Multitenancy

state of the union

Airflow Summit
Let’s flow together
September 19-21, 2023,
Toronto, Canada
About us

Vincent Beck
Software Engineer, AWS
Airflow committer

Jarek Potiuk
Independent Open Source Contributor and Advisor
Airflow Committer & PMC Member, ASF Member

Mateusz Henc
Senior Software Engineer
in Google Cloud Composer
What is multi-tenancy?
Initial multi-tenancy goals

- Single database
- Shared scheduler and webserver instances
- Allow multiple teams to have isolated environments
- Isolated/different code dependencies
- Authentication integrated with (some) SSO
Who needs multi-tenancy?

- Individual / Small teams?
  - No
- Medium / Big companies
  - Most likely
- Service providers
  - Multi-tenancy per customer
AIP-44

Internal API
AIP-44 Why?
AIP-44 Why?
AIP-44 How?
AIP-44 Status

- New Component - done
- Migration of all methods to Internal API - almost there
- Migration of Operators code - not done
- Running CI tests in DBless mode is - not done
- Regressions and performance tests - not done
AIP-56
Extensible user management
What is AIP-56?

User → Rest API → Web server → CLI

Airflow

Core Airflow
- DAG processors
- Triggers
- Executor
- Database
- Scheduler
- User management
- Workers
What is AIP-56?
What is AIP-56?
What is AIP-56?

User

Rest API

Web server

CLI

FAB auth manager

Auth manager A

Auth manager B

Airflow

Core Airflow

DAG processors

Trigerrers

Executor

Database

Scheduler

User management

Workers
What is an auth manager?

1. User management API
   a. User authentication
   b. User authorization
What is the status of AIP-56?

1. Defining auth manager interface. 90% done.
2. Migrating Airflow code to use auth manager API. 40% done.
3. Implementing FAB auth manager. 60% done.
4. Implementing another auth manager based on external tools (e.g. KeyCloak). Not started.
Where are we now?
Multitenancy as a platform

- Airflow is a platform
- Multi-tenancy is not a (simple) feature to turn on
- Enabling building blocks
  - AIP-43 Dag Processor Separation
  - AIP-44 Internal API
  - AIP-56 Extensible User Management
  - ??? What do we miss ??
Proposed next steps
What do we miss?

- Documenting how to approach it
  - Absolutely
- Per tenant access to resources?
  - Yes
- Per tenant Python / System dependencies
  - Certainly
- Per task authorization to API calls?
  - Probably not
Kill all birds with single stone?

Queues

Queue == Tenant
Queues (alias tenant) in DAG File Processor

● One DAG subfolder = one queue = one tenant

● Per-tenant standalone processor

● `airflow dag-processor --tenant`

● **Force** queue = tenant

● Per-tenant environment (per-tenant image)
Queues (alias tenant) in Triggerer

- Per tenant triggerer
- `airflow triggerer --tenant`
- Pick only deferred tasks with `queue = tenant`
- Per-tenant environment (same per-tenant image)
Queues (alias tenant) in Workers (k8s Pods)

- Per tenant worker
- `airflow celery worker --tenant` (alias for `--queue`)
- Pick only tasks with `queue = tenant`
- Per-tenant environment (same per-tenant image)
Queues (alias tenant) in internal-api server

- Per tenant internal-api
- `airflow internal-api --tenant`
- Only accepts calls for DAGs with `queue = tenant`
- Per-tenant environment (same per-tenant image)
- Per tenant connections and variables (new ‘tenant’ field for DB and secret backends)
Tenants in webservice

- AIP-56 to the rescue
- Auth Manager with queue = tenant access mapping
AIP-58 ?
Multi-tenancy is (almost) here
Q&A

Multitenancy: state of the union