**What is Apache Airflow**

Apache Airflow is an open-source tool to programmatically author, schedule, and monitor workflows. It is used by Data Engineers for orchestrating workflows or pipelines. One can easily visualize your data pipelines' dependencies, progress, logs, code, trigger tasks, and success status. Complex data pipelines are managed using it. These data pipelines are used to deliver datasets that are easily used by business intelligence applications or machine learning models where a huge amount of data is required. It is one of the most robust platforms for data engineers. Batch-oriented workflows are developed, scheduled, and monitored efficiently. Apache Airflow is a workflow engine that  easily schedules and runs  complex data pipelines.

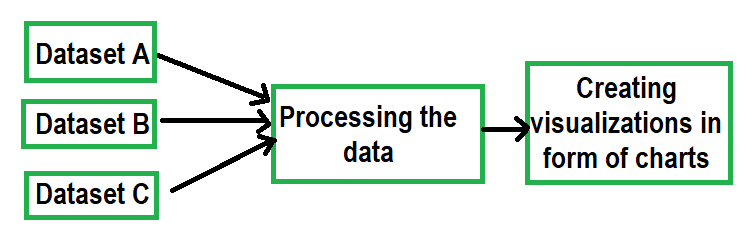
**Working of Airflow and DAG:**

Workflow refers to the process of achieving some goal. They always have an end goal which could be something like creating visualizations for some data as given here. Directed Acyclic Graphs (abbreviated as DAG) are used to represent the workflow.

A diagram of a network

AI-generated content may be incorrect.

In the above-directed graph, if we traverse along the direction of the edges, and find no closed loop, we can conclude that no directed cycles are present. This type of graph is called a directed acyclic graph.



**Apache Airflow Concepts**

Apache Airflow is a platform for programmatically authoring, scheduling, and monitoring workflows. The key concepts include **DAGs, Tasks, Operators, Scheduler, and Executor**.

A **DAG (Directed Acyclic Graph)** is a collection of all the tasks you want to run, organized in a way that reflects their relationship and dependencies. DAGs are defined using Python code and are stored in the DAGs directory. The Airflow scheduler looks for files with "dag" or "airflow" strings and parses all the DAGs at regular intervals.

A **Task** is a unit of work within a DAG. Tasks can be Operators, Sensors, or SubDags. Each task is represented as a node in the DAG.Tasks have dependencies on other tasks, and these dependencies define the order in which tasks execute.

**An Operator is a predefined task template that provides integration with external services**. Examples include the BashOperator, PythonOperator, and MySQLOperator. Operators allow you to execute commands, run SQL queries, and interact with various services.2

**The Scheduler is responsible for triggering the execution of tasks based on their defined schedules**. It monitors all tasks and DAGs, then triggers the task instances once their dependencies are complete. The scheduler uses the configured Executor to run tasks that are ready.

**The Executor handles the execution of tasks**. In the default Airflow installation, this runs everything inside the scheduler, but most production-suitable executors push task execution out to workers. The executor interacts with the scheduler and the webserver to manage task execution.

