Oozie

Additionally see the customized Hadoop training courses (onsite or at public venues) – [http://courses.coreservlets.com/hadoop-training.html](http://courses.coreservlets.com/hadoop-training.html)

Customized Java EE Training: [http://courses.coreservlets.com/](http://courses.coreservlets.com/)
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at your location.

For live customized Hadoop training (including prep for the Cloudera certification exam), please email info@coreservlets.com
Taught by recognized Hadoop expert who spoke on Hadoop several times at JavaOne, and who uses Hadoop daily in real-world apps. Available at public venues, or customized versions can be held on-site at your organization.

• Courses developed and taught by Marty Hall
  – JSF 2.2, PrimeFaces, servlets/JSP, Ajax, jQuery, Android development, Java 7 or 8 programming, custom mix of topics
  – Courses available in any state or country. Maryland/DC area companies can also choose afternoon/evening courses.
• Courses developed and taught by coreservlets.com experts (edited by Marty)
  – Spring, Hibernate/JPA, GWT, Hadoop, HTML5, RESTful Web Services
Contact info@coreservlets.com for details
Agenda

• Introduce Oozie
• Oozie Installation
• Write Oozie Workflow
• Deploy and Run Oozie Workflow

Oozie

• Workflow scheduler for Hadoop
  – Java MapReduce Jobs
  – Streaming Jobs
  – Pig
• Top level Apache project
  – Comes packaged in major Hadoop Distributions
    • Cloudera Distribution for Hadoop (CDH)
    – http://incubator.apache.org/oozie
• Provides workflow management and coordination of those workflows
• Manages Directed Acyclic Graph (DAG) of actions
Oozie

- Runs HTTP service
  - Clients interact with the service by submitting workflows
  - Workflows are executed immediately or later
- **Workflows are defined via XML**
  - Instead of writing Java code that implements Tool interface and extending Configured class
Action and Control Nodes

- Control Flow
  - start, end, kill
  - decision
  - fork, join

- Actions
  - map-reduce
  - java
  - pig
  - hdfs

```
<workflow-app name="foo-wf">
  <start to="[NODE-NAME]"/>
  <map-reduce>
    ...
    ...
  </map-reduce>
  <kill name="[NODE-NAME]">
    <message>Error occurred</message>
  </kill>
  <end name="[NODE-NAME]"/>
</workflow-app>
```
Oozie Coordination Engine

- Oozie Coordination Engine can trigger workflows by
  - Time (Periodically)
  - Data Availability (Data appears in a directory)

Install Oozie

- $ mkdir <OOZIE_HOME>/libext
- Download ExtJS and place under <OOZIE_HOME>/libext
  - ext-2.2.zip
- Place Hadoop libs under libext
  - $ cd <OOZIE_HOME>
  - $ tar xvf oozie-hadooplibs-3.1.3-cdh4.0.0.tar.gz
  - $ cp oozie-3.1.3-cdh4.0.0/hadooplibs/hadooplib-2.0.0-cdh4.0.0/*.jar libext/
- Configure Oozie with components under libext
  - $ bin/oozie-setup.sh
Install Oozie

• Create environment variable for default url
  – export OOZIE_URL=http://localhost:11000/oozie
  – This allows you to use $oozie command without providing url

• Update oozie-site.xml to point to Hadoop configuration
  <property>
  <name>oozie.service.HadoopAccessorService.hadoop.configurations</name>
  <value>*=/home/hadoop/Training/CDH4/hadoop-2.0.0-cdh4.0.0/conf</value>
  </property>

• Setup Oozie database
  – $./bin/ooziedb.sh create -sqlfile oozie.sql -run DB Connection.

Install Oozie

• Update core-site.xml to allow Oozie become “hadoop” and for that user to connect from any host
  <property>
  <name>hadoop.proxyuser.hadoop.groups</name>
  <value>*</value>
  <description>Allow the superuser oozie to impersonate any members of the group group1 and group2</description>
  </property>
  <property>
  <name>hadoop.proxyuser.hadoop.hosts</name>
  <value>*</value>
  <description>The superuser can connect only from host1 and host2 to impersonate a user</description>
  </property>

• Learn more:
  http://hadoop.apache.org/common/docs/r1.0.3/Secure_Impersonation.html
Start Oozie

$ oozie-start.sh

Setting OOZIE_HOME: /home/hadoop/Training/CDH4/oozie-3.1.3-cdh4.0.0
Setting OOZIE_CONFIG: /home/hadoop/Training/CDH4/oozie-3.1.3-cdh4.0.0/conf
Sourcing: /home/hadoop/Training/CDH4/oozie-3.1.3-cdh4.0.0/conf/oozie-env.sh
setting OOZIE_LOG=/home/hadoop/Training/logs/oozie
setting CATALINA_PID=/home/hadoop/Training/hadoop_work/pids/oozie.pid
Setting OOZIE_CONFIG_FILE: oozie-site.xml
Setting OOZIE_DATA: /home/hadoop/Training/CDH4/oozie-3.1.3-cdh4.0.0/data
Using OOZIE_LOG: /home/hadoop/Training/logs/oozie
Setting OOZIE_LOG4J_FILE: oozie-log4j.properties
Setting OOZIE_LOG4J_RELOAD: 10
Setting OOZIE_HTTP_HOSTNAME: localhost
Setting OOZIE_HTTP_PORT: 11000
Setting OOZIE_ADMIN_PORT: 11001
...
...
...

Test Installation

$ oozie admin -status
System mode: NORMAL

http://localhost:11000/oozie/
Running Oozie Examples

- Extract examples packaged with Oozie
  - $ cd $OOZIE_HOME
  - $ tar xvf oozie-examples.tar.gz

- Copy examples to HDFS to user’s home directory
  - $ hdfs dfs -put examples examples

- Run an example
  - $ oozie job -config examples/apps/map-reduce/job.properties -run

- Check Web Console
  - http://localhost:11000/oozie/

Oozie Workflows

- Defined in XML
- Uses Process Definition Language

```xml
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
  ...
  <decision name="mydecision">
    <switch>
      <case to="reconsolidatejob">
        ${fs:fileSize(secondjobOutputDir) gt 10 * GB}
      </case>
      <case to="rexpandjob">
        ${fs:fileSize(secondjobOutputDir) lt 100 * MB}
      </case>
      <case to="recomputejob">
        ${ hadoop:counters('secondjob') [RECORDS] [REDUCE_OUT] lt 1000000 }"
      </case>
      <default to="end"/>
    </switch>
  </decision>
  ...
</workflow-app>
```
Oozie Workflows

- Workflows consist of
  - Action nodes
    - MapReduce, Pig, Hive
    - Streaming, Java, etc...
  - Control flow nodes
    - Logic decisions between action nodes
    - Execute actions based on conditions or in parallel

- Workflows begin with START node
- Workflows succeed with END node
- Workflows fail with KILL node
- Several actions support JSP Expression Language (EL)

Most Occurrences Workflows

START → Count Each Letter MapReduce → OK → Find Max Letter MapReduce → OK → Clean Up → END

- Action Node
- Control Flow Node
- Control Node

This source is in HadoopSamples project under /src/main/resources/mr/workflows
Most Occurrences Workflows

```xml
<workflow-app xmlns="uri:oozie:workflow:0.2" name="most-seen-letter">
  <start to="count-each-letter"/>
  <action name="count-each-letter">
    <map-reduce>
      <job-tracker>${jobTracker}</job-tracker>
      <name-node>${nameNode}</name-node>
      <prepare>
        <delete path="${nameNode}${outputDir}"/>
        <delete path="${nameNode}${intermediateDir}"/>
      </prepare>
      <configuration>
        <property>
          <name>mapreduce.job.map.class</name>
          <value>mr.wordcount.StartsWithCountMapper</value>
        </property>
        ...
      </configuration>
    </map-reduce>
    <ok to="find-max-letter"/>
    <error to="fail"/>
  </action>
  ...
</workflow-app>
```

START Action Node to count-each-letter
MapReduce action
MapReduce have optional Prepare section
Pass property that will be set on MapReduce job’s Configuration object
In case of success, go to the next job; in case of failure go to fail node

First map-reduce Action

```xml
<map-reduce>
  <job-tracker>${jobTracker}</job-tracker>
  <name-node>${nameNode}</name-node>
  <prepare>
    <delete path="${nameNode}${outputDir}"/>
    <delete path="${nameNode}${intermediateDir}"/>
  </prepare>
  <configuration>
    <property>
      <name>mapred.mapper.new-api</name>
      <value>true</value>
    </property>
    <property>
      <name>mapred.reducer.new-api</name>
      <value>true</value>
    </property>
    <property>
      <name>mapred.job.queue.name</name>
      <value>${queueName}</value>
    </property>
    ...
  </configuration>
</map-reduce>
```

Administrative items to indicate where namenode and resource manager is
Optional prepare section; allows to execute command prior running the job
By default “old api” is used; specify to use new api
Specify which queue to submit this job to Resource Manager
First map-reduce Action

...<property>
  <name>mapreduce.job.map.class</name>
  <value>mr.wordcount.StartsWithCountMapper</value>
</property>
<property>
  <name>mapreduce.job.combine.class</name>
  <value>mr.wordcount.StartsWithCountReducer</value>
</property>
<property>
  <name>mapreduce.job.reduce.class</name>
  <value>mr.wordcount.StartsWithCountReducer</value>
</property>
<property>
  <name>mapreduce.job.inputformat.class</name>
  <value>org.apache.hadoop.mapreduce.lib.input.TextInputFormat</value>
</property>
<property>
  <name>mapreduce.job.outputformat.class</name>
  <value>org.apache.hadoop.mapreduce.lib.output.TextOutputFormat</value>
</property>...

Specify Mapper, Reducer, Input and Output formats; this is instead of Tool implementation

This action will produce a file of tab separated key-value pairs as specified by TextOutputFormat

First map-reduce Action (continued)

...<property>
  <name>mapreduce.job.output.key.class</name>
  <value>org.apache.hadoop.io.Text</value>
</property>
<property>
  <name>mapreduce.job.output.value.class</name>
  <value>org.apache.hadoop.io.IntWritable</value>
</property>
<property>
  <name>mapreduce.input.fileinputformat.inputdir</name>
  <value>${inputFile}</value>
</property>
<property>
  <name>mapreduce.output.fileoutputformat.outputdir</name>
  <value>${intermediateDir}</value>
</property>...

These properties are substituted from job.properties file
Most Occurrences Workflows

Second MapReduce job

Token substituted from application properties file

Namenode and Yarn Resource Manager Location

Control Flow Node

Second map-reduce Action

Specify Mapper, Reducer and Combiner
Second map-reduce Action (continued)

First map-reduce action produced a file with tab separated key-value pairs; second step utilizes KeyValueTextInputFormat to read these pairs as text

...<property>
   <name>mapreduce.job.inputformat.class</name>
   <value>org.apache.hadoop.mapreduce.lib.input.KeyValueTextInputFormat</value>
</property>
<property>
   <name>mapreduce.job.outputformat.class</name>
   <value>org.apache.hadoop.mapreduce.lib.output.TextOutputFormat</value>
</property>
<property>
   <name>mapreduce.job.output.key.class</name>
   <value>org.apache.hadoop.io.Text</value>
</property>
<property>
   <name>mapreduce.job.output.value.class</name>
   <value>org.apache.hadoop.io.IntWritable</value>
</property>
...

Most Occurrences Workflows

...<action name="clean-up">
   <fs>
      <delete path='${nameNode}${intermediateDir}'/>
   </fs>
   <ok to="end"/>
   <error to="end"/>
</action>

<kil name="fail">
   <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
</kill>

<end name="end"/>
</workflow-app>
Package and Run Your Workflow

1. Create application directory structure with workflow definitions and resources
   – Workflow.xml, jars, etc..
2. Copy application directory to HDFS
3. Create application configuration file
   – specify location of the application directory on HDFS
   – specify location of the namenode and resource manager
4. Submit workflow to Oozie
   – Utilize oozie command line
5. Monitor running workflow(s)

1: Oozie Application Directory

• Must comply to directory structure spec

mostSeenLetter-oozieWorkflow
|--lib/
 | |--HadoopSamples.jar
|--workflow.xml

Workflow.xml defines workflow
Libraries should be placed under lib directory
1: Oozie Application Directory

- Can use a build tool to generate this structure
  - Samples use maven plugins (see pom.xml)
    - Maven-dependency-plugin
    - Maven-resources-plugin
  - Run `mvn clean package`
    - Will create `mostSeenLetter-oozieWorkflow` directory with dependencies and workflow definitions

2: Copy Application Directory to HDFS

- Oozie utilizes HDFS to load applications

```
hdfs dfs -put mostSeenLetter-oozieWorkflow
```

Copies directory from local files system onto HDFS; directory gets copied to user’s home directory
3: Create Application Configuration File

- **job.properties** - Needs to exist locally, required for submission

```properties
nameNode=hdfs://localhost:8020
jobTracker=localhost:8021
queueName=default

inputFile=/training/data/hamlet.txt
intermediateDir=/training/playArea/mostSeenLetter-oozieWorkflow-tmp
outputDir=/training/playArea/oozieWorkflow

oozie.wf.application.path=${nameNode}/user/${user.name}/mostSeenLetter-oozieWorkflow
```

Properties for required locations such as namenode and resource manage

Properties needed for the MapReduce actions in the workflow

Most importantly HDFS location of the application must be specified

4: Submit Workflow to Oozie

- **Use oozie command line tool**
  - For usage: $oozie help

```
$ oozie job -config job.properties -run
job: 0000001-120711224224630-oozie-hado-W
```

Application configuration file

Application ID; use this ID to get status
5: Monitor Running Workflow(s)

- **Two options**
  - Command line ($oozie$)
  - Web Interface (http://localhost:11000/oozie)

5: Monitor Running Workflow(s) - Command Line

$ oozie job -info 0000001-120711224224630-oozie-hado-W
Job ID : 0000001-120711224224630-oozie-hado-W

<table>
<thead>
<tr>
<th>Workflow Name : most-seen-letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Path      : hdfs://localhost:8020/user/hadoop/mostSeenLetter-oozieWorkflow</td>
</tr>
<tr>
<td>Status        : RUNNING</td>
</tr>
<tr>
<td>Run           : 0</td>
</tr>
<tr>
<td>User          : hadoop</td>
</tr>
<tr>
<td>Group         : -</td>
</tr>
<tr>
<td>Created       : 2012-07-13 03:08</td>
</tr>
<tr>
<td>Started       : 2012-07-13 03:08</td>
</tr>
<tr>
<td>Last Modified : 2012-07-13 03:08</td>
</tr>
<tr>
<td>Ended         : -</td>
</tr>
<tr>
<td>CoordAction ID: -</td>
</tr>
</tbody>
</table>

**Actions**

<table>
<thead>
<tr>
<th>ID</th>
<th>Status</th>
<th>Ext ID</th>
<th>Ext Status</th>
<th>Err Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000001-120711224224630-oozie-hado-W@count-each-letter</td>
<td>OK</td>
<td>job_1342136595052_0006</td>
<td>SUCCEEDED</td>
<td>-</td>
</tr>
<tr>
<td>0000001-120711224224630-oozie-hado-W@find-max-letter</td>
<td>RUNNING</td>
<td>job_1342136595052_0008</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5: Monitor Running Workflow(s) - Web Interface

http://localhost:11000/oozie

- Web Interface

Clicking on a particular application/job will bring up list of all the actions; click on each action to get further details.
5: Monitor Running Workflow(s) - Web Interface

Link to Resource Manager to view details of the job for this particular Oozie Action

This view displays details for a selected action.

5: Monitor Running Workflow(s) - Web Interface

Oozie assigns a name to each job

Clicking on the “Console Url” from action view will take you to the details of the job for that action.
mostSeenLetter-oozieWorkflow Result

$ hdfs dfs -cat /training/playArea/oozieWorkflow/part-r-00000 t 3711

Reminder: This source is in HadoopSamples project under /src/main/resources/mr/workflows

Oozie Resources

- **Home Page:**
  - Quick start, functional specifications for workflows, coordinators, and expression language

- **Mailing Lists**
  - [http://oozie.apache.org/mail-lists.html](http://oozie.apache.org/mail-lists.html)

- **Chapter about Oozie**

Hadoop: The Definitive Guide
Tom White (Author)
O'Reilly Media; 3rd Edition (May6, 2012)
Wrap-Up

Summary

• We learned about
  – Oozie Features
  – Oozie Installation

• We learned how to
  – Implement an Oozie Workflow
  – Deploy and Run Oozie Workflow
Questions?

More info:
- http://courses.coreservlets.com/Hadoop-training.html – Customized Hadoop training courses, at public venues or onsite at your organization
- http://www.coreservlets.com/java-8-tutorial – Java 8 tutorial
- http://www.coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training

Customized Java EE Training: http://courses.coreservlets.com/
Hadoop, Java, JSF 2, PrimeFaces, Servlets, JSP, Ajax, jQuery, Spring, Hibernate, RESTful Web Services, Android.
Developed and taught by well-known author and developer. At public venues or onsite at your location.