Writing, Configuring, and Running Apache Spark Applications

The Spark Shell and Spark Applications

- **The Spark shell allows interactive exploration and manipulation of data**
  - REPL using Python or Scala
- **Spark applications run as independent programs**
  - For jobs such as ETL processing, streaming, and so on
  - Python, Scala, or Java

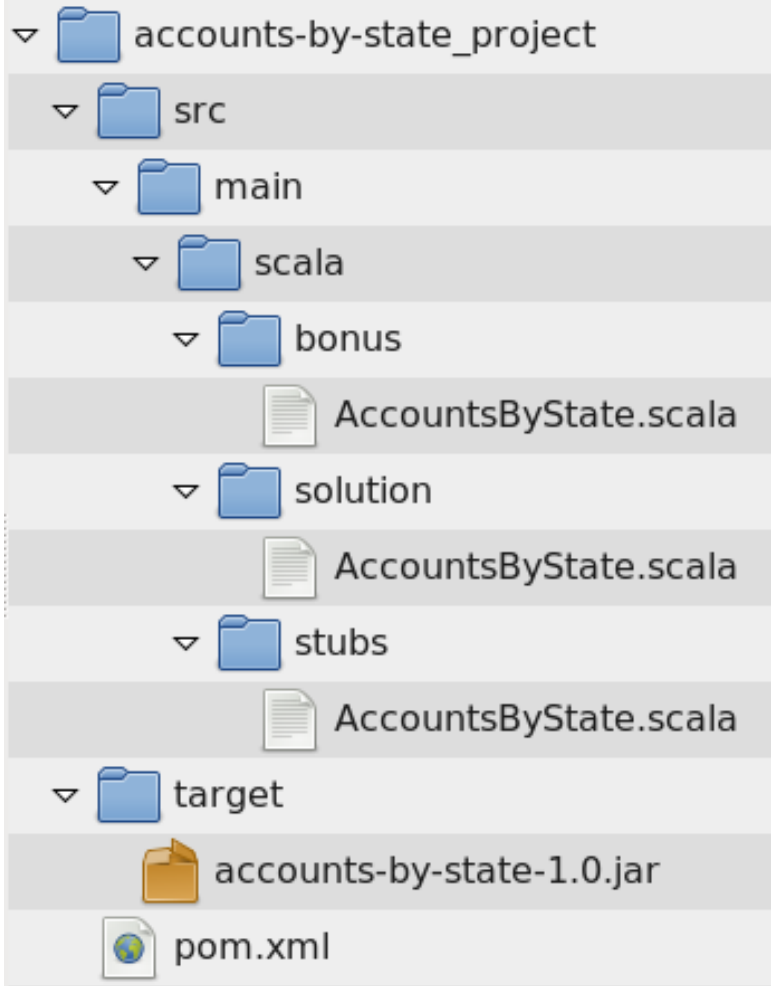
Python Example: Name List

```
import sys

from pyspark.sql import SparkSession

if__name__== "__main__":
  if len(sys.argv) < 3:
    print >> sys.stderr,
      "Usage: NameList.py <input-file> <output-file>"
    sys.exit()

  spark = SparkSession.builder.getOrCreate()
  spark.sparkContext.setLogLevel("WARN")

  peopleDF = spark.read.json(sys.argv[1])
  namesDF = peopleDF.select("firstName","lastName")
  namesDF.write.option("header","true").csv(sys.argv[2])

  spark.stop()
```

Language: *Python*

Building Scala Applications in the Hands-On Exercises

- **Basic Apache Maven projects are provided in the exercise directory**
  - `stubs`: starter Scala files—do exercises here
  - `solution`: exercise solutions
  - `bonus`: bonus solutions
- **Build command: `mvn package`**



Running a Spark Application

- **The easiest way to run a Spark application is to use the submit script**
  - Python

```
$ spark2-submit NameList.py people.json namelist/
```

  - Scala or Java

```
$ spark2-submit --class NameList MyJarFile.jar \
  people.json namelist/
```

Submit Script Options

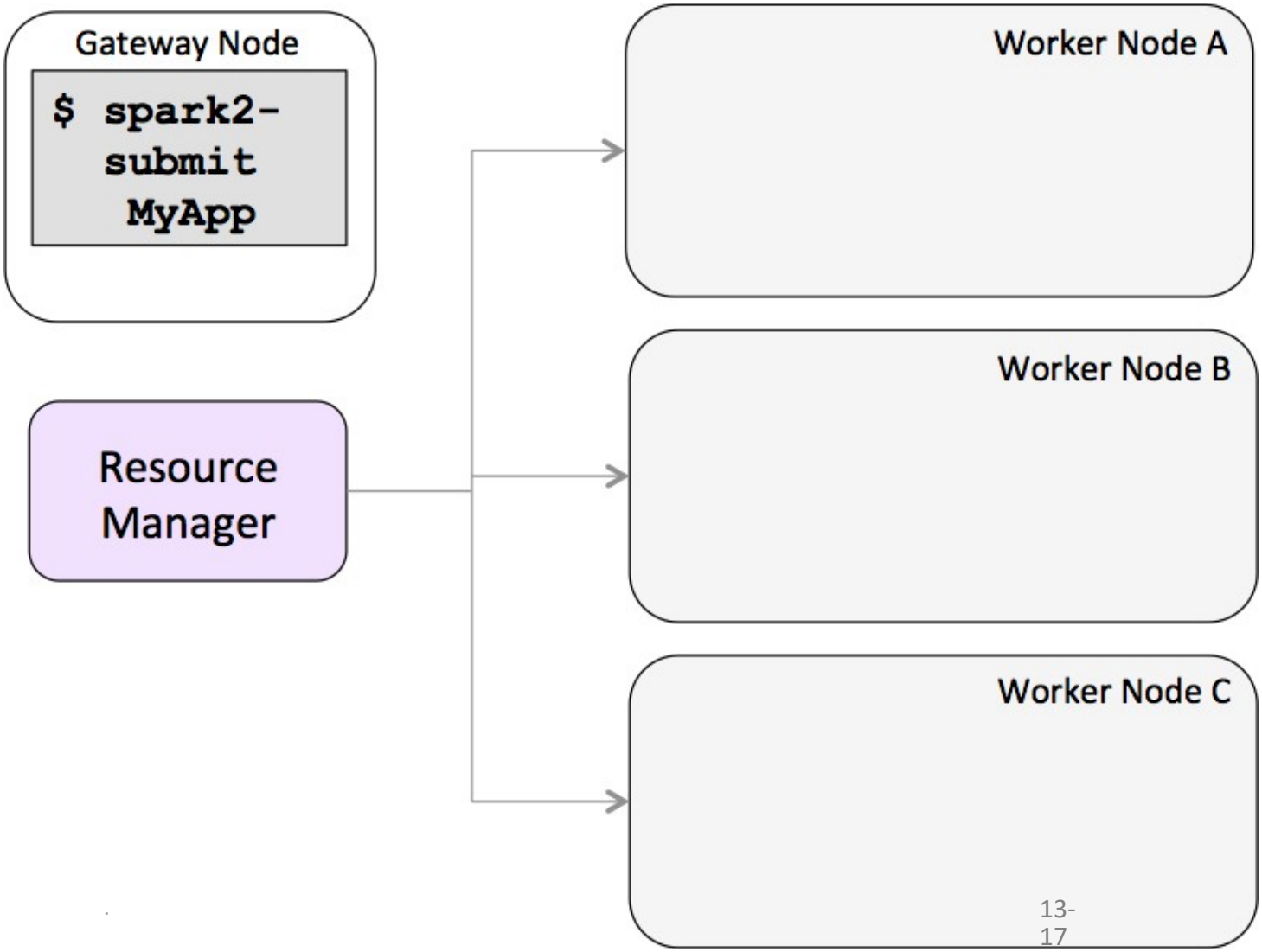
- **The Spark submit script provides many options to specify how the application should run**
  - Most are the same as for `pyspark2` and `spark2-shell`
- **General submit flags include**
  - `master`: `local`, `yarn`, or a Mesos or Spark Standalone cluster manager URI
  - `jars`: Additional JAR files (Scala and Java only)
  - `pyfiles`: Additional Python files (Python only)
  - `driver-java-options`: Parameters to pass to the driver JVM
- **YARN-specific flags include**
  - `num-executors`: Number of executors to start application with
  - `driver-cores`: Number cores to allocate for the Spark driver
  - `queue`: YARN queue to run in
- **Show all available options**
  - `help`

Application Deployment Mode

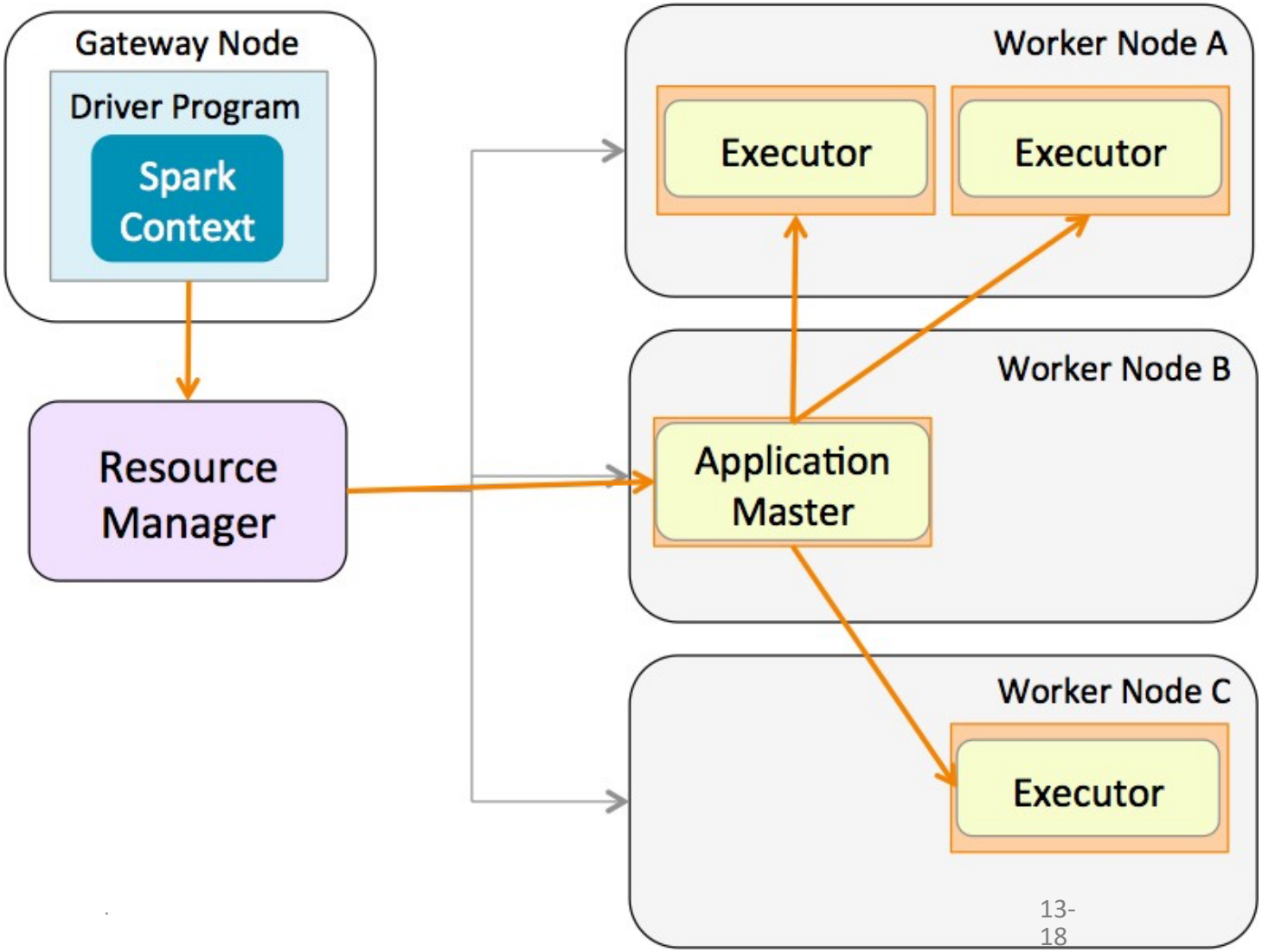
- **Spark applications can run**
  - Locally with one or more threads
  - On a cluster
    - In `client` mode (default), the driver runs locally on a gateway node
      - Requires direct communication between driver and cluster worker nodes
    - In `cluster` mode, the driver runs in the application master on the cluster
      - Common in production systems
- **Specify the deployment mode when submitting the application**

```
$ spark2-submit --master yarn --deploy-mode cluster \
  NameList.py people.json namelist/
```

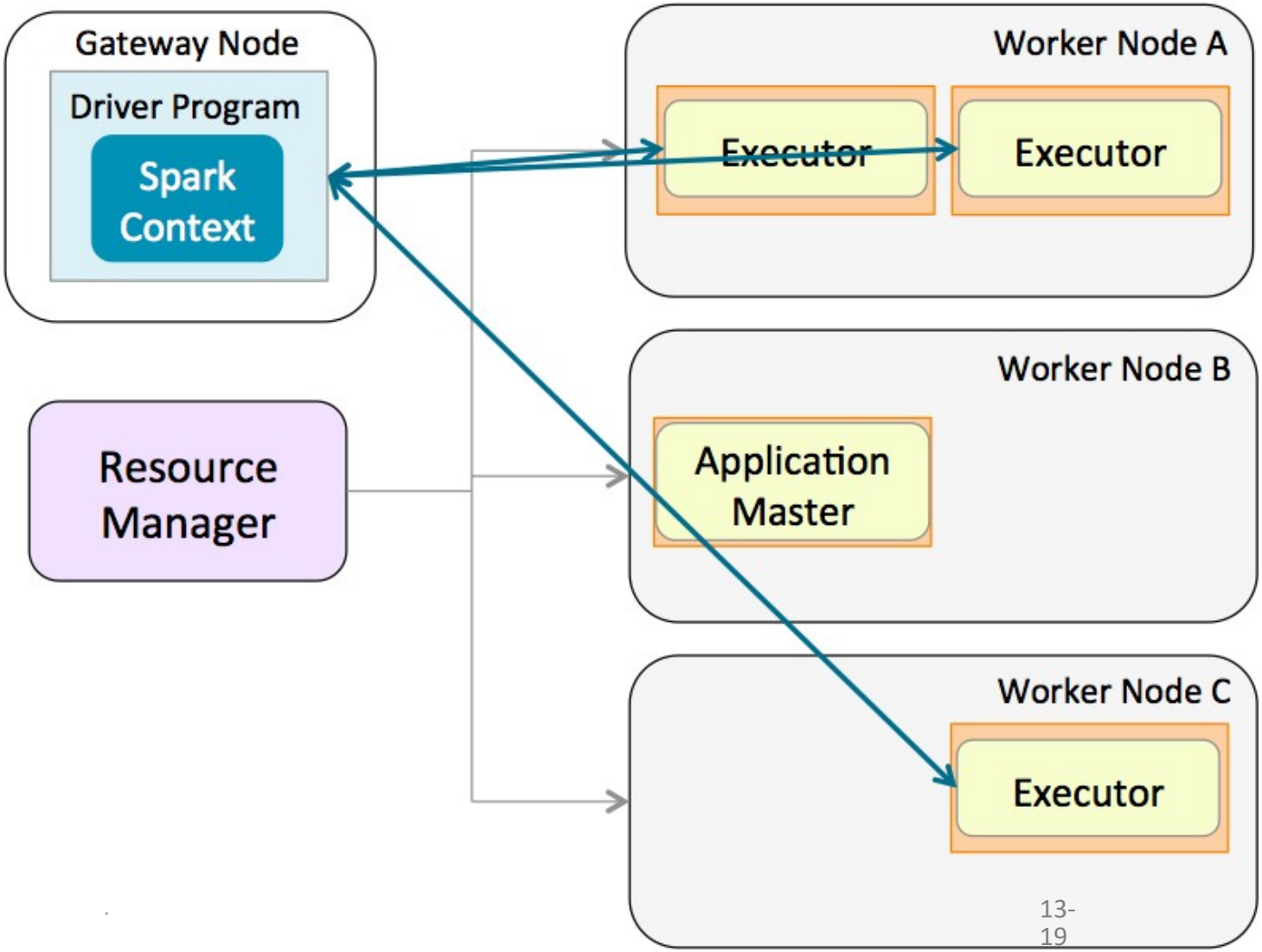
Spark Deployment Mode on YARN: Client Mode (1)



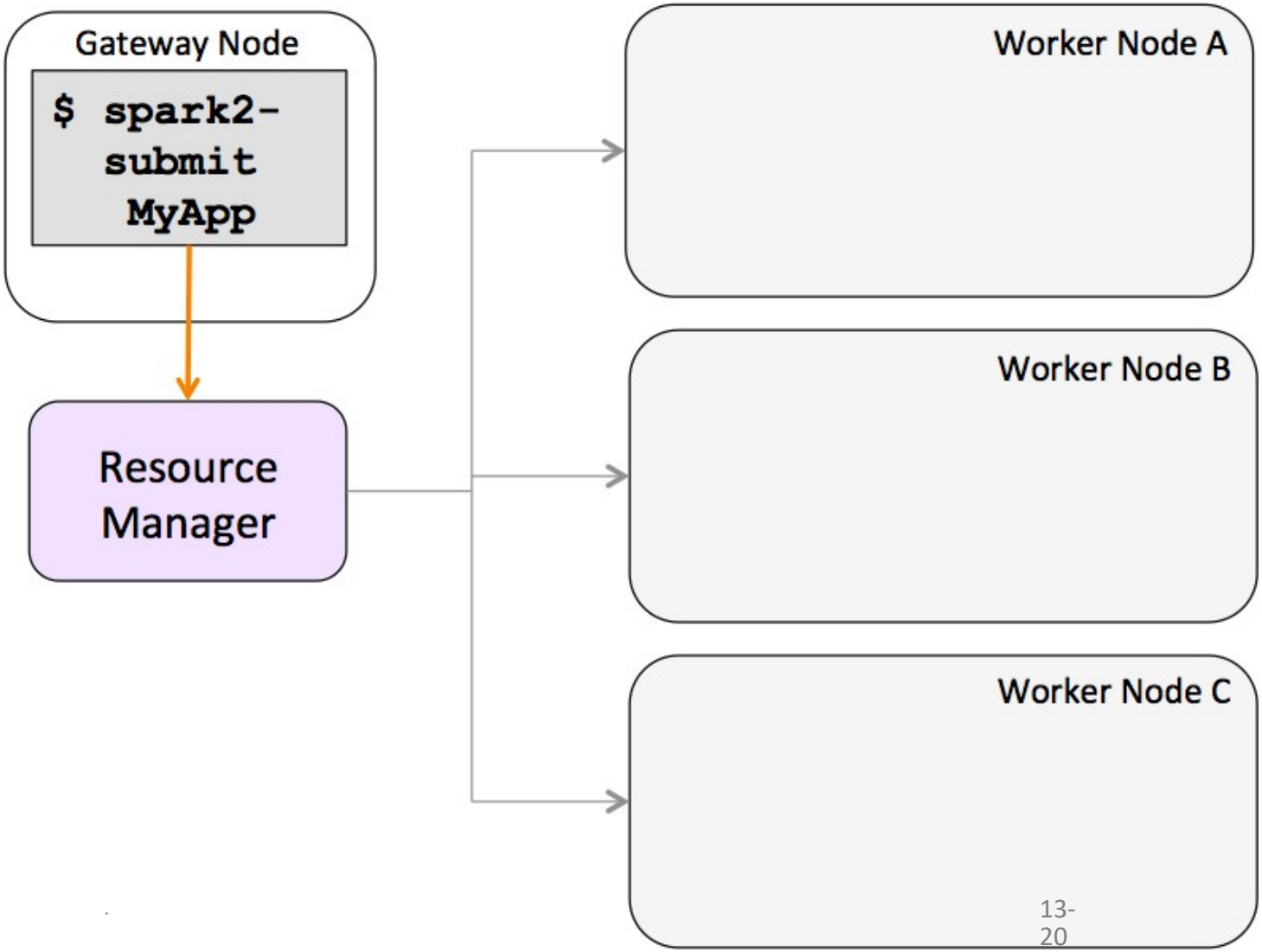
Spark Deployment Mode on YARN: Client Mode (2)



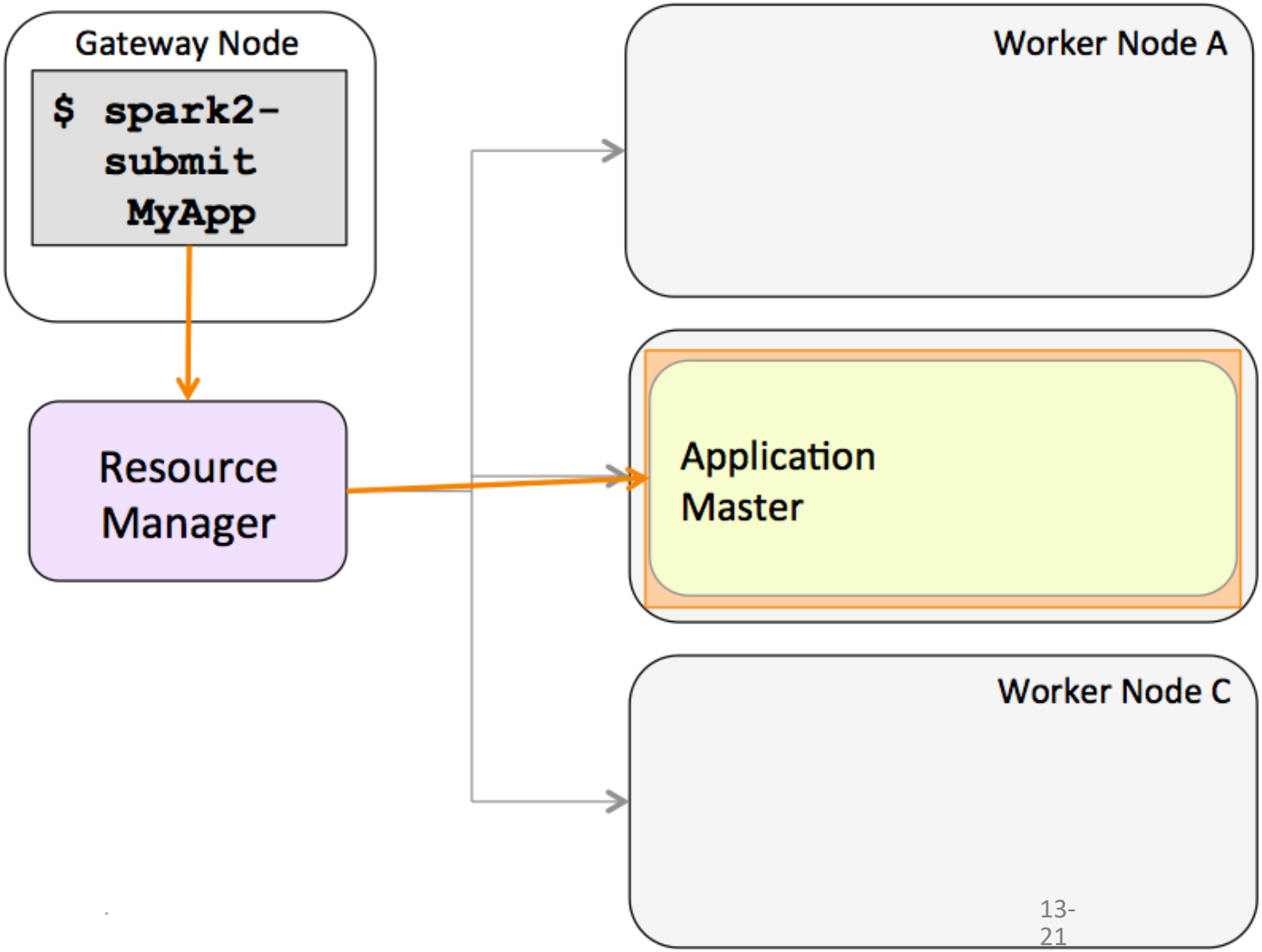
Spark Deployment Mode on YARN: Client Mode (3)



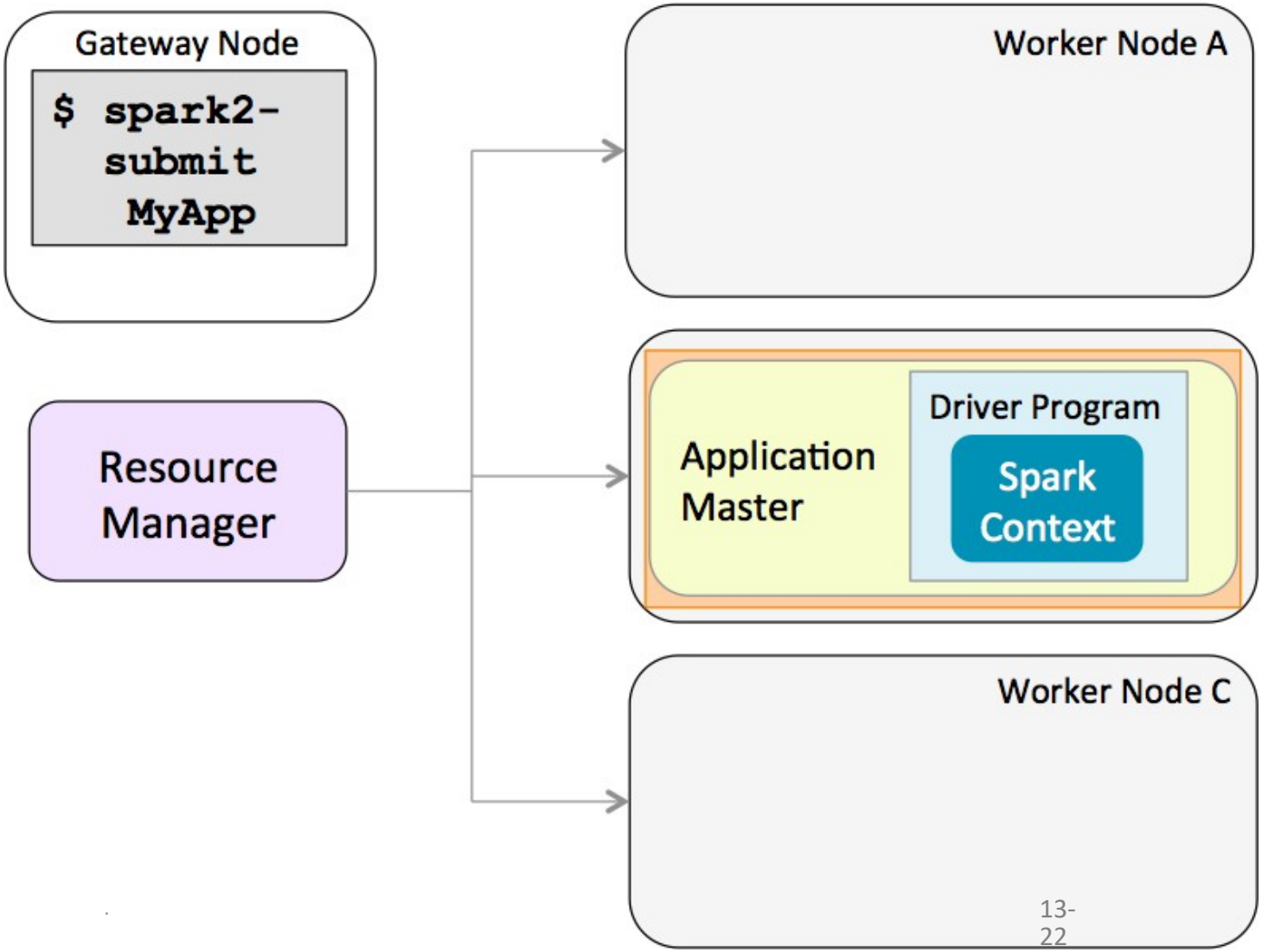
Spark Deployment Mode on YARN: Cluster Mode (1)



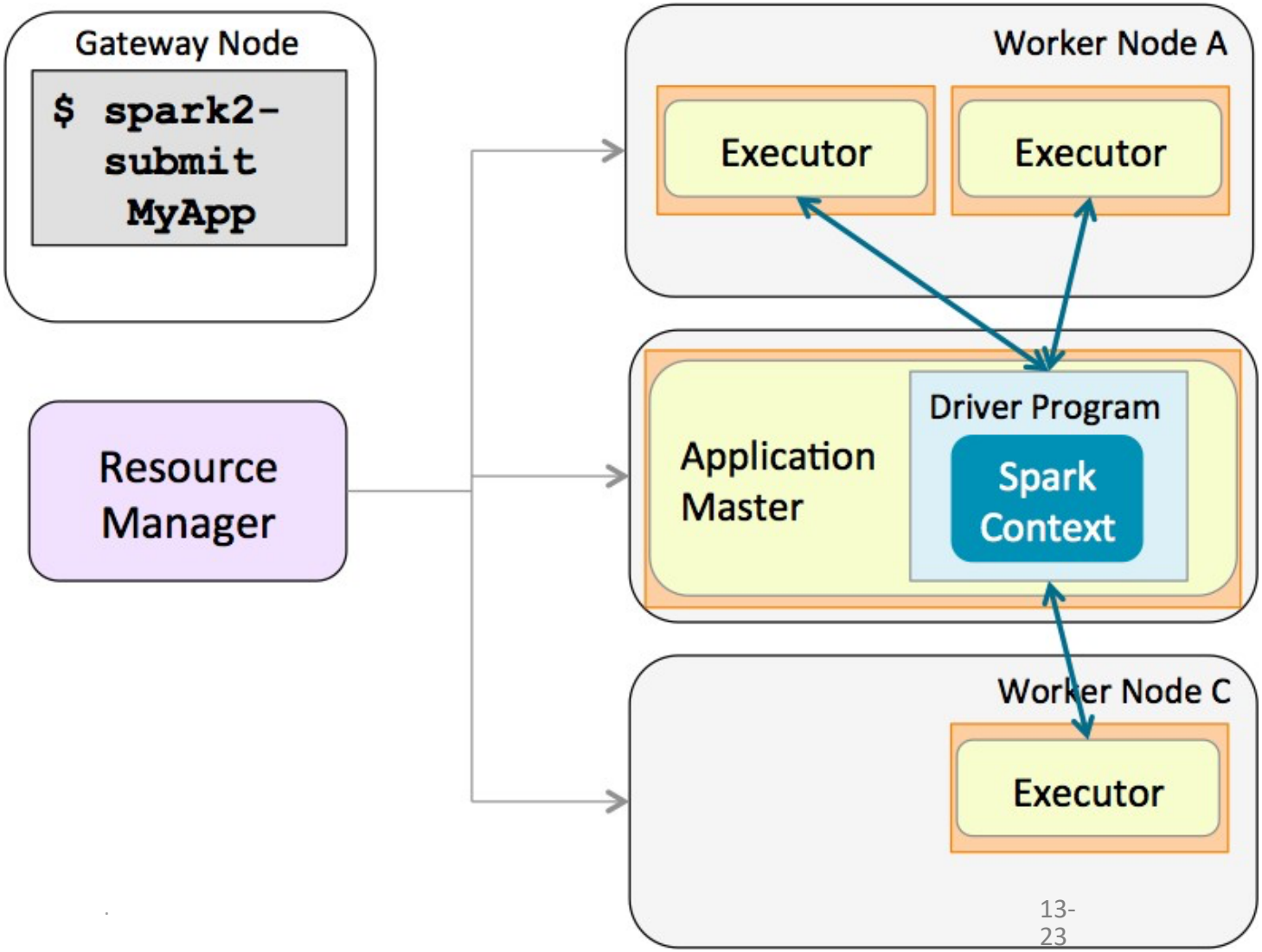
Spark Deployment Mode on YARN: Cluster Mode (2)



Spark Deployment Mode on YARN: Cluster Mode (3)



Spark Deployment Mode on YARN: Cluster Mode (4)



The Spark Application Web UI

- **The Spark UI lets you monitor running jobs, and view statistics and configuration**

Spark 2.1.0.cloudera1 | Jobs | Stages | Storage | Environment | Executors | SQL | Spark shell application UI

Executors

Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Blacklisted
Total(4)	13	267.6 KB / 1.5 GB	0.0 B	3	1	0	60	61	12 s (0.4 s)	35.6 MB	0.0 B	2.2 MB	0
Dead(2)	4	87.8 KB / 768.2 MB	0.0 B	2	0	0	60	60	12 s (0.4 s)	35.6 MB	0.0 B	2.2 MB	0
Active(2)	9	179.8 KB / 768.2 MB	0.0 B	1	1	0	0	1	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	0

Executors

Show 20 entries Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Logs	Thread Dump
driver	10.0.6.229:40885	Active	7	148.5 KB / 384.1 MB	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B		Thread Dump
1	worker-2:46762	Dead	2	30.6 KB / 384.1 MB	0.0 B	1	0	0	1	1	2 s (49 ms)	65.5 KB	0.0 B	0.0 B	stdout stderr	Thread Dump
2	worker-2:36897	Dead	2	57.2 KB / 384.1 MB	0.0 B	1	0	0	59	59	10 s (0.3 s)	35.5 MB	0.0 B	2.2 MB	stdout stderr	Thread Dump
3	worker-2:42528	Active	2	31.3 KB / 384.1 MB	0.0 B	1	1	0	0	1	0 ms (0 ms)	0.0 B	0.0 B	0.0 B	stdout stderr	Thread Dump

Showing 1 to 4 of 4 entries Previous 1 Next

Spark 2.1.0.cloudera1 | Jobs

Spark Jobs (?)

User: training
Total Uptime: 50 min
Scheduling Mode: FIFO
Completed Jobs: 4

Event Timeline

Completed Jobs (4)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	saveAsTextFile at <console>:33	2017/05/26 06:52:57	12 s	3/3	41/41
2	saveAsTextFile at <console>:33	2017/05/26 06:50:38	2 s	1/1 (2 skipped)	18/18 (23 skipped)
1	saveAsTextFile at <console>:33	2017/05/26 06:50:19	13 s	3/3	41/41
0	first at <console>:27	2017/05/26 06:46:47	15 s	1/1	1/1

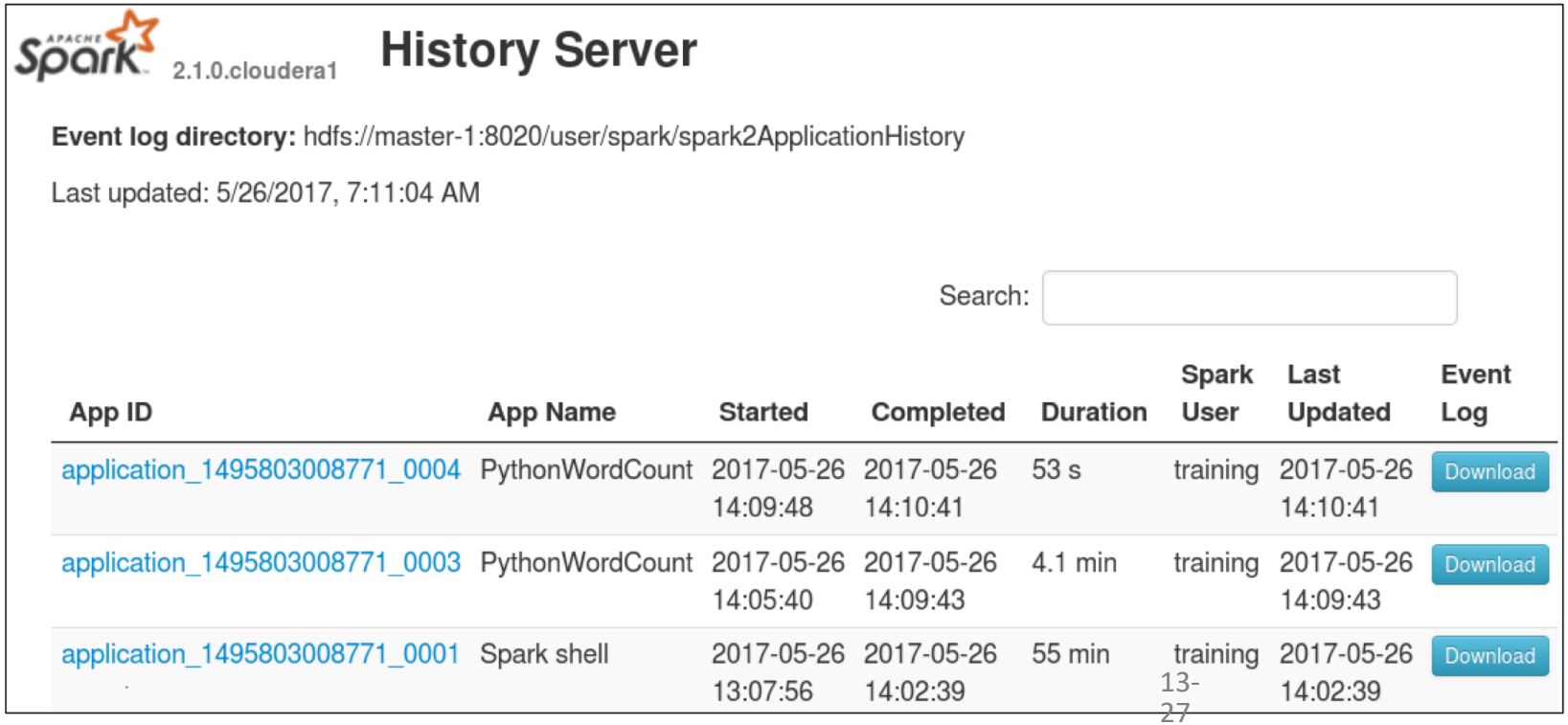
Accessing the Spark UI

- **The web UI is run by the Spark driver**
  - When running locally: `http://localhost:4040`
  - When running in client mode: `http://gateway:4040`
  - When running in cluster mode, access via the YARN UI

ueue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI
sers.training	Fri May 26 07:12:50 -0700 2017	N/A	RUNNING	UNDEFINED	1	1	1024		ApplicationMaster
sers.training	Fri May 26 07:09:52 -0700 2017	Fri May 26 07:10:41 -0700 2017	FINISHED	SUCCEEDED	N/A	N/A	N/A		History
sers.training	Fri May 26 07:05:45 -0700 2017	Fri May 26 07:09:44 -0700 2017	FINISHED	SUCCEEDED	N/A	N/A	N/A		History

Spark Application History UI

- **The Spark UI is only available while the application is running**
- **Use the Spark application history server to view metrics for a completed application**
  - Optional Spark component



Spark 2.1.0.cloudera1 History Server

Event log directory: hdfs://master-1:8020/user/spark/spark2ApplicationHistory

Last updated: 5/26/2017, 7:11:04 AM

Search:

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated	Event Log
application_1495803008771_0004	PythonWordCount	2017-05-26 14:09:48	2017-05-26 14:10:41	53 s	training	2017-05-26 14:10:41	Download
application_1495803008771_0003	PythonWordCount	2017-05-26 14:05:40	2017-05-26 14:09:43	4.1 min	training	2017-05-26 14:09:43	Download
application_1495803008771_0001	Spark shell	2017-05-26 13:07:56	2017-05-26 14:02:39	55 min	training	2017-05-26 14:02:39	Download

Viewing the Application History UI

- **You can access the history server UI by**
  - Using a URL with host and port configured by a system administrator
  - Following the History link in the YARN UI

ueue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB	Progress	Tracking UI
sers.training	Fri May 26 07:12:50 -0700 2017	N/A	RUNNING	UNDEFINED	1	1	1024		ApplicationMaster
sers.training	Fri May 26 07:09:52 -0700 2017	Fri May 26 07:10:41 -0700 2017	FINISHED	SUCCEEDED	N/A	N/A	N/A		History
sers.training	Fri May 26 07:05:45 -0700 2017	Fri May 26 07:09:44 -0700 2017	FINISHED	SUCCEEDED	N/A	N/A	N/A		History

Spark Application Configuration Properties

- **Spark provides numerous properties to configure your application**
- **Some example properties**
  - `spark.master`: Cluster type or URI to submit application to
  - `spark.app.name`: Application name displayed in the Spark UI
  - `spark.submit.deployMode`: Whether to run application in client or cluster mode (default: `client`)
  - `spark.ui.port`: Port to run the Spark Application UI (default `4040`)
  - `spark.executor.memory`: How much memory to allocate to each Executor (default `1g`)
  - `spark.pyspark.python`: Which Python executable to use for Pyspark applications
  - And many more…
    - See the Spark Configuration page in the Spark documentation for more details

Setting Configuration Properties

- **Most properties are set by system administrators**
  - Managed manually or using Cloudera Manager
  - Stored in a *properties file*
- **Developers can override system settings when submitting applications by**
  - Using submit script flags
  - Loading settings from a custom properties file instead of the system file
  - Setting properties programmatically in the application
- **Properties that are not set explicitly use Spark default values**

Overriding Properties Using Submit Script

- **Some Spark submit script flags set application properties**
  - For example
    - Use `--master` to set `spark.master`
    - Use `--name` to set `spark.app.name`
    - Use `--deploy-mode` to set `spark.submit.deployMode`
- **Not every property has a corresponding script flag**
  - Use `--conf` to set any property

```
$ spark2-submit \
  --conf spark.pyspark.python=/usr/bin/python2.7
```

Setting Properties in a Properties File

- **System administrators set system properties in properties files**
  - You can use your own custom properties file instead

```
spark.master                local[*]
spark.executor.memory       512k
spark.pyspark.python        /usr/bin/python2.7
```

- **Specify your properties file using the `properties-file` option**

```
$ spark2-submit \
  --properties-file=dir/my-properties.conf
```

- **Note that Spark will load *only* your custom properties file**
  - System properties file is ignored
  - Copy important system settings into your custom properties file
  - Custom file will not reflect future changes to system settings

Setting Configuration Properties Programmatically

- **Spark configuration settings are part of the Spark session or Spark context**
- **Set using the Spark session builder functions**
  - `appName` sets `spark.app.name`
  - `master` sets `spark.master`
  - `config` can set any property

```
import org.apache.spark.sql.SparkSession
…
val spark = SparkSession.builder.
  appName("my-spark-app").
  config("spark.ui.port","5050").
  getOrCreate()
…
```

Priority of Spark Property Settings

- **Properties set with higher priority methods override lower priority methods**
  1. Programmatic settings
  2. Submit script (command line) settings
  3. Properties file settings
     - Either administrator site-wide file or custom properties file
  4. Spark default settings
     - See the Spark Configuration guide

Viewing Spark Properties

- **You can view the Spark property settings two ways**
  - Using `--verbose` with the submit script
  - In the Spark Application UI Environment tab

Spark 2.1.0.cloudera1 | Jobs | Stages | Storage | Environment | Executors | SQL | **PySparkShell** application UI

Environment

Runtime Information

Name	Value
Java Home	/usr/java/jdk1.7.0_67-cloudera/jre
Java Version	1.7.0_67 (Oracle Corporation)
Scala Version	version 2.11.8

Spark Properties

Name	Value
spark.app.id	application_1495803008771_0005
spark.app.name	PySparkShell
spark.authenticate	false
spark.driver.appUIAddress	http://10.0.6.229:4040
spark.driver.extraLibraryPath	/opt/cloudera/parcels/CDH-5.11.0-1.cdh5.11.0.p0.34 /lib/hadoop/lib/native
spark.driver.host	10.0.6.229
spark.driver.port	50884

Submit Spark Job to YARN CLuster