Multi-tenancy is coming
All you wanted to know about multi-tenancy
... but were afraid to ask
Airflow Survey!

About us

Jarek Potiuk
Independent Open-Source Contributor and Advisor
Airflow Committer & PMC member
Twitter: @jarekpotiuk

Mateusz Henc
Senior Software Engineer
in Google Cloud Composer
Use cases

- I want to run my dags independently from each others
- I want different teams to share Airflow infrastructure
- I want to manage a single Airflow installation
Multi-tenancy today?
Is multi-tenancy possible today?

- It’s just assigning permissions in the UI, right?
  - Nope. UI is just a view on DAGs.
- I can just give access to sub-dirs to separate teams?
  - Good start, but no. Workers are shared between teams.
- But I can have cluster policies to prevent it?
  - Yeah. But tasks have access to the DB and can modify it!
Challenges

• Original Airflow model: “trust-everyone”
• Multiple-teams - multiple Airflow installations
• All dags in single physical location
• Direct database access
• No isolation between dags/tasks and Airflow code
• No fine-grained access to Airflow MetaData DB
• No fine-grained access to Secrets
Way forward
Path to Multi-tenancy

1. Gradual
   Adding feature by feature

2. No breaking changes
   All features opt-in

3. Iterative
   Each step is usable solution

4. Open
   There is room for follow-up ideas
Principles

- Slow introduction - no big-bang
- Feature Flags
- Cooperation with many stakeholders
  - Google, Astronomer, AirBnB, Cloudera, Amazon
- Semi-regular meetings (Recordings, Minutes)
Concepts
Trusted vs untrusted components

**Trusted**
- Airflow community code execution
- No user plugins/pypi-packages
- No access to DAG files
- Direct access to MetaData DB

**Untrusted**
- User code execution
- Possible plugins/pypi-packages
- Access to DAGs storage
- No direct access to MetaData DB
Managed vs Standalone Airflow

**Managed Airflow**

- Service Provider
  - Timetables
  - Triggers
  - Webserver plugins
- Airflow Admin
  - Runtime Plugins
  - Runtime Packages
- DAG authors
  - DAG code

**Standalone Airflow**

- Airflow Admin
  - Timetables
  - Triggers
  - Webserver Plugins
  - Runtime Plugins
  - Runtime Packages
- DAG authors
  - DAG code
Various degrees of multi-tenancy

- Separate Dag Authors and Airflow Admins
- Separate dependencies for different teams
- No direct MetaDB access for DAG authors
- Fine-grained secret access for DAGs/Tasks
- Standalone tasks with all resources needed
Airflow 2.2 architecture

- Dag Processor
- Scheduler
- DB
- Triggerer
- DAGs
- Workers
- Webserver
Single-tenant legacy: Airflow 2.2

Shared infrastructure

Dag Processor

Scheduler

DB

Workers

Triggerer

Webserver

Should be per-tenant
Dag Processor separation
Dag Processor Separation

Scheduler

Dag Processor

Scheduler

Dag Processor
Airflow 2.3: AIP-43 (partial) - scheduler code runtime separation

Single, separate tenant

Dag Processor → DAGs → Workers

Scheduler

DB

Webserver

Code runtime separation

Shared infrastructure

Triggerer
Dag Processor Separation

- **Dag Processor refactoring**
  - Zombie detection moved to Scheduler Job
- **Callbacks**
  - Through database
- **New configuration**
  - `[scheduler]standalone_dag_processor`
- **New CLI command**
  - `airflow dag-processor`
- **AIP-43**
Dag Processor Separation
Future
Runtime isolation
Per team runtime isolation

- Complete AIP-43 + AIP-46 combined
  - Google + AirBnB
- Combine separate dag processors and Docker Runtime
- Docker Runtime allows for environment separation
- Same Docker Runtime shared between Processor and Worker
- Easily configurable per tenant
Airflow 2.?: AIP-43 + AIP-46: Tenant Runtime isolation

Tenant 1
- Dag Processor
- DAGs
- Workers

Tenant 2
- Dag Processor
- DAGs
- Workers

Shared infrastructure
- Scheduler
- DB
- Triggerer
- Webserver
DB Isolation
Airflow Internal API

- No direct access to DB from Workers and Dag Processor
- Only certain operations allowed via Internal API
- Temporary authorization for the time of processing
- No fine-grained Meta Data DB access (yet)
- AIP-44 - Airflow Internal API
Airflow 2.?: AIP-44 - DB access isolation

Tenant 1
- Dag Processor
- DAGs
- Workers

Tenant 2
- Dag Processor
- DAGs
- Workers

Shared infrastructure
- Scheduler
- Webserver
- No direct DB access

Triggerer
- Internal API
- DB
Status of AIP-44

- First pass of reviews passed. On hold due to many changes in 2.3
- RPC-like interface replacing current internal methods
- No duplication of code for local/remote:
  - Internal vs. RPC calls
  - POC in progress
- Authorization:
  - Temporary tokens generated by Scheduler/Processor
Fine-grained access
Fine-grained access to MetaData DB

- No AIP yet - discussions must happen
- Temporary tokens with selective access per task
- Only access resources that are needed
  - DAG/Task/DagRun/XCom
- Still no Secrets isolation separation
Fine-grained access to resources

- No AIP yet - discussions must happen
- Challenges:
  - Mapping DAGs/Tasks to secrets
  - Likely require changing DAG definition
  - Likely require adding Tenant entity in DB
  - Likely we can “embed” credentials in workload
Airflow 2.?: AIP-? - Fine grained access to resources

Tenant 1
- Dag Processor
- Credentials
- DAGs
- Workers
- Credentials

Tenant 2
- Dag Processor
- Credentials
- DAGs
- Workers
- Credentials

- Fine-grained DAG runtime access
- Fine grained task runtime access

Scheduler
Internal API
DB
Webserver

Shared infrastructure
Web Server
per-tenant access
Per Tenant Webserver Access to DAGs

- Possible but not part of Airflow as a product
  - Cloud Composer - custom approach
- No AIP yet - discussions must happen
- Challenges:
  - Mapping DAGs/Tasks to user groups
  - Permission management for task groups
  - Likely require changing DAG definition
  - Likely require adding Tenant entity in DB
Airflow 2.? - AIP-? - Per-tenant Webserver access to DAGs

Tenant 1
- Dag Processor
- Credentials
- DAGs
- Workers
- Credentials

Tenant 2
- Dag Processor
- Credentials
- DAGs
- Workers
- Credentials

Fine-grained DAG runtime access
- Triggerer
- Internal API
- Scheduler
- DB
- Webserver

Fine-grained task runtime access
Airflow 3.0+
Open questions
Airflow 3.0 and beyond

- Transition to isolation mode
- Replacement of DB queries
- Multi-tenancy flag or feature flags?
- Multi-tenancy by default?
- No opting-out?
Thank you!

Q&A