WIKIMEDIA FOUNDATION

Enhancing Airflow for Analytics, Data Engineering, and ML at Wikimedia

Ben Tullis, Balthazar Rouberol
FOSDEM 2025

WIKIMEDIA FOUNDATION

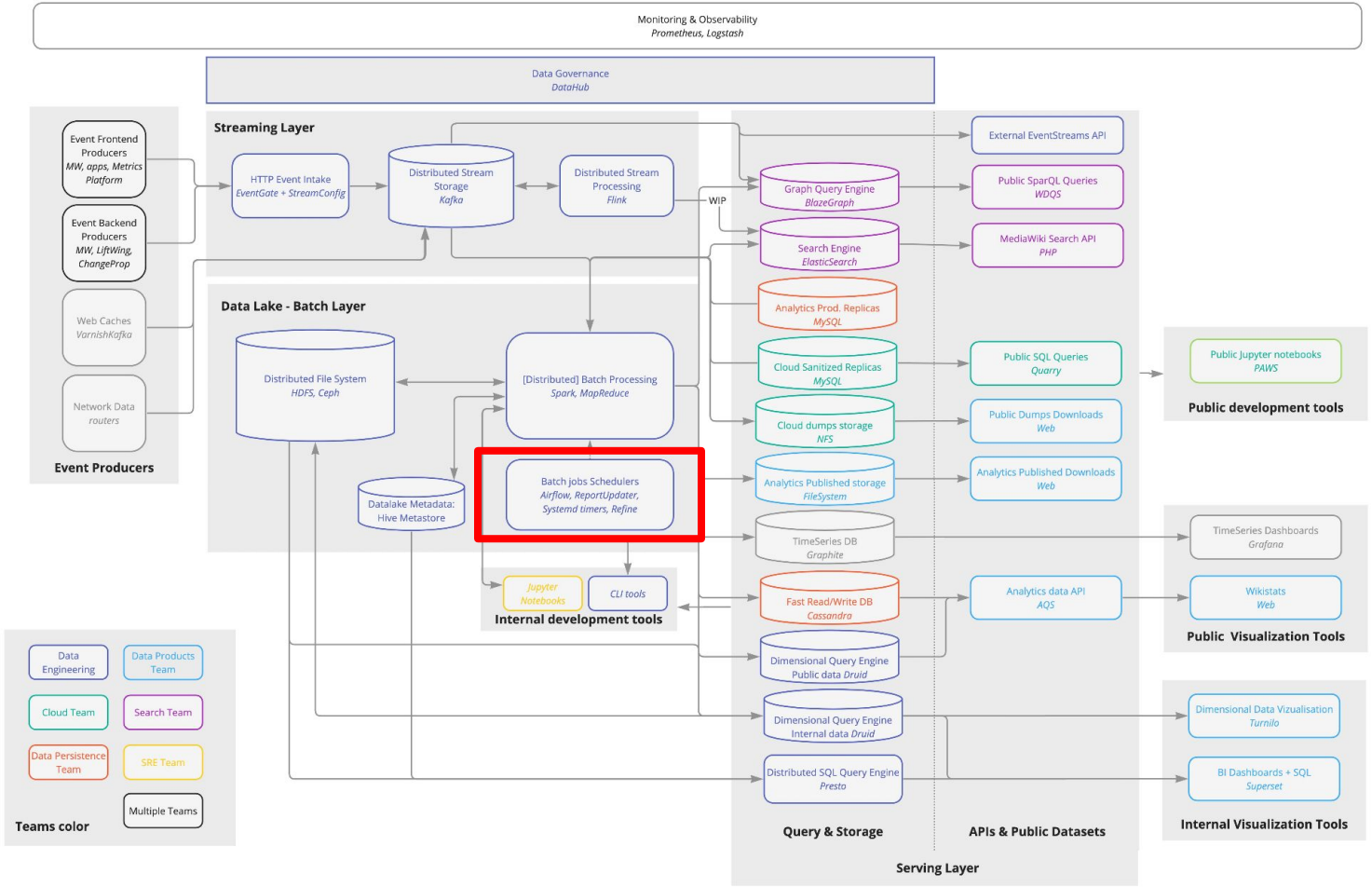
Let’s set the scene

Wikimedia's Data Platform

"A collection of systems and services that enable data producers and consumers to discover, use, and collect data to derive insights, conduct research, and build new data products."

https://wikitech.wikimedia.org/wiki/Data_Platform

- Data Engineering
- Experiment Platform
- Metrics Platform
- Search Platform
- Data Platform SRE


Monitoring & Observability
Prometheus, Logstash
Data Governance
DataHub
Event Frontend Producers
MW, apps, Metrics Platform
Event Backend Producers
MW, LiftWing, ChangeProp
Web Caches
VarnishKafka
Network Data
routers
Event Producers
Streaming Layer
HTTP Event Intake
EventGate + StreamConfig
Distributed Stream Storage
Kafka
Distributed Stream Processing
Flink
WIP
Data Lake - Batch Layer
Distributed File System
HDFS, Ceph
[Distributed] Batch Processing
Spark, MapReduce
Datalake Metadata:
Hive Metastore
Batch jobs Schedulers
Airflow, ReportUpdater, Systemd timers, Refine
Jupyter Notebooks
CLI tools
Internal development tools
Graph Query Engine
BlazeGraph
Search Engine
ElasticSearch
Analytics Prod. Replicas
MySQL
Cloud Sanitized Replicas
MySQL
Cloud dumps storage
NFS
Analytics Published storage
FileSystem
TimeSeries DB
Graphite
Fast Read/Write DB
Cassandra
Dimensional Query Engine
Public data Druid
Dimensional Query Engine
Internal data Druid
Distributed SQL Query Engine
Presto
Query & Storage
External EventStreams API
Public SparQL Queries
WDQS
MediaWiki Search API
PHP
Public SQL Queries
Quarry
Public Dumps Downloads
Web
Analytics Published Downloads
Web
Analytics data API
AQS
APIs & Public Datasets
Serving Layer
Public Jupyter notebooks
PAWS
Public development tools
TimeSeries Dashboards
Grafana
Wikistats
Web
Public Visualization Tools
Dimensional Data Vizualisation
Turnilo
BI Dashboards + SQL
Superset
Internal Visualization Tools
Data Engineering
Data Products Team
Cloud Team
Search Team
Data Persistence Team
SRE Team
Multiple Teams
Teams color

Airflow within the Data Platform

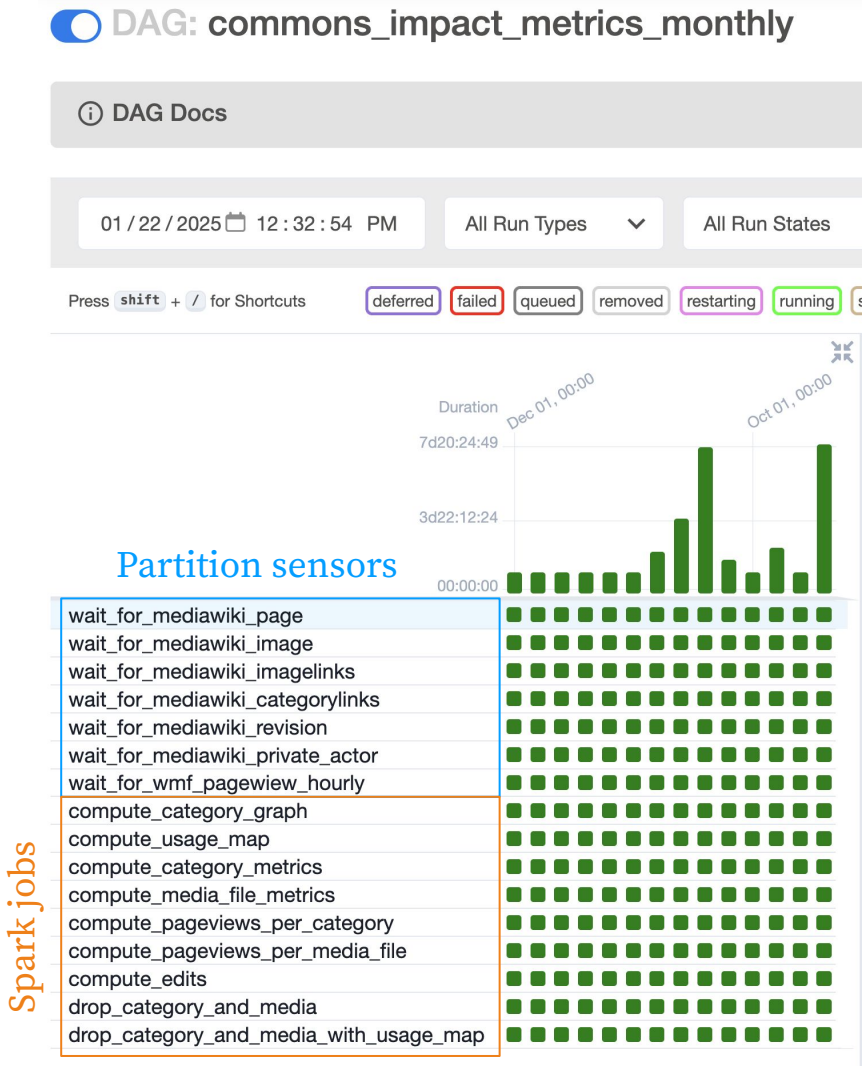
Self service data pipelines for WMF engineering and data-science teams.

- **Analytics**: produce quality analytics metrics and datasets for consumption by WMF and the community
- **Search**: facilitate the timely update of Mediawiki search and Wikidata graph DB queries
- **Research**: generate datasets for our research teams, related to demographic, pageviews, etc
- **Product analytics**: enable WMF engineering teams to work with feature instrumentation

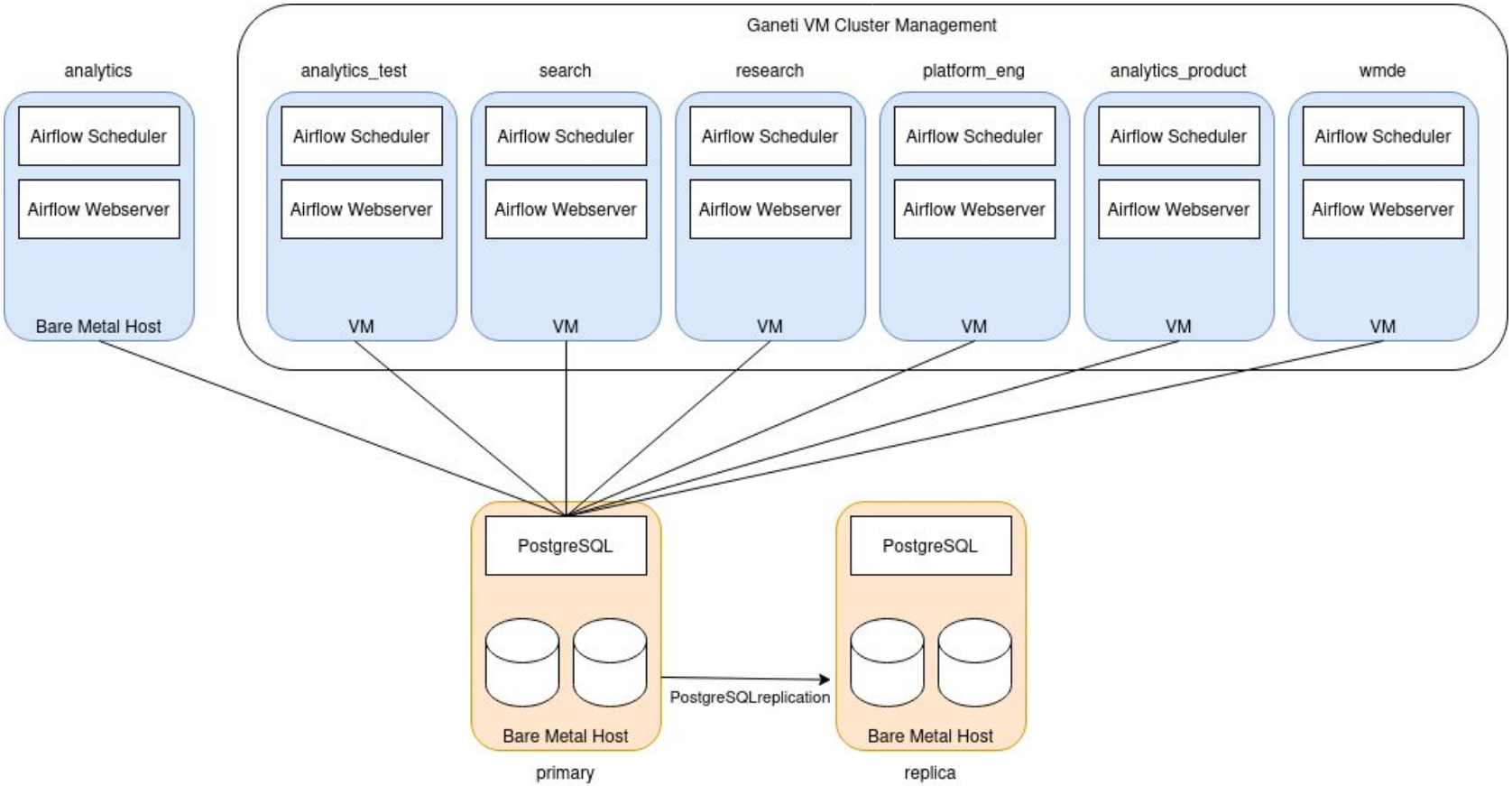
A typical data pipeline

What: batch data processing jobs sourcing data from HDFS and writing to HDFS, Cassandra, Druid, Swift, ...
How: mostly Spark jobs running on a YARN cluster, working on data partitions, based on their availability

Airflow is used to **orchestrate** these Spark jobs.



The initial Airflow infrastructure



The initial Airflow infrastructure

- Airflow components run as systemd services, using custom Puppet code
- Kerberos authentication to Hadoop, YARN, Hive & Spark
- Local Executor
- Local task logging
- A shared PostgreSQL server with manual failover

Problem statement: Reliability

- LocalExecutor forced us to cause downtime when we had to restart the airflow scheduler
- The lack of HA PostgreSQL caused DB maintenances to affect all instances
- Local logging and DAGs generating a lot (1500+) of mapped tasks sometimes filled up our disk, causing outages
- High number of tasks causes a high number of connections on PG

Problem statement: Scalability

- Some instances were CPU/network/storage bound, causing delays during unexpected backfills
- LocalExecutor made it hard to horizontally scale
- Using bare metal hosts made it hard to vertically scale often

Problem statement: UX & Security

- No RBAC / logging in place as the UIs are only internally exposed, through SSH tunnels
- Everybody is Admin
- No per-component firewall rules

WIKIMEDIA FOUNDATION

And they have a plan 🤖

https://phabricator.wikimedia.org/T362788

General direction

- Single points-of-failure should be eliminated
- Airflow services should be resilient to the downtime of any host.
- The system should make use of systems we already invested in
- The migration should be as transparent as possible for users

The Data Platform Kubernetes cluster

```
root@deploy2002:~# kubectl get nodes
NAME                                STATUS   ROLES           AGE
dse-k8s-ctrl1001.eqiad.wmnet        Ready    control-plane   707d
dse-k8s-ctrl1002.eqiad.wmnet        Ready    control-plane   707d
dse-k8s-worker1001.eqiad.wmnet      Ready    <none>          706d
dse-k8s-worker1002.eqiad.wmnet      Ready    <none>          706d
dse-k8s-worker1003.eqiad.wmnet      Ready    <none>          706d
dse-k8s-worker1004.eqiad.wmnet      Ready    <none>          706d
dse-k8s-worker1005.eqiad.wmnet      Ready    <none>          703d
dse-k8s-worker1006.eqiad.wmnet      Ready    <none>          703d
dse-k8s-worker1007.eqiad.wmnet      Ready    <none>          703d
dse-k8s-worker1008.eqiad.wmnet      Ready    <none>          703d
dse-k8s-worker1009.eqiad.wmnet      Ready    <none>          153d
```

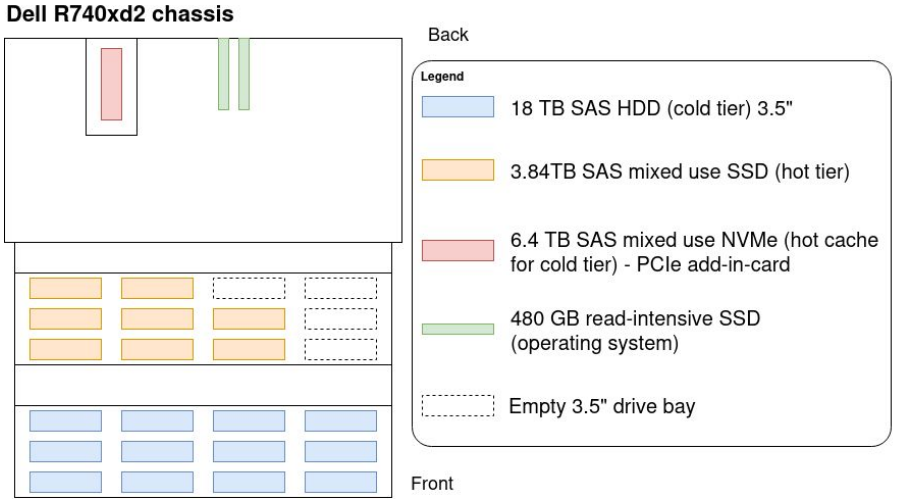
The Data Platform Ceph Cluster

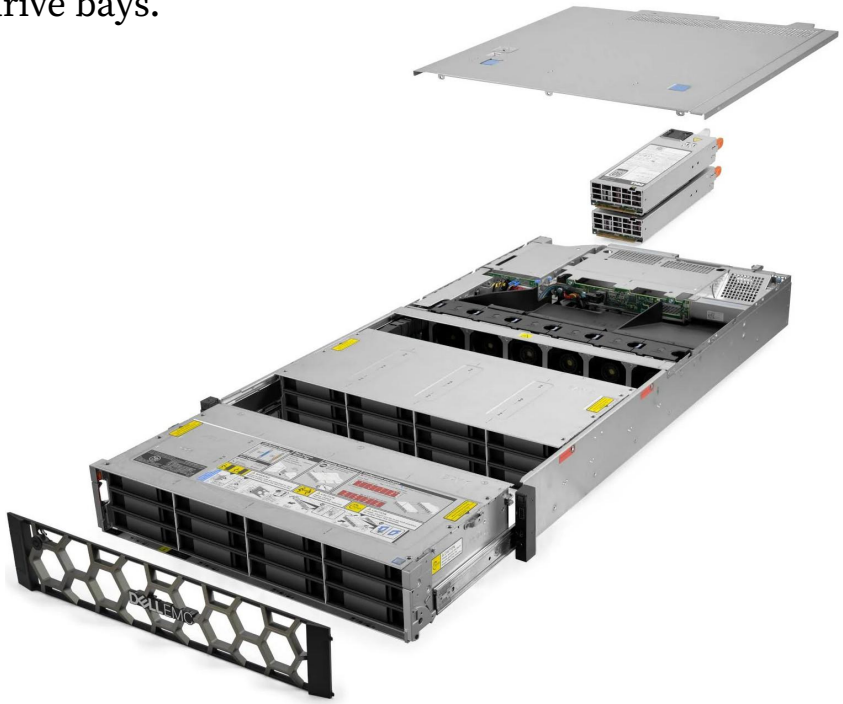
Five servers racked by November 2022

A high-density chassis, optimised for disk storage. 24 drive bays.

1PB of available HDD space

150TB of available SSD space





Proposed architecture

Airflow

- Uses the `KubernetesExecutor` to run DAG tasks as Pods
- Stores the task logs remotely
- Has its dedicated HA PostgreSQL cluster

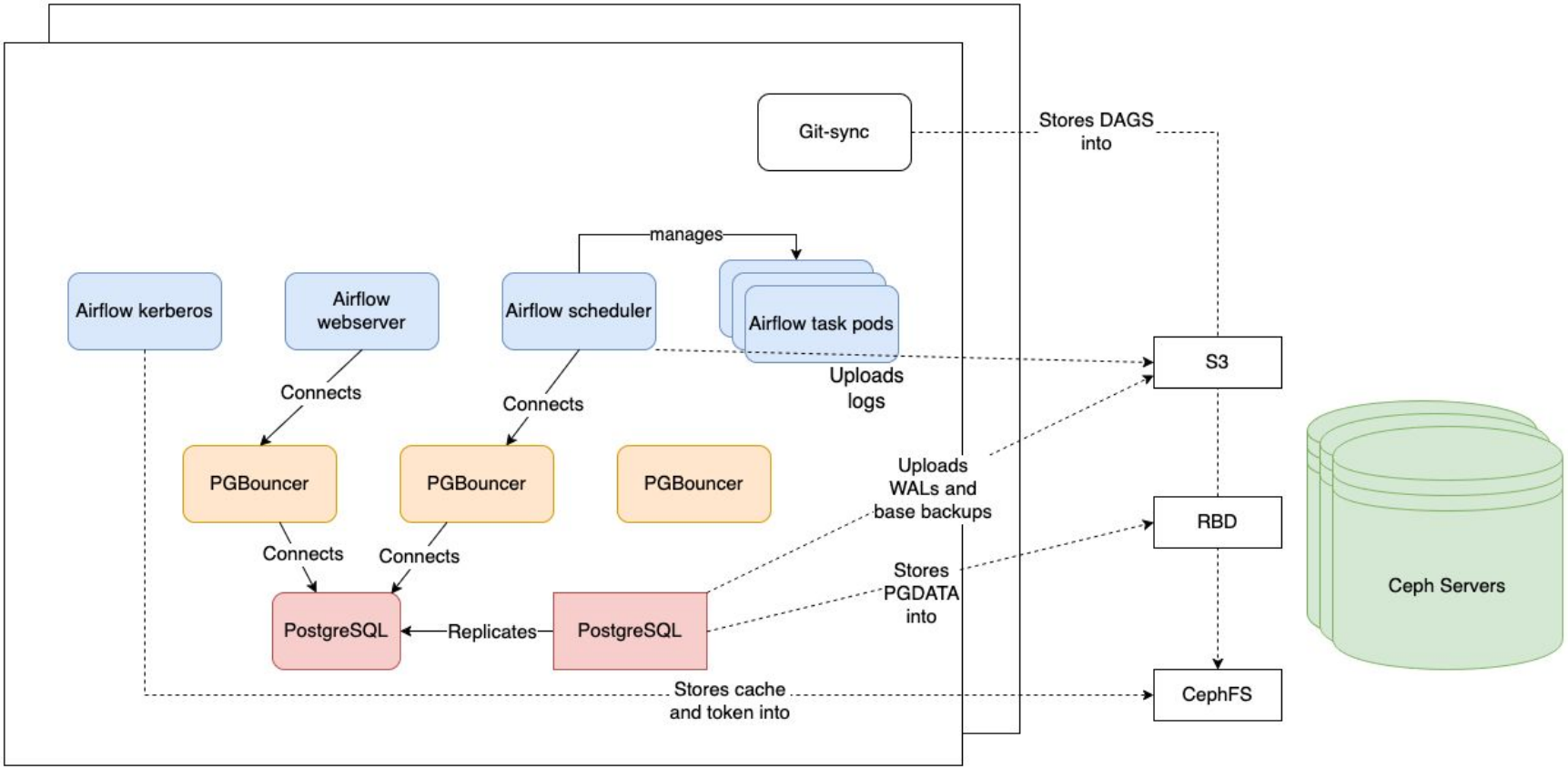
PostgreSQL

- Managed by a Kubernetes Operator
- Automatic leader failover
- Exports WALs and base backups to Ceph

Ceph

- RBD (block device) for PG data
- CephFS (distributed filesystem) for DAGs and Kerberos tokens
- RadosGW (S3) for PG WALs, PG base backups and task logs

Proposed architecture



Setting up Ceph with Kubernetes

- Integration of RBD with Kubernetes done using `ceph-csi-rbd`
- Integration of CephFS with Kubernetes done using `ceph-csi-cephfs`
- RadosGW/S3 exposed behind an anycast endpoint to minimize network hops and maximize bandwidth

```
brouberol@deploy2002:~$ kubectl get pvc
NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
airflow-dags-pvc                    Bound    pvc-3bd48ea4-993f-4dde-bcdd-fb20aa60867d   100Mi      RWX            ceph-cephfs-ssd
airflow-kerberos-token-pvc          Bound    pvc-3178b8e1-c0a0-4cff-9b8e-8bc50ce05b49   12Mi       RWX            ceph-cephfs-ssd
postgresql-airflow-test-k8s-1       Bound    pvc-763458f4-4af7-401f-bf76-5b9e2cab6486   5Gi        RWO            ceph-rbd-ssd
postgresql-airflow-test-k8s-1-wal   Bound    pvc-aea298f1-2550-4678-97ef-b522dc5c3b40   15Gi       RWO            ceph-rbd-ssd
postgresql-airflow-test-k8s-2       Bound    pvc-0a8ea1d8-ad55-4b21-9791-0aed1f2fe84b   5Gi        RWO            ceph-rbd-ssd
postgresql-airflow-test-k8s-2-wal   Bound    pvc-c3a97fcc-1689-4bbc-9baf-fbe5e23e77c7   15Gi       RWO            ceph-rbd-ssd
```

Running PostgreSQL in Kubernetes

- We decided* to use `cloudnative-pg`
- Each cluster is composed of 2 PG servers and 3 PGBouncer pods
- WALs and base backups are uploaded to S3
- Backups exported outside of Ceph every night

```
brouberol@deploy2002:~$ kubectl get cluster
NAME                          AGE    INSTANCES   READY   STATUS                     PRIMARY
postgresql-airflow-test-k8s   117d   2           2       Cluster in healthy state   postgresql-airflow-test-k8s-2
```

* https://phabricator.wikimedia.org/T362999

Security and RBAC

- Public facing: https://airflow-analytics.wikimedia.org
- Authenticated UI (OIDC)
- Authenticated API (Kerberos)
- User roles and permissions derived from LDAP groups

Our migration plan

- Migrate the webserver first
- Test a DAG of each Operator in our test instance
- Migrate the database
- Migrate the scheduler and DAGs
- Fix DAG errors until morale improves

What held us back

- Kerberos API authentication was broken
  - Fixed by https://github.com/apache/airflow/pull/43662
  - Released in `apache-airflow-providers-fab 1.5.1`
- The codegen'd Airflow API client does not support Kerberos
  - We have an internal hack to make it work
- Migrating instances with a lot of DAGs was a large lift and shift
  - Some took several attempts as we had to rewrite/fix DAGs
  - It took some finessing to integrate task pods with our service mesh

What we get out of it today

- Improved reliability
- Improved security
- Improved scalability
- Improved UX
- Reusable infrastructure building blocks
- Increased adoption

What we'll invest on

- Leverage `KubernetesPodOperator` to allow DAG authors to run custom Docker images
- Migrate the monthly Wikipedia dumps* bespoke system to an Airflow DAG
- Help other teams get started with Airflow

* https://meta.wikimedia.org/wiki/Data_dumps

WIKIMEDIA FOUNDATION

Thank you!