Resource and Application Management

Mirek Riedewald

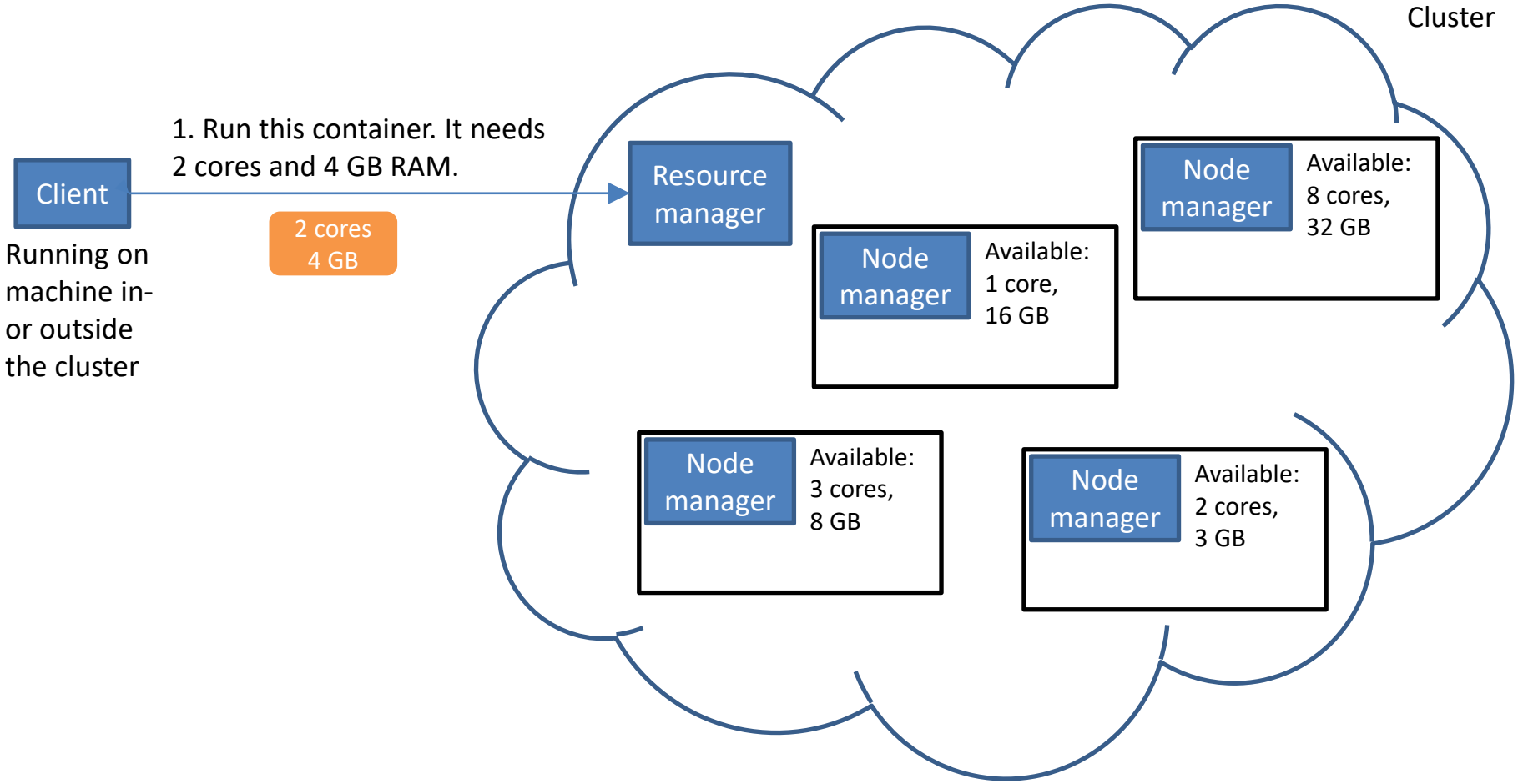
Key Learning Goals

- What is the difference between cluster resource manager and application master?
- What is the driver of a job?
- What is the difference between a job and a task?
- What is the main reason for separating resource management from application management?

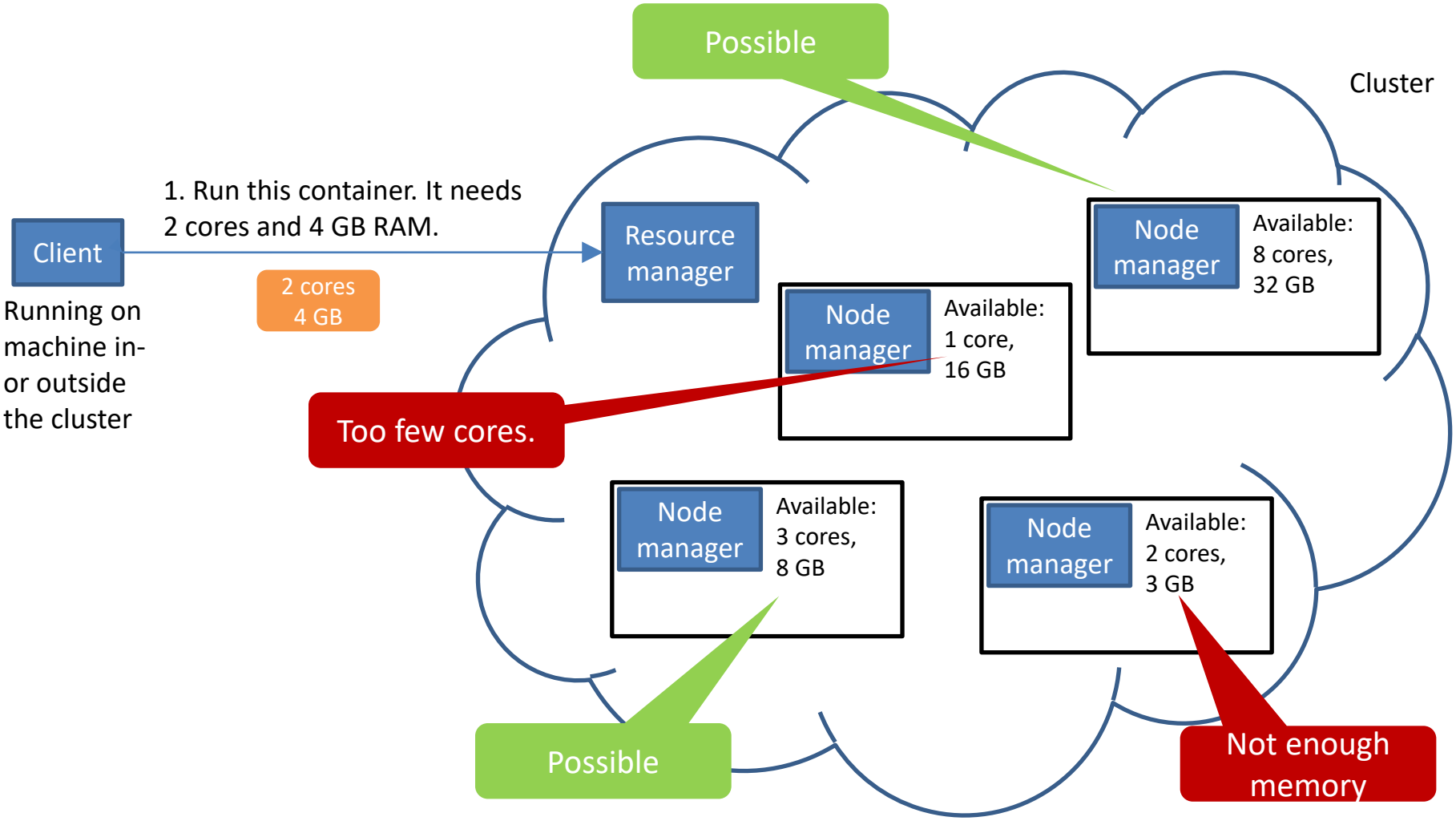
Introduction

- Clusters have resources; jobs and services need resources. The resource manager decides *who* gets *which* resources *when*. Cluster resource managers usually focus on CPU cores and memory.
- There is a single resource manager for the entire cluster. As we have seen for the distributed file system, this greatly simplifies consensus in a distributed system. And like the GFS master process, the resource manager only exchanges small control messages with processes requesting resources.
  - In addition to MapReduce and Spark, many other services and applications may run in a cluster. They all communicate with the same cluster resource manager.
- Scheduling is a classic computer-science problem with many solutions. We focus on the functionality of YARN, a popular open-source scheduler.

Basic View of Cluster Resource Management



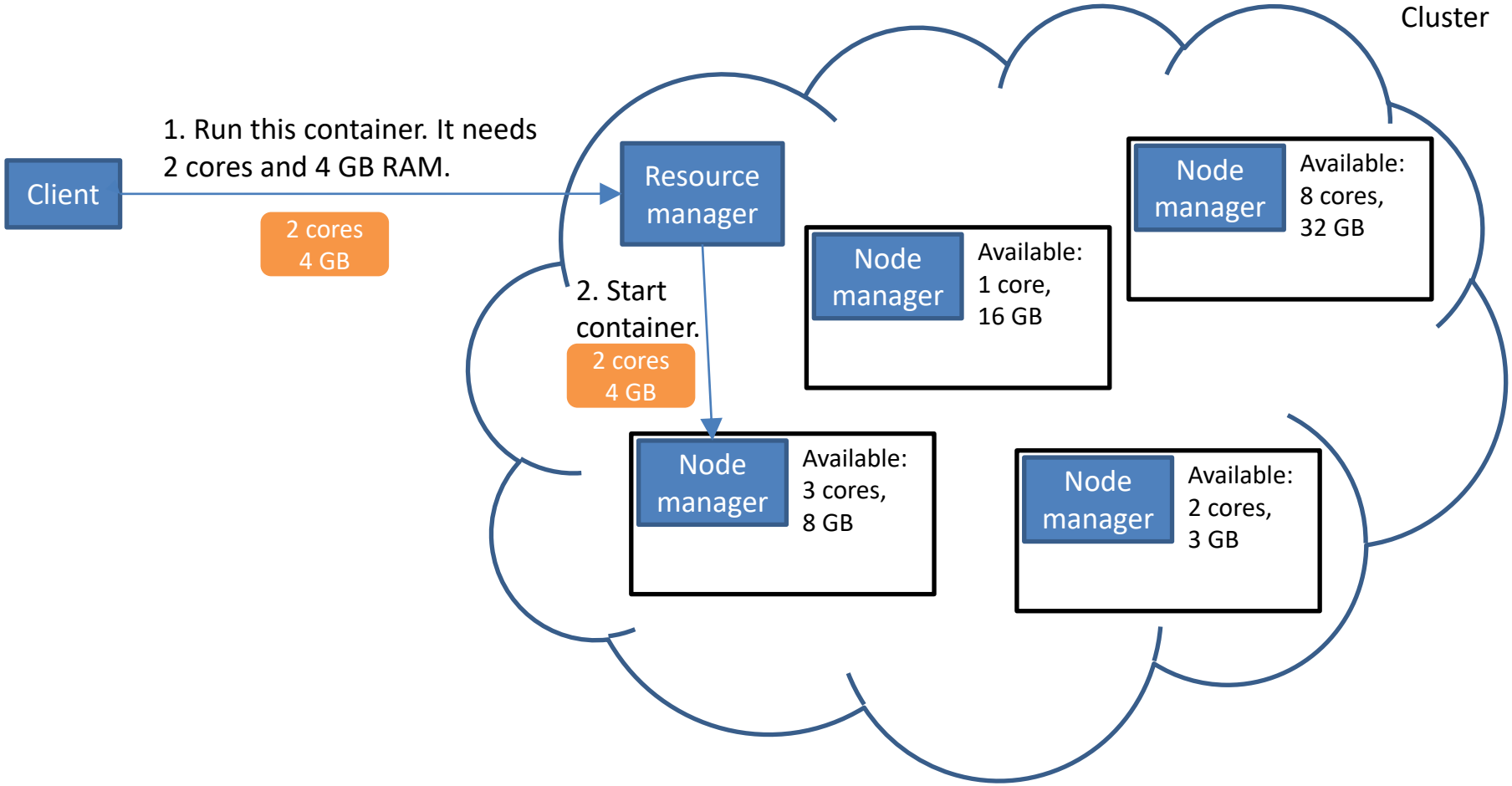
Who Will Run the Application?



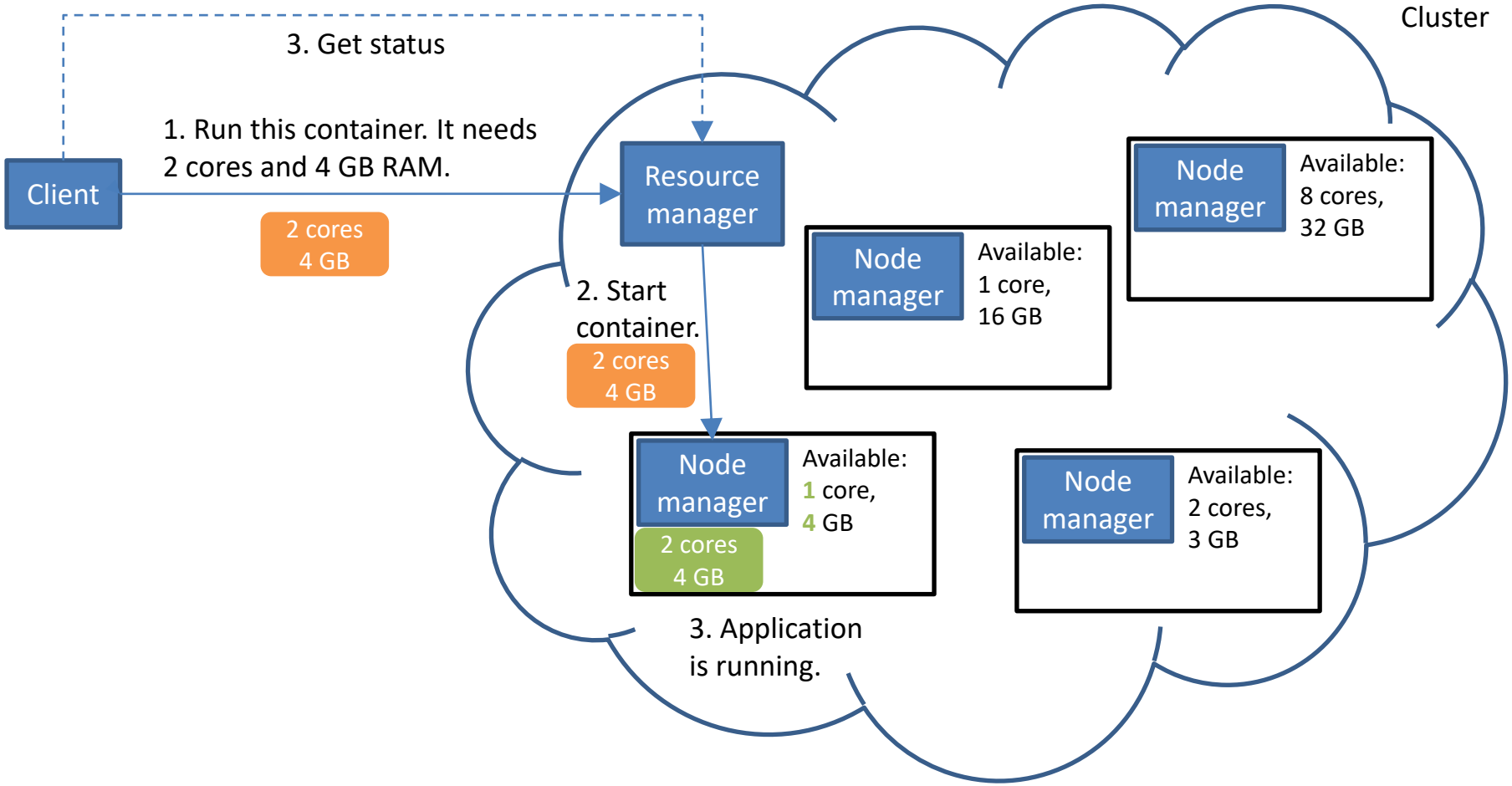
Who Will Run the Application?

- The resource manager communicates with the node managers running on the machines to keep an up-to-date view about available resources.
- Any machine with sufficient resources could run the application container. In practice, the scheduler will typically use a heuristic to make a choice. There are many reasonable options:
  - For even load distribution, select the least loaded machine or the machine with the most resources available.
  - To avoid small resource leftovers, assign the application to the machine with the "best fit." This means that the machine will have the least leftover resources.
- For our example, let's assume the latter strategy is used. The resource manager will communicate with the corresponding node manager to start the container.

Basic View of Cluster Resource Management (cont.)



Basic View of Cluster Resource Management (cont.)



Scheduling Policies

- The resource manager also decides what to do when there are *insufficient* resources. It collects requests in a queue, where they wait until resources become available from completed or terminated jobs.
- A scheduling policy decides which request will be next in line. Here are some examples:
  - First-in first-out (FIFO) is the simplest approach: requests are served in order of arrival.
  - FAIR: the next request served is for the user who has been waiting the longest.
  - Priority: requests with higher priority are served first.

How Many Resources to Request?

- The resource manager, together with the node managers, enforces resource consumption limits. This means that if an application exceeds its requested resources, it will be terminated. However, asking for too much makes it harder to find a machine with enough memory and CPUs.
- Hadoop MapReduce and Spark can automatically determine the number of cores needed for application master and worker processes.
- The user only needs to worry about the amount of memory. This is essentially the same problem as determining the heap size limit for a Java program.
  - A good programmer will analyze memory consumption of their program and then choose accordingly. In the worst case, one can apply trial-and-error: start with a "good guess," then increase container memory size if necessary.

Application Management

- A distributed data-processing job consists of many tasks that are running on different machines. These tasks must be coordinated, as we will discuss in a future module.
- Should the cluster resource manager do this?
  - Pro: We already have a centralized process that is aware of available resource, so let's use it. The old Hadoop 0.* and 1.* versions took this route.
  - Con: The resource manager would have to be aware of application semantics. For MapReduce, it would have to understand in what order Map or Reduce tasks can be scheduled and what to do when one fails. For Spark or a database service, there are other task types and different ways to react to failures.
- The current trend is to separate resource management from application management. This way a single generic resource manager like YARN can support diverse applications and services on the same cluster.

Spark Application Management

- To see how resource and application management interact, we take a closer look at the execution of a Spark job. You will become more familiar with terminology like job, task, and executor as we progress in the course. For now, be aware that a Spark job consists of many tasks, which can run independently on different machines.
- The Spark job is initiated by a client process running in a Java Virtual Machine (JVM). In cluster-deploy mode, the client first requests resources from the resource manager to start up the application master, including the driver program. There is exactly one master per Spark job.
  - Notice again that a single coordinator is used to achieve consensus between many actors, in this case the tasks of a job.
- The application master then requests resources for executors from the resource manager. Executors then communicate with the master to (1) receive Spark tasks, (2) inform the driver about their status, and (3) emit their output.
  - The cluster resource manager is not involved in the application-specific communication or data transfer. It only assigns the applications to resources as requested. Then the application master takes over. This way resource and application management are cleanly separated.

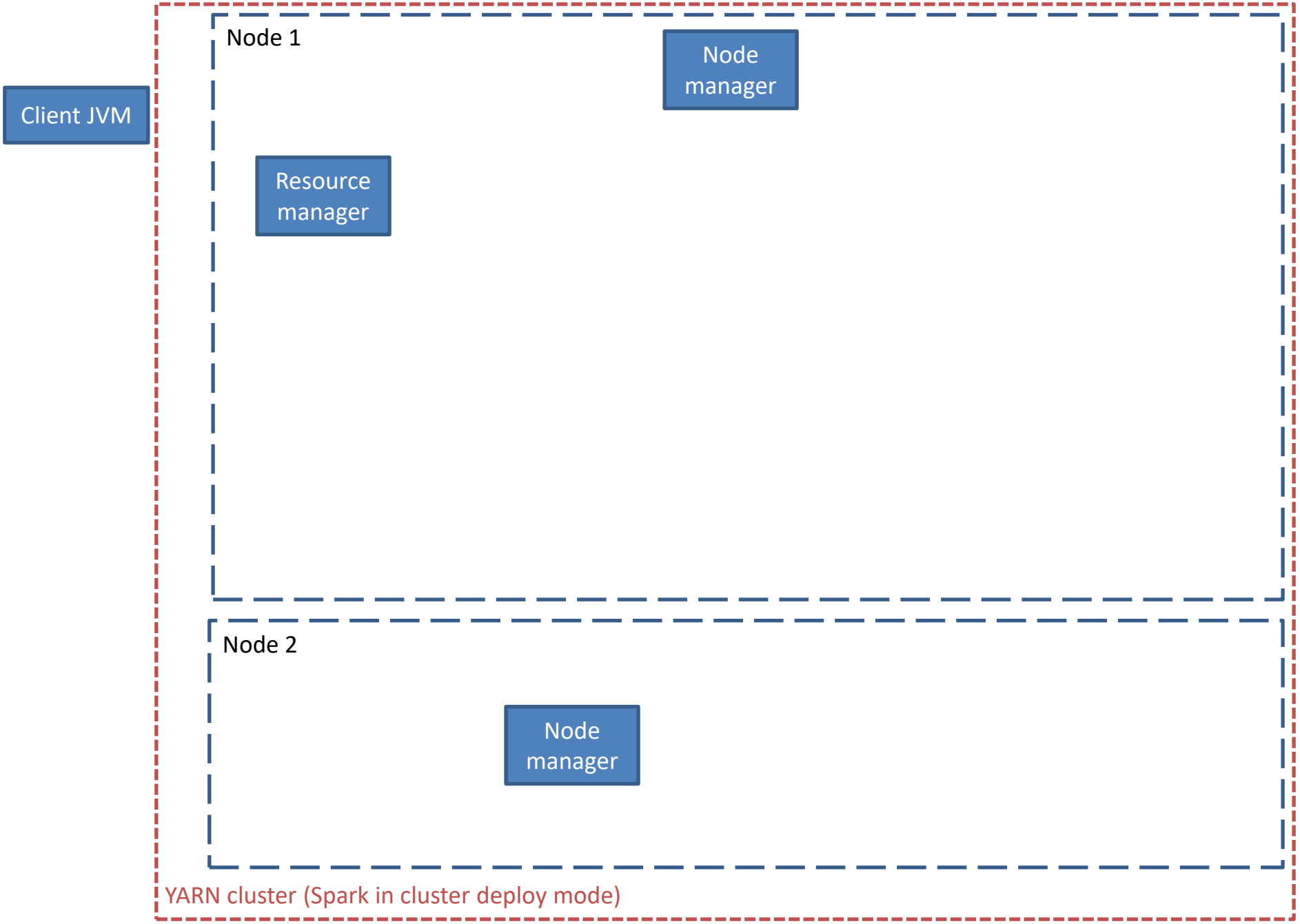


Illustration based on Zecevic/Bonaci book

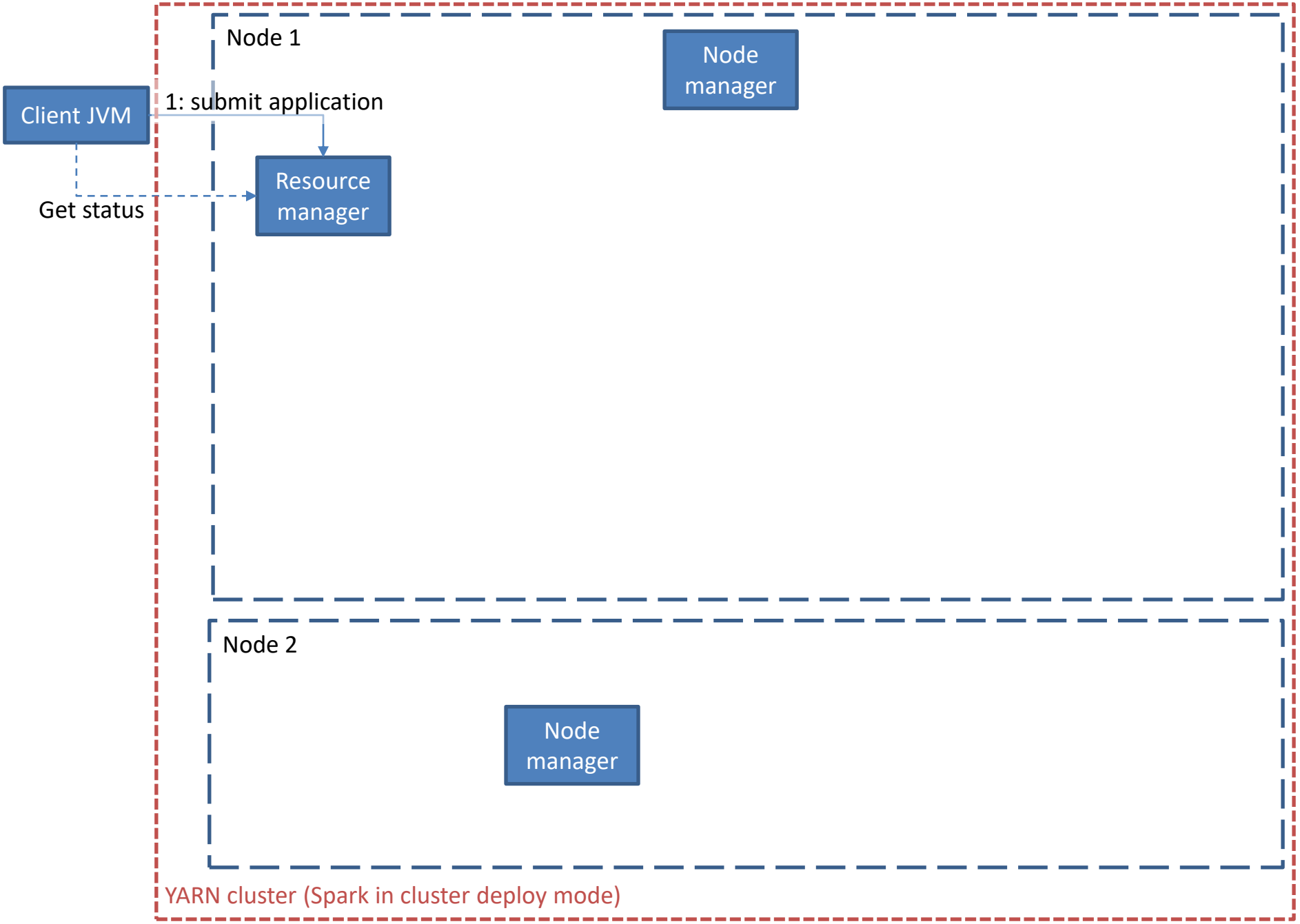


Illustration based on Zecevic/Bonaci book

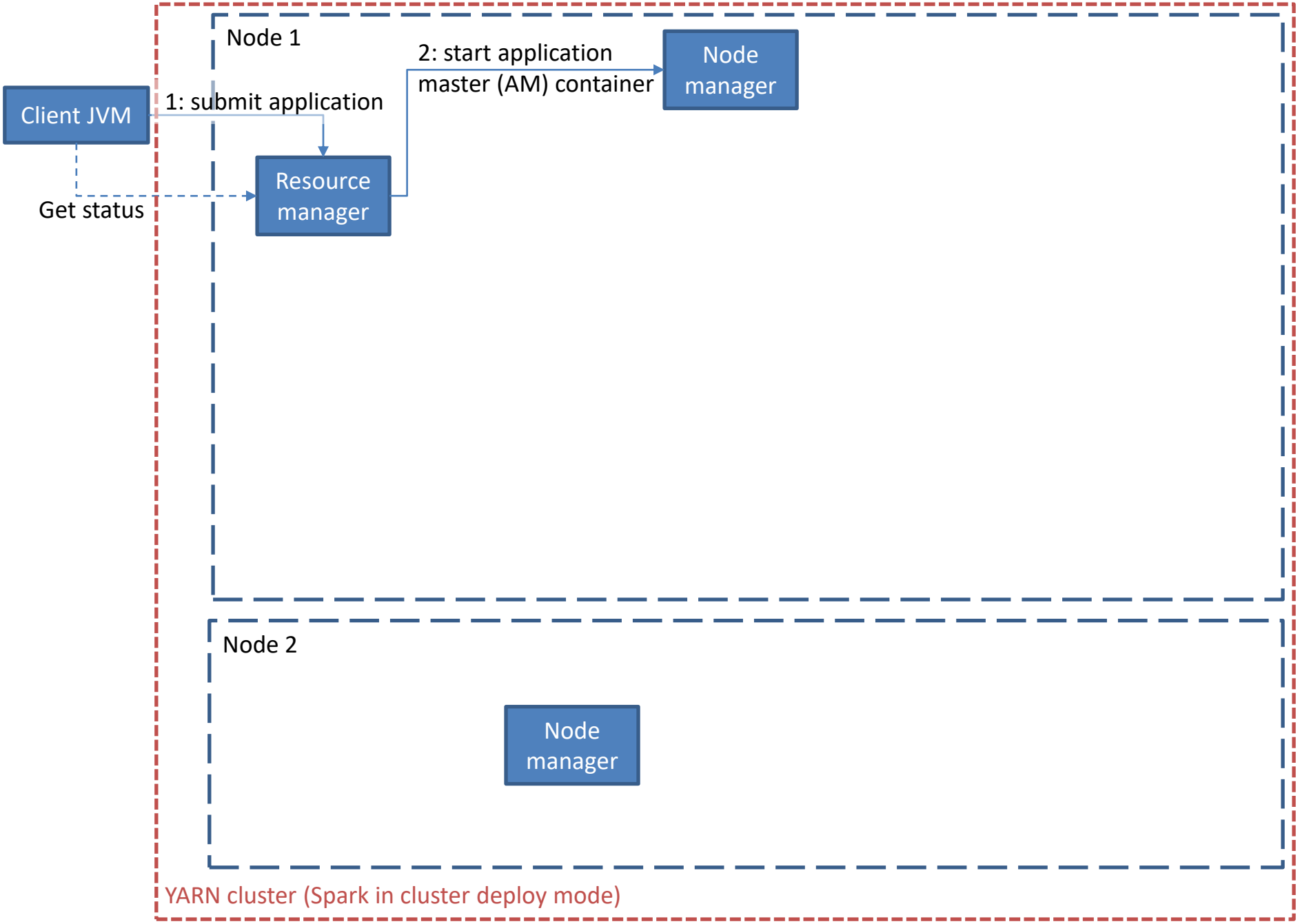


Illustration based on Zecevic/Bonaci book

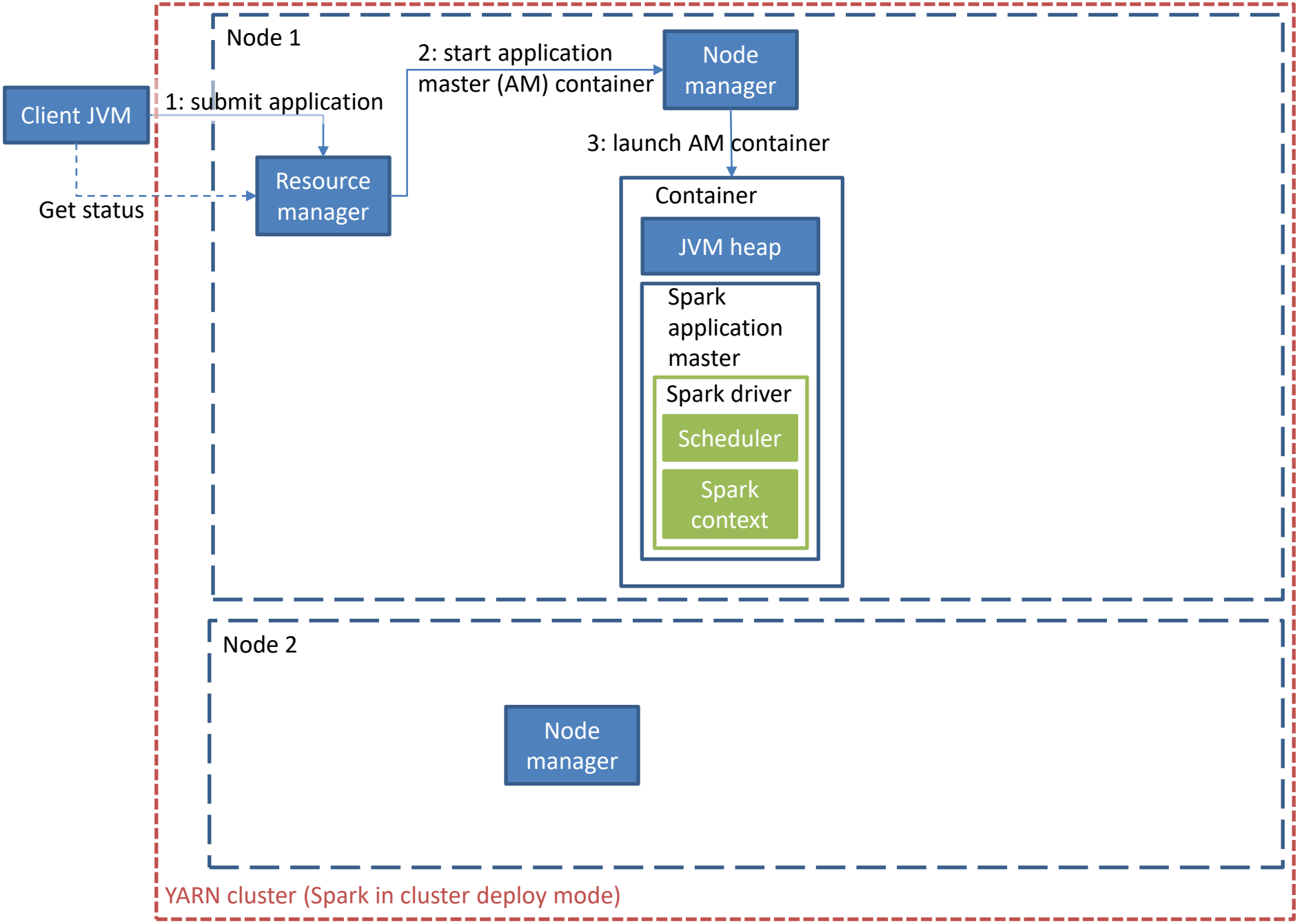


Illustration based on Zecevic/Bonaci book

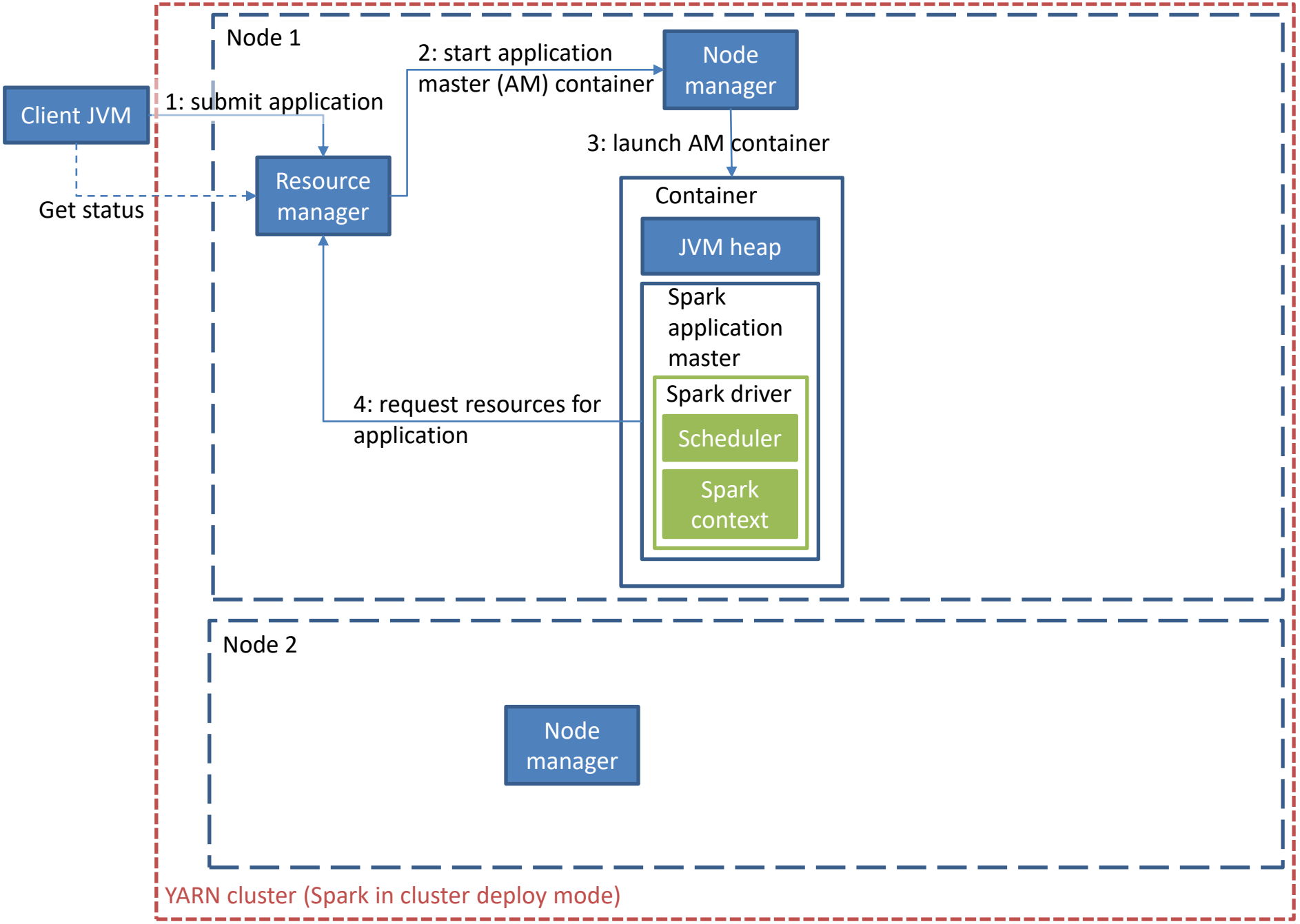


Illustration based on Zecevic/Bonaci book

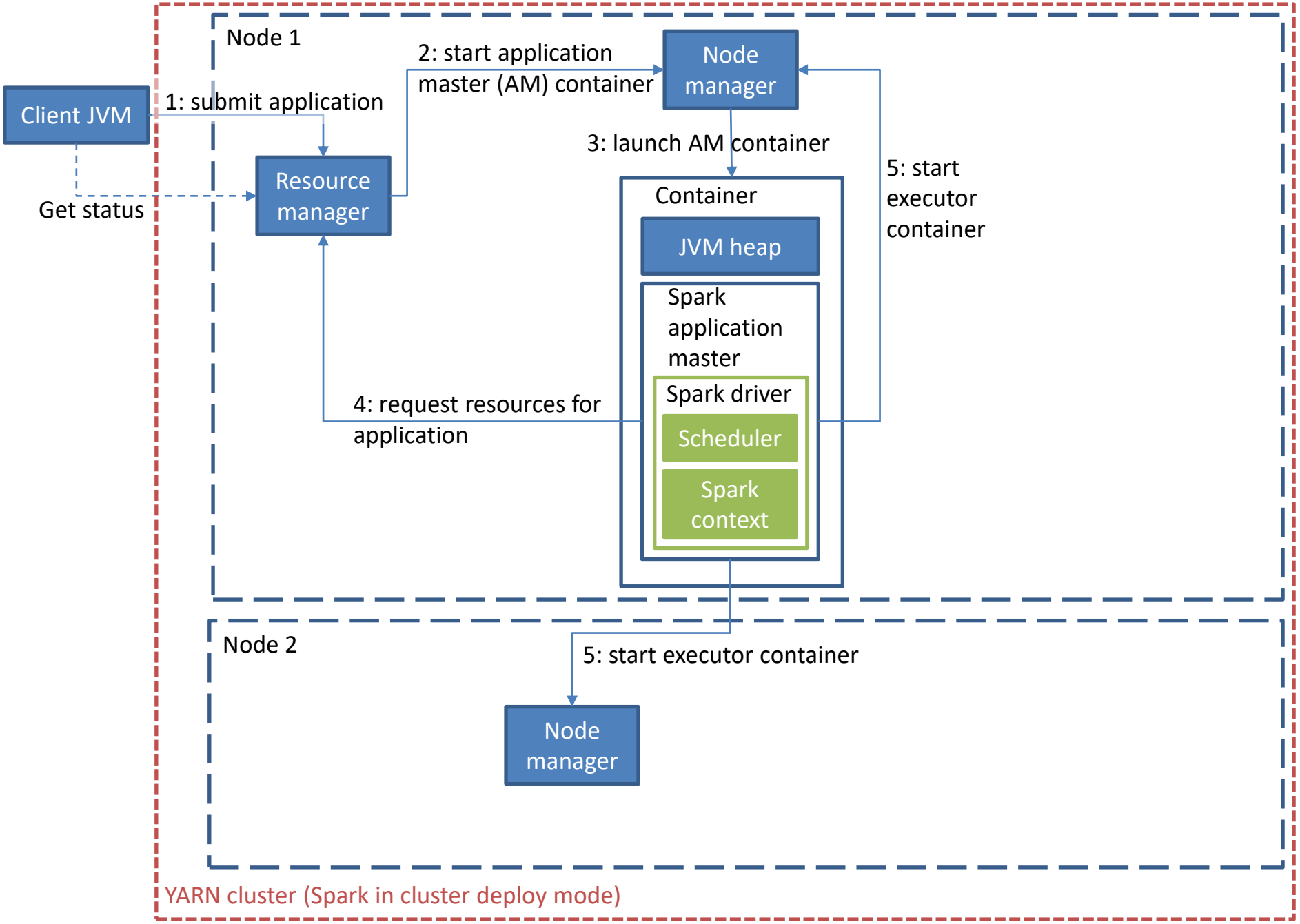


Illustration based on Zecevic/Bonaci book

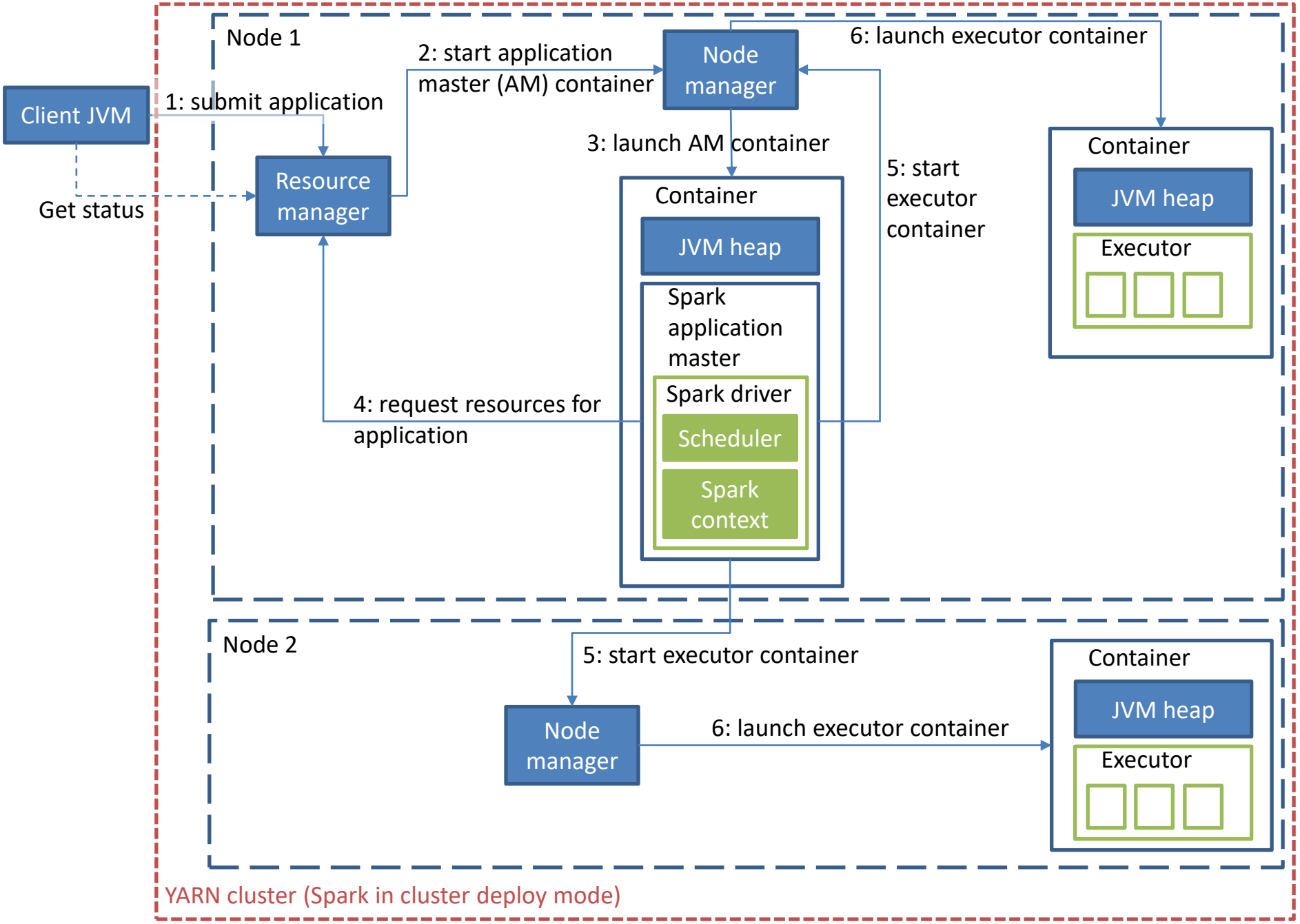


Illustration based on Zecevic/Bonaci book

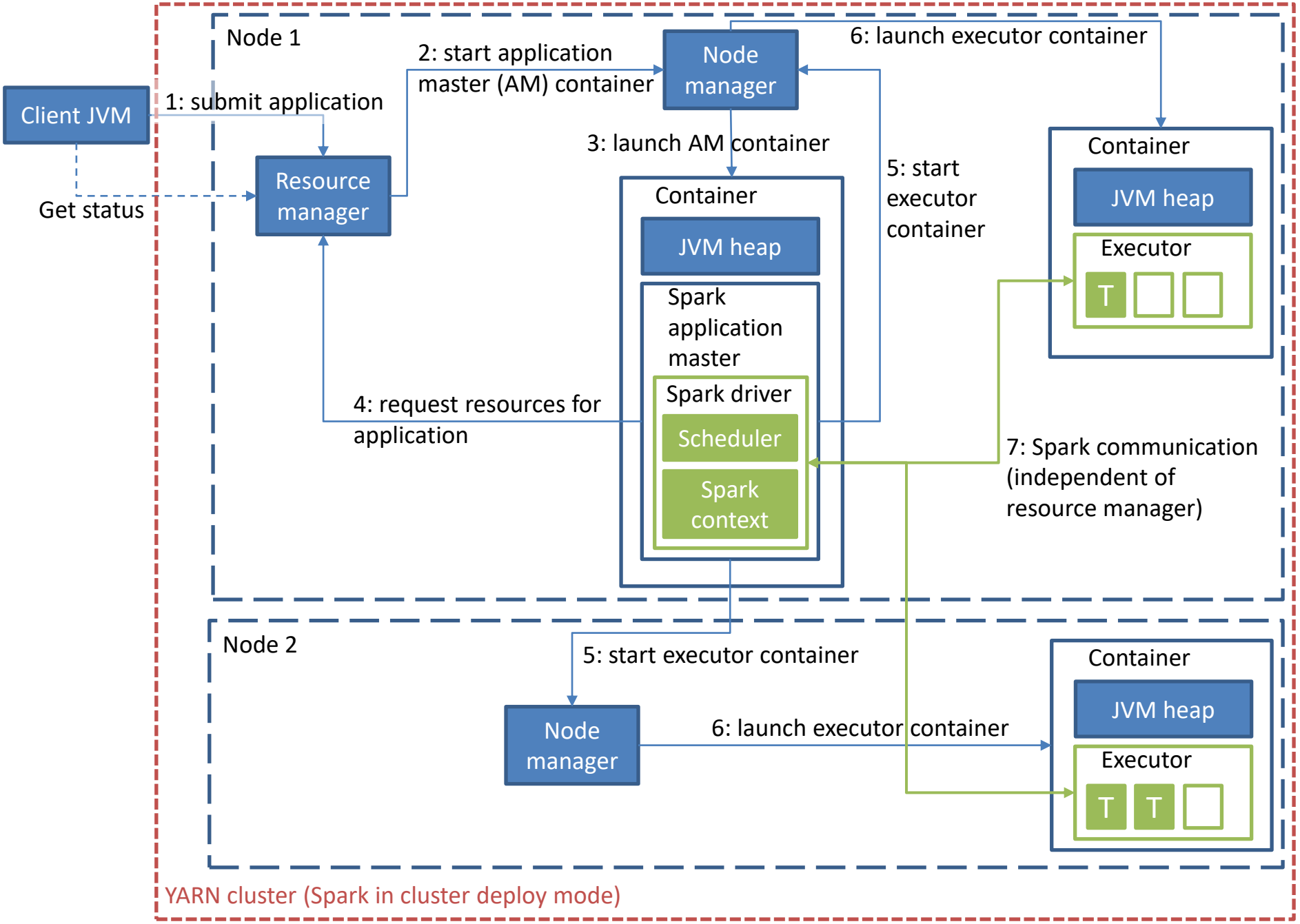


Illustration based on Zecevic/Bonaci book

References

- Petar Zecevic and Marko Bonaci: Spark in Action. Manning Publications, 2016