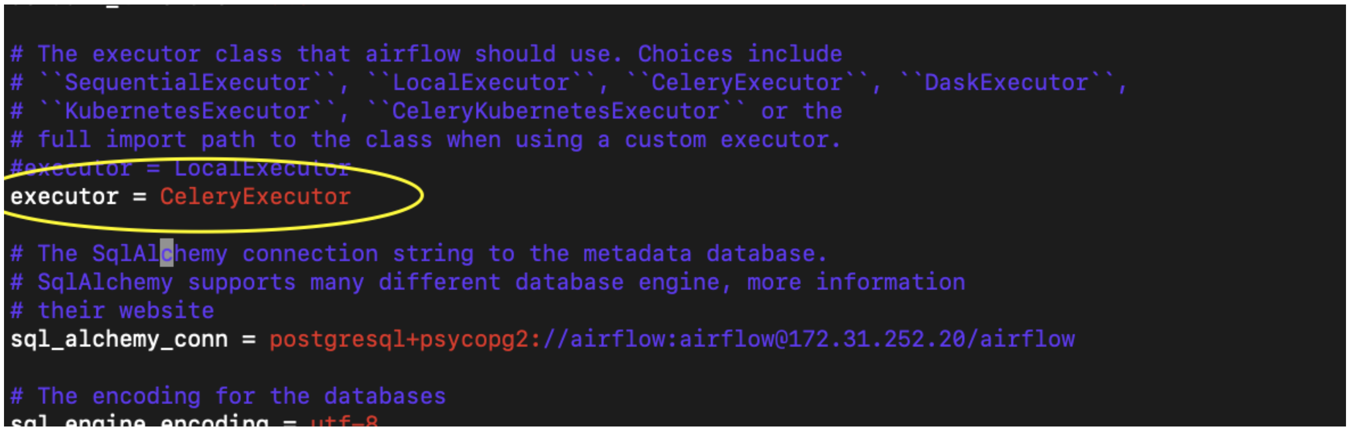
Enabling CeleryExecutor in Airflow

Scheduler VM:

- Setup Redis with password enabled

```
sudo docker run -d   -e REDIS_PASSWORD=123   -p 6379:6379   --name redis   --restart always   redis:6.2
/bin/sh -c 'redis-server --appendonly yes --requirepass ${REDIS_PASSWORD}'
```

- Install `redis` and `celery` pip packages in python3 virtualenv
- Set airflow executor as `CeleryExecutor` and `sql_alchemy_conn` in `airflow.cfg` file



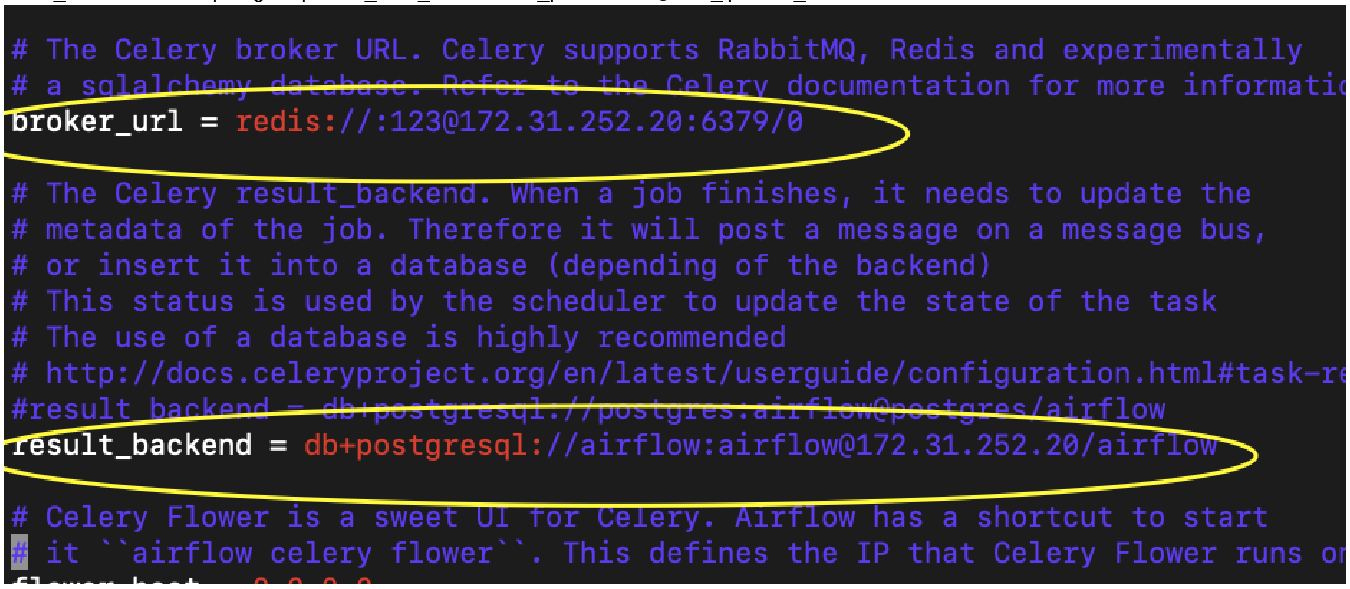
```
# The executor class that airflow should use. Choices include
# ``SequentialExecutor``, ``LocalExecutor``, ``CeleryExecutor``, ``DaskExecutor``,
# ``KubernetesExecutor``, ``CeleryKubernetesExecutor`` or the
# full import path to the class when using a custom executor.
#executor = LocalExecutor
executor = CeleryExecutor

# The SqlAlchemy connection string to the metadata database.
# SqlAlchemy supports many different database engine, more information
# their website
sql_alchemy_conn = postgresql+psycopg2://airflow:airflow@172.31.252.20/airflow

# The encoding for the databases
sql_engine_encoding = utf-8
```

- Set `broker_url` to connect with Redis and set `result_backend` to connect with Postgresdb

broker_url = redis://:<redis_password>@<redis_vm_ip>:<redis_port>/<db_number>
result_backend = db+postgresql://<db_user_name>:<db_password>@<db_ip>/<db_name>



```
# The Celery broker URL. Celery supports RabbitMQ, Redis and experimentally
# a sqlalchemy database. Refer to the Celery documentation for more informatic
broker_url = redis://:123@172.31.252.20:6379/0

# The Celery result_backend. When a job finishes, it needs to update the
# metadata of the job. Therefore it will post a message on a message bus,
# or insert it into a database (depending of the backend)
# This status is used by the scheduler to update the state of the task
# The use of a database is highly recommended
# http://docs.celeryproject.org/en/latest/userguide/configuration.html#task-re
#result_backend = db+postgresql://postgres:airflow@postgres/airflow
result_backend = db+postgresql://airflow:airflow@172.31.252.20/airflow

# Celery Flower is a sweet UI for Celery. Airflow has a shortcut to start
# it ``airflow celery flower``. This defines the IP that Celery Flower runs or
```

- Start Scheduler and Webserver Service
- Start Flower service(optional)

```
airflow celery flower > flower_log.log 2>&1 &
```

Worker VM:

- Create New VM to setup Airflow Worker
- Install Python3, Airflow packages in worker node with same version as in Airflow Scheduler VM, also maintain same folder structure as in Scheduler VM
- Create SSH Keys for user `airflow` in worker node using `ssh-keygen` tool, add the generated public-key in `/home/airflow/.ssh /authorized_keys` file in user `airflow` in Scheduler VM
- Install rsyncd service in worker node

```
sudo yum install -y rsync
sudo systemctl start rsyncd
sudo systemctl enable rsyncd
```

- Need to create `/etc/sysconfig/airflow` file and add configuration as same as other Worker-node
- `/etc/sysconfig/airflow`

```
HADOOP_CONF_DIR=/etc/hadoop/conf
SPARK_DIST_CLASSPATH=$(hadoop classpath)
JAVA_HOME=/usr/java/jdk1.8.0_201-amd64/
SPARK_HOME=/opt/spark
AIRFLOW_HOME=/home/airflow/airflow
AIRFLOW_CONFIG=/home/airflow/airflow/airflow.cfg
PATH=/opt/python3_venv/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr
/sbin:/home/airflow/.local/bin:/home/airflow/bin
```

- Create systemd script for airflow worker service , which contains a commands to start worker with queue name and also to rsync the required folders from Scheduler node
  `/etc/systemd/system/airflow-worker.service`

```
[Unit]
Description=Airflow Worker daemon

[Service]
User=airflow
Group=airflow
Type=simple
EnvironmentFile=/etc/sysconfig/airflow

ExecStartPre=/usr/bin/rsync  -azrv --delete <airflow_scheduler_vm_ip>:
/home/airflow/airflow/ /home/airflow/airflow/
ExecStartPre=/usr/bin/rsync  -azrv --delete <airflow_scheduler_vm_ip>:
/opt/python3_venv/  /opt/python3_venv/
ExecStartPre= /usr/bin/rsync  -azrv --delete
<airflow_scheduler_vm_ip>:/opt/airflow/ /opt/airflow/

ExecStart=/opt/python3_venv/bin/python  /opt/python3_venv/bin/airflow
celery worker -q worker_q2
Restart=on-failure
RestartSec=5s
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

- All files which we are syncing should be in airflow user
- Start Airflow worker service

```
sudo systemctl start airflow-worker
sudo systemctl enable airflow-worker
```

- Set crontab in user `airflow` to rsync the airflow scheduler folders periodically.

```
*/3 * * * * /usr/bin/rsync  -azrv --delete <airflow_scheduler_vm_ip>:
/home/airflow/airflow/ /home/airflow/airflow/
*/3 * * * * /usr/bin/rsync  -azrv --delete <airflow_scheduler_vm_ip>:
/opt/python3_venv/  /opt/python3_venv/
*/3 * * * * /usr/bin/rsync  -azrv --delete <airflow_scheduler_vm_ip>:
/opt/airflow/ /opt/airflow/
```