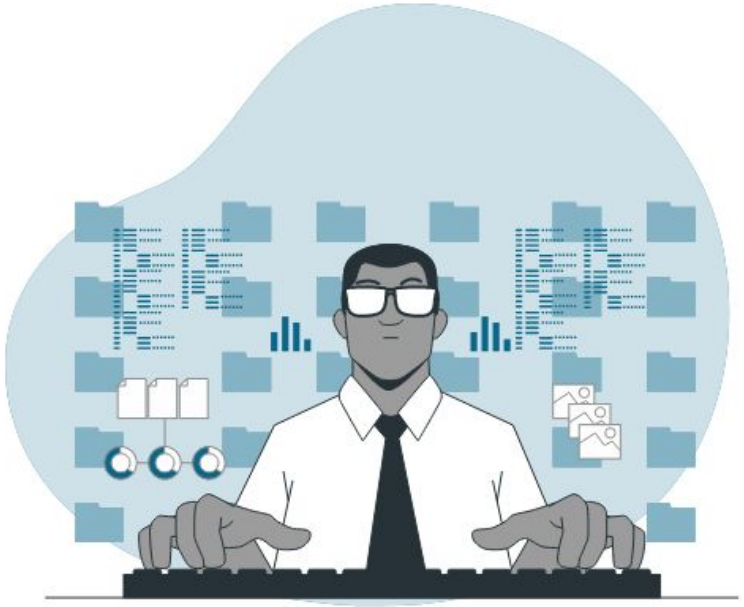
Create A Data Pipeline based on Messaging Using PySpark and Airflow

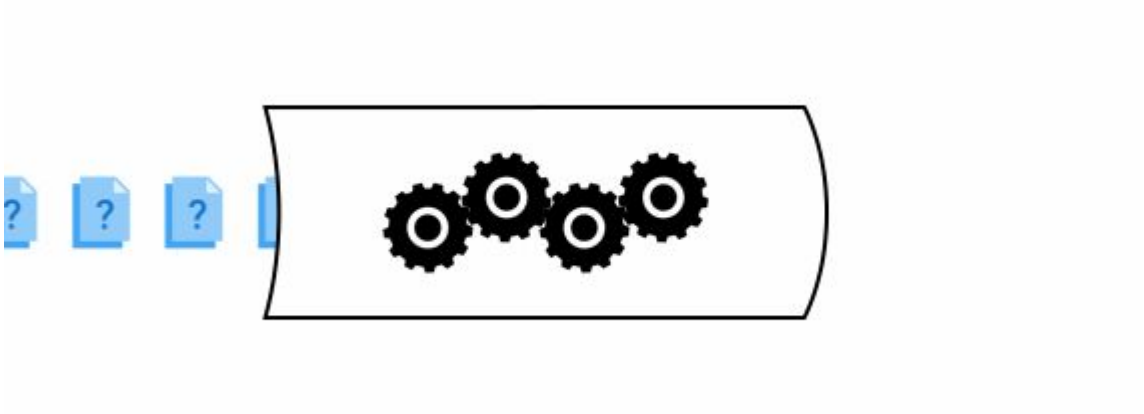
Welcome

Create A Data Pipeline based on Messaging Using PySpark and Airflow

What is Data Engineering



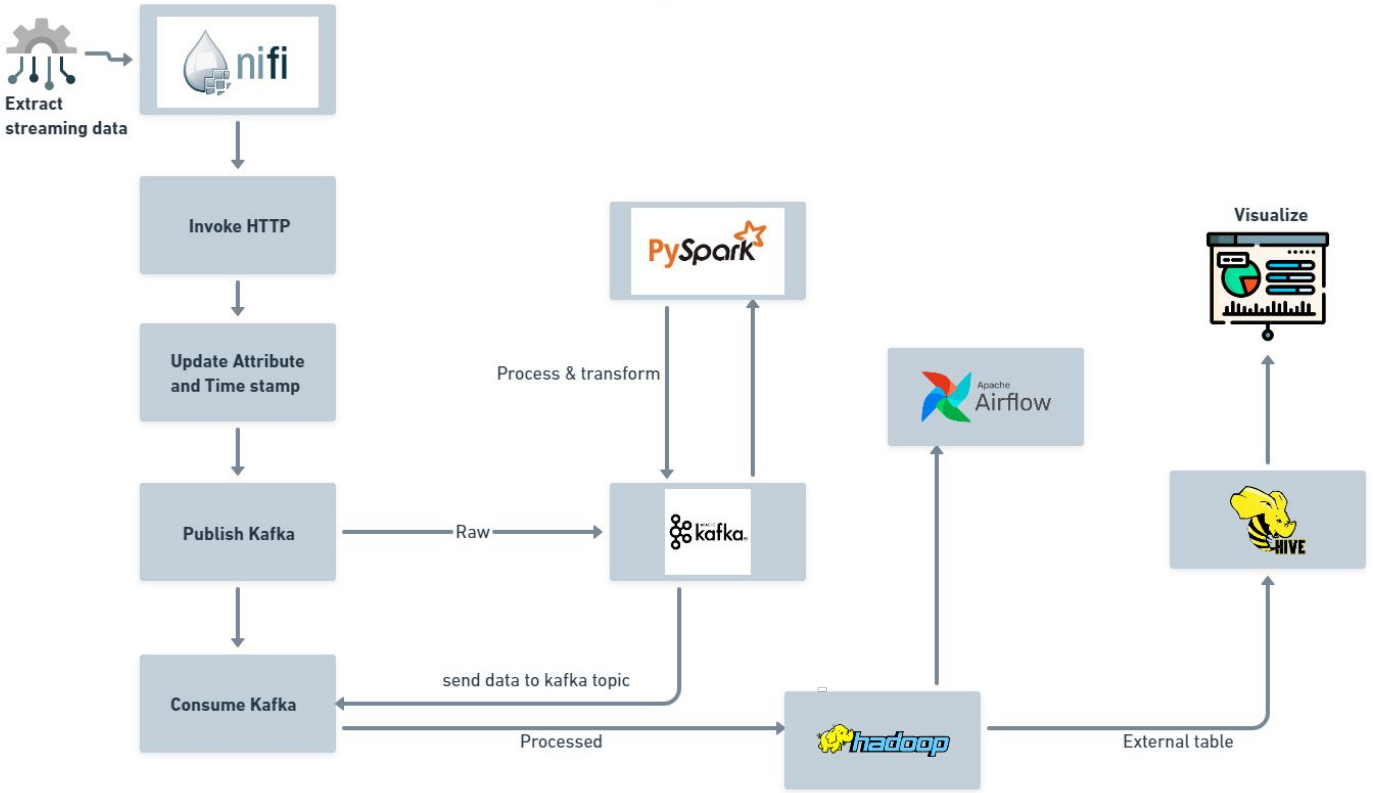
Introduction to Data Pipeline



Project Overview

Problem Description

Data architecture



Understand the application scope

1. The speed of getting data pipelines up and running
2. Getting new engineers up to speed
3. Spending less time worrying about the service management
4. Complexities
5. Providing business value
6. Overall engineering cost

Tour to existing solution

Component	Responsibility	Examples
`Scheduler`	Starting data pipelines at their scheduled frequency.	Airflow scheduler, cron, dbt cloud, etc
`Executor`	Running the data processing code. The executor can also call out other services to process the data.	python, data warehouse, Spark, k8s, dbt, etc
`Orchestrator`	Ensuring that the data pipeline tasks are executed in the right order, retrying on failures, storing metadata, and displaying progress via UI.	Airflow, Prefect, Dagster, dbt
`Source`	System where data is to be read from.	OLTP databases, cloud storage, SFTP/FTP servers, REST APIs, etc
`Destination`	Making data available for the end-user.	data warehouses, Elastic search, NoSQL, CSV files, etc

`Visualization/BI tool`	Enabling business users to look at data patterns and build shareable dashboards.	Looker, Tableau, Apache Superset, Metabase, etc
`Queue`	Accepting continuously incoming data (aka streaming) and making it available for the consuming system to read from.	Kafka, Pulsar, AWS Kinesis, Nats, RabbitMQ, etc
`Event triggers`	Triggering an action in response to a defined event occurring.	AWS lambda triggers, Watchdog, etc
`Monitoring & Alerting`	Continuously monitoring data pipelines and alerting in case of breakage or delay.	Datadog, Newrelic, Grafana,
`Data quality check`	Checking if data confines to your expectations.	custom scripts checking for data constraints & business rules, Great expectations, dbt tests, etc

Data Infrastructure : Components used

Apache NiFi

- Open source for automating and managing data flow between systems.
- Process distributed data and executes on the JVM on host OS.
- Web based UI for creating, monitoring & controlling data flows.
- Supports buffering of all Queued data.
- Processors connect to many source and destination systems.
- Easy to troubleshoot and flow optimisation.
- Role based authentication
- Build user defined processors and controller services.
- Easy of use- need not code much

HDFS

- **Data replication.** This is used to ensure that the data is always available and prevents data loss. For example, when a node crashes or there is a hardware failure, replicated data can be pulled from elsewhere within a cluster, so processing continues while data is recovered.
- **Fault tolerance and reliability.** HDFS' ability to replicate file blocks and store them across nodes in a large cluster ensures fault tolerance and reliability.
- **High availability.** As mentioned earlier, because of replication across notes, data is available even if the NameNode or a DataNode fails.
- **Scalability.** Because HDFS stores data on various nodes in the cluster, as requirements increase, a cluster can scale to hundreds of nodes.
- **High throughput.** Because HDFS stores data in a distributed manner, the data can be processed in parallel on a cluster of nodes. This, plus data locality (see next bullet), cut the processing time and enable high throughput.
- **Data locality.** With HDFS, computation happens on the DataNodes where the data resides, rather than having the data move to where the computational unit is. By minimizing the distance between the data and the computing process, this approach decreases network congestion and boosts a system's overall throughput.

Apache Kafka

- Apache Kafka is an open-source stream-processing software which buffers the records as they occur without data loss
- Uses I/O efficiently by batching and compressing records
- Supports streaming data and is scalable, durable and fault tolerant
- Uses file system for caching and storage
- Works on publish and subscribe mechanism
- Producers are source systems , consumers are destination systems
- Messages stored in topic
- Why Kafka over Apache Flume or Apache Storm?

Kafka Vs Flume

Kafka	Flume
Used as a bridge between source and destination systems	Mostly used to gather data into Hadoop from various sources
No data loss and more general purpose system can be used on any data	No data loss but mostly used for log based aggregations and log data
Multiple publishers and subscribers can share same topic	It's a tool to gather data into HDFS
Makes data available even after single point failure	Depends on the configuration

Hive

- Hive is scalable, fast, and uses familiar concepts
- Schema gets stored in a database, while processed data goes into a Hadoop Distributed File System (HDFS)
- Tables and databases get created first; then data gets loaded into the proper tables
- Hive supports four file formats: ORC, SEQUENCEFILE, RCFILE (Record Columnar File), and TEXT FILE
- The most significant difference between the Hive Query Language (HQL) and SQL is that Hive executes queries on Hadoop infrastructure instead of on a traditional database

Airflow

Features of Apache Airflow

1. **Easy to Use**: If you have a bit of python knowledge, you are good to go and deploy on Airflow.
2. **Open Source**: It is free and open-source with a lot of active users.
3. **Robust Integrations**: It will give you ready to use operators so that you can work with Google Cloud Platform, Amazon AWS, Microsoft Azure, etc.
4. **Use Standard Python to code**: You can use python to create simple to complex workflows with complete flexibility.
5. **Amazing User Interface:** You can monitor and manage your workflows. It will allow you to check the status of completed and ongoing tasks.

Data Visualization Tools

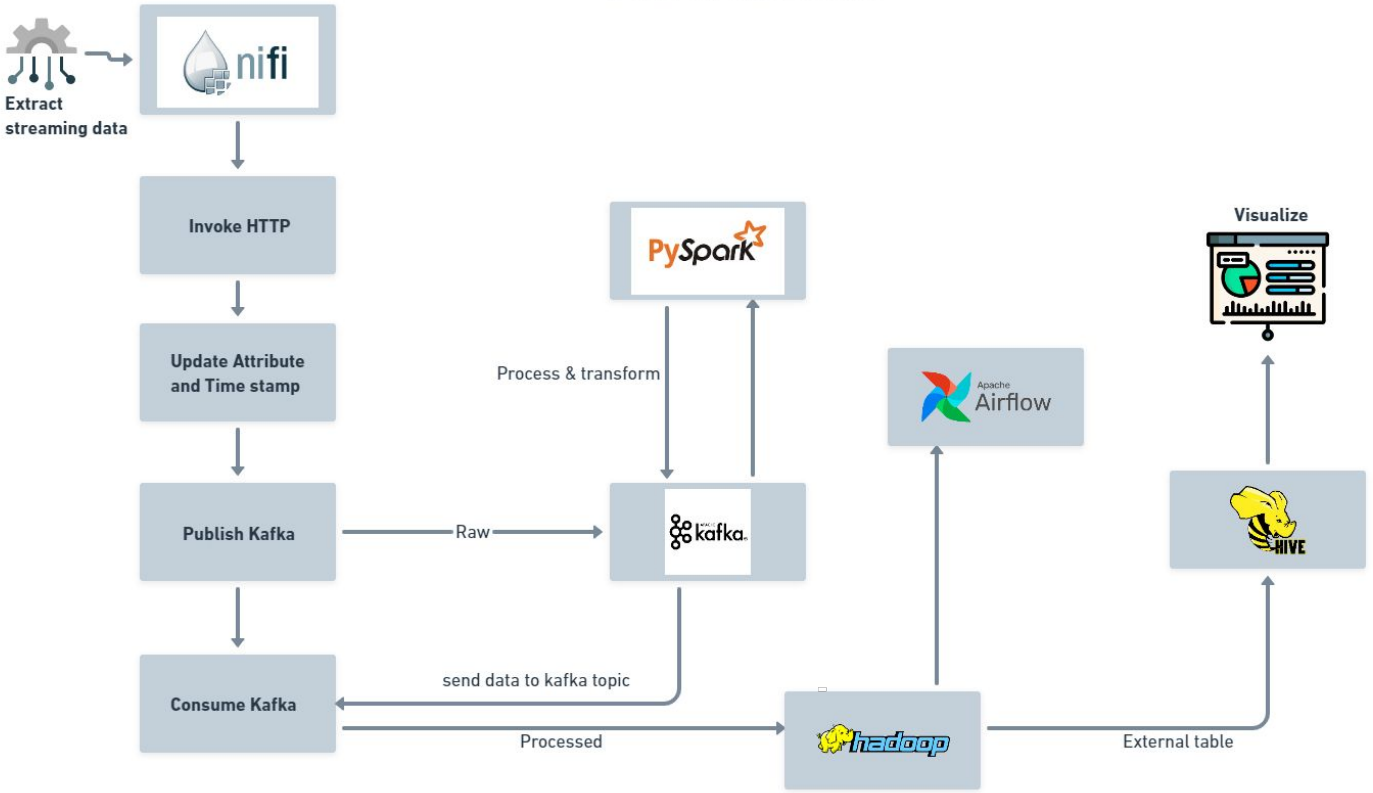
How to gather requirements to Build data pipeline

How to gather requirements for data pipeline

1. Think like the end user
2. Know the why
3. End user interviews
4. Create High Level Design
5. End user Walkthrough for proposed solution
6. Timelines and Deliverables

Tour to Architecture diagram

Data architecture




thank you!