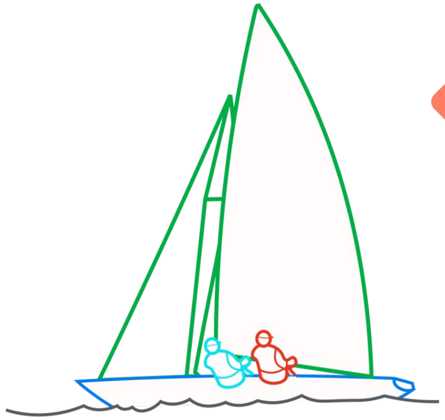
Airflow Executors Past, Present and Future

Nikolas (Niko) Oliveira



Airflow Summit
Let's flow together

September 19-21, 2023,
Toronto, Canada

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
- Spent much of the last year working on Airflow executors

Past: What is an Executor?

- Executors facilitate the running of Airflow tasks (Task Instances)
- The Airflow scheduler decides *when* a task should run and the executor decides *how*.
- Examples: CeleryExecutor, KubernetesExecutor, LocalExecutor
- Runs within the Airflow scheduler process.
- Pluggable, kind of...

Past: What is an Executor?

- 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**

Past: What is an Executor?

- 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**

Past: What is an Executor?

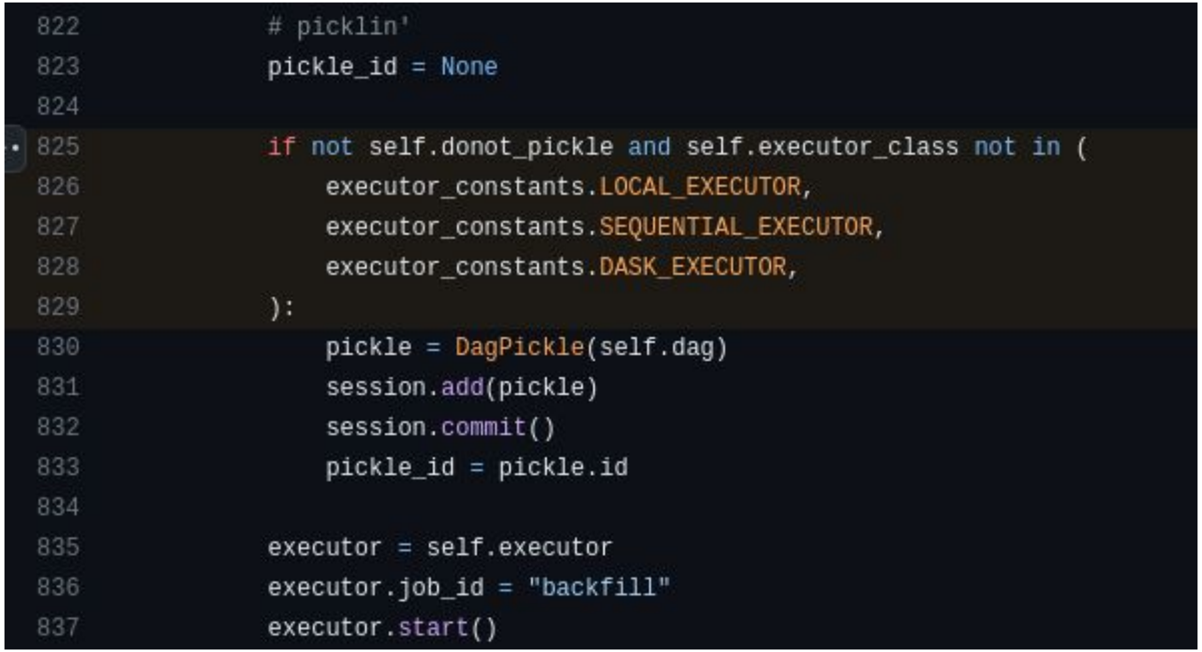
- There are many types of Airflow executors, but some major ones include:
    - Local Executors: Airflow tasks are executed on the same host that the executor (i.e. scheduler) is running on. E.g.: **LocalExecutor**
    - Remote Queued/Batched Executors: Airflow tasks are sent to a central queue where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

Past: What is an Executor?

- Airflow executors implement/inherit from the BaseExecutor class
    - This represents the public interface for executors
- It was always possible to write your own executor, however, there were some issues:
    - This interface was not strictly public in the past
    - Many executor features/use cases were baked into Airflow core code rather than the interface, we call this Executor Coupling

Past: Example of Executor Coupling

- This snippet is from the Airflow Backfill job.
- You can see the core Airflow code is hard coding executors, and must know their behaviour and implementation
- How do we fix this?



```
        # picklin'
        pickle_id = None

        if not self.donot_pickle and self.executor_class not in (
            executor_constants.LOCAL_EXECUTOR,
            executor_constants.SEQUENTIAL_EXECUTOR,
            executor_constants.DASK_EXECUTOR,
        ):
            pickle = DagPickle(self.dag)
            session.add(pickle)
            session.commit()
            pickle_id = pickle.id

        executor = self.executor
        executor.job_id = "backfill"
        executor.start()
```

Present: AIP-51 - Executor Decoupling

- AIP-51 - Described the instances of Executor Coupling in the Airflow code base as well as proposals for how to fix them

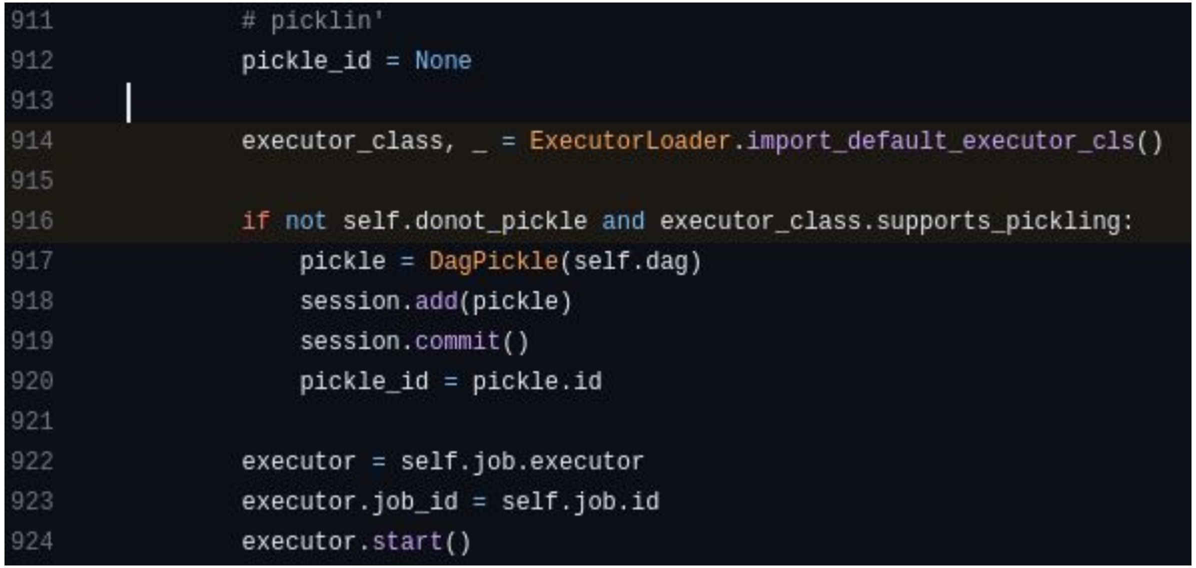
- Community effort to implement the fixes for each source of coupling

- Many couplings were simple compatibility checks, but more interesting instances included Executors vending CLI commands as well as Airflow task logs

Present: Example Executor Coupling Fix

- Pickling support is now part of the public BaseExecutor interface

- Core code no longer coupled to specific executors and interacts with a known public API



```
        # picklin'
        pickle_id = None

        executor_class, _ = ExecutorLoader.import_default_executor_cls()

        if not self.donot_pickle and executor_class.supports_pickling:
            pickle = DagPickle(self.dag)
            session.add(pickle)
            session.commit()
            pickle_id = pickle.id

        executor = self.job.executor
        executor.job_id = self.job.id
        executor.start()
```

Present: Executor Migration

- Now that the separation of Airflow executors from core Airflow is more distinct, some executors that used to live within Airflow can be moved to their own provider packages

- The CeleryExecutor and KubernetesExecutors:
  - These Executors were updated during the AIP-51 project to comply with the BaseExecutor interface
  - They have since been moved out of Airflow core to their own providers (thanks Jarek!)

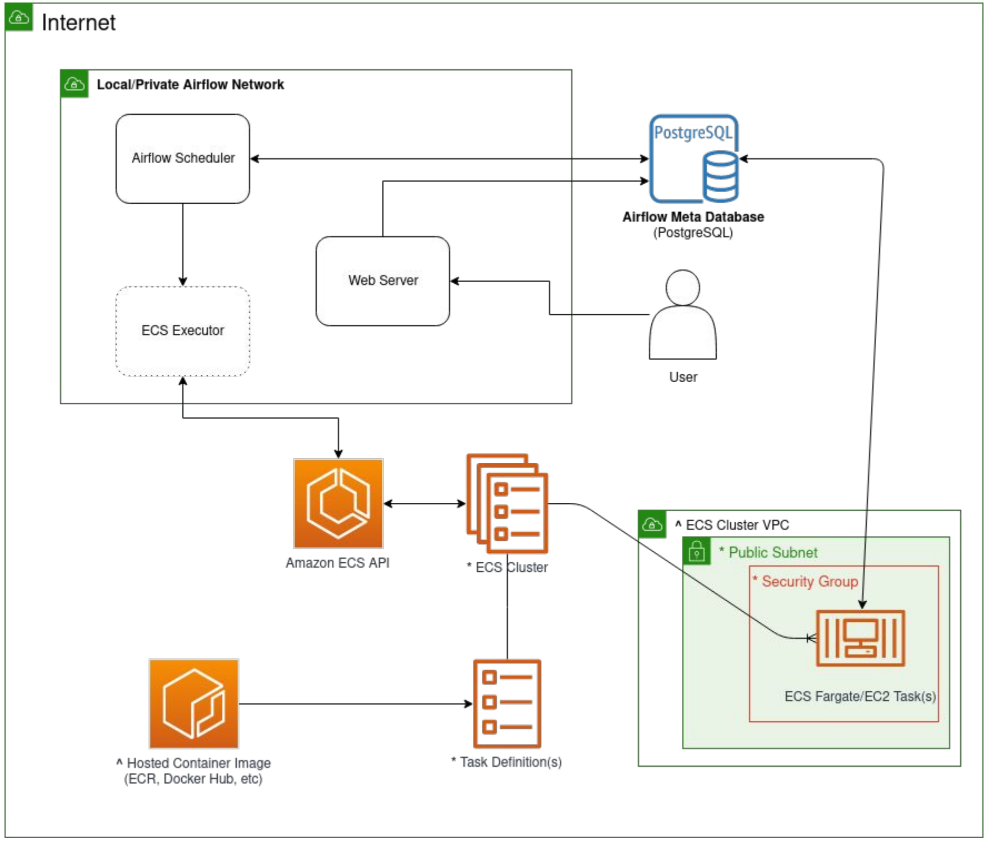
Present: Writing Your Own Executor

- Now that implementing an Airflow executor is more supported than ever, it’s easy to write your own. So that’s what we did!
- Myself and some folks from the AWS OSS Airflow team, along with an initial contribution from Ahmed Elzeiny (aelzeiny/airflow-aws-executors) have been working on a new Airflow executor that leverages AWS technology
- https://github.com/apache/airflow/pull/34381

Present: ECS Executor

- Implemented in Amazon Provider Package, leveraging the public BaseExecutor interface

- Each task that Airflow schedules for execution is run within its own ECS container



Future: More Executors!

- We have more executors on the horizon built on AWS technology that many of you will be familiar with, including:
    - AWS Batch: Queued based executor with compute backed by ECS/EC2
    - Amazon EKS: Container based executor backed by Kubernetes/EKS
    - And more to come, stay tuned!

Future: Airflow Hybrid Execution

- Airflow Executors are easier to write, and more options are arriving now and in the future, wouldn’t it be nice to leverage more than one executor at once?
- Each executor has its own pros and cons and committing to just one restricts the capabilities of any one Airflow environment
- Hardcoded hybrid executors exist (e.g. CeleryKuberentesExecutor), but are not ideal
- Expect an AIP in the near future proposing full native support for multiple executors

Questions?

github.com/o-nikolas

linkedin.com/in/niko-oliveira-aws