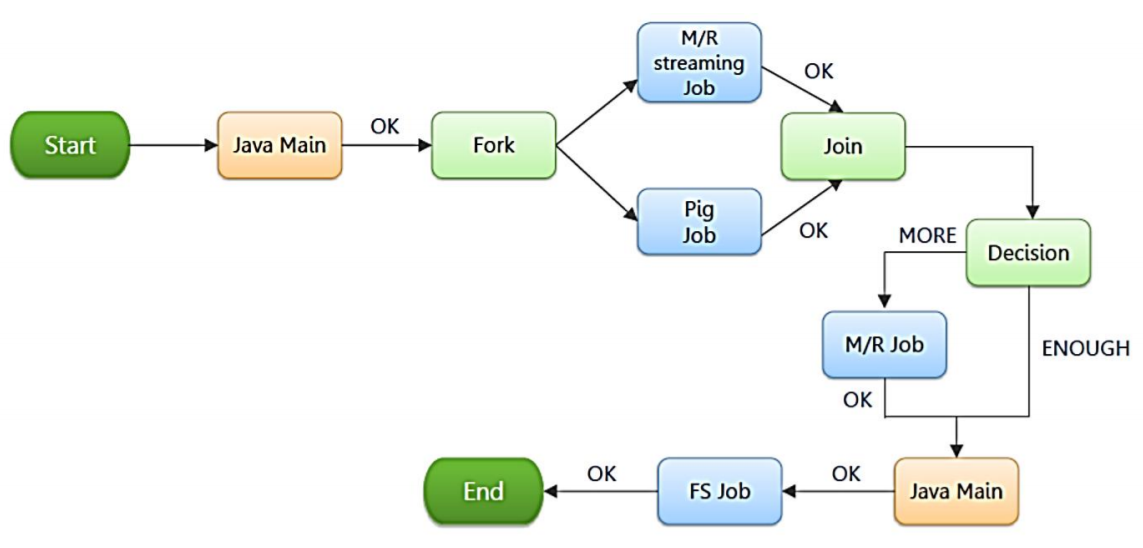
**Session 10 Assignment 1**

* **The workflow of Oozie and its Benefits**
* Oozie is a server based Workflow Engine specialized in running workflow jobs with actions that run Hadoop Map/Reduce and Pig jobs and is a Java Web-Application that runs in a Java servlet-container. For The purposes of Oozie, a workflow is a collection of actions (i.e. Hadoop Map/Reduce jobs, Pig jobs) arranged in a control dependency DAG (Direct Acyclic Graph). "control dependency" from one action to another means that the second action can't run until the first action has completed. It’s workflow definitions are written in hPDL (a XML Process Definition Language similar to JBOSS JBPM jPDL) and contain control flow nodes and action nodes.

Oozie workflow actions start jobs in remote systems (i.e. Hadoop, Pig). Upon action completion, the remote systems callback Oozie to notify the action completion, at this point Oozie proceeds to the next action in the workflow.

Control flow nodes define the beginning and the end of a workflow (start, end and fail nodes) and provide a mechanism to control the workflow execution path (decision, fork and join nodes). Action nodes are the mechanism by which a workflow triggers the execution of a computation/processing task. Oozie provides support for different types of actions: Hadoop map-reduce, Hadoop file system, Pig, SSH, HTTP, eMail and Oozie sub-workflow. Oozie can be extended to support additional type of actions. Oozie workflows can be parameterized (using variables like ${inputDir} within the workflow definition). When submitting a workflow job values for the parameters must be provided. If properly parameterized (using different output directories) several identical workflow jobs can concurrently. Below is the Workflow image,



Sample Workflow ->

<workflow-app name="Name of Workflow" xmlns="uri:oozie:workflow:0.1">

<start to="Start Node"/>

<action name="Name of Node">

<hive xmlns="uri:oozie:hive-action:0.2">

<job-tracker>Job Tracker Address</job-tracker>

<name-node>Name Node Address</name-node>

<job-xml>Path of hive-site.xml in HDFS</job-xml>

<configuration>

<property>

<name>Configuration Name</name>

<value>Configuration Value</value>

</property>

</configuration>

<script>Hive Query File</script>

</hive>

<ok to="end"/>

<error to="end"/>

</action>

• To run oozie workflows, two files are needed.

1. workflow.xml (stored in HDFS)

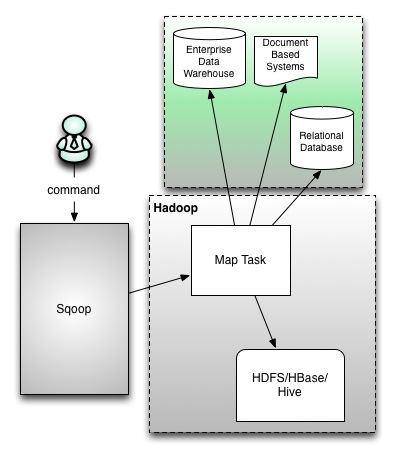
• It contains the structure of workflow.

2. job.properties (stored in local)

• It contains the configuration properties.

some benefits that are worth considering:

* + Oozie is designed to scale in a Hadoop cluster. Each job will be launched from a different data node. This means that the workflow load will be balanced and no single machine will become overburdened by launching workflows. This also means that the capacity to launch workflows will grow as the cluster grows.
  + Oozie is well integrated with Hadoop security. This is especially important in a kerberized cluster. Oozie knows which user submitted the job and will launch all actions as that user, with the proper privileges. It will handle all the authentication details for the user as well.
  + Oozie is the only workflow manager with built-in Hadoop actions, making workflow development, maintenance and troubleshooting easier.
  + Oozie UI makes it easier to drill down to specific errors in the data nodes. Other systems would require significantly more work to correlate job tracker jobs with the workflow actions.
  + Oozie is proven to scale in some of the world’s largest clusters. The white paper discusses a deployment at Yahoo! that can handle 1250 job submissions a minute.
  + Oozie gets call-backs from MapReduce jobs so it knows when they finish and whether they hang without expensive polling. No other workflow manager can do this.
  + Oozie Coordinator allows triggering actions when files arrive at HDFS. This will be challenging to implement anywhere else.
* **The workflow of Sqoop and its Benefits**
* Sqoop is a bulk data transfer tool that allows easy import/export of data from structured datastores such as relational databases, enterprise data warehouses, and NoSQL systems. Using Sqoop, you can provision the data from an external system into HDFS, as well as populate tables in Hive and HBase. Similarly, Sqoop integrates with the workflow coordinator Apache Oozie (incubating), allowing you to schedule and automate import/export tasks. Sqoop uses a connector-based architecture which supports plugins that provide connectivity to additional external systems. Below is the 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.

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

Example:

<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>

Some of the benefits are ->

* + Allows the transfer of data with a variety of structured data stores like Postgres, Oracle, Teradata, and so on.
  + Since the data is transferred and stored in Hadoop, Sqoop allows us to offload certain processing done in the ETL (Extract, Load and Transform) process into low-cost, fast, and effective Hadoop processes.
  + Sqoop can execute the data transfer in parallel, so execution can be quick and more cost effective.
  + Helps to integrate with sequential data from the mainframe. This helps not only to limit the usage of the mainframe, but also reduces the high cost in executing certain jobs using mainframe hardware.