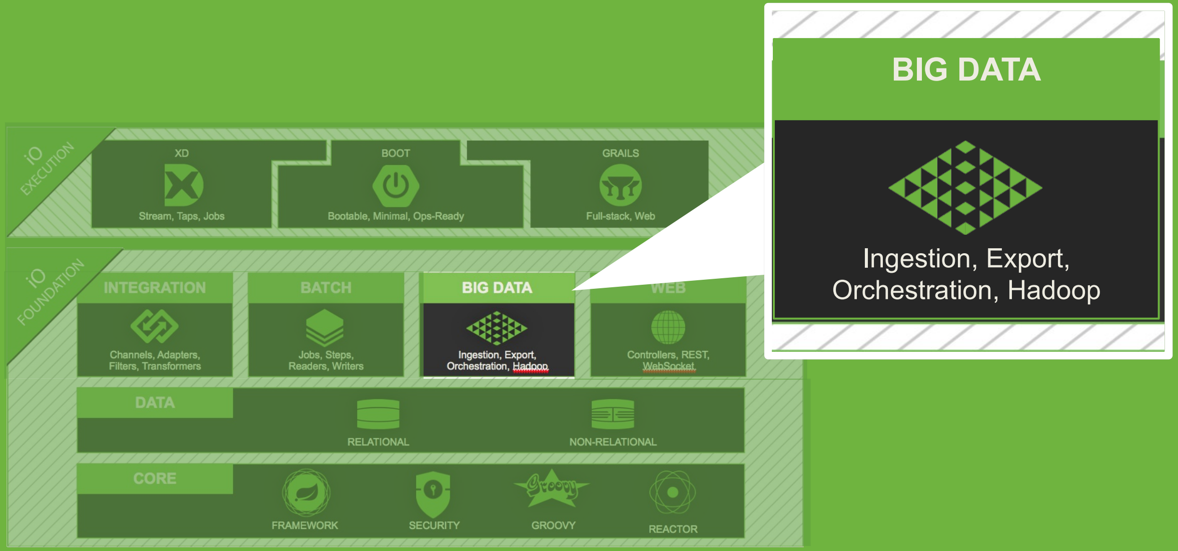
# Getting started with Spring Data and Apache Hadoop

Thomas Risberg
Janne Valkealahti

## Spring IO FOUNDATION:

# Spring for Apache Hadoop


iO EXECUTION
XD
Stream, Taps, Jobs
BOOT
Bootable, Minimal, Ops-Ready
GRAILS
Full-stack, Web
iO FOUNDATION
INTEGRATION
Channels, Adapters, Filters, Transformers
BATCH
Jobs, Steps, Readers, Writers
BIG DATA
Ingestion, Export, Orchestration, Hadoop
WEB
Controllers, REST, WebSocket
DATA
RELATIONAL
NON-RELATIONAL
CORE
FRAMEWORK
SECURITY
GROOVY
REACTOR
BIG DATA
Ingestion, Export, Orchestration, Hadoop

# Just a Note

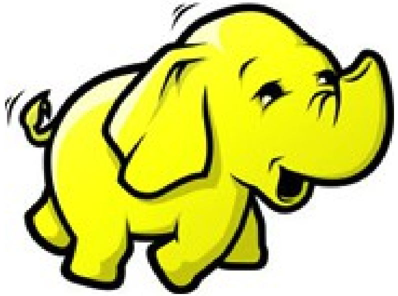
**in the making of this presentation**

# About us ...

- Thomas
  - Working on the Spring Data engineering team at Pivotal
  - Joined Spring Framework team in 2003 working on JDBC support
  - co-author of “Professional Java Development with Spring Framework” from Wrox 2005 and “Spring Data” book from O'Reilly 2012
- Janne
  - Member of the Spring Data engineering team at Pivotal
  - contributes to Spring for Apache Hadoop and Spring XD projects
  - Previously Consultant for SpringSource vFabric team
  - 10-year career at a biggest online stock broker in Finland

# About Apache Hadoop

- An Apache Project
- Modeled after Google File System and Map Reduce
- Provides:
  - Distributed file system
  - Map Reduce
  - General resource managemt for workloads with YARN (Hadoop v2)
  - Started as open source project at Yahoo and Facebook
  - Initial development by Doug Cutting and Mike Cafarella
- We hope you attended “Hadoop - Just the Basics for Big Data Rookies” with Adam Shook earlier; we will not cover Hadoop itself in detail today



# About Spring Data

- Bring classic Spring value propositions to new data technologies
  - Productivity
  - Programming model consistency
- Support for new data technologies like NOSQL databases, Hadoop, SOLR, ElasticSearch, Querydsl
- Many entry points to use
  - Low level data access and Opinionated APIs
  - Repository Support and Object Mapping
  - Guidance

# Hadoop trends

- Many organizations are currently using or evaluating Hadoop
- One common usage is Hadoop HDFS as a "data-lake" landing zone
  - *Collect all data and store it in HDFS, worry about analysis later*
- Many companies are now looking to YARN for running non-map-reduce workloads on a Hadoop cluster
- Lots of interest in using SQL on top of HDFS data – Hive/Stinger, Impala and HAWQ

Image credit: Myriam Maul http://www.sxc.hu/photo/722484

# What we will be talking about today

- Getting started with:
  - running **Apache Hadoop** for development/testing
  - writing map-reduce jobs for **Apache Hadoop**
  - writing apps using **Spring for Apache Hadoop**
  - writing apps with **Spring Yarn**

# Getting Started with Hadoop

… can seem like an uphill struggle at times

… let's take one step at a time

Some ways to get started

1. Standalone Mode
2. Pre-configured VM
3. Pseudo-distributed cluster

Image credit: Steve Garvie http://www.flickr.com/photos/rainbirder/5031379872/

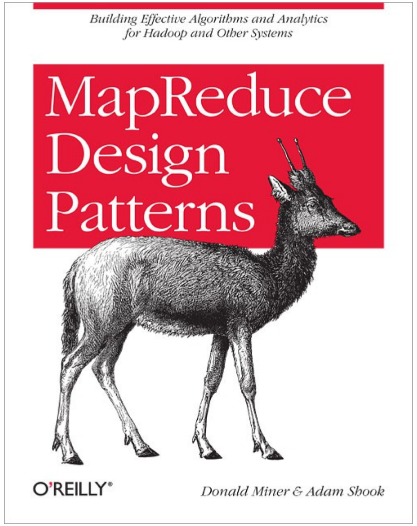
# Hadoop in Standalone Mode

- Download Apache Hadoop from
  - http://hadoop.apache.org/releases.html#Download
- Create a directory and unzip the download, set PATH and test

```
~$ mkdir ~/test
~$ cd ~/test
~/test$ tar xvzf ~/Downloads/hadoop-1.2.1-bin.tar.gz
~/test$ export HADOOP_INSTALL=~/test/hadoop-1.2.1
~/test$ export PATH=$PATH:$HADOOP_INSTALL/bin:$HADOOP_INSTALL/sbin
~/test$ export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64
~/test$ hadoop version
Hadoop 1.2.1
Subversion https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r 1503152
Compiled by mattf on Mon Jul 22 15:23:09 PDT 2013
From source with checksum 6923c86528809c4e7e6f493b6b413a9a
This command was run using /home/trisberg/test/hadoop-1.2.1/hadoop-core-1.2.1.jar
```

# Our first Map Reduce job – TweetHashTagCounter ...

- We will count the number of occurrences of #hashtags in a collection of tweeets collected during the 2013 NBA Finals
- Based on “Word Count” example from “MapReduce Design Patterns” book
- We need
  - a Mapper class
  - and a Reducer class
  - and a driver class

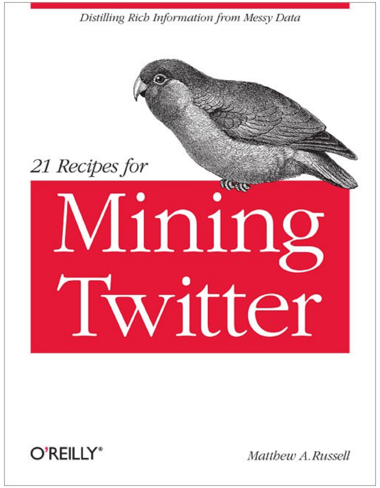

Building Effective Algorithms and Analytics for Hadoop and Other Systems
MapReduce Design Patterns
O'REILLY
Donald Miner & Adam Shook

Book: MapReduce Design Patterns, Miner & Shook http://shop.oreilly.com/product/0636920025122.do

# Some input data – tweets captured during NBA finals

```
{
        "id": 348115421360164864,
        "text": "RT @NBA: The Best of the 2013 #NBAFinals set to 'Radioactive' by Imagine Dragons! http://t.co/EA198meYpC",
        "createdAt": 1371832158000,
        "fromUser": "I_Nikki_I",
...
        "retweetedStatus": {
                "id": 348111916452950016,
                "text": "The Best of the 2013 #NBAFinals set to 'Radioactive' by Imagine Dragons! http://t.co/EA198meYpC",
                "createdAt": 1371831323000,
                "fromUser": "NBA",
...
        },
...
        "entities": {
                "hashTags": [{
                        "text": "NBAFinals",
                        "indices": [30, 40]
                }]
        },
        "retweet": true
}
```

*The data file has the entire JSON document for each tweet on a single line*



Distilling Rich Information from Messy Data

21 Recipes for

Mining Twitter

O'REILLY

Matthew A. Russell

Book: 21 Recipes for Mining Twitter, Matthew A. Russell   http://shop.oreilly.com/product/0636920018261.do

# Our first Mapper class

```
public class TweetCountMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable ONE = new IntWritable(1);
    private final ObjectMapper mapper = new ObjectMapper(new JsonFactory());

    @Override
    protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {

        Map<String, Object> tweet = mapper.readValue(value.toString(),
                new TypeReference<HashMap<String,  Object>>(){});
        Map<String, Object> entities = (Map<String, Object>) tweet.get("entities");
        List<Map<String, Object>> hashTagEntries = null;
        if (entities != null) {
            hashTagEntries = (List<Map<String, Object>>) entities.get("hashTags");
        }
        if (hashTagEntries != null && hashTagEntries.size() > 0) {
            for (Map<String, Object> hashTagEntry : hashTagEntries) {
                String hashTag = hashTagEntry.get("text").toString();
                context.write(new Text(hashTag), ONE);
            }
        }
    }
}
```

# Our first Reducer class

```
public class IntSumReducer extends Reducer<Text, IntWritable, Text, IntWritable>{

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable value : values) {
            sum += value.get();
        }
        context.write(key, new IntWritable(sum));
    }

}
```

# Our first Driver class

```java
public class TweetHashTagCounter {

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();
        String[] myArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        if (myArgs.length != 2) {
            System.err.println("Usage: TweetHashTagCounter <input path> <output path>");
            System.exit(-1);
        }
        Job job =  Job.getInstance(conf, "Tweet Hash Tag Counter");
        job.setJarByClass(TweetHashTagCounter.class);

        FileInputFormat.addInputPath(job, new Path(myArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(myArgs[1]));

        job.setMapperClass(TweetCountMapper.class);
        job.setReducerClass(IntSumReducer.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}
```

# Need to build the app – our pom.xml

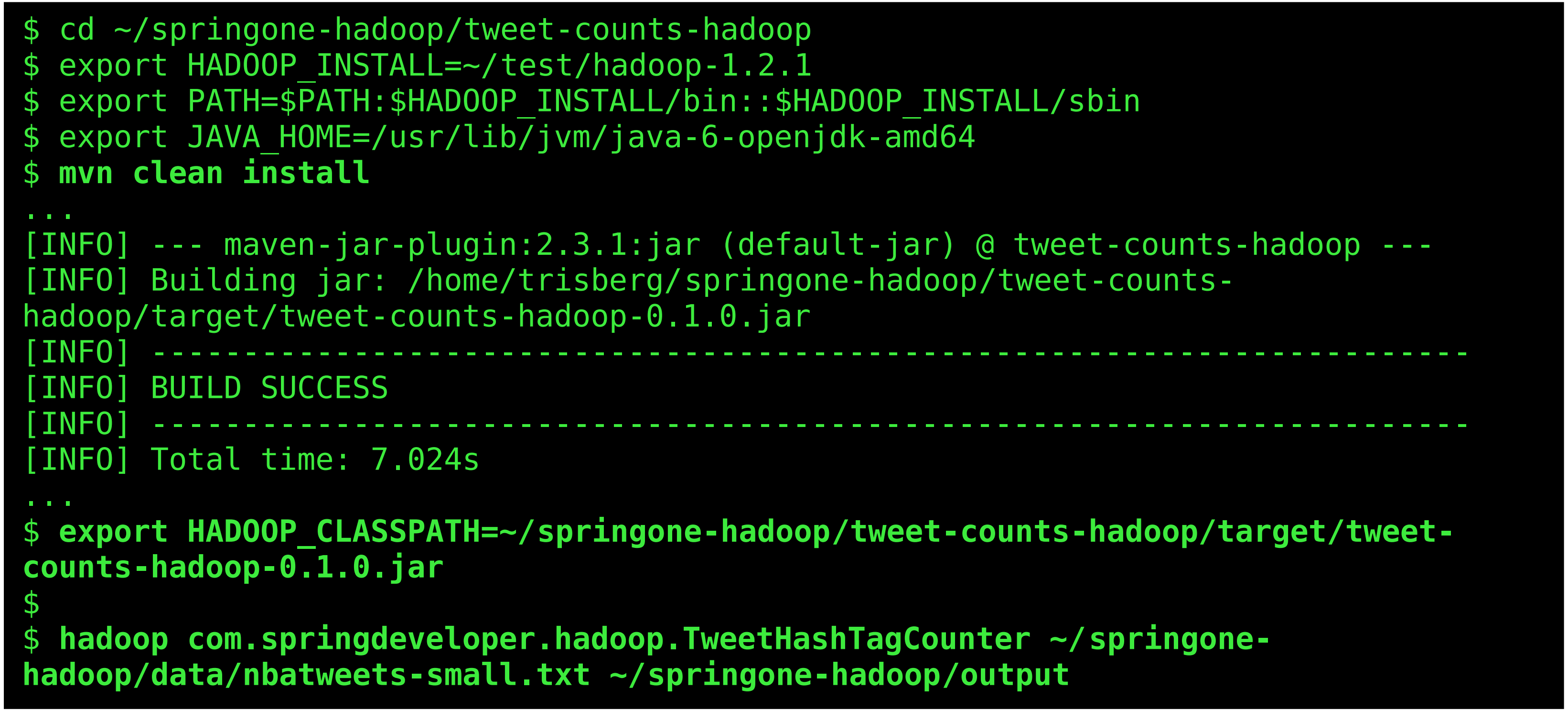
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
...
     <groupId>com.springdeveloper.hadoop</groupId>
     <artifactId>tweet-counts-hadoop</artifactId>
     <version>0.1.0</version>
     <packaging>jar</packaging>
     <name>Tweet Counts</name>
...
     <properties>
          <hadoop.version>1.2.1</hadoop.version>
     </properties>
...
     <dependencies>
          <dependency>
               <groupId>org.apache.hadoop</groupId>
               <artifactId>hadoop-core</artifactId>
               <version>${hadoop.version}</version>
          </dependency>
...
     </dependencies>
```

# Example code repository

- All code for the Hadoop HDFS and Map Reduce examples can be downloaded from GitHub
- Repository
  - https://github.com/trisberg/springone-hadoop.git

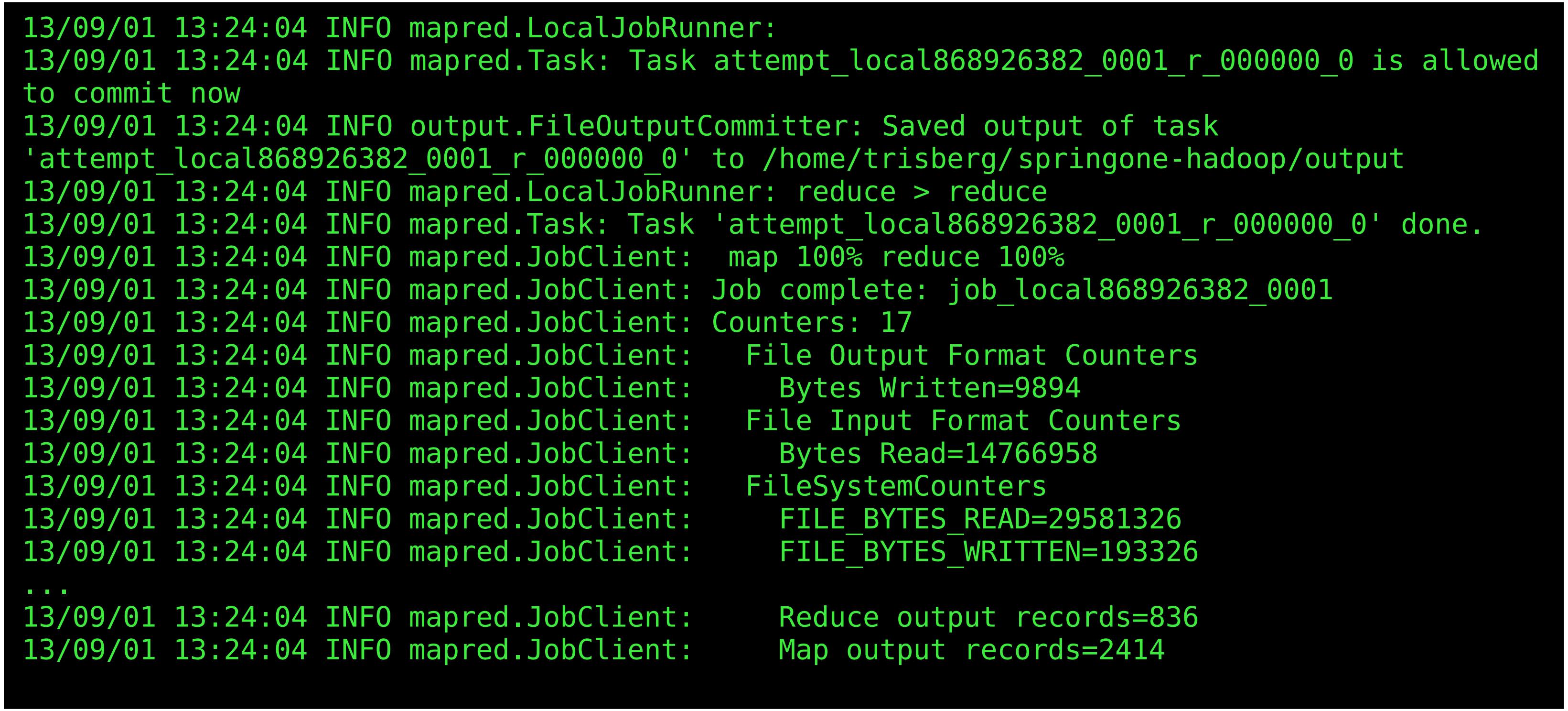
```
$ cd ~
$ git clone https://github.com/trisberg/springone-hadoop.git
$ cd ~/springone-hadoop
```

# Let's build and run the app



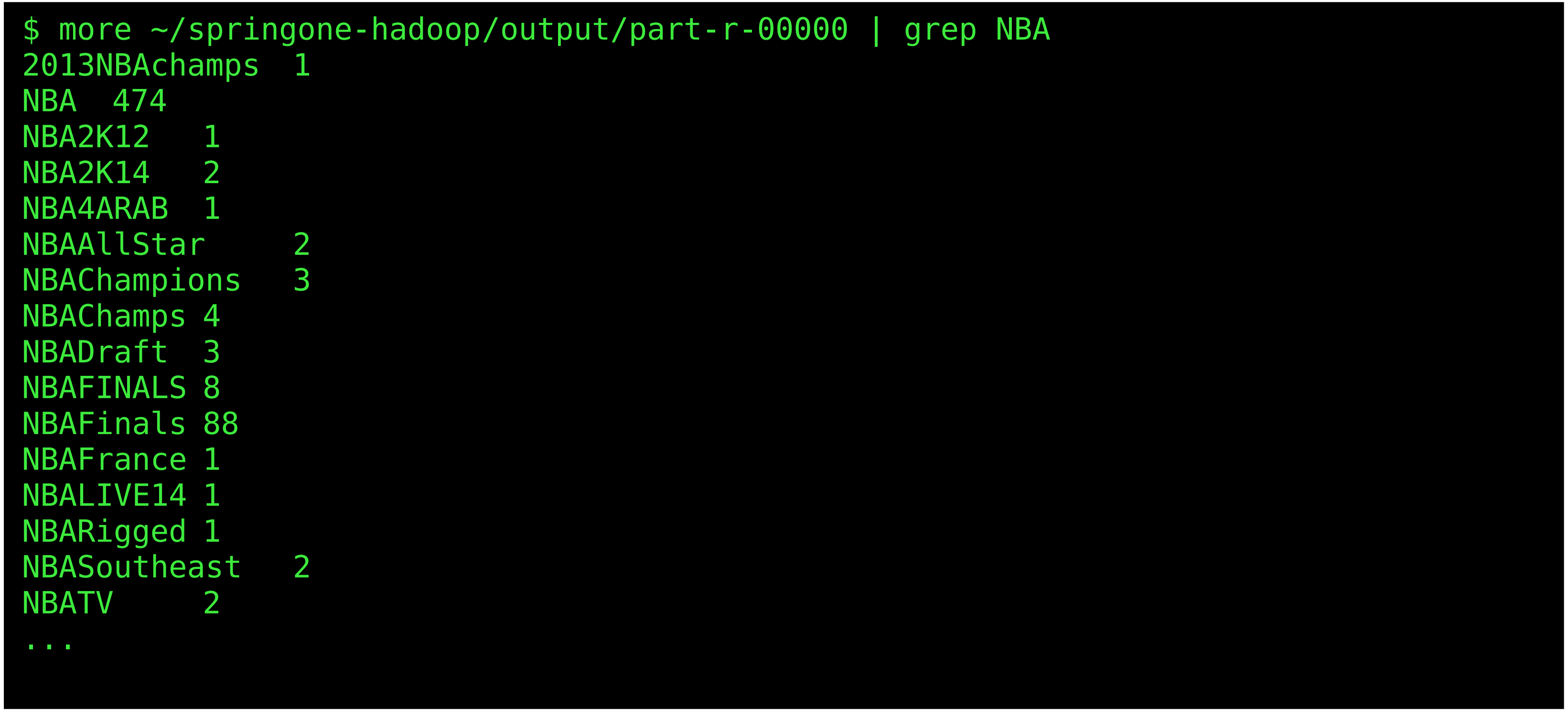
```
$ cd ~/springone-hadoop/tweet-counts-hadoop
$ export HADOOP_INSTALL=~/test/hadoop-1.2.1
$ export PATH=$PATH:$HADOOP_INSTALL/bin::$HADOOP_INSTALL/sbin
$ export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64
$ mvn clean install
...
[INFO] --- maven-jar-plugin:2.3.1:jar (default-jar) @ tweet-counts-hadoop ---
[INFO] Building jar: /home/trisberg/springone-hadoop/tweet-counts-
hadoop/target/tweet-counts-hadoop-0.1.0.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 7.024s
...
$ export HADOOP_CLASSPATH=~/springone-hadoop/tweet-counts-hadoop/target/tweet-
counts-hadoop-0.1.0.jar
$
$ hadoop com.springdeveloper.hadoop.TweetHashTagCounter ~/springone-
hadoop/data/nbatweets-small.txt ~/springone-hadoop/output
```

# App log output ...



```
13/09/01 13:24:04 INFO mapred.LocalJobRunner:
13/09/01 13:24:04 INFO mapred.Task: Task attempt_local868926382_0001_r_000000_0 is allowed
to commit now
13/09/01 13:24:04 INFO output.FileOutputCommitter: Saved output of task
'attempt_local868926382_0001_r_000000_0' to /home/trisberg/springone-hadoop/output
13/09/01 13:24:04 INFO mapred.LocalJobRunner: reduce > reduce
13/09/01 13:24:04 INFO mapred.Task: Task 'attempt_local868926382_0001_r_000000_0' done.
13/09/01 13:24:04 INFO mapred.JobClient:  map 100% reduce 100%
13/09/01 13:24:04 INFO mapred.JobClient: Job complete: job_local868926382_0001
13/09/01 13:24:04 INFO mapred.JobClient: Counters: 17
13/09/01 13:24:04 INFO mapred.JobClient:   File Output Format Counters
13/09/01 13:24:04 INFO mapred.JobClient:     Bytes Written=9894
13/09/01 13:24:04 INFO mapred.JobClient:   File Input Format Counters
13/09/01 13:24:04 INFO mapred.JobClient:     Bytes Read=14766958
13/09/01 13:24:04 INFO mapred.JobClient:   FileSystemCounters
13/09/01 13:24:04 INFO mapred.JobClient:     FILE_BYTES_READ=29581326
13/09/01 13:24:04 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=193326
...
13/09/01 13:24:04 INFO mapred.JobClient:     Reduce output records=836
13/09/01 13:24:04 INFO mapred.JobClient:     Map output records=2414
```

# And the results ...



```
$ more ~/springone-hadoop/output/part-r-00000 | grep NBA
2013NBAchamps  1
NBA  474
NBA2K12   1
NBA2K14   2
NBA4ARAB  1
NBAAllStar     2
NBAChampions   3
NBAChamps 4
NBADraft  3
NBAFINALS 8
NBAFinals 88
NBAFrance 1
NBALIVE14 1
NBARigged 1
NBASoutheast   2
NBATV     2
...
```

# Developer observations on Hadoop

- For Spring developers, Hadoop has a fairly poor out-of-the-box programming model
- Lots of low-level configuration and exception handling code
- Non trivial applications often become a collection of scripts calling Hadoop command line applications
- Spring aims to simplify development for Hadoop applications
  - Leverage Spring's configuration features in addition to several Spring eco-system projects

# Spring for Apache Hadoop

> “Spring for Apache Hadoop provides extensions to Spring, Spring Batch, and Spring Integration to build manageable and robust pipeline solutions around Hadoop.

# Spring for Apache Hadoop – Features

- Consistent programming and declarative configuration model
  - Create, configure, and parametrize Hadoop connectivity and all job types
  - Environment profiles – easily move application from dev to qa to production
- Developer productivity
  - Create well-formed applications, not spaghetti script applications
  - Simplify HDFS access and FsShell API with support for JVM scripting
  - Runner classes for MR/Pig/Hive/Cascading for small workflows
  - Helper "Template" classes for Pig/Hive/HBase

# Spring for Apache Hadoop – Use Cases

- Apply across a wide range of use cases
  - Ingest: Events/JDBC/NoSQL/Files to HDFS
  - Orchestrate: Hadoop Jobs
  - Export: HDFS to JDBC/NoSQL
- Spring Integration and Spring Batch make this possible

# Spring for Apache Hadoop – Status

- 1.0 GA in February 2013 – supported up to Hadoop 1.0.4
- 1.0.1 GA last week – supports all Hadoop 1.x stable, 2.0-alpha and 2.1-beta
- Default is Apache Hadoop 1.2.1 stable
- Distribution specific “flavors” via a suffix on version:

| | |
|---|---|
| – 1.0.1.RELEASE-cdh4 | Cloudera CDH 4.3.1 |
| – 1.0.1.RELEASE-hdp13 | Hortonworks HDP 1.3 |
| – 1.0.1.RELEASE-phd1 | Pivotal HD 1.0 |
| – 1.0.1.RELEASE-hadoop21 | Hadoop 2.1.0-beta |

# Spring for Apache Hadoop – Future

- New structure for 2.0
  - New sub-projects
    - Core M/R, FSShell, Hive, Pig etc. and basic configuration
    - Batch is separate with separate namespace
    - Cascading separate with separate namespace
    - Test sub-project for integration testing
    - Adding **spring-yarn** sub-project for 2.0 based builds
  - Just released first 2.0.0.M1 milestone release

# Core and Batch

- Running Map Reduce jobs
- HDFS shell scripting
- Running Pig and Hive scripts
- Configuration
- Configuring batch jobs with Spring Batch

**Examples built using spring-data-hadoop 1.0.1.RELEASE**

# Hadoop Configuring M/R

- Standard Hadoop APIs

```java
Configuration conf = new Configuration();
Job job =  Job.getInstance(conf, "Tweet Hash Tag Counter");
job.setJarByClass(TweetHashTagCounter.class);

FileInputFormat.addInputPath(job, new Path(myArgs[0]));
FileOutputFormat.setOutputPath(job, new Path(myArgs[1]));

job.setMapperClass(TweetCountMapper.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

System.exit(job.waitForCompletion(true) ? 0 : 1);
```

# Configuring Hadoop with Spring

```
<context:property-placeholder location="hadoop-dev.properties"/>

<hdp:configuration>
  fs.default.name=${hd.fs}
  mapred.job.tracker=${hd.jt}
</hdp:configuration>

<hdp:job id="word-count-job"
         input-path="${input.path}"
         output-path="${output.path}"
         jar="hadoop-examples.jar"
         mapper="examples.WordCount.WordMapper"
         reducer="examples.WordCount.IntSumReducer"/>

<hdp:job-runner id="runner" job-ref="word-count-job"
                            run-at-startup="true" />
```

applicationContext.xml

Automatically determines
Output key and class

```
input.path=/wc/input/
output.path=/wc/word/
hd.fs=hdfs://localhost:8020
hd.jt=localhost:8021
```

hadoop-dev.properties

# Injecting Jobs

- Use DI to obtain reference to Hadoop Job
  - Perform additional runtime configuration and submit

```
public class WordService {

  @Autowired
  private Job mapReduceJob;

  public void processWords() {
    mapReduceJob.submit();
  }
}
```

# Streaming Jobs and Environment Configuration

```
bin/hadoop jar hadoop-streaming.jar \
    -input /wc/input -output /wc/output \
    -mapper /bin/cat -reducer /bin/wc \
    -files stopwords.txt
```

```
<context:property-placeholder location="hadoop-${env}.properties"/>

<hdp:streaming id="wc" input-path="${input}" output-path="${output}"
  mapper="${cat}" reducer="${wc}"
  files="classpath:stopwords.txt">
</hdp:streaming>
```

```
hadoop-dev.properties
  input.path=/wc/input/
  output.path=/wc/word/
  hd.fs=hdfs://localhost:9000
```

```
hadoop-qa.properties
  input.path=/gutenberg/input/
  output.path=/gutenberg/word/
  hd.fs=hdfs://darwin:9000
```

```
Java -Denv=dev -jar SpringLauncher.jar applicationContext.xml
```

# HDFS and Hadoop Shell as APIs

- Access all “bin/hadoop fs” commands through Spring’s FsShell helper class
  - mkdir, chmod, test

```
class MyScript {
  @Autowired FsShell fsh;

  @PostConstruct void init() {
    String outputDir = "/data/output";
    if (fsShell.test(outputDir)) {
     fsShell.rmr(outputDir);
   }
  }
}
```

# HDFS and Hadoop Shell as APIs

copy-files.groovy

```
// use the shell (made available under variable fsh)

if (!fsh.test(inputDir)) {
    fsh.mkdir(inputDir);
    fsh.copyFromLocal(sourceFile, inputDir);
    fsh.chmod(700, inputDir)
}
if (fsh.test(outputDir)) {
    fsh.rmr(outputDir)
}
```

# HDFS and Hadoop Shell as APIs

- Reference script and supply variables in application configuration

app-context.xml

```
<script id="setupScript" location="copy-files.groovy">
  <property name="inputDir" value="${wordcount.input.path}"/>
  <property name="outputDir" value="${wordcount.output.path}"/>
  <property name=“sourceFile“ value="${localSourceFile}"/>
</script>
```

# Small workflows

- Often need the following steps
  - Execute HDFS operations before job
  - Run MapReduce Job
  - Execute HDFS operations after job completes
- Spring’s JobRunner helper class sequences these steps
  - Can reference multiple scripts with comma delimited names

```
<hdp:job-runner id="runner" run-at-startup="true"
                pre-action="setupScript"
                job="wordcountJob‶
                post-action=‶tearDownScript"/>
```

# Runner classes

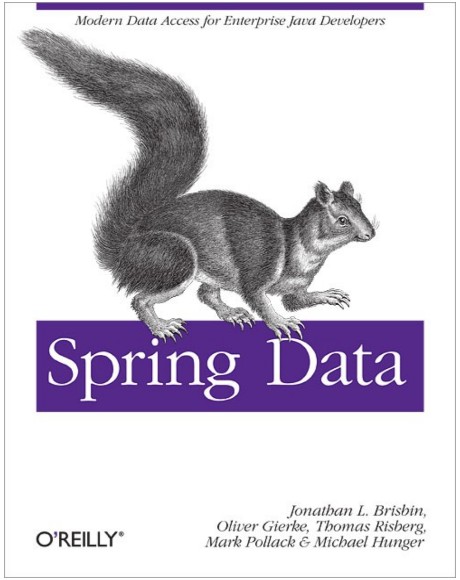
- Similar runner classes available for Hive and Pig
- Implement JDK callable interface
- Easy to schedule for simple needs using Spring

```
<hdp:job-runner id="runner“ run-at-startup=“false"
                            pre-action="setupScript“
                            job="wordcountJob“
                            post-action=“tearDownScript"/>

<task:scheduled-tasks>
  <task:scheduled ref="runner" method="call" cron="3/30 * * * * ?"/>
</task:scheduled-tasks>
```

- Can later ‘graduate’ to use Spring Batch for more complex workflows
  - Start simple and grow, reusing existing configuration

# Our first Spring Configured Map Reduce job

- We will reuse the TweetHashTagCounter example
- Loosely based on “Spring Word Count” example from “Spring Data” book
- We need an application context
  - and a properties file
  - and a driver class


Modern Data Access for Enterprise Java Developers
Spring Data
O'REILLY
Jonathan L. Brisbin, Oliver Gierke, Thomas Risberg, Mark Pollack & Michael Hunger

https://github.com/trisberg/springone-hadoop.git

Book: Spring Data, *Modern Data Access for Enterprise Java* http://shop.oreilly.com/product/0636920024767.do

# Our application context

```
<context:property-placeholder location="hadoop.properties"/>

<configuration>
     fs.default.name=${hd.fs}
     mapred.job.tracker=${hd.jt}
</configuration>

<job id="tweetCountJob"
     input-path="${tweetcount.input.path}"
     output-path="${tweetcount.output.path}"
     libs="file:${app.repo}/tweet-counts-hadoop-0.1.0.jar"
     mapper="com.springdeveloper.hadoop.TweetCountMapper"
     reducer="com.springdeveloper.hadoop.IntSumReducer"/>

<script id="setupScript" location="file-prep.groovy">
     <property name="localSourceFile" value="${app.home}/${localSourceFile}"/>
     <property name="inputDir" value="${tweetcount.input.path}"/>
     <property name="outputDir" value="${tweetcount.output.path}"/>
</script>

<job-runner id="runner" run-at-startup="true"
     pre-action="setupScript"
     job-ref="tweetCountJob" />
```

```
hd.fs=hdfs://sandbox:8020
hd.jt=sandbox:50300

tweetcount.input.path=/tweets/input
tweetcount.output.path=/tweets/results
localSourceFile=data/nbatweets-small.txt
```

# Our Spring Driver class

```
public class TweetCount {

    private static final Log log = LogFactory.getLog(TweetCount.class);

    public static void main(String[] args) throws Exception {
        AbstractApplicationContext context = new ClassPathXmlApplicationContext(
                "/META-INF/spring/application-context.xml", TweetCount.class);
        log.info("TweetCount Application Running");
        context.registerShutdownHook();
    }
}
```

# A pom.xml to build and run the app – part 1

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
...
      <properties>
            <spring.framework.version>3.2.4.RELEASE</spring.framework.version>
            <spring.hadoop.version>1.0.1.RELEASE</spring.hadoop.version>
            </properties>
      <dependencies>
            <dependency>
                  <groupId>com.springdeveloper.hadoop</groupId>
                  <artifactId>tweet-counts-hadoop</artifactId>
                  <version>0.1.0</version>
                  <scope>runtime</scope>
            </dependency>
            <dependency>
                  <groupId>org.springframework</groupId>
                  <artifactId>spring-context-support</artifactId>
                  <version>${spring.framework.version}</version>
            </dependency>
            <dependency>
                  <groupId>org.springframework.data</groupId>
                  <artifactId>spring-data-hadoop</artifactId>
                  <version>${spring.hadoop.version}</version>
            </dependency>
...
```

# A pom.xml to build and run the app – part 2

```
...
            <dependency>
                  <groupId>org.codehaus.groovy</groupId>
                  <artifactId>groovy</artifactId>
                  <version>1.8.5</version>
                  <scope>runtime</scope>
            </dependency>
            <dependency>
                  <groupId>log4j</groupId>
                  <artifactId>log4j</artifactId>
                  <version>1.2.14</version>
            </dependency>
...
      </dependencies>
      <plugins>
            <plugin>
                  <groupId>org.codehaus.mojo</groupId>
                  <artifactId>appassembler-maven-plugin</artifactId>
                  <version>1.2.2</version>
            </plugin>
...
      </plugins>
```

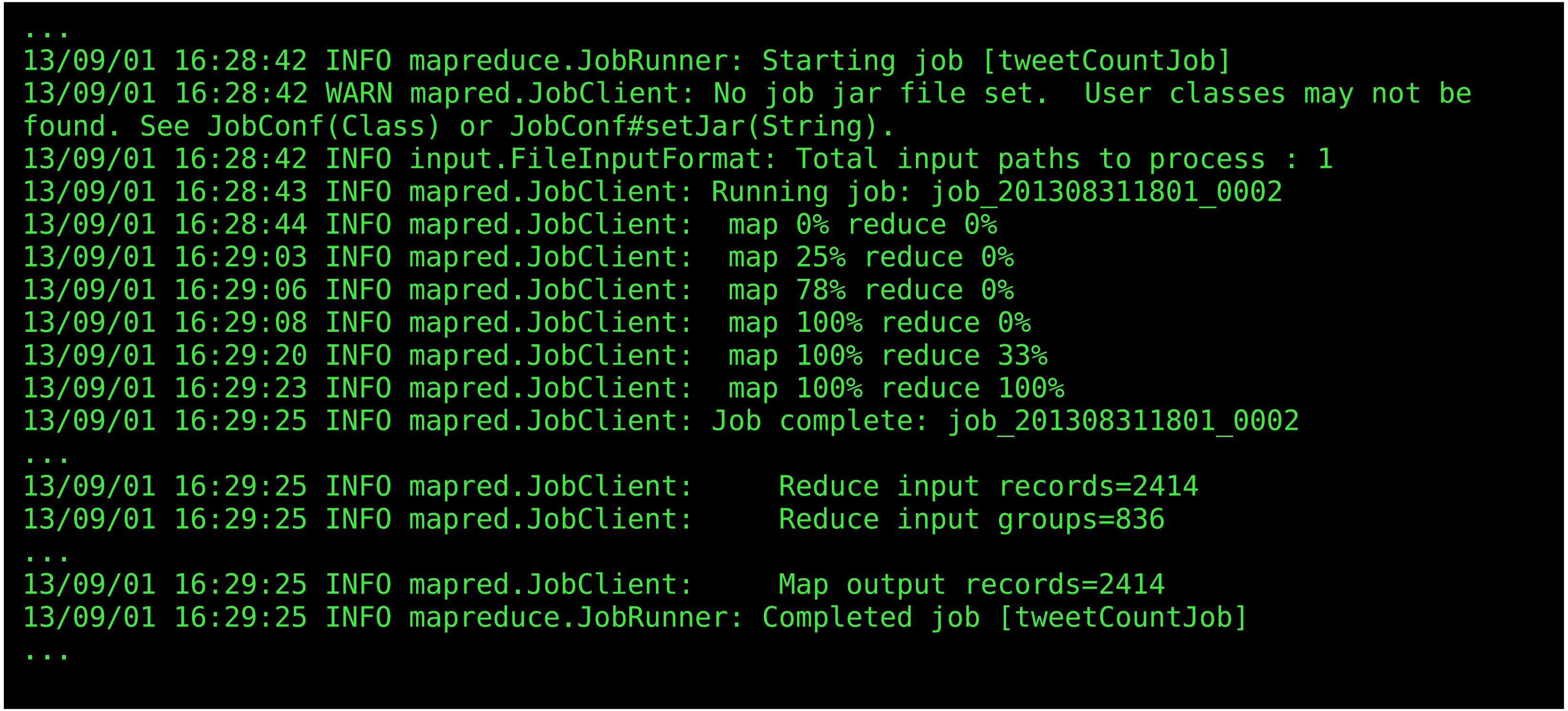
# Testing with Hadoop – using a pre-configured VM

- VMs “ready to run” - most distro companies provide one:
  - Hortonworks Sandbox HDP 1.3 and HDP 2.0
  - Pivotal HD 1.0 Single Node VM
  - Cloudera Quickstart CDH4
- Which one to use? Depends on what your company uses.
- If starting from scratch Hortonworks Sandbox HDP 1.3 is based on Hadoop 1.2.0 and a good place to start …
  - HDFS configured to listen on the VM network making it easy to connect from host system
  - Uses only 2GB of memory making it easy to use on laptops
  - Compatible with Spring for Apache Hadoop 1.0.1.RELEASE and its transitive dependencies

# Let's build and run the Spring app

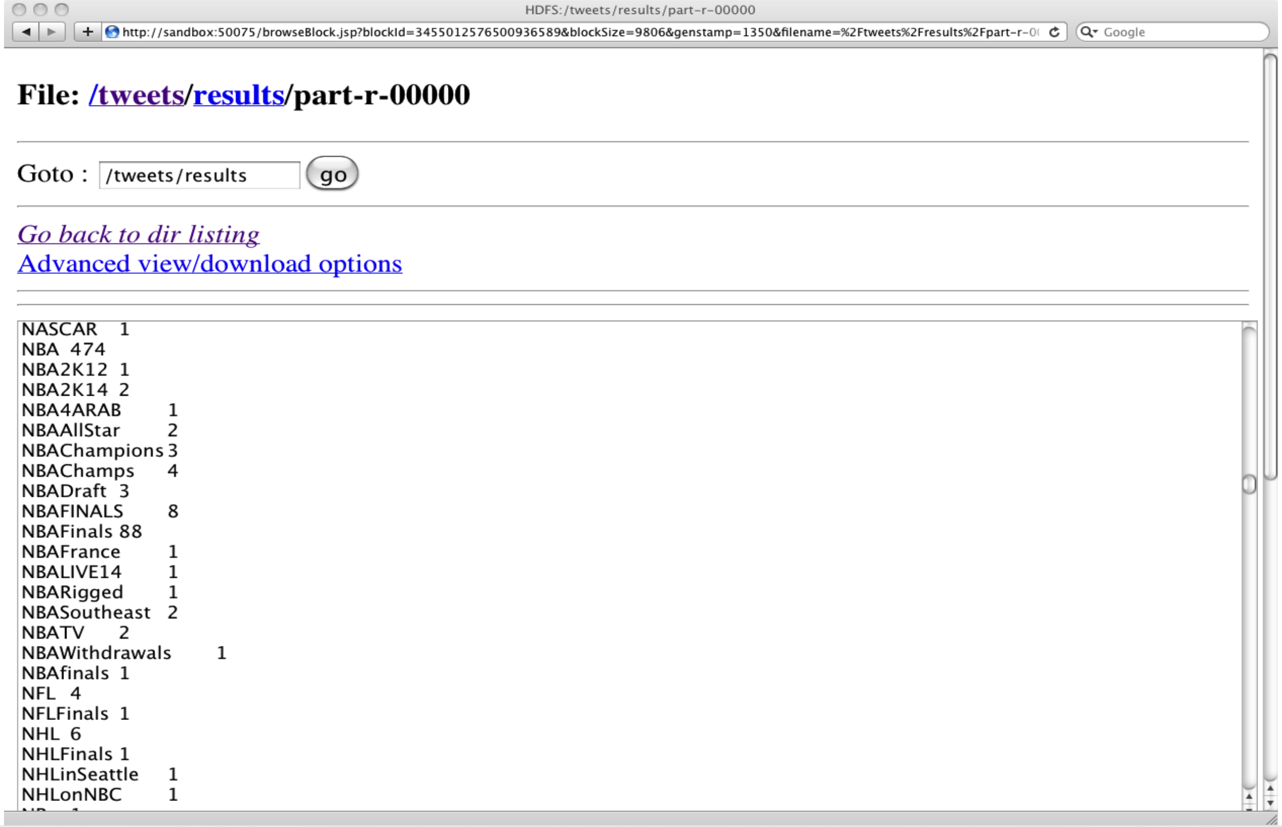
```
$ cd ~/springone-hadoop/tweet-counts-spring
$ mvn clean package
...
[INFO] --- maven-antrun-plugin:1.3:run (config) @ tweet-counts-spring ---
[INFO] Executing tasks
     [copy] Copying 1 file to /home/trisberg/springone-hadoop/tweet-counts-
spring/target/appassembler/data
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 11.710s
...
$ sh ./target/appassembler/bin/tweetcount
```

# App log output ...



```
...
13/09/01 16:28:42 INFO mapreduce.JobRunner: Starting job [tweetCountJob]
13/09/01 16:28:42 WARN mapred.JobClient: No job jar file set.  User classes may not be
found. See JobConf(Class) or JobConf#setJar(String).
13/09/01 16:28:42 INFO input.FileInputFormat: Total input paths to process : 1
13/09/01 16:28:43 INFO mapred.JobClient: Running job: job_201308311801_0002
13/09/01 16:28:44 INFO mapred.JobClient:  map 0% reduce 0%
13/09/01 16:29:03 INFO mapred.JobClient:  map 25% reduce 0%
13/09/01 16:29:06 INFO mapred.JobClient:  map 78% reduce 0%
13/09/01 16:29:08 INFO mapred.JobClient:  map 100% reduce 0%
13/09/01 16:29:20 INFO mapred.JobClient:  map 100% reduce 33%
13/09/01 16:29:23 INFO mapred.JobClient:  map 100% reduce 100%
13/09/01 16:29:25 INFO mapred.JobClient: Job complete: job_201308311801_0002
...
13/09/01 16:29:25 INFO mapred.JobClient:     Reduce input records=2414
13/09/01 16:29:25 INFO mapred.JobClient:     Reduce input groups=836
...
13/09/01 16:29:25 INFO mapred.JobClient:     Map output records=2414
13/09/01 16:29:25 INFO mapreduce.JobRunner: Completed job [tweetCountJob]
...
```

# And the results ...



HDFS:/tweets/results/part-r-00000
http://sandbox:50075/browseBlock.jsp?blockId=3455012576500936589&blockSize=9806&genstamp=1350&filename=%2Ftweets%2Fresults%2Fpart-r-0(
Google

## File: /tweets/results/part-r-00000

Goto : /tweets/results go

*Go back to dir listing*
Advanced view/download options

```
NASCAR	1
NBA	474
NBA2K12	1
NBA2K14	2
NBA4ARAB	1
NBAAllStar	2
NBAChampions	3
NBAChamps	4
NBADraft	3
NBAFINALS	8
NBAFinals	88
NBAFrance	1
NBALIVE14	1
NBARigged	1
NBASoutheast	2
NBATV	2
NBAWithdrawals	1
NBAfinals	1
NFL	4
NFLFinals	1
NHL	6
NHLFinals	1
NHLinSeattle	1
NHLonNBC	1
```

# Spring’s PigRunner

- Execute a small Pig workflow

```
<pig-factory job-name=“analysis“ properties-location="pig-server.properties"/>

<script id="hdfsScript” location="copy-files.groovy">
  <property name=“sourceFile" value="${localSourceFile}"/>
  <property name="inputDir" value="${inputDir}"/>
  <property name="outputDir" value="${outputDir}"/>
</script>

<pig-runner id="pigRunner“ pre-action="hdfsScript” run-at-startup="true">
  <script location=“wordCount.pig">
    <arguments>
      inputDir=${inputDir}
      outputDir=${outputDir}
    </arguments>
  </script>
</pig-runner>
```

# PigTemplate - Configuration

- Helper class that simplifies the programmatic use of Pig
  - Common tasks are one-liners

```
<pig-factory id="pigFactory“ properties-location="pig-server.properties"/>

<pig-template pig-factory-ref="pigFactory"/>
```

- Similar XxxTemplate helper classes for Hive and HBase

# PigTemplate – Programmatic Use

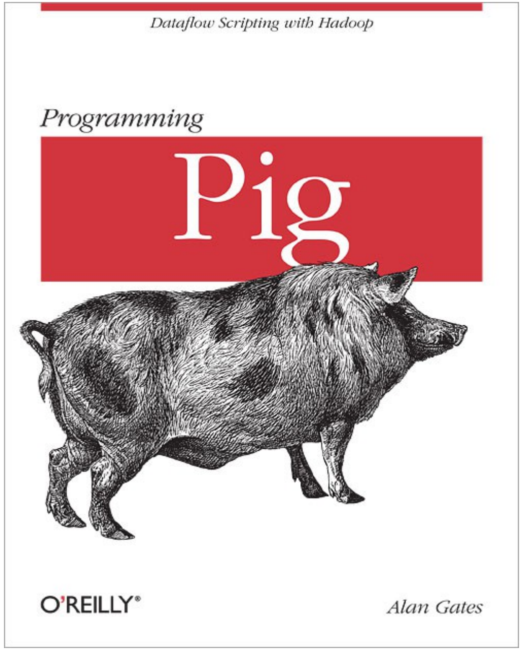
```java
public class PigPasswordRepository implements PasswordRepository {
  @Autowired
  private PigTemplate pigTemplate;
  @Autowired
  private String outputDir;
  private String pigScript = "classpath:password-analysis.pig";

  public void processPasswordFile(String inputFile) {

    Properties scriptParameters = new Properties();
    scriptParameters.put("inputDir", inputFile);
    scriptParameters.put("outputDir", outputDir);

    pigTemplate.executeScript(pigScript, scriptParameters);
  }
}
```

# Pig example using Spring

- We will use the output from the TweetHashTagCounter example
- Sort and select the top 10 #hashtags
- We need an application context
  - With an embedded Pig server
  - and a properties file
  - and a driver class

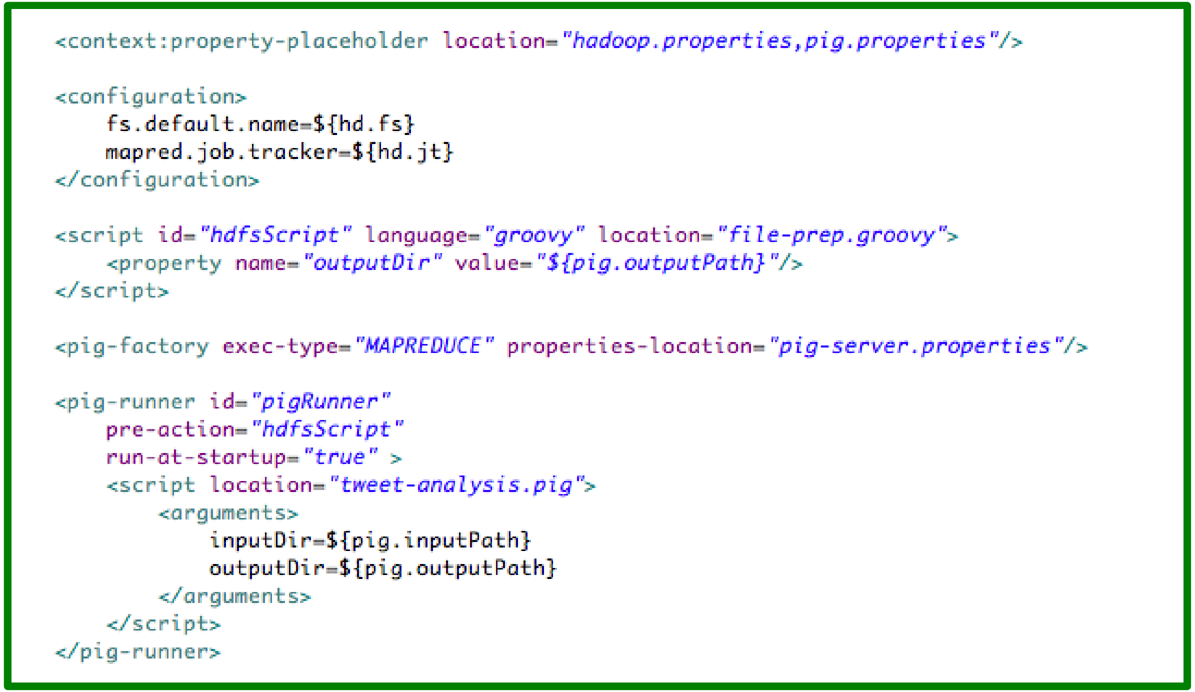
# Our Pig script

```
hashtags = LOAD '$inputDir' USING PigStorage('\t') AS (hashtag:chararray, count:int);
sorted = ORDER hashtags BY count DESC;
top10 = LIMIT sorted 10;
STORE top10 INTO '$outputDir';
```



Book: Programming Pig, Alan Gates http://shop.oreilly.com/product/0636920018087.do

# DEMO - Pig



```
<context:property-placeholder location="hadoop.properties,pig.properties"/>

<configuration>
    fs.default.name=${hd.fs}
    mapred.job.tracker=${hd.jt}
</configuration>

<script id="hdfsScript" language="groovy" location="file-prep.groovy">
    <property name="outputDir" value="${pig.outputPath}"/>
</script>

<pig-factory exec-type="MAPREDUCE" properties-location="pig-server.properties"/>

<pig-runner id="pigRunner"
    pre-action="hdfsScript"
    run-at-startup="true" >
    <script location="tweet-analysis.pig">
        <arguments>
            inputDir=${pig.inputPath}
            outputDir=${pig.outputPath}
        </arguments>
    </script>
</pig-runner>
```

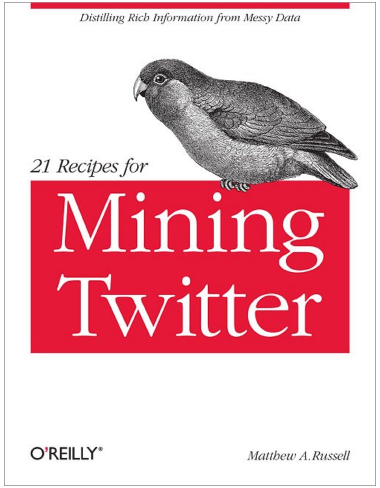
https://github.com/trisberg/springone-hadoop.git

# Hive example using Spring

- We will count the number of retweets per original user account found in the collection of tweeets collected during the 2013 NBA Finals
- Sort and select the top 10 users based on the number of retweets found – this should give us the influential users
- We need an application context
  - With and embedded Hive server
  - and a properties file
  - and a driver class

# Same input data – tweets captured during NBA finals

```
{
      "id": 348115421360164864,
      "text": "RT @NBA: The Best of the 2013 #NBAFinals set to 'Radioactive' by Imagine Dragons! http://t.co/EA198meYpC",
      "createdAt": 1371832158000,
      "fromUser": "I_Nikki_I",
...
      "retweetedStatus": {
            "id": 348111916452950016,
            "text": "The Best of the 2013 #NBAFinals set to 'Radioactive' by Imagine Dragons! http://t.co/EA198meYpC",
            "createdAt": 1371831323000,
            "fromUser": "NBA",
...
      },
...
      "entities": {
            "hashTags": [{
                  "text": "NBAFinals",
                  "indices": [30, 40]
            }]
      },
      "retweet": true
}
```

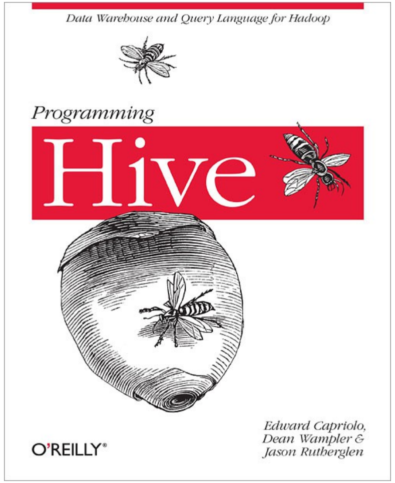


*The data file has the entire JSON document for each tweet on a single line*

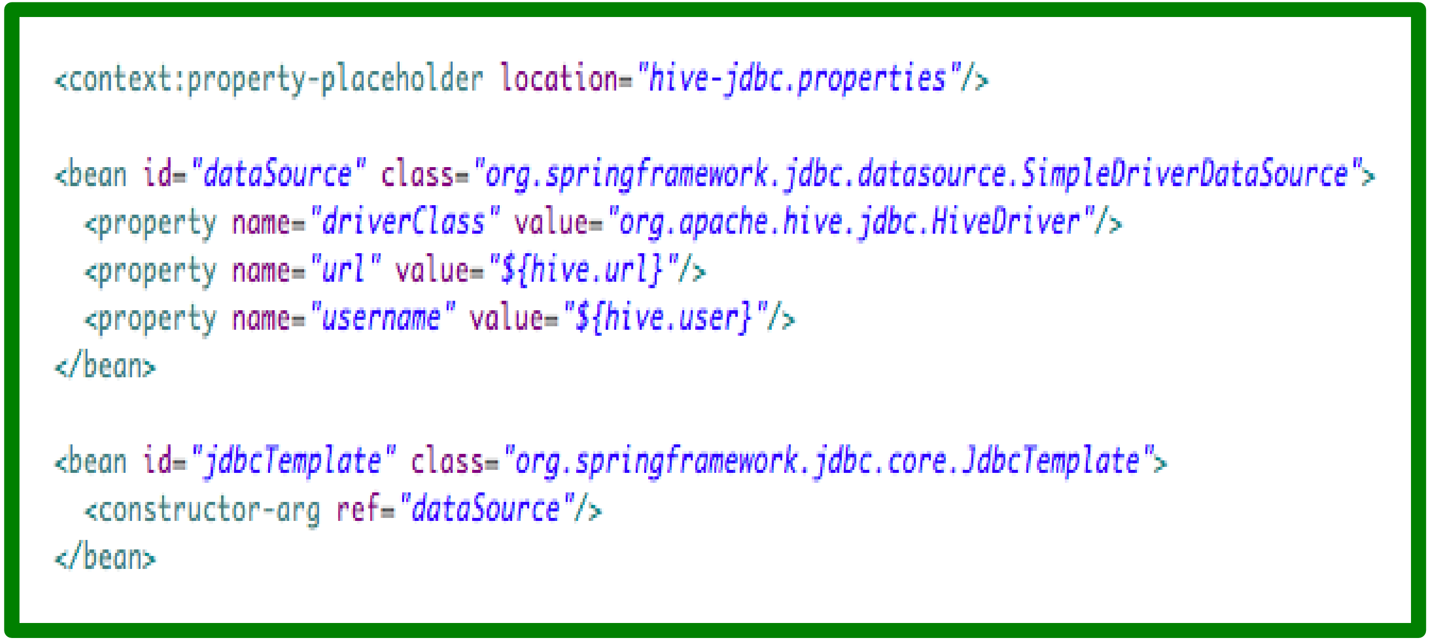
Book: 21 Recipes for Mining Twitter, Matthew A. Russell http://shop.oreilly.com/product/0636920018261.do

# Our Hive script

```
create external table tweetdata (value STRING) LOCATION '/tweets/input';

select r.retweetedUser, count(r.retweetedUser) as count
  from tweetdata j
       lateral view json_tuple(j.value, 'retweet', 'retweetedStatus') t as retweet, retweetedStatus
       lateral view json_tuple(t.retweetedStatus, 'fromUser') r as retweetedUser
 where t.retweet = 'true'
 group by r.retweetedUser order by count desc limit 10;
```



# DEMO - Hive

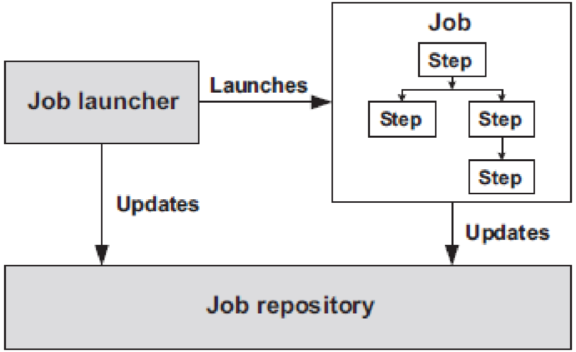


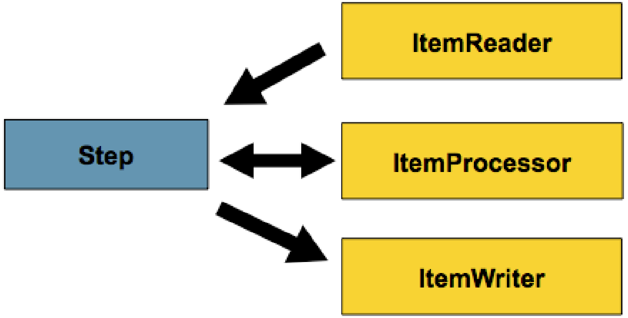
```
<context:property-placeholder location="hive-jdbc.properties"/>

<bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
  <property name="driverClass" value="org.apache.hive.jdbc.HiveDriver"/>
  <property name="url" value="${hive.url}"/>
  <property name="username" value="${hive.user}"/>
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <constructor-arg ref="dataSource"/>
</bean>
```

https://github.com/trisberg/springone-hadoop.git

# Spring Batch

- Framework for batch processing
  - Basis for JSR-352
- Born out of collaboration with Accenture in 2007
- Features
  - parsers, mappers, readers, writers
  - automatic retries after failure
  - periodic commits
  - synchronous and asynch processing
  - parallel processing
  - partial processing (skipping records)
  - non-sequential processing
  - job tracking and restart





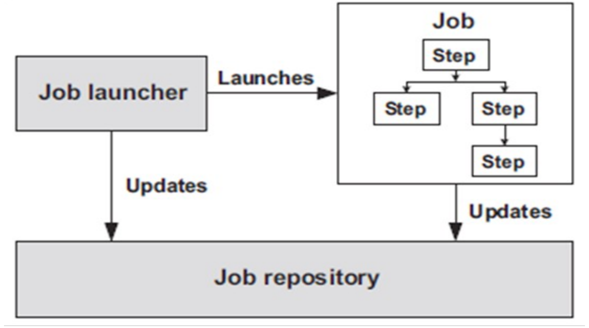
# Spring Batch workflows for Hadoop

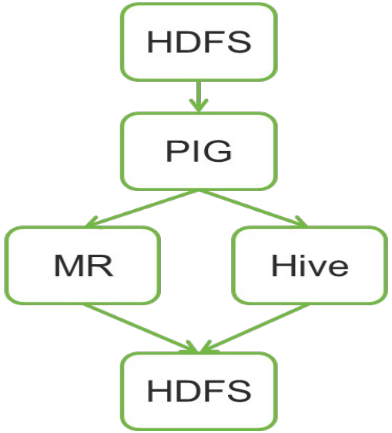
- Batch Ingest/Export
  - Examples
    - Read log files on local file system, transform and write to HDFS
    - Read from HDFS, transform and write to JDBC, HBase, MongoDB,…
- Batch Analytics
  - Orchestrate Hadoop based workflows with Spring Batch
  - Also orchestrate non-hadoop based workflows

# Hadoop Analytical workflow managed by Spring Batch

- Reuse same Batch infrastructure and knowledge to manage  Hadoop workflows

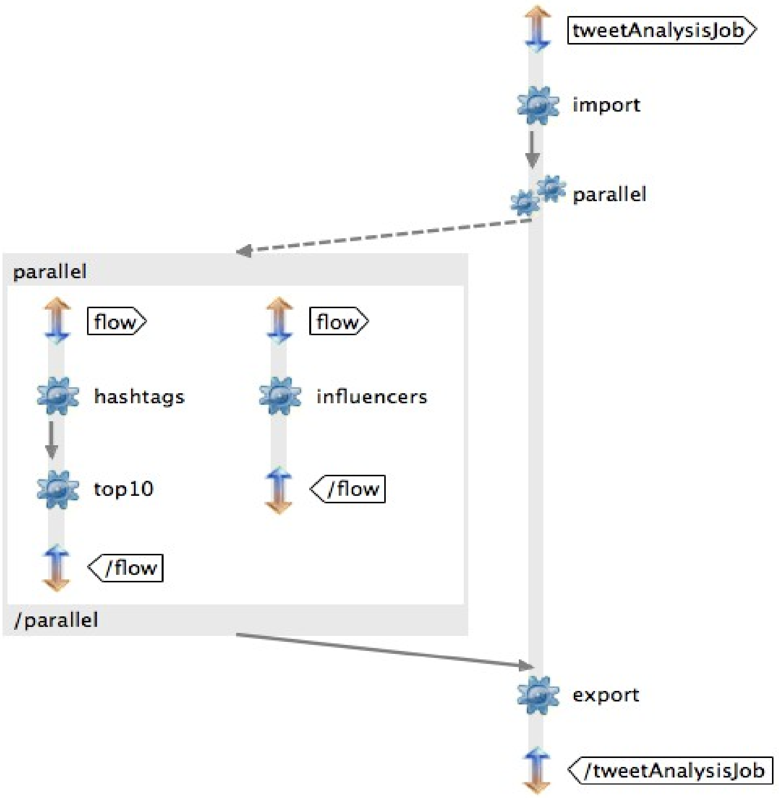
- Step can be any Hadoop job type or HDFS script





# Spring Batch Configuration for Hadoop

```
<batch:job id="tweetAnalysisJob">
   <batch:step id="import" next="parallel">
    <batch:tasklet ref="scriptTasklet"/>
  </batch:step>
  <batch:split id="parallel" task-executor="taskExec
    <batch:flow>
      <batch:step id="hashtags" next="top10">
        <batch:tasklet ref="hashtag-tasklet" />
      </batch:step>
      <batch:step id="top10">
        <batch:tasklet ref="top10-tasklet" />
      </batch:step>
    </batch:flow>
    <batch:flow>
      <batch:step id="influencers">
        <batch:tasklet ref="influencers-tasklet" />
      </batch:step>
    </batch:flow>
  </batch:split>
  <batch:step id="export" parent="export-step"/>
</batch:job>
```

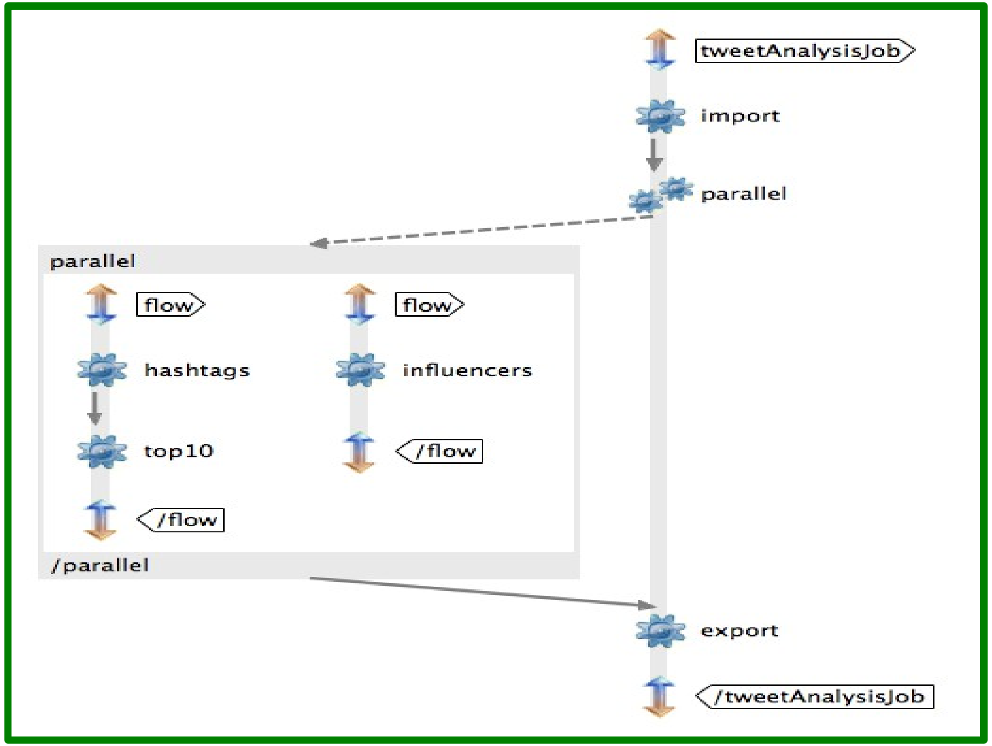


# Exporting HDFS to JDBC

- Use Spring Batch’s
  - MutliResourceItemReader + FlatFileItemReader
  - JdbcBatchItemWriter

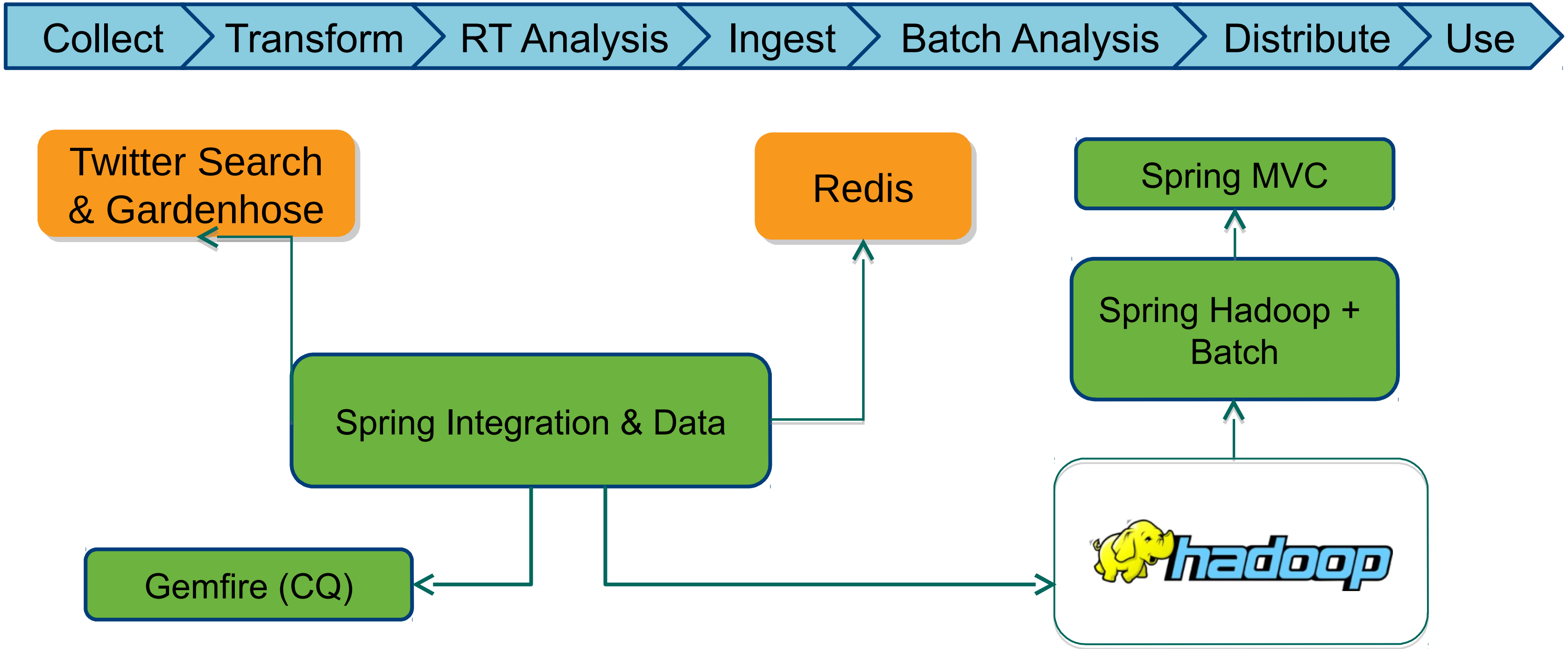
```
<step id="step1">
   <tasklet>
      <chunk reader="hdfsReader" processor="itemProcessor"
             writer="jdbcWriter"
             commit-interval="1000"
             skip-limit="3"/>
      </chunk>
   </tasklet>
</step>
```

# DEMO - Batch

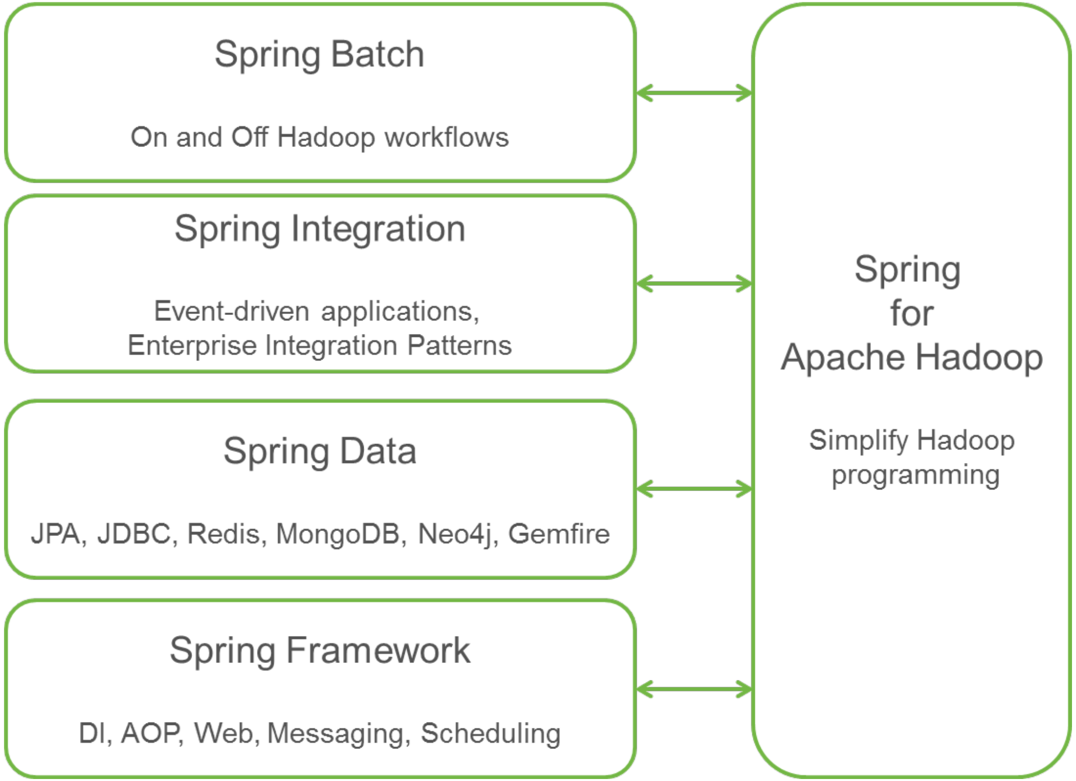


https://github.com/trisberg/springone-hadoop.git

# Big Data problems are also integration problems



# Relationship between Spring Projects



Spring Batch

On and Off Hadoop workflows

Spring Integration

Event-driven applications,
Enterprise Integration Patterns

Spring Data

JPA, JDBC, Redis, MongoDB, Neo4j, Gemfire

Spring Framework

DI, AOP, Web, Messaging, Scheduling

Spring
for
Apache Hadoop

Simplify Hadoop
programming

# Next Steps – Spring XD

- New open source umbrella project to support common big data use cases
  - High throughput distributed data ingestion into HDFS
    - From a variety of input sources
  - Real-time analytics at ingestion time
    - Gathering metrics, counting values, Gemfire CQ…
  - On and off Hadoop workflow orchestration
  - High throughput data export
    - From HDFS to a RDBMS or NoSQL database.

**<u>Tackling Big Data Complexity with Spring</u>**
**2:30 - 4:00 PM**
SCCC Theatre

Don't miss!

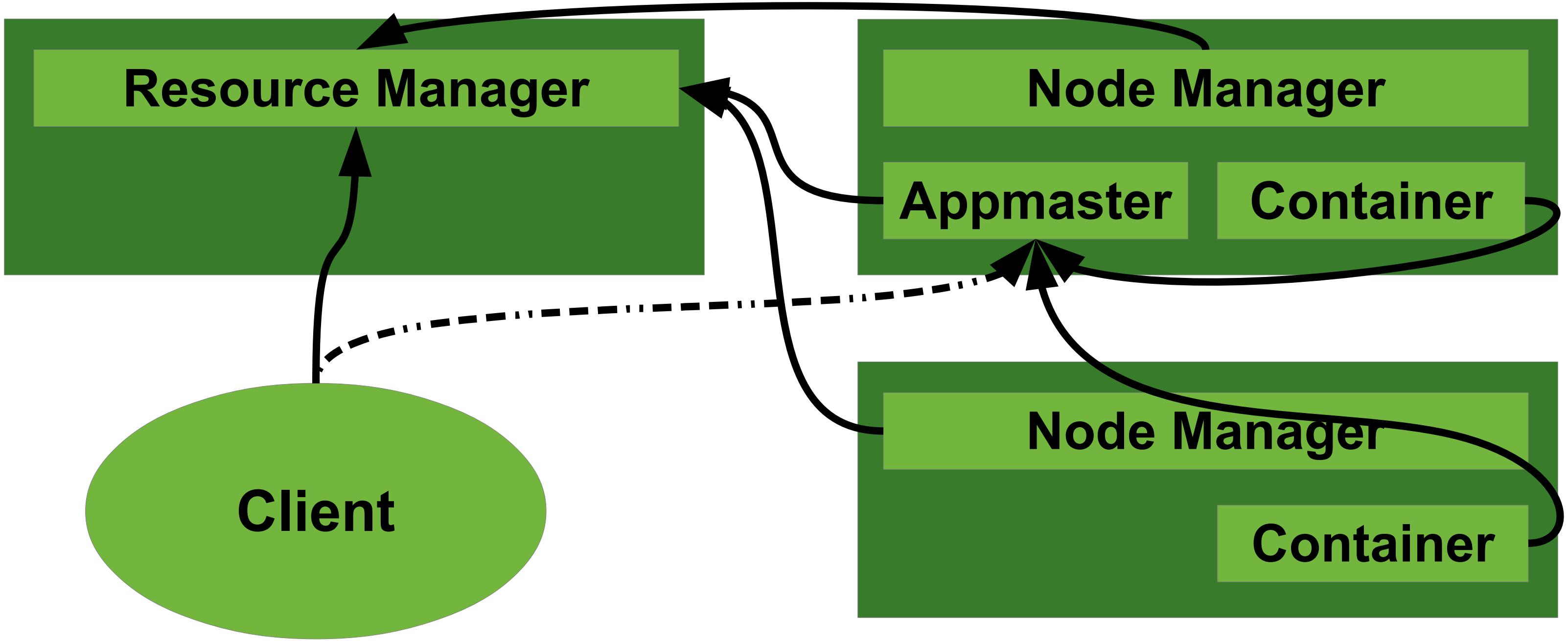
# Spring Yarn

> “*Spring Yarn provides features from the Spring programming model to make developing Yarn applications as easy as developing regular Spring applications.*

# Hadoop Yarn

- Hadoop v1 vs. v2
- Is a Resource Scheduler
- Is not a Task Scheduler
- YARN != Hadoop v2
- MapReduce v2 is a YARN Application
- Big Investment – Re-use Outside of MapReduce

# YARN Components



# Spring Yarn

- Is a Framework
- Run Spring Contexts on YARN
- Application Configuration
- No Boilerplate for Something Simple
- Extend to Create more Complex Applications

# Spring Yarn Concepts

- Configuration XML vs. JavaConfig (Milestone 2)
- Client
- Appmaster
- Container
- Bootstrap / Control

# Concepts - Configuration

- Familiar Spring Config Styles
  - XML – Namespace

```
<beans ...>

  <yarn:configuration />

  <yarn:client />
  <yarn:appmaster />
  <yarn:container />

</beans>
```

# Concepts – Configuration (Milestone 2)

- Familiar Spring Config Styles
  - JavaConfig – Builder / Configurers

```
@Configuration
@EnableYarn(enable=Enable.CLIENT)
class Config extends SpringYarnConfigurerAdapter
  @Override
  public void configure(YarnConfigBuilder config)
      throws ... {
    ...
  }
}
```

# Concepts - Client

- Access Yarn Cluster
- Submit / Control Running Applications
- Launch Context for Appmaster
  - Config
  - Libraries (Localization)
  - Environment

# Concepts - Appmaster

- Control the Running Application
- Appmaster is a main() of the Application
- Lifecycle
- Controls and Launches Containers
- Launch Context for Container
  - Config
  - Libraries (Localization
  - Environment

# Concepts - Container

- Real Job or Task is Done Here
- Run / Do Something and Exit
- Interact with Custom Services

# Concepts – Bootstrap / Control

- Application Context Having a YarnClient
  - Submit / Control
- CommandLineClientRunner
- Spring Boot
- Things to Remember
  - Dependencies for Hadoop Yarn Libs
  - Dependencies for Your Custom Code
  - Container Localized Files

# Project Setup

- Custom Class Files / Context Configs
- Testing Files if Needed
- Spring Yarn Examples
- Normal Spring Project

```
src/main/java/.../MultiContextContainer.java
src/main/resources/application-context.xml
src/main/resources/appmaster-context.xml
src/main/resources/container-context.xml
src/test/java/.../MultiContextTests.java
src/test/resources/MultiContextTests-context.xml
```

# Demo

- Simple Example
  - Run Multiple Containers
  - Let Containers Just Exit
  - Application Master is Finished
  - Application is Completed

# Testing with YARN

- Testing is Difficult
- Spring Yarn to Rescue
- Spring Test / Spring Yarn Test
- @MiniYarnCluster
- AbstractYarnClusterTests
- Yarn Configuration from a Mini Cluster

# Test – Client Context Config

```
<beans ...>

  <!-- where is our yarn config? -->

  <yarn:localresources />
  <yarn:environment />
  <yarn:client app-name="myAppName">
    <yarn:master-runner />
  </yarn:client>

</beans>
```

# Test - JUnit

```
@ContextConfiguration
(loader=YarnDelegatingSmartContextLoader.class)
@MiniYarnCluster
public class AppTests extends AbstractYarnClusterTests {
  @Test
  public void testApp() throws IOException {
    YarnApplicationState state =
      SubmitApplicationAndWait();
    assertNotNull(state);
    assertTrue(state.equals(
      YarnApplicationState.FINISHED));
  }
}
```

# Advanced Topic - Appmaster Services

- Link Between Appmaster and Container
  - Command / Control Container Internals
- Link Between Appmaster and Client
  - Command Your Custom Appmaster

# Advanced Topic - Container Locality

- Task Accessing Data on HDFS
- Container “near” HDFS Blocks
  - On Nodes
  - On Racks

# Advanced Topic - Spring Batch

- Execute Batch Partitioned Steps on Hadoop
- Proxy for Remote Job Repository
- Appmaster Runs the Batch Job

# Spring Yarn Future?

- M2 planned for Q4
- Java Config support
- 2.1.x-beta Overhauls Yarn APIs
  - Incompatible with Hadoop 2.0 alpha based distributions
- Potential Extensions
  - Thrift
  - Heartbeating
  - Container Grid/Groups

# Installing Hadoop

> *“A couple of ways to install a small Hadoop cluster that can be used to test your new Hadoop applications.*

# Hortonworks HDP 1.3 Sandbox

- Download:
  - http://hortonworks.com/products/hortonworks-sandbox/
- VMs available for:
  - VirtualBox
  - VMware Fusion or Player
  - Hyper-V

Hortonworks

# Installing HDP 1.3 Sandbox for VMware

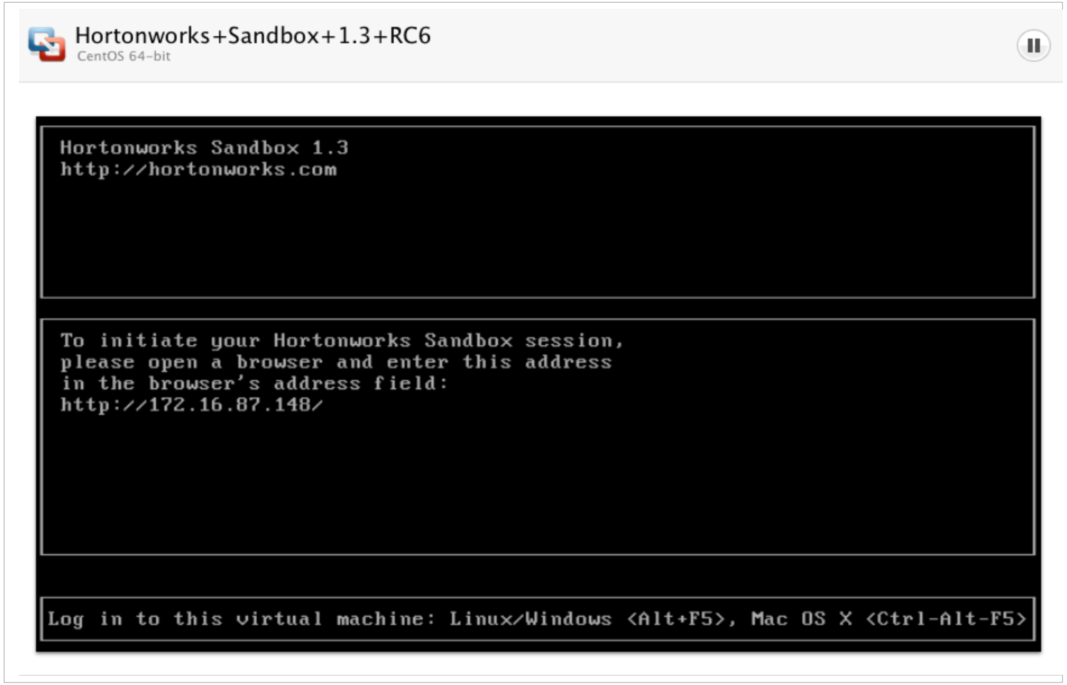
- Configured to use 2 processors
- Uses 2048MB memory
- Network shared with host

  sandbox resolves to IP assigned  to VM
- User/password:

  root/hadoop
- Listens on ports:

  HDFS - sandbox:8020

  JobTracker - sandbox:50300



```
Hortonworks Sandbox 1.3
http://hortonworks.com

To initiate your Hortonworks Sandbox session,
please open a browser and enter this address
in the browser's address field:
http://172.16.87.148/

Log in to this virtual machine: Linux/Windows <Alt+F5>, Mac OS X <Ctrl-Alt-F5>
```

# Using HDP 1.3 Sandbox for VMware

- Add to /etc/hosts on your local system *(adjust IP address to the one on startup screen)*:

```
172.16.87.148   sandbox
```

- Now you can access Hadoop on the sandbox:

```
$ hadoop dfs -ls hdfs://sandbox:8020/
Found 4 items
drwxr-xr-x   - hdfs   hdfs          0 2013-05-30 13:34 /apps
drwx------   - mapred hdfs          0 2013-08-23 12:31 /mapred
drwxrwxrwx   - hdfs   hdfs          0 2013-06-10 17:39 /tmp
drwxr-xr-x   - hdfs   hdfs          0 2013-06-10 17:39 /user
```

# Hadoop in Pseudo-distributed Mode (Single Node)

- Download Apache Hadoop (hadoop-2.0.6-alpha)
  - http://hadoop.apache.org/releases.html#Download
- Create a directory and unzip the download
  - I use ~/Hadoop on my system
- Modify `$HADOOP_INSTALL/etc/hadoop/hadoop-env.sh`
  - modify this line : `export JAVA_HOME=${JAVA_HOME}`
  - to be: `export JAVA_HOME="/usr/lib/jvm/java-6-openjdk-amd64"`

    *or to what your local Java installations home is*

# Update configuration files in etc/hadoop

core-site.xml

```
<configuration>

  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:8020</value>
    <final>true</final>
  </property>

</configuration>
```

mapred-site.xml

```
<configuration>

  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>

</configuration>
```

hdfs-site.xml

```
<configuration>

    <property>
        <name>dfs.support.append</name>
        <value>true</value>
    </property>

    <property>
      <name>dfs.webhdfs.enabled</name>
      <value>true</value>
    </property>

    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>

</configuration>
```

# Update configuration files in `etc/hadoop`

`yarn-site.xml`

```
<configuration>

    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce.shuffle</value>
    </property>

    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>

</configuration>
```

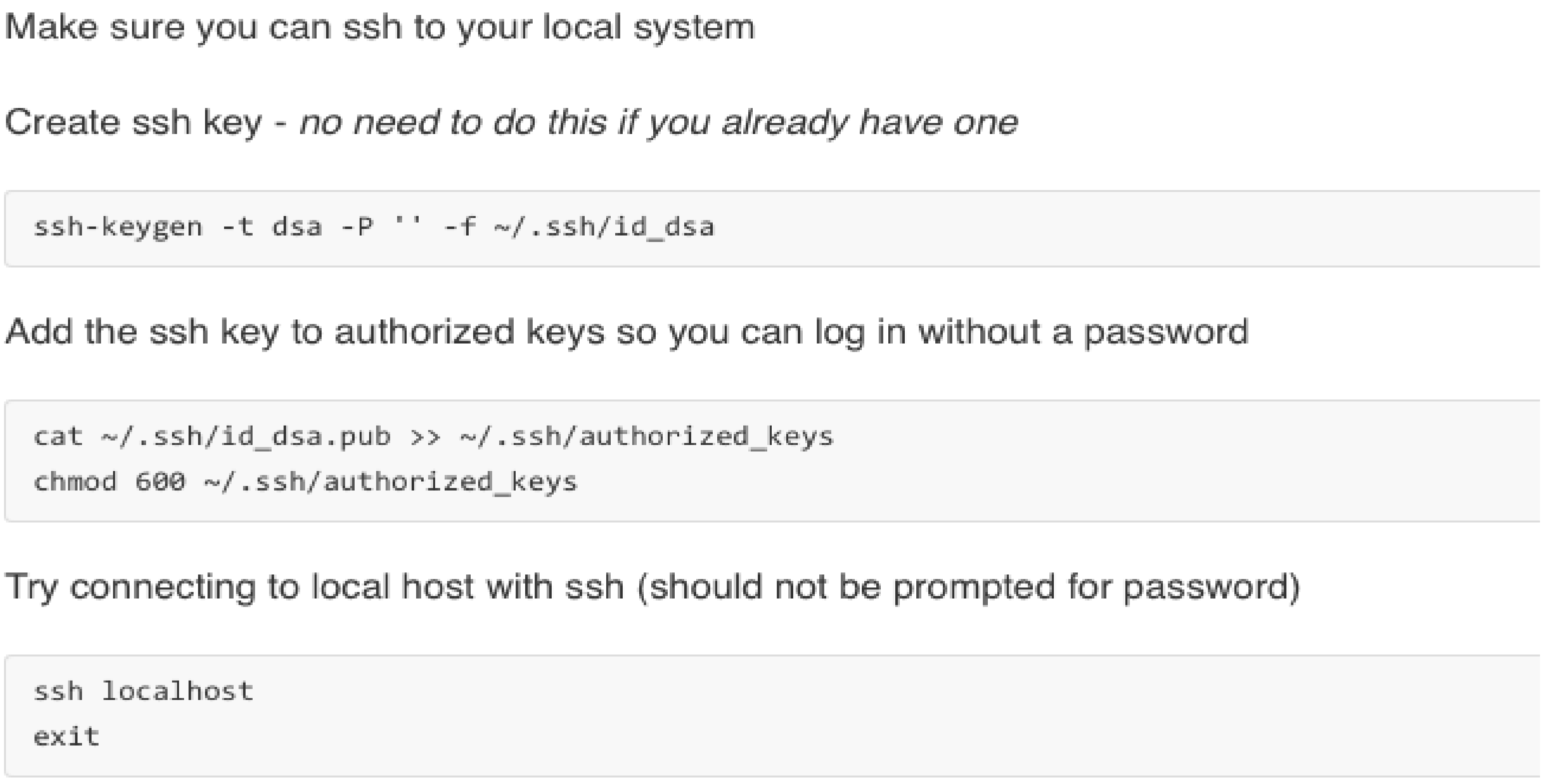
You can download these config files from:
https://github.com/trisberg/springone-hadoop/tree/master/hadoop-config/2.0.6-alpha

# Configure your environment settings

hadoop-2.0.6-env

```
export HADOOP_INSTALL=~/Hadoop/hadoop-2.0.6-alpha
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64

export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_YARN_HOME=$HADOOP_INSTALL

export HADOOP_CONF_DIR=$HADOOP_INSTALL/etc/hadoop

export PATH=$HADOOP_INSTALL/bin:$HADOOP_INSTALL/sbin:$PATH
```

# Configure your SSH settings



Make sure you can ssh to your local system

Create ssh key - *no need to do this if you already have one*

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
```

Add the ssh key to authorized keys so you can log in without a password

```
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

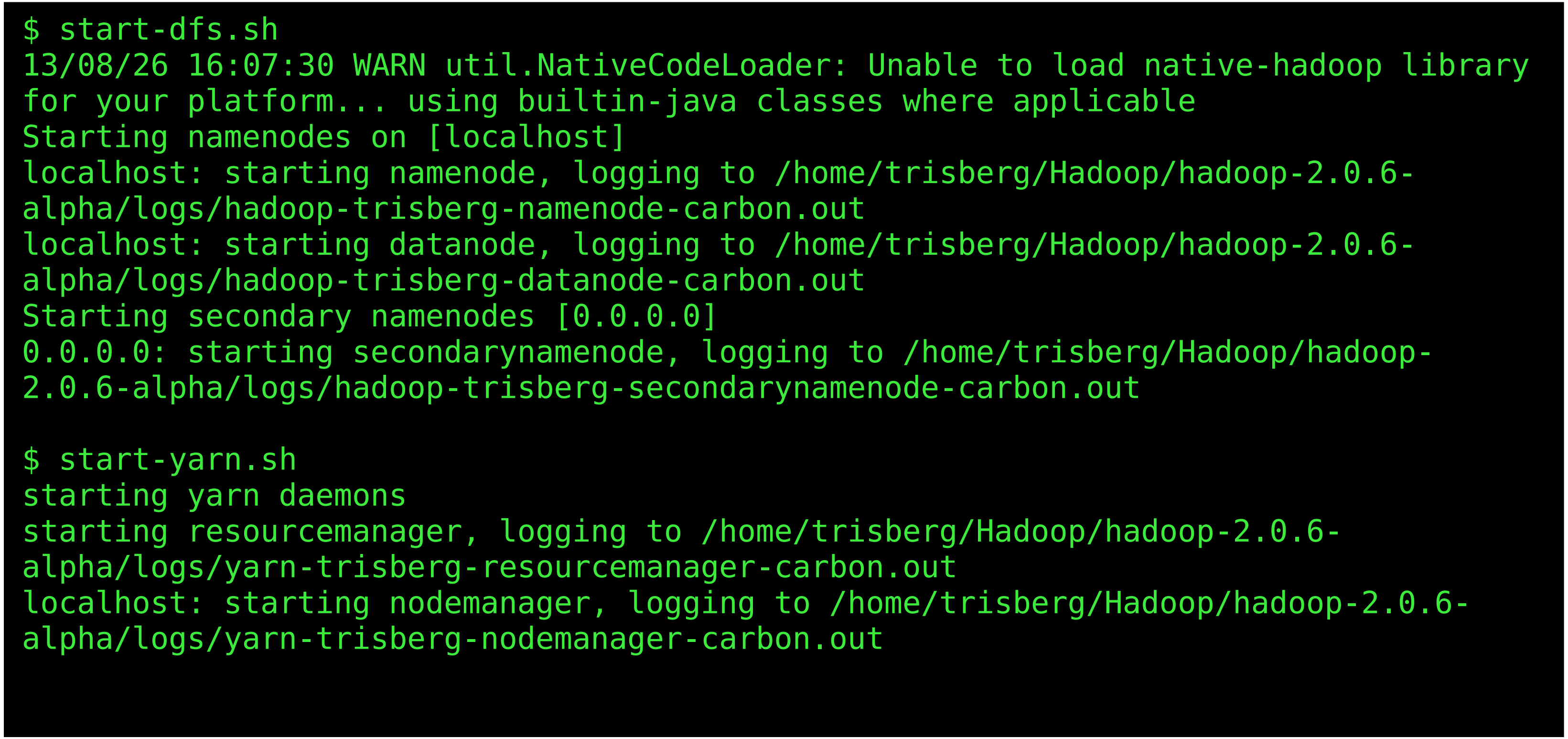
Try connecting to local host with ssh (should not be prompted for password)

```
ssh localhost
exit
```

# Let's start by formatting the namenode

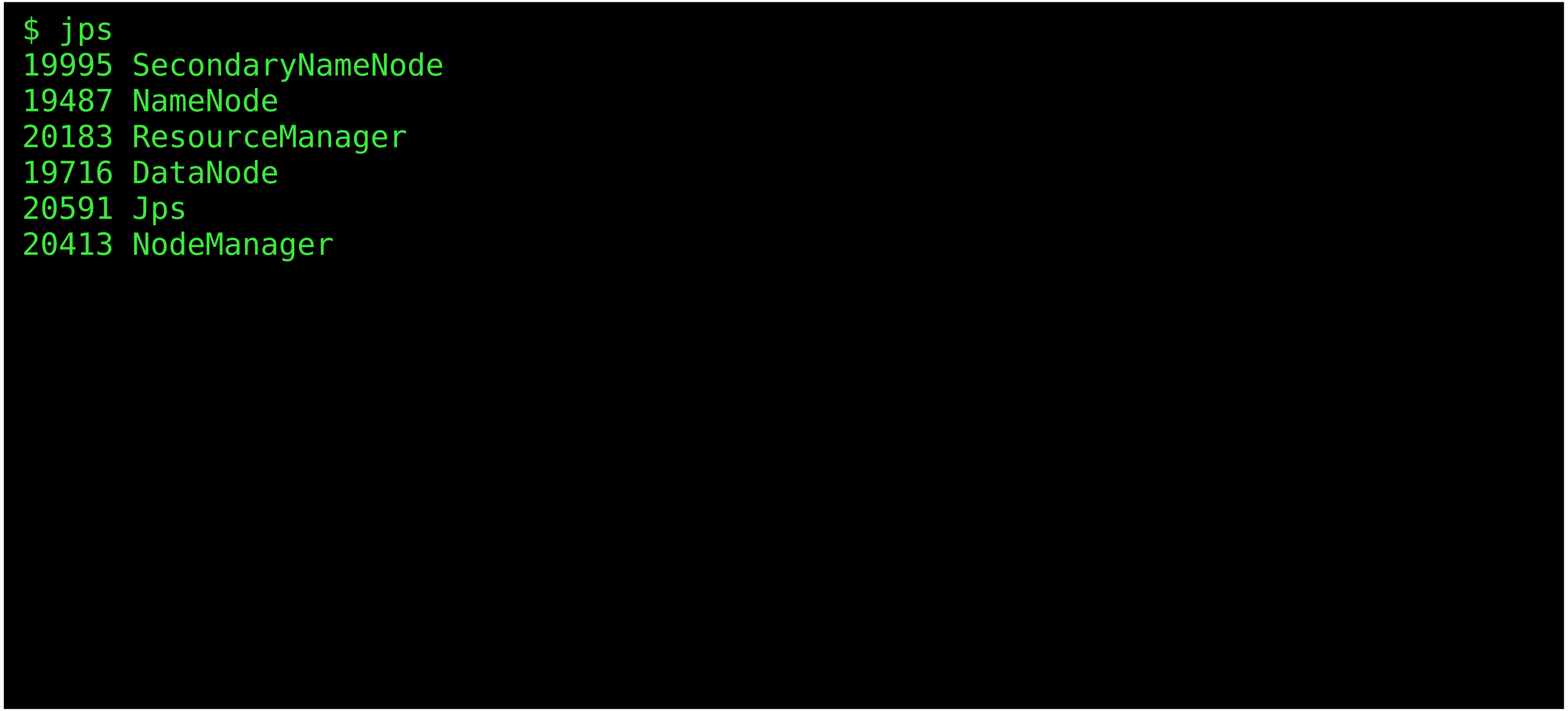
```
$ cd ~/Hadoop
$ source hadoop-2.0.6-env
$ hdfs namenode -format
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = carbon/192.168.0.114
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.0.6-alpha
...
Formatting using clusterid: CID-919300bd-2c08-483b-ab8d-a38ce1e31b1c
...
13/08/26 16:15:06 INFO common.Storage: Storage directory /tmp/hadoop-
trisberg/dfs/name has been successfully formatted.
13/08/26 16:15:06 INFO namenode.FSImage: Saving image file /tmp/hadoop-
trisberg/dfs/name/current/fsimage.ckpt_0000000000000000000 using no compression
...
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at carbon/192.168.0.114
************************************************************/
```

# Next, start the Hadoop “cluster”



```
$ start-dfs.sh
13/08/26 16:07:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library
for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/trisberg/Hadoop/hadoop-2.0.6-
alpha/logs/hadoop-trisberg-namenode-carbon.out
localhost: starting datanode, logging to /home/trisberg/Hadoop/hadoop-2.0.6-
alpha/logs/hadoop-trisberg-datanode-carbon.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/trisberg/Hadoop/hadoop-
2.0.6-alpha/logs/hadoop-trisberg-secondarynamenode-carbon.out

$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /home/trisberg/Hadoop/hadoop-2.0.6-
alpha/logs/yarn-trisberg-resourcemanager-carbon.out
localhost: starting nodemanager, logging to /home/trisberg/Hadoop/hadoop-2.0.6-
alpha/logs/yarn-trisberg-nodemanager-carbon.out
```

# Check that all daemons are running

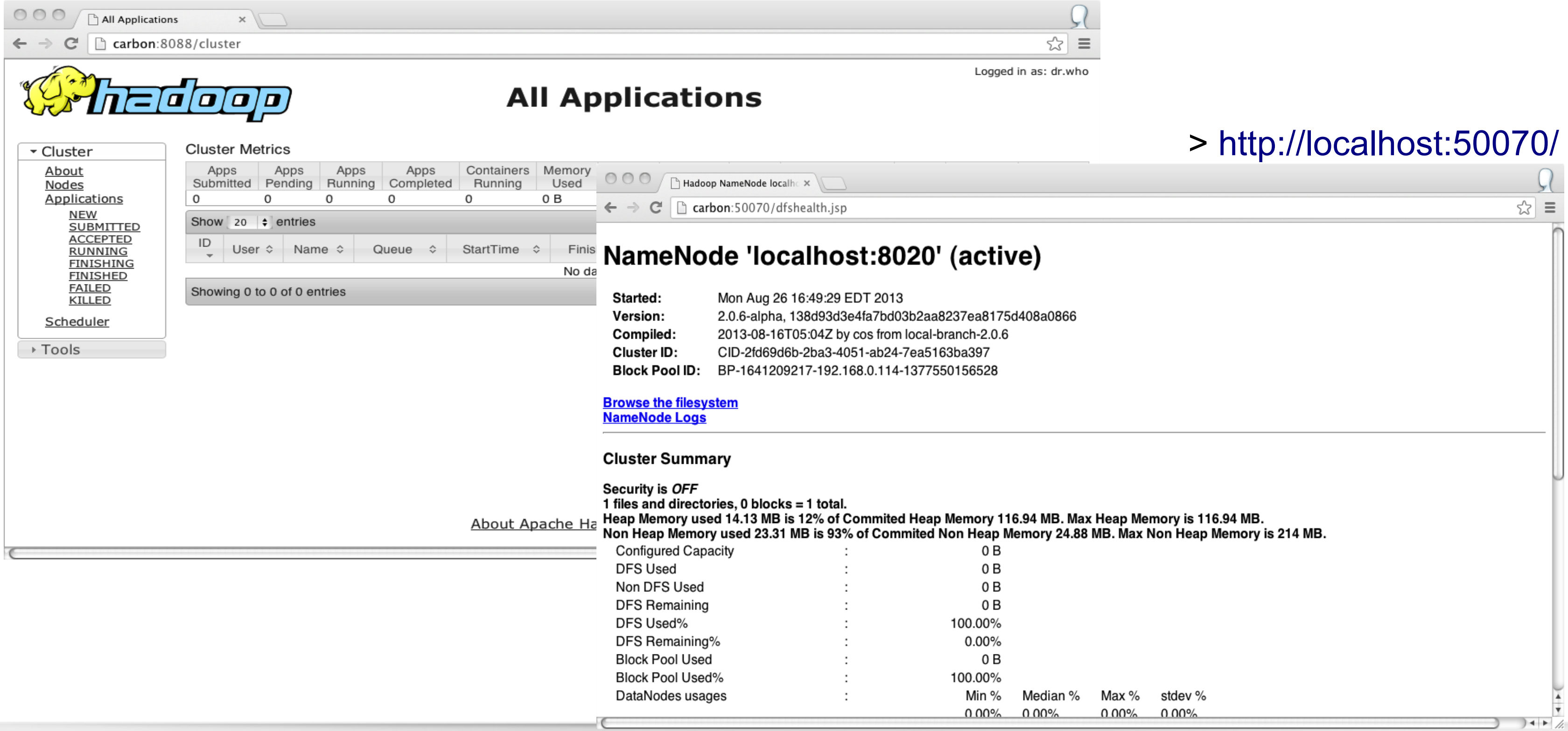


```
$ jps
19995 SecondaryNameNode
19487 NameNode
20183 ResourceManager
19716 DataNode
20591 Jps
20413 NodeManager
```

# Check cluster and hdfs web pages

> http://localhost:8088/

> http://localhost:50070/



## All Applications

Cluster Metrics

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Memory Used |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 B |

Show 20 entries

| ID | User | Name | Queue | StartTime | Finis |
|---|---|---|---|---|---|
| No da | | | | | |

Showing 0 to 0 of 0 entries

## NameNode 'localhost:8020' (active)

| | |
|---|---|
| **Started:** | Mon Aug 26 16:49:29 EDT 2013 |
| **Version:** | 2.0.6-alpha, 138d93d3e4fa7bd03b2aa8237ea8175d408a0866 |
| **Compiled:** | 2013-08-16T05:04Z by cos from local-branch-2.0.6 |
| **Cluster ID:** | CID-2fd69d6b-2ba3-4051-ab24-7ea5163ba397 |
| **Block Pool ID:** | BP-1641209217-192.168.0.114-1377550156528 |

Browse the filesystem
NameNode Logs

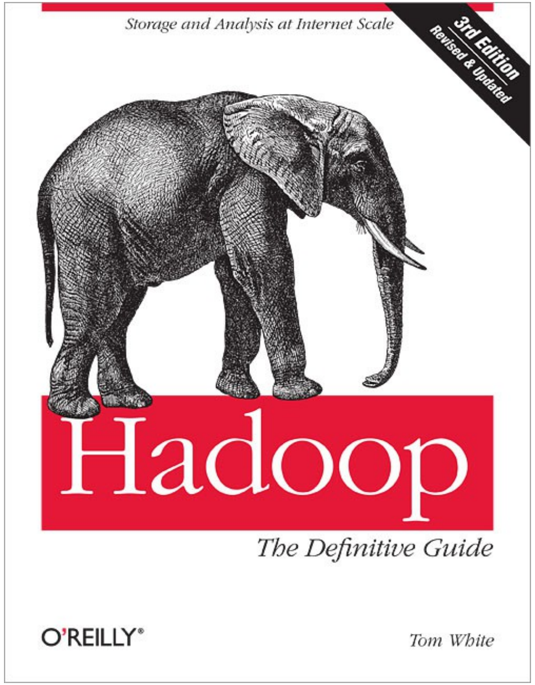
### Cluster Summary

**Security is *OFF***
**1 files and directories, 0 blocks = 1 total.**
**Heap Memory used 14.13 MB is 12% of Commited Heap Memory 116.94 MB. Max Heap Memory is 116.94 MB.**
**Non Heap Memory used 23.31 MB is 93% of Commited Non Heap Memory 24.88 MB. Max Non Heap Memory is 214 MB.**

| | | | | | |
|---|---|---|---|---|---|
| Configured Capacity | : | 0 B | | | |
| DFS Used | : | 0 B | | | |
| Non DFS Used | : | 0 B | | | |
| DFS Remaining | : | 0 B | | | |
| DFS Used% | : | 100.00% | | | |
| DFS Remaining% | : | 0.00% | | | |
| Block Pool Used | : | 0 B | | | |
| Block Pool Used% | : | 100.00% | | | |
| DataNodes usages | : | Min % | Median % | Max % | stdev % |
| | | 0.00% | 0.00% | 0.00% | 0.00% |

# For more detail ...

- This has been a brief intro to getting Apache Hadoop installed for development
- Lots more to learn ...



Book: Hadoop: The Definitive Guide, 3rd Edition, Tom White http://shop.oreilly.com/product/0636920021773.do

# Project Links

- **Source**:
  - https://github.com/spring-projects/spring-hadoop
- **Samples**:
  - https://github.com/spring-projects/spring-hadoop-samples
- **Project**:
  - http://projects.spring.io/spring-hadoop/
- **Forum**:
  - http://forum.spring.io/forum/spring-projects/data/hadoop

# Learn More.  Stay Connected.

We need your feedback -

http://forum.spring.io/forum/spring-projects/data/hadoop

- Talk to us on Twitter: @springcentral
- Find Session replays on YouTube: spring.io/video