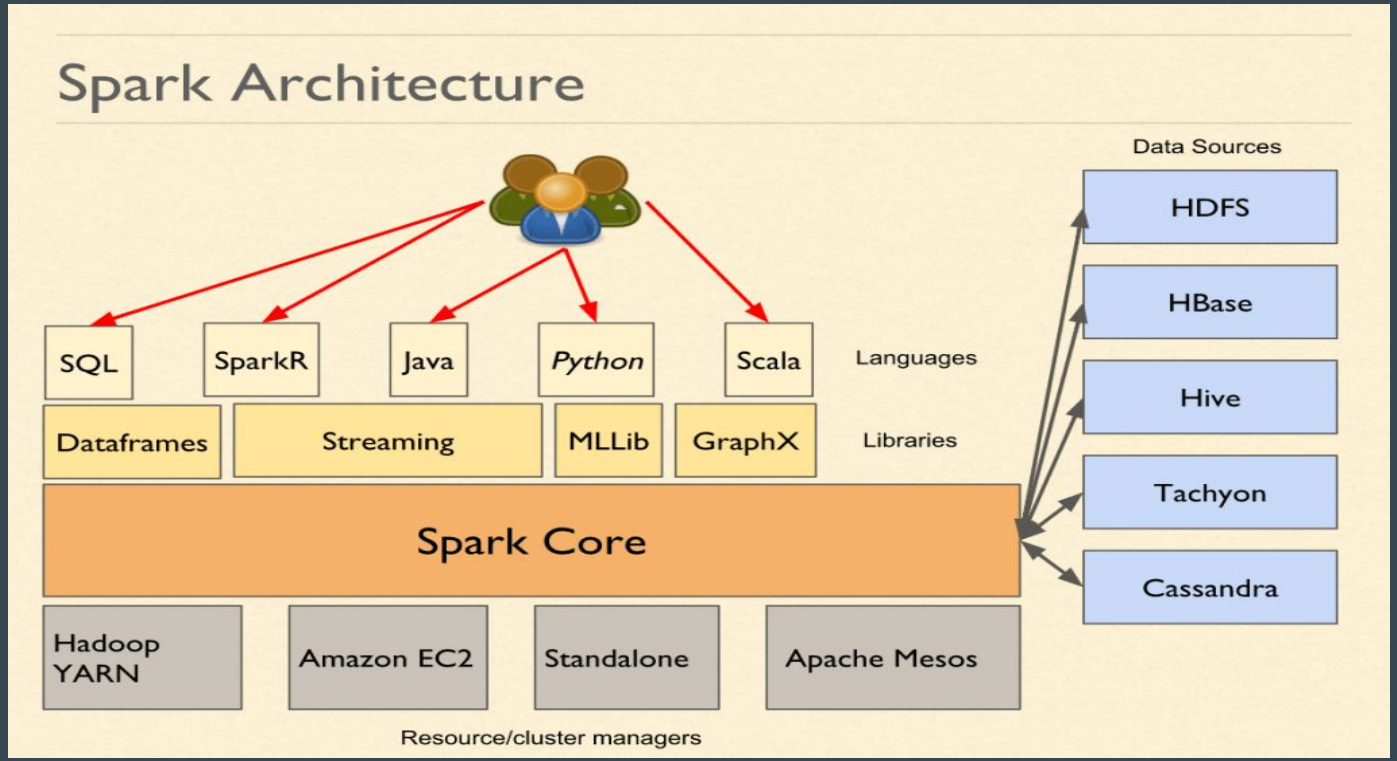
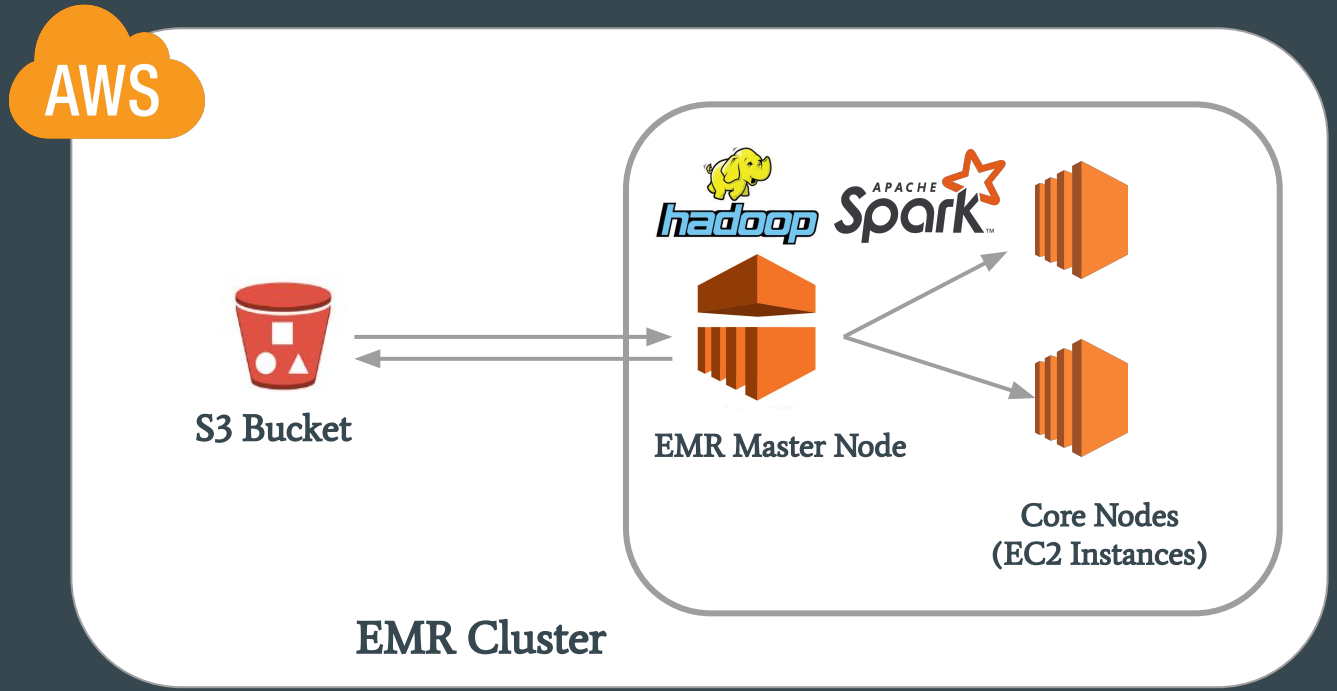
Big Data Processing Using Spark

A Guide by Michelle Feng and Elisa Miller
Created for ECE 4150 Spring 2024

High Level Overview



High Level Overview II



Introduction

Big Data Analysis Using Apache Spark and EMR

EMR

- Use S3 bucket as its file system
- Use EC2 instances as worker nodes
- Pre-configured with Spark + Hadoop

Apache Spark

- Exponentially faster than Hadoop
- More advanced
- Use Hadoop and Spark together

Big Data

- Popular in the real world
- Additional applications
    - Business Intelligence, Financial Analysis, Healthcare Analytics...

Example

Guided Example to Run Apache Spark for Data Analytics

Data

- Survey results of people using Stack Overflow
- Different fields like occupation, country, compensation, experience, etc. are represented

Analysis

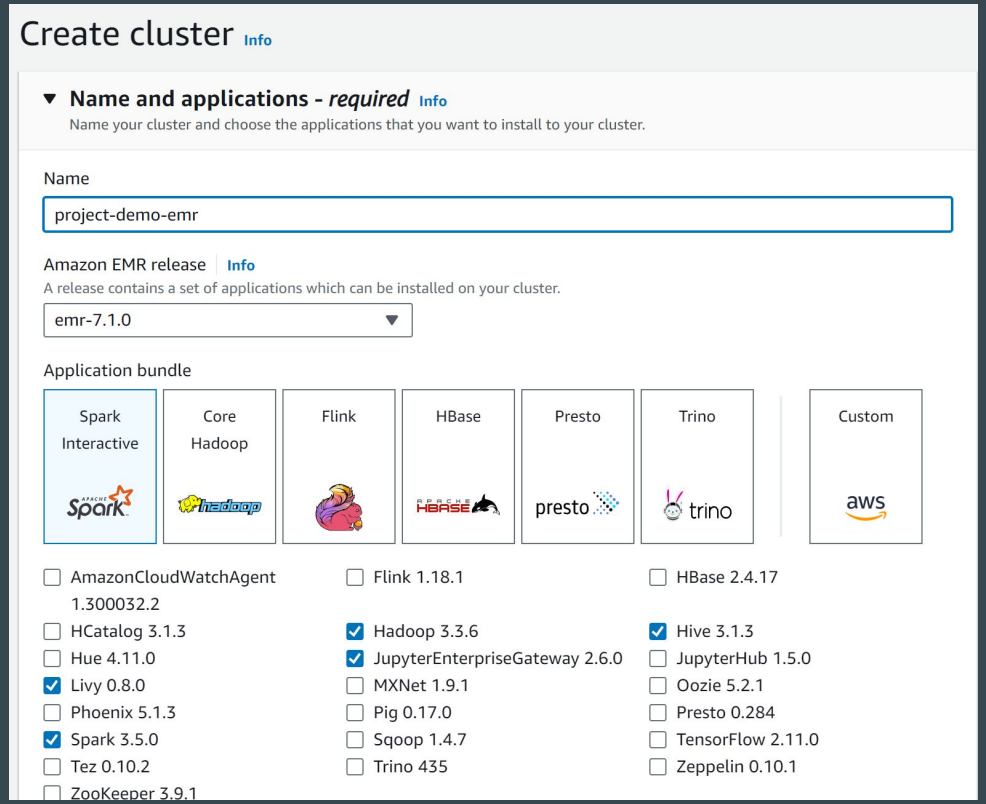
- Filter data based on specific characteristics using Apache Spark
- Store results in S3 bucket
- Run further analysis for visual representation of big data

Steps

Overview of the Steps Involved in this Guide

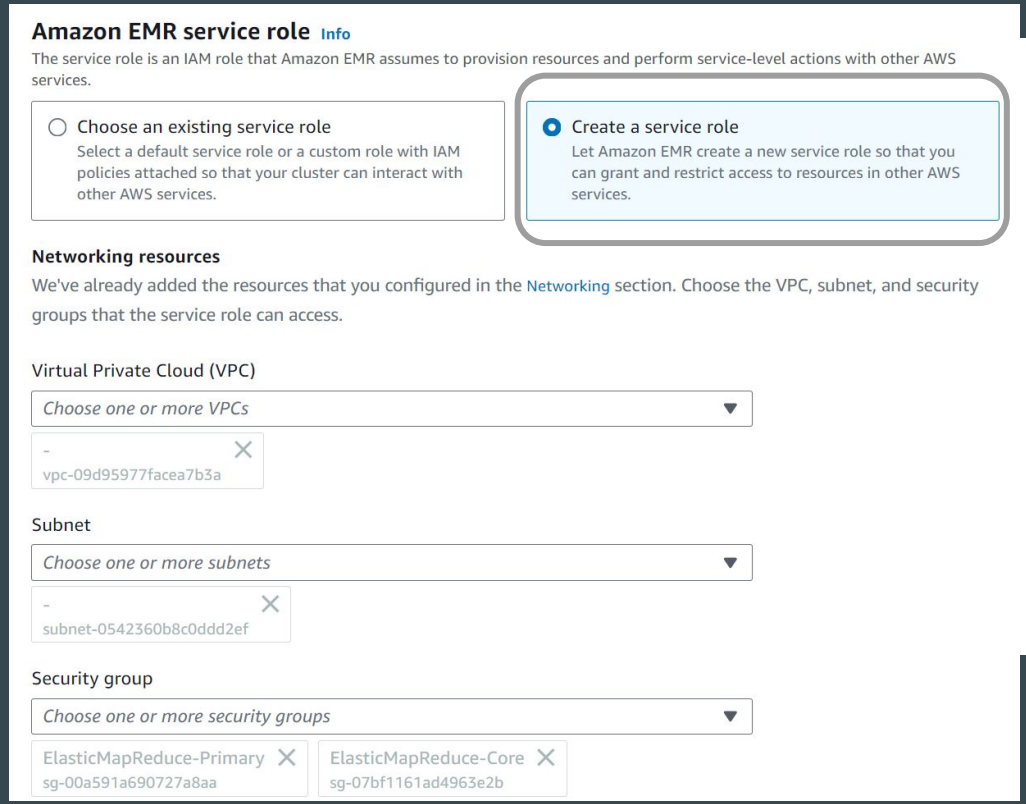
1. Create and configure EMR cluster for Spark
2. Create and configure S3 Bucket file system
3. Write Python code for data processing
4. SSH into Hadoop
5. Create and run Spark code in the Hadoop terminal
6. View results in S3 bucket

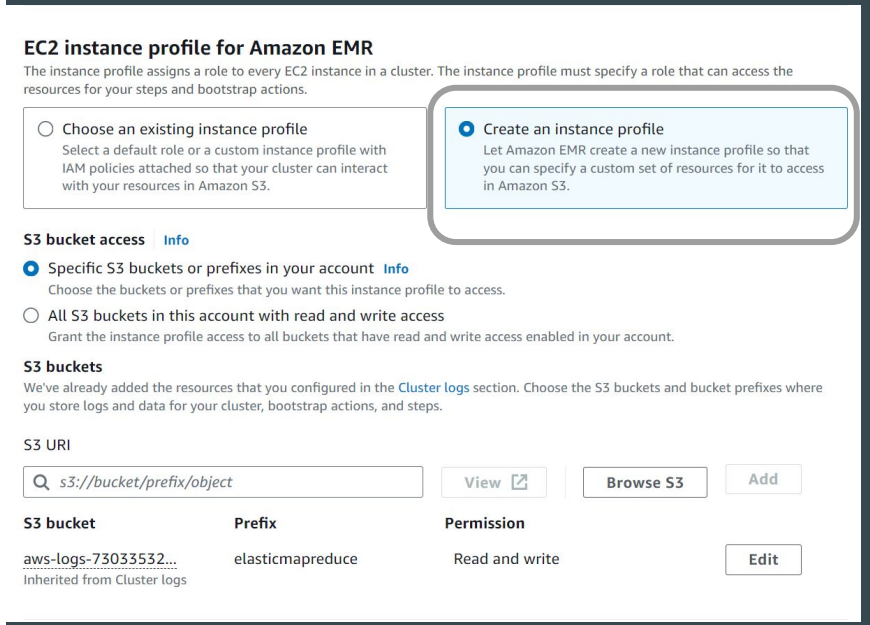
Create a EMR instance with Apache Spark



- Choose Spark Application bundle
- **Cluster configuration** - remove task instance group
- Keep the defaults for Cluster scaling and provisioning and Networking
- **Security configuration and EC2 key pair** - choose an EC2 pair you already have or create one

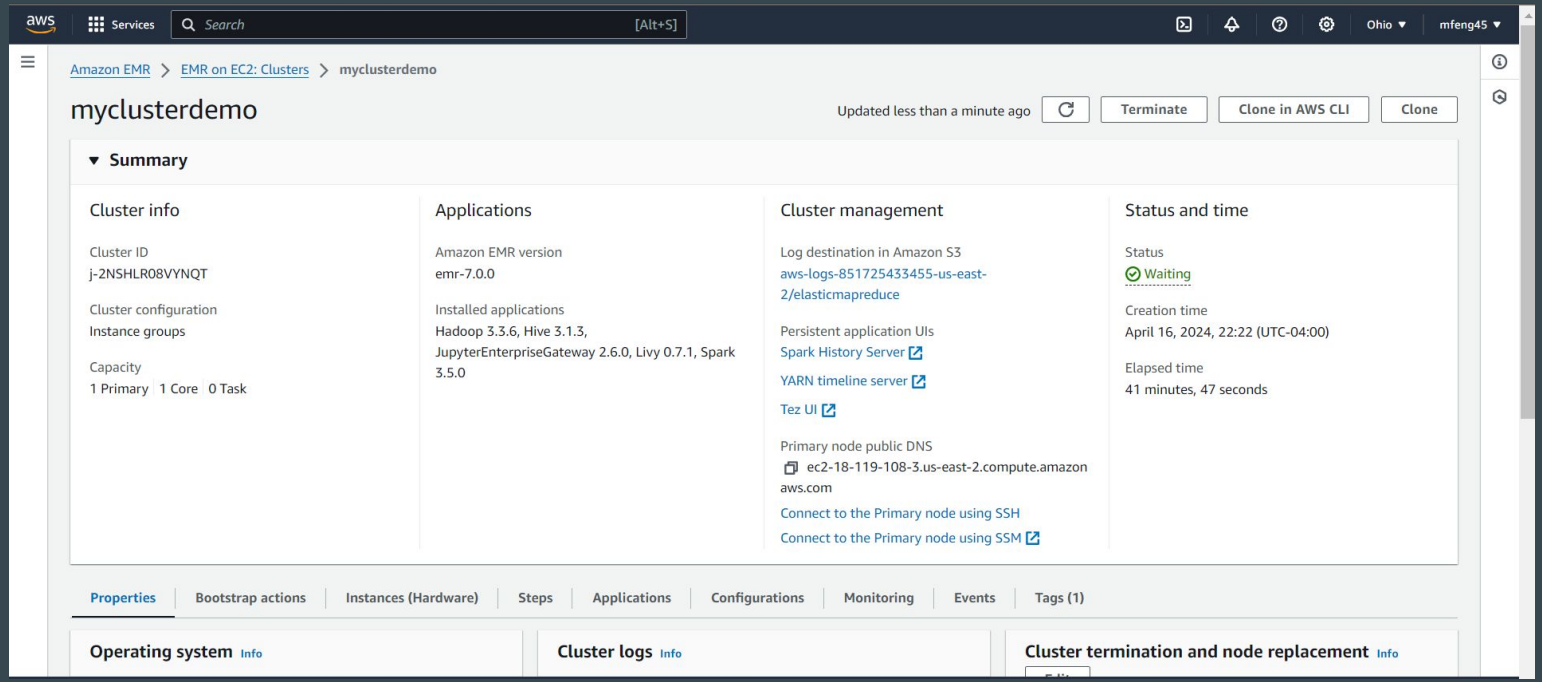
Identity and Access Management (IAM) roles





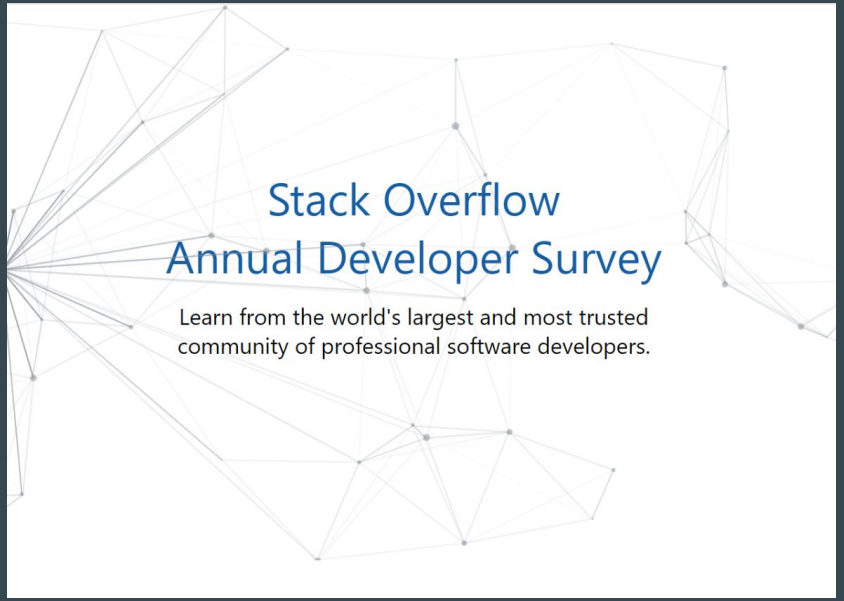
Leave these values as default, but need to tell EMR to create its own roles to access S3 bucket

Start the EMR instance with Apache Spark



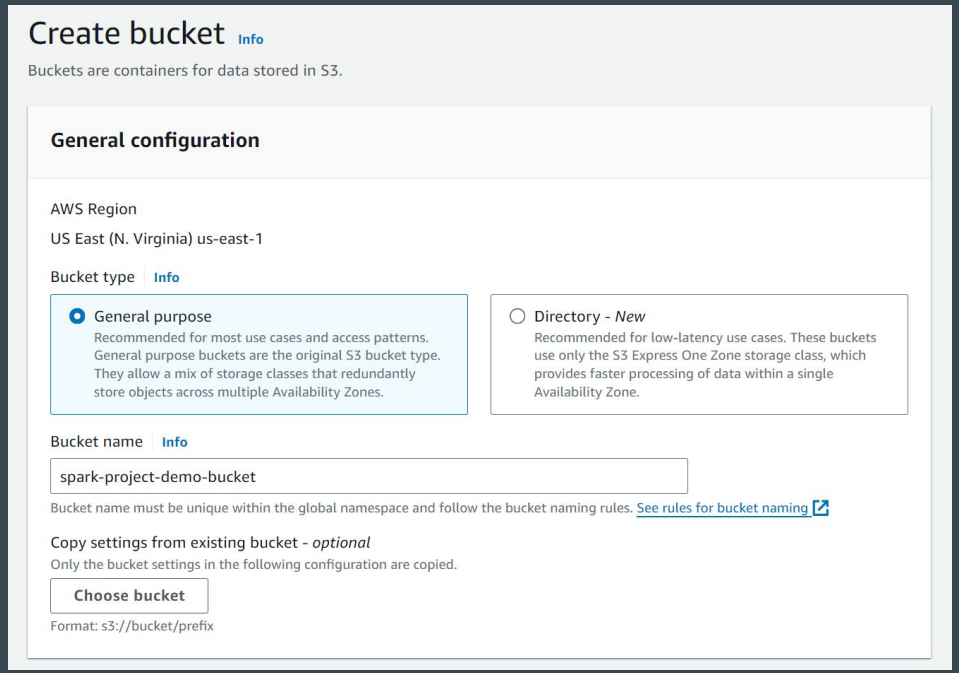
Wait until the Status is in the Waiting stage, this will take around 10 minutes

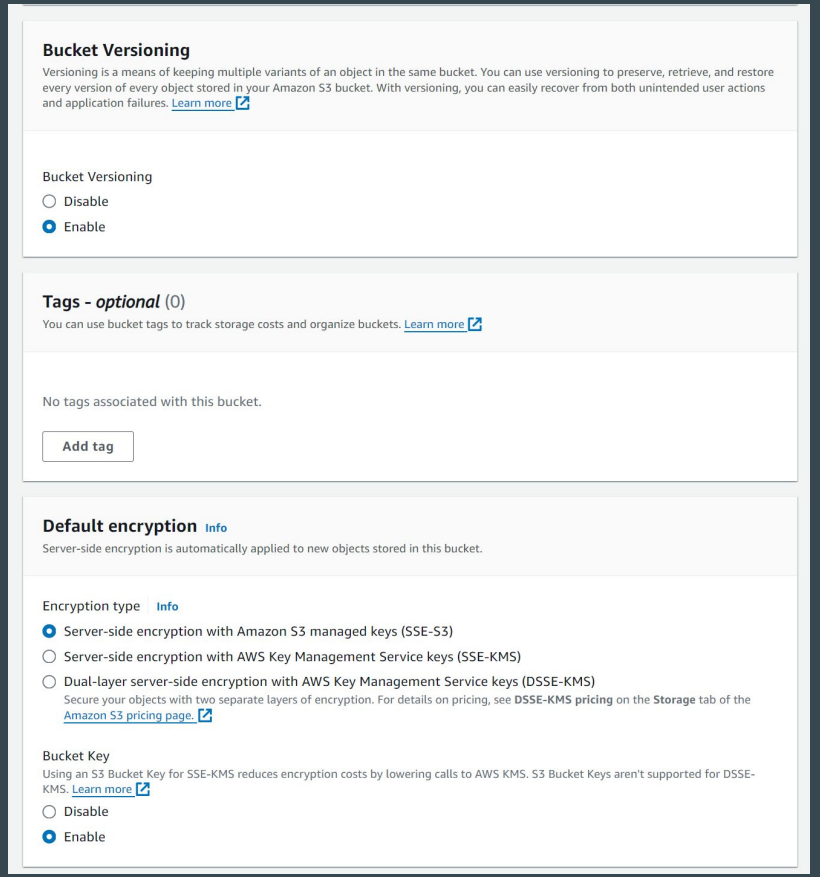
Download Data for Processing



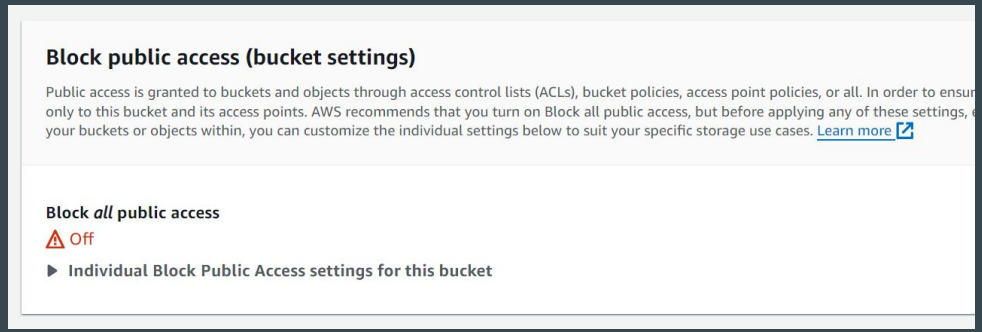
- For this guide, we will be using data from a Stack Overflow Survey from 2022
- You can use any dataset you want to analyze!

Create S3 Bucket to Store Data

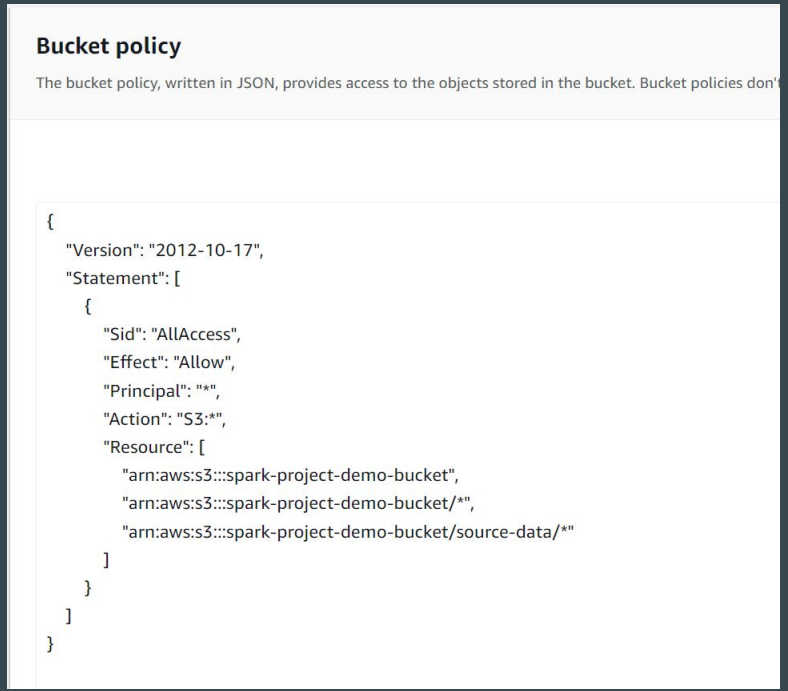




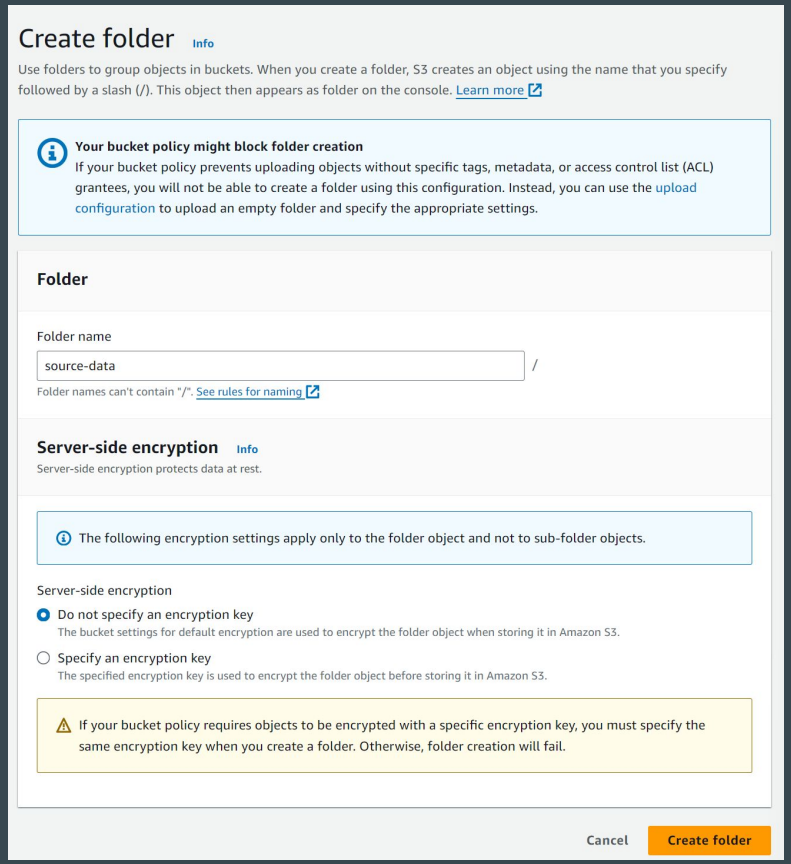
Ensure Public Access to S3 Bucket

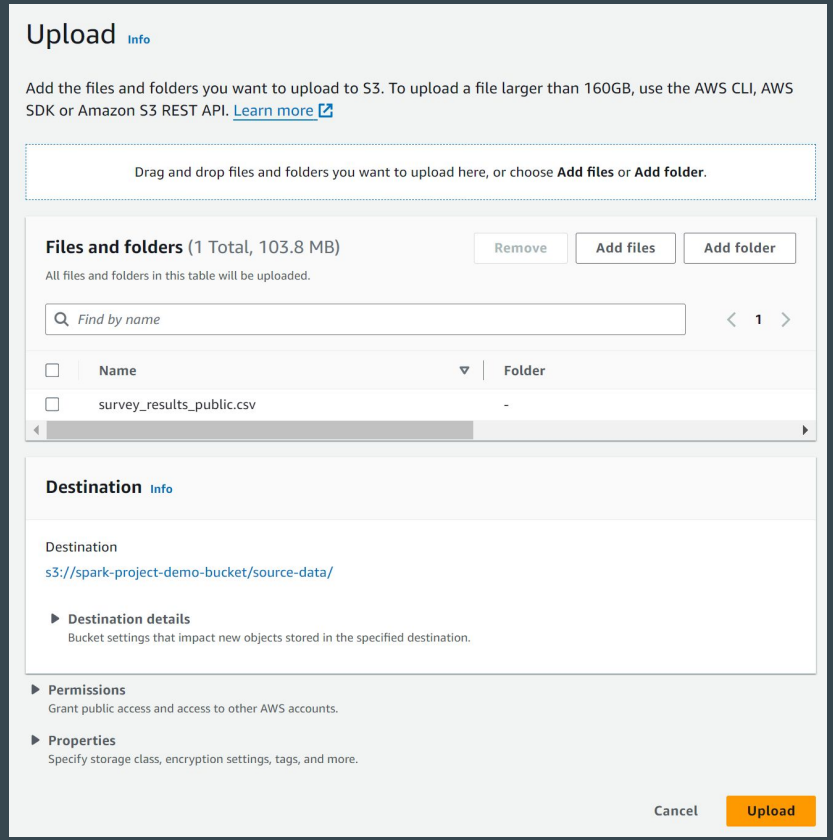


- Turn OFF Block all public access.
- Add the following Bucket policy:



Create Folder in S3 Bucket and Upload Data





Write Python Code for Spark Processing

final_project_processing.py 2

C: > gatech > ece 4150 > project > final_project_processing.py > ...

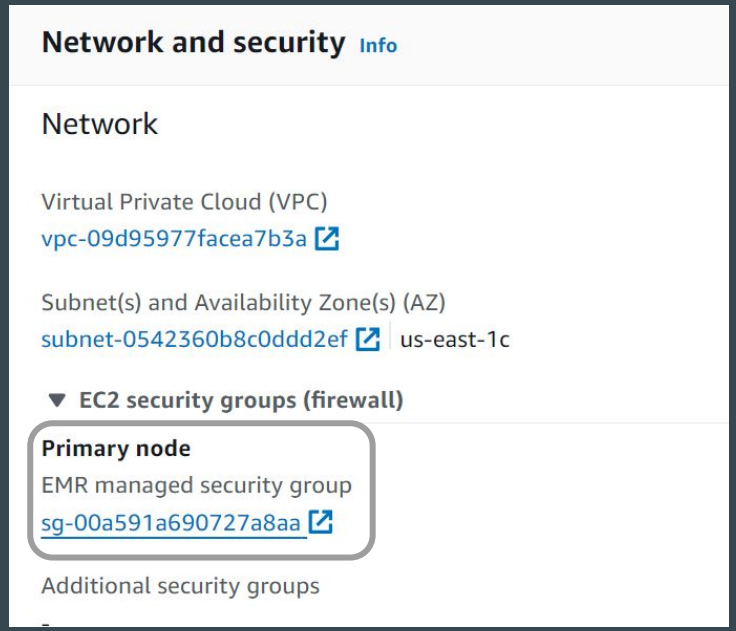
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# CHANGE BUCKET NAME and FOLDER ACCORDINGLY
S3_DATA_SOURCE_PATH = 's3://spark-project-demo-bucket/source-data/survey_results_public.csv'
S3_OUTPUT_SOURCE_PATH = 's3://spark-project-demo-bucket/output-data'


def main():
    spark = SparkSession.builder.appName('ProjectDemoApp').getOrCreate()
    all_data = spark.read.csv(S3_DATA_SOURCE_PATH, header=True)
    print('Total number of records in the source data: %s' % all_data.count())
    selected_data = all_data.where((col('Country') == 'United States of America') & (col('YearsCode') > 15))
    print('The number of Stack Overflow users who live in the USA and have been learning code for more than 15 years
    is: %s' % selected_data.count())
    selected_data.write.mode('overwrite').csv(S3_OUTPUT_SOURCE_PATH)
    print('Selected data was sucessfully saved to s3: %s' % S3_OUTPUT_SOURCE_PATH)

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

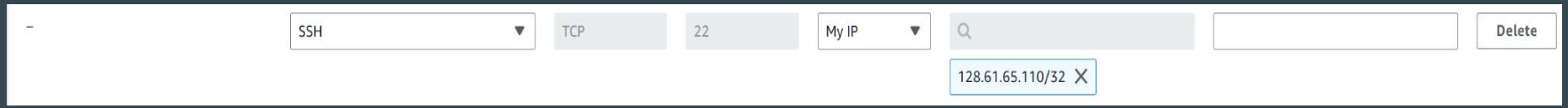
s3://YourS3Bucket/YourData.csv

s3://YourS3Bucket/output-data

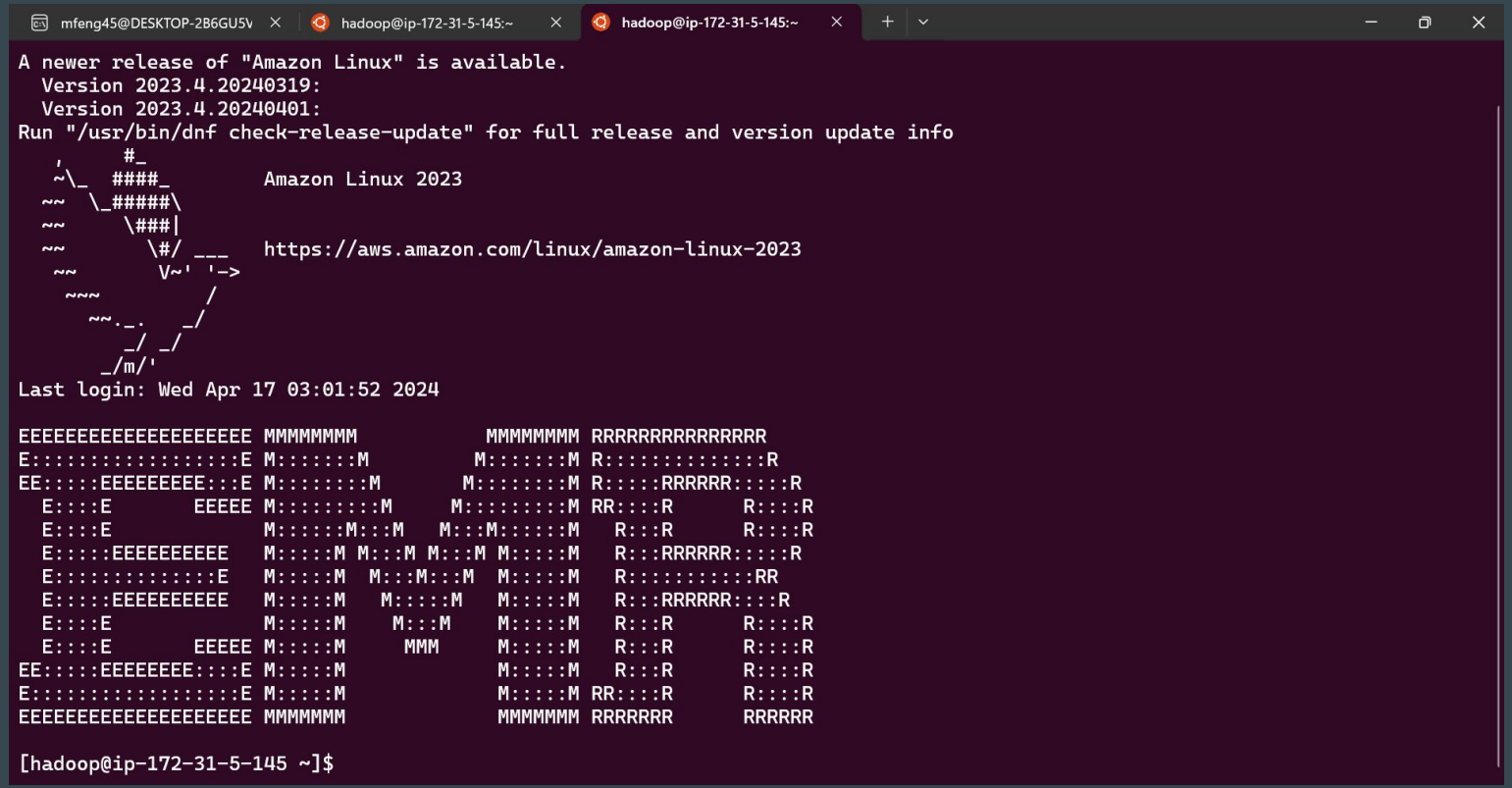
Before you SSH, you need to edit the EC2 SecurityGroup

- Go to the EMR instance
- Go to Network Security - Security Groups and open the security group for the master cluster (ElasticMapReduce-master)
- Edit the inbound rules and add a rule for SSH: TCP: 22: My IP and save changes





SSH into Hadoop in a terminal containing your key



Create main.py in the Hadoop Terminal

```
[hadoop@ip-172-31-5-145 ~]$ vim main.py
```

hadoop@ip-172-31-10-31:~

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# CHANGE BUCKET NAME and FOLDER ACCORDINGLY
S3_DATA_SOURCE_PATH = 's3://spark-project-demo-bucket/source-data/survey_results_public.csv'
S3_OUTPUT_SOURCE_PATH = 's3://spark-project-demo-bucket/output-data'


def main():
    spark = SparkSession.builder.appName('ProjectDemoApp').getOrCreate()
    all_data = spark.read.csv(S3_DATA_SOURCE_PATH, header=True)
    print('Total number of records in the source data: %s' % all_data.count())
    selected_data = all_data.where((col('Country') == 'United States of America') & (col('YearsCode') > 15))
    print('The number of Stack Overflow users who live in the USA and have been learning code for more than 15 years is: %s' % selected_
data.count())
    selected_data.write.mode('overwrite').csv(S3_OUTPUT_SOURCE_PATH)
    print('Selected data was sucessfully saved to s3: %s' % S3_OUTPUT_SOURCE_PATH)

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

Run:

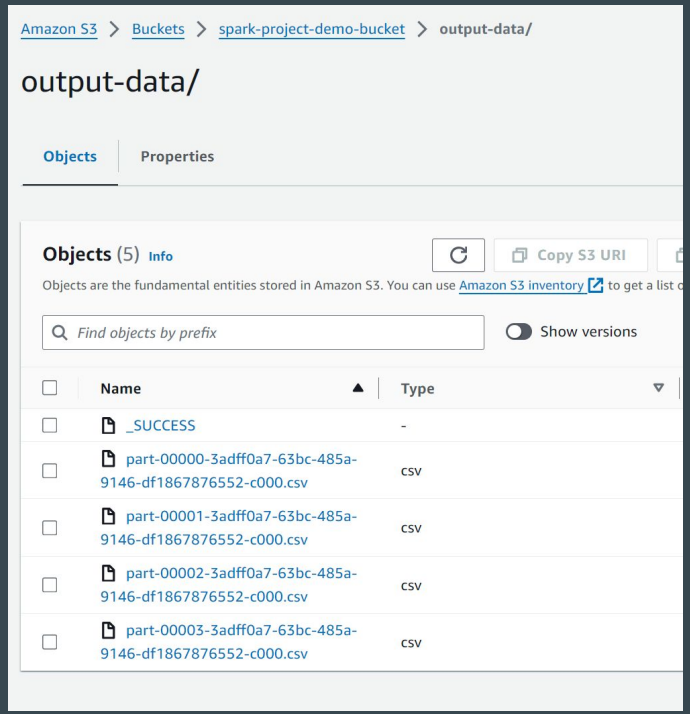
```
[hadoop@ip-172-31-10-31 ~]$ spark-submit main.py
```

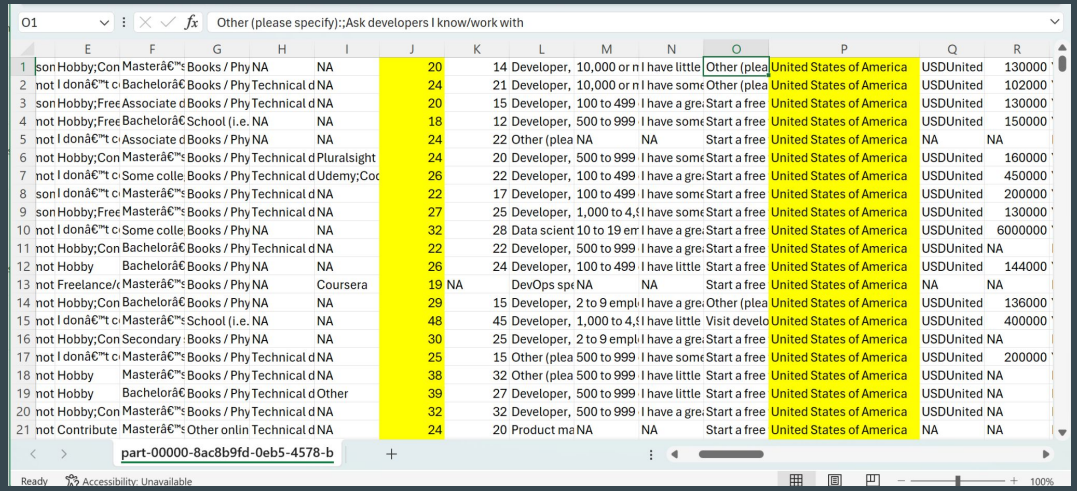
- The command line will look like this when the job is complete.
- It’s hard to read so let’s check the S3 bucket

```
24/04/17 17:48:15 INFO YarnClientSchedulerBackend: Shutting down all executors
24/04/17 17:48:15 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
24/04/17 17:48:15 INFO YarnClientSchedulerBackend: YARN client scheduler backend Stopped
24/04/17 17:48:15 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
24/04/17 17:48:15 INFO MemoryStore: MemoryStore cleared
24/04/17 17:48:15 INFO BlockManager: BlockManager stopped
24/04/17 17:48:15 INFO BlockManagerMaster: BlockManagerMaster stopped
24/04/17 17:48:15 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
24/04/17 17:48:15 INFO SparkContext: Successfully stopped SparkContext
24/04/17 17:48:15 INFO ShutdownHookManager: Shutdown hook called
24/04/17 17:48:15 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-957e64a9-732e-41e2-9179-7356e86f03ff/pyspark-69297b7e-
-418f-8e2d-c989f3dc2bbb
24/04/17 17:48:15 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-63174745-91a8-4e6a-9f9c-cc82f95f6d72
24/04/17 17:48:15 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-957e64a9-732e-41e2-9179-7356e86f03ff
[hadoop@ip-172-31-10-31 ~]$
```

```
24/04/17 17:54:09 INFO DAGScheduler: Job 4 finished: count at NativeMethodAccessorImpl.java:0, took 0.049885 s
The number of Stack Overflow users who live in the USA and have been learning code for more than 15 years is: 5117
```

Check out the output in the S3 bucket





Highlighted columns for visibility show ‘YearCode’ and ‘Country’

Write More Python Code...

- Similarly, you can write more code to process different columns of the data
  - For this data, say I want to look at Stack Overflow users that are ‘developer’s by profession’ and received a ‘CompTotal’ greater than $150,000 USD
- Spark has huge applications in data analytics in the real world!
  - Such as Social Media Analysis, Text Analytics, Operations Management, Fraud Detection, and more!

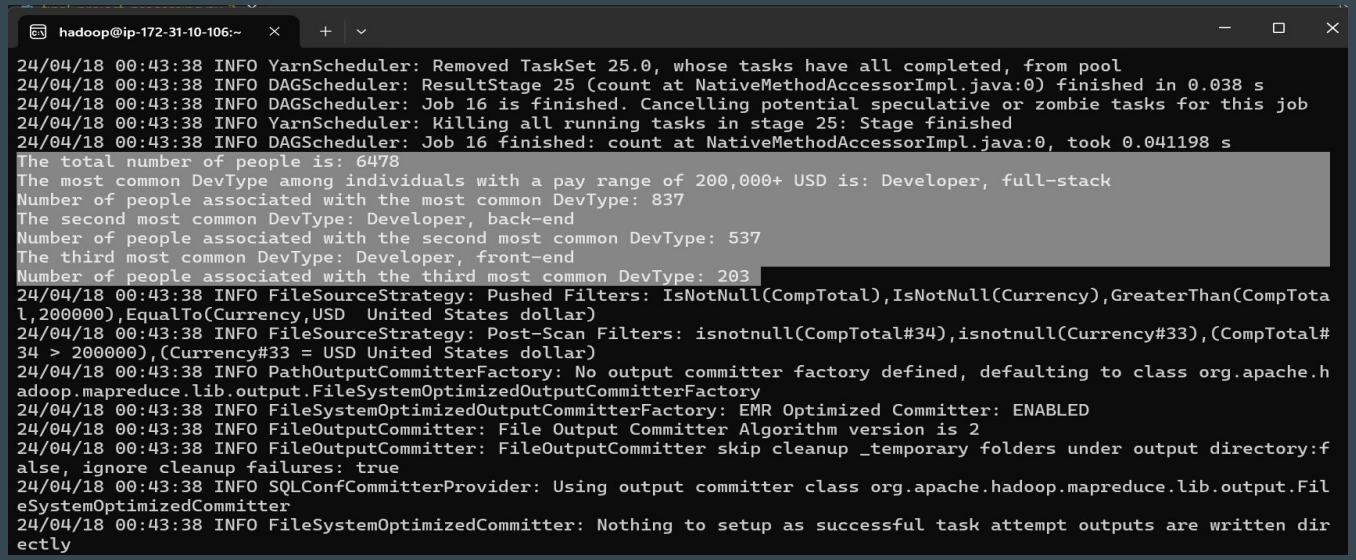
```
selected_data = all_data.where((col('MainBranch') == 'I am a developer by profession') &
                               (col('CompTotal') > '150000') &
                               (col('Currency') == 'USD United States dollar'))
print('The number of Stack Overflow users are professional developers and receive over 150000 USD in pay: %s' %
selected_data.count())
selected_data.write.mode('overwrite').csv(S3_OUTPUT_SOURCE_PATH)
print('Selected data was sucessfully saved to s3: %s' % S3_OUTPUT_SOURCE_PATH)
```

More on Big Data...

```
# CONTINUATION OF PROBLEM 2 IN ORDER TO FIGURE OUT WHAT PROFESSION MAKES THE MOST MONEY
# PROBLEM 3: PROCESS DATA FOR MORE CRITERIA - DEVTYPE BY PROFESSION MAKING OVER 200,000+ USD
selected_data = all_data.where((col('CompTotal') > '200000') &
                               (col('Currency') == 'USD United States dollar'))
print('The number of Stack Overflow users that receive over 200000 USD in pay: %s' % selected_data.count())
# NOW FILTER THE DATA TO DETERMINE THE DEVTYPE THAT MOST COMMONLY MAKES 200,000+
filtered_data = all_data.where((col('CompTotal') > '200000') &
                               (col('Currency') == 'USD United States dollar'))
devtype_counts = filtered_data.groupBy('DevType').count()
# Order by count descending and collect the result
ordered_devtypes = devtype_counts.orderBy(col('count').desc()).collect()
# Extract the first, second, and third most common DevTypes and their counts
most_common_devtype = ordered_devtypes[0]['DevType']
most_common_count = ordered_devtypes[0]['count']
second_common_devtype = ordered_devtypes[1]['DevType'] if len(ordered_devtypes) > 1 else None
second_common_count = ordered_devtypes[1]['count'] if len(ordered_devtypes) > 1 else None
third_common_devtype = ordered_devtypes[2]['DevType'] if len(ordered_devtypes) > 2 else None
third_common_count = ordered_devtypes[2]['count'] if len(ordered_devtypes) > 2 else None
# Print the results
print('The total number of people is: %s' % filtered_data.count())
print('The most common DevType among individuals with a pay range of 200,000+ USD is:', most_common_devtype)
print('Number of people associated with the most common DevType:', most_common_count)
print('The second most common DevType:', second_common_devtype)
print('Number of people associated with the second most common DevType:', second_common_count)
print('The third most common DevType:', third_common_devtype)
print('Number of people associated with the third most common DevType:', third_common_count)
```

Filter DevType by profession making over $200,000

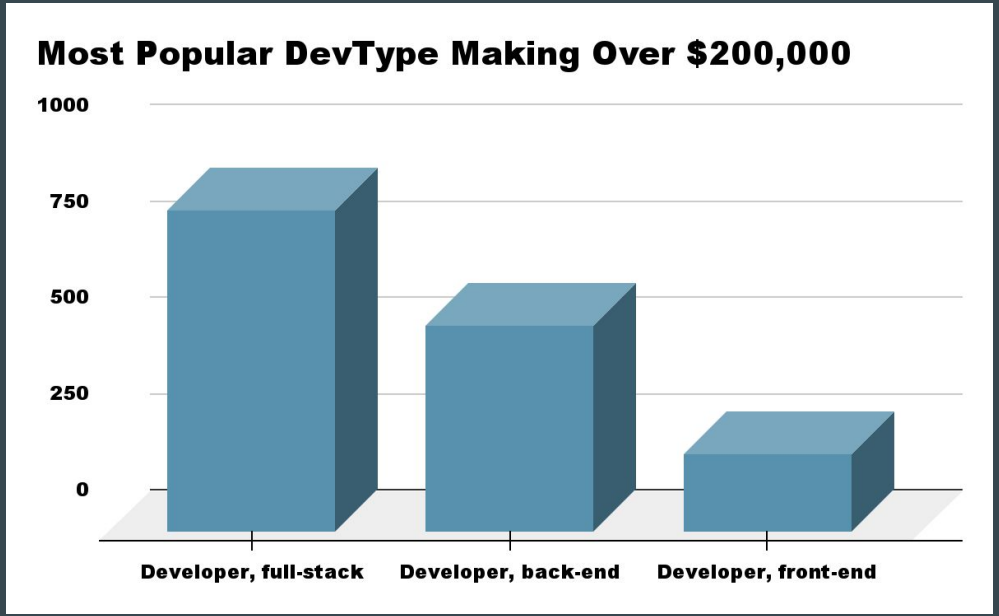
Big Data Continued...



We are going to use the highlighted portion of data to reveal the most common DevType that makes $200,000+

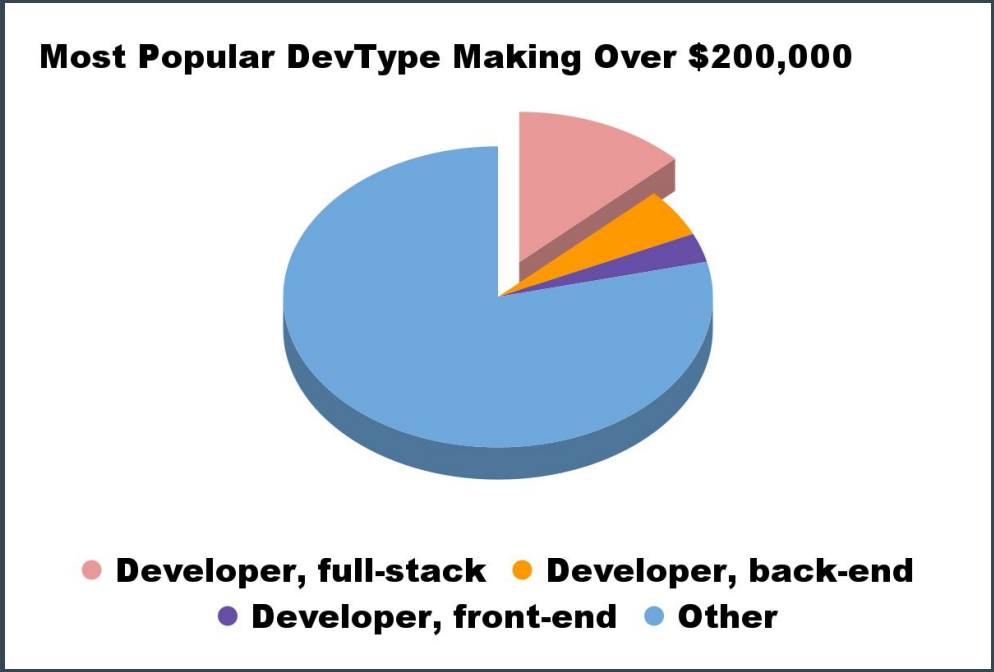
Let's Take a Look at the Visual Representation of Data

- This column chart depicts the Developer, full-stack as the most popular DevType making over $200,000 by far
- However, let's take a look at a different visual representation of the data



Visual Representation of Data Continued...

- This pie chart is critical in order to understand the data collected
- While the previous column chart depicted Developer, full-stack to be the most popular, it didn't take into account the total number of occupations. Thus, skewing the perception of the results



To Sum it All Up

- At the end of the day, the way you represent data matters
- In this guide, you used AWS EMR, AWS S3, Apache Spark, and more!
- Apache Spark is just one tool that can be used to draw conclusions from big data
- Spark was used to filter, organize, and export data!

Resources

Spark Tutorials:

- Intro to Amazon EMR - Big Data Tutorial using Spark
- AWS EMR Big Data Processing with Spark and Hadoop | Python, PySpark, Step by Step Instructions

Stack Overflow:

- Stack Overflow Annual Developer Survey

Images:

- https://avinash333.files.wordpress.com/2019/08/spark-architecture.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/5/5c/AWS_Simple_Icons_AWS_Cloud.svg/2560px-AWS_Simple_Icons_AWS_Cloud.svg.png
- https://miro.medium.com/v2/resize:fit:1400/1*pD4tIQG8Tn2-XQkI5-jqUg.png
- https://www.zuar.com/blog/content/images/2022/02/emr.png
- https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.linkedin.com%2Fpulse%2Fsetting-up-hadoop-cluster-top-aws-checking-existence-replica-kumar&psig=AOvVaw2jW57_vKBezieWCmj89LCr&ust=1713467403324000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCLCf3ND5yYUDFQAAAAAdAAAAABAI
- https://miro.medium.com/v2/resize:fit:1400/1*hz7npUiTzL9ieTVhCRNHEw.png