# LAPORAN TUGAS PERTEMUAN 7

# BIG DATA

**Oleh :**

**MOCHAMMAD ZAKARO AL FAJRI 2241720175**

## D-IV TEKNIK INFORMATIKA

## JURUSAN TEKNOLOGI INFORMASI

## POLITEKNIK NEGERI MALANG

## 2025

**Tugas 7 – Spark Docker**

**Praktikum :** Interaksi dengan Spark di Lingkungan Windows Menggunakan Docker

**Hasil :**

1. Pull Image Spark Resmi, dengan menggunakan perintah :
   **docker pull apache/spark:latest**
   Bukti :

```
Terminal

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
0f3083818c14: Pull complete
4f4fb700ef54: Pull complete
391ef20df327: Pull complete
d3c7b6bd77aa: Pull complete
5762a181dda2: Pull complete
4d9bb71a5e54: Pull complete
d9802f032d67: Pull complete
3058f73b8f49: Pull complete
Digest: sha256:39321d67b23e2e0953f81b60778f74bf40c40a18dfb0e881e6a38593af60afa1
Status: Downloaded newer image for apache/spark:latest
docker.io/apache/spark:latest
PS C:\Users\KAKA>
```

2. Menjalankan Spark Master
   Sebelumnya buat docker network sebagai berikut :

```
PS C:\Users\KAKA> docker network create spark-net
6497f560747107a9eb717951db17363ae04edfd499ce60d2d28f836d6d
dffae0
```

   Kemudian jalankan spark-master dalam network tersebut :

```
PS C:\Users\KAKA> docker run -d -p 8080:8080 -p 7077:7077
--name spark-master --network spark-net -m 2g --cpus=2 apa
che/spark:latest /opt/spark/bin/spark-class org.apache.spa
rk.deploy.master.Master
d847dfe66deca9f661a4e4b2a6d5e4435062defc2fd045c8e08744c56a
f032da
```

3. Menjalankan Spark Worker dengan memperhatikan beberapa hal yaitu :
   a. Mengalokasikan resource misalnya 2G memori dan 2 core CPU
   b. Nama yang berbeda untuk membuat beberapa worker. Misalnya spark-worker1, spark-worker2, dan seterusnya

   Bukti :

```
PS C:\Users\KAKA> docker run -d --name spark-worker1 --network spark-net -m 2g -
-cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.w
orker.Worker spark://spark-master:7077 --memory 1g --cores 1
b624f3bedda96b31824e655db7c23cfa9e75db63f19553d9cad1ff968a4b6306
PS C:\Users\KAKA> docker run -d --name spark-worker2 --network spark-net -m 2g -
-cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.w
orker.Worker spark://spark-master:7077 --memory 1g --cores 1
fba813be0eaab8481d80cc6f0db0bbe4669ab376846bd8f28d75314a5584fdff
PS C:\Users\KAKA> docker run -d --name spark-worker3 --network spark-net -m 2g -
-cpus=2 apache/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.w
orker.Worker spark://spark-master:7077 --memory 1g --cores 1
```

**Containers** Give feedback

View all your running containers and applications. Learn more

| | | | | | | |
|---|---|---|---|---|---|---|
| spark-master | d847dfe66dec | apache/spark:latest | 7077:7077 Show all ports (2) | 0.22% | 8 minutes ago |
| spark-worker1 | abca593cdd8b | apache/spark:latest | | 0.23% | 1 minute ago |
| spark-worker2 | fc325ee22b9f | apache/spark:latest | | 0.19% | 58 seconds ago |
| spark-worker3 | 6fb1bfda1964 | apache/spark:latest | | 0.16% | 47 seconds ago |

4. Mengakses Spark Web UI pada url : http://localhost:8080/

**Spark Master at spark://172.18.0.2:7077**

**URL:** spark://172.18.0.2:7077
**Alive Workers:** 3
**Cores in use:** 3 Total, 0 Used
**Memory in use:** 3.0 GiB Total, 0.0 B Used
**Resources in use:**
**Applications:** 0 Running, 0 Completed
**Drivers:** 0 Running, 0 Completed
**Status:** ALIVE

**Workers (3)**

| Worker Id | Address | State | Cores | Memory | Resources |
|---|---|---|---|---|---|
| worker-20250422093927-172.18.0.3-44569 | 172.18.0.3:44569 | ALIVE | 1 (0 Used) | 1024.0 MiB (0.0 B Used) | |
| worker-20250422093944-172.18.0.4-38895 | 172.18.0.4:38895 | ALIVE | 1 (0 Used) | 1024.0 MiB (0.0 B Used) | |
| worker-20250422093957-172.18.0.5-40647 | 172.18.0.5:40647 | ALIVE | 1 (0 Used) | 1024.0 MiB (0.0 B Used) | |

**Running Applications (0)**

| Application ID | Name | Cores | Memory per Executor | Resources Per Executor | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|---|

**Completed Applications (0)**

| Application ID | Name | Cores | Memory per Executor | Resources Per Executor | Submitted Time | User | State | Duration |
|---|---|---|---|---|---|---|---|---|

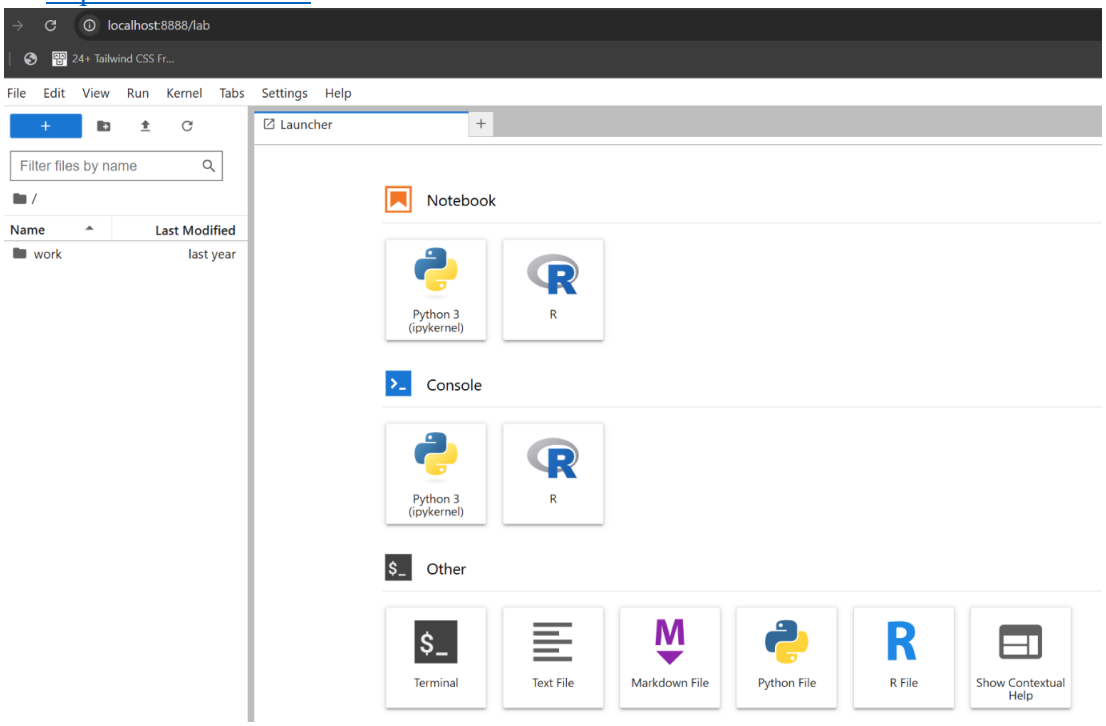
5. Menjalankan Spark Shell

```
PS C:\Users\KAKA> docker run -it --rm --name spark-shell --network spark-net --link spark-master:spark-master apache/spark:latest /opt/spark/bin/spark-shell
 --master spark://spark-master:7077
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/22 09:47:02 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://efdb3ab19486:4040
Spark context available as 'sc' (master = spark://spark-master:7077, app id = app-20250422094705-0000).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.5.5
      /_/

Using Scala version 2.12.18 (OpenJDK 64-Bit Server VM, Java 11.0.26)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

6. Menggunakan Jupyter Notebook dengan Spark dengan mengakses Jupyter Notebook di: http://localhost:8888




7. Untuk menghentikan container:
**docker stop spark-master spark-worker**
**docker rm spark-master spark-worker**

```
PS C:\Users\KAKA> docker stop spark-master spark-worker1 spark-worker2 spark-wor
ker3
spark-master
spark-worker1
spark-worker2
spark-worker3
PS C:\Users\KAKA> docker rm spark-master spark-worker1 spark-worker2 spark-worke
r3
spark-master
spark-worker1
spark-worker2
spark-worker3
```

**Contoh Program Word Count dengan Spark di Docker**

Berikut adalah contoh program Word Count (menghitung kemunculan kata) menggunakan Apache Spark yang bisa dijalankan di lingkungan Docker:

- **Cara 1: Menggunakan Spark Shell**
  a. Jalankan Spark Shell di Docker seperti contoh di atas

  b. Ketikkan kode berikut di Spark Shell:

```
scala> val textData = List("Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes Spark easy"
textData: List[String] = List(Hello Spark, Hello Docker, Spark is awesome, Docker makes Spark easy)

scala> val rdd = sc.parallelize(textData)
rdd: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> val wordCounts = rdd
  .flatMap(line => line.split(" "))      // Memecah setiap baris jadi kata
  .map(word => (word, 1))                // Membuat pasangan (kata, 1)
  .reduceByKey(_ + _)                    // Menggabungkan jumlah per kata
wordCounts: org.apache.spark.rdd.RDD[String] = ParallelCollectionRDD[0] at parallelize at <console>:24

scala> res0: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at flatMap at <console>:24

] = ShuffledRDD[3] at reduceByKey at <console>:24

scala> wordCounts.collect().foreach(println)
```

  c. Output :

```
scala> wordCounts.collect().foreach(println)
Hello
Spark
Hello
Docker
Spark
is
awesome
Docker
makes
Spark
easy
```

- **Cara 2: Menggunakan PySpark (Python)**
  a. Jalankan PySpark Shell di Docker

  b. Ketikkan kode Python berikut:

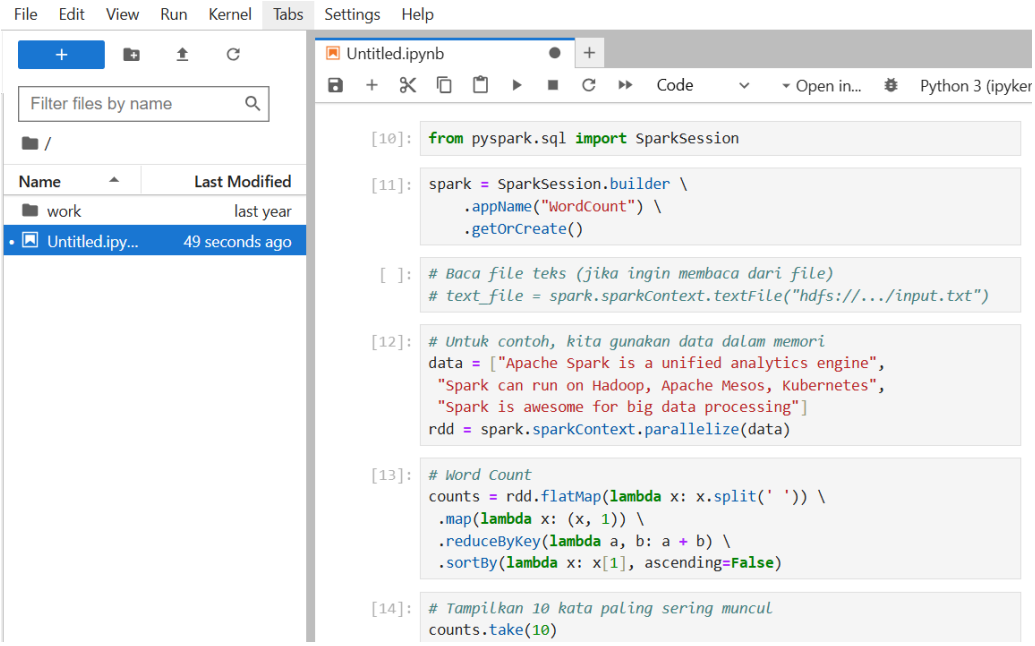
```
>>> spark = SparkSession.builder.appName("WordCount").getOrCreate()
>>> data = ["Hello Spark", "Hello Docker", "Spark is awesome", "Docker m
akes Spark easy"]
>>> rdd = spark.sparkContext.parallelize(data)
>>> word_counts = rdd.flatMap(lambda line: line.split(" ")) \
... .map(lambda word: (word, 1)) \
... .reduceByKey(lambda a, b: a + b)
>>> word_counts.collect()
```

c. Output :

```
>>> word_counts.collect()
[('Spark', 3), ('awesome', 1), ('Docker', 2), ('easy', 1), ('Hello', 2),
 ('is', 1), ('makes', 1)]
>>>
```

- **Cara 3: Menggunakan Jupyter Notebook**

a. Jika Anda menggunakan Jupyter Notebook (seperti di container jupyter/all-spark-notebook), ketikkan kode seperti berikut:




b. Output :



```
[14]: [('Spark', 3),
       ('is', 2),
       ('Apache', 2),
       ('Mesos,', 1),
       ('awesome', 1),
       ('run', 1),
       ('Kubernetes', 1),
       ('processing', 1),
       ('Hadoop,', 1),
       ('analytics', 1)]
```

- **Cara 4 : Menjalankan Program sebagai Script**

a. Buat file wordcount.py dengan isi berikut**:**

```
README.md M    wordcount.py ×
C: > Users > KAKA > wordcount.py > ...
from pyspark import SparkContext

# Buat Spark Context
sc = SparkContext(appName="SimpleWordCount")

# Data dummy
data = sc.parallelize(["hello world", "hello docker", "hello spark"])

# Wordcount
counts = data.flatMap(lambda line: line.split(" ")) \
            .map(lambda word: (word, 1)) \
            .reduceByKey(lambda a, b: a + b)

# Ambil hasil dan print
for word, count in counts.collect():
    print("%s: %i" % (word, count))

# Stop Spark Context
sc.stop()
```

b. Jalankan script, jangan lupa juga mendifinisikan network spark-net dengan kode berikut :

```
PS C:\Users\KAKA> docker run --rm --network spark-net -v ${PWD}:/app apache/spar
k:latest /opt/spark/bin/spark-submit --master spark://spark-master:7077 /app/wor
dcount.py
25/04/26 14:49:47 INFO SparkContext: Running Spark version 3.5.5
25/04/26 14:49:47 INFO SparkContext: OS info Linux, 5.15.167.4-microsoft-standar
d-WSL2, amd64
```

c. Output :

```
25/04/26 15:00:05 INFO DAGScheduler: Job 0 finished: collect at /app/wordcount.py:15, to
world: 1
docker: 1
hello: 3
spark: 1
25/04/26 15:00:05 INFO SparkContext: SparkContext is stopping with exitCode 0.
```