



**Amazon Web Services**

**Amazon EMR**

# Contents

---

- ✓ Amazon EMR – Basics
- ✓ Creating EMR Cluster
- ✓ Connecting to EMR using SSH
- ✓ Demo 1: Copy files from S3 to EMRFS/HDFS
- ✓ Demo 2: Run a MapReduce Program on EMR
- ✓ Demo 3: Run a Hive Script on EMR
- ✓ Demo 4: Access Hive from Presto on EMR
- ✓ Demo 5: Integrate Hive with DynamoDB
- ✓ Demo 6: Run PySpark Scripts on EMR



# Amazon EMR

---

Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data

- Amazon EMR is a cloud big data platform for running large-scale distributed data processing jobs, interactive SQL queries, and machine learning applications using open-source analytics frameworks such as Apache Spark, Apache Hive, and Presto.
- EMR Uses:
  - Storage: S3
  - Computing: EC2 instances



# Benefits of Amazon EMR

---

- **Reduces the cost of physical infrastructure** as there are no upfront costs involved related to purchasing hardware, software, maintenance and administration. You only pay per use.
- **Resource Utilization** – in EMR storage and compute can be decoupled. You can spin up EC2 instances and cluster when needed without having to worry about data. You can achieve optimum resource utilization as do not have to keep your cluster idle.
- **Saves Time** related to system management and administration.



# How EMR works ?

---

## Upload



Upload your data and processing application to S3.

## Create



Configure and create your cluster by specifying data inputs, outputs, cluster size, security settings, etc.

## Monitor



Monitor the health and progress of your cluster. Retrieve the output in S3.



# EMR Vs. Redshift - When to use what ?

---

- **Use Redshift when**
  - Traditional data warehouse
  - When you need the data relatively hot for analytics such as BI
  - When there is no data engineering team
  - When you require joins
  - When u need a cluster 24X7
- **Use EMR (Spark, Presto, Hive) when**
  - When you don't need a cluster 24X7
  - When elasticity is important (auto scaling on tasks)
  - When cost is important: spots
  - Until a few hundred TB's, In some cases PB's will work.
  - When you want to separate compute and storage (external table + task node + auto scaling)



# Creating an EMR Cluster

---

- Open the EMR Service and Click on 'Create Cluster' button
- Fill the following details:
  - General Configuration
    - Cluster name: DemoEMR1
    - Launch Mode: Cluster
  - Software configuration
    - Core Hadoop (select base on what services you want to run)
  - Hardware configuration
    - Instance Type: m4.large
    - Number of instances: 2 (even 1 is fine)
  - Security and access
    - EC2 key pair: Choose an existing key-pair
- Click on Create Cluster button



# Creating an EMR Cluster

Summary

Application user interfaces

Monitoring

Hardware

Configurations

Events

Steps

Bootstrap actions

Summary

ID: j-1W4C6I0GCATET

Creation date: 2022-01-02 13:21 (UTC+5:30)

Elapsed time: 2 minutes

After last step completes: Cluster waits

Termination protection: Off [Change](#)

Tags: -- [View All](#) / [Edit](#)

Master public DNS: ec2-35-172-223-44.compute-1.amazonaws.com [🔗](#)

[Connect to the Master Node Using SSH](#)

Configuration details

Release label: emr-5.34.0

Hadoop distribution: Amazon 2.10.1

Applications: Hive 2.3.8, Hue 4.9.0, Mahout 0.13.0, Pig 0.17.0, Tez 0.9.2

Log URI: s3://aws-logs-157549686651-us-east-1/elasticmapreduce/ [🔗](#)

EMRFS consistent view: Disabled

Custom AMI ID: --

Application user interfaces

Persistent user interfaces [🔗](#): --

On-cluster user interfaces [🔗](#): Not Enabled [Enable an SSH Connection](#)

Network and hardware

Availability zone: us-east-1f

Subnet ID: [subnet-048f404f74a1da27d](#) [🔗](#)

Master: **Provisioning** 1 m4.large

Core: **Provisioning** 1 m4.large

Task: --

Cluster scaling: Not enabled

Auto-termination: Terminate if idle for 1 hour

Security and access

Key name: NVirginia-KP

EC2 instance profile: EMR\_EC2\_DefaultRole

EMR role: EMR\_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master: [sg-0f9d78a3de6c24505](#) [🔗](#) (ElasticMapReduce-master)

Security groups for Core & Task: [sg-0c0dfe0e6c597560d](#) [🔗](#) (ElasticMapReduce-slave)





# Creating an EMR Cluster

Cluster: DemoCluster16

Running

Running step

Summary

Application user interfaces

Monitoring

Hardware

Configurations

Events

Steps

Bootstrap actions

Summary

ID: j-1W4C6I0GCATET

Creation date: 2022-01-02 13:21 (UTC+5:30)

Elapsed time: 12 minutes

After last step completes: Cluster waits

Termination protection: Off [Change](#)

Tags: — [View All](#) / [Edit](#)

Master public DNS: ec2-35-172-223-44.compute-1.amazonaws.com [Connect to the Master Node Using SSH](#)

Configuration details

Release label: emr-5.34.0

Hadoop distribution: Amazon 2.10.1

Applications: Hive 2.3.8, Hue 4.9.0, Mahout 0.13.0, Pig 0.17.0, Tez 0.9.2

Log URI: s3://aws-logs-157549686651-us-east-1/elasticmapreduce/ [📁](#)

EMRFS consistent view: Disabled

Custom AMI ID: —

Application user interfaces

Persistent user interfaces [🔗](#): [YARN timeline server](#), [Tez UI](#)

On-cluster user interfaces [🔗](#): Not Enabled [Enable an SSH Connection](#)

Network and hardware

Availability zone: us-east-1f

Subnet ID: [subnet-048f404f74a1da27d](#) [🔗](#)

Master: **Running** 1 m4.large

Core: **Running** 1 m4.large

Task: —

Cluster scaling: Not enabled

Auto-termination: Terminate if idle for 1 hour

Security and access

Key name: NVirginia-KP


EC2 instance profile: EMR\_EC2\_DefaultRole

EMR role: EMR\_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master: [sg-0f9d78a3de6c24505](#) [🔗](#) (ElasticMapReduce-master)

Security groups for Core & Task: [sg-0c0dfe0e6c597560d](#) [🔗](#) (ElasticMapReduce-slave)



# Creating an EMR Cluster

Cluster: DemoCluster16 **Waiting** Cluster ready after last step completed.

Summary

Application user interfaces

Monitoring

Hardware

Configurations

Events

Steps

Bootstrap actions

## Summary

ID: j-1W4C6I0GCATET

Creation date: 2022-01-02 13:21 (UTC+5:30)

Elapsed time: 14 minutes

After last step completes: Cluster waits

Termination protection: Off [Change](#)

Tags: — [View All / Edit](#)

Master public DNS: ec2-35-172-223-44.compute-1.amazonaws.com   
[Connect to the Master Node Using SSH](#)

## Configuration details

Release label: emr-5.34.0

Hadoop distribution: Amazon 2.10.1

Applications: Hive 2.3.8, Hue 4.9.0, Mahout 0.13.0, Pig 0.17.0, Tez 0.9.2


Log URI: s3://aws-logs-157549686651-us-east-1/elasticmapreduce/ 

EMRFS consistent view: Disabled

Custom AMI ID: —

## Application user interfaces

Persistent user interfaces : YARN timeline server, Tez UI

On-cluster user interfaces : Not Enabled [Enable an SSH Connection](#)

## Network and hardware

Availability zone: us-east-1f

Subnet ID: [subnet-048f404f74a1da27d](#) 

Master: **Running** 1 m4.large

Core: **Running** 1 m4.large

Task: —

Cluster scaling: Not enabled

Auto-termination: Terminate if idle for 1 hour

## Security and access

Key name: NVirginia-KP

EC2 instance profile: EMR\_EC2\_DefaultRole

EMR role: EMR\_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master: [sg-0f9d78a3de6c24505](#)  (ElasticMapReduce-master)

Security groups for Core & Task: [sg-0c0dfe0e6c597560d](#)  (ElasticMapReduce-slave)



# SSH to EMR Master Instance

---

- When you create an EMR instance, an EC2 cluster with one master and two slaves (or as per your hardware configuration options) is launched.
- You can SSH to master node via PuTTY
- **NOTE:** Make sure SSH is added to the Inbound rules of the Security Group of the master instance.



# Demo 1: Copying files from S3 to EMRFS

---

- Open the EMR cluster and Go to **Steps** Tab
- Add a step to copy files from S3 to EMRFS
  - Step Type: Custom JAR
  - Name: CopyDataFromS3 (can be any name)
  - JAR Location: command-runner.jar
  - Arguments: (entire command should be in a single line)
    - `s3-dist-cp --s3Endpoint=s3.amazonaws.com --src=s3://iquiz.emr/sampleddata/file1.txt --dest=hdfs:///output`
- Check the output in SSH terminal
  - Login to SSH terminal (user: ec2-user)
  - Type the following command to check the output:
    - `hadoop fs -ls hdfs:///output`



## Demo 2: Run a MapReduce program

---

- Add your MR Jar file and input file to an S3 bucket.
- Open the EMR cluster and Go to **Steps** Tab
- Add a step to run an MR program
  - Step Type: Custom JAR
  - Name: MRWordCount (can be any name)
  - JAR Location: <Browse S3 and locate your MR jar file>
  - Arguments:  
`WordCount s3://iquiz.mapreduce/wordcount/input.txt`  
`s3://iquiz.mapreduce/wordcount/output`



## Demo 3: Run a Hive Script

---

- Add your Hive script file and input file to an S3 bucket
- Open the EMR cluster and Go to **Steps** Tab
- Add a step to run a hive script
  - Step Type: Hive Program
  - Name: Churn Modelling
  - Script S3 Location:  
`s3://iquiz.emr/hive/hive_script_churn_modelling.hql`
  - Input S3 Location:  
`s3://iquiz.emr/hive/Churn_Modelling.csv`
  - Output S3 Location:  
`s3://iquiz.emr/hive/output`
- The program will be executed and output files are created in the mentioned output location.



## Demo 4: Using Presto with Hive

---

- Spin up an EMR cluster.
  - Make sure to select an EMR version that has Presto available.
- SSH to the EMR master node as **'hadoop'** user
- Connect to hive CLI and create an external table.
- Query the table from Hive shell.

```
create external table ratings
(userId int, movieId int, rating double, ts bigint)
row format delimited
fields terminated by ','
lines terminated by '\n'
location 's3://iquiz.emr/hive/movielens_ratings/'
```





## Demo 4: Using Presto with Hive

---

- Exit the HiveCLI and connect to Presto CLI Shell

```
$ presto-cli --catalog hive
```

- Query from the table created in Hive

```
$ select * from default.ratings limit 10;
```





## Demo 5: Integrating Hive & DynamoDB

---

- Spin up an EMR cluster with a version containing Hive and SSH to the master node of the cluster.
- Copy the dataset that you want to load to a hive table from s3 to your local machine.
- Connect to Hive shell, create a hive table and load data into it.

```
$ aws s3 cp s3://iquiz.datasets/features-csv/features.csv .
```

```
hive> create table hive_features (id bigint, name string, class  
string, state string, lat double, long double, elev_in_feet  
bigint) row format delimited fields terminated by ',' lines  
terminated by '\n';
```

```
hive> load data local inpath './features.csv' overwrite into  
table hive_features;
```



## Demo 5: Integrating Hive & DynamoDB

---

- Go to DynamoDB service and create a table (Features, key: Id - Number)
- Go back to Hive CLI and create an external hive table with DynamoDB as the storage format.

```
hive> create external table dynamodb_features (id bigint, name string,  
class string, state string, lat double, long double, elev_in_feet bigint)  
row format delimited fields terminated by ',' lines terminated by '\n'  
stored by 'org.apache.hadoop.hive.dynamodb.DynamoDBStorageHandler'  
tblproperties("dynamodb.table.name" = "Features",  
              "dynamodb.column.mapping"=  
"id:Id,name:Name,class:Class,state:State,lat:Latitude,long:Longitude,elev  
_in_feet:Elevation");
```



## Demo 5: Integrating Hive & DynamoDB

---

- Load data into the hive external table which is mapped to DynamoDB. As you load data into this table, the DynamoDB table is automatically populated with the data.

```
hive> insert overwrite table dynamodb_features select id, name, class,  
state, lat, long, elev_in_feet from hive_features limit 100;
```



# Demo 6: Running Spark on EMR

- Load the **scripts** and **data files** to S3
- Spin up an EMR cluster with Spark
- SSH to the EC2 instance of the master node.
- Run the spark-submit command.

- `spark-submit s3://iquiz.emr/pyspark/top10movies.py`
- `sudo spark-submit s3://iquiz.emr/pyspark/top10movies.py`

- NOTE: Make sure the user has write permissions. You can run using **sudo** if you are getting permission issues.

Release  ⓘ

Applications

- ☐ Core Hadoop: Hadoop 2.10.1, Hive 2.3.8, Hue 4.9.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
- ☐ HBase: HBase 1.4.13, Hadoop 2.10.1, Hive 2.3.8, Hue 4.9.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
- ☐ Presto: Presto 0.261 with Hadoop 2.10.1 HDFS and Hive 2.3.8 Metastore
- ☒ Spark: Spark 2.4.8 on Hadoop 2.10.1 YARN and Zeppelin 0.10.0

☐ Use AWS Glue Data Catalog for table metadata ⓘ





THANK  
YOU