The workflow of Oozie and its Benefits

Apache Oozie is a scheduler system to run and **manage Hadoop jobs** in a distributed environment. It allows to combine multiple complex jobs to be run in a sequential order to achieve a bigger task. Within a sequence of task, two or more jobs can also be programmed to run parallel to each other.

One of the main advantages of Oozie is that it is tightly integrated with Hadoop stack supporting various Hadoop jobs like **Hive, Pig, Sqoop** as well as system-specific jobs like **Java and Shell**.

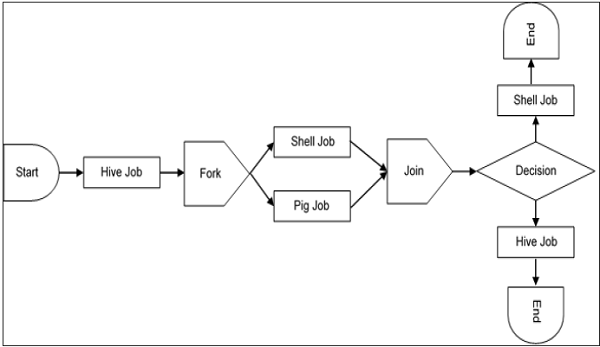
Oozie is an **Open Source Java Web-Application** available under Apache license 2.0. It is responsible for triggering the workflow actions, which in turn uses the Hadoop execution engine to actually execute the task. Hence, Oozie is able to leverage the existing Hadoop machinery for load balancing, fail-over, 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.

Apache Oozie is used by Hadoop system administrators to run complex log analysis on **HDFS**. Hadoop Developers use Oozie for performing ETL operations on data in a sequential order and saving the output in a specified format (Avro, ORC, etc.) in HDFS.

Following three types of jobs are common in Oozie −

* **Oozie Workflow Jobs** − These are represented as Directed Acyclic Graphs (DAGs) to specify a sequence of actions to be executed.
* **Oozie Coordinator Jobs** − These consist of workflow jobs triggered by time and data availability.
* **Oozie Bundle** − These can be referred to as a package of multiple coordinator and workflow jobs.



Oozie is scalable and can manage timely execution of thousands of workflows (each consisting of dozens of jobs) in a Hadoop cluster.

Oozie is very much flexible, as well. One can easily start, stop, suspend and rerun jobs. Oozie makes it very easy to rerun failed workflows.

Workflow in Oozie is a sequence of actions arranged in a control dependency DAG (Direct 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.

* **WorkFlow**

Sample nodes of workflow.xml file are as follows.

1. Start node:

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

1. Action node:

<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.hive</script>

</hive>

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

<error to = "kill\_job" />

</action>

1. Fork:

<fork name = "fork\_node">

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

<path start = "Create\_orc\_Table"/>

</fork>

1. Join:

<join name = "join\_node" to = "Insert\_into\_Table"/>

1. Decision:

<decision name = "external\_table\_exists">

<switch>

<case to = "Create\_External\_Table">${fs:exists('/test/abc') eq 'false'}

</case>

<default to = "orc\_table\_exists" />

</switch>

</decision>

1. Kill job:

<kill name = "kill\_job">

<message>Job failed</message>

</kill>

1. End node:

<end name = "end" />

* **Property file:**

Oozie workflows can be parameterized. The parameters come from a configuration file called as property file. We can run multiple jobs using same workflow by using multiple .property files

Sample property file

nameNode = hdfs://rootname

jobTracker = xyz.com:8088

script\_name\_external = hdfs\_path\_of\_script/external.hive

script\_name\_orc=hdfs\_path\_of\_script/orc.hive

script\_name\_copy=hdfs\_path\_of\_script/Copydata.hive

database = database\_name

Using above parameters in workflow.xml

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

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

<job-tracker>${jobTracker}</job-tracker>

<name-node>${nameNode}</name-node>

<script>${script\_name\_copy}</script>

<param>${database}</param>

</hive>

<ok to = "end" />

<error to = "kill\_job" />

</action>

Use property file while running the job:

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

--config edgenode\_path/job1.properties -D oozie.wf.application.path

hdfs://Namenodepath/pathof\_workflow\_xml/workflow.xml –run

At run time, all the parameters in ${} will be replaced by its corresponding value in the .properties file.

* **Coordinator:**

Example:

<coordinator-app xmlns = "uri:oozie:coordinator:0.2" name =

"coord\_copydata\_from\_external\_orc" frequency = "5 \* \* \* \*" start =

"2016-00-18T01:00Z" end = "2025-12-31T00:00Z"" timezone = "America/Los\_Angeles">

<controls>

<timeout>1</timeout>

<concurrency>1</concurrency>

<execution>FIFO</execution>

<throttle>1</throttle>

</controls>

<action>

<workflow>

<app-path>pathof\_workflow\_xml/workflow.xml</app-path>

</workflow>

</action>

</coordinator-app>

* **start** − It means the start datetime for the job. Starting at this time the actions will be materialized.
* **end** − The end datetime for the job. When actions will stop being materialized.
* **timezone** − The timezone of the coordinator application.
* **frequency** − The frequency, in minutes, to materialize actions.