ARTHUR WIEDMER / FEBRUARY 2017

Apache Airflow
@
Airbnb

AND BEYOND

(IF THERE IS TIME)

About Me

- Data Engineer on the Data Platform Team at Airbnb.
- Working on Airflow since 2014, Apache Airflow committer
- I work on both Airflow and building internal frameworks on top of it.
- Most of my free time is spent with my wife and our 1 year-old son :)

What is Airflow?

What can Airflow do for you?

What should you know before you start?


Airflow?

Why does Airflow exist?

- Companies grow to have a **complex network of processes that have intricate dependencies**.
- Analytics & batch processing are **mission critical**. They serve decision makers and power machine learning models that can feed into production.
- There is a lot of **time invested in writing and monitoring jobs and troubleshooting issues**.

What is Airflow?

An open source platform to author, orchestrate and monitor batch processes

- It's the **glue** that binds your data ecosystem together
- It **orchestrates** tasks in a complex networks of job dependencies
- It's **Python** all the way down
- It's **popular** and has a thriving open source community
- It's **expressive** and **dynamic**, workflows are defined in code

Concepts

- **Workflows** are called **DAGs** for Directed Acyclic Graph.


On
DAG: airflow_maintenance
schedule: 1 day, 0:00:00
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
failed
Run:
scheduled__2017-02-15T00:00:00
Layout:
Left->Right
Go
Search for...
GenericTransfer
MySqlOperator
MySqlToHiveTransfer
TimeSensor
success
running
failed
skipped
retry
queued
no status
export_dag_table
wait_for_end_of_day
export_task_instance_table
export_db_exports_sla_summary
wait_for_9am
export_log_table
export_core_data_sla_summary
sla_table_landing
export_hive_stats
export_core_sessions_sla_summary
sla_metadata_update
sla_metadata
m2m
sla_metadata_update_metastore

```
dag = DAG(
    'tutorial',
    default_args=default_args,
    description='A simple tutorial DAG',
    schedule_interval=timedelta(days=1))
```

Concepts

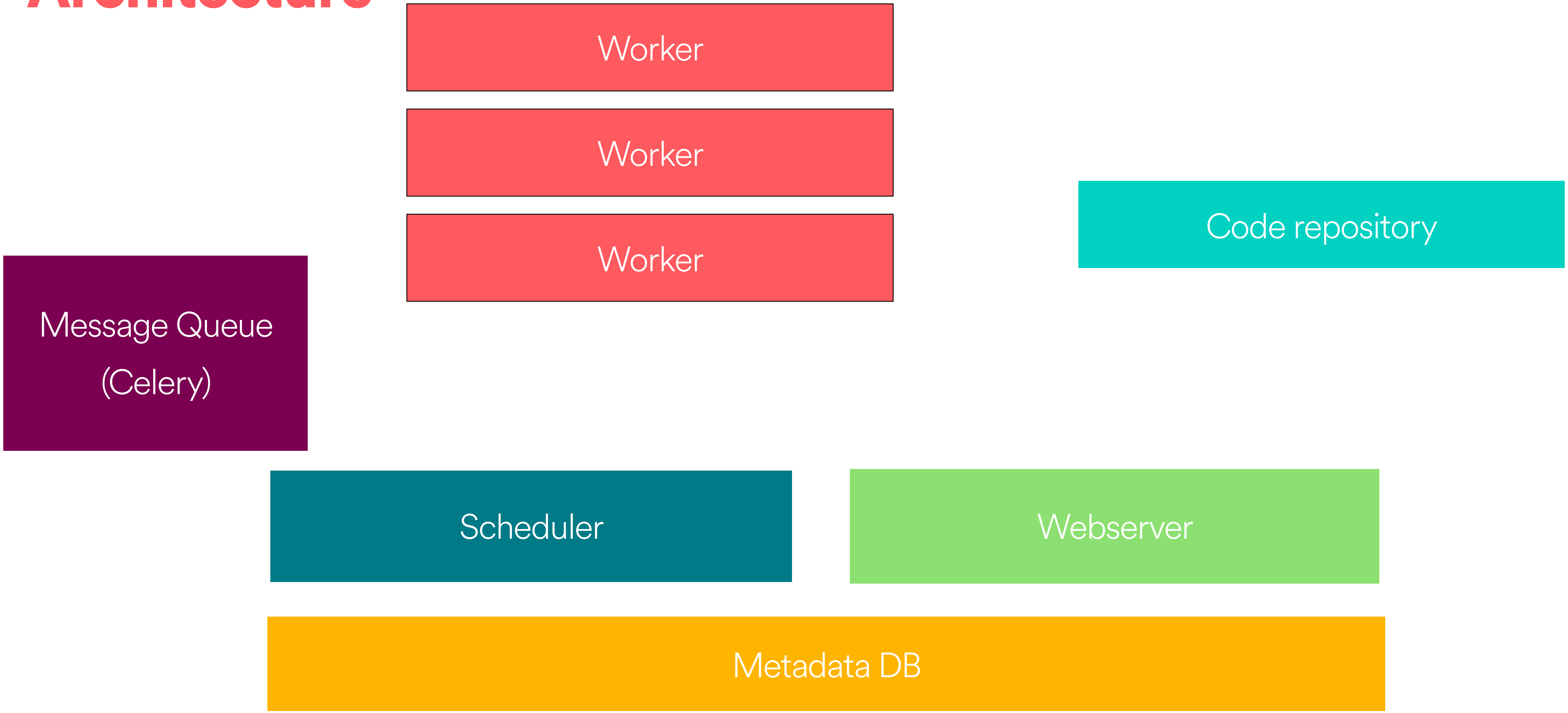
- **Tasks:** Workflows are composed of tasks called **Operators.**
- Operators can do pretty much anything that can be run on the Airflow machine.
- We tend to classify operators in 3 categories : **Sensors, Operators, Transfers.**

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

Setting dependencies

```
t2.set_upstream(t1)
```

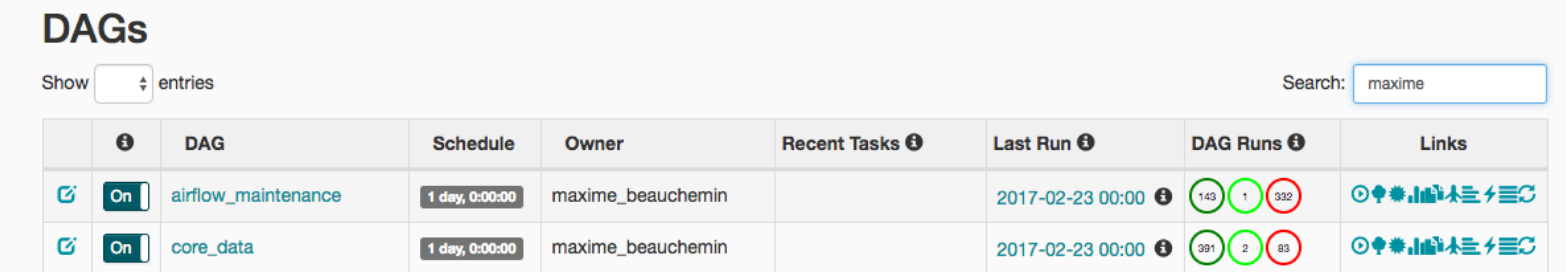
Architecture



What can Airflow do for you?

Monitoring

Monitoring DAG Status



DAGs

Show [ ] entries

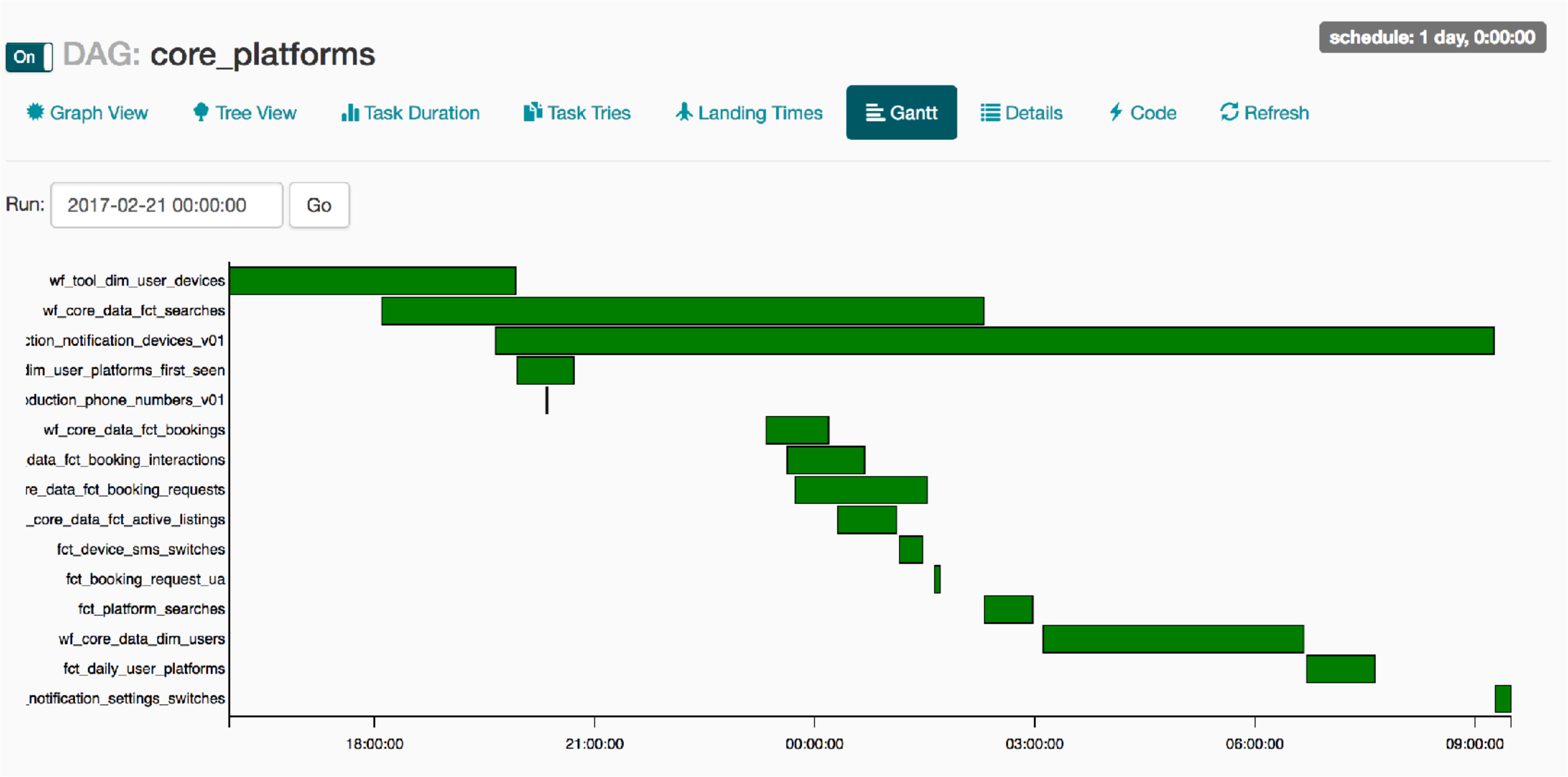
Search: maxime

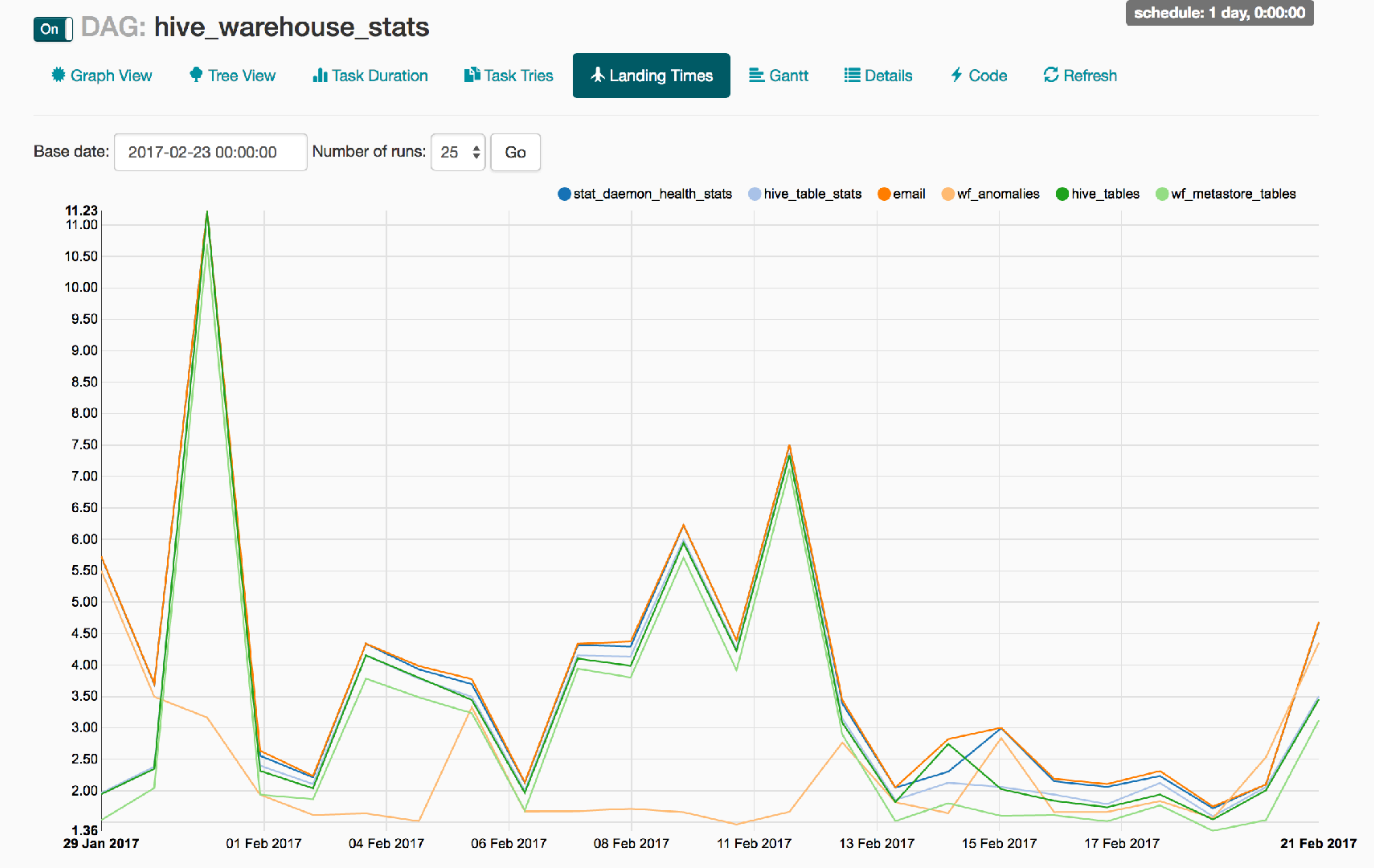
		DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
	On	airflow_maintenance	1 day, 0:00:00	maxime_beauchemin		2017-02-23 00:00	143 1 332	
	On	core_data	1 day, 0:00:00	maxime_beauchemin		2017-02-23 00:00	391 2 83	

Last Run	DAG Runs
2017-02-23 00:00	143 1 332
2017-02-23 00:00	391 2 83

Monitoring DAG Status

Monitoring Gantt Chart style





On
DAG: hive_warehouse_stats
schedule: 1 day, 0:00:00
Graph View
Tree View
Task Duration
Task Tries
Landing Times
Gantt
Details
Code
Refresh
Base date: 2017-02-23 00:00:00
Number of runs: 25
Go
stat_daemon_health_stats
hive_table_stats
email
wf_anomalies
hive_tables
wf_metastore_tables
11.23
11.00
10.50
10.00
9.50
9.00
8.50
8.00
7.50
7.00
6.50
6.00
5.50
5.00
4.50
4.00
3.50
3.00
2.50
2.00
1.36
29 Jan 2017
01 Feb 2017
04 Feb 2017
06 Feb 2017
08 Feb 2017
11 Feb 2017
13 Feb 2017
15 Feb 2017
17 Feb 2017
21 Feb 2017

Scale

Airflow @ Airbnb : scale

- We currently run 800+ DAGs and about ~**80k tasks** as day.
- We have DAGs running at **daily, hourly and 10 minute** granularities. We also have ad hoc DAGs.
- About **100 people @ Airbnb have authored or contributed to a DAG** directly and 500 have contributed or modified a configuration to one of our frameworks.
- We use the Celery executor with Redis as a backend.

Flexibility

Airflow @ Airbnb

Experimentation

Growth Analytics

Search Ranking

Operational Work

Data Warehousing

Engagement Analytics

Anomaly Detection

Infrastructure Monitoring

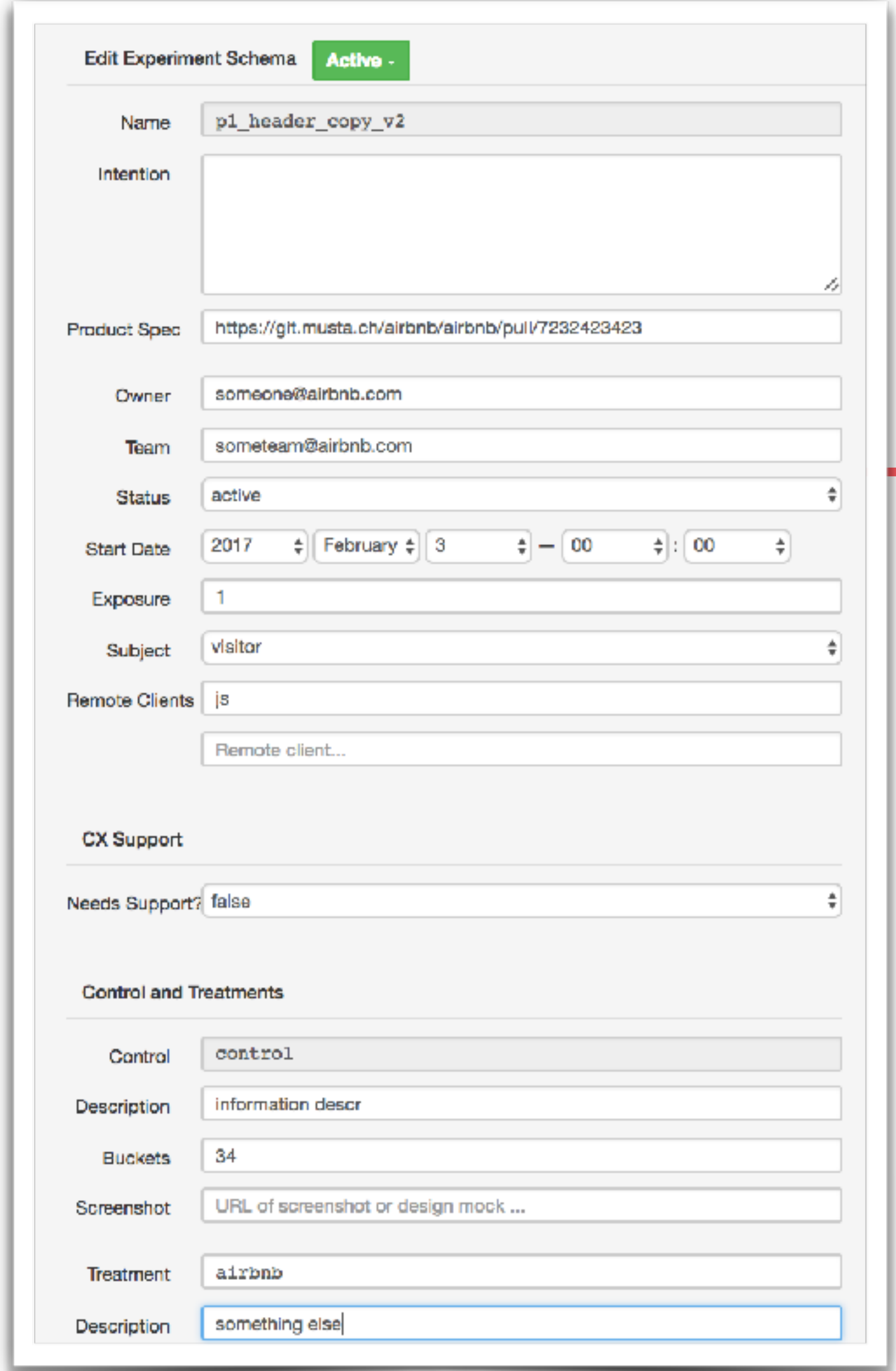
Data Exports
from/to production

Sessionization

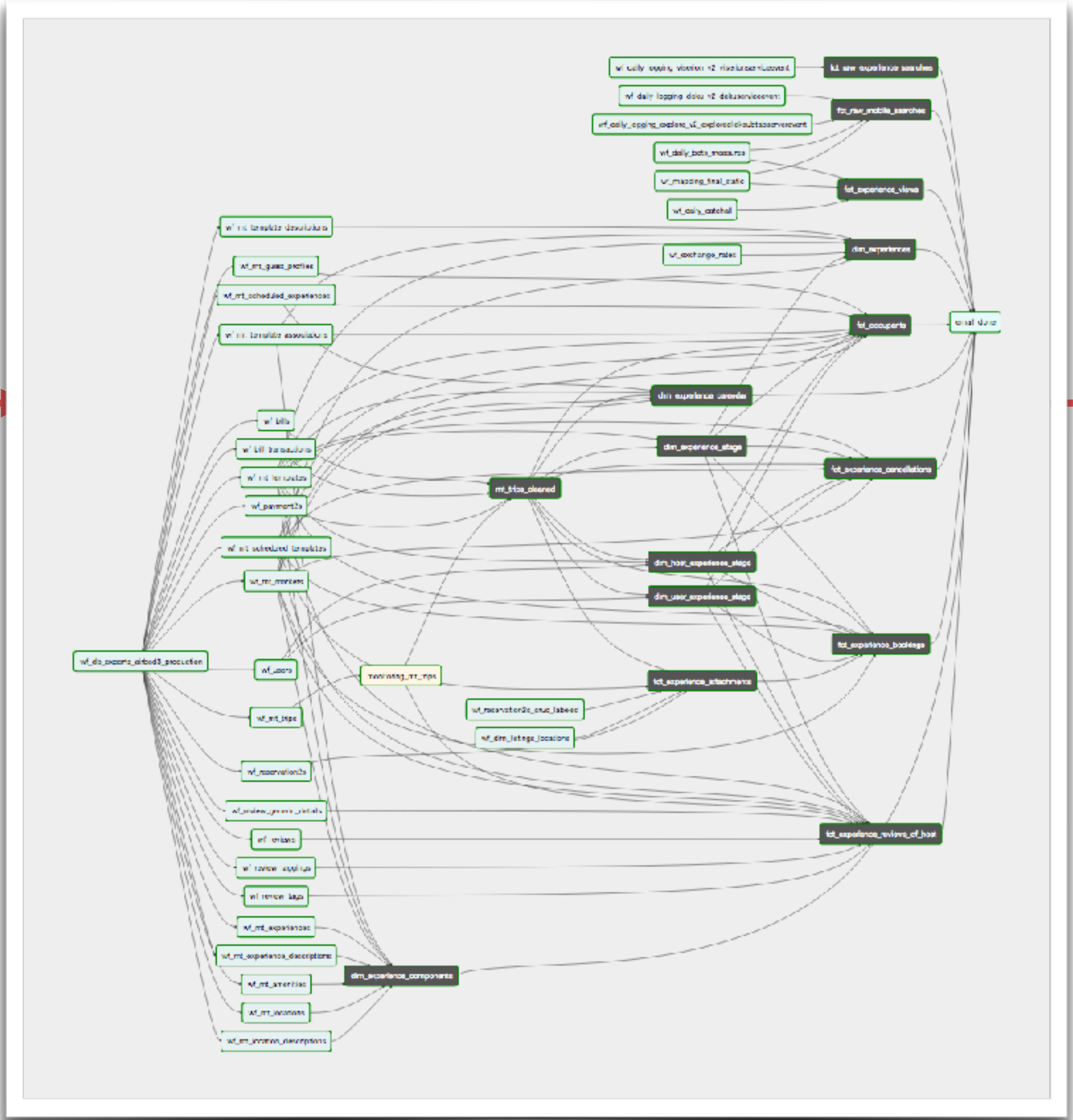
Email Targeting

Common Pattern

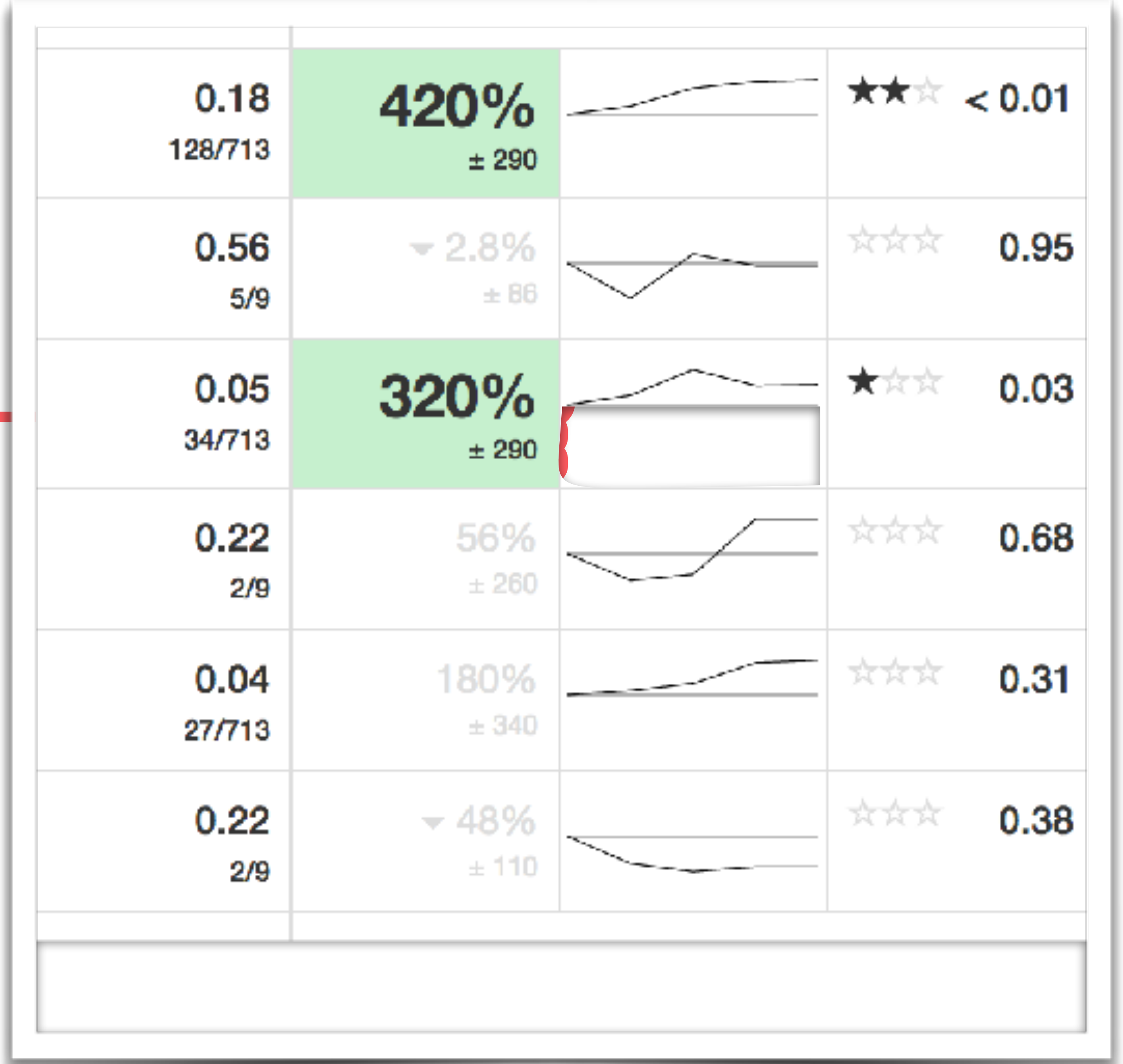
Input



Data Processing



Output



CumSum

Efficient cumulative metrics computation

- Live to date metrics per subject (user, listings, advertiser, ...) are a common pattern
- Computing the SUM since beginning of time is inefficient, it's preferable to add up today's metrics to yesterday's total

```
SELECT userid, COUNT(*) as ltd_bookings
FROM fct_bookings
WHERE ds <= '{{ ds }}'
GROUP BY userid
```

```
metric: bookings
owner: data-engineering@airbnb.com
subject: user
sql: |
    SELECT userid, COUNT(1) as metric
    FROM fct_bookings
    WHERE ds = '{{ ds }}'
    GROUP BY userid
dependencies:
    core_data.fct_bookings:
        partition: ds
```

```
SELECT userid, SUM(ltd_bookings)
FROM (
    -- Yesterday's total from summary table
    SELECT userid, ltd_bookings
    FROM cumsum
    WHERE ds = '{{ yesterday_ds }}'
    UNION ALL
    --- Today's transaction to add to existing total
    SELECT userid, 1 AS ltd_bookings
    FROM fct_bookings
    WHERE ds = '{{ ds }}'
) as subqry
GROUP BY userid
```

CumSum

Efficient cumulative metrics computation

- Live to date metrics per subject (user, listings, advertiser, ...) are a common pattern
- Computing the SUM since beginning of time is inefficient, it's preferable to add up today's metrics to yesterday's total

Outputs

- An efficient pipeline
- Easy / efficient backfilling capabilities
- A centralized table, partitioned by metric and date, documented by code
- Allows for efficient time range deltas by scanning 2 partitions

```
SELECT use
FROM fct_b
WHERE ds <
GROUP BY u
```

```
metric: bo
owner: dat
subject: us
sql: |
    SELECT
    FROM f
    WHERE ds = '{{ ds }}'
    GROUP BY userid
dependencies:
    core_data.fct_bookings:
        partition: ds
```

```
table

existing total
```

What should you know to get started?

Monitoring and Alerting

- Enable the email feature and EmailOperator/SlackOperator for monitoring.
- Ease of monitoring will help you keep track of your jobs as their number grows.
- Checkout the SLA feature to know when your jobs are not completing on time.
- The scheduler is still the weakest link as it is a single point of failure. Enabling service monitoring with runit, monit can be useful if you need to guarantee uptime.

Metadata Database

- As the number of jobs you run on Airflow increases, so does the load on the Airflow database.
- SQLite is used for tutorials but cannot handle concurrent connections. We highly recommend switching to MySQL/MariaDB or Postgres.
- Some people have tried other databases, but we cannot currently test against them, so it might break in the future.

Best Practices about DAG building

- Try to make you tasks idempotent. Airflow will then be able to handle retrying for you in case of failure.

- You can setup pools for resource management.

- SubDAGs are still not completely without issues.

Configuration as Code!

As an alternative to static YAML, JSON or worse: drag and drop tools

- Code is more expressive, powerful & compact
- Reusable components (functions, classes, object factories) come naturally in code
- An API has a clear specification with defaults, input validation and useful methods
- Nothing gets lost into translation: Python is the language of Airflow.
- The API can be derived/extended as part of the workflow code. Build your own Operators, Hooks etc...
- In its minimal form, it's as simple as static configuration

The future of Airflow

Quick history of Airflow @ Airbnb

- Back in 2014 we were using **Chronos** a Framework for long running jobs on top of **Mesos**.
- Defining data dependencies was near impossible. Debugging why data was not landing on time was really difficult.
- Max Beauchemin joined Airbnb and was interested in open sourcing an entirely rewritten version of Data Swarm, the job authoring platform at Facebook.
- Introduced **Jan 2015** for our main warehouse pipeline.
- Open sourced in **early 2015**, donated to the **Apache Foundation** for Incubation in **march 2016**.

Apache Airflow

- The community is **currently working on version 1.8.0**. To be released soon.
- The **focus** has been on **stability** and **monitoring/ troubleshooting** enhancements.
- We hope to graduate to Top Level Project this year.
- We are looking for contributors. Check out the project and come hack with us.

Resources

Airflow Resources

- The Airflow community is active on **Gitter** at https://gitter.im/apache/incubator-airflow and has a lot of user to user help.

- If you have more advanced questions, the dev mailing list at http://mail-archives.apache.org/mod_mbox/incubator-airflow-dev/ has the core developers on it.

- The documentation is available at **https://airflow.incubator.apache.org/**

- The project also has a wiki : https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow+Home

Airflow Talks

- The Bay Area Airflow meet up :

https://www.meetup.com/Bay-Area-Apache-Airflow-Incubating-Meetup/

- Matt Davis at PyBay 2016:

https://speakerdeck.com/pybay2016/matt-davis-a-practical-introduction-to-airflow

- Laura Lorenz at PyData DC 2016 How I learned to time travel, or, data pipelining and scheduling with Airflow :

https://www.youtube.com/watch?v=60FUHEkcPyY


Questions?

airbnb