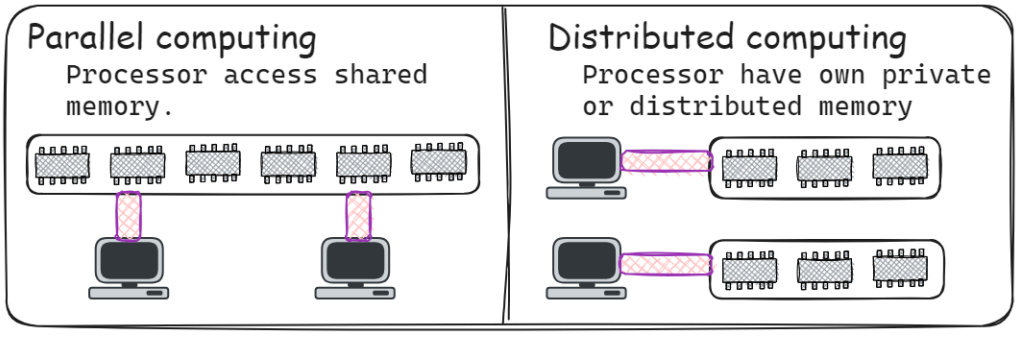
02 Spark Runtime Architecture

Spark

- Spark is a distributed computing platform
- Any Spark application is a distributed application which runs on a cluster
- Cluster technologies for Apache Spark
- Hadoop YARN
- Kubernetes
- Apache Mesos
- Spark Standalone



What is a cluster?

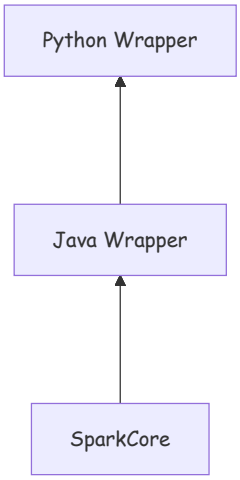
A pool of computers working together but viewed as a single system

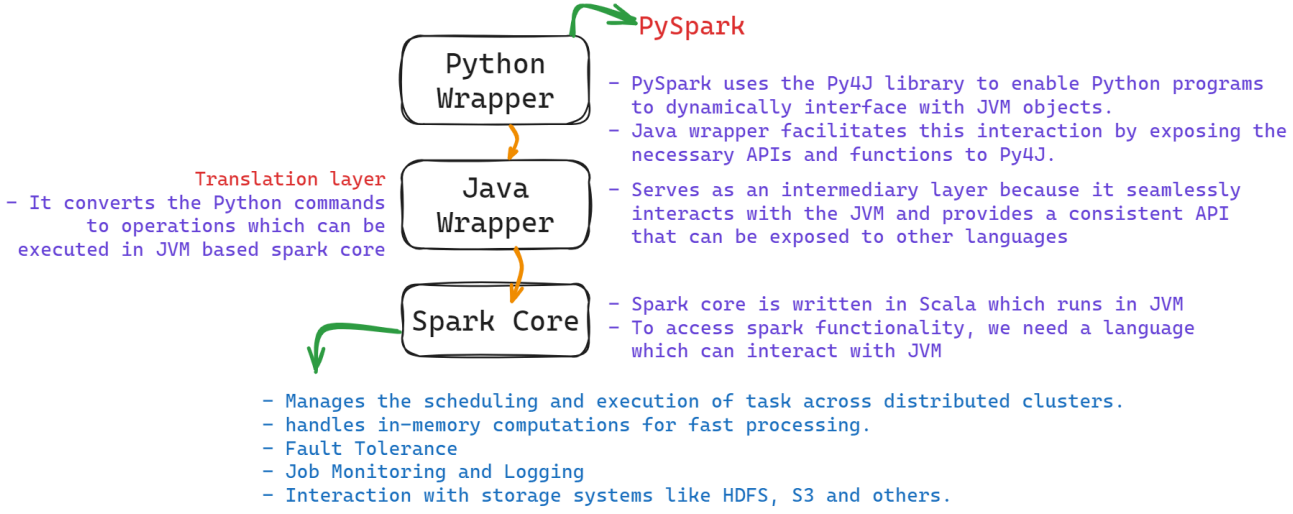
- Example cluster configuration
- Worker Node Capacity
- 16 CPU cores
- 64 GB RAM
- Cluster Capacity
- 160 CPU cores
- 640 GB RAM

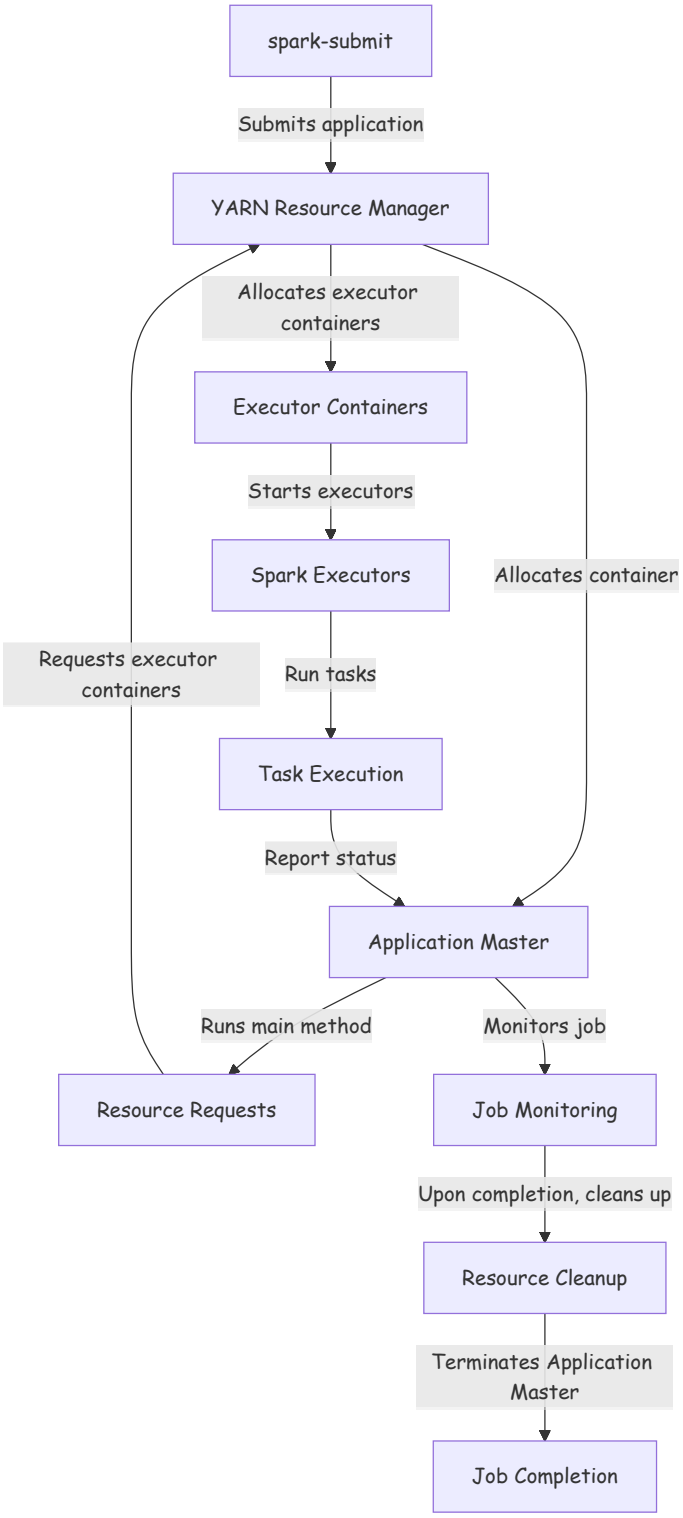
Spark Runtime Architecture

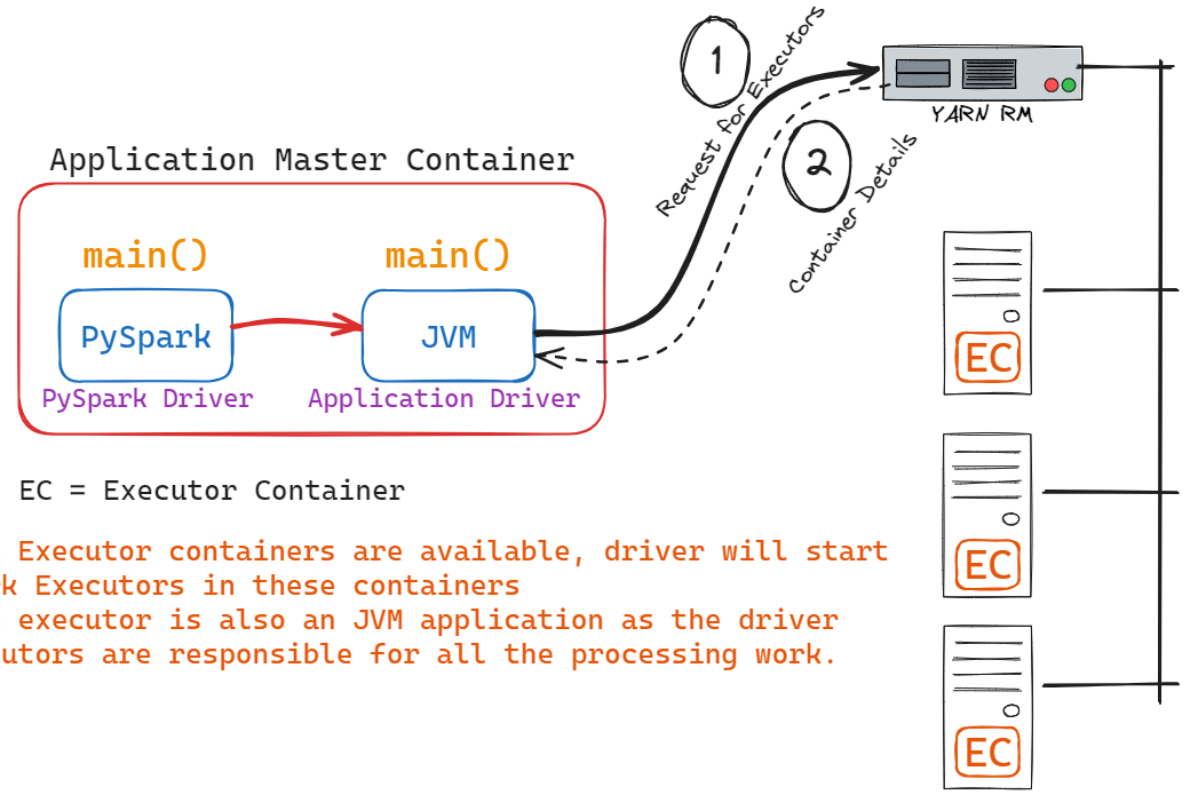
User submits the Spark Code by `spark-submit`, request goes to YARN RM.
YARN RM creates a Application Master container on a worker node and runs the main method of the application

- Container is an isolated virtual runtime environment with its own CPU RAM
  Container runs the main() method of the application. Two possibilities.
- PySpark
- Scala
  Spark is written in Scala, it runs in JVM



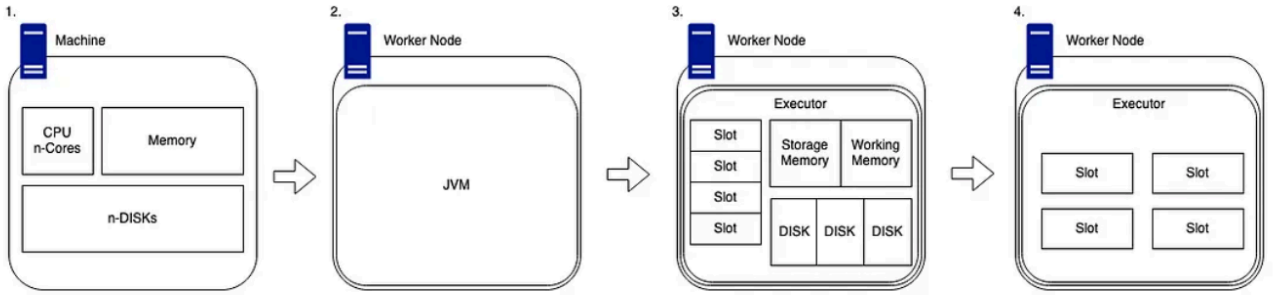



spark-submit
Submits application
YARN Resource Manager
Allocates executor containers
Executor Containers
Starts executors
Spark Executors
Allocates container
Requests executor containers
Run tasks
Task Execution
Report status
Application Master
Runs main method
Monitors job
Resource Requests
Job Monitoring
Upon completion, cleans up
Resource Cleanup
Terminates Application Master
Job Completion

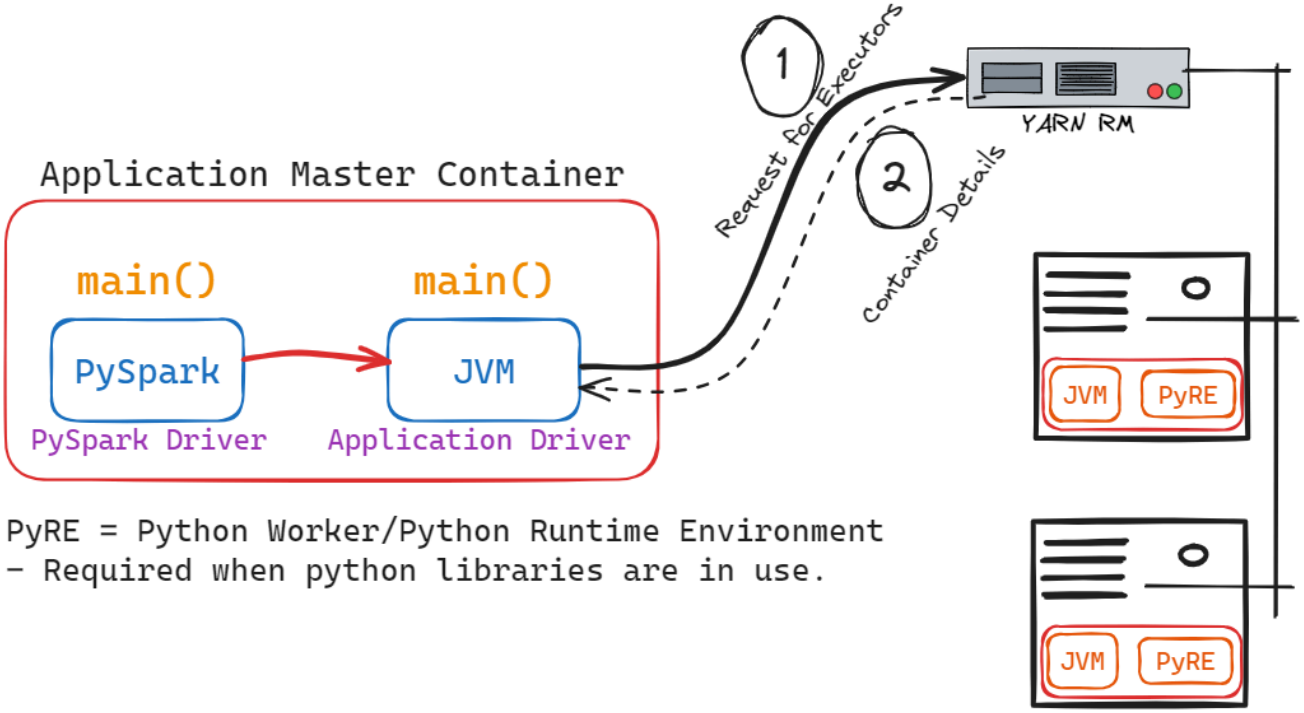


EC = Executor Container

- When Executor containers are available, driver will start Spark Executors in these containers
- Each executor is also an JVM application as the driver
- Executors are responsible for all the processing work.



Worker Node Step by Step (created by Luke Thorp)

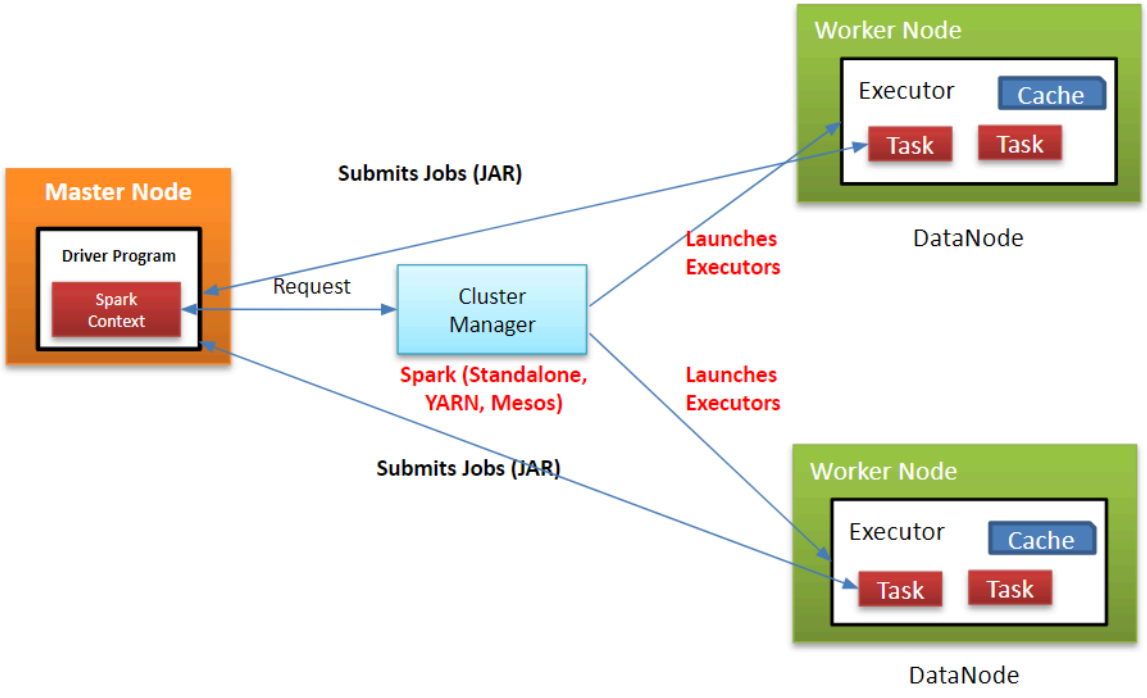


PyRE = Python Worker/Python Runtime Environment
- Required when python libraries are in use.

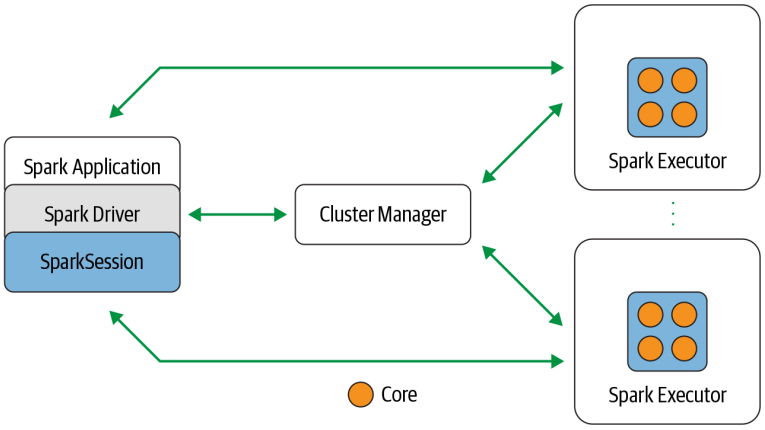
Spark Application Submission with YARN

1. **Submitting the Application:**
   - When you run `spark-submit`, it packages your application and configuration into a request and submits it to the YARN Resource Manager (RM).
2. **Resource Allocation:**
   - The YARN RM receives the application submission request and allocates a container for the Application Master (AM).
3. **Application Master Initialization:**
   - The Application Master is launched in the allocated container. For a Spark application, the AM is specifically the Spark Application Master. It initializes by running the `main()` method of your Spark application.
4. **Application Master Responsibilities:**
   - **Resource Requests:** The Application Master requests resources (additional containers) from the YARN RM to execute various parts of your Spark job.
   - **Task Scheduling:** It schedules tasks and assigns them to the allocated containers, distributing the workload across the cluster.
   - **Monitoring:** It monitors the progress of the application, handling task failures and retries, and aggregating logs and metrics.
5. **Executor Containers:**
   - The YARN RM allocates additional containers as per the request from the Application Master. These containers are used to launch Spark Executors, which are responsible for running individual tasks of your Spark application.
   - The Executors start executing tasks (such as transformations and actions specified in your Spark job) and report their status back to the Application Master.

Spark Architecture



[



Spark Architecture Terminologies

Spark Driver

Spark driver program is a main entry point which runs the main() method and is the place where the SparkContext is created.

- responsible for instantiating a `SparkSession`, the Spark driver has multiple roles:
- it communicates with the cluster manager;
- it requests resources (CPU, memory, etc.) from the cluster manager for Spark's executors (JVMs);
- and it transforms all the Spark operations into DAG computations, schedules them, and distributes their execution as tasks across the Spark executors. Once the resources are allocated, it communicates directly with the executors.

SparkContext :

Spark context is a gateway to all the Spark functionalities. It is similar to your database connection. Any command you execute in your database goes through the database connection.

SparkSession

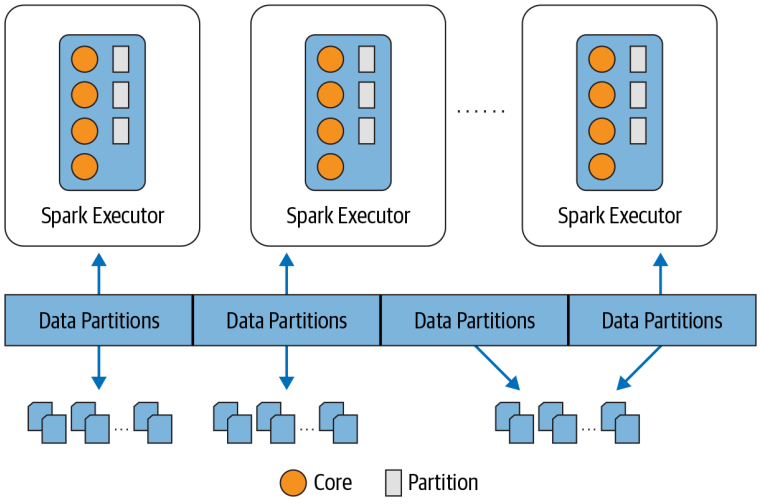
In Spark 2.0, the `SparkSession` became a unified conduit to all Spark operations and data. Not only did it subsume previous entry points to Spark like the `SparkContext`, `SQLContext`, `HiveContext`, `SparkConf`, and `StreamingContext`, but it also made working with Spark simpler and easier.

Cluster Manager : An external service to manage resource on the cluster

Worker Node : Node that run the application program in the cluster

Executors : Executors are worker nodes' processes in charge of running individual tasks in a given Spark job.

Task : A task is a unit of work that is sent to the executor. Each stage has some task, one task per partition. The Same task is done over different partitions of RDD.



Step by Step working of Spark

- **Step 01** : A code is submitted by the user which contains Transformations and Actions.
- **Step 02** : The driver converts the piece of code to DAG(Directed Acyclic Graph) which also optimizes i.e, pipelining transformations.

- **Step 03** : Driver node then converts DAG into a Physical Execution Plan which consists a set of Stages.
- **Step 04** : Driver Node then converts Physical Execution units (TASKS). A set of tasks is created under each Stages.
- **Step 05** : Now the Driver node talks to the Cluster Manager and asks for resources.
- **Step 06** : Driver Program initiates the spark context.
- **Step 07** : Cluster Manager in turns launches Executors on worker nodes on behalf of the driver.
- **Step 08** : Driver node will send the task to the Executors based on data placement.
- **Step 09** : When the executor starts they will register themselves to the drivers. So drivers will have complete view of the executors.
- **Step 10** : Executors starts executing the tasks assigned by the driver program. At any time, the driver program monitors the executors.
- **Step 11** : When sc.stop() is called it also terminates the executors and release resources from Cluster manager.