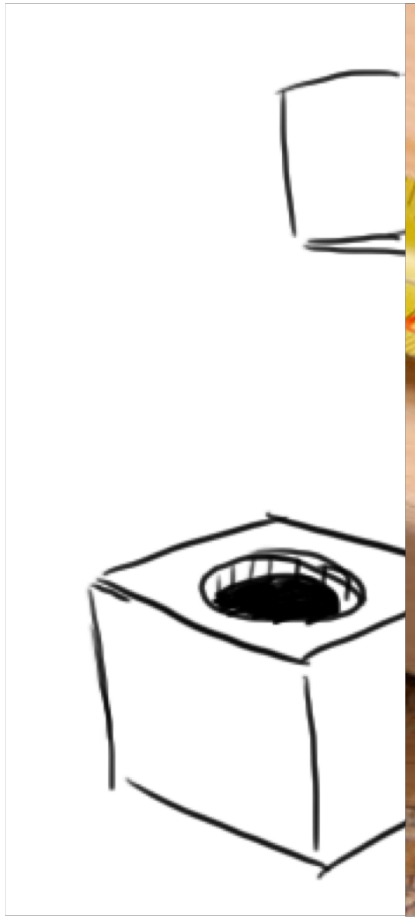
Who are we?

- Globally active bank, based in Amsterdam, the Netherlands
- 52.000 employees, 38.4 million customers
- Wholesale Banking Advanced Analytics (WBAA)
  - Works for the corporate clients, like Shell and Unilever
  - Consists of mostly Data Scientists(booo!) and Data Engineers(yeaaaah!)
  - Build data-driven algorithmic products

- Big Data and Data Science Consultancy, based in Amsterdam, The Netherlands
- 40 people, 2/3 data scientist, 1/3 data engineer
- Provide expertise in machine learning, big data, cloud, and scalable architectures
- Help organizations to become more data-driven
- Develop production-ready data applications

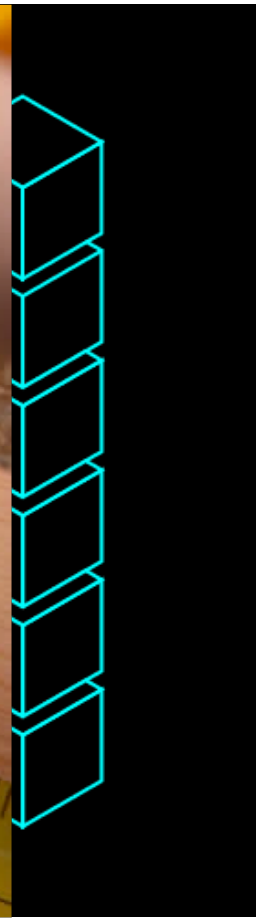
Real data sucks

Square peg, round hole



Test the pegs!

So, you thought it was square?

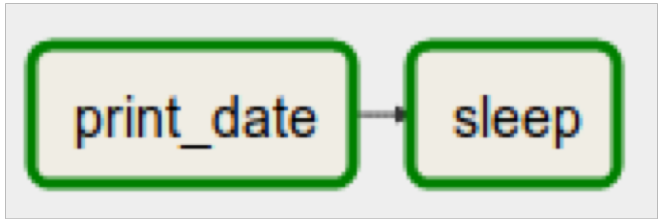


Data quality is a problem everywhere and always!

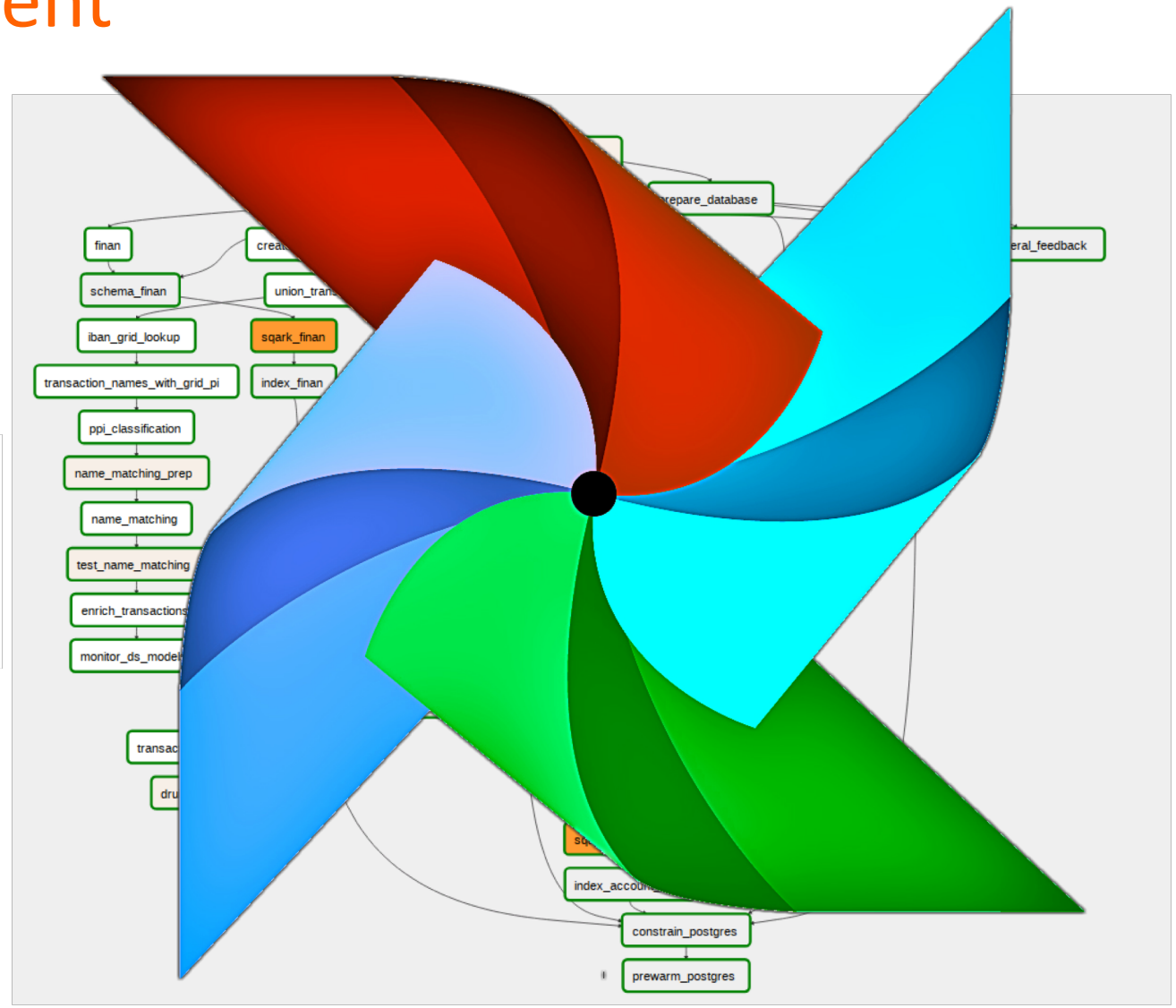
Airflow 101

Define your data pipeline in Python, as a (complex) sequence of tasks called a Directed Acyclic Graph (DAG) which can be scheduled

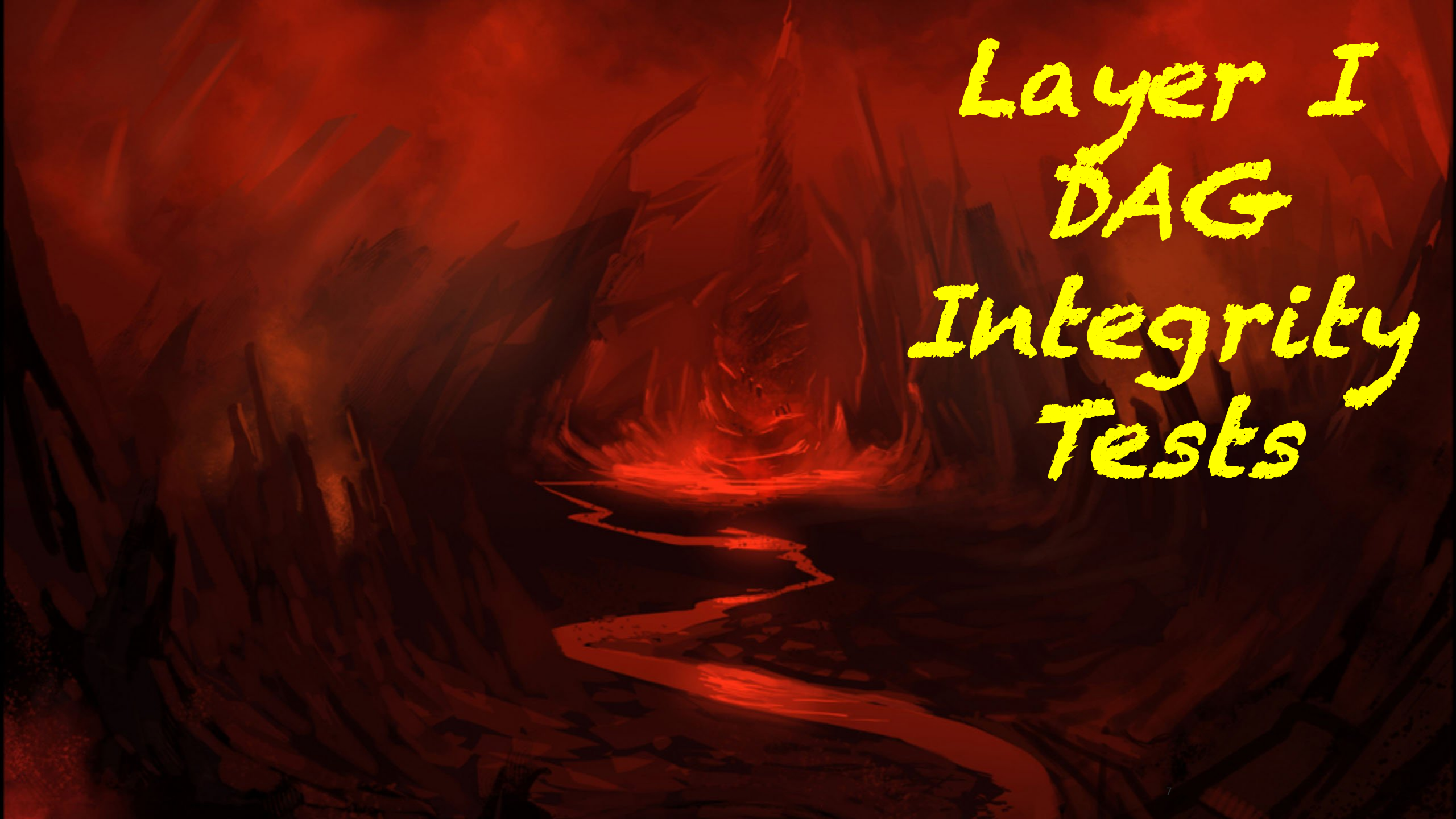
```
t1 = BashOperator(
  task_id='print_date',
  bash_command='date',
  dag=dag)

t2 = BashOperator(
  task_id='sleep',
  bash_command='sleep 5',
  retries=3,
  dag=dag)

t2.set_upstream(t1)
```



Our environment



Seems like a highway to hell?



Layer I DAG Integrity Tests

DAG integrity test (Layer 1)

CI

2 main use cases:

- Airflow version upgrade testing in Continuous Integration pipeline (CI)
- Sanity and typo checking of our DAGs

Broken DAG: [/usr/share/airflow/dags/finan_ingestion/airflowfile.py] No module named ssh_operator

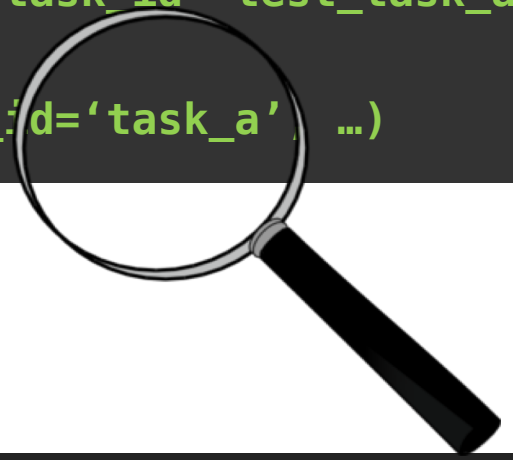
ssh_execute_operator → ssh_operator

ING

DAG integrity test (Layer 1)

CI

```
task_a = BashOperator(task_id='task_a', …)

test_task_a = BashOperator(task_id='test_task_a', …)

task_b = BashOperator(task_id='task_a', …)
```



```
E            airflow.exceptions.AirflowException: Cycle detected in DAG. Faulty task: <Task(BashOperator): templated>

../../../virtualenv/python3.5.3/lib/python3.5/site-packages/airflow/models.py:2364: AirflowException
========================== 1 failed in 0.83 seconds ==========================
```


ING

DAG integrity test (Layer 1)

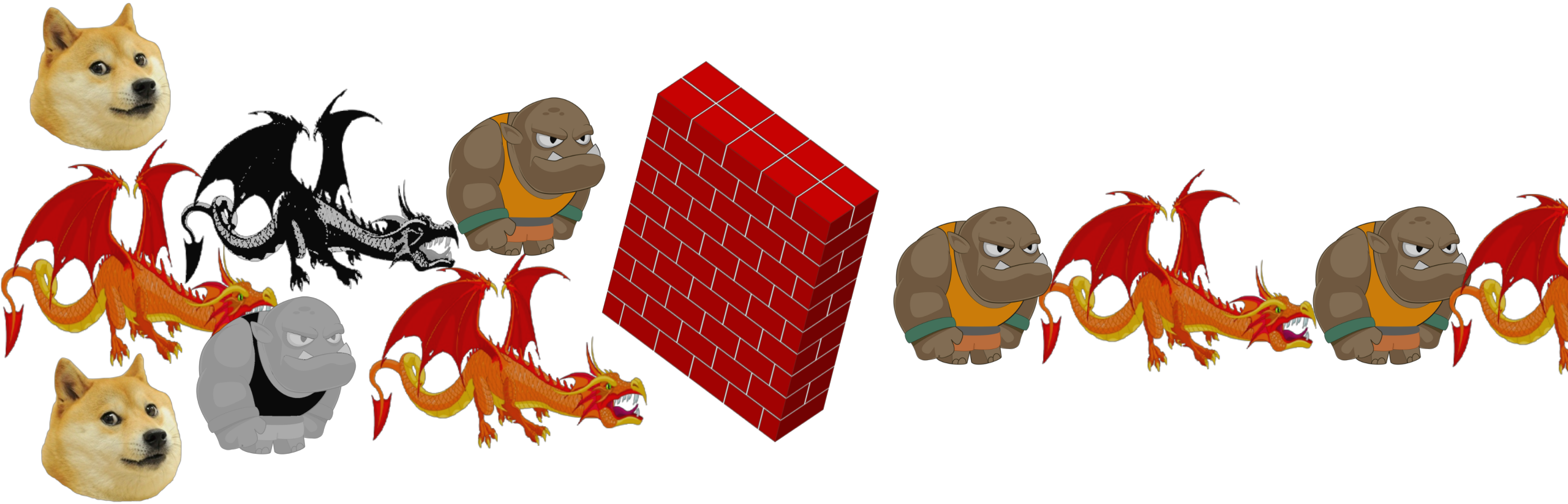
CI

```
for all DAG.py in /dags/:
    assert all objects are valid Airflow DAGs
```

- We spend a lot of time fixing typos and logic errors in DAGs, after we upload it to Airflow to actually run
- You can now do this in your CI thanks to the guys and girls at CloverHealth

Layer II
Split data ingestion from data deployments

Split your data ingestion from your data deployment (Layer 2)



Split your data ingestion from your data deployment (Layer 2)

- Data comes in from many different sources
- Create an ingestion DAG per source
- Create an interface for systems that do the same thing i.e. payment transactions
- Let your data deployment pipeline for your project work with “clean” data
- Make sure you add a debugging column from your source, like a unique ID, if none exists. Also add a column indicating from which source it came


Layer III
Data Tests

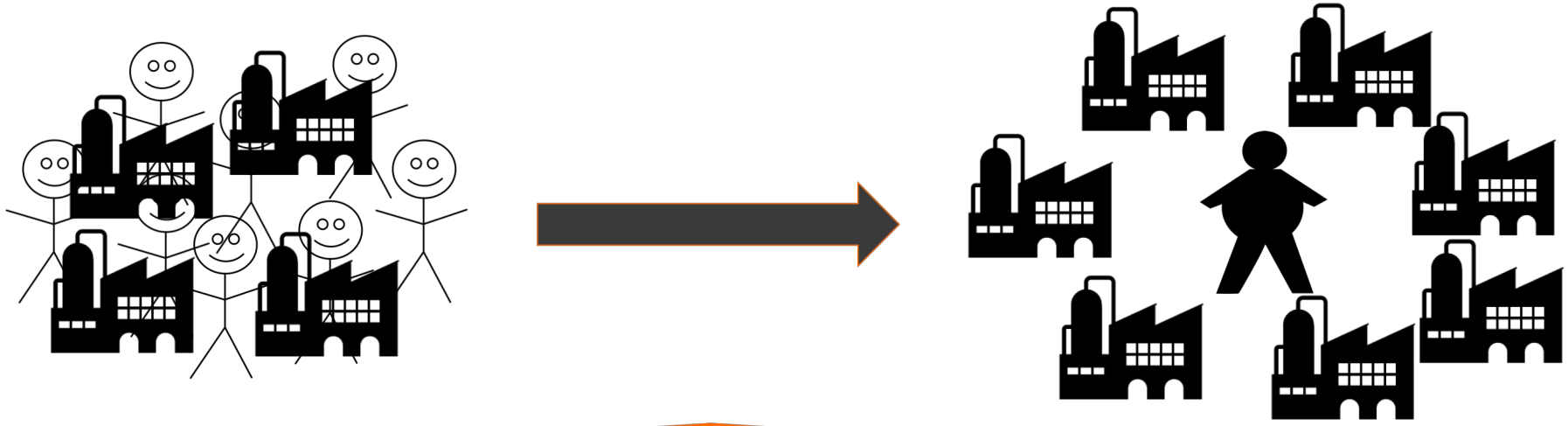
Data tests (Layer 3)

- After every action we take we have a test to check if that step has gone as expected
- We split this up in testing ingestion of data sources and testing data deployment steps

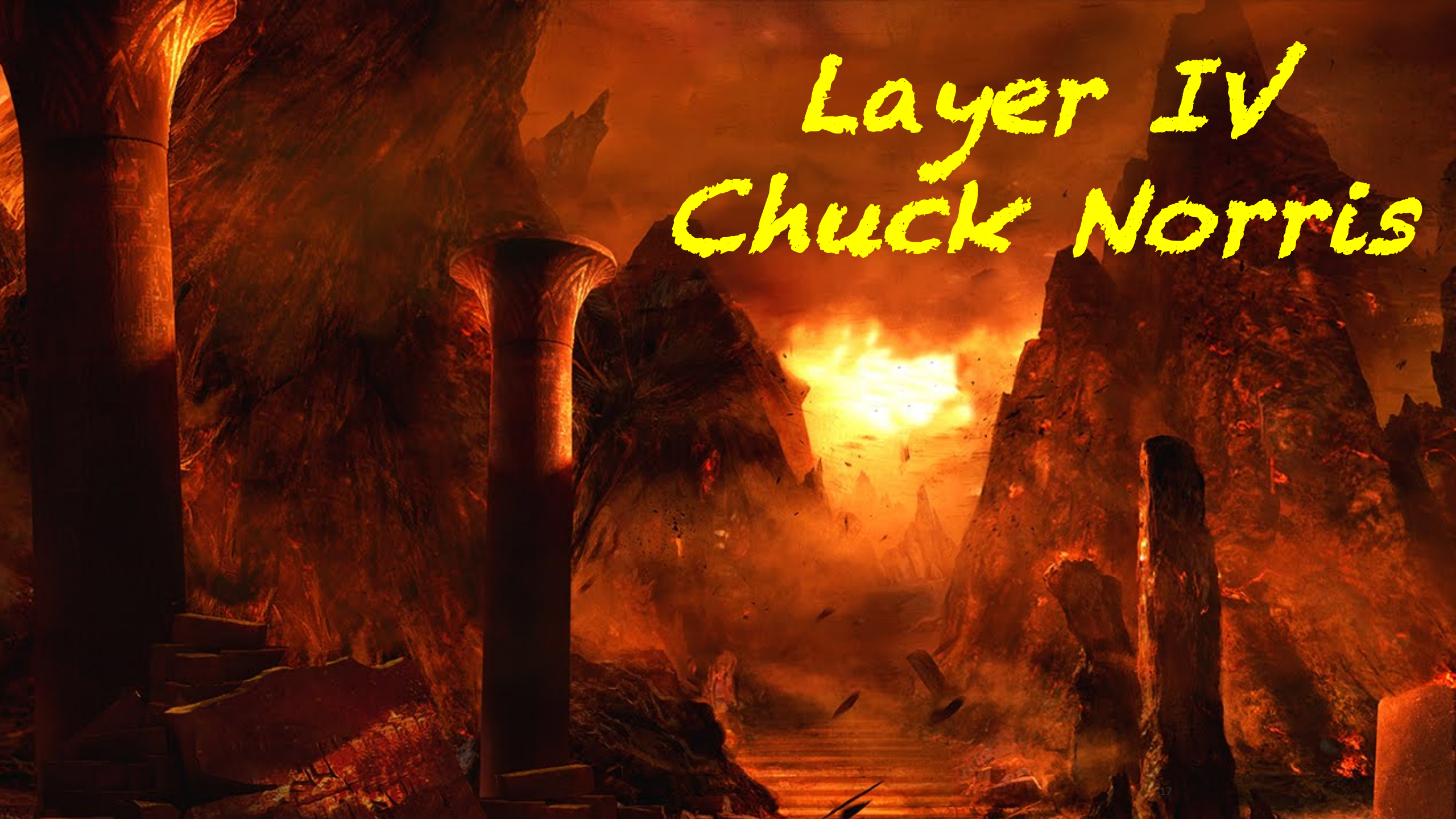
- Are there files available for ingestion?
- Did we get the columns that we expected?
- Are the rows that are in there valid? (Join it)
- Did the count only increase?

Data tests (Layer 3)

The data deployment pipeline also contains tests along the way



- Are the known private individuals filtered out?
- Are the known companies still in?
- Do all the output rows have a classification?
- Has the aggregation of PI's worked correctly?


Layer IV
Chuck Norris

Alerting by Chuck Norris (Layer 4)

Thursday, December 14th

Chuck Norris APP 00:34
data_trends Failure! Task failed: <Task(BashOperator): test_FINAN> Check log at: scheduled__2017-12-12T23:30:00

Go take a look, something blew up

Chuck Norris APP 09:23
sam Failure! Task failed: <Task(BashOperator): pyspark_data_test> Check log at: scheduled__2017-12-13T04:00:00

Chuck Norris APP 09:59
domino_data_deployment_dev Failure! Task 7-12-13T16:04:44

Chuck Norris APP 10:28
domino_data_deployment_dev Failure! Task 7-12-13T16:04:44

daniel 11:49
@aerdem @jonas ^

Chuck Norris APP 12:21
domino_data_deployment_dev Failure! Task 7-12-13T16:04:44

Chuck Norris APP 14:04
grid_ingestion Failure! Task failed: <Task(Ba 2838

Chuck Norris APP 15:02
peer_detection_dev Failure! Task failed: <Ta anual__2017-11-27T16:00:47.332361

Chuck Norris APP 15:25
domino_data_deployment_dev Failure! Task

Chuck Norris APP 15:32
peer_detection_dev Failure! Task failed: <Ta anual__2017-11-27T16:00:47.332361
domino_data_deployment_dev Failure! Task

Chuck Norris APP 16:10
domino_data_deployment_dev Failure! Task failed: <Task(SparkJDBCOperator): sqark_company_info> Check log at: manual__2017-12-14T14:14:12.173398
domino_data_deployment_dev Failure! Task failed: <Task(SparkSubmitOperator): name_matching> Check log at: manual__2017-12-14T14:14:12.173398

Chuck Norris APP 16:22
domino_data_deployment_dev Failure! Task failed: <Task(SparkSubmitOperator): name_matching> Check log at: None

1 reply 4 days ago

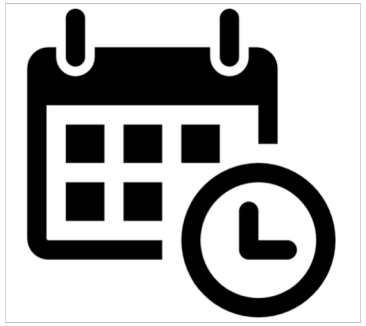
People owning their mistakes ☺

Chuck Norris APP 18:34
domino_data_deployment_dev Failure! Task failed: <Task(BashOperator): promote_branch_to_tst> Check log at: manual__2017-12-14T14:14:12.173398


Layer V
Nuclear GIT

Nuclear GIT (Layer 5)

```
if PRD≈DEV:
```



Nuclear GIT (Layer 5)

```
# this will hard reset all repos to the
version on master branch
# any local commits that have not been
pushed yet will be lost.
echo "Resetting "${dir%/*}
git fetch
git checkout -f master
git reset --hard origin/master
git clean -df
git submodule update --recursive --force
```

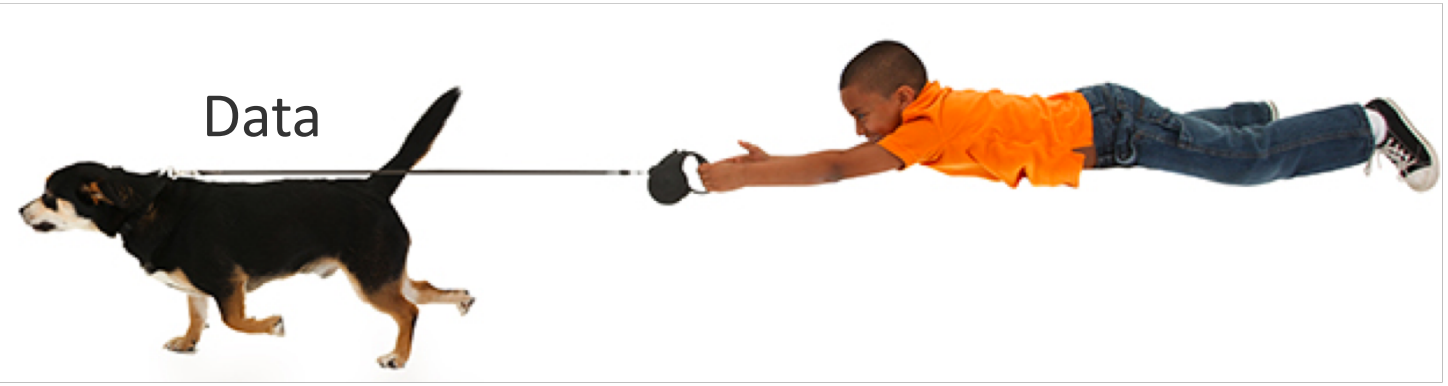
Don’t copy paste this at home!


Layer VI
Mock Pipeline
Tests

Mock pipeline tests (Layer 6)

CI

Two variables; code and data



Control the exact data that goes in to your pipeline

Code is the variable; allowing you to test your logic

Mock pipeline tests (Layer 6)

CI

Step 1: Create fake data that looks like your real data in a pytest fixture

```
PERSONS =  [('name': 'Kim Yong Un', 'country': 'North Korea',
'iban': 'NK99NKBK0000000666'), …]
TRANSACTIONS= [('iban': 'NK99NKBK0000000666', 'amount': 10 ), …]
```

Step 2: Run your code in pytest

```
filter_data(spark, PERSONS, TRANSACTIONS)
```

Step 3: Check if your task returns the data that you expect:

```
assert spark.sql("""SELECT COUNT(*) ct FROM filtered_data WHERE
iban = 'NK99NKBK0000000666'""").first().ct == 0
```


Layer 7
DTAP

DTAP (Layer 7)

So.

DTAP (Layer 7)

DEV

- Quickly run your pipeline on a very small subset of your data
- In our case 0.0025% of all data
- Nothing will make sense, but it's a nice integration test

TST

- Select a subset of your data for data that you know
- Immediately see if something is off
- Still quick to run

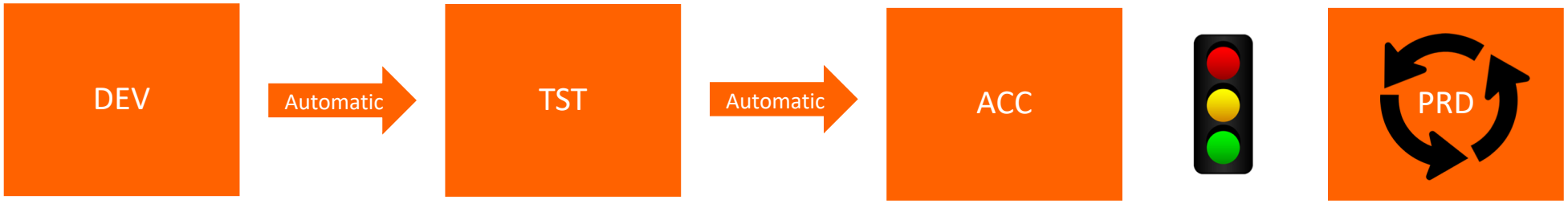
ACC

- Carbon copy of production
- You can check if you feel comfortable pushing to PRD
- Give access to a Product Owner for them to check

PRD

- Greenlight procedure for merging from ACC to PRD
- Manual operation

DTAP (Layer 7)



- 4 branches, dev, tst, acc, prd, each separately checked out in your /dags/ directory
- An environment.conf file outside of GIT in the corresponding directory
- Automatic promotion of code from dev to tst and tst to acc if everything went “green” in the DAG
- TriggerDagRunOperator to trigger the next DAG automatically for dev to tst and tst to acc

Local testing of Airflow with Whirl

Colleagues Bas Beelen and Kris Geusebroek made some very nice improvements after our time on this project.

https://www.youtube.com/watch?v=jqK_HCOJ9Ak

High level overview:

- Data is confidential, can't take this local
- There are many different DAGs, some of which are very complex
- Whirl speeds up development by
  - Being able to reuse standard components of a DAG
  - Test your DAG locally end to end with fake data using Docker
- Open source code is in the pipeline with Bas and Kris ☺

Now take your time to understand it all!

danielvdende Change dag schedule to prevent auto-running ··· Latest commit d400cc5 19 hours ago

dags	Change dag schedule to prevent auto-running	19 hours ago
docker_files	Fix styling	22 hours ago
integrity_tests	Change dag schedule to prevent auto-running	19 hours ago
mock_pipeline_tests	Add tox.ini	3 days ago
.gitignore	Add skeleton for testing components	a month ago
.travis.yml	Bump CI python version	3 days ago
Dockerfile	Fix incorrect dag trigger	21 hours ago
LICENSE	Initial commit	a month ago
README.md	Update README with better descriptions	3 days ago
tox.ini	Add tox.ini	3 days ago

README.md

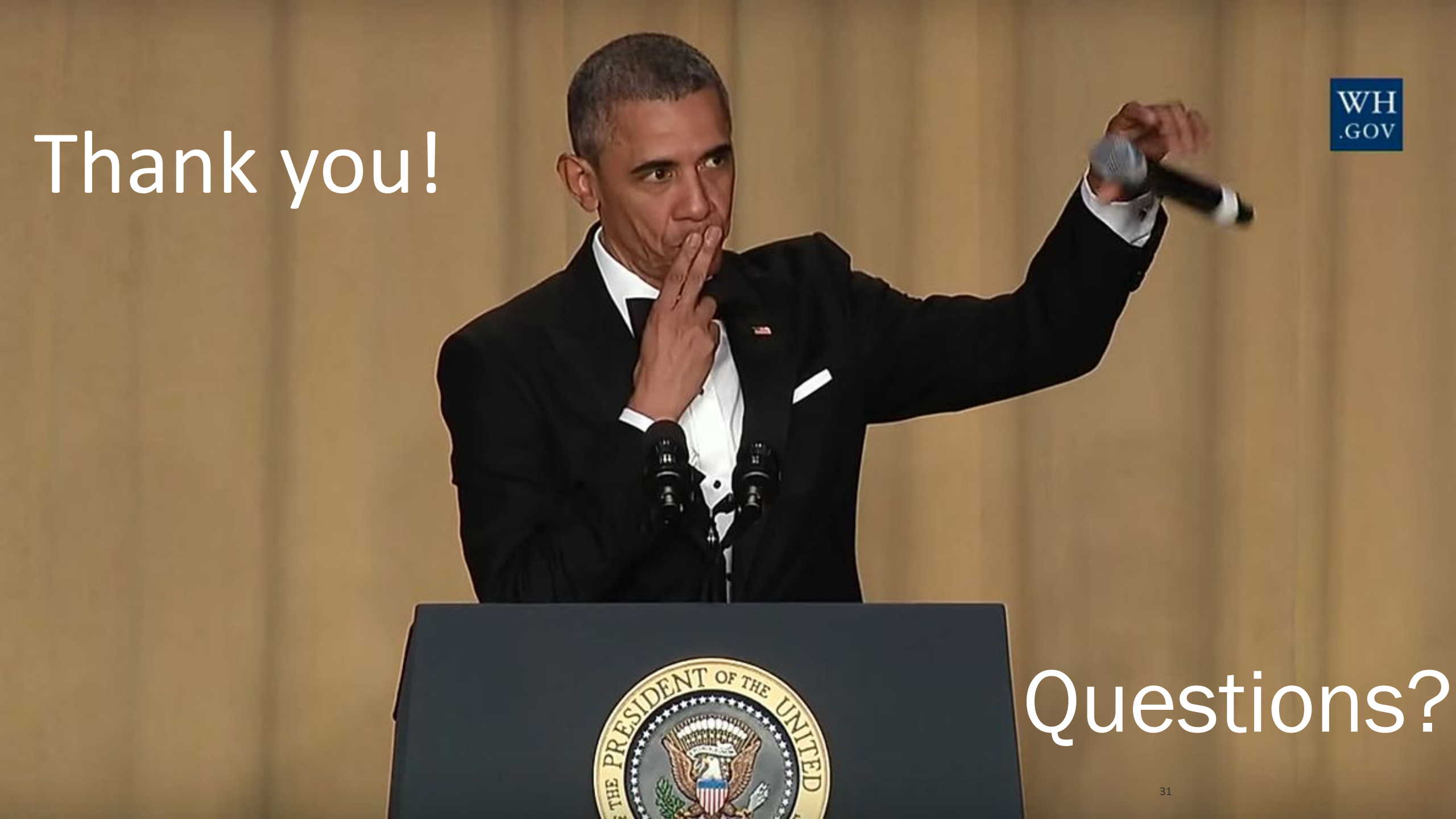
Data Testing With Airflow

build passing

This repository contains simple examples of how to implement some of the Nine Circles of Data Tests described in our blogpost. A docker container is provided to run the DTAP and Data Tests examples. The Mock Pipeline tests and DAG integrity tests are implemented in Travis CI tests.

Blogpost: https://medium.com/@ingwbaa/datas-inferno-7-circles-of-data-testing-hell-with-airflow-cef4adff58d8

Github: https://github.com/danielvdende/data-testing-with-airflow


Thank you!
WH
.GOV
Questions?
THE PRESIDENT OF THE UNITED