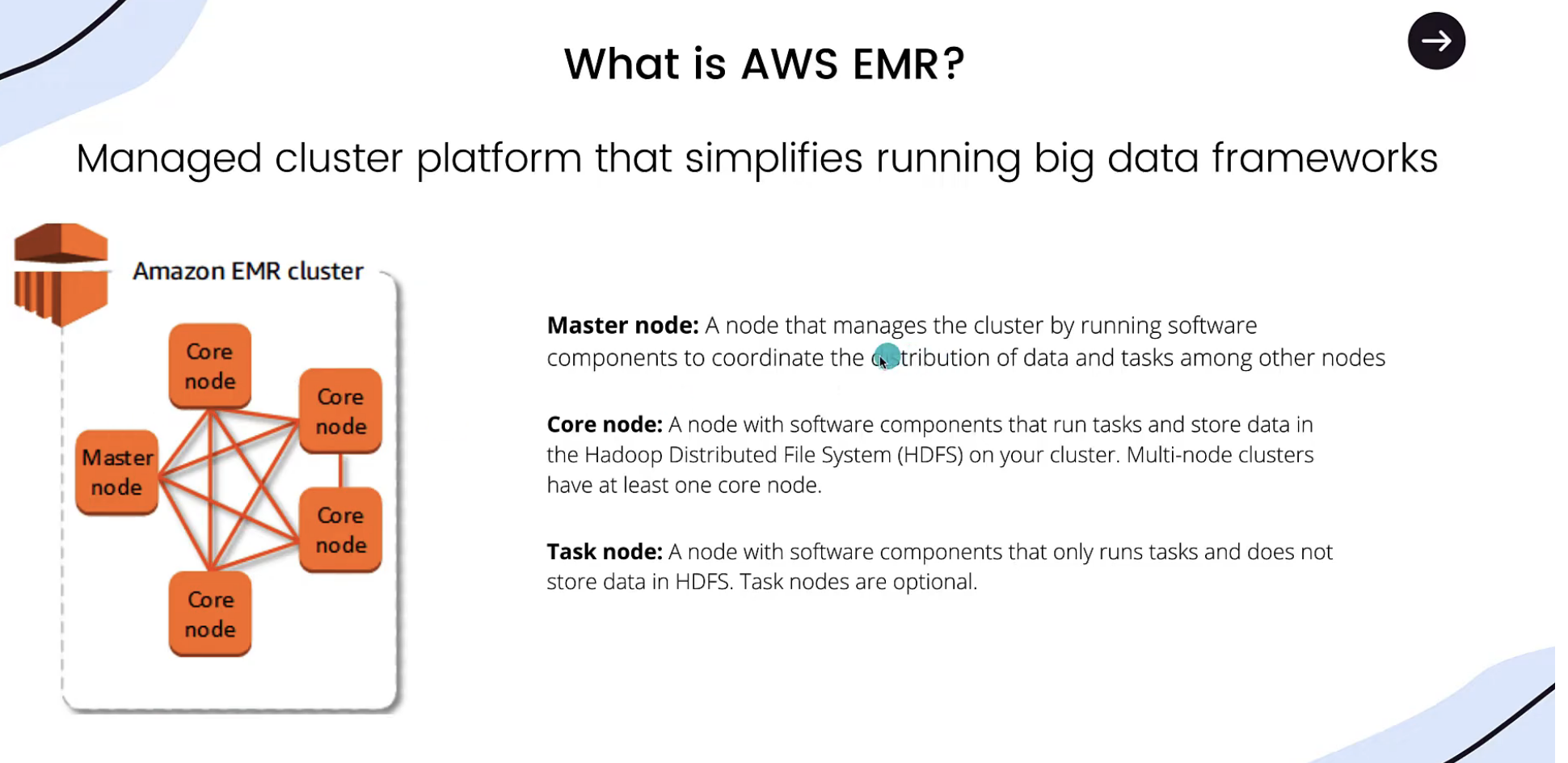
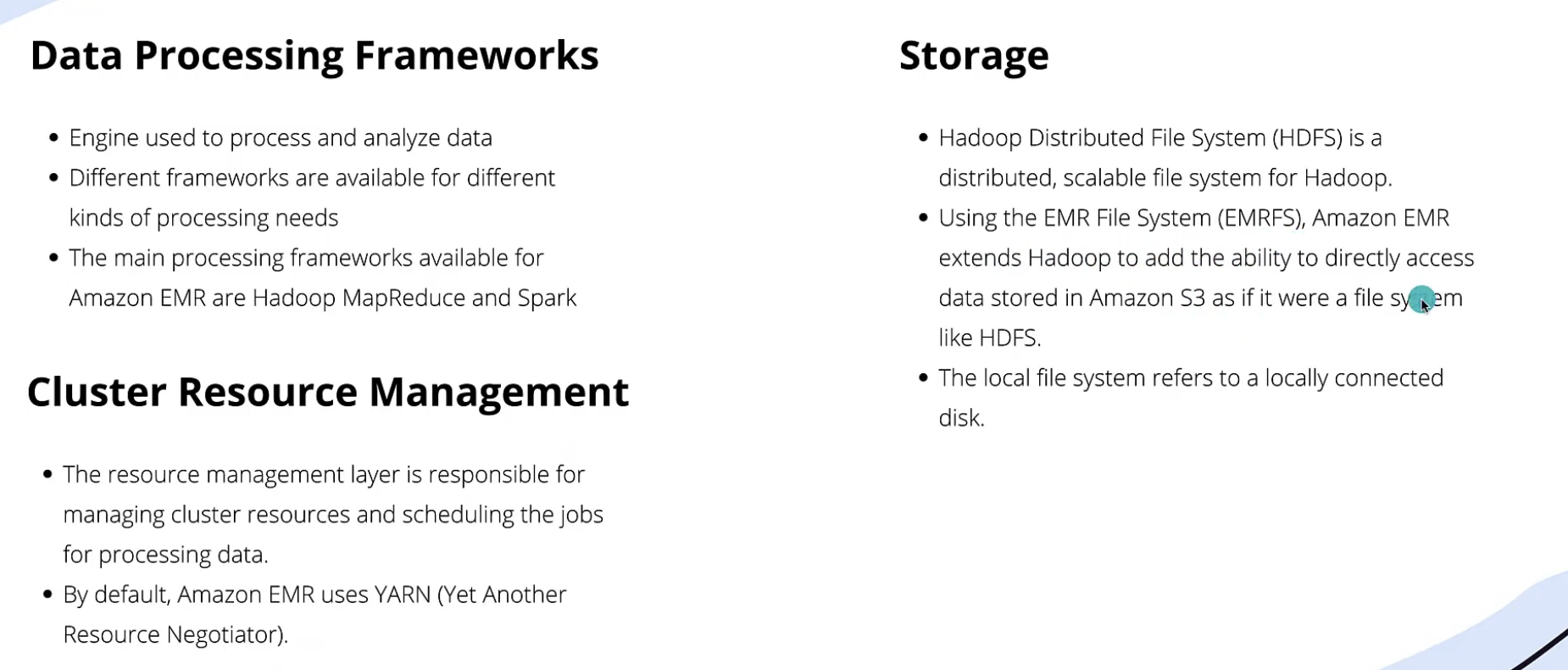
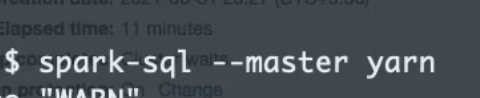
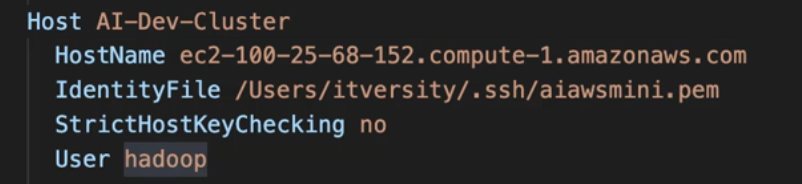
EMR:

* 
* 

<https://github.com/itversity/mastering-emr>

* If we enable Accidental terminate then we can’t delete cluster, we need to disable to terminate
* Bootstrap is nothing but installing few pre-requisite to run our application in EMR
*  or spark-sql will also connect to yarn
* Show databases to see all databases in glue
* To start pyspark 🡪 pyspark
* Spark-shell for scala based
* Elastic IP is nothing but static IP
* 
* We can check bootstrap logs
* 
* A diagram of a code

  Description automatically generated
* A diagram of a network

  Description automatically generated

**Step 1: Ensure Cluster Access**

1. **Get EMR Master Node Public DNS**  
   Use the AWS Management Console or CLI to find the Master Node's Public DNS:

bash

Copy code

aws emr describe-cluster --cluster-id <cluster-id> --query 'Cluster.MasterPublicDnsName'

1. **Open SSH Access**  
   Ensure the security group attached to your EMR cluster allows inbound traffic on port 22 from your IP.
2. **Obtain the Key Pair**  
   Ensure you have the .pem file for the key pair associated with the EMR cluster.

**Step 2: Set Up SSH Configuration**

1. **Edit SSH Config File**  
   Open or create the SSH configuration file ~/.ssh/config:

bash

Copy code

nano ~/.ssh/config

1. **Add EMR Cluster Configuration**  
   Add the following lines (replace placeholders with actual values):

typescript

Copy code

Host emr-master

HostName <master-node-public-dns>

User hadoop

IdentityFile ~/.ssh/<key-pair-name>.pem

1. **Set Permissions**  
   Ensure the private key and SSH config file have the correct permissions:

bash

Copy code

chmod 600 ~/.ssh/<key-pair-name>.pem

chmod 600 ~/.ssh/config

1. **Test Connection**  
   Verify SSH access to the EMR Master Node:

bash

Copy code

ssh emr-master

**Step 3: Install VS Code Remote - SSH Extension**

1. Open VS Code.
2. Go to the Extensions view (Ctrl+Shift+X or Cmd+Shift+X on Mac).
3. Search for **Remote - SSH** and install it.

**Step 4: Connect to EMR Master Node in VS Code**

1. Press F1 or Cmd+Shift+P to open the command palette.
2. Type **Remote-SSH: Connect to Host** and select it.
3. Choose the host configured in your SSH config file (emr-master).
4. VS Code will open a new window connected to the EMR Master Node

A screen shot of a computer

Description automatically generated

In EMR pyspark is present in : A close up of a computer screen

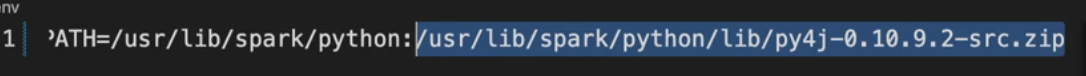
Description automatically generated

Set python path to get pyspark related library

In order to run pyspark code below path needs to be set:

A black background with white text and blue letters

Description automatically generated

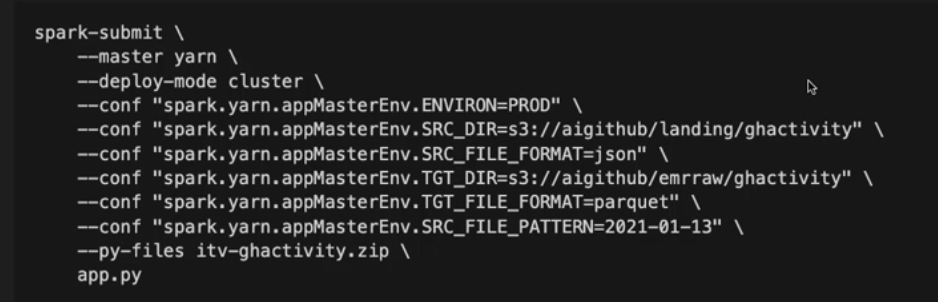


A screen shot of a computer

Description automatically generated

A screen shot of a computer screen

Description automatically generated



CLI command to create EMR cluster and run steps:

aws emr create-cluster \

--name "My cluster" \

--log-uri "s3://aws-logs-816069129520-ap-south-1/elasticmapreduce" \

--release-label "emr-7.4.0" \

--service-role "arn:aws:iam::816069129520:role/emr-role" \

--ec2-attributes '{"InstanceProfile":"emr-role-for-ec2","EmrManagedMasterSecurityGroup":"sg-0b2bcef5ee2dc8414","EmrManagedSlaveSecurityGroup":"sg-0c0f673dbfcd72cba","KeyName":"atul-aws-key","AdditionalMasterSecurityGroups":[],"AdditionalSlaveSecurityGroups":[],"SubnetId":"subnet-04b70fbf3ac9cf368"}' \

--applications Name=Hadoop Name=Hive Name=JupyterEnterpriseGateway Name=Spark \

--configurations '[{"Classification":"hive-site","Properties":{"hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory"}},{"Classification":"spark-hive-site","Properties":{"hive.metastore.client.factory.class":"com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory"}}]' \

--instance-groups '[{"InstanceCount":1,"InstanceGroupType":"MASTER","Name":"Primary","InstanceType":"m5.xlarge","EbsConfiguration":{"EbsBlockDeviceConfigs":[{"VolumeSpecification":{"VolumeType":"gp2","SizeInGB":32},"VolumesPerInstance":2}]}}]' \

--steps '[{"Name":"read\_json\_pysaprk","ActionOnFailure":"CONTINUE","Jar":"command-runner.jar","Properties":"","Args":["spark-submit","--deploy-mode","cluster","--py-files","spark\_app\_code.zip","s3://atul.data/app.py"],"Type":"CUSTOM\_JAR"},{"Name":"read\_json\_pysaprk","ActionOnFailure":"CONTINUE","Jar":"command-runner.jar","Properties":"","Args":["spark-submit","--deploy-mode","cluster","--py-files","s3://atul.data/spark\_app\_code.zip","s3://atul.data/app.py"],"Type":"CUSTOM\_JAR"}]' \

--scale-down-behavior "TERMINATE\_AT\_TASK\_COMPLETION" \

--auto-termination-policy '{"IdleTimeout":3600}' \

--region "ap-south-1"

With modify cluster we can only modify step concurrency only, nothing else

Create Cluster:

import boto3

# Initialize the EMR client

emr\_client = boto3.client('emr', region\_name='ap-south-1')

# Create the cluster

response = emr\_client.run\_job\_flow(

Name="My cluster",

LogUri="s3://aws-logs-816069129520-ap-south-1/elasticmapreduce",

ReleaseLabel="emr-7.4.0",

ServiceRole="arn:aws:iam::816069129520:role/emr-role",

AutoTerminationPolicy={

'IdleTimeout': 3600 # Terminate after 1 hour of idle time

},

ScaleDownBehavior="TERMINATE\_AT\_TASK\_COMPLETION",

Applications=[

{'Name': 'Hadoop'},

{'Name': 'Hive'},

{'Name': 'JupyterEnterpriseGateway'},

{'Name': 'Spark'}

],

Configurations=[

{

'Classification': 'hive-site',

'Properties': {

'hive.metastore.client.factory.class': 'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory'

}

},

{

'Classification': 'spark-hive-site',

'Properties': {

'hive.metastore.client.factory.class': 'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory'

}

}

],

Instances={

'InstanceGroups': [

{

'InstanceCount': 1,

'InstanceRole': 'MASTER', # Specify the role of this instance group

'Name': 'Primary',

'InstanceType': 'm5.xlarge',

'EbsConfiguration': {

'EbsBlockDeviceConfigs': [

{

'VolumeSpecification': {

'VolumeType': 'gp2',

'SizeInGB': 32

},

'VolumesPerInstance': 2

}

]

}

}

],

'Ec2KeyName': 'atul-aws-key',

'EmrManagedMasterSecurityGroup': 'sg-0b2bcef5ee2dc8414',

'EmrManagedSlaveSecurityGroup': 'sg-0c0f673dbfcd72cba',

'Ec2SubnetId': 'subnet-04b70fbf3ac9cf368' # Correct parameter

},

Steps=[

{

'Name': 'read\_json\_pyspark\_2',

'ActionOnFailure': 'CONTINUE',

'HadoopJarStep': {

'Jar': 'command-runner.jar',

'Args': [

'spark-submit',

'--deploy-mode', 'cluster',

'--py-files', 's3://atul.data/spark\_app\_code.zip',

's3://atul.data/app.py'

]

}

}

],

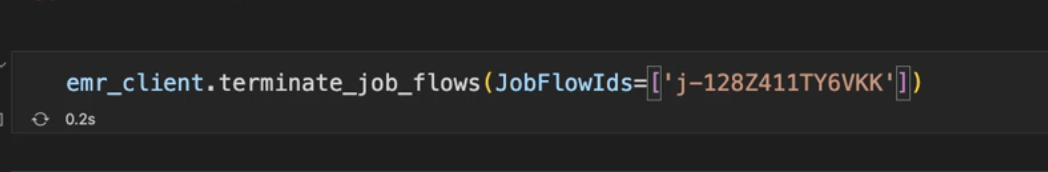
JobFlowRole="emr-role-for-ec2"

)

# Print the response

print("Cluster created with ID:", response['JobFlowId'])

Delete emr cluster using boto 3:



RUN azure blob code using pyspark using additional Jars:

A screenshot of a computer

Description automatically generated

**Cluster Mode:**

**spark-submit --deploy-mode cluster --master yarn --packages com.microsoft.azure:azure-storage:3.0.0,org.apache.hadoop:hadoop-azure:3.3.4 --repositories https://repo1.maven.org/maven2 test.py**

**Client Mode:**

**spark-submit --deploy-mode client --master yarn --packages com.microsoft.azure:azure-storage:3.0.0,org.apache.hadoop:hadoop-azure:3.3.4**

from pyspark.sql import SparkSession

import boto3

spark = SparkSession.builder \

.appName("ReadFromAzureBlob") \

.config("fs.azure.account.key.qatest.blob.core.windows.net", "OFJP0OJ7vOdK+uLY+bB7jwqlQ4P2RCllXIVAbqB7UODTs+jKaPdwkU1dWLJJ4HEM2RUQIqB8/Jpw+AStclETaA==") \

.getOrCreate()

# Azure Blob URL and file path

storage\_account\_name = "icebergpoc"

container\_name = "qatest"

blob\_file\_path = "automation\_dataset/car\_data\_auto.csv"

# Construct the full path

blob\_url = f"wasbs://{container\_name}@{storage\_account\_name}.blob.core.windows.net/{blob\_file\_path}"

# Read the data into a DataFrame

df = spark.read.format("csv").option("header", "true").load(blob\_url)

# Show the data

df.show()

df.write.format("csv").mode("overwrite").save("s3://atul.data/temp\_data\_1")

from pyspark.sql import SparkSession

import boto3

spark = SparkSession.builder \

.master("yarn") \

.appName("dsd") \

.getOrCreate()

# Read the data into a DataFrame

df = spark.read.format("csv").option("header", "true").load("s3://atul.data/customers/customer.csv")

# Show the data

df.show()

df.write.format("csv").mode("overwrite").save("s3://atul.data/temp\_data\_1")

import boto3

emr\_client = boto3.client('emr', region\_name='ap-south-1')

# Create the cluster

response = emr\_client.run\_job\_flow(

Name="My cluster",

LogUri="s3://aws-logs-816069129520-ap-south-1/elasticmapreduce",

ReleaseLabel="emr-7.4.0",

ServiceRole="arn:aws:iam::816069129520:role/emr-role",

BootstrapActions=[

{

"Name": "Custom Bootstrap Action",

"ScriptBootstrapAction": {

"Path": "s3://atul.data/emr\_bootstrap.sh",

"Args": []

}

}

],

AutoTerminationPolicy={

'IdleTimeout': 3600 # Terminate after 1 hour of idle time

},

ScaleDownBehavior="TERMINATE\_AT\_TASK\_COMPLETION",

Applications=[

{'Name': 'Hadoop'},

{'Name': 'Hive'},

{'Name': 'JupyterEnterpriseGateway'},

{'Name': 'Spark'}

],

Configurations=[

{

'Classification': 'hive-site',

'Properties': {

'hive.metastore.client.factory.class': 'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory'

}

},

{

'Classification': 'spark-hive-site',

'Properties': {

'hive.metastore.client.factory.class': 'com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory'

}

}

],

Instances={

'InstanceGroups': [

{

'InstanceCount': 1,

'InstanceRole': 'MASTER', # Specify the role of this instance group

'Name': 'Primary',

'InstanceType': 'm5.xlarge',

'EbsConfiguration': {

'EbsBlockDeviceConfigs': [

{

'VolumeSpecification': {

'VolumeType': 'gp2',

'SizeInGB': 32

},

'VolumesPerInstance': 2

}

]

}

}

],

'Ec2KeyName': 'atul-aws-key',

'EmrManagedMasterSecurityGroup': 'sg-0b2bcef5ee2dc8414',

'EmrManagedSlaveSecurityGroup': 'sg-0c0f673dbfcd72cba',

'Ec2SubnetId': 'subnet-04b70fbf3ac9cf368' # Correct parameter

},

Steps=[

{

'Name': 'aws',

'ActionOnFailure': 'CONTINUE',

'HadoopJarStep': {

'Jar': 'command-runner.jar',

'Args': [

"spark-submit", "--deploy-mode", "cluster", "--packages", "com.microsoft.azure:azure-storage:3.0.0,org.apache.hadoop:hadoop-azure:3.3.4",

"--repositories", "https://repo1.maven.org/maven2", "s3://atul.data/pyspark\_new\_scripts/test\_azure\_blob.py"

]

}

},

{

'Name': 'aws',

'ActionOnFailure': 'CONTINUE',

'HadoopJarStep': {

'Jar': 'command-runner.jar',

'Args': [

"spark-submit", "--deploy-mode", "cluster", "s3://atul.data/pyspark\_new\_scripts/test\_aws\_s3.py"

]

}

}

],

JobFlowRole="emr-role-for-ec2"

)

# Print the response

print("Cluster created with ID:", response['JobFlowId'])

AWS Step Functions:

* It is used to co-ordinate distributed applications
* State Machine: Synonym for pipelines or workflow
* Activities: State Machine are built using activities
* Output of previous step will be stored under attribute
* A screenshot of a computer

  Description automatically generated
* In state machine for loop is nothing but map
* If we are triggering one state machine from another we need to have all IAM policies attached

While creating Step function, select JSON not JSONata

A screenshot of a computer

Description automatically generated

To save variable from response use:

A screenshot of a computer program

Description automatically generated

If you want to process an attribute at runtime end it with .$, you json path to extract

For iteration use MAP

A screenshot of a computer

Description automatically generated

To Use variable $VARIBALE\_NAME

A screenshot of a computer

Description automatically generated

Spark Sql Step:

A screenshot of a computer

Description automatically generated

We can use waiters in EMR to wait for EMR clusters and steps

EMR Serverless:

* We don’t specify resources and presently we can run only hive and spark based jobs in EMR serverless
* <https://github.com/johnny-chivers/emr-serverless>