AWS Project for Batch Processing with PySpark on AWS EMR

CookBook

Table of Content:

Introduction

In this training, we will perform batch processing on Wikipedia data. Batch processing can be a useful technique for processing large volumes of data from Wikipedia. Wikipedia provides regular data dumps that contain the entire content of the site, which can be analyzed using batch processing techniques. For example, text analysis can be performed on Wikipedia articles to identify patterns in the content, such as sentiment analysis or topic modeling. Statistical analysis can also be performed on Wikipedia data to identify trends and patterns, such as analyzing the frequency of words or phrases across multiple articles or languages. Additionally, batch processing can be used to analyze the links between Wikipedia articles and train machine learning models using Wikipedia data. By using batch processing techniques on Wikipedia data, researchers and data scientists can efficiently analyze large amounts of data to gain insights and improve their understanding of the content and structure of the encyclopedia.

Processing refers to the manipulation of data in order to produce a desired output. It involves a series of operations performed on data, such as data input, data validation, data transformation, data storage, data retrieval, and data output.

Processing can be done manually, by humans, or automatically, by computers or other electronic devices. The process may involve a single operation or a series of operations that are performed in a specific order to achieve a specific outcome. The outcome may be a report, a database, a chart, a graphic, or any other type of information that is useful for decision-making.

Processing can be classified into various types based on different criteria, such as the nature of the data, the volume of data, the speed of processing, and the complexity of processing. Some common types of processing include batch processing, real-time processing, stream processing, parallel processing, interactive processing, and distributed processing.

Batch processing is a method of processing large volumes of data in which multiple transactions are grouped together into a batch and processed at the same time. In batch processing, data is collected, processed, and outputted in batches, rather than being processed continuously or in real-time.

The process of batch processing typically involves several stages, including:

1. **Data collection:** Data is collected from various sources and stored in a temporary repository, such as a file or a database.
2. **Data validation:** The collected data is validated to ensure that it meets the required format and standards. Data that does not meet the required standards is rejected and sent back for correction.
3. **Data transformation:** The collected data is transformed into a format that can be processed by the batch processing system. This may involve converting data from one format to another, filtering and sorting data, and performing other operations to prepare the data for processing.
4. **Processing:** The transformed data is processed by the batch processing system using one or more processing algorithms. The processing may involve performing calculations, updating records, generating reports, and other tasks.

5. **Output:** The results of the processing are outputted to a storage location, such as a database, a file, or a report. The output may also be sent to other systems or applications for further processing or analysis.

Batch processing is commonly used in situations where large volumes of data need to be processed at regular intervals, such as payroll processing, financial transactions, and billing systems. Batch processing allows for more efficient processing and can help to reduce errors by processing data in a controlled and consistent manner. It can be done manually or through automated software systems, and can be customized to meet specific processing requirements.

What is the need of Batch processing?

Batch processing is needed in situations where large amounts of data need to be processed efficiently and accurately. Here are some specific reasons why batch processing is used:

- **Efficient use of resources:** Batch processing allows for the efficient use of computing resources by processing multiple transactions in a single batch. This can help to reduce the processing time and lower the cost of processing.
- **Consistent processing:** Batch processing ensures that data is processed in a consistent manner, with the same rules and algorithms applied to each transaction in the batch. This can help to reduce errors and ensure accuracy in processing.
- **Scheduling and automation:** Batch processing can be scheduled to run at specific times or intervals, allowing for automated processing without the need for manual intervention. This can save time and increase efficiency.
- **Error handling:** Batch processing can be designed to handle errors and exceptions in a systematic way, with the ability to stop processing and alert operators when errors occur. This can help to reduce the risk of data loss or corruption.
- **Historical reporting:** Batch processing can be used to generate historical reports that provide insight into past transactions and processing performance. This can be useful for monitoring and analyzing trends in data processing.

Overall, batch processing is a valuable tool for processing large volumes of data efficiently and accurately, while minimizing the risk of errors and reducing processing costs. It is commonly used in industries such as finance, healthcare, manufacturing, and retail.

Batch processing is used in many different industries and applications. Here are some specific use cases where batch processing is commonly used:

- **Payroll processing:** Payroll processing involves calculating salaries, wages, and other benefits for employees. This typically involves processing a large volume of data, including time cards, tax deductions, and other payroll-related information. Batch processing is commonly used to efficiently process this data in a timely and accurate manner.
- **Financial transactions:** Batch processing is widely used in the financial industry for processing transactions such as credit card transactions, fund transfers, and securities trades. These transactions involve large volumes of data and require high levels of accuracy and security.
- **Billing and invoicing:** Batch processing is often used to generate bills and invoices for customers. This involves collecting and processing data such as customer orders, product pricing, and shipping information.

- **Inventory management:** Batch processing is used in inventory management systems to track and manage inventory levels. This involves processing data such as sales orders, purchase orders, and inventory counts to determine stock levels and order replenishments.
- **Data warehousing:** Batch processing is commonly used in data warehousing to extract, transform, and load (ETL) data from multiple sources into a central repository for analysis and reporting. This involves processing large volumes of data from multiple sources and transforming it into a format that can be used for analysis.

Overall, batch processing is a valuable tool for processing large volumes of data in a wide range of applications, allowing for efficient processing, accurate results, and improved productivity.

Services used in the project

AWS S3

Amazon S3 (Simple Storage Service) is a cloud-based object storage service provided by Amazon Web Services (AWS). It allows businesses and individuals to store and retrieve data from anywhere on the internet. Amazon S3 is designed to be highly scalable, reliable, and cost-effective, making it a popular choice for storing a wide variety of data, including images, videos, documents, and backups.

Some of the key features of Amazon S3 include:

1. **Scalability:** Amazon S3 is highly scalable and can handle storage needs ranging from a few gigabytes to multiple petabytes.
2. **Security:** Amazon S3 provides several security features to protect data, including encryption in transit and at rest, access controls, and multi-factor authentication.
3. **Durability and availability:** Amazon S3 store data across multiple data centers, providing high durability and availability. It also provides a service-level agreement (SLA) for availability of 99.999999999% (11 nines).
4. **Integration:** Amazon S3 integrates with a wide range of AWS services, including compute, database, analytics, and machine learning services.
5. **Cost-effective:** Amazon S3 offers a pay-as-you-go pricing model, allowing customers to pay only for the storage and data transfer they use.

Here are some disadvantages of using Amazon S3:

1. **Complexity:** The various features and configuration options of Amazon S3 can make it difficult to set up and manage for users without a technical background.
2. **Cost:** Although Amazon S3 can be cost-effective for smaller storage needs, costs can quickly add up for large volumes of data and high usage.
3. **Data transfer fees:** There are additional fees for transferring data in and out of Amazon S3, which can increase costs for businesses with high data transfer needs.
4. **Performance:** While Amazon S3 offers high durability and availability, its performance can be slower compared to other storage options for certain use cases.
5. **Vendor lock-in:** Moving data out of Amazon S3 to another storage solution can be challenging, potentially creating a vendor lock-in situation.

Overall, Amazon S3 is a powerful and flexible storage solution that can be used for a wide range of use cases, from backup and disaster recovery to big data analytics and web hosting.

Amazon EMR

Amazon EMR (Elastic MapReduce) is a fully-managed big data processing service provided by Amazon Web Services (AWS). It simplifies the processing of large amounts of data using popular big data frameworks such as Apache Hadoop, Apache Spark, and Presto. Amazon EMR can automatically provision, scale, and manage clusters of virtual servers running Hadoop, Spark, and other big data frameworks. It provides an easy-to-use web interface and command-line tools to create and manage EMR clusters, as well as APIs to automate cluster creation and management.

Here are some key features of Amazon EMR:

1. **Fully managed:** Amazon EMR is a fully-managed service that takes care of the underlying infrastructure, software installation, configuration, and maintenance of the big data frameworks and tools.
2. **Scalable:** Amazon EMR can automatically scale the processing capacity up or down based on the size and complexity of the data processing workloads.
3. **Flexible:** Amazon EMR supports a wide range of big data frameworks and tools, such as Apache Hadoop, Apache Spark, Apache Hive, Apache Pig, and Presto.
4. **Cost-effective:** Amazon EMR uses a pay-as-you-go pricing model, allowing you to pay only for the processing resources you use. You can also use spot instances to reduce costs.
5. **Integration:** Amazon EMR integrates with other AWS services such as Amazon S3, Amazon DynamoDB, and Amazon Redshift, making it easy to move data in and out of EMR.

Here are some disadvantages of using Amazon EMR:

1. **Complexity:** Setting up and configuring EMR clusters can be complex, especially for users who are new to big data processing. It requires knowledge of big data frameworks and the Amazon Web Services environment.
2. **Cost:** While EMR uses a pay-as-you-go pricing model, costs can add up quickly, especially if you are processing large volumes of data or using high-performance instances.
3. **Data transfer costs:** If you are moving data between AWS services or from outside AWS into EMR, there may be additional data transfer costs that you need to consider.
4. **Limited control:** As a managed service, EMR may not give you full control over the underlying infrastructure or software. This may limit your ability to customize or optimize your clusters for specific workloads.

5. **Limited support for certain tools:** While EMR supports many big data frameworks and tools, there may be certain tools or features that are not fully supported or integrated.
6. **Reliance on AWS:** If you are using EMR, you are relying on AWS for your big data processing needs. This may limit your ability to use other cloud providers or run big data workloads on-premises.

Some of the use cases for Amazon EMR include:

1. **Data processing and analysis:** Amazon EMR can be used for processing and analyzing large volumes of data, such as log files, social media data, and sensor data.
2. **ETL (Extract, Transform, Load):** Amazon EMR can be used for ETL workloads, such as transforming and loading data from various sources into a data warehouse.
3. **Machine learning:** Amazon EMR can be used for building and training machine learning models using frameworks like Apache Spark MLlib and TensorFlow.
4. **Real-time processing:** Amazon EMR can be used for real-time processing of streaming data using tools like Apache Kafka and Apache Flink.

In summary, Amazon EMR is a powerful, flexible, and cost-effective big data processing service that can help businesses process and analyze large amounts of data efficiently. Its ease of use and integration with other AWS services make it a popular choice for data engineers, data analysts, and data scientists.

Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL queries. It is a serverless, fully managed service that doesn't require any infrastructure setup or maintenance, making it easy to get started with querying data immediately.

Here are some key features of Amazon Athena:

1. **Serverless:** Amazon Athena is a serverless service, which means that it automatically scales up or down based on your query workload. You don't need to provision any servers or manage any infrastructure.
2. **Fully managed:** Amazon Athena is a fully managed service, which means that Amazon takes care of all the underlying infrastructure, including backups, security, and patching.
3. **Pay-as-you-go pricing:** Amazon Athena uses a pay-as-you-go pricing model, which means that you only pay for the queries that you run.
4. **Integration:** Amazon Athena integrates seamlessly with other AWS services, including Amazon S3, AWS Glue, and Amazon Redshift.
5. **SQL-based queries:** Amazon Athena supports standard SQL queries, making it easy for data analysts and business users to get started with querying data without having to learn new query languages or tools.
6. **Schema-on-read:** Amazon Athena uses a schema-on-read approach, which means that you can store data in Amazon S3 without having to define a schema beforehand. This allows for greater flexibility and agility when working with data.

Here are some disadvantages of using Amazon Athena:

1. **Performance:** Amazon Athena performance can be affected by the size and complexity of the data you are querying, as well as the complexity of the SQL queries you are running.
2. **Limited functionality:** Amazon Athena is designed for ad hoc querying and exploration of data in Amazon S3 using standard SQL. While it can be used for more complex analytics and data processing tasks, it has limited functionality compared to other big data tools and platforms.
3. **Cost:** While Amazon Athena uses a pay-as-you-go pricing model, costs can add up quickly, especially if you are querying large volumes of data or running complex queries.
4. **Integration:** Amazon Athena is designed to work with data stored in Amazon S3, which may limit its usability if you need to work with data stored in other locations or formats.
5. **Data formats:** Amazon Athena works best with data stored in columnar formats like Apache Parquet or Apache ORC. If your data is stored in other formats, you may need to convert it to a supported format before querying it with Athena.

6. **Data availability:** Amazon Athena requires that your data be stored in Amazon S3 and be available at the time of querying. If your data is not immediately available, you may need to wait for it to be processed by other AWS services like AWS Glue.

Some of the use cases for Amazon Athena include:

1. **Ad hoc querying:** Amazon Athena is ideal for ad hoc querying and exploration of large datasets in Amazon S3.
2. **Business intelligence:** Amazon Athena can be used for business intelligence and reporting, allowing business users to easily access and analyze data without requiring IT support.
3. **Log analysis:** Amazon Athena can be used for analyzing log data stored in Amazon S3, allowing you to gain insights into application and system performance.
4. **Machine learning:** Amazon Athena can be used for preparing and transforming data for machine learning models, allowing you to quickly iterate and refine your models.

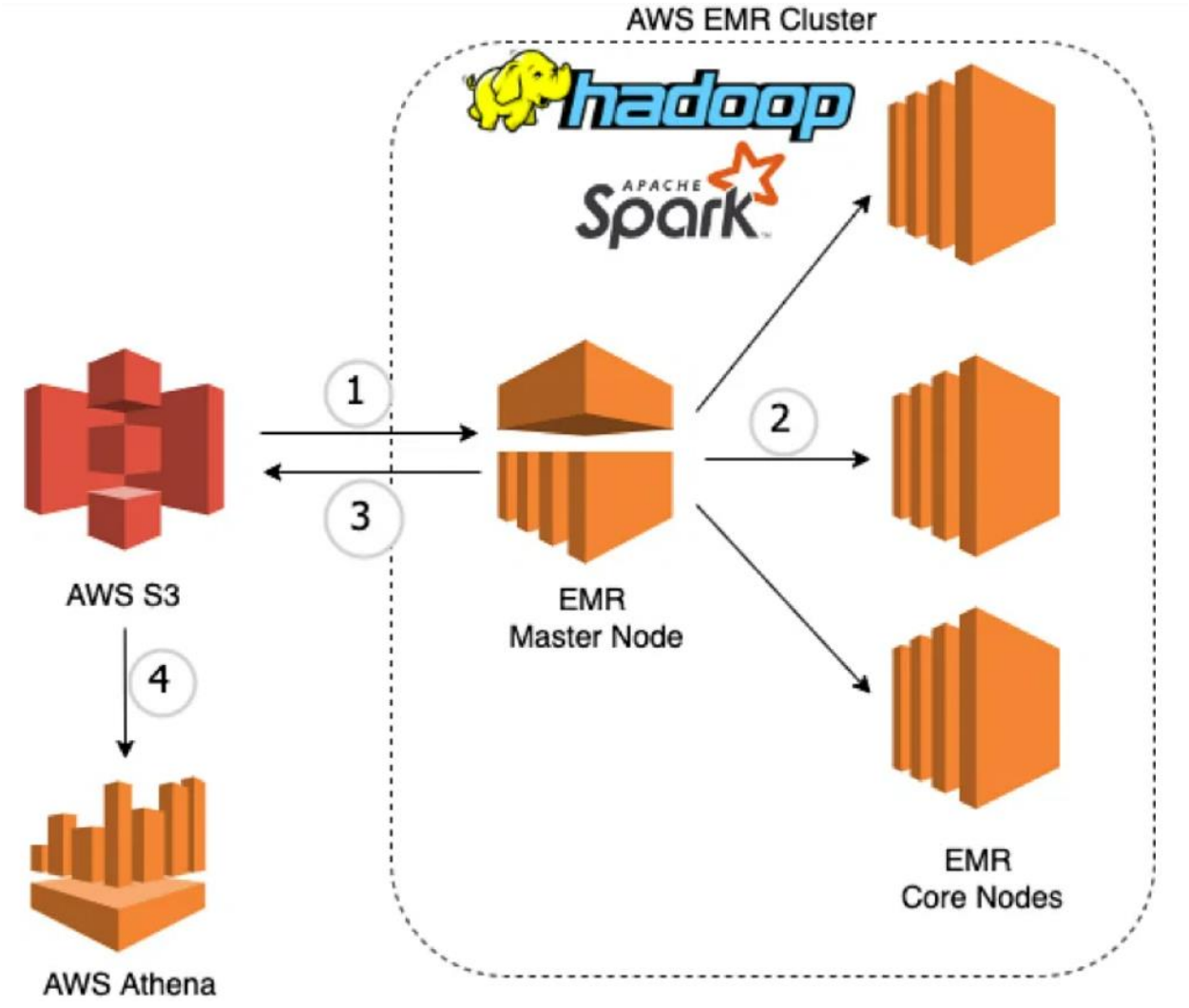
In summary, Amazon Athena is a powerful and easy-to-use query service that makes it easy to analyze data stored in Amazon S3 using standard SQL queries. Its serverless and fully managed architecture, pay-as-you-go pricing model, and integration with other AWS services make it a popular choice for data analysts, business users, and data scientists.

ProjectPro

Use-case depicted in this project

In this project, we will perform batch processing on Wikipedia data using AWS services. We will use AWS S3 for storage, Amazon EMR for processing data, and Athena for querying the processed results. Batch processing can be a useful technique for processing large volumes of data from Wikipedia. Wikipedia provides regular data dumps that contain the entire content of the site, which can be analyzed using batch processing techniques. By using batch processing techniques on Wikipedia data, researchers and data scientists can efficiently analyze large amounts of data to gain insights and improve their understanding of the content and structure of the encyclopedia.

The Architecture of the pipeline is as follows:



Breakdown of the Use-Case

- Setup required:
  - Create an AWS account
  - Download the dataset
  - Download the Code
  - This is an OS-independent project, as the whole pipeline is set up on AWS Cloud

- Ways to interact with the AWS services:
  - Console- Using the Web-based UI
  - CLI- Using the AWS CLI tool in Terminal/CMD
  - CDK- Using infrastructure-as-a-code, e.g., CloudFormation
  - SDK- Programmatically, e.g., 'boto3' in python
  - For this project, we will not be using the CDK(CloudFormation); instead, the rest of the three methods will be used for execution.

- Best Practices for AWS Account
  - Enable Multi-Factor Authentication(MFA)
  - Protect the account using IAM policies
  - Avoid using the root account
  - Choose the AWS region that is closest to you
  - Create a Budget and set up a notification
  - Rotate all keys and passwords periodically
  - Follow least privilege principle

Project Approach

1. AWS S3
   - Create AWS S3 bucket
   - Upload the raw data in the AWS S3 bucket folder
2. Amazon EMR
   - Create Amazon EMR cluster and log in to the EMR cluster using SSH
   - Run PySpark code on the EMR cluster. This code will fetch the data from the S3 bucket, perform filtering and aggregation on this data, and push the processed data back into the new folder of the AWS S3 bucket.
3. Amazon Athena
   - Create a table on top of the processed data stored in the AWS S3 bucket by providing the relevant schema.
   - Perform queries on the data stored in the Athena table.

AWS S3

We will upload the "wikiticker-2015-09-12-sampled.json" file in the S3 bucket. The file "wikiticker-2015-09-12-sampled.json" is a dataset containing a sample of Wikipedia edits that were made on September 12, 2015. The data is stored in JSON format, which is a lightweight data interchange format that is easy for computers to parse and read.

The dataset includes information about each edit, such as the title of the article that was edited, the username or IP address of the editor, the timestamp of the edit, and the text of the edit itself. This information can be used for various analyses and research purposes related to Wikipedia, such as studying patterns in the types of edits that are made, examining the behavior of different types of editors, or tracking the evolution of particular articles over time.

JSON file:


```
{
  "time": "2015-09-12T00:46:58.771Z",
  "channel": "#en.wikipedia",
  "cityName": null,
  "comment": "added project",
  "countryIsoCode": null,
  "countryName": null,
  "isAnonymous": false,
  "isMinor": false,
  "isNew": false,
  "isRobot": false,
  "isUnpatrolled": false,
  "metroCode": null,
  "namespace": "Talk",
  "page": "Talk:Oswald Tilghman",
  "regionIsoCode": null,
  "regionName": null,
  "user": "GELongstreet",
  "delta": 36,
  "added": 36,
  "deleted": 0
}
```


Create AWS S3 bucket

To create an S3 bucket in AWS, follow these steps:

1. Log in to your AWS Management Console.
2. Navigate to the S3 service by typing "S3" in the search box at the top of the page and selecting "S3" from the list of services.

3. Click on the "Create bucket" button in the top-left corner of the page.
4. Enter a unique name for your bucket. Note that bucket names must be globally unique across all of AWS, so you may need to try a few different names before finding one that is available.
5. Select the region where you want your bucket to be located. Choose a region that is geographically close to your users to minimize latency.
6. Choose the appropriate settings for your bucket. For example, you can choose whether to enable versioning or object-level logging.
7. Click on the "Create bucket" button at the bottom of the page to create your bucket.

Your S3 bucket will now be created, and you can start uploading objects to it.

Upload the raw data in the AWS S3 bucket folder

To upload data to an S3 bucket in AWS, you can follow these steps:

1. Log in to your AWS Management Console and navigate to the S3 service.
2. Click on the name of the bucket you want to upload data to.
3. Click on the "Upload" button in the top-left corner of the page.
4. In the "Upload" dialog box, click on the "Add files" button to select the file(s) you want to upload.
5. Choose the file(s) you want to upload and click on the "Open" button. You can also drag and drop the files into the dialog box.
6. In the next step, you can configure some settings for your upload, such as setting permissions, adding tags or metadata, and enabling server-side encryption.
7. Once you have configured the settings, click on the "Upload" button to start the upload process.
8. You will see a progress bar that shows the status of the upload. Once the upload is complete, you will see a message confirming that the file(s) have been successfully uploaded.

That's it! Your data has been uploaded to your S3 bucket in AWS. You can now access it through the S3 console or programmatically using the AWS SDK or API.

Amazon EMR

We will process the data stored in AWS S3 bucket using PySpark code running on Amazon EMR cluster. Before we create the Amazon EMR cluster, we need to create EC2 key pair.

To create an EC2 key pair, you can follow these steps:

1. Log in to the AWS Management Console and navigate to the EC2 dashboard.
2. Click on the "Key Pairs" link in the left-hand menu.
3. Click on the "Create Key Pair" button.
4. Enter a name for your key pair in the "Key pair name" field.
5. Choose the file format for your key pair. The recommended format is "PEM."
6. Click on the "Create Key Pair" button.
7. The key pair will be downloaded to your local computer. Save it in a secure location as you will need it to access your EC2 instances.

By following these steps, you can easily create an EC2 key pair that can be used to securely access your EC2 instances. The key pair is used to authenticate your SSH (Secure Shell) connections to your EC2 instances, and it's important to keep it secure and protected.

Create Amazon EMR cluster

To create an EMR cluster using the AWS Management Console, you can follow these steps:

1. Log in to the AWS Management Console and navigate to the EMR dashboard.
2. Click on the "Create cluster" button.
3. On the "Create Cluster" page, select the desired options for your cluster, including:
    - Cluster name: Enter a name for your EMR cluster.
    - Software configuration: Choose the software and version you want to use for your cluster, such as Hadoop or Spark.
    - Instance type: Select the type of EC2 instances to use for your cluster.
    - Number of instances: Choose the number of instances you want to use for your cluster.
    - Security and access: Configure security groups and SSH key pair to access the cluster.
    - Additional options: Choose any additional options you want to configure, such as debugging or logging.
4. Click on the "Create cluster" button at the bottom of the page.
5. Wait for the EMR cluster to be created. This may take a few minutes.
6. Once the EMR cluster is created, you can navigate to the "Cluster List" page to see a list of your clusters.
7. From the "Cluster List" page, you can select your cluster and view details about it, such as the status, instances, and applications running on it.
8. To access your EMR cluster, you can use SSH to connect to the master node using the SSH key pair you specified during cluster creation.

By following these steps, you can easily create an EMR cluster using the AWS Management Console. The console provides an easy-to-use interface for configuring and launching EMR clusters, and allows you to manage and monitor your clusters from a central location.

We have successfully created an EMR cluster, let's add a SSH port to the security group of the EMR cluster. To add an SSH port (port 22) to the security group of your EMR cluster, you can follow these steps:

1. Go to the Amazon EMR console.
2. Select your EMR cluster.
3. Click on the "Security and access" tab.
4. Under "Security groups", click on the link to the security group assigned to your EMR cluster.
5. In the security group details page, click on the "Edit inbound rules" button.
6. Click on the "Add rule" button and select "SSH" from the list of predefined rules.
7. Set the source to "My IP" to restrict access to your own IP address. Alternatively, you can set the source to "Anywhere" to allow access from any IP address.
8. Click on the "Save rules" button to apply the changes.

By following these steps, you can easily add the SSH port to the security group of your EMR cluster, allowing you to connect to the master node of the cluster using SSH. It's important to restrict access to the SSH port to only the IP addresses that require access, in order to improve the security of your EMR cluster.

To connect to an EMR cluster using SSH via the command prompt or terminal, you can follow these steps:

1. Open the command prompt or terminal application on your local machine.
2. Retrieve the public DNS name of the master node for your EMR cluster from the EMR console. To do this, select your cluster from the cluster list and navigate to the "Summary" tab.
3. Navigate to the directory where your SSH key pair is stored using the cd command.

   ```
   cd /path/to/your/ssh/keypair
   ```

   Replace /path/to/your/ssh/keypair with the local file path to your SSH key pair.

4. Use the following command to connect to the master node of your EMR cluster via SSH:

   ```
   ssh -i keypair.pem hadoop@<master-node-public-dns>
   ```

   Replace keypair.pem with the name of your SSH key pair file, and <master-node-public-dns> with the public DNS name of the master node for your EMR cluster.

5. If prompted, type "yes" to confirm the connection to the server.
6. You are now connected to the master node of your EMR cluster via SSH. From here, you can run commands and manage your EMR cluster using the command line.

By following these steps, you can easily connect to your EMR cluster using SSH via the command prompt or terminal application. It's important to keep your SSH key pair secure and protected, as it provides access to your EMR cluster.

Run PySpark code on EMR cluster

Code to filter out records by following conditions:

- isRobot: False
- countryName: “United States”

PySpark code:

```
from pyspark.sql import SparkSession

#Defining input path of data
S3_DATA_INPUT_PATH="s3://s3-projectpro-emr-athena-sripad/source-folder/wikiticker-2015-09-12-sampled.json"

#Defining output path of data
S3_DATA_OUTPUT_PATH_FILTERED="s3://s3-projectpro-emr-athena-sripad/data-output/filtered"

def main():

    #Creating spark session object
    spark = SparkSession.builder.appName('projectProDemo').getOrCreate()

    #Reading source data
    df = spark.read.json(S3_DATA_INPUT_PATH)
    print(f'The total number of records in the source data set is {df.count()}')

    #Filter out the required data using following conditions
    filtered_df = df.filter((df.isRobot == False) & (df.countryName == 'United States'))
    print(f'The total number of records in the filtered data set is {filtered_df.count()}')
    filtered_df.show(10)
    filtered_df.printSchema()

    #Push the processed/filtered data to filtered bucket in parquet format
    filtered_df.write.mode('overwrite').parquet(S3_DATA_OUTPUT_PATH_FILTERED)
    print('The filtered output is uploaded successfully')

if __name__ == '__main__':
    main()
```

To create a Python file using the vi editor in an EMR cluster, you can follow these steps:

1. Connect to the master node of your EMR cluster using SSH. You can use the command prompt or terminal application as described in the previous steps.
2. Navigate to the directory where you want to create the Python file. You can use the cd command to navigate to the desired directory.
3. Use the vi command to create the Python file. For example, to create a file named main.py, you can use the following command:

```
vi main.py
```

4. Press the **`i`** key to enter the insert mode.
5. Type your Python code into the file.
6. Press the **`Esc`** key to exit the insert mode.
7. Type **`:wq`** and press Enter to save and exit the file.
    - The **`:`** puts **`vi`** into command mode.
    - The **`w`** command writes the file to disk.
    - The **`q`** command quits vi.

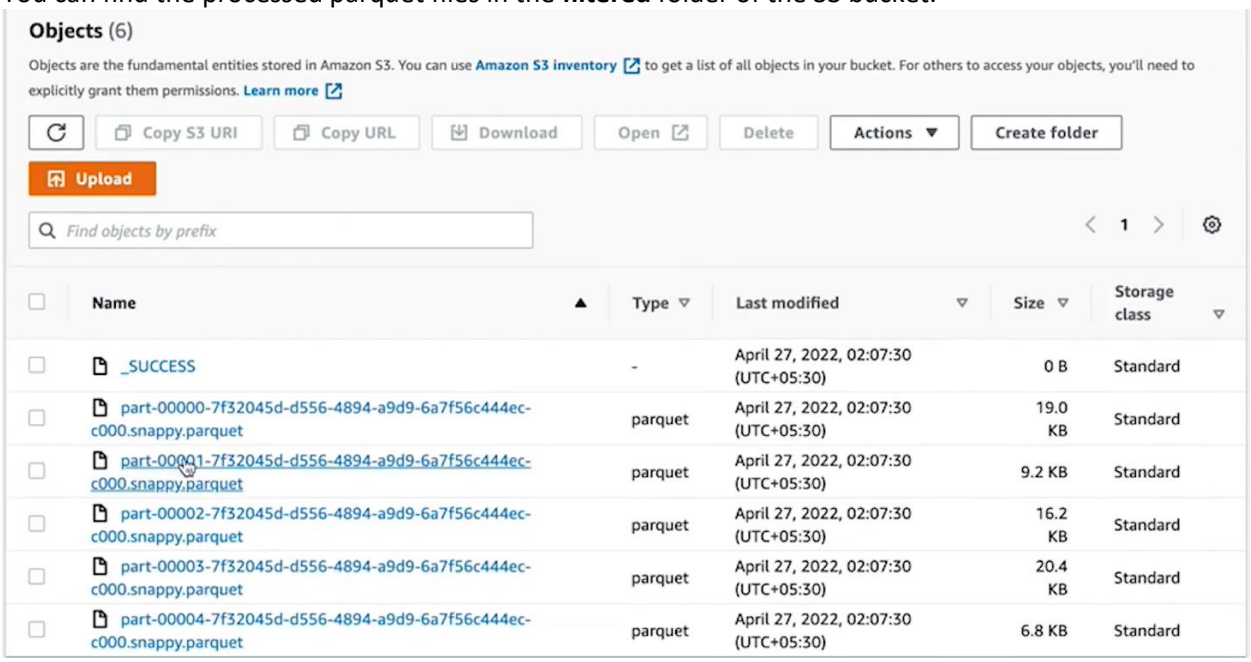
Now you have successfully created a Python file using vi in your EMR cluster. You can run the Python file using the spark-submit command to execute it on your EMR cluster.

Use the following command to run the Python file in EMR cluster:

```
spark-submit main.py
```

Output:

You can find the processed parquet files in the **filtered** folder of the S3 bucket.



Code to aggregate the data by "Channel"

PySpark code:

```
from pyspark.sql import SparkSession

#Defining input path of data
S3_DATA_INPUT_PATH="s3://s3-projectpro-emr-athena/source-folder/wikiticker-2015-09-12-sampled.json"

#Defining output path of data
S3_DATA_OUTPUT_PATH_AGGREGATED="s3://s3-projectpro-emr-athena/data-output/aggregated"

def main():
    #Creating spark session object
    spark = SparkSession.builder.appName('projectProDemo1').getOrCreate()

    #Reading source data
    df = spark.read.json(S3_DATA_INPUT_PATH)
    print(f'The total number of records in the input data set is {df.count()}')

    #Applying group by operator on channel and taking the count
    aggregated_df = df.groupBy(df.channel).count()
    print(f'The total number of records in the aggregated data set is {aggregated_df.count()}')
    aggregated_df.show(10)
    aggregated_df.printSchema()

    #Push the processed/aggregated data to filtered bucket in parquet format
    aggregated_df.write.mode('overwrite').parquet(S3_DATA_OUTPUT_PATH_AGGREGATED)
    print('The aggregated data has been uploaded successfully')


if __name__ == '__main__':
    main()
```

To create a Python file using the vi editor in an EMR cluster, you can follow these steps:

1. Connect to the master node of your EMR cluster using SSH. You can use the command prompt or terminal application as described in the previous steps.
2. Navigate to the directory where you want to create the Python file. You can use the cd command to navigate to the desired directory.
3. Use the vi command to create the Python file. For example, to create a file named main.py, you can use the following command:

   ```
   vi aggregation.py
   ```

4. Press the **`i`** key to enter the insert mode.
5. Type your Python code into the file.
6. Press the **`Esc`** key to exit the insert mode.
7. Type **`:wq`** and press Enter to save and exit the file.
    - The **`:`** puts **`vi`** into command mode.
    - The **`w`** command writes the file to disk.
    - The **`q`** command quits vi.

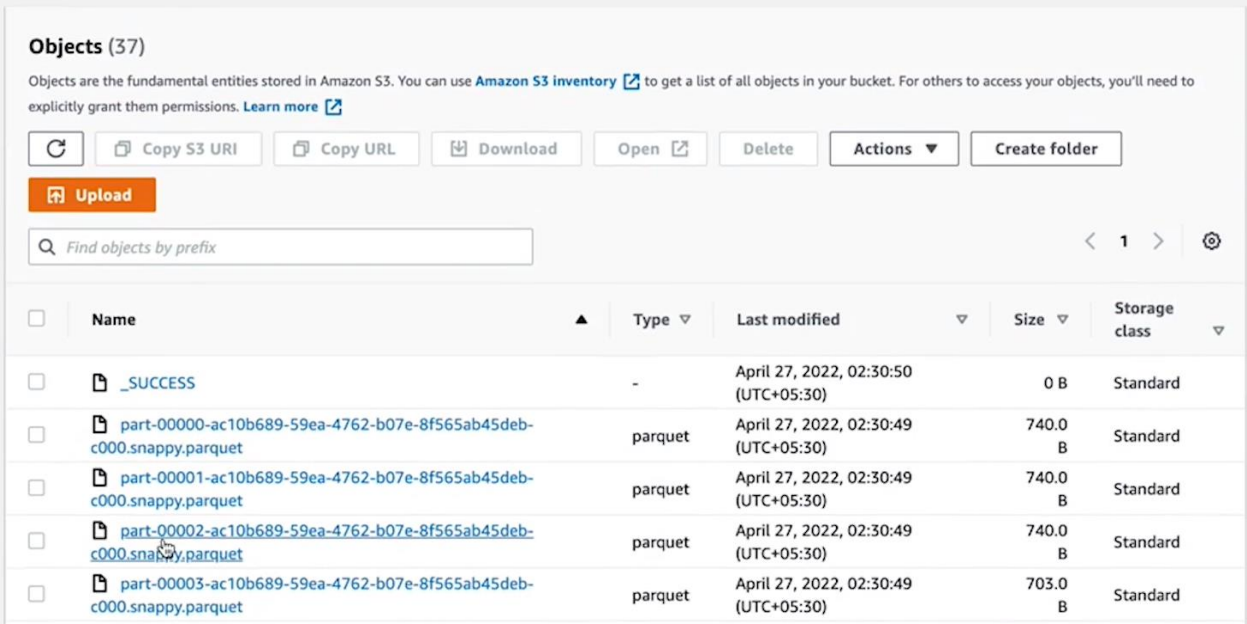
Now you have successfully created a Python file using vi in your EMR cluster. You can run the Python file using the spark-submit command to execute it on your EMR cluster.

Use the following command to run the Python file in EMR cluster:

```
spark-submit aggregation.py
```

Output:

You can find the processed parquet files in the **aggregated** folder of the S3 bucket.



Amazon Athena

We will create tables on top of the processed data stored in the AWS S3 bucket by providing the relevant schema in Athena.

To create tables in Amazon Athena on top of data stored in AWS S3, follow these steps:

1. Open the Athena console in the AWS Management Console.
2. Click on the "Query Editor" tab in the left navigation pane.
3. In the query editor, click on the "Database" dropdown in the top-left corner and select the database where you want to create the table.
4. Click on the "Create Table" button in the top-right corner of the query editor.
5. In the "Create Table" wizard, select the "From S3 bucket data" option.
6. In the "From S3 bucket data" form, enter the following information:
    - S3 location: The location of the S3 bucket and folder where your data is stored.
    - File format: The file format of your data (e.g. CSV, JSON, Parquet).
    - Table name: The name of the table you want to create.
    - Database name: The database where you want to create the table.
    - IAM role: The IAM role that has access to your S3 bucket and folder.
    - Additional options: Additional options to specify, such as custom delimiter, compression, or schema.
      Schema:

```
added bigint, channel string, cityName string, comment string, countryIsoCode string, countryName string, deleted bigint, delta bigint, isAnonymous boolean, isMinor boolean, isNew boolean, isRobot boolean, isUnpatrolled boolean, metroCode bigint, namespace string, page string, regionIsoCode string, regionName string, time string, user string
```

7. Click on the "Preview table" button to preview the data in your S3 file.
8. Once you have confirmed that the preview looks correct, click on the "Create table" button to create the table.
9. After the table is created, you can query the data using SQL queries in the query editor.

When creating a table in Athena on top of data stored in S3, make sure to specify the correct data format, delimiter, and file type used in the S3 file. Additionally, you must have the appropriate IAM permissions to access the S3 bucket and folder where your data is stored. Now you can perform various queries to perform analysis on data.

Summary

- In this training, we performed batch processing on Wikipedia data.
- Batch processing is a technique used in data processing where a large volume of data is processed at once, rather than processing it individually or in real-time. Wikipedia, being a massive online encyclopedia, generates a tremendous amount of data every day.
- The reusability of the proposed pipeline in the various industries was discussed.
- We understood each service used in this project in detail, their advantages, disadvantages and use cases.
- We created an EMR cluster and processed raw data using PySpark.
- Performed queries on Amazon Athena to analyse the data.
- Various AWS services were leveraged to implement the concepts with the most cost-optimized configurations that can even serve 100x the amount of demo data used.