Successful Execution of a PySpark Job Inside a Docker Container Using docker-compose

# Overview

This document describes the successful execution of a PySpark job inside a Docker container using docker-compose. The process includes container creation, Spark initialization, job execution, and clean shutdown.

# Step-by-Step Breakdown

## 1. Container Initialization

The Docker container named `pyspark-app` was successfully created using the `docker-compose up` command:  
✔ Container pyspark-app Created

## 2. Bitnami Spark Container Logs

The Bitnami Spark container initializes and displays the environment setup:  
- Spark version 3.5.5  
- OS: Linux with WSL2  
- Java version: 17.0.15

## 3. Spark Context and Resources Setup

Spark initializes internal components such as SparkContext, BlockManager, and the Spark Web UI, which is available at port 4040 inside the container.

## 4. Job Execution

A PySpark job runs that displays a DataFrame with three rows. Sample output:  
+-------+---+  
| name|age|  
+-------+---+  
| Alice| 30|  
| Bob| 25|  
|Charlie| 35|  
+-------+---+

## 5. Clean Shutdown

Spark shuts down all services and components, and the container exits cleanly with exit code 0.

# Conclusion

The PySpark job successfully ran inside a Docker container using docker-compose. Spark components were initialized, a simple DataFrame job was executed, and the application exited without errors.