APACHE AIRFLOW

INTRODUCTION TO DATA WAREHOUSING AND DATA LAKES

DATA WAREHOUSING:

- **DEFINITION:** A data warehouse is a centralized repository for storing large volumes of structured data from multiple sources. It is designed for query and analysis rather than transaction processing.
- **CHARACTERISTICS:**
    - Structured data
    - Schema-on-write
    - Optimized for read-heavy operations
    - Supports complex queries and reporting
- **USE CASES:** Business intelligence, reporting, and data analysis.

DATA LAKES:

- **DEFINITION:** A data lake is a storage repository that holds a vast amount of raw data in its native format until it is needed. It can store structured, semi-structured, and unstructured data.
- **CHARACTERISTICS:**
    - Schema-on-read
    - Can handle large volumes of data
    - Supports a variety of data types
    - Flexible and scalable
- **USE CASES:** Big data analytics, machine learning, and data exploration.

DESIGNING DATA WAREHOUSING FOR AN ETL DATA PIPELINE

STEPS:

1. **REQUIREMENT ANALYSIS:** Understand the business requirements and data sources.
2. **DATA MODELING:** Design the schema (e.g., star schema, snowflake schema).
3. **ETL PROCESS DESIGN:**
    - **EXTRACTION:** Identify and extract data from various sources.
    - **TRANSFORMATION:** Cleanse, transform, and aggregate data.
    - **LOADING:** Load the transformed data into the data warehouse.

4. **DATA INTEGRATION:** Integrate data from different sources to provide a unified view.
5. **PERFORMANCE OPTIMIZATION:** Optimize for query performance and storage efficiency.
6. **SECURITY AND GOVERNANCE:** Implement security measures and data governance policies.

DESIGNING DATA LAKES FOR AN ETL DATA PIPELINE

STEPS:

1. **REQUIREMENT ANALYSIS:** Understand the types of data and business needs.
2. **DATA INGESTION:** Ingest data from various sources in its raw format.
3. **DATA STORAGE:** Store data in a scalable and cost-effective manner (e.g., HDFS, S3).
4. **DATA CATALOGING:** Implement a data catalog to manage metadata and data lineage.
5. **DATA PROCESSING:** Use tools like Apache Spark or Hadoop for data processing.
6. **DATA ACCESS:** Provide mechanisms for data access and querying (e.g., Presto, Hive).
7. **SECURITY AND GOVERNANCE:** Implement security measures and data governance policies.

ETL VS ELT

ETL (EXTRACT, TRANSFORM, LOAD):

- **PROCESS:**
    1. **EXTRACT:** Data is extracted from source systems.
    2. **TRANSFORM:** Data is transformed into the desired format.
    3. **LOAD:** Transformed data is loaded into the target system (e.g., data warehouse).
- **USE CASES:** Traditional data warehousing, where transformation is done before loading.

ELT (EXTRACT, LOAD, TRANSFORM):

- **PROCESS:**
    1. **EXTRACT:** Data is extracted from source systems.
    2. **LOAD:** Raw data is loaded into the target system (e.g., data lake).
    3. **TRANSFORM:** Data is transformed within the target system.
- **USE CASES:** Modern data lakes and big data environments, where transformation is done after loading.

FUNDAMENTALS OF AIRFLOW

OVERVIEW:

- **DEFINITION:** Apache Airflow is an open-source platform to programmatically author, schedule, and monitor workflows.
- **ARCHITECTURE:**
    - **DAGs (Directed Acyclic Graphs):** Define the workflow.
    - **OPERATORS:** Define individual tasks.
    - **SCHEDULER:** Schedules the tasks.
    - **EXECUTOR:** Executes the tasks.
    - **METADATA DATABASE:** Stores the state of the tasks and workflows.
    - **WEB INTERFACE:** Provides a user interface to monitor and manage workflows.

KEY CONCEPTS:

- **TASKS:** The basic unit of execution.
- **DAG:** A collection of tasks with dependencies.
- **OPERATORS:** Define what each task does (e.g., BashOperator, PythonOperator).
- **SENSORS:** Wait for a certain condition to be met before executing a task.

WORK MANAGEMENT WITH AIRFLOW

FEATURES:

- **SCHEDULING:** Schedule tasks to run at specific intervals.
- **MONITORING:** Monitor task execution and workflow status.
- **LOGGING:** Centralized logging for all tasks.
- **ALERTING:** Set up alerts for task failures or retries.
- **RETRY MECHANISM:** Automatically retry failed tasks.
- **TASK DEPENDENCIES:** Define dependencies between tasks to control execution order.

AUTOMATING AN ENTIRE DATA PIPELINE WITH AIRFLOW

STEPS:

1. **DEFINE THE DAG:** Create a DAG to represent the workflow.
2. **DEFINE TASKS:** Use operators to define tasks (e.g., data extraction, transformation, loading).

3. **SET TASK DEPENDENCIES:** Define the order in which tasks should be executed.
4. **SCHEDULE THE DAG:** Set the schedule interval for the DAG.
5. **MONITOR AND MANAGE:** Use the Airflow web interface to monitor and manage the workflow.
6. **HANDLE ERRORS AND RETRIES:** Implement error handling and retry mechanisms.
7. **OPTIMIZE PERFORMANCE:** Optimize task execution and resource usage.

EXAMPLE DAG:

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta

default_args = {
  'owner': 'airflow',
  'depends_on_past': False,
  'start_date': datetime(2023, 1, 1),
  'email_on_failure': False,
  'email_on_retry': False,
  'retries': 1,
  'retry_delay': timedelta(minutes=5),
}

def extract_data():
  # Code to extract data
  pass

def transform_data():
  # Code to transform data
  pass
```

```
def load_data():
  # Code to load data
  pass

dag = DAG(
  'etl_pipeline',
  default_args=default_args,
  description='An ETL data pipeline',
  schedule_interval=timedelta(days=1),
)

extract_task = PythonOperator(
  task_id='extract_data',
  python_callable=extract_data,
  dag=dag,
)

transform_task = PythonOperator(
  task_id='transform_data',
  python_callable=transform_data,
  dag=dag,
)

load_task = PythonOperator(
  task_id='load_data',
  python_callable=load_data,
  dag=dag,
)
```

```
extract_task >> transform_task >> load_task
```