# HDFS, Parquet and Avro 

## Big Data E22

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

# General

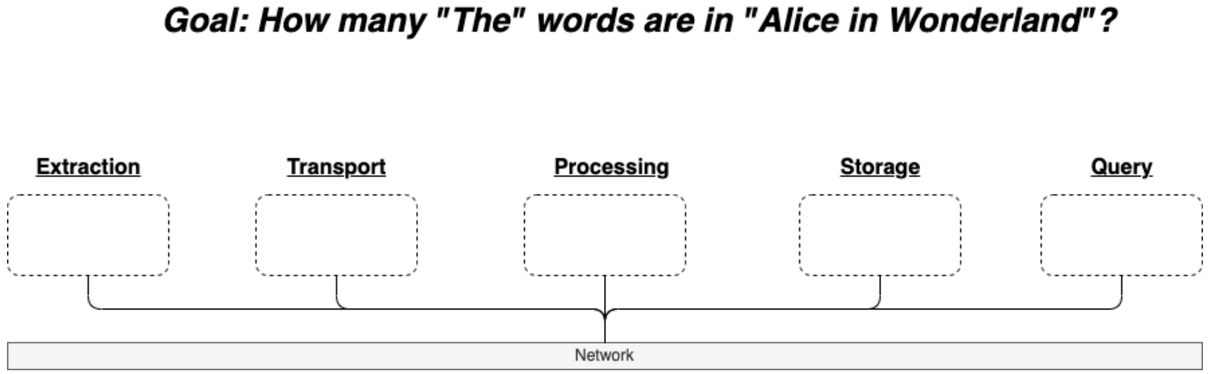
## Exercise repository

- 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?

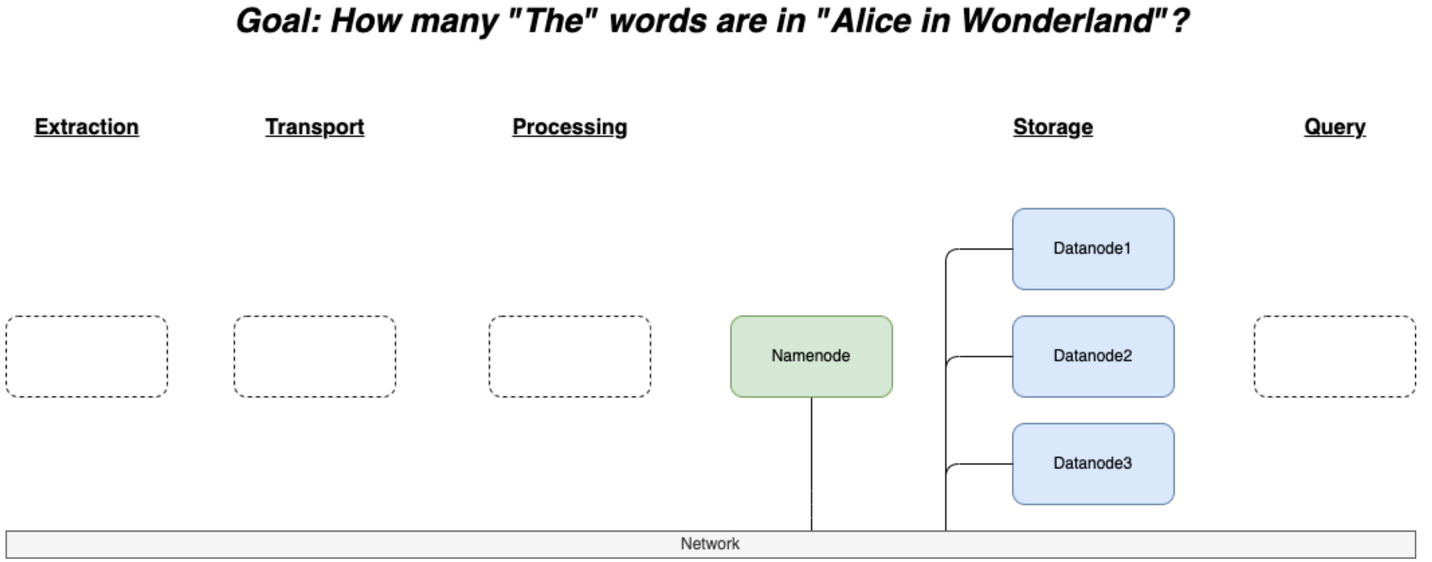
- A data-pipeline to count how many “The” words are in Alice in Wonderland



# Context

## What are we doing today?

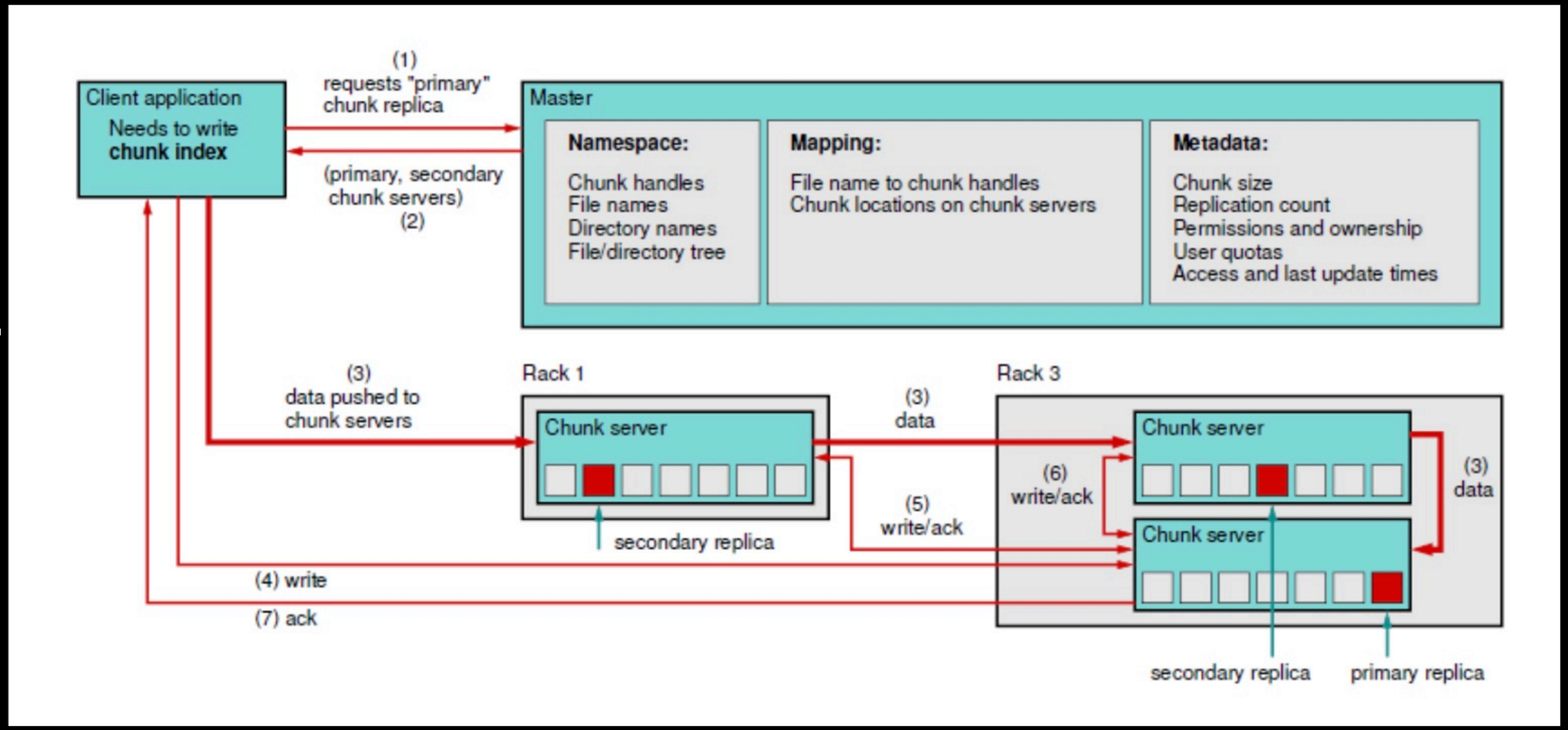
- Setting up a HDFS cluster
  - 1 namenode
  - 3 datanodes
  - 1 network



# HDFS

## What is a namenode?

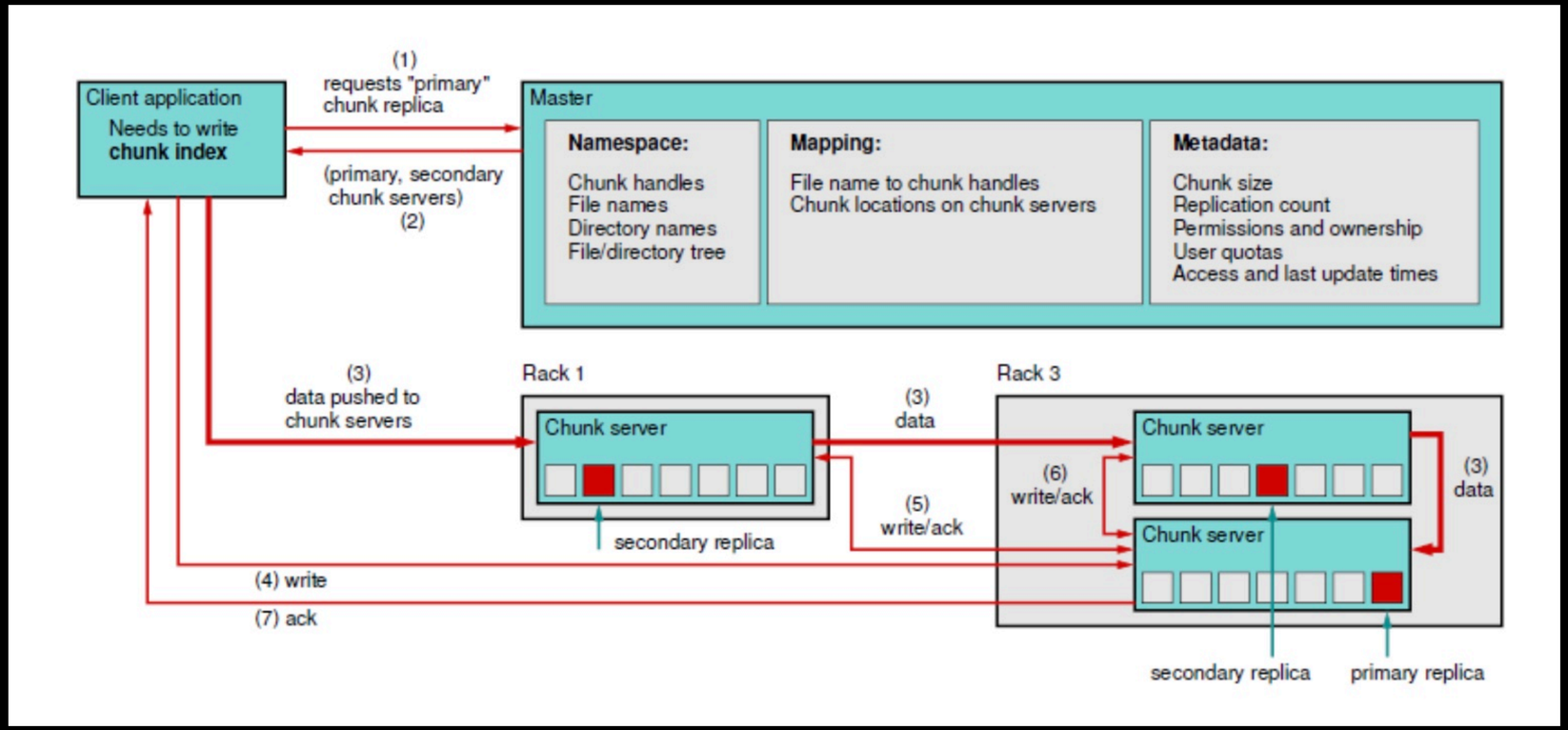
- Knows where data is located.
  - Metadata - Information that describes the data.
  - Chunks - A part of the data.
  - Mapping - We ask the namenode where to read/write our data.



# HDFS

## What is a datanode?

- Contains the data.
  - Chunks
  - Replicas
  - High availability requires 3 nodes.



# HDFS

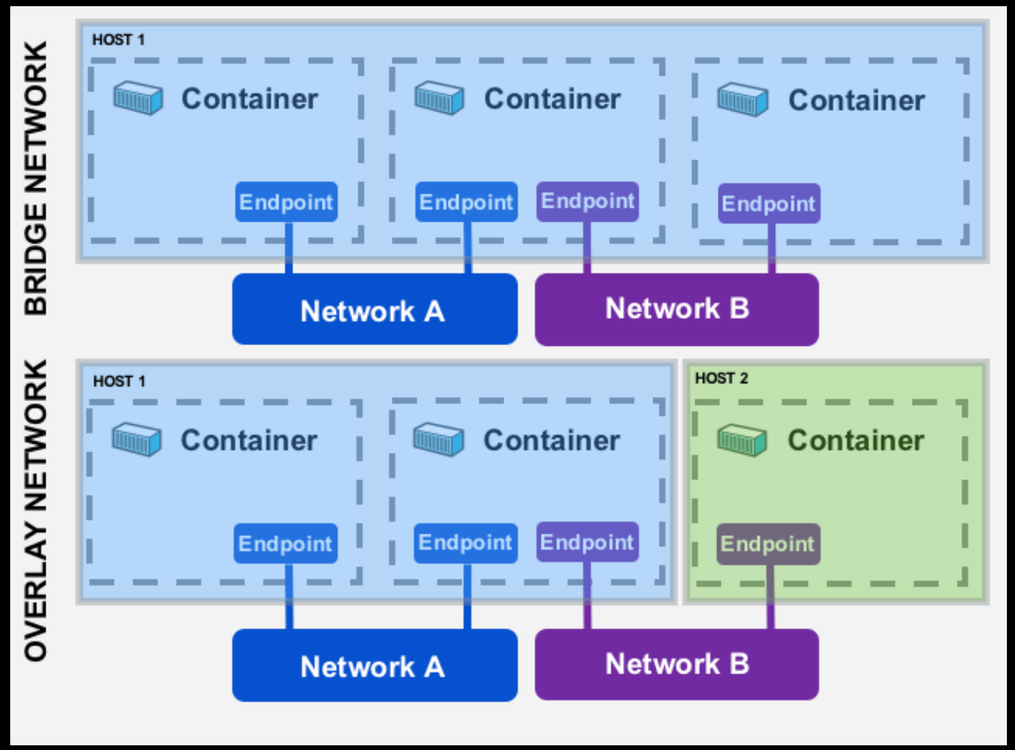
## What is a docker network?

- Do anyone know what a docker network is?

# HDFS

## What is a docker network?

- Do anyone know what a docker network is?
- Bridge networks
- Overlay networks
- Allow containers to connect to each other.



# Exercise 01

## Composing a HDFS Cluster

- To work with HDFS we need to setup a HDFS cluster.
- To do this you need to do the following:
  1. `cd ./lecture02/`
  2. Run `docker compose up -d` in `./lecture02`
  3. What does `-d` mean/do?
  4. What did the command do?
     - You should get familiar with the docker-compose.yaml file!

# Exercise 02

## Accessing the namenode and interacting with the HDFS CLI

- Now that we have a HDFS cluster, lets try and access a node, and play around.
- To access the HDFS cluster do the following:
    1. `docker exec -ti namenode /bin/bash`
- Now we have access to the namenode container that is able to interact with the HDFS command-line interface (CLI).
- Now lets try to run some HDFS shell commands, and see what they do 👀.
    - We will use the `hdfs dfs -[command] [path]` command
    2. `hdfs dfs -ls [path]` to list all files in a specified folder.
    3. `hdfs dfs -cat [path]` to print the contents of a file on a specific path.
    4. `hdfs dfs -put [sourcePath] [targetPath]` to add a file to hdfs. The source path should point to a file inside the container, and the target path should point to a path inside hdfs.
        - You might want to create a file first:
            - `touch fileName.txt` to create a file.
            - `apt update && apt install nano -y` to update the apt registry and install the nano editor.
            - `nano fileName.txt` to change the contents of the file.

# Exercise 03

## Adding data to HDFS

- We should add some adventurous data to HDFS, so let’s download a book we all know!

    1. `apt update && apt install -y wget` to update the apt registry and install wget (a tool to download files through HTTP from a linux terminal).

    2. `wget -O alice-in-wonderland.txt https://www.gutenberg.org/files/11/11-0.txt` to download the fairy tale Alice In wonderland to the namenode.

    3. `hdfs dfs -put alice-in-wonderland.txt /` to add the fairy tale to HDFS.

# Exercise 04

## Access txt data on HDFS from a Python client (Read and write)

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

1. `cd ./lecture02/simple-python-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-client:latest` to build a docker image with the python client.
4. `docker run -rm —network hadoop —name python-client python-client` to run the docker image as a new container.
   - -rm ensures that any existing python-client container is removed and replaced by a new python-client container.
   - —name sets the name of the running container, so it is easier for you to exec into it etc.
5. What is the content of the /write.txt file? How does this relate to the Alice in Wonderland text?

# Exercise 05

## Access JSON data on HDFS from a Python client (Counting words)

- Now that we have a python client that can read and write to HDFS, we can try to write and read JSON from HDFS.

    1. `cd ./lecture02/json-word-count` 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-client:latest` to build a docker image with the python client.

    4. `docker run -rm —network hadoop —name python-client python-client` to run the docker image as a new container.

    5. What are the five most common words in Alice in Wonderland, and how many times are the repeated?

# Exercise 06

## Access Avro data on HDFS from a Python client (Counting words)

- Now it is getting exciting, because now we need to work with a new file format, for many of you. Lets try and read and write data to Avro from our Python client.
    1. `cd ./lecture02/avro-word-count` 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-client:latest` to build a docker image with the python client.
    4. `docker run -rm —network hadoop —name python-client python-client` to run the docker image as a new container.
    5. How does the Avro Schema look?
    6. What are the five most common words in Alice in Wonderland, and how many times are the repeated?

# Exercise 07

## Access Parquet data on HDFS from a Python client (Counting words)

- Oh no! Our Python client is not complete, and now our customer want us to write data to Parquet. What the heck is Parquet?? Lets try and solve this issue before the customer abandons us!! 😭

    1. `cd ./lecture02/parquet-word-count` 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. Write the missing code in the example.py file.
        - You can use the pyarrow.parquet, pandas and pyarrow libraries to read and write from parquet
            - Parquet Docs: https://fastparquet.readthedocs.io/en/latest/
            - Pyarrow Docs: https://arrow.apache.org/docs/python/parquet.html
        - Pandas allows you to create data frames from text.
        - Pyarrow allows you to create a pandas table from a data frame.
        - Pyarrow.parquet allows you to write pandas tables to parquet files.
    4. `docker build . -t python-client:latest` to build a docker image with the python client.
    5. `docker run -rm —network hadoop —name python-client python-client` to run the docker image as a new container.
    6. How does the Parquet Schema column names look?
    7. What are the five most common words in Alice in Wonderland, and how many times are the repeated?

# Exercise 08

## Teardown of cluster and resources

1. `docker image ls -a` to see all images.
    - Notice there are quite a few dangling images (images that are marked as <none>)
    - Dangling images are images you have run, that have exited. These take up space on your SSD/HDD. So we should remove them.
2. Cleanup images by running `docker image prune`
    - You can also do a full cleanup by running `docker system prune`. Be aware that this will remove:
        - All stopped containers
        - All networks not used by at least one container
        - All dangling images
        - All dangling build cache.
    - Most of the time this is a non issue, but sometimes it is not!

- What about containers?
    - Notice that we have been running all containers with the `-rm` argument which removes the container after it is terminated. So no cleanup is needed.
- What about the compose HDFS cluster?

3. `cd ./lecture02` to change your working directory to the specified path.
4. Run `docker compose down` to teardown the cluster.