Use-Cases for Batch Data Pipelines

Big Data on Kubernetes – [Day 3]

LUAN MORENO

CEO & Data Architect
Data Engineer & MVP

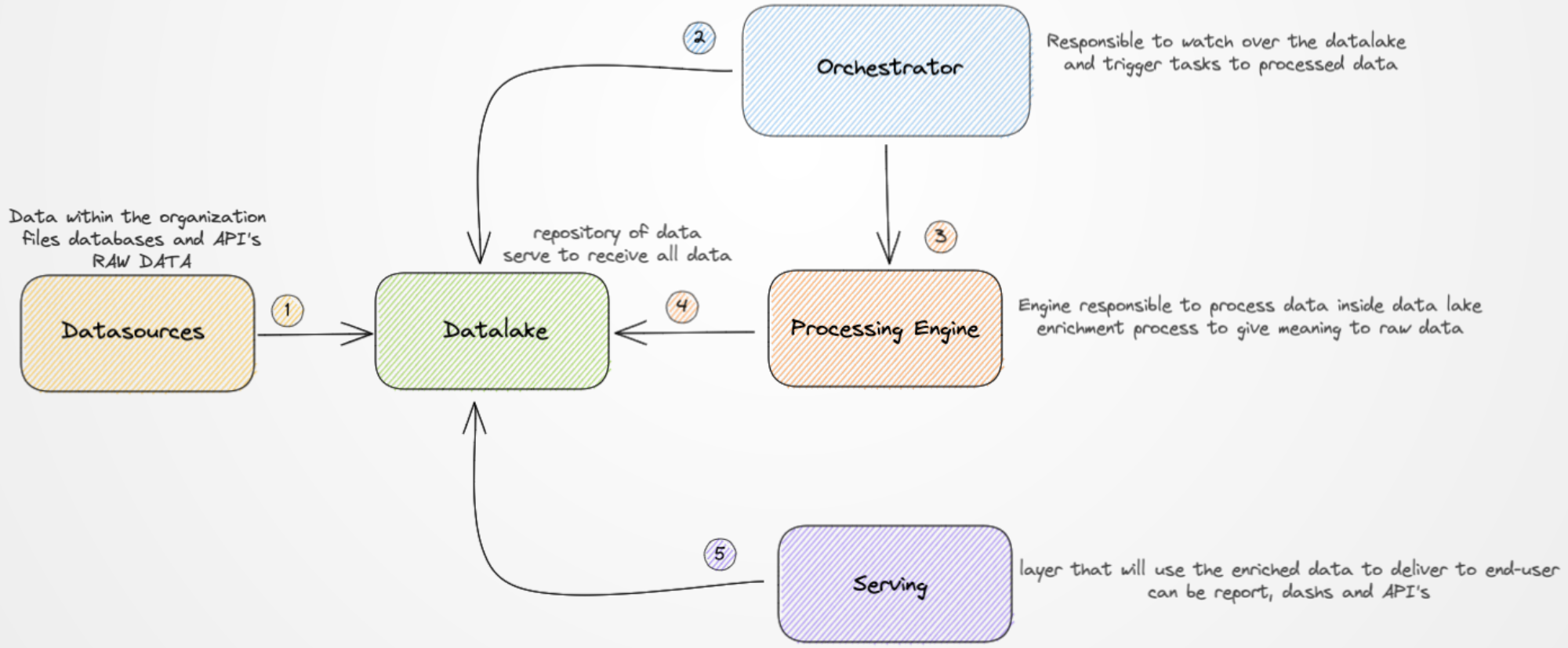
MATEUS OLIVEIRA

Big Data Architect
Data In-Motion Specialist

Batch Data Pipelines on Kubernetes

Symbiosis of Big Data and Kubernetes Infrastructure
Backbone System for Batch Data Pipelines at Scale

1. Data is uploaded in Data lake in the initial folder [landing zone]
2. Orchestrator watch data lake new data arrival to trigger processing tasks
3. Processing engine is triggered by orchestrator move to another folder [processing zone]
4. Enriched data is added into Data lake in another folder [curated zone]
5. Serving application access the curated zone folder to retrieve the enriched data

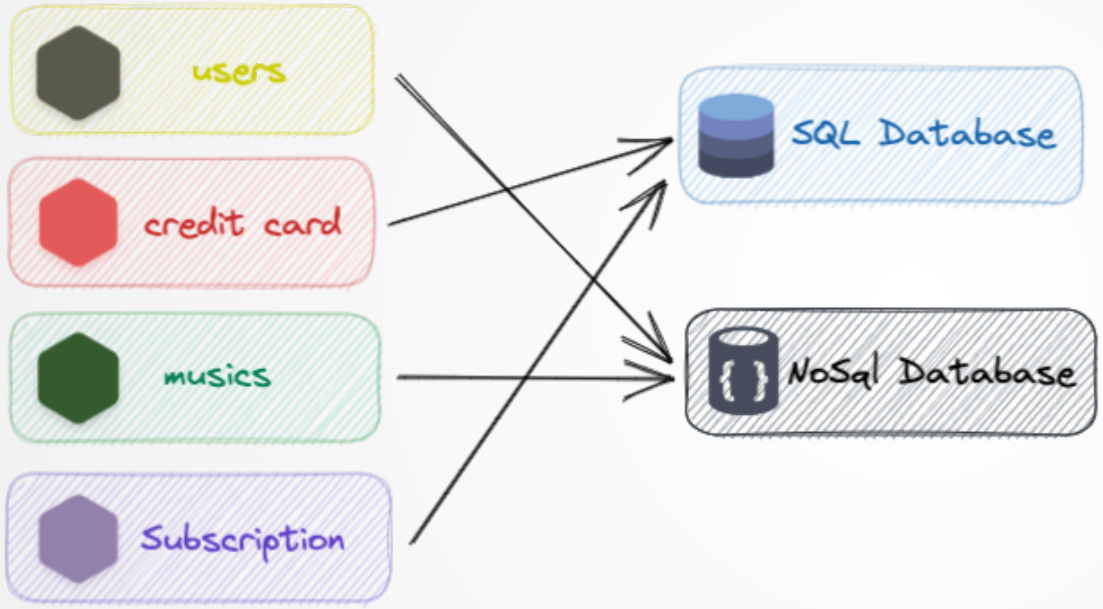


Use-Cases

- Using a Distributed Data Store for Microservices Applications
- Processing Files on Data Lake using Apache Spark and Delta Lake
- Data Exploration with Apache Spark and Apache Zeppelin
- Querying Data Sources using Trino for Analytics at Scale
- Orchestrating Data Pipelines Efficiently using Apache Airflow 2.0

Using a Distributed Data Store for Microservices Applications

Business Use-Case

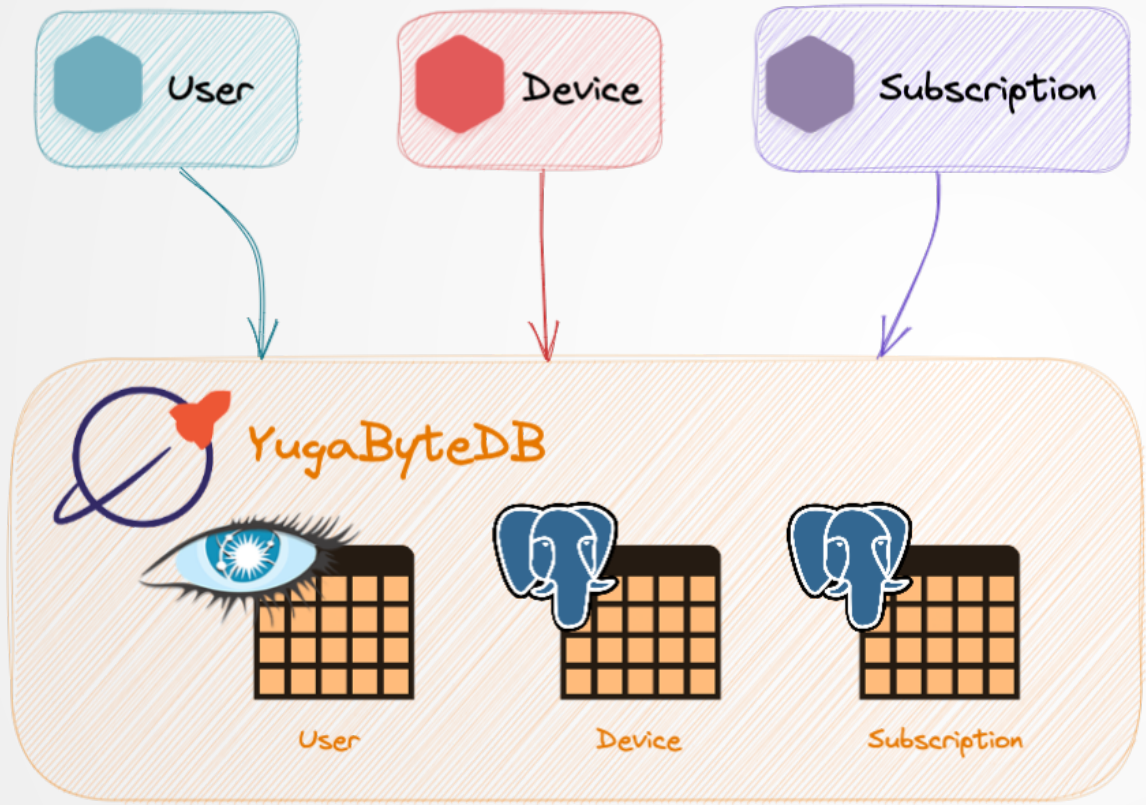


Business Use-Case Scenario

- Multiple Microservices Storing Data in a Variety of Different Data Stores
- Managing Microservices Communication is Hard (Use of Service Mesh)
- CQRS Decouples Reads and Writes - Requests & Analytics
- Complex Maintenance of Data Stores
- Usually, Data Stores Provisioned as a PaaS or SaaS
- Extract Analytics from Microservices is Complex

Using a Distributed Data Store for Microservices Applications

Stack



Technology Stack

- Microservices Written in Python
- Using a Distributed Database Engine (YugaByteDB)
- Postgres & Apache Cassandra API
- Unified Location to Query and Store Data

Advantages

- Usage of Microservices Pattern
- Increase of Development Time
- Simplify Data Store Strategy
- Unify Data Needs in a Single Place

Demo

Processing Files on Data Lake using Apache Spark and Delta Lake

Business Use-Case

Data from backend applications are inserted into a Data lake
The need is to choose a processing engine using best practices of the market
must be fast, scale, and distributed engine
don't forget to think is some lakehouse architecture

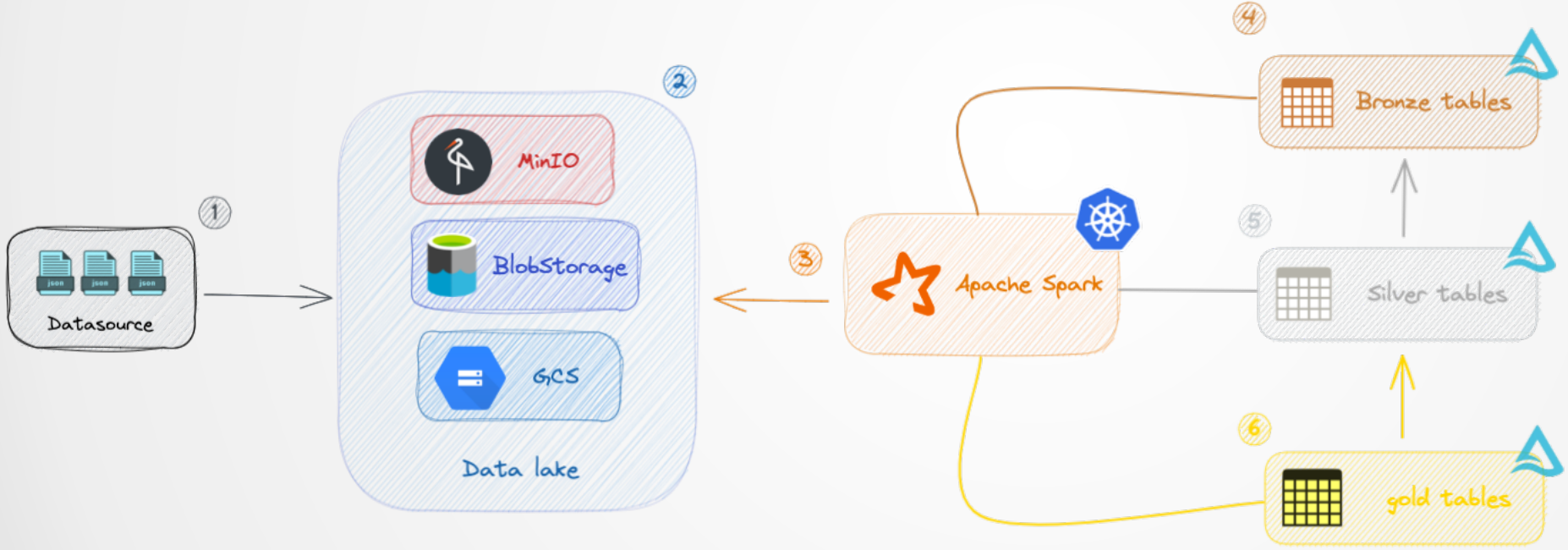
Business Use-Case Scenario

- Multiple Data Sources Writing into a Data Lake Store
- Managing Data Swamp Problems
- Processing Efficiently Data inside of Data Lake
- Reduce of Cost and Storage Capacity
- Using Appropriate File Types for Processing at Scale

Processing Files on Data Lake using Apache Spark and Delta Lake

Stack

1. Raw data coming from databases and apps
2. Data lake technologies available OSS MinIO, Azure blob storage, and Google Cloud GCS [landing zone]
3. Apache Spark access data lake landing zone folder, the first step access raw data
4. Transform raw data into a delta bronze table, copy the data as-is. [bronze zone]
5. Access delta bronze table make transformations (clean data, change columns names,etc) and the result of transform we create delta silver table [silver zone]
6. Using the delta silver table we create delta gold table based on specific business view, for example gold table for anti-fraud department use case



Technology Stack

- Use of Open-Source Cloud-Native Data Lakes
- The Powerful Processing Engine - Apache Spark
- The Best File Format - Delta Lake
- The Delta Architecture to Create a Data Pipeline
- Building a Data Lakehouse Paradigm

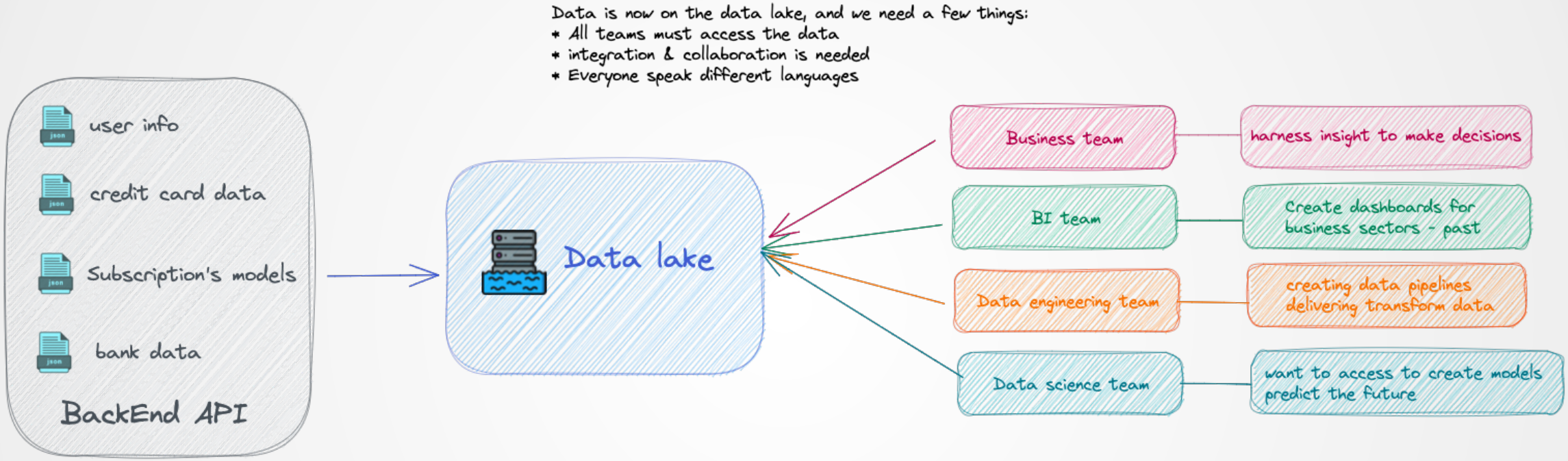
Advantages

- Data Lakes = Without Inbound and Outbound Costs
- Apache Spark = Better Utilization of Resources
- Delta Lake = Storage Engine for Fast Access

Demo

Data Exploration with Apache Spark and Apache Zeppelin

Business Use-Case



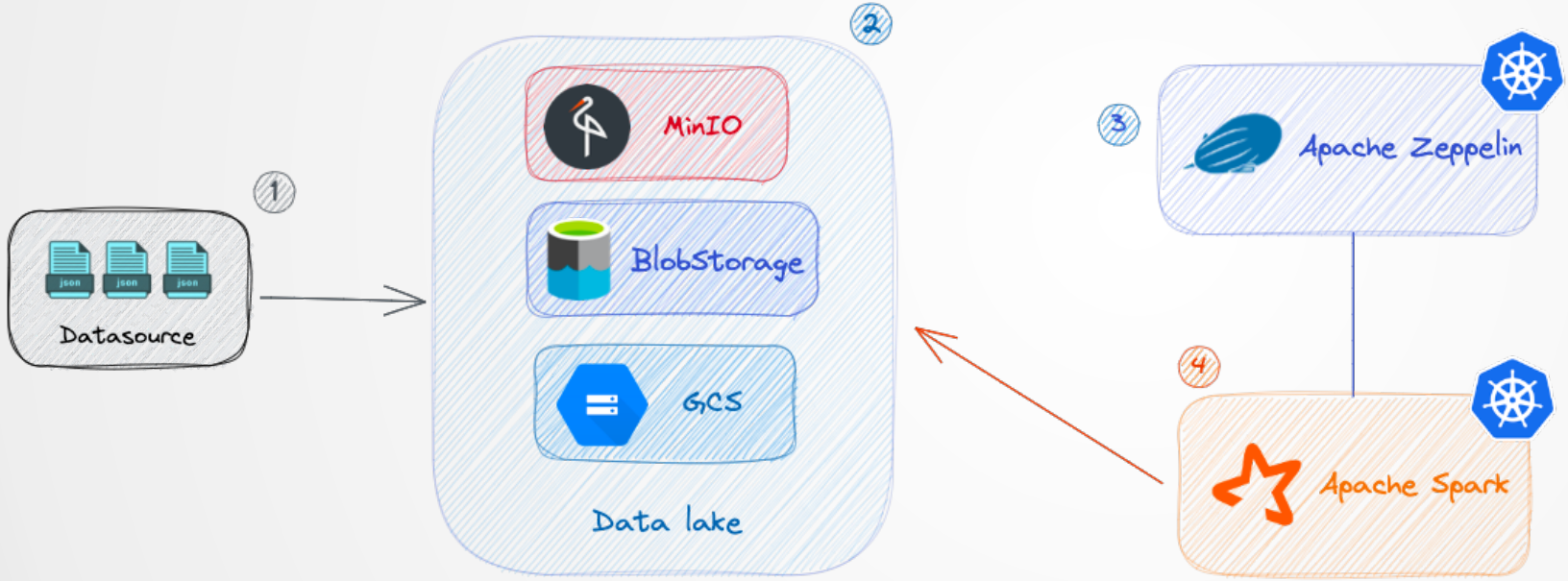
Business Use-Case Scenario

- Analytics is Bloodstream for Finding Patterns
- Multiple Teams Analyzing Data from Data Lake
- Tight Integration with Processing Engines
- Notebook Experience for Development and Data Exploration

Data Exploration with Apache Spark and Apache Zeppelin

Stack

1. raw data coming from databases and apps
2. data lake technologies available OSS MinIO, Azure blob storage, and Google Cloud GCS [landing zone]
3. Apache Zeppelin is a notebook experience tool that communicates with Apache Spark, to run spark codes
4. Apache Spark do that heavy lifting executing codes from Zeppelin acquiring data for exploration



Technology Stack

- Use of Open-Source Cloud-Native Data Lakes
- The Powerful Processing Engine - Apache Spark
- Data Exploration with Notebook Experience – Apache Zeppelin

Advantages

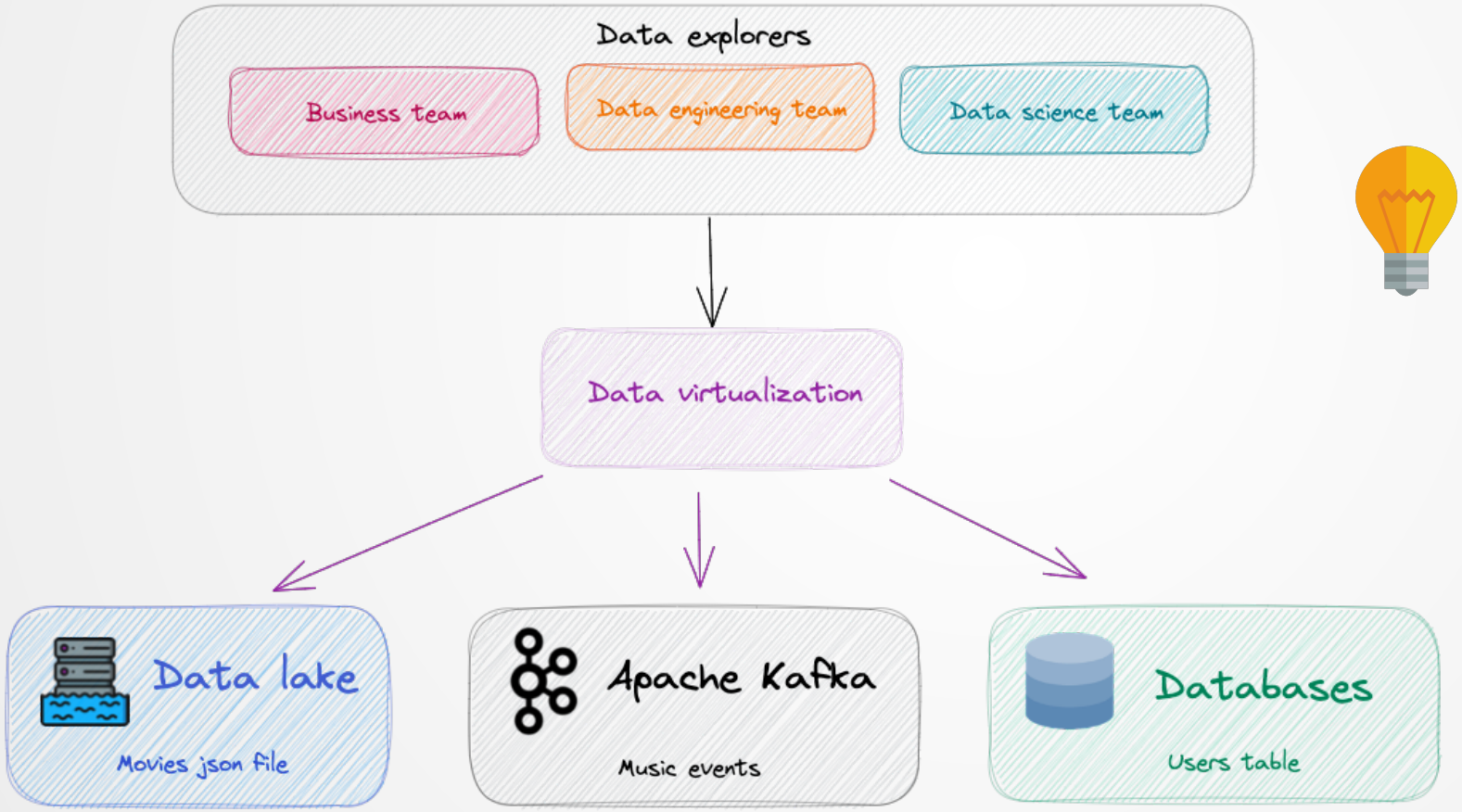
- Development using Notebook Experience for Multiple Teams
- Data Exploration for Development of Data Pipelines
- Fully Integrated with Apache Spark
- Open-Source and Costwise

Demo

Query Data Sources using Trino for Analytics at Scale

Business Use-Case

with more and more data in different stores, one problem surfaces in the head
how we going to query data from different places, using one place, and improve for this kind of operation
In this case, we can use data virtualization to access different stores using SQL



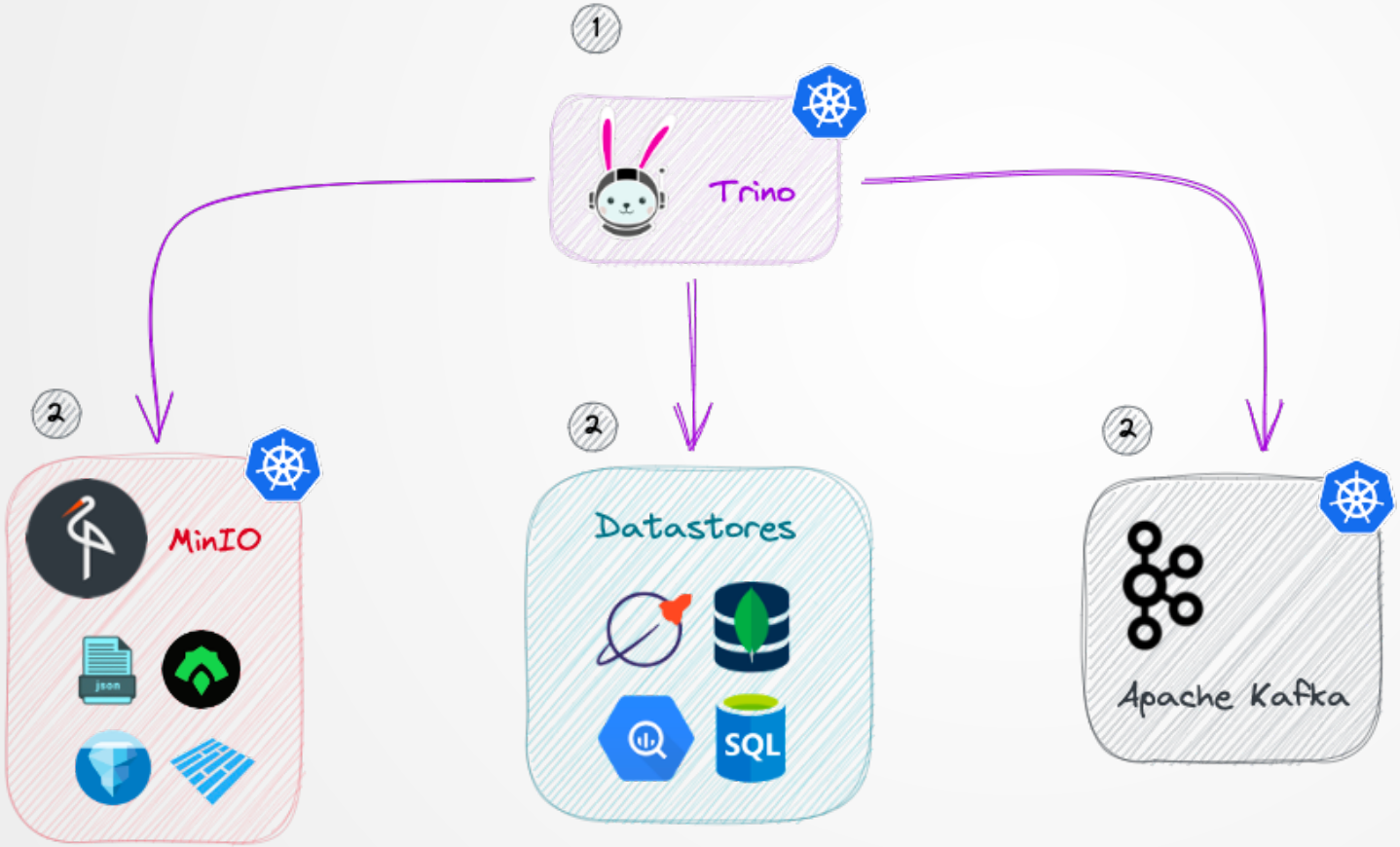
Business Use-Case Scenario

- Data Split Across Multiple Data Sources within Company
- Different Kind of Data Store Technologies
- Multiple ETL Processes for Crunching Data
- Duplication of Data, Increase Overall Storage Cost
- Explore Data at 360° View

Query Data Sources using Trino for Analytics at Scale

Stack

1. Trino access data sources through a catalog (YAML file config)
2. In MinIO, you need to specify a create table statement
2. In databases such as YugabyteDB he makes in schema infer, only query
2. Kafka reads events in real-time, with a function call json_extract_scalar we can access nested columns



Technology Stack

- Virtualization of Data
- Trino
- Presto

Advantages

- Data Exploration to Understand Data Sources
- Query SQL for Analytical Purpose
- Join Multiple Data Sources in One Technology
- Discover Patterns for Data Pipeline Build

Demo

Orchestrating Data Pipelines Efficiently using Apache Airflow 2.0

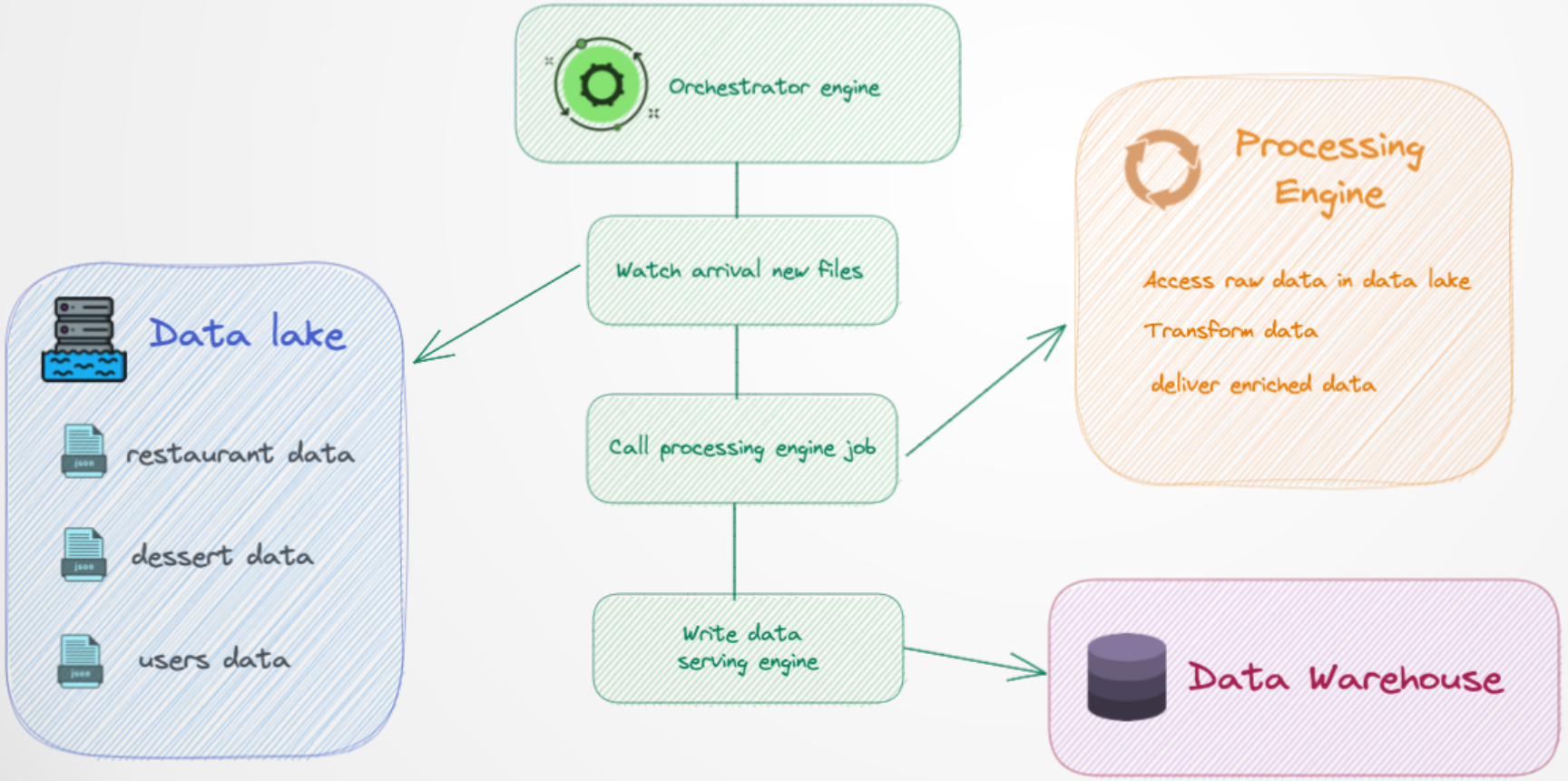
Business Use-Case

Instead of jobs that executes time to time we can create a data pipeline orchestrated

* Verify if new files landed in data lake
* call the job from execution engine
* write the data into a DW or Data lake [curated zone]

Business Use-Case Scenario

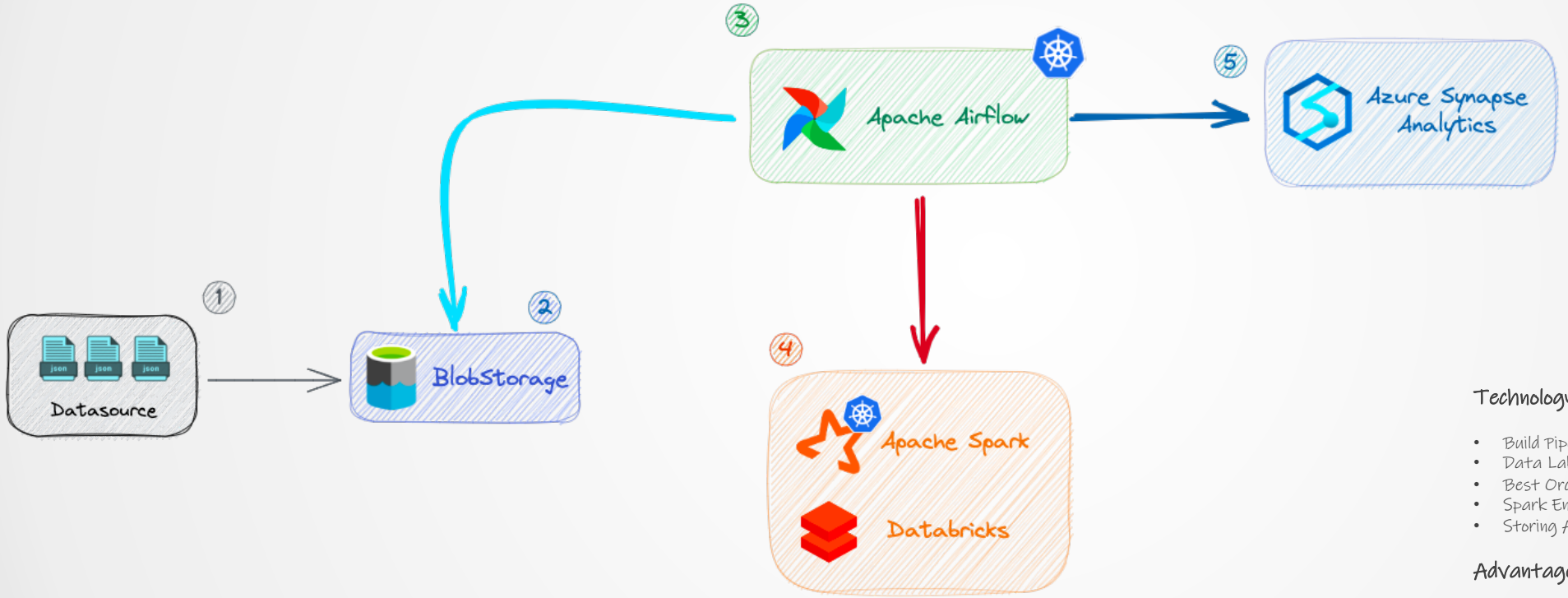
- Manage Multiple Data Pipelines
- Organize and Schedule Interconnected Pieces
- Channeling Data Arrival and Data Enrichment
- Validate and Troubleshooting Problems
- Main Location to Monitor and Watchdog Process



Orchestrating Data Pipelines Efficiently using Apache Airflow 2.0

Stack

1. Data source with multiple files write in blob storage
2. blob storage receive new files into landing zone folder
3. Airflow DAG execute and runs a task: sensor to verify new files in blob storage
4. Airflow DAG execute and runs a task: call apache spark job to process new data
5. Airflow DAG execute and runs a task: write data into Synapse Analytics [DW]



Technology Stack

- Build Pipelines using Python
- Data Lake Cloud Provider – Blob Storage
- Best Orchestration Tool for Data Pipeline – Airflow
- Spark Engines – Databricks [Enterprise], Spark on K8s OSS
- Storing Analytical Data – Synapse Analytics

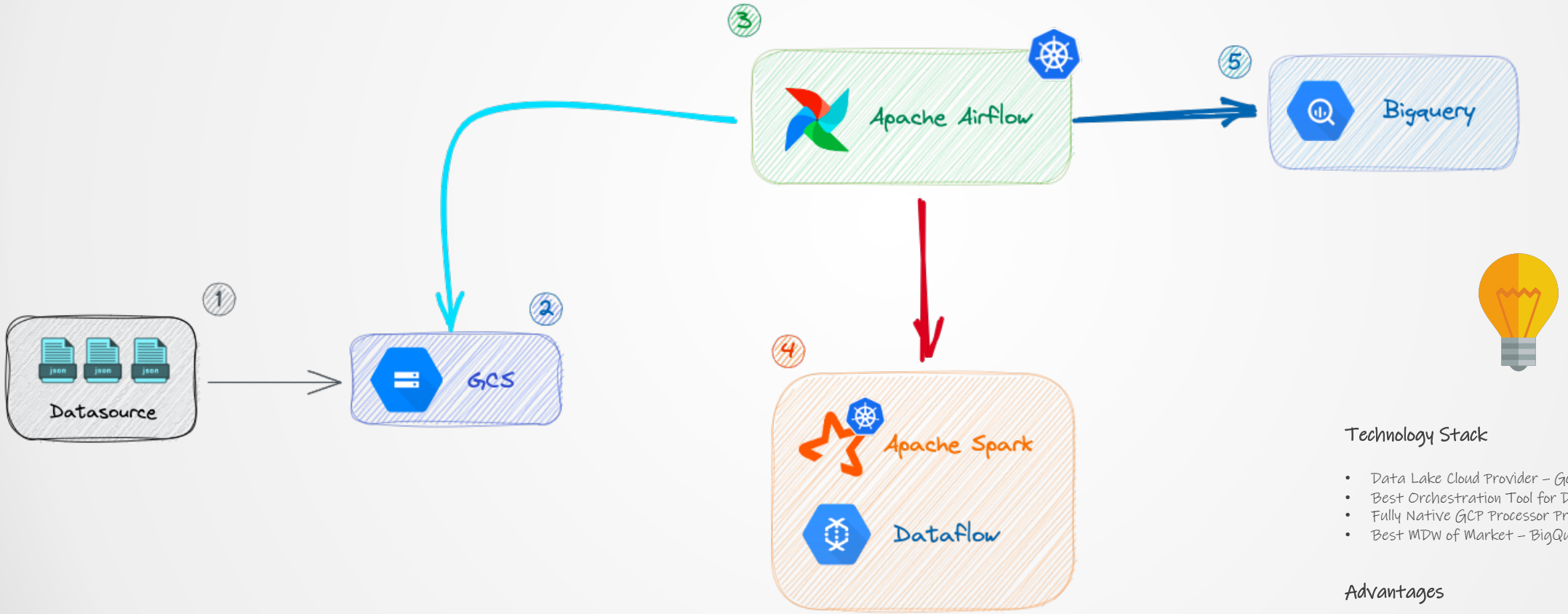
Advantages

- Hybrid Data Pipeline for Batching in Azure
- Incremental Processing
- Cost Reduction by Using Kubernetes Big Data Stack
- Easy Scale

Orchestrating Data Pipelines Efficiently using Apache Airflow 2.0

Stack

1. Data source with multiple files write in GCS
2. GCS receive new files into landing zone bucket
3. Airflow DAG, execute and runs a task: sensor to verify new files in landing bucket
4. Airflow DAG, execute and runs a task: call apache spark job or dataflow job to process new data
5. Airflow DAG, execute and runs a task: write data into BigQuery



Technology Stack

- Data Lake Cloud Provider – Google Cloud Storage
- Best Orchestration Tool for Data Pipeline – Airflow
- Fully Native GCP Processor Product – Dataflow
- Best MDW of Market – BigQuery

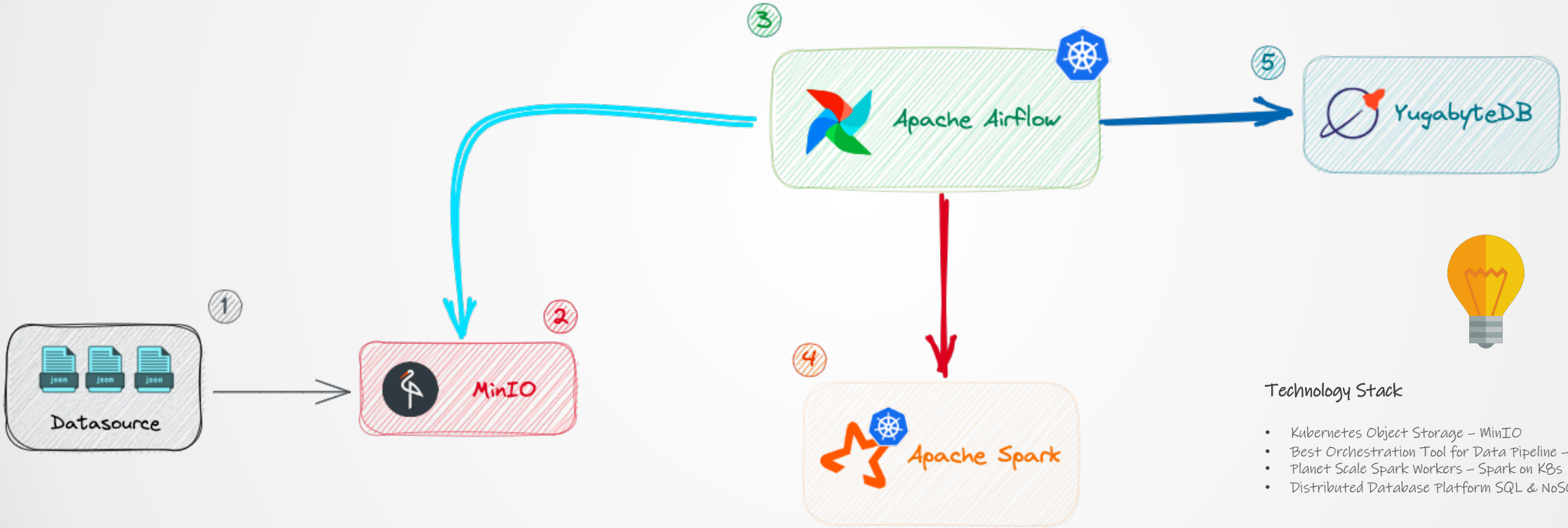
Advantages

- Hybrid Data Pipeline in GCP using OSS to Orchestrate
- Processing using Apache Spark on K8s or Dataflow

Orchestrating Data Pipelines Efficiently using Apache Airflow 2.0

Stack

1. Data source with multiple files write in MinIO
2. MinIO receive new files into landing zone bucket
3. Airflow DAG execute and runs a task: sensor to verify new files in landing bucket
4. Airflow DAG execute and runs a task: call apache spark job using spark operator to process new data
5. Airflow DAG execute and runs a task: write data into YugabyteDB



Technology Stack

- Kubernetes Object Storage – MinIO
- Best Orchestration Tool for Data Pipeline – Airflow
- Planet Scale Spark Workers – Spark on K8s
- Distributed Database Platform SQL & NoSQL - YugabyteDB

Advantages

- Full Open-Source Data Pipeline in Batch [Without Lock-In]
- Migration for Any Cloud Provider is Simple – 0% Complexity
- Everything Deployed and Managed in One Single Place
- Costwise for Better Resource Utilization

Demo


With ordinary talent and extraordinary perseverance, all things are attainable
Thomas Fowell Buxton
quotefancy