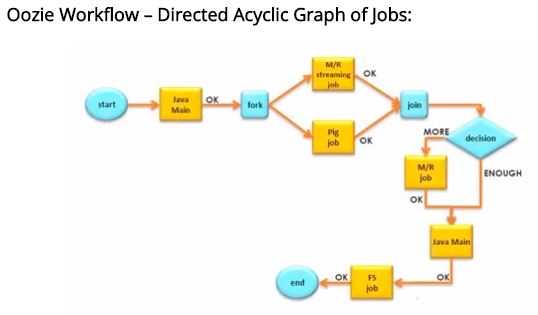
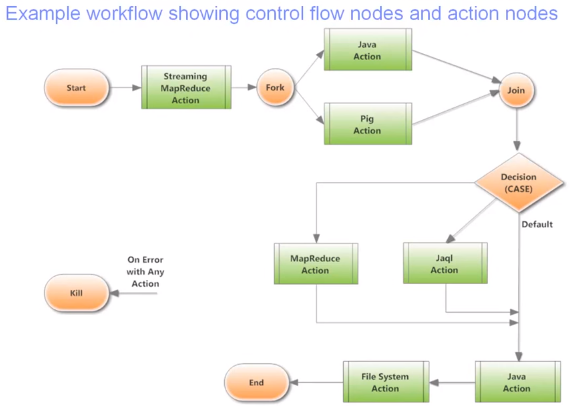
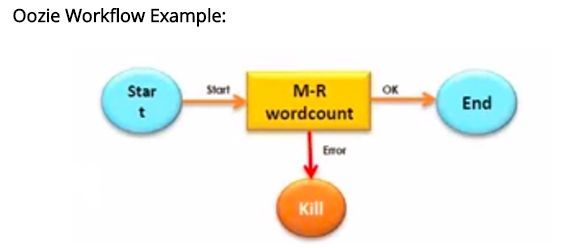
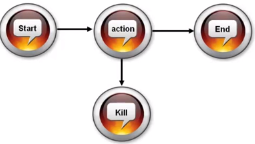
**Quick Ref: Oozie**

****

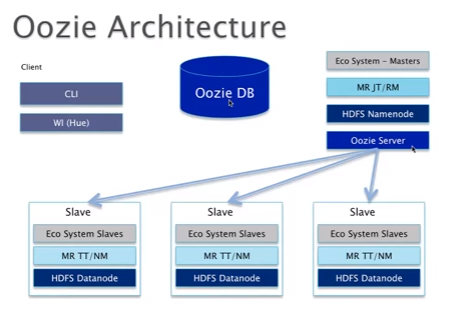
|  |  |
| --- | --- |
| Fork and Join | 1. Fork: The Fork splits an execution path into multiple concurrent execution paths 2. Join: Join waits until all the forked paths have executed 3. The fork and join must be used in pairs |
| Decision | 1. Capability to select the execution path 2. Implemented as a case statement 3. Consists of list if predicate and pairs. Predicates are evaluates in the order of appearance 4. Plus a default transitions. |

****

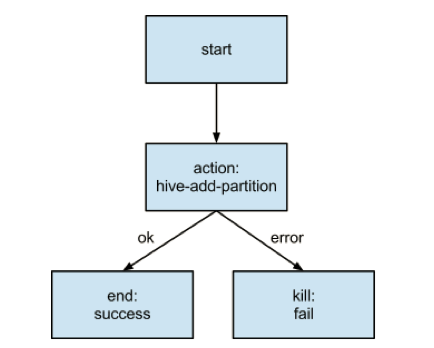
****

****

|  |  |
| --- | --- |
| Workflow | <workflow-app nome='wordcount –wf’>  <start to= ‘wordcount’/>  <action name=’Wordcount'>  <map-reduce>  <job-tracker>foo.com:9001</job-tracker>  <name-node>hdfs://bar.com:9000</name-node>  <configuration>  <property>  <name>mapred.input.dir</name>  <value>${inputDir}</value,>  </property>  <property>  <name>mapred.output.dir</name>  <value> ${outputDir}</value>  </property>  </configuration>  </map-reduce>  <ok to='end’/>  <error to='kill'/>  </action>  <kill name='kill'/>  <end name='end'/>  </Workflow-app> |
| Execution of WF | 2016-06-26 04:32:55,558 INFO ActionStartXCommand:520 - SERVER[ip-172-31-13-154.ec2.internal] USER[saranvisa2830] GROUP[-] TOKEN[] APP[Kumar\_WF1] JOB[0000870-160604182128967-oozie-oozi-W] ACTION[0000870-160604182128967-oozie-oozi-W@:start:] Start action [0000870-160604182128967-oozie-oozi-W@:start:]  with user-retry state : userRetryCount [0], userRetryMax [0], userRetryInterval [10] |
|  |  |

****

**Simplified workflow architecture:**

****

|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Desc** |
|  | **Web Ref** |  |
|  | About Oozie | <http://www.edureka.co/blog/brief-introduction-to-oozie/> |
|  | Schedule Job | <http://blog.cloudera.com/blog/2013/01/how-to-schedule-recurring-hadoop-jobs-with-apache-oozie/> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **General Info** | Oozie is a workflow scheduler system to manage Apache Hadoop jobs. Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs such as Java MapReduce, Streaming MapReduce, Pig, Hive and Sqoop. Oozie is a scalable, reliable and extensible system. Oozie is used in production at Yahoo!, running more than 200,000 jobs every day |
|  | **Features of Oozie** | * Execute and monitor workflows in Hadoop * Periodic scheduling of workflows * Trigger execution of data availability * HTTP and command line interface and web console |
|  | **Oozie Architecture** | 1. Oozie Repository 2. Database to store workflow definition and details 3. Customize Oozie Repository with MySQL 4. Go to the host on which Oozie is running. 5. Install Mysql connector 6. Go to MySQL db and create db name Oozie 7. clouder manager -> Oozie -> configuration (view/edit)->change DB parameter 8. Also enable create db by adding parameter using safety valve for oozie-site.xml 9. Start service 10. Oozie Server 11. Oozie Client |
|  | **Expression Language (EL) functions**  *<build-in functions>* | # Built-in Functions (EL functions)   1. Basic EL Constants 2. KB: 1024 (kilo bite) 3. MB: 1024 \* KB 4. GB: 1024 \* MB 5. TB: 1024 \* GB 6. PB: 1024 \* TB 7. Basic EL Functions 8. String function: trim(), Concat() 9. Date & Time functions: timestamp() 10. etc 11. etc |
|  | **Commands**  *<samples given below>* | Application Deployment:  $ hadoop fs-put wordcount-wf hdfs://bar.com:9000/usr/abc/wordcount  Workflow Job Parameters:  $ cat job.properites  Oozie.wf.application.path=hdfs://bar.com:9000/usr/abc/wordcount  Input=/usr/abc/input-data  Output=/usr/abc/output-data  Job Execution:  $ oozie job –run –config job.properties  Job:1-20090525161321-oozie-xyz-W |
| 1 | **About Oozie** | 1. Oozie helps to chain multiple hadoop jobs(mapreduce, pig, hive, sqoop) together to form a data pipeline application. 2. Oozie is map reduce based work flow tool 3. When you submit a workflow, the workflow itself will be running as a MapReduce job. 4. It facilitates you to define work flows for related jobs 5. A workflow is DAG (Directed Acyclic Graph). 6. workflow will be stored as XML 7. Workflow is stored as hPDL (XML) 8. hPDL is a fairly compact language 9. It runs the entire work flow using one or more map reduce jobs. In larger clusters, Gateway nodes can become bottlenecks and Oozie can mitigate that issue as jobs are run in the cluster in the form of map reduce programs 10. Integrated with most of the Hadoop eco system tools |
| 2 | **Two components in Oozie** | 1. Oozie Workflows 2. Oozie Scheduler |
| 2.1 | **Oozie workflow** | Oozie work flow can have more than one jobs from   1. MapReduce –or -- 2. Hive --or 3. Pig –or 4. Sqoop –or   Combination of all the above |
| 3 | **Admin scope** | 1. Admin has to setup the cluster to support Oozie. And do a basic check before share to developer 2. Developer will develop the workflows. Admin will use the Oozie tool to schedule/run the workflows and submit Oozie jobs. |
| 4 | Oozie is the right candidate for ONLY large cluster | # Oozie is the right candidate for large cluster   1. In smaller cluster, we can submit jobs from gateway node where as in large cluster, if we use gateway node for all the jobs it will be over loaded. 2. Consider you have 100 MapReduce jobs to be submitted. Oozie comes in picture here, it can helps to chain multiple Hadoop jobs using work flow (like informatica workflow). So that we can submit only Oozie in gateway and Oozie will distribute loads to cluster 3. So oozie is not required in small cluster as Oozie itself will also use resources in cluster. |
| 5 | Parameter file | Oozie-site.xml |
| 6 | **Oozie folder structure** | # sample Oozie folder structure  # 'src', 'apps', 'inputdata' folders, in wchich apps uses inputdata folder to process jobs  # main directory is apps. it has multiple sub folders represents the Oozie supported systems (Ex: hive, pig, shell, streaming, cron, etc)  # So Oozie is not only supporting hadoop echo system like (hive, pig), it also supports non-hadoop eco systems (like sheel, cron, etc) |
| 7 | **Workflow.xml** | # workflow.xml: This is the main DAG, it controls the workflow  # All the below information will be given by developer  # Workflow is used to handle any type of applications like Yarn/MapReduce, Pig, hive, sqoop  # Common steps in Workflow: source data -> load into hadoop -> ETL on top of it -> After ETL, report the data back to reporting layer ->  # Ex: The below example shows we need at least 3 actions. It will be coordinated via oozie  # **Step1/Action1:** might use sqoop job to get data from source to Hadoop  # **Step2/Action2:** Use etl for some actions  # **Step3/Action3:** After ETL Copy data to Reporting layer  # In case of Hive, workflow.xml, you will see action name=Hive-node  # In case of MapReduce, workflow.xml, you will see action name=mr-node  # In case of pig, workflow.xml, you will see action name=pig-node |
| 7.1 | Workflow.xml Type | 1. Serial : 2nd job will start after the first one is complete 2. Parallel: run first and 2nd parallel and join those two result set for further process |
| 7.2 | Workflow.xml  *<sample>* | ## The below workflow is sample for serial  # You can also do parallel , then apply join logic to join both the datasets  <workflow-app xmlns="url:oozie:workflow:0.2" name ="map-reduce-wf">  <start to ="mr-node">  # In this example, only one action is given for one workflow, but in production, there would be multiple actions with same/different workflows  <action name=mr-node> # This is the important tag, talks about the action  <map-reduce> # Type of workflow  <job-tracker>${JobTracker}</job-tracker> # JobTracker will be used from job.properties. Also we can hardcode, but it is not recommanded  <name-node>${namenode}</name-node>  <prepare>  # examplesRoot will be used from job.properties. Also we can hardcode, but it is not recommanded  <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-date/${outputDir}"  <prepare>  <configuration>  <property>  <name>mapre.job.queue.name</name>  <value>${queueName}</value>  </property>  <property>  <name>mapred.mapper.class</name>  <value>org.apache.oozie.example.SampleMapper</value>  </property>  <name>mapred.output.value.class</name>  <value>org.apache.hadoop.io.NullWritable</value>  </property>  <property>  <name>mapred.map.tasks</name>  <value>1</value>  </property>  <property>  <name>mapred.input.dir</name>  <value>/user/${wf:user()}/lca/input</value>  </property>  <property>  <name>mapred.output.dir</name>  <value>/user/${wf:user()}/${exampleRoot/output-data/${OutputDir}}</value>  </property>  </Configuration>  </map-reduce>  <ok to='pig-node'>  <error to='fail'>  </action>  # In case of Pig, workflow.xml, you will see action name=pig-node  <action name='pig-node'>  <pig>  <job-tracker>${JobTracker}</job-tracker>  <name-node>${namenode}</name-node>  <prepare>  <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-date/${outputDir}"  <prepare>  <configuration>  <property>  ..........................  </pig>  <ok to='end'>  <error to='fail'>  ...............  <kill name='fail'>  <message>MapReduce failed, error message [${wf:errormessage(wf:lasterrornode())}]  </kill>  <end name='end'/>  </workflow-app> |
| 8 | Job.properties | Some property & path to be passed as an argument, so that you can reuse it in many places… |
| 8.1 | Job.properties  *<update it>* | >vi job.properties # Do the below changes  # job.properties has the following info  # In production, we have to use actual ip address instead of localhost  # \*Note: If you have High Availability Enabled then use hdfs://service\_ID instead of IP Address (no port mentioned)  namenode=hdfs://localhost:8020 -- Default value with port 8020  namenode=hdfs://service\_ID  # \*Note: if you are using Yarn Resource Manager then use jobtracker=hostname\_where\_u\_hv\_RM:8032 and port 8032  else  jobtracker=localhost:8021 -- Default value with port 8021. 8021 is for classic and 8032 is for YARN  queuename=default  examplesRoot=examples # This is the parameter used in below oozie command. Replace 'examples' with proper path  # below parameter is hdfs directory  oozie.wf.applicatin.path=${namenode}/user/${user.name}/${examplesRoot}/apps/map-reduce/workflow.xml # Uses 'examplesRoot' as parameter from above  outputDir=map-reduce |
| 9 | Lib  *<folder>* | # lib: has the jar file which needs to be executed. oozie-examples-4.0.0-cdh5.2.1.jar |
| 10 | Default log path | # Note: As jobs are submitted as MapReduce or Yarn or Hive, etc… you need to go through respective logs in addition to oozie logs to trouble shoot any issue  /var/log/oozie |
| 11 | **Scheduling Recurring Workflows** | # Scheduling Recurring Workflows (Schedule jobs)  Oozie has another type of a job called a coordinator application. Coordinator applications allow users to schedule more complex workflows, including workflows that are scheduled regularly, or that have dependencies on the output from other workflows  For more details with sample:  <http://blog.cloudera.com/blog/2013/01/how-to-schedule-recurring-hadoop-jobs-with-apache-oozie/> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **Oozie Command** |  |
| 1 | After configuration | #Go to gateway node and do the below steps |
| 2 | Oozie status after configuration | # To check the status of Oozie  >oozie admin -oozie http://ip\_where\_oozie\_installed:11000/oozie -status |
| 3 | Oozie version | # To check the Version of Oozie  >oozie admin -oozie http://ip\_where\_oozie\_installed:11000/oozie -version  Oozie Server build version : 4.0.0-cdh5.2.1 |
| 4 | How to run a jar file | # First find the jar using  >sudo find / -name "\*filename.jar"  >find . -name "\*.jar"  # Run the below command. If it shows error 'jar not found' then run the next two commands  >jar tvf lib/oozie-examples-4.0.0-cdh5.2.1.jar |
| 4.1 | How to run a jar file | # The below command shows the jar existence in your system. Once you confirmed  > /usr/java/jdk1.7.0\_67-coudera/bin/jar  # Run the below command from map-reduce directory. It shows all the classes for the 'src' directory  map-reduce>/usr/java/jdk1.7.0\_67-coudera/bin/jar -tvf lib/oozie-examples-4.0.0-cdh5.2.1.jar  #what is tvf in linux? |
| 5 | Move file to hdfs | #Once you make all the changes in local, move it to hdfs  >hadoop fs -put examples /user/hduser |
| 6 | Valiedate job.properties | # Valiedate job.properties (oozie.wf.applicatin.path)  #Check the oozie.wf.applicatin.path mentioned in job.properties path is availablein hdfs  oozie.wf.applicatin.path=${namenode}/user/${user.name}/${examplesRoot}/apps/map-reduce/workflow.xml  >hadoop fs -ls /user/hduser/examples/apps/map-reduce/workflow.xml |
| 7 | Run Oozie job | #Note: Now we are running Oozie from the OS commandline, so it won't understand hdfs path, so give local path where job.property available  >oozie job -oozie http://ip\_where\_Oozie\_is\_installed:11000/oozie -config /home/hduser/demo/oozie/examples/apps/mapreduce/job.properties –run  # it will give the oozie\_job-id |
| 8 | Validate Oozie Job | >oozie job -oozie http://ip\_where\_Oozie\_is\_installed:11000/oozie -info oozie\_job\_id |
| 9 | Monitor using Resource Manager UI | <http://RMinstalledhost:8088>  # Get Output directory from WUI -> Configuration -> search (outputdir)  hdfs://nameservice1/user/hduser/emamples/output-data/map-reduce  >hadoop fs -ls hdfs://nameservice1/user/hduser/emamples/output-data/map-reduce  >hadoop fs -cat hdfs://nameservice1/user/hduser/emamples/output-data/map-reduce/part\* |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **Troubleshoot the issue** |  |
|  | Log | * 1. Wf.id()   2. Wf.name()   3. Wf.user()   4. Wf.errorcode() – returns error code for the specified action node   5. Wf.errormessage()   6. Wf.lastErrorNode() – returns the name of the last workflow action node that exited with an error |
|  | Log  *<Practical Example>* | ….  WARN HiveActionExecutor:523 - SERVER[ip-172-31-13-154.ec2.internal] USER[saranvisa2830] GROUP[-] TOKEN[] APP[Kumar\_WF1] JOB[0000870-160604182128967-oozie-oozi-W] ACTION[0000870-160604182128967-oozie-oozi-W@hive-d9bc] Launcher ERROR, reason: Main class [org.apache.oozie.action.hadoop.HiveMain], exit code [10001]  2016-06-26 04:33:24,933 INFO ActionEndXCommand:520 - SERVER[ip-172-31-13-154.ec2.internal] USER[saranvisa2830] GROUP[-] TOKEN[] APP[Kumar\_WF1] JOB[0000870-160604182128967-oozie-oozi-W] ACTION[0000870-160604182128967-oozie-oozi-W@hive-d9bc] ERROR is considered as FAILED for SLA  …..  2016-06-26 04:33:24,960 INFO ActionStartXCommand:520 - SERVER[ip-172-31-13-154.ec2.internal] USER[saranvisa2830] GROUP[-] TOKEN[] APP[Kumar\_WF1] JOB[0000870-160604182128967-oozie-oozi-W] ACTION[0000870-160604182128967-oozie-oozi-W@Kill] [\*\*\*0000870-160604182128967-oozie-oozi-W@Kill\*\*\*]Action status=DONE |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  | **Oozie Example** |  |
|  | Oozie workflow for hive query | <https://www.youtube.com/watch?v=7lCaL2gZiXo> |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. I have three different type of jobs running on the data in HDFS. These three jobs have to be run separately in the current scenario. Now, we want to run the three jobs together by piping the OUTPUT data of one job to the other job without writing the data in HDFS to improve the architecture and overall performance?

Oozie helps to chain multiple hadoop jobs(mapreduce, pig, hive, java etc.) together to form a data pipeline application. The built-in support of scheduling and hadoop-related functions makes dev's life much easier to manage complex hadoop related jobs.

However Oozie doesn't necessarily eliminate data storage in HDFS or other forms such as local file system or database, etc. To do that you would need to introduce some in-memory data store, message-queue systems or other system which works for the scale of data you have