**Airflow Architecture Components**

**DAG (Directed Acyclic Graph)**

● A DAG defines a workflow: a collection of tasks and their dependencies.

● Tasks in a DAG are organized in a way that avoids circular dependencies.

**Scheduler**

● Responsible for orchestrating the execution of workflows by determining when tasks should run.

● Periodically checks the DAGs to identify tasks that are ready to execute.

● Queues tasks for execution and ensures task dependencies are met.

. **Executor**

● Executes the tasks scheduled by the Scheduler.

● Determines how and where tasks are executed.

● Common Executors:

○ SequentialExecutor: Runs one task at a time (for testing and small-scale setups).

○ LocalExecutor: Runs multiple tasks in parallel on the same machine.

○ CeleryExecutor: Distributes tasks to multiple worker nodes for parallel execution.

○ KubernetesExecutor: Runs tasks in Kubernetes pods, offering scalability and isolation.

**Worker**

●Executes the actual task (e.g., running Python code, querying a database, or making API calls).

● Workers can be distributed across multiple nodes to handle tasks in parallel.

**Task**:

* basic unit of execution

**Convert Local Time to UTC:**

import pendulum

local\_time = pendulum.datetime(2024, 2, 25, 12, 0, tz="Asia/Kolkata") # IST Time

utc\_time = local\_time.in\_tz("UTC")

print(utc\_time) # Output: 2024-02-25T06:30:00+00:00 (Converted to UTC)

—------------------------------------------------------------------------------------------------

# Get current time in

UTC utc\_time = pendulum.now("UTC")

# Convert UTC time to a different time zone, e.g., 'America/New\_York'

ny\_time = utc\_time.in\_timezone("America/New\_York")

—-------------------------------------------------------------------------------------

**start\_date** determines when the DAG starts running.

**end\_date** defines when the DAG will stop running.

**catchup=False** Airflow will run all missed DAG runs from the start\_date up to the current date, even if you haven't manually triggered those runs.

**catchup=True** Airflow will not backfill missed runs and will only run for the current schedule.

**Mode**:

reschedule: The task is rescheduled when not finished instead of being retried.

poke: The task keeps poking for a condition at intervals.

**replaceWhere:**

df.write.partitionBy("year","month","day")\

.format("delta")\

.mode("overwrite")\

.option("mergeSchema", "true")\

.option("overwriteSchema","True")\

.option("replaceWhere",x)\

.option("path",path)\

.saveAsTable("{}.{}".format(database\_name, table\_name))