**Workflow of Oozie and its Benefits**

Oozie is a workflow/coordination system that can be used to manage Apache Hadoop jobs.

Oozie workflow provides support for defining and executing a controlled sequence of MapReduce, Hive and Pig jobs.

Workflow in Oozie is a sequence of actions arranged in a control dependency DAG (Directed Acyclic Graph). The actions are in controlled dependency as the next action can only run as per the output of current action. Subsequent actions are dependent on its previous action. A workflow action can be a Hive action, Pig action, Java action, Shell action, etc. There can be decision trees to decide how and on which condition a job should run.

A fork is used to run multiple jobs in parallel. Oozie workflows can be parameterized (variables like **${nameNode}** can be passed within the workflow definition). These parameters come from a configuration file called as property file.

**WORKFLOW EXAMPLE**

Consider we want to load a data from external hive table to an ORC Hive table.

Step 1: Creating an external table

DDL for Hive external table (say **external.hql**)

Create external table external\_person(

name string,

yearofbirth int,

age int,

address string,

zip int

)

row format delimited

fields terminated by ','

stored as textfile

location '/hive\_data';

**Step 2: Creating an ORC table**

DDL for Hive ORC table (say orc.hql)

Create Table orc\_table

(

name string, -- Concatenate value of first name and last name with space as seperator

yearofbirth int,

age int, -- Current year minus year of birth

address string,

zip int

)

STORED AS ORC;

Step 3: Hive Script to insert data from external table to ORC table (say copydata.hql)

use ${database\_name}; -- input from Oozie

insert into table orc\_table

select

concat(first\_name,' ',last\_name) as name,

yearofbirth,

year(from\_unixtime) --yearofbirth as age,

address,

zip

from external\_person;

**Step 4** − Create a workflow to execute all the above three steps. (let’s call it workflow.xml)

<!-- This is a comment -->

<workflow-app xmlns = "uri:oozie:workflow:0.4" name = "simple-Workflow">

<start to = "Create\_External\_Table" />

<!—Step 1 -->

<action name = "Create\_External\_Table">

<hive xmlns = "uri:oozie:hive-action:0.4">

<job-tracker>xyz.com:8088</job-tracker>

<name-node>hdfs://rootname</name-node>

<script>hdfs\_path\_of\_script/external.hql</script>

</hive>

<ok to = "Create\_orc\_Table" />

<error to = "kill\_job" />

</action>

<!—Step 2 -->

<action name = "Create\_orc\_Table">

<hive xmlns = "uri:oozie:hive-action:0.4">

<job-tracker>xyz.com:8088</job-tracker>

<name-node>hdfs://rootname</name-node>

<script>hdfs\_path\_of\_script/orc.hql</script>

</hive>

<ok to = "Insert\_into\_Table" />

<error to = "kill\_job" />

</action>

<!—Step 3 -->

<action name = "Insert\_into\_Table">

<hive xmlns = "uri:oozie:hive-action:0.4">

<job-tracker>xyz.com:8088</job-tracker>

<name-node>hdfs://rootname</name-node>

<script>hdfs\_path\_of\_script/copydata.hive</script>

<param>database\_name</param>

</hive>

<ok to = "end" />

<error to = "kill\_job" />

</action>

<kill name = "kill\_job">

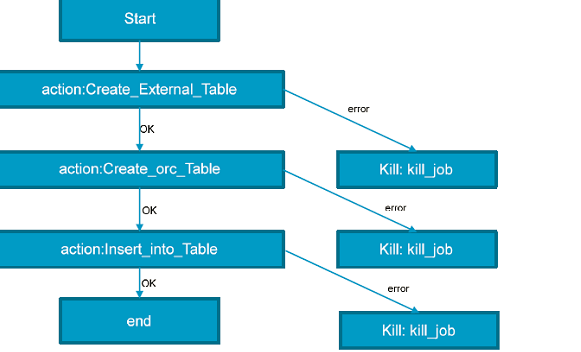
<message>Job failed</message>

</kill>

<end name = "end" />

</workflow-app>

**The above workflow will translate into following DAG:**



**Running the Workflow:**

oozie job --oozie http://host\_name:8080/oozie -D

oozie.wf.application.path=hdfs://namenodepath/pathof\_workflow\_xml/ workflow.xml -run

**Note** − *The workflow and hive scripts should be placed in HDFS path before running the workflow.*

To check the status of job you can go to Oozie web console -- [**http://host\_name:8080/**](http://host_name:8080/)

The possible states for workflow jobs are: **PREP, RUNNING, SUSPENDED, SUCCEEDED, KILLED and FAILED**.

**Benefits of Oozie Workflow:**

1. It provides support for defining and executing a controlled sequence of MapReduce, Hive and Pig jobs.
2. Automating the steps of hadoop jobs.
3. Workflow gets translated into DAG so it’s easy to understand the step by step sequence of jobs.

**Workflow of Sqoop and its Benefits**

We can configure Oozie workflow for Sqoop commands. Below is an example to run sample Oozie Sqoop action to get data from MySQL table on HDFS:

1. Configure job.properties

**nameNode=hdfs://<namenode-host>:8020**

**jobTracker=<rm-host>:8050**

**queueName=default**

**examplesRoot=examples**

**oozie.use.system.libpath=true**

**oozie.wf.application.path=${nameNode}/user/${user.name}**

**oozie.libpat=/user/root**

1. Configure Workflow.xml

**<?xml version="1.0" encoding="UTF-8"?>**

**<workflow-app xmlns="uri:oozie:workflow:0.2" name="sqoop-wf">**

**<start to="sqoop-node"/>**

**<action name="sqoop-node">**

**<sqoop xmlns="uri:oozie:sqoop-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>**

**<command>import --connect jdbc:mysql://<mysql-server-hostname>:3306/<database-name> --username <mysql-database-username> --table <table-name> --driver com.mysql.jdbc.Driver --m 1</command>**

**</sqoop>**

**<ok to="end"/>**

**<error to="fail"/>**

**</action>**

**<kill name="fail">**

**<message>Sqoop failed, error message[${wf:errorMessage(wf:lastErrorNode())}]</message>**

**</kill>**

**<end name="end"/>**

**</workflow-app>**

#### **Upload workflow.xml and shell script to "**oozie.wf.application.path" **defined in job.properties.**

#### **Follow below command to run Oozie workflow**

**oozie job -oozie http://<oozie-server-hostname>:11000/oozie -config /$PATH/job.properties -run**