SO WHAT DO WE HAVE SO FAR...

Hadoop installed and running

Oozie installed and running

Let's actually do something with these...

WORKFLOWS

WORKFLOWS

This is a core Oozie building block

A workflow brings together different **actions**

These are the individual units of work which make up the workflow

WORKFLOWS

actions

An action does the actual processing in a workflow

MR jobs

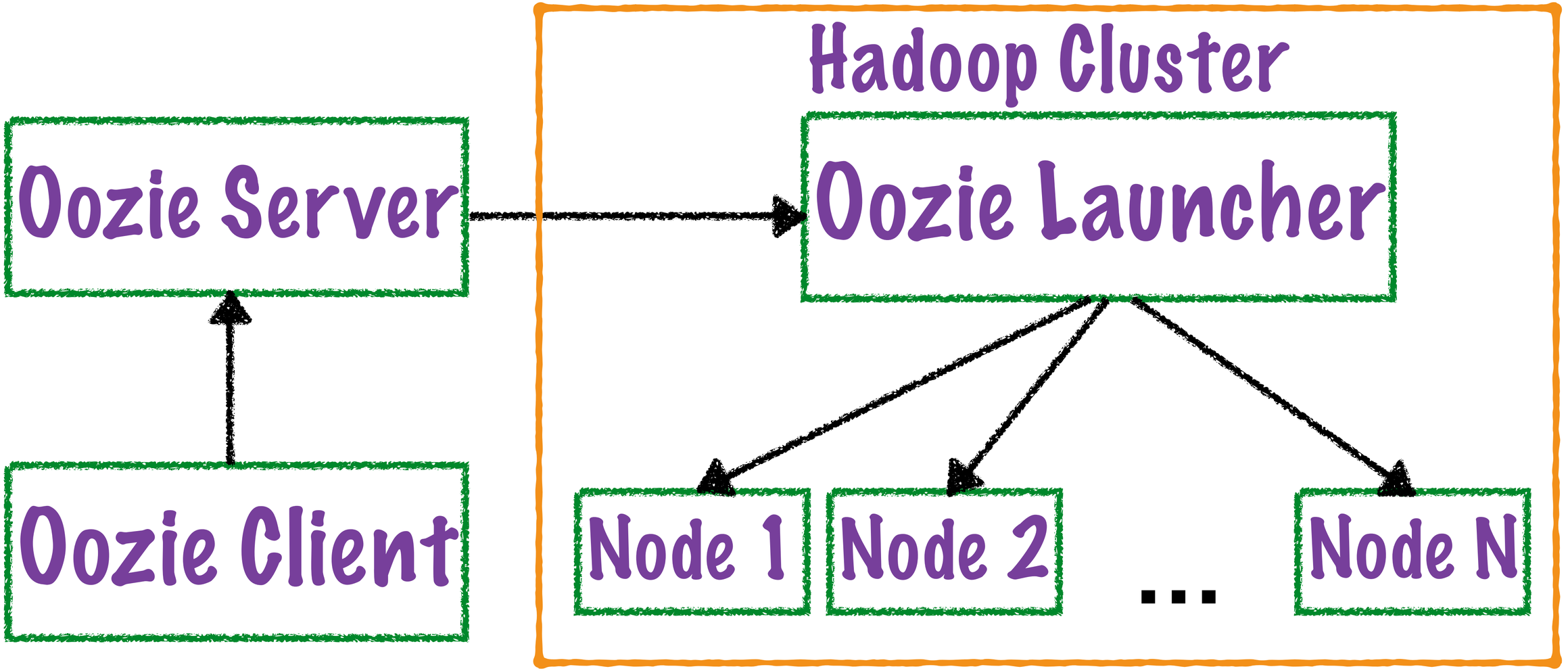
Hive jobs

Pig jobs

Scripts or Java programs

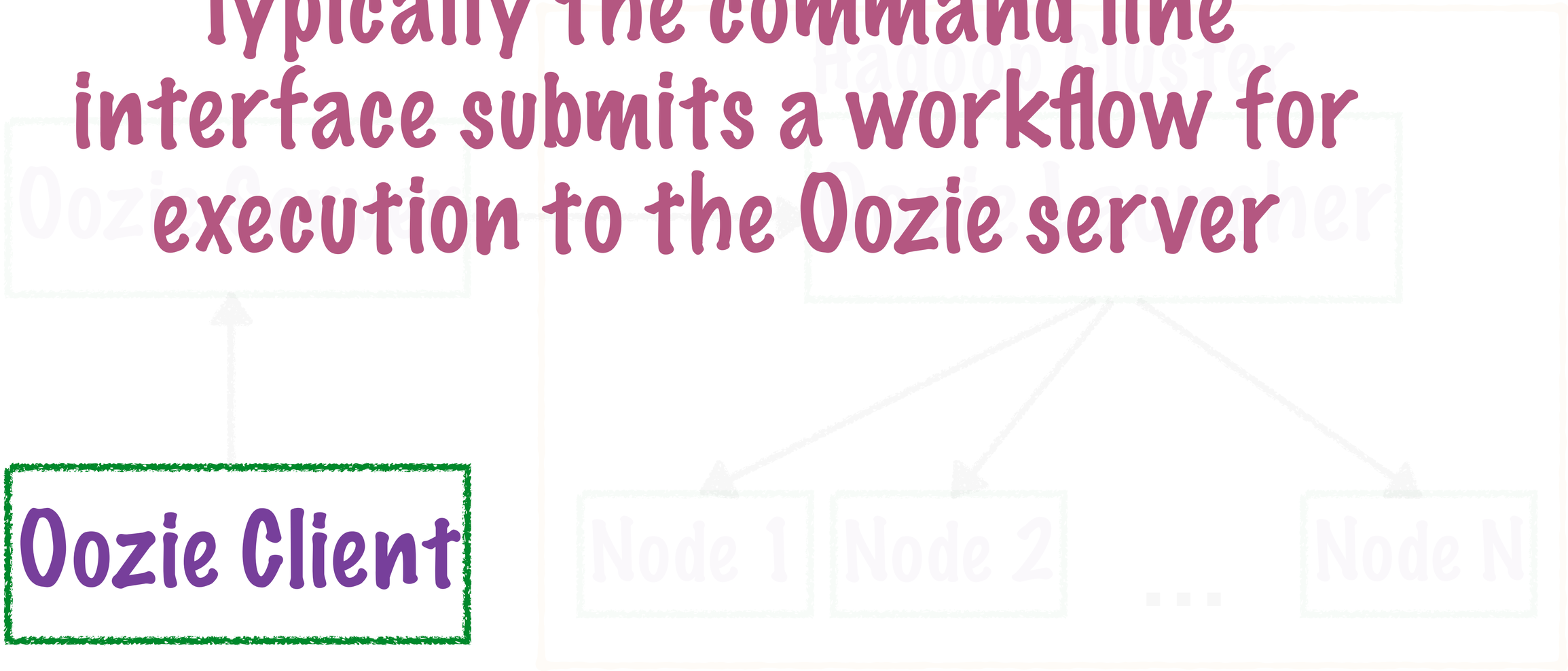
WORKFLOWS

How is an action executed on Oozie?



WORKFLOWS

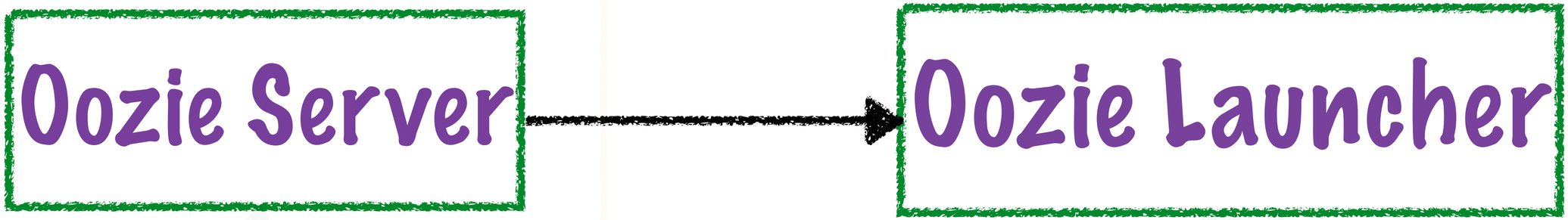
Typically the command line interface submits a workflow for execution to the Oozie server



Oozie Client

WORKFLOWS

The server runs a Launcher MapReduce to execute Oozie jobs

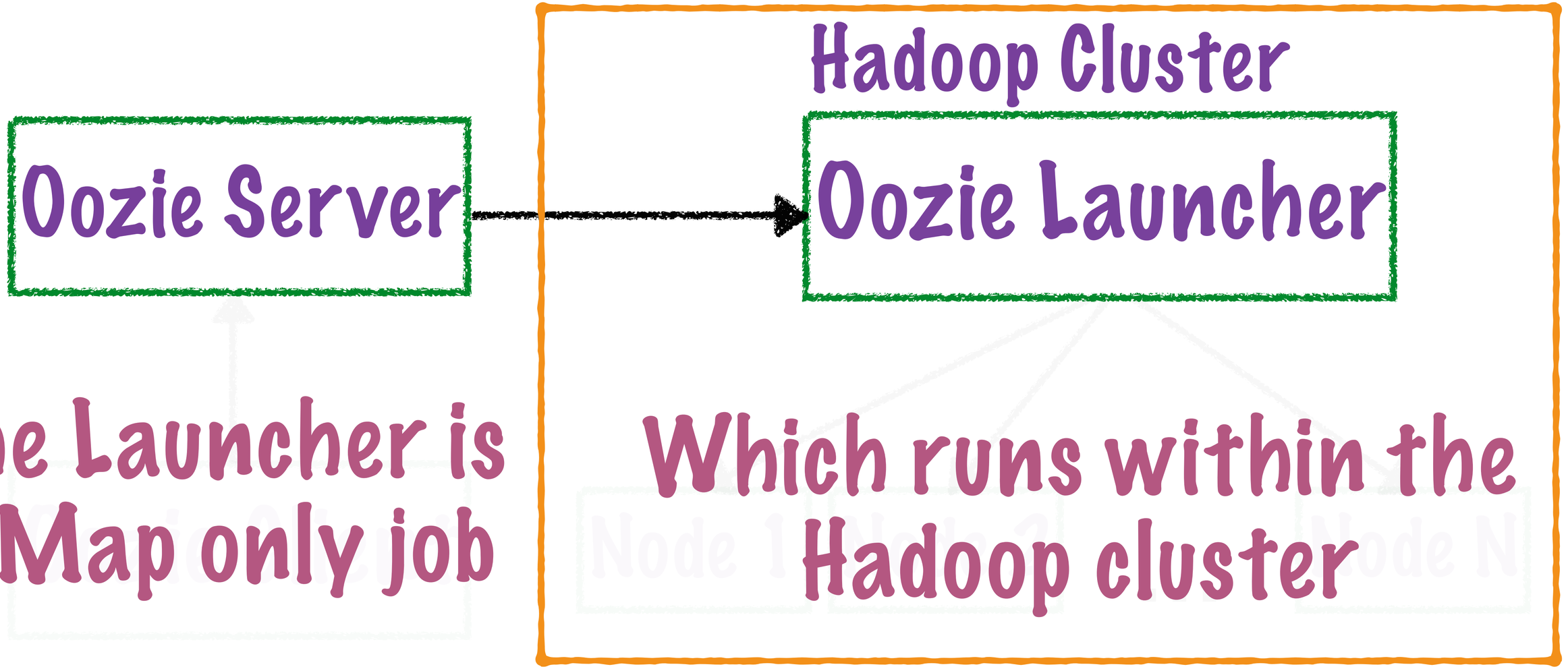


This MR invokes the correct libraries for the action

Hadoop, Pig or Hive libraries

WORKFLOWS

Launcher MapReduce to execute Oozie jobs



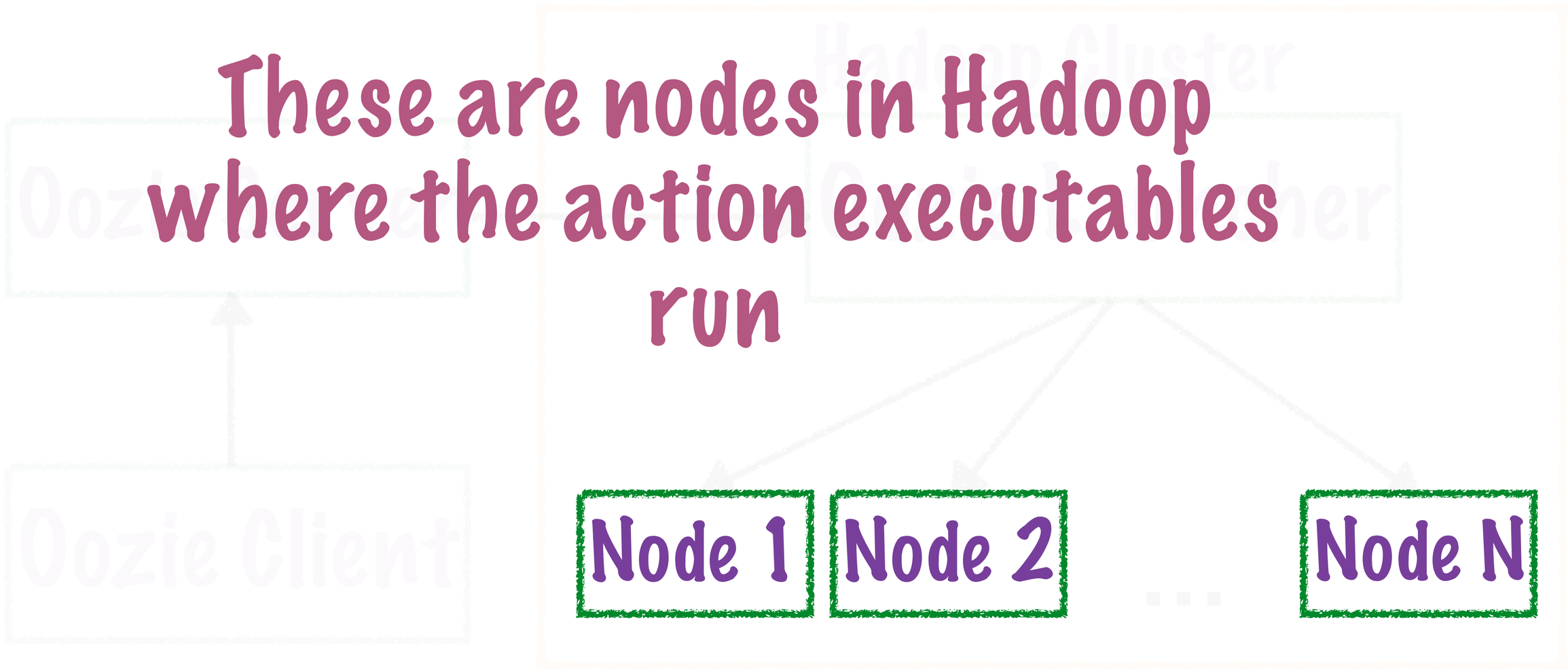
The Launcher is a Map only job

Which runs within the Hadoop cluster

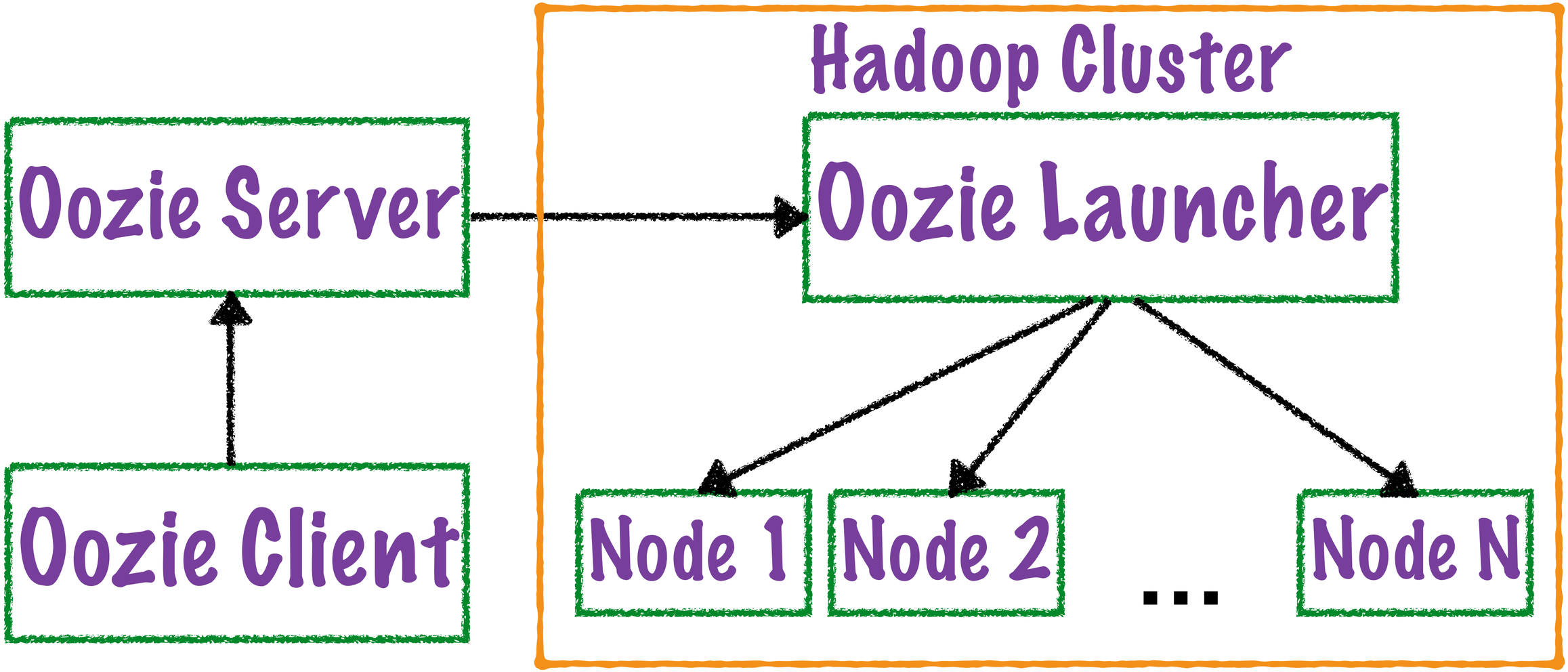
WORKFLOWS

Launcher MapReduce to execute Oozie jobs

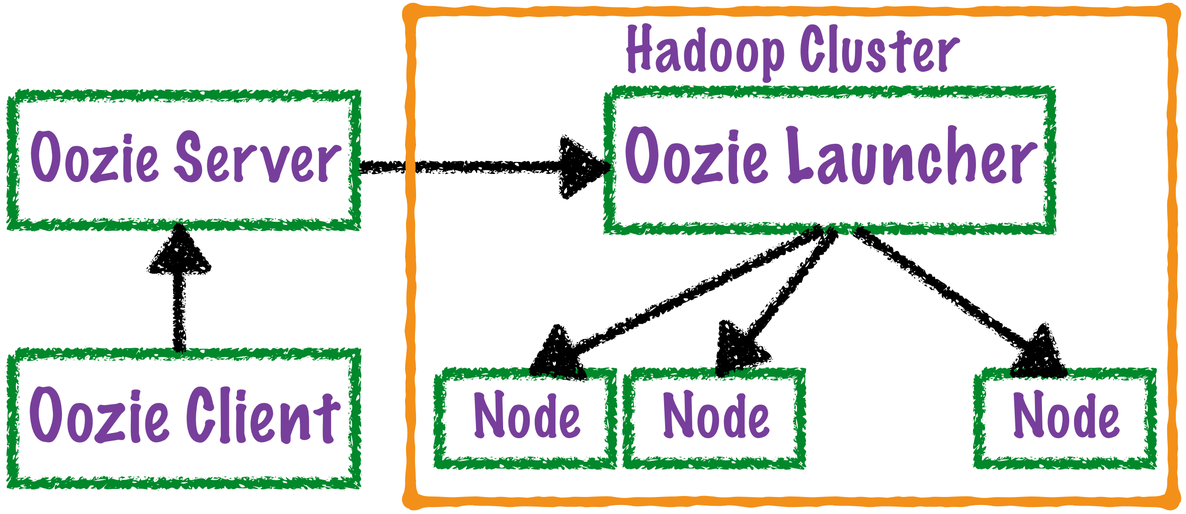
These are nodes in Hadoop where the action executables run



WORKFLOWS


Hadoop Cluster
Oozie Server
Oozie Launcher
Oozie Client
Node 1
Node 2
...
Node N

WORKFLOWS



This architecture **reduces the load** on the Oozie server

User code in the actions is **isolated** from Oozie code

This makes the Oozie server more **stable** and it can remain stateless

WORKFLOWS

Insert video here 1 & 2

WORKFLOWS

We've run our very first MapReduce workflow, now what exactly did we do to achieve that?

WORKFLOWS

Let's start with the job.properties file

This file lives on the local file system not on HDFS

```
oozie job -oozie http://localhost:11000/oozie
-config /Users/jananiravi/Desktop/
iMovieLibrary/Oozie/Workflows/MapReduce/
job.properties -run
```

WORKFLOWS

Let's start with the job.properties file

```
oozie job -oozie http://localhost:11000/oozie -config /Users/jananiravi/
Desktop/iMovieLibrary/Oozie/Workflows/MapReduce/job.properties -run
```

This contains the job configuration to send to Oozie to invoke the workflow

WORKFLOWS

Let's start with the job.properties file

```
oozie job -oozie http://localhost:11000/oozie -config /Users/jananiravi/
Desktop/iMovieLibrary/Oozie/Workflows/MapReduce/job.properties -run
```

This contains the arguments for the Oozie workflow application

WORKFLOWS
job.properties

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.application.path=${nameNode}/user/$
{user.name}/${oozieRoot}/map-reduce/
workflow.xml
```

WORKFLOWS

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.application.path=hdfs://localhost:9000/user/
${user.name}/${oozieRoot}/map-reduce/
workflow.xml
```

This is location of the HDFS name node in the Hadoop installation

WORKFLOWS

```
jobTracker=localhost:8032
```

This is the port where the Yarn Resource Manager runs (in the latest versions of Hadoop)

WORKFLOWS

```
jobTracker=localhost:8032
```

In earlier versions it referred to the JobTracker which helps run MR jobs

WORKFLOWS

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.application.path=${nameNode}/user/$
{user.name}/${oozieRoot}/apps/workflow
workflow.xml
```

These are parameters used by the Oozie application

WORKFLOWS

These are parameters used by the Oozie application

```
nameNode=hdfs://localhost:
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.applica
{user.name}/${o
workflow.
```

Use of parameters make applications flexible and changes are easier to make

WORKFLOWS

This tells Oozie to look for JARs and libraries in the `sharelib` path

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie

oozie.system.libpath=true

oozie.wf.application.path=${nameNode}/user/$
[illegible]/oozie/map-reduce/
workflow.xml
```

We set this up during the Oozie install

Hive and DistCp jobs require this flag to be set

WORKFLOWS

This points to the application's root directory where the workflow files live

```
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.application.path=${nameNode}/user/$
{user.name}/${oozieRoot}/map-reduce/
workflow.xml
```

WORKFLOWS

This points to the application's root directory where the workflow files live

This tell the Oozie job where to find the files to run the workflow

```
oozie.wf.application.path=${nameNode}/user/$
{user.name}/${oozieRoot}/map-reduce/
workflow.xml
```

WORKFLOWS

Note the use of the parameters we just set up

```
nameNode=hdfs://localhost:8020
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.application.path=${nameNode}/user/${user.name}/${oozieRoot}/map-reduce/workflow.xml
```

WORKFLOWS

This points to the application's root directory where the workflow files live

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.system.libpath=true

oozie.wf.application.path=${nameNode}/user/$
{user.name}/${oozieRoot}/map-reduce/
workflow.xml
```

WORKFLOWS

The job.properties pass in arguments to the Oozie application to configure how it runs

Now let's look at our very first Oozie application, the Workflow

WORKFLOWS

```xml
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start 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()}/${oozieRoot}/map-reduce/output-data"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.apache.oozie.example.SampleMapper</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.apache.oozie.example.SampleReducer</value>
                </property>
                <property>
                    <name>mapred.map.tasks</name>
                    <value>1</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start to="mr-node"/>
    <action name="mr-node">
        [redacted]
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[$
{wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

This is the basic structure of the workflow XML

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start 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()}/${oozieRoot}/map-reduce/output-data"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.apache.oozie.example.SampleMapper</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.apache.oozie.example.SampleReducer</value>
                </property>
                <property>
                    <name>mapred.map.tasks</name>
                    <value>1</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to="end" />
        <error to="fail" />
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end" />
</workflow-app>
```

The workflow-app element holds the entire workflow

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start 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()}/${oozieRoot}/map-reduce/output-data"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.apache.oozie.example.SampleMapper</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.apache.oozie.example.SampleReducer</value>
                </property>
                <property>
                    <name>mapred.map.tasks</name>
                    <value>1</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to="end" />
        <error to="fail" />
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end" />
</workflow-app>
```

Oozie schemas are validated and this attribute indicates the version to use for the validation

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start to="mr-node"/>
    <action name="mr-node">
        ...
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[$
{wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

This kick starts the workflow

The "to" attribute sends the control to the node to process

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start to="mr-node"/>
    <action name="mr-node">
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[$
{wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

The action element holds the configuration of the job to execute

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start 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()}/${oozieRoot}/map-reduce/output-data"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.apache.oozie.example.SampleMapper</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.apache.oozie.example.SampleReducer</value>
                </property>
                <property>
                    <name>mapred.map.tasks</name>
                    <value>1</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to="end" />
        <error to="fail" />
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

This element kills the job, usually called when the job fails for any reason

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start to="mr-node"/>
    <action name="mr-node">
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[$
{wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

Element indicates the end of the job

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.2" name="map-reduce-wf">
    <start 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()}/${oozieRoot}/map-reduce/output-data"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
                <property>
                    <name>mapred.mapper.class</name>
                    <value>org.apache.oozie.example.SampleMapper</value>
                </property>
                <property>
                    <name>mapred.reducer.class</name>
                    <value>org.apache.oozie.example.SampleReducer</value>
                </property>
                <property>
                    <name>mapred.map.tasks</name>
                    <value>1</value>
                </property>
                <property>
                    <name>mapred.input.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
                </property>
                <property>
                    <name>mapred.output.dir</name>
                    <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
                </property>
            </configuration>
        </map-reduce>
        <ok to="end" />
        <error to="fail" />
    </action>
    <kill name="fail">
        <message>Map/Reduce failed, error message[$
{wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

This is the basic structure of the workflow XML

WORKFLOWS

```
<action name="mr-node">
    <map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
        <prepare>
            <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
        </prepare>
        <configuration>
            <property>
                <name>mapred.job.queue.name</name>
                <value>${queueName}</value>
            </property>
            <property>
                <name>mapred.mapper.class</name>
                <value>org.apache.oozie.example.SampleMapper</value>
            </property>
            <property>
                <name>mapred.reducer.class</name>
                <value>org.apache.oozie.example.SampleReducer</value>
            </property>
            <property>
                <name>mapred.map.tasks</name>
                <value>1</value>
            </property>
            <property>
                <name>mapred.input.dir</name>
                <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
            </property>
            <property>
                <name>mapred.output.dir</name>
                <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
            </property>
        </configuration>
    </map-reduce>
    <ok to="end"/>
    <error to="fail"/>
</action>
```

This is the configuration for the job that is executed in this workflow

WORKFLOWS

```
<action name="mr-node">
     <map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
        <prepare>
            <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
        </prepare>
        <configuration>
            <property>
                <name>mapred.job.queue.name</name>
                <value>${queueName}</value>
            </property>
            <property>
                <name>mapred.mapper.class</name>
                <value>org.apache.oozie.example.SampleMapper</value>
            </property>
            <property>
                <name>mapred.reducer.class</name>
                <value>org.apache.oozie.example.SampleReducer</value>
            </property>
            <property>
                <name>mapred.map.tasks</name>
                <value>1</value>
            </property>
            <property>
                <name>mapred.input.dir</name>
                <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
            </property>
            <property>
                <name>mapred.output.dir</name>
                <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
            </property>
        </configuration>
     </map-reduce>
      <ok to="end"/>
      <error to="fail"/>
</action>
```

The map-reduce element indicates that the action is an MR job

WORKFLOWS

```
<action name="mr-node">
     <map-reduce>
 <job-tracker>${jobTracker}</job-tracker>
<name-node>${nameNode}</name-node>
<prepare>
   <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
</prepare>
<configuration>
    <property>
        <name>mapred.job.queue.name</name>
        <value>${queueName}</value>
    </property>
    <property>
        <name>mapred.mapper.class</name>
        <value>org.apache.oozie.example.SampleMapper</value>
    </property>
    <property>
        <name>mapred.reducer.class</name>
        <value>org.apache.oozie.example.SampleReducer</value>
    </property>
    <property>
        <name>mapred.map.tasks</name>
        <value>1</value>
    </property>
    <property>
        <name>mapred.input.dir</name>
        <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
    </property>
    <property>
        <name>mapred.output.dir</name>
        <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
    </property>
</configuration>
     </map-reduce>
     <ok to="end"/>
     <error to="fail"/>
</action>
```

The job can complete successfully in which case it goes to the "ok" element

The job can fail, in which case it goes to the "error" element

WORKFLOWS

```
<action name="mr-node">
     <map-reduce>
 <job-tracker>${jobTracker}</job-tracker>
<name-node>${nameNode}</name-node>
<prepare>
    <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
</prepare>
<configuration>
    <property>
        <name>mapred.job.queue.name</name>
        <value>${queueName}</value>
    </property>
    <property>
        <name>mapred.mapper.class</name>
        <value>org.apache.oozie.example.SampleMapper</value>
    </property>
    <property>
        <name>mapred.reducer.class</name>
        <value>org.apache.oozie.example.SampleReducer</value>
    </property>
    <property>
        <name>mapred.map.tasks</name>
        <value>1</value>
    </property>
    <property>
        <name>mapred.input.dir</name>
        <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
    </property>
    <property>
        <name>mapred.output.dir</name>
        <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
    </property>
</configuration>
     </map-reduce>
     <ok to="end"/>
     <error to="fail"/>
</action>
```

The job can complete successfully in which case it goes to the "ok" element

The job can fail, in which case it goes to the "error" element

Each of these use the "to" attribute to send it to other elements

WORKFLOWS

```
<map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
    <prepare>
        <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
    </prepare>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
        <property>
            <name>mapred.mapper.class</name>
            <value>org.apache.oozie.example.SampleMapper</value>
        </property>
        <property>
            <name>mapred.reducer.class</name>
            <value>org.apache.oozie.example.SampleReducer</value>
        </property>
        <property>
            <name>mapred.map.tasks</name>
            <value>1</value>
        </property>
        <property>
            <name>mapred.input.dir</name>
            <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
        </property>
        <property>
            <name>mapred.output.dir</name>
            <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
        </property>
    </configuration>
</map-reduce>
```

These are required elements for a MR action to give it information about the Hadoop cluster

WORKFLOWS

```
<map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
    <prepare>
        <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
    </prepare>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
        <property>
            <name>mapred.mapper.class</name>
            <value>org.apache.oozie.example.SampleMapper</value>
        </property>
        <property>
            <name>mapred.reducer.class</name>
            <value>org.apache.oozie.example.SampleReducer</value>
        </property>
        <property>
            <name>mapred.map.tasks</name>
            <value>1</value>
        </property>
        <property>
            <name>mapred.input.dir</name>
            <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
        </property>
        <property>
            <name>mapred.output.dir</name>
            <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
        </property>
    </configuration>
</map-reduce>
```

Note the use of the parameters specified in the job.properties

WORKFLOWS

```
<map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
    <prepare>
        <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
    </prepare>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
        <property>
            <name>mapred.mapper.class</name>
            <value>org.apache.oozie.example.SampleMapper</value>
        </property>
        <property>
            <name>mapred.reducer.class</name>
            <value>org.apache.oozie.example.SampleReducer</value>
        </property>
        <property>
            <name>mapred.map.tasks</name>
            <value>1</value>
        </property>
        <property>
            <name>mapred.input.dir</name>
            <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
        </property>
        <property>
            <name>mapred.output.dir</name>
            <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
        </property>
    </configuration>
</map-reduce>
```

Oozie supports the JSP Expression Language syntax for variables, functions and complex parameters

WORKFLOWS

```
<action name="mr-node">
    <map-reduce>
        <prepare>
            <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
        </prepare>
</action>
```

The prepare statements are the set up before the job can run

An MR requires a non-existent output directory so we delete the directory before we run the MR

WORKFLOWS

```
<action name="mr-node">
    <map-reduce>
        <job-tracker>${jobTracker}</job-tracker>
        <name-node>${nameNode}</name-node>
        <prepare>
            <delete path="${nameNode}/user/${wf:user()}/${oozieRoot}/map-reduce/output-data"/>
        </prepare>
        <configuration>
            <property>
                <name>mapred.job.queue.name</name>
                <value>${queueName}</value>
            </property>
            <property>
                <name>mapred.mapper.class</name>
                <value>org.apache.oozie.example.SampleMapper</value>
            </property>
            <property>
                <name>mapred.reducer.class</name>
                <value>org.apache.oozie.example.SampleReducer</value>
            </property>
            <property>
                <name>mapred.map.tasks</name>
                <value>1</value>
            </property>
            <property>
                <name>mapred.input.dir</name>
                <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
            </property>
            <property>
                <name>mapred.output.dir</name>
                <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
            </property>
        </configuration>
    </map-reduce>
    <ok to="end"/>
    <error to="fail"/>
</action>
```

Note the use of a EL expression to get the user name which is in the HDFS directory path

WORKFLOWS

```
<configuration>
      <property>
          <name>mapred.job.queue.name</name>
          <value>${queueName}</value>
      </property>
</configuration>
```

This is the queue name where this MR will run

WORKFLOWS

```
<configuration>
        <property>
            <name>mapred.mapper.class</name>
            <value>org.apache.oozie.example.SampleMapper</value>
        </property>
        <property>
            <name>mapred.reducer.class</name>
            <value>org.apache.oozie.example.SampleReducer</value>
        </property>
</configuration>
```

The full path of the mapper and reducer class in the JAR we've added to the "lib/" folder

WORKFLOWS

```
<configuration>
    <property>
        <name>mapred.map.tasks</name>
        <value>1</value>
    </property>
</configuration>
```

This is how you'd specify the number of mapper processes

WORKFLOWS

And finally the input and output directory for the MR job

```
<configuration>
    <property>
        <name>mapred.input.dir</name>
        <value>/user/${wf:user()}/${oozieRoot}/map-reduce/input-data</value>
    </property>
    <property>
        <name>mapred.output.dir</name>
        <value>/user/${wf:user()}/${oozieRoot}/map-reduce/output-data</value>
    </property>
</configuration>
```

WORKFLOWS

Hadoop offers 2 APIs for MapReduce, the older **mapred** and the newer **mapreduce** API

Oozie supports the **mapred** API out of the box, the new API requires additional set up

WORKFLOWS

Other Oozie actions are Hive action, Pig action, Email action, DistCp action, File System action etc

Let's now see a **Shell** action which allows you execute terminal commands and scripts

WORKFLOWS

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.use.system.libpath=true

oozie.wf.application.path=${nameNode}/user/$
{user.name}/${oozieRoot}/simple-shell
```

WORKFLOWS

Only the application path is different, this is the same path where you need to copy over the example

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.use.system.libpath=true

oozie.wf.application.path=${nameNode}/user/${user.name}/${oozieRoot}/simple-shell
```

WORKFLOWS

In order to run the Workflow follow the same steps as you did for the MR workflow

```
nameNode=hdfs://localhost:9000
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.use.system.libpath=true

oozie.wf.application.path=${nameNode}/user/${user.name}/${oozieRoot}/simple-shell
```

WORKFLOWS

Just copy over the files to the oozie/simple-shell directory in HDFS

```
nameNode=hdfs://localhost:8020
jobTracker=localhost:8032
queueName=default
oozieRoot=oozie
oozie.use.system.libpath=true

oozie.wf.application.path=${nameNode}/user/${user.name}/${oozieRoot}/simple-shell
```

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="shell-wf">
    <start to="shell-node"/>
    <action name="shell-node">
        <shell xmlns="uri:oozie:shell-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <exec>echo</exec>
            <argument>my_output=Hello Oozie</argument>
            <capture-output/>
        </shell>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Shell action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="shell-wf">
    <start to="shell-node"/>
    <action name="shell-node">
        <shell xmlns="uri:oozie:shell-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <exec>echo</exec>
            <argument>my_output=Hello Oozie</argument>
            <capture-output/>
        </shell>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Shell action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

These parts are the same for any Workflow

WORKFLOWS

```
<action name="shell-node">
     <shell xmlns="uri:oozie:shell-action:0.2">
          <job-tracker>${jobTracker}</job-tracker>
          <name-node>${nameNode}</name-node>
          <configuration>
              <property>
                  <name>mapred.job.queue.name</name>
                  <value>${queueName}</value>
              </property>
          </configuration>
          <exec>echo</exec>
          <argument>my_output=Hello Oozie</argument>
          <capture-output/>
     </shell>
      <ok to="end"/>
      <error to="fail"/>
 </action>
```

The shell action indicates a shell script

WORKFLOWS

```
<action name="shell-node">
     <shell xmlns="uri:oozie:shell-action:0.2">
          <exec>echo</exec>
          <argument>my_output=Hello Oozie</argument>
          <capture-output/>
     </shell>
 </action>
```

The echo command is the script we want to execute specified in the "exec" element

WORKFLOWS

```
<action name="shell-node">
     <shell xmlns="uri:oozie:shell-action:0.2">
          <job-tracker>${jobTracker}</job-tracker>
          <name-node>${nameNode}</name-node>
          <configuration>
               <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
               </property>
          </configuration>
          <exec>echo</exec>
          <argument>my_output=Hello Oozie</argument>
          <capture-output/>
     </shell>
     <ok to="end"/>
     <error to="fail"/>
 </action>
```

It can take in command line arguments

WORKFLOWS

```
<action name="shell-node">
      <shell xmlns="uri:oozie:shell-action:0.2">
          <job-tracker>${jobTracker}</job-tracker>
          <name-node>${nameNode}</name-node>
          <configuration>
              <property>
                  <name>mapred.job.queue.name</name>
                  <value>${queueName}</value>
              </property>
          </configuration>
          <exec>echo</exec>
          <argument>my_output=Hello Oozie</argument>
          <capture-output/>
      </shell>
      <ok to="end"/>
      <error to="fail"/>
 </action>
```

This element causes the output of the shell command to be captured by Oozie

The output will be made available to the Workflow application using the `action:output()` EL function

WORKFLOWS

A **Shell** action can run Python scripts as well!

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="shell-wf">
    <start to="shell-node"/>
    <action name="shell-node">
        <shell xmlns="uri:oozie:shell-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <exec>/usr/bin/python</exec>
            <argument>test.py</argument>
            <argument>boo</argument>
            <file>test.py#test.py</file>
            <capture-output/>
        </shell>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Shell Python failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

WORKFLOWS

Execute Python from this path

```
<exec>/usr/bin/python</exec>
<argument>test.py</argument>
<argument>boo</argument>
<file>test.py#test.py</file>
```

WORKFLOWS

The Python script is the first argument and should be in the same directory as the workflow.xml file

```
<exec>/usr/bin/python</exec>
<argument>test.py</argument>
<argument>boo</argument>
<file>test.py#test.py</file>
```

WORKFLOWS

Specify an argument to the Python script as well

```
<exec>/usr/bin/python</exec>
<argument>test.py</argument>
<argument>boo</argument>
<file>test.py#test.py</file>
```

WORKFLOWS

This is an element to support the native Hadoop way of packaging files and JARs

```
<exec>/usr/bin/python</exec>
<argument>test.py</argument>
<argument>boo</argument>
<file>test.py#test.py</file>
```

WORKFLOWS

This is the symbolic link to the file path and indicates to the action where the file can be found

```
<exec>/usr/bin/python</exec>
<argument>test.py</argument>
<argument>boo</argument>
<file>test.py#test.py</file>
```

This is how Hadoop distributes files and archives using the distributed cache

WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="shell-wf">
    <start to="shell-node"/>
    <action name="shell-node">
        <shell xmlns="uri:oozie:shell-action:0.2">
            <job-tracker>${jobTracker}</job-tracker>
            <name-node>${nameNode}</name-node>
            <configuration>
                <property>
                    <name>mapred.job.queue.name</name>
                    <value>${queueName}</value>
                </property>
            </configuration>
            <exec>echo</exec>
            <argument>my_output=Hello Oozie</argument>
            <capture-output/>
        </shell>
        <ok to="end"/>
        <error to="fail"/>
    </action>
    <kill name="fail">
        <message>Shell action failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>
    </kill>
    <end name="end"/>
</workflow-app>
```

WORKFLOWS

Basic outline of a workflow

1. Control nodes
2. Action nodes
3. Global configuration

WORKFLOWS

Control nodes

These control the start, end and basic execution of the workflow

`<start>`

`<end>`

`<kill>`

`<ok>`

`<error>`

Let's see some more interesting ones which control execution flow

Fork and Join WORKFLOWS

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <fork name="forking">
        <path start="firstparalleljob"/>
        <path start="secondparalleljob"/>
    </fork>
    <action name="firstparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job1.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <action name="secondparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job2.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <join name="joining" to="nextaction"/>
    ...
</workflow-app>
```

Fork and Join WORKFLOWS

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <fork name="forking">
        <path start="firstparalleljob"/>
        <path start="secondparalleljob"/>
    </fork>
    <action name="firstparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job1.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <action name="secondparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job2.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <join name="joining" to="nextaction"/>
    ...
</workflow-app>
```

Here are two MR actions to run in this workflow

They can be run in **parallel**, they are not dependent on one another

Fork and Join WORKFLOWS

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    <fork name="forking">
        <path start="firstparalleljob"/>
        <path start="secondparalleljob"/>
    </fork>
    <action name="firstparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job1.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <action name="secondparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job2.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <join name="joining" to="nextaction"/>
    ...
</workflow-app>
```

The fork specifies the paths that can be run in parallel

Fork and Join WORKFLOWS

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <fork name="forking">
        <path start="firstparalleljob"/>
        <path start="secondparalleljob"/>
    </fork>
    <action name="firstparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job1.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <action name="secondparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job2.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <join name="joining" to="nextaction"/>
    ...
</workflow-app>
```

The workflow does not proceed beyond the join till all the parallel paths have been complete

Fork and Join WORKFLOWS

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <fork name="forking">
        <path start="firstparalleljob"/>
        <path start="secondparalleljob"/>
    </fork>
    <action name="firstparallejob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job1.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <action name="secondparalleljob">
        <map-reduce>
            <job-tracker>foo:9001</job-tracker>
            <name-node>bar:9000</name-node>
            <job-xml>job2.xml</job-xml>
        </map-reduce>
        <ok to="joining"/>
        <error to="kill"/>
    </action>
    <join name="joining" to="nextaction"/>
    ...
</workflow-app>
```

Decision WORKFLOWS

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
...
<decision name="decision">
    <switch>
        <case to="mapreduce">
            ${jobType eq "mapreduce"}
        </case>
        <case to="hive">
            ${jobType eq "hive"}
        </case>
        <case to="pig">
            ${jobType eq "pig"}
        </case>
        <default to="end"/>
    </switch>
</decision>
...
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>

</workflow-app>
```

Decision WORKFLOWS

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
...
<decision name="decision">
    <switch>
        <case to="mapreduce">
            ${jobType eq "mapreduce"}
        </case>
        <case to="hive">
            ${jobType eq "hive"}
        </case>
        <case to="pig">
            ${jobType eq "pig"}
        </case>
        <default to="end"/>
    </switch>
</decision>
...
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>

</workflow-app>
```

Here are 3 possible actions that can be executed by this workflow

Decision WORKFLOWS

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
...
<decision name="decision">
    <switch>
        <case to="mapreduce">
            ${jobType eq "mapreduce"}
        </case>
        <case to="hive">
            ${jobType eq "hive"}
        </case>
        <case to="pig">
            ${jobType eq "pig"}
        </case>
        <default to="end"/>
    </switch>
</decision>
...
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>

</workflow-app>
```

The decision node allows us to choose one of these actions based on a variable set

Decision WORKFLOWS

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
...
<decision name="decision">
    <switch>
        <case to="mapreduce">
            ${jobType eq "mapreduce"}
        </case>
        <case to="hive">
            ${jobType eq "hive"}
        </case>
        <case to="pig">
            ${jobType eq "pig"}
        </case>
        <default to="end"/>
    </switch>
</decision>
...
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>

</workflow-app>
```

This is an EL condition to evaluate whether the variable is equal to mapreduce

Decision WORKFLOWS

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
...
<decision name="decision">
      <switch>
            <case to="mapreduce">
              ${jobType eq "mapreduce"}
            </case>
        <case to="hive">
            ${jobType eq "hive"}
        </case>
        <case to="pig">
            ${jobType eq "pig"}
        </case>
        <default to="end"/>
    </switch>
</decision>
...
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>

</workflow-app>
```

The switch node allows us to specify multiple conditions

If any condition in a case element matches the control moves to the specified node

Decision WORKFLOWS

```
<workflow-app name="foo-wf" xmlns="uri:oozie:workflow:0.1">
...
<decision name="decision">
    <switch>
        <case to="mapreduce">
            ${jobType eq "mapreduce"}
        </case>
        <case to="hive">
            ${jobType eq "hive"}
        </case>
        <case to="pig">
            ${jobType eq "pig"}
        </case>
        <default to="end"/>
    </switch>
</decision>
...
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>

</workflow-app>
```

If there is no match the control moves to the node specified by the default element

WORKFLOWS

Basic outline of a workflow

1. Control nodes
2. Action nodes
3. Global configuration

Action nodes WORKFLOWS

Action nodes are the ones which specifies the unit of execution

<map-reduce>

<pig>

<shell>

<fs>

<email>

<hive>

Action nodes

<fs>

<map-reduce>

<shell>

<email> <pig>

<hive>

All actions are executed asynchronously by Oozie other than the File System action which is synchronous

Action nodes

<fs>

<map-reduce>

<shell>

<email> <pig>

<hive>

Actions have two transitions to ok and to error

WORKFLOWS

Basic outline of a workflow

1. Control nodes
2. Action nodes
3. Global configuration

Global configuration WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="wf-name">
<global>
    <job-tracker>${job-tracker}</job-tracker>
    <name-node>${name-node}</name-node>
    <job-xml>job1.xml</job-xml>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
    </configuration>
</global>
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>
...
</workflow-app>
```

Global configuration WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="wf-name">
<global>
    <job-tracker>${job-tracker}</job-tracker>
    <name-node>${name-node}</name-node>
    <job-xml>job1.xml</job-xml>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
    </configuration>
</global>
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>
...
</workflow-app>
```

Here are 3 actions which are part of this workflow

Global configuration WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="wf-name">
<global>
    <job-tracker>${job-tracker}</job-tracker>
    <name-node>${name-node}</name-node>
    <job-xml>job1.xml</job-xml>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
    </configuration>
</global>
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>
...
</workflow-app>
```

They have many configuration properties in common

Global configuration WORKFLOWS

```
<workflow-app xmlns="uri:oozie:workflow:0.4" name="wf-name">
<global>
    <job-tracker>${job-tracker}</job-tracker>
    <name-node>${name-node}</name-node>
    <job-xml>job1.xml</job-xml>
    <configuration>
        <property>
            <name>mapred.job.queue.name</name>
            <value>${queueName}</value>
        </property>
    </configuration>
</global>
<action name="mapreduce">
    ...
</action>
<action name="hive">
    ...
</action>
<action name="pig">
    ...
</action>
...
</workflow-app>
```

These can be defined in the global parameters rather than once for each action

WORKFLOWS

Basic outline of a workflow

1. Control nodes
2. Action nodes
3. Global configuration