Project Title	**Full-Stack On-Premise Data Pipeline for IoT & Weather Data**
Skills take away From This Project	**Python,SQL,MongoDB,Hive,Spark,Sqoop,Kakfa, Airflow,Docker,Grafanna**
Domain	

Problem Statement:

You are tasked with building an on-premise data engineering pipeline for a hypothetical weather analytics company.

Your system must:

- **Ingest real-time weather data** from a public API and send it to a **Kafka** topic.
- **Generate synthetic data** using **Faker** and save it into a **CSV file** every minute.
- **Mock database records** using Faker and **insert them into a MySQL table**.
- Use **Apache Spark** for:
    - **Streaming**: Consume Kafka topic and save as **Parquet** files every 5 minutes.
    - **Batch Processing**: Join CSV file + MySQL table, perform ETL, and load into **Hive** and a final **MySQL** table.
- **Store and manage data** in:
    - Hive Tables
    - MySQL Tables
    - Parquet files
- **Orchestrate** workflows using **Apache Airflow**.
- **Monitor** pipelines using **Grafana + Prometheus**.
- **Containerize** the full system using **Docker Compose**.(Optional)

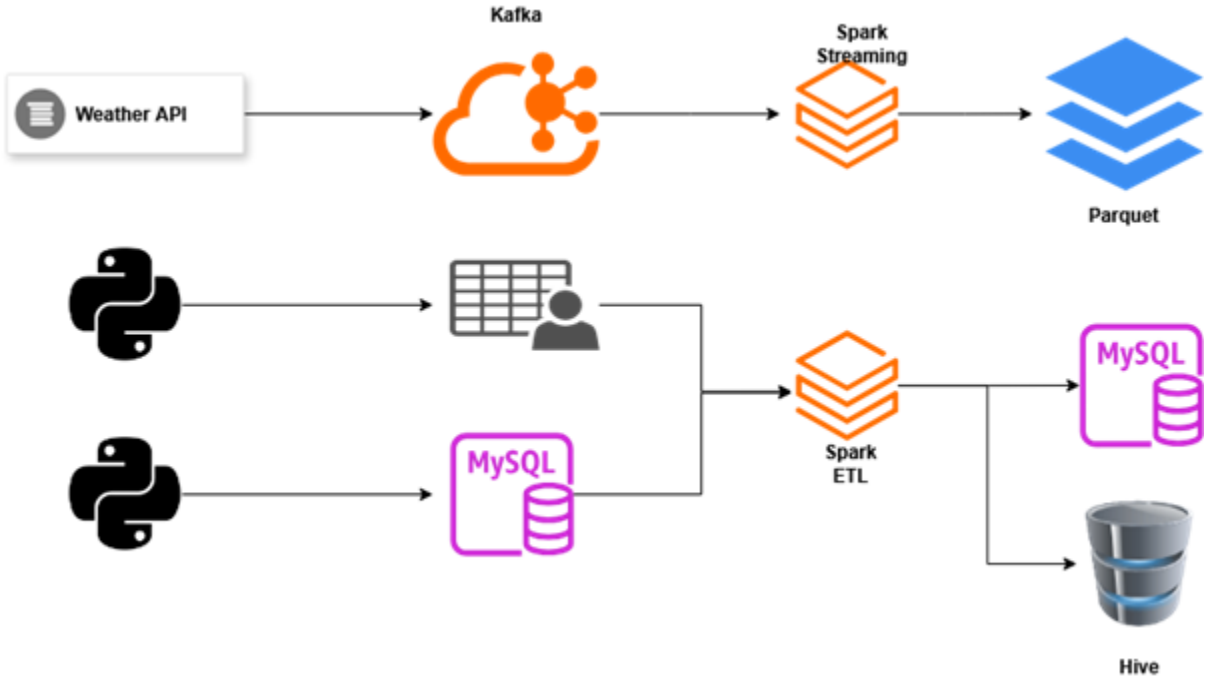
Business Use Cases:

You are tasked with building an on-premise data engineering pipeline for a hypothetical weather analytics company.

Your system must:

- **Ingest real-time weather data** from a public API and send it to a **Kafka** topic.
- **Generate synthetic data** using **Faker** and save it into a **CSV file** every minute.
- **Mock database records** using Faker and **insert them into a MySQL table**.
- Use **Apache Spark** for:
    - **Streaming**: Consume Kafka topic and save as **Parquet** files every 5 minutes.
    - **Batch Processing**: Join CSV file + MySQL table, perform ETL, and load into **Hive** and a final **MySQL** table.
- **Store and manage data** in:
    - Hive Tables
    - MySQL Tables
    - Parquet files
- **Orchestrate** workflows using **Apache Airflow**.
- **Monitor** pipelines using **Grafana + Prometheus**.
- **Containerize** the full system using **Docker Compose**.(Optional)

Approach:



3.1 Ingestion

- **Weather API to Kafka:**
  - **Use Python to pull data from OpenWeatherMap API every minute.**
  - **Push JSON data to Kafka Topic: `weather-topic`.**
  - **Schedule using Airflow DAG: `weather_to_kafka_dag.py`.**
- **Faker to CSV:**
  - **Generate fake user weather logs (e.g., Name, City, Temperature) using Faker Python library.**
  - **Save the data as a CSV file every minute.**
- **Mock MySQL Data:**
  - **Insert fake sensor/device data into MySQL database using Python script.**

3.2 Processing

- **Spark Streaming:**
  - **Stream data from Kafka Topic.**
  - **Write the streamed data as Parquet files (every 5 minutes).**
- **Spark Batch ETL:**

- Read CSV files and MySQL table.
- Perform transformations (e.g., Join, Filter, Select).
- Save output:
  - Into a Hive Table (`final_table`).
  - Into a MySQL Final Table (`final_table`).

3.3 Loading & Storage

- Save Kafka-processed data into Parquet format in a specified directory.
- Load ETL processed data into:
  - Hive Table (`final_table`)
  - MySQL Final Table (`final_table`)
- Export the Hive Table to CSV using Spark.

3.4 Orchestration and Monitoring

- Use Apache Airflow to orchestrate:
  - Weather API ingestion
  - Batch ETL job triggers
- Use Grafana to visualize health of:
  - Airflow jobs (optional)
  - Docker containers (optional)
  - Spark Applications (optional)

Technologies and Tools

Technology	Purpose
Apache Kafka	Real-time ingestion
Apache Spark (Streaming & Batch)	Processing
Apache Hive	Data Lake Table
MySQL	Relational Storage
Apache Airflow	Workflow Orchestration
Docker Compose	Infrastructure Setup
Grafana + Prometheus	Monitoring

Results:

Each student must submit:

1. **Project Code Repository**:
    - Structured folders for Airflow, Spark jobs, Kafka producers, Faker scripts, Docker Compose files.
2. **Airflow DAGs**:
    - `weather_to_kafka_dag.py`
    - `batch_etl_dag.py`
3. **Spark Jobs**:
    - Streaming Job: `streaming_kafka_to_parquet.py`
    - Batch ETL Job: `batch_etl.py`
4. **Docker Compose File**: (Optional)
    - Must include Kafka, MySQL, Hive, Spark, Airflow, Grafana, Prometheus.
5. **SQL Scripts** (if any Hive table creation)
6. **Monitoring Dashboard JSON** (Grafana dashboard exported) (Optional)
7. **README.md** explaining the setup and running steps

Project Evaluation metrics:

Technology	Purpose
Weather API to Kafka Working	10
Faker CSV Generator Working	10
Mock MySQL Data Loader	10
Spark Streaming to Parquet	15
Spark Batch ETL (Join, Transform, Load)	15
Hive Table and Export to CSV	15
MySQL Final Table Data Loaded Correctly	10
Airflow DAGs Scheduled Correctly	10
README Documentation	5

Dockerized End-to-End Deployment	+5 Bonus Points

Data Set:

Weather API Data

- Source: OpenWeatherMap API
- Frequency: Collected every minute via Airflow and pushed to AWS Kinesis

Local Sensor Simulation File

- Source: A locally generated `.csv` file simulating IoT sensor readings
- Frequency: Generated every minute by a background Python job
- Destination: Uploaded to S3 Bucket (`weather-data-ingestion-bucket`)

MySQL Weather Records (Historical)

- Source: Simulated historical weather data stored in a local MySQL database
- Migrated to: AWS RDS MySQL for further ETL processing

Data Set Explanation:

1. OpenWeatherMap API Data

- **Purpose**: Real-time ingestion of weather metrics such as temperature, humidity, wind speed, pressure, etc.
- **Fields Collected**:
    - `timestamp`, `city`, `latitude`, `longitude`, `temperature`, `humidity`, `pressure`, `wind_speed`, `weather_description`
- **Format**: JSON response → Converted to structured format before pushing to Kinesis
- **Preprocessing**:
    - Convert timestamps to UTC
    - Normalize nested weather fields (e.g., `weather[0].description` → `weather_description`)
    - Ensure schema consistency before Kinesis ingestion

---

2. Local File Data (Simulated IoT Sensor)

- **Purpose**: Emulates edge-device file drops from weather stations
- **Fields Collected**:
    - `sensor_id`, `location`, `timestamp`, `temp_celsius`, `humidity_percent`, `rain_mm`
- **Format**: CSV
- **Preprocessing**:

- Rename headers to match standard schema
- Validate numeric values
- Move to a staging S3 bucket using a Python uploader job

---

3. Local MySQL → AWS RDS MySQL Historical Data

- **Purpose**: Simulates backfilled data from previous months or years
- **Fields**:
  - `station_id`, `date`, `min_temp`, `max_temp`, `avg_humidity`, `precipitation`
- **Format**: Relational schema (SQL table)
- **Preprocessing**:
  - Data migration via Airflow → Docker MySQL → AWS RDS MySQL
  - Deduplicate using date + station ID
  - Cleansed using Pandas + stored in Hive-compatible CSV format

Project Deliverables:

1. **GitHub Repository**
   - Push your final project to your GitHub account.
   - Repository must be private during the assignment period.
   - Add the instructor's GitHub ID as a collaborator for review.
2. **Naming Convention**
   - GitHub Repo Name: `onprem-data-pipeline-{yourname}`
   - Example: `onprem-data-pipeline-amit`

Project Guidelines:

1. **GitHub Repository**
    - Push your final project to your GitHub account.
    - Repository must be private during the assignment period.
    - Add the instructor's GitHub ID as a collaborator for review.
2. **Naming Convention**
    - GitHub Repo Name: `onprem-data-pipeline-{yourname}`
    - Example: `onprem-data-pipeline-amit`

Timeline:

Define the project timeline, including milestones and deadlines.

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.
Note: Book the slot at least before 12:00 Pm on the same day

Timing: Monday-Saturday (4:00PM to 5:00PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

For DE/BADM project/class topic doubt slot clarification session:

Booking link : https://forms.gle/NtkQ4UV9cBV7Ac3C8

Session timing:

For DE: 04:00 pm to 5:00 pm every saturday
For BADM 05:00 to 07:00 pm every saturday

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.
Note: This form will Open only on Saturday (after 2 PM) and Sunday on Every Week

Timing:

For BADM and DE
Monday-Saturday (11:30AM to 1:00PM)

For DS and AIML
Monday-Saturday (05:30PM to 07:00PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurRA