零基础学习Spark 1.x应用开发系列课程

Spark JobHistory与Spark on YARN

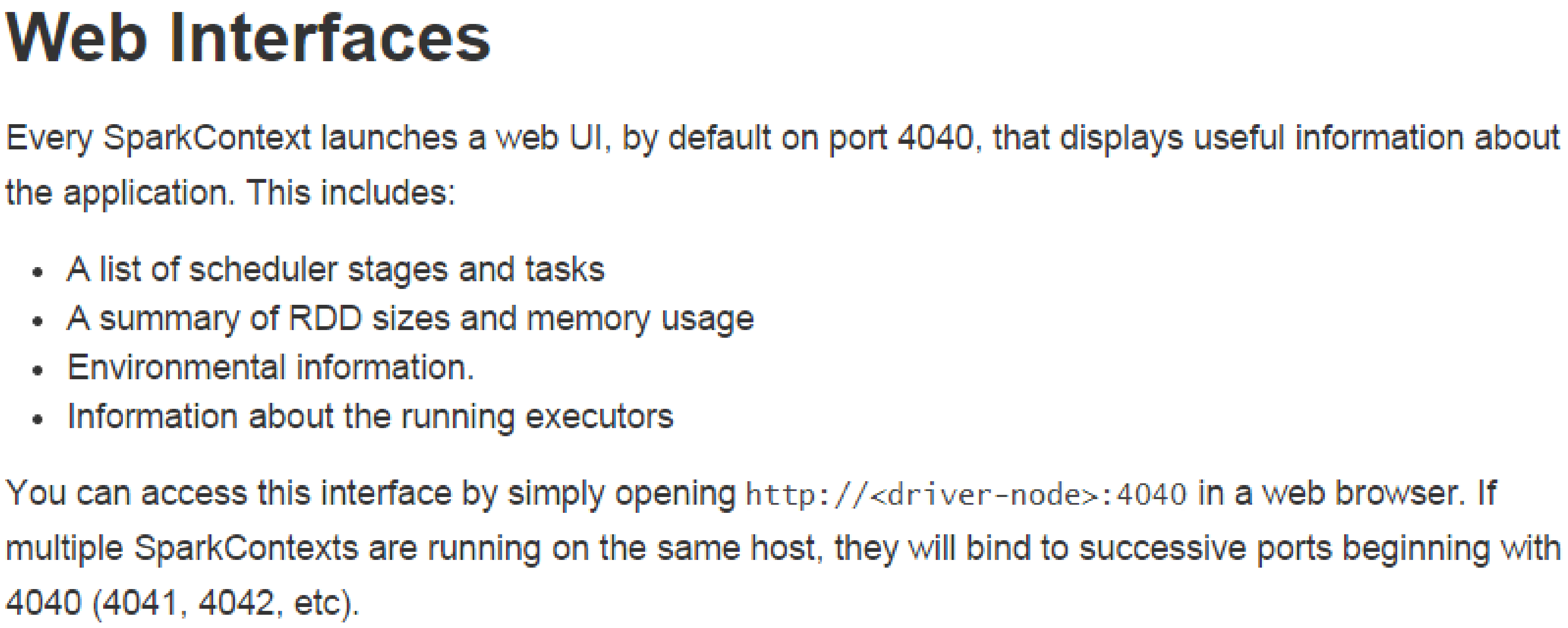
讲师-梦琪

版权声明

课程详情访问云帆大数据官方培训网站

http://www.cloudyhadoop.com

Spark Application Monitor



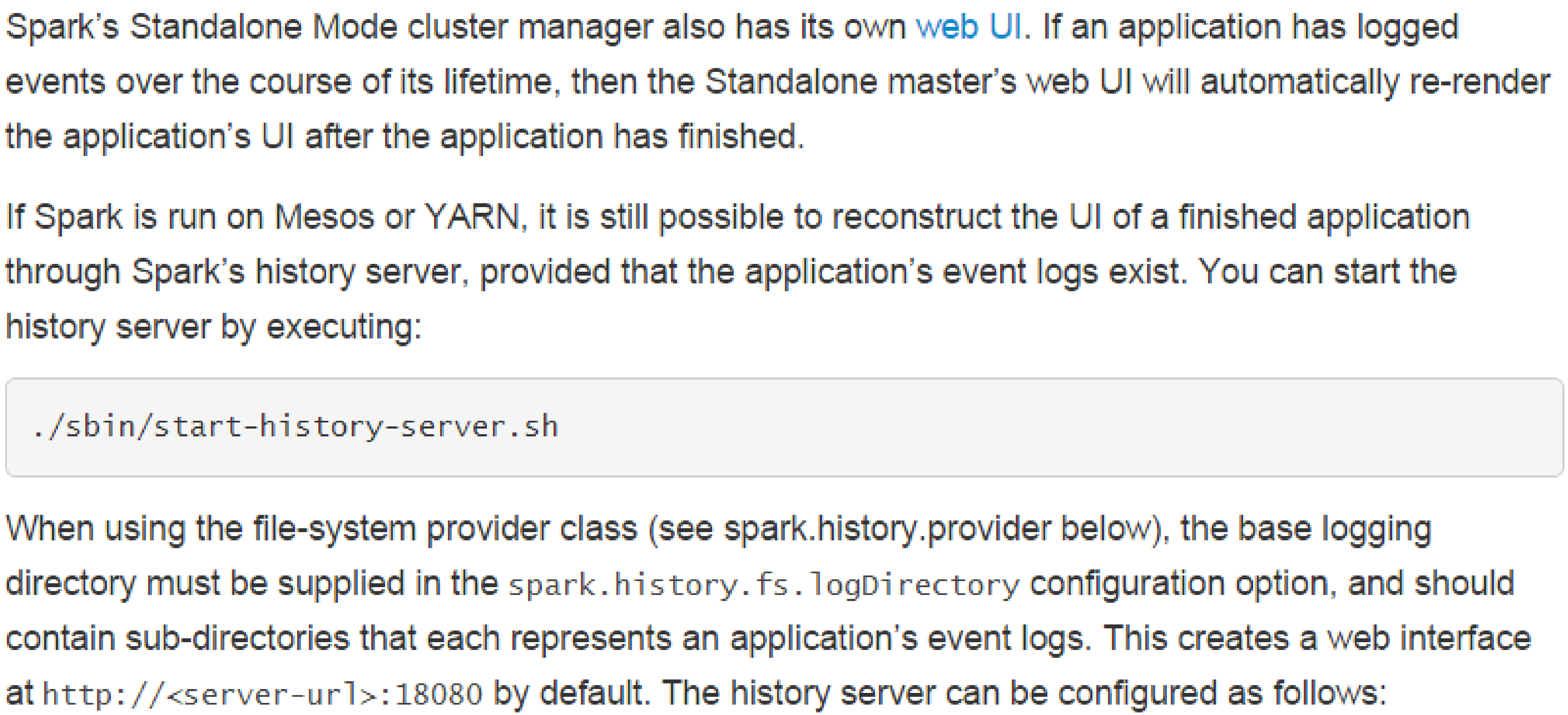
Web Interfaces

Every SparkContext launches a web UI, by default on port 4040, that displays useful information about the application. This includes:

- A list of scheduler stages and tasks
- A summary of RDD sizes and memory usage
- Environmental information.
- Information about the running executors

You can access this interface by simply opening `http://<driver-node>:4040` in a web browser. If multiple SparkContexts are running on the same host, they will bind to successive ports beginning with 4040 (4041, 4042, etc).

Spark Application Monitor



Spark's Standalone Mode cluster manager also has its own web UI. If an application has logged events over the course of its lifetime, then the Standalone master's web UI will automatically re-render the application's UI after the application has finished.

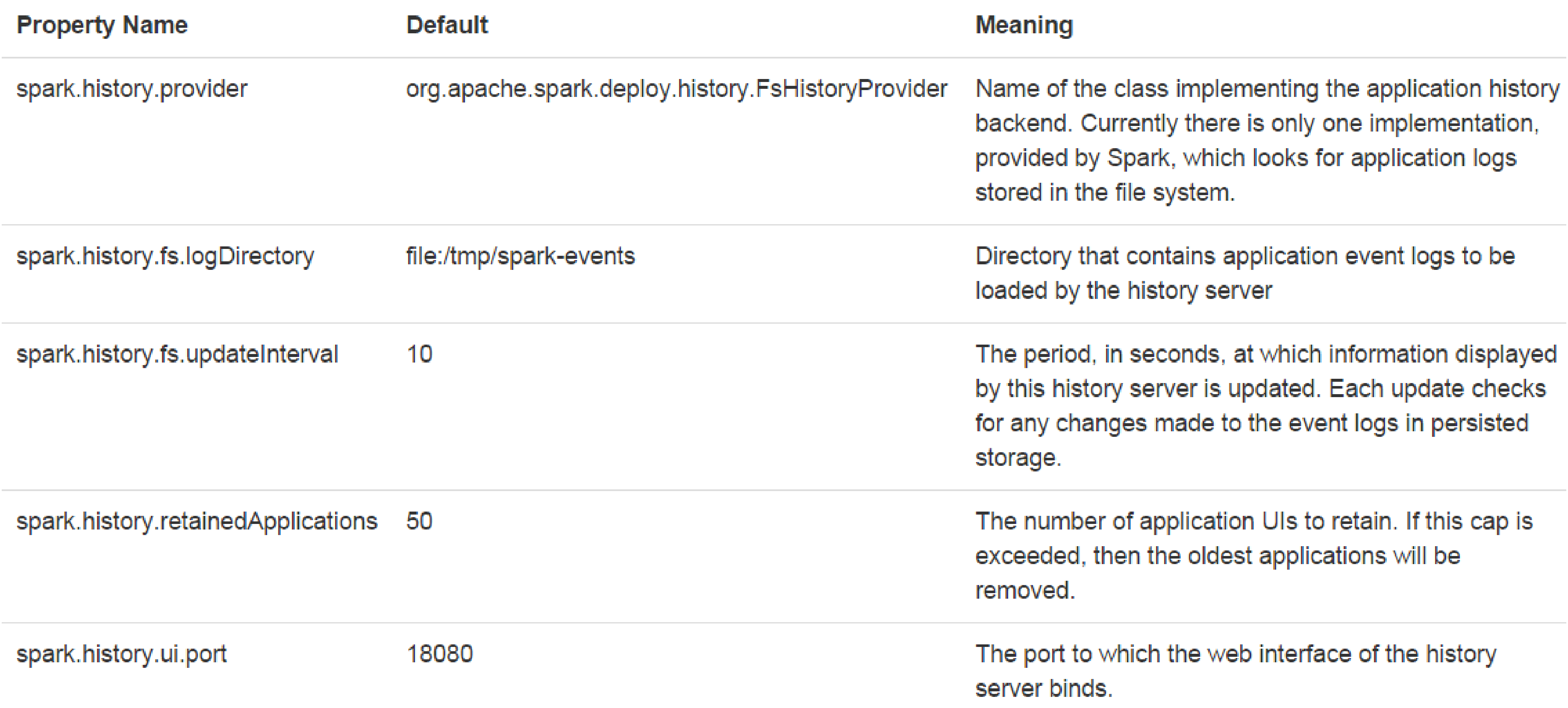
If Spark is run on Mesos or YARN, it is still possible to reconstruct the UI of a finished application through Spark's history server, provided that the application's event logs exist. You can start the history server by executing:

```
./sbin/start-history-server.sh
```

When using the file-system provider class (see spark.history.provider below), the base logging directory must be supplied in the `spark.history.fs.logDirectory` configuration option, and should contain sub-directories that each represents an application's event logs. This creates a web interface at `http://<server-url>:18080` by default. The history server can be configured as follows:

服务器端配置

配置在spark-env.sh中的SPARK_HISTORY_OPTS

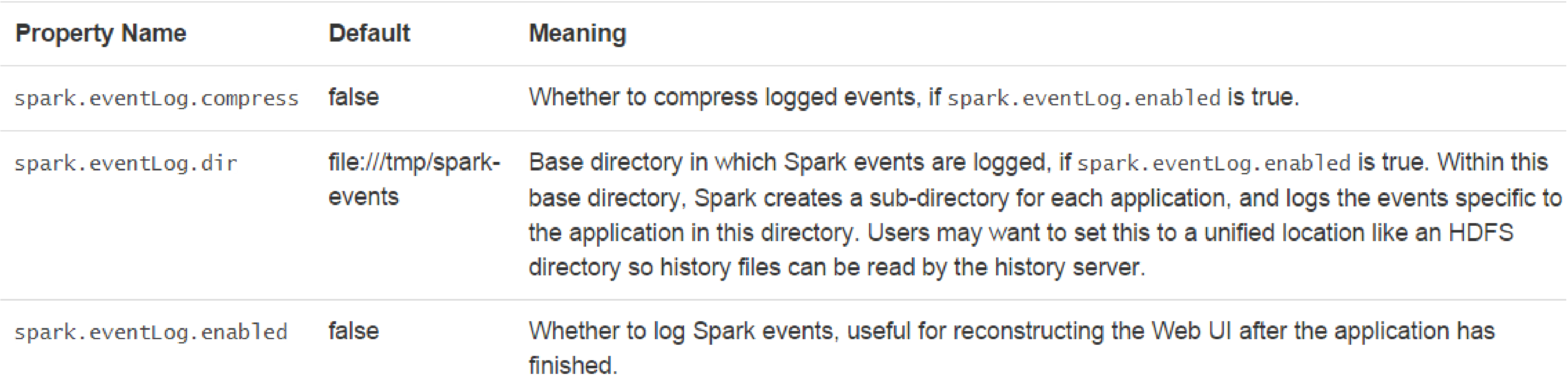


Property Name	Default	Meaning
spark.history.provider	org.apache.spark.deploy.history.FsHistoryProvider	Name of the class implementing the application history backend. Currently there is only one implementation, provided by Spark, which looks for application logs stored in the file system.
spark.history.fs.logDirectory	file:/tmp/spark-events	Directory that contains application event logs to be loaded by the history server
spark.history.fs.updateInterval	10	The period, in seconds, at which information displayed by this history server is updated. Each update checks for any changes made to the event logs in persisted storage.
spark.history.retainedApplications	50	The number of application UIs to retain. If this cap is exceeded, then the oldest applications will be removed.
spark.history.ui.port	18080	The port to which the web interface of the history server binds.

http://spark.apache.org/docs/latest/monitoring.html

客户端配置

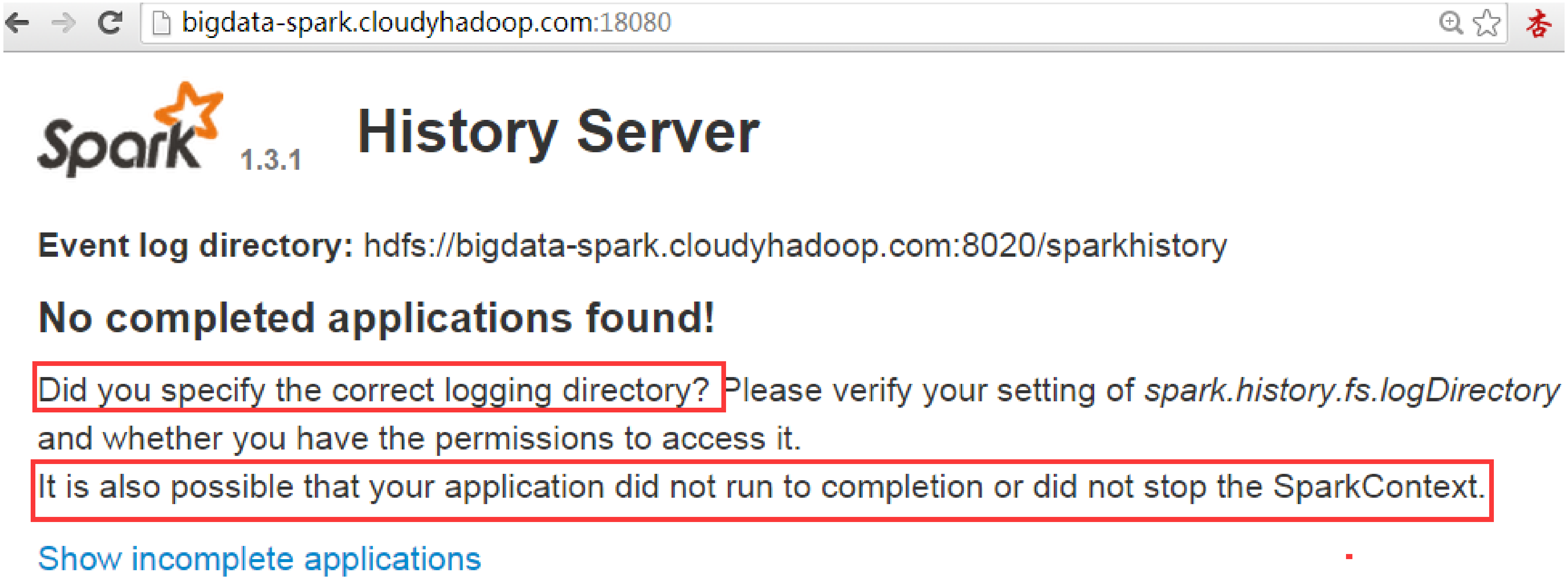
配置在spark-defaults.conf



Property Name	Default	Meaning
`spark.eventLog.compress`	false	Whether to compress logged events, if `spark.eventLog.enabled` is true.
`spark.eventLog.dir`	file:///tmp/spark-events	Base directory in which Spark events are logged, if `spark.eventLog.enabled` is true. Within this base directory, Spark creates a sub-directory for each application, and logs the events specific to the application in this directory. Users may want to set this to a unified location like an HDFS directory so history files can be read by the history server.
`spark.eventLog.enabled`	false	Whether to log Spark events, useful for reconstructing the Web UI after the application has finished.

http://spark.apache.org/docs/latest/configuration.html#spark-ui

Spark History Server



History Server

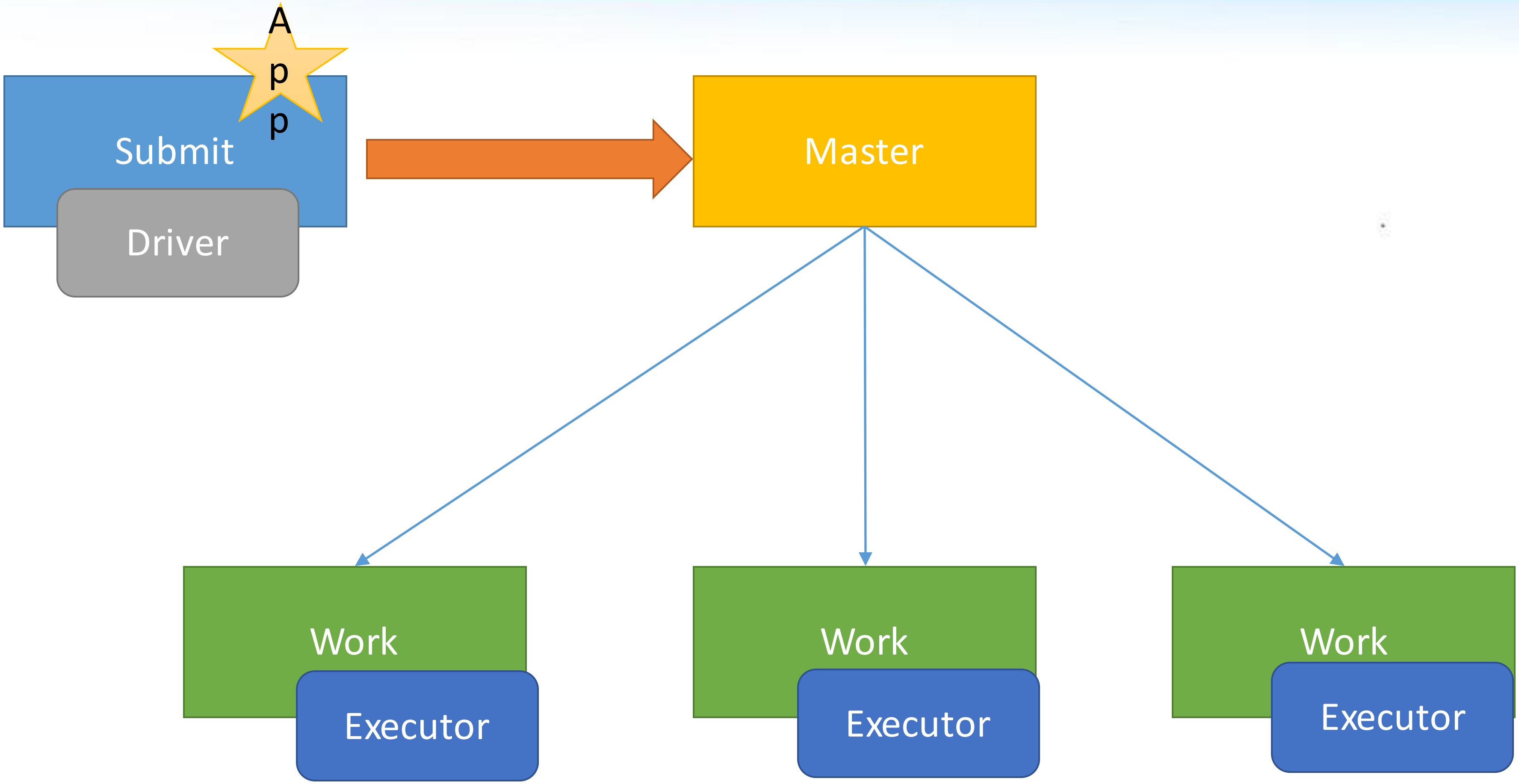
Event log directory: hdfs://bigdata-spark.cloudyhadoop.com:8020/sparkhistory

No completed applications found!

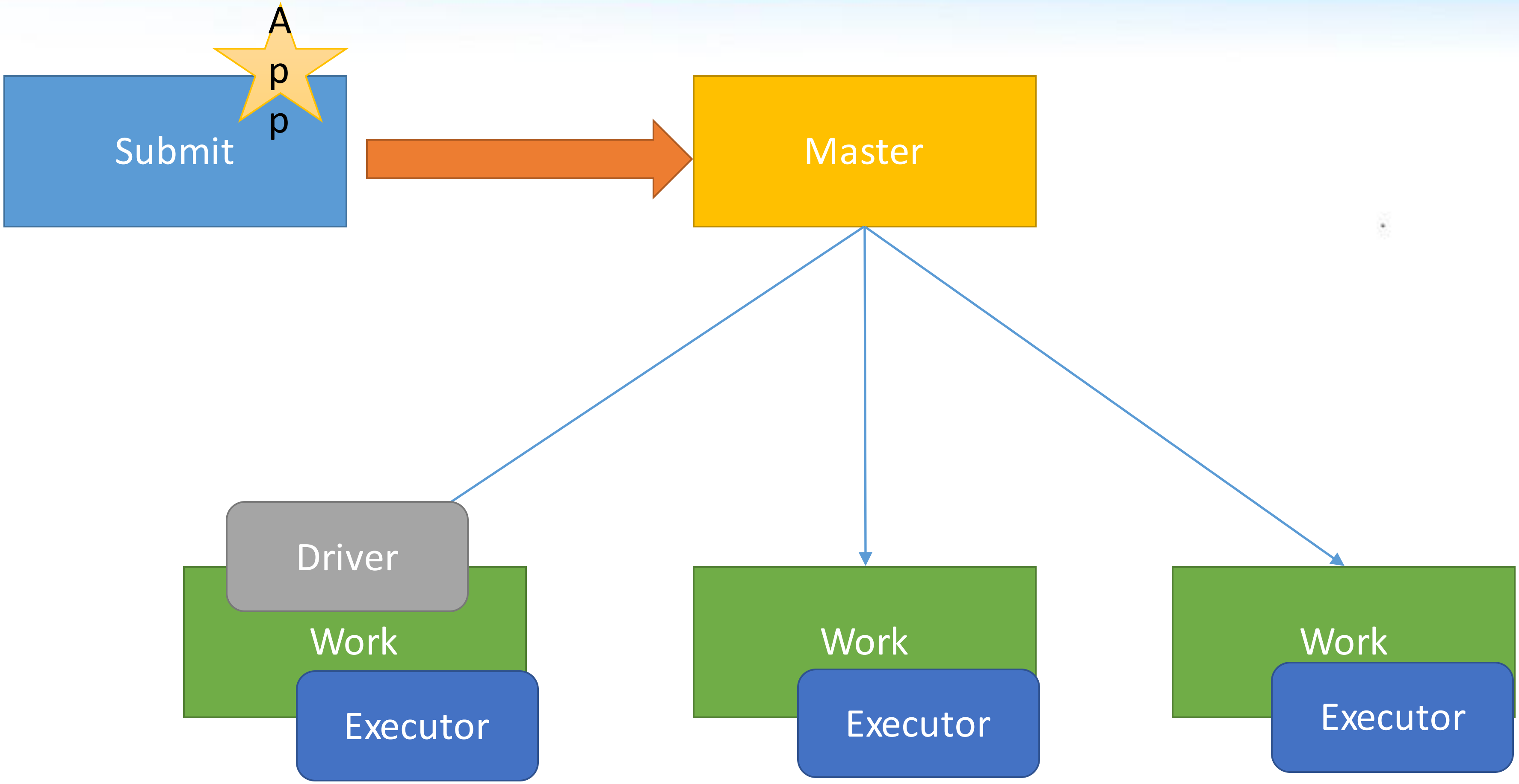
Did you specify the correct logging directory? Please verify your setting of *spark.history.fs.logDirectory* and whether you have the permissions to access it.
It is also possible that your application did not run to completion or did not stop the SparkContext.

Show incomplete applications

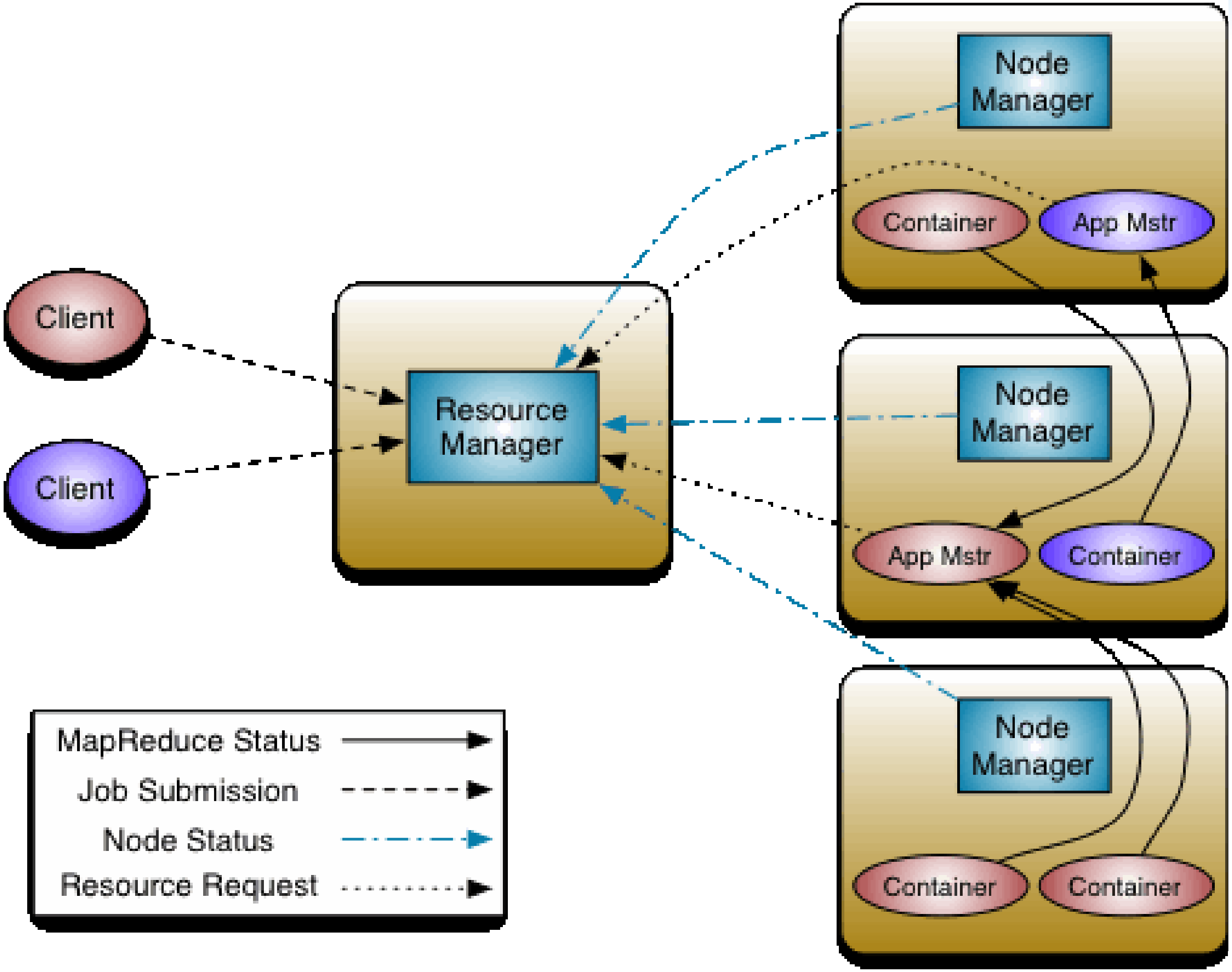
client


A
p
p
Submit
Driver
Master
Work
Executor
Work
Executor
Work
Executor

cluster


App
Submit
Master
Driver
Work
Executor
Work
Executor
Work
Executor

YARN 架构图

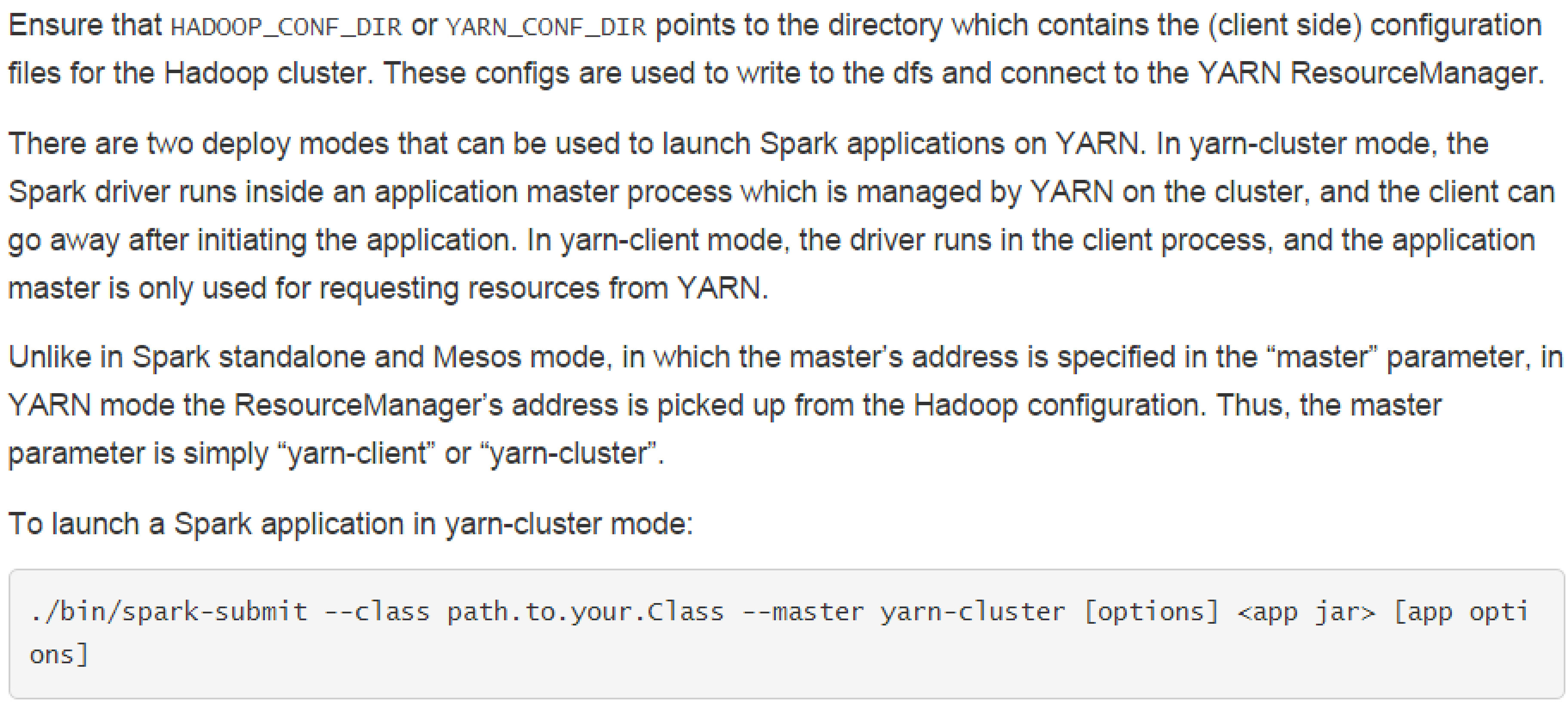

Client
Client
Resource Manager
Node Manager
Container
App Mstr
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

YARN 服务功能

- ResourceManager
  - 处理客户端请求
  - 启动/监控ApplicationMaster
  - 监控NodeManager
  - 资源分配与调度
- NodeManager
  - 单个节点上的资源管理
  - 处理来自ResourceManager的命令
  - 处理来自ApplicationMaster的命令

- ApplicationMaster
  - 数据切分
  - 为应用程序申请资源，并分配给内部任务
  - 任务监控与容错
- Container
  - 对任务运行环境的抽象，封装了CPU、内存等多维资源以及环境变量、启动命令等任务运行相关的信息

Launching Spark on YARN



Ensure that HADOOP_CONF_DIR or YARN_CONF_DIR points to the directory which contains the (client side) configuration files for the Hadoop cluster. These configs are used to write to the dfs and connect to the YARN ResourceManager.

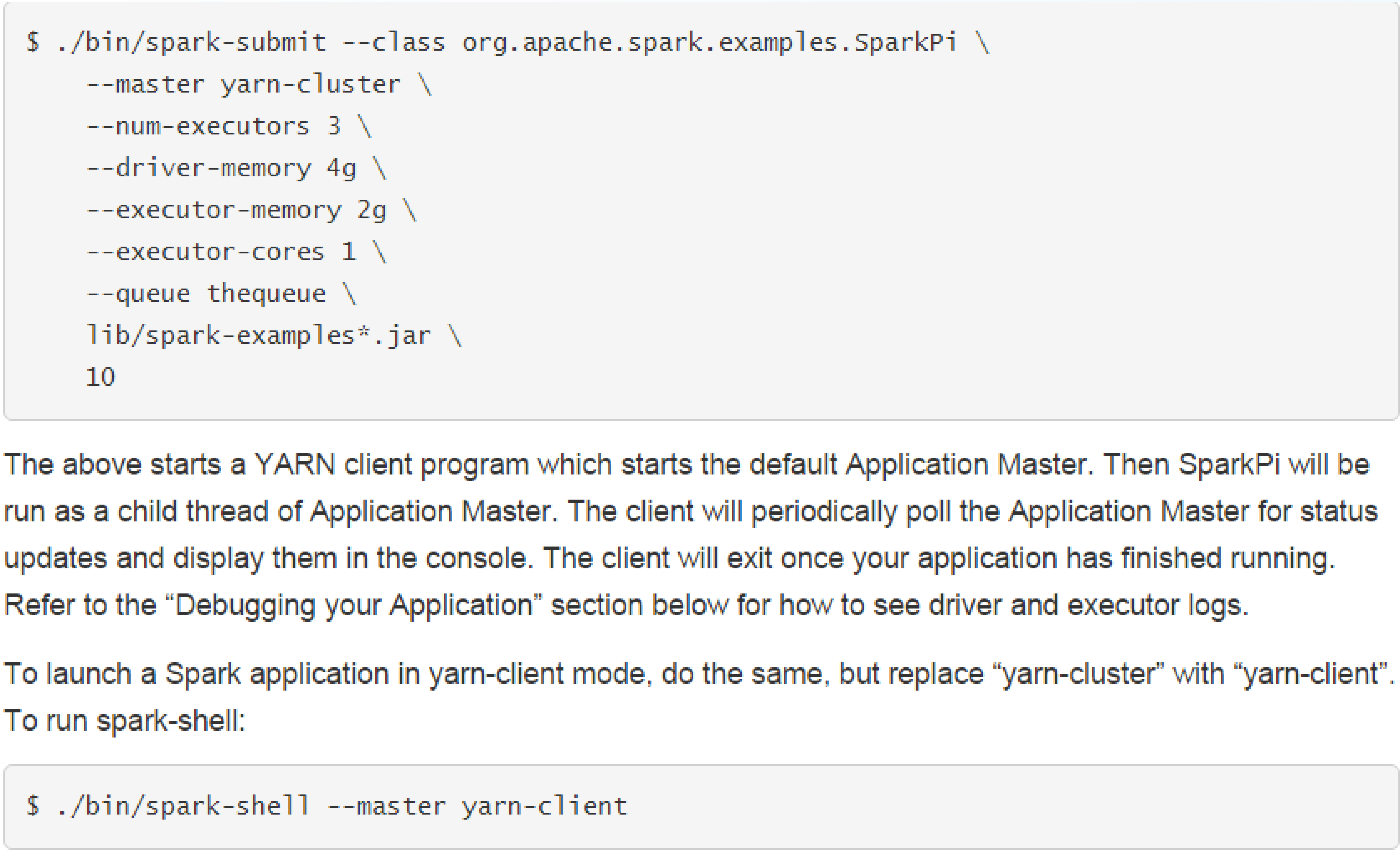
There are two deploy modes that can be used to launch Spark applications on YARN. In yarn-cluster mode, the Spark driver runs inside an application master process which is managed by YARN on the cluster, and the client can go away after initiating the application. In yarn-client mode, the driver runs in the client process, and the application master is only used for requesting resources from YARN.

Unlike in Spark standalone and Mesos mode, in which the master's address is specified in the “master” parameter, in YARN mode the ResourceManager's address is picked up from the Hadoop configuration. Thus, the master parameter is simply “yarn-client” or “yarn-cluster”.

To launch a Spark application in yarn-cluster mode:

```
./bin/spark-submit --class path.to.your.Class --master yarn-cluster [options] <app jar> [app opti
ons]
```

Launching Spark on YARN



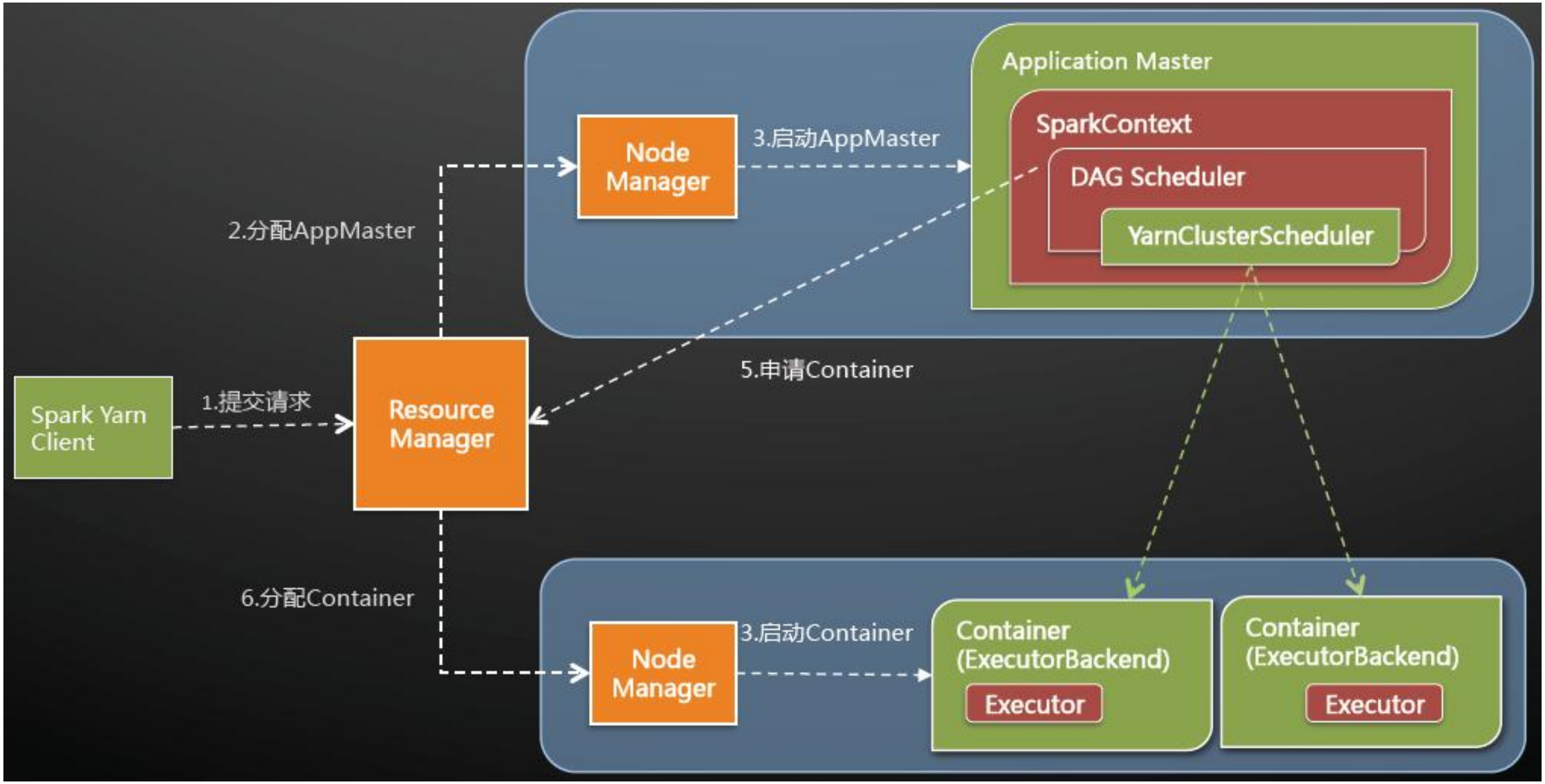
```
$ ./bin/spark-submit --class org.apache.spark.examples.SparkPi \
    --master yarn-cluster \
    --num-executors 3 \
    --driver-memory 4g \
    --executor-memory 2g \
    --executor-cores 1 \
    --queue thequeue \
    lib/spark-examples*.jar \
    10
```

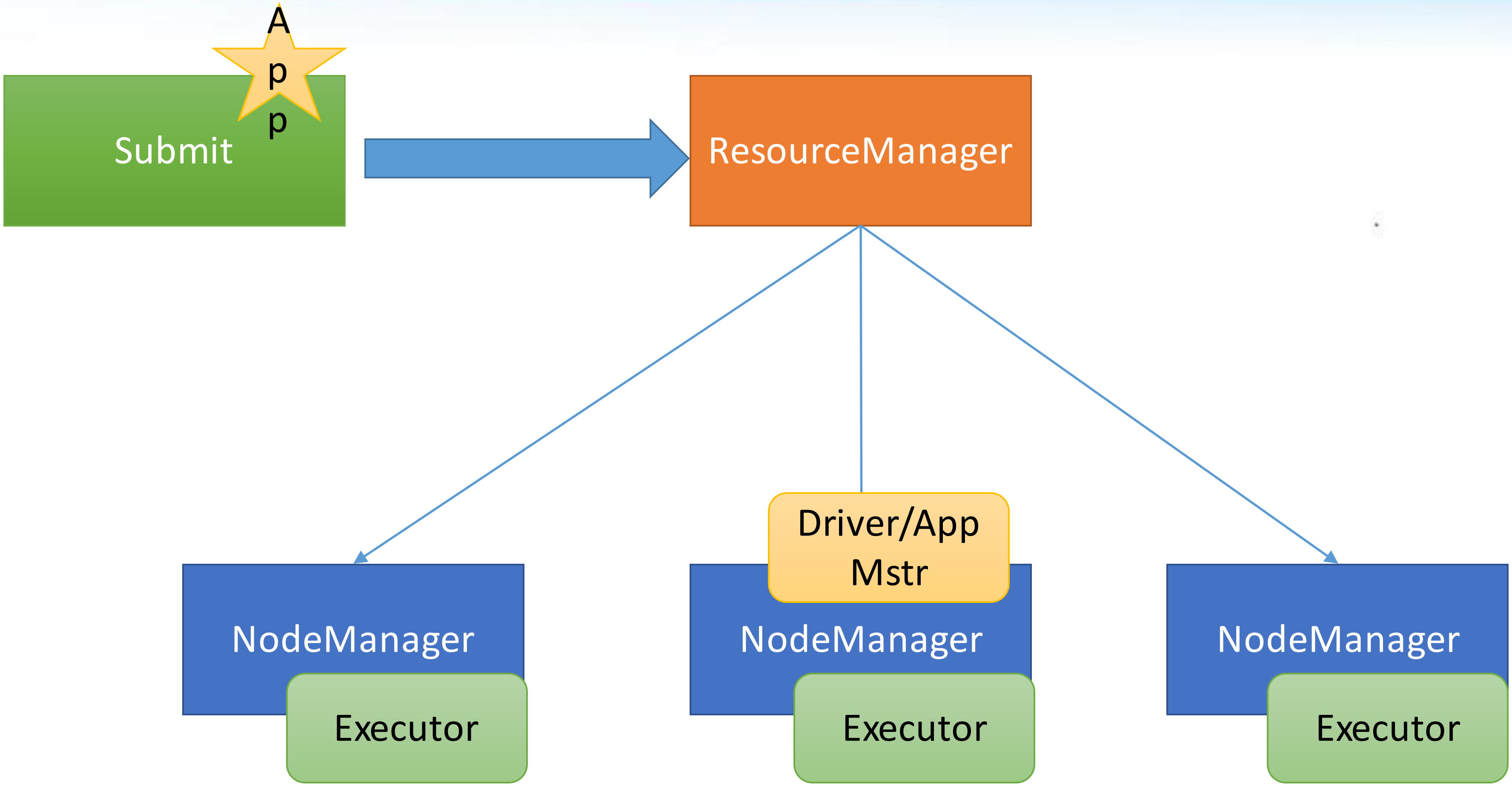
The above starts a YARN client program which starts the default Application Master. Then SparkPi will be run as a child thread of Application Master. The client will periodically poll the Application Master for status updates and display them in the console. The client will exit once your application has finished running. Refer to the “Debugging your Application” section below for how to see driver and executor logs.

To launch a Spark application in yarn-client mode, do the same, but replace “yarn-cluster” with “yarn-client”. To run spark-shell:

```
$ ./bin/spark-shell --master yarn-client
```

Spark on YARN




App
Submit
ResourceManager
Driver/App Mstr
NodeManager
Executor
NodeManager
Executor
NodeManager
Executor

云帆大数据网络课程

- 云帆大数据是国内首家坚持实时在线授课、提供高端开发课程网络培训机构。采用新兴的互联网教育模式，坚持实时在线授课模式，既继承传统教育的学习交流特点，又发挥互联网的无处不在的时空特性，将天南地北有志向的人才组织在一起学习交流，使原先孤立的个体学习，组合成有组织的学习探讨，并且把原先的学费用降低到十分之一左右，使更多的人能学习到最新的高端课程技术。云帆大数据同时是一个平台，如果你是一个学员，可以尽情的学习和交流；如果你是一个有梦想有才华的人，可以联系我们，给你提供一片驰骋的原野。
- 云帆大数据已推出国内首家《企业级Hadoop 2.x 应用开发课程》、《企业级Hadoop 2.x 项目实战课程》和《企业级Spark 1.x 应用开发课程》，更多其他详情，请登录我们的培训网站http://www.cloudyhadoop.com。

实时在线授课，专业课程辅导