



ACADGILD

Session 10: Oozie and Sqoop

Assignment 1 Question -

Problem Statement

Note:-Assignment problem has been changed by Prateek

Explain in brief:

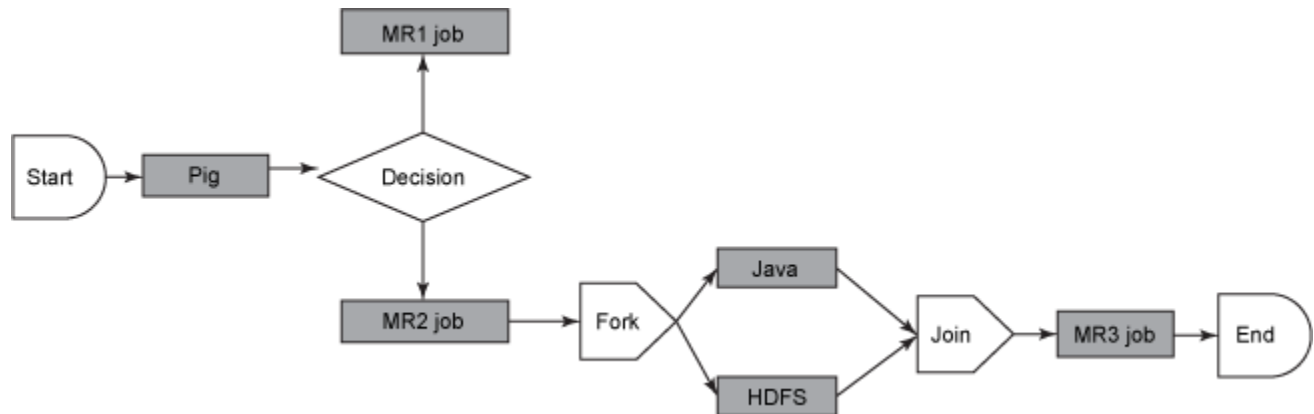
- *The complete structure and the working of “Oozie Workflow scheduler”*

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.

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 script, 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 Action and Decision Nodes*

Solution:

Action Nodes:

- All computation/processing tasks triggered by an action node are executed asynchronously by Oozie. For most types of computation/processing tasks triggered by workflow action, the workflow job has to wait until the computation/processing task completes before transitioning to the following node in the workflow.
- The exception is the fs action that is handled as a synchronous action.
- Oozie can detect completion of computation/processing tasks by two different means, callbacks and polling.
- When a computation/processing task is started by Oozie, Oozie provides a unique callback URL to the task, the task should invoke the given URL to notify its completion.
- For cases that the task failed to invoke the callback URL for any reason (i.e. a transient network failure) or when the type of task cannot invoke the callback URL upon completion, Oozie has a mechanism to poll computation/processing tasks for completion.
- If a computation/processing task -triggered by a workflow- completes successfully, it transitions to ok.
- If a computation/processing task -triggered by a workflow- fails to complete successfully, it transitions to error.
- If a computation/processing task exits in error, there computation/processing task must provide error-code and error-message information to Oozie. This information can be used from decision nodes to implement a fine grain error handling at workflow application level.

- Each action type must clearly define all the error codes it can produce
- Oozie provides recovery capabilities when starting or ending actions.

Decision Nodes:

- A decision 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.
- A decision node consists of a list of predicates-transition pairs plus a default transition.
- Predicates are evaluated in order of 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.
- Each case element contains a predicate and a transition name.
- The default element indicates the transition to take if none of the predicates evaluates to true.
- All decision nodes must have a default element to avoid bringing the workflow into an error state if none of the predicates evaluates to true.

● *Oozie Workflow Nodes*

Solution:

Control Flow nodes:

Control flow nodes define the beginning and the end of a workflow (the start, end and kill nodes) and provide a mechanism to control the workflow execution path (the decision, fork and join nodes).

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.

The end node is the end for a workflow job, it indicates that the workflow job has completed successfully.

When a workflow job reaches the end it finishes successfully (SUCCEEDED).

If one or more actions started by the workflow job are executing when the end node is reached, the actions will be killed. In this scenario the workflow job is still considered as successfully run.

A workflow definition must have one end node.

The kill node allows a workflow job to kill itself.

When a workflow job reaches the kill, it finishes in error (KILLED).

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.

Action Nodes:

Action nodes are the mechanism by which a workflow triggers the execution of a computation/processing task.

All computation/processing tasks triggered by an action node are executed asynchronously by Oozie.

For most types of computation/processing tasks triggered by workflow action, the workflow job has to wait until the computation/processing task completes before transitioning to the following node in the workflow.

- Fork and Join

Solution:

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.

The join node assumes concurrent execution paths are children of the same fork node.

- Oozie Web Console

Solution:

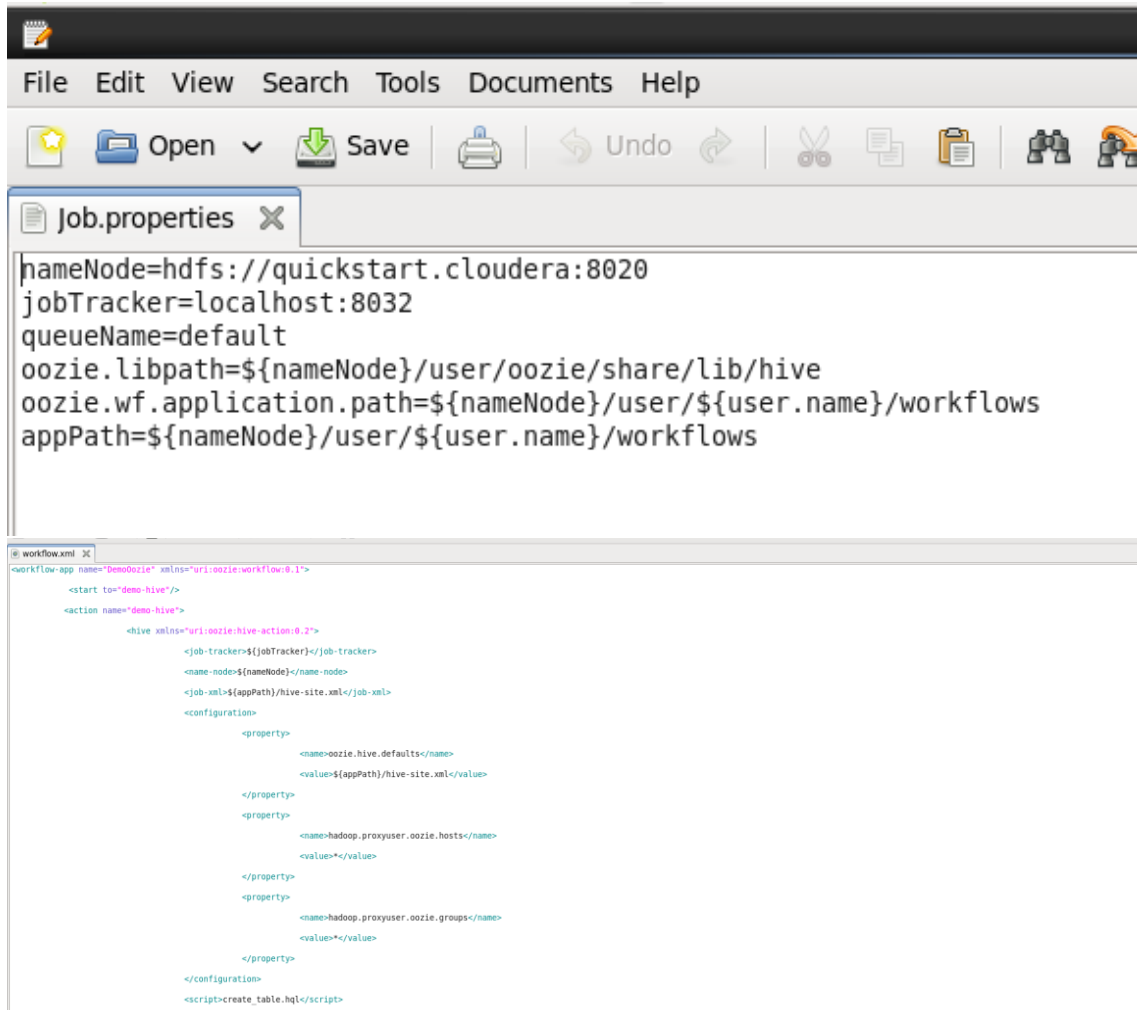
Oozie web console is one which is used for monitoring the Status of the job, workflow applications status and workflow jobs status.

It is read only interface and we cannot create any workflow.

We can only read only.

Following Procedure applied for oozie scheduling :-

Created three files create_table.hql, workflow.xml and job properties



```
        </configuration>

        <script>create_table.hql</script>

    </hive>

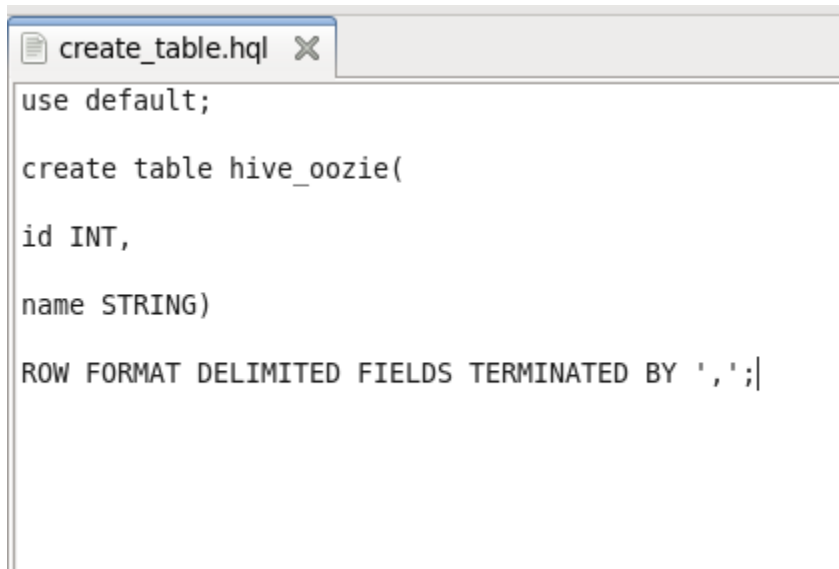
    <ok to="end"/>

    <error to="end"/>

</action>

<end name="end"/>

</workflow-app>
```



```
create_table.hql X
use default;

create table hive_oozie(
id INT,
name STRING)
ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
```

Command to run oozie job: -

```
[cloudera@quickstart oozie]$ oozie job -oozie
https://quickstart.cloudera://11000/oozie -config job.properties -run
```