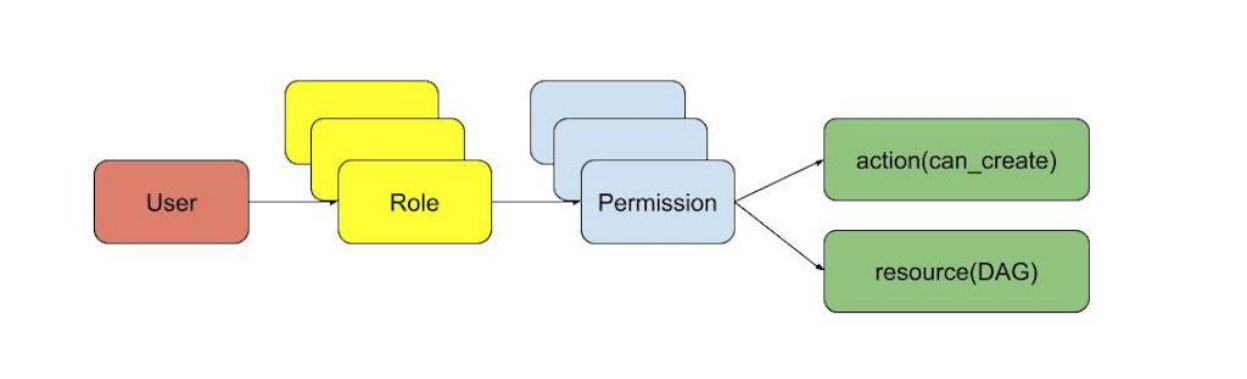
Airflow Security

Airflow has built the role based access control (rbac) of the web console on top of Flask AppBuilder(F.A.B). A user can have multiple roles and different users can share the same role. Each role can have multiple permissions.

Each permission allows the user an action against a resource



This security model gives us a lot of flexibility to manage user permissions. For example, we can allow a user being able to clear/rerun a task, but not being able to configure connections and pools.

Airflow provided 5 default roles with predefined permissions - Admin, Public, User, Public, Viewer ad Op. Airflow allows Admin to create Customer Roles allowing you to put the permissions you want under the role.

Admin

Admin users have all possible permissions, including granting or revoking permissions from other users.

Public

Public users (anonymous) don't have any permissions.

Viewer

Viewer users have limited viewer permissions

User

User users have Viewer permissions plus additional user permissions

Op

Op users have User permissions plus additional op permissions

Airflow also allows you to have DAG level roles. So you can only allow project owners and Admin to edit the DAGs under a project.

DAG-level permissions

For DAG-level permissions exclusively, access can be controlled at the level of all DAGs or individual DAG objects.

This includes DAGs.can_create, DAGs.can_read, DAGs.can_edit and DAGs.can_delete. When these permissions are listed, access is granted to users who either have the listed permission or the same permission for the specific DAG being acted upon

. For individual DAGs, the resource name is DAG: + the DAG ID.

Airflow auth is built on top of F.A.B which supports the 5 auth types below. More details about auth in F.A.B can be found here.

- Database: username and password style that is queried from the database to match. Passwords are kept hashed on the database.
- OAuth: Authentication using OAUTH (v1 or v2). You need to install authlib.
- OpenID: Uses the user's email field to authenticate on Gmail, Yahoo etc...

- LDAP: Authentication against an LDAP server, like Microsoft Active Directory.
- REMOTE_USER: Reads the REMOTE_USER web server environ var, and verifies if it’s authorized with the framework users table.
- It’s the web server's responsibility to authenticate the user, useful for intranet sites, when the server (Apache, Nginx) is configured to use kerberos, no need for the user to login with username and password on F.A.B.