

Evolution of Airflow at Uber



Shobhit Shah

STAFF SOFTWARE ENGINEER
@ Uber



Sumit Maheshwari

TECH LEAD
@ Uber



AGENDA

01 Data workflow management at Uber

02 Doubling down on Airflow

03 Diverging & scaling

04 Multi - executors, serializers, region

05 Piper EcoSystem

06 Piper/Airflow catchup

About Uber

70

#Countries

10k

#Cities

30M

#Trips per day

156M

#Monthly active users

7.4M

#Monthly active drivers

1M

Eats merchant partners

Key Data Platform Use Cases



Business Intelligence and Analytics

- Financial reporting (P&L, budget)
- Track gross bookings, promo spend etc
- Partners / Merchants dashboards
- Billing for contractors, 3rd party vendors



Business and Customer Processes


- Track orders and deliveries issues rate
- Reporting trip data to government
- Audit customers complaints and reported incidents
- Send promotional emails, recommendations for eater, rider etc



Predictive Analytics, Risk & Fraud

- Forecasting resource needs
- Identify safety incidents that require intervention
- Users sign up fraud detection and prevention
- Malicious attacks detection and prevention

Data workflow management at Uber

- 
- 2016**
Data workflow management at Uber
 - 2017**
Doubling down on Airflow
 - 2019**
Diverging & scaling
 - 2022**
Multi - executors, serializers, region
 - 2024**
Hybrid cloud architecture

Data Workflow Landscape



Gaps & Opportunities



Lack of Standardization

- Different teams using different tools
- Collaboration obstacles
- Duplicated efforts



No Centralized Support


- Painful Upgrades
- Difficult Migrations
- Security & compliance risks



Inefficient Resource Utilization

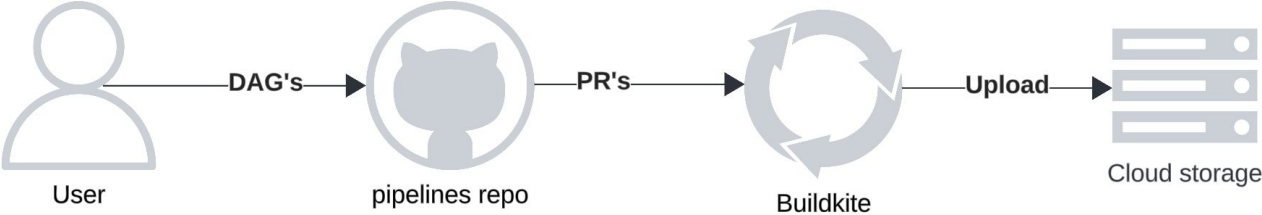
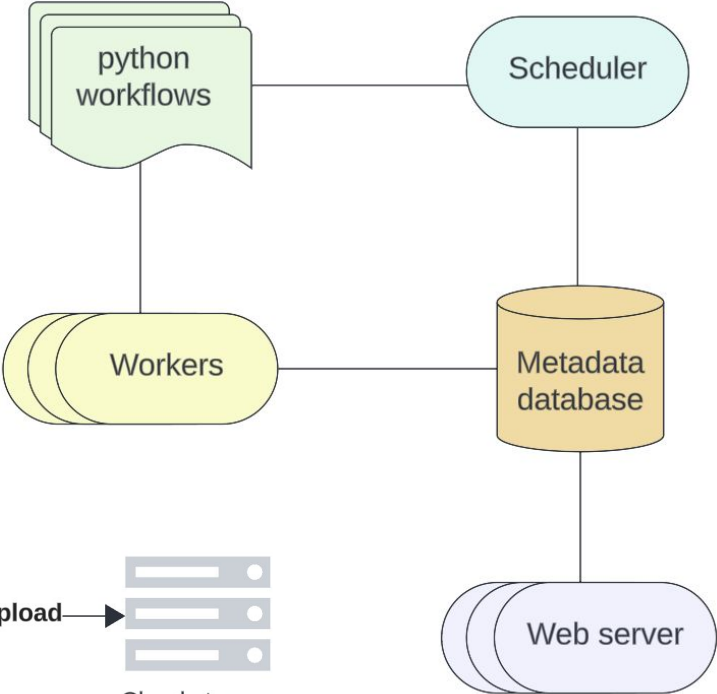
- Underutilized resources
- Overprovisioning
- Wasted compute and storage resources

Doubling down on Airflow

- 
- 2016**
Data workflow management at Uber
 - 2017**
Doubling down on Airflow
 - 2019**
Diverging & scaling
 - 2022**
Multi - executors, serializers, region
 - 2024**
Hybrid cloud architecture

V1 - Bootstrapping Airflow as Piper

- Pipelines git mono repo
- Airflow DSL



Gaps & Opportunities

Scalability Issues

Difficulty processing high number of DAG's

Performance

High processing and scheduling time DAG's

Integration

Platform support for only Java & Go

Diverging to Scale



2016

Data workflow management at Uber



2017

Doubling down on Airflow



2019

Diverging & scaling



2022

Multi - executors, serializers, region

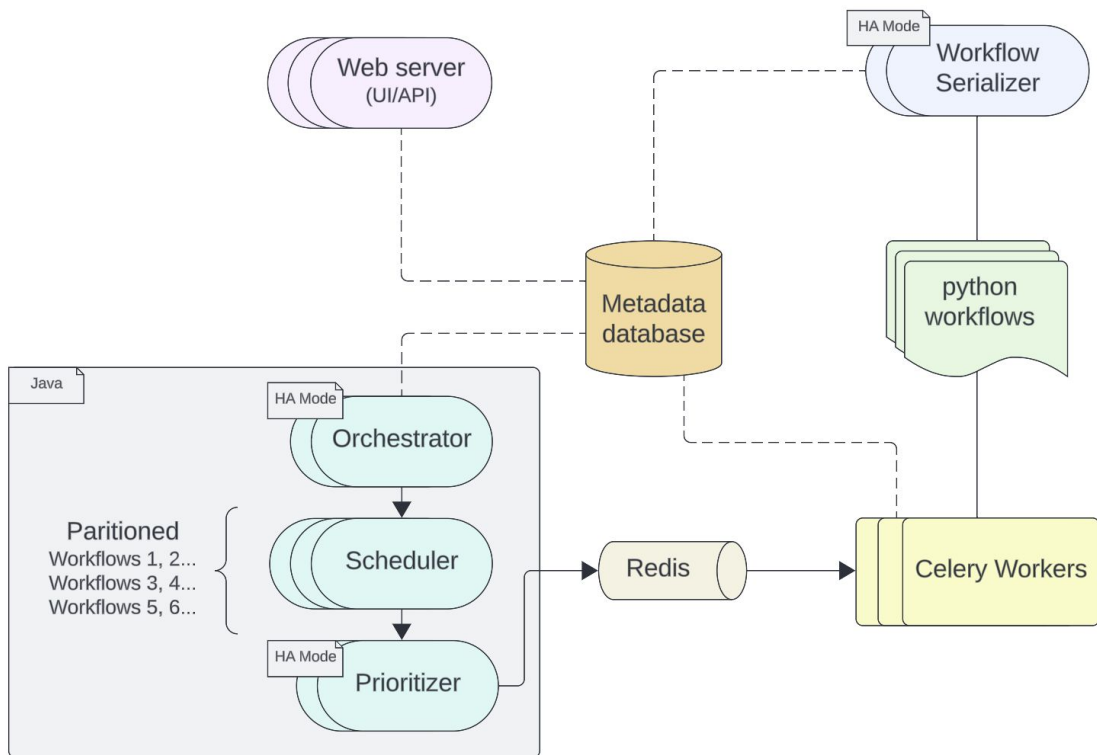


2024

Hybrid cloud architecture

V2 - Diverging & Scaling

- Scheduler isolated from user code
- Serialization format defined
- Scaled-out & re-written in Java
- Zookeeper for HA & leader election
- Jumpstart (Freshness based triggers)
- UI based authoring experience



Gaps & Opportunities

Noisy neighbours

Low tier tasks consuming higher resources and affecting high tier tasks

Performance

High delay in new pipeline processing and updating DAG's

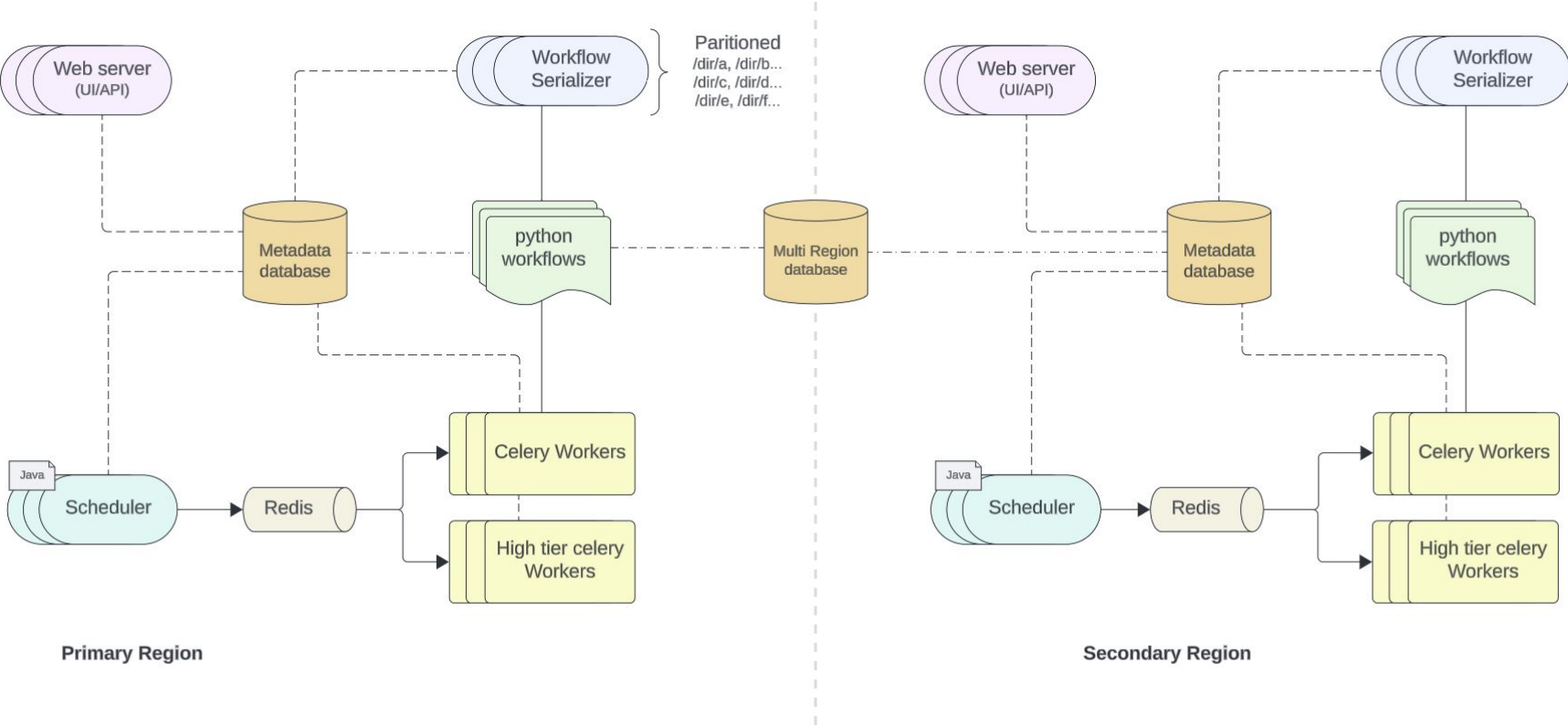
Disaster Recovery

No failover support in case of on-prem region failure


Multi - Executors Serializers Region

- 
- 2016**
Data workflow management at Uber
 - 2017**
Doubling down on Airflow
 - 2019**
Diverging & scaling
 - 2022**
Multi - executors, serializers, region
 - 2024**
Hybrid cloud architecture

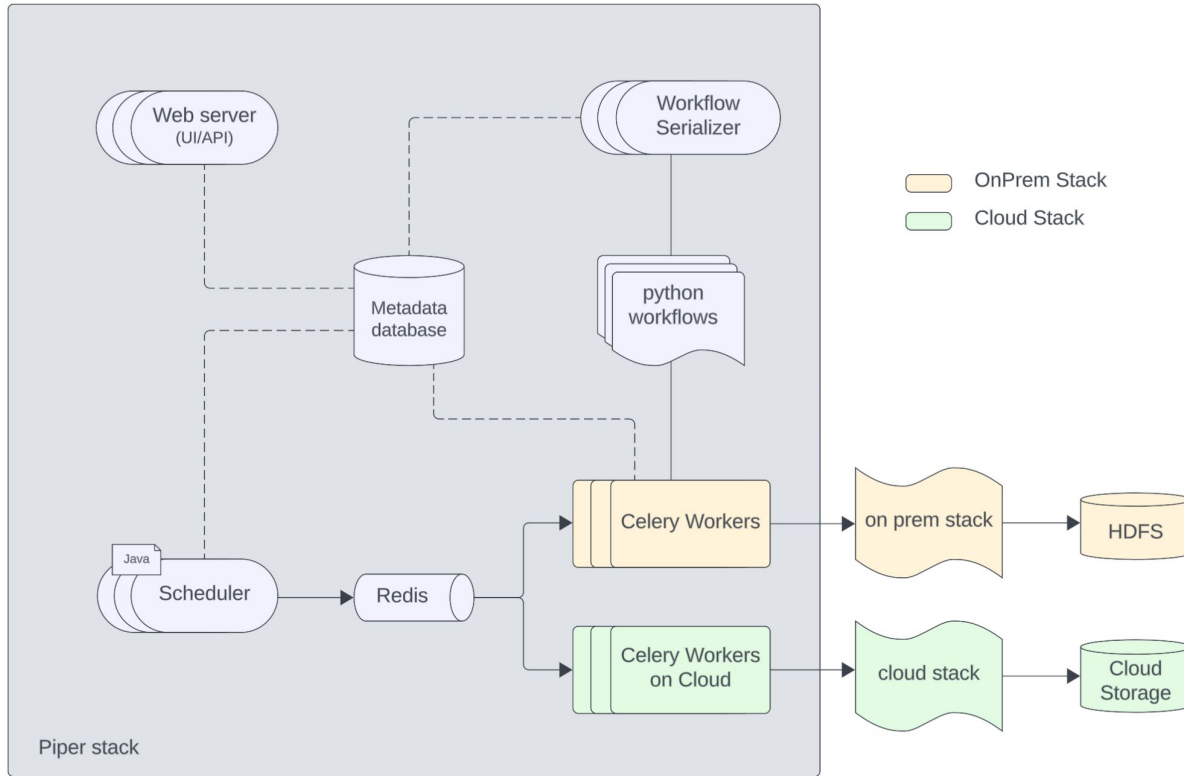
V3 - Partitioned with DR support



Hybrid Cloud Architecture

- 
- 2016**
Data workflow management at Uber
 - 2017**
Doubling down on Airflow
 - 2019**
Diverging & scaling
 - 2022**
Multi - executors, serializers, region
 - 2024**
Hybrid cloud architecture

V4 - Hybrid Cloud architecture

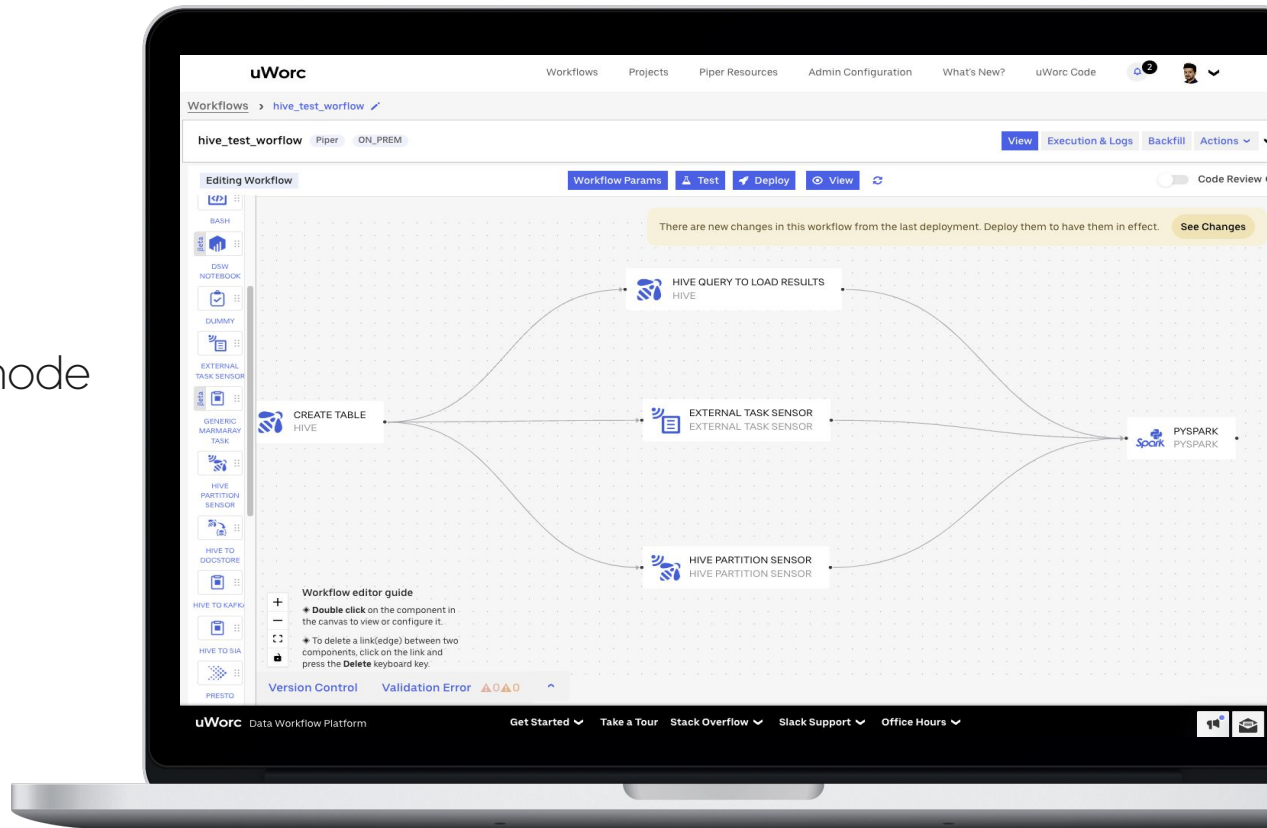


An aerial, top-down view of a city street grid. The streets are filled with cars, and there are several large buildings with flat roofs. The lighting is soft, suggesting late afternoon or early morning. The overall color palette is muted, with greys, blues, and browns.

Piper Ecosystem

uWorc

- Drag and drop UI
- Versioning
- Backfill workflows
- Import and export
- Running workflow in test mode
- Built-in integrations
- Multi engine

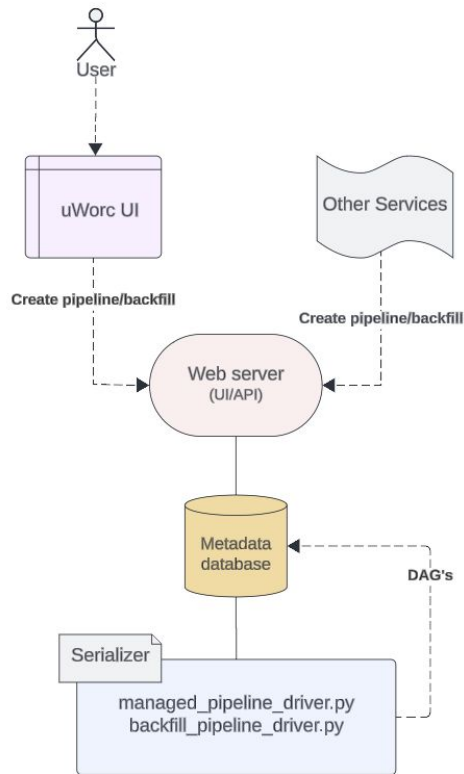


Managed Pipelines

- Fixed contract to create & manage pipelines via API
- Backbone of creating GUI authoring tool (uWorc)
- ~50% of total pipeline in Uber are managed pipelines

Backfill Pipelines

- Piper took diff approach
- Backfills stored in DB and processed by Scheduler
- A new backfill pipeline would be created
- Support executions to a diff pool



JumpStart - Data Aware Scheduling

- Trigger based scheduling
- Enhances scheduling flexibility
- Trigger signals:
 - Data freshness
 - Data completeness
 - Pipeline execution history

```
pipelines:  
  
- pipeline_id: HIVE_ETL_demo  
  dstn_table: demo.sample_hive_table  
  last_task: promote_hive  
  rule: '({{most_latent_src_updated}} or {{seconds_since_execution_end}} >= 14400) and not {{running}}'
```

Workflow Governance

Establishes clear methods, delineate responsibilities, and implement robust processes to standardize and safeguard workflows

Prioritization

- Tier wise segregation of workflows
- Resource management

Reliability

- Alerting policies
- Enforced ownership

Cost Efficiency

- Workflow retention
- Cleanup of inactive pipelines

OneETL Framework

Config-driven ETL framework that supports any source to any sink data loading & transformations

Config directory format for pipelines

```
one_etl/  
  config/  
    ddl/  
      //CREATE TABLE statement  
      <table_1_name>.ddl  
      <table_2_name>.ddl  
    jobs/  
      //Pipeline configuration  
      <table_1_name>.yaml  
      <table_2_name>.yaml  
    sql/  
      //Transformation logic  
      <table_1_name>.sql  
      <table_2_name>.sql
```

Example pipeline yaml

```
owner: alice  
hive_table: schema.table  
spark_opts:  
  queue: spark-adhoc  
ddl_file: <TEAM>/ddl/example_ddl.ddl  
sql_file: <TEAM>/sql/example_sql.sql  
execution_engine: spark  
start_date: '2024-09-15'  
hive_staging_schema: stg_schema  
hive_test_schema: test_schema  
hive_partition_key: datestr
```



OneETL Salient Features

Efficiency

- Supports incremental data load
- Auto generate sample datasets
- Pre-load tests to ensure datasets health
- Post cleanup to delete tmp datasets

Reliability

- Multi stage pipelines to support checkpoints
- Auto publishing lineage for compute freshness
- Regression tests to compare/validate data

Productivity

- No Piper/Python knowledge required
- YAML inheritance for config reusability
- SQL templates for code reusability
- Unit testable

Piper Usage

200k

active pipelines

450k

average daily pipeline runs

750k

average daily task runs

1k

teams using piper

8k

distinct users

3.5k

avg monthly commits to piper-core-pipelines

1000

celery executors

10 mins

serializer avg processing time

1 min

avg task scheduling delay

Piper vNext

Uber Specific

- Disaster recovery with partial failovers*
- Hybrid cloud support*
- Safe deployment of pipelines

Parity with Airflow

- Async operator*
- K8s executors
- Dynamic tasks

Converging to Airflow

- Booting Airflow as an alternative
- Migrating simple use-cases to Airflow
- Contributing back to Airflow



we are same

Piper

you are different

Airflow

Questions?



Shobhit Shah



Sumit Maheshwari

