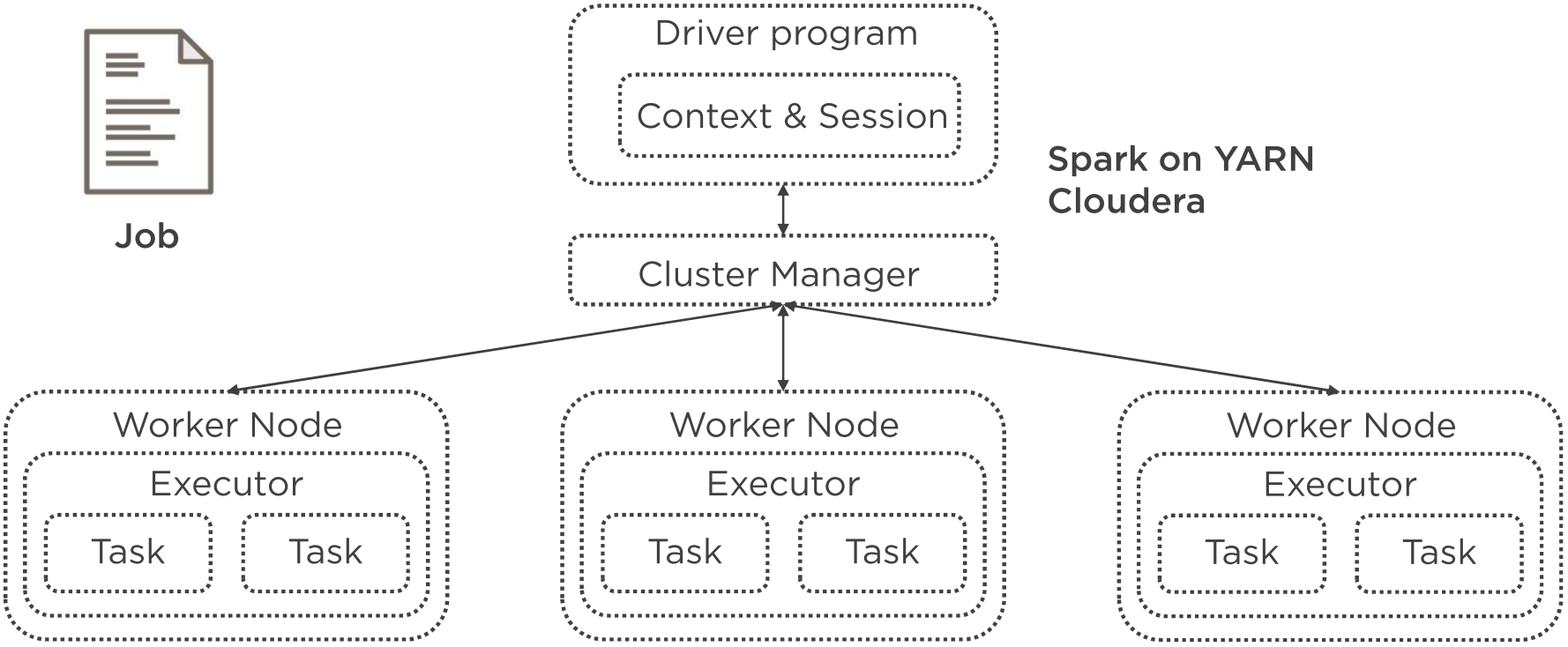
Getting Technical with Spark

Xavier Morera

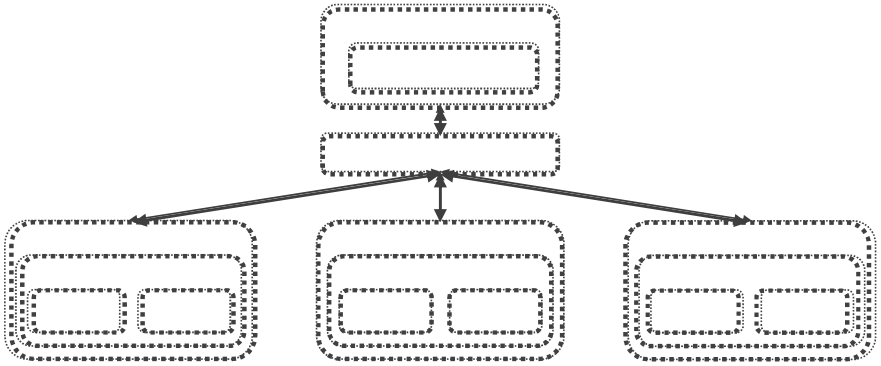
HELPING DEVELOPERS UNDERSTAND SEARCH & BIG DATA

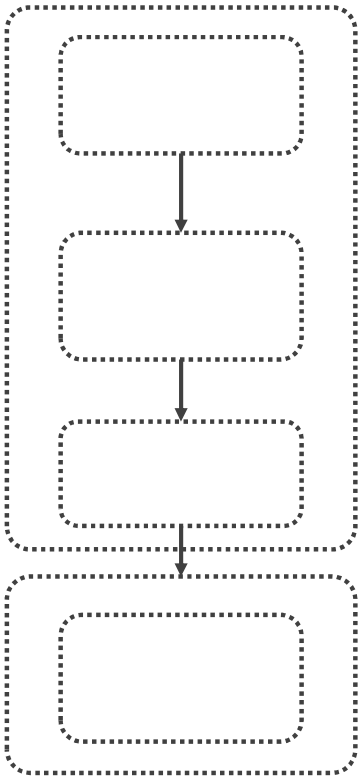
@xmorera www.xaviermorera.com

Spark's Architecture

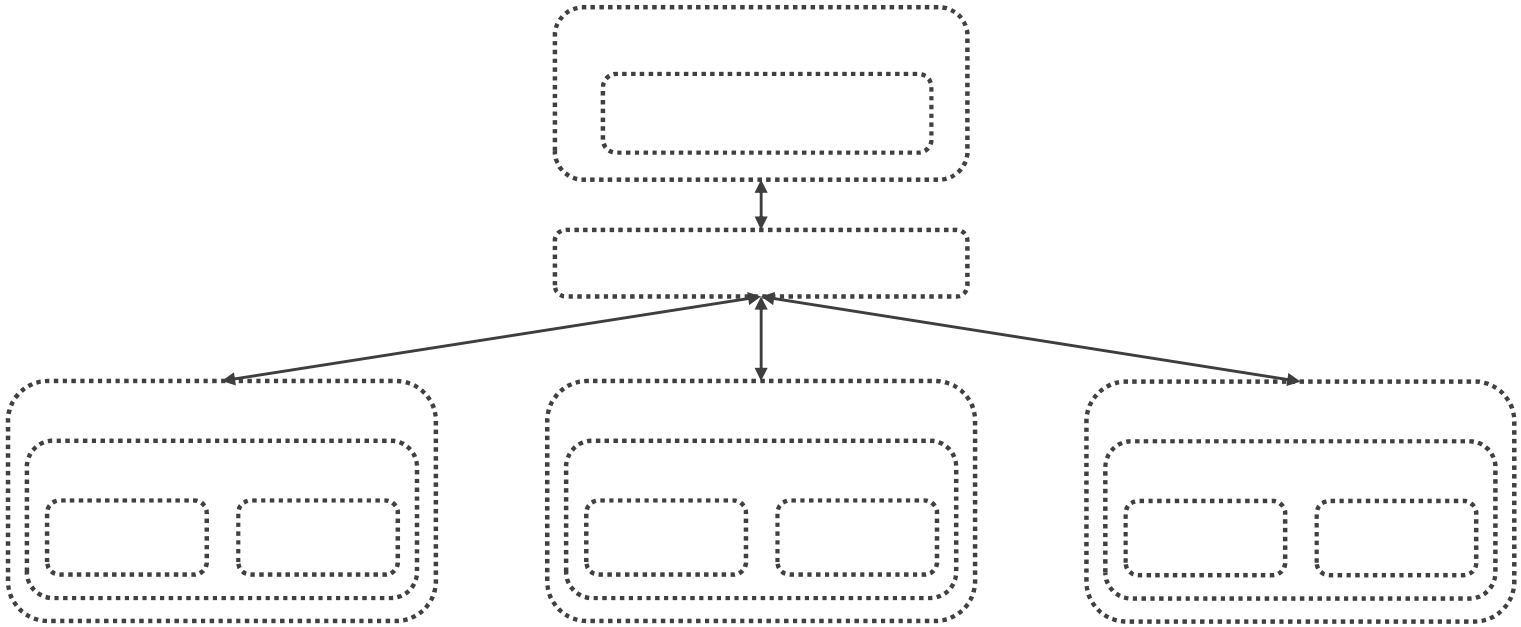


Developing Spark Applications

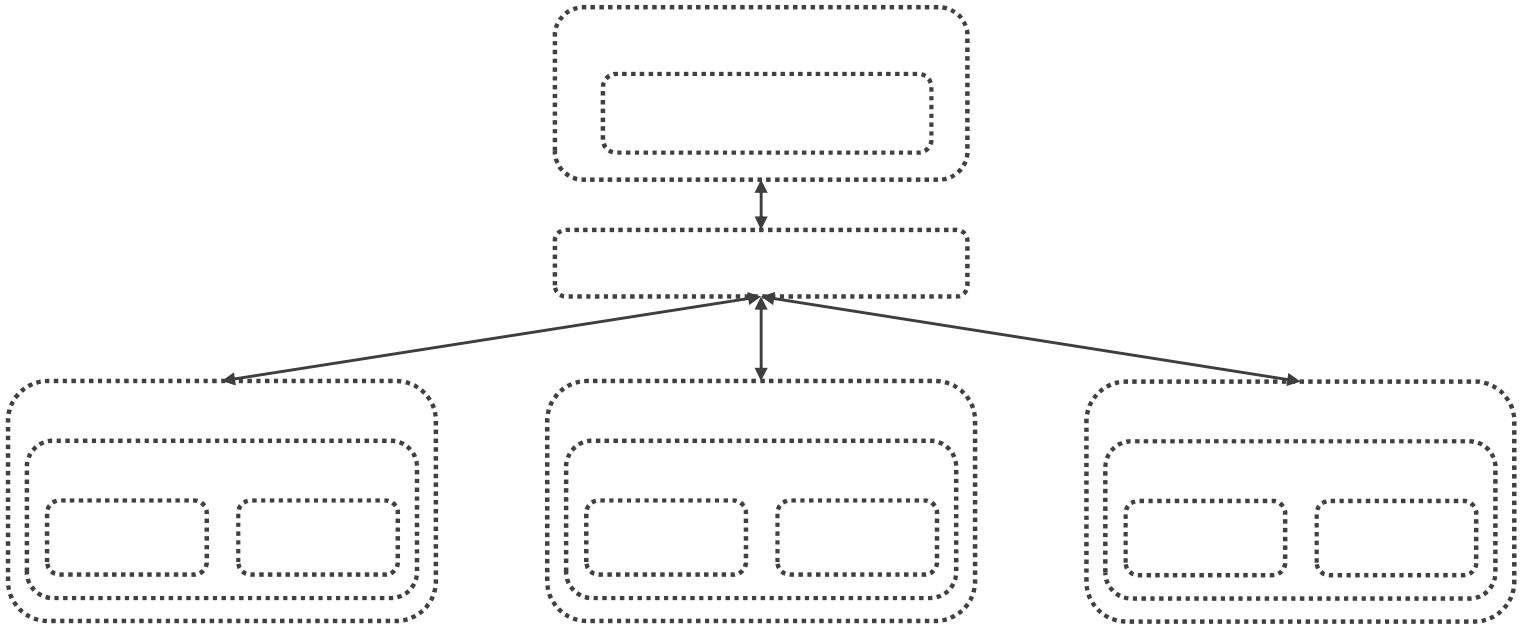


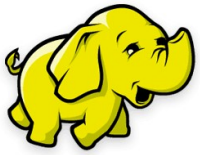


What's Missing?



Storage





Reading from Different Data Sources

```
sc.textFile("/user/cloudera/spark-committers-no-header.tsv").take(10)
```

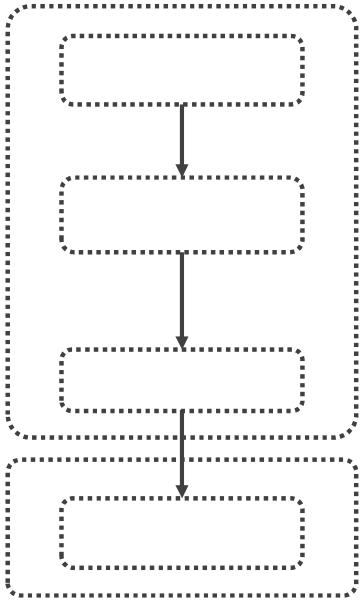
```
sc.textFile("file:///stackexchange/spark-committers-no-header.tsv").take(10)
```

```
sc.textFile("s3a://pluralsight-spark-cloudera-scala/spark-committers-no-header.tsv").take(10)
```

Let's Talk APIs

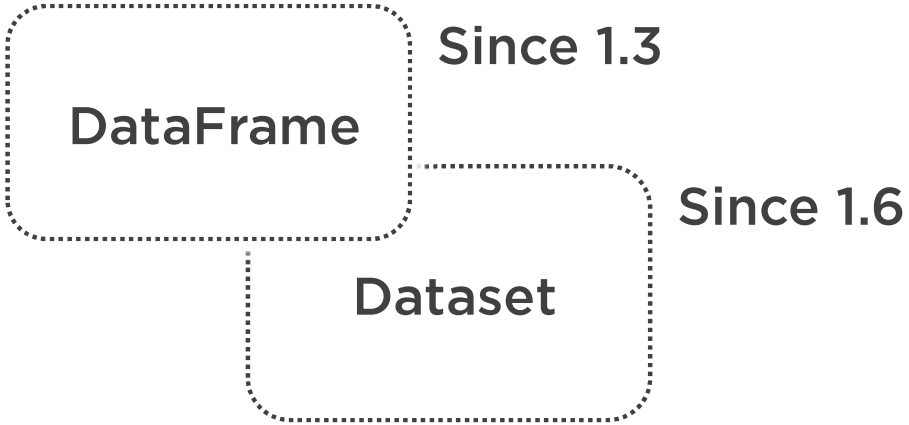
Let's Talk APIs

RDD



Since the beginning

DataFrame (and Dataset)



Google Search for messages

Groups

Spark Users ›

glom?

5 posts by 3 authors

☆	**Mark Hamstra** In reading through the Spark source to discover all of the undocumented goodies, I've come across the glom() method of RDDs. Okay, so I can turn an RDD into a GlommedRDD.	9/11/12
☆	**Scott Smith** x.glom().flatMap(_ -> _) == x It's great for taking all the rows of a split and turning them into a single larger object for doing more efficient operations than map would allow (like matrix	9/12/12
☆	**Matei Zaharia** It's meant for algorithms that want to work on an RDD's entire partition at once, such as stochastic gradient descent (a way to do gradient descent where you just merge in one point	9/12/12

Mark Hamstra 9/12/12

☆ Is there a plan to deprecate glom (or any thing else) in 0.6? How much more do you anticipate the api evolving before the 0.6 release?

- show quoted text -

Matei Zaharia 9/12/12

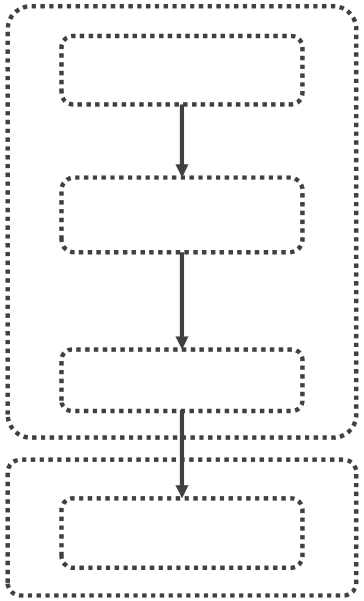
☆ I think we'll keep all of the APIs the same. I don't like removing these kinds of operations because some user code might depend on them and they're not that hard to maintain.

Matei

- show quoted text -

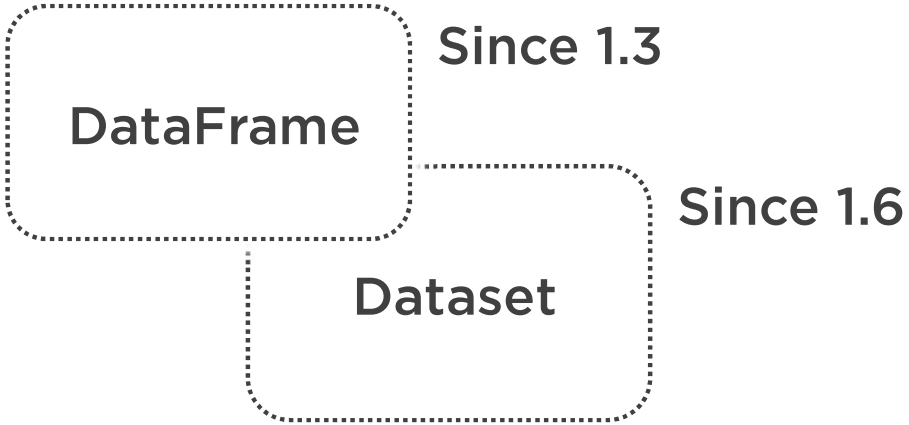
Let's Talk APIs

RDD



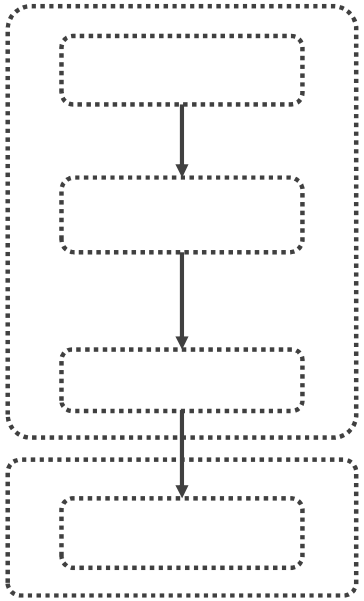
Since the beginning

DataFrame (and Dataset)



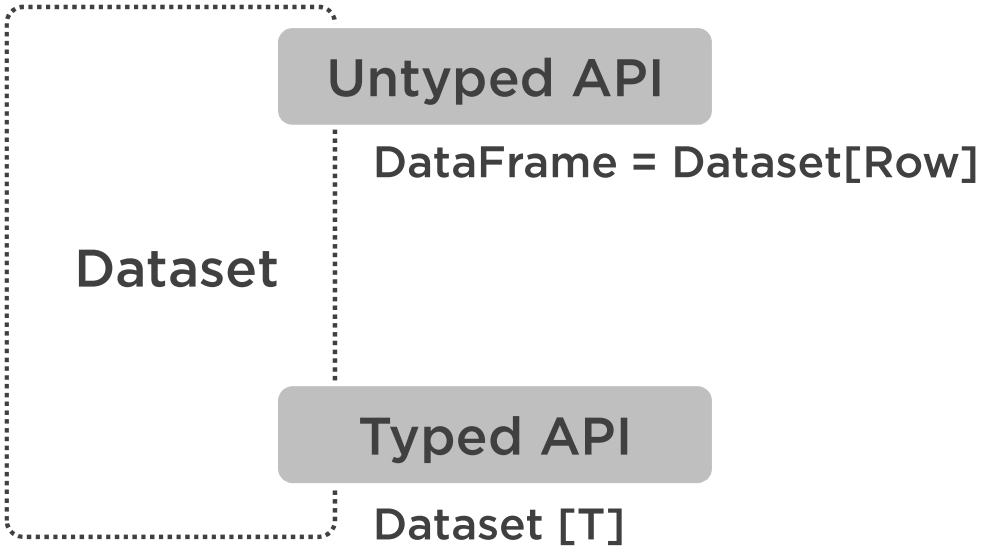
Let's Talk APIs

RDD



Since the beginning

DataFrame (and Dataset)



Since 2.0

Upgrading From Spark SQL 2.1 to 2.2

- Spark 2.1.1 introduced a new configuration key: `spark.sql.hive.caseSensitiveInferenceMode`. It had a default setting of `NEVER_INFER`, which kept behavior identical to 2.1.0. However, Spark 2.2.0 changes this setting's default value to `INFER_AND_SAVE` to restore compatibility with reading Hive metastore tables whose underlying file schema have mixed-case column names. With the `INFER_AND_SAVE` configuration value, on first access Spark will perform schema inference on any Hive metastore table for which it has not already saved an inferred schema. Note that schema inference can be a very time consuming operation for tables with thousands of partitions. If compatibility with mixed-case column names is not a concern, you can safely set `spark.sql.hive.caseSensitiveInferenceMode` to `NEVER_INFER` to avoid the initial overhead of schema inference. Note that with the new default `INFER_AND_SAVE` setting, the results of the schema inference are saved as a metastore key for future use. Therefore, the initial schema inference occurs only at a table's first access.

Upgrading From Spark SQL 2.0 to 2.1

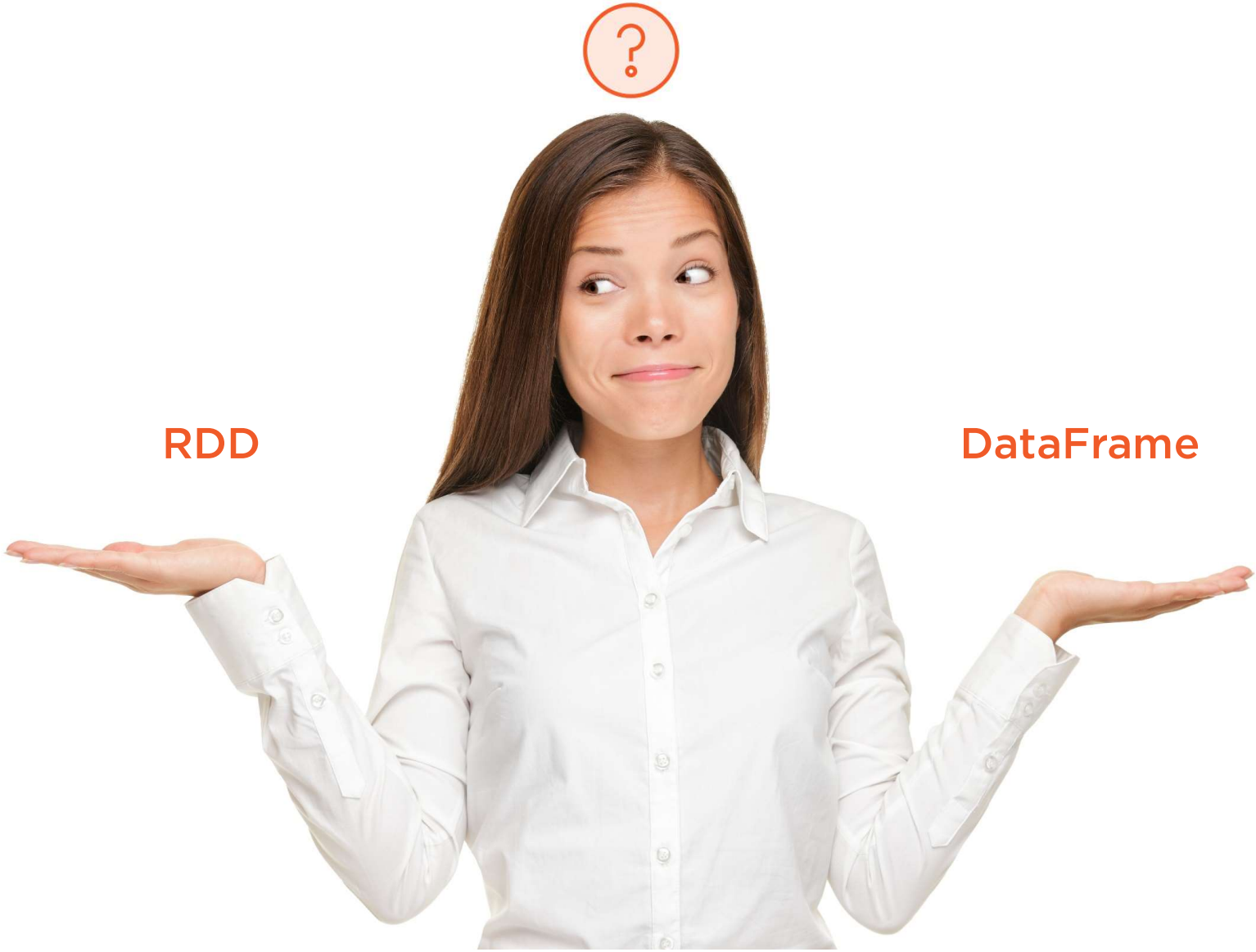
- Datasource tables now store partition metadata in the Hive metastore. This means that Hive DDLs such as `ALTER TABLE PARTITION ... SET LOCATION` are now available for tables created with the Datasource API.
  - Legacy datasource tables can be migrated to this format via the `MSCK REPAIR TABLE` command. Migrating legacy tables is recommended to take advantage of Hive DDL support and improved planning performance.
  - To determine if a table has been migrated, look for the `PartitionProvider: Catalog` attribute when issuing `DESCRIBE FORMATTED` on the table.
- Changes to `INSERT OVERWRITE TABLE ... PARTITION ...` behavior for Datasource tables.
  - In prior Spark versions `INSERT OVERWRITE` overwrote the entire Datasource table, even when given a partition specification. Now only partitions matching the specification are overwritten.
  - Note that this still differs from the behavior of Hive tables, which is to overwrite only partitions overlapping with newly inserted data.

Upgrading From Spark SQL 1.6 to 2.0

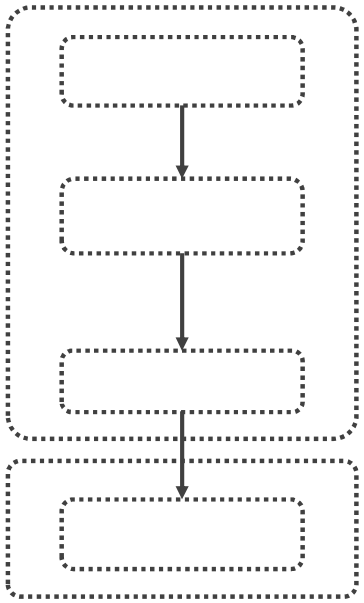
- `SparkSession` is now the new entry point of Spark that replaces the old `SQLContext` and `HiveContext`. Note that the old SQLContext and HiveContext are kept for backward compatibility. A new `catalog` interface is accessible from `SparkSession` - existing API on databases and

RDD or Dataset/DataFrame?

RDD	Dataset/DataFrame
Unstructured data	Structured or semi structured data
Data manipulation with lambdas	Much easier to understand
Low level transformations	Equivalent to database table
Functional	But better
	Leverage optimizations
	Relational
	Datasets have types


RDD
DataFrame

RDD Optimizations



Catalyst Optimizer

Project Tungsten

Catalyst Optimizer

Framework, part of Spark SQL

Optimize queries

Objectives

- Easily add new optimization techniques
- Enable external developers to extend it

Catalyst Optimizer

- Analyzes queries
- Creates an unresolved logical plan
- Looks at the Catalog
- Creates an optimized logical plan
- Optimized physical plan
- Optimized code generation

Project Tungsten

Push the boundaries of performance

Optimize for CPU and memory efficiency

Focuses on hardware where Spark runs

Project Tungsten

Tungsten row format

Manages memory explicitly

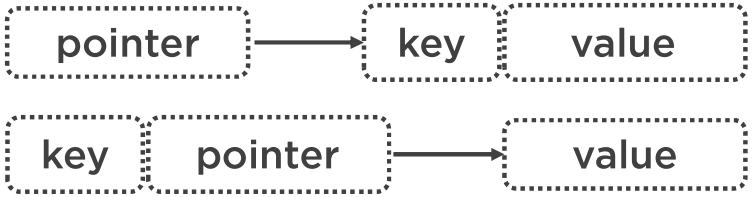
Cache aware computation

Project Tungsten

Tungsten row format

Manages memory explicitly

Cache aware computation



Project Tungsten

Tungsten row format

Manages memory explicitly

Cache aware computation

Whole-stage Code Generation

SparkContext

SparkSession

SparkContext

Located in the Spark Driver

Entry point for RDDs

The Spark application

- One SparkContext per application

Created for you in the REPL

You need to create for spark2-submit

SparkContext

```
sc
sc.version
sc.getClass
sc.appName
sc.getClass // try adding .getName
sc //hit tab, autocomplete will kick in
sc.uiWebUrl
sc.applicationId
sc.sparkUser
sc.textFile("/user/cloudera/spark-committers-no-header.tsv").take(10)
sc.parallelize(Array("Matei Zaharia", "Josh Rosen", "Holden Karau"))
```

SparkSession

Entry point to Spark SQL

Merges SQLContext and HiveContext

Access SparkContext

Can have multiple SparkSession objects

Created for you in REPL

You need to create for spark2-submit

SparkSession

```
spark
spark.getClass.getName
spark.sparkContext
spark.sql("select * from committers").show(10)
cmDF.select("Name").show(10)
spark.catalog.listDatabases().show(truncate=false)
```

SparkContext

SparkSession

Spark Configuration

Properties

Application parameters

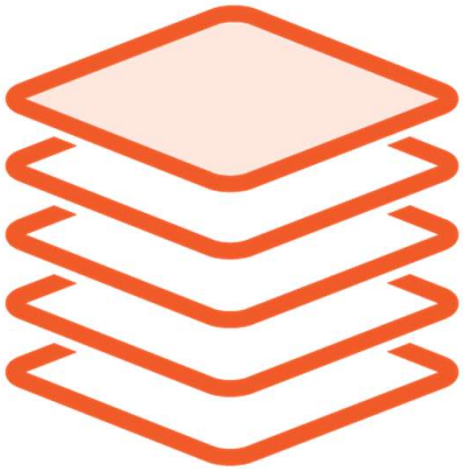
Environment Vars

Per machine settings

Logging

In log4j.properties

Configuration Precedence



Application (in code)

Flags passed to spark2-submit/spark2-shell

spark-defaults.conf

Environment

Runtime Information

Name	Value
Java Home	/usr/java/jdk1.8.0_144/jre
Java Version	1.8.0_144 (Oracle Corporation)
Scala Version	version 2.11.8

Spark Properties

Name	Value
spark.app.id	application_1511382093107_0012
spark.app.name	PySparkShell
spark.authenticate	false
spark.driver.appUIAddress	http://10.0.2.101:4040
spark.driver.extraLibraryPath	/opt/cloudera/parcels/CDH-5.12.1-1.cdh5.12.1.p0.3/lib/hadoop/lib/native
spark.driver.host	10.0.2.101
spark.driver.port	34232
spark.dynamicAllocation.enabled	true
spark.dynamicAllocation.executorIdleTimeout	60
spark.dynamicAllocation.minExecutors	0
spark.dynamicAllocation.schedulerBacklogTimeout	1
spark.eventLog.dir	hdfs://mgt01.cloudera:8020/user/spark/spark2ApplicationHistory
spark.eventLog.enabled	true
spark.executor.extraLibraryPath	/opt/cloudera/parcels/CDH-5.12.1-1.cdh5.12.1.p0.3/lib/hadoop/lib/native
spark.executor.id	driver
spark.executorEnv.PYTHONPATH	/opt/cloudera/parcels/SPARK2-2.2.0.cloudera1-1.cdh5.12.0.p0.142354/lib/spark2/python/lib/py4j-0.10.4-src.zip:/opt/cloudera/parcels/SPARK2-2.2.0.cloudera1-1.cdh5.12.0.p0.142354/lib/spark2/python/:<CPS>/opt/cloudera/parcels/SPARK2-2.2.0.cloudera1-1.cdh5.12.0.p0.142354/lib/spark2/python/lib/py4j-0.10.4-src.zip<CPS>/opt/cloudera/parcels/SPARK2-2.2.0.cloudera1-1.cdh5.12.0.p0.142354/lib/spark2/python/lib/pyspark.zip
spark.hadoop.mapreduce.application.classpath	
spark.hadoop.yarn.application.classpath	
spark.master	yarn
spark.org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter.param.PROXY_HOSTS	dn04.cloudera

```
import org.apache.spark.{SparkContext, SparkConf}
val conf = new SparkConf()
    .setMaster("yarn")
    .setAppName("Stack Overflow Test")
    .set("spark.executor.memory","1g")
val sc = new SparkContext(conf)
```

SparkConf with RDD API

Initialize SparkConf object

And pass on context creation

Object is cloned, so can't be modified

Time and Size Formats

5ms for 5 milliseconds

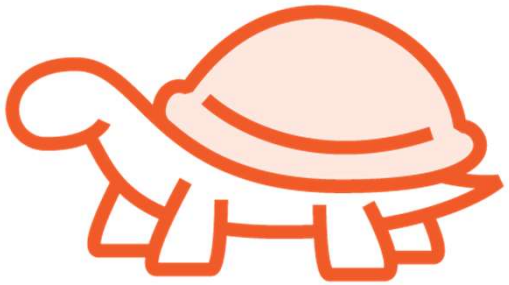
10m for 10 minutes

1y for 1 year

1g / 1gb for 1 gigabyte

1t / 1tb for 1 terabyte

~~Time and Size Formats~~



Imagine if you had to do single value

31622400 seconds for 1 year

```
spark2-submit --executor-memory 4g stackoverflowtest.scala
```

SparkConf with spark2-submit

You can pass configuration options when using spark2-submit

Dynamically set properties on execution

Also possible to use --config "spark.executor.memory=4g"

```
scala>  sc.getConf.getAll

res35: Array[(String, String)] =
Array((spark.driver.host,10.0.2.104),
(spark.eventLog.enabled,true),
(spark.driver.extraLibraryPath,/opt/cloudera/parcels/CDH-
5.12.1-1.cdh5.12.1.p0.3/lib/hadoop/lib/native),

scala> sc.getConf.get("spark.driver.host")
```

Reviewing Configuration

And you can review the configuration with sc.getConf.getAll

Or one specific configuration with sc.getConf.get

```
import org.apache.spark.sql.SparkSession
val spark = SparkSession.
   builder().
   appName("StackOverflow Test").
   config("spark.executor.memory","1g").
   getOrCreate()
```

SparkConf with Dataset API

Import SparkSession

Builder pattern: complex object, one step at a time

Named properties like master and appName

```
val spark = SparkSession.
    builder().
    appName("StackOverflow Test").
    config("spark.submit.deployMode","").
    getOrCreate()
```

Another Useful Configuration

Deployment mode

Specifies location of where the Spark driver executes

```
val spark = SparkSession.
    builder().
    appName("StackOverflow Test").
    config("spark.submit.deployMode","client").
    getOrCreate()
```

Client Deployment Mode

Driver runs where Spark application was launched

- i.e. local machine

Process killed if driver disconnected

```
val spark = SparkSession.
    builder().
    appName("StackOverflow Test").
    config("spark.submit.deployMode","cluster").
    getOrCreate()
```

Cluster Deployment Mode

Driver runs in a node within the cluster even if launched from outside

Process not killed if computer where submitted is disconnected

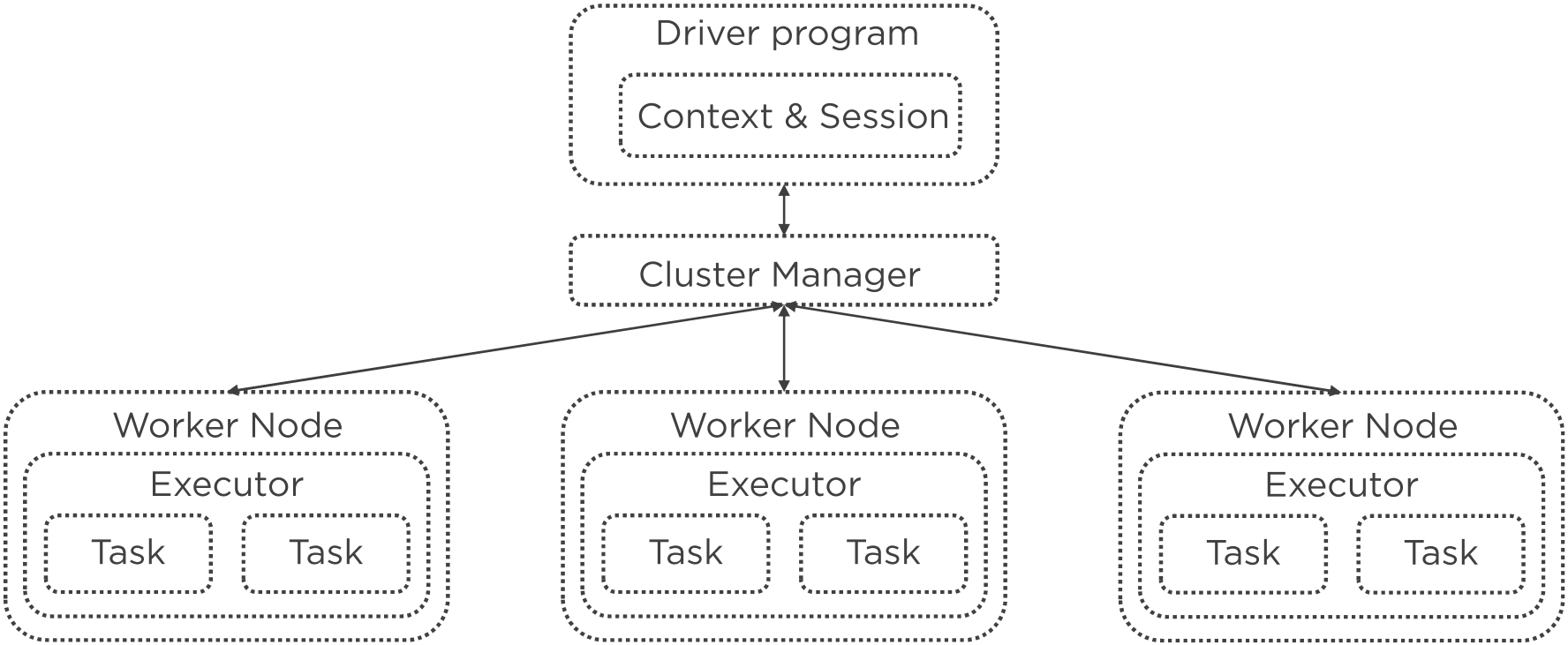
Does not support shell

Spark Configuration

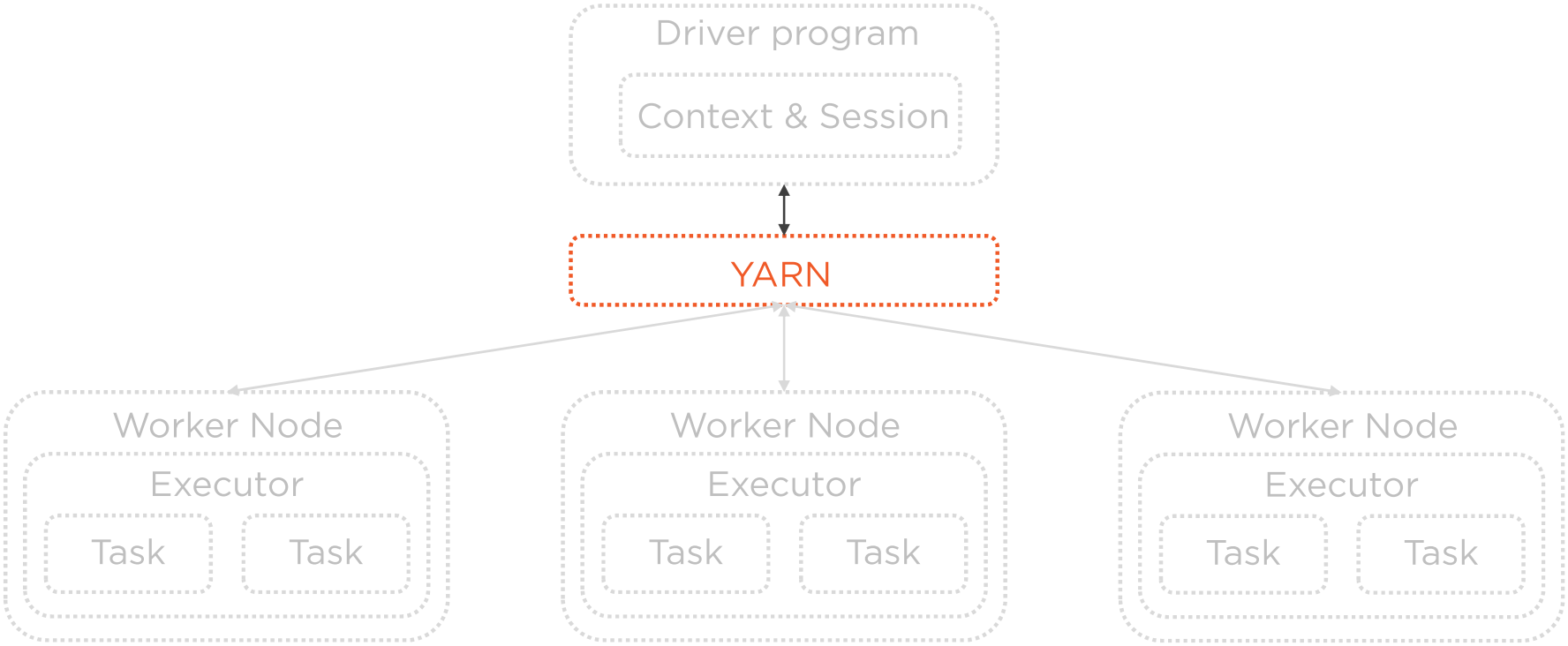
Configure Spark and fine tune it

- Your requirements & resources
- Workload type

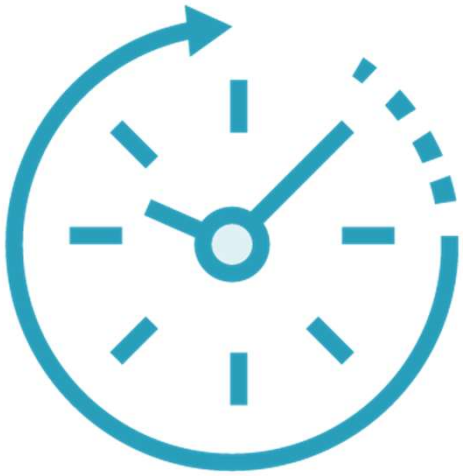
Spark's Architecture



Spark on YARN



YARN



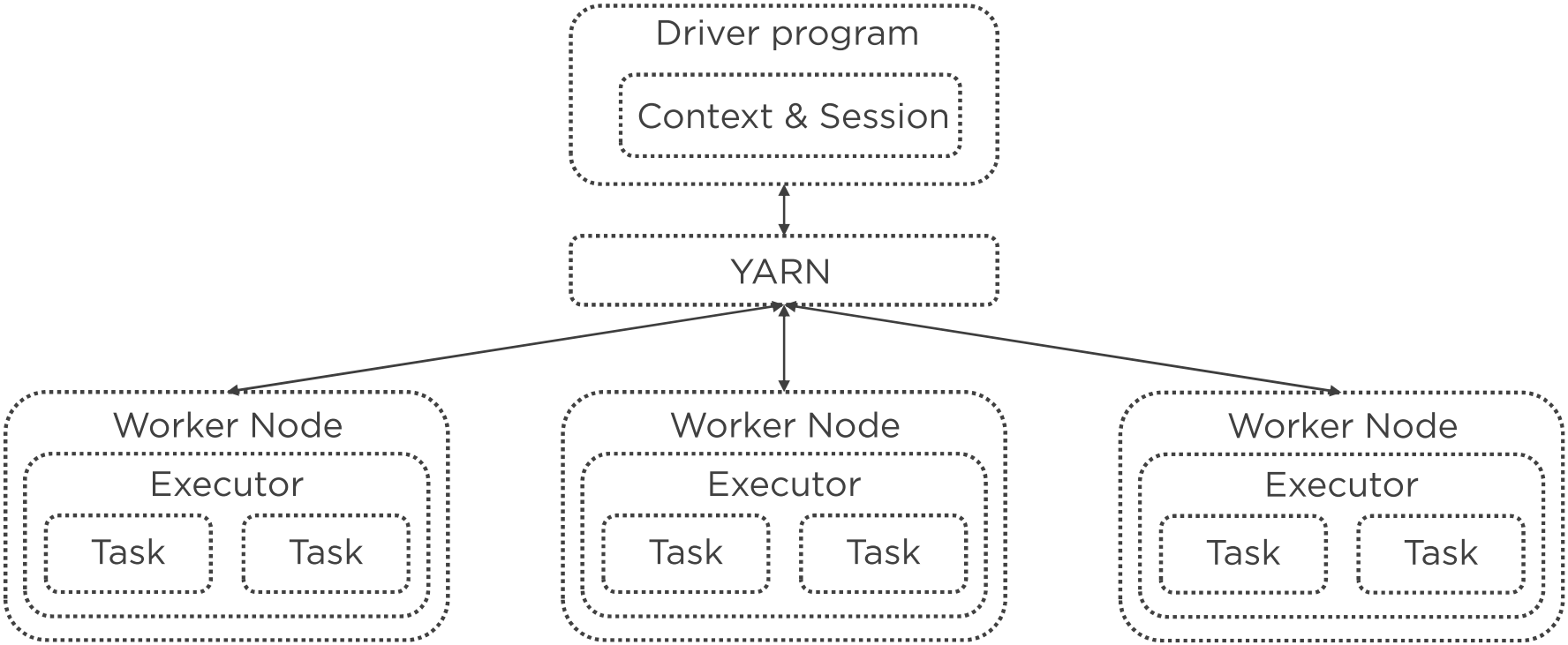
Yet Another Resource Negotiator

Cluster management technology

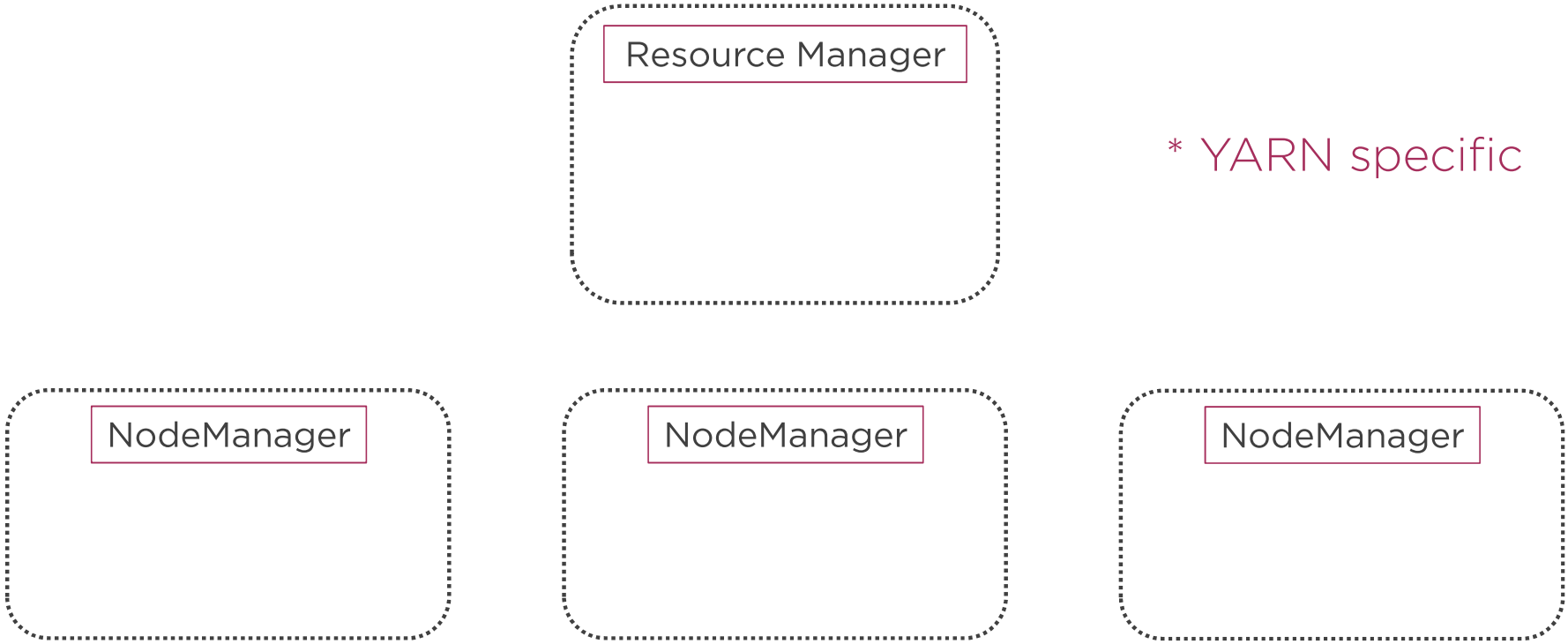
In charge of

- Resource management
- Job scheduling/monitoring

Spark on YARN

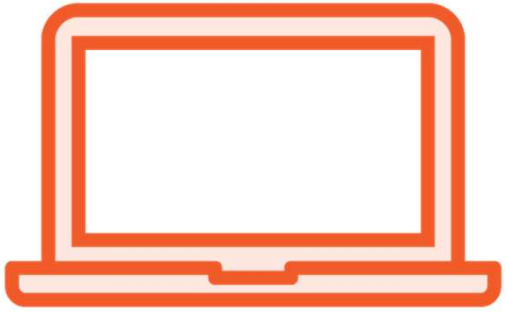


Running Spark Jobs



* YARN specific

Spark Cluster Mode



Resource Manager

NodeManager

NodeManager

Driver Program

NodeManager

Spark Client Mode

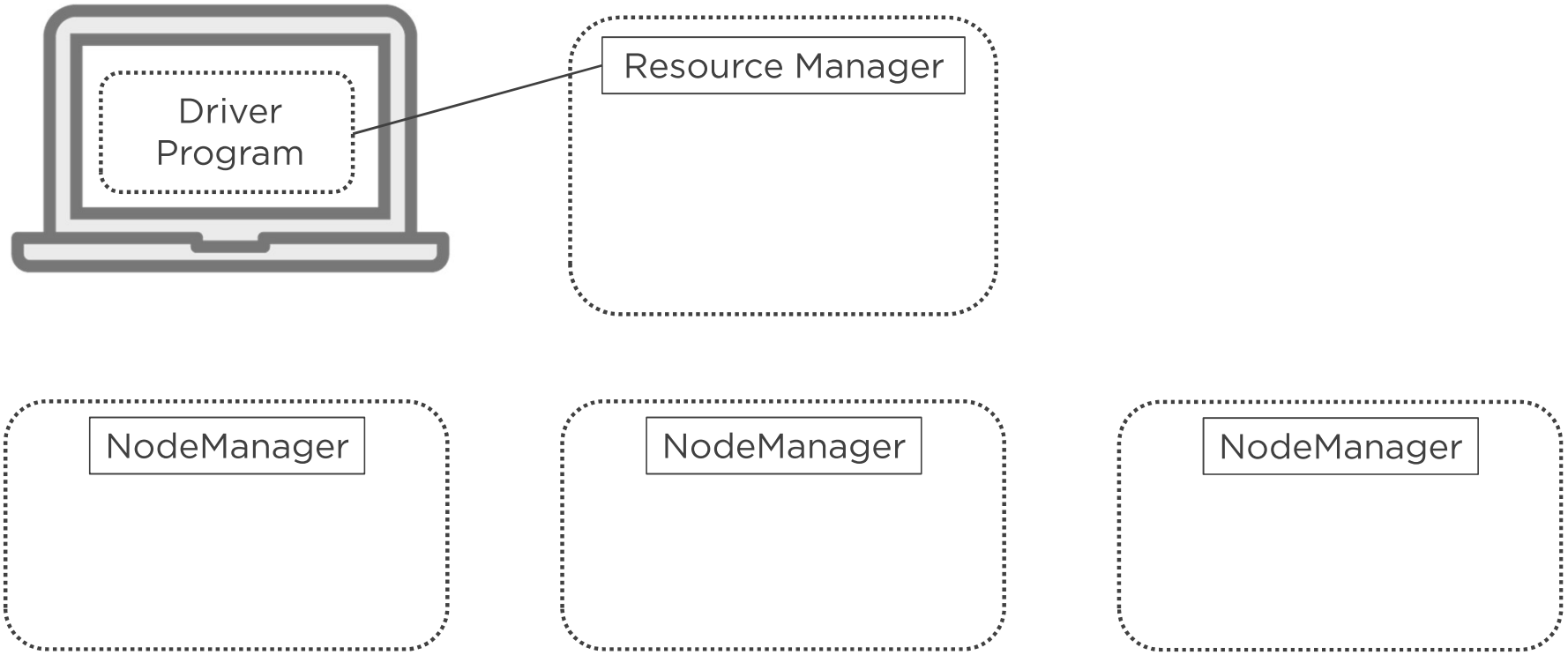
Resource Manager

NodeManager

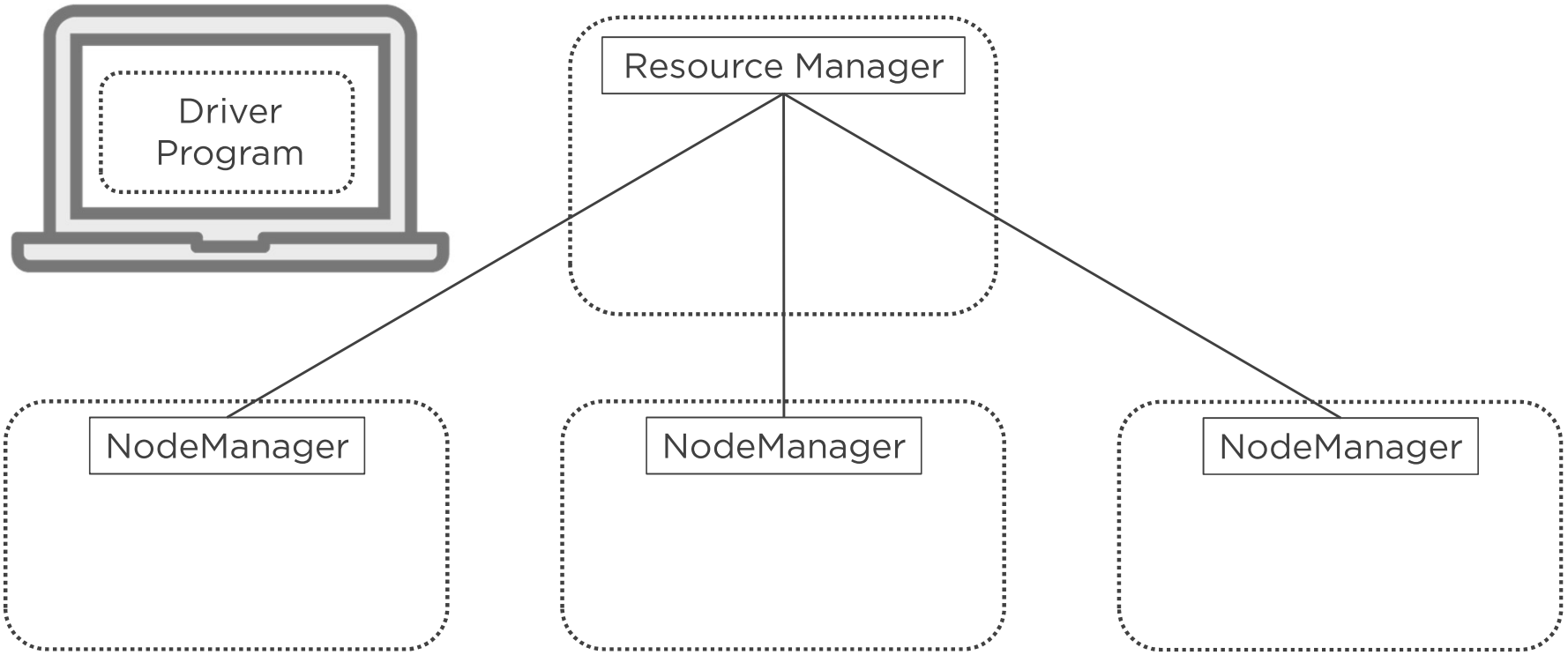
NodeManager

NodeManager

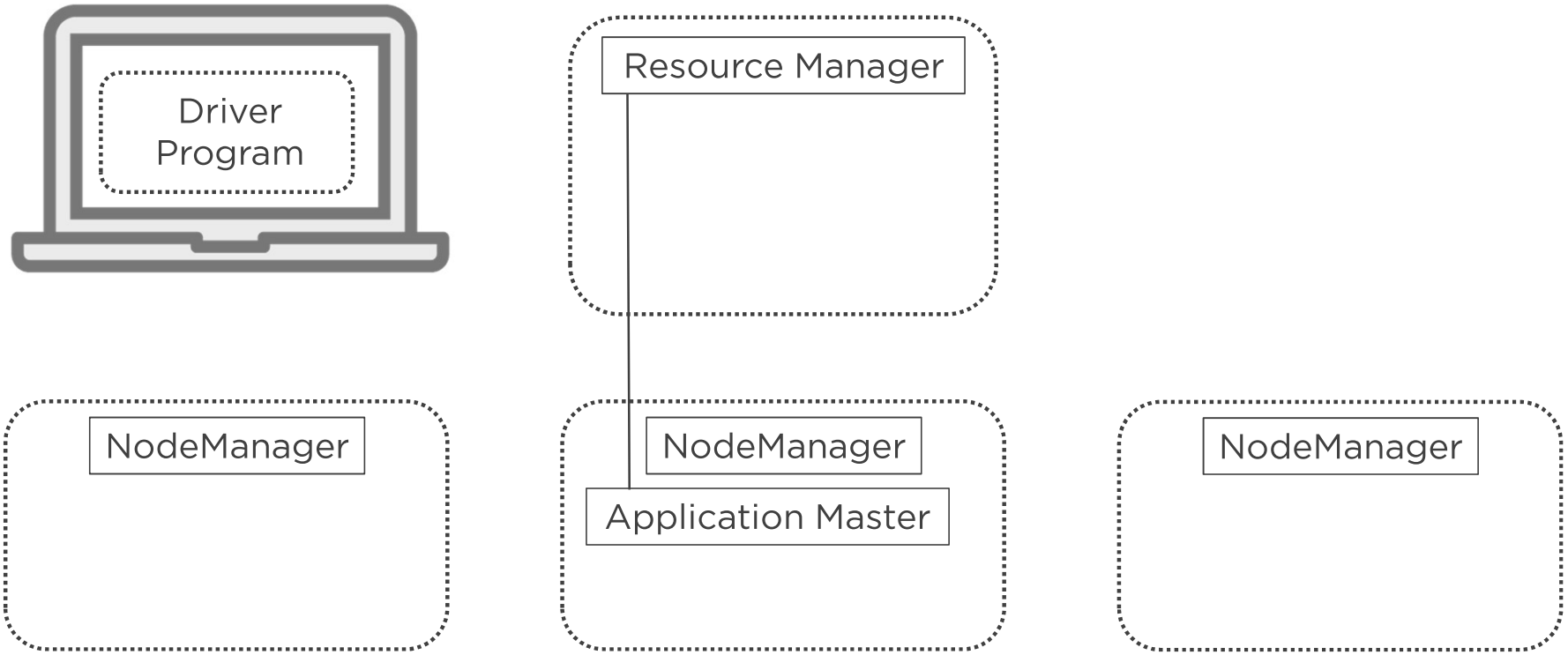
Spark Client Mode



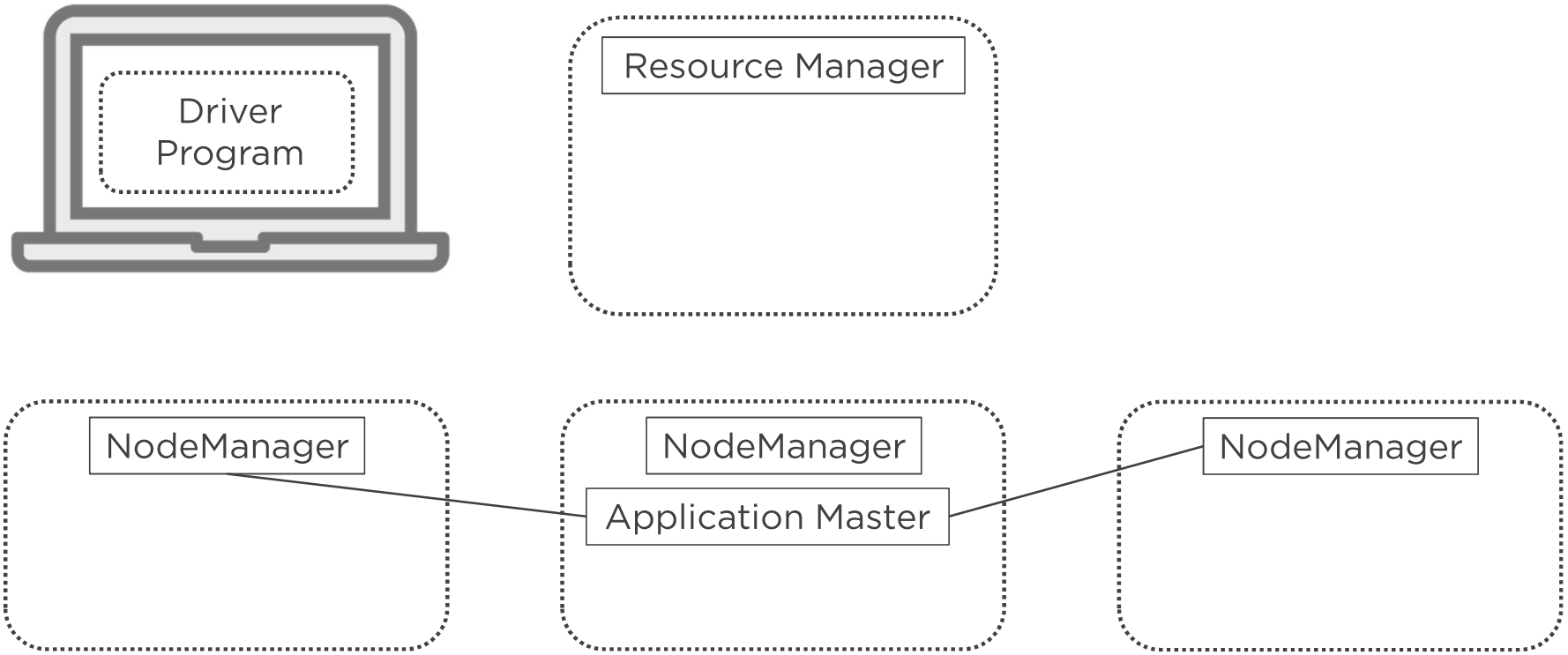
Spark Client Mode



Spark Client Mode



Spark Client Mode



Spark Client Mode

Resource Manager

NodeManager

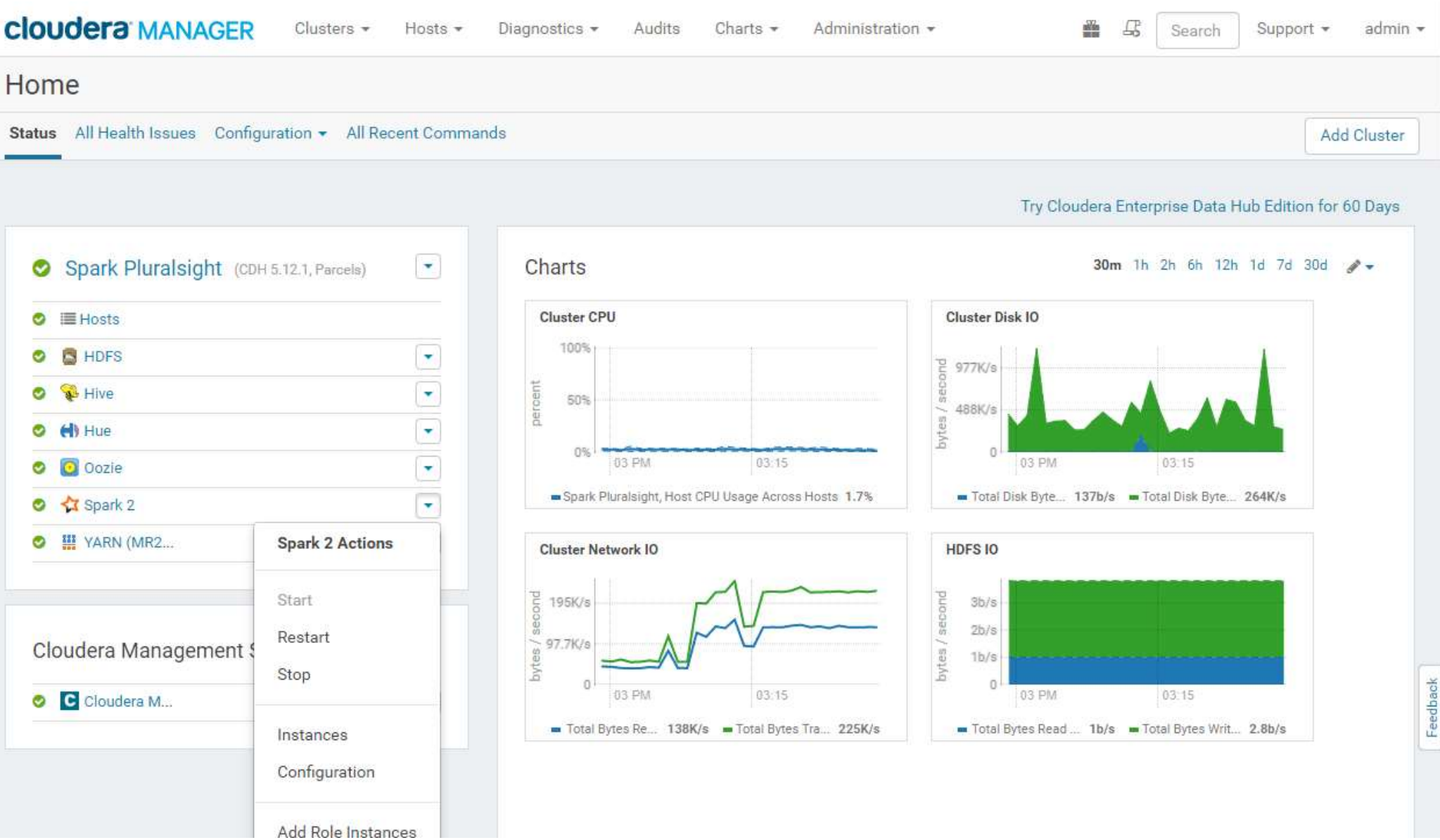
Executor

NodeManager

Application Master

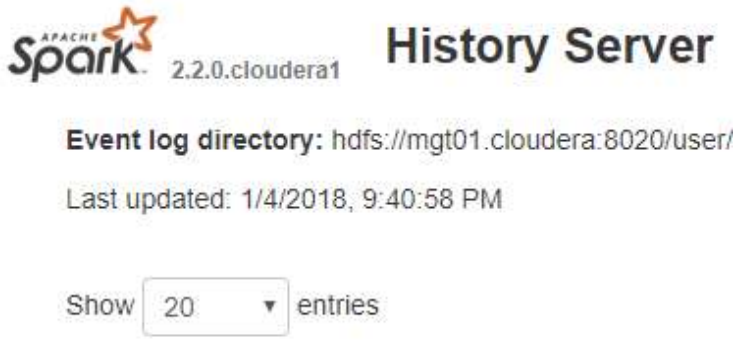
NodeManager

Executor


cloudera MANAGER
Clusters
Hosts
Diagnostics
Audits
Charts
Administration
Search
Support
admin
Home
Status
All Health Issues
Configuration
All Recent Commands
Add Cluster
Try Cloudera Enterprise Data Hub Edition for 60 Days
Spark Pluralsight (CDH 5.12.1, Parcels)
Hosts
HDFS
Hive
Hue
Oozie
Spark 2
YARN (MR2...
Spark 2 Actions
Start
Restart
Stop
Instances
Configuration
Add Role Instances
Cloudera Management S
Cloudera M...
Charts
30m 1h 2h 6h 12h 1d 7d 30d
Cluster CPU
100%
50%
0%
percent
03 PM
03:15
Spark Pluralsight, Host CPU Usage Across Hosts 1.7%
Cluster Disk IO
977K/s
488K/s
0
bytes / second
03 PM
03:15
Total Disk Byte... 137b/s
Total Disk Byte... 264K/s
Cluster Network IO
195K/s
97.7K/s
0
bytes / second
03 PM
03:15
Total Bytes Re... 138K/s
Total Bytes Tra... 225K/s
HDFS IO
3b/s
2b/s
1b/s
0
bytes / second
03 PM
03:15
Total Bytes Read ... 1b/s
Total Bytes Writ... 2.8b/s
Feedback

Visualizing Your Spark App: Web UI & History Server

Spark Web UI



Spark History Server

Spark Web UI

Interface of a running Spark application

Launched by the SparkContext

Located in port 4040

- Configurable
- If port busy, will move to 4041 and on...

Spark Web UI

Available

- Timeline
- Resources
- DAG
- Storage
- Environment
- Executors
- SQL

Spark History Server

Web UI

History

- Completed apps
- Failed and killed apps
- Running apps

Port 18089

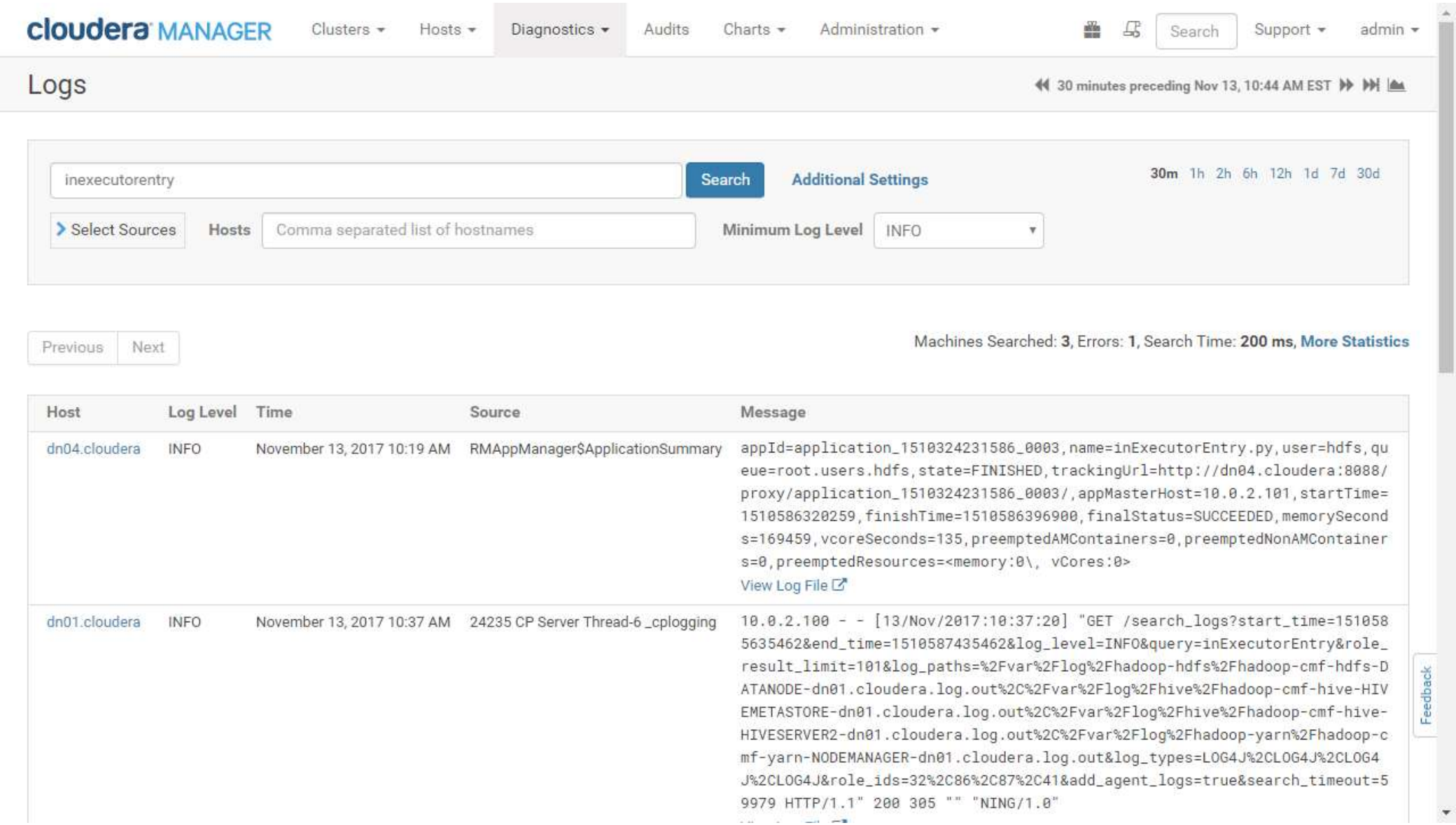
Spark History Server

Because Spark Web UI is removed after application exits

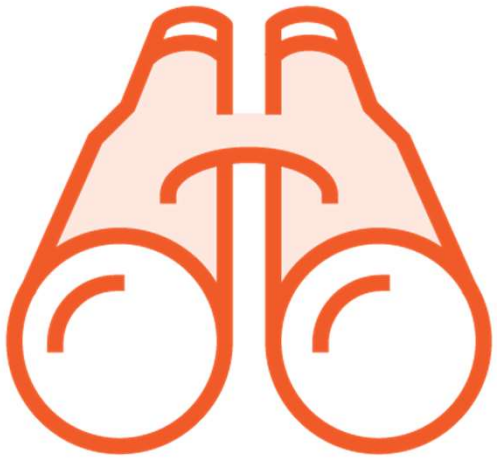
Log File

File that records events or messages

```
17/11/13 10:15:33 DEBUG ipc.ProtobufRpcEngine: Call: setTimes took 2ms
17/11/13 10:15:33 TRACE ipc.ProtobufRpcEngine: 35: Response <- mgt01.cloudera/10.0.2.100:8
020: setTimes {}
17/11/13 10:15:33 INFO cluster.YarnClientSchedulerBackend: Interrupting monitor thread
17/11/13 10:15:33 TRACE client.TransportClient: Sending RPC to /10.0.2.100:52130
17/11/13 10:15:33 TRACE client.TransportClient: Sending request 7575887306779588512 to /10
.0.2.100:52130 took 1 ms
17/11/13 10:15:33 TRACE protocol.MessageDecoder: Received message RpcResponse: RpcResponse
{requestId=7575887306779588512, body=NettyManagedBuffer{buf=PooledUnsafeDirectByteBuf(ridx
: 21, widx: 68, cap: 2048)}}
17/11/13 10:15:33 INFO cluster.YarnClientSchedulerBackend: Shutting down all executors
17/11/13 10:15:33 INFO cluster.YarnSchedulerBackend$YarnDriverEndpoint: Asking each execut
or to shut down
17/11/13 10:15:33 INFO cluster.SchedulerExtensionServices: Stopping SchedulerExtensionServ
ices
(serviceOption=None,
 services=List(),
 started=false)
17/11/13 10:15:33 DEBUG service.AbstractService: Service: org.apache.hadoop.yarn.client.ap
i.impl.YarnClientImpl entered state STOPPED
17/11/13 10:15:33 DEBUG ipc.Client: stopping client from cache: org.apache.hadoop.ipc.Clie
nt@21461fc
17/11/13 10:15:33 INFO cluster.YarnClientSchedulerBackend: Stopped
17/11/13 10:15:33 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoin
t stopped!
17/11/13 10:15:33 INFO memory.MemoryStore: MemoryStore cleared
17/11/13 10:15:33 INFO storage.BlockManager: BlockManager
```


cloudera MANAGER
Clusters
Hosts
Diagnostics
Audits
Charts
Administration
Search
Support
admin
Logs
30 minutes preceding Nov 13, 10:44 AM EST
inexecutorentry
Search
Additional Settings
30m 1h 2h 6h 12h 1d 7d 30d
Select Sources
Hosts
Comma separated list of hostnames
Minimum Log Level
INFO
Previous
Next
Machines Searched: 3, Errors: 1, Search Time: 200 ms, More Statistics
Host
Log Level
Time
Source
Message
dn04.cloudera
INFO
November 13, 2017 10:19 AM
RMAppManager$ApplicationSummary
appId=application_1510324231586_0003,name=inExecutorEntry.py,user=hdfs,queue=root.users.hdfs,state=FINISHED,trackingUrl=http://dn04.cloudera:8088/proxy/application_1510324231586_0003/,appMasterHost=10.0.2.101,startTime=1510586320259,finishTime=1510586396900,finalStatus=SUCCEEDED,memorySeconds=169459,vcoreSeconds=135,preemptedAMContainers=0,preemptedNonAMContainers=0,preemptedResources=<memory:0\, vCores:0>
View Log File
dn01.cloudera
INFO
November 13, 2017 10:37 AM
24235 CP Server Thread-6 _cplogging
10.0.2.100 - - [13/Nov/2017:10:37:20] "GET /search_logs?start_time=1510585635462&end_time=1510587435462&log_level=INFO&query=inExecutorEntry&role_result_limit=101&log_paths=%2Fvar%2Flog%2Fhadoop-hdfs%2Fhadoop-cmf-hdfs-DATANODE-dn01.cloudera.log.out%2C%2Fvar%2Flog%2Fhive%2Fhadoop-cmf-hive-HIVEMETASTORE-dn01.cloudera.log.out%2C%2Fvar%2Flog%2Fhive%2Fhadoop-cmf-hive-HIVESERVER2-dn01.cloudera.log.out%2C%2Fvar%2Flog%2Fhadoop-yarn%2Fhadoop-cmf-yarn-NODEMANAGER-dn01.cloudera.log.out&log_types=LOG4J%2CLOG4J%2CLOG4J%2CLOG4J&role_ids=32%2C86%2C87%2C41&add_agent_logs=true&search_timeout=59979 HTTP/1.1" 200 305 "" "NING/1.0"
Feedback

How Can I Find the Log Files?



Using the terminal, in Spark Log Dir

- Local
- HDFS

Spark UI

Spark History Server

Cloudera Manager

HUE

YARN

```
sc.setLogLevel("ALL")
```

Logging Level

Specify log4j's logging levels

Using setLogLevel()

Log Levels

OFF	No logging
FATAL	Only very severe errors, those that might cause application to abort
ERROR	Log errors, but that still may allow application to keep executing

Log Levels

WARN — Logs potentially harmful scenarios

INFO — Informational messages, that highlight progress of application

DEBUG — Used mostly by developers, mainly when debugging the application

Log Levels

TRACE	Even finer grained than DEBUG
ALL	Turns on all logging

Navigating the Documentation

Takeaway

Spark's architecture

- Driver, cluster manager, executor, tasks
- Storage
- Deployment modes
- YARN

Takeaway

Visualizing your Spark application

- Web UI
- History Server

Logging

Navigating the documentation

Takeaway

RDD API