Executors

Name: Priyeshwar

Mail: priyesh2664@gmail.com

Executors in Apache Airflow are the mechanisms responsible for running task instances. They are **pluggable** and configurable via the executor option in the [core] section of the Airflow configuration file. Executors can be built-in, custom, or third-party, enabling flexibility depending on deployment needs.

To check the currently configured executor:

```
airflow config get-value core executor
```

Executor Types

Executors define how tasks are run — either **locally** or **remotely**.

1. Local Executors

- Run tasks **within the scheduler process**.
- **Pros**: Simple setup, low latency, minimal overhead.
- **Cons**: Limited scalability, resource sharing with scheduler.
- **Example**: LocalExecutor.

2. Remote Executors

Tasks are executed by external workers, often via queues or containers.

a. Queued/Batch Executors

- Tasks placed in a queue, processed by persistent workers.
- **Pros**: Robust, scalable, efficient for parallel workloads.
- **Cons**: Resource competition ("noisy neighbor"), potential cost inefficiency with idle workers.
- **Examples**: CeleryExecutor, BatchExecutor, EdgeExecutor (experimental).

b. Containerized Executors

- Tasks run in **isolated containers/pods**.
- **Pros**: Strong isolation, customizable environments, cost-efficient (pay-per-task).

- **Cons**: Startup latency, potentially costly for short tasks, requires container orchestration (e.g., Kubernetes).
- **Examples**: KubernetesExecutor, EcsExecutor.

Multiple Executors (Airflow 2.10+)

Airflow supports **multi-executor configurations**, enabling different executors for different workloads.

- Configured via a comma-separated list in [core].
- The **first executor** acts as the default.
- Aliases can simplify configuration.

Examples:

```
[core]
executor = LocalExecutor
executor = LocalExecutor,CeleryExecutor
executor = KubernetesExecutor,my.custom.ExecutorClass
```

Writing DAGs and Tasks with Executors

Executors can be set at **task** or **DAG** level:

```
# Task level
BashOperator(
    task_id="hello_world",
    executor="LocalExecutor",
    bash_command="echo 'hello world!'",
)


# DAG level
with DAG(
    dag_id="hello_worlds",
    default_args={"executor": "LocalExecutor"}
) as dag:
    ...
```

Monitoring

- Metrics are tracked per executor (e.g., executor.open_slots.<executor_name>).
- Logging works the same way as with single executors.

Deprecated Hybrid Executors

Static hybrids (e.g., LocalKubernetesExecutor, CeleryKubernetesExecutor) are discouraged due to maintenance issues and misuse of the queue field. Multi-executor support replaces this need.

Custom Executors

All executors must implement the BaseExecutor interface.

Mandatory Methods:

- sync: Updates task states during heartbeats.
- execute_async: Runs workloads asynchronously.

Optional Methods:

- start, end, terminate
- try_adopt_task_instances
- get_cli_commands, get_task_log

Compatibility Attributes:

- supports_pickling, supports_sentry, is_local, is_production, etc.

Workloads

A workload is the unit of execution for an executor (e.g., an Airflow task). Executors queue, execute, and monitor workloads.

CLI & Logging

- Executors can **add custom CLI commands** for setup or management.

- They can **extend task logs** by fetching logs from external execution environments (e.g., Kubernetes pod logs).

Next Steps

To use a custom executor, specify it in Airflow’s config:

[core]

executor = my_company.executors.MyCustomExecutor
