Content

Manage Airflow environment with AWS auth manager

When the AWS auth manager is used, all users and their permissions are no longer managed by Airflow itself but by AWS through two different services: AWS IAM Identity Center (users) and Amazon Verified Permissions (permissions).

Manage users through AWS IAM Identity Center

Users

All users having access to the Airflow environment must be defined in AWS IAM Identity Center. You can use AWS IAM Identity center as identity source or as a proxy between the identity source (e.g. Active Directory) and the Airflow environment. See documentation for more details on how to manage your identity source.

You can see the list of users defined in AWS IAM Identity Center by following these steps.

1. Open the IAM Identity Center console.
2. Choose **Users**.

Groups

You can use groups in AWS IAM Identity center to group users in logical entities (e.g. by team, by department, …). Later, these groups can be used in Amazon Verified Permissions to assign permissions to a group of users. See documentation to add users to groups.

You can see the list of groups defined in AWS IAM Identity Center by following these steps.

1. Open the IAM Identity Center console.
2. Choose **Groups**.

Assign users and groups to the Airflow environment

! Note

All users and groups defined in AWS IAM Identity Center do not have automatically access to the Airflow environment. You need to manually assign which user can access to Airflow.

To assign users and groups to Airflow, please follow the steps below.

1. Open the IAM Identity Center console.

! Note

If you manage users in AWS Managed Microsoft AD, make sure that the IAM Identity Center console is using the AWS Region where your AWS Managed Microsoft AD directory is located before taking the next step.

2. Choose **Applications**.
3. Choose the **Customer managed** tab.
4. In the list of applications, choose the application name `Airflow`.
5. On the application details page, in the **Assigned users and groups** section, choose **Assign users and groups**.
6. In the **Assign users or groups** page, select the different users and groups you want to assign to Airflow. You can also search users and groups. You can specify multiple users or groups by selecting the applicable accounts as they appear in search results.

7. Choose **Assign users**.

Manage user permissions through Amazon Verified Permissions

AWS auth manager uses Amazon Verified Permissions to define and verify user permissions. It uses Cedar language to define fine-grained permissions for users. AWS auth manager uses one policy store to store all policies related to the Airflow environment. To manage these policies, please follow the steps below.

1. Open the Amazon Verified Permissions console.
2. Choose the policy store used by Airflow (by default its description is `Airflow`).
3. In the navigation pane on the left, choose **Policies**.
4. On the **Policies** page, you can see the list of policies. To create a new policy, please follow steps below.

    1. Choose **Create policy**.
    2. Under **Create policy**, choose **Create static policy**.

Policy format

Policies are defined using Cedar language in Amazon Verified Permissions. For a complete documentation, see cedar language website. In cedar language, a policy is composed of three elements:

- **Principal**. Who is making the request?
- **Action**. What operation does the principal want to perform?
- **Resource**. What does the principal want to perform the action on?

Each of these three elements can have limited values in the context of the Airflow environment. You can see the list of principals, actions and resources in the policy store schema by following the steps below.

1. Open the Amazon Verified Permissions console.
2. Choose the policy store used by Airflow (by default its description is `Airflow`).
3. In the navigation pane on the left, choose **Schema**.

Example of policies

Here are some example of policies you can define in Amazon Verified Permissions. You can use them as-is if they fit exactly your use case. You can also modify and/or combine them to create your owned tailor made policies.

Give all permissions to specific user

```
permit(
  principal == Airflow::User::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action,
  resource
);
```

! Note

You must reference the user by its user ID and not its username or any other attribute. You can find the user ID in AWS IAM Identity Center.

Give all permissions to a group of users

```
permit(
  principal in Airflow::Group::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action,
  resource
);
```

! Note

You must reference the group by its group ID and not its name. You can find the group ID in AWS IAM Identity Center.

Give read-only permissions to a group of users

```
permit(
  principal in Airflow::Group::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action in [
    Airflow::Action::"Configuration.GET",
    Airflow::Action::"Connection.GET",
    Airflow::Action::"Custom.GET",
    Airflow::Action::"Dag.PUT",
    Airflow::Action::"Dag.GET",
    Airflow::Action::"Menu.MENU",
    Airflow::Action::"Pool.GET",
    Airflow::Action::"Variable.GET",
    Airflow::Action::"Dataset.GET",
    Airflow::Action::"View.GET"
  ],
  resource
);
```

Give DAG specific permissions to a group of users

The policy below gives all DAG related permissions of the DAG `test` to a group of users.

```
permit(
  principal in Airflow::Group::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action,
  resource == Airflow::Dag::"test"
);
```

The policy below gives all DAG related permissions of the DAGs `financial-1` and `financial-2` to a group of users.

```
permit(
  principal in Airflow::Group::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action,
  resource in [Airflow::Dag::"financial-1", Airflow::Dag::"financial-2"]
);
```

The policy below gives access to logs of the DAG `test` to a group of users.

```
permit(
  principal in Airflow::Group::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action,
  resource == Airflow::Dag::"test"
) when {
  context has dag_entity && context.dag_entity == "TASK_LOGS"
};
```

Forbid specific action to specific user

All policies defined in Amazon Verified Permissions are taken into account when doing an authorization check. For example, if both one **permit** and one **forbid** policies match the request, the access will be denied to the user. This can be useful if, for example, you want to restrict access to a specific user who belongs to a group that is granted all permissions.

The policy below removes access of DAGs `secret-dag-1` and `secret-dag-2` from a specific user.

```
forbid(
  principal == Airflow::User::"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
  action,
  resource in [Airflow::Dag::"secret-dag-1", Airflow::Dag::"secret-dag-2"]
);
```

Previous

Next

Join community

License
Donate
Thanks
Security