Kafka

Big Data E22

Nikolai Emil Damm - nidam16@student.sdu.dk
Bende Siewertsen - besie18@student.sdu.dk

General

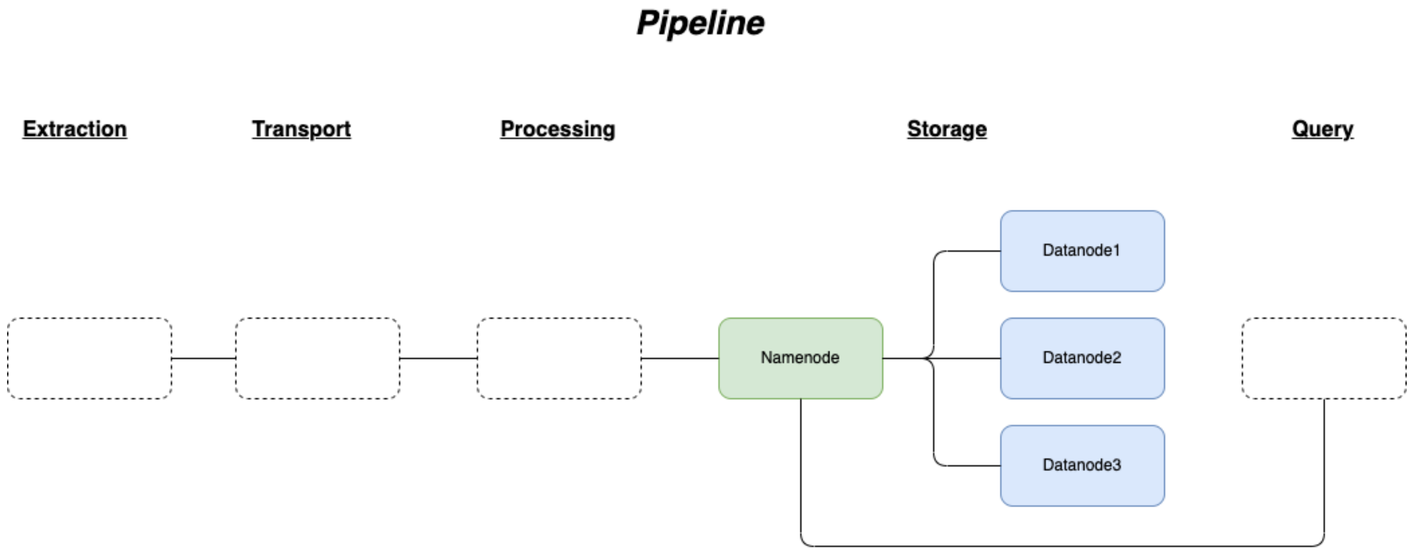
Exercise repository

- We will use a public repository for all coming lectures. You can fork it to have your own version.
  - Use the sync option visible on your forked version to keep your repo updated with the origin.
- The repo: https://github.com/jakobhviid/BigDataCourseExercises

Context

What are we doing?

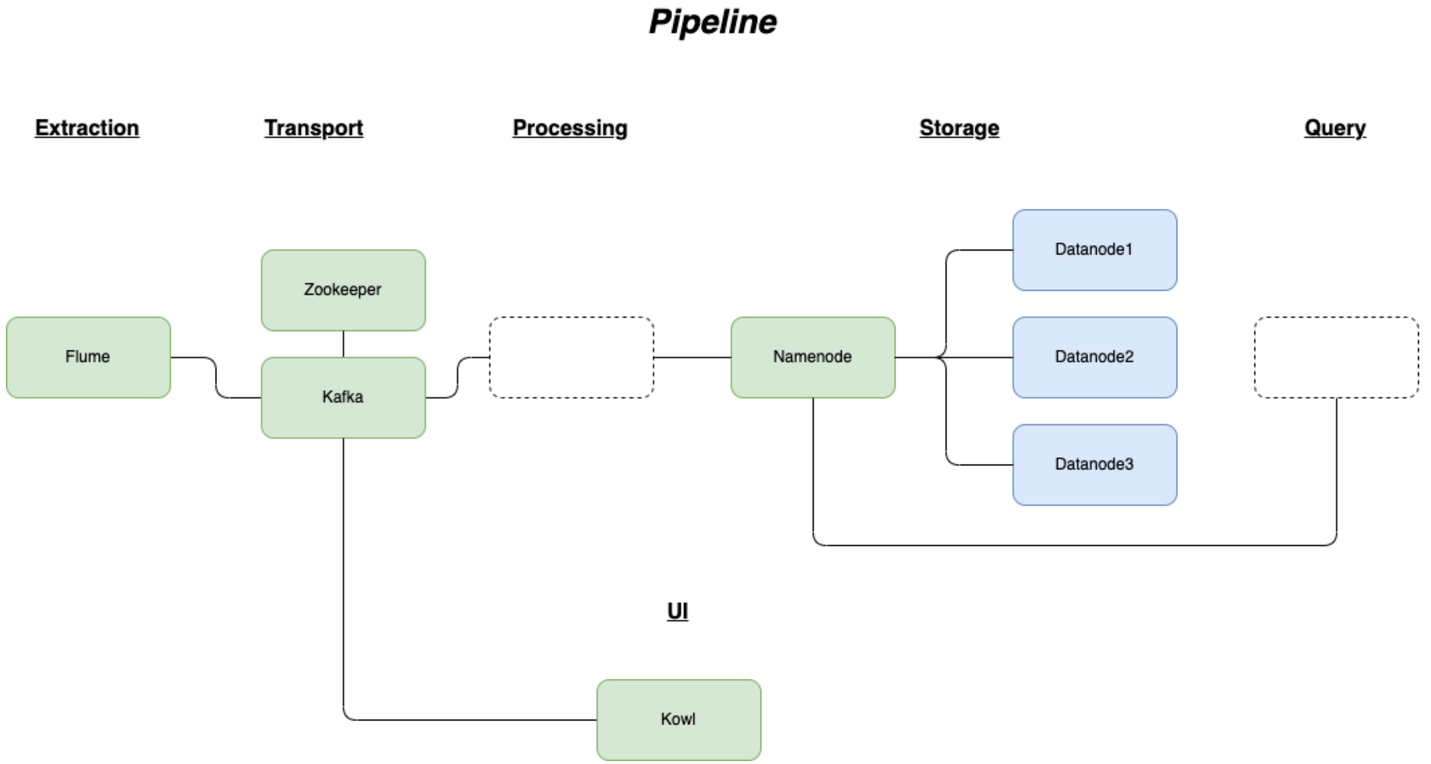
- A data-pipeline to count how many “The” words are in Alice in Wonderland
- Last time we introduced storage



Context

What are we doing today?

- Setting up a Kafka cluster
  - 1 Kafka broker
  - 1 Zookeeper node
  - 1 Kowl instance
  - 1 network



Kowl

Interacting with Kafka

- Quick demonstration of using Kowl.

Exercise 01

Composing a Kafka Cluster

- To work with Kafka we need to setup a Kafka cluster with Zookeeper, a Kafka broker and Kowl.
- To do this you need to do the following:
    1. cd ./lecture03/
    2. Examine the docker-compose.yml file
        1. How is the Environment variable ZOOKEEPER_CLIENT_PORT related to the advertised docker ports?
        2. What purpose does the KAFKA_ADVERTISED_LISTENER have?
        3. How do you access Kowl in the browser?
    3. Run docker compose up -d in ./lecture03
    4. What did the command do?
        - You should get familiar with the docker-compose.yaml file!

Exercise 02

Interacting with Kafka using Kowl

1. What does the Brokers view show you?
    1. What is the default value for log.retention.hours?
    2. What is the default offsets.topic.replication.factor? And what is the current value?
2. What does the Topics view show you?
    1. Use Kowl to create a topic.
    2. Use Kowl to insert a JSON message (it is possible).
    3. Use Kowl to insert a text message.
    4. Use Kowl to insert a message with a key and a value that matches.
    5. Use Kowl to delete the oldest message.

Exercise 03

Create topics on Kafka using the CLI

1. Exec into the cp-kafka container
2. Use kafka-topics to create a topic
    1. —bootstrap-server kafka:9092
    2. —create —if-not-exists —topic <topic name>
    3. —partitions <partition amount>
    4. —replication-factor <replication amount>
3. Use kafka-topics to check what is in the partition
    1. —bootstrap-server kafka:9092
    2. —describe —topic <topic name>

Exercise 04

Consume and produce message on Kafka using the CLI

1. Open two terminals.
2. Exec into the cp-kafka container in both terminals
3. In the first terminal
    1. Use kafka-console-consumer to consume messages from a topic
        1. —bootstrap-server kafka:9092
        2. —topic <topic name>
4. In the second terminal
    1. Use kafka-console-producer to produce messages to a topic
        1. —bootstrap-server kafka:9092
        2. —topic <topic name>
5. Try to send messages using the producer and see what happens in the consumer!
    1. You can also check Kowl to visualize your actions

Exercise 05

Consume messages on Kafka using Python

- Now we want to access Kafka from a client. To do this we will create a Python client that can consume from Kafka.
- Start the Kafka consumer
  - Disclaimer: If you do not like Python, you can try with something else (there are client libraries for Kafka for most programming languages), but throughout this course we will use Python.

    1. cd ./lecture03/consumer-client to change your working directory to the specified path.
    2. Examine the Dockerfile, the example.py, and the requirements.txt file. You should become familiar with these files as they are the basis of your Python Client.
    3. docker build . -t python-consumer-client:latest to build a docker image with the python client.
    4. docker run —rm -it —network shared_network —name python-consumer-client python-consumer-client to run the docker image as a new container.
       - —rm ensures that the container you started is removed after exiting.
       - —name sets the name of the running container, so it is easier for you to exec into it etc.
    5. Use Kowl to produce message and check if the consumer has received the message.
    6. DO NOT CLOSE YOUR CONSUMER! You will need it for the next exercise.

Exercise 06

Produce messages to Kafka using Python

- Now we want to access Kafka from a client. To do this we will create a Python client that can produce messages to Kafka.
- Start the Kafka producer
    - Disclaimer: If you do not like Python, you can try with something else (there are client libraries for Kafka for most programming languages), but throughout this course we will use Python.

    1. cd ./lecture03/producer-client to change your working directory to the specified path.
    2. Examine the Dockerfile, the example.py, and the requirements.txt file. You should become familiar with these files as they are the basis of your Python Client.
    3. Upps.. where is the implementation? Now you customer might be mad, fix it fast!
        1. Implement a kafka producer with library we used for the consumer
        2. Implement it so it sends a series of messages and shuts down afterwards
        3. Implement it to it listens for your input.
    4. docker build . -t python-producer-client:latest to build a docker image with the python client.
    5. docker run —rm -ti —network shared_network —name python-producer-client python-producer-client to run the docker image as a new container.
        - —rm ensures that the container you started is removed after exiting.
        - —name sets the name of the running container, so it is easier for you to exec into it etc.

        1. Check your consumer, we hope you did not close it 😉

Exercise 07

Consume messages with two consumers

- Start another consumer
- Produce some messages
- Do both consumers receive the same messages?
  - Why? Why not?

Exercise 08

How can we get the consumers to receive the same messages?

1. Either change the topic partition using Kowl or using the CLI
2. Using the CLI
    1. Use kafka-topics
        1. --bootstrap-server kafka:9092
        2. --alter --topic <topic name>
        3. --partitions <partition amount>
3. Produce new messages and check the consumers
4. Did both consumers receive the messages? Why? Why not?

Exercise 09

Alice in Kafka-Land 👸🐇

1. Create a python client from the template in lecture03-exercises/alice-in-kafkaland-producer
    1. Create an insecure HDFS client.
    2. Read alice-in-wonderland.txt from HDFS.
    3. Create a Kafka producer that publishes each sentence of the fairy-tale to a kafka topic (alice-in-kafkaland)
2. Create a python client from the template in lecture03-exercises/alice-in-kafkaland-consumer
    1. Consume messages from the alice-in-kafkaland kafka topic
    2. Combine the messages to a single string
    3. Write the string to HDFS in a file called ‘alice-in-kafkaland.txt’
3. Start HDFS cluster found in previous lecture
4. Run the consumer first, then the producer.
    1. Check that

(Optional) Exercise

Produce messages from a file to Kafka using Flume

1. Check Documentation
    1. Flume user guide: https://flume.apache.org/FlumeUserGuide.html
        1. Examples for a flume configuration including a docker image can be found in lecture03-exercises/flume-example
        2. In the Dockerfile, there might be some changes required based on the chip architecture of your computer
        3. Try to publish Alice-in-wonderland.txt to kafka using flume.