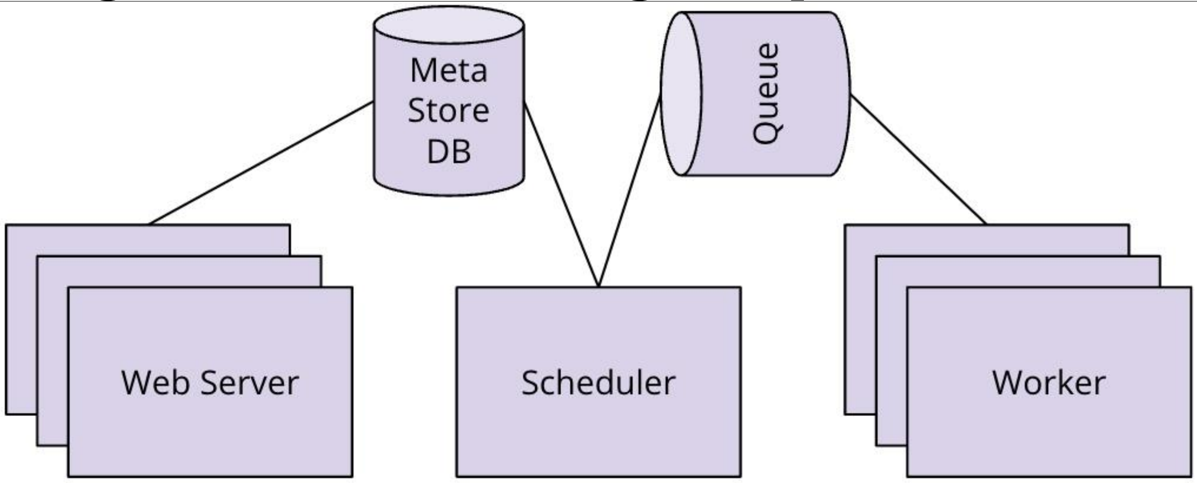
PostgresHook and PostgresOperator



Metastore Backend in Context

MetastoreBackend

The MetastoreBackend python class connects to the **Airflow Metastore Backend** to retrieve credentials and other data needed to connect to outside systems.

The below code creates an aws_connection object:

- aws_connection.login contains the **Access Key ID**
- aws_connection.password contains the **Secret Access Key**

MetastoreBackend Usage ``` from airflow.decorators import dag from airflow.secrets.metastore import MetastoreBackend

```
@dag(
    start_date=pendulum.now()
)
def load_data_to_redshift_dag():

    @task
    def copy_task():
        metastoreBackend = MetastoreBackend()
```

```
aws_connection=metastoreBackend.get_connection("aws_credentials")
        logging.info(vars(aws_connection))

```
```

Logging Output ``` [2022-08-11, 16:16:20 UTC] {l2_e4_s3_to_redshift copy.py:21} INFO - {'_sa_instance_state': <sqlalchemy.orm.state.InstanceState object at 0x7f4342ca2e10>, 'id': 1, 'conn_type': 'aws', 'login': 'AKIA4QE4NTH3R7EBEANN', 'conn_id': 'aws_credentials', '_password': '***'}

```
```

Using PostgresHook

The `PostgresHook` class is a superclass of the Airflow `DbApiHook`. When you instantiate the class, it creates an object that contains all the connection details for the Postgres database. It retrieves the details from the Postgres connection you created earlier in the Airflow UI.

Just pass the connection id that you created in the Airflow UI.

```
from airflow.providers.postgres.operators.postgres import
PostgresOperator
. . .
        redshift_hook = PostgresHook("redshift")
```

Call `.run()` on the returned `PostgresHook`object to execute SQL statements.

```
       redhisft_hook.run("SELECT * FROM trips")
```

Using PostgresOperator

The `PostgresOperator` class executes sql, and accepts the following parameters:

- a `postgres_conn_id`
- a `task_id`
- the `sql` statement
- optionally a `dag`

```
from airflow.hooks.postgres_hook import PostgresHook
from airflow.decorators import dag

@dag(
```
```

```
    start_date=pendulum.now()
)
def load_data_to_redshift_dag():


. . .
    create_table_task=PostgresOperator(
        task_id="create_table",
        postgres_conn_id="redshift",
        sql=sql_statements.CREATE_TRIPS_TABLE_SQL
    )
. . .
    create_table_task >> copy_data
```

Tip:

Notice the `PostgresOperator` **doesn't** have a `dag` parameter in the above example. That is because we used the `@dag` decorator on the dag function:

PreviousNext
Send Page Feedback