



NAMA : SUKMA BAGUS WAHASDWIKA

NIM : 2241720223

KELAS : TI - 3D

## BIG DATA - 07 Spark Docker

### Interaksi dengan Spark di Lingkungan Windows Menggunakan Docker

#### Prasyarat

1. Windows 10/11 (64-bit) dengan versi Pro, Enterprise, atau Education
2. Docker Desktop untuk Windows diinstal dan berjalan
3. WSL 2 (Windows Subsystem for Linux versi 2) diaktifkan

#### Langkah - langkah

##### 1. Pull Image Spark Resmi

`docker pull apache/spark:latest`

```
D:\POLINEMA\SEMESTER 6\Big Data>docker pull apache/spark:latest
latest: Pulling from apache/spark
391ef20df327: Pull complete
d9802f032d67: Pull complete
5762a181dda2: Pull complete
f937e0a2086c: Pull complete
d3c7b6bd77aa: Pull complete
0f3083818c14: Pull complete
b072aa17899d: Pull complete
4f4fb700ef54: Pull complete
1ba3910f6ba2: Pull complete
4d9bb71a5e54: Pull complete
3058f73b8f49: Pull complete
Digest: sha256:39321d67b23e2e0953f81b60778f74bf40c40a18dfb0e881e6a38593af60afa1
Status: Downloaded newer image for apache/spark:latest
docker.io/apache/spark:latest
```

##### 2. Menjalankan Spark Master

Sebelumnya buat docker network sebagai berikut

```
PS C:\Users\sukma bagus> docker network create spark-net
```

Kemudian jalankan spark-master dalam network tersebut

```
PS C:\Users\sukma bagus> docker run -d -p 8080:8080 -p 7077:7077 --name spark-master --network spark-net -m 2g --cpus=2 apache
/spark:latest /opt/spark/bin/spark-class org.apache.spark.deploy.master.Master
cf40a98f79ba4419a89be84e27d99a13aaee6a9b587da40db20c9b09d93f1b1f
PS C:\Users\sukma bagus>
```

##### 3. Menjalankan Spark Worker

```
PS C:\Users\sukma bagus> docker run -d --name spark-worker3 --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin/spark-class or
g.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
d2994679fada4fccccf506965f37bf2b41b675dbdc771c62e0de587289b19c751
PS C:\Users\sukma bagus>
```

jalankan perintah di atas beberapa kali dengan nama yang berbeda untuk membuat beberapa worker. Misalnya spark-worker1, spark-worker2, dan seterusnya



NAMA : SUKMA BAGUS WAHASDWIKA  
NIM : 2241720223  
KELAS : TI - 3D

## BIG DATA - 07 Spark Docker

### 4. Mengakses Spark Web UI

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 (4)

| Worker Id                              | Address          | State | Cores      | Memory                  | Resources |
|--|------------------|-------|------------|-------------------------|-----------|
| worker-20250422095351-172.18.0.3-36615 | 172.18.0.3:36615 | DEAD  | 1 (0 Used) | 1024.0 MiB (0.0 B Used) |           |
| worker-20250422095421-172.18.0.3-42661 | 172.18.0.3:42661 | ALIVE | 1 (0 Used) | 1024.0 MiB (0.0 B Used) |           |
| worker-20250422095441-172.18.0.4-42043 | 172.18.0.4:42043 | ALIVE | 1 (0 Used) | 1024.0 MiB (0.0 B Used) |           |
| worker-20250422095451-172.18.0.5-39111 | 172.18.0.5:39111 | 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

```
Terminal
PS C:\Users\sukma bagus> 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 10:05:16 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Spark context Web UI available at http://63b68e7fc121:4040
Spark context available as 'sc' (master = spark://spark-master:7077, app id = app-20250422100517-0000).
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.
```



NAMA : SUKMA BAGUS WAHASDWIKA  
NIM : 2241720223  
KELAS : TI - 3D

## BIG DATA - 07 Spark Docker

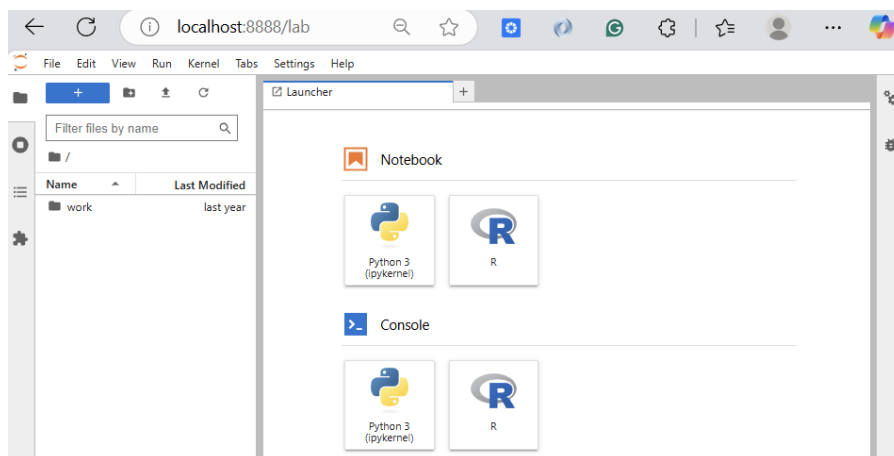
### 6. Menggunakan Jupyter Notebook dengan Spark

```
PS C:\Users\sukma bagus> docker run -it -p 8888:8888 -p 4040  
:4040 --network spark-net jupyter/all-spark-notebook
```

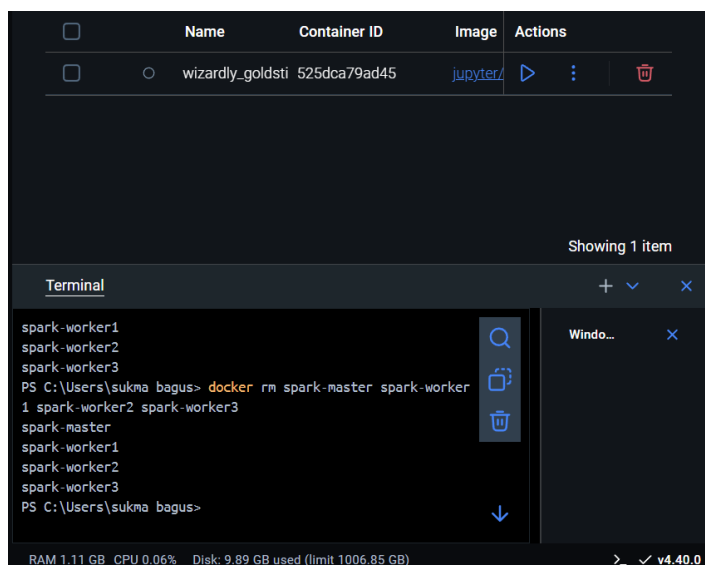
```
Terminal
7024bb03412a: Pull complete
7c128e9d2ddd: Pull complete
a21a167f7127: Pull complete
9a165b6e9dc7: Pull complete
Digest: sha256:b63bac2d9d34779ac969deeb4834efd838991f77269ca9a76bf6bd1f8678d29
Status: Downloaded newer image for jupyter/all-spark-notebook:latest
Entered start.sh with args: jupyter lab
Running hooks in: /usr/local/bin/start-notebook.d as uid: 1000 gid: 100
Done running hooks in: /usr/local/bin/start-notebook.d
Running hooks in: /usr/local/bin/before-notebook.d as uid: 1000 gid: 100
Sourcing shell script: /usr/local/bin/before-notebook.d/spark-config.sh

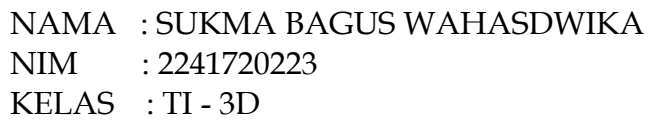
RAM 2.12 GB CPU 0.00% Disk: 9.89 GB used (limit 1006.85 GB)
```

Setelah itu, akses Jupyter Notebook di <http://localhost:8888>



Menghentikan container





```
docker stop spark-master spark-worker  
docker rm spark-master spark-worker
```

## Cara 1 : Menggunakan Spark Shell

- [illegible]

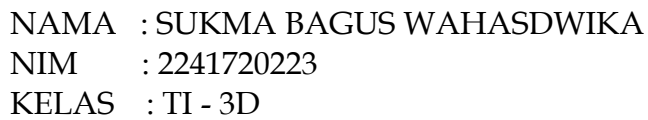
- ```
Terminal

scala> val textData = List("Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes Spark easy")
val rdd = sc.parallelize(textData)
val wordCounts = rdd.flatMap(line => line.split(" "))
  .map(word => (word, 1))
  .reduceByKey(_ + _)
wordCounts.collect().foreach(println)
textData: List[String] = List(Hello Spark, Hello Docker, Spark is awesome, Docker makes Spark easy)

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

scala> wordCounts: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at flatMap at <console>:23
Docker
Spark
is
awesome
Docker
makes
Spark
easy
```

```
scala> System.exit(0)
PS C:\Users\sukma bagus>
```



## Cara 2 : Menggunakan PySpark (Python)

- ```
PS C:\Users\sukma bagus> docker run -it --rm --name pyspark-shell --network spark-net --link spark-master:spark-master apache/
formation.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/23 09:45:49 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
where applicable
Welcome to

  _--_
 _/  \_  _--_/_/
 _\  /_  _\  /_  /_
/_/  /  _/_/_/_/  /_/_\  version 3.5.5
  /_

Using Python version 3.8.10 (default, Feb  4 2025 15:02:54)
Spark context Web UI available at http://2972282ec293:4040
Spark context available as 'sc' (master = spark://spark-master:7077, app id = app-20250423094550-0002).
SparkSession available as 'spark'.
>>>
```

- ```
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.appName("WordCount").getOrCreate()
25/04/23 09:59:19 WARN SparkSession: Using an existing Spark session; only runtime SQL configurations will take effect.
>>> data = ["Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes 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()
[Stage 0:>  [['Docker', 2], ('Spark'
, 3), ('awesome', 1), ('easy', 1), ('Hello', 2), ('is', 1), ('makes', 1)]]
>>>
```

### Cara 3 : Menggunakan Jupyter Notebook

- ```
PS C:\Users\sukma bagus> docker run -it -p 8888:8888 -p 4040
:4040 --network spark-net jupyter/all-spark-notebook
```



NAMA : SUKMA BAGUS WAHASDWIKA

NIM : 2241720223

KELAS : TI - 3D

## BIG DATA - 07 Spark Docker

Buka browser di <http://localhost:8888> , kemudian buat file dan tulis kode program

The screenshot shows a JupyterLab interface in a web browser at [127.0.0.1:8888/lab/tree/...](http://127.0.0.1:8888/lab/tree/...). The interface includes a file browser on the left, a code editor in the center, and a console/output area at the bottom. The code in the notebook is as follows:

```
[1]: from pyspark.sql import SparkSession

# Inisialisasi Spark
spark = SparkSession.builder \
    .appName("WordCount") \
    .getOrCreate()

# Baca file teks (jika ingin membaca dari file)
# text_file = spark.sparkContext.textFile("hdfs://.../input.txt")

# Untuk contoh, kita gunakan data dalam memori
data = ["Apache Spark is a unified analytics engine",
        "Spark can run on Hadoop, Apache Mesos, Kubernetes",
        "Spark is awesome for big data processing"]
rdd = spark.sparkContext.parallelize(data)

# Word Count
counts = rdd.flatMap(lambda x: x.split(' ')) \
    .map(lambda x: (x, 1)) \
    .reduceByKey(lambda a, b: a + b) \
    .sortBy(lambda x: x[1], ascending=False)

# Tampilkan 10 kata paling sering muncul
counts.take(10)
```

The output of the code is displayed in the console:

```
[1]: [('Spark', 3),
      ('Apache', 2),
      ('is', 2),
      ('a', 1),
      ('can', 1),
      ('Hadoop', 1),
      ('analytics', 1),
      ('awesome', 1),
      ('for', 1),
      ('run', 1)]
```

### Menjalankan Program sebagai Script

1. Buat file wordcount.py :

```
wordcount.py > ...
1  from pyspark.sql import SparkSession
2
3  if __name__ == "__main__":
4      spark = SparkSession.builder.appName("WordCount").getOrCreate()
5
6      # Untuk versi membaca file
7      # lines = spark.read.text("input.txt").rdd.map(lambda r: r[0])
8
9      # Untuk versi data contoh
10     data = ["Hello Spark", "Hello Docker", "Spark is awesome"]
11     lines = spark.sparkContext.parallelize(data)
12
13     counts = lines.flatMap(lambda x: x.split(' ')) \
14         .map(lambda x: (x, 1)) \
15         .reduceByKey(lambda a, b: a + b)
16
17     output = counts.collect()
18     for (word, count) in output:
19         print("%s: %i" % (word, count))
20
21     spark.stop()
```



NAMA : SUKMA BAGUS WAHASDWIKA  
NIM : 2241720223  
KELAS : TI - 3D

## BIG DATA - 07 Spark Docker

2. Jalankan script, jangan lupa juga mendefinisikan network spark-net

```
PS C:\Users\sukma bagus\script> cd ..
PS C:\Users\sukma bagus> cd app
PS C:\Users\sukma bagus\app> docker run --rm --network spark-net -v ${PWD}:/app --link spark-master:spark-master apache/spark:latest
/opt/spark/bin/spark-submit --master spark://spark-master:7077 /app/wordcount.py
25/04/28 09:44:16 INFO SparkContext: Running Spark version 3.5.5
25/04/28 09:44:16 INFO SparkContext: OS info Linux, 5.15.167.4-microsoft-standard-WSL2, amd64
25/04/28 09:44:16 INFO SparkContext: Java version 11.0.26
25/04/28 09:44:16 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
```

Output program:

```
25/04/28 09:44:23 INFO DAGScheduler: Job 0 finished: collect
at /app/wordcount.py:17, took 4.903890 s
Hello: 2
Spark: 2
is: 1
awesome: 1
Docker: 1
25/04/28 09:44:23 INFO SparkContext: SparkContext is stoppin
g with exitCode 0.
```