EMR and Glue- Learning document

Let's begin with EMR first, to start EMR installation go to the EMR console and click go to advanced options.

Step 1- Software and Steps

1. Software Configuration – Choose the EMR release and the product to install.
2. Multiple Master Nodes – On 24X 7 clusters, mark this checkbox because if you lose the master node – you lose the cluster and lose the data on HDFS. You may end up in unexpected results of your data pipeline. On transient clusters it is less important.
3. AWS Glue Data Catalog – We can choose if we want to use Glue data Catalog. AWS Glue will allow us to query external database tables. For example, from Athena.
4. Edit software setting – In order to change the installation configuration.
5. Steps – You can configure automation steps.

Step 2- Hardware

A. Cluster Composition

This section allows to modify your cluster instance:

a. Uniform instance groups – The more recommended option due to the configuration of executor (CPU and RAM) utilization. In this option you select the instance configuration.

b. Instance fleets -You can configure each instance independently.

B. Networking

You pick the VPC and Subnet of the cluster.

C. Cluster Nodes and Instances

i. Master Node – Choose on demand to ensure cluster stability. If you lose the master node you lose the cluster, lose the data on HDFS, and may end up in unexpected results of your data pipeline.

ii. Core Node – No need more than 2 instances, as most of the data is expected to be on AWS S3. Unfortunately, you can remove the core node completely. Coming next, the replication factor is relevant to HDFS, not to S3.

iii. Task Node- Use task node for Auto Scaling, as it has no HDFS local storage, and it is used only to add more compute power dynamically.

iv. General resources recommendations- Use EC2 R family instances. The reason is that about 50% of the cluster memory is spent on JAVA, YARN, and OS overhead. And be consistent in the instance types across the cluster- do not mix as they may affect cluster utilization.

D. Cluster scaling
Choose min/max core/task. AWS EMR will change it automatically depending on the demand usage.

E. EBS Root
Volume – Choose the hard disk for the cluster.

Step 3- General Cluster Settings
1. General Options- nothing to add.
2. Tags- create tags.
3. Additional Options.
4. Bootstrap Actions- you can configure scripts that will run while the instance is starting.

Step 4- Security
1. EC2 key pair– Choose the key to connect the cluster.
2. Permissions– Choose the role for the cluster (EMR will create new if you did not specified).
3. Security configuration – skip for now, used to setup encryption at rest and in motion.
4. EC2 security groups– Choose the security group for the Master instance and the Slave instance (EMR will create new if one does not specify).

Note: You can export your EMR cluster creation process to a AWS CLI command via pressing the "AWS CLI EXPORT" button on the console.

AWS Glue

To summarize, we will build one full ETL process: we'll create an S3 bucket, upload our raw data to the bucket, start the glue database, add a crawler that browses the data in the above S3 bucket, will create a GlueJob, which can be run on a schedule, on a trigger, or on-demand, and finally updated data back to the S3 bucket.

1. Create an IAM role to access AWS Glue + EC2 + CloudWatch + S3
You need an appropriate role to access the different services you are going to be using in this process. An IAM role is similar to an IAM user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. When you get a role, it provides you with temporary security credentials for your role session.

- Open Amazon IAM console
- Click on Roles → Create Role.
- Choose Glue service from "Choose the service that will use this role"
- Choose Glue from the "Select your use case" section

- Select “AWSGlueServiceRole” from the Attach Permissions Policies section.
- Click on Next: Tags. Leave the Add tags section blank. Create role.
- Your role now gets full access to AWS Glue and other services

2. Upload source CSV files to Amazon S3

- On the Amazon S3 console, click on the Create a bucket where you can store files and folders.
- Enter a bucket name, select a Region and click on Next
- The remaining configuration settings can remain empty now. Click Next to create your S3 bucket.
- Create a new folder in your bucket and upload the some CSV files

3. Start the AWS Glue Database

In order to add data to a Glue data catalog, which helps to hold the metadata and the structure of the data, we need to define a Glue database as a logical container. So, we need to initialize the glue database

4. Create and Run Glue Crawlers

As we have Glue Database ready, we need to feed our data into the model. So, what we are trying to do is this: We will create crawlers that basically scan all available data in the specified S3 bucket. The crawler identifies the most common classifiers automatically including CSV, JSON, and Parquet.

- On the left pane in the AWS Glue console, click on Crawlers -> Add Crawler
- Click the blue **Add crawler** button.
- Make a crawler a **name,** and leave it as it is for **“Specify crawler type.**
- In **Data Store,** choose S3 and select the bucket you created. Drill down to select the read folder
- In **IAM role,** choose the role you created above
- Leave the Frequency on “Run on Demand” now. You can always change to schedule your crawler on your interest later.
- In **Output,** specify a Glue database you created above (sampledb)
- Then, a Glue Crawler that reads all the files in the specified S3 bucket is generated
- Click the checkbox and Run the crawler by clicking Run Crawler
- Once it’s done, you should see its status as ‘Stopping’. And ‘Last Runtime’ and ‘Tables Added’ are specified.
- Then, Databases → Tables on the left pane let you verify if the tables were created automatically by the crawler.

5. Define Glue Jobs

With the final tables in place, we know create Glue Jobs, which can be run on a schedule, on a trigger, or on-demand. The interesting thing about creating Glue jobs is that it can actually be an

almost entirely GUI-based activity, with just a few button clicks needed to auto-generate the necessary python code. However, I will make a few edits in order to synthesize multiple source files and perform in-place data quality validation. By default, Glue uses DynamicFrame objects to contain relational data tables, and they can easily be converted back and forth to PySpark DataFrames for custom transforms. Note that at this step, you have an option to spin up another database (i.e. AWS RedShift) to hold final data tables if the size of the data from the crawler gets big. For now, we will put the processed data tables directly back to another S3 bucket

- In the left pane, click on Jobs, then click on Add Job
- Give a name and then select IAM role previously created for AWS Glue
- Select Spark for the Type and select Spark 2.4, Python 3 for Glue Version
- You can edit the number of DPU (Data processing unit) values in the Maximum capacity field of Security configuration, script libraries, and job parameters (optional).
- The remaining configuration is optional
- Choose a data source table from Choose a data source section. You can choose only a single data source.
- Add a JDBC connection to AWS Redshift. We need to choose a place where we would want to store the final processed data. You can choose your existing database if you have one. Or you can re-write back to the S3 cluster. For this tutorial, we are going ahead with the default mapping. The business logic can also later modify this.
- Open the Python script by selecting the recently created job name. Click on Action -> Edit Script.
- The left pane shows a visual representation of the ETL process. The right-hand pane shows the script code and just below that you can see the logs of the running Job.
- Save and execute the Job by clicking on Run Job.
- We get history after running the script and get the final data populated in S3 (or data ready for SQL if Redshift is selected as the final data storage)