

# **Simplifying Big Data Workflows for Hadoop**

## **Using Spring for Apache Hadoop**

Thomas Risberg, Pivotal

Raghavan "Rags" Srinivas, EMC

# Introduction

- Thomas Risberg (@trisberg) is a member of the Spring team at Pivotal contributing to the Spring XD project and leading the Spring for Apache Hadoop and JDBC Extensions projects.
- Raghavan "Rags" Srinivas (@ragss) works as an Architect/Developer Evangelist for EMC CODE(emccode.github.io) goaled with helping developers build highly scalable and available systems.

# Spring



Image credit: Thomas Risberg

<http://web.archive.org/web/20050324191121/http://www.springframework.org/images/spring2004.jpg>

# Spring Framework ...

- **Since 2003** the goal has been to simplify enterprise Java development
- **Spring Framework** uses POJO based programming model providing abstractions over many Java enterprise APIs
  - **Dependency Injection** helps to improve configuration and testability
  - **Declarative transaction management** makes your code simpler and easier to test
  - **MVC framework** originally an alternative to Apache Struts, extensively used and constantly evolving

# Spring Projects ...

- **Spring Batch** addresses common scenarios for batch processing
- **Spring Integration** for implementing enterprise integration patterns using messaging
- **Spring Boot** makes it easier to get started writing Spring apps
- **Spring Cloud** makes it easier to write Cloud Native applications
- **Spring XD** makes it easier to write Big Data applications for data ingestion, real time analytics, batch processing, and data export

# Hadoop



Image credit: Martine Lemmens  
<http://www.freeimages.com/photo/1328068>

# Forrester Hadoop Predictions 2015

*The Hadoop skills shortage will disappear. Hadoop is not that hard to understand. It is a file system, albeit distributed, and it is a computing platform, albeit distributed. The APIs are Java. What's the big deal? ...*

# What actually will happen

- The problem isn't **learning** the API
- It is **working** with a low level API
  - productivity suffers
  - code is harder to maintain
- You need to move to a much higher level of abstraction
  - we have seen this so many times - JDBC, servlet API, EJB ...
- BTW, “distributed” is the really hard part



# Forrester Hadoop Predictions 2015 (2)

*The shortage of Hadoop skills will quickly disappear as enterprises turn to their existing application development teams to implement projects such as filling data lakes and developing MapReduce jobs using Java. ...*

# What actually will happen (2)

- Few developers will ingest data writing Java code
  - they will use tools like Sqoop, Flume or Spring XD
- No developers will be writing new MapReduce code in Java
  - they will use higher level abstraction frameworks like Spark
  - or they will simply use SQL with Hive, HAWQ, Drill or Impala

# Need for Orchestration

- We still need tools to orchestrate all these pieces
  - Command line scripts and manual scheduling has its limitations
  - Spring Batch and Spring XD are great options for Java/Spring users
  - Oozie commonly used for pure Hadoop Eco System users

# Spring + Hadoop



# 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.*

”

# A quick example ...

- Display a directory listing of the Hadoop Distributed File System (HDFS)
- We need:
  - Spring Boot
  - URL for the Hadoop Namenode
  - An instance of the File System Shell

# Let's get Groovy ...

```
@Grab("org.springframework.data:spring-data-hadoop-boot:2.1.2.RELEASE")

import org.apache.hadoop.fs.FileStatus
import org.springframework.data.hadoop.fs.FsShell

public class Application implements CommandLineRunner {

    @Autowired FsShell fsShell

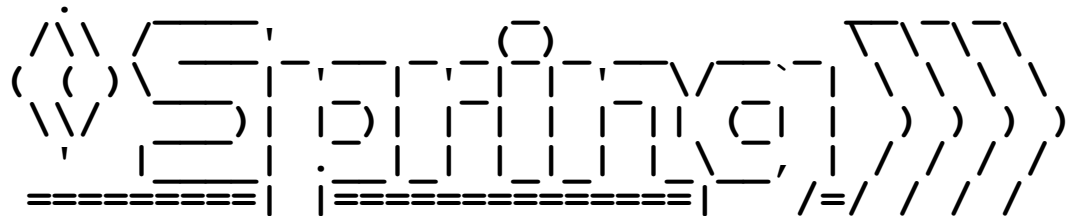
    void run(String... args) {
        println("Hello!")
        for (FileStatus fs : fsShell.ls("/")) {
            println("> ${fs.path.name}")
        }
    }
}
```

application.properties

spring.hadoop.fsUri=hdfs://borneo:8020

# Running the app ...

```
$ spring run app.groovy
```



The logo for Spring Boot, featuring the word "Spring" in a stylized font with a green outline, followed by "Boot" in a similar font. Below the text is a green horizontal line. To the right of the line is the text "(v2.1.3.RELEASE)".

:: Spring Boot :: (v2.1.3.RELEASE)

```
Hello:
```

```
> /  
> /tmp  
> /user  
> /xd
```



# Let's try Java ...

```
@ComponentScan
@EnableAutoConfiguration
public class Application implements CommandLineRunner {

    @Autowired
    private FsShell fsShell;

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Override
    public void run(String... strings) throws Exception {
        System.out.println("*** HDFS content:");
        for (FileStatus fs : fsShell.ls("/")) {
            System.out.println(fs.getOwner() +
                               " " + fs.getGroup() +
                               ": /" + fs.getPath().getName());
        }
    }
}
```

# How do I build it?

Use Maven or Gradle

Use Spring Boot for running app

Use Spring Boot parent pom for dependency management

Specify the version for Spring for Apache Hadoop and Hadoop:

```
spring-data-hadoop.version
```

```
hadoop.version
```

# Add Spring Boot

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.2.3.RELEASE</version>
  <relativePath/>
</parent>

<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

# And dependencies ...

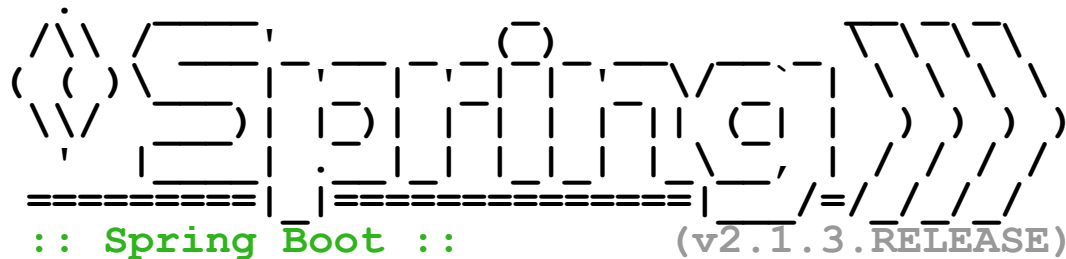
```
<properties>
  <spring-data-hadoop.version>2.1.2.RELEASE</spring-data-hadoop.version>
  <hadoop.version>2.6.0</hadoop.version>
  <java.version>1.7</java.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-autoconfigure</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-log4j</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.data</groupId>
    <artifactId>spring-data-hadoop-boot</artifactId>
    <version>${spring-data-hadoop.version}</version>
  </dependency>
</dependencies>
```

# Build and then Run ...

```
$ mvn clean package
```

```
...  
$ java -jar target/hello-hadoop-0.1.0.jar
```

The logo for Spring Boot, featuring a stylized green 'S' and 'B' with a green circle in the center, and the text 'Spring Boot' in green. To the right of the logo is the text '(v2.1.3.RELEASE)' in green.

:: Spring Boot :: (v2.1.3.RELEASE)

```
*** HDFS content:  
hdfs supergroup : /  
hdfs supergroup : /tmp  
hdfs supergroup : /user  
trisberg supergroup : /xd
```

# Spring for Apache Hadoop - Features

- **Consistent programming and declarative configuration model**

- Create, configure, and parameterize Hadoop connectivity and all job types
- Support for running MapReduce jobs, streaming, tool, jars
- Configure Hadoop's distributed cache
- Environment profiles – easily move application from dev to qa to prod
- Support for working with Hive, Pig and HBase
- Writing to HDFS – partitioning, many data formats
- Support for YARN programming

# Spring for Apache Hadoop - Features

- **Developer productivity**

- Create well-formed applications, not spaghetti script applications
- Simplify HDFS access and FsShell API with support for JVM scripting
- Helper “Template” classes for Pig/Hive/HBase
- Runner classes for MR/Pig/Hive for small workflows
- Tasklet implementations for larger Spring Batch flows

# 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 XD makes it even easier



# Spring For Apache Hadoop - History

- Project started by Dave Syer and Costin Leau in 2011
- First 1.0 GA release in February 2013
- Current versions:
  - 1.1.0 supports Hadoop v1 & v2 (1.0.4 - 2.2.0)
  - 2.0.4 supports Hadoop v1 & v2 (1.2.1 - 2.6.0)
  - 2.1.0 Hadoop v2 only (2.2.0 - ...)
- Next version:
  - 2.2.0 Hadoop v2 and Java 7+

# Versions, versions, versions ...

This presentation and all accompanying examples uses:

- Hadoop v2 APIs
- Apache Hadoop version 2.6.0
- Spring for Apache Hadoop 2.1.2.RELEASE
- Hive version 0.13.1, hiveserver2
- Spring XD version 1.1.1.RELEASE

# A unified model for different distributions

- Spring for Apache Hadoop provides several “flavors” to match dependencies with Hadoop distributions from:
  - Apache Hadoop
  - Cloudera CDH
  - Hortonworks HDP
  - Pivotal HD

# Supported distributions for 2.1

**hadoop24 hadoop25 hadoop26\*** (Apache)

**cdh5** (Cloudera)

**hdp22** (Hortonworks)

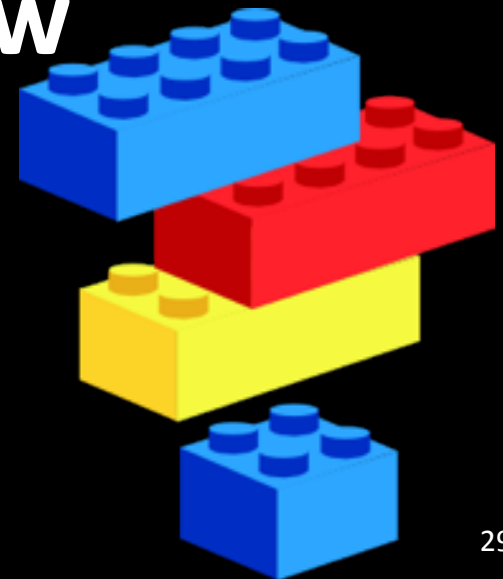
**phd21 phd30** (Pivotal)

Specified like this:

**2.1.2.RELEASE-hadoop24**

\* hadoop26 is the default

# Building Blocks for a Simple Workflow



# Spring For Apache Hadoop - Configuration

- XML namespace

```
<hadoop:configuration>
  fs.defaultFS=${spring.hadoop.fsUri}
  yarn.resourcemanager.address=${spring.hadoop.resourceManagerHost}
</hadoop:configuration>
```

- @Bean

```
@Value("${spring.hadoop.fsUri}")
String defaultFS;
@Value("${spring.hadoop.resourceManagerAddress}")
String resourceManager;

@Bean
Configuration hadoopConfiguration() {
    Configuration hadoopConfiguration = new Configuration();
    hadoopConfiguration.set("fs.defaultFS", defaultFS);
    hadoopConfiguration.set("yarn.resourcemanager.address", resourceManager);
    return hadoopConfiguration;
}
```

# Spring For Apache Hadoop - Jobs

- Runners for common jobs

job-runner, jar-runner, tool-runner, pig-runner, hive-runner

```
hdfs.input.path=/tweets/input/workflow  
hdfs.output.path=/tweets/results
```

```
<job id="tweetCountJob"  
  input-path="${hdfs.input.path}"  
  output-path="${hdfs.output.path}"  
  jar="file:${systemProperties['user.dir']}/target/lib/tweets-mapreduce.jar"  
  mapper="com.springdeveloper.hadoop.TweetCountMapper"  
  reducer="com.springdeveloper.hadoop.IntSumReducer"/>
```

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

# Spring For Apache Hadoop - Scripting

- Scripting support for File System Shell commands
  - mkdir, cp, chmod, rm
  - JavaScript, Groovy

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

```
<script id="setupScript" location="file-prep.groovy">  
    <property name="localFile" value="#${systemProperties['user.dir']}/${local.file}"/>  
    <property name="inputDir" value="${hdfs.input.path}"/>  
    <property name="outputDir" value="${hdfs.output.path}"/>  
</script>
```

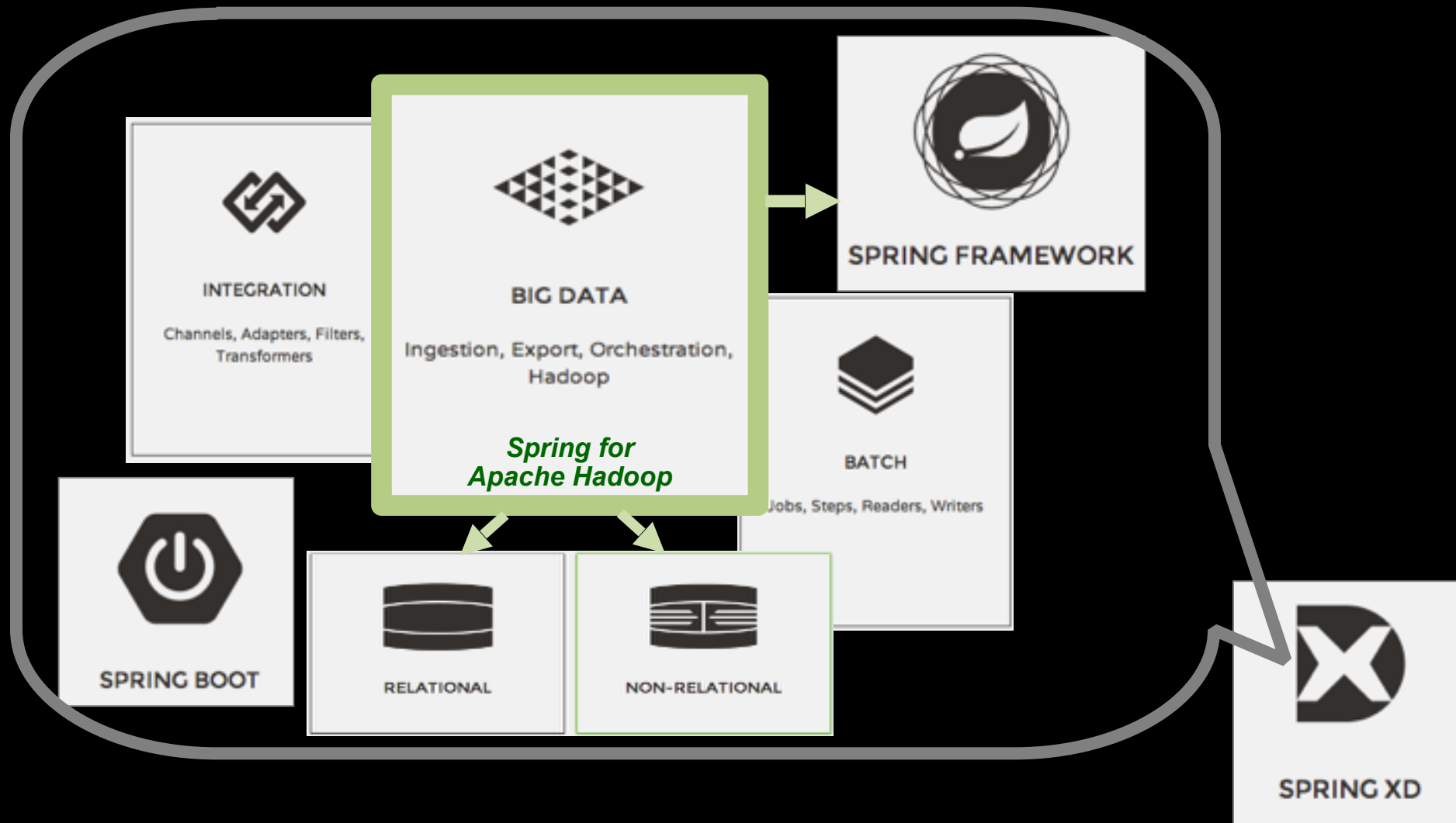
```
local.file=../data/hadoop-tweets_2014-09-02.txt  
hdfs.input.path=/tweets/input/workflow  
hdfs.output.path=/tweets/results
```



# Bigger Workflows

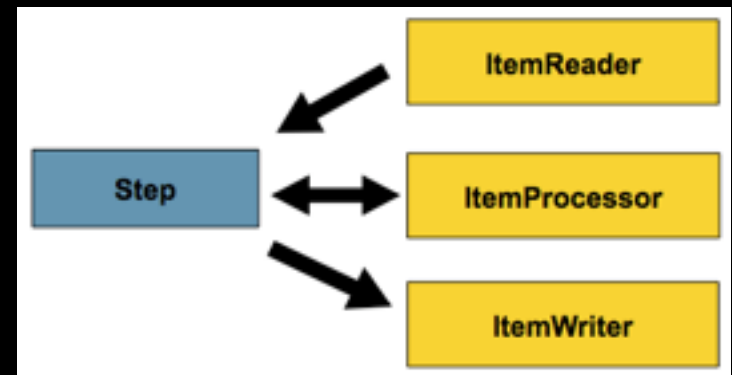
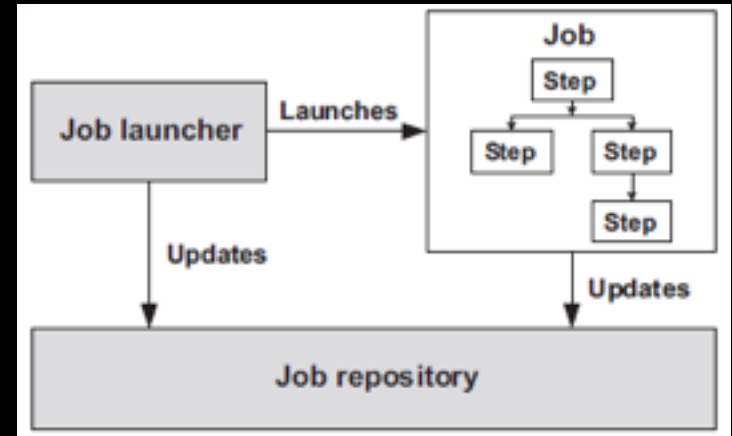


# Spring Projects for Big Data



# 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 data from RDBMS/NoSQL DB, transform and write to HDFS

- Read from HDFS, transform and write to RDBMS, HBase, MongoDB,...

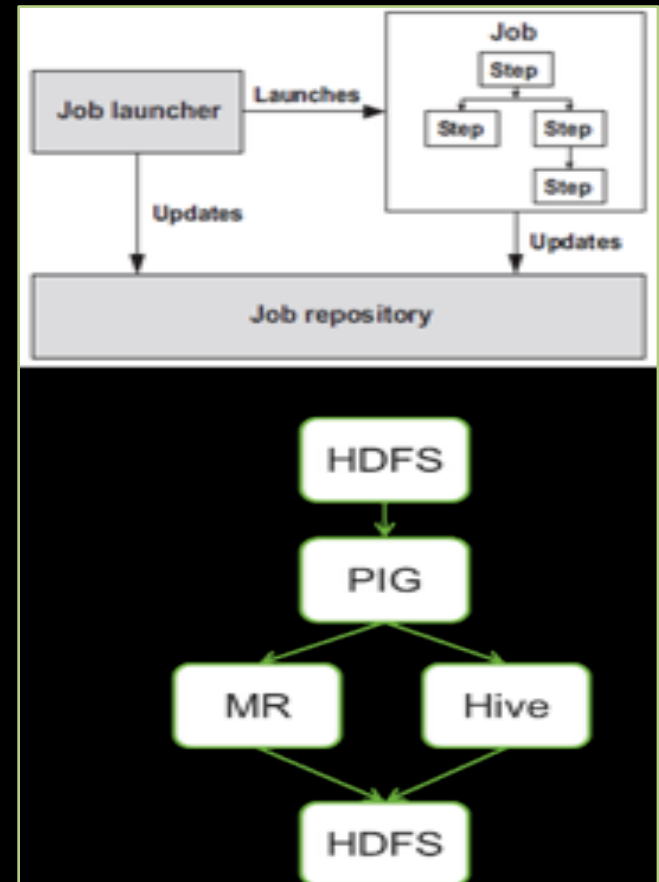
- Batch Analytics

- Orchestrate Hadoop based workflows with Spring Batch

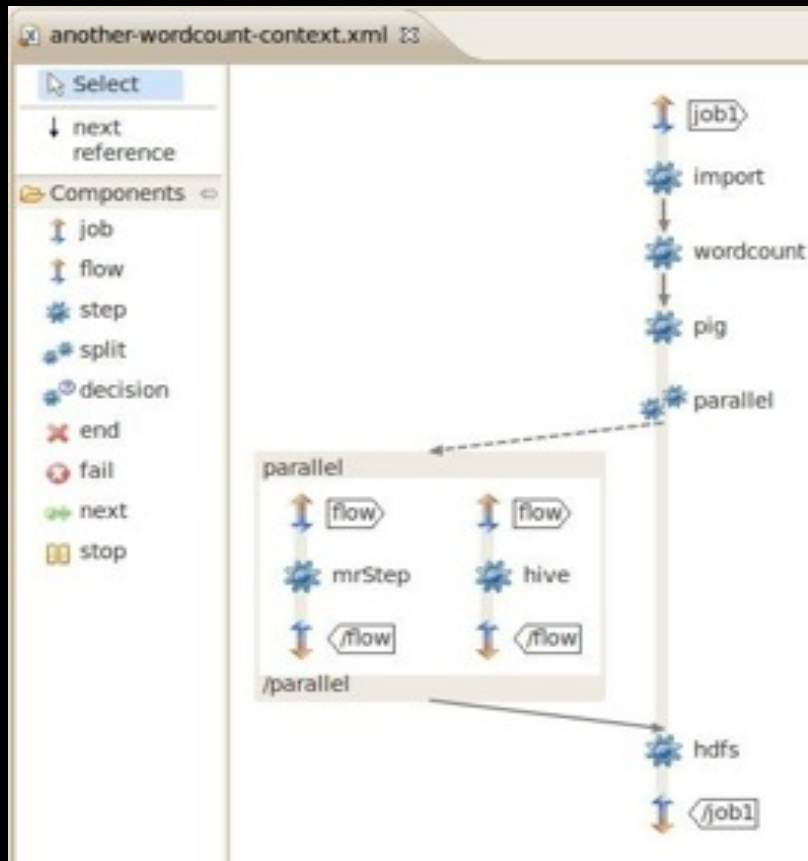
- Also orchestrate non-hadoop based workflows

# Hadoop Analytical Workflow with 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



```
<job id="job1">
  <step id="import" next="wordcount">
    <tasklet ref="import-tasklet"/>
  </step>
  <step id="wc" next="pig">
    <tasklet ref="wordcount-tasklet"/>
  </step>
  <step id="pig">
    <tasklet ref="pig-tasklet"/>
  </step>
  <split id="parallel" next="hdfs">
    <flow><step id="mrStep">
      <tasklet ref="mr-tasklet"/>
    </step></flow>
    <flow><step id="hive">
      <tasklet ref="hive-tasklet"/>
    </step></flow>
  </split>
  <step id="hdfs">
    <tasklet ref="hdfs-tasklet"/>
  </step>
</job>
```

# Exporting HDFS to JDBC

- Use Spring Batch's
  - MutliFileItemReader
  - JdbcItemWriter

```
<step id="step1">
  <tasklet>
    <chunk reader="flatFileItemReader" processor="itemProcessor"
            writer="jdbcItemWriter"
            commit-interval="100" retry-limit="3"/>
    </chunk>
  </tasklet>
</step>
```

# Batch Workflow Example

- We are getting daily files containing tweets from a twittersearch for #hadoop
- Files are in a /var/hadoop-data/{date} directory
- We need to:
  - Copy them to HDFS directory
  - Run MapReduce and Hive Jobs
  - Run some steps in parallel
  - Export data to RDBMS



# DEMO

## Bigger Workflow Example

# Running Spring+Hadoop Apps in the Cloud



# Hadoop in the Cloud

- Docker images
- AWS with Cloudbreak/Ambari
- Cloud Foundry Lattice
- Pivotal Cloud Foundry
- IBM BlueMix BigInsights
- Amazon Elastic MapReduce
- Microsoft Azure HDInsight

# Spring Cloud Connectors

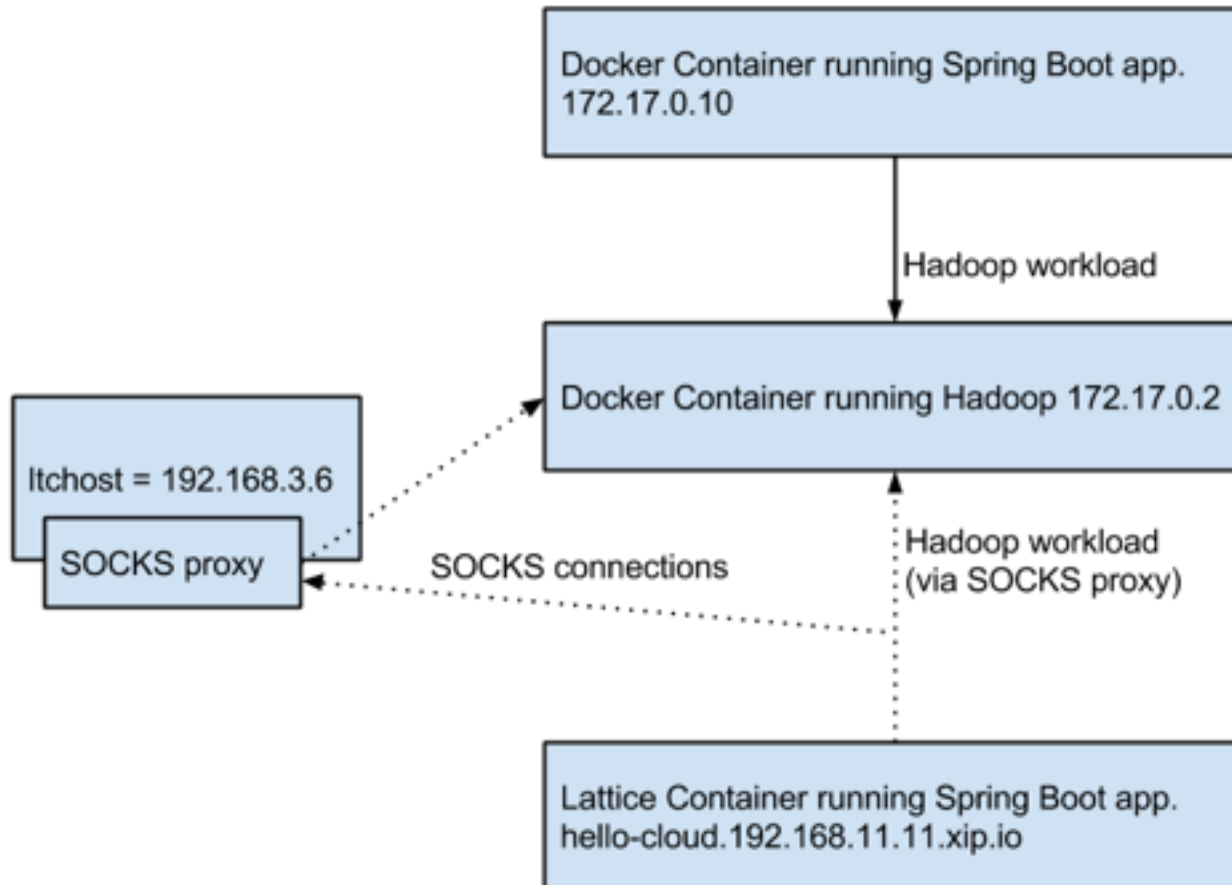
- Provides configuration management for runtime cloud services
- Supports RDBMS, NoSQL, Messaging etc.
  - need to add options for Hadoop services

<http://cloud.spring.io/spring-cloud-connectors/>

# Spring Profiles

- Provides configuration for multiple environments
  - switch by passing environment variable
- Also works with cloud connectors

# Cloud Demo



# DEMO

**Spring+Hadoop App  
in the “Cloud”**

# Resources

Demo Source: <https://github.com/trisberg/big-data-2015>

Project: <http://projects.spring.io/spring-hadoop/>

Questions:

<http://stackoverflow.com/questions/tagged/spring-data-hadoop>

Twitter: <https://twitter.com/springcentral>

YouTube: <http://youtube.com/user/SpringSourceDev>



Q&A

Thank You

Thomas Risberg (@trisberg)

Raghavan "Rags" Srinivas (@ragss)