Building Streaming Application Using Kafka

Kodjo Klouvi | kodjo.klouvi@gmail.com

What is Kafka?

- Event-driven architecture: publisher – subscriber topology

- Not a client-server architecture:
  - multiple brokers playing the same role
  - any broker can be used to target the cluster (bootstrap server for producer/consumer)

- Used to build streaming applications and data pipeline

- High scalability, fault tolerant, support data retention

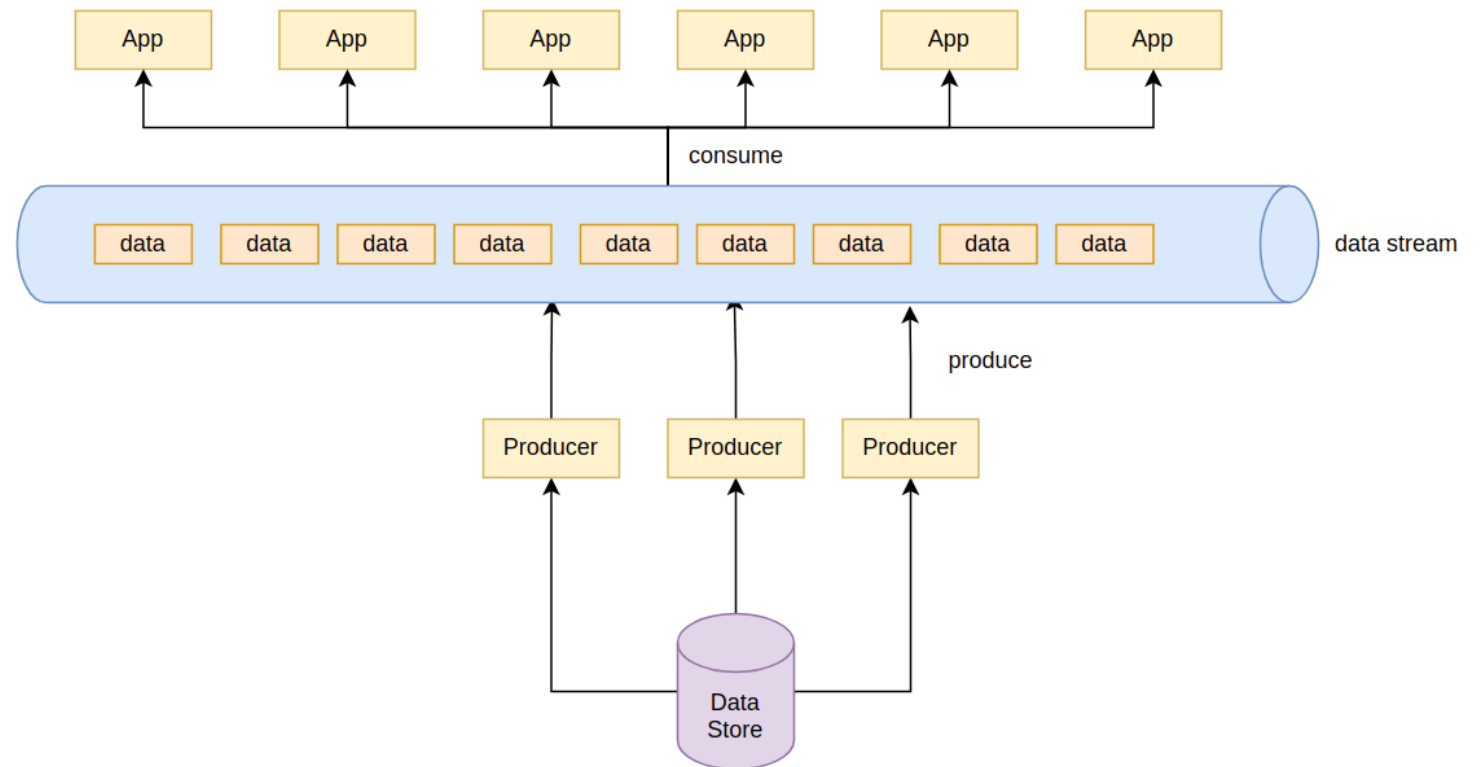- Many options to configure and some require advanced expertise!

Flashback...

- Multiple applications access the same data at the same time and apply the same transformations/operations
  - Risk of bottleneck
  - Massive resources consumption
  - Duplication
  - Heavy to maintain
- Synchronous actions (read – write)
 - Time consuming
  - Not parallelable
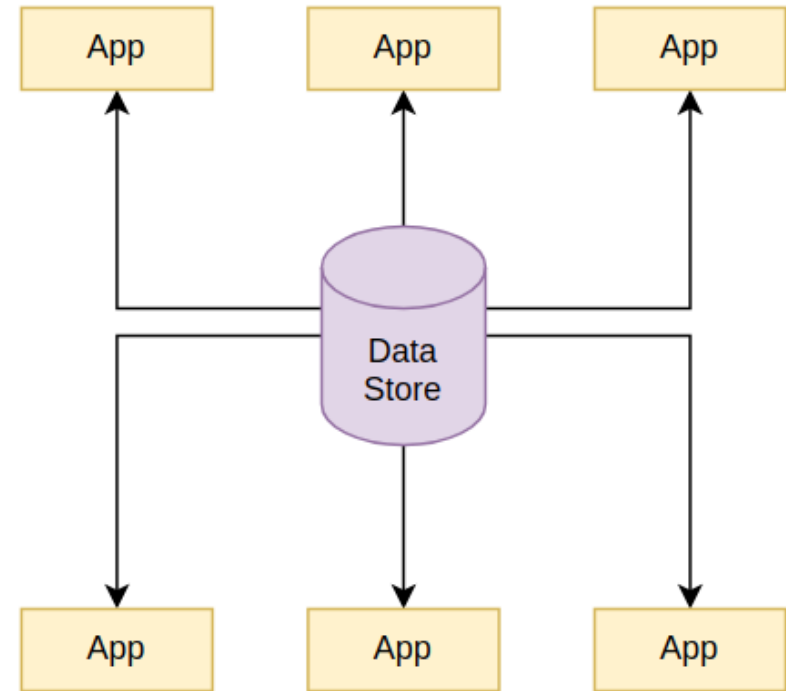
Kafka Architecture

- **Message queue**: data structure to store messages (events)

- **Producer**: application that publishes messages into the queue

- **Consumer**: application that reads messages from the queue

App   App   App   App   App   App

consume

| data | data | data | data | data | data | data | data | data | data stream |

produce

Producer   Producer   Producer

Data Store

4

Kafka Architecture

- **Broker**: application that manages the data.

- **Zookeeper**: manage the infrastructure, the connectivity (network discovery) and meta data

- **Asynchronous actions**
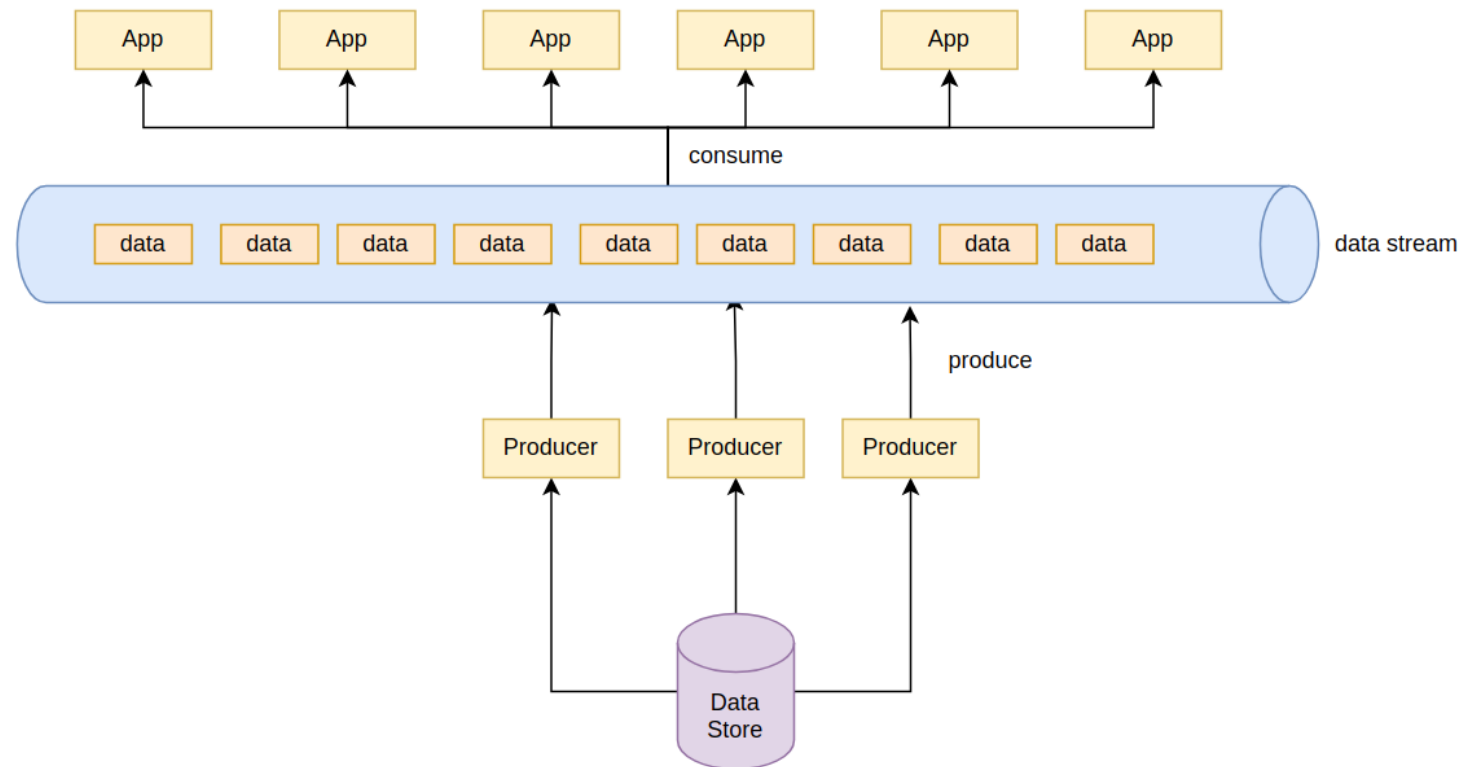
Kafka Architecture

- **Topic**: data structure to store the same type of data (for the same purpose)

- **Partition**: division of a topic. Each topic can have multiple partitions allowing horizontal scalability across many servers.

- **Offset**: unique identifier assigned to each message within partition. The position of the message in the partition.
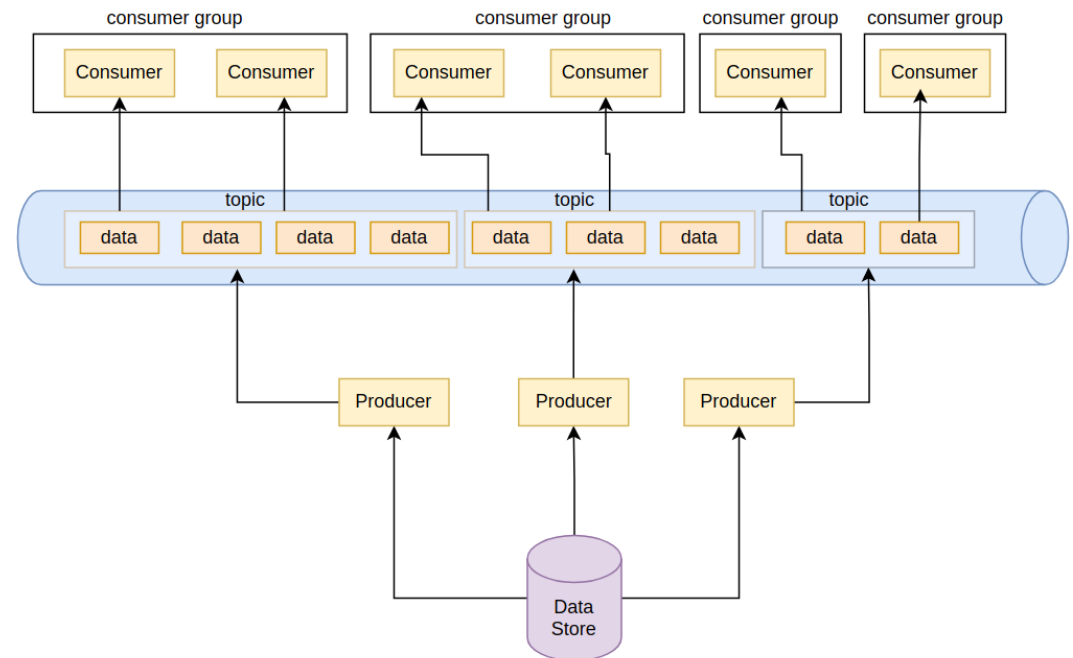
Kafka Architecture

- **Consumer group**: group of applications/consumers that read the same topic – they work in parallel using partitions and offset paradigm

- **Multiple brokers**: data are stored on multiple nodes (replication)

- **Replication**: each message is stored on multiple nodes (replication factor) and has its own leader (broker)
  - fast and parallel read
  - fault tolerant, etc.
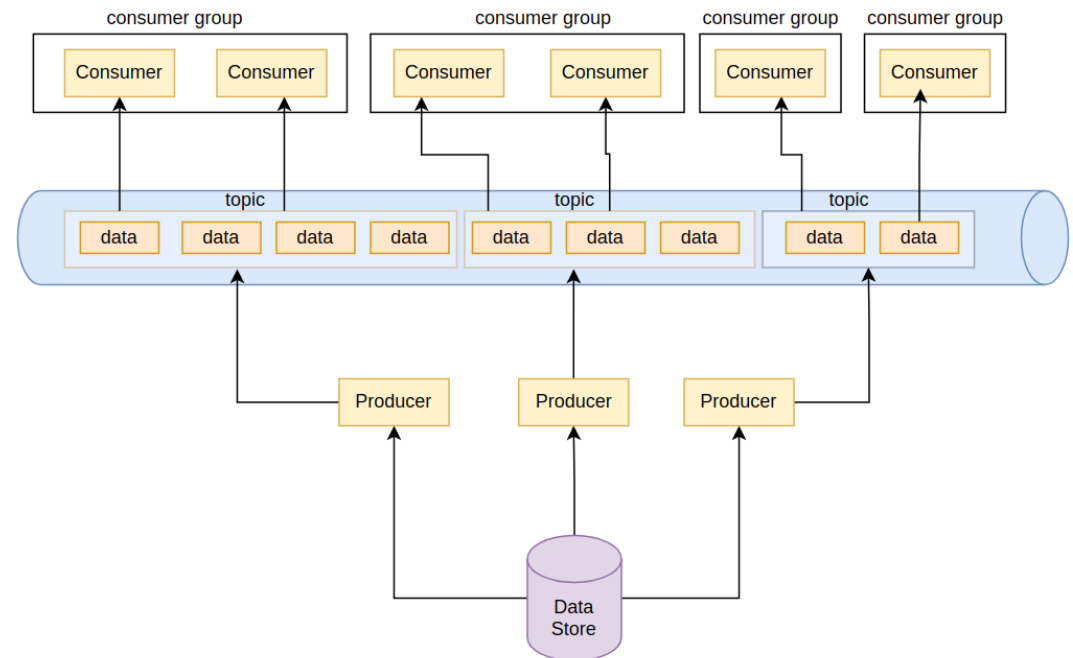
Kafka Architecture

- Only one broker per machine
  - Kafka doesn't support horizontal scalability at the machine level

- Each broker has its own ID (broker.id)

- Data retention period: keep data (in the log) for some period

- Programming language agnostic

- Sensitive to the data structure/format, serialization/deserialization

Use Cases

- Real-Time Data Streaming and Analytics

- Data Pipeline Modernisation (ETL)

- Microservices Communication

- Event Sourcing

- Log Aggregation

- Stream Processing (Kafka Stream API)

Use Cases

- Data Integration Across Systems (Kafka Connect)

- IoT Data Management

- Replication and Disaster Recovery

- Edge Computing

Hands-On

- https://github.com/osekoo/hands-on-kafka

- Preparing development environment

- Practice 1: Get started

- Practice 2: Implement online word definition search (dictionary)

- Practice 3: Homework (advanced dictionary)

PyCharm

- Python Integrated Development Environment

- Multiple modules available for Python

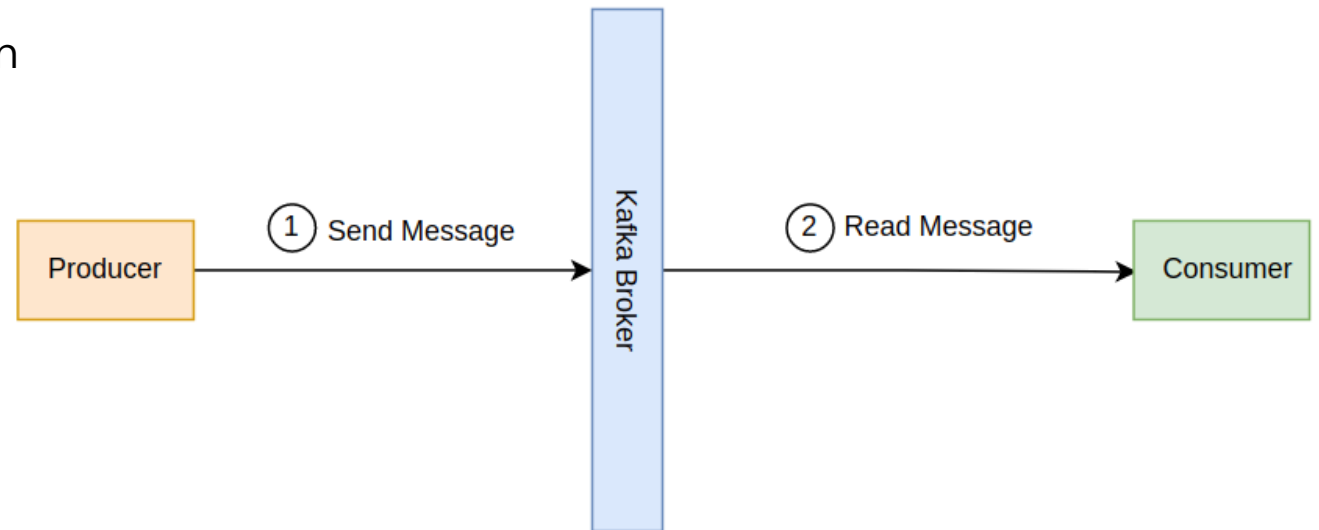- Enable debugging, Python applications packaging, etc.

- Even more...

Docker

- Platform as a service (PaaS)

- OS abstraction

- Self-contained applications running in containers

- Simplify applications delivery/distribution

- Enable horizontal scaling

Practice 1: Get started

- Build a very simple Kafka based application

- Implement "*push-pull*" message queue

Practice 2: Online dictionary

- Build a Kafka based word definition search dictionary

- User specifies his nickname and the dictionary language (FR, EN)

- He/she interacts with the application by sending a word

- Our application search for the word definition by scrapping online dictionaries
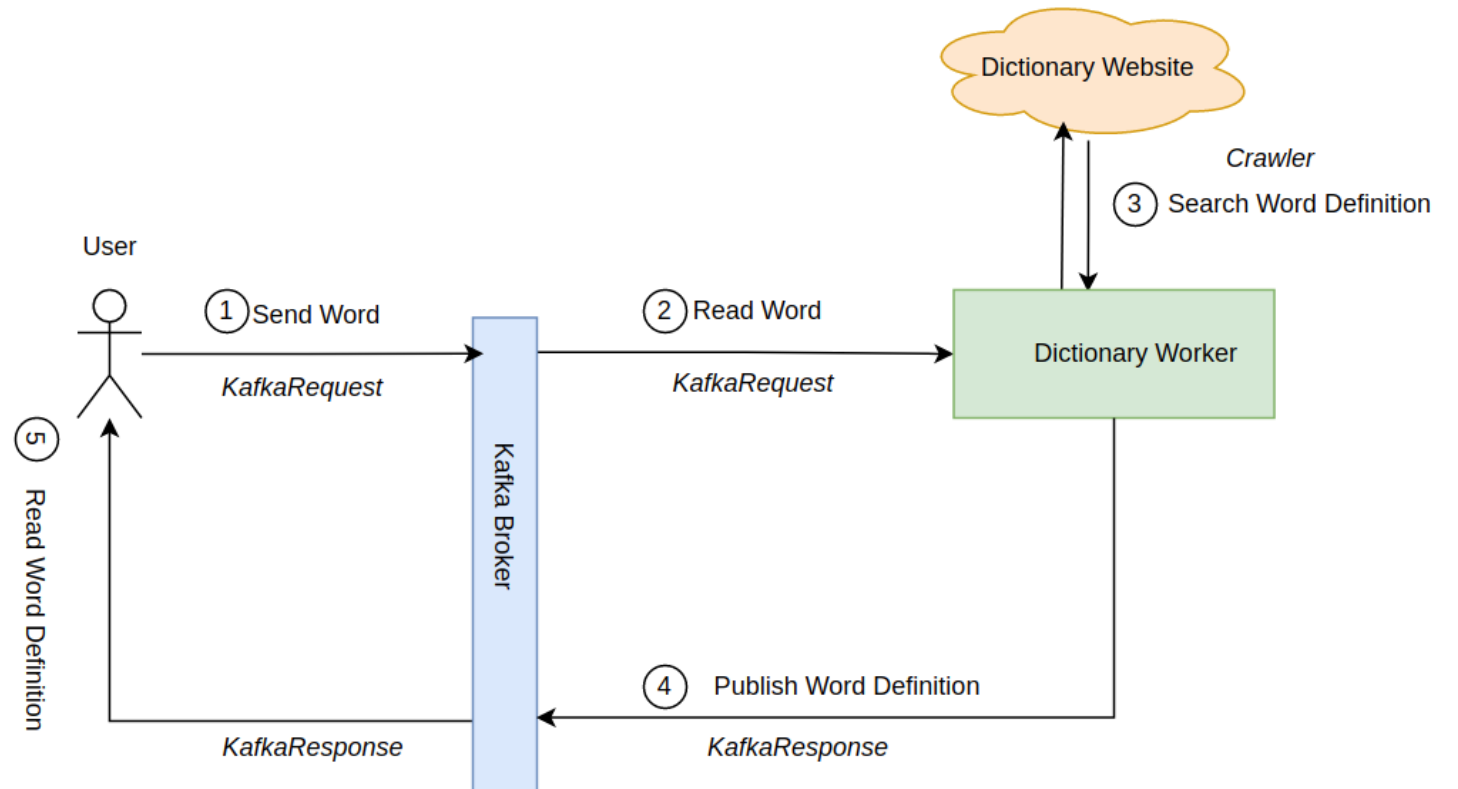
Practice 3: Advanced dictionary

- Homework
- Searching word definition for website
- Extend language support (ES, CN, GE, etc.)
- Further details are available on Github
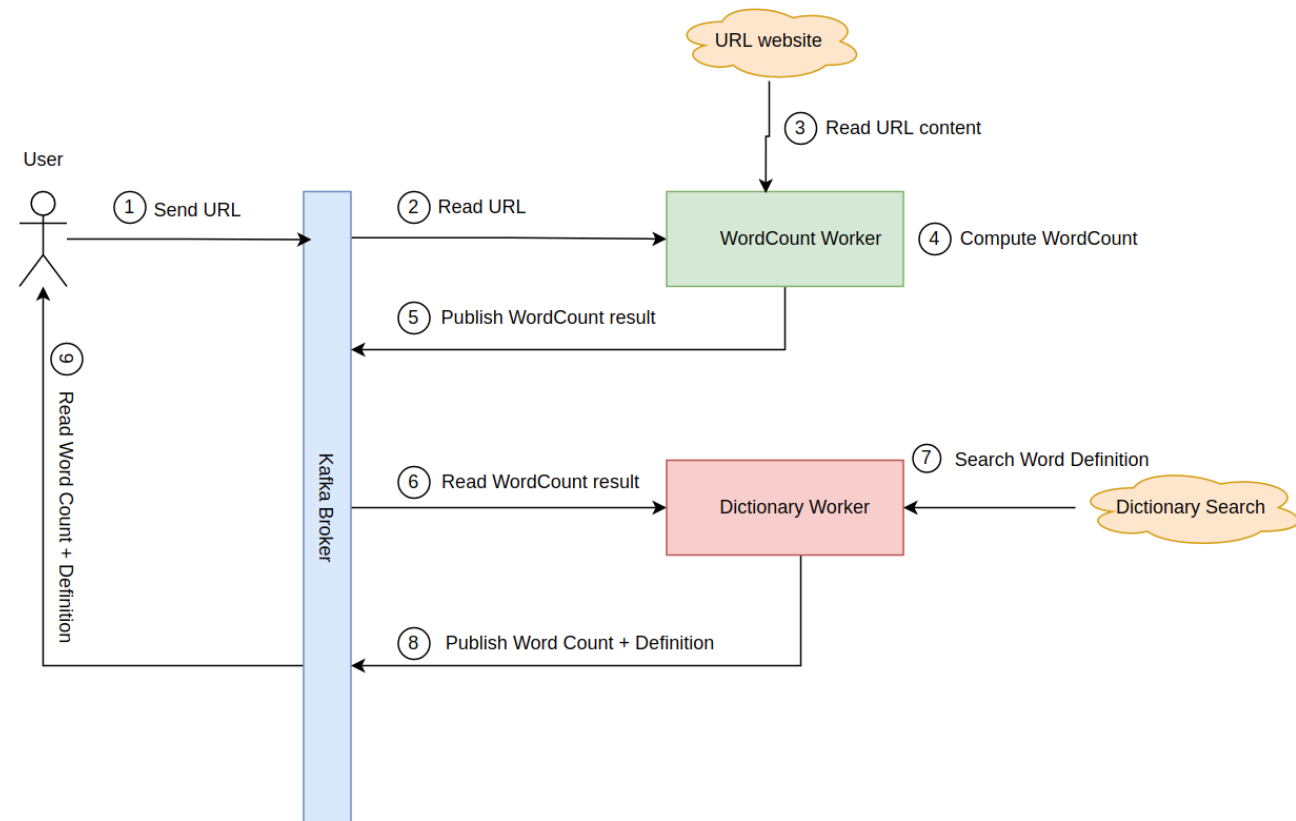
Troubleshooting

- Raise any question during the session or via email
  kodjo.klouvi@gmail.com