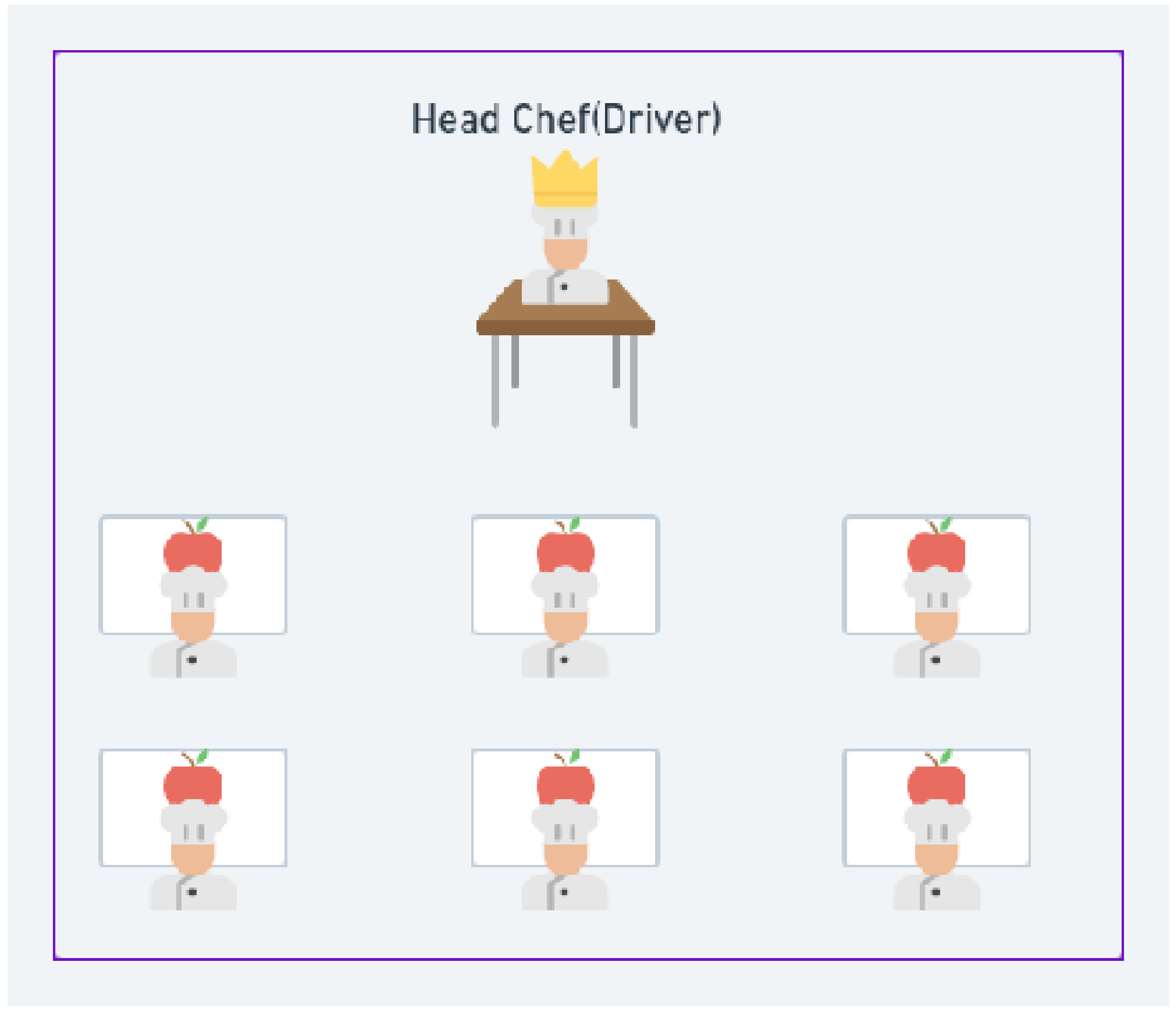
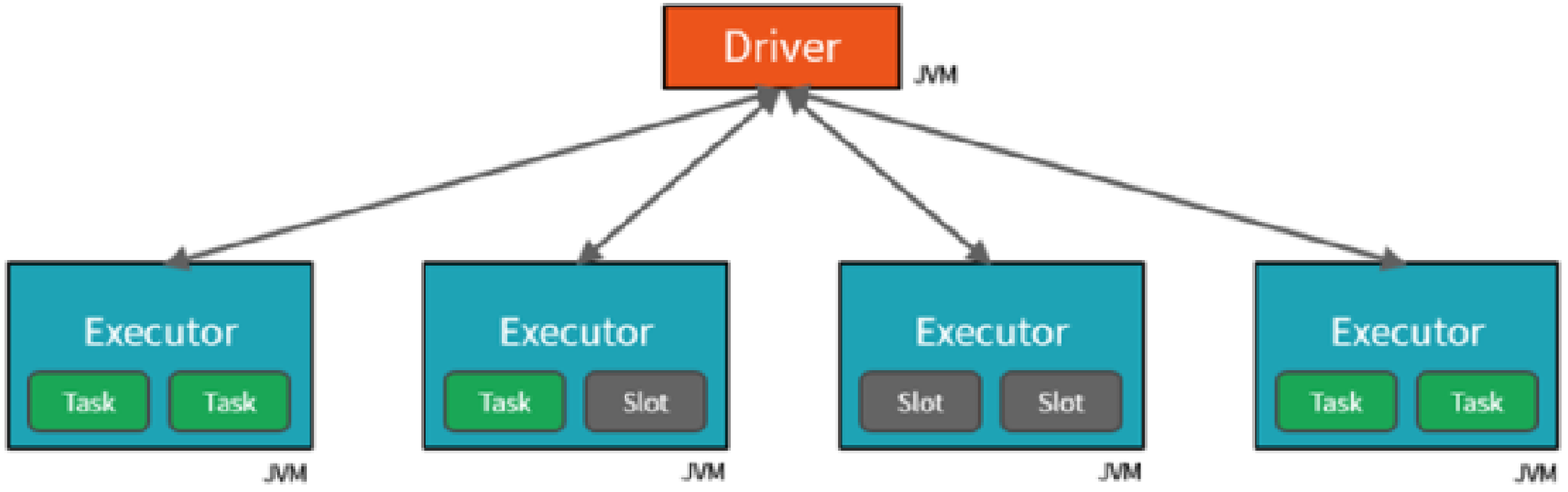
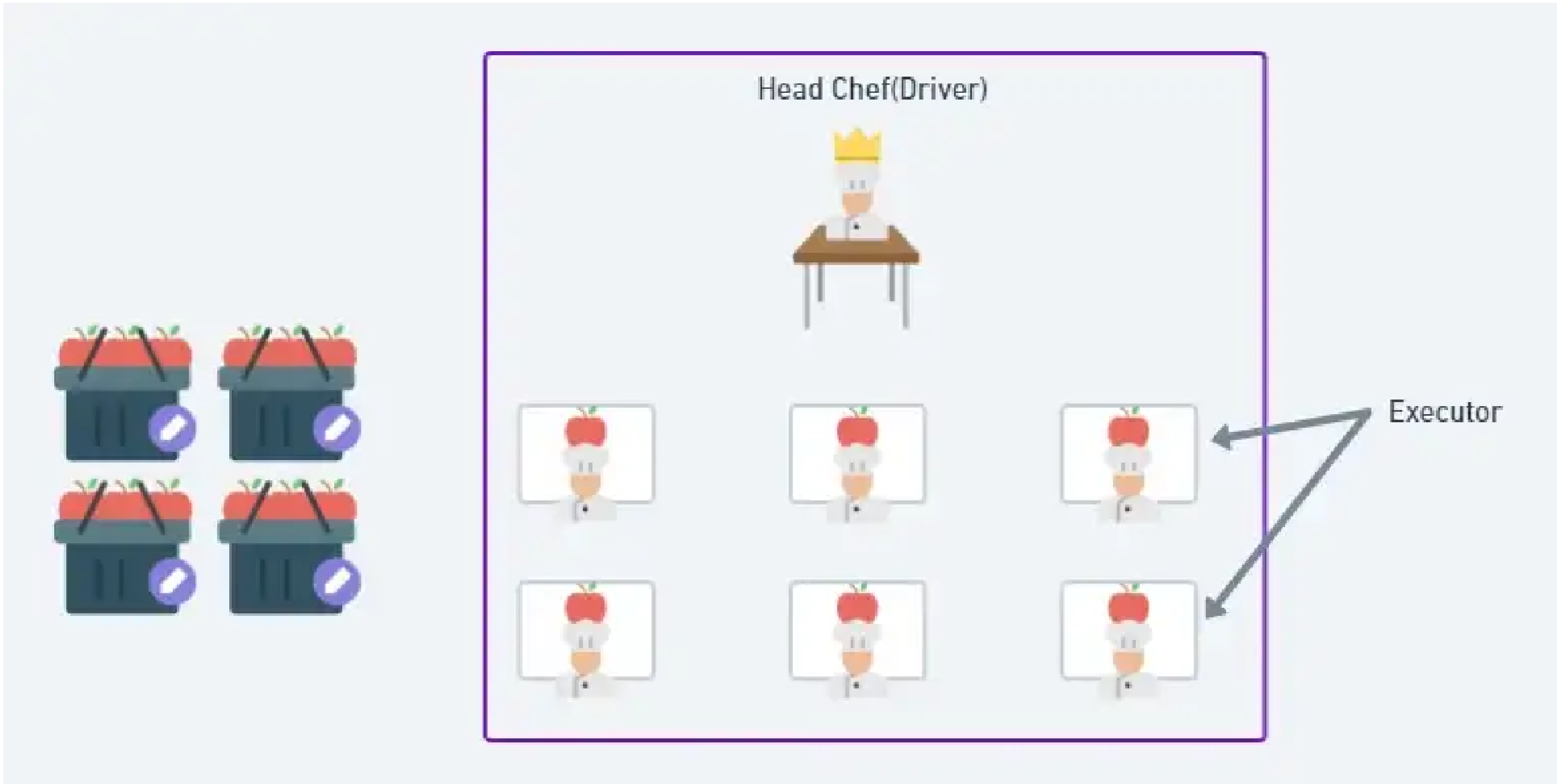


Driver: Consider Driver as a head chef/manager of the restaurant who distributes the work among the other chef.

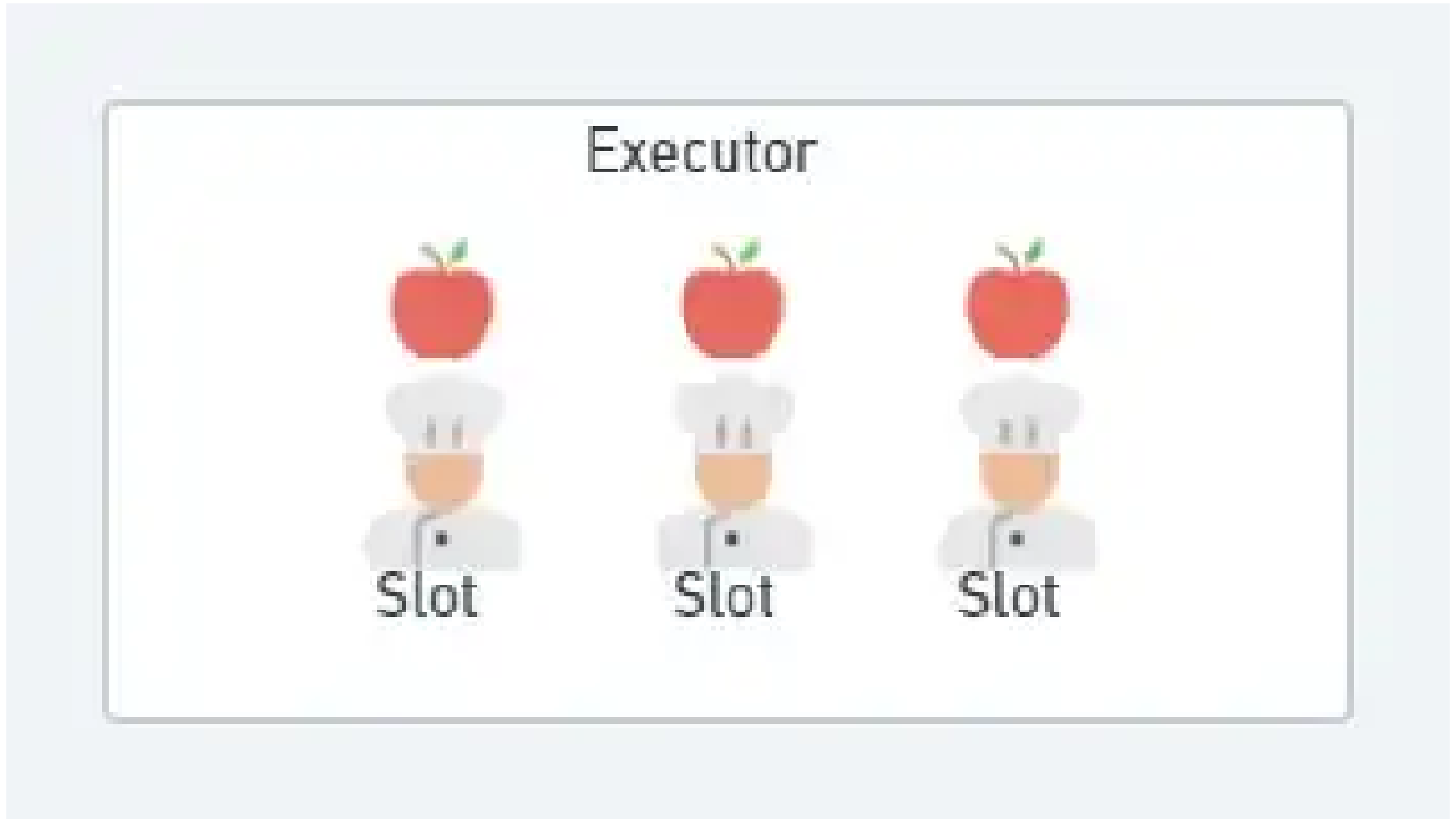
In spark, the Driver is responsible for assigning Tasks to the executors. From a Data engineer's perspective if we want to filter a table consisting of 1000 rows these rows will be divided and distributed across the executors by Driver. Note that the driver will not be allowed to view the data.


Driver
JVM
Executor
Task
Task
JVM
Executor
Task
Slot
JVM
Executor
Slot
Slot
JVM
Executor
Task
Task
JVM

Executors: Consider executors as a workstation/chopping board where the chef performs the task(cutting apples).



In spark, Cores/Slots are the CPU that performs the actual task. The total number of cores present is responsible for executing more tasks parallelly.



In spark, the executors are the Java Virtual Machine where your task is being performed. The executors are responsible for running the individual task and sending the result to the driver.

Slots/Cores: Slots or Cores are referred to the chefs performing the tasks of cutting apples. An executor consists of Cores/Slots. An executor(workstation) can consist of one or more cores/slots.

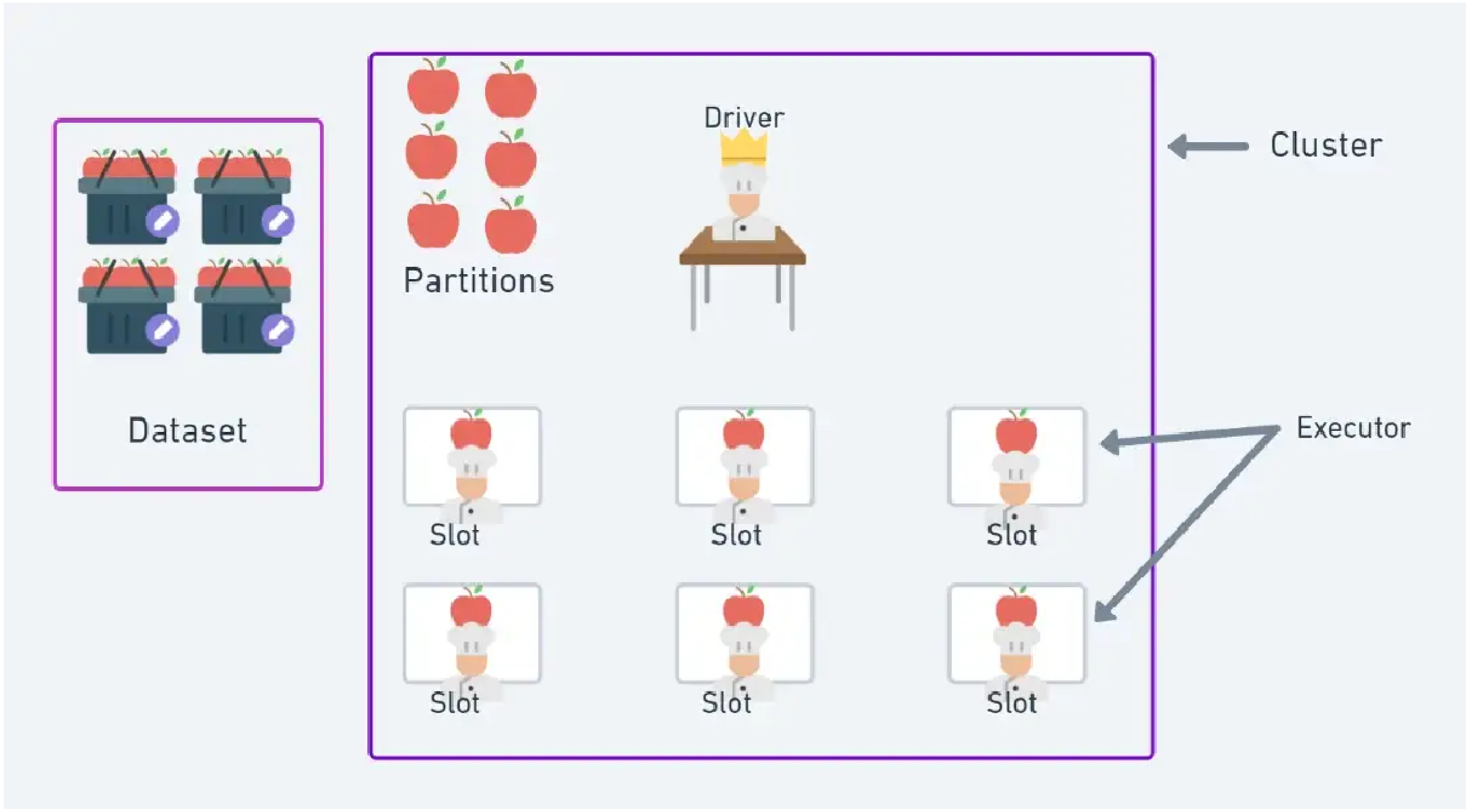
Jobs, Tasks, Dataset, Partitions:

Dataset is a collection of apples in a basket.

Partitioning is the process of splitting the dataset into smaller parts. Partitioning is done by the driver.

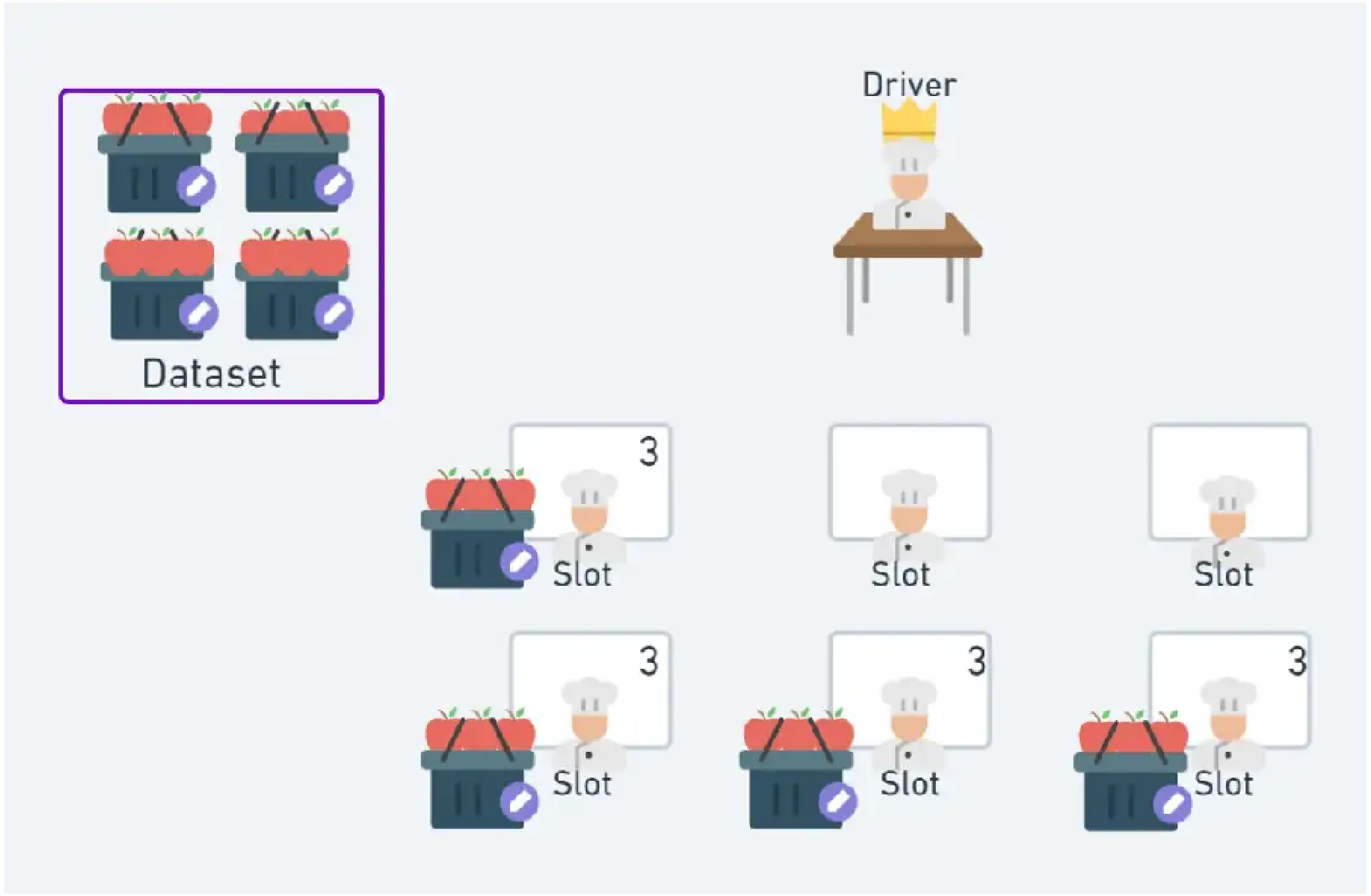
Tasks are usually the total number of partitions that needs to process for example if we have 4 baskets of 3 apples in each then we have 12 apples(Dataset) to be cut. This is divided into 12 partitions by driver hence 12 tasks have to be performed in total. Each task is performed by each core. Hence if we have 3 executors of 2 cores in each executor then we have 6 cores in total therefore there can be only a maximum of 6 tasks that can be performed parallelly.

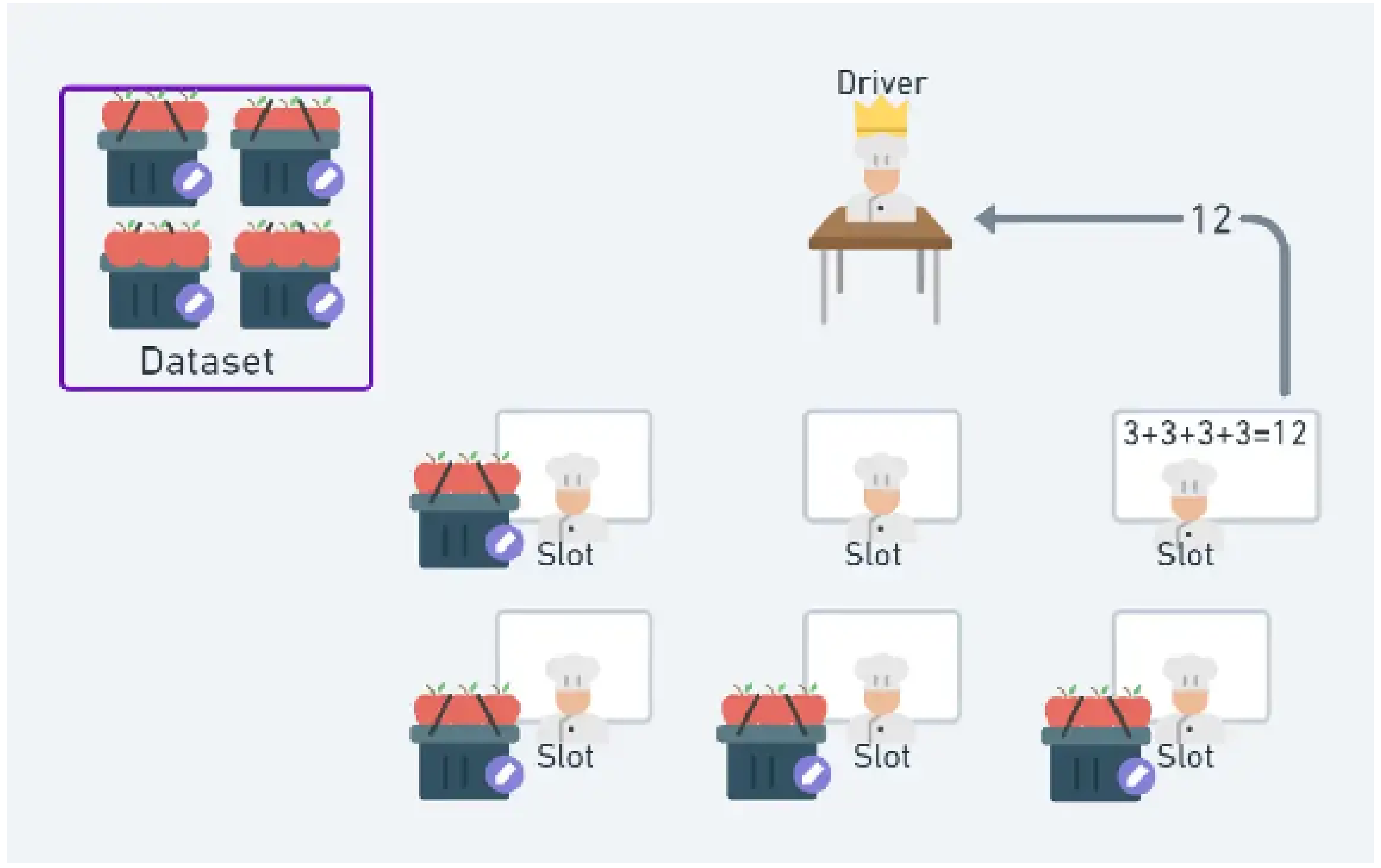
Job is the overall instruction that the driver receives. In this case, we need to cut 4 baskets of 3 apples in each.


Dataset
Partitions
Driver
Cluster
Executor
Slot
Slot
Slot
Slot
Slot
Slot

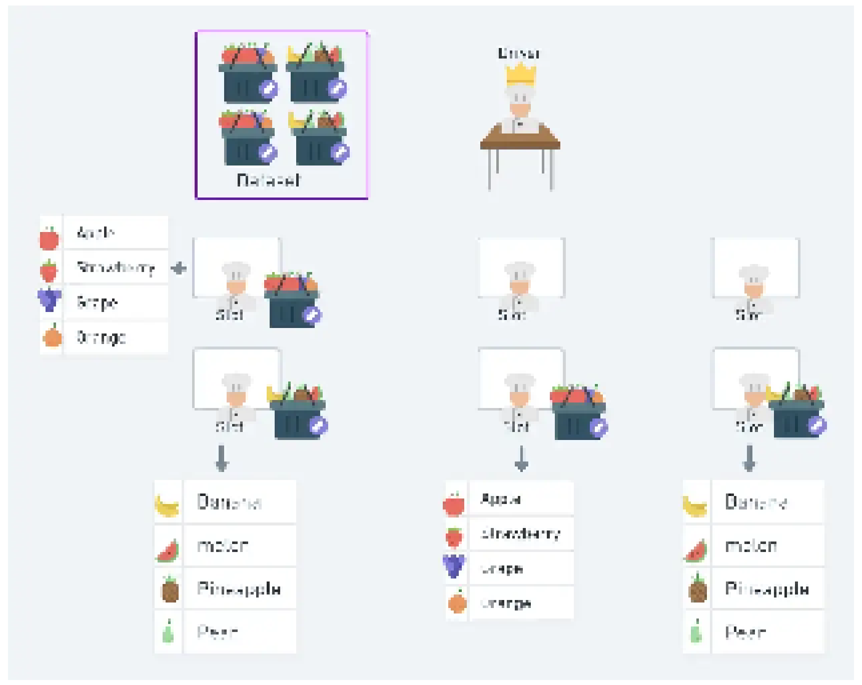
Stages:
Stages in Spark represent groups of tasks that can be executed together to compute the same operation on multiple machines. As we know already that the driver is not allowed to see the dataset then have you wondered how the driver knows the total no of rows in the dataset or apples in our example? Spark tackles this issue by distributing the dataset across executors and asking them for the count. Let’s understand the entire process by going through two stages.

Stage 1: In this stage, the driver divides the dataset and distributes them to a set of executors. The driver now asks the executors to count the number of rows in the dataset. In this stage, the work of the executors is to find the no of apples in each basket. Now the count is stored within each executor on the disk to use in later stages this is also called shuffle data.


Dataset
Driver
3
Slot
Slot
Slot
3
Slot
3
Slot
3
Slot



Stage 2: Even now each of the executors can't give the result directly to the driver and ask it to count the sum. So this process of adding the count of apples from each executor is performed by another executor and the total sum is given to the driver finally.

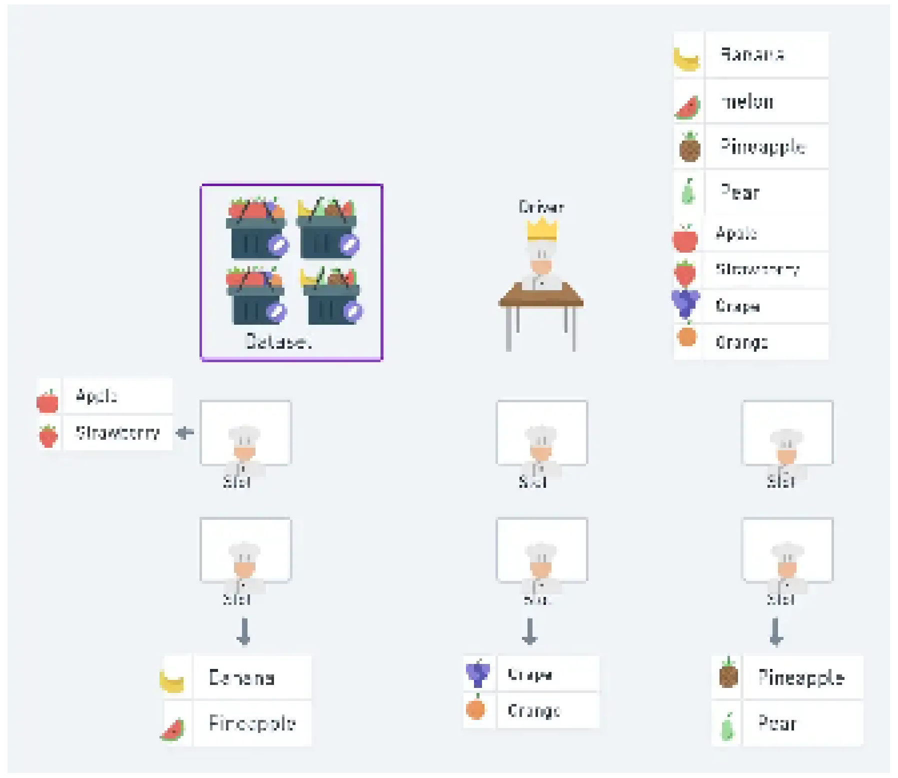


Since an executor is required to collect the information from all the other executors these operations are called wide transformation. Transformation is an instruction given by the spark to produce a new dataframe from an existing one.

Whenever there is an exchange of data between executors then we have a wide transformation and if no exchange of data is done then it's called a narrow transformation.

Now after these two stages are done we have now a total no of apples in our 12 it is. Now the driver exactly knows how to divide these apples among the other chefs. Now let us convert our understanding into stages.

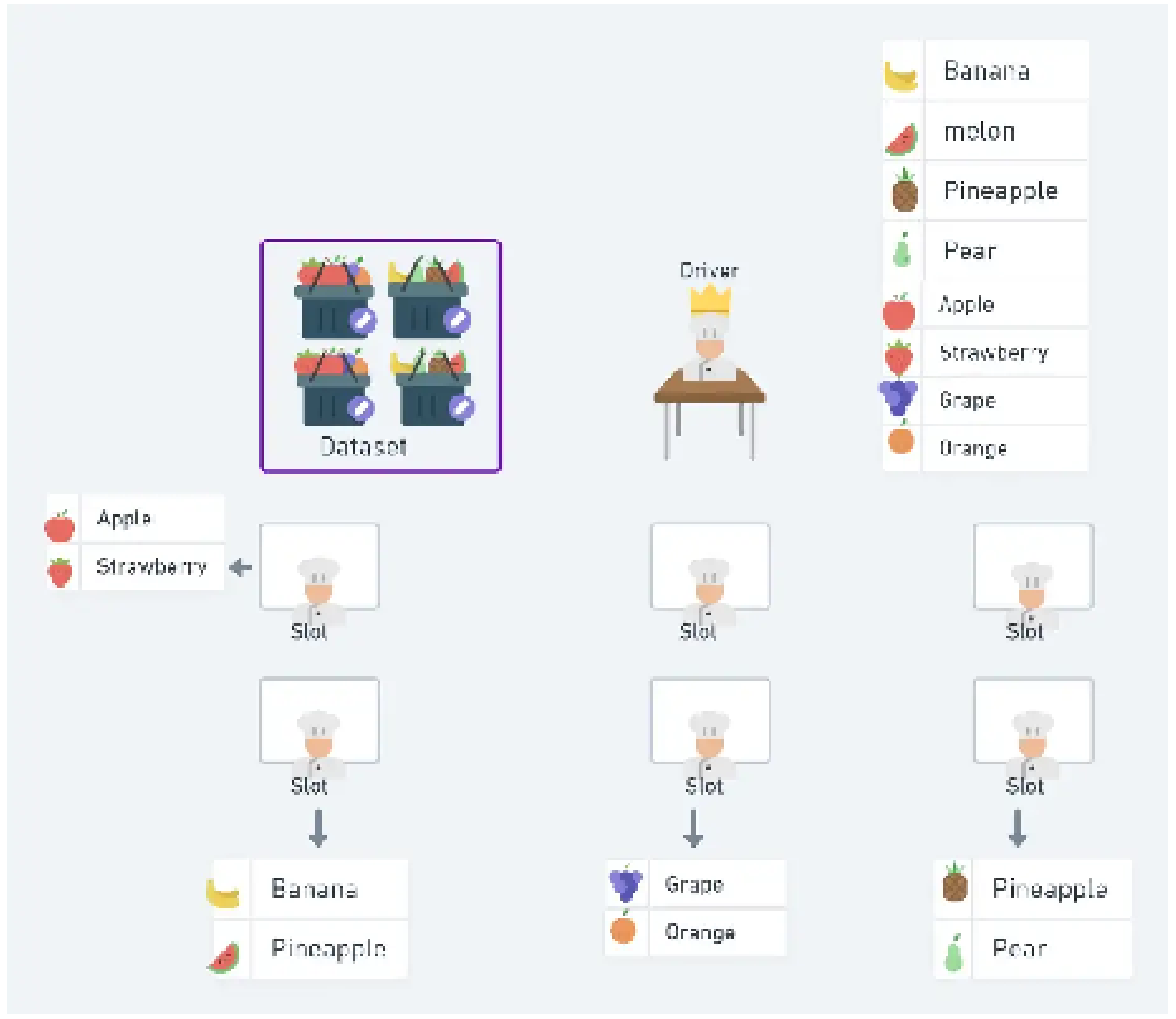
Stage 1:Divide the baskets among executors
Stage 2:Count the total and deliver the result to the executor
Stage 3:Driver divides the apple among the executor and the executor cut them into pieces.
Stage 4: An executor is assigned to collect all the pieces from the other executors and deliver the result to the driver.

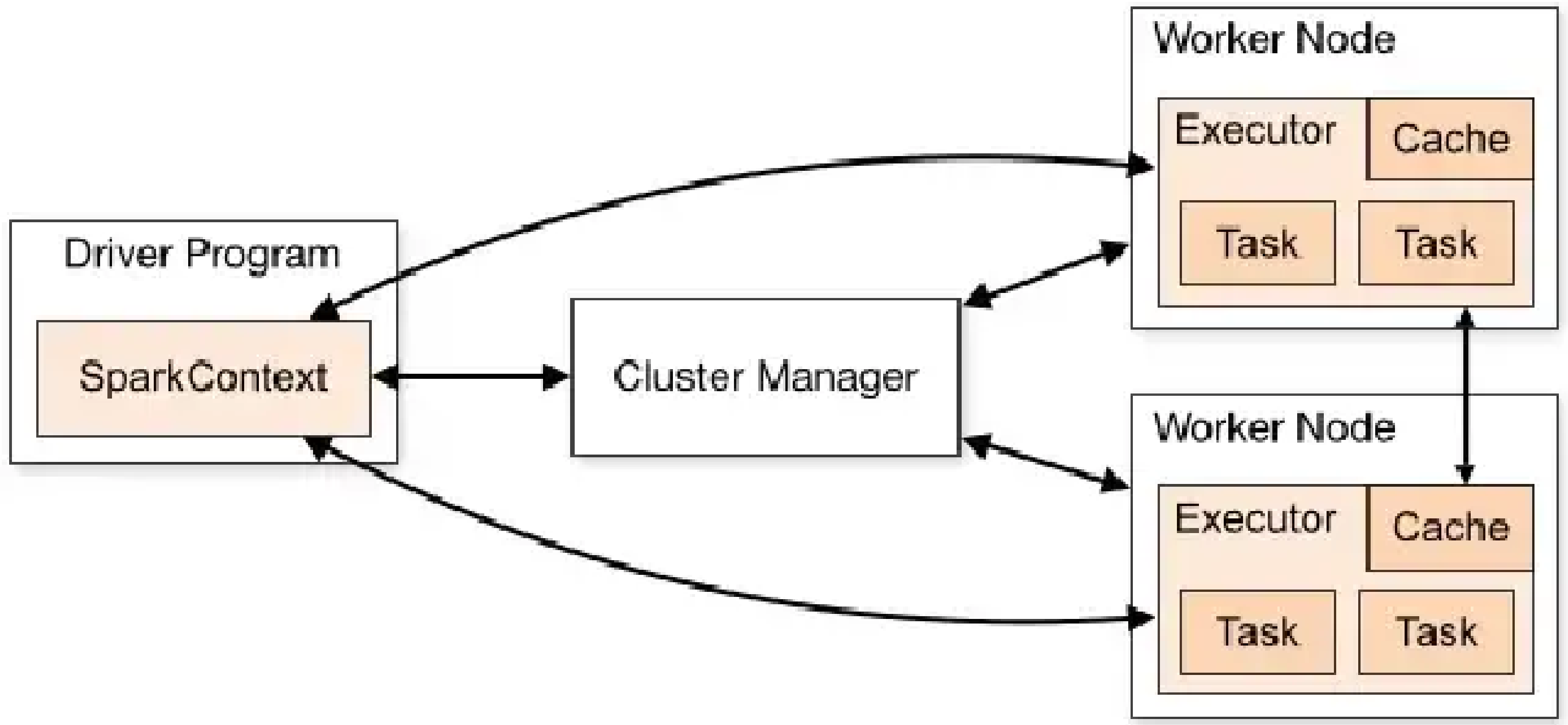


Shuffling:
Shuffling is the process of rearranging data within the cluster between stages. To understand shuffle better, we will take an example now our task is to count the distinct() fruit types present in our basket. Same as before now this task is done in two stages.

Stage 1: The basket is distributed among executors. Now each executor is performing in counting the distinct fruits in the basket. This data is now stored in the disk of each executor.

Stage 2: Now each executor reads the data stored in another executor to find the total distinct fruits. This process of sharing data across the executors is known as shuffling. Shuffling is an expensive process in terms of architecture. Hence it's best to reduce our number of wide transformations as much as possible. Most of the time spark does it by it selves and the process is known as pipelining.



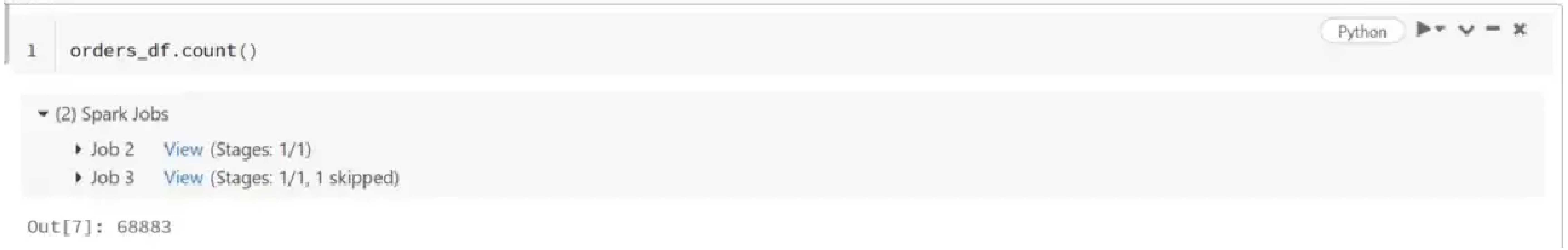


The cluster manager is responsible for maintaining the driver and the executors. Currently, there are three cluster managers available standalone cluster manager, Apache Mesos, and Hadoop YARN.

Step 1: The user will submit a spark application. Here we are requesting the cluster manager to accept our request and allocate the resources for our application. You can find more on how to write a spark-submit application with examples here.

Step 2: Now your application should create a SparkSession. This initializes the drivers and executors present in the cluster. The SparkSession will subsequently communicate with the cluster manager, asking it to launch Spark executor processes across the cluster. The number of executors and their relevant configurations are set by the user via the command-line arguments in the original spark-submit call.

orders_df=spark.read.csv('dbfs:/FileStore/tables/Data/orders/part_00000',schema="""order_id INT,order_date DATE,order_customer_id INT,order_status STRING""")



```
orders_df.count()
```

▾ (2) Spark Jobs
- ▸ Job 2 View (Stages: 1/1)
- ▸ Job 3 View (Stages: 1/1, 1 skipped)

```
Out[7]: 68883
```