# Executors in Spark

so in spark executor are nothing but the processes running   means executors are present in worker nodes ?

Yes, you're absolutely right — in **Apache Spark**, executors are **processes** that **run on the worker nodes**.

Here's a clear breakdown:

---

## ✅ What are Executors?

- **Executors are JVM processes** launched by the Spark driver on **worker nodes**.
- They are **responsible for executing tasks** (from the DAG) and **storing data in memory or disk (caching, shuffling, etc.)**.
- Each executor runs **multiple tasks in parallel** using multiple threads (depending on the number of cores assigned).

---

## 🖥️ Where are Executors Present?

- Executors are **always present on worker nodes, not on the driver node**.
- In **standalone, YARN, or Kubernetes** clusters:
  - **Driver**: runs on one node (can be master or any worker, depending on mode).
  - **Executors**: are distributed across **worker nodes**.

---

## 🔁 Executor Lifecycle

1. **Spark Driver** (running on some node) connects to the **Cluster Manager** (like YARN/K8s/Standalone).
2. Cluster manager allocates **worker nodes** to launch **executors**.
3. Executors start as **JVM processes** and stay alive throughout the Spark job (unless dynamic allocation is enabled).
4. Once a job ends, executors shut down.

---

## 📦 Example Visualization

| Cluster Node | Role | What's Running |
|---|---|---|
| Node A | Driver | Spark Driver process only |
| Node B | Worker Node | Executor 1 (with N cores, M memory) |

| Cluster Node | Role | What's Running |
|---|---|---|
| Node C | Worker Node | Executor 2 (with N cores, M memory) |
| Node D | Worker Node | Executor 3 (with N cores, M memory) |

---

## ⚠️ Important Notes

- Executors **do not run on the driver node** (unless it's a local mode job).
- Each executor **reports back** to the driver with task status, metrics, etc.
- Executors are **short-lived** — they last only for the duration of the Spark application unless reused (like in Spark Streaming with dynamic allocation off).

---

Let me know if you'd like to see this with a visual diagram or a specific example with code or `spark-submit` config.

ok means executors are the one who does the job process:
so we can also cal excecutors as workers or labours who do the job?

Yes! That’s a **very good analogy** 🔧💼

You can absolutely think of:

> **Executors = Workers / Labourers who do the actual job**

Here’s how it fits perfectly:

---

## 🧠 Simple Analogy: Spark as a Construction Site

| Spark Component | Real-World Equivalent | Role |
|---|---|---|
| **Driver** | Site Supervisor / Manager | Plans the work, assigns tasks, coordinates everything |
| **Executors** | Labourers / Workers | Do the actual work: lift bricks, pour cement, build walls (i.e., run tasks) |
| **Cluster Manager** | Contractor / HR Manager | Hires workers (allocates machines), decides who goes where |
| **Worker Node** | Work Site (Machine) | A place where workers (executors) do the job |

---

## 🔍 Technical Mapping

- **Driver**:
    - Converts your Spark code into a **DAG of stages and tasks**
    - Sends these tasks to **executors** on worker nodes

  - Collects results and handles coordination
- **Executors**:
  - Execute **tasks** on **partitions of your data**
  - Perform transformations (e.g., `map`, `filter`, `join`)
  - Store data (cache/persist), shuffle data, write outputs

---

## ✅ So yes, Spark Executors:

- Are like **workers/labourers** doing the hands-on job
- Run on **worker nodes**
- Managed by the **driver (manager)**

Would you like me to give you a **diagram or simple code** to visualize this with actual cluster behavior?