Apache Airflow

Enterprise data orchestration for academic research (?)

Andy Holguin | Application Developer's SIG | April 13, 2021

Agenda

- IET Research Programming Service
- Airflow Features
- AWS Example
- UI Demo
- Closing Thoughts / Conclusion

IET | Research Programming Service

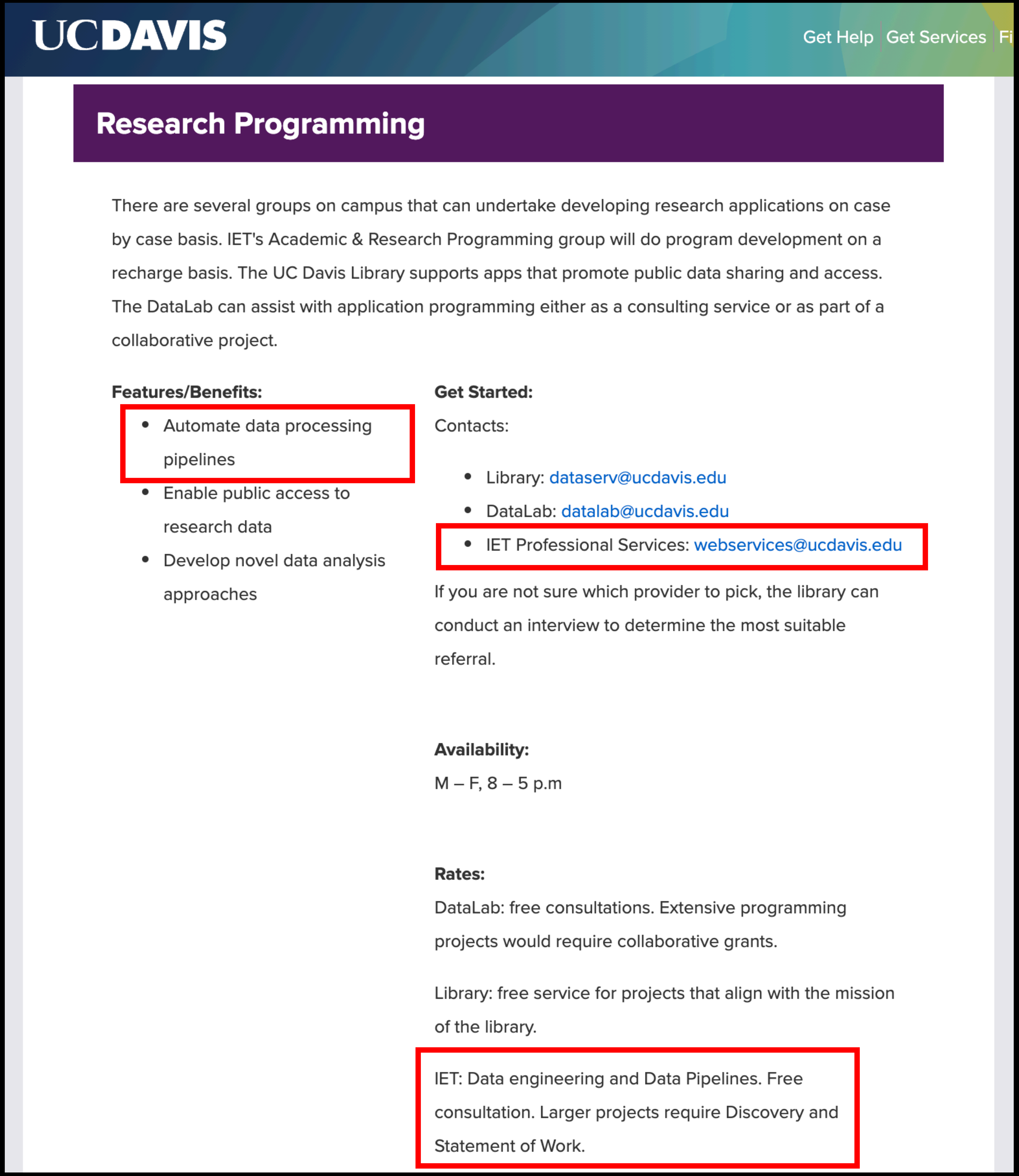
IT Service Catalog: https://servicehub.ucdavis.edu/servicehub?id=it_catalog_content&spa=1&sys_id=4a7669191b9798103f4286ae6e4bcbfa

- Data engineering & pipelines
- API development
- Web apps & Dashboards

→ Research Data

NOT

- Institutional data
- Student data
- Business data



Research Programming

There are several groups on campus that can undertake developing research applications on case by case basis. IET's Academic & Research Programming group will do program development on a recharge basis. The UC Davis Library supports apps that promote public data sharing and access. The DataLab can assist with application programming either as a consulting service or as part of a collaborative project.

Features/Benefits:

- Automate data processing pipelines
- Enable public access to research data
- Develop novel data analysis approaches

Get Started:

Contacts:

- Library: dataserv@ucdavis.edu
- DataLab: datalab@ucdavis.edu
- IET Professional Services: webservices@ucdavis.edu

If you are not sure which provider to pick, the library can conduct an interview to determine the most suitable referral.

Availability:

M – F, 8 – 5 p.m

Rates:

DataLab: free consultations. Extensive programming projects would require collaborative grants.

Library: free service for projects that align with the mission of the library.

IET: Data engineering and Data Pipelines. Free consultation. Larger projects require Discovery and Statement of Work.

Apache Airflow

“Workflow automation and scheduling that can be used to author and manage data pipelines”[1]

- Workflows
- Scheduling
- Error handling
- Monitoring
- Reporting
- Scaling

- Open source
- Hosted services on AWS, GCP, etc.
- **Run tasks in a repeatable & reliable manner**

[1]: https://projects.apache.org/project.html?airflow

Airflow

History | Who uses it?

- 2014 | project started at Airbnb
- 2016 | Apache Software Foundation Incubator program
- 2019 | Apache Top-Level Project
- 2020 | Airflow v. 2.0

- Widely used - Over 400 organizations[1]
- Apache License 2.0

[1]: https://github.com/apache/incubator-airflow#who-uses-apache-airflow

Airflow
Concepts

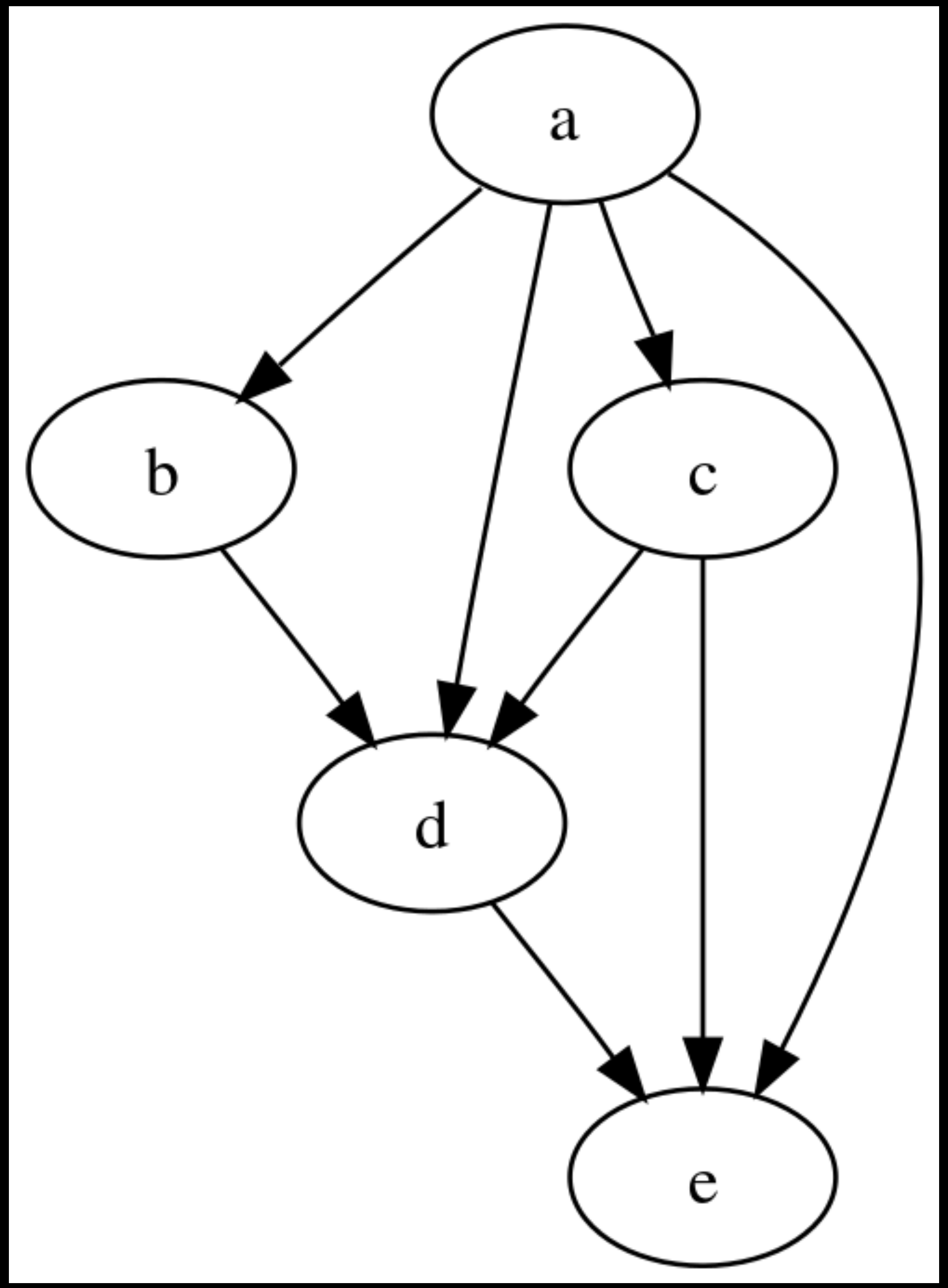
- **Workflows/DAGs**: Directed Acyclic Graphs

  "Collection of all the tasks you want to run, organized in a way that reflects their relationships and dependencies"[1]

  - Defined in Python

- Tasks/components:

  - Downloading, filtering, etc.

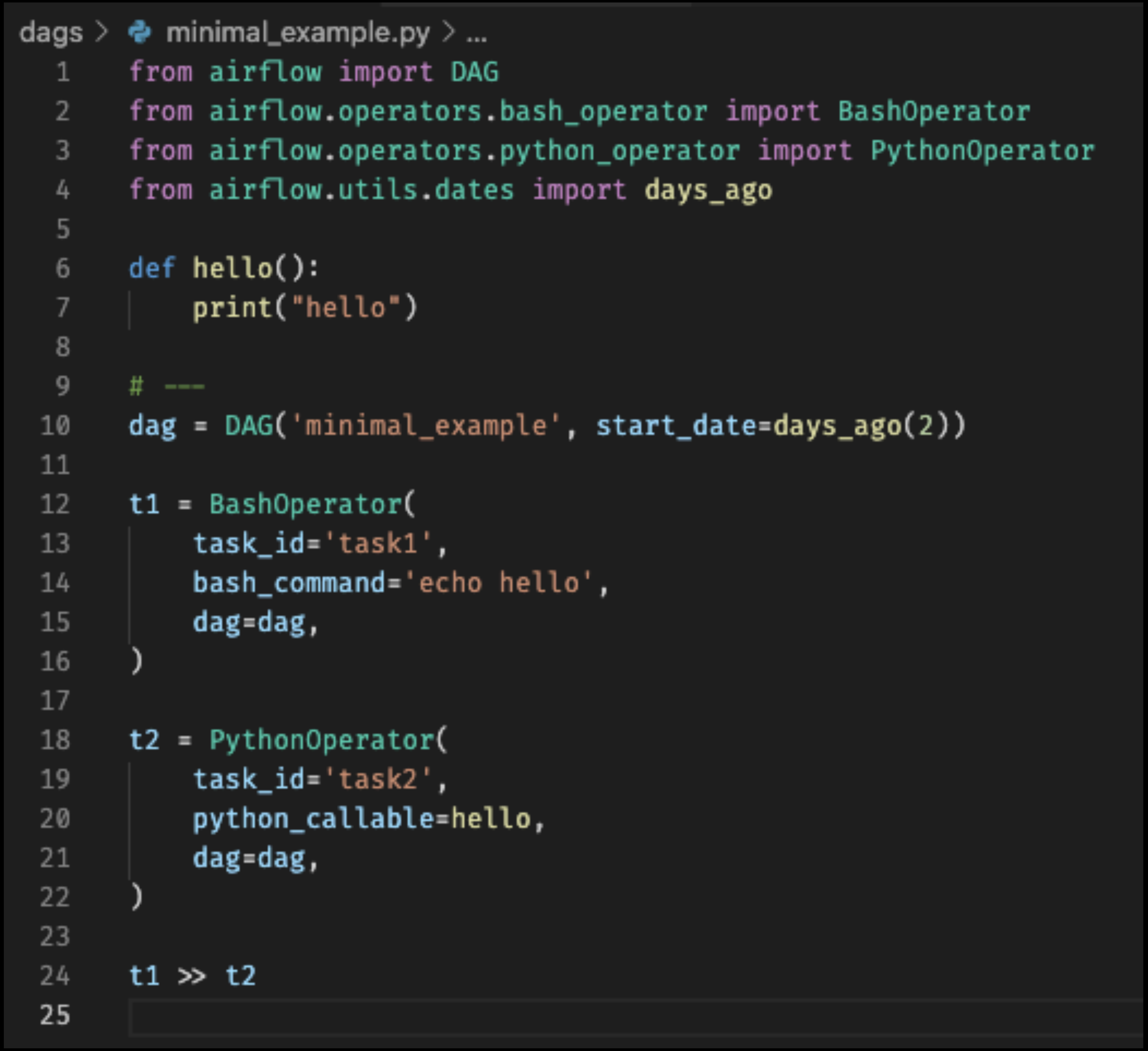

$$\begin{tikzcd}
 & a \arrow[dl] \arrow[d] \arrow[dr] \arrow[ddr, bend left] & \\
b \arrow[r] & d \arrow[dr] & c \arrow[l] \arrow[d] \\
 & & e
\end{tikzcd}$$

[1]: https://airflow.apache.org/docs/apache-airflow/1.10.15/concepts.html#dags

Operators/tasks

Concepts

- Single task in a workflow
  - **`BashOperator`**
  - **`PythonOperator`**
  - **`EmailOperator`**
  - `etc.`
- Dependecies define DAGs
  - >> and <<



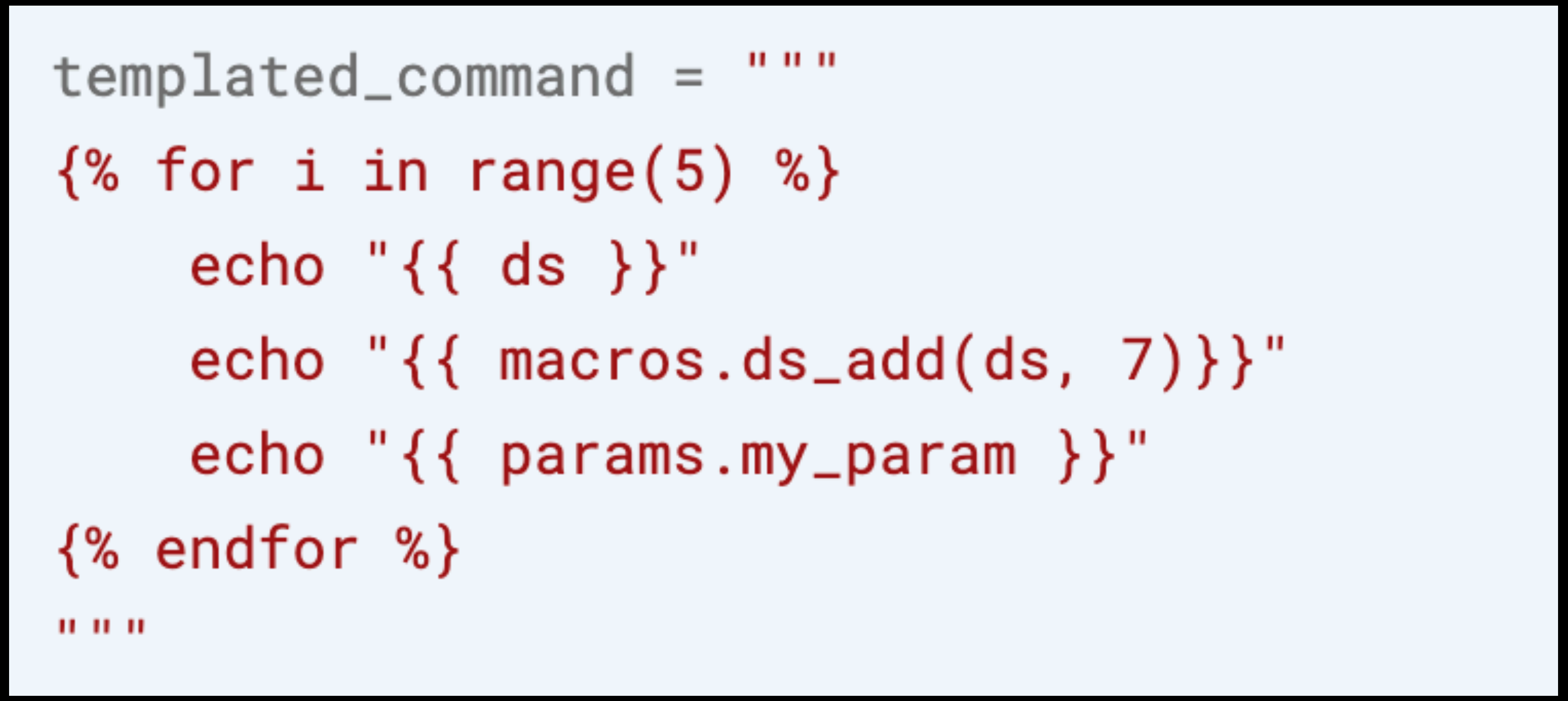
dags > minimal_example.py > ...

```python
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.operators.python_operator import PythonOperator
from airflow.utils.dates import days_ago

def hello():
    print("hello")

# ---
dag = DAG('minimal_example', start_date=days_ago(2))

t1 = BashOperator(
    task_id='task1',
    bash_command='echo hello',
    dag=dag,
)

t2 = PythonOperator(
    task_id='task2',
    python_callable=hello,
    dag=dag,
)

t1 >> t2

```

Templating

Concepts

- Substitute info when running DAG
- Jinja templating language
- Built in macros
  - datetime, uuid, etc.



```
templated_command = """
{% for i in range(5) %}
    echo "{{ ds }}"
    echo "{{ macros.ds_add(ds, 7)}}"
    echo "{{ params.my_param }}"
{% endfor %}
"""
```

Sensors

Concepts

- Wait for condition to be true
  - **`FileSensor`** - check for the existence of a file
  - **`ExternalTaskSensor`** - task in another DAG
  - **`HttpSensor`** - request URL and check content
  - **`SqlSensor`** - run query and check result
  - Contrib/Providers (AWS, Azure, Redis, etc.)

DAG Runs/Scheduling

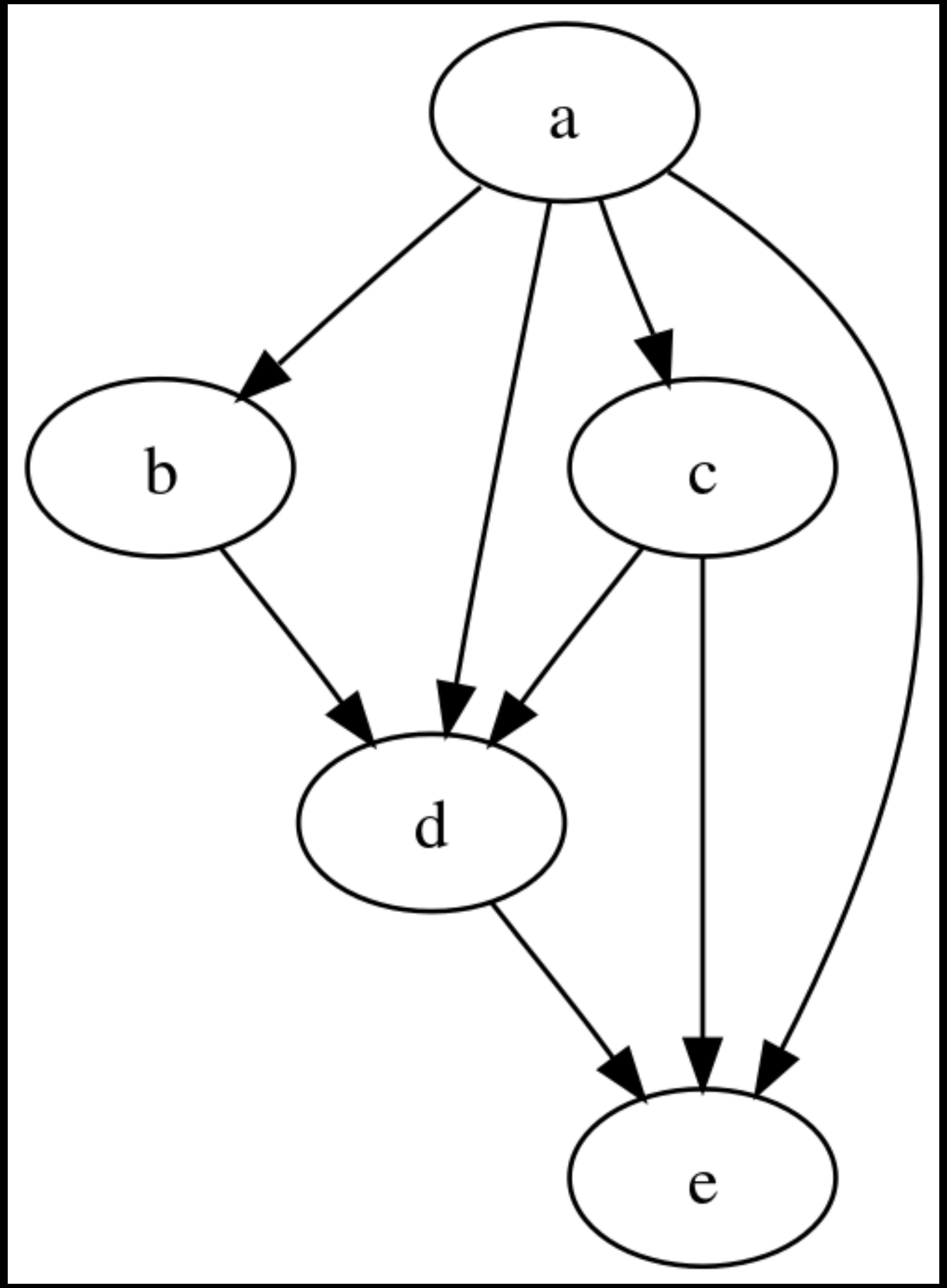
Concepts

- Scheduler
  - Manual
  - Scheduled
  - External trigger
- State
  - Running, paused, queued, failed, succeeded, etc.

Executors

Concepts

- Runs the tasks
  - **`SequentialExecutor`** - run one task at a time (useful for testing)
  - **`LocalExecutor`** - run tasks on a single system (using available resources)
  - **`CeleryExecutor`** - use Celery as task manager on cluster
  - **`KubernetesExecutor`**



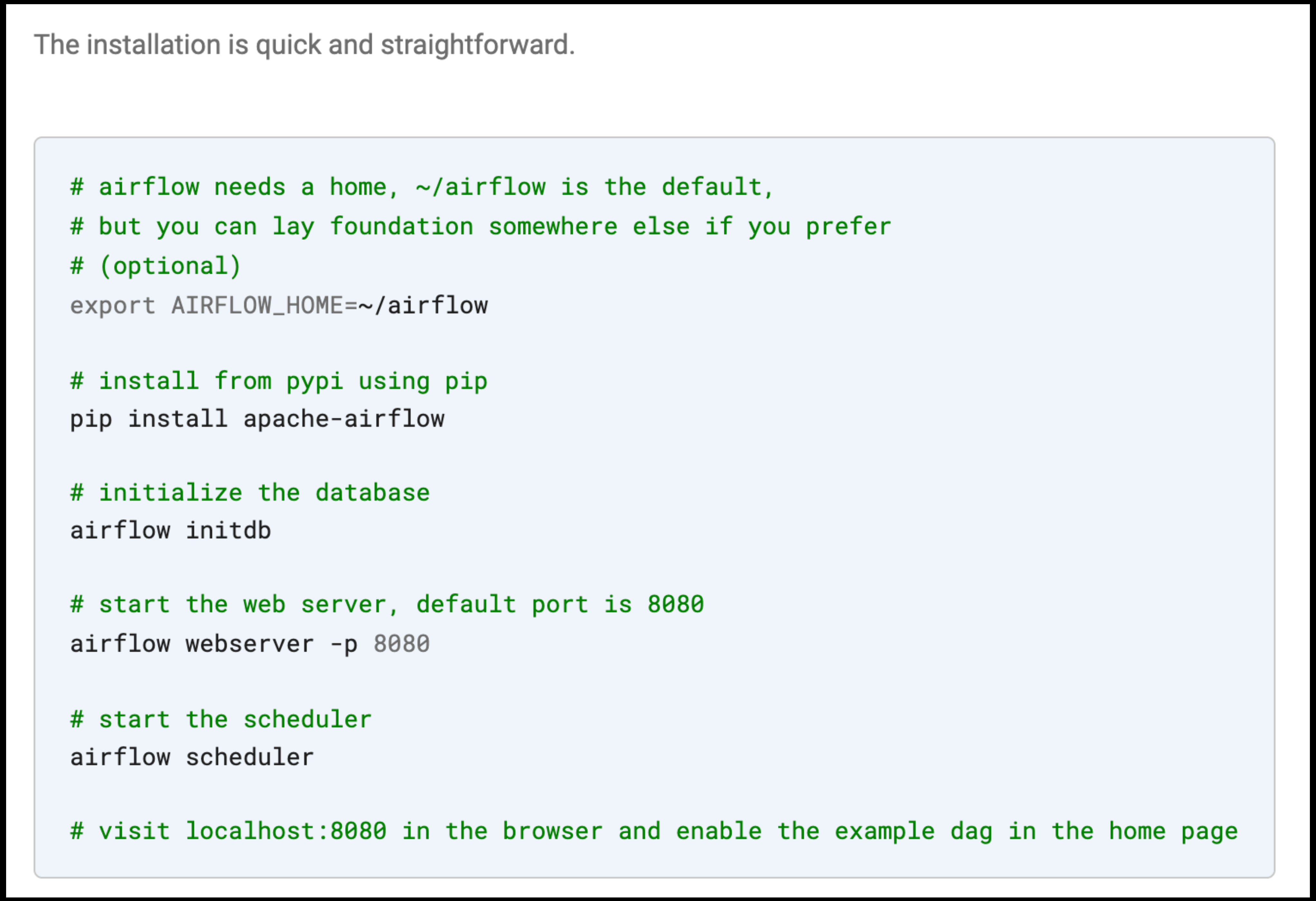
Many more features…

- UI
- Hooks
- Pools
- Connections
- XComs
- Branching
- Backfilling
- SubDAGs
- Etc.

Installation

- Run locally (mostly for testing)

https://airflow.apache.org/docs/apache-airflow/1.10.15/start.html



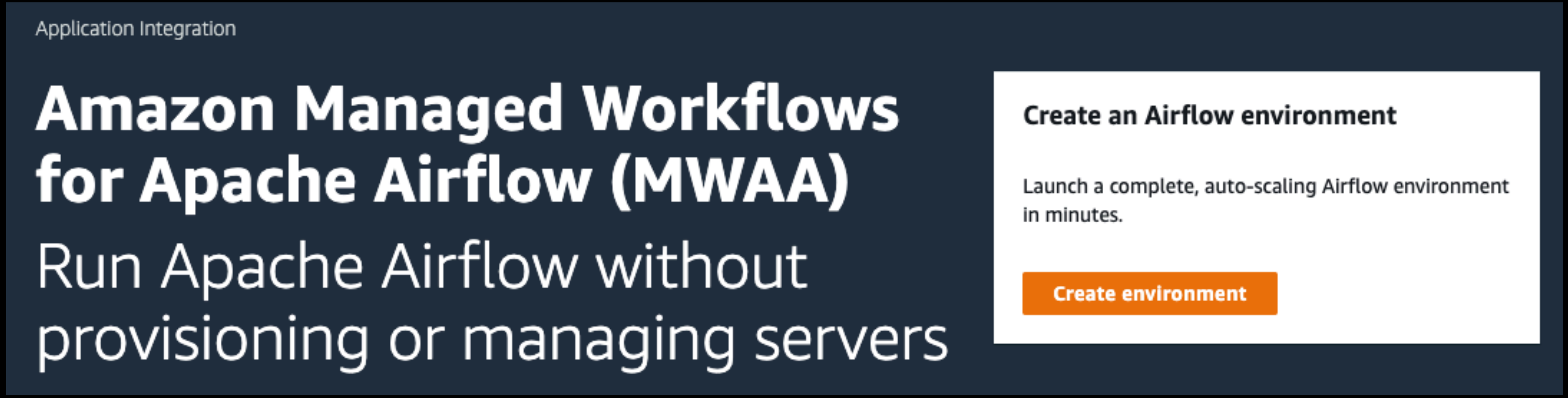
The installation is quick and straightforward.

```
# airflow needs a home, ~/airflow is the default,
# but you can lay foundation somewhere else if you prefer
# (optional)
export AIRFLOW_HOME=~/airflow

# install from pypi using pip
pip install apache-airflow

# initialize the database
airflow initdb

# start the web server, default port is 8080
airflow webserver -p 8080

# start the scheduler
airflow scheduler

# visit localhost:8080 in the browser and enable the example dag in the home page
```

Hosting

- Open Source —> host it yourself
- Managed Services
  - AWS | Amazon Managed Workflows for Apache Airflow (MWAA)
  - GCP | Cloud Composer
  - Astronomer (https://www.astronomer.io/)

AWS Setup



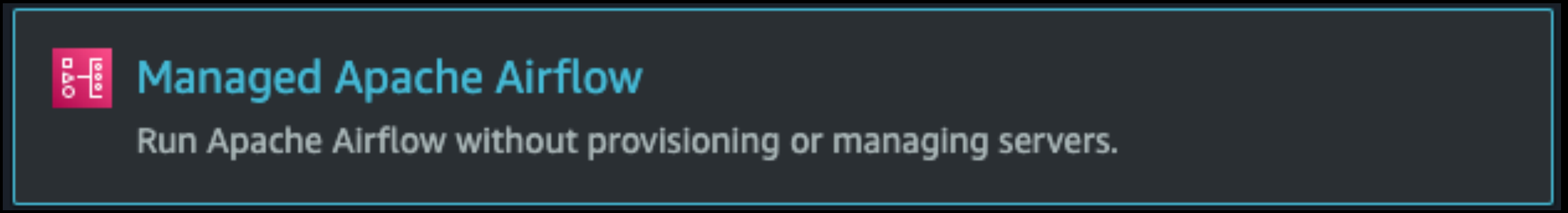
Application Integration

Amazon Managed Workflows for Apache Airflow (MWAA)

Run Apache Airflow without provisioning or managing servers

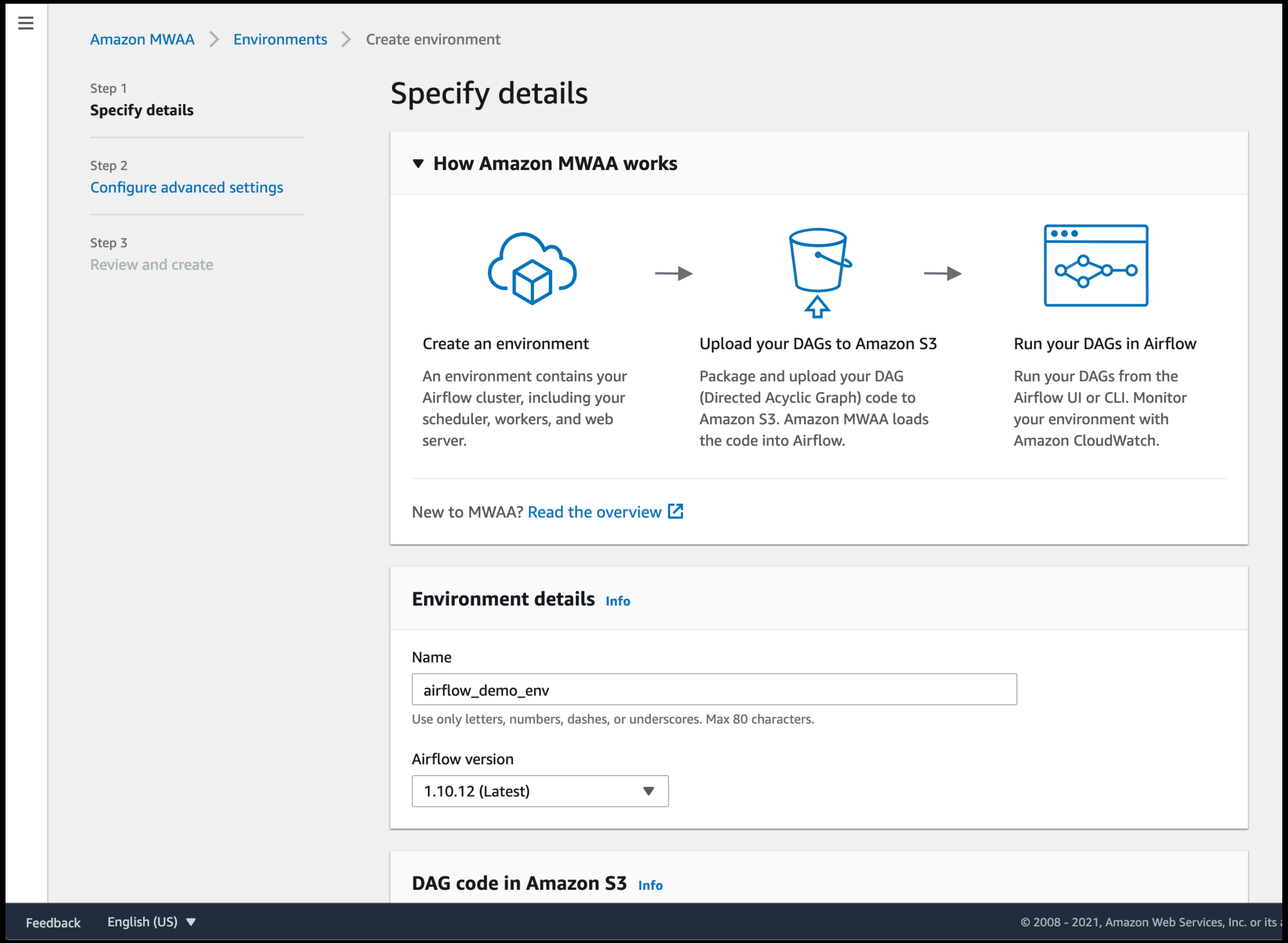
Create an Airflow environment

Launch a complete, auto-scaling Airflow environment in minutes.



Managed Apache Airflow

Run Apache Airflow without provisioning or managing servers.



Amazon MWAA > Environments > Create environment

Step 1
Specify details

Step 2
Configure advanced settings

Step 3
Review and create

Specify details

How Amazon MWAA works

Create an environment
An environment contains your Airflow cluster, including your scheduler, workers, and web server.

Upload your DAGs to Amazon S3
Package and upload your DAG (Directed Acyclic Graph) code to Amazon S3. Amazon MWAA loads the code into Airflow.

Run your DAGs in Airflow
Run your DAGs from the Airflow UI or CLI. Monitor your environment with Amazon CloudWatch.

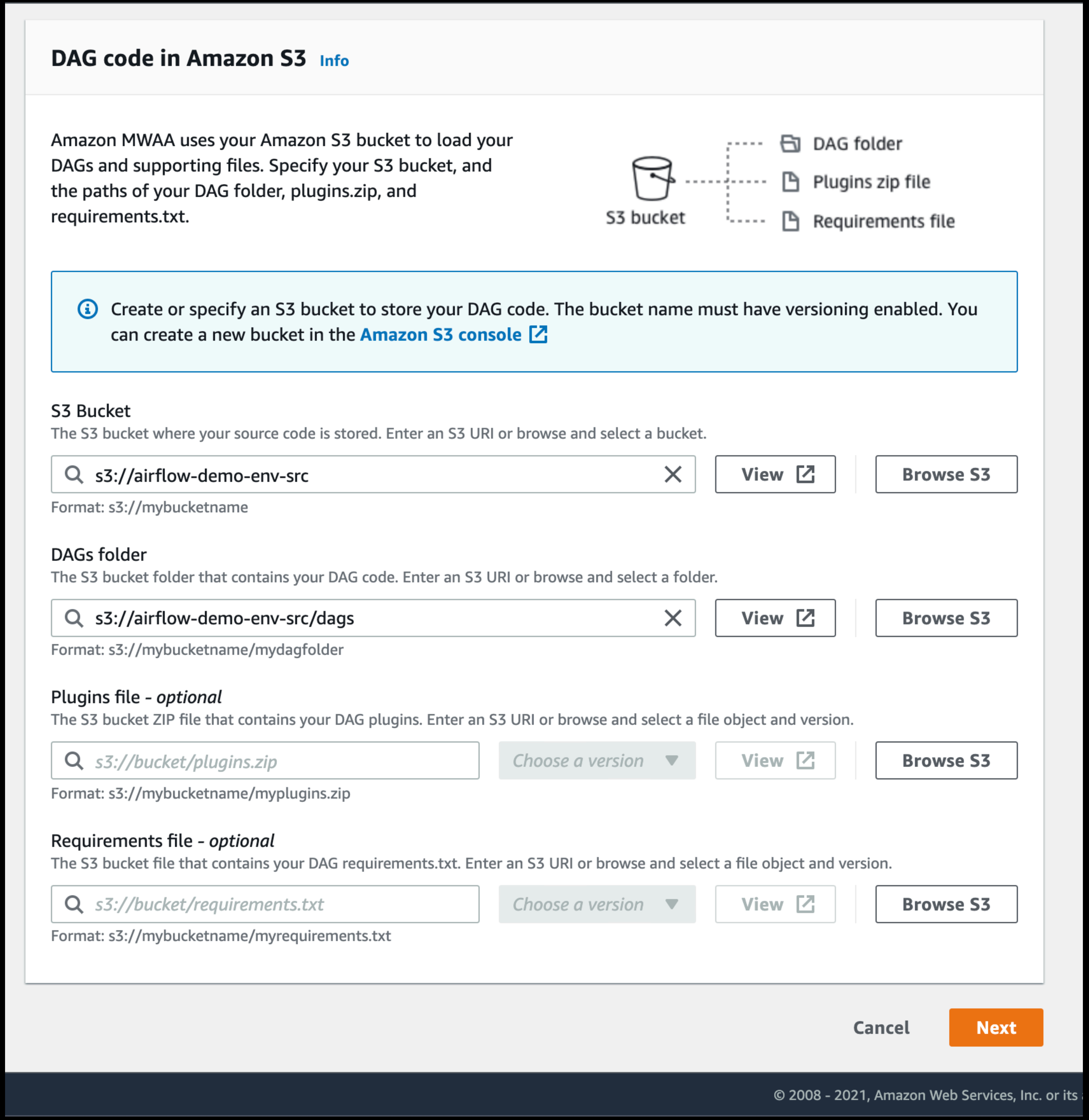
New to MWAA? Read the overview

Environment details Info

Name
airflow_demo_env
Use only letters, numbers, dashes, or underscores. Max 80 characters.

Airflow version
1.10.12 (Latest)

DAG code in Amazon S3 Info



DAG code in Amazon S3 Info

Amazon MWAA uses your Amazon S3 bucket to load your DAGs and supporting files. Specify your S3 bucket, and the paths of your DAG folder, plugins.zip, and requirements.txt.

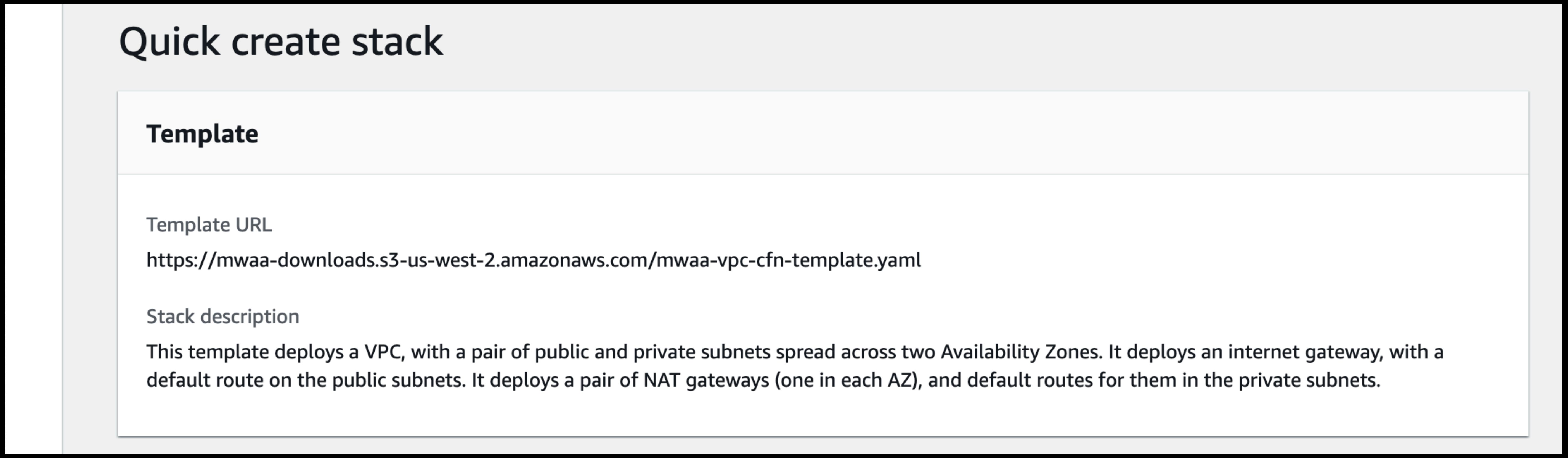
Create or specify an S3 bucket to store your DAG code. The bucket name must have versioning enabled. You can create a new bucket in the Amazon S3 console

S3 Bucket
The S3 bucket where your source code is stored. Enter an S3 URI or browse and select a bucket.
s3://airflow-demo-env-src
Format: s3://mybucketname

DAGs folder
The S3 bucket folder that contains your DAG code. Enter an S3 URI or browse and select a folder.
s3://airflow-demo-env-src/dags
Format: s3://mybucketname/mydagfolder

Plugins file - *optional*
The S3 bucket ZIP file that contains your DAG plugins. Enter an S3 URI or browse and select a file object and version.
s3://bucket/plugins.zip — *Choose a version*
Format: s3://mybucketname/myplugins.zip

Requirements file - *optional*
The S3 bucket file that contains your DAG requirements.txt. Enter an S3 URI or browse and select a file object and version.
s3://bucket/requirements.txt — *Choose a version*
Format: s3://mybucketname/myrequirements.txt



Quick create stack

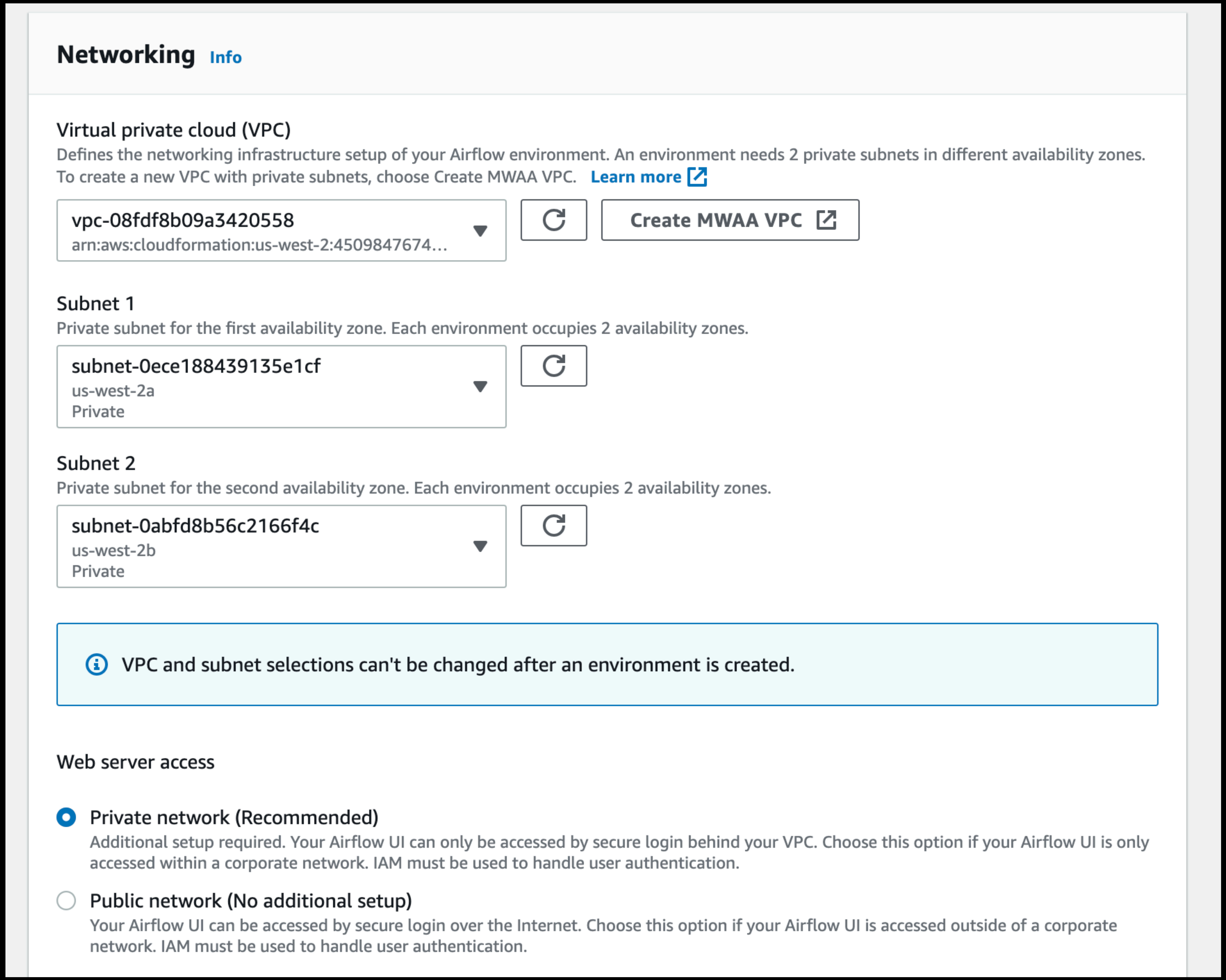
Template

Template URL

https://mwaa-downloads.s3-us-west-2.amazonaws.com/mwaa-vpc-cfn-template.yaml

Stack description

This template deploys a VPC, with a pair of public and private subnets spread across two Availability Zones. It deploys an internet gateway, with a default route on the public subnets. It deploys a pair of NAT gateways (one in each AZ), and default routes for them in the private subnets.



Networking Info

Virtual private cloud (VPC)

Defines the networking infrastructure setup of your Airflow environment. An environment needs 2 private subnets in different availability zones. To create a new VPC with private subnets, choose Create MWAA VPC. Learn more

vpc-08fdf8b09a3420558
arn:aws:cloudformation:us-west-2:4509847674…

Create MWAA VPC

Subnet 1

Private subnet for the first availability zone. Each environment occupies 2 availability zones.

subnet-0ece188439135e1cf
us-west-2a
Private

Subnet 2

Private subnet for the second availability zone. Each environment occupies 2 availability zones.

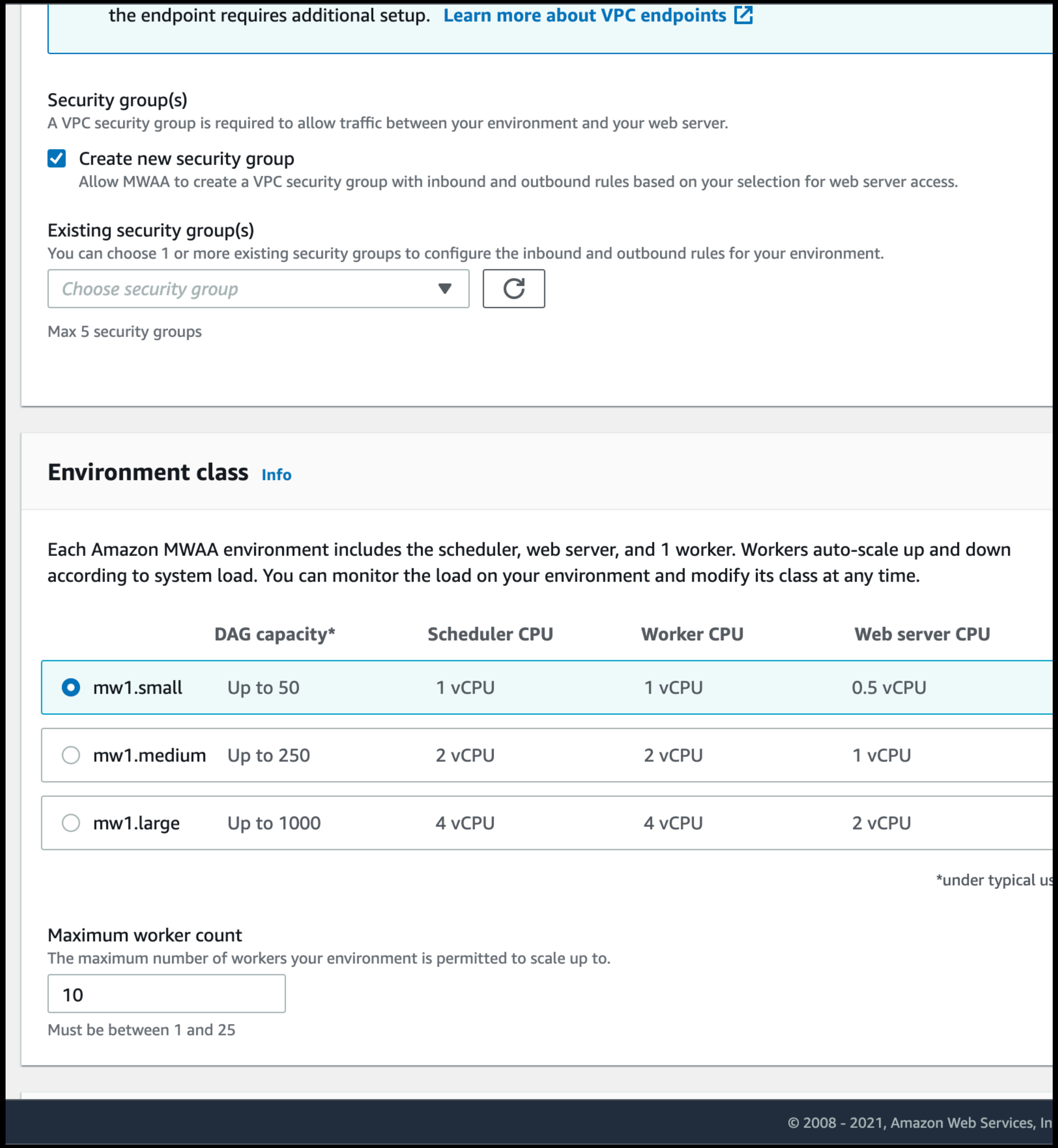
subnet-0abfd8b56c2166f4c
us-west-2b
Private

VPC and subnet selections can't be changed after an environment is created.

Web server access

[✓ Private network (Recommended)]
Additional setup required. Your Airflow UI can only be accessed by secure login behind your VPC. Choose this option if your Airflow UI is only accessed within a corporate network. IAM must be used to handle user authentication.

Public network (No additional setup)
Your Airflow UI can be accessed by secure login over the Internet. Choose this option if your Airflow UI is accessed outside of a corporate network. IAM must be used to handle user authentication.



the endpoint requires additional setup. Learn more about VPC endpoints

Security group(s)

A VPC security group is required to allow traffic between your environment and your web server.

[✓ Create new security group]
Allow MWAA to create a VPC security group with inbound and outbound rules based on your selection for web server access.

Existing security group(s)

You can choose 1 or more existing security groups to configure the inbound and outbound rules for your environment.

Choose security group

Max 5 security groups

Environment class Info

Each Amazon MWAA environment includes the scheduler, web server, and 1 worker. Workers auto-scale up and down according to system load. You can monitor the load on your environment and modify its class at any time.

	DAG capacity*	Scheduler CPU	Worker CPU	Web server CPU
● mw1.small	Up to 50	1 vCPU	1 vCPU	0.5 vCPU
○ mw1.medium	Up to 250	2 vCPU	2 vCPU	1 vCPU
○ mw1.large	Up to 1000	4 vCPU	4 vCPU	2 vCPU

*under typical us

Maximum worker count

The maximum number of workers your environment is permitted to scale up to.

10

Must be between 1 and 25

© 2008 - 2021, Amazon Web Services, In

airflow_demo_env is being created. This takes 20-30 minutes.

Amazon MWAA > Environments

Airflow environments

Next steps
While you wait for your environment to be created, learn about how to access the Airflow UI and work with your DAGs.

Learn more

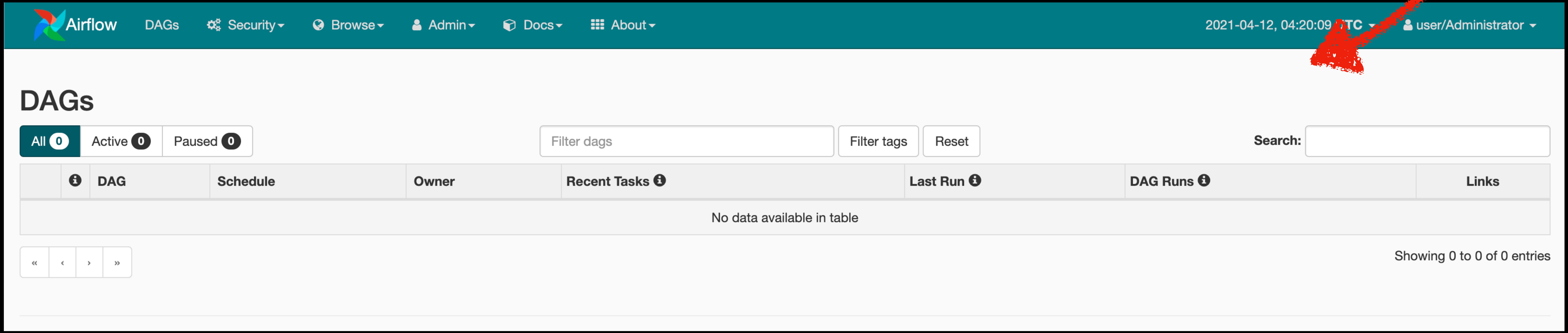
Environments (1)

Edit · Delete · Actions · Create environment

Find environments

Name	Status	Created date	Airflow version	Airflow UI
airflow_demo_env	Available	Apr 11, 2021 20:57:42 (UTC-07:00)	1.10.12	Open Airflow UI

https://< ID >.us-west-2.airflow.amazonaws.com/home



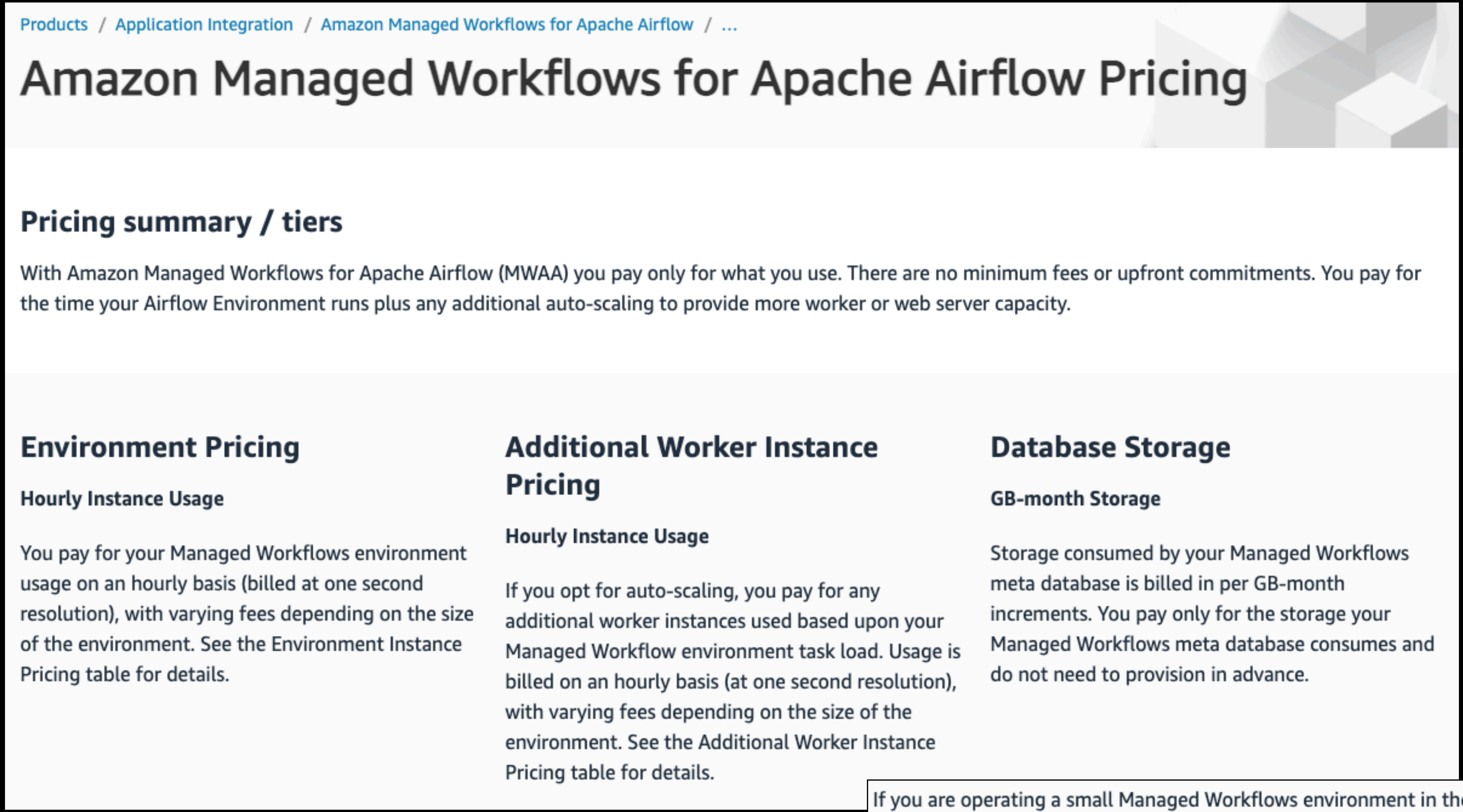
DAGs

All 0 · Active 0 · Paused 0

Filter dags · Filter tags · Reset · Search:

	DAG	Schedule	Owner	Recent Tasks	Last Run	DAG Runs	Links
No data available in table							

Showing 0 to 0 of 0 entries



Products / Application Integration / Amazon Managed Workflows for Apache Airflow / …

Amazon Managed Workflows for Apache Airflow Pricing

Pricing summary / tiers

With Amazon Managed Workflows for Apache Airflow (MWAA) you pay only for what you use. There are no minimum fees or upfront commitments. You pay for the time your Airflow Environment runs plus any additional auto-scaling to provide more worker or web server capacity.

Environment Pricing

Hourly Instance Usage

You pay for your Managed Workflows environment usage on an hourly basis (billed at one second resolution), with varying fees depending on the size of the environment. See the Environment Instance Pricing table for details.

Additional Worker Instance Pricing

Hourly Instance Usage

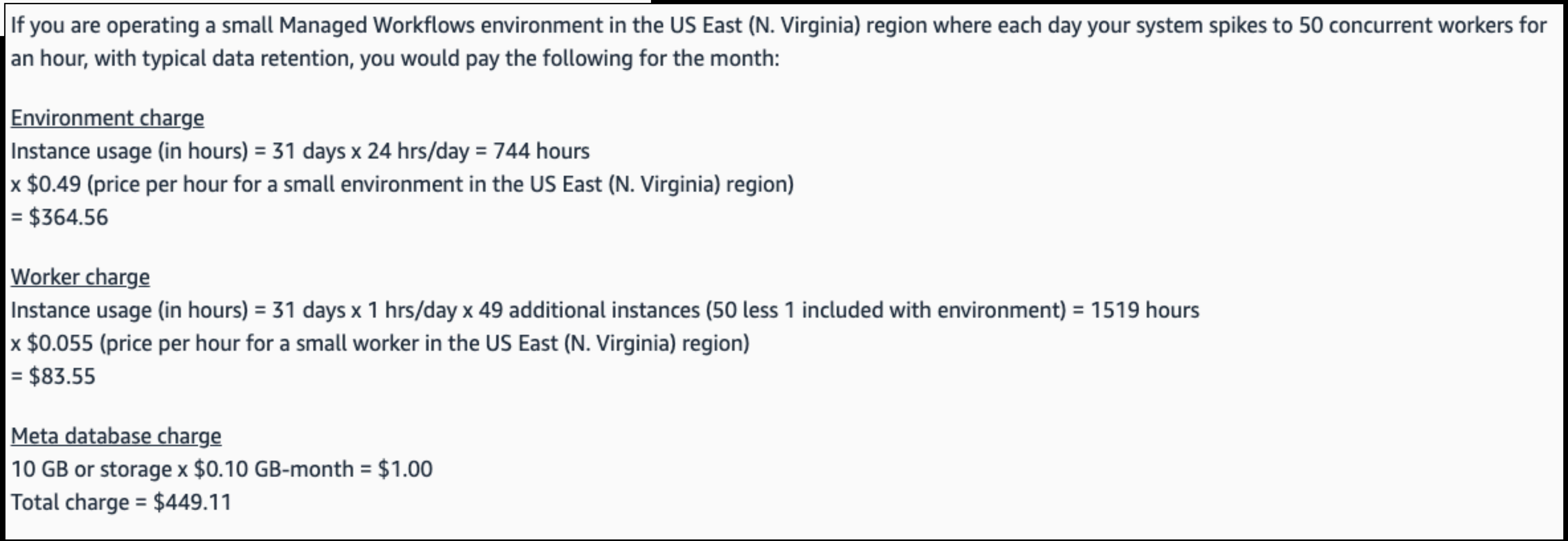
If you opt for auto-scaling, you pay for any additional worker instances used based upon your Managed Workflow environment task load. Usage is billed on an hourly basis (at one second resolution), with varying fees depending on the size of the environment. See the Additional Worker Instance Pricing table for details.

Database Storage

GB-month Storage

Storage consumed by your Managed Workflows meta database is billed in per GB-month increments. You pay only for the storage your Managed Workflows meta database consumes and do not need to provision in advance.

Billing Example



If you are operating a small Managed Workflows environment in the US East (N. Virginia) region where each day your system spikes to 50 concurrent workers for an hour, with typical data retention, you would pay the following for the month:

Environment charge
Instance usage (in hours) = 31 days x 24 hrs/day = 744 hours
x $0.49 (price per hour for a small environment in the US East (N. Virginia) region)
= $364.56

Worker charge
Instance usage (in hours) = 31 days x 1 hrs/day x 49 additional instances (50 less 1 included with environment) = 1519 hours
x $0.055 (price per hour for a small worker in the US East (N. Virginia) region)
= $83.55

Meta database charge
10 GB or storage x $0.10 GB-month = $1.00
Total charge = $449.11

UI Demo

Alternatives

- Argo, Prefect, Dagster, Luigi, AWS step functions, etc.
- https://github.com/pditommaso/awesome-pipeline

Closing thoughts / Conclusions

Pro

- Open source & portable
- Nice UI
- Scheduling, logging, retries, etc.
- Clear separation of tasks

Con

- Cost / not serverless
- Some effort to integrate existing tasks (e.g., dependencies, storage)
- Mostly for running production data engineering tasks

Use cases

- Projects with frequent standardized data processing tasks
- Projects already running IET services on AWS (e.g., SRCE)
- Connect to scalable AWS/GCP infrastructure

END

- Has anyone used Airflow?
- Other workflow/pipeline/ETL systems?
- Projects that would benefit from Airflow?

Contact:

Andy Holguin

ajholguin@ucdavis.edu

UCD Slack: @Andrew Holguin