Intro to Apache Airflow

CCOSS 2019 - Jakob Homan & Temo Ojeda & Aizhamal Numamat kyzy

The Plan

Agenda

- Become familiar with Airflow and its architecture
  (this talk - 60 min - Jakob)
- Break *(11:30)*
- Learn about Apache (ASF) and how to contribute it
  (12 :00 - 30 min - Jakob)
- Stand up a local Airflow for testing and development
  (12:20 - 30 min - Temo)
- Lunch *(13:00 - 60 min)*
- Create and submit your first patch to Airflow
  (15:30 - Temo and Jakob and Aizhamal)

Who we are

Jakob Homan

Apache Airflow committer, PMC Member

Data @ Lyft

Temo Ojeda

Apache Airflow contributor

Data @ Lyft

Opportunities to contribute

Low bar to contribution, good for new users

Medium effort, a significant contribution

Big architectural changes that drive Airflow's future

Shout out questions, comments

Creative commons: https://commons.wikimedia.org/wiki/File:Arti_Hands_up.jpg

This talk's goal

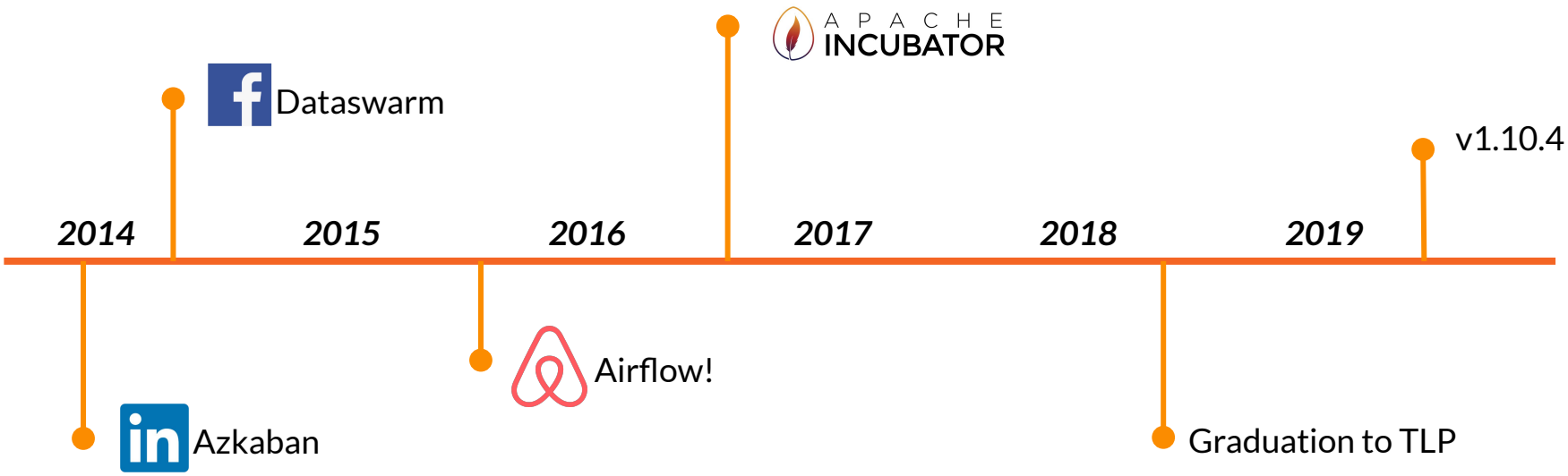
- Clear understanding of what Airflow is (and is not)
- Case study of a common Airflow use case
- Medium dive into how an Airflow clusters
- Identify lots of areas for future Airflow improvement
- Not deep diving into all of Airflow's features:
    - Missing out: xcoms, cli, lots of Operator configs, backfills, etc.

Airflow's backstory

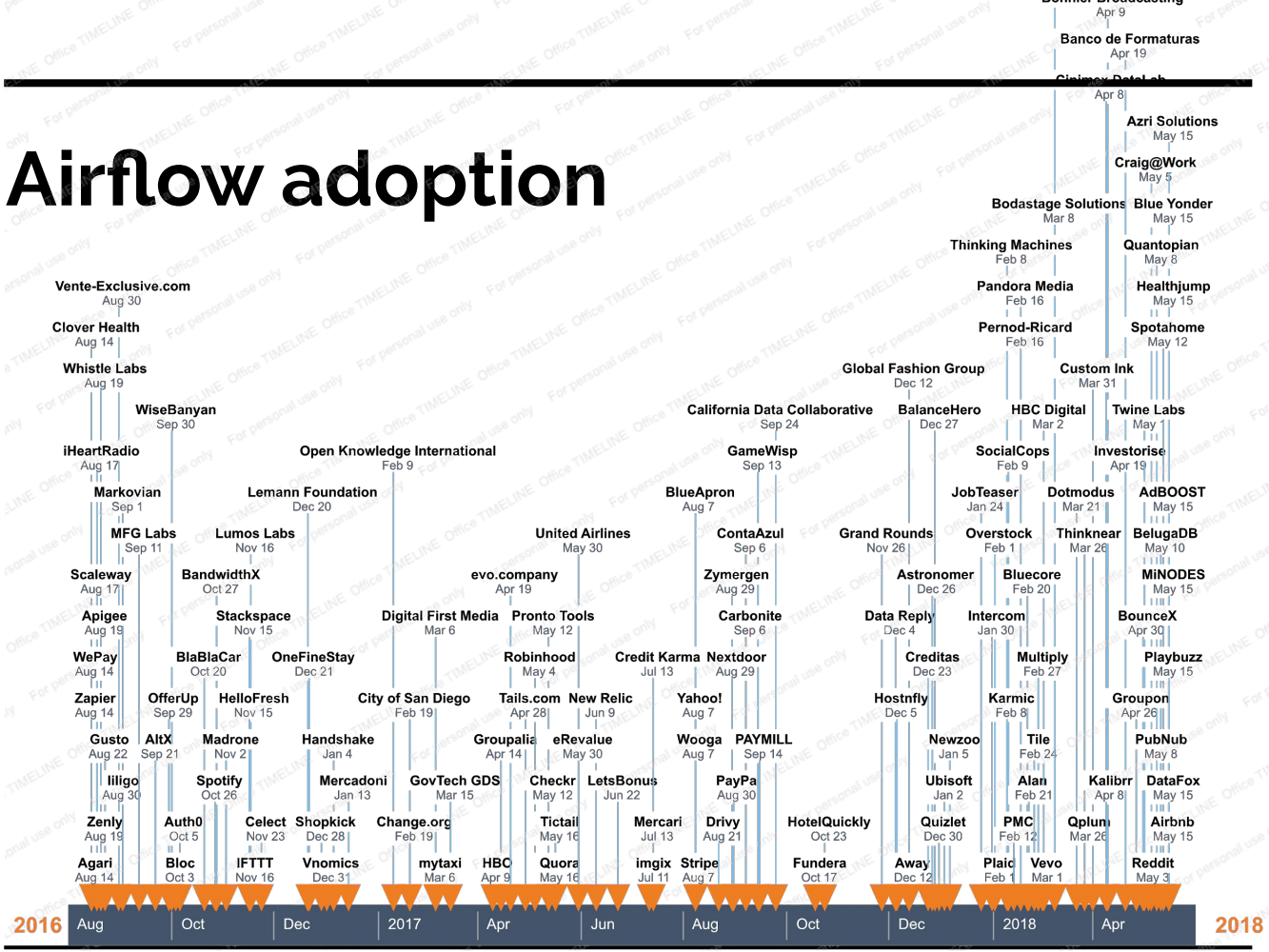
So, what is Airflow anyway?

Apache Airflow is a
super-duper
ultra-mega
totally awesome
workflow scheduler

The timeline

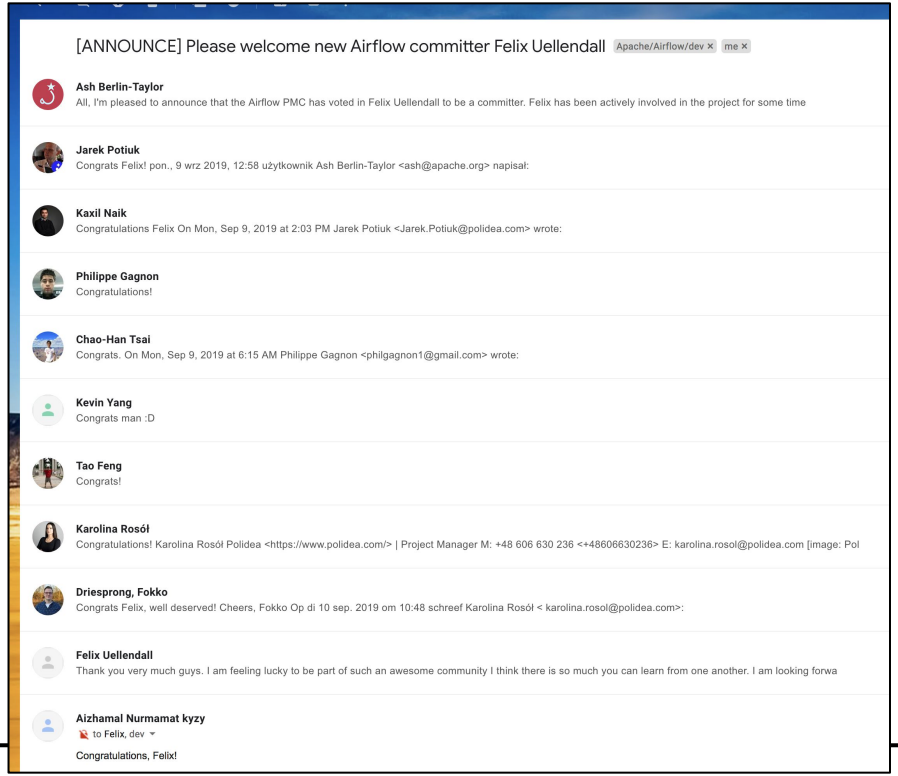


Airflow adoption



Very active project

And very welcoming too...

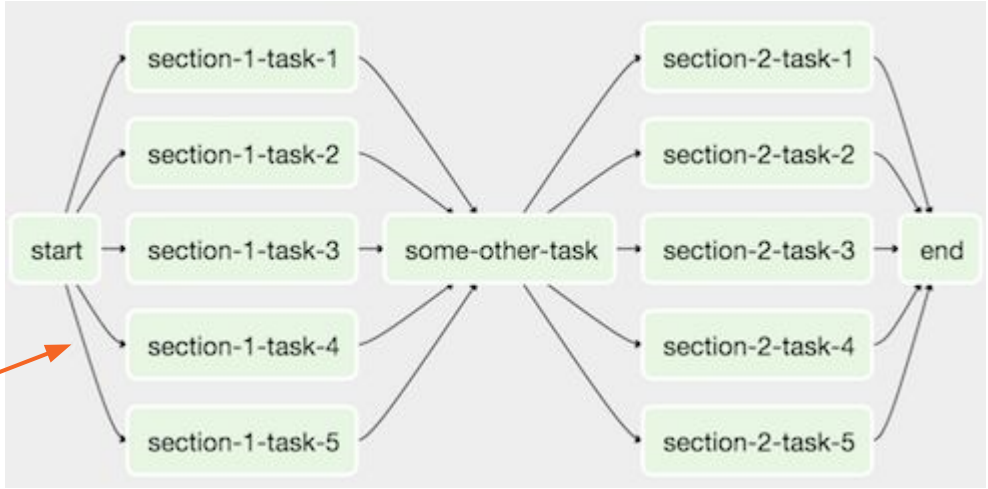


Important Airflow Concepts

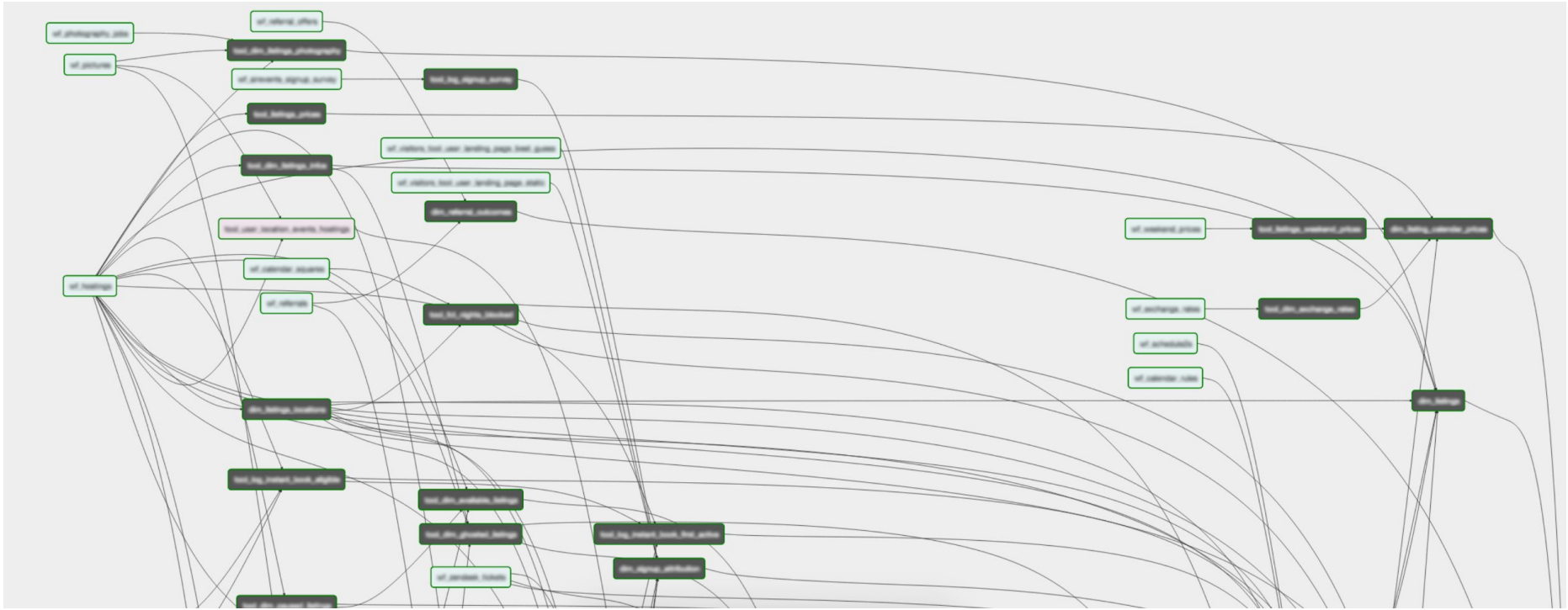
The core concept: The DAG

Dags have operators

Operators are connected



DAGs can be arbitrarily complex



https://medium.com/@hafizbadrie/airflow-when-your-dag-is-far-behind-the-schedule-ea11bf02e44c

DAGs are made of Operators

Operators do something… anything.

- BashOperator - run a bash command
- MySqlOperator - run script on MySQL
- SlackOperator - send a message to a Slack channel
- EmailOperator - send an email
- etc...

Some important operators to know

PythonOperator - execute arbitrary method

```
def do_something_really_complex (ds, **kwargs):
    print("Maybe some numpy or something..." )

run_some_arbitrary_python = PythonOperator(
    task_id='run_some_arbitrary_python' ,
    provide_context =True,
    python_callable =do_something_really_complex,
    dag=dag)
```

Some important operators to know

BranchingOperator - provide control flow

```
def only_run_on_first_of_month(self, context):
    """
    Run an extra branch on the first day of the month
    """
    if context['execution_date'].day == 1:
        return 'monthly_operator'
    else:
        return 'daily_operator'

branch_op = BranchPythonOperator(
    task_id='choose_monthly_or_daily',
    provide_context=True,
    python_callable=only_run_on_first_of_month,
    dag=dag)
```

Some important operators to know

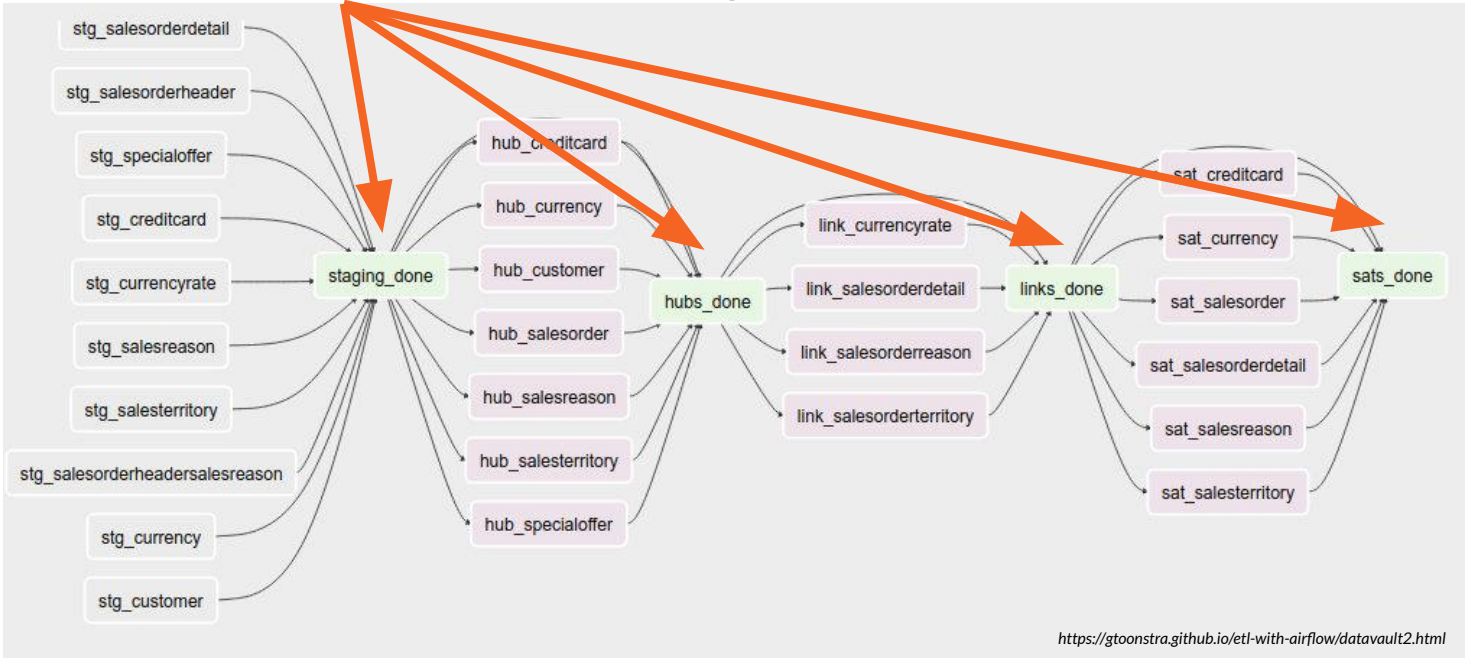
Sensors - wait for some condition

- S3KeySensor - wait for an S3 key to appear
- HttpSensor - wait for a 200 response from some endpoint
- SQLSensor - wait for a specified SQL query to return 0 rows
- etc.

```
wait_for_new_data = SqlSensor(
    task_id='wait_for_new_data',
    conn_id='redshift',
    sql='select * from database where seconds_old > 10',
    poke_interval=30,
    timeout=3600,
    dag=dag)
```

Some important operators to know

DummyOperator - group portions of a DAG



And many, many, many more

adls_list_operator.py
adls_to_gcs.py
aws_athena_operator.py
aws_sqs_publish_operator.py
awsbatch_operator.py
azure_container_instances_operator.py
azure_cosmos_operator.py
bigquery_check_operator.py
bigquery_get_data.py
bigquery_operator.py
bigquery_table_delete_operator.py
bigquery_to_bigquery.py
bigquery_to_gcs.py
bigquery_to_mysql_operator.py
cassandra_to_gcs.py
databricks_operator.py
dataflow_operator.py
dataproc_operator.py
datastore_export_operator.py
datastore_import_operator.py
dingding_operator.py
discord_webhook_operator.py
docker_swarm_operator.py
druid_operator.py
dynamodb_to_s3.py
ecs_operator.py
emr_add_steps_operator.py
emr_create_job_flow_operator.py
emr_terminate_job_flow_operator.py
file_to_gcs.py
file_to_wasb.py
gcp_bigtable_operator.py
gcp_cloud_build_operator.py
gcp_compute_operator.py
gcp_container_operator.py
gcp_dlp_operator.py
gcp_function_operator.py
gcp_natural_language_operator.py
gcp_spanner_operator.py
gcp_speech_to_text_operator.py
gcp_sql_operator.py
gcp_tasks_operator.py
gcp_text_to_speech_operator.py
gcp_transfer_operator.py
gcp_translate_operator.py
gcp_translate_speech_operator.py
gcp_video_intelligence_operator.py
gcp_vision_operator.py
gcs_acl_operator.py
gcs_delete_operator.py
gcs_download_operator.py
gcs_list_operator.py
gcs_operator.py
gcs_to_bq.py
gcs_to_gcs.py
gcs_to_gcs_transfer_operator.py
gcs_to_gdrive_operator.py
gcs_to_s3.py
grpc_operator.py
hipchat_operator.py
hive_to_dynamodb.py
imap_attachment_to_s3_operator.py
jenkins_job_trigger_operator.py
jira_operator.py
kubernetes_pod_operator.py
mlengine_operator.py
mongo_to_s3.py
mssql_to_gcs.py
mysql_to_gcs.py
opsgenie_alert_operator.py
oracle_to_azure_data_lake_transfer.py
oracle_to_oracle_transfer.py
postgres_to_gcs_operator.py
pubsub_operator.py
qubole_check_operator.py
qubole_operator.py
redis_publish_operator.py
s3_copy_object_operator.py
s3_delete_objects_operator.py
s3_list_operator.py
s3_to_gcs_operator.py
s3_to_gcs_transfer_operator.py
s3_to_sftp_operator.py
sagemaker_base_operator.py
sagemaker_endpoint_config_operator.py
sagemaker_endpoint_operator.py
sagemaker_model_operator.py
sagemaker_training_operator.py
sagemaker_transform_operator.py
sagemaker_tuning_operator.py
segment_track_event_operator.py
sftp_operator.py
sftp_to_s3_operator.py
slack_webhook_operator.py
snowflake_operator.py
sns_publish_operator.py
spark_jdbc_operator.py
spark_sql_operator.py
spark_submit_operator.py
sql_to_gcs.py
sqoop_operator.py
ssh_operator.py
vertica_operator.py
vertica_to_hive.py
vertica_to_mysql.py
wasb_delete_blob_operator.py
winrm_operator.py

Let’s talk use cases

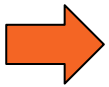
Typical first Airflow use case

users_funnel.sql
ab_test_results.sql
users_by_region.sql

cron:
0 2 * * * /etl/run_sql.sh users_funnel.sh
0 2 * * * /etl/run_sql.sh ab_test_resuts.sql
0 2 * * * /etl/run_sql.sh users_by_region.sql

But then comes more SQL

users_funnel.sql
ab_test_results.sql
users_by_region.sql



users_funnel.sql, ***must run after*** normalize_users.sql
ab_test_results.sql
users_by_region.sql ***must run*** before normalize_users.sql
normalize_users.sql
aggregrate_items_by_category.sql
calculate_risk_factors ***must run after*** normalize_users.sql, incident_reports.sql
user_actions_by_version.sql
some_more_silly_business_stuff.sql
even_more_counting_of_things.sql

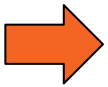
Enter the custom etl framework...

cron:
0 2 * * * /etl/run_sql.sh users_funnel.sh
0 2 * * * /etl/run_sql.sh ab_test_resuts.sql
0 2 * * * /etl/run_sql.sh users_by_region.sql

cron:
0 2 * * * /etl/etl_runner.***py***

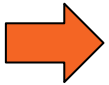
Airflow the rescue

Full set of reliability features



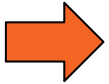
```
args = { 'owner': 'DataEng', 'start_date': datetime(2019, 05, 03),
         'email_on_failure': de@mycompany.com, 'retries: 3', 'depends_on_past': True,
         'queue': 'business_etl', 'sla': datetime.timedelta(3 hours),
         'execution_timeout': datetime.timedelta(1 hour), }
```

Create the DAG

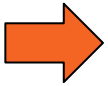


```
dag = DAG(dag_id='daily_business_etl', default_args=args, schedule_interval='0 2 * * *',)

# Read in all the sql files we want to run
sqls_to_run = ['users_funnel', 'ab_test_resuts', ...]

# And populate them into a map
sql_ops = dict()
for sql in sqls_to_run:
   sql_operator[sql] = SQLOperator(read_sql_file(sql), conn_id='redshift')
```

Build an Operator for each SQL

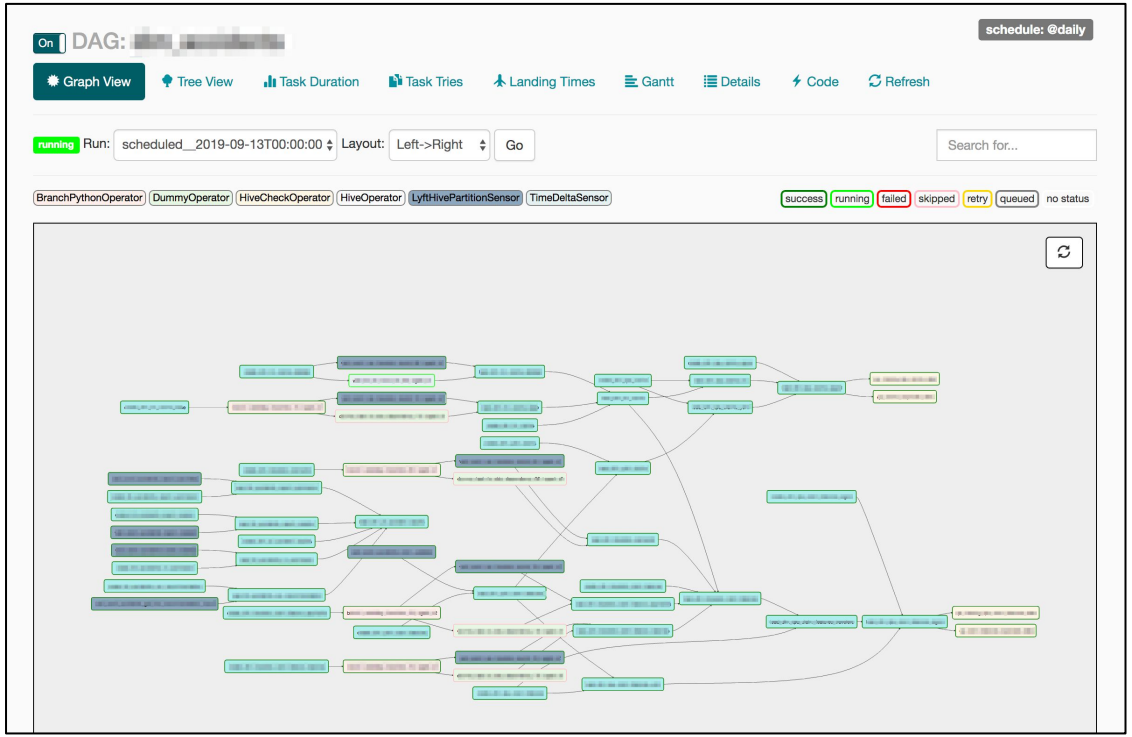


Wire up dependencies



```
# Wire up dependencies
sql_ops['user_funnel'].set_upstream(sql_ops['normalize_users'])
sql_ops['normalize_users'].set_upstream(sql_ops['users_by_region'])
sql_ops['calcuate_risk_factors'].set_upstream(sql_ops['normalize_users'])
sql_ops['calcuate_risk_factors'].set_upstream(sql_ops['incident_reports'])
```

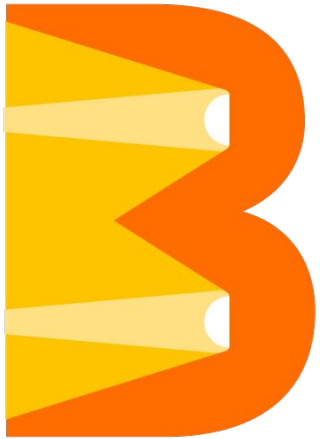
For the win



ETLs like this should be built into Airflow

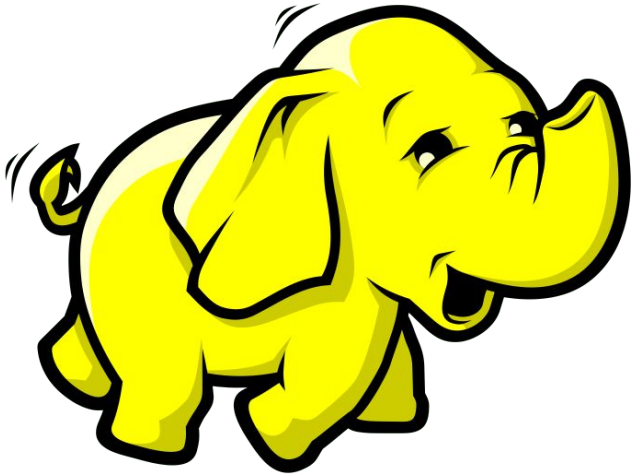
What Airflow is not, part 1

Airflow is not a stream processor, like Apache Flink or Beam

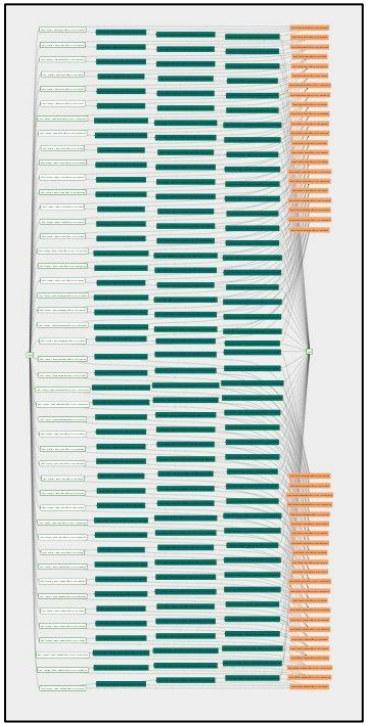


What Airflow is not, part 2

Airflow isn't a Map-Reduce engine, like Hadoop or Hive

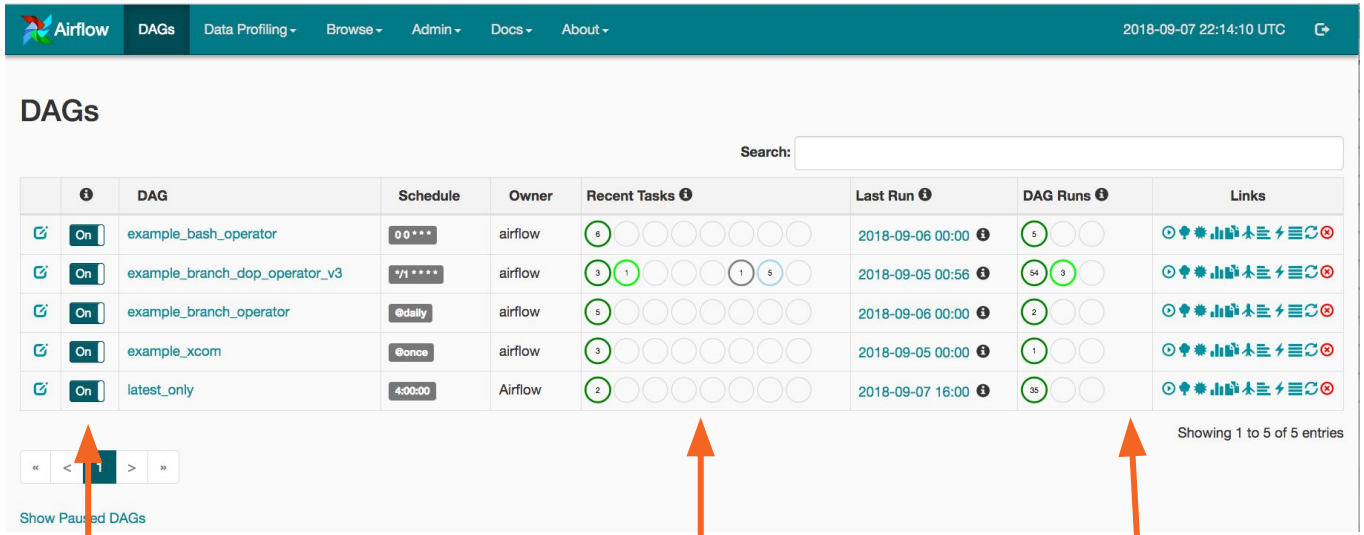


A case study in what Airflow isn't



The web interface

DAGs page

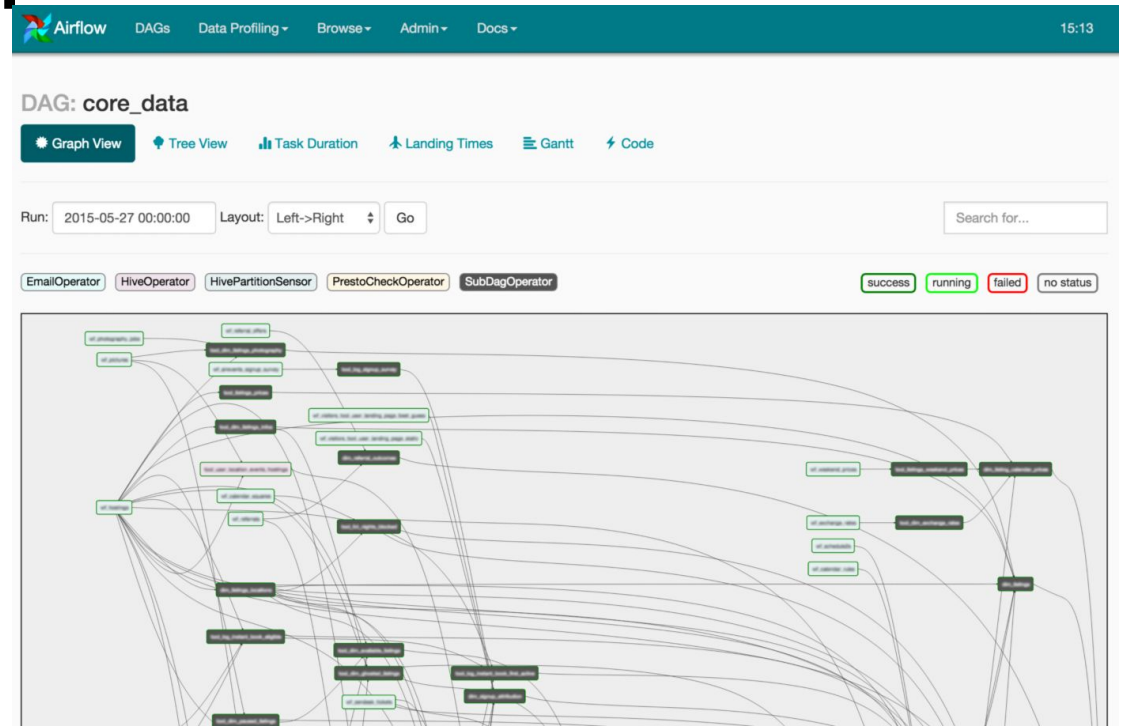


Is the DAG enabled?

How many operators recently succeeded, failed, are retrying, etc.

How many runs recently succeeded, failed, etc.

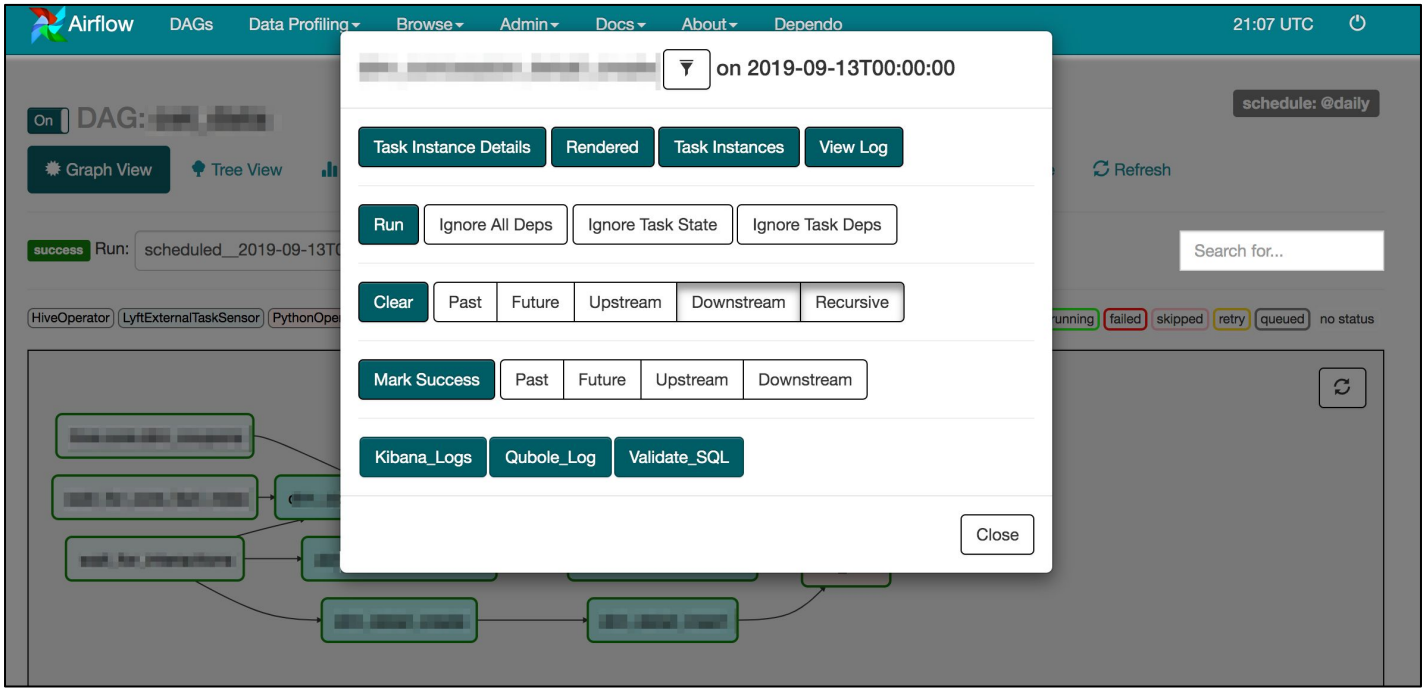
Graph view



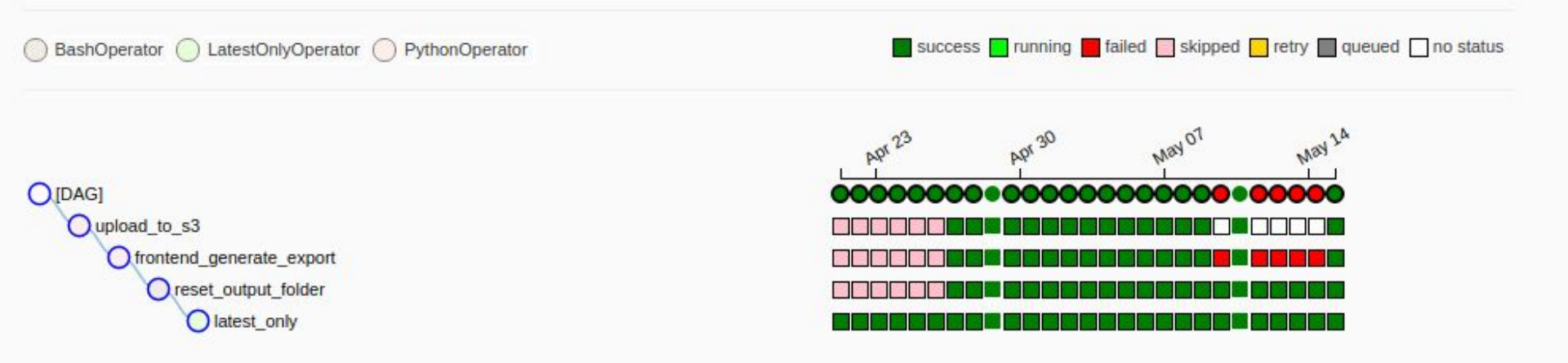
Gets super slow on big DAGs

https://medium.com/a-r-g-o/installing-apache-airflow-on-ubuntu-aws-6ebac15db211

Task Instance details



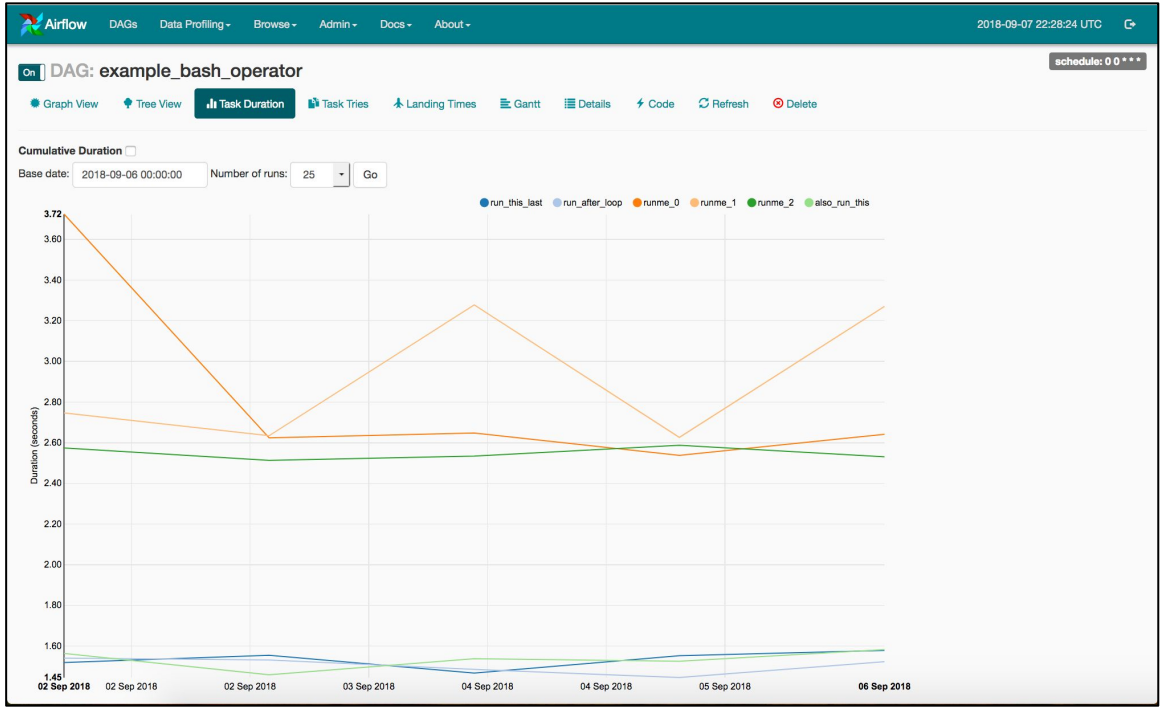
Tree view page (super useful)



http://flagzeta.org/blog/intro-to-apache-airflow/

Needs to expand, collapse, filter

Duration view

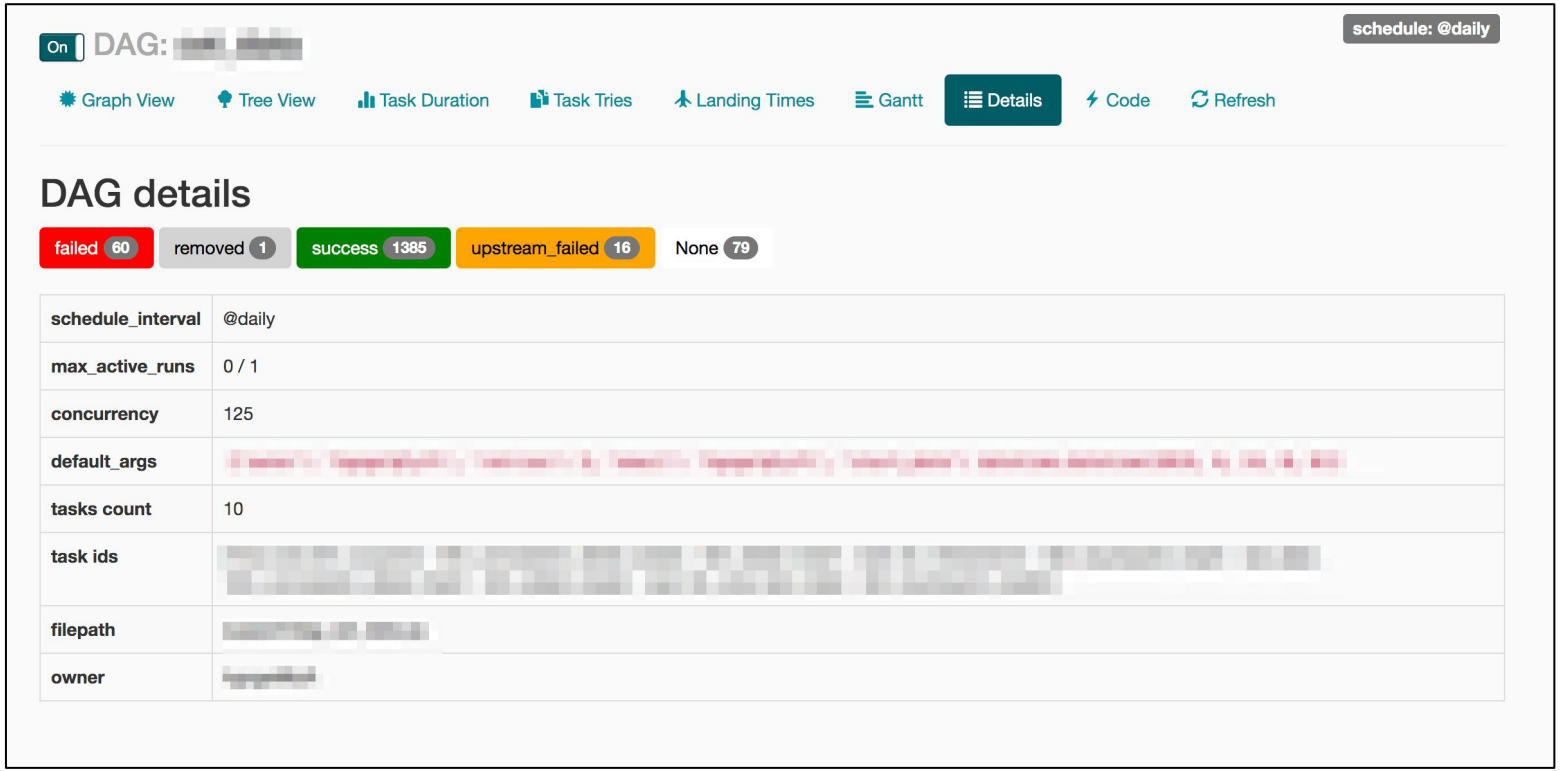


Totally broken for Really big DAGs

Gantt chart

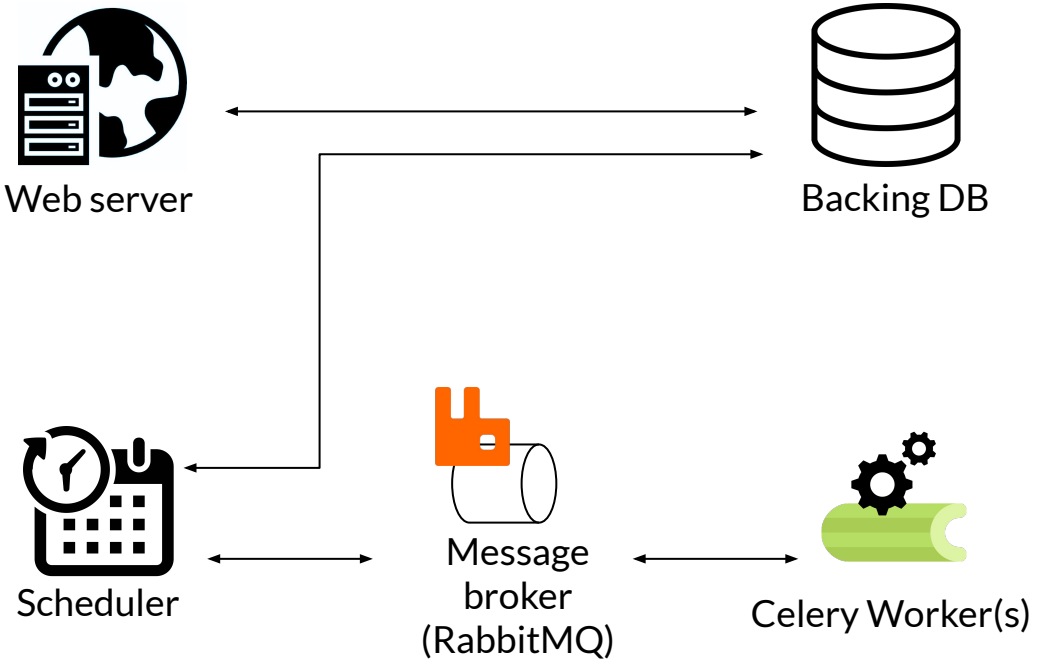

On
DAG:
schedule: @daily
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Run:
2019-09-13 00:00:00
Go
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00

Details view



Deployment

Anatomy of an Airflow cluster



Anatomy of an Airflow cluster

Web server

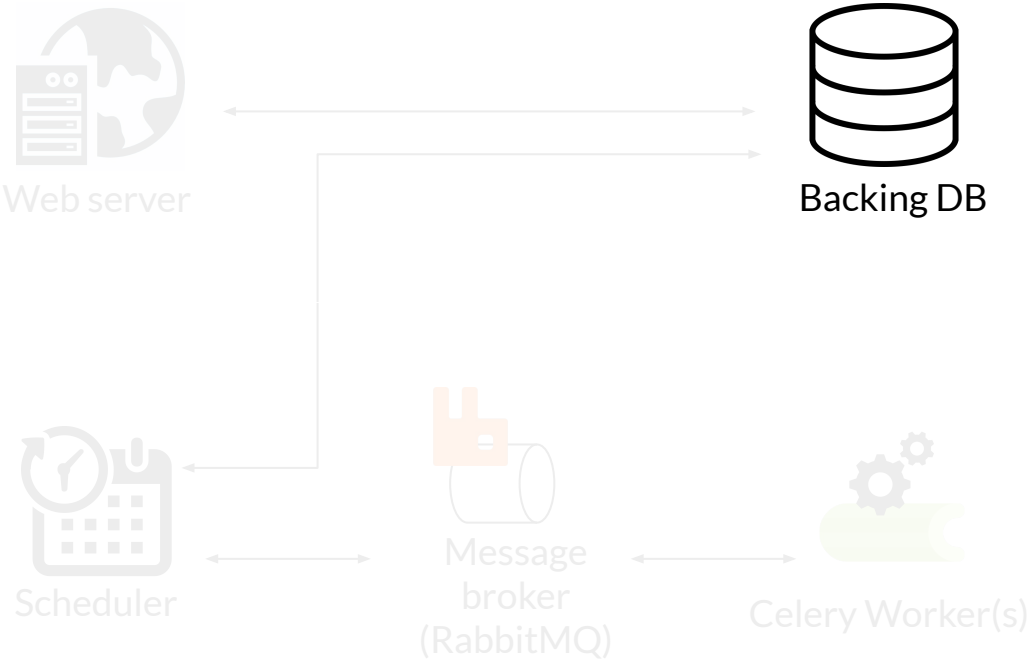
- Stateless Flask application that talks to Backing DB to get/set task and DAG status.

Scaling issues, UX improvements

Anatomy of an Airflow cluster

Backing DB

- Stores all state for DAGs, tasks, XComs, etc.
- Managed solutions (RDS) work well here.
- Can be perf bottleneck

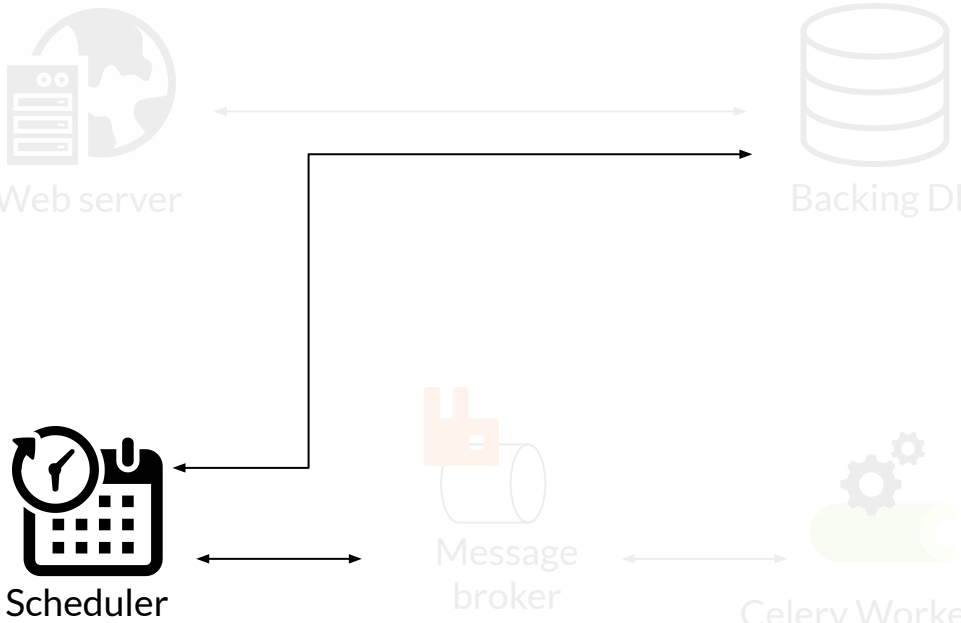


Anatomy of an Airflow cluster

Scheduler

- Kicks off jobs as they're read to be run
- Reasonably buggy
- Written in the wrong language

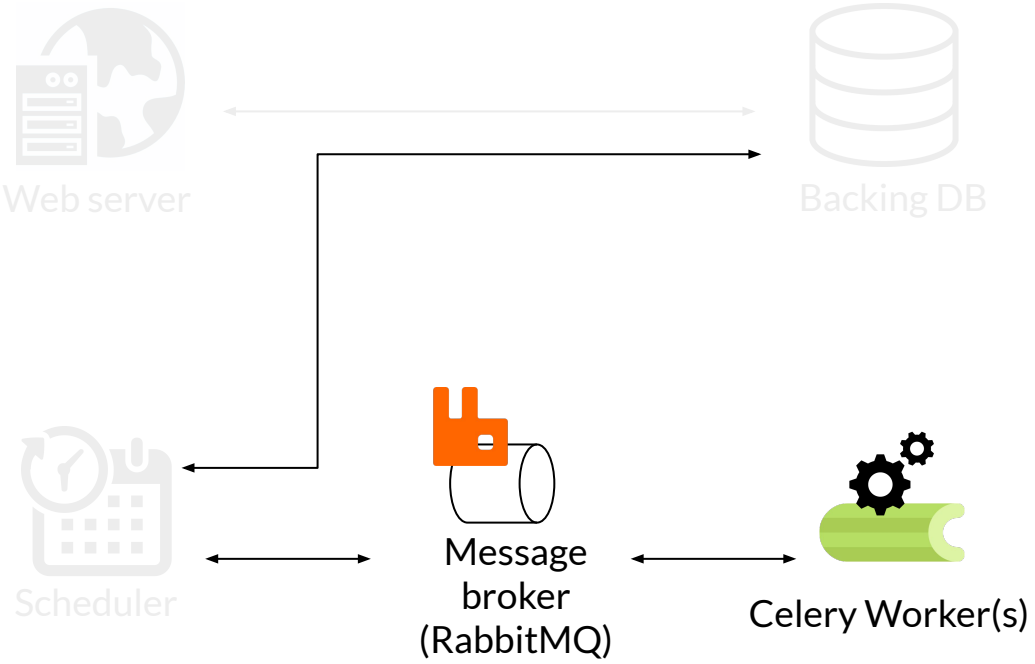
Tests, correctness, features, etc.



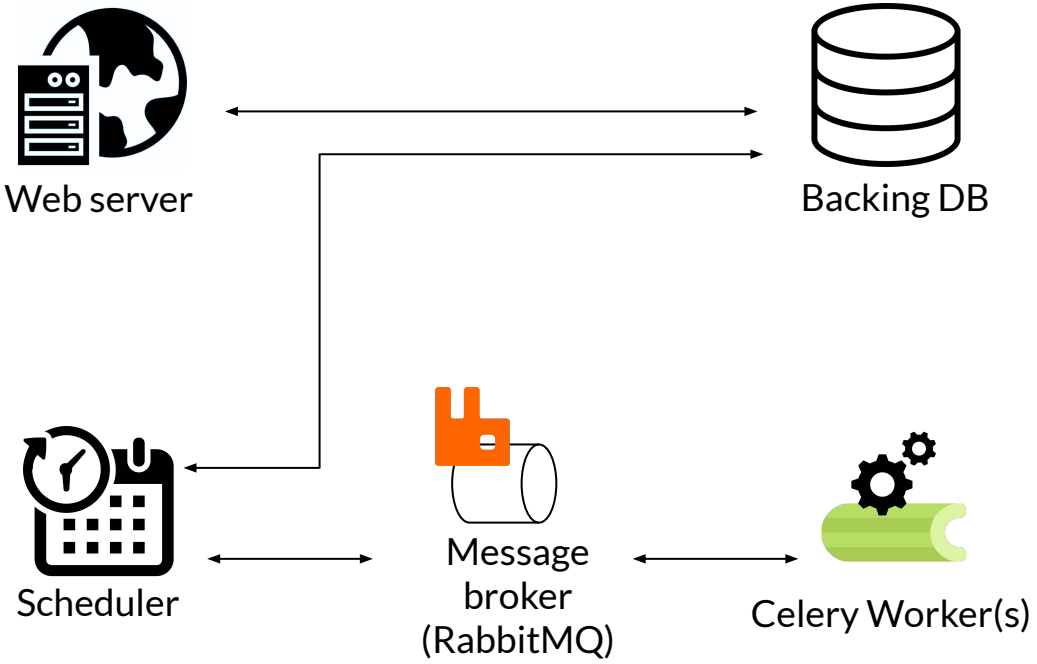
Anatomy of an Airflow cluster

Message Queue and Workers

- Takes requests from scheduler to execute Task Instances
- Scalable, but better support needed



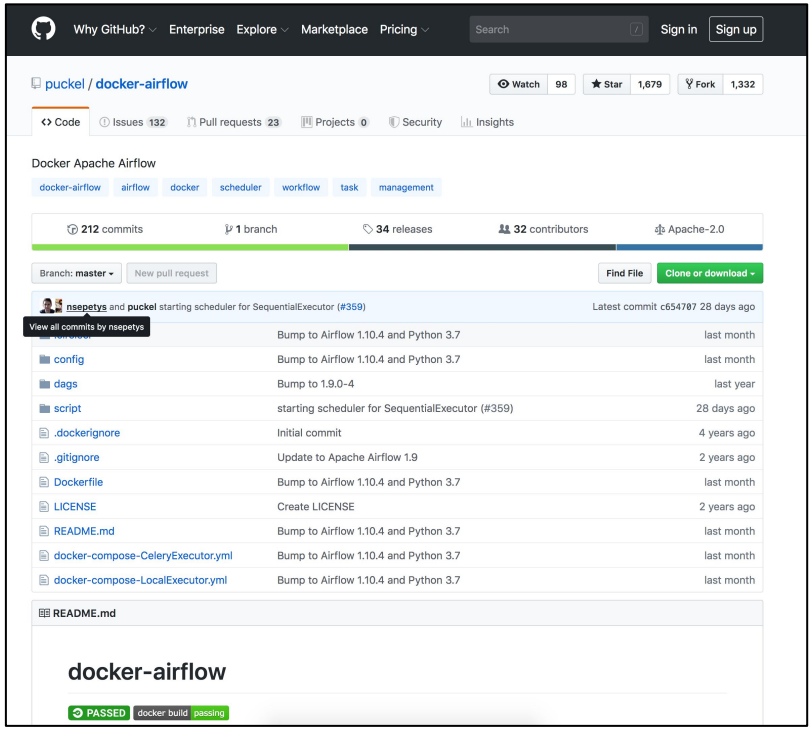
Anatomy of an Airflow cluster



Types of executors

- **CeleryExecutor**
  - Reliable, scalable, hugely popular
- **LocalExecutor**
  - Only for hello world and maybe testing
  - Yeah, don't use it. It's a pain.
- **KubernetesExecutor**
  - Added in 1.10
  - Lots of momentum from Google, others

Docker-based solutions



Gracias!

https://www.linkedin.com/in/jghoman - https://twitter.com/BlueBoxTraveler

First patch to Airflow

- Trivial change in code base to exercise the mechanics of contributing
  - Checking out code
  - Making a change
  - Testing change
  - Uploading change
  - Getting feedback from the community
- Visit this spreadsheet to grab a ticket: http://bit.ly/CCOSS-Airflow-tickets