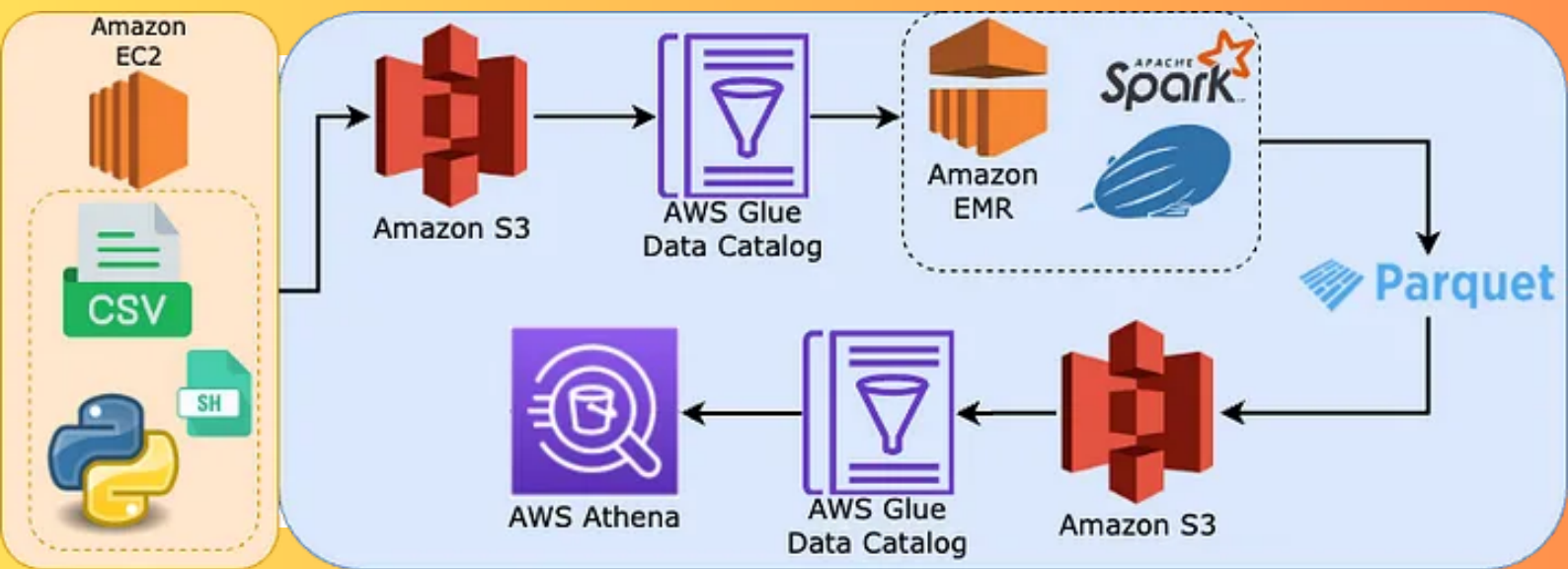


AWS Cloud Data Engineering End-to-End Project – EMR, EC2, Glue, S3, Spark, Zeppelin



Tech Stack

- AWS EMR
- AWS S3
- AWS Glue
- AWS EC2
- AWS Athena
- Apache Spark
- Zeppelin Notebook
- SQL
- Shell Scripting
- Parquet file format

Overview

In this project, we are going to upload a CSV file into an S3 bucket either with automated Python/Shell scripts or manually. We are going to create a corresponding Glue Data Catalog table. The main part will be establishing a new EMR cluster. After creating it, we are going to run a Spark job with Zeppelin Notebook and modify the data. After modifications, we are going to write the data to S3 as a parquet file. A Glue Data Catalog table will also be created. We will monitor the data using AWS Athena and S3 Select in the end.

S3 Bucket

In this project, we will need 3 buckets: source, target, and log. We will upload the source data into the source bucket. The source bucket's name will be a unique name that describes the process (dirty-transactions-from-csv-to-parquet for this project). We will upload our initial CSV file into this bucket with the key

dirty_transactions/dirty_transactions.csv. If we want to upload the data automatically from inside the EC2 instance, all details can be found in the below article.

How to Automate Data Upload to Amazon S3

If we want to create the S3 bucket manually, we can do it via the S3 dashboard directly or upload the CSV file using AWS CLI. As the target bucket, we will be using a bucket named aws-glue-emr-from-csv-to-parquet. This part is important since the bucket name should include “aws-glue”, if not we should define some other permissions. In the end, we are going to create a bucket s3-from-csv-to-parquet-aws-emr-logs which we will choose as the bucket where we will keep the logs of EMR later.

General configuration

Bucket name

s3-from-csv-to-parquet-aws-emr-logs

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

AWS Region

EU (Frankfurt) eu-central-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Glue Data Catalog

In this part, the first thing is creating a new database. We can create the database from AWS Glue -> Databases -> Add database. We can name it dirty-transactions-from-csv-to-parquet, the same as the bucket name.

The second part is creating a new table in this database. We are going to use this table to keep the metadata of the object we recently put into the S3 bucket. We can define the schema manually. Our table name will be `dirty_transactions` (the same as the S3 prefix).

Set table properties

Table details

Name

If you plan to access the table from Amazon Athena, then the name should be under 256 characters and contain only lowercase letters (a-z), numbers (0-9), and underscore (_). For more information, see [Athena names](#).

Database

▼

Description - optional

Descriptions can be up to 2048 characters long.

We are going to choose the source data as S3 and browse the location of our newly created object. Be careful that we should choose the directory instead of the file itself at this point.

For our data `dirty_transactions`, we should choose the data type as CSV and define the schema manually.

Schema (1/9)

[Edit schema as JSON](#)

View and manage the table schema.

<input type="checkbox"/>	#	Column name	Data type	Partition
<input type="checkbox"/>	1	store_id	string	-
<input type="checkbox"/>	2	store_location	string	-
<input type="checkbox"/>	3	product_category	string	-
<input type="checkbox"/>	4	product_id	string	-
<input type="checkbox"/>	5	mrp	string	-
<input type="checkbox"/>	6	cp	string	-
<input type="checkbox"/>	7	discount	string	-
<input type="checkbox"/>	8	sp	string	-
<input checked="" type="checkbox"/>	9	date	date	-

After all, we can create our Glue Data Catalog table.

Security Group

To be able to define while creating the EMR cluster, we should create a security group named `aws_emr_security_group`. We can define all the necessary inbound rules after creation. The necessary cluster-related inbound rules will be created after the creation of the cluster.

Create security group Info

A security group acts as a virtual firewall for your instance to control inbound and outbound traffic. To cr

Basic details

Security group name Info

Name cannot be edited after creation.

Description Info

VPC Info

Amazon EMR

We are going to create an AWS EMR cluster and use Spark with Zeppelin Notebook. We will follow the below steps to create the cluster. Our cluster's name will be csv-from-s3-to-parquet-cluster.

We are going to check Use for Spark table metadata so that we are going to use the Glue Data Catalog table along with EMR. We can choose the latest release for Amazon EMR.

Create cluster [Info](#)

Name and applications [Info](#)

Name

csv-from-s3-to-parquet-cluster

Amazon EMR release [Info](#)

A release contains a set of applications which can be installed on your cluster.

emr-6.12.0

Application bundle

Spark 	Core Hadoop 	Flink 	HBase 	Presto 	Trino 	Custom 
--	--	--	--	---	--	---

Applications included in bundle

Spark 3.4.0 on Hadoop 3.3.3 YARN with and Zeppelin 0.10.1

AWS Glue Data Catalog settings

Use the AWS Glue Data Catalog to provide an external metastore for your application.

☒ Use for Spark table metadata

Operating system options [Info](#)

☒ Amazon Linux release

☐ Custom Amazon Machine Image (AMI)

☒ Automatically apply latest Amazon Linux updates

When it comes to the instances, we will only need 1 primary and 1 core instance. We can remove the task instance since we don't need it. The instance type depends on our workload and I chose m5.xlarge for this project. If suitable, you may choose cheaper instances as well.

Instance groups

Primary

Choose EC2 instance type

m5.xlarge

4 vCore 16 GiB memory EBS only storage
On-Demand price: \$0.230 per instance/hour
Lowest Spot price: \$0.077 (eu-central-1c)

Actions ▼

☐ Use multiple primary nodes

To improve cluster availability, use 3 primary nodes with the same configuration and bootstrap actions. You can not use multiple primary nodes with instance fleets.

► Node configuration - optional

Core

Remove instance group

Choose EC2 instance type

m5.xlarge

4 vCore 16 GiB memory EBS only storage
On-Demand price: \$0.230 per instance/hour
Lowest Spot price: \$0.077 (eu-central-1c)

Actions ▼

► Node configuration - optional

Add task instance group

You can add up to 48 more task instance groups.

We will choose the cluster instance size as 1 for this specific project.

Cluster scaling and provisioning option [Info](#)

Amazon EMR console only supports EMR-managed scaling. To create a cluster with auto-scaling, use CLI or SDK.

Choose an option

☒ Set cluster size manually

Use this option if you know your workload patterns in advance.

☐ Use EMR-managed scaling

Monitor key workload metrics so that EMR can optimize the cluster size and resource utilization.

Provisioning configuration

Set the size of your core instance group. Amazon EMR attempts to provision this capacity when you launch your cluster.

Name	Instance type	Instance(s) size	Use Spot purchasing option
Core	m5.xlarge	<input type="text" value="1"/>	<input type="checkbox"/>

We can choose the default VPC or we can also create a specific VPC for this use case. We can also choose private/public subnet. Since this is a demo project, we can choose a public subnet. But I definitely recommend choosing a private subnet for

production use. If you want to have more information about establishing a VPC, you may see the below article.

Establishing a VPC for Amazon S3, Lambda, RDS and EC2

Networking [Info](#)

Virtual private cloud (VPC) [Info](#)

[Browse](#)

[Create VPC](#) [↗](#)

Subnet [Info](#)

[Browse](#)

[Create subnet](#) [↗](#)

▶ [EC2 security groups \(firewall\)](#)

Since the instances will cost us a bit much, we better set a termination time for our clusters. I chose it to be 1 hour, but you might define it depending on your use case.

Cluster termination [Info](#)

- ☐ Manually terminate cluster
- ☐ Automatically terminate cluster after last step ends
- ☒ Automatically terminate cluster after idle time (Recommended)

Idle time

Enter the time until your cluster terminates.

Choose a time that is greater than 1 minute (00:01:00) and less than 7 days. The time is in hh:mm:ss (24-hour) format.

- ☒ **Use termination protection**
Protect your EC2 instances from accidental termination.

If this is the first time we create an EMR cluster, we better choose “Create a service role”. VPC and subnet will be determined automatically since we already defined them in the previous steps. We can choose the security group as the one we created previously (aws_emr_security_group).

Identity and Access Management (IAM) roles [Info](#)

Choose or create a service role and instance profile for the EC2 instances in your cluster.

Amazon EMR service role [Info](#)

The service role is an IAM role that Amazon EMR assumes to provision resources and perform service-level actions with other AWS services.

☐ **Choose an existing service role**
Select a default service role or a custom role with IAM policies attached so that your cluster can interact with other AWS services.

☒ **Create a service role**
Let Amazon EMR create a new service role so that you can grant and restrict access to resources in other AWS services.

Networking resources

We've already added the resources that you configured in the [Networking](#) section. Choose the VPC, subnet, and security groups that the service role can access.

Virtual Private Cloud (VPC)

Choose one or more VPCs

- [redacted] X

Subnet

Choose one or more subnets

- [redacted] X

Security group

Choose one or more security groups

aws_emr_security_group X

We can choose the key pair that is used for the EC2 instance.

Security configuration and EC2 key pair - optional [Info](#)

Security configuration

Select your cluster encryption, authentication, authorization, and instance metadata service settings.

Q Choose a security configuration



Browse

Create security configuration

Amazon EC2 key pair for SSH to the cluster [Info](#)

Q [redacted] X

Browse

Create key pair

We already created a dedicated bucket for our EMR logs. We can choose that bucket for the Cluster logs section. Please be careful that we choose directories, not files.

▼ Cluster logs - optional [Info](#)

☒ Publish cluster-specific logs to Amazon S3

Amazon S3 location



[View](#)

[Browse S3](#)

☐ Encrypt cluster-specific logs

Last but not least, we can allow access to S3 buckets, but it is not recommended for production use. We better specify the buckets we will use for our project.

EC2 instance profile for Amazon EMR

The instance profile assigns a role to every EC2 instance in a cluster. The instance profile must specify a role that can access the resources for your steps and bootstrap actions.

☐ Choose an existing instance profile

Select a default role or a custom instance profile with IAM policies attached so that your cluster can interact with your resources in Amazon S3.

☒ Create an instance profile

Let Amazon EMR create a new instance profile so that you can specify a custom set of resources for it to access in Amazon S3.

S3 bucket access [Info](#)

☐ Specific S3 buckets or prefixes in your account [Info](#)

Choose the buckets or prefixes that you want this instance profile to access.

☒ All S3 buckets in this account with read and write access

Grant the instance profile access to all buckets that have read and write access enabled in your account.

After all, we can click on Create cluster button. Once created, the cluster will be running in about 5 minutes. We can see that the status will be “Waiting” instead of “Starting”. Once created, we have to be careful about the following steps:

We should modify the IAM role for Instance profile of EC2. Once we click on the role’s name on EMR dashboard, we will be redirected to the IAM page. We should add related S3 and Glue policies to the role so that we will be able to access S3 and Glue.

We should also modify the security group’s inbound rules according to our use case. If we want to access the cluster from a specific service (Zeppelin notebook for example), we should define its security group ID as the inbound rule.

Spark Job

--	--

glue_etl_job_data_catalog_s3/emr_zeppelin/emr_zeppelin_notebook.ipynb at main · ...

Glue ETL job or EMR Spark that gets from data catalog, modifies and uploads to S3 and Data Catalog ...

We will be using Zeppelin Notebook to run our Spark job. We should modify the inbound rule for Zeppelin Notebook's ID (For demo purposes, we can allow all IPs but this will make our cluster so vulnerable). After all, we will access the Zeppelin notebook in the Applications section. Now, we are going to go through the Spark job.

Application UIs on the primary node

These require SSH tunneling to be enabled.

Application

HDFS Name Node

Resource Manager

Spark History Server

Zeppelin

First of all, we are going to choose Spark for the use purpose. Once we run the below code, our Spark session will be created.

```
%pyspark
spark.version
```

Some libraries and packages will be required for the UDFs we are going to create. We will import all of them.

```
%pyspark from pyspark.sql.functions import regexp_replace, regexp_extract from
pyspark.sql.types import StringType, FloatType, IntegerType
```

We are going to get the CSV data from the S3 bucket and will see its result.

```
%pyspark
df = spark.read.format("csv")\
.option("header", True)\
.option("inferSchema", True)\
.option("sep", ",")\
.load("s3://dirty-transactions-from-csv-to-parquet/dirty_transactions/dirty_tra

df.show(5)
```

We may also retrieve the data from the Glue Data Catalog table we already created.

```
%pyspark
df_glue_table = spark.table("dirty-transactions-from-csv-to-parquet.dirty_trans

df_glue_table.show(5)
```

Since we defined the column names in lowercase for the Glue table, choosing `df_glue_table` as our main data frame will be a wise choice. Then, we are going to define and register UDFs for this specific use case. This part might differ depending on the use case of the project.

```
%pyspark
def extract_city_name(string):
    cleaned_string = regexp_replace(string, r'^\w\s', '')
    city_name = cleaned_string.strip()
    return city_name

def extract_only_numbers(string):
    numbers = regexp_extract(string, r'\d+', 0)
```

```

        return "".join(numbers)

def extract_floats_without_sign(string):
    string_without_dollar = regexp_replace(string, r'\$', '')
    return float(string_without_dollar)

spark.udf.register("extract_city_name", extract_city_name, StringType())
spark.udf.register("extract_only_numbers", extract_only_numbers, IntegerType())
spark.udf.register("extract_floats_without_sign", extract_floats_without_sign,

```

Once we define the UDFs, we can apply them to our main data frame and create the final data frame.

```

%pyspark
# choose df_glue_table since column names are lowercase
df_final = df_glue_table.selectExpr(
    "store_id",
    "extract_city_name(store_location) as store_location",
    "product_category",
    "extract_only_numbers(product_id) as product_id",
    "extract_floats_without_sign(mrp) as mrp",
    "extract_floats_without_sign(cp) as cp",
    "extract_floats_without_sign(discount) as discount",
    "extract_floats_without_sign(sp) as sp",
    "date"
)

df_final.show(5)

```

To conclude, we are going to write our final clean data frame in the parquet format both to the S3 bucket and target Glue table as below. We can check both after the Spark job's run finishes.

```

%pyspark
df_final.write\
    .saveAsTable('dirty-transactions-from-csv-to-parquet.clean_transactions', format=
        path='s3://aws-glue-emr-from-csv-to-parquet/clean_transactions_parq

```

Monitor the Data Using S3 Select and AWS Athena


Once we run the above Spark job using Zeppelin Notebook, we are going to see our resulting data both in the S3 bucket as a parquet file and as a Glue Data Catalog table.


We can run the below query in AWS Athena to see the most recent product_category and discount per store_location.


```
WITH dirty_transactions_rn AS (  
  SELECT  
    store_location,  
    product_category,  
    discount,  
    row_number() OVER(PARTITION BY store_location ORDER BY date DESC) as rn  
  FROM 'dirty-transactions-from-csv-to-parquet.clean_transactions'  
)  
  
SELECT  
  store_location,  
  product_category,  
  discount,  
  FROM dirty_transactions_rn  
  WHERE rn = 1
```



Objects (2)


Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list





 Copy S3 URI

 Copy URL

 Download

 Open

 Find objects by prefix

<input type="checkbox"/>	Name ▲	Type ▼	
<input type="checkbox"/>		-	A (
<input type="checkbox"/>	 c000.snappy.parquet	parquet	A (

We can also see the resulting data using S3 Select for the target parquet file in the S3 bucket. The below query will give us the below result.

Query results

Query results are not available after you choose **Close** or navigate away. Choose **Download results** to download a copy of the following query results.

Status

 Successfully returned 5 records in 841 ms

Bytes returned: 330 B

Raw

Formatted

```
YR1475,New York,Electronics,90175476,38.0,25.84,3.42,34.58,2019-11-26
0J2470,Denver,Kitchen,75536193,96.0,70.08,4.8,91.2,2019-11-26
XV5488,Washington,Kitchen,58647460,68.0,51.0,2.04,65.96,2019-11-26
0J9422,Denver,Fashion,60402526,74.0,51.8,6.66,67.34,2019-11-26
XV7687,Washington,Kitchen,68167114,78.0,56.16,5.46,72.54,2019-11-26
```

You may also see the resulting parquet file [here](#).

EMR

AWS

Spark

Parquet

Athena

Thanks for reading!

**Follow for more content
like this**



Ganesh R

Azure Data Engineer



<https://www.linkedin.com/in/rganesh0203/>

Like



Comment



Share

