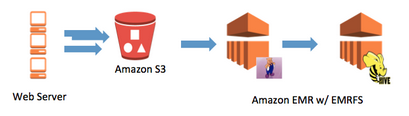
**ETL processing with Data Pipeline and Pig/Hive**

This lab covers how to build an ETL workflow that uses AWS Data Pipeline to schedule an Amazon Elastic MapReduce (Amazon EMR) cluster to clean and process web server logs stored in an Amazon Simple Storage Service (Amazon S3) bucket.

You will be creating the following ETL workflow:



This ETL workflow pushes webserver logs to an Amazon S3 bucket, cleans and filters the data using Pig scripts, and then generates analytical reports from this data using Hive scripts. AWS Data Pipeline allows you to run this workflow for a schedule in the future and lets you backfill data by scheduling a pipeline to run from a start date in the past.

We will use the AWS CLI to create and activate our ETL pipeline.

**Six-step Workflow**

Step 1: Check if log files are available in the Amazon S3 bucket.

Step 2: Create an Amazon EMR cluster with EMRFS on it.

Step 3: Run emrfs sync to update metadata with contents of  the Amazon S3 bucket.

Step 4: Submit a Pig job on Amazon EMR cluster as step.

Step 5: Reuse the same Amazon EMR cluster to launch a Hive job for generating reports.

Step 6: Clean EMRFS metadata that is older than two days (emrfs delete).

# Step 1: Determine Whether Log Files are Available in an Amazon S3 Bucket

The first step is to check whether files are available in the s3://emr-worksop/data/input/{YYYY-MM-DD} location. This is passed as a variable myInputData. A variable can be referenced in other places by #{variable\_name}.

In our example, we use the scheduledStartTime variable provided by AWS Data Pipeline and do a date minus operation to get the data of yesterday’s logs.

#{format(minusDays(@scheduledStartTime,1),’YYYY-MM-dd’)} references to a day before the scheduled date of workflow.

This is done using the Precondition feature of AWS Data Pipeline.

{

"id": "myPreconditionId",

"name": "DefaultPrecondition1",

"maximumRetries": "10",

"role": "DataPipelineDefaultRole",

"s3Prefix": "#{myInputData}/#{format(minusDays(@scheduledStartTime,1),'YYYY-MM-dd')}",

"type": "S3PrefixNotEmpty"

}

# Step 2: Creating an Amazon EMR Cluster with EMRFS

The data pipeline JSON code below launches a 2-node Amazon EMR cluster with EMRFS with one master and 1 core node. [terminateAfter](http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-ec2resource.html" \t "_blank) terminates the cluster after one hour.

{

"terminateAfter": "1 Hour",

"id": "myEmrClusterId",

"amiVersion": "3.2.1",

"schedule": {

"ref": "mySchedule"

},

"keyPair": "#{myKeyPair}",

"masterInstanceType": "m3.xlarge",

"bootstrapAction": "s3://elasticmapreduce/bootstrap-actions/configure-hadoop,-e,fs.s3.consistent=true,-e, fs.s3.consistent.metadata.tableName=EmrFSMetadata, -e, fs.s3.consistent.retryCount=5, -e, fs.s3.consistent.retryPeriodSeconds=10, -e, fs.s3.enableServerSideEncryption=true",

"coreInstanceType": "m3.xlarge",

"enableDebugging": "true",

"name": "DefaultEmrCluster1",

"coreInstanceCount": "2",

"type": "EmrCluster"

}

# Step 3: Run emrfs sync to Update Metadata with the Contents of the Amazon S3 Bucket

For this, we use the preStepCommand feature. Amazon EMR’s documentation provides more information about this feature.

“preStepCommand”: “/home/hadoop/bin/emrfs sync #{myInputData}”,

# Step 4: Launch a Pig Job on the Amazon EMR Cluster

The Pig job is submitted to the Amazon EMR cluster as a Step. The input to this job are log files present in an Amazon S3 bucket referenced by variable myInputData. After processing the log file, the output is written to another Amazon S3 location referenced by the variable myTransformedData. A sub-folder is created under this bucket for each scheduled date.

The Pig script is referenced by variable myPigScript.

"step": [

"s3://us-east-1.elasticmapreduce/libs/script-runner/script-runner.jar,s3://us-east-1.elasticmapreduce/libs/pig/pig-script,--base-path,s3://us-east-1.elasticmapreduce/libs/pig/, --pig-versions,latest,--run-pig-script,--args,-f,#{myPigScript},-p,INPUT=#{myInputData}/#{format(minusDays(@scheduledStartTime,1),'YYYY-MM-dd')},-p,OUTPUT=#{myTransformedData}/#{format(minusDays(@scheduledStartTime,1),'YYYY-MM-dd')}"

]

# Step 5: Reuse the Amazon EMR Cluster to Launch a Hive job for Generating Reports

After the Pig job has cleaned the data, we kick off the Hive job to generate reports on same Amazon EMR cluster. Let’s take a closer look at this Hive script ([download the script](https://emr-workshop.s3.amazonaws.com/scripts/hive_sample.q" \t "_blank) to check it out). The script creates two tables: one from the cleaned up log reports and one from the subset of data about Google and Bing search terms (both were outputs from the Pig step).

The Hive job is submitted to the Amazon EMR cluster as a step using the command below. The myHiveScript variable is defined as a expression, the value of which is substituted AWS Data Pipeline at run time. The AWS Data Pipeline documentation provides details about the expressions.

"step": "s3://us-east-1.elasticmapreduce/libs/script-runner/script-runner.jar,s3://us-east-1.elasticmapreduce/libs/hive/hive-script,--base-path,s3://us-east-1.elasticmapreduce/libs/hive/,--hive-versions,latest,--run-hive-script,--args,-f,#{myHiveScript},-d,INPUT=#{myTransformedData}/#{format(minusDays(@scheduledStartTime,1),'YYYY-MM-dd')},-d,OUTPUT=#{myOutputData},-d,DATE=#{format(minusDays(@scheduledStartTime,1),'YYYY-MM-dd')}"

# Step 6: Clean EMRFS Metadata Older than Two Days (emrfs delete)

We do this post processing after both the jobs have completed on cluster. Amazon EMR documentation provides more information about EMRFS delete. For this we use the postStepCommand feature of AWS Data Pipeline which runs the command after successfully completing the step.

      “postStepCommand”: “/home/hadoop/bin/emrfs delete –time 2 -u days”

## Defining a Schedule for the Workflow

The Schedule object of the pipeline can be used to define the schedule on which the pipeline should execute. You can define a schedule by specifying a startDateTime, an endDataTime and the interval at which the pipeline should run, such as once a day.

The endDateTime field is optional. If the endDateTime is not specified, the pipeline will run until it is shut down. [AWS Data Pipeline documentation](http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-object-schedule.html" \t "_blank) provides details about pipeline scheduling.

For our ETL example we define a schedule to run the pipeline for a day in the past.

{

"id": "ScheduleId\_E8jhD",

"startDateTime": "2014-09-22T00:00:00",

"name": "MyEMRJobSchedule",

"type": "Schedule",

"period": "1 Day",

"endDateTime": "2014-09-23T00:00:00"

}

Now that we’ve defined the required objects in our JSON pipeline definition, we can create and activate our pipeline.

**Creating and Activating the Pipeline**

You can download the sample [myWorkflow.json](https://github.com/awslabs/aws-big-data-blog/blob/master/aws-blog-datapipeline-etl-processing/myWorkflow.json" \t "_blank) used in this example and create a pipeline using the AWS Data Pipeline CLI. This pipeline will run once for a day in the past. Before you create the pipeline, replace **<<MY\_BUCKET>>** and **<<MY\_KEYPAIR>>** values with your Amazon S3 bucket and keypair.

Creating the pipeline is simple using [create-pipeline](http://docs.aws.amazon.com/cli/latest/reference/datapipeline/create-pipeline.html" \t "_blank).  The output of this command will be pipeline id.

$ aws datapipeline create-pipeline --name myETLWorkflow --unique-id myETLWorkflow

{

"pipelineId": "df-XXXXXXXXXXX "

}

Next we [put the pipeline definition](http://docs.aws.amazon.com/cli/latest/reference/datapipeline/put-pipeline-definition.html" \t "_blank). Any errors/warnings in your JSON appear on the screen. Creating the pipeline does not schedule it for execution. You must [activate](http://docs.aws.amazon.com/cli/latest/reference/datapipeline/activate-pipeline.html" \t "_blank) the pipeline before it will run on the schedule.

$ aws datapipeline put-pipeline-definition --pipeline-id df-XXXXXXXXXXX --pipeline-definition file://myWorkflow.json

$ aws datapipeline activate-pipeline --pipeline-id df- XXXXXXXXXXX

Observe that the pipeline should have been launched successfully. You need to wait for about 20-30 minutes before you will be able to see data in your output bucket.

Congratulations! You have completed the lab.