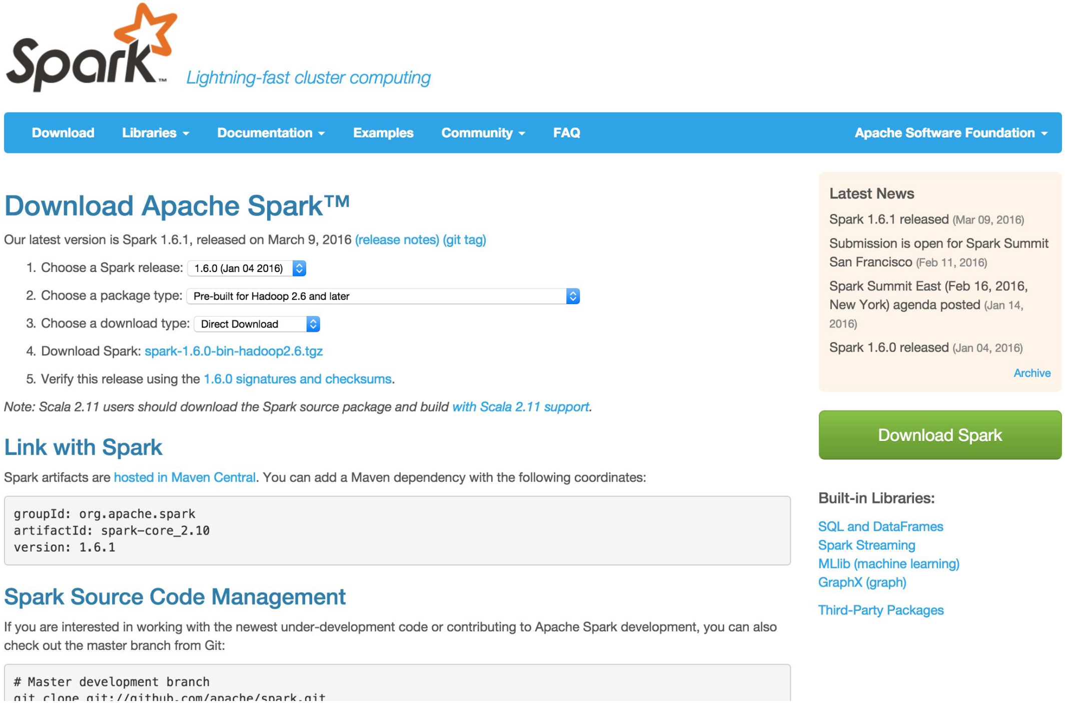
02 - Install Spark - Deployments

Demi Ben-Ari

Panorays

Official Spark Site

- http://spark.apache.org/



Download Apache Spark™

Our latest version is Spark 1.6.1, released on March 9, 2016 (release notes) (git tag)

1. Choose a Spark release: 1.6.0 (Jan 04 2016)
2. Choose a package type: Pre-built for Hadoop 2.6 and later
3. Choose a download type: Direct Download
4. Download Spark: spark-1.6.0-bin-hadoop2.6.tgz
5. Verify this release using the 1.6.0 signatures and checksums.

Note: Scala 2.11 users should download the Spark source package and build with Scala 2.11 support.

Link with Spark

Spark artifacts are hosted in Maven Central. You can add a Maven dependency with the following coordinates:

```
groupId: org.apache.spark
artifactId: spark-core_2.10
version: 1.6.1
```

Spark Source Code Management

If you are interested in working with the newest under-development code or contributing to Apache Spark development, you can also check out the master branch from Git:

```
# Master development branch
git clone git://github.com/apache/spark.git
```

Latest News

Spark 1.6.1 released (Mar 09, 2016)

Submission is open for Spark Summit San Francisco (Feb 11, 2016)

Spark Summit East (Feb 16, 2016, New York) agenda posted (Jan 14, 2016)

Spark 1.6.0 released (Jan 04, 2016)

Archive

Built-in Libraries:

- SQL and DataFrames
- Spark Streaming
- MLlib (machine learning)
- GraphX (graph)
- Third-Party Packages

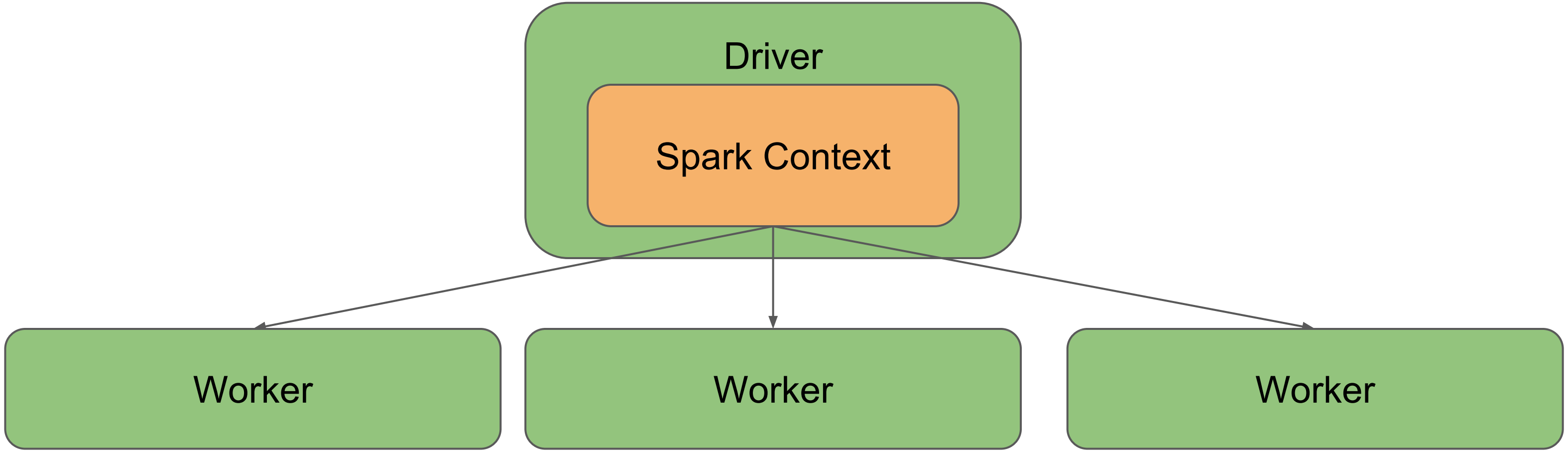
Some Configuration

- SPARK_HOME=<Spark Root>
    - Add $SPARK_HOME/bin to the $PATH environment variable
- Logging
    - $SPARK_HOME/conf => log4j.properties
    - Better switching all to ERROR

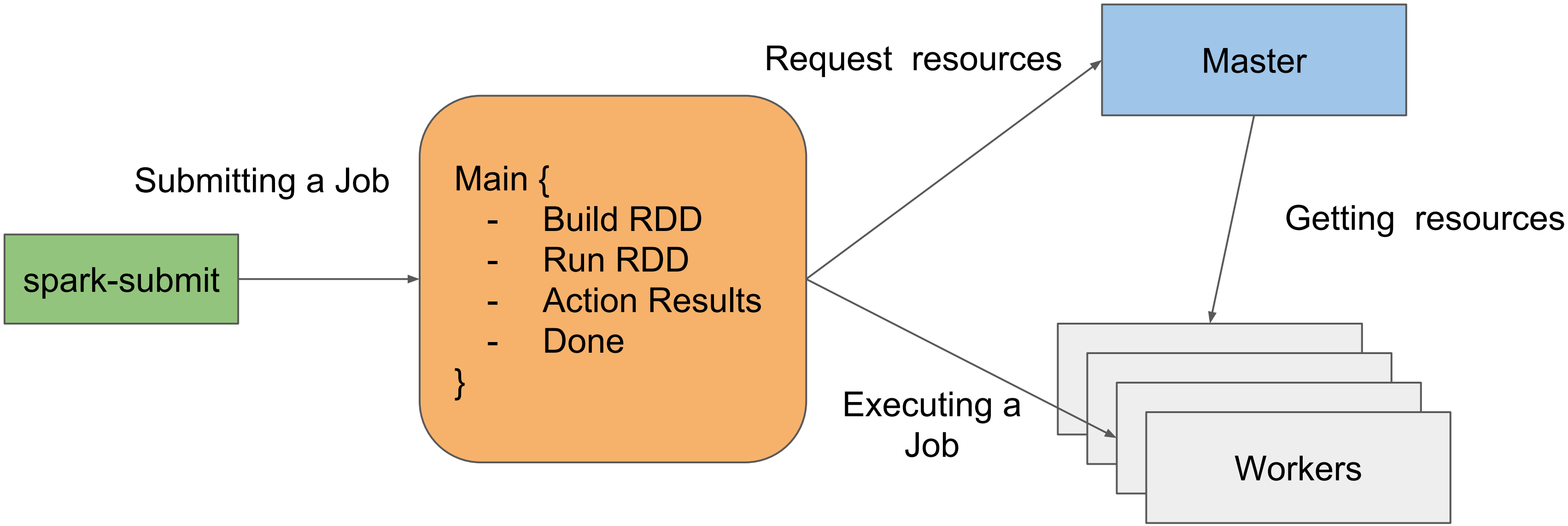
Running Spark on Windows

- There are no many (Or some at all that I know of) deployments on Windows of Spark.
- When Running Windows:
  add “winutils.exe” to the /bin/ directory
  Spark ticked: SPARK-2356
  Add HADOOP_HOME variable to the path that **“winutils.exe”** exists at.

Spark Mechanics



Spark-submit process



spark-submit

Submitting a Job

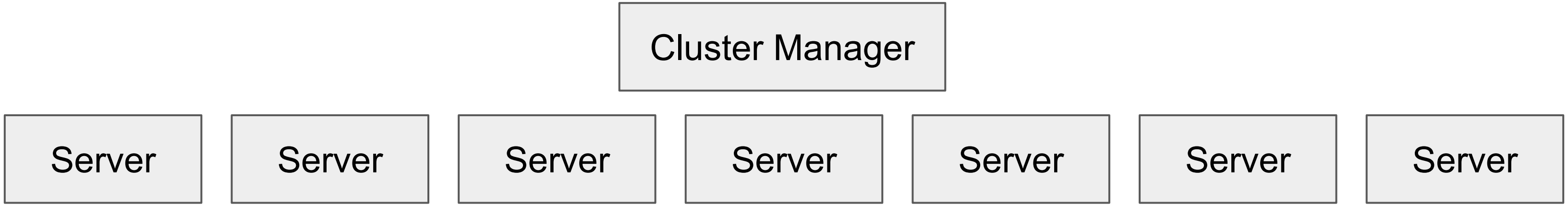
```
Main {
    -    Build RDD
    -    Run RDD
    -    Action Results
    -    Done
}
```

Panorays

Cluster Managers - configuration

- --master
    - spark://host:port
    - mesos://host:port
    - yarn (client : default, cluster)
    - Local[] (number of cores)
- --class - Application to run (Full path)
- --name
- --conf
- --properties-file <File>
- --driver-memory
- --executor-memory

Cluster Managers



hadoop

--master yarn
HADOOP/YARN_CONF_DIR
client / cluster
--num-executors #
--executor-cores #

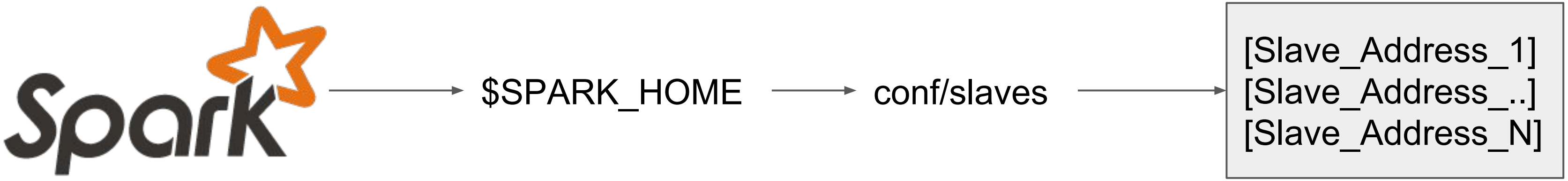
Spark

--master spark://host:7077
client / cluster
spark.deploy.spreadOut=true
--total-executor-cores #
--executor-cores #

MESOS

--master mesos://host:5050
client / cluster
spark.mesos.coarse=false
--total-executor-cores #

Spark Standalone



$SPARK_HOME → conf/slaves →

[Slave_Address_1]
[Slave_Address_..]
[Slave_Address_N]

Should have a passwordless ssh to all of the slaves from the master

- ./sbin/start-all.sh
- Processes
  - bin/spark-class org.apache.deploy.master.Master
  - bin/spark-class org.apache.deploy.master.Worker
    - spark://[Master]:7077
  - Application files: $SPARK_HOME/work by default
- http://spark.apache.org/docs/latest/spark-standalone.html

Useful facts

- Spark UI launches always by default on port 4040
  - If it doesn’t succeed it increments by 1 (4041, 4042...)
  - Port 4045 is inaccessible by browsers. (Can be hogged manually)
  - Default maximum UIs is 16.
  - Applications can be launched with no UI via: **“spark.ui.enabled false”** (ERROR org.apache.spark.ui.SparkUI- Failed to bind SparkUI java.net.BindException: **Address already in use: Service 'SparkUI'** failed after 16 retries!)

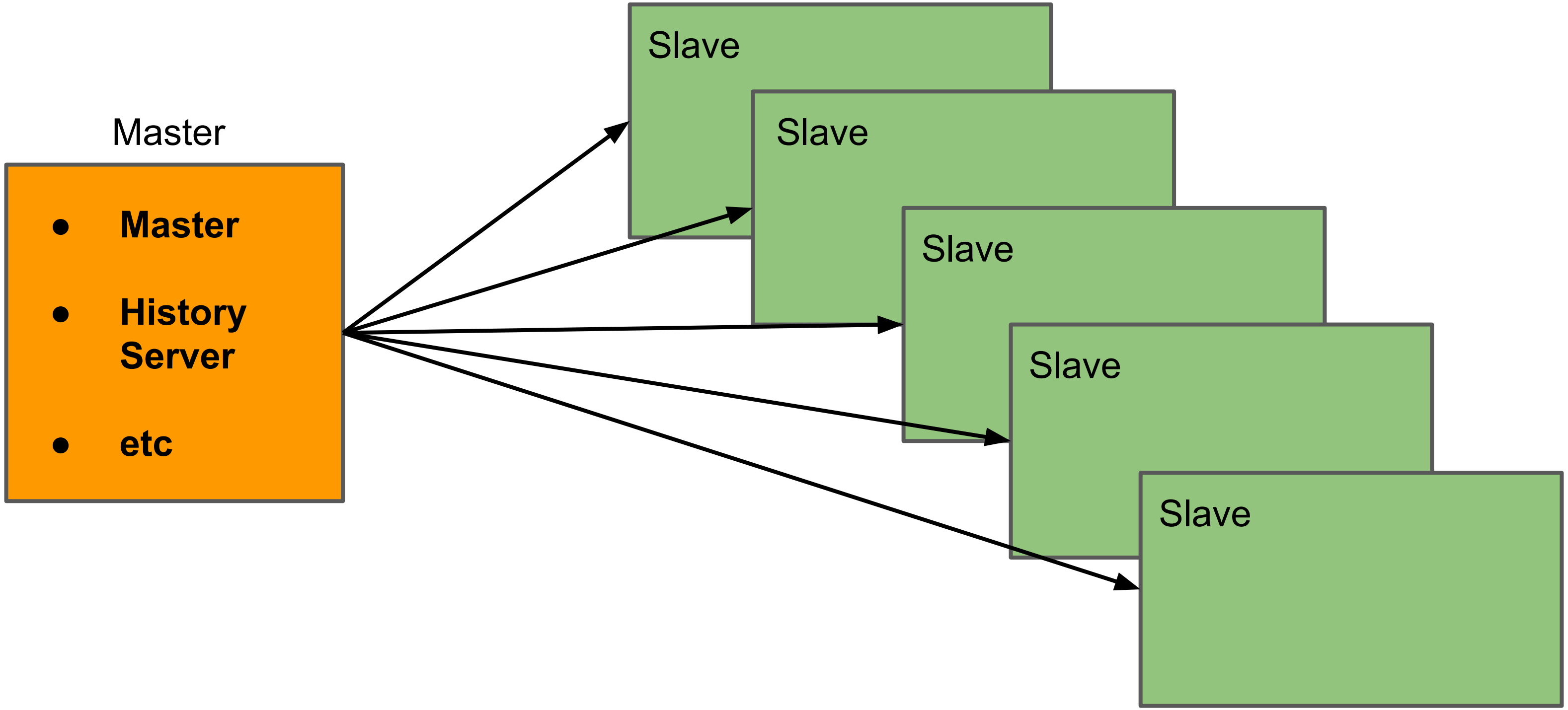
Spark Maintenance

- Nodes might go down
  - Monitoring is the solution
- Know your application's workload
  - Have visibility on the running applications and on the cluster utilization
- Monitor Whatever you can
  - Application runs
  - Application execution times
  - Failure logs
  - Spark History server

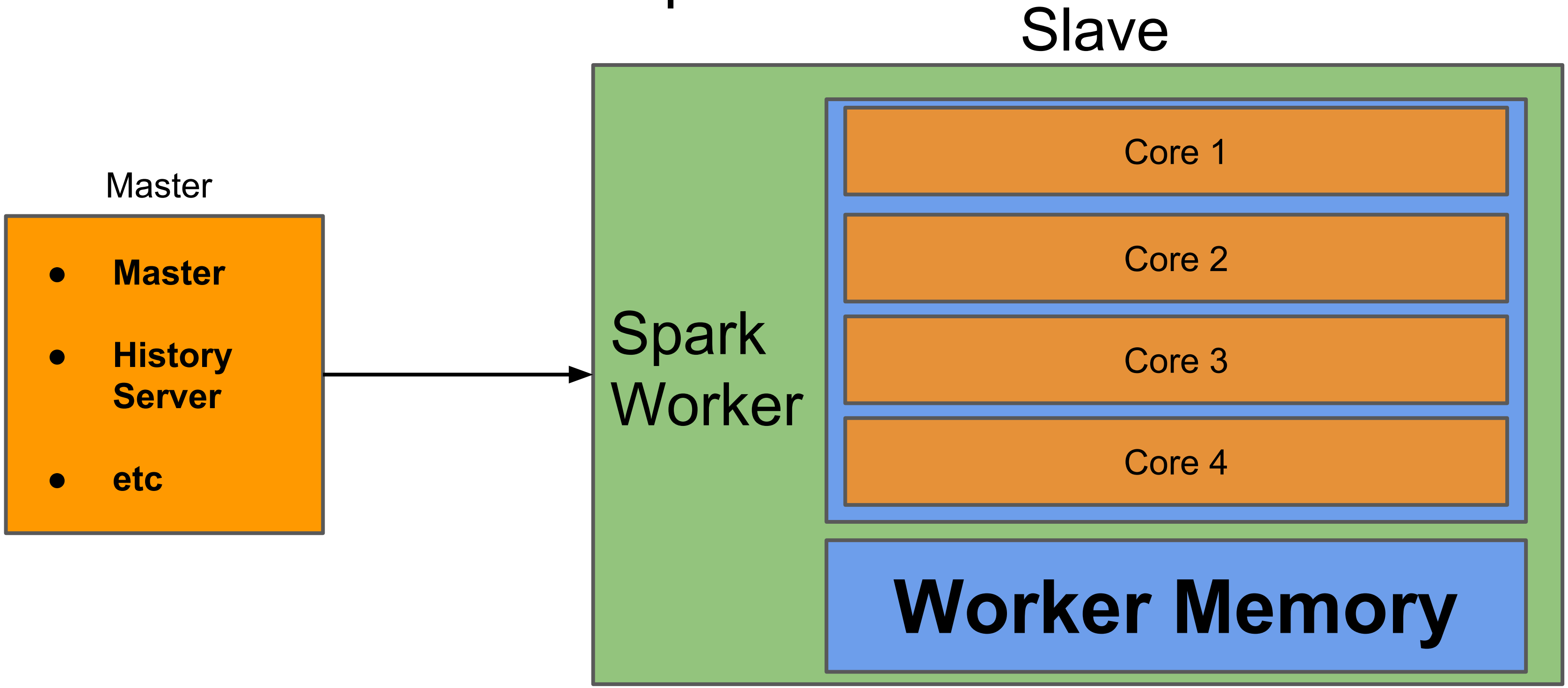
Standalone Deployment Tale

- You can read in more elaboration in the next blog post
- This hack was made mostly to handle a better distribution of data
  - Trying to be more predictable

Spark Standalone Cluster - Architecture

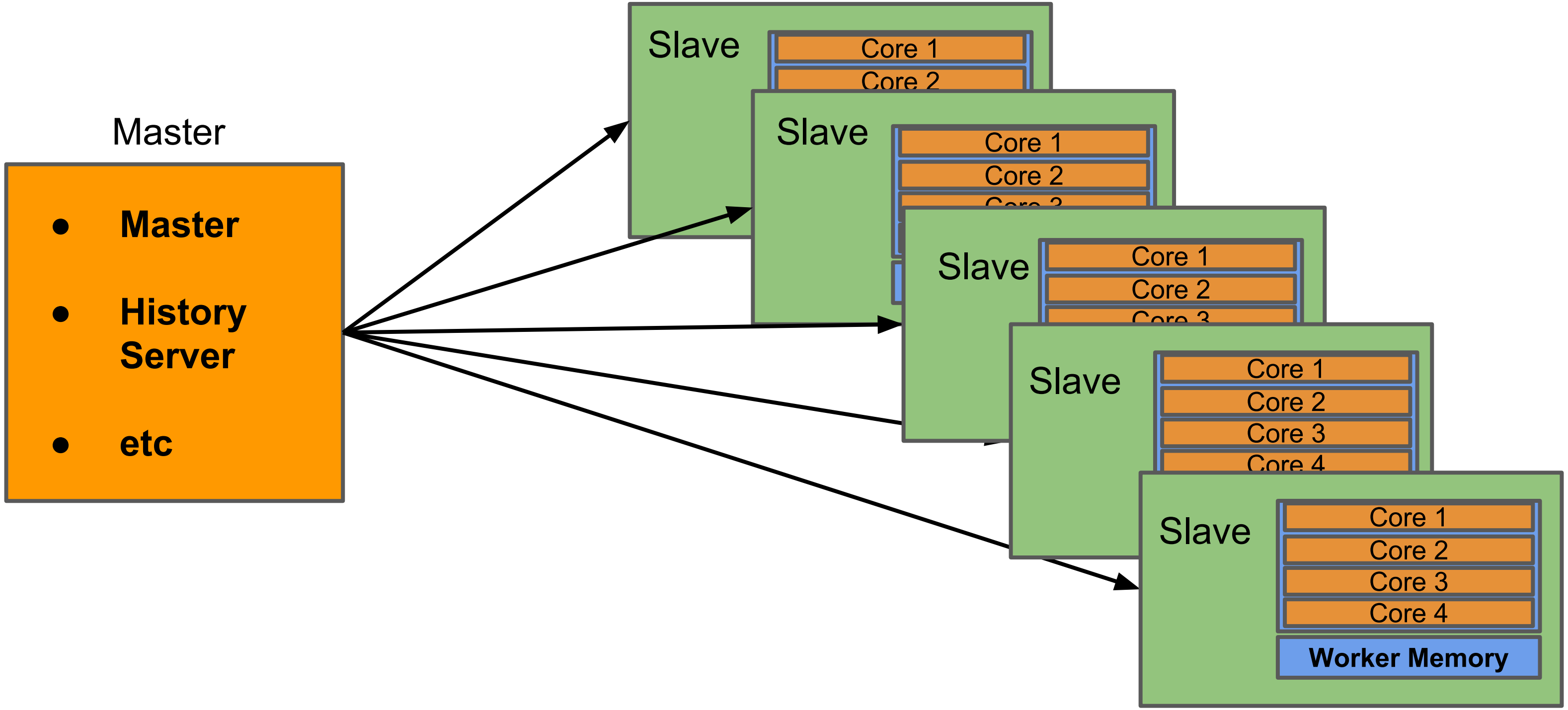


Slave Structure - Deep Dive

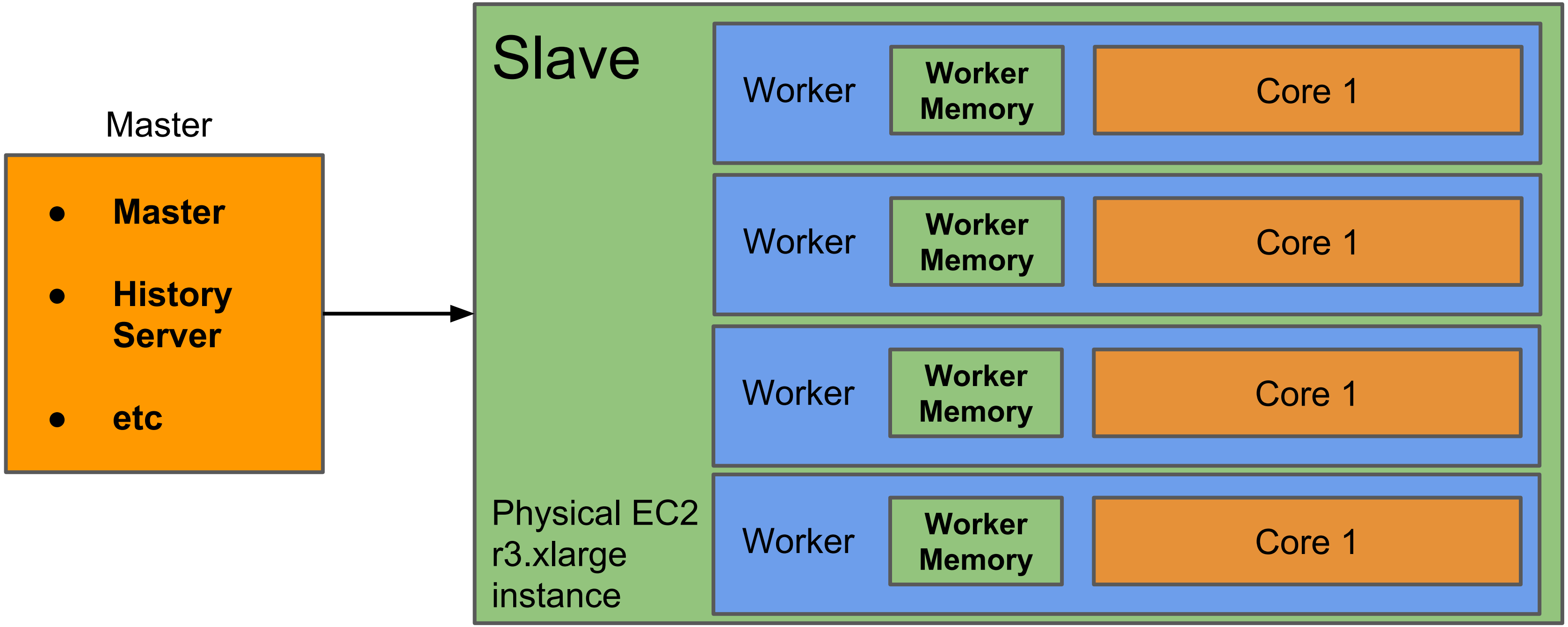


Panorays

Spark Standalone Cluster - Architecture



“Hacked” Slave Structure - Deep Dive



Master

- Master
- History Server
- etc

Slave

Physical EC2 r3.xlarge instance

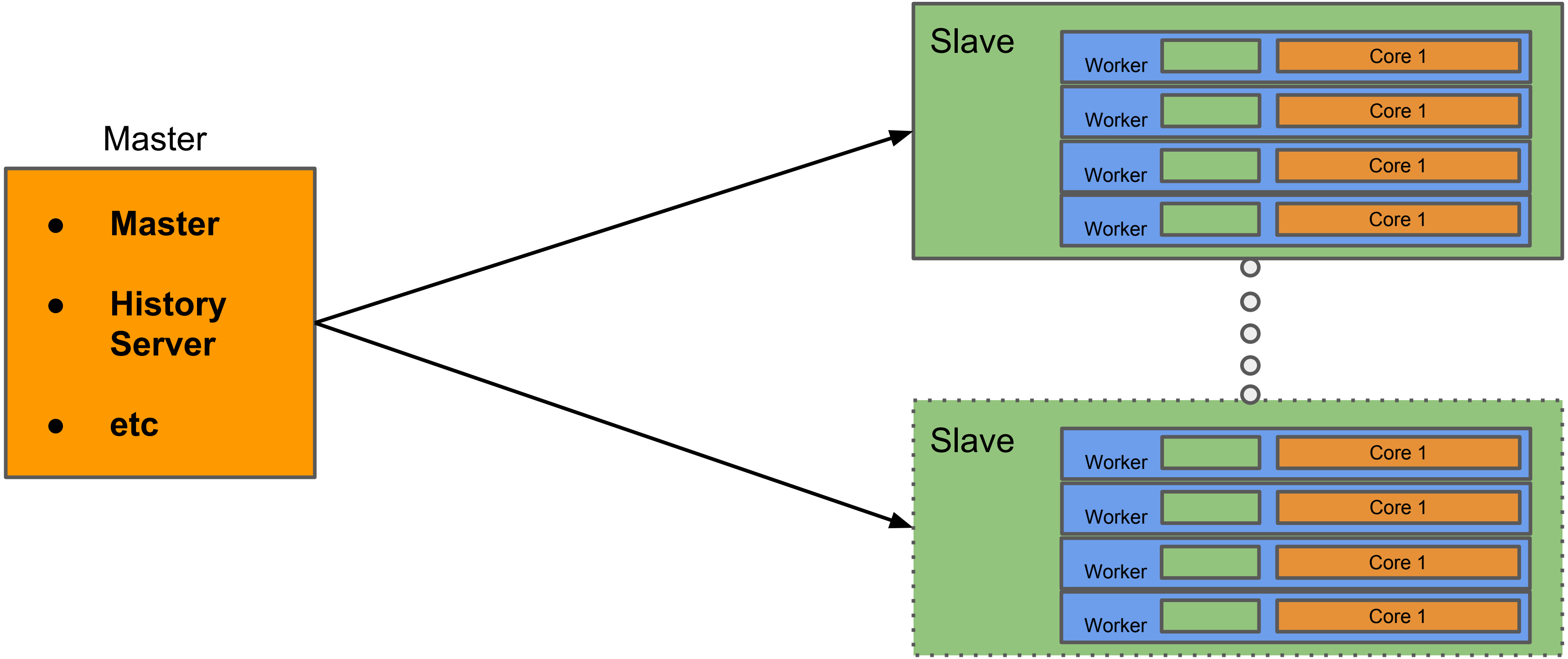
Worker | Worker Memory | Core 1

Worker | Worker Memory | Core 1

Worker | Worker Memory | Core 1

Worker | Worker Memory | Core 1

“Hacked“ Spark Standalone Cluster - Architecture



Panorays

Panorays

Mind the Attack Surface