Reasoning Reliability in Wrike's Data Pipeline

Wrike - A Collaborative Work Management Platform

Founded in 2006

10 Offices Globally

20,000+ Customers Globally

1000+ Employees

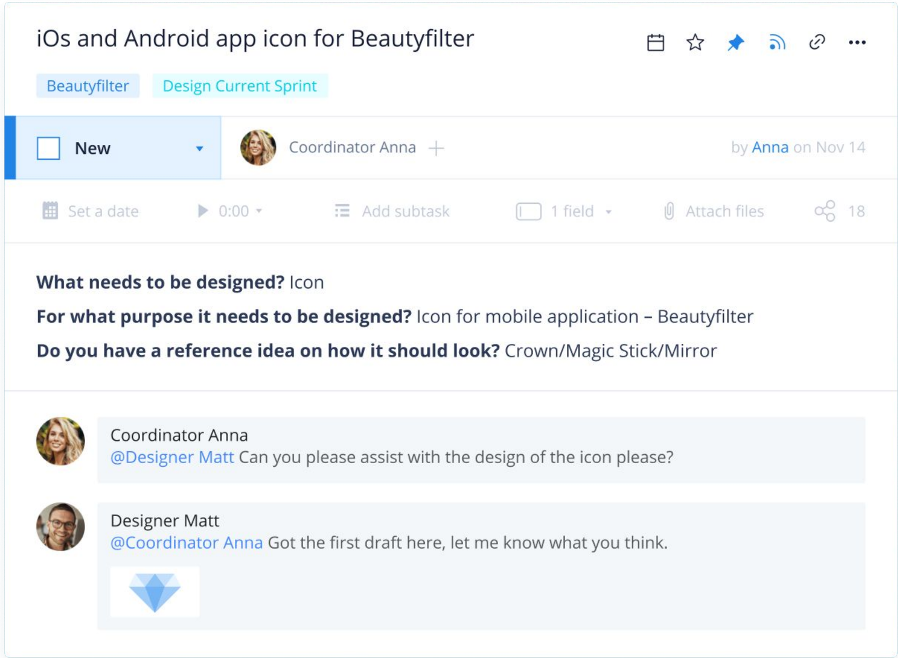
5 years in the Fast 500

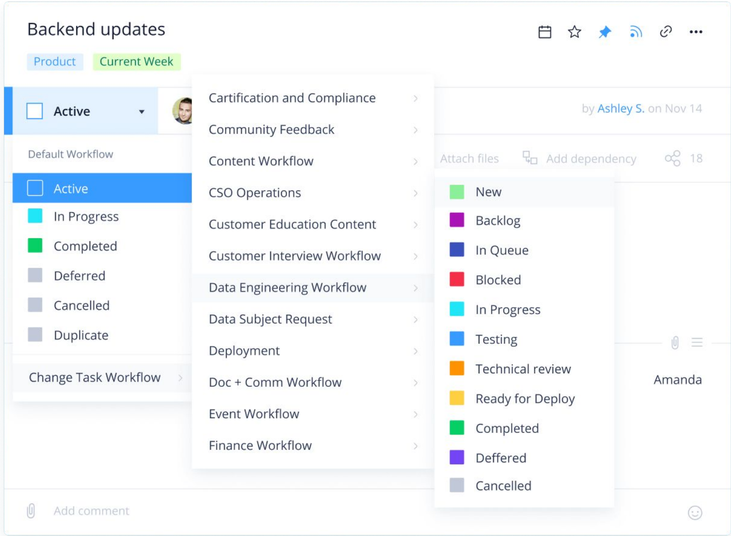
20,000+

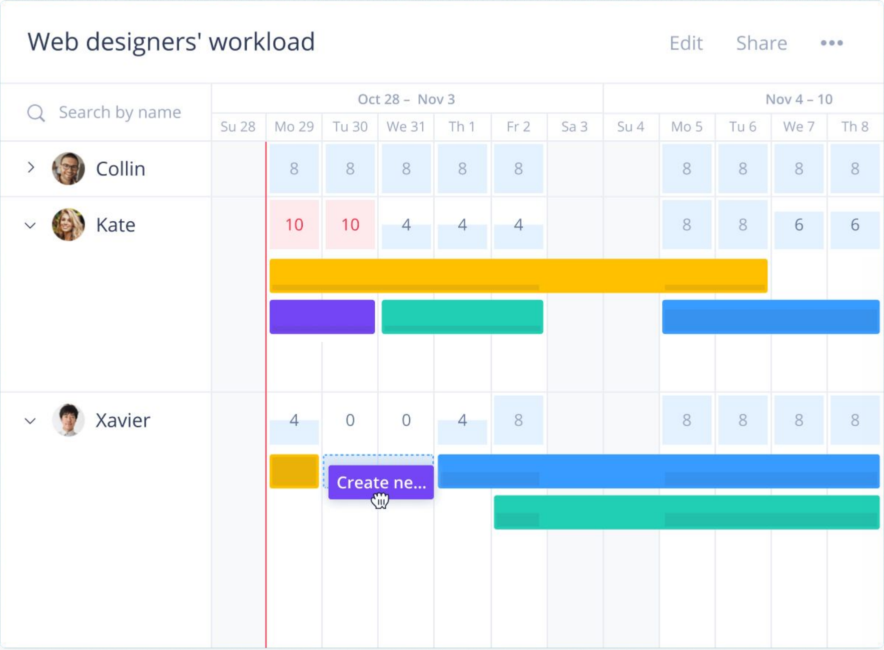
Organizations choose Wrike to orchestrate their digital work

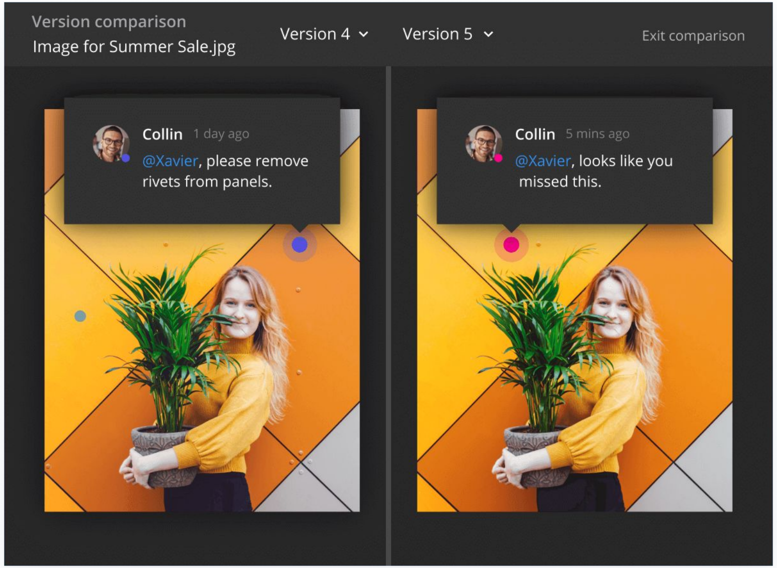
With an additional 35,000 starting trials each month

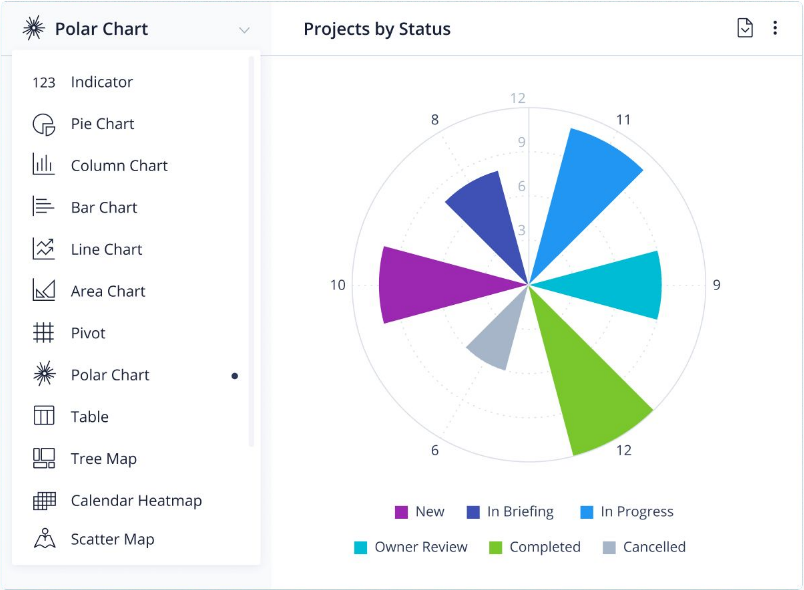
- 2M users
- 130+ countries
- 10 languages
- 100M+ completed tasks

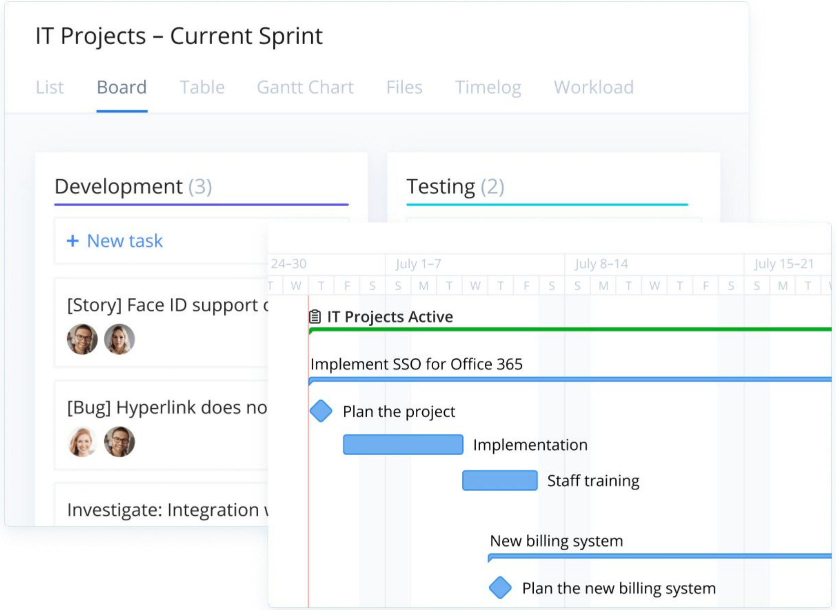

iOs and Android app icon for Beautyfilter
Beautyfilter
Design Current Sprint
New
Coordinator Anna
by Anna on Nov 14
Set a date
0:00
Add subtask
1 field
Attach files
18
What needs to be designed? Icon
For what purpose it needs to be designed? Icon for mobile application – Beautyfilter
Do you have a reference idea on how it should look? Crown/Magic Stick/Mirror
Coordinator Anna
@Designer Matt Can you please assist with the design of the icon please?
Designer Matt
@Coordinator Anna Got the first draft here, let me know what you think.

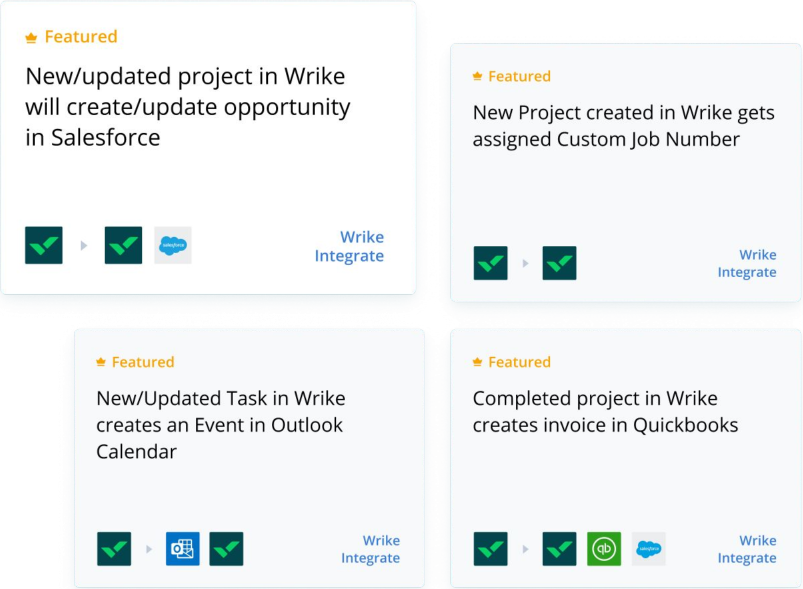

Backend updates
Product
Current Week
Active
by Ashley S. on Nov 14
Default Workflow
Active
In Progress
Completed
Deferred
Cancelled
Duplicate
Change Task Workflow
Cartification and Compliance
Community Feedback
Content Workflow
CSO Operations
Customer Education Content
Customer Interview Workflow
Data Engineering Workflow
Data Subject Request
Deployment
Doc + Comm Workflow
Event Workflow
Finance Workflow
Attach files
Add dependency
18
New
Backlog
In Queue
Blocked
In Progress
Testing
Technical review
Ready for Deploy
Completed
Deffered
Cancelled
Amanda
Add comment

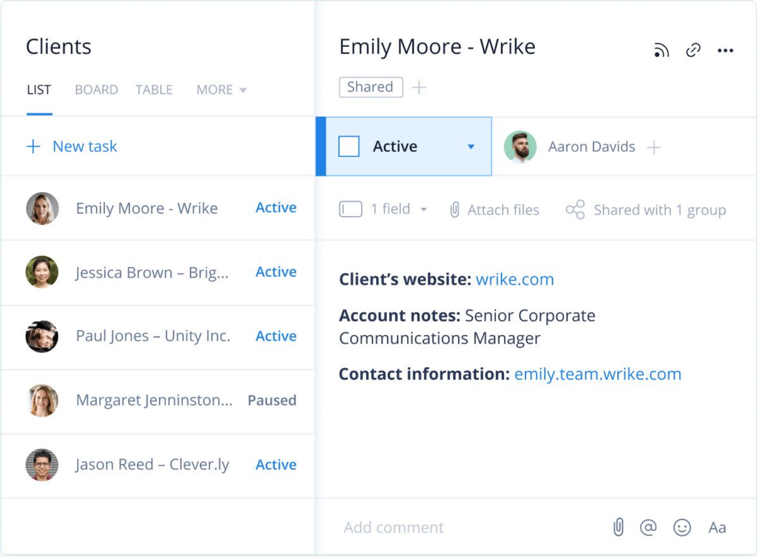

Web designers' workload
Edit
Share
Search by name
Oct 28 – Nov 3
Nov 4 – 10
Su 28
Mo 29
Tu 30
We 31
Th 1
Fr 2
Sa 3
Su 4
Mo 5
Tu 6
We 7
Th 8
Collin
Kate
Xavier
Create ne...


Version comparison
Image for Summer Sale.jpg
Version 4
Version 5
Exit comparison
Collin 1 day ago
@Xavier, please remove rivets from panels.
Collin 5 mins ago
@Xavier, looks like you missed this.


Polar Chart
Projects by Status
123 Indicator
Pie Chart
Column Chart
Bar Chart
Line Chart
Area Chart
Pivot
Polar Chart
Table
Tree Map
Calendar Heatmap
Scatter Map
12
8
11
9
6
3
10
9
6
12
New
In Briefing
In Progress
Owner Review
Completed
Cancelled


IT Projects – Current Sprint
List
Board
Table
Gantt Chart
Files
Timelog
Workload
Development (3)
Testing (2)
+ New task
[Story] Face ID support c
[Bug] Hyperlink does no
Investigate: Integration v
24-30
July 1-7
July 8-14
July 15-21
IT Projects Active
Implement SSO for Office 365
Plan the project
Implementation
Staff training
New billing system
Plan the new billing system


Featured
New/updated project in Wrike will create/update opportunity in Salesforce
Wrike Integrate
Featured
New Project created in Wrike gets assigned Custom Job Number
Wrike Integrate
Featured
New/Updated Task in Wrike creates an Event in Outlook Calendar
Wrike Integrate
Featured
Completed project in Wrike creates invoice in Quickbooks
Wrike Integrate


Clients
LIST
BOARD
TABLE
MORE
New task
Emily Moore - Wrike
Active
Jessica Brown – Brig...
Active
Paul Jones – Unity Inc.
Active
Margaret Jenninston...
Paused
Jason Reed – Clever.ly
Active
Emily Moore - Wrike
Shared
Active
Aaron Davids
1 field
Attach files
Shared with 1 group
Client's website: wrike.com
Account notes: Senior Corporate Communications Manager
Contact information: emily.team.wrike.com
Add comment
Aa

Data Engineering in Wrike

- SaaS means that we
  - Create
  - Support
  - Sell our product, and
  - Attract leads
- Help these teams speak the language of data
- We’ve got big space for data democratization

Data Engineering Team in Wrike

- 16 data engineers in 4 teams
- We’re supporting 250+ DAGs on production
- Up to 1200 tasks
- With median of 13 tasks
- ~10 updates of production or acceptance each day
- Helped 5 other teams to start using Airflow
- ~10-15% of our colleagues are using data engineering infrastructure and sources every month directly (>50% are using analytical reports or through integrations)

We’ve Started With

- First analysts using new Data Warehouse based on Google BigQuery
- Data provided by a single instance of Airflow
  - A lot of bugs found on production data
  - A lot of changes during review
  - A lot of delays in data
  - Partially available data
  - Lack of the full picture during code review and architecture problems
- And we wanted to start democratization
  - Reliable production
  - No changes on production, at least unexpected ones
    - No changes in Data Structure
    - No changes in Data Freshness

Acceptance Could Help

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
- Great for git blame

Via Data's Inferno by Wholesale Banking Advanced Analytics

Acceptance Environment

- Acceptance is an environment where changes are welcome
- To make sure that we aren't going to need them on production

No Changes on Production, at Least Unexpected Ones

- No Changes in Data Structure
- No Changes in Data Freshness
- No Changes during release from Acceptance to Production

No Changes in Data Structure

Implementation of Acceptance

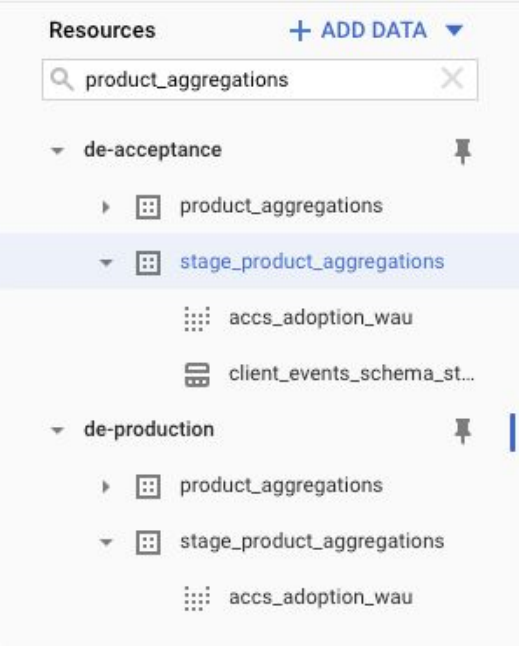
DEV	TST
• Quickly run your pipeline on a very small subset of your data • In our case 0.0025% of all data • Nothing will make sense, but it's a nice integration test	• Select a subset of your data for data that you know • Immediately see if something is off • Still quick to run

ACC	PRD
• Carbon copy of production • You can check if you feel comfortable pushing to PRD • Give access to a Product Owner for them to check	• Greenlight procedure for merging from ACC to PRD • Manual operation • Great for git blame

Via Data's Inferno by Wholesale Banking Advanced Analytics

Acceptance on DB Side. BigQuery

- Acceptance and production are **different projects** in the notation of BigQuery
- Isolated quotas and limits (resources)
- BigQuery allows for cross-project queries
  - So we store on acceptance only changed data
  - And take source data from production.



Dataflow Example

```
`de-acceptance.aggregations.client`
(v1)
```

```
SELECT ...
FROM
`de-production.events.client`
GROUP BY ...
```

Dataflow Example

```
SELECT ...
FROM
`de-production.events.client`
GROUP BY ...
```

```
`de-acceptance.aggregations.client`
(v1)
```

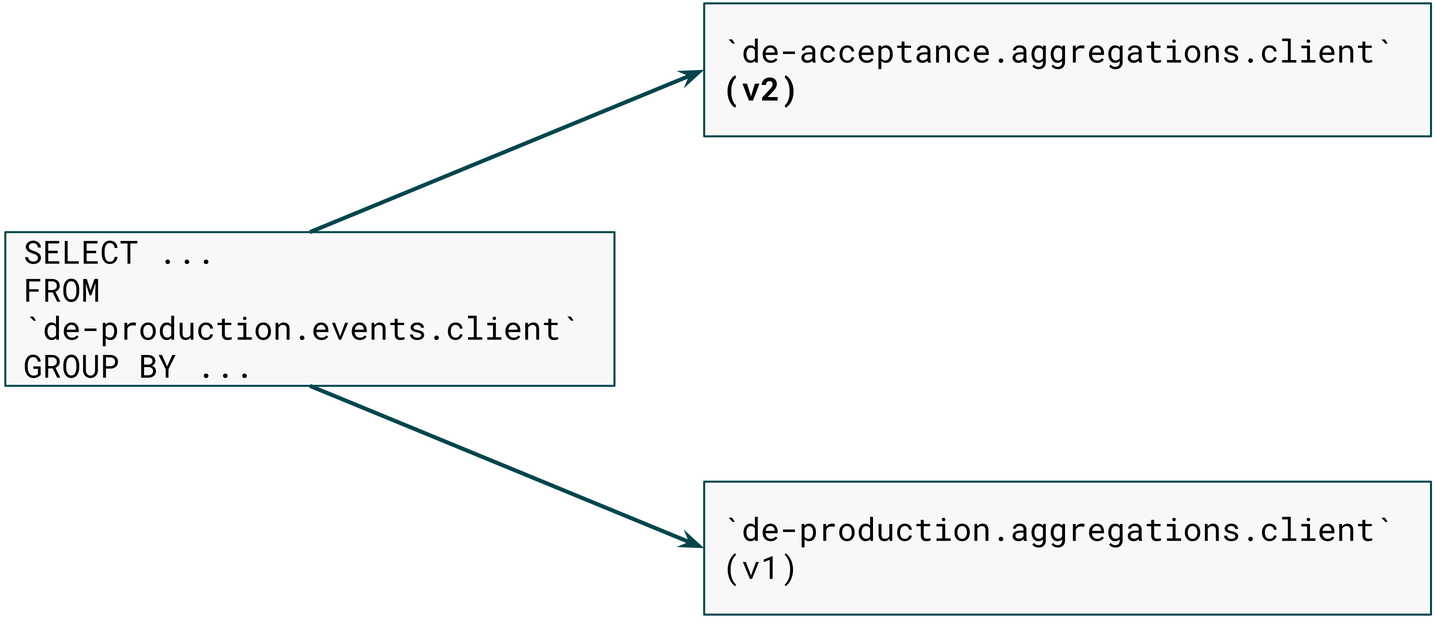
```
`de-production.aggregations.client`
(v1)
```

Dataflow Example

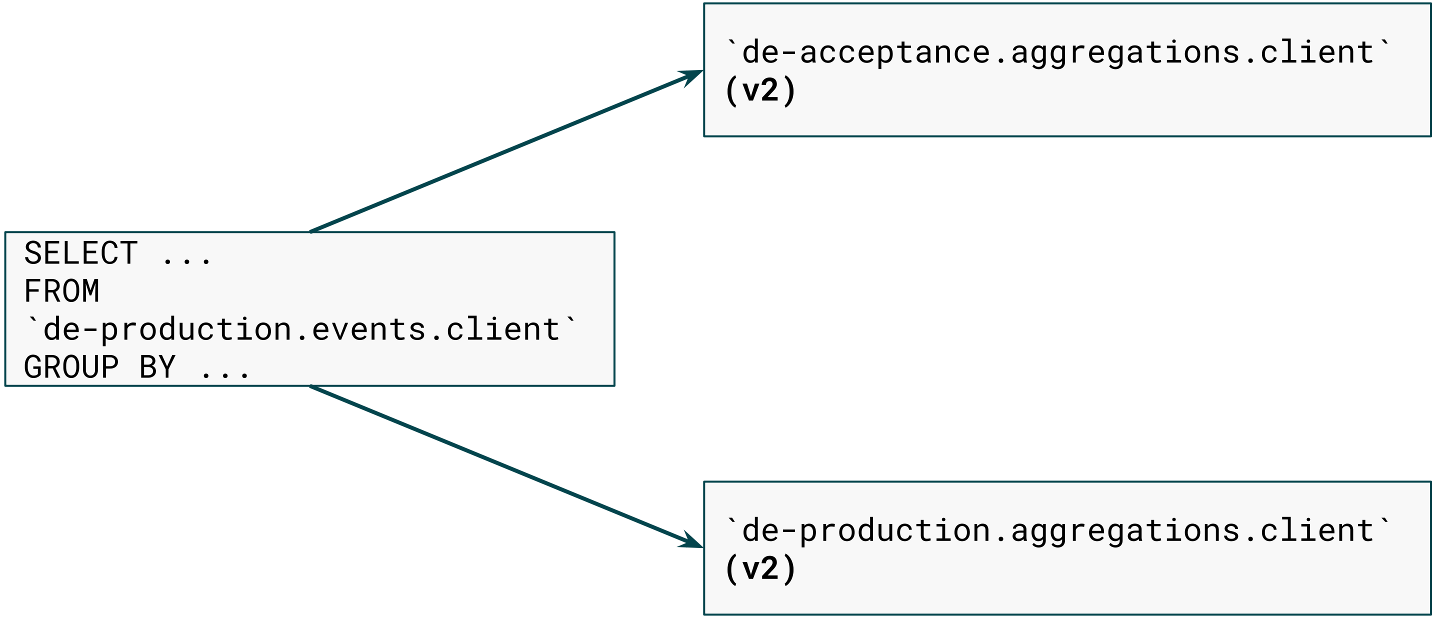
```
SELECT ...
FROM
`de-production.events.client`
GROUP BY ...
```

```
`de-production.aggregations.client`
(v1)
```

Dataflow Example



Dataflow Example

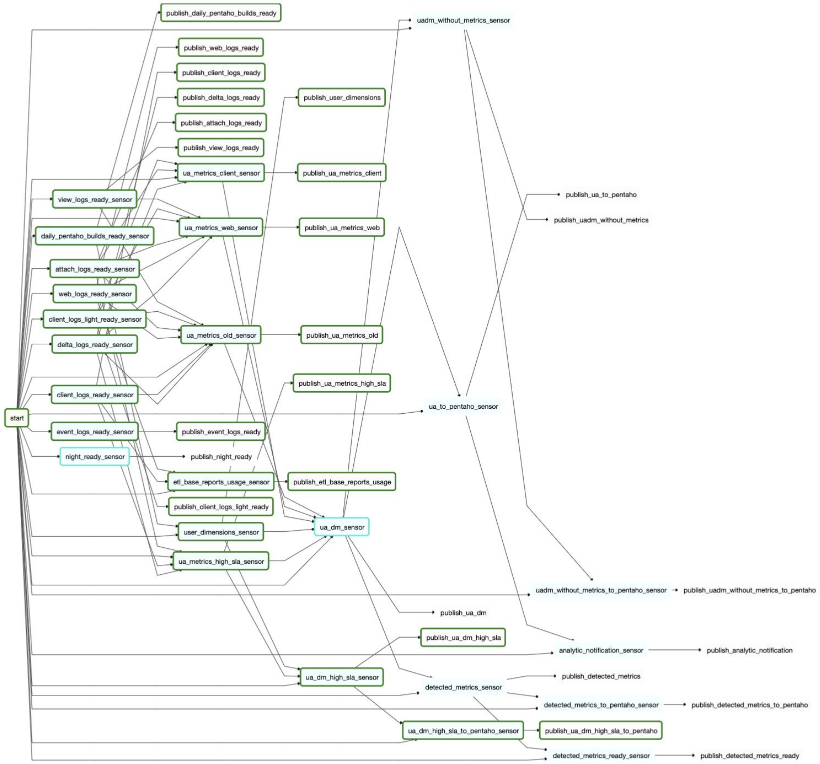


Interface Separation on Other DBs

- Look for interface separation and resource isolation
  - And think about cost tradeoffs
- Approaches for **interface separation**
  - Schemas
  - Base directory name
  - Naming (bucket names for example)
  - Separate DBs
- Approaches for **resource isolation** (several trade offs with cost)
  - On service layer (separate DBs)
  - On DB side (e.g. roles, connection pools, quotas)
  - Airflow side (e.g. pools, priority, parallelism limit)
  - On monitoring side (e.g. query killer)

No Changes in Data Freshness

Beautiful DAG with 150 Tasks



No Changes in Data Freshness (2 out of 3)

Dataflow Example

```
SELECT ...
FROM
`de-production.events.client`
GROUP BY ...
```

DAG: events loader (prod)

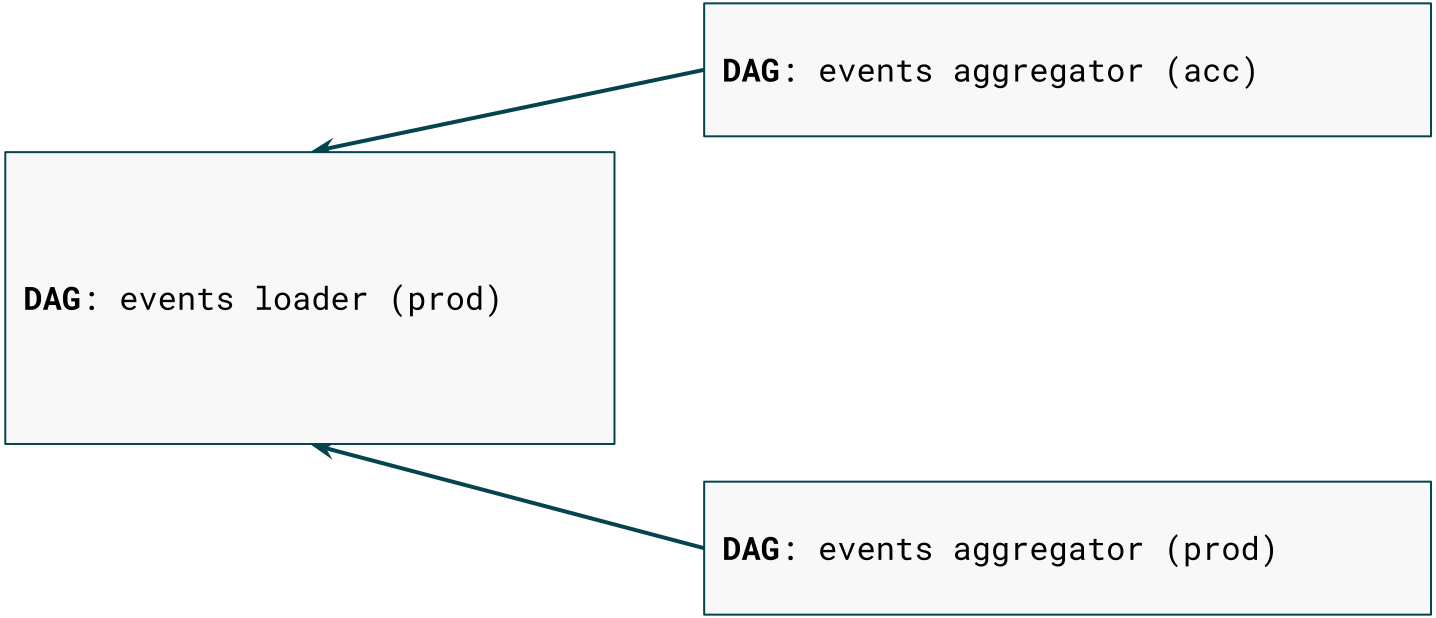
```
`de-acceptance.aggregations.client`
```

DAG: events aggregator (acc)

```
`de-production.aggregations.client`
```

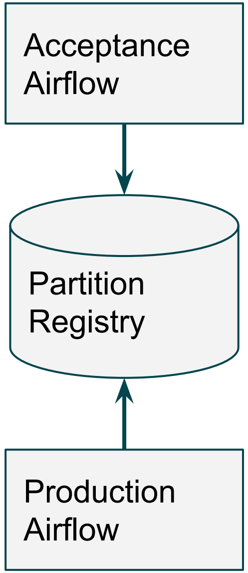
DAG: events aggregator (prod)

Execution Example



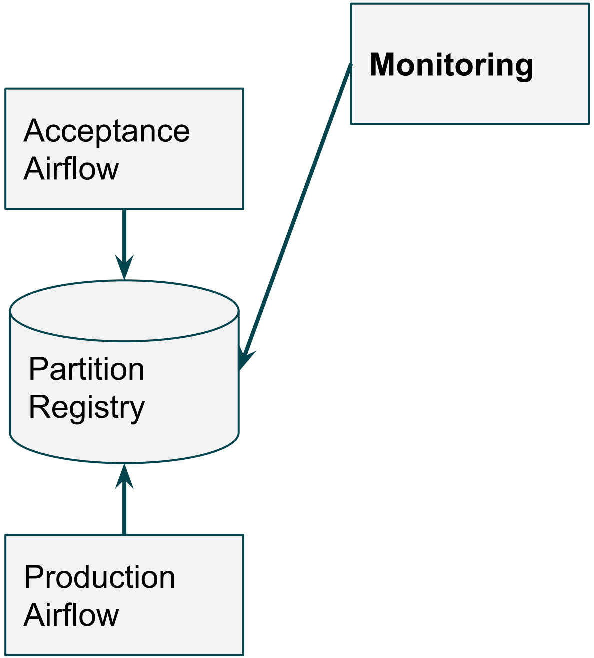
Separate Airflows

- Coordinated via Postgres database named Partition Registry
    - Inspired by [Functional Data Engineering](#) by Maxime Beauchemin
    - Partition — unit of work for DAG, typically hour/day/week in a table
- State of partition published using operator
    - Explicitly publish sources
    - After all data validations have passed
- Wait for dependent sources using sensor
    - Automatically identify the strategy for interval
        - Week-on-hour, Month-on-day, custom catch-ups, etc.



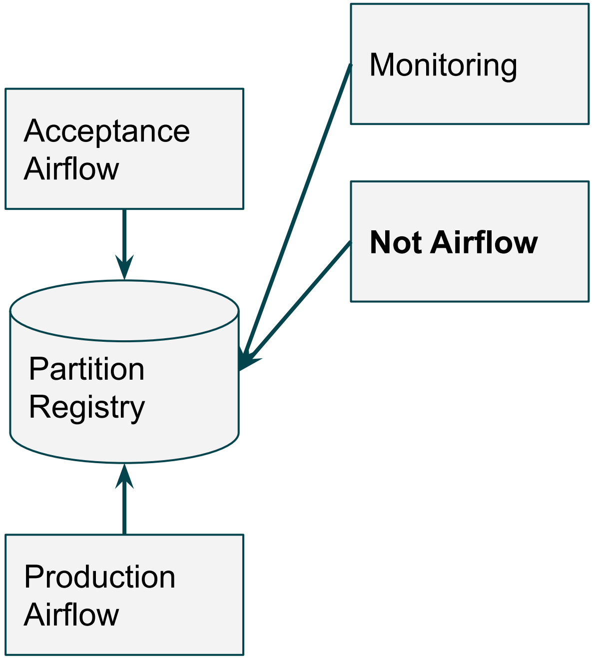
Partition Registry Now

- Custom monitoring and alerts:
  - Severity of delays for partitions (DAG SLAs)
  - Base for data lineage



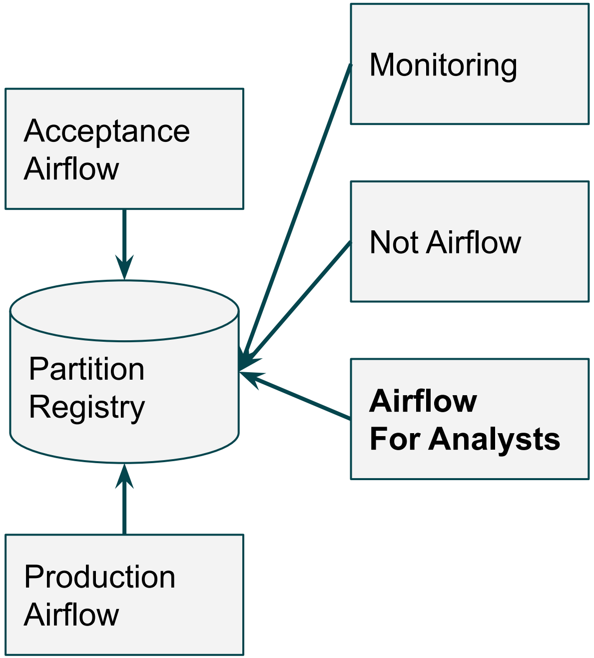
Partition Registry Now

- Custom monitoring and alerts:
  - Severity of delays for partitions (DAG SLAs)
  - Base for data lineage
- Not Airflow: Pentaho DI and Old Jenkins Pipelines



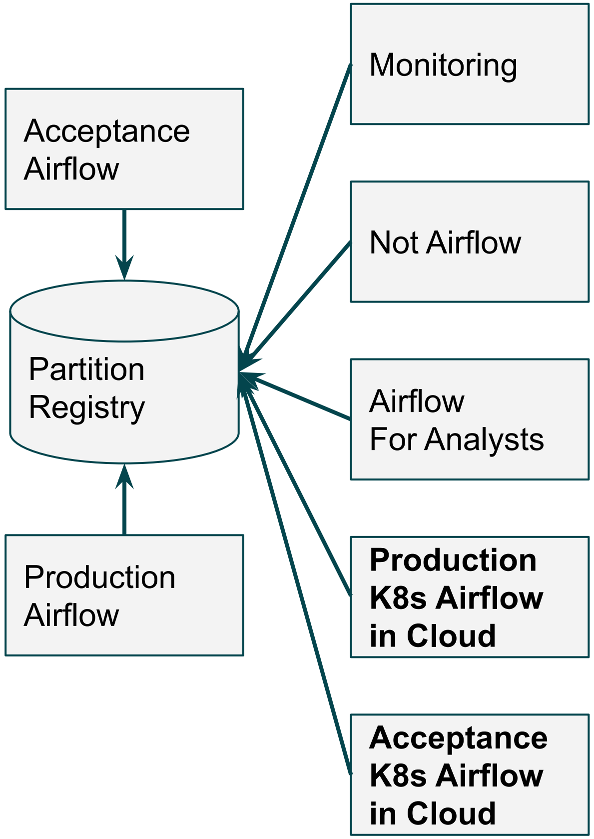
Partition Registry Now

- Custom monitoring and alerts:
  - Severity of delays for partitions (DAG SLAs)
  - Base for data lineage
- Not Airflow: Pentaho DI and Old Jenkins Pipelines
- Airflow for Analysts: isolated resources and credentials



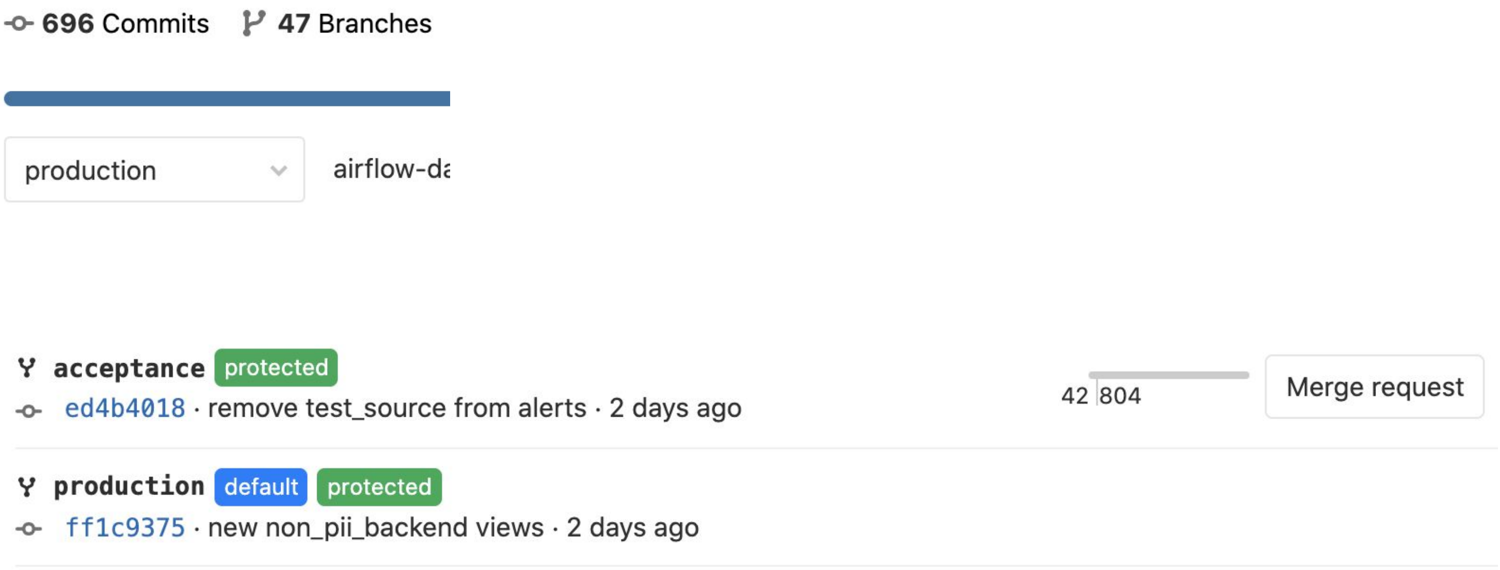
Partition Registry Now

- Custom monitoring and alerts:
  - Severity of delays for partitions (DAG SLAs)
  - Base for data lineage
- Not Airflow: Pentaho DI and Old Jenkins Pipelines
- Airflow for Analysts: isolated resources and credentials
- K8s Airflow in Cloud
  - Easy switch with on-prem
  - Zero downtime migration
  - Data locality



No Changes During Release from Acc to Prod

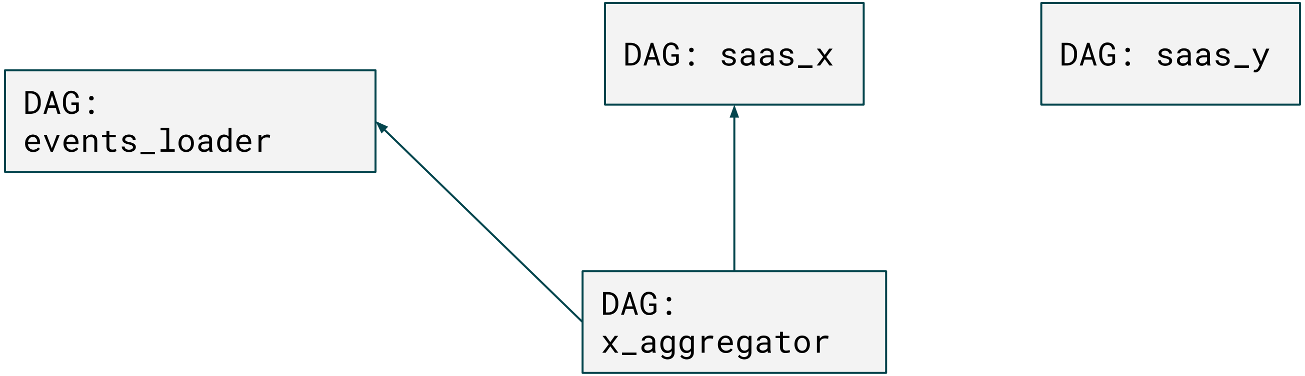
Acceptance Told Us Where We Went Wrong



Fast and Reliable Release

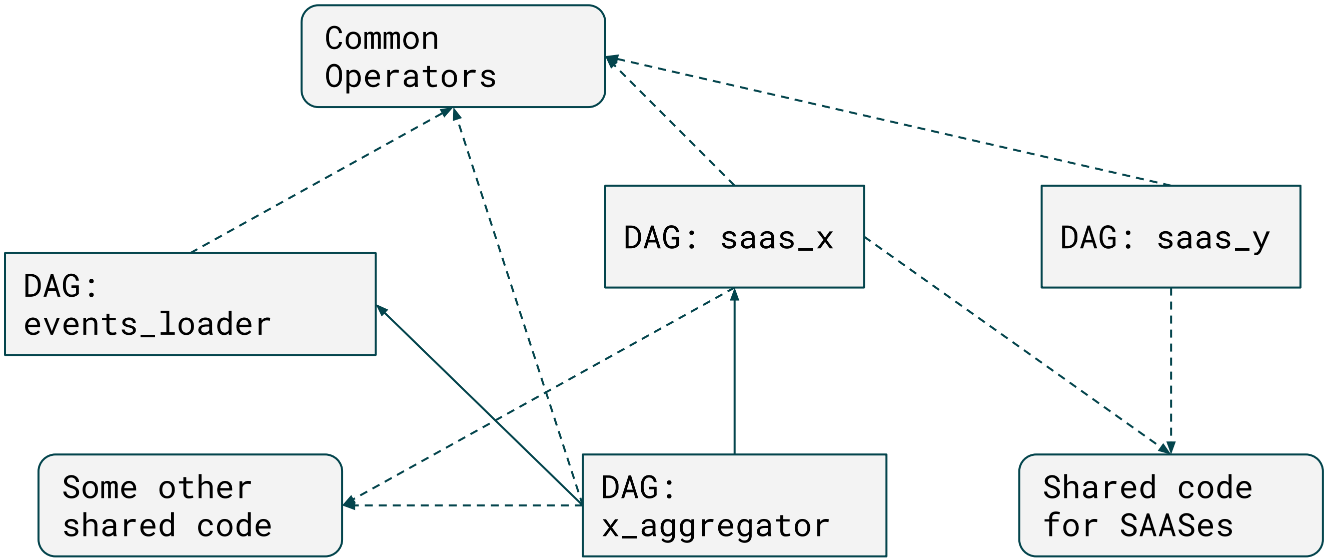
- We need code freeze to test dependent parts
- But we need 10 releases per day
    - So, we need to freeze as little as possible
        - But still review and test every change made

Dependency Scheme



No Changes During Release Process (3 out of 3)

Dependency Scheme with Code



No Changes During Release Process Means

- Good data isolation during release
- Good code isolation during release

Bad Data Isolation Is When

- You recalculate your data and get different results
- Data distribution changes
- Data distribution does not change when it should
- Analytical dashboard starts to focus on the wrong things
- You achieve your results a lot faster :)
- Something else is wrong and you don’t know about it.

So if Data Changes

- It’s safe to assume
  - Review is no longer valid
  - Manual testing is no longer valid
  - Data sources may be corrupted
- So before the release of data change
  - Notifying all stakeholders of all changed dependent sources
  - Checking that everything works correctly on acceptance
  - Making atomic release
- We’re helping to implement recalculation strategies
  - Recalculating everything and keeping it up-to-date
  - Preserving history for metrics in prestaging
  - Supporting and gradual deprecation of old version of metrics

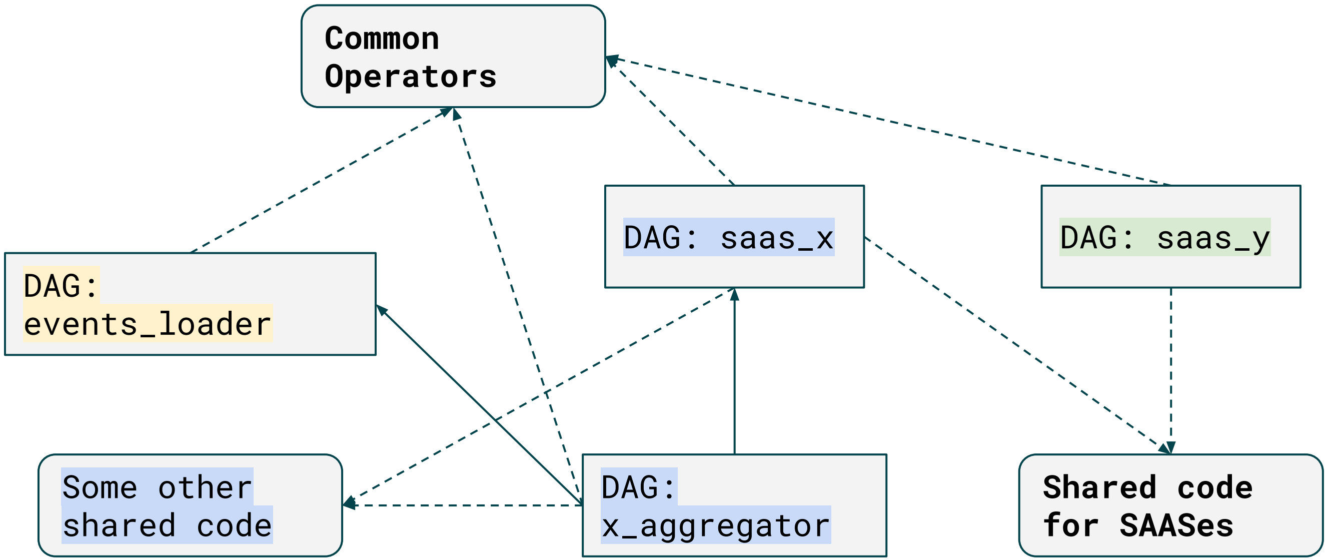
Keeping Track of Data Isolation

- Knowing when dependencies are updated after release to production
  - Notifications from other teams
  - Dependency on exact version of partition
    - Makes it easier to switch between acc and prod in code
  - Validation of data on your side
    - Great Expectations to explicitly specify your assumptions on data nature
    - Anomaly detection
- Finding all dependent sources before release to the production
  - Manual
    - BigQuery history
    - Search in git repository
  - Data Lineage + release process
  - Autotests

Good Code Isolation

- Bad code isolation means you have a bug and your pipeline is not working
- This happens when when 2+ DAGs use the same code
  - You update code or library and other DAG fails
- Two types of failure
  - Scheduler/Web Server — appears immediately, hard isolation (fat-zip, boilerplate)
  - Worker — visible during execution, easy isolation (k8s, venv)
    - Can be at the end of a 4 hour-long task at the start of the next month :(
- How do we avoid this?
  - There is 20% of code used in 80% of cases
    - We're moving it to the library, test and track backward compatibility
  - We have a shared code that is changed rarely
    - This code should be as private as possible to make sure that we're not reusing it
      - The main reason for DAGs to be included in the single repo or merge request

Dependency Scheme with Code



How Do We Reason About Reliability?

- Our production is very predictable
- All interface changes reviewed on separate environment
  - We keep track of all data dependencies and communicate the change to all stakeholders throughout the pipeline
  - Every source on production is reviewed, supported by several data engineers, have a clear time of readiness and all errors are communicated to all stakeholders
- We're using partition registry
  - To isolate resources of acceptance
    - As little recalculation as possible
  - To integrate Airflow with separate creds and resources to other teams
- Acceptance could be made cheaper

Thank You!
Any Questions?

Alexander Eliseev at Airflow Slack
alexander.eliseev@team.wrike.com
https://github.com/eliseealex