Running on a Cluster

Spark Runtime Architecture

Spark Runtime Architecture

Machine or Node 1

Machine or Node 2

Machine or Node 3

Spark Runtime Architecture

Machine or Node 1

Worker Node

Machine or Node 2

Worker Node

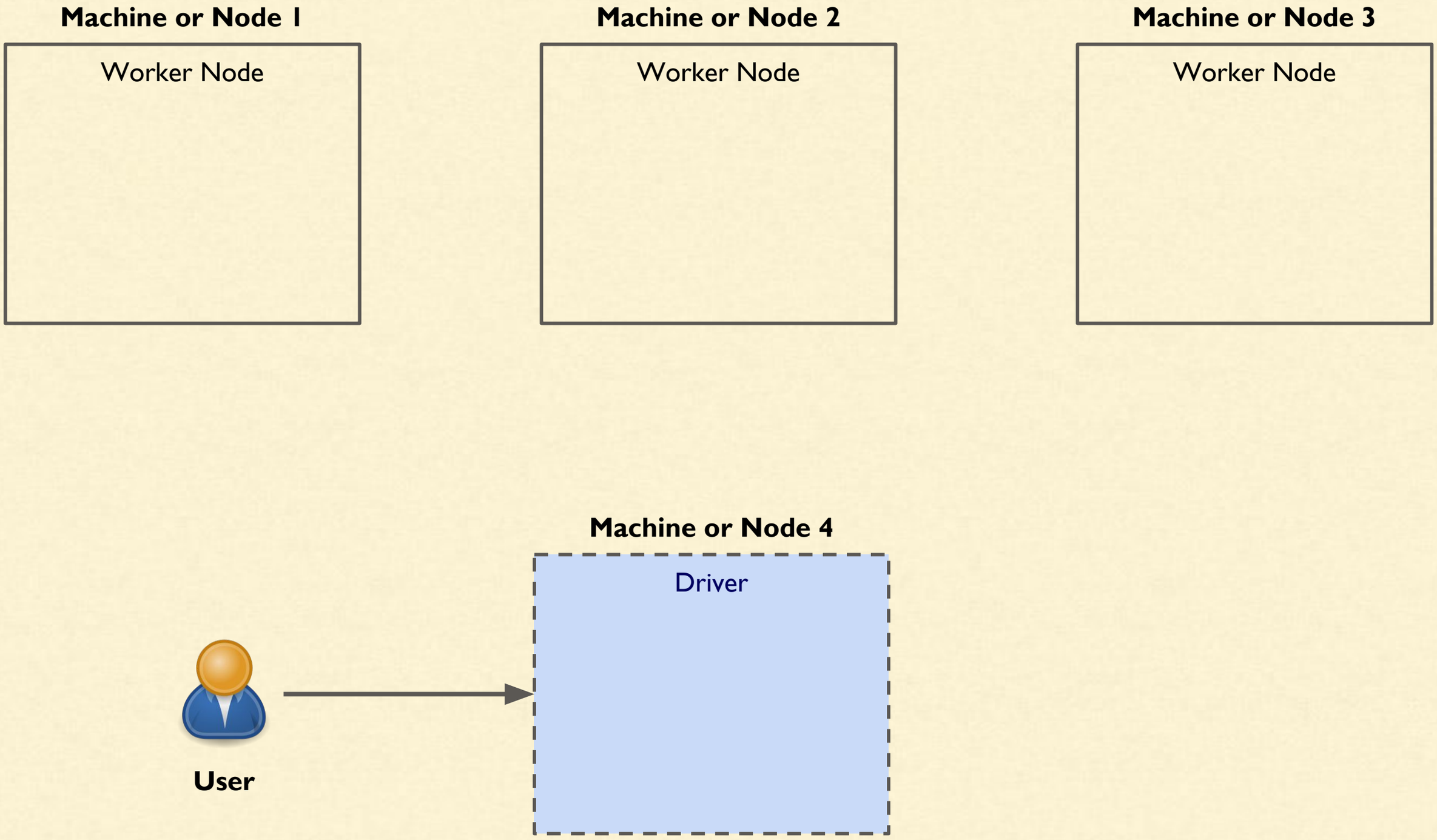
Machine or Node 3

Worker Node

Machine or Node 4

Driver

Spark Runtime Architecture



The Driver

- Process where main() method runs
- When you launch a Spark shell, you've created a driver program
- Once the driver terminates, the application is finished.

The Driver

- Process where main() method runs
- When you launch a Spark shell, you've created a driver program
- Once the driver terminates, the application is finished.
- While running it performs following:
  - Converting a user program into tasks
    - Convert a user program into tasks - units of execution.
    - Converts DAG (logical graph) into a physical execution plan

The Driver

- Process where main() method runs
- When you launch a Spark shell, you've created a driver program
- Once the driver terminates, the application is finished.
- While running it performs following:
  - Converting a user program into tasks
    - Convert a user program into tasks - units of execution.
    - Converts DAG (logical graph) into a physical execution plan
  - Scheduling tasks on executors

Driver: Scheduling tasks on executors

Driver: Scheduling tasks on executors

- Coordinate the scheduling of individual tasks on executors

Driver: Scheduling tasks on executors

- Coordinate the scheduling of individual tasks on executors
- Schedule tasks based on data placement

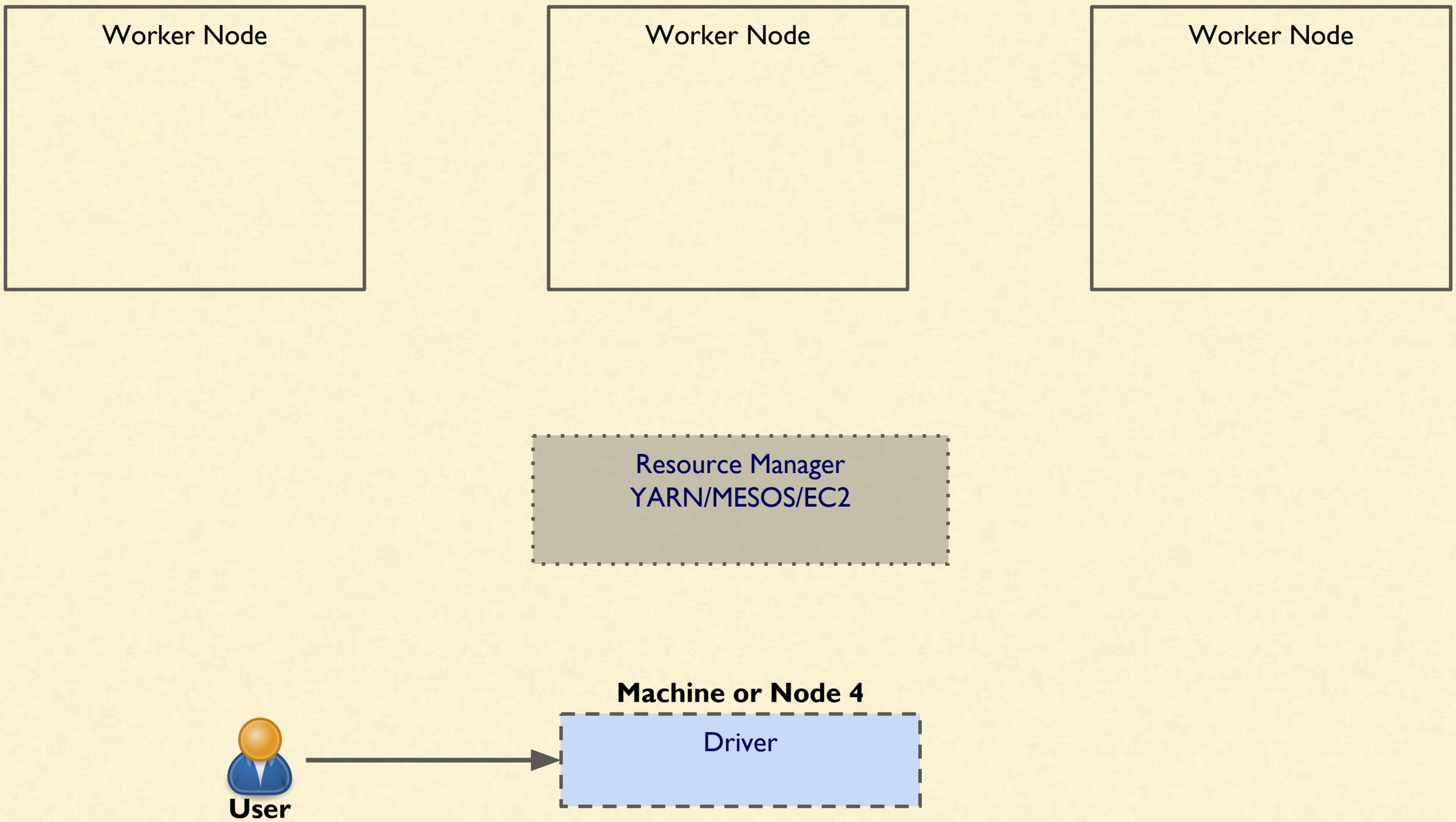
Driver: Scheduling tasks on executors

- Coordinate the scheduling of individual tasks on executors
- Schedule tasks based on data placement
- Tracks cached data and uses it to schedule future tasks

Driver: Scheduling tasks on executors

- Coordinate the scheduling of individual tasks on executors
- Schedule tasks based on data placement
- Tracks cached data and uses it to schedule future tasks
- Runs Spark web interface at port 4040.

Understanding The Architecture



Understanding The Architecture

Machine or Node 1

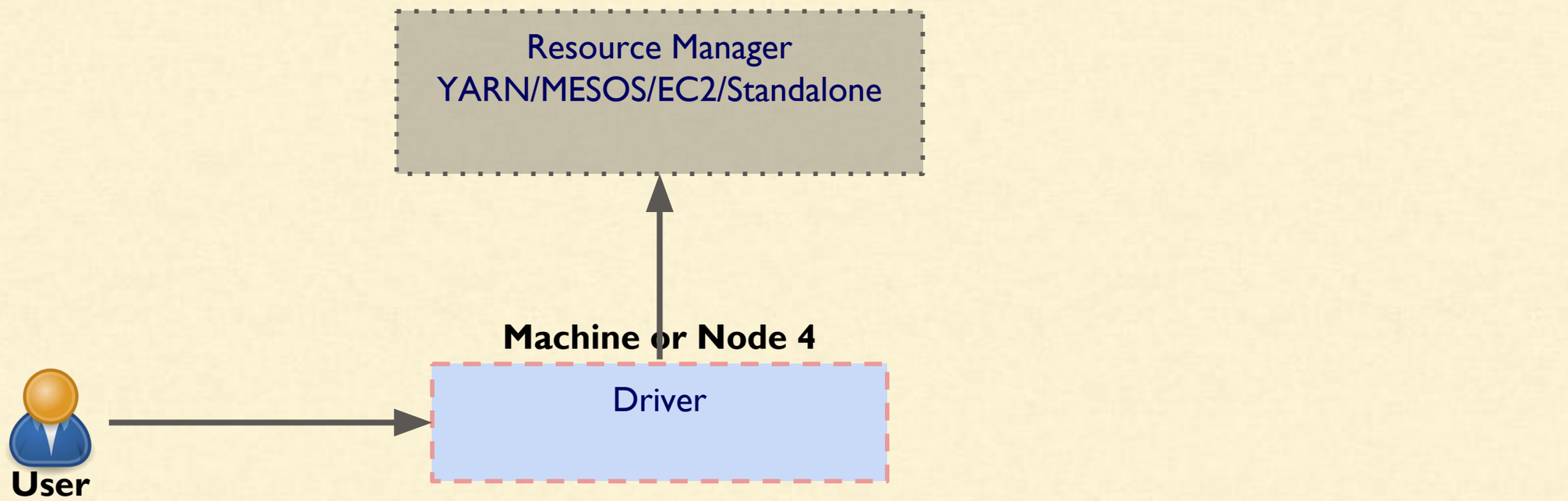
Worker Node

Machine or Node 2

Worker Node

Machine or Node 3

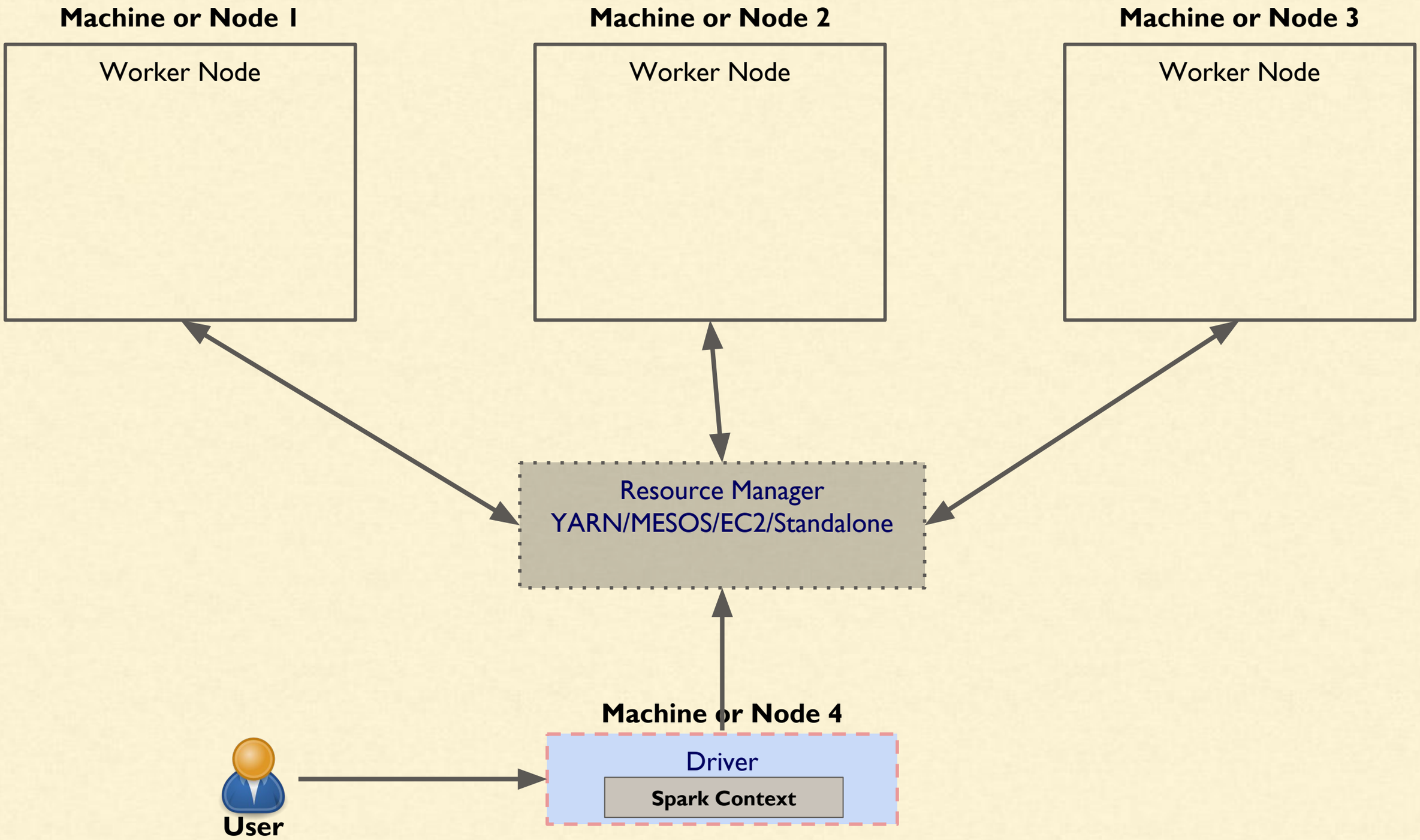
Worker Node



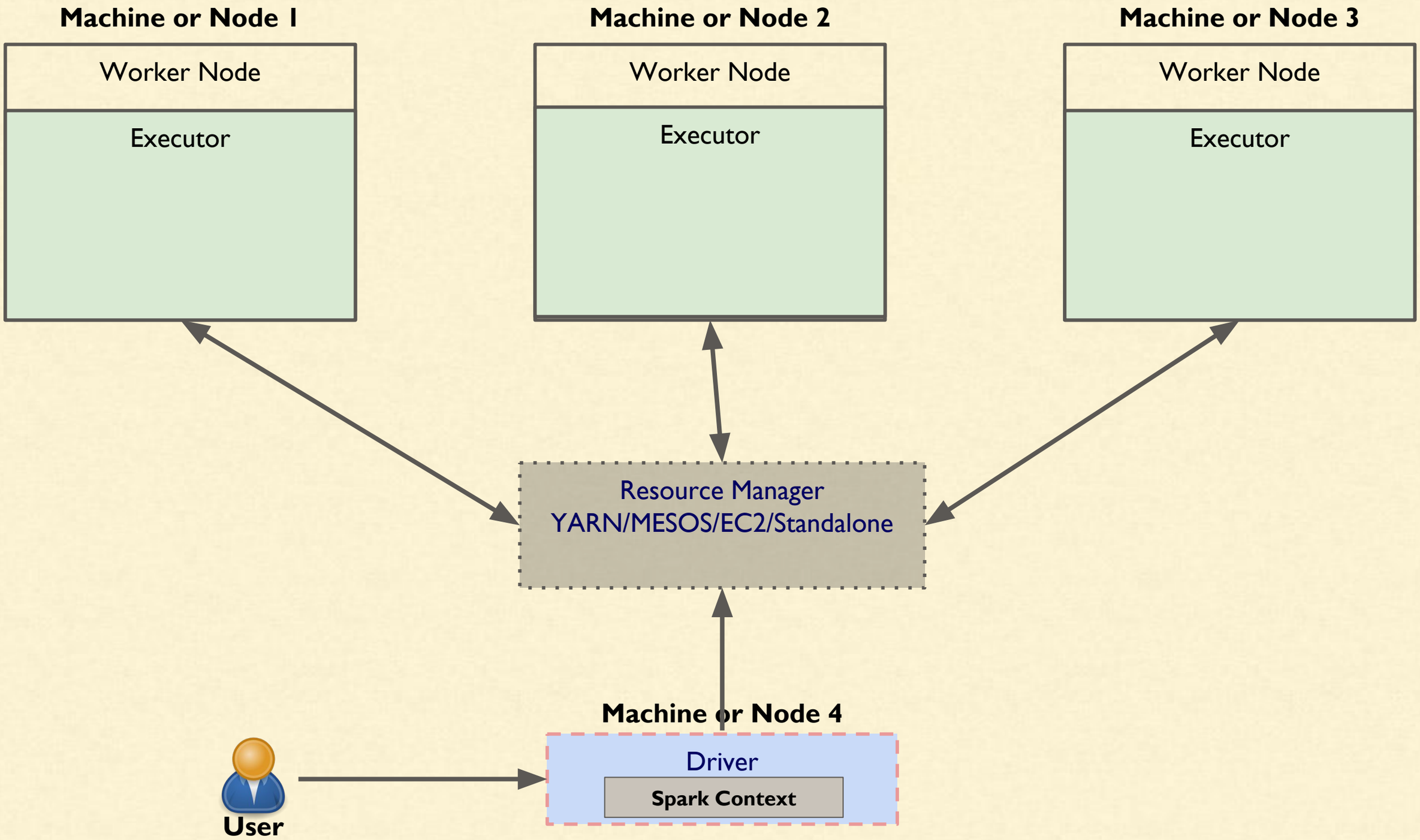
Cluster Manager

- It is a pluggable component in Spark.
- This allows Spark to run on YARN, Mesos & builtin Standalone

Understanding The Architecture



Understanding The Architecture



Executors

- Worker processes that run tasks of a job

Executors

- Worker processes that run tasks of a job
- Return results to the driver

Executors

- Worker processes that run tasks of a job
- Return results to the driver
- Launched once at the beginning of a Spark application

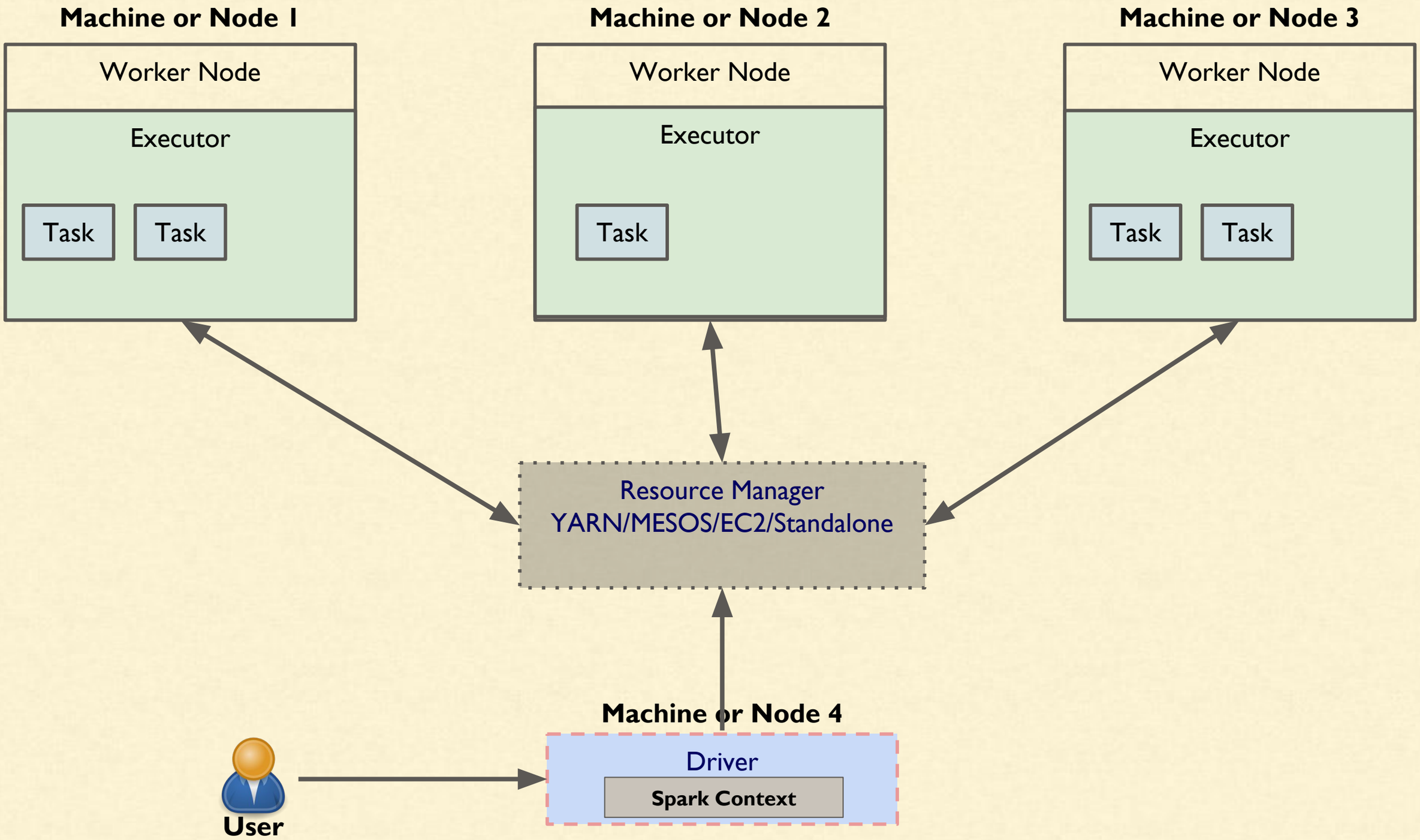
Executors

- Worker processes that run tasks of a job
- Return results to the driver
- Launched once at the beginning of a Spark application
- Run for the entire lifetime of an application,

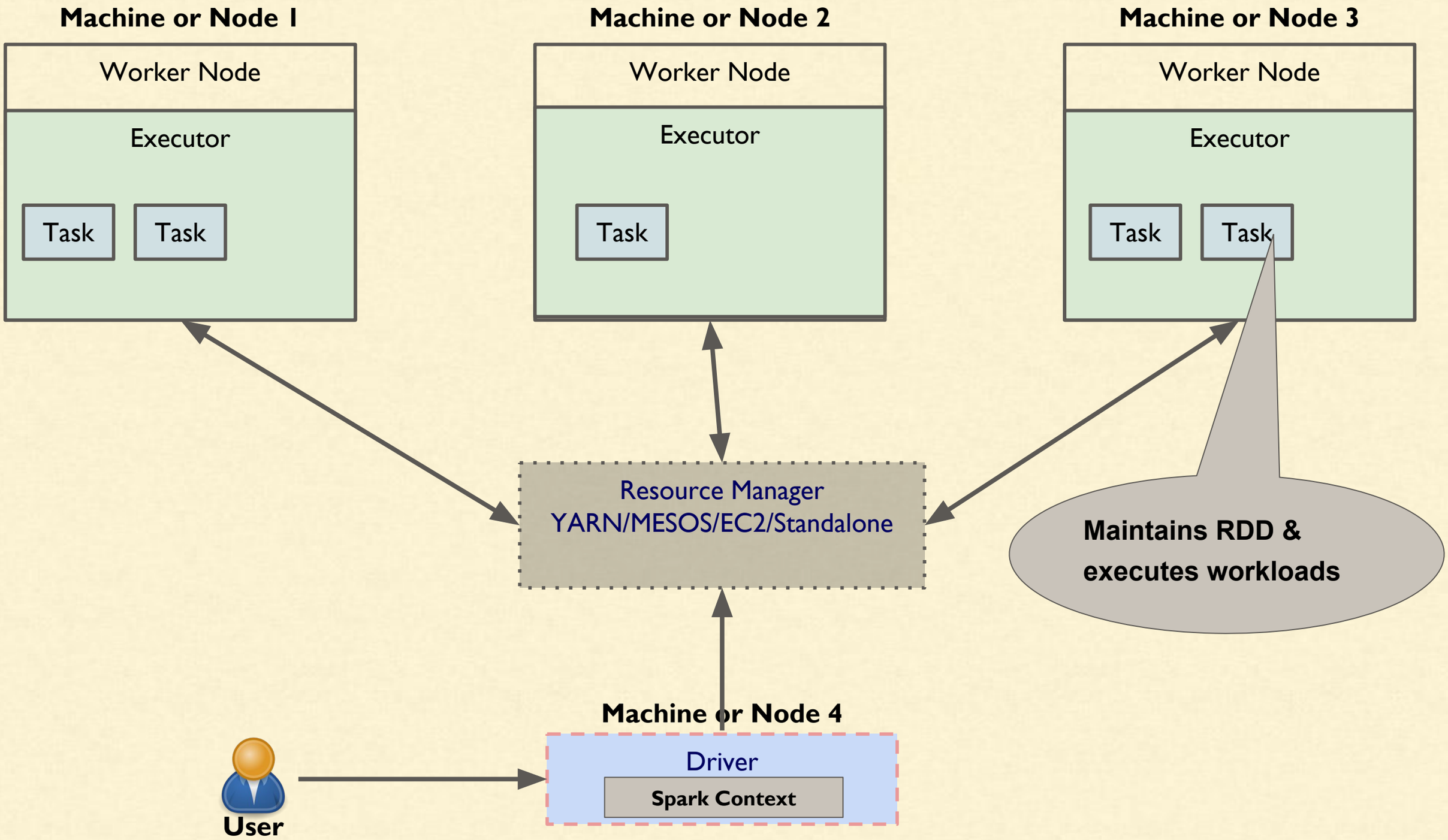
Executors

- Worker processes that run tasks of a job
- Return results to the driver
- Launched once at the beginning of a Spark application
- Run for the entire lifetime of an application,
- Provide in-memory storage for cached RDDs via Block Manager

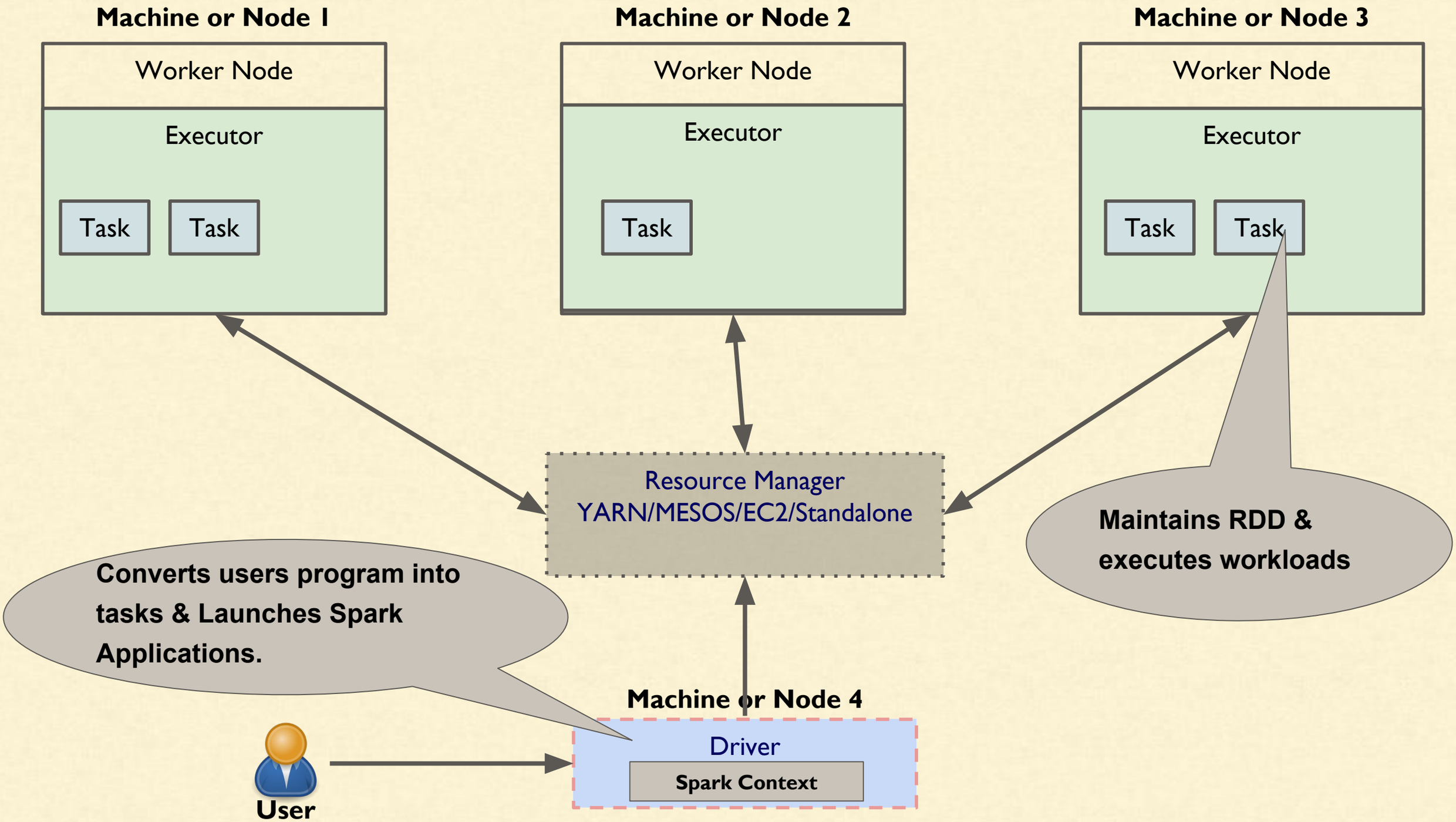
Understanding The Architecture



Understanding The Architecture



Understanding The Architecture



Launching a Program

Launching a Program

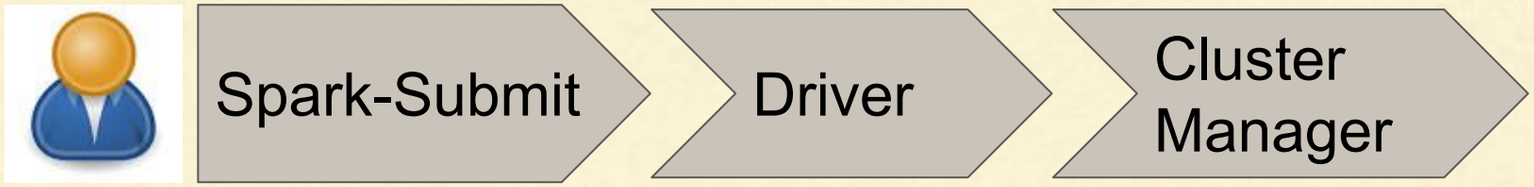
Spark-Submit

- The user submits an application using spark-submit.

Launching a Program

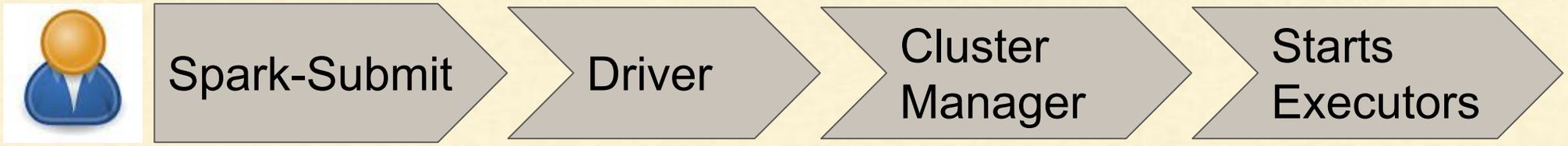
- The user submits an application using spark-submit.
- spark-submit launches the driver program

Launching a Program



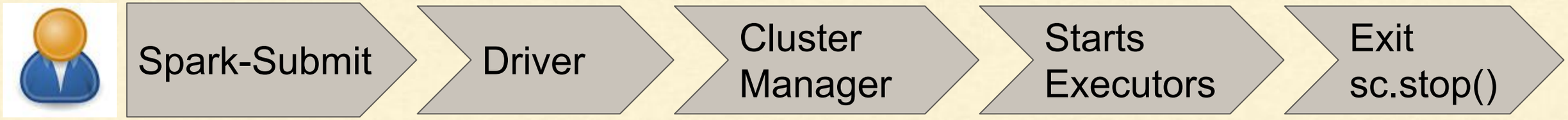
- The user submits an application using spark-submit.
- spark-submit launches the driver program
- driver invokes the main() and creates spark context
- The driver program contacts the cluster manager for resources

Launching a Program



- The user submits an application using spark-submit.
- spark-submit launches the driver program
- driver invokes the main() and creates spark context
- The driver program contacts the cluster manager for resources
- The cluster manager launches executors
- The driver process runs through the user application.
- the driver sends work to executors in the form of tasks.
- Tasks are run on executor processes to compute and save results.

Launching a Program



- The user submits an application using spark-submit.
- spark-submit launches the driver program
- driver invokes the main() and creates spark context
- The driver program contacts the cluster manager for resources
- The cluster manager launches executors
- The driver process runs through the user application.
- the driver sends work to executors in the form of tasks.
- Tasks are run on executor processes to compute and save results.
- Terminate the executors and release resources if driver's main() exit or sc.stop()

Getting Started - Two Modes

1. Local Mode
2. Cluster Mode

Getting Started - Two Modes

1. Local Mode

2. Cluster Mode

```
Spark-shell --master ....
```

Local Mode or Spark in-process

1. Default Mode

2. Does not require any resource manager

    a. Simply download and run.

3. Good for utilizing multiple cores for processing

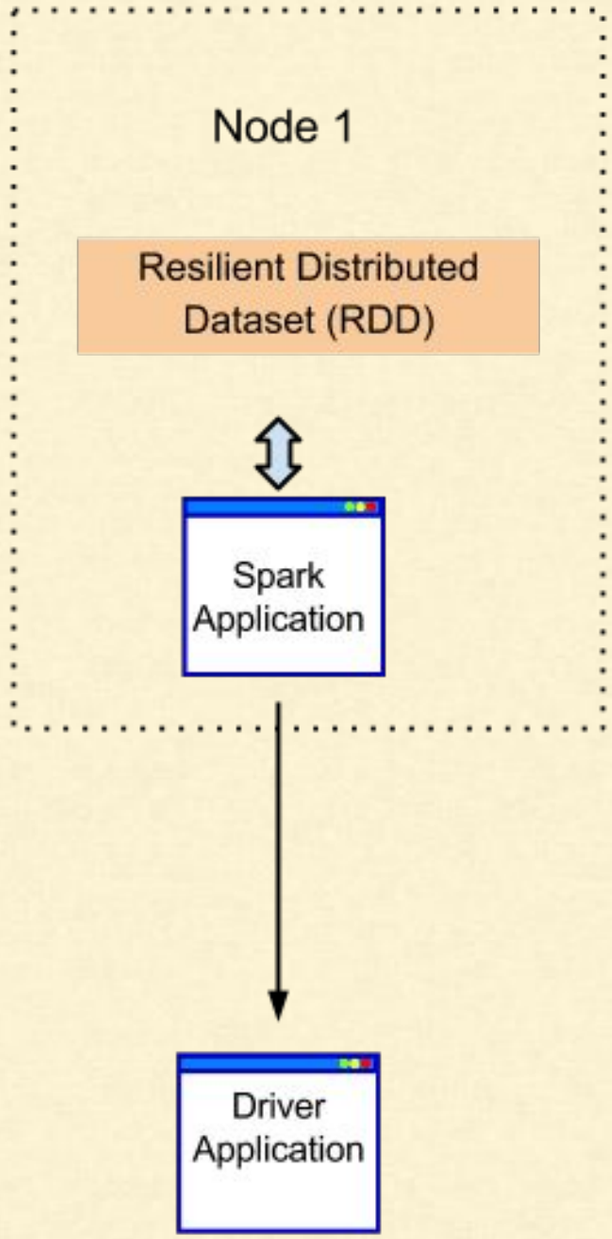
4. Partitions are generally equal to number of CPUs.

5. Used generally for testing

Getting Started - Local Mode

We can run spark-shell, spark-submit with

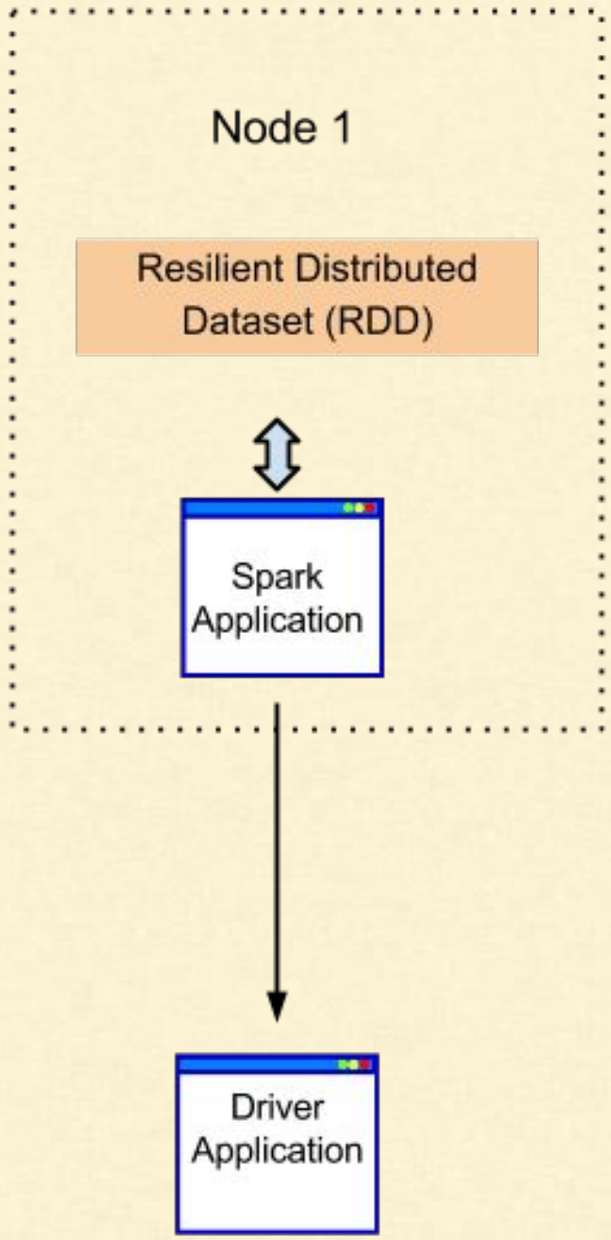
- *spark-shell*
- *spark-shell --master local*
- *spark-shell --master local[n]*
- *spark-shell --master local[*]*



Local Mode - Check!

```
scala> sc.isLocal

res0: Boolean = true
```



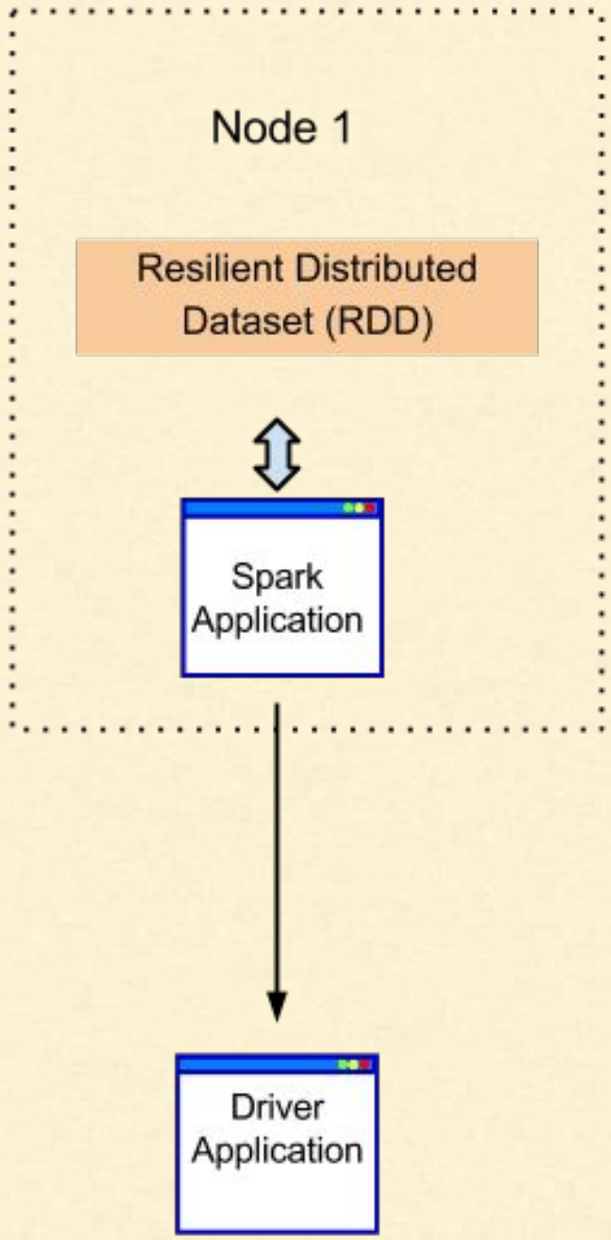
Local Mode - Check!

```
scala> sc.isLocal

res0: Boolean = true
```

```
scala> sc.master

res0: String = local[*]
```



Local Mode - HandsOn!

Cluster Modes

Different kind of Resource Managers

a. Standalone

b. YARN

c. Mesos

d. EC2

Cluster Mode - Standalone

Uses inbuilt manager resource manager

How to setup?

a. Install spark on all nodes.
b. Inform all nodes about each other
c. Launch spark on all nodes.
d. The spark nodes will discover each other

Installing Standalone Cluster

1. Copy a compiled version of Spark to the same location on all your machines—for example, /home/yourname/spark.
2. Set up password-less SSH access from your master machine to the others.
3. Edit the conf/slaves file on your master and fill in the workers' hostnames.
4. run sbin/start-all.sh on your master
5. Check http://masternode:8080
6. To stop the cluster, run bin/stop-all.sh on your master node.

Cluster Mode - YARN

To run spark inside Hadoop's YARN.
Tasks are run inside the yarn's containers

How to use?

```
export YARN_CONF_DIR=/etc/hadoop/conf/
export HADOOP_CONF_DIR=/etc/hadoop/conf/

spark-shell --master yarn
```

Launching a program on yarn - Hands On

1. export YARN_CONF_DIR=/etc/hadoop/conf/
2. export HADOOP_CONF_DIR=/etc/hadoop/conf/
3. *spark-submit --master yarn --class org.apache.spark.examples.SparkPi /usr/hdp/current/spark-client/lib/spark-examples-*.jar 10*

Launching a program on yarn

Hands On video

Cluster Mode - MESOS

1. Mesos Is a general-purpose cluster manager
2. it runs both analytics workloads and long-running services (DBs)
3. To use Spark on Mesos, pass a mesos:// URI to spark-submit:
   spark-submit --master mesos://masternode:5050 yourapp
4. You can use ZooKeeper to elect master in mesos in case of multi-master
5. Use a mesos://zk:// URI pointing to a list of ZooKeeper nodes.
6. Ex:, if you have 3 nodes (n1, n2, n3) having ZK on port 2181, use URI:
   mesos://zk://n1:2181/mesos,n2:2181/mesos,n2:2181/mesos

Cluster Mode - Amazon EC2

- Spark comes with a built-in script to launch clusters on Amazon EC2.
- First create an Amazon Web Services (AWS) account
- Obtain an access key ID and secret access key.
- export these as environment variables:
  - export AWS_ACCESS_KEY_ID="..."
  - export AWS_SECRET_ACCESS_KEY="..."
- Create an EC2 SSH key pair and download its private key file (helps in SSH)
- Launch command of the spark-ec2 script:
  - cd /path/to/spark/ec2
  - ./spark-ec2 -k mykeypair -i **mykeypair.pem** launch mycluster

Deployment Modes

- Based on where does driver run.
- Two ways:
  - **Client**
  - **Cluster**

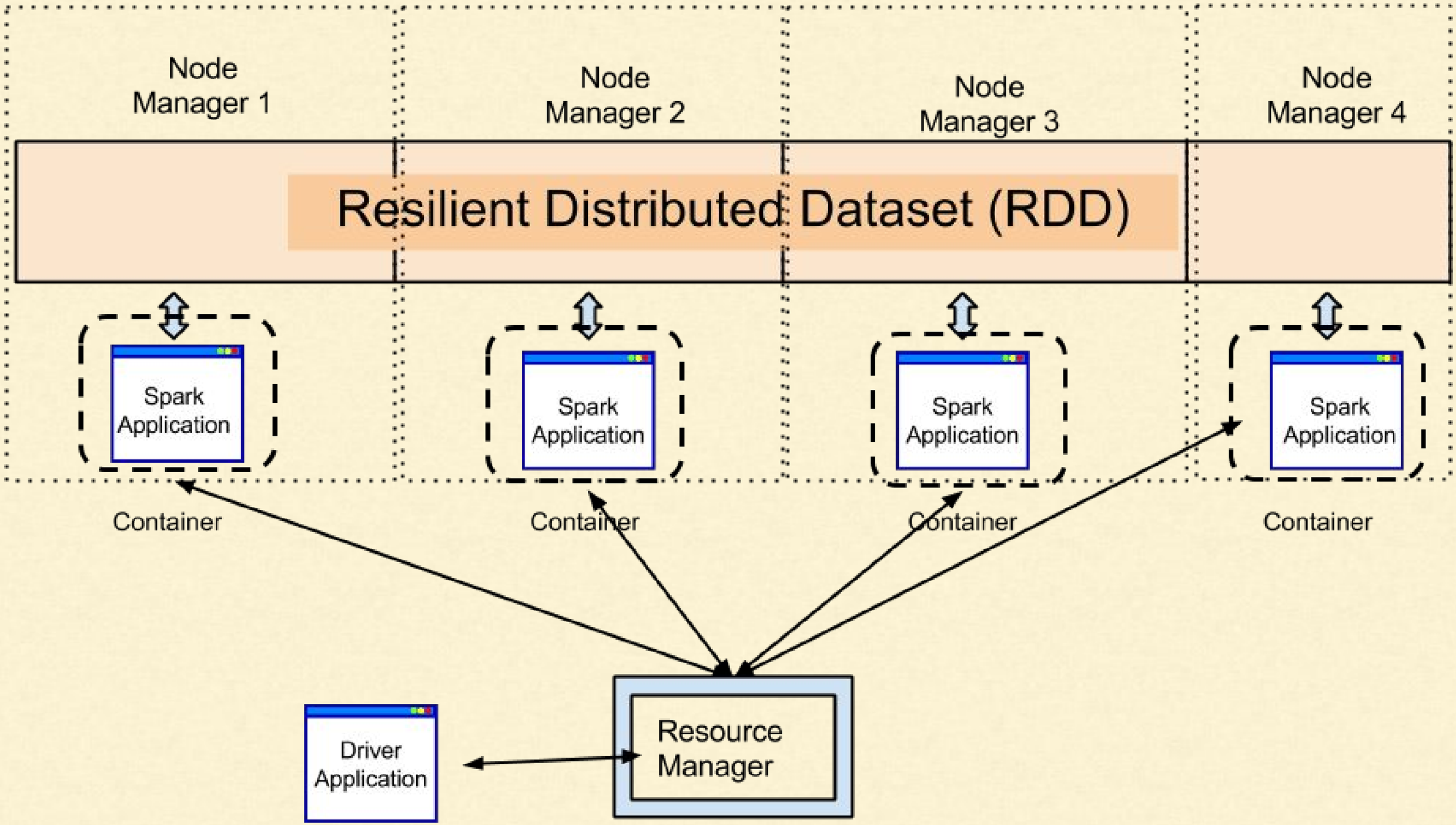
Deployment Modes

- Based on where does driver run.
- Two ways:
  - **Client -** launch the driver program locally. Default
  - **Cluster**

Deployment Modes

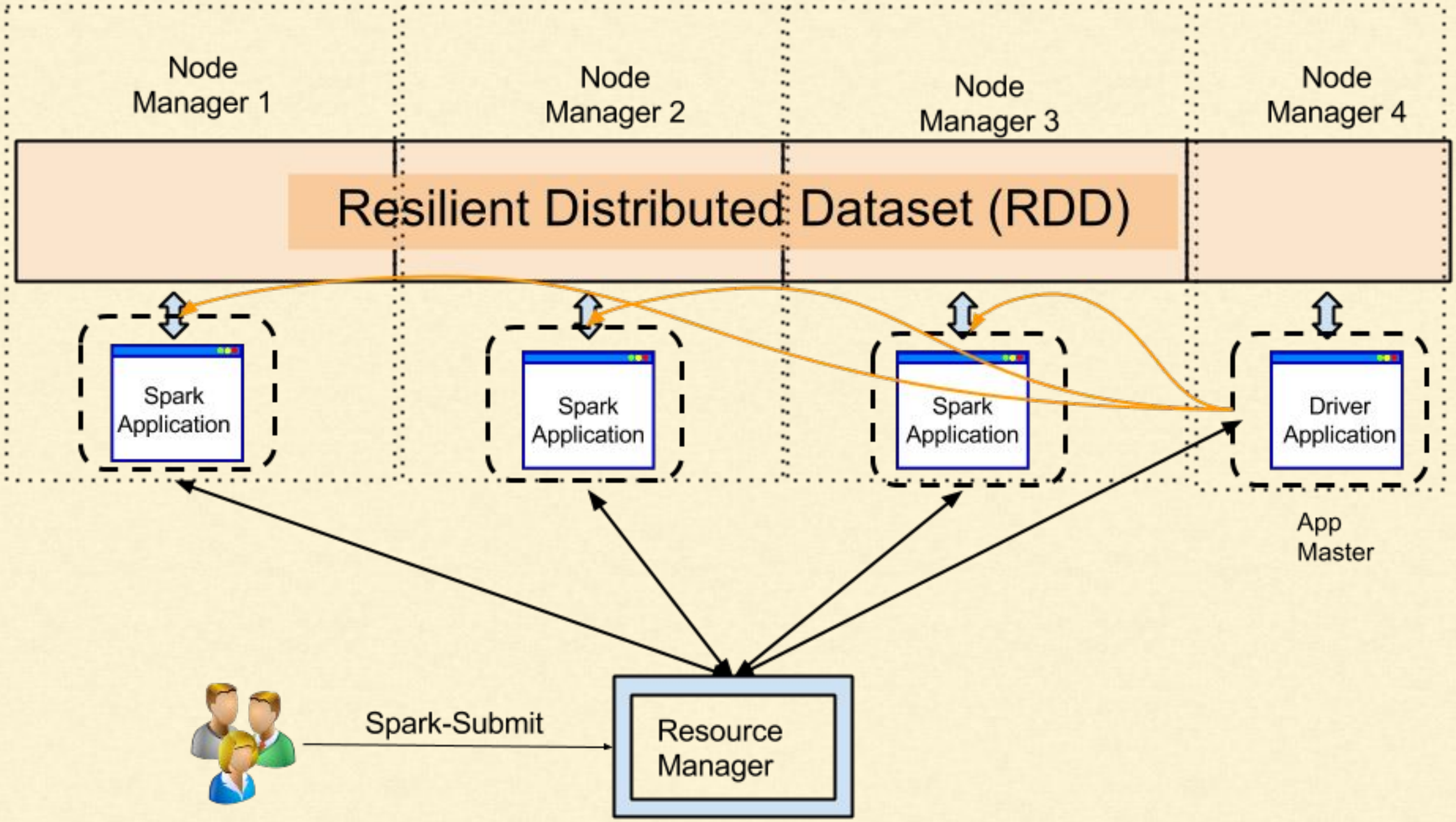
- Based on where does driver run.
- Two ways:
  - **Client -** launch the driver program locally. Default
  - **Cluster -** on one of the worker machines inside the cluster

Architecture Yarn Client Mode



1. Driver Application is runs outside yarn
   a. On machine where it is launched
2. If Driver Application shuts down the process is killed
3. Does not have resilience but is quicker to run.

Architecture Yarn Cluster Mode



1. Driver Application runs inside yarn in application master
2. If launcher shuts down the process continues like a batch process
   a. in background
3. Preferred way to run the long running processes

Architecture Yarn cluster Mode - Example

export YARN_CONF_DIR=/etc/hadoop/conf/
export HADOOP_CONF_DIR=/etc/hadoop/conf/

spark-submit --master yarn --deploy-mode cluster --class org.apache.spark.examples.SparkPi /usr/hdp/current/spark-client/lib/spark-examples-.jar 10*

To check the status, use:

- http://e.cloudxlab.com:4040/
- http://a.cloudxlab.com:8088/cluster

Architecture Yarn cluster Mode - Demo

Hand On Video

Which Cluster Manager to Use?

1. Start with a local mode if this is a new deployment.
2. To use richer resource scheduling capabilities (e.g., queues), use YARN and Mesos
3. When sharing amongst many users is primary criteria, use Mesos
4. In all cases, it is best to run Spark on the same nodes as HDFS for fast access to storage.
   a. You can either install Mesos or Standalone cluster on Datanodes
   b. Or Hadoop distributions already install YARN and HDFS together

Packaging Your Code and Dependencies

1. Bundle all the libraries that your program depends upon
2. No need to bundle the spark libraries (org.apache.spark) and language libraries (java…)
3. Python users can:
   a. Either install on all nodes using pip or easy_install
   b. Or use --py-files argument (take files to every node's cwd) of spark-submit
4. Java & Scala
   a. Submit libraries using --jars
   b. But there are many libraries, use build tool such as sbt or maven

Common flags for spark-submit

Flag	Explanation
master	Indicates the cluster manager to connect to. The options for this flag are described earlier.

Common flags for spark-submit

Flag	Explanation
deploy-mode	Whether to launch the driver program locally ("client") or on one of the worker machines inside the cluster ("cluster"). In client mode spark-submit will run your driver on the same machine where spark-submit is itself being invoked. In cluster mode, the driver will be shipped to execute on a worker node in the cluster. The default is client mode.

Common flags for spark-submit

Flag	Explanation
class	The “main” class of your application if you’re running a Java or Scala program.

Common flags for spark-submit

Flag	Explanation
name	A human-readable name for your application. This will be displayed in Spark's web UI.

Common flags for spark-submit

Flag	Explanation
jars	A list of JAR files to upload and place on the classpath of your application. If your application depends on a small number of third-party JARs, you can add them here.

Common flags for spark-submit

Flag	Explanation
files	A list of files to be placed in the working directory of your application. This can be used for data files that you want to distribute to each node.

Common flags for spark-submit

Flag	Explanation
py-files	A list of files to be added to the PYTHONPATH of your application. This can contain .py, .egg, or .zip files.

Common flags for spark-submit

Flag	Explanation
executor-memory	The amount of memory to use for executors, in bytes. Suffixes can be used to specify larger quantities such as “512m” (512 megabytes) or “15g” (15 gigabytes).

Common flags for spark-submit

Flag	Explanation
driver-memory	The amount of memory to use for the driver process, in bytes. Suffixes can be used to specify larger quantities such as “512m” (512 megabytes) or “15g” (15 gigabytes).

Running on a Cluster

Thank you!

Deployment Mode

As yarn-client	**yarn cluster**
1. Driver runs on the client 2. Client can't disconnect 3. --master yarn-client	1. Driver runs on Application master insite yarn 2. Client can disconnect after starting 3. --master yarn-cluster