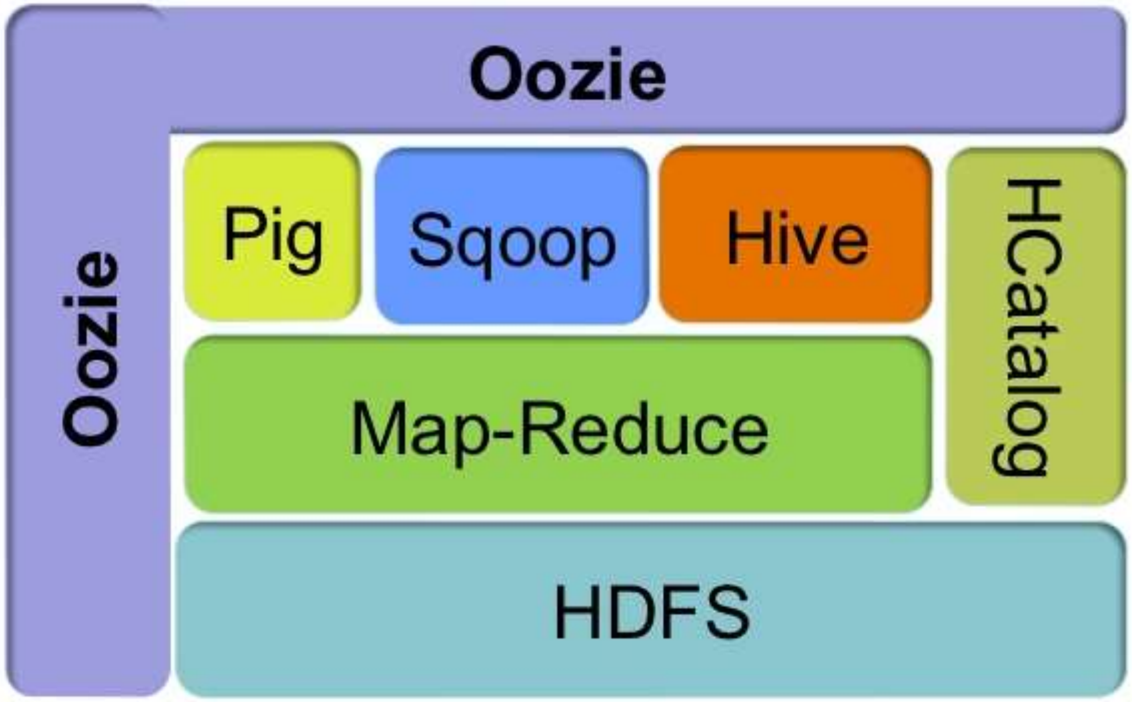
What is Oozie

- Oozie is a workflow scheduler system to manage Apache Hadoop jobs.
- Its a system for running workflows of dependent jobs
- Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions.
- Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts).

Oozie Features

- Designed to scale
- Can manage the timely execution of thousands of workflows in a Hadoop cluster
- Makes rerunning failed workflows more tractable
- Runs as a service in the cluster
- Clients can submit workflow definitions for immediate or later execution

Oozie in Hadoop Eco-System



Oozie Components

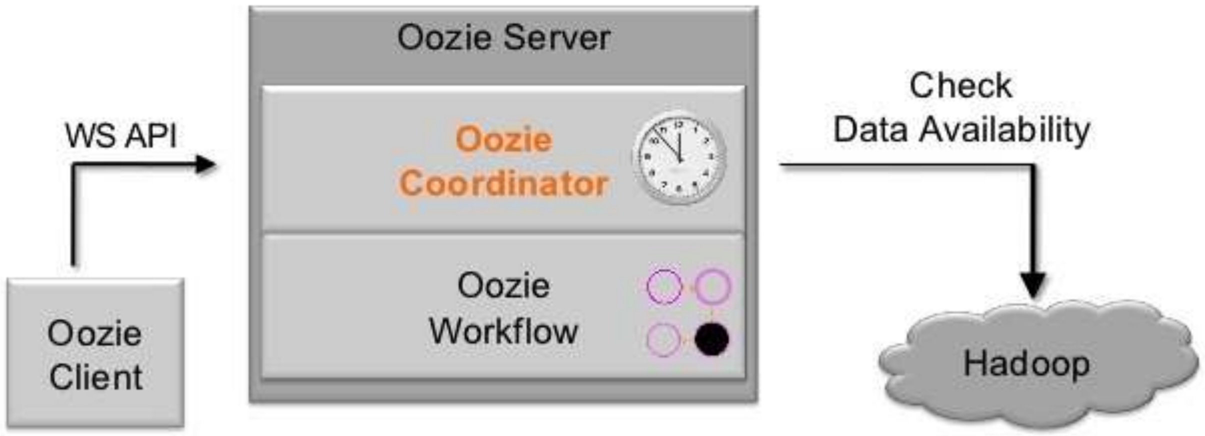
- Composed of 2 parts:
  - Workflow engine
    - Stores and runs workflows composed of different types of Hadoop jobs
  - Coordinator engine
    - Runs workflow jobs based on predefined schedules and data availability

Workflow

- Workflow is a DAG(Directed Acyclic Graph) of *action nodes* and *control-flow nodes*.
- *Action node*
  - performs a workflow task, such as moving files in HDFS, running a MapReduce, Streaming, Pig, or Hive job
- *Control-flow node*
  - governs the workflow execution between actions

A Schedular

- Oozie executes workflow based on:
  - Time Dependency (Frequency)
  - Data Dependency



Oozie Server Setup

- Oozie is distributed as two separate packages, a client package (oozie-client) and a server package (oozie).
- We will install oozie server which also installs oozie-client.
- **$ yum –y install oozie**
- When you install Oozie from an RPM, Oozie server creates all configuration, documentation and runtime files in the standard Unix directories, as follows:

Oozie Server Setup

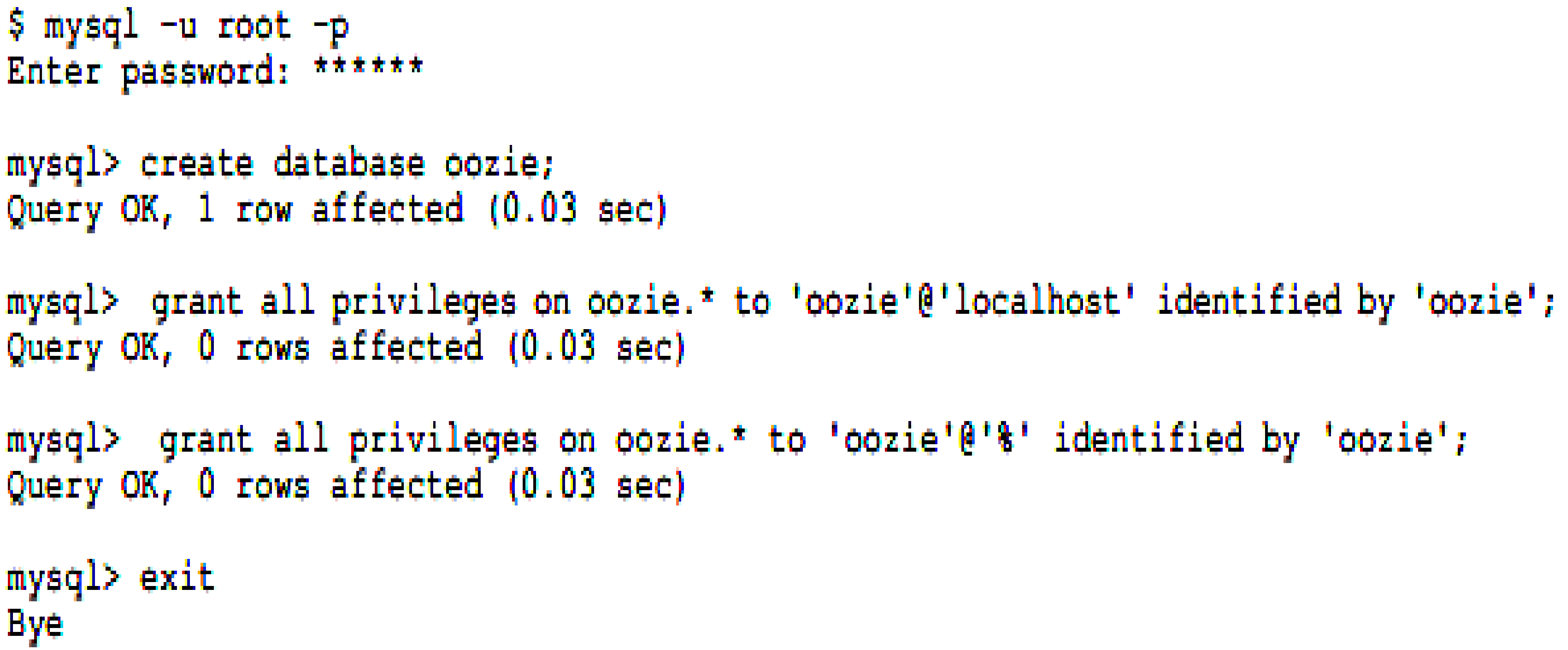
Type of File	Where installed
Binaries	/usr/lib/oozie/
Configuration	/etc/oozie/conf/
Documentation	/user/share/doc/oozie/
Examples	/user/share/doc/oozie/
Sharelib TAR.GZ	/usr/lib/oozie/
Data	/var/lib/oozie/
Logs	/var/log/oozie/

Configuring Oozie to Use MySQL

- Oozie needs a database to store all the workflow job information
- We will be configuring it to use Mysql as database
- Step 1: Install and start MySQL 5.x
- **$ yum –y install mysql-server**

Configuring Oozie to Use MySQL

- Step 2: Create the Oozie database and Oozie MySQL user



```
$ mysql -u root -p
Enter password: ******

mysql> create database oozie;
Query OK, 1 row affected (0.03 sec)

mysql>  grant all privileges on oozie.* to 'oozie'@'localhost' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql>  grant all privileges on oozie.* to 'oozie'@'%' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql> exit
Bye
```

Configuring Oozie to Use MySQL

- Step 3: Configure Oozie to use MySQL
  - Edit properties in the oozie-site.xml file as follows:

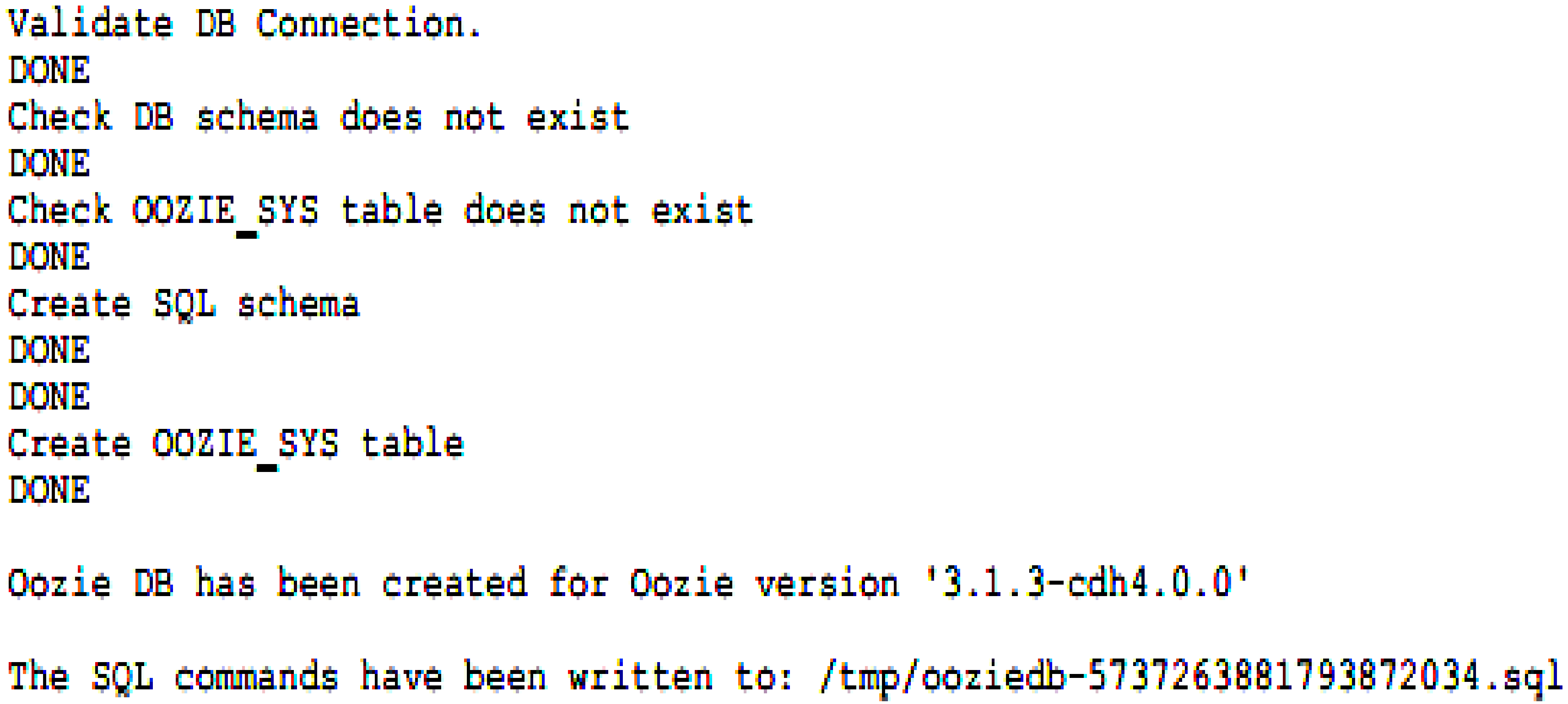
```
...
    <property>
        <name>oozie.service.JPAService.jdbc.driver</name>
        <value>com.mysql.jdbc.Driver</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.url</name>
        <value>jdbc:mysql://localhost:3306/oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.password</name>
        <value>oozie</value>
    </property>
    ...
```

Configuring Oozie to Use MySQL

- Step 4: Add the MySQL JDBC driver JAR to Oozie
  - **$ ln -s /usr/share/java/mysql-connector-java.jar /var/lib/oozie/mysql-connector-java.jar**
- Step 5:Creating the Oozie Database Schema
- After configuring Oozie database information and creating the corresponding database, create the Oozie database schema. Oozie provides a database tool for this purpose.
  - **$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh create –run**

Configuring Oozie to Use MySQL

- You should see output such as the following:



```
Validate DB Connection.
DONE
Check DB schema does not exist
DONE
Check OOZIE_SYS table does not exist
DONE
Create SQL schema
DONE
DONE
Create OOZIE_SYS table
DONE

Oozie DB has been created for Oozie version '3.1.3-cdh4.0.0'

The SQL commands have been written to: /tmp/ooziedb-5737263881793872034.sql
```

Enabling the Oozie Web Console

- By default Oozie does not enable web console. Following steps must be followed to enable it
- Step 1: Download the Library
  - **$ wget http://dev.sencha.com/deploy/ext-2.2.zip**
- Step 2: Install the Library
  - **$ unzip ext-2.2.zip**
  - **$ cp -r ext-2.2 /var/lib/oozie/**

Installing the Oozie ShareLib in HDFS

- The Oozie installation bundles Oozie ShareLib, which contains all of the necessary JARs to enable workflow jobs to run streaming, DistCp, Pig, Hive, and Sqoop actions.
- ShareLib must be copied in the home directory of oozie user in HDFS:
  - **$ sudo –u hdfs hadoop fs –mkdir /user/oozie**
  - **$ sudo –u hdfs hadoop fs –chown oozie:oozie /user/oozie**
  - **$ mkdir /tmp/ooziesharelib**
  - **$ cd /tmp/ooziesharelib**
  - **$ tar –xzf /user/lib/oozie/oozie-sharelib.tar.gz**
  - **$ sudo –u oozie hadoop fs –put share /user/oozie/share**

Starting, Stopping, and Accessing the Oozie Server

- Starting the Oozie Server
  - **$ service oozie start**
- Stopping the Oozie Server
  - **$ service oozie stop**
- Accessing the Oozie Server with the Oozie Client
  - The Oozie client is a command-line utility that interacts with the Oozie server via the Oozie web-services API
  - Use the /usr/bin/oozie script to run the Oozie client.
  - For example, if you want to invoke the client on the same machine where the Oozie server is running:
  - **$ oozie admin –oozie http://localhost:11000/oozie -status**
    - System mode: NORMAL
- Accessing the Oozie Server with a Browser
  - If you have enabled the Oozie web console by adding the ExtJS library, you can connect to the console at
  - **http://localhost:11000/oozie**

Defining an Oozie Workflow

- Workflow definitions are written in XML using the Hadoop Process Definition Language
- Consists of 2 components
  - Control Node
    - Start
    - End
    - Decision
    - Fork
    - Join
    - Kill
  - Action Node
    - Map-reduce
    - Pig, etc..

Control Flow Nodes

- **Start Control Node**
  - The start node is the entry point for a workflow job
  - It indicates the first workflow node the workflow job must transition to
  - When a workflow is started, it automatically transitions to the node specified in the start
  - A workflow definition must have one start node
- Syntax
  - <workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
  - ...
  - <start to="[NODE-NAME]"/>
  - ...
  - </workflow-app>

The node name(action) from which the workflow should start

Control Flow Nodes

- **End Control Node**
  - The end node is the end for a workflow job
  - Indicates that the workflow job has completed successfully
  - When a workflow job reaches the end it finishes successfully
  - If one or more actions started by the workflow job are executing when the end node is reached, the actions will be killed
  - A workflow definition must have one end node.
- Syntax
  - <workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
  - ...
  - <end name="[NODE-NAME]"/>
  - ...
  - </workflow-app>

The node name(action) on which the workflow should end

Control Flow Nodes

- **Kill Control Node**
  - The kill node allows a workflow job to kill itself
  - When a workflow job reaches the kill it finishes in error
  - If one or more actions started by the workflow job are executing when the kill node is reached, the actions will be killed
  - A workflow definition may have zero or more kill nodes
- Syntax

```
<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
...
<kill name="[NODE-NAME]">
<message>[MESSAGE-TO-LOG]</message>
</kill>
...
</workflow-app>
```

If the workflow execution reaches this node the workflow will be

Control Flow Nodes

- **Decision Control Node**
  - Enables a workflow to make a selection on the execution path to follow
  - The behavior of a decision node can be seen as a switch-case statement
  - Predicates are evaluated in order or appearance until one of them evaluates to true and the corresponding transition is taken
  - If none of the predicates evaluates to true the default transition is taken
- Syntax
  - <decision name="[NODE-NAME]">
  - <switch> <case to="[NODE_NAME]">[PREDICATE]</case>
  - ...
  - <case to="[NODE_NAME]">[PREDICATE]</case>
  - <default to="[NODE_NAME]"/>
  - </switch> </decision>

Switch case to decide between the execution of nodes

Control Flow Nodes

- **Fork and Join Control Nodes**
  - A fork node splits one path of execution into multiple concurrent paths of execution
  - A join node waits until every concurrent execution path of a previous fork node arrives to it
  - The fork and join nodes must be used in pairs
  - Actions at fork runs parallel ✓
- Syntax
  - <fork name="[FORK-NODE-NAME]">
  - <path start="[NODE-NAME]" />
  - ...
  - <path start="[NODE-NAME]" />
  - </fork>
  - <join name="[JOIN-NODE-NAME]" to="[NODE-NAME]" /> ✓

Workflow Action Nodes

- Action Basis
  - Action Computation/Processing is always remote
  - Actions are Asynchronous
  - Actions have two transitions, ok and error
  - Action Recovery
    - Oozie provides recovery capabilities when starting or ending actions
    - Recovery strategies differ on the nature of failure
    - For non-transient failures action is suspended
    - For transient failures Oozie will perform retries after a fixed time interval

Workflow Action Nodes

- **Fs(HDFS) Action**
  - The fs action allows to manipulate files and directories in HDFS from a workflow application
  - The supported commands are move , delete and mkdir
  - The FS commands are executed synchronously from within the FS action
  - Syntax
    - <action name="[NODE-NAME]">
    - <fs>
    - **<delete path='[PATH]'/>**
    - ...
    - **<mkdir path='[PATH]'/>**
    - ...
    - **<move source='[SOURCE-PATH]' target='[TARGET-PATH]'/>**
    - </fs> <ok to="[NODE-NAME]"/>
    - <error to="[NODE-NAME]"/>
    - </action>

Workflow Action Nodes

Pig Action

- The pig action starts a Pig job
- The workflow job will wait until the pig job completes before continuing to the next action
- The pig action has to be configured with the job-tracker, name-node, pig script and the necessary parameters and configuration to run the Pig job.
- The configuration properties are loaded in the following order, job-xml and configuration , and later values override earlier values.
- Hadoop mapred.job.tracker and fs.default.name properties must not be present in the job-xml and inline configuration

Workflow Action Nodes

- Pig Action
  - Syntax
    - <pig>
    - <job-tracker>[JOB-TRACKER]</job-tracker>
    - <name-node>[NAME-NODE]</name-node>
    - <prepare> <delete path="[PATH]"/>
    - ... <mkdir path="[PATH]"/> ... </prepare>
    - <job-xml>[JOB-XML-FILE]</job-xml>

necessary configuration

Workflow Action Nodes

- <configuration>
- <property>
- <name>[PROPERTY-NAME]</name> <value>[PROPERTY-VALUE]</value>
- </property>
- ... </configuration>
- <script>[PIG-SCRIPT]</script>
- <param>[PARAM-VALUE]</param>
- ... <param>[PARAM-VALUE]</param>
- <argument>[ARGUMENT-VALUE]</argument>
- ... <argument>[ARGUMENT-VALUE]</argument>
- <file>[FILE-PATH]</file>
- ... <archive>[FILE-PATH]</archive>
- ... </pig>

Cluster wide configura tion

Pig script, its parameters and arguments

Oozie Job States

- A workflow job can have be in any of the following states:
  - **PREP**: When a workflow job is first created it will be in PREP state. The workflow job is defined but it is not running.
  - **RUNNING**: When a CREATED workflow job is started it goes into RUNNING state, it will remain in RUNNING state while it does not reach its end state, ends in error or it is suspended.
  - **SUSPENDED**: A RUNNING workflow job can be suspended, it will remain in SUSPENDED state until the workflow job is resumed or it is killed.
  - **SUCCEEDED**: When a RUNNING workflow job reaches the end node it ends reaching the SUCCEEDED final state.
  - **KILLED**: When a CREATED , RUNNING or SUSPENDED workflow job is killed by an administrator or the owner via a request to Oozie the workflow job ends reaching the KILLED final state.
  - **FAILED**: When a RUNNING workflow job fails due to an unexpected error it ends reaching the FAILED final state.

Example

- $ cp /usr/share/doc/oozie-3.3.2+49/oozie-examples.tar.gz .
- $ tar -xvf oozie-examples.tar.gz
- $ hadoop fs -put examples/ .
- $ cd examples/apps/pig/
- $ oozie job --oozie http://localhost:11000/oozie -config job.properties –run
- $ oozie job -oozie http://localhost:11000/oozie -info <job_id>

Understand the Example

- Pig Script
  - **$ cat id.pig**

```
A = load '$INPUT' using PigStorage(':');
B = foreach A generate $0 as id;
store B into '$OUTPUT' USING PigStorage();
```

Understand the Example

- Workflow xml
  - **$ cat workflow.xml**

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="pig-wf">
    <start to="pig-node"/>
    <action name="pig-node">
        <pig>
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <prepare>
                <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/pig"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
                    <name>mapred.compress.map.output</name>
                    <value>true</value>
                </property>
            </configuration>
```

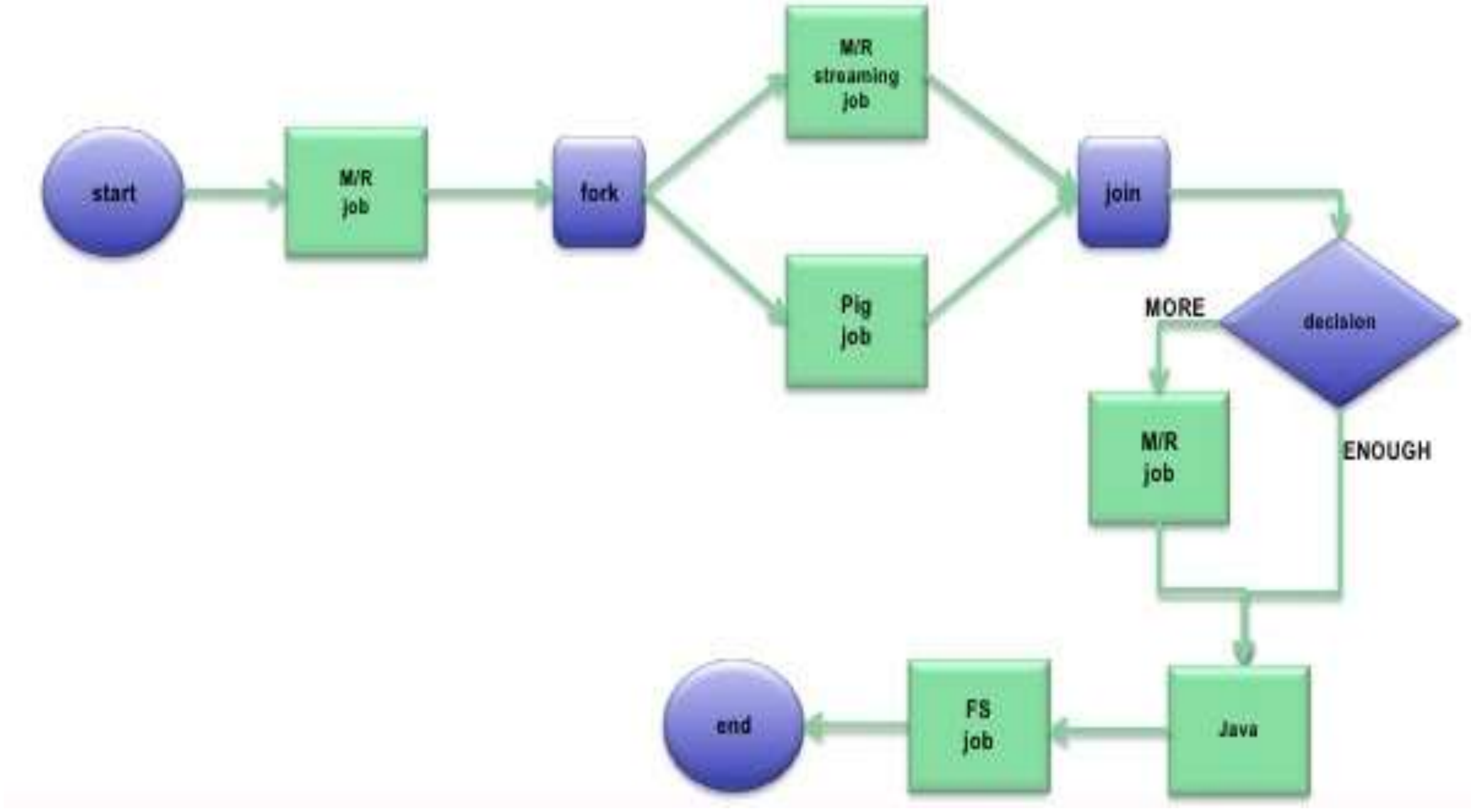
Understand the Example

- <script>id.pig</script>
-             <param>INPUT=/user/${wf:user()}/${examplesRoot}/input-data/text</param>
-             <param>OUTPUT=/user/${wf:user()}/${examplesRoot}/output-data/pig</param>
-         </pig>
-         <ok to="end"/>
-         <error to="fail"/>
-     </action>
-     <kill name="fail">
-         <message>Pig failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
-     </kill>
-     <end name="end"/>
- </workflow-app>

Understand the Example

- $ cat job.properties
  - nameNode=hdfs://localhost:8020
  - jobTracker=localhost:8021
  - queueName=default
  - examplesRoot=examples
  - oozie.use.system.libpath=true
  - oozie.wf.application.path=${nameNode}/user/${user.name}/${examplesRoot}/apps/pig

A Workflow Job



A Workflow Job

- $ cd /root/examples/apps/demo
- $ cat workflow.xml

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="demo-wf">
   <start to="cleanup-node"/>
   <action name="cleanup-node">
      <fs>
         <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-
data/demo"/>
      </fs>
      <ok to="fork-node"/>
      <error to="fail"/>
   </action>

   <fork name="fork-node">
      <path start="pig-node"/>
      <path start="streaming-node"/>
   </fork>
```

A Workflow Job

```
<action name="pig-node">
        <pig>
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <prepare>
                <delete
path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-
data/demo/pig-node"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
```

A Workflow Job

- <property>
- <name>mapred.map.output.compress</name>
- <value>false</value>
- </property>
- </configuration>
- <script>id.pig</script>
- <param>INPUT=/user/${wf:user()}/${examplesRoot}/input-data/text</param>
- <param>OUTPUT=/user/${wf:user()}/${examplesRoot}/output-data/demo/pig-node</param>
- </pig>
- <ok to="join-node"/>
- <error to="fail"/>
- </action>

A Workflow Job

```
<action name="streaming-node">
    <map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
        <prepare>
            <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-
data/demo/streaming-node"/>
        </prepare>
        <streaming>
            <mapper>/bin/cat</mapper>
            <reducer>/usr/bin/wc</reducer>
        </streaming>
        <configuration>
            <property>
                <name>mapred.job.queue.name</name>
                <value>${queueName}</value>
            </property>
```

A Workflow Job

```
<property>
    <name>mapred.input.dir</name>
    <value>/user/${wf:user()}/${examplesRoot}/input-data/text</value>
</property>
<property>
    <name>mapred.output.dir</name>
    <value>/user/${wf:user()}/${examplesRoot}/output-data/demo/streaming-node</value>
</property>
</configuration>
</map-reduce>
<ok to="join-node"/>
<error to="fail"/>
</action>
```

A Workflow Job

- <join name="join-node" to="mr-node"/>

- <action name="mr-node">
-         <map-reduce>
-             <job-tracker>${jobTracker}</job-tracker>
-             <name-node>${nameNode}</name-node>
-             <prepare>
-                 <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/demo/mr-node"/>
-             </prepare>
-             <configuration>
-                 <property>
-                     <name>mapred.job.queue.name</name>
-                     <value>${queueName}</value>
-                 </property>

A Workflow Job

```
<property>
    <name>mapred.mapper.class</name>

<value>org.apache.oozie.example.DemoMapper</value>
</property>
<property>
    <name>mapred.mapoutput.key.class</name>
    <value>org.apache.hadoop.io.Text</value>
</property>
<property>
    <name>mapred.mapoutput.value.class</name>
    <value>org.apache.hadoop.io.IntWritable</value>
</property>
```

A Workflow Job

```
▪ <property>
▪             <name>mapred.reducer.class</name>
▪             <value>org.apache.oozie.example.DemoReducer</value>
▪           </property>
▪           <property>
▪             <name>mapred.map.tasks</name>
▪             <value>1</value>
▪           </property>
▪           <property>
▪             <name>mapred.input.dir</name>
▪             <value>/user/${wf:user()}/${examplesRoot}/output-data/demo/pig-
  node,/user/${wf:user()}/${examplesRoot}/output-data/demo/streaming-node</value>
▪           </property>
▪           <property>
▪             <name>mapred.output.dir</name>
▪             <value>/user/${wf:user()}/${examplesRoot}/output-data/demo/mr-node</value>
▪           </property>
▪         </configuration>
▪       </map-reduce>
▪       <ok to="decision-node"/>
▪       <error to="fail"/>
▪     </action>
```

A Workflow Job

```
<decision name="decision-node">
    <switch>
        <case to="hdfs-node">${fs:exists(concat(concat(concat(concat(concat(nameNode, '/user/'), wf:user()), '/'), examplesRoot), '/output-data/demo/mr-node')) == "true"}</case>
        <default to="end"/>
    </switch>
</decision>
```

A Workflow Job

- <action name="hdfs-node">
-         <fs>
-             <move source="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/demo/mr-node"
-                   target="/user/${wf:user()}/${examplesRoot}/output-data/demo/final-data"/>
- &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</fs>
-         <ok to="end"/>
-         <error to="fail"/>
- &nbsp;   </action>

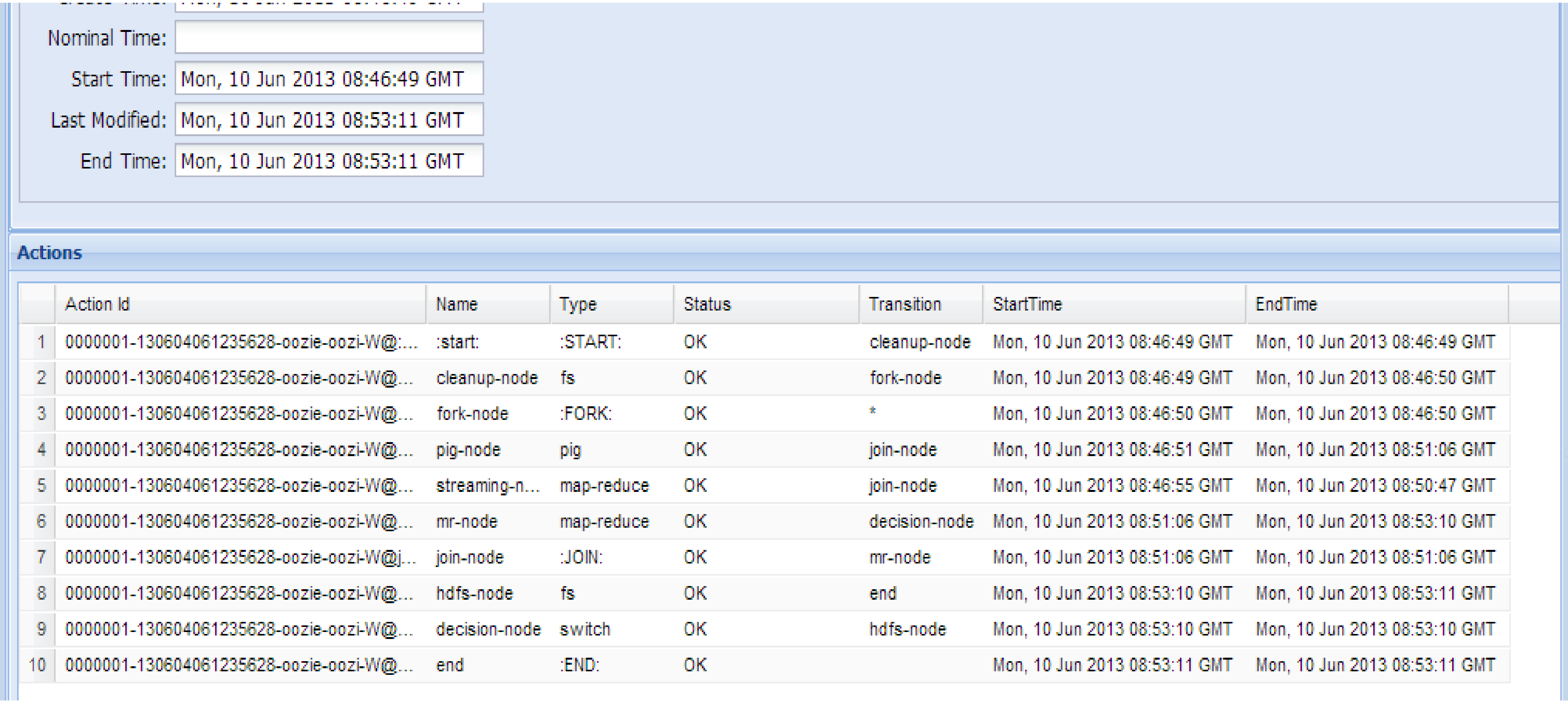
-     <kill name="fail">
-         <message>Demo workflow failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
- &nbsp;&nbsp;&nbsp;&nbsp;</kill>

-     <end name="end"/>

- </workflow-app>

A Workflow Job

- At the end of the Job Completion you will see something like this:



Nominal Time:	
Start Time:	Mon, 10 Jun 2013 08:46:49 GMT
Last Modified:	Mon, 10 Jun 2013 08:53:11 GMT
End Time:	Mon, 10 Jun 2013 08:53:11 GMT

Actions

	Action Id	Name	Type	Status	Transition	StartTime	EndTime
1	0000001-130604061235628-oozie-oozi-W@:...	:start:	:START:	OK	cleanup-node	Mon, 10 Jun 2013 08:46:49 GMT	Mon, 10 Jun 2013 08:46:49 GMT
2	0000001-130604061235628-oozie-oozi-W@...	cleanup-node	fs	OK	fork-node	Mon, 10 Jun 2013 08:46:49 GMT	Mon, 10 Jun 2013 08:46:50 GMT
3	0000001-130604061235628-oozie-oozi-W@...	fork-node	:FORK:	OK	*	Mon, 10 Jun 2013 08:46:50 GMT	Mon, 10 Jun 2013 08:46:50 GMT
4	0000001-130604061235628-oozie-oozi-W@...	pig-node	pig	OK	join-node	Mon, 10 Jun 2013 08:46:51 GMT	Mon, 10 Jun 2013 08:51:06 GMT
5	0000001-130604061235628-oozie-oozi-W@...	streaming-n...	map-reduce	OK	join-node	Mon, 10 Jun 2013 08:46:55 GMT	Mon, 10 Jun 2013 08:50:47 GMT
6	0000001-130604061235628-oozie-oozi-W@...	mr-node	map-reduce	OK	decision-node	Mon, 10 Jun 2013 08:51:06 GMT	Mon, 10 Jun 2013 08:53:10 GMT
7	0000001-130604061235628-oozie-oozi-W@j...	join-node	:JOIN:	OK	mr-node	Mon, 10 Jun 2013 08:51:06 GMT	Mon, 10 Jun 2013 08:51:06 GMT
8	0000001-130604061235628-oozie-oozi-W@...	hdfs-node	fs	OK	end	Mon, 10 Jun 2013 08:53:10 GMT	Mon, 10 Jun 2013 08:53:11 GMT
9	0000001-130604061235628-oozie-oozi-W@...	decision-node	switch	OK	hdfs-node	Mon, 10 Jun 2013 08:53:10 GMT	Mon, 10 Jun 2013 08:53:10 GMT
10	0000001-130604061235628-oozie-oozi-W@...	end	:END:	OK		Mon, 10 Jun 2013 08:53:11 GMT	Mon, 10 Jun 2013 08:53:11 GMT