



# AWS Lambda Executor: The Speed of Local Execution with the Advantages of Remote

Nikolas (Niko) Oliveira

3.0

# Who Am I?



- Apache Airflow committer
- Sr. software engineer at Amazon
  - Amazon Managed Workflows for Apache Airflow (MWAA)
  - Founding member of the Amazon Apache Airflow Open Source Team

# Executors in Airflow



- Executors facilitate the running of Airflow tasks (Task Instances)
- The Scheduler decides *when* a task runs; the executor decides *where* and *how*
- Executors run within the Scheduler process
- Pluggable and extensible, you can write your very own!

# Executors in Airflow



- There are many types of Airflow executors, but some major ones include:
  - **Local Executors:** Airflow tasks are executed on the same host that the executor (i.e. scheduler) is running on. E.g.: **LocalExecutor**

# Executors in Airflow



- There are many types of Airflow executors, but some major ones include:
  - **Local Executors:** Airflow tasks are executed on the same host that the executor (i.e. scheduler) is running on. E.g.: **LocalExecutor**
  - **Remote Queued/Batched Executors:** Airflow tasks are sent to a central queue where remote workers pull tasks to execute. Often workers are persistent and run multiple tasks at once: E.g.: **CeleryExecutor, AwsBatchExecutor**

# Executors in Airflow

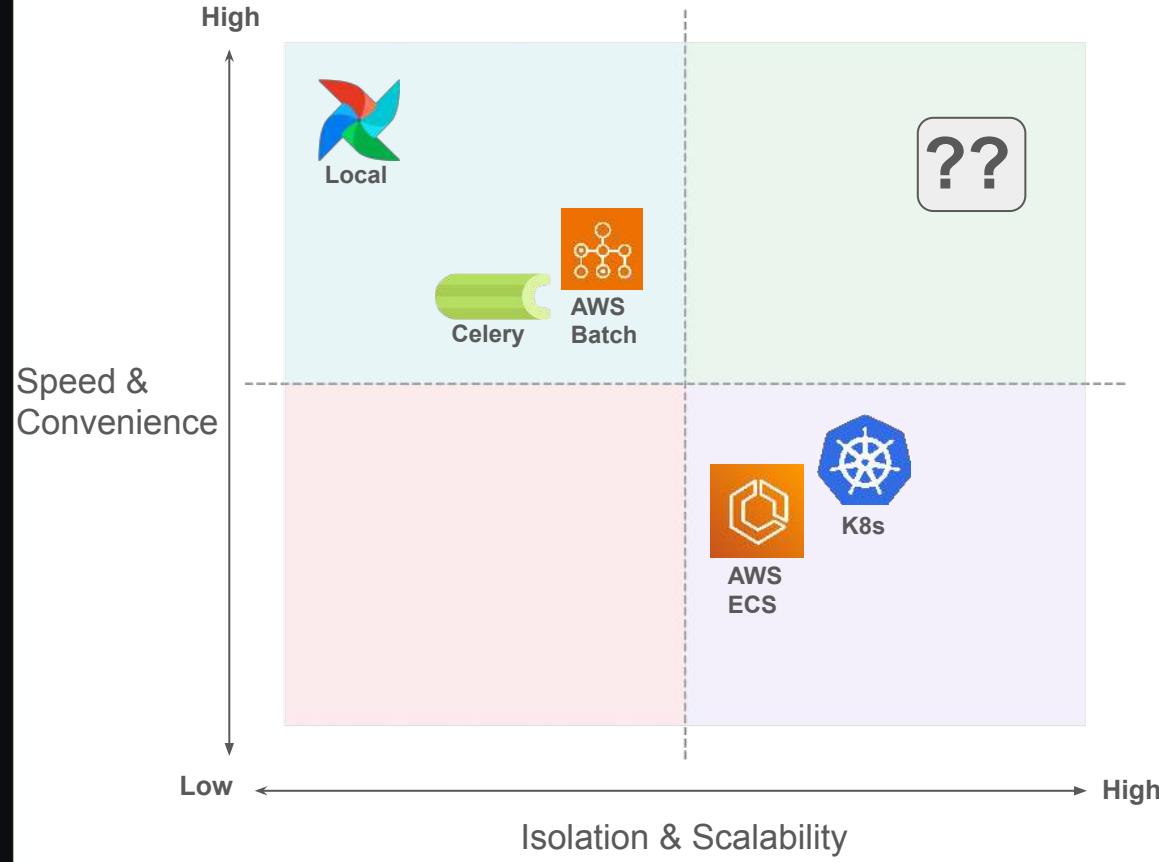


- There are many types of Airflow executors, but some major ones include:
  - **Local Executors:** Airflow tasks are executed on the same host that the executor (i.e. scheduler) is running on. E.g.: **LocalExecutor**
  - **Remote Queued/Batched Executors:** Airflow tasks are sent to a central queue where remote workers pull tasks to execute. Often workers are persistent and run multiple tasks at once: E.g.: **CeleryExecutor, AwsBatchExecutor**
  - **Remote Containerized Executors:** Airflow tasks are executed ad hoc inside containers/pods. Each task is isolated in its own environment. E.g.: **KubernetesExecutor, AwsEcsExecutor**



## Speed vs Isolation

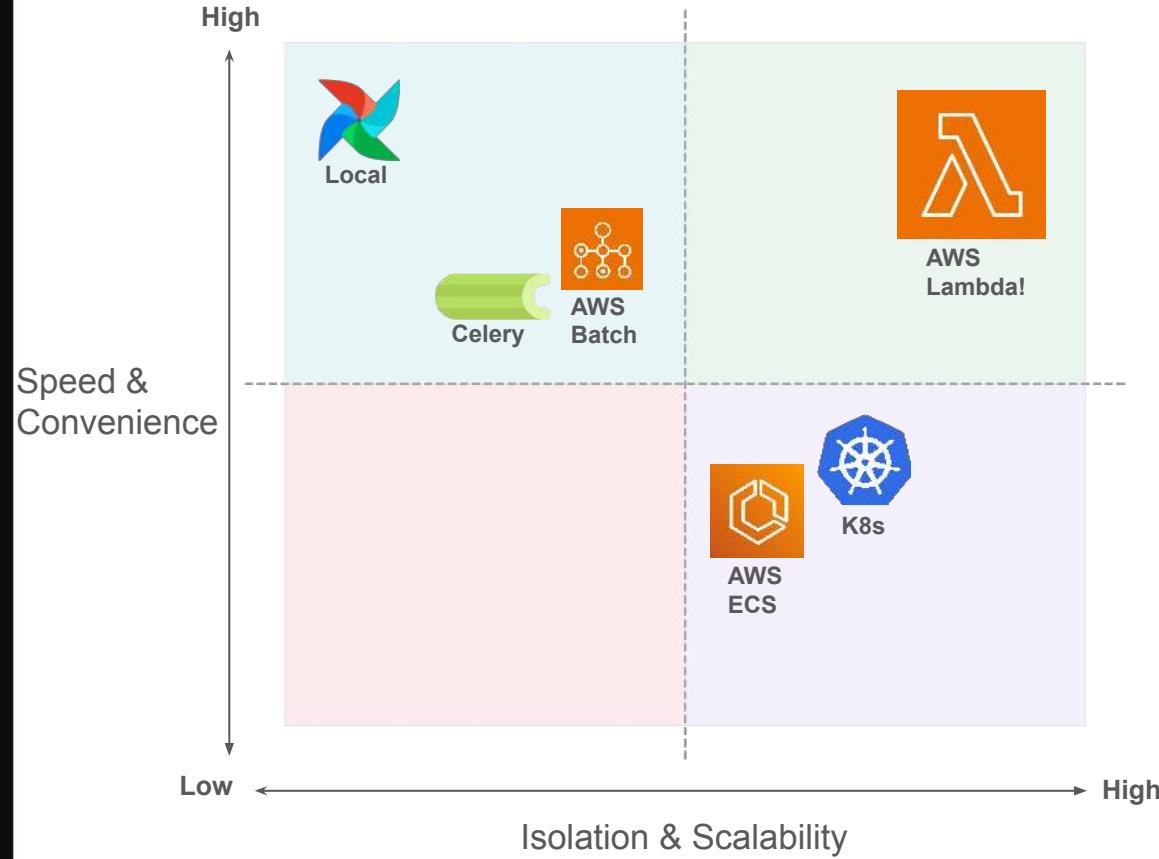
Speed & convenience vs isolation & scalability can you have it all?





## Speed vs Isolation

Yes, with the Lambda Executor!



# Introducing: Lambda Executor



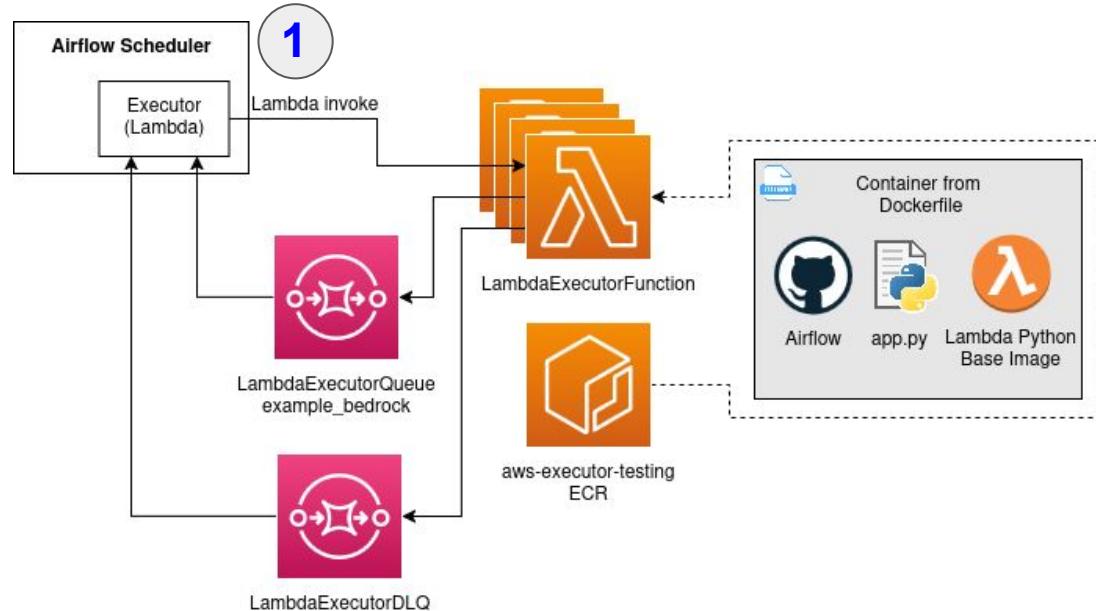
- Ephemeral containerized compute environment
- Massively scalable architecture
- Low latency (warm pools, Python runtime)
- Resilient: Highly available, retries, and dead letter queue
- Ideal for short to medium running tasks:
  - 15 min max execution time
  - Restrictive memory and storage (~10GB)



## Lambda Executor Architecture

1

Lambda Executor receives task from the Scheduler. Queueing them and ultimately calling an asynchronous invoke for each Airflow Task to be run.





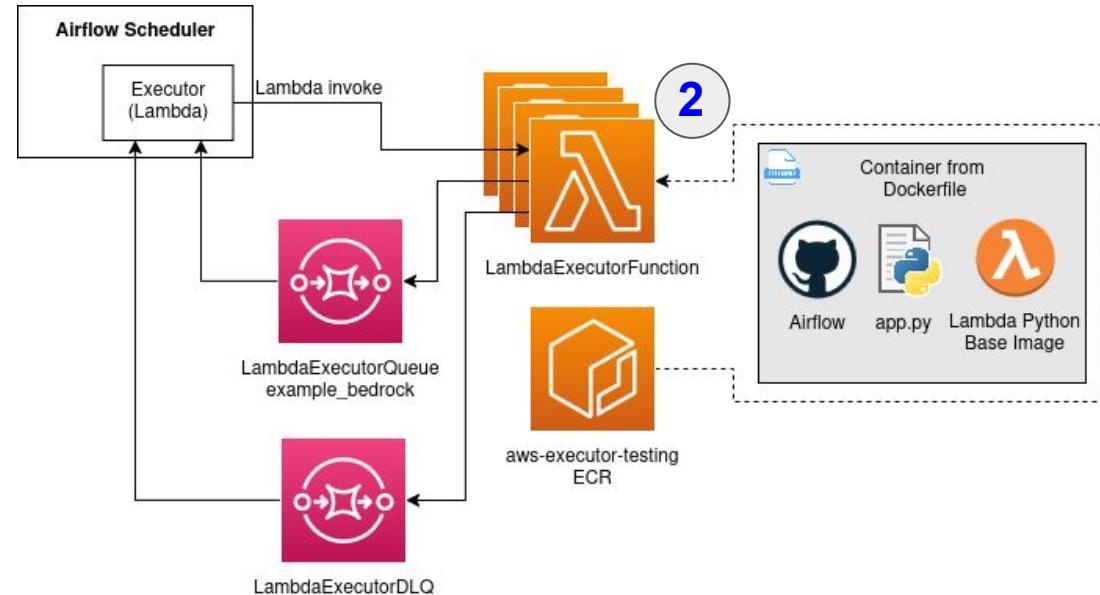
## Lambda Executor Architecture

2

Lambda starts a Firecracker VM based off of the provided image.

Users must build the image to their specification and create the Lambda Function.

Example app/handler code and image building tips found [here](#).



## Lambda Executor Architecture

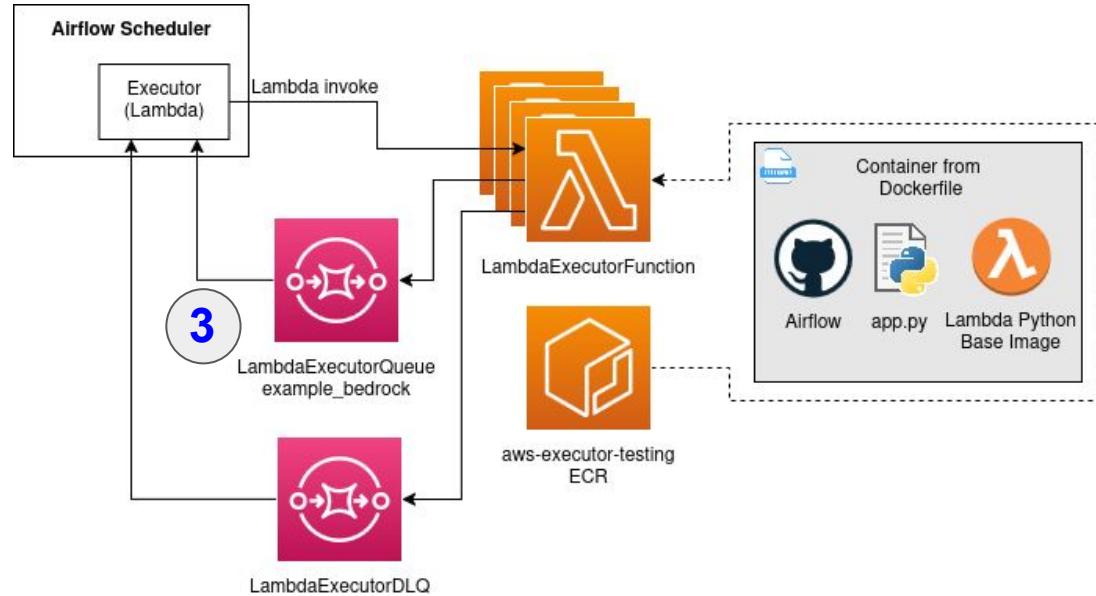
3

On Airflow Task completion, a message is sent to a results queue indicating pass/fail (SQS only as of now).

**Async Lambdas cannot be described!**

It is the responsibility of the Lambda app/handler code to send the results.

See provided example app.py in docs [here](#).



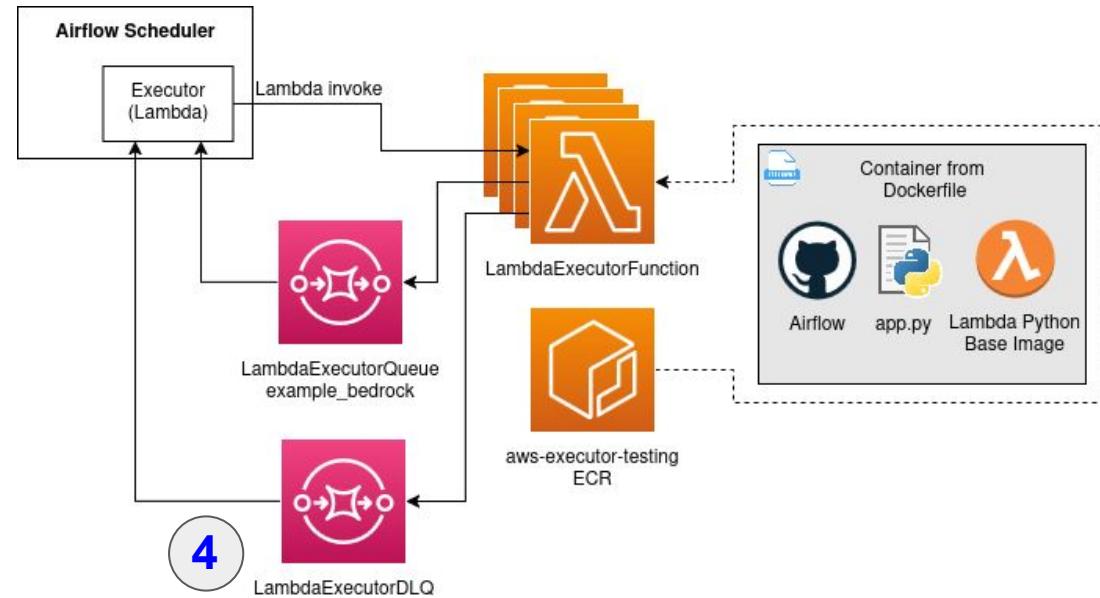


## Lambda Executor Architecture

4

The Lambda function should be configured with a Dead Letter Queue (DLQ).

If an invocation fails catastrophically (OOM, uncaught exception, etc) the Lambda function will send a message to the DLQ.

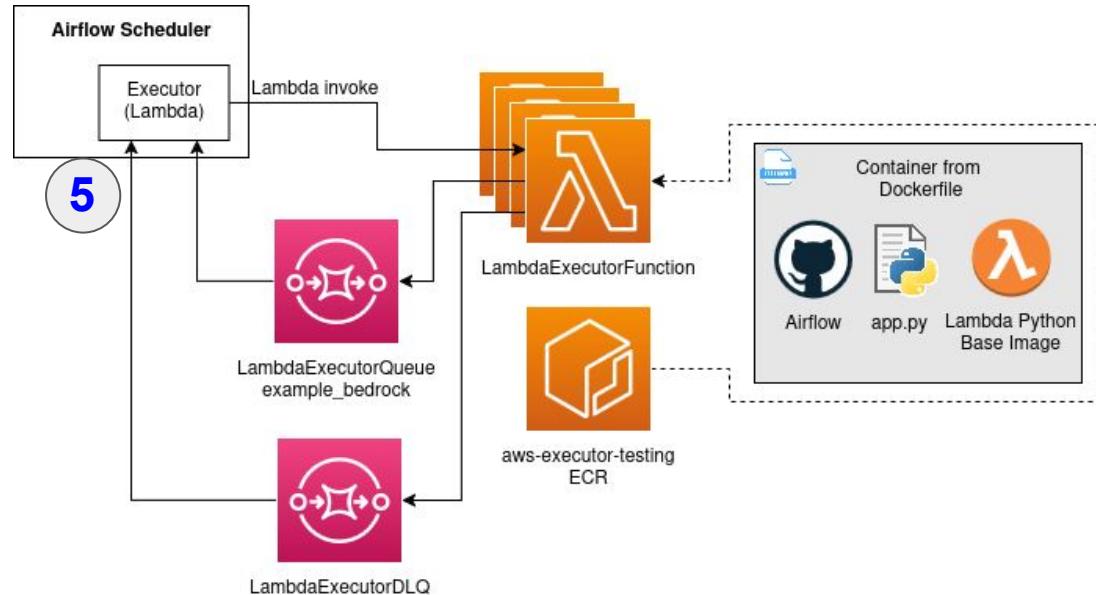


## Lambda Executor Architecture

5

The LambdaExecutor periodically reads from both SQS queues for results.

Updating Task state and communicating events back to the Scheduler.



# Local vs Lambda



Triggered hello\_world with new Run ID manual\_\_2025-10-02T00:43:23.984706+00:00, it should start any moment now.

DAG: hello\_world

Schedule: 1 day, 0:00:00

Next Run ID: 2025-10-02, 00:00:00 UTC

10 / 02 / 2025 12:43:25 AM

All Run Types

All Run States

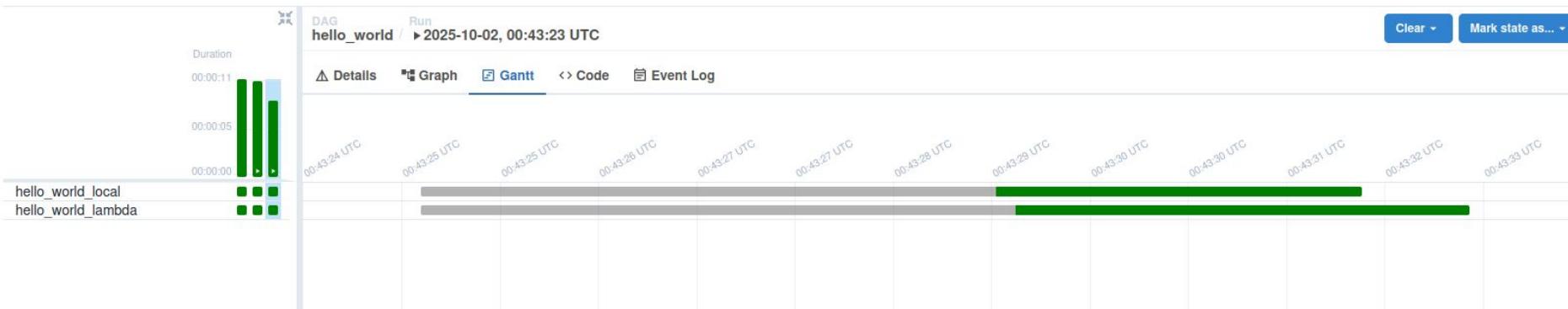
[Clear Filters](#)

Auto-refresh  25

Press **shift** + **!** for Shortcuts

deferred failed queued removed restarting running scheduled shutdown skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status

[Clear](#) [Mark state as...](#)



# Local vs Lambda



Triggered hello\_world with new Run ID manual\_\_2025-10-02T00:43:23.984706+00:00, it should start any moment now.

DAG: hello\_world

Schedule: 1 day, 0:00:00

Next Run ID: 2025-10-02, 00:00:00 UTC



10 / 02 / 2025 12:43:25 AM

All Run Types

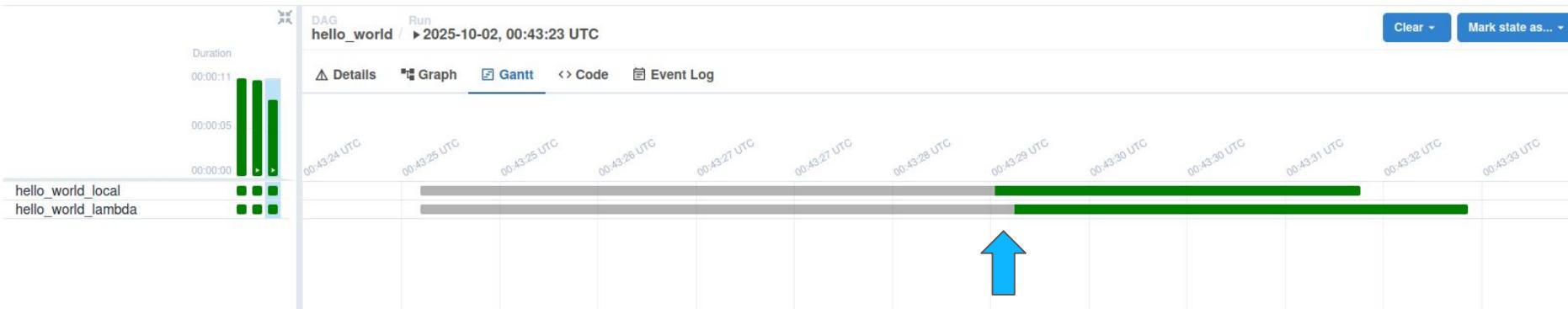
All Run States

[Clear Filters](#)

Auto-refresh  25

Press **shift** + **!** for Shortcuts

deferred failed queued removed restarting running scheduled shutdown skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status



# Local vs Lambda



Triggered hello\_world with new Run ID manual\_\_2025-10-02T00:43:23.984706+00:00, it should start any moment now.

DAG: hello\_world

Schedule: 1 day, 0:00:00

Next Run ID: 2025-10-02, 00:00:00 UTC

10 / 02 / 2025 12:43:25 AM

All Run Types

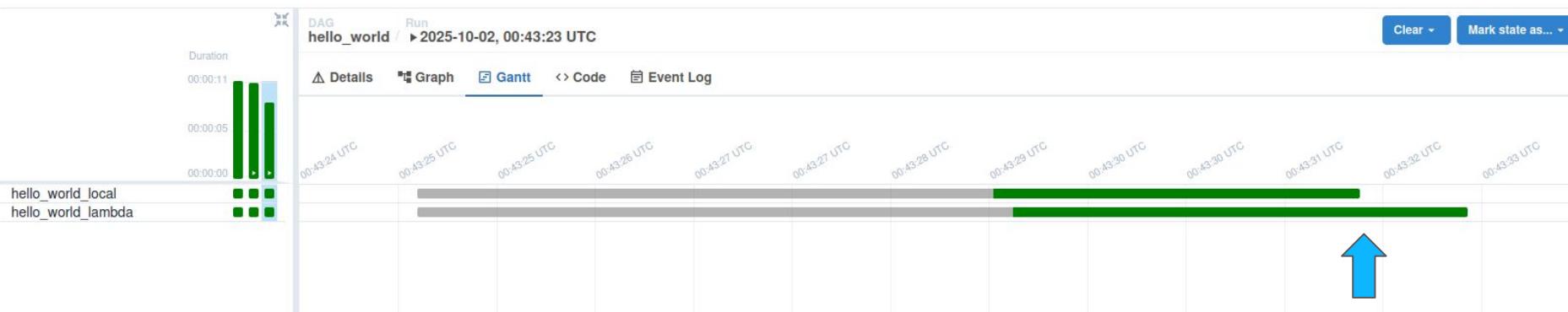
All Run States

[Clear Filters](#)

Auto-refresh  25

Press **shift** + **!** for Shortcuts

deferred failed queued removed restarting running scheduled shutdown skipped success up\_for\_reschedule up\_for\_retry upstream\_failed no\_status



# System tests: ECS vs Lambda

## ECS Executor



## Lambda Executor

### Apache Airflow - Amazon Provider Package Health

#### View the health of AWS service integrations for Apache Airflow

This live dashboard displays the current health of AWS service integrations available in the [Amazon Provider package](#) of Apache Airflow.

The following table shows data for all runs from the past 7 days of the [AWS System Tests](#) using the latest [Apache Airflow codebase](#).

The data currently being displayed reflects the tests run using the ecs executor.

[Local](#) | [ECS executor](#) **Batch executor** | [Lambda executor](#)

System name	▲	Successes	▼	Failures	▼	Duration	▼
<a href="#">example_appflow_run</a>		28		0		24 minutes	
<a href="#">example_athena</a>		27		1		33 minutes	
<a href="#">example_batch</a>		28		0		an hour	
<a href="#">example_bedrock</a>		28		0		17 minutes	
<a href="#">example_bedrock_batch_inference</a>		28		0		28 minutes	
<a href="#">example_cloudformation</a>		27		1		17 minutes	
<a href="#">example_comprehend</a>		28		0		24 minutes	
<a href="#">example_comprehend_document_classifier</a>		28		0		33 minutes	
<a href="#">example_datasync</a>		27		1		39 minutes	
<a href="#">example_dynamodb</a>		28		0		17 minutes	

### Apache Airflow - Amazon Provider Package Health

#### View the health of AWS service integrations for Apache Airflow

This live dashboard displays the current health of AWS service integrations available in the [Amazon Provider package](#) of Apache Airflow.

The following table shows data for all runs from the past 7 days of the [AWS System Tests](#) using the latest [Apache Airflow codebase](#).

The data currently being displayed reflects the tests run using the lambda executor.

[Local](#) | [ECS executor](#) | [Batch executor](#) | **Lambda executor**

System name	▲	Successes	▼	Failures	▼	Duration	▼
<a href="#">example_appflow_run</a>		20		7		5 minutes	
<a href="#">example_athena</a>		20		7		6 minutes	
<a href="#">example_batch</a>		20		7		8 minutes	
<a href="#">example_bedrock</a>		20		7		4 minutes	
<a href="#">example_bedrock_batch_inference</a>		20		7		10 minutes	
<a href="#">example_cloudformation</a>		20		7		4 minutes	
<a href="#">example_comprehend</a>		20		7		9 minutes	
<a href="#">example_comprehend_document_classifier</a>		20		7		11 minutes	
<a href="#">example_datasync</a>		20		7		6 minutes	
<a href="#">example_dynamodb</a>		20		7		4 minutes	

# System tests: ECS vs Lambda

## ECS Executor



## Lambda Executor

### Apache Airflow - Amazon Provider Package Health

#### View the health of AWS service integrations for Apache Airflow

This live dashboard displays the current health of AWS service integrations available in the [Amazon Provider package](#) of Apache Airflow.

The following table shows data for all runs from the past 7 days of the [AWS System Tests](#) using the latest [Apache Airflow codebase](#).

The data currently being displayed reflects the tests run using the ecs executor.

[Local](#) | [ECS executor](#) **Batch executor** | [Lambda executor](#)

System name	▲	Successes	▼	Failures	▼
<a href="#">example_appflow_run</a>		28		0	
<a href="#">example_athena</a>		27		1	
<a href="#">example_batch</a>		28		0	
<a href="#">example_bedrock</a>		28		0	
<a href="#">example_bedrock_batch_inference</a>		28		0	
<a href="#">example_cloudformation</a>		27		1	
<a href="#">example_comprehend</a>		28		0	
<a href="#">example_comprehend_document_classifier</a>		28		0	
<a href="#">example_datasync</a>		27		1	
<a href="#">example_dynamodb</a>		28		0	

Duration
24 minutes
33 minutes
an hour
17 minutes
28 minutes
17 minutes
24 minutes
33 minutes
39 minutes
17 minutes

### Apache Airflow - Amazon Provider Package Health

#### View the health of AWS service integrations for Apache Airflow

This live dashboard displays the current health of AWS service integrations available in the [Amazon Provider package](#) of Apache Airflow.

The following table shows data for all runs from the past 7 days of the [AWS System Tests](#) using the latest [Apache Airflow codebase](#).

The data currently being displayed reflects the tests run using the lambda executor.

[Local](#) | [ECS executor](#) | [Batch executor](#) | **Lambda executor**

System name	▲	Successes	▼	Failures	▼
<a href="#">example_appflow_run</a>		20		7	
<a href="#">example_athena</a>		20		7	
<a href="#">example_batch</a>		20		7	
<a href="#">example_bedrock</a>		20		7	
<a href="#">example_bedrock_batch_inference</a>		20		7	
<a href="#">example_cloudformation</a>		20		7	
<a href="#">example_comprehend</a>		20		7	
<a href="#">example_comprehend_document_classifier</a>		20		7	
<a href="#">example_datasync</a>		20		7	
<a href="#">example_dynamodb</a>		20		7	

Duration
5 minutes
6 minutes
8 minutes
4 minutes
10 minutes
4 minutes
9 minutes
11 minutes
6 minutes
4 minutes

# System tests: Batch vs Lambda



## Batch Executor

### Apache Airflow - Amazon Provider Package Health

#### View the health of AWS service integrations for Apache Airflow

This live dashboard displays the current health of AWS service integrations available in the [Amazon Provider package](#) of Apache Airflow.

The following table shows data for all runs from the past 7 days of the [AWS System Tests](#) using the latest [Apache Airflow codebase](#).

The data currently being displayed reflects the tests run using the batch executor.

System name	▲	Successes	▼	Failures	▼	Duration	▼
Local							
ECS executor							
Batch executor							
System name	▲	Successes	▼	Failures	▼	Duration	▼
example_appflow_run		28		0		14 minutes	
example_athena		28		0		20 minutes	
example_batch		28		0		33 minutes	
example_bedrock		28		0		10 minutes	
example_bedrock_batch_inference		28		0		21 minutes	
example_cloudformation		28		0		10 minutes	
example_comprehend		28		0		18 minutes	
example_comprehend_document_classifier		28		0		25 minutes	
example.datasync		28		0		25 minutes	
example_dynamodb		28		0		10 minutes	

## Lambda Executor

### Apache Airflow - Amazon Provider Package Health

#### View the health of AWS service integrations for Apache Airflow

This live dashboard displays the current health of AWS service integrations available in the [Amazon Provider package](#) of Apache Airflow.

The following table shows data for all runs from the past 7 days of the [AWS System Tests](#) using the latest [Apache Airflow codebase](#).

The data currently being displayed reflects the tests run using the lambda executor.

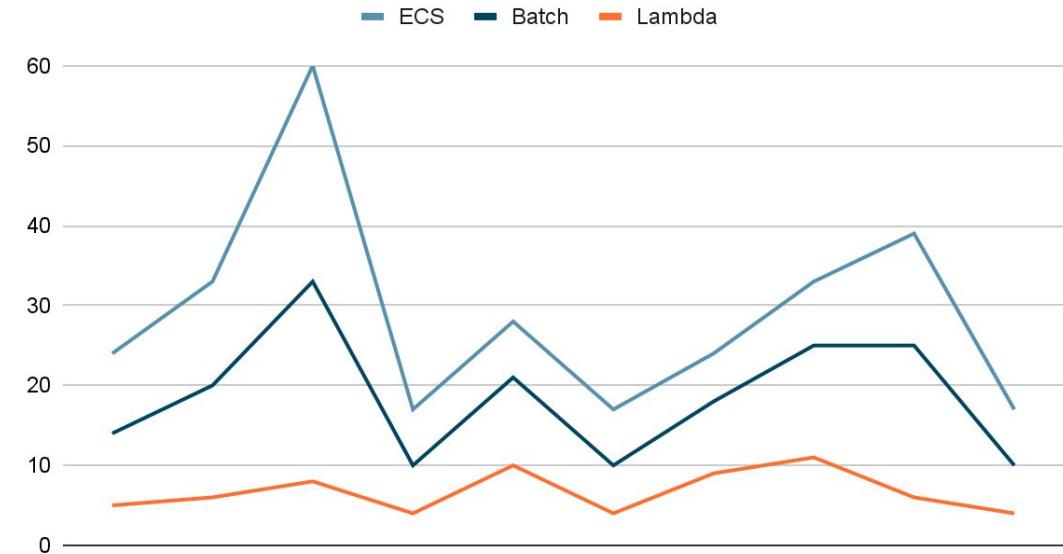
System name	▲	Successes	▼	Failures	▼	Duration	▼
Local							
ECS executor							
Batch executor							
System name	▲	Successes	▼	Failures	▼	Duration	▼
example_appflow_run		20		7		5 minutes	
example_athena		20		7		6 minutes	
example_batch		20		7		8 minutes	
example_bedrock		20		7		4 minutes	
example_bedrock_batch_inference		20		7		10 minutes	
example_cloudformation		20		7		4 minutes	
example_comprehend		20		7		9 minutes	
example_comprehend_document_classifier		20		7		11 minutes	
example.datasync		20		7		6 minutes	
example_dynamodb		20		7		4 minutes	



## ECS vs Batch vs Lambda

The same data from the previous slides plotted alongside each other

Runtime per Executor (minutes)

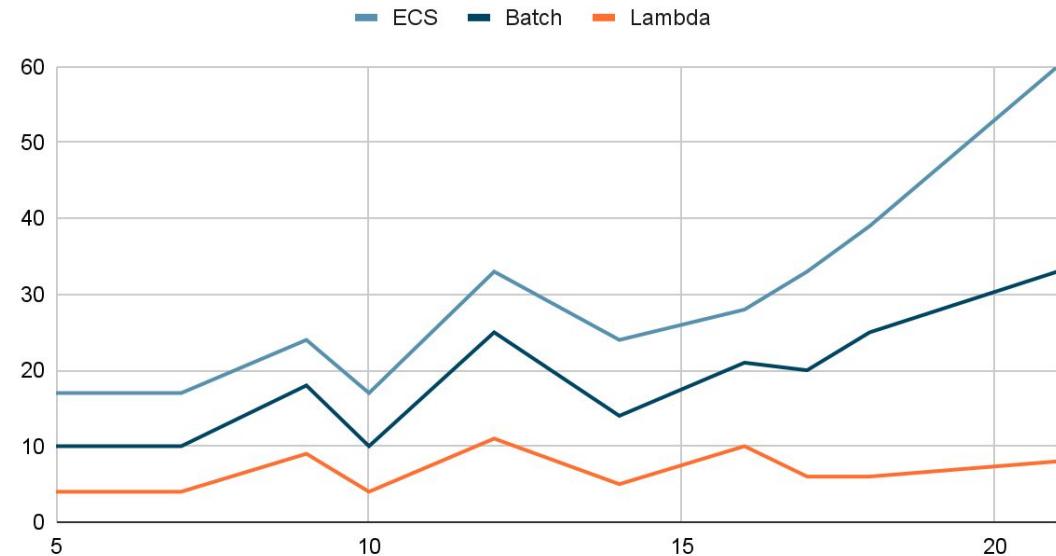




## ECS vs Batch vs Lambda

If you order the data by the number of tasks you start to see the defining factor. Startup latency!

Runtime per Executor (minutes) - Ordered by Number of Tasks

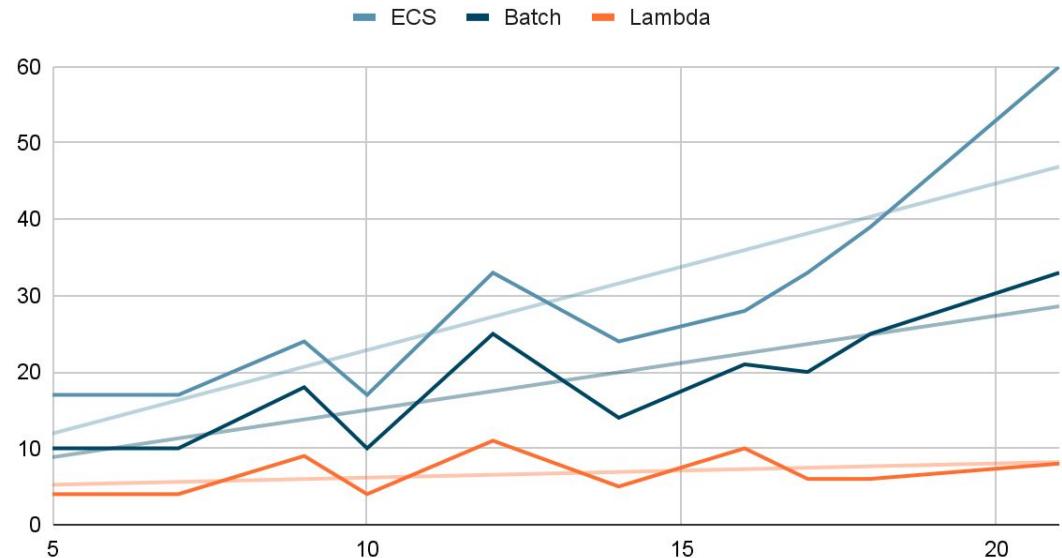




## ECS vs Batch vs Lambda

Plotting a trend, you can see Lambda runtime as tasks are added grows at a slower rate.

Runtime per Executor (minutes) - Ordered by Number of Tasks





## What about the 15min Limit?

Of all the AWS Operators we test regularly. Only 1 was not usable with the Lambda Executor, in its basic use case.

Other Operators or usage patterns may differ.

Deferrable operators are your friend!



150+

A large green circle with a thick green border. Inside the circle, the text "150+" is displayed in a large, bold, green font.

1

A large red circle with a thick red border. Inside the circle, the number "1" is displayed in a large, bold, red font.

# Not just Lambda: FaaS(t) Executors



- A new type of executor: Function as a Service (FaaS) --> FaaSt
- Distinctly different:
  - Queued/async execution pattern
  - FAST/resilient/scalable
  - Restrictive memory/storage
  - Execution time limits

# Questions?

[Link to the docs](#)

Try it out now! Supports Airflow 2 and 3:

Currently Experimental



[github.com/o-nikolas](https://github.com/o-nikolas)



[linkedin.com/in/niko-oliveira-aws](https://linkedin.com/in/niko-oliveira-aws)

