04 Spark Submit and Deploy Modes

Deploy Modes

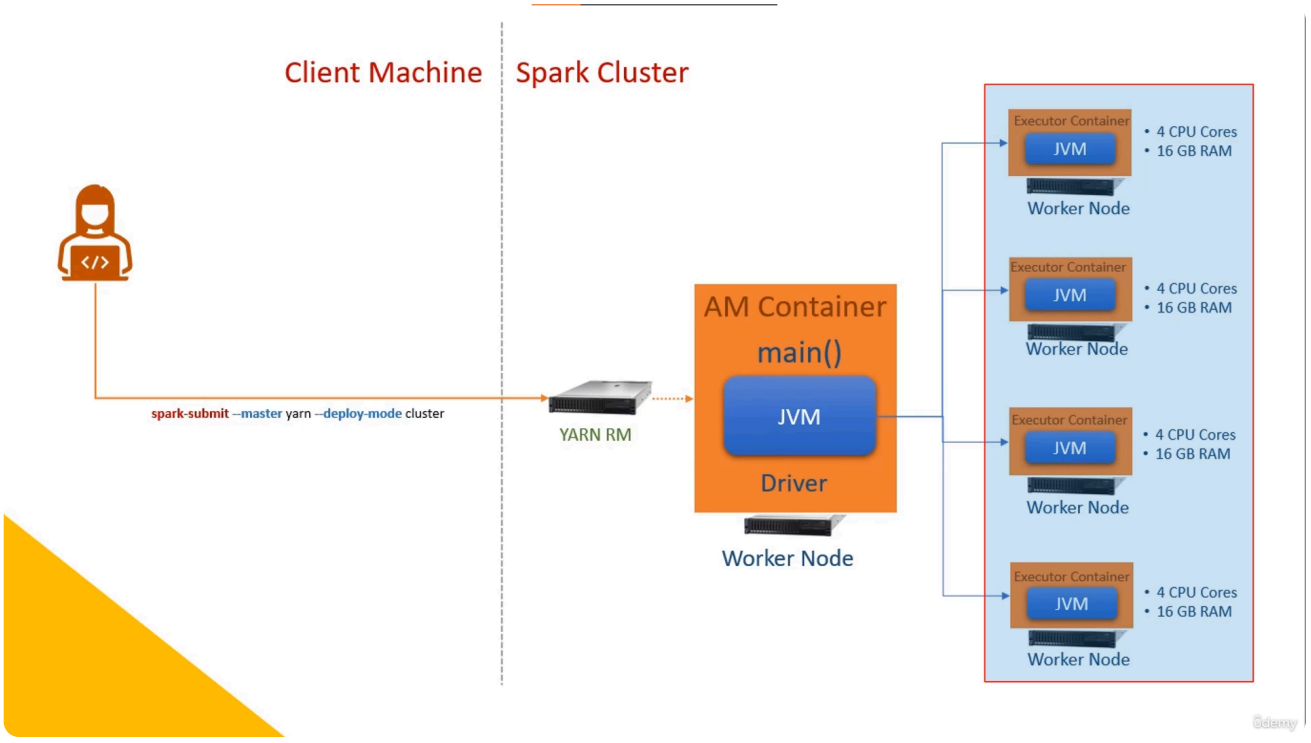
Cluster Mode VS Client Mode

Cluster Mode

1. **Cluster Mode:** Driver program runs **on one of the nodes in the cluster**.

   Benefits:

   - **Scalability:** Can leverage the processing power of multiple machines in the cluster for large datasets.
   - **High Availability:** If a node fails, the driver can be restarted on another node with minimal disruption.
   - **Security:** Driver runs within the secure environment of the cluster.
   - No network latency
   - **When to use:**
     - Production workloads requiring large-scale data processing.
     - Applications demanding high availability and fault tolerance.



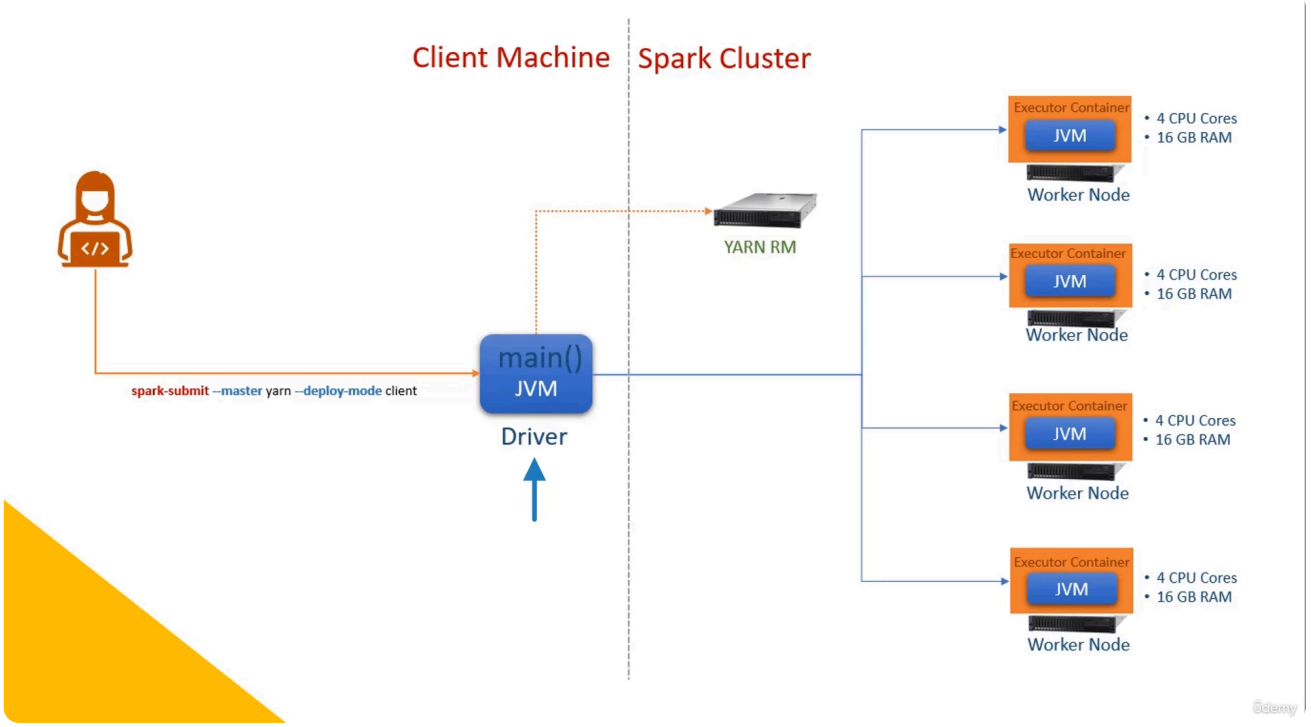
Client Mode

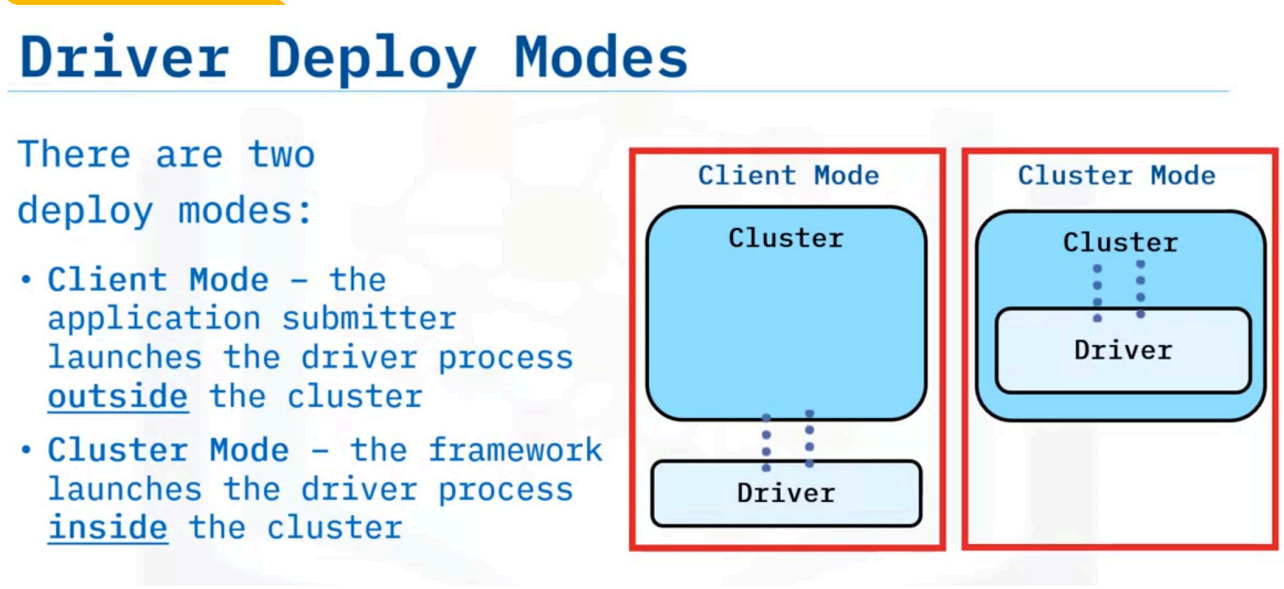
Client Mode: Driver program runs **in the local machine's JVM (Java Virtual Machine)**.

Benefits:

- **Simplicity:** Easier to set up and use, especially for development and testing.
- **Faster startup:** Driver doesn't need to be deployed on the cluster, reducing initial overhead.
- **Development interaction:** Easier to monitor and interact with the driver program during development.

- **When to use:**
  - Development and testing of Spark applications.
  - Smaller datasets that can be processed efficiently on a single machine.
  - Interactive Spark sessions where immediate feedback is necessary.
  - PySpark, Spark-Shell, spark-sql





Local vs. cluster modes

https://spark.apache.org/docs/latest/rdd-programming-guide.html#local-vs-cluster-modes

```
counter = 0
rdd = sc.parallelize(data)

# Wrong: Don't do this!!
def increment_counter(x):
    global counter
    counter += x
```

```
rdd.foreach(increment_counter)

print("Counter value: ", counter)
```

The behavior of the above code is undefined, and may not work as intended. To execute jobs, Spark breaks up the processing of RDD operations into tasks, each of which is executed by an executor. Prior to execution, Spark computes the task's **closure**. The closure is those variables and methods which must be visible for the executor to perform its computations on the RDD (in this case `foreach()`). This closure is serialized and sent to each executor.

The variables within the closure sent to each executor are now copies and thus, when **counter** is referenced within the `foreach` function, it's no longer the **counter** on the driver node. There is still a **counter** in the memory of the driver node but this is no longer visible to the executors! The executors only see the copy from the serialized closure. Thus, the final value of **counter** will still be zero since all operations on **counter** were referencing the value within the serialized closure.

In local mode, in some circumstances, the `foreach` function will actually execute within the same JVM as the driver and will reference the same original **counter**, and may actually update it.

To ensure well-defined behavior in these sorts of scenarios one should use an `Accumulator`. Accumulators in Spark are used specifically to provide a mechanism for safely updating a variable when execution is split up across worker nodes in a cluster. The Accumulators section of this guide discusses these in more detail.

In general, closures - constructs like loops or locally defined methods, should not be used to mutate some global state. Spark does not define or guarantee the behavior of mutations to objects referenced from outside of closures. Some code that does this may work in local mode, but that's just by accident and such code will not behave as expected in distributed mode. Use an Accumulator instead if some global aggregation is needed.

---

spark-submit

`spark-submit` is designed to run the application, complete its tasks, and then shut down.

A command line tool that allows you to submit the Spark application to the cluster

```
@NikhilSharma ➜ C:\spark (base 3.12.7)
[70%] cd $env:SPARK_HOME\bin
@NikhilSharma ➜ C:\..\bin (base 3.12.7)
[70%] ls

    Directory: C:\spark\spark-3.3.1-bin-hadoop2\bin


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        10/15/2022   3:11 PM           1089 beeline
-a----        10/15/2022   3:11 PM           1064 beeline.cmd
-a----        10/15/2022   3:11 PM          11283 docker-image-tool.sh
-a----        10/15/2022   3:11 PM           1935 find-spark-home
-a----        10/15/2022   3:11 PM           2685 find-spark-home.cmd
-a----        10/15/2022   3:11 PM           2337 load-spark-env.cmd
-a----        10/15/2022   3:11 PM           2678 load-spark-env.sh
-a----        10/15/2022   3:11 PM           2636 pyspark
-a----        10/15/2022   3:11 PM           1170 pyspark.cmd
-a----        10/15/2022   3:11 PM           1542 pyspark2.cmd
-a----        10/15/2022   3:11 PM           1030 run-example
-a----        10/15/2022   3:11 PM           1223 run-example.cmd
-a----        10/15/2022   3:11 PM           3539 spark-class
-a----        10/15/2022   3:11 PM           1180 spark-class.cmd
-a----        10/15/2022   3:11 PM           2812 spark-class2.cmd
-a----        10/15/2022   3:11 PM           3122 spark-shell
-a----        10/15/2022   3:11 PM           1178 spark-shell.cmd
-a----        10/15/2022   3:11 PM           1818 spark-shell2.cmd
-a----        10/15/2022   3:11 PM           1065 spark-sql
-a----        10/15/2022   3:11 PM           1173 spark-sql.cmd
-a----        10/15/2022   3:11 PM           1118 spark-sql2.cmd
-a----        10/15/2022   3:11 PM           1040 spark-submit
-a----        10/15/2022   3:11 PM           1180 spark-submit.cmd
-a----        10/15/2022   3:11 PM           1155 spark-submit2.cmd
-a----        10/15/2022   3:11 PM           1039 sparkR
-a----        10/15/2022   3:11 PM           1168 sparkR.cmd
-a----        10/15/2022   3:11 PM           1097 sparkR2.cmd
```

```
PS C:\spark\bin> spark-submit --help
Usage: spark-submit [options] <app jar | python file | R file> [app arguments]
Usage: spark-submit --kill [submission ID] --master [spark://...]
Usage: spark-submit --status [submission ID] --master [spark://...]
Usage: spark-submit run-example [options] example-class [example args]

Options:
  --master MASTER_URL         spark://host:port, mesos://host:port, yarn,
                              k8s://https://host:port, or local (Default: local[*]).
  --deploy-mode DEPLOY_MODE   Whether to launch the driver program locally
("client") or
                              on one of the worker machines inside the cluster
("cluster")
                              (Default: client).
  --class CLASS_NAME          Your application's main class (for Java / Scala apps).
  --name NAME                 A name of your application.
  --jars JARS                 Comma-separated list of jars to include on the driver
                              and executor classpaths.
```

```
spark-submit --master yarn --deploy-mode cluster --driver-cores 2 --driver-memory 8G
--num-executors 4 --executor-cores 4 --executor-memory 16G example_spark.py
```

```
spark-submit --master local[*] --driver-memory 8G --executor-memory 8G --executor-
cores 4 --total-executor-cores 4 ratings-counter.py
```

Driver container will be having 2 Cores and 8GB RAM and also we request YARM RM to provide 4 executors each with 4Cores and 16GB RAM.

> https://sparkbyexamples.com/spark/spark-submit-command/