# 2️⃣ Lab 2 – Spark RDD

## Objective

- Install Spark on a Docker cluster.
- Run Python Spark RDD programs on the cluster.
- Perform basic RDD operations like counting lines, computing averages, sorting by key, and aggregating by key.

---

## 1️⃣ Setup Docker Cluster

**Step 1: Remove old network (optional)**

```
docker network rm hadoop
```

> ⚠️ Only needed if the network already exists. You cannot remove it if containers are still using it.

**Step 2: Create new network**

```
docker network create hadoop
```

**Step 3: Run containers**

```
docker run -itd --net=hadoop -p 8080:8080 --expose 22 --name hadoop-master --hostname hadoop-master liliasfaxi/spark-hadoop:hv-2.7.2

docker run -itd --net=hadoop --expose 22 --name hadoop-slave1 --hostname hadoop-slave1 liliasfaxi/spark-hadoop:hv-2.7.2
```

```
docker run -itd --net=hadoop --expose 22 --name hadoop-slave2 --hostnam
e hadoop-slave2 liliasfaxi/spark-hadoop:hv-2.7.2
```

> Notes:
>
> - `-net=hadoop` connects all containers in the same network.
> - `p 8080:8080` exposes Spark Master UI.
> - `-expose 22` is optional if you want SSH access.

**Step 4: Enter master container**

```
docker exec -it hadoop-master bash
```

---

# 2️⃣ Configure Spark

**Step 1: Create `slaves` file**

```
touch /usr/local/spark/conf/slaves
```

Add the names of slave nodes:

```
echo "hadoop-slave1" >> /usr/local/spark/conf/slaves
echo "hadoop-slave2" >> /usr/local/spark/conf/slaves
```

**Step 2: Configure Python for PySpark (optional)**

```
which python3
cd /usr/local/spark/conf
cp spark-env.sh.template spark-env.sh
```

Add this line to `spark-env.sh`:

```
echo "export PYSPARK_PYTHON=/usr/bin/python3" >> spark-env.sh
```

> Note: Some Docker images already have Python configured, so this may not be necessary.

---

## 3️⃣ Start Spark Services

**Step 1: Start master**

```
cd /usr/local/spark/sbin
./start-master.sh
```

**Step 2: Start slaves**

```
./start-slave.sh spark://hadoop-master:7077
```

> Notes:
>
> - Master URL: `spark://hadoop-master:7077`
> - Slaves register with the master automatically.

---

## 4️⃣ Copy Data and Python Scripts

**Step 1: Create folder inside master container**

```
mkdir -p /root/local-vr
```

**Step 2: Copy `arbres.csv`**

```
docker cp arbres.csv hadoop-master:/root/arbres.csv
docker cp arbres.csv hadoop-slave1:/root/arbres.csv
docker cp arbres.csv hadoop-slave2:/root/arbres.csv
```

### Step 3: Copy Python programs

```
docker cp p1.py hadoop-master:/root/local-vr/p1.py
docker cp p2.py hadoop-master:/root/local-vr/p2.py
docker cp p3.py hadoop-master:/root/local-vr/p3.py
docker cp p4.py hadoop-master:/root/local-vr/p4.py
```

> Note: Always make sure the target folder exists inside the container (mkdir -p) before using docker cp.

---

## 5️⃣ Run Spark RDD Programs

### Step 1: Enter folder with scripts

```
cd /root/local-vr
```

### Step 2: Submit Spark jobs

```
spark-submit --master spark://hadoop-master:7077 p1.py
spark-submit --master spark://hadoop-master:7077 p2.py
spark-submit --master spark://hadoop-master:7077 p3.py
spark-submit --master spark://hadoop-master:7077 p4.py
```

> What each program does:
>
> - `p1.py`: Count total lines in `arbres.csv`
> - `p2.py`: Compute average height of trees
> - `p3.py`: Find the genre of the tallest tree (sortByKey)
> - `p4.py`: Count the number of trees per genre (reduceByKey)

---

## 6️⃣ Notes / Observations

- Spark distributes computations across master and slave nodes (cluster computing).
- PySpark uses RDDs (Resilient Distributed Datasets) for processing data in parallel.
- Logs show stages, job progress, executor messages, and output size.
- Temporary files are automatically deleted after job completion.

> Important: Make sure you have Python installed in the container (which python3).
>
> All data and scripts must be inside the container; Docker cannot access local files directly during execution.

---

## 7️⃣ Restart / Re-do Lab (Start Over)

If you want to **reset the lab environment**:

1. **Stop containers**

```
docker stop hadoop-master hadoop-slave1 hadoop-slave2
```

1. **Remove containers**

```
docker rm hadoop-master hadoop-slave1 hadoop-slave2
```

1. **Remove network**

```
docker network rm hadoop
```

1. **Recreate network and containers** (repeat steps from section 1).

2. **Copy data/scripts again** (repeat section 4).

> ✅ This ensures you have a clean cluster without conflicts from previous runs.