What is a DAG?

The concept to understand

Definition

In mathematics and computer science, a Directed Acyclic Graph (DAG), is a finite directed graph with no directed cycles. That is, it consists of finitely many vertices and edges, with each edge directed from one vertex to another, such that there is no way to start at any vertex **v** and follow a consistently-directed sequence of edges that eventually loops back to **v** again.

Equivalently, a DAG is a directed graph that has a topological ordering, a sequence of the vertices such that every edge is directed from earlier to later in the sequence.

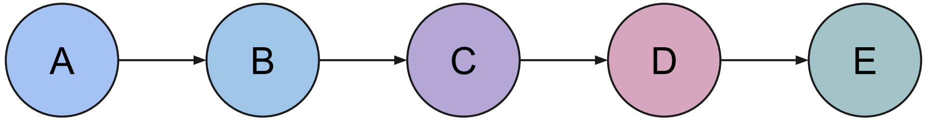
Wikipedia

In a More Concise Way

- A DAG (Directed Acyclic Graph) is a finite directed graph that doesn’t have any cycles (loops).
  - A cycle is a serie of vertices that connect back to each other making a loop.
- In Apache Airflow, a DAG represents a collection of tasks to run, organized in a way that represent their dependencies and relationships.
  - Its job is to make sure that tasks happen at the right time in the right order with the right handling of any unexpected issues.
  - It ultimately defines our Workflow.
- Each node is a Task.
- Each edge is a Dependency.

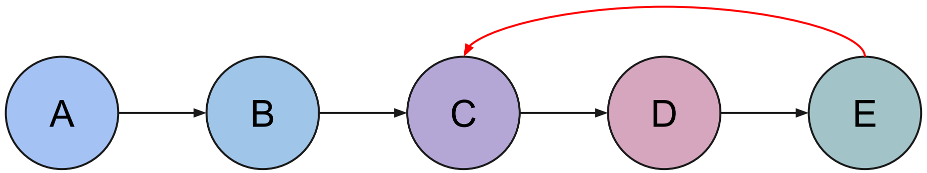
An Example

- Here a very simple DAG example where we could imagine that
  - Node A could be downloading data
  - Node B could be sending that data for processing
  - Node C could be monitoring the data processing
  - Node D could be generating a report
  - Node E could be sending an email to the DAG's owner


$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \arrow[r] & D \arrow[r] & E \end{tikzcd}$$

An Example

- This is not a DAG because there is a cycle.



$$\begin{tikzcd} A \arrow[r] & B \arrow[r] & C \arrow[r] & D \arrow[r] & E \arrow[ll, bend right] \end{tikzcd}$$

Important Properties

- DAGs are defined in Python files placed into Airflow's DAG_FOLDER (usually ~/airflow/dags).
- `dag_id` serve as a unique identifier for your DAG.
- `description` the description of your DAG
- `start_date` tell when your DAG should start.
- `schedule_interval` define how often your DAG runs.
- `depend_on_past` run the next DAGRun if the previous one completed successfully.
- `default_args` a dictionary of variables to be used as constructor keyword parameter when initialising operators.

Coding Time!

Enough theory, let’s start coding your first DAG!