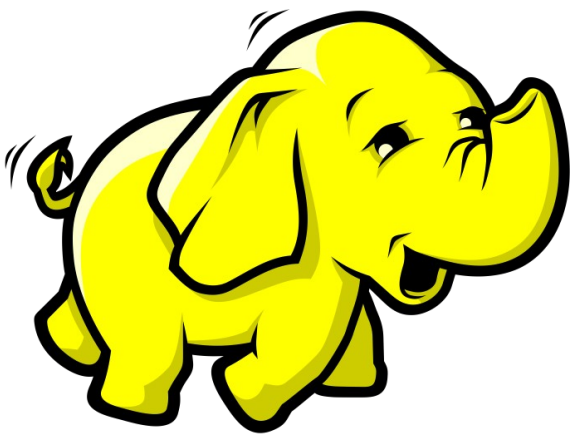
Hadoop Stack



The Hadoop stack

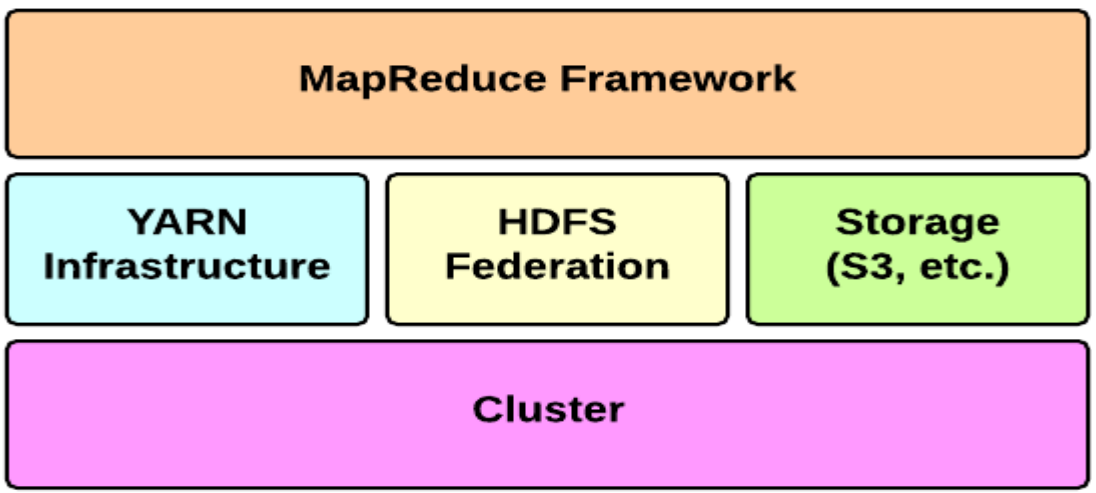


Image from: http://ercoppa.github.io/HadoopInternals/

Distributed storage

- Assumptions:
  - Large files (Terabytes)
  - Write once, read many
- Challenges:
  - Throughput
  - Reliability
  - Locality

Hadoop Distributed File System

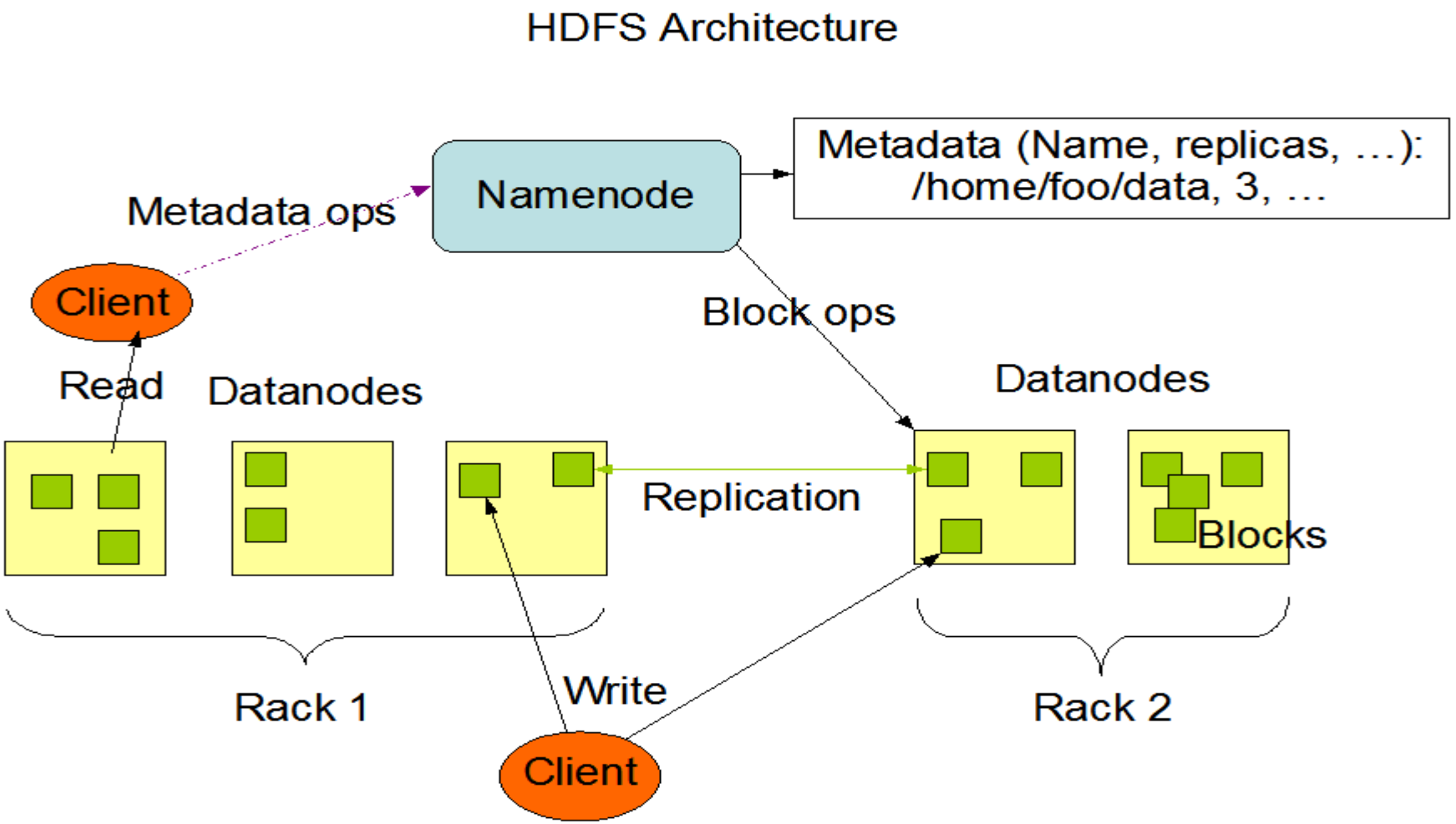


Image from: https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Map Reduce

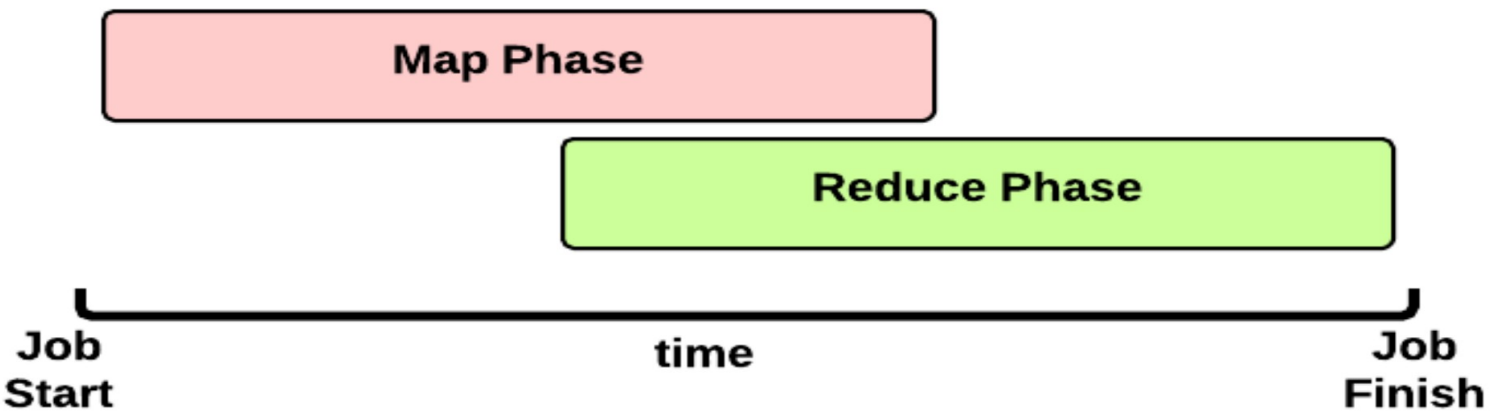


Image from: http://ercoppa.github.io/HadoopInternals/

MapReduce tasks

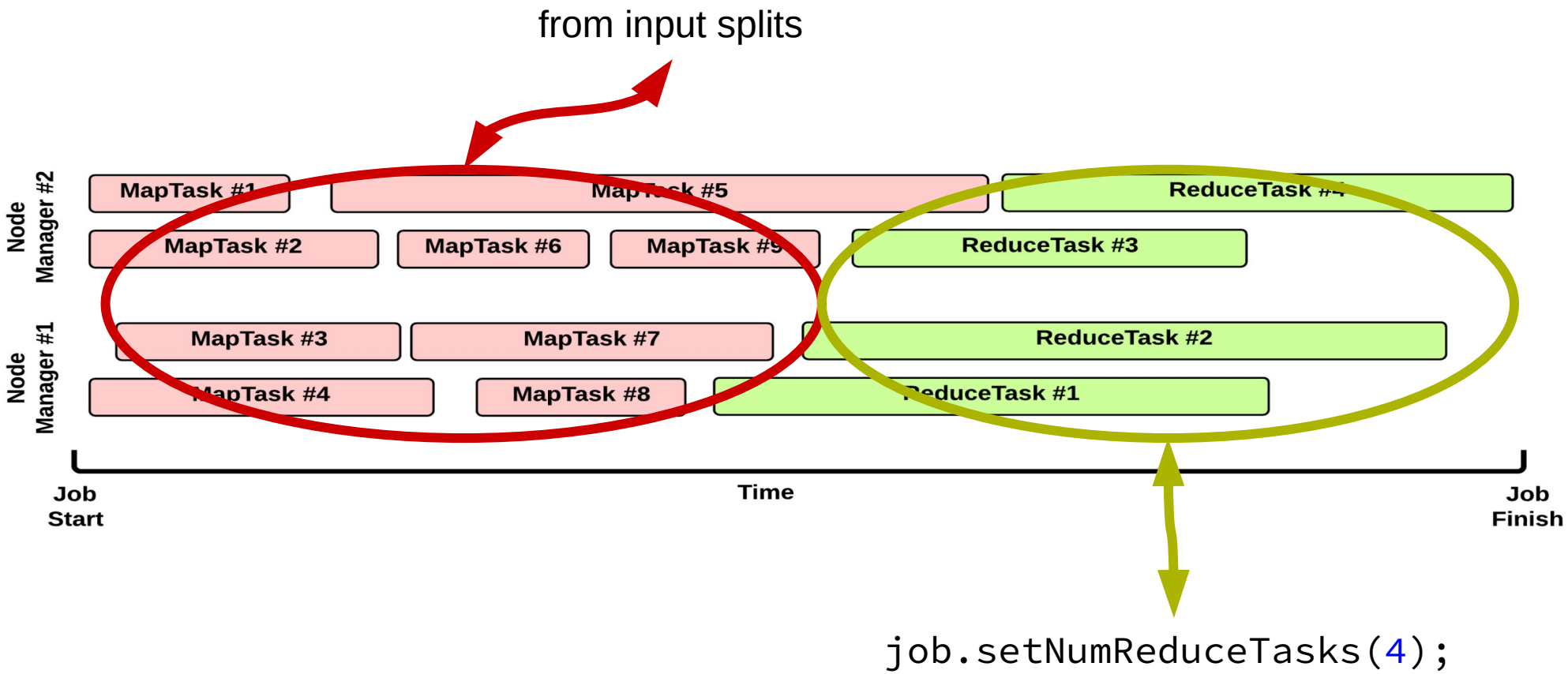
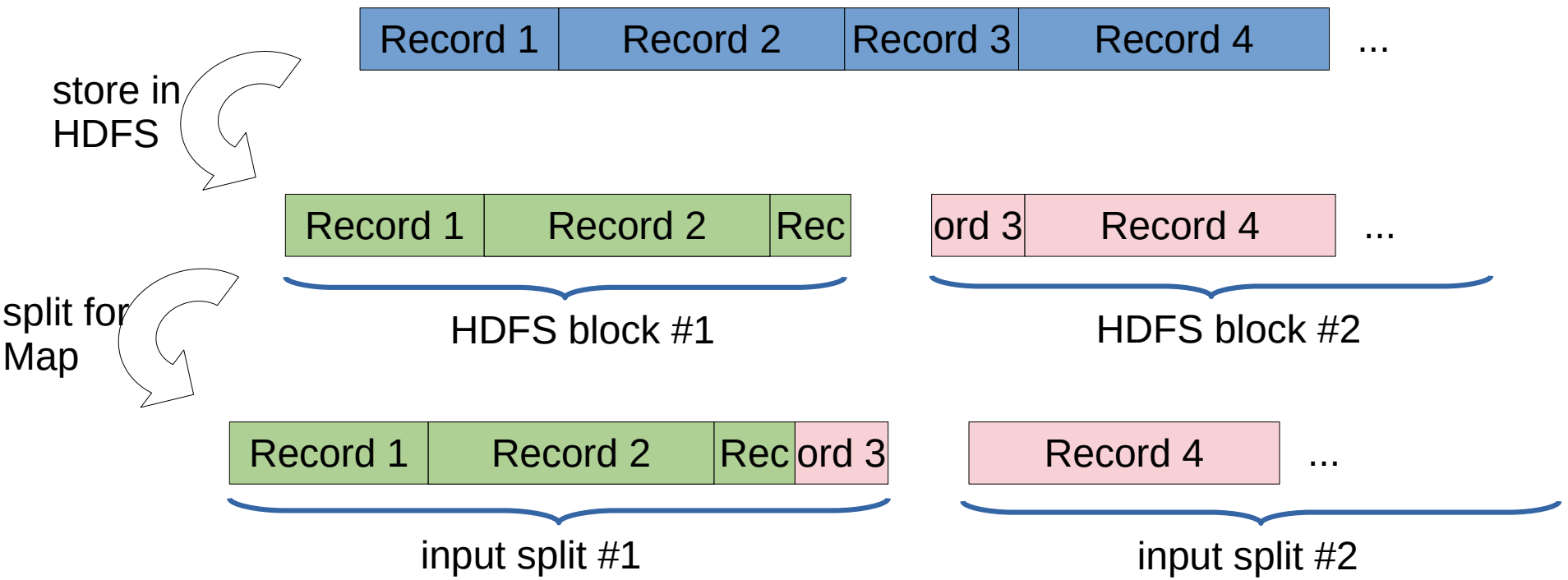


Image from: http://ercoppa.github.io/HadoopInternals/

Input splits



Input splits

- How to split:
  - Keep records aligned with blocks
  - Keep indexes for start of records
  - Search for record separator (e.g., ‘\n’)
  - Do not allow splits (e.g., .gz files)
- Used to launch task container close to corresponding data

Map stage

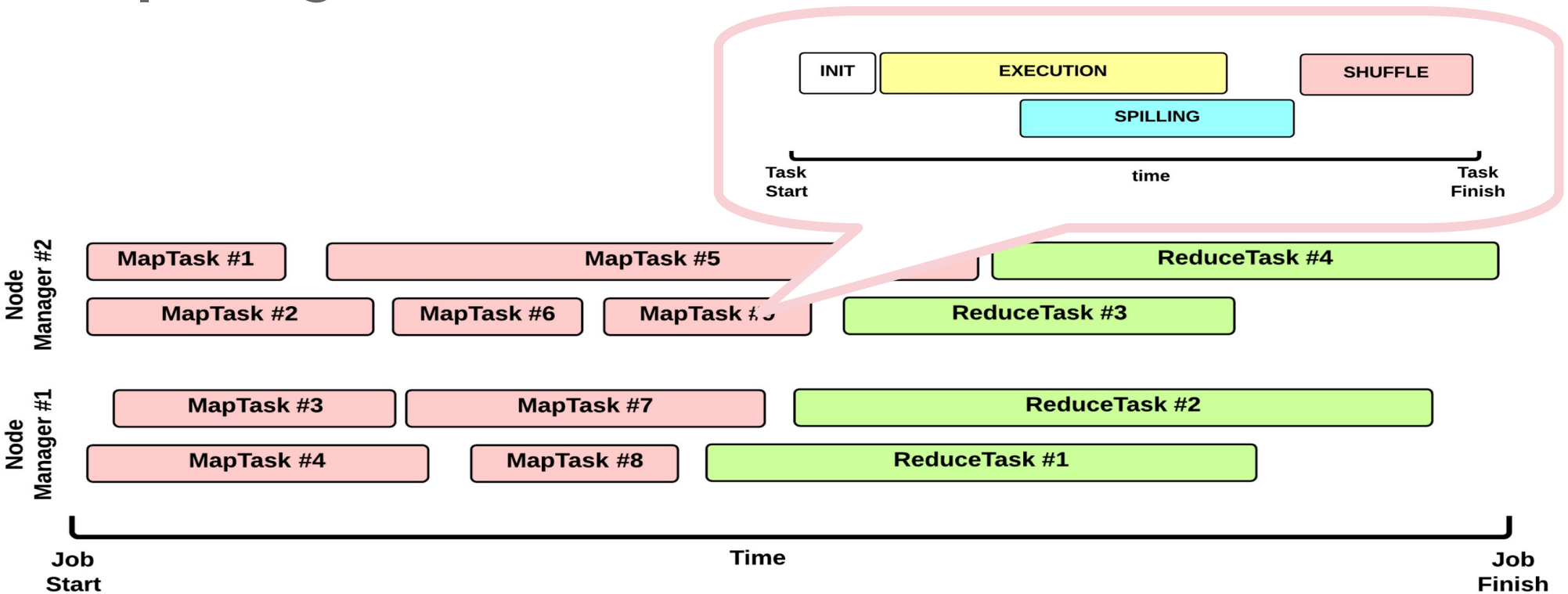


Image from: http://ercoppa.github.io/HadoopInternals/

Map spilling

- Map results are kept in memory and then sorted and stored on disk (“spilled”)

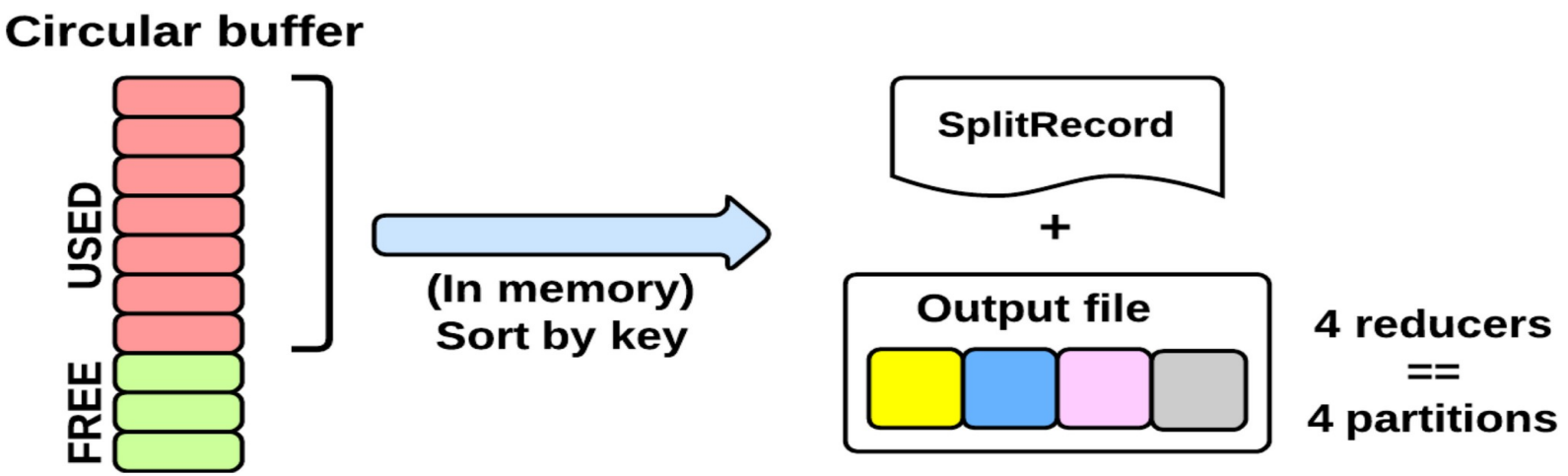


Image from: http://ercoppa.github.io/HadoopInternals/

Map shuffle

- “Spill” files are then combined using external merge sort:

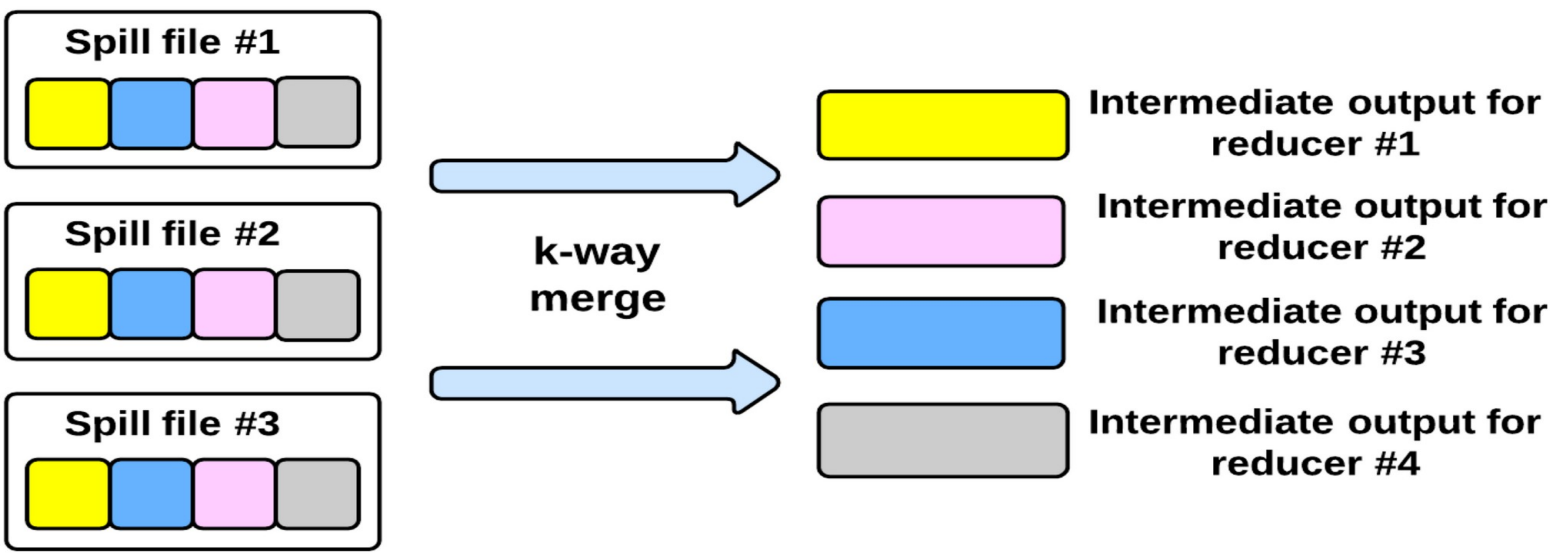


Image from: http://ercoppa.github.io/HadoopInternals/

Reduce stage

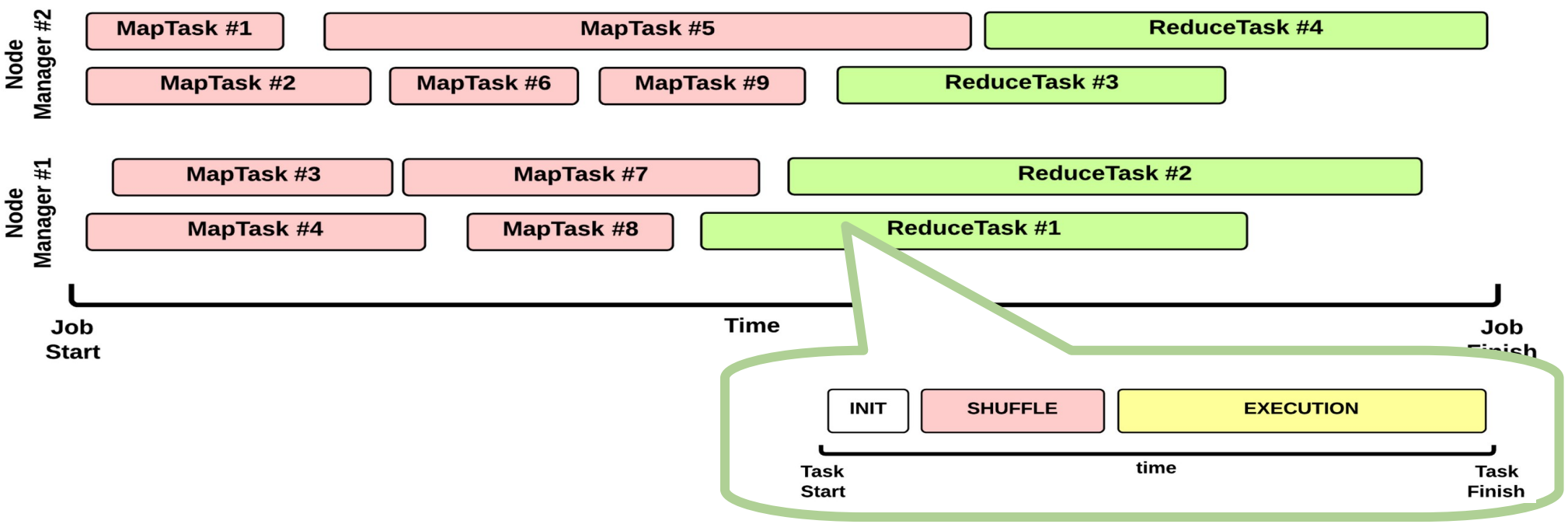
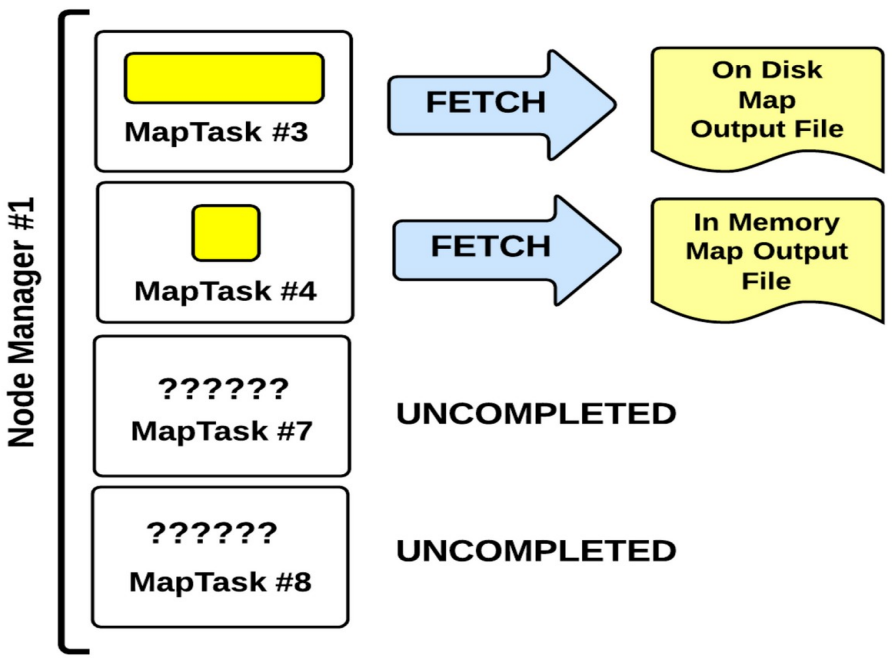


Image from: http://ercoppa.github.io/HadoopInternals/

Reduce shuffle



- Reducer tasks fetch corresponding files from mappers
  - Bigger files to disk
  - Smaller files are stored in memory and combined

Image from: http://ercoppa.github.io/HadoopInternals/

Reduce shuffle

- All map output files are finally combined for the reducer task

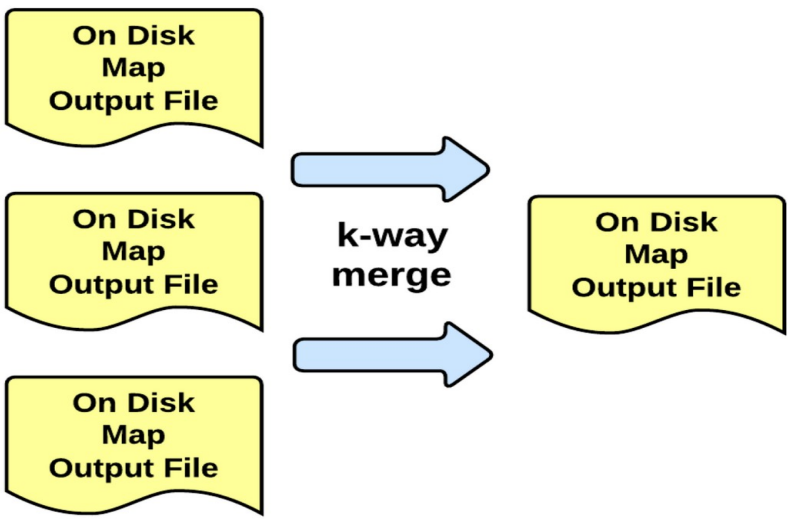


Image from: http://ercoppa.github.io/HadoopInternals/

Map Reduce summary

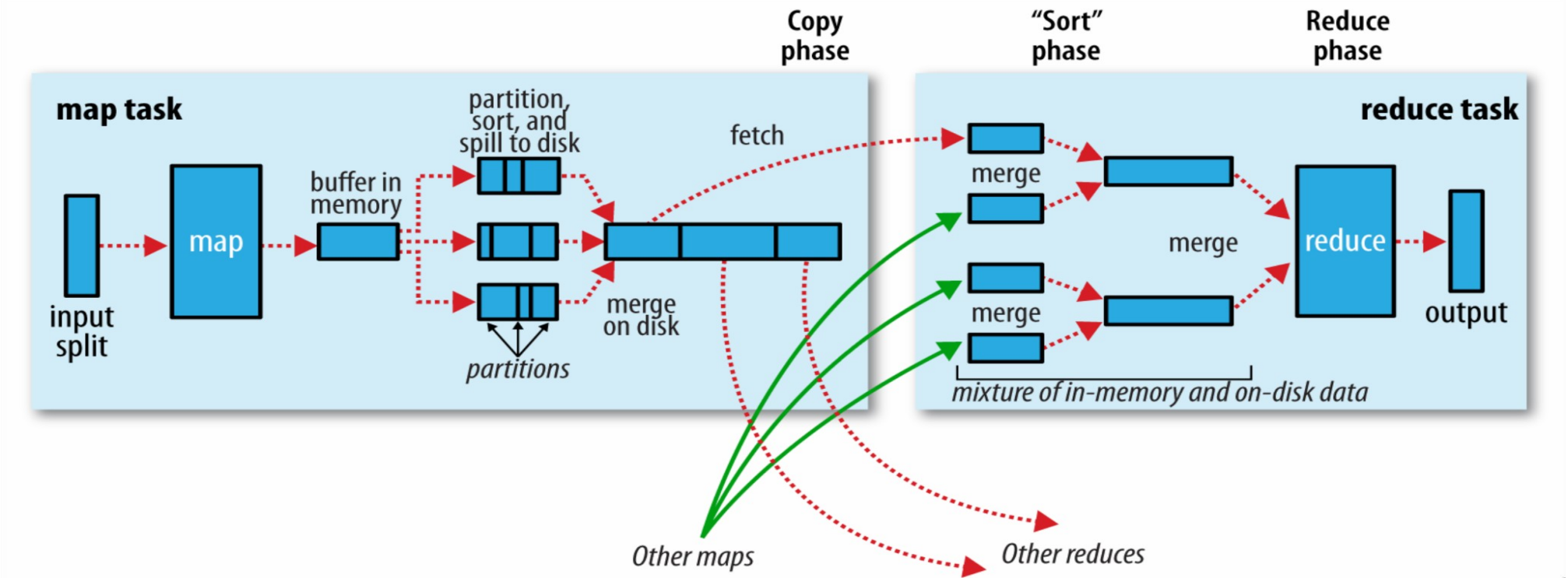


Image from: Tom White. Hadoop: The Definitive Guide (4th Edition).

Deployment

Docker deployment

- Check out and deploy:

  $ git clone https://github.com/big-data-europe/docker-hadoop.git

  $ docker-compose pull

  $ docker-compose up

- File system access (inside a connected container):

  # hdfs …

  … dfs -put url /dir
  (-D dfs.blocksize=bytes)

  … dfs -ls, -mkdir, -rm, -rmr, ...

  … fsck / -files -blocks -locations

Examples

- Loading files from the Web:

  $ docker exec -it namenode bash

  # curl https://... | hdfs dfs -put - /filename

- ... and decompressing GZip on-the-fly:

  $ docker exec -it namenode bash

  # curl https://..../.../filename.gz | gzip -d | hdfs dfs -put - /filename

- Uploading local files:
  - $ docker run --env-file hadoop.env \

    --network docker-hadoop_default \

    -v /home/me/somefolder:/data \

    -it bde2020/hadoop-base \

    hdfs dfs -put /data/somefile /filename

Packaging

- No need to include Hadoop dependencies or log4 configuration:

```
<dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
        <version>3.2.1</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Docker deployment

- Sample Dockerfile:

```
FROM bde2020/hadoop-base
COPY target/jarname.jar /
CMD ["hadoop", "jar", "/jarname.jar", "mainclassname"]
```

- Run options:

  --network docker-hadoop_default

  --env-file /...path-to.../docker-hadoop/hadoop.env

- Clean up with:

  $ docker-compose down --volumes