  
  
**Workflow**   
1. Extracting data from API.

2. Streaming data from Azure App Service Application through Kafka

3. Saving data from Kafka consumer to a Delta table, storing CSV files in DBFS.

4. In Spark: Converting data to proper schemas and cleaning.

5. In Spark: sorting, aggregating, determining, etc. for Analysis.

**Data Sources**

**Primary Data Source**

* **Blockchain.com Explorer API** (<https://www.blockchain.com/explorer/api>)
* <https://api.blockchain.com/v3/?python#authentication>
* **Access Method:** REST API or WebSocket [Still Exploring
* **Data Extracted:**
  + Live blockchain transactions
  + Block confirmations
  + Mempool activity
  + Mining statistics

**Supplementary Data (Enrichment via CSV Files)**

* Historical transaction data
* Exchange rate data (e.g., BTC/USD, ETH/USD)
* Metadata about known wallet addresses
* Block size and hash rate trends

**Technical Approach**

1. **Data Extraction & Ingestion**

* **Option 1: REST API** (Easier implementation, but data updates are periodic.)
* **Option 2: WebSockets** (More complex but provides real-time streaming.)
* Use a Python/Scala script to fetch data from the **Blockchain.com API** and push it into Kafka.
* Enrich real-time data with **CSV files** stored in **Databricks File System (DBFS)**.

1. **Data Streaming via Kafka**

* Deploy an **Azure App Service** application to collect and forward streaming data.
* Use **Apache Kafka** as the messaging queue:
  + Producer: Fetches data from the API/WebSocket and pushes it into Kafka.
  + Consumer: Reads the data and processes it in Spark.

1. **Data Storage in Delta Lake & DBFS**

* Store raw blockchain transaction data in a **Kafka-backed Delta Lake table**.
* Store historical CSV files in **DBFS** for reference and enrichment.
* Checkpointing enabled to ensure fault tolerance and exactly-once processing.

1. **Data Processing & Cleaning in Spark**

* **Schema Inference & Data Parsing:** Convert incoming JSON data to structured format.
* **Data Deduplication:** Ensure unique transactions and valid data points.
* **Filtering:** Remove irrelevant or spam transactions.

1. **Data Analysis & Insights Generation**

* **Aggregation & Sorting:** Track transaction volume, miner rewards, and mempool growth.
* **Anomaly Detection:** Detect unusual transaction spikes or network congestion.
* **Trend Analysis:** Identify trends in block confirmation times and network fees.

**Business Questions to Answer**

**4.1. Transaction Analysis**

* What are the most frequent transaction amounts being processed on the blockchain?
* How has the number of transactions changed over time?

**4.2. Network Congestion & Fees**

* How does transaction volume impact confirmation times and fees?
* Are there patterns in network congestion during specific times of the day?

**4.3. Miner & Block Activity**

* Which miners process the most blocks?
* How often do orphan blocks appear, and what causes them?

**6. Implementation Plan**

**Phase 1: Setup & Data Ingestion**

* Configure **Kafka** on Azure
* Develop API/WebSocket **data extraction script**
* Set up **Kafka producer and consumer**
* Store raw data in Delta Lake

**Phase 2: Data Processing & Cleaning**

* Implement **schema inference** and validation in Spark
* Enrich data with CSV files (exchange rates, metadata)
* Perform **basic aggregations & filtering**

**Phase 3: Advanced Analysis & Reporting**

* Implement **network congestion analysis**
* Detect **anomalies & suspicious transactions**
* Build **real-time dashboards** in Power BI/Tableau

**Phase 4: Optimization & Deployment**

* Optimize **Spark performance & storage formats**
* Automate data pipeline execution
* Deploy dashboards & schedule reports