Recap

Let’s review what we’ve learned so far

Summary

- The most basic configuration which is by default is Sequential Executor with SQLite. This configuration is perfect for debugging but it does not allow to run multiple concurrent tasks limiting Apache Airflow's performances. SQLite supports an unlimited number of readers but only one writer at a time.
- Only Sequential Executors can be used with SQLite.
- The difference between concurrency and parallelism is that a concurrent system can support two or more actions in progress at the same time whereas a parallel system supports two or more actions running simultaneously at the same time (they are at the same instruction to execute).
- Do not use Sequential Executors with SQLite in production since this configuration doesn't scale.

Summary

- The second configuration is Local Executor with PostgreSQL (or MySQL). PostgreSQL is a scalable client-server database allowing concurrent connections in reading and writing.
- Local Executors execute tasks locally in parallel by using the multiprocessing Python library and queues to parallelize the execution of tasks.
- They run tasks by spawning processes in a controlled fashion in different modes on the same machine.
- In the Airflow configuration file if you change `parallelism`with
  - `0` : Unlimited parallelism, every task submitted to the Local Executor will be executed in its own process as they arrive..
  - `> 0` : Limited parallelism, spawn the number of processes equal to the value of `parallelism`at start time using a “task_queue” to coordinate the ingestion of tasks.

Summary

- The last configuration is Celery Executor with PostgreSQL (or MySQL) and RabbitMQ.
- RabbitMQ is an open source message queuing software where multiple producers send messages to a queue where those messages are pull out by consumers.
- Celery is a Python Task-Queue system that handle distribution of tasks on workers across threads or network nodes.
- Celery Executors allow to interact with Celery backend in order to distribute and execute task instances on multiple worker nodes giving a way to high availability and horizontal scaling.
- Celery needs to use a broker in order to pull out from the worker nodes the task instances to execute justifying the need of RabbitMQ.
- This configuration is greatly recommended in production because it scales very well and allow to achieve best performances.

What’s next?

In this section, we have seen how to use the three most common used executors. You are now able to use Apache Airflow in standalone, pseudo distributed and distributed mode.

In the following section we are going to see some advanced concepts in order to make your DAGs even more dynamics.