# Concepts

## Scheduler

It is a process which monitors DAG directory for changes (it look at files and detects new or modified DAGs), determines which tasks need to be ran and places them in queue for execution.

For example about how it works please refer to the ‘Example workflow – Scheduler, Executor and Worker‘ section of this documentation.

## Executor

It decides how / where to run tasks from DAGs. It prepares a task to be executed by a Worker. What executor is doing exactly depends on what executor we are using.

For example about how it works please refer to the ‘Example workflow – Scheduler, Executor and Worker‘ section of this documentation.

## Worker

Executes code from tasks from DAGs.

For example about how it works please refer to the ‘Example workflow – Scheduler, Executor and Worker‘ section of this documentation.

## Example workflow – Scheduler, Executor and Worker

Example workflow then Celery executor is used:

* Scheduler sees tasks’ schedules and based on that, at a proper time, it asks Executor to execute a task.
* Executor puts the task in a queue (Celery queue).
* Workers (Celery workers) sees that task in a queue and pick it up.

Example workflow then Kubernetes executor is used:

* Scheduler sees tasks’ schedules and based on that, at a proper time, it asks Executor to execute a task.
* Executor tells Kubernetes to start a Pod (Worker) for this task.
* Kubernetes creates a Pod.
* Pod runs a task.

## Web server

Provides a web application for monitoring DAGs and performing different actions on them (starting, stopping).

## Metadata database

Database (usually PostgreSQL) with information about DAGs, tasks, job statuses etc.

## DAGs

DAGs are written in Python and they define workflows, that is what tasks will be executed in which order.

# Airflow deployment

There are a few options for deploying Airflow:

* Azure Data Factory
* Google Cloud Composer
* MWAA
* Astronomer
* Docker / Kubernetes

Azure Data Factory, Google Cloud Composer and MWAA are simple to use but there is a lot of limitations, you can’t modify too much the environment in which Airflow is running. For example you can’t install additional tools needed for Airflow dags like database drivers.

Astronomer offers much more flexibility. You can modify there Docker images which prepares environment in which Airflow is running.

Ranking of solutions according to flexibility:

1. VM
2. Docker / Kubernetes serverless services
3. Astronomer
4. ADF / GCC / MWAA

Ranking of solutions according to simplicity of usage:

1. ADF / GCC / MWAA
2. Astronomer
3. Docker / Kubernetes serverless services
4. VM

# Materials to learn from

* [Why Airflow? The Top 5 Reasons To Use It!](https://www.youtube.com/watch?v=vEApEfa8HXk&list=PL79i7SgJCJ9hf7JgG3S-3lOpsk2QCpWkD) - playlist
* [Apache Airflow Overview | Architecture | What is DAG | Tasks | Operators | Use Cases](https://www.youtube.com/watch?v=s6PgXq-SO4I&list=PLc2EZr8W2QIAI0cS1nZGNxoLzppb7XbqM) - playlist
* [Airflow Tutorial for Beginners - Full Course in 2 Hours 2022](https://www.youtube.com/watch?v=K9AnJ9_ZAXE&list=PLwFJcsJ61oujAqYpMp1kdUBcPG0sE0QMT)
* [Tutorials — Airflow Documentation](https://airflow.apache.org/docs/apache-airflow/stable/tutorial/index.html)

# PoCs

## MS SQL - Data Lake data ingestion

I prepared one PoC about how to orchestrate a Python script which is ingesting data incrementally from a MS SQL server into an Azure Data Lake using Airflow.

There is the PowerPoint presentation about that and videos in the ‘Airflow Data Lake data ingestion’ folder in the same folder as this document. It describes the on a high level what this PoC is about.

Links to repositories with code:

* <https://github.com/bulka4/airflow_docker> – Airflow application
* <https://github.com/bulka4/azure_terraform> – Terraform code for creating the Azure MS SQL db and Azure Data Lake
* <https://github.com/bulka4/data_lake_data_ingestion> - Code for ingesting data into the Azure MS SQL db and creating container in the Data Lake