ASSIGNMENT 10.1

Apache Oozie is an open source project based on Java™ technology that simplifies the process of creating workflows and managing coordination among jobs. In principle, Oozie offers the ability to combine multiple jobs sequentially into one logical unit of work. One advantage of the Oozie framework is that it is fully integrated with the Apache Hadoop stack and supports Hadoop jobs for Apache MapReduce, Pig, Hive, and Sqoop. In addition, it can be used to schedule jobs specific to a system, such as Java programs. Therefore, using Oozie, Hadoop administrators are able to build complex data transformations that can combine the processing of different individual tasks and even sub-workflows. This ability allows for greater control over complex jobs and makes it easier to repeat those jobs at predetermined periods.

In practice, there are different types of Oozie jobs:

Oozie Workflow jobs — Represented as directed acyclical graphs to specify a sequence of actions to be executed.

Oozie Coordinator jobs — Represent Oozie workflow jobs triggered by time and data availability.

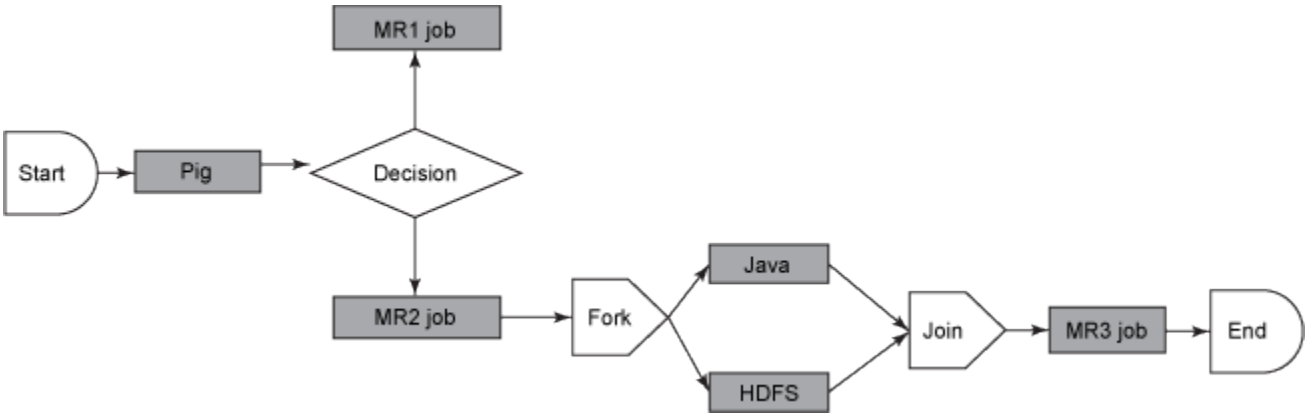
Oozie Bundle— Facilitates packaging multiple coordinator and workflow jobs, and makes it easier to manage the life cycle of those jobs.

OOZIE WORKFLOW:-

An Oozie workflow is a collection of actions arranged in a directed acyclic graph (DAG). This graph can contain two types of nodes: control nodes and action nodes.*Control nodes*, which are used to define job chronology, provide the rules for beginning and ending a workflow and control the workflow execution path with possible decision points known as fork and join nodes.*Action nodes*are used to trigger the execution of tasks. In particular, an action node can be a MapReduce job, a Pig application, a file system task, or a Java application. (The shell and ssh actions have been deprecated).

Oozie is a native Hadoop stack integration that supports all types of Hadoop jobs and is integrated with the Hadoop stack. In particular, Oozie is responsible for triggering the workflow actions, while the actual execution of the tasks is done using Hadoop MapReduce. Therefore, Oozie becomes able to leverage existing Hadoop machinery for load balancing, fail-over, etc. Oozie detects completion of tasks through callback and polling. When Oozie starts a task, it provides a unique callback HTTP URL to the task, and notifies that URL when it is complete. If the task fails to invoke the callback URL, Oozie can poll the task for completion. Figure 1 illustrates a sample Oozie workflow that combines six action nodes (Pig scrip, MapReduce jobs, Java code, and HDFS task) and five control nodes (Start, Decision control, Fork, Join, and End). Oozie workflows can be also parameterized. When submitting a workflow job, values for the parameters must be provided. If the appropriate parameters are used, several identical workflow jobs can occur concurrently.

Figure 1. Sample Oozie workflow



In practice, it is sometimes necessary to run Oozie workflows on regular time intervals, but in coordination with other conditions, such as the availability of specific data or the completion of any other events or tasks. In these situations, Oozie Coordinator jobs allow the user to model workflow execution triggers in the form of the data, time, or event predicates where the workflow job is started after those predicates get satisfied. The Oozie Coordinator can also manage multiple workflows that are dependent on the outcome of subsequent workflows. The outputs of subsequent workflows become the input to the next workflow. This chain is called a *data application pipeline*.

Oozie workflow definition language is XML-based and it is called the *Hadoop Process Definition Language.*Oozie comes with a command-line program for submitting jobs. This command-line program interacts with the Oozie server using REST. To submit or run a job using the Oozie client, give Oozie the full path to your workflow.xml file in HDFS as a parameter to the client. Oozie does not have a notion of global properties. All properties, including the *jobtracker* and the *namenode*, must be submitted as part of every job run. Oozie uses an RDBMS for storing state.

Oozie in action

Use an Oozie workflow to run a recurring job. Oozie workflows are written as an XML file representing a directed acyclic graph. Let's look at the following simple workflow example that chains two MapReduce jobs. The first job performs an initial ingestion of the data and the second job merges data of a given type.

To run oozieworkflows, two files are needed.

1. workflow.xml (stored in HDFS) \It contains the structure of workflow.

2.job.properties(stored in local)\It contains the configuration properties.

Benefits of oozie workflow :-

Oozie supports automated starting of oozie workflow process.

Oozie allow to run a series of map-reduce, pig, java & scripts actions a single workflow job.

Oozie supports: mapreduce (java, streaming, pipes), pig, java, filesystem, ssh, sub-workflow.

Oozie supports decision nodes allowing the workflow to make decisions.

Oozie interval job scheduling is time & input-data-dependent based.

Working with oozie job example

hadoop fs -mkdir -p /user/oozie/workflows/

hadoop fs -put workflow.xml /user/oozie/workflows/

hadoop fs -put create_table.hql /user/oozie/workflows/

hadoop fs -put /var/lib/ambari-server/resources/stacks/HDP/2.1/services/HIVE/configuration/hive-site.xml /user/oozie/workflows/hive-site.xml

sudo -u oozie oozie job -oozie http://127.0.0.1:11000/oozie -config job.properties -submit

0000000-160723204217175-oozie-oozi-W

sudo -u oozie oozie job -oozie http://127.0.0.1:11000/oozie -start 0000000-160723204217175-oozie-oozi-W

Equivalently,

sudo -u oozie oozie job -oozie http://127.0.0.1:11000/oozie -config job.properties -run

sudo -u oozie hive -e "drop table company_demo"

Oozie Console:

http://127.0.0.1:11000/oozie/

Working oozie as root user

Go to Ambari Server

http://localhost:8080

HDFS --> Configs --> Custom core-site.xml

hadoop.proxyuser.oozie.groups is set to users

usermod -a -G <groupname> username

usermod -a -G users root

---

hadoop fs -mkdir -p /user/root/workflows

hadoop fs -put workflow.xml /user/root/workflows/

hadoop fs -put create_table.hql /user/root/workflows/

hadoop fs -put /var/lib/ambari-server/resources/stacks/HDP/2.1/services/HIVE/configuration/hive-site.xml /user/root/workflows/hive-site.xml

oozie job -oozie http://127.0.0.1:11000/oozie -config job.properties -run

oozie job -oozie http://127.0.0.1:11000/oozie -info 0000005-160904225720176-oozie-oozi-W

Sqoop Workflow

The sqoop action runs a Sqoop job.

The workflow job will wait until the Sqoop job completes before continuing to the next action.

To run the Sqoop job, you have to configure the sqoop action with the =job-tracker=name-node and Sqoop command or arg elements as well as configuration.

A sqoop action can be configured to create or delete HDFS directories before starting the Sqoop job.

Sqoop configuration can be specified with a file, using the job-xml element, and inline, using the configuration elements.

Oozie EL expressions can be used in the inline configuration. Property values specified in the configuration element override values specified in the job-xml file.

Note that Hadoop mapred.job.tracker and fs.default.name properties must not be present in the inline configuration.

As with Hadoop map-reduce jobs, it is possible to add files and archives in order to make them available to the Sqoop job. Refer to the [WorkflowFunctionalSpec#FilesAchives][Adding Files and Archives for the Job] section for more information about this feature.

Syntax:

```
<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.1">
    ...
    <action name="[NODE-NAME]">
        <sqoop xmlns="uri:oozie:sqoop-action:0.2">
            <job-tracker>[JOB-TRACKER]</job-tracker>
            <name-node>[NAME-NODE]</name-node>
            <prepare>
```

```
            <delete path="[PATH]"/>
            ...
            <mkdir path="[PATH]"/>
            ...
        </prepare>
        <configuration>
            <property>
                <name>[PROPERTY-NAME]</name>
                <value>[PROPERTY-VALUE]</value>
            </property>
            ...
        </configuration>
        <command>[SQOOP-COMMAND]</command>
        <arg>[SQOOP-ARGUMENT]</arg>
        ...
        <file>[FILE-PATH]</file>
        ...
        <archive>[FILE-PATH]</archive>
        ...
    </sqoop>
    <ok to="[NODE-NAME]"/>
    <error to="[NODE-NAME]"/>
</action>
...
</workflow-app>
```

The prepare element, if present, indicates a list of paths to delete or create before starting the job. Specified paths must start with hdfs://HOST:PORT .

The job-xml element, if present, specifies a file containing configuration for the Sqoop job. As of schema 0.3, multiple job-xml elements are allowed in order to specify multiple job.xmlfiles.

The configuration element, if present, contains configuration properties that are passed to the Sqoop job.

Sqoop command

The Sqoop command can be specified either using the command element or multiple arg elements.

When using the command element, Oozie will split the command on every space into multiple arguments.

When using the arg elements, Oozie will pass each argument value as an argument to Sqoop.
The arg variant should be used when there are spaces within a single argument.
Consult the Sqoop documentation for a complete list of valid Sqoop commands.
All the above elements can be parameterized (templatized) using EL expressions.
Examples:
Using the command element:

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <action name="myfirsthivejob">
        <sqoop xmlns="uri:oozie:sqoop-action:0.2">
            <job-traker>foo:8021</job-tracker>
            <name-node>bar:8020</name-node>
            <prepare>
                <delete path="${jobOutput}"/>
            </prepare>
            <configuration>
                <property>
                    <name>mapred.compress.map.output</name>
                    <value>true</value>
                </property>
            </configuration>
            <command>import  --connect jdbc:hsqldb:file:db.hsqldb --table TT --target-dir
hdfs://localhost:8020/user/tucu/foo -m 1</command>
        </sqoop>
        <ok to="myotherjob"/>
        <error to="errorcleanup"/>
    </action>
    ...
</workflow-app>
```

The same Sqoop action using arg elements:

```
<workflow-app name="sample-wf" xmlns="uri:oozie:workflow:0.1">
    ...
    <action name="myfirsthivejob">
        <sqoop xmlns="uri:oozie:sqoop-action:0.2">
            <job-traker>foo:8021</job-tracker>
```

```
            <name-node>bar:8020</name-node>
            <prepare>
               <delete path="${jobOutput}"/>
            </prepare>
            <configuration>
               <property>
                  <name>mapred.compress.map.output</name>
                  <value>true</value>
               </property>
            </configuration>
            <arg>import</arg>
            <arg>--connect</arg>
            <arg>jdbc:hsqldb:file:db.hsqldb</arg>
            <arg>--table</arg>
            <arg>TT</arg>
            <arg>--target-dir</arg>
            <arg>hdfs://localhost:8020/user/tucu/foo</arg>
            <arg>-m</arg>
            <arg>1</arg>
        </sqoop>
        <ok to="myotherjob"/>
        <error to="errorcleanup"/>
    </action>
    ...
</workflow-app>
```

NOTE: The arg elements syntax, while more verbose, allows to have spaces in a single argument, something useful when using free from queries.

Sqoop Action Counters

The counters of the map-reduce job run by the Sqoop action are available to be used in the workflow via the[hadoop:counters() EL function](#).

If the Sqoop action run an import all command, the hadoop:counters() EL will return the aggregated counters of all map-reduce jobs run by the Sqoop import all command.

Sqoop Action Logging

Sqoop action logs are redirected to the Oozie Launcher map-reduce job task STDOUT/STDERR that runs Sqoop.

From Oozie web-console, from the Sqoop action pop up using the 'Console URL' link, it is possible to navigate to the Oozie Launcher map-reduce job task logs via the Hadoop job-tracker web-console.

The logging level of the Sqoop action can set in the Sqoop action configuration using the property oozie.sqoop.log.level

The default value is INFO .