**Quick answer:**

Use [this docker setup](https://github.com/puckel/docker-airflow).

1

git clone https://github.com/puckel/docker-airflow.git

1

cd docker-airflow

Be sure to set shared volumes to your local DAG & plugins folders and persist psql data in the celery .yml file.

1

docker-compose -f docker-compose-CeleryExecutor.yml up -d

To restart airflow:

1

docker-compose -f docker-compose-CeleryExecutor.yml restart

or

1

docker-compose -f docker-compose-CeleryExecutor.yml down

2

3

docker-compose -f docker-compose-CeleryExecutor.yml up -d

Delete your DAG runs and all data (helpful for debugging and re-running DAG on backfills):

1

docker-compose -f docker-compose-CeleryExecutor.yml run --rm webserver airflow delete\_dag -y sparkify\_etl

Where sparkify\_etl is the DAG name.

**Details:**

I recommend one of two options: using this docker container (<https://github.com/puckel/docker-airflow>) or use GCP's Cloud Composer. I don't recommend a local install directly for reasons stated at the end. Cloud Composer will cost something like $5-$20 to complete the project (and it can be a little slow waiting for it to startup and install python packages), but it works well. With GCP you should get $300 of credit for signing up with a new account, too.

You can see more on how to run these things here: <https://github.com/nateGeorge/airflow_udacity_project#running-airflow>

With the docker container, first clone the github repo. Then update the .yml file you want to use (celery or localexecutor, they both do parallel work) to set the shared folders (volumes) so that it references the dags and plugins folders for the project. You also want to uncomment the part in the celery yml file that will persist data like your redshift and aws creds if you don't want to enter those repeatedly. You also need to add a requirements.txt file with boto3 at least. Then you can run the command:

docker-compose -f docker-compose-CeleryExecutor.yml up -d

To shut it down:

docker-compose -f docker-compose-CeleryExecutor.yml down

To run a CLI command:

docker-compose -f docker-compose-CeleryExecutor.yml run --rm webserver airflow list\_dags

If you add a new file, the current airflow.cfg file in the repo is set to check for new files every 5 minutes (I recommend changing it to something like 5s or 1s), so it's easier to restart the docker cluster:

docker-compose -f docker-compose-CeleryExecutor.yml restart

For GCP, create an account, and head to Cloud Composer (under big data in the menu I think). Then create an instance with mostly default settings (I use the Iowa/central region because I think that's cheapest), then it's fairly intuitive to figure out. There's a webui link, a link to the google bucket where you drop DAG and plugin files, and once you click on the instance name, there's a pypi tab where you can install python packages. Note it's pretty slow (5+ minutes to start up and 5+ minutes to install a python package). The plugins use boto3 so you have to install that.

Running an airflow CLI command on GCP is like:

1

gcloud composer environments run udacity-p4 --location us-central1 delete\_dag -- -y sparkify\_etl

You can also install airflow directly as Mike suggested, but I recommend against that. Installing directly works well \*if\* it works with no problems. If you run into problems like I did, prepare for lengthy and frustrating debugging. I ended up giving up on the local install because it was too much work, especially compared with docker and GCP. And with docker/GCP, you remove any dependence on your local environment, so your work and airflow installation is reproducible by others, unlike a local install (none of the "but it works on my machine" issues). It's also more work with a local install if you want to use a parallel executor like local or celery (which you do, because that's how it's set up in the workspace -- localexecutor). Parallel execution runs faster than sequentialexeuctor.

If you don't have the proper version of Windows to run docker, you might search google shopping for windows 10. I think you need Win 10 pro for it to work properly. Personally, I dual-boot Ubuntu 18.LTS and Windows and almost always use Ubuntu.