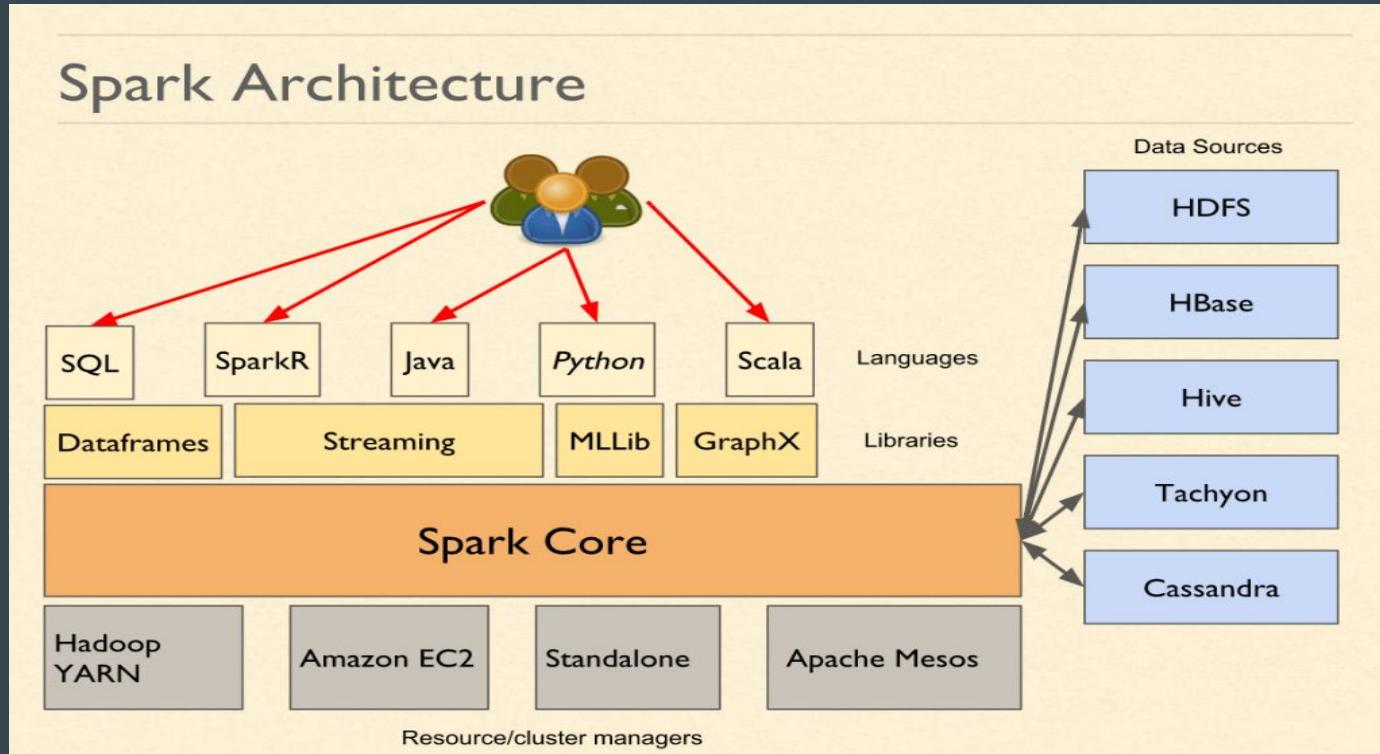


Big Data Processing Using Spark

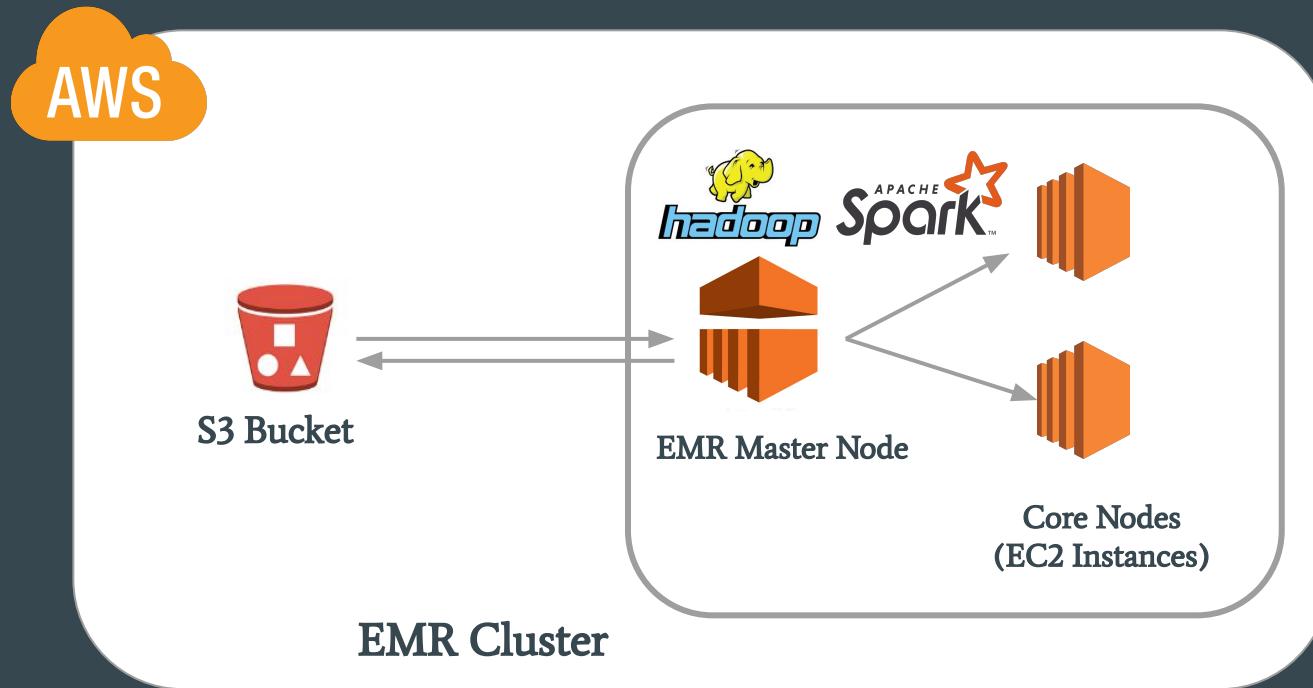
...

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

Create cluster [Info](#)

▼ Name and applications - *required* [Info](#)

Name your cluster and choose the applications that you want to install to your cluster.

Name

project-demo-emr

Amazon EMR release | [Info](#)

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

emr-7.1.0

Application bundle

Spark Interactive Core Hadoop Flink HBase Presto Trino Custom

<input type="checkbox"/> AmazonCloudWatchAgent 1.300032.2	<input type="checkbox"/> Flink 1.18.1	<input type="checkbox"/> HBase 2.4.17
<input checked="" type="checkbox"/> HCatalog 3.1.3	<input checked="" type="checkbox"/> Hadoop 3.3.6	<input checked="" type="checkbox"/> Hive 3.1.3
<input type="checkbox"/> Hue 4.11.0	<input checked="" type="checkbox"/> JupyterEnterpriseGateway 2.6.0	<input type="checkbox"/> JupyterHub 1.5.0
<input checked="" type="checkbox"/> Livy 0.8.0	<input type="checkbox"/> MXNet 1.9.1	<input type="checkbox"/> Oozie 5.2.1
<input type="checkbox"/> Phoenix 5.1.3	<input type="checkbox"/> Pig 0.17.0	<input type="checkbox"/> Presto 0.284
<input checked="" type="checkbox"/> Spark 3.5.0	<input type="checkbox"/> Sqoop 1.4.7	<input type="checkbox"/> TensorFlow 2.11.0
<input type="checkbox"/> Tez 0.10.2	<input type="checkbox"/> Trino 435	<input type="checkbox"/> Zeppelin 0.10.1
<input type="checkbox"/> ZooKeeper 3.9.1		

- 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

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



vpc-09d95977facea7b3a

Subnet

Choose one or more subnets



subnet-0542360b8c0ddd2ef

Security group

Choose one or more security groups

ElasticMapReduce-Primary

sg-00a591a690727a8aa

ElasticMapReduce-Core

sg-07bf1161ad4963e2b

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.

S3 buckets

We've already added the resources that you configured in the [Cluster logs](#) section. Choose the S3 buckets and bucket prefixes where you store logs and data for your cluster, bootstrap actions, and steps.

S3 URI

s3://bucket/prefix/object



[Browse S3](#)

[Add](#)

S3 bucket

aws-logs-73033532...

Inherited from Cluster logs

Prefix

elasticmapreduce

Permission

Read and write

[Edit](#)

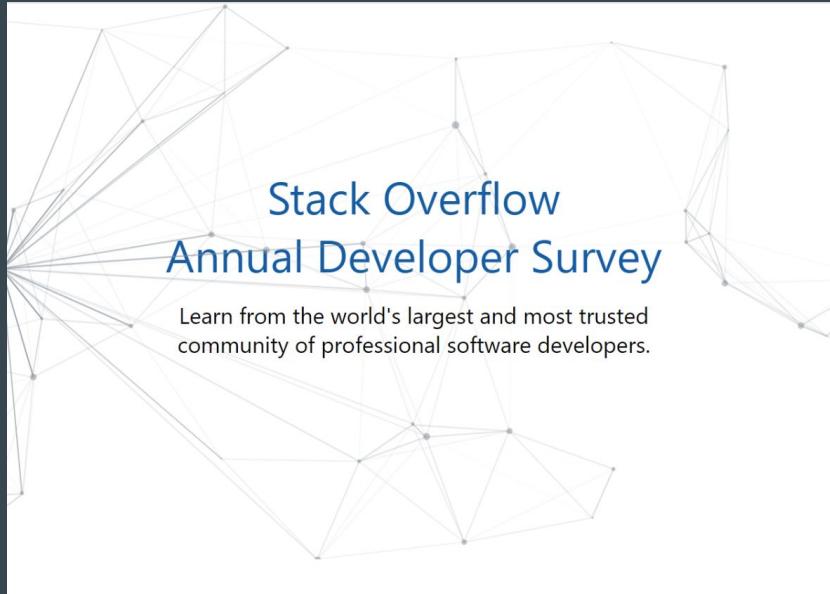
Leave these values as default, but need to tell EMR to create its own roles to access S3 bucket

Start the EMR instance with Apache Spark

The screenshot shows the AWS EMR Cluster Summary page for a cluster named "myclusterdemo". The cluster ID is j-2NSHLR08VYNQT, running version emr-7.0.0. It has one Primary instance and one Core instance. The applications installed include Hadoop 3.3.6, Hive 3.1.3, JupyterEnterpriseGateway 2.6.0, Livy 0.7.1, and Spark 3.5.0. The cluster management section shows log destination in Amazon S3 at aws-logs-851725433455-us-east-2/elasticmapreduce, and persistent application UIs for Spark History Server, YARN timeline server, and Tez UI. The status is currently Waiting. The creation time was April 16, 2024, at 22:22 UTC-04:00, and the elapsed time is 41 minutes, 47 seconds. The primary node's public DNS is ec2-18-119-108-3.us-east-2.compute.amazonaws.com. There are tabs for Properties, Bootstrap actions, Instances (Hardware), Steps, Applications, Configurations, Monitoring, Events, and Tags (1). Below the main summary are sections for Operating system, Cluster logs, and Cluster termination and node replacement.

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

Create bucket [Info](#)

Buckets are containers for data stored in S3.

General configuration

AWS Region

US East (N. Virginia) us-east-1

Bucket type [Info](#)

General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory - New

Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name [Info](#)

spark-project-demo-bucket

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

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. [Learn more](#)

Bucket Versioning

Disable

Enable

Tags - *optional* (0)

You can use bucket tags to track storage costs and organize buckets. [Learn more](#)

No tags associated with this bucket.

[Add tag](#)

Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

Server-side encryption with Amazon S3 managed keys (SSE-S3)

Server-side encryption with AWS Key Management Service keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the Storage tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable

Enable

Ensure Public Access to S3 Bucket

The screenshot shows the 'Block public access (bucket settings)' section. It includes a note about public access being granted through various methods like ACLs and bucket policies. A link to learn more is provided. Below this, there's a section for 'Block all public access' which is currently set to 'Off'. A link to 'Individual Block Public Access settings for this bucket' is also present.

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

The screenshot displays a JSON-based Bucket policy. The policy allows full access ('AllAccess') to all users ('*') for all actions ('S3:*') on specific resources within the bucket. These resources include the main bucket, its versioned objects, and objects in a specific folder named 'source-data'.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "S3:*",
      "Resource": [
        "arn:aws:s3:::spark-project-demo-bucket",
        "arn:aws:s3:::spark-project-demo-bucket/*",
        "arn:aws:s3:::spark-project-demo-bucket/source-data/*"
      ]
    }
  ]
}
```

Create Folder in S3 Bucket and Upload Data

Create folder Info

Use folders to group objects in buckets. When you create a folder, S3 creates an object using the name that you specify followed by a slash (/). This object then appears as folder on the console. [Learn more](#)

Your bucket policy might block folder creation

If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the [upload configuration](#) to upload an empty folder and specify the appropriate settings.

Folder

Folder name

 /

Folder names can't contain "/". [See rules for naming](#)

Server-side encryption Info

Server-side encryption protects data at rest.

The following encryption settings apply only to the folder object and not to sub-folder objects.

Server-side encryption

Do not specify an encryption key

The bucket settings for default encryption are used to encrypt the folder object when storing it in Amazon S3.

Specify an encryption key

The specified encryption key is used to encrypt the folder object before storing it in Amazon S3.

⚠ If your bucket policy requires objects to be encrypted with a specific encryption key, you must specify the same encryption key when you create a folder. Otherwise, folder creation will fail.

Cancel

Create folder

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose [Add files](#) or [Add folder](#).

Files and folders (1 Total, 103.8 MB)

All files and folders in this table will be uploaded.

Find by name		
<input type="checkbox"/>	Name	Folder
<input type="checkbox"/>	survey_results_public.csv	-

Destination Info

Destination

s3://spark-project-demo-bucket/source-data/

▶ Destination details

Bucket settings that impact new objects stored in the specified destination.

▶ Permissions

Grant public access and access to other AWS accounts.

▶ Properties

Specify storage class, encryption settings, tags, and more.

Cancel

Upload

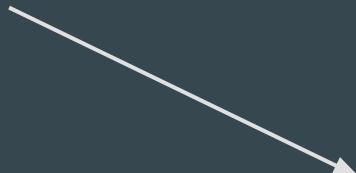
Write Python Code for Spark Processing

```
C: > gattech > ece 4150 > project > final_project_processing.py > ...
1   from pyspark.sql import SparkSession
2   from pyspark.sql.functions import col
3
4   # CHANGE BUCKET NAME and FOLDER ACCORDINGLY
5   S3_DATA_SOURCE_PATH = 's3://spark-project-demo-bucket/source-data/survey_results_public.csv'
6   S3_OUTPUT_SOURCE_PATH = 's3://spark-project-demo-bucket/output-data'
7
8
9  def main():
10    spark = SparkSession.builder.appName('ProjectDemoApp').getOrCreate()
11    all_data = spark.read.csv(S3_DATA_SOURCE_PATH, header=True)
12    print('Total number of records in the source data: %s' % all_data.count())
13    selected_data = all_data.where((col('Country') == 'United States of America') & (col('YearsCode') > 15))
14    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())
15    selected_data.write.mode('overwrite').csv(S3_OUTPUT_SOURCE_PATH)
16    print('Selected data was sucessfully saved to s3: %s' % S3_OUTPUT_SOURCE_PATH)
17
18  if __name__ == '__main__':
19    main()
```

The code block shows a Python script named `final_project_processing.py`. It imports `pyspark.sql` and `pyspark.sql.functions`. The script defines two constants, `S3_DATA_SOURCE_PATH` and `S3_OUTPUT_SOURCE_PATH`, both set to `'s3://spark-project-demo-bucket/source-data/survey_results_public.csv'` and `'s3://spark-project-demo-bucket/output-data'` respectively. A red arrow points from the text `s3://YourS3Bucket/YourData.csv` to the line defining `S3_DATA_SOURCE_PATH`. Another red arrow points from the text `s3://YourS3Bucket/output-data` to the line defining `S3_OUTPUT_SOURCE_PATH`.

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



Network and security [Info](#)

Network

Virtual Private Cloud (VPC)
[vpc-09d95977facea7b3a](#)

Subnet(s) and Availability Zone(s) (AZ)
[subnet-0542360b8c0ddd2ef](#) | us-east-1c

▼ EC2 security groups (firewall)

Primary node

EMR managed security group
[sg-00a591a690727a8aa](#)

Additional security groups

-	SSH	TCP	22	My IP	<input type="text" value="128.61.65.110/32"/>	<input type="button" value="Delete"/>
---	-----	-----	----	-------	---	---------------------------------------

SSH into Hadoop in a terminal containing your key

```
mfeng45@DESKTOP-2B6GU5V ~ | hadoop@ip-172-31-5-145:~ | hadoop@ip-172-31-5-145:~ + - X
A newer release of "Amazon Linux" is available.
Version 2023.4.20240319:
Version 2023.4.20240401:
Run "/usr/bin/dnf check-release-update" for full release and version update info
      #_
      ~\_\_ #####_          Amazon Linux 2023
      ~~ \_\#####\
      ~~ \###|
      ~~ \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
      ~~      V~' '-->
      ~~~   /
      ~~.-. /-
      _/-/_/
      _/m/`_
Last login: Wed Apr 17 03:01:52 2024

EEEEEEEEEEEEEEEEEE MMMMMMMM      MMMMMMMR RRRRRRRRRRRRRR
E:::::::::::E M:::::M      M:::::M R:::::R:::::R
EE:::::EEEEE::::E M:::::M      M:::::M R:::::RRRRRR:::::R
  E:::E     EEEEE M:::::M      M:::::M RR::::R      R::::R
  E:::E     M:::::M:::M      M:::M::::M R:::::R      R::::R
  E:::::EEEEE::::E M:::::M M:::M:::M M:::::M R:::::RRRRRR:::::R
  E:::::::::::E M:::::M M:::M:::M M:::::M R:::::::::::R
  E:::::EEEEE::::E M:::::M M:::::M M:::::M R:::::RRRRRR:::::R
  E:::E     M:::::M M:::M      M:::::M R:::::R      R::::R
  E:::E     EEEEE M:::::M     MMM  M:::::M R:::::R      R::::R
EE:::::EEEEE::::E M:::::M      M:::::M R:::::R      R:::::R
E:::::::::::E M:::::M      M:::::M RR:::::R      R:::::R
EEEEEEEEEEEEEEEEEE MMMMMMMR      MMMMMMR RRRRRR      RRRRRR

[hadoop@ip-172-31-5-145 ~]$
```

Create main.py in the Hadoop Terminal

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

```
hadoop@ip-172-31-10-31:~ x + v - □ ×

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

Amazon S3 > Buckets > spark-project-demo-bucket > output-data/

output-data/

Objects Properties

Objects (5) Info

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

Find objects by prefix Show versions

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	_SUCCESS	-
<input type="checkbox"/>	part-00000-3adff0a7-63bc-485a-9146-df1867876552-c000.csv	csv
<input type="checkbox"/>	part-00001-3adff0a7-63bc-485a-9146-df1867876552-c000.csv	csv
<input type="checkbox"/>	part-00002-3adff0a7-63bc-485a-9146-df1867876552-c000.csv	csv
<input type="checkbox"/>	part-00003-3adff0a7-63bc-485a-9146-df1867876552-c000.csv	csv

O1 E F G H I J K L M N O P Q R

fx Other (please specify);Ask developers I know/work with

1	sonHobby;Con Masterâ€™s Books / Phy NA	NA	20	14 Developer, 10,000 or n i have little	Other (plea	United States of America	USDUnited	130000		
2	not I donâ€™t c Bachelorâ€¢ Books / Phy Technical d NA	NA	24	21 Developer, 10,000 or n i have some	Other (plea	United States of America	USDUnited	102000		
3	sonHobby;Free Associate d Books / Phy Technical d NA	NA	20	15 Developer, 100 to 499	I have a gre;Start a free	United States of America	USDUnited	130000		
4	not Hobby;Free Bachelorâ€¢Schoot (i.e. NA	NA	18	12 Developer, 500 to 999	I have some Start a free	United States of America	USDUnited	150000		
5	not I donâ€™t c Associate d Books / Phy NA	NA	24	22 Other (plea	NA NA	Start a free	United States of America	NA NA		
6	not Hobby;Con Masterâ€™s Books / Phy Technical d Pluralsight	NA	24	20 Developer, 500 to 999	I have some;Start a free	United States of America	USDUnited	160000		
7	not I donâ€™t c Some colle Books / Phy Technical d Udemy;Co	NA	26	22 Developer, 100 to 499	I have a gre;Start a free	United States of America	USDUnited	450000		
8	son I donâ€™t c Masterâ€¢ Books / Phy Technical d NA	NA	22	17 Developer, 100 to 499	I have some Start a free	United States of America	USDUnited	200000		
9	son Hobby;Free Masterâ€¢ Books / Phy Technical d NA	NA	27	25 Developer, 1,000 to 4,1	I have some;Start a free	United States of America	USDUnited	130000		
10	not I donâ€™t c Some colle Books / Phy NA	NA	32	28 Data scient 10 to 19 em	I have a gre;Start a free	United States of America	USDUnited	600000		
11	not Hobby;Con Bachelorâ€¢ Books / Phy Technical d NA	NA	22	22 Developer, 500 to 999	I have a gre;Start a free	United States of America	USDUnited	NA		
12	not Hobby Bachelorâ€¢ Books / Phy NA	NA	26	24 Developer, 100 to 499	I have little	Start a free	United States of America	USDUnited	144000	
13	not Freelance/ Masterâ€¢ Books / Phy NA	Coursera	19	NA DevOps sp	NA NA	Start a free	United States of America	NA NA		
14	not Hobby;Con Bachelorâ€¢ Books / Phy NA	NA	29	15 Developer, 2 to 9 empl	I have a gre;Other	(plea	United States of America	USDUnited	136000	
15	not I donâ€™t c Masterâ€¢ Books / Secondary School (i.e. NA	NA	48	45 Developer, 1,000 to 4,1	I have little	Visit develo	United States of America	USDUnited	400000	
16	not Hobby;Con Secondary Books / Phy NA	NA	30	25 Developer, 2 to 9 empl	I have a gre;Start a free	United States of America	USDUnited	NA		
17	not I donâ€™t c Masterâ€¢ Books / Phy Technical d NA	NA	25	15 Other (plea	500 to 999	I have some Start a free	United States of America	USDUnited	200000	
18	not Hobby Masterâ€¢ Books / Phy Technical d NA	NA	38	32 Other (plea	500 to 999	I have little	Start a free	United States of America	USDUnited	NA
19	not Hobby Bachelorâ€¢ Books / Phy Technical d Other	NA	39	27 Developer, 500 to 999	I have little	Start a free	United States of America	USDUnited	NA	
20	not Hobby;Con Masterâ€¢ Books / Phy Technical d NA	NA	32	32 Developer, 500 to 999	I have a gre;Start a free	United States of America	USDUnited	NA		
21	not Contribute Masterâ€¢ Books / Phy Technical d NA	NA	24	20 Product ma	NA NA	Start a free	United States of America	NA NA		

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!

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...

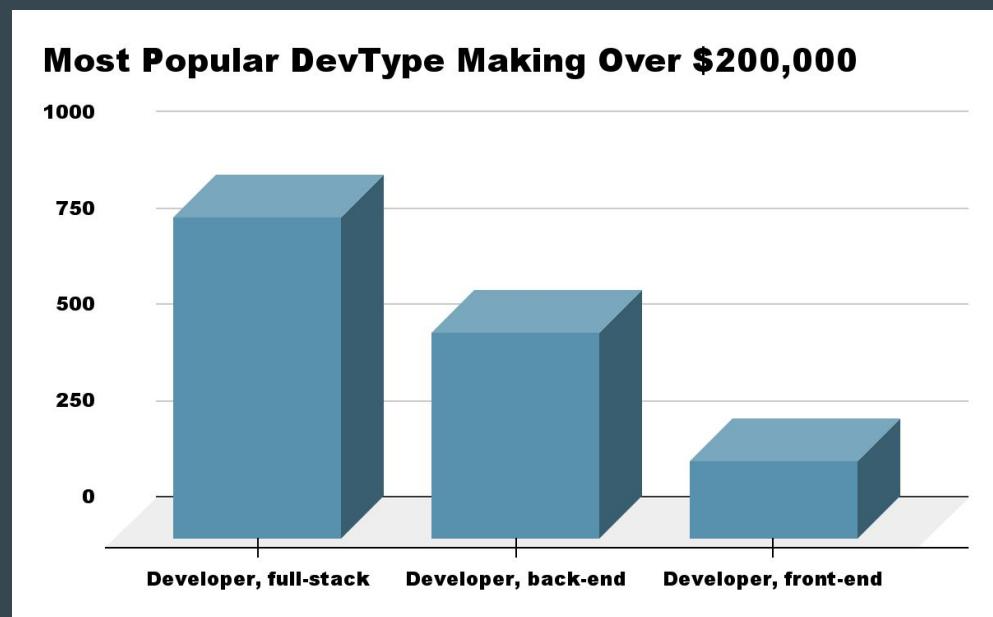
```
hadoop@ip-172-31-10-106:~
```

```
24/04/18 00:43:38 INFO YarnScheduler: Removed TaskSet 25.0, whose tasks have all completed, from pool
24/04/18 00:43:38 INFO DAGScheduler: ResultStage 25 (count at NativeMethodAccessorImpl.java:0) finished in 0.038 s
24/04/18 00:43:38 INFO DAGScheduler: Job 16 is finished. Cancelling potential speculative or zombie tasks for this job
24/04/18 00:43:38 INFO YarnScheduler: Killing all running tasks in stage 25: Stage finished
24/04/18 00:43:38 INFO DAGScheduler: Job 16 finished: count at NativeMethodAccessorImpl.java:0, took 0.041198 s
The total number of people is: 6478
The most common DevType among individuals with a pay range of 200,000+ USD is: Developer, full-stack
Number of people associated with the most common DevType: 837
The second most common DevType: Developer, back-end
Number of people associated with the second most common DevType: 537
The third most common DevType: Developer, front-end
Number of people associated with the third most common DevType: 203
24/04/18 00:43:38 INFO FileSourceStrategy: Pushed Filters: IsNotNull(CompTotal),IsNotNull(Currency),GreaterThan(CompTotal,200000),EqualTo(Currency,USD United States dollar)
24/04/18 00:43:38 INFO FileSourceStrategy: Post-Scan Filters: isnotnull(CompTotal#34),isnotnull(Currency#33),(CompTotal#34 > 200000),(Currency#33 = USD United States dollar)
24/04/18 00:43:38 INFO PathOutputCommitterFactory: No output committer factory defined, defaulting to class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedOutputCommitterFactory
24/04/18 00:43:38 INFO FileSystemOptimizedOutputCommitterFactory: EMR Optimized Committer: ENABLED
24/04/18 00:43:38 INFO FileOutputCommitter: File Output Committer Algorithm version is 2
24/04/18 00:43:38 INFO FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory: false, ignore cleanup failures: true
24/04/18 00:43:38 INFO SQLConfCommitterProvider: Using output committer class org.apache.hadoop.mapreduce.lib.output.FileSystemOptimizedCommitter
24/04/18 00:43:38 INFO FileSystemOptimizedCommitter: Nothing to setup as successful task attempt outputs are written directly
```

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

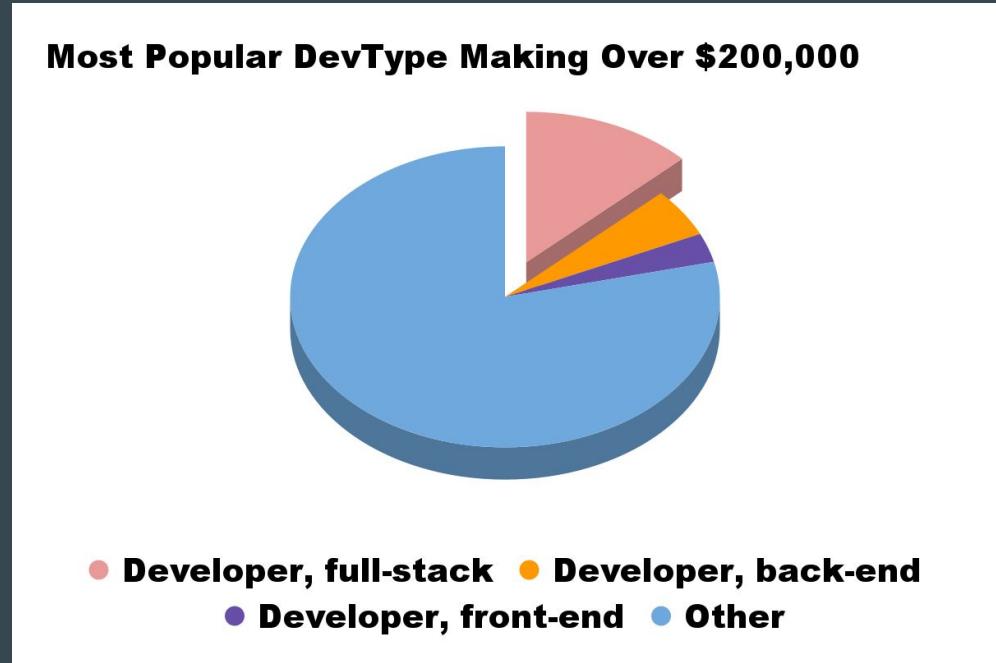
Let's Take a Look at the Visual Representation of Data

- 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/l*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=0CBIQjRxqFwoTCLCf3ND5yYUDFQAAAAAAdAAAAABAI
- https://miro.medium.com/v2/resize:fit:1400/l*hz7npUiTzL9ieTVhCRNHEw.png