Airflow RBAC proposal

- Introduction
- Design
  - View Level Access Control (VLAC)
    - Database Diagram
    - User-Group Mapping
    - User-Role and Group-Role Mapping
    - Role-Permission Mapping
  - Dag Level Access Control (DLAC)
    - DAG Roles
    - DAG Permissions
    - DAG Role Permission Mapping
    - Access-Control attribute
    - Database Diagram
  - Role Enforcement
- Implementation Plan
  - Phase 1
    - Goal
    - Requirements
  - Phase 2
    - Goal
    - Requirements
- Open Questions

Introduction

Presently, Airflow doesn't have a built-in Role Based Access Control (RBAC) capability. It does provide very limited authorization capability by providing admin, data_profiler, and user roles. However, associating these roles to authenticated identities is not a simple effort. The current proposal aims at resolving this issue by building RBAC into Airflow and simplifying access management via the Airflow UI.

Design

I propose implementing Role based Access Control within Airflow at two levels:

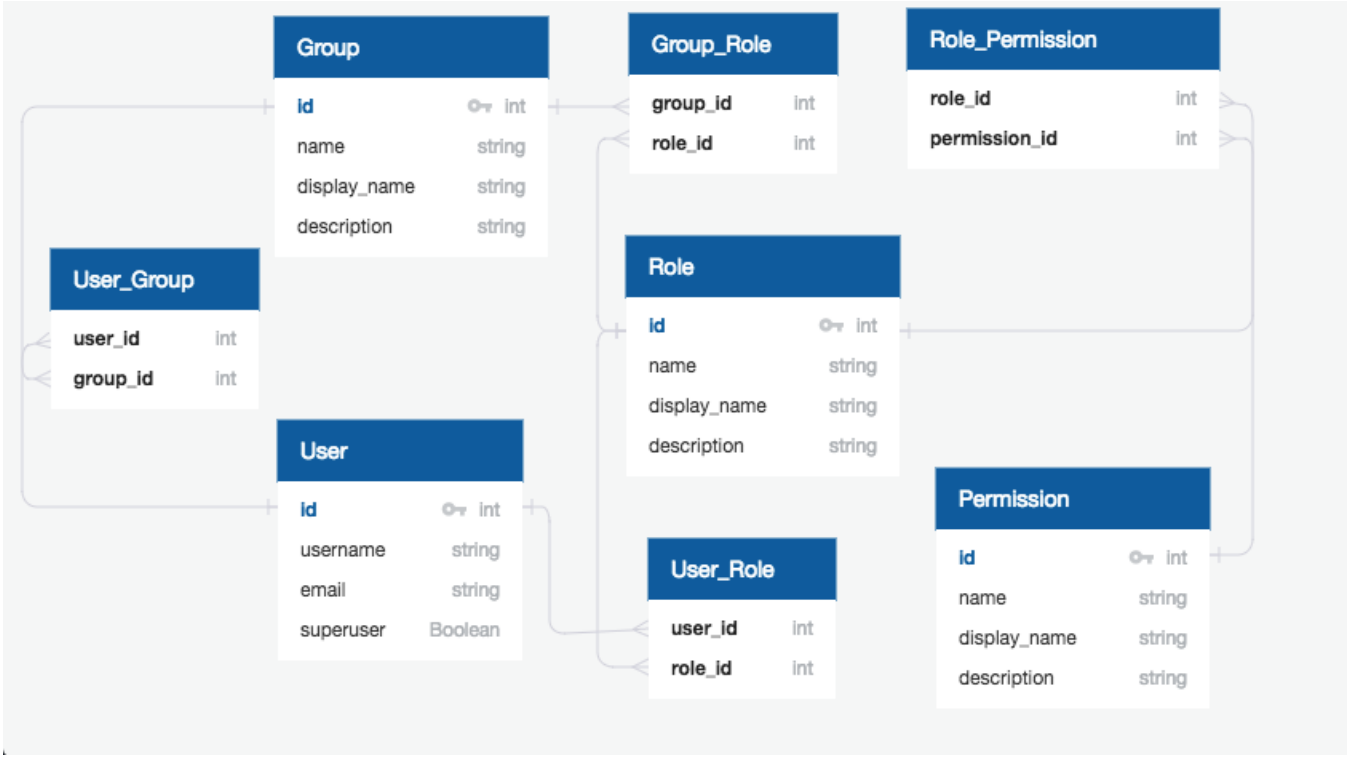
1. View Level
2. Dag Level

View Level Access Control (VLAC)

Airflow UI exposes a number of views grouped under various categories like admin, data profiling, etc. To implement VLAC, we propose switching to following five roles

No.	Role	Description
1	Administrator	Has access to all the functions and views inside airflow
2	Ops	Has access to all views except User view
3	Data Profiler	Only has access to the views under Data Profiling category.
4	User	This role corresponds to regular user and will have access to all non-admin and non-data profiling views.
5	Read_Only	Only has read-only access to all the non-admin and non-data profiling Airflow views.

Database Diagram



User-Group Mapping

User-Group mappings can be handled by Administrator within Airflow UI when the Authentication backend doesn't provide the Group info for the authenticated user. If the Authentication backend does provide the group information, it will be recorded by Airflow on User login and will be written to the backend database. This information will be persisted until the user's next successful login when it will be rewritten. This is done to account for any modifications made to user-group mapping within the authentication backed since the user's last login to Airflow.

User-Role and Group-Role Mapping

User-Role and Group-Role mappings will be handled inside Airflow. Only users with Administrator role will be able to manage these mappings.

Role-Permission Mapping

Role and Permissions are very specific to Airflow application and their values will be used as decorators within Airflow source code. Hence, none of the roles will be allowed to modify Role, Permission models. Even the Role-Permission mapping won't be editable from the Airflow UI.

Dag Level Access Control (DLAC)

As Dags are not created using Airflow UI, the latter lacks information to make decisions pertaining to DAG level access. Hence, information pertaining to DAG level Access needs to be supplied in the DAG file that will be read by Airflow when the DAG is imported.

DAG Roles

To implement DLAC, I propose following DAG level roles.

No.	Role	Description
1	DAG_Viewer	Can only view the DAG
2	DAG_Editor	Can only view, edit, and execute the DAG
3	DAG_Executor	Can only view and execute the DAG

DAG Permissions

No.	Permission	Description
1	READ_DAG	Allows the user to view the DAG and the associated attributes.
2	WRITE_DAG	Allows the user to clear or change the task/DAG status.
3	EXECUTE_DAG	Allows the user to run the DAG
4	REFRESH_DAG	Allows the user to refresh the DAG

DAG Role Permission Mapping

No.	Role	Permission(s)
1	DAG_Viewer	READ_DAG
2	DAG_Editor	READ_DAG, WRITE_DAG, EXECUTE_DAG, REFRESH_DAG
3	DAG_Executor	READ_DAG, EXECUTE_DAG

Access-Control attribute

Presently, DAG can specify the owner using the ‘owner’ attribute. However, the ‘owner’ attribute doesn’t necessarily denote the creator(s) of the DAG. The airflow documentation recommends the owner attribute to be the unix username under which that DAG needs to run. At this time, we do not have the data to know how many Airflow deployments deviate from this recommendation.

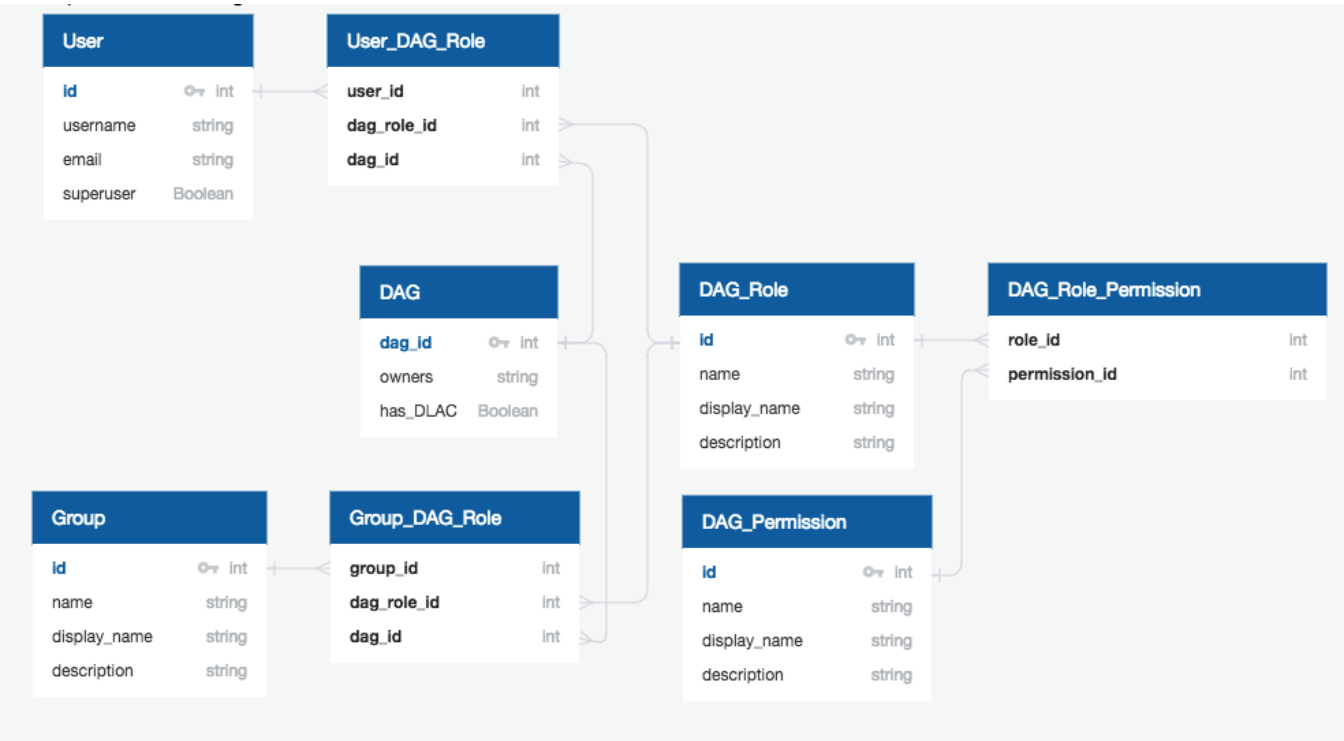
I wanted to ensure that users are able to adopt DLAC smoothly and standardizing the use of owner attribute will be a breaking change. Hence, I’ve decided not to use the owner attribute in DLAC.

Instead, I propose addition of a new DAG attribute called “access_control” with following syntax:

access_control={‘DAG_level_role’:{‘groups’:[list_of_group_names],’users’:[list_of_user_names]}}

This attribute allows the DAG to declare association between DAG roles and users/groups. This will be an optional attribute. Absence of this attribute would mean the DAG doesn’t want to declare any DAG level access control.

Database Diagram



Role Enforcement

For every logged-in user, VLAC is enforced before DLAC. For certain View Level Roles, DLAC is ignored.

Following table shows whether DLAC is ignored/honored depending on user's View-level role.

No.	View-level Role	DAG Level Roles(Ignored/Honored)
1	Administrator	Ignored
2	Ops	Ignored
3	Data Profiler	Not Applicable
4	User	Honored
5	Read_Only	DAG_Viewer Role is enforced for all the DAGs

Implementation Plan

The proposal will be implemented in two phases

Phase 1

Goal

Limit access to views within Airflow UI based on view-level access control (VLAC)

Requirements

1. Add a User Management menu under Admin Category
2. Models to be created:
    a. Group

    b. Role
    c. Permission
3. Adding necessary Flask-Principal decorators to the views

Phase 2

Goal

Limit access to specific DAGs based on DAG-level roles (DLAC)

Requirements

1. Models to be created:
    a. DAG_Role
    b. DAG_Permission
2. Models to be modified:
    a. DAG: has_DLAC column needs to be added
3. Addition of new “access_control” attribute to the DAG.

Open Questions

1. Should DAG_Executor role be allowed to refresh the DAG?