

# Deploy to Astro

It is time! Time to deploy the DAG you've built during this course on Astro!

Without further ado, let's go!

## Prerequisites

The DAG `extract_stars.py` under the folder  `dags`  should look like that:

```
from airflow import DAG
from airflow.operators.bash import BashOperator
from airflow.providers.http.operators.http import SimpleHttpOperator
from airflow.operators.python import PythonOperator

import json
from datetime import datetime

def _print_stargazers(github_stats: str, date: str):
    github_stats_json = json.loads(github_stats)
    airflow_stars = github_stats_json.get("stargazers_count")
    print(f"As of {date}, Apache Airflow has {airflow_stars} stars on Github!")

with DAG('extract_stars', schedule_interval='@daily', start_date=datetime(2022, 1, 1), catchup=False) as dag:

    get_date = BashOperator(
        task_id="get_date",
        bash_command="echo {{ ds.format('yyyy') }}"
    )

    query_github_stats = SimpleHttpOperator(
        task_id="query_github_stats",
        endpoint="{{ var.value.endpoint }}",
        method="GET",
        http_conn_id="github_api",
        log_response=True
    )

    print_stargazers = PythonOperator(
        task_id="print_stars",
        python_callable=_print_stargazers,
        op_args=["{{ ti.xcom_pull(task_ids='query_github_stats') }}", "{{ ti.xcom_pull(task_ids='get_date') }}"]
    )
```

```
get_date >> query_github_stats >> print_stargazers
```

With the Connection:

- Connection Id: `github_api`
- Connection Type: `HTTP`
- Host: `api.github.com`
- Schema: `https`

And the Variable:

- Key: `endpoint`
- Val: `repos/apache/airflow`

## Authenticate to Astro

Once you've tested your DAGs locally with `astro dev parse`

Authenticate to the Astro CLI by running `astro login`

```
~/astro ➤ astro login
Welcome to the Astro CLI 🚀

To learn more about Astro, go to https://docs.astronomer.io

Logging in with saved user marc@astronomer.io

Press Enter to open the browser to log in or ^C to quit...
```

After running this command, you will be prompted to open your web browser and log in to the Cloud UI. Once you complete this login, you will be automatically authenticated to the CLI.



Congratulations, you're all set!  
Your device is now connected.

## Add Variables and Connections to Astro

---

Before deploying your DAG, you need to recreate the connections/variables that you've created locally on Astro.

Got to your Astro deployment `dev`

Click `Add Variables`

Deployments > dev Open Airflow Logs

**HEALTHY** **NAMESPACE** spherical-protogalaxy-3131 **CLUSTER** FE - Development - US West 2 **ASTRO RUNTIME** 5.0.7 (based on Airflow v2.3.3) **DOCKER IMAGE** deploy-2022-08-22T11-26 **UPDATED** 28 minutes ago **CREATED** an hour ago

Worker queues are now available as the newest feature in Astro. They enable a simpler way to size workers and allow you to create optimized execution environments for different tasks. To learn more, see [Astro documentation](#).

5 DAGs: 0 of 0 runs failed    Tasks: 0 of 0 tasks failed    Worker CPU: 18% max of 2 CPUs    Worker Memory: 12% max of 7.5GiB

[Edit Details](#)

**WORKER QUEUES** [Add Worker Queue](#)

NAME	WORKER TYPE	MAX TASKS PER WORKER	WORKER COUNT (MIN-MAX)	
default	m5.xlarge (2 CPU, 14.535Gi MEM)	16	1-10	<a href="#">Edit</a>

**SCHEDULER SETTINGS**

**RESOURCES** **SCHEDULER COUNT** [Edit](#)

5 AU (0.5 CPUs, 1.88 GiB MEM) 1

Environment Variables (0) [Add Variables](#)

API Keys (0) [Add API Key](#)

Alert Emails (0)

We are going to create two environment variables. One for the connection, one for the variable.

For the connection:

KEY: `AIRFLOW_CONN_GITHUB_API`

VALUE: `http://api.github.com/https`

Click `Add`

For the variable:

KEY: `AIRFLOW_VAR_ENDPOINT`

VALUE: `repos/apache/airflow`

Update Environment Variables

KEY

VALUE

AIRFLOW\_CONN\_GITHUB\_API

http://api.github.com/ht  
tps

Secret?

AIRFLOW\_VAR\_ENDPOINT

repos/apache/airflow

Secret?

KEY

value

Secret?

+

Add

Save Variables

Cancel

Click `Save Variables`

## Deploy to Astro

Back to the Astro CLI, you can list workspaces with `astro workspaces list`

NAME	ID
Dogfood Production	g4elywerg190081hzgdc08cei
Onboarding - Customer Success	342dg4ely190081hzgdc08cei
Education	cl48g4edfess234422dwc08cei
Data Engineering	cl74nlx32434324fyoa6vtvj86

Pick your workspace with `astro workspace switch <workspace-id>`

Next, run `astro deploy` and pick your deployment

```

Authenticated to Astro

Current Workspace: Data Engineering

Select which Deployment you want to deploy to:
#      DEPLOYMENT NAME      RELEASE NAME      DEPLOYMENT ID
1      dev                  spherical-protogalaxy-3131  cl74npow0485901j0ikhpsgf4r

> 1
Deploying: spherical-protogalaxy-3131
Building image...
[+] Building 1.6s (11/11) FINISHED

```

After a few minutes, you should end up with something like that:

```

Deployed Image Tag: deploy-2022-08-22T11-26
Successfully pushed Docker image to Astronomer registry. Navigate to the Astronomer UI for confirmation that your deploy was successful.

Deployment can be accessed at the following URLs:

Deployment Dashboard: cloud.astronomer.io/cknaqyipv05731evsry6cj4n0/deployments/cl74npow0485901j0ikhpsgf4r
Airflow Dashboard: astronomer.astronomer.run/dhpsgf4r?orgId=org_0FGQMRjnZdcx3EDk

```

Back to your Deployment on Astro, wait until the status turned back to Healthy

The screenshot shows the Astronomer UI for a deployment named 'spherical-protogalaxy-3131' in the 'dev' namespace. The deployment status is 'HEALTHY'. The interface includes a table with deployment details and a section for worker queues.

NAME	WORKER TYPE	MAX TASKS PER WORKER	WORKER COUNT (MIN-MAX)
default	m5.xlarge (2 CPU, 14.535Gi MEM)	16	1-10

## Run you DAG

Click [Open Airflow](#)

Turn on the toggle of [extract\\_stars](#) DAG and wait a little bit.

dev

AI 1 Active 2 Paused 3

Filter DAGs by tag

Search DAGs

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks	Actions	Links
astronomer_monitoring_dag	airflow	1	*/5 * * * *	2022-08-22, 12:00:00	2022-08-22, 12:05:00	1	[Run] [Stop]	...
example_dag_advanced	community	0	@daily		2022-08-21, 00:00:00	0	[Run] [Stop]	...
example_dag_basic	airflow	0	@daily		2022-08-21, 00:00:00	0	[Run] [Stop]	...
extract_stars	airflow	1	@daily	2022-08-21, 00:00:00	2022-08-22, 00:00:00	1	[Run] [Stop]	...

Click `extract_stars`

Then click on the task `print_stars` and `Log`

DAG: extract\_stars

Schedule: @daily Next Run: 2022-08-22, 00:00:00

Grid Graph Calendar Task Duration Task Tries Landing Times Gantt Details Code Audit Log

22/08/2022, 12:07:42 25 All Run Types All Run States Clear Filters

Auto-refresh

Duration

00:00:00

00:00:02

00:00:04

get\_date

query\_github\_stats

print\_stars

Task Actions

Ignore All Deps Ignore Task State Ignore Task Deps Run

Past Future Upstream Downstream Recursive Failed Clear

Past Future Upstream Downstream Mark Failed

Past Future Upstream Downstream Mark Success

Download Log (by attempts):

1

Status: success

You should get something like that

Log by attempts

1

Jump To End

Toggle Wrap

Download

```
*** Reading remote log from s3://airflow-logs-ckv1i6hv002n0srx2pwpbpye/cl74npow0485901j0ikhp5gf4r/dag_id=extract_stars/run_id=scheduled__2022-08-21T00:00:00+00:00/task_id=print_stars/attempt=1.log.
[2022-08-22, 12:05:55 UTC] {taskinstance.py:1100} INFO - Dependencies all met for <TaskInstance: extract_stars.print_stars scheduled__2022-08-21T00:00:00+00:00 [queued]>
[2022-08-22, 12:05:55 UTC] {taskinstance.py:1180} INFO - Dependencies all met for <TaskInstance: extract_stars.print_stars scheduled__2022-08-21T00:00:00+00:00 [queued]>
[2022-08-22, 12:05:55 UTC] {taskinstance.py:1377} INFO -
-----
[2022-08-22, 12:05:55 UTC] {taskinstance.py:1370} INFO - Starting attempt 1 of 1
[2022-08-22, 12:05:55 UTC] {taskinstance.py:1379} INFO -
-----
[2022-08-22, 12:05:56 UTC] {taskinstance.py:1390} INFO - Executing <Task(PythonOperator): print_stars> on 2022-08-21 00:00:00+00:00
[2022-08-22, 12:05:56 UTC] {standard_task_runner.py:52} INFO - Started process 128 to run task
[2022-08-22, 12:05:56 UTC] {standard_task_runner.py:79} INFO - Running: ['airflow', 'tasks', 'run', 'extract_stars', 'print_stars', 'scheduled__2022-08-21T00:00:00+00:00', '--job-id', '25', '--raw', '--subdir', 'D
[2022-08-22, 12:05:56 UTC] {standard_task_runner.py:80} INFO - Job 25: Subtask print_stars
[2022-08-22, 12:05:56 UTC] {task_command.py:371} INFO - Running <TaskInstance: extract_stars.print_stars scheduled__2022-08-21T00:00:00+00:00 [running]> on host 10.0.0.230
[2022-08-22, 12:05:56 UTC] {taskinstance.py:1590} INFO - Exporting the following env vars:
AIRFLOW_CTX_DAG_OWNER=airflow
AIRFLOW_CTX_DAG_ID=extract_stars
AIRFLOW_CTX_TASK_ID=print_stars
AIRFLOW_CTX_EXECUTION_DATE=2022-08-21T00:00:00+00:00
AIRFLOW_CTX_TRY_NUMBER=1
AIRFLOW_CTX_DAG_RUN_ID=scheduled__2022-08-21T00:00:00+00:00
[2022-08-22, 12:05:56 UTC] {logging_mixin.py:115} INFO - As of 2022-08-21, Apache Airflow has 27101 stars on Github!
[2022-08-22, 12:05:56 UTC] {python.py:173} INFO - Done. Returned value was: None
[2022-08-22, 12:05:56 UTC] {taskinstance.py:1416} INFO - Marking task as SUCCESS. dag_id=extract_stars, task_id=print_stars, execution_date=20220821T000000, start_date=20220822T120555, end_date=20220822T120556
[2022-08-22, 12:05:56 UTC] {local_task_job.py:156} INFO - Task exited with return code 0
[2022-08-22, 12:05:56 UTC] {local_task_job.py:273} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

Well done! You've just successfully built and deployed your first DAG on Astro! 🎉

## Additional resources

Astro Environment Variables: <https://docs.astronomer.io/astro/environment-variables>

Deploy on Astro: <https://docs.astronomer.io/astro/deploy-code>