What is Oozie?

It is an automation framework apart of Hadoop ecosystem component which helps us to automate the work & link several work together in Hadoop Programming

Say you want to perform below operation serially.

Local Input File→load it to hdfs→Run Map-Reduce Job->run pig job->sqoop to export it back to database->end of the program

Let's say somebody pushes the data into the local file system daily at certain time say 8:00 AM and you would like to start the above operation exactly at 10.00 AM. Since it is daily job, it will be difficult to perform daily and we would definitely like to automate it.

We can use oozie for that.

In oozie the above sequence of work is called work flow.

1. Oozie is a workflow schedular system to manage Apache Hadoop jobs.
2. Oozie workflow jobs are Directed A-Cyclical Graphs(DAG) of actions.
3. Oozie Coordinator jobs are recurrent Oozie workflow jobs triggered by time and data availability
4. You can use Map-reduce, Pig, Hive, Sqoop, Distcp, Java programs, shell scripts etc as a part of the job.
5. Oozie is scalable,reliable and extensible system

Main Features of Oozie

- Execute and monitor workflow in hadoop
- Periodic scheduling of workflows
- Trigger execution by data availability
- HTTP and Command line interface +web Console
- In Production, Yahoo uses more than 2,00,000 job/day, where they need some sort of scheduling.
- We can integrate hadoop ecosystem with external world using oozie ,coz oozie supports webservices

What is workflow

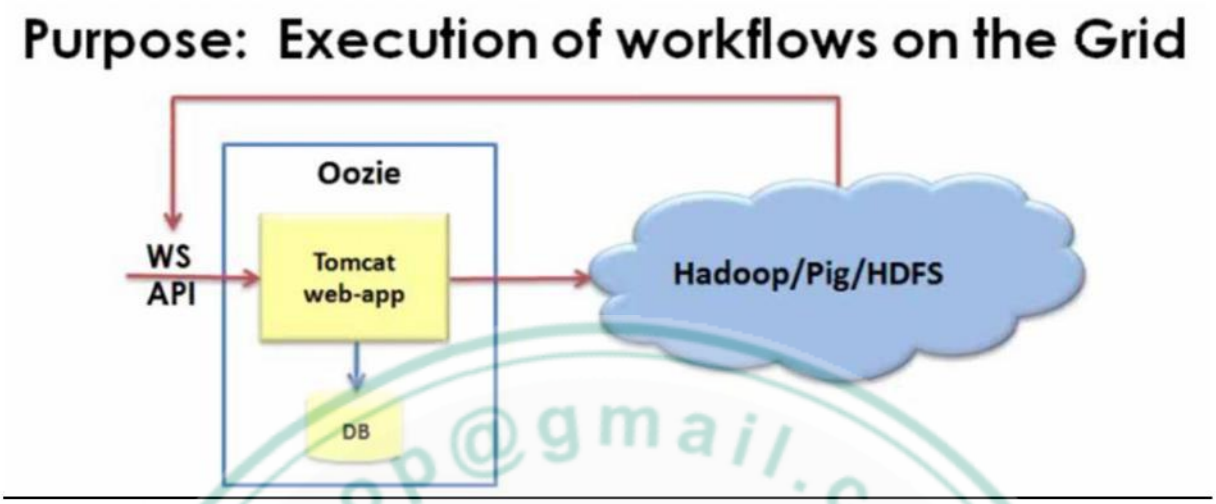
Continuous Process where one Job output will be input to other job.

1. 1st step could be some sort of Map-reduce job, doing data cleaning, ETL , Pre-processing
2. 2nd step cud be running pig or hive queries over the output of the 1st.
3. 3rd step could be execute the algorithm and analytics as per your business logic
4. 4th store the output to NO-SQL database or RDBMS as per your choice
5. 5th Your downstream application will read the data from database as per their need and display to the end user.

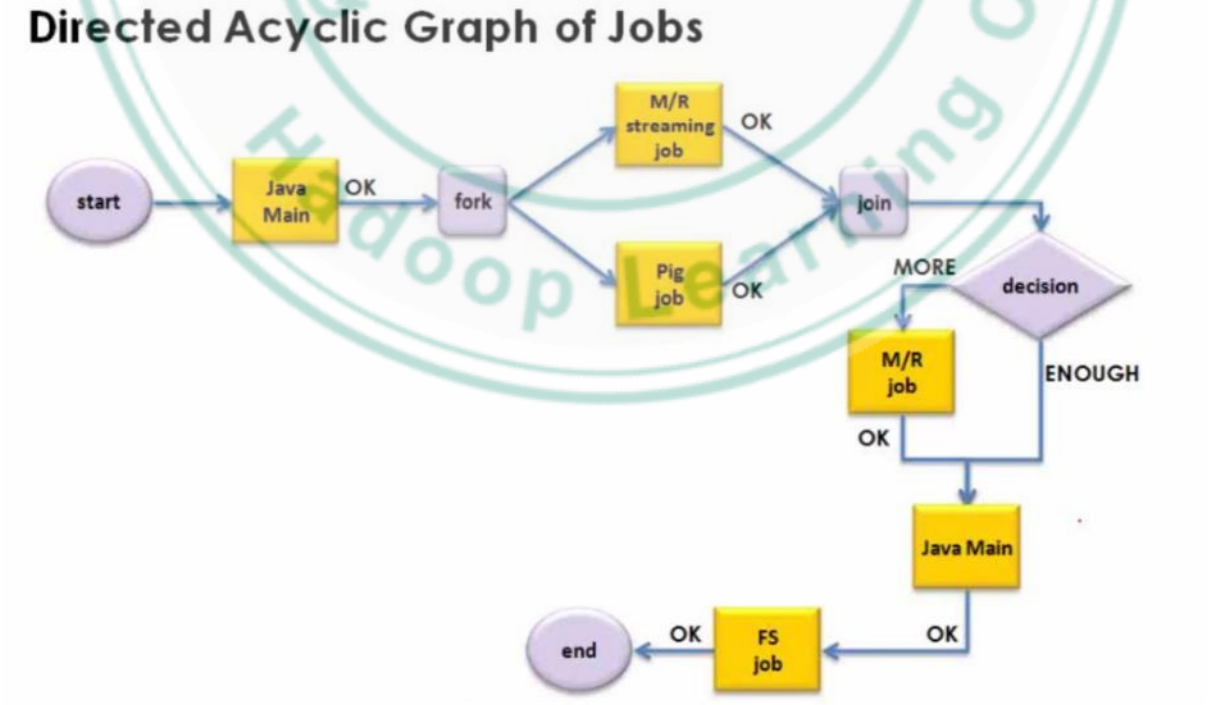
Mostly we will have a requirement where we would be using all these ecosystem together in our job. Oozie workflow can contain N number of steps, as per the business requirement, and in each of these steps we can place any ecosystem component(like map-reduce, pig, hive, sqoop ...)

It is practically important and easy to follow component.

Oozie Workflow(Birds eye view)



1. Oozie is actually a web-application, that is running on Tomcat Server.
2. Default database in derby, but we can use any database
3. Cloudera uses mysql database whereas pivotal/Hortonworks uses postgres
4. Database will store information about the workflow for example the sequences, which workflow ran at what time, its status, next Action name and so on.
5. We can also use external java API to run the workflow instead of just running it from terminal.
6. The oozie component can also be exposed as web services so as to call it from any programming language. Now dot-Net application can also call oozie to perform the job and activities.



1. Yellow: Action/Processing nodes(Hadoop components + Java components)

2. Light blue: Control flow nodes, defines the flow that can be controlled
3. FS jobs means hdfs operation like write data to hdfs.
4. Java Main means Java Application
5. Fork means doing things in parallel
6. Join means merging the jobs together.

2 Types of nodes

- Control Flow(Colored light-blue in the above diagram)
  - start/end/kill every work flow has start action ,end action and kill(termnate the workflow)
  - decision what do do when action fails or success
  - fork/join(forking means doing something parallel) executing the action in parallel
- Actions (Colored yellow in the above diagram) Actions are the steps in workflow that perfrom some specific action or task using these components
  - map-reduce
  - pig
  - hdfs
  - sub-workflow
  - java-run (custom java code)
- Extension Action(https://oozie.apache.org/docs/4.2.0/)
  - Email Action
  - Shell Action
  - Hive Action
  - Hive 2 Action
  - sqoop Action
  - Ssh Action
  - DistCP Action
  - Spark Action
  - Writing a Custom Action Executor

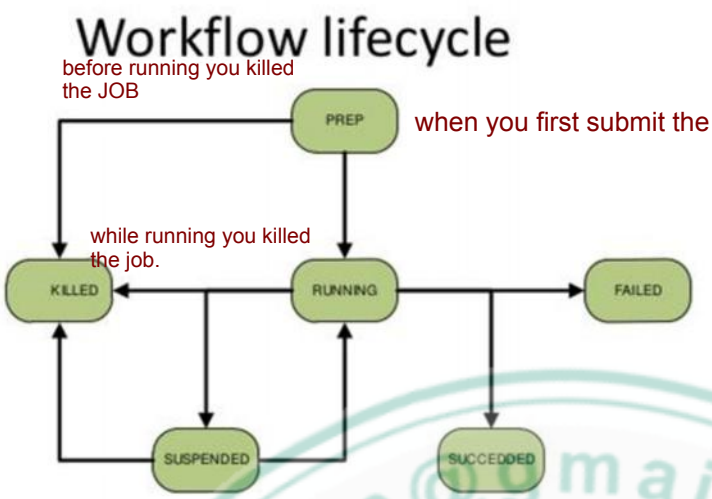
In practice, there are different types of Oozie jobs:

3 Types of Job in Oozie

There are 3 types of jobs in oozie:

- **Oozie workflow jobs :**DAGs of actions which are jobs such as shell scripts, Map-Reduce, Sqoop, Streaming, Pipes, Pig, Hive etc
- **Oozie coordinator jobs:** Invoke Oozie workflow jobs based on specified event triggers-date/time or data availability
- **Oozie bundle jobs:** Facilitates packaging multiple coordinator and workflow jobs, and makes it easier to manage the life cycle of those jobs.

Life cycle of Oozie job:



Can we do the above work without using oozie?

Yes. Using java api you can write map-reduce, Pig, Hive,....

JAVA Way	Oozie Way
If Workflow sequence changes then java code needs to be changed. The flow should not be programmed. The java code need to be recompiled and re-packaged.	In oozie the workflow is flexible and can be changed just by changing the configuration.
If your workflow stops in the middle due to some problem. Then we need to restart it from the beginning	In Oozie you can resume from the place where it was failed last time
You will have to manually use JavaMail API to send the status of a job in case it got failed	We can easily configure email whenever a Job starts or ends or gets failed
You may have to use Batik API,other reporting tools to display the workflow in graphical way	Oozie shows you graphical interface of your workflow by default

More about OOzie

- Oozie runs as a server and a client submits a workflow to the server.
- In oozie workflow is a DAG of action nodes and control flow nodes.
- Action flow nodes performs a workflow task, like moving the file, running a map-reduce job, running a Pig Job.
- A control flow node govern the workflow execution between actions by allowing such construct as conditional logic(so different execution branches may be followed depending on the result of an earlier action node) or parallel execution.

Call back Mechanism:-

When a workflow completes, oozie can make an http call-back to the client to inform it of the workflow status. It is also possible to receive call back every time the workflow enters or exists an action node.

Oozie allows failed workflows to be re-run from an arbitrary point (failed point).This is useful for dealing with transient errors when the early actions in the workflow are time consuming to re-execute.

Assume you have a oozie job, where the final step is to write the data on hdfs or local file system. But due to some permission issue you job execution failed. We can resume the job later from the failed point once the permission are given.

Resource needed to run oozie workflow in HDFS

1. workflow.xml combination of Actions
2. config-default.xml(if needed)
3. lib directory with all the libraries needed (if any)
4. Pig script (if needed)
5. Hive queries (if needed)
6. job.properties (Initializing properties) passing parameter to jobs

Sample Workflow.xml MUST REFER THE EXMAPLE of WORKFLOW PROVIDED BY cloudera

there wont be any driver program fro map reduce when you schedule job in oozie you need to give all the properties in workflow.xml only

They contain Job Tracker ,Namenode information. It is a Map-reduce Action.This tag varies and will be different for pig,hive sqoop...

```
<workflow-app name="wordcount-wf"> name of workflow/job
    <start to="wordcount" /> starting action point.it is compulsary even if u have one action

    <action name="wordcount"> unique name of actions identified within workflow
        <map-reduce>
            <job-tracker>${jobTracker}</job-tracker> actual value of job tracker present in properties.xml file or you can give job
            <name-node>${nameNode}</name-node>       tracker value here only

            <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="myEnd" />
        <error to="myFail" />
    </action>

    <kill name="myFail">
        <message>Map/Reduce failed</message>
    </kill>

    <end name="myEnd" />
</workflow-app>
```

This is just the sample xml file. In the real xml you will also have mapper, reducer, combiner, no of reducers ...configured within the <properties>.You can otherwise pass Mapper, reducer... in a separate properties file(job.properties) instead of passing it here.

You can also have tags like prepare which will be executed before running the job

```
<prepare>
        <delete path="${nameNode}/user/${wf:user()}/${examplesRoot}/output-data/${outputDir}"/>
        <mkdir path=""/>
</prepare>
```

Session 3:Running Predefined Oozie example in Oozie 5.5.0

We will be using cloudera quickstart 5.5.0-vmware for this.
The following setting is recommended for smooth execution of cloudera.
Cloudera 5+: 8 GB of RAM +2 processors
Cloudera 4+: 4 GB of RAM +2 processors

1. **Copy the examples bundle from its location to desktop** example provided by cloudera for oozie

```
[cloudera@localhost ~]$ cp /usr/share/doc/oozie-4.1.0+cdh5.5.0+222/oozie-examples.tar.gz ~/Desktop
```

2. **Extract oozie-examples.tar.gz into a folder**

```
$ cd ~/Desktop
[cloudera@localhost Desktop]$ tar xvf oozie-examples.tar.gz
```

The above command will extract the zip into examples directory in desktop. Inside the examples folder you will have below directories.

```
examples(Main Folder)
    apps
    input-data
    src
```

3. **Copy the entire examples directory to HDFS**

```
[cloudera@localhost Desktop]$ hadoop fs -put examples /user/cloudera/examples
```

4. **Verify if your copy is successful.**

```
$ hadoop fs -ls /user/cloudera/examples
```

Or use HUE to navigate and check the file content.

5. In the local file system navigate to the below location. This is the location where your examples are present.

```
$cd /home/cloudera/Desktop/examples/apps
```

6. **Execute the below command from the above location**

```
$ oozie job -run --oozie http://localhost:11000/oozie -config  map-reduce/job.properties
job: 0000000-150727122224358-oozie-oozi-W
```

You may get this Error

Error: E0901: NameNode [localhost:8020] not allowed, not in Oozie's white list. Allowed values are: [quickstart.cloudera:8020]
E0900: JobTracker [localhost:8021] not allowed, not in Oozie's white list. Allowed values are: [quickstart.cloudera:8032]

Solution:-

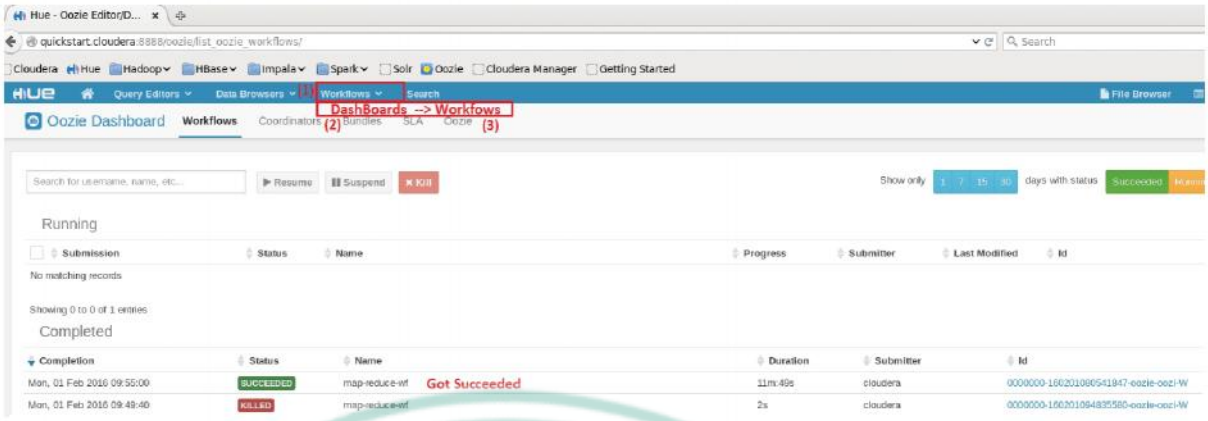
```
[cloudera@quickstart apps]$ vim map-reduce/job.properties
```

```
nameNode=hdfs://quickstart.cloudera:8020
jobTracker=quickstart.cloudera:8032
queueName=default
examplesRoot=examples
```

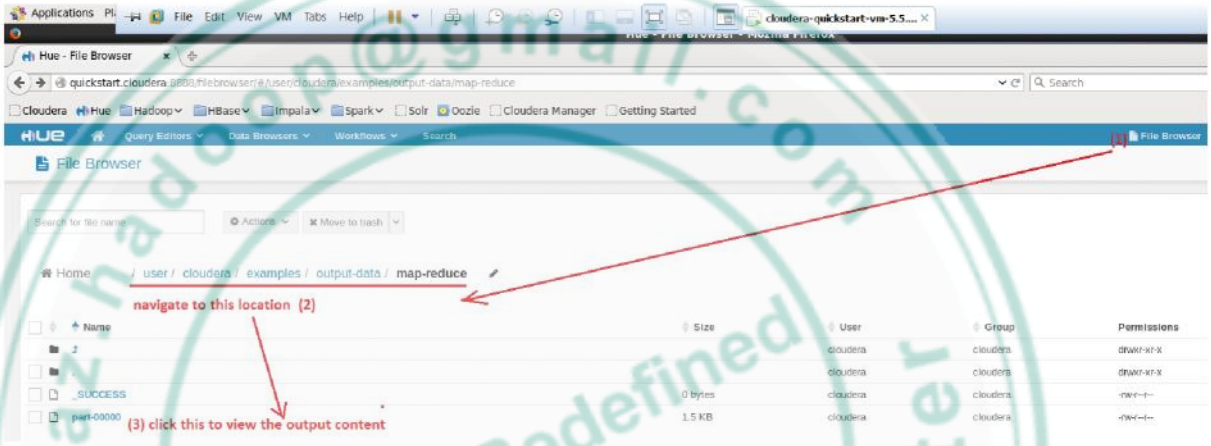
We have changed these from localhost to quickstart.cloudera

Again submit the job you should be successful.

7. **Check Your Job Progress using Hue.**
   a. Launch Hue using http://localhost:8888/home
   b. Use the credential to login Username/Password :cloudera / cloudera
   c. Click the "WorkFlows->DashBoards->Workflows".
   d. You will see your workflow running. Of course, you can use the oozie command to get the status as well.
   e. In case your job failed, try restarting your Cloudera Manager all the services and again try it



8. **See the output directory of the above job execution using File Browser**

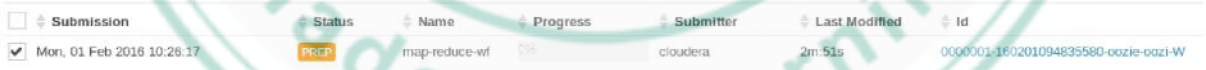


Oozie Life Cycle

PREP→START→RUNNING

Submit a workflow: The job will be created, but it will not be started, it will be in `PREP` status.

```
$ oozie job -submit --oozie http://localhost:11000/oozie -config map-reduce/job.properties
job: 0000001-160201094835580-oozie-oozi-W
```



starting a workflow:

Syntax:oozie job -start --oozie http://localhost:11000/oozie <JOB-ID>

```
$ oozie job --oozie http://localhost:11000/oozie  -start   0000001-160201094835580-oozie-oozi-W
```

The `start` option starts a previously submitted job that is in `PREP` status.
After the command is executed the job will be in `RUNNING` status

Directly Running a Workflow

```
$oozie job --oozie http://localhost:11000/oozie -config job.properties -run
```

The `run` option creates and starts job.The job will be in `RUNNING` status.

Suspending a Workflow, Coordinator or Bundle Job

```
$ oozie job -oozie http://localhost:8080/oozie -suspend    0000001-160201094835580-oozie-oozi-W
```

The `suspend` option suspends a job in `RUNNING` status. After the command is executed the job will be in `SUSPENDED` status.

Resuming a Workflow, Coordinator or Bundle Job

```
$ oozie job -oozie http://localhost:8080/oozie -resume 0000001-160201094835580-oozie-oozi-W
```

The `resume` option resumes a job in `SUSPENDED` status. After the command is executed the workflow job will be in `RUNNING` status.

Killing a Workflow, Coordinator or Bundle Job

```
$ oozie job -oozie http://localhost:8080/oozie -kill 0000001-160201094835580-oozie-oozi-W
```

The `kill` option kills a workflow job in `PREP` , `SUSPENDED` or `RUNNING` status and a coordinator/bundle job in `PREP`, `RUNNING` , `PREPSUSPENDED` , `SUSPENDED` , `PREPPAUSED` , or `PAUSED` status.

After the command is executed the job will be in `KILLED` status.

Rerunning a Workflow Job

```
$ oozie job -oozie http://localhost:8080/oozie -config job.properties -rerun 0000001-160201094835580-oozie-oozi-W
```

The `rerun` option reruns a completed (`SUCCCEDED` , `FAILED` or `KILLED`) job skipping the specified nodes. You must supply a list of action nodes to skip inside of the job.properties file using the oozie.wf.rerun.skip.nodes variable.

Session 4:Running WordCount Example in Cloudera 5.5.0

1. Copy the input_data folder into Cloudera desktop.
2. Open Terminal & issue the below command to copy the data.

```
$ hadoop  fs  -put  ~/Desktop/input_data  /user/cloudera/
```

3. Confirm if the data is copied successfully to hdfs using command or FileBrowser(HUE)

```
$ hadoop fs -ls /user/cloudera/input_data
Found 2 items
-rw-r--r--   3 cloudera cloudera         82 2015-08-09 22:25 /user/cloudera/input_data/word1.txt
-rw-r--r--   3 cloudera cloudera         48 2015-08-09 22:25 /user/cloudera/input_data/word2.txt
```

4. Create a java project & add Mapper, Reducer, Driver program.
   While writing Driver program,use port number 8020 for hdfs
   hdfs://localhost:8020/user/cloudera/input_data/
5. Add the below jar file to the class path.

available in /usr/jars

- avro-1.7.6-cdh5.5.0.jar
- commons-cli-1.2.jar
- commons-collections-3.2.1.jar
- commons-configuration-1.6.jar
- commons-httpclient-3.1.jar
- commons-lang-2.6.jar
- commons-logging-1.2.jar
- guava-15.0.jar
- hadoop-auth-2.6.0-cdh5.5.0.jar
- hadoop-common-2.6.0-cdh5.5.0.jar
- hadoop-hdfs-2.6.0-cdh5.5.0.jar
- hadoop-mapreduce-client-common-2.6.0-cdh5.5.0.jar
- hadoop-mapreduce-client-core-2.6.0-cdh5.5.0.jar
- hadoop-yarn-api-2.6.0-cdh5.5.0.jar
- hadoop-yarn-common-2.6.0-cdh5.5.0.jar
- htrace-core-3.2.0-incubating.jar
- htrace-core4-4.0.1-incubating.jar
- jackson-core-asl-1.9.12.jar
- jackson-mapper-asl-1.9.12.jar
- log4j-1.2.17.jar
- protobuf-java-2.5.0.jar
- servlet-api-2.5.jar
- slf4j-api-1.7.5.jar

Note: This are the minimum Jars needed to run Map-reduce on Cloudera Box

6. Run your java Application from Eclipse.

Lets execute the Program in cluster Mode:-

1. Copy input_data folder present along with this document into cloudera Desktop and issue the below command to copy the input_data into hdfs location

```
[cloudera@localhost Desktop]$ hadoop fs -put input_data /user/cloudera/
```

2. To run the program in cluster mode,Simply create a Jar file of the project.Use the jar present along with the folder.While creating the Jar file,no need to include lib folder that contains all the jars.Since we are going to execute the jar in the cluster mode Just include the Driver,Mapper,Reducer class while creating the jar.
   Note: If you create the jar file from _1WordCountExampleHardcodedIOPaths project ,create the jar in your Desktop and execute the below code

```
[cloudera@localhost Desktop]$ hadoop jar MyWordCount.jar p1.MyDriver
```

   However if you create the jar from _2WordCountExampleSoftCodedIOPaths project,then create the jar in your Desktop and execute the below code.We have removed the hardcoded path in 2nd program so we need to provide the path explicitly while executing the jars.

```
[cloudera@localhost Desktop]$ hadoop jar MyWordCount.jar p1.MyDriver input_data output_data
15/08/10 00:57:59 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments.
Applications should implement Tool for the same.
15/08/10 00:57:59 INFO input.FileInputFormat: Total input paths to process : 1
15/08/10 00:58:00 INFO mapred.JobClient: Running job: job_201508092223_0001
15/08/10 00:58:01 INFO mapred.JobClient:  map 0% reduce 0%
15/08/10 00:58:07 INFO mapred.JobClient:  map 100% reduce 0%
15/08/10 00:58:11 INFO mapred.JobClient:  map 100% reduce 100%
```

3. Check the ouput of the program in the output_data folder

```
[cloudera@localhost Desktop]$ hadoop fs  -cat  /user/cloudera/output_data/part-r-00000
```

Chapter 5:Running WordCount example using oozie workflow

1. Create a folder named "wordcount-oozie" in cloudera Desktop.
2. Create a folder named "lib" inside "wordcount-oozie" and place "MyWordCount.jar" that we had created in previous class inside it.
   Note: In case you are creating MyWordCount.jar just to use it in oozie workflow,then you dont need to include the DriverClass and libraries as well.Oozie doesnot need any Driver Class.All the configuration in the Driver class will be done in workflow.xml.
3. Create the below 2 files inside "wordcount-oozie" folder.
   a. job.properties
   b. workflow.xml

   Note: Actually job.properties can be present anywhere, you don't need to keep it under wordcount-oozie folder. We will upload the wordcount-oozie folder in the hdfs but while execute the oozie job, we will use job.properties ie present in local file system.
   job.properties need not be present in the hdfs inside wordcount-oozie.
   job.properties contains a variable pointing to workflow.xml i.e, present in hdfs.

   But let's create everything in a single place so that we can always take them together.
   So the entire directory structure should be as below

```
wordcount-oozie
        -- job.properties
        -- workflow.xml
        -- lib
           -- MyWordCount.jar
```

4. Write the job.properties & workflow.xml as present in wordcount-oozie folder
5. Put the input_data folder from cloudera desktop to hdfs /user/cloudera.The input_data is present along with this example.You need to copy them to cloudera Desktop and then to hdfs

```
$ hadoop fs -put /home/cloudera/Desktop/input_data  /user/cloudera/
```

6. Copy the whole wordcount-oozie folder created in step 3 into cloudera desktop and then to hdfs.

```
$ hadoop fs -put /home/cloudera/Desktop/wordcount-oozie  /user/cloudera/
```

7. Execute the oozie workflow.change your pwd to Desktop/Wordcount-oozie

```
$ cd ~/Desktop/wordcount-oozie
$ oozie job   -oozie   http://localhost:11000/oozie -config   job.properties    -run
job: 0000000-150809222353171-oozie-oozi-W
```

   Note: In your local system you just need job.properties,whereas in hdfs you need workflow.xml and lib folder and not the job.properties.

8. Navigate to Hue and see the progress of Oozie job & check your output folder.
   Note: While creating a jar you need not include Driver program if you just want to execute the them as part of oozie

Chapter 6:Running WordCount Example as Co-ordinator job

1. Simply copy "wordcount-oozie-coordinator" folder into cloudera Desktop.
2. Edit coordinator.properties and change your start date and enddate and frequecy as needed
3. Copy the folder into hdfs location

```
$hadoop fs -put ~/Desktop/wordcount-oozie-coordinator .
```

4. Issue the below command to execute the co-ordinator job.

```
$oozie job -oozie http://localhost:11000/oozie -config ~/Desktop/wordcount-oozie-coordinator/coordinator.properties -run

job: 0000004-160207032040720-oozie-oozi-C
```

5. Go to Hue->Workflows->Dashboard ->Coordinator and you will see the coordinator getting scheduled.
6. The coordinator job is executed as workflow job, so you will need to check workflow to see the job getting executed every 5 minutes.

Requirement:

Let's say we have an xml file, containing employees details. You will get this file in input_data folder present in the same folder where this document is present

Sample empDetails.xml

```
<employees>
    <employee>
        <empId>101</empId>
        <empName>suraj</empName>
        <empSalary>5000</empSalary>
        <empCompany>TCS</empCompany>
        <empEmail>suraj@tcs.com</empEmail>
    </employee>

    <employee>
        <empId>102</empId>
        <empName>shyam</empName>
        <empSalary>7000</empSalary>
        <empCompany>capgemini</empCompany>
        <empEmail>shyam@capgemini.com</empEmail>
    </employee>

   ...
</employees>
```

We need to write a oozie workflow which takes xml file->executes map-reduce for parsing-> execute Pig for getting top3Paid employee->executes hive to get some column from the data.

1. Import the given project into Cloudera eclipse IDE
2. Create a jar file called "XMLProcessing.jar" from the Project.While creating the jar file you need not include the driver program as it is not needed in oozie.
3. Write your pig script,hive script,job.properties,workflow.xml to get below structure

```
xmlParsing
  -- job.properties
  -- workflow.xml
  -- Top3HighlyPaid.pig
  -- hiveScript.q
  -- lib
     -- XMLProcessing.jar
```

4. Lets copy the entire xmlParsing folder into cloudera Desktop.
5. Also copy input_xml folder into cloudera desktop.
6. Open Linux shell and change your working directory to Destkop.
7. Copy xmlParsing & input_xml folder into hdfs

```
$cd Desktop
$hadoop fs -put input_xml   /user/cloudera/
$ hadoop fs -put /home/cloudera/Desktop/xmlParsing  .
```

. means copy xmlParsing folder and its content to default directory in hdfs(/user/cloudera)

8. Execute the below command from Desktop to run your workflow

```
$ oozie job -oozie http://localhost:11000/oozie -config  xmlParsing/job.properties  -run
job: 0000000-150809222353171-oozie-oozi-W
```

9. Navigate to Hue and see the progress of Oozie job and check your output folder after completion.

Running WordCount Example as Co-ordinator job

1. Simply copy "wordcount-oozie-coordinator" folder into cloudera Desktop.
2. Check the current date and time-zone in your cloudera Desktop. By default the timezone is PDT.

```
[cloudera@quickstart ~]$ date
Mon Jun  6 09:02:08 PDT 2016
```

3. Let's say you want to configure job to run at 6-June-2016 09:10:00(After 8 min)
   Edit coordinator.properties and change your start date(Intended Time+7 Hours 5 minutes) and end-date and frequency as needed. As per our requirement we will have 9:10+7:05 = 16:15 in our co-ordinator properties (jobStart=2016-06-06T16:15Z)

4. Copy the folder into hdfs location

```
$hadoop  fs -put    ~/Desktop/wordcount-oozie-coordinator  .
```

5. Issue the below command to execute the co-ordinator job.

```
$oozie  job   -oozie   http://localhost:11000/oozie -config  ~/Desktop/wordcount-oozie-coordinator/coordinator.properties    -run

job: 0000004-160207032040720-oozie-oozi-C
```

6. Go to Hue->Workflows->Dashboard ->Coordinator and you will see the coordinator getting scheduled.
7. The coordinator job is executed as workflow job, so you will need to check workflow to see the job getting executed every 5 minutes.

Note: The timeZone is specified as IST(No value of this, It is always talking UTC time-format)

Assignment:-

It's now 06-June-2016 1:30 PM(at cloudera time). I want to schedule the job to run at 1:35 PM. What is the startTime I need to configure?

Answer: add 7:05 hours to it. It becomes 8:40 PM.= 20:40

jobStart=2016-06-06T20:40Z

Cloudera Timings Vs IST

Cloudera clock time is PDT(Pacific Daylight Time) which is -12:30 hours behind IST.

IST-12:30 = PDT

IST= PDT+12:30

Assume it's 07-June-2016 02:15 AM IST,and you want to schedule a job at 2:20 AM IST.

What would be your configuration in the coordinator.properties

Configuration time= PDT+7:05 hours

Lets find out PDT equivalent.

PDT= IST-12:30 07-June-2016 02:15 AM -12:30

PDT= 06-June-2016 1:45 PM

Conf time= 06-June-2016 1:45 PM+ 7:05

= 06-June-2016 8:50 PM

Session 7: Integrating Map-reduce Pig Hive using oozie workflow.

Let's say we have an empDetails.xml as shown below. You will get this file in input_xml folder present in the same folder where this document is present

Sample empDetails.xml

```
<employees>
    <employee>
        <empId>101</empId>
        <empName>suraj</empName>
        <empSalary>5000</empSalary>
        <empCompany>TCS</empCompany>
        <empEmail>suraj@tcs.com</empEmail>
    </employee>

    <employee>
        <empId>102</empId>
        <empName>shyam</empName>
        <empSalary>7000</empSalary>
        <empCompany>capgemini</empCompany>
        <empEmail>shyam@capgemini.com</empEmail>
    </employee>

   ...
</employees>
```

We need to write an oozie workflow which takes xml file->executes map-reduce for parsing-> execute Pig for getting top3Paid employee->executes hive to get some column from the data.

1. Import the given project into Cloudera eclipse IDE
2. Create a jar file called "XMLProcessing.jar" from the Project. While creating the jar file you need not include the driver program as it is not needed in oozie.
3. Write your pig script, hive script, job.properties, workflow.xml to get below structure

```
xmlParsing
  -- job.properties
  -- workflow.xml
  -- Top3HighlyPaid.pig
  -- hiveScript.q
  -- lib
     -- XMLProcessing.jar
```

4. Lets copy the entire xmlParsing folder into cloudera Desktop.
5. Also copy input_xml folder into cloudera desktop.
6. Open Linux shell and change your working directory to Destkop.
7. Copy xmlParsing & input_xml folder into hdfs

```
$cd Desktop
$hadoop fs -put input_xml   /user/cloudera/
$ hadoop fs -put /home/cloudera/Desktop/xmlParsing  .
```

. means copy xmlParsing folder and its content to default directory in hdfs(/user/cloudera)

8. Execute the below command from Desktop to run your workflow

```
$ oozie job -oozie http://localhost:11000/oozie -config  xmlParsing/job.properties  -run
job: 0000000-150809222353171-oozie-oozi-W
```

9. Navigate to Hue and see the progress of Oozie job and check your output folder after completion.

Running Job based on Data-availability

User story:- Let's assume there are 2 locations in hdfs where data comes from different sources. The 2 locations are

1. hdfs:///user/cloudera/data1
2. hdfs:///user/cloudera/data2

Assume everyday 2 different databases are queried and the results from these are kept as per the date. Like in the path shown below

hdfs:///user/cloudera/data1/${YEAR}/${MONTH}/${DAY}/
hdfs:///user/cloudera/data2/${YEAR}/${MONTH}/${DAY}/
You would like to trigger the workflow only when you get data in these 2 folders every day.

Steps:-

1. copy data1 folder given along the document to cloudera Desktop.
2. Change your current directory to Desktop

```
[cloudera@quickstart ~]$ cd Desktop
```

3. Create data1 and data2 folder with the current date information

```
[cloudera@quickstart  Desktop]$ hadoop fs -mkdir -p /user/cloudera/data1/2016/10/15
[cloudera@quickstart  Desktop]$ hadoop fs -mkdir -p /user/cloudera/data2/2016/10/15
```

4. Put some data in data1 and data2 folder of hdfs

```
[cloudera@quickstart  Desktop]$  hadoop fs -put data1/* /user/cloudera/data1/2016/10/15/
[cloudera@quickstart  Desktop]$  hadoop fs -put data1/* /user/cloudera/data2/2016/10/15/
```

Note: We have kept the same file in both data1 and data2

5. Copy the oozie project into cloudera desktop and then copy them into the hdfs

```
[cloudera@quickstart Desktop]$ hadoop  fs -put   Nyse_Processing_Pig_Data_Availablility  .
```

6. Let's say you want to configure job to run at 6-June-2016 09:10:00(After 8 min)
   Edit coordinator.properties and change your start date(Intended Time+7 Hours 5 minutes) and end-date and frequency as needed. As per our requirement we will have 9:10+7:05 = 16:15 in our co-ordinator properties (jobStart=2016-06-06T16:15Z)

7. Issue the below command to execute the co-ordinator job.

```
$ oozie  job   -oozie   http://localhost:11000/oozie -config
~/Desktop/Nyse_Processing_Pig_Data_Availablility/coordinator.properties    -run

job: 0000004-160207032040720-oozie-oozi-C
```

8. Go to Hue->Workflows->Dashboard ->Coordinator and you will see the coordinator getting scheduled.
9. The coordinator job is executed as workflow job, so you will need to check workflow to see the job getting executed.