Interaksi dengan Spark di Lingkungan Windows Menggunakan Docker

Dalam praktikum ini kita akan menjalankan Apache Spark di Windows menggunakan Docker dan mencoba membuat job sederhana dengan berbagai macam alternatif cara.

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

Terminal

```
f937e0a2086c: Pull complete
0f3083818c14: Pull complete
d3c7b6bd77aa: Pull complete
4d9bb71a5e54: Pull complete
b072aa17899d: Pull complete
5762a181dda2: Pull complete
1ba3910f6ba2: Pull complete
4f4fb700ef54: Pull complete
391ef20df327: Pull complete
Digest: sha256:39321d67b23e2e0953f81b60778f74bf40c40a18dfb0e881e6a38593af60afa1
Status: Downloaded newer image for apache/spark:latest
docker.io/apache/spark:latest
PS C:\Users\Acer>
```

2. Menjalankan Spark Master

Sebelumnya buat docker network sebagai berikut

```
PS C:\Users\Acer> docker network create spark-net
00568fcfd04abf0f4a9cf5d98aa568e7230e2dac924bee0bd70257c10ca3b3db
```

Kemudian jalankan spark-master dalam network tersebut

```
PS C:\Users\Acer> 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
5bbd45e9dcf3af50dba8c63c111bb2b824f65a768fbb224a41bb795c51c28d70
```

Kita alokasikan resource untuk memastikan tidak kekurangan resource dalam menjalankan job.

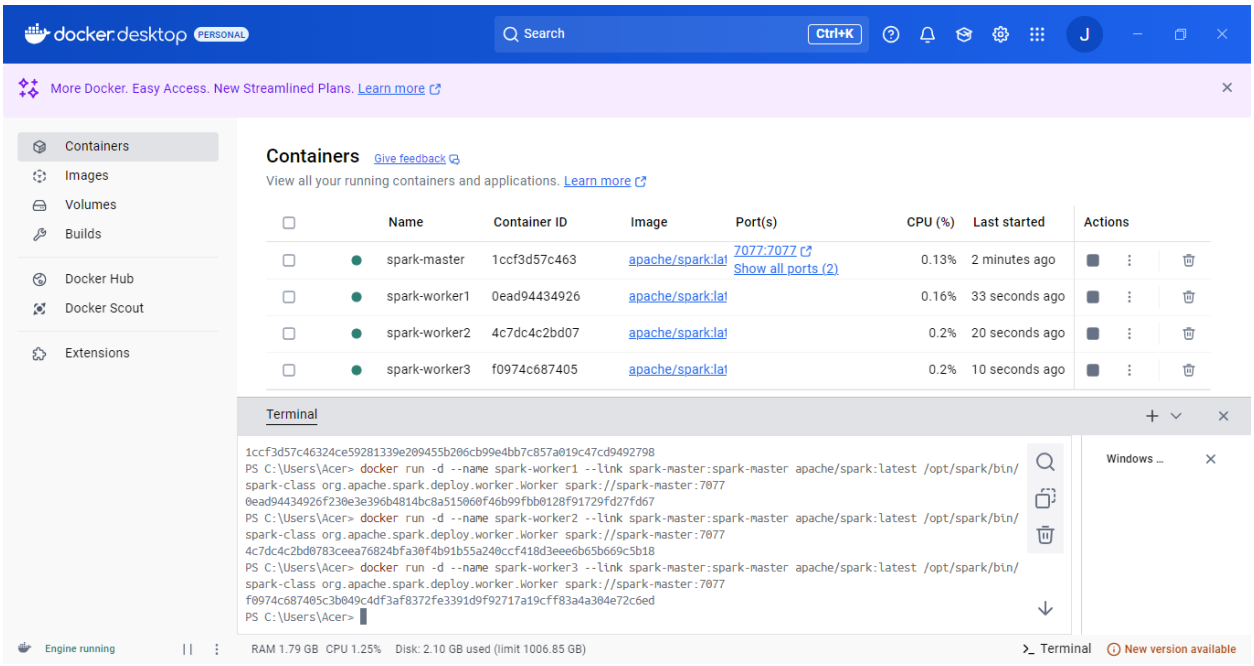
3. Menjalankan Spark Worker

```
PS C:\Users\Acer> docker run -d --name spark-worker --network spark-net -m 2g --cpus=2 apache/spark:latest /opt/spark/bin
/spark-class org.apache.spark.deploy.worker.Worker spark://spark-master:7077 --memory 1g --cores 1
58c18a90d6c65b112cc2e72ed7cad42c991bc8bb2710f459e2bce6b53168f9a8
```

Kita perlu alokasikan resource misalnya 2G memori dan 2 core CPU

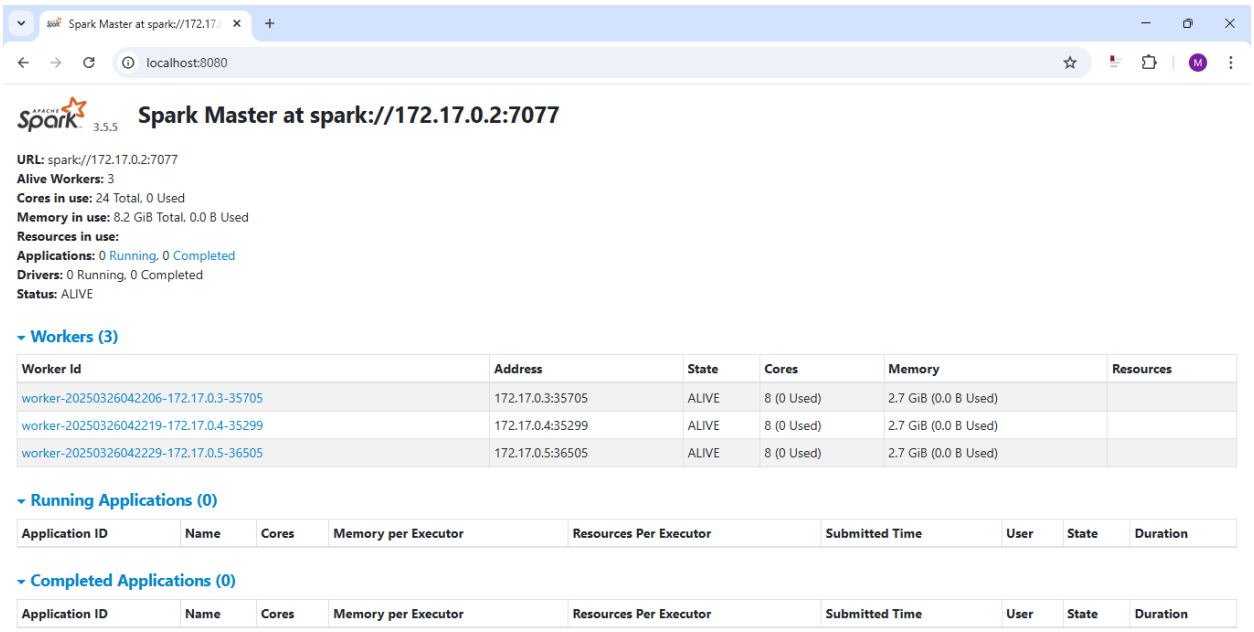
jalankan perintah di atas beberapa kali dengan nama yang berbeda untuk membuat beberapa worker. Misalnya spark-worker1, spark-worker2, dan seterusnya

Contoh menggunakan 3 worker.



4. Mengakses Spark Web UI

http://localhost:8080



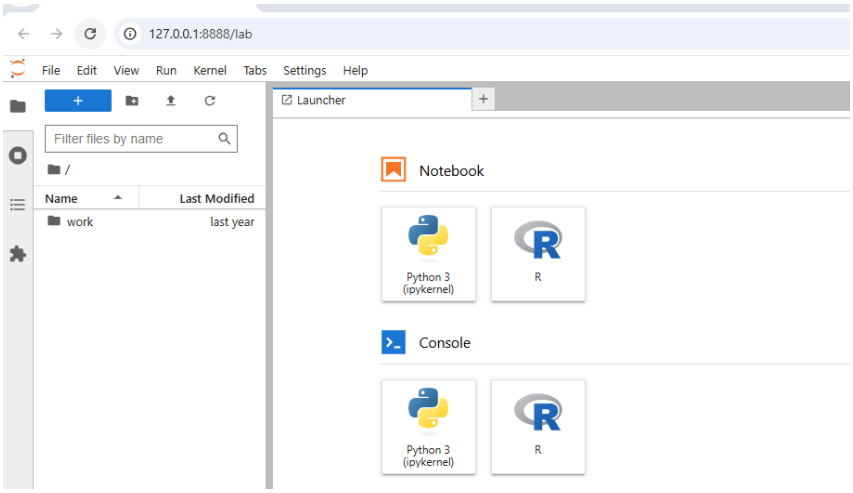
5. Menjalankan Spark Shell

```
PS C:\Users\Acer> docker run -it --rm --name spark-shell --network spark-net --link spark-master:spark-master apache/spar
k: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).
25/03/26 04:52:31 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cla
sses where applicable
Spark context Web UI available at http://03f37d151387:4040
Spark context available as 'sc' (master = spark://spark-master:7077, app id = app-20250326045233-0000).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
```

6. Menggunakan Jupyter Notebook dengan Spark

```
PS C:\Users\Acer> docker run -it -p 8888:8888 -p 4040:4040 --network spark-net jupyter/all-spark-notebook
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
Done running hooks in: /usr/local/bin/before-notebook.d
```

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



Untuk menghentikan container:

docker stop spark-master spark-worker

docker rm spark-master spark-worker

```
PS C:\Users\Acer> docker stop spark-master spark-worker1 spark-worker2
spark-master
spark-worker1
spark-worker2
PS C:\Users\Acer> docker rm spark-master spark-worker1 spark-worker2
spark-master
spark-worker1
spark-worker2
PS C:\Users\Acer>
```

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

1. Jalankan Spark Shell di Docker seperti contoh di atas
2. Ketikkan kode berikut di Spark Shell:

```
// Buat RDD dari koleksi teks
val textData = List("Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes Spark easy")
val rdd = sc.parallelize(textData)

// Lakukan word count
val wordCounts = rdd.flatMap(line => line.split(" "))  // Split menjadi kata-kata
                .map(word => (word, 1))          // Map setiap kata ke tuple (word, 1)
                .reduceByKey(_ + _)             // Reduce dengan menjumlahkan

// Tampilkan hasil
wordCounts.collect().foreach(println)
```

Terminal

```
scala> res1: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[3] at reduceByKey at <console>:24

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

Untuk keluar dari spark-shell gunakan:

System.exit(0)

Cara 2: Menggunakan PySpark (Python)

1. Jalankan PySpark Shell di Docker:

Dalam command juga terdapat definisi network juga.

```
PS C:\Users\Acer> docker run -it --rm --name pyspark-shell --network spark-net --link spark-master:spark-master apache/sp
ark:latest /opt/spark/bin/pyspark --master spark://spark-master:7077
Python 3.8.10 (default, Feb  4 2025, 15:02:54)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/03/26 05:09:36 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cla
sses where applicable
Welcome to
```

2. Ketikkan kode Python berikut:

```
from pyspark.sql import SparkSession

# Buat SparkSession
spark = SparkSession.builder.appName("WordCount").getOrCreate()

# Data contoh
data = ["Hello Spark", "Hello Docker", "Spark is awesome", "Docker makes Spark easy"]
rdd = spark.sparkContext.parallelize(data)

# Proses Word Count
word_counts = rdd.flatMap(lambda line: line.split(" ")) \
            .map(lambda word: (word, 1)) \
            .reduceByKey(lambda a, b: a + b)

# Tampilkan hasil
word_counts.collect()
```

```
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.appName("WordCount").getOrCreate()
25/03/26 05:11:39 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()
[('Hello', 2), ('Spark', 3), ('is', 1), ('awesome', 1), ('Docker', 2), ('makes', 1), ('easy', 1)]
>>>
```

Untuk keluar dari pyspark-shel menggunakan: exit()

Cara 3: Menggunakan Jupyter Notebook

Jika Anda menggunakan Jupyter Notebook (seperti di container jupyter/all-spark-notebook):

```
from pyspark.sql import SparkSession



# Inisialisasi Spark
```

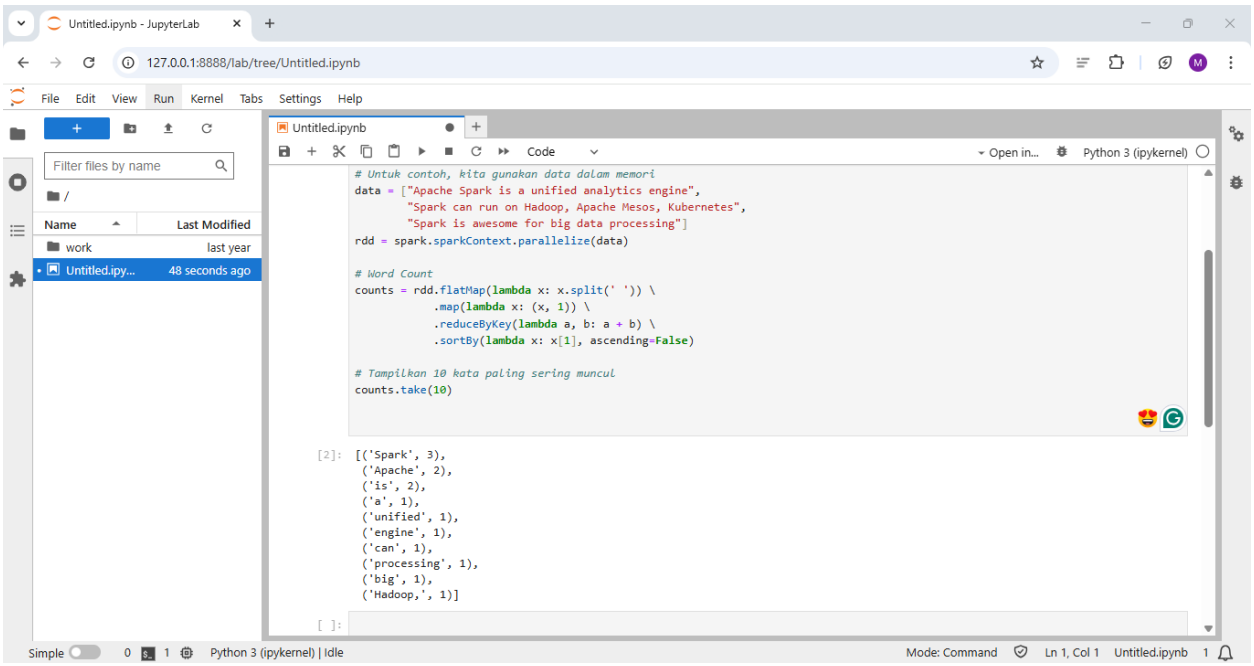
```
spark = SparkSession.builder \
    .appName("WordCount") \
    .getOrCreate()

# 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)
```



Menjalankan Program sebagai Script

1. Buat file wordcount.py dengan isi berikut:

```
from pyspark.sql import SparkSession

if __name__ == "__main__":
    spark = SparkSession.builder.appName("WordCount").getOrCreate()

    # Untuk versi membaca file
    # lines = spark.read.text("input.txt").rdd.map(lambda r: r[0])

    # Untuk versi data contoh
    data = ["Hello Spark", "Hello Docker", "Spark is awesome"]
    lines = spark.sparkContext.parallelize(data)

    counts = lines.flatMap(lambda x: x.split(' ')) \
                  .map(lambda x: (x, 1)) \
                  .reduceByKey(lambda a, b: a + b)

    output = counts.collect()
    for (word, count) in output:
        print("%s: %i" % (word, count))

    spark.stop()
```

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

```
PS C:\Users\Acer> 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
25/03/26 05:20:21 INFO SparkContext: Running Spark version 3.5.5
25/03/26 05:20:21 INFO SparkContext: OS info Linux, 5.15.167.4-microsoft-standard-WSL2, amd64
25/03/26 05:20:21 INFO SparkContext: Java version 11.0.26
```

Perhatikan dalam command tersebut mendifinisikan akses data ke local.

Program-program di atas akan menghasilkan output seperti:

Hello: 2

Spark: 3

Docker: 2

is: 1

awesome: 1

...

```
25/03/26 05:20:32 INFO DAGScheduler: Job 0 finished: collect at /app/wordcount.py:17, took 7.986877 s
Hello: 2
Spark: 2
is: 1
awesome: 1
Docker: 1
25/03/26 05:20:32 INFO SparkContext: SparkContext is stopping with exitCode 0.
```