

## Data Lake options on AWS

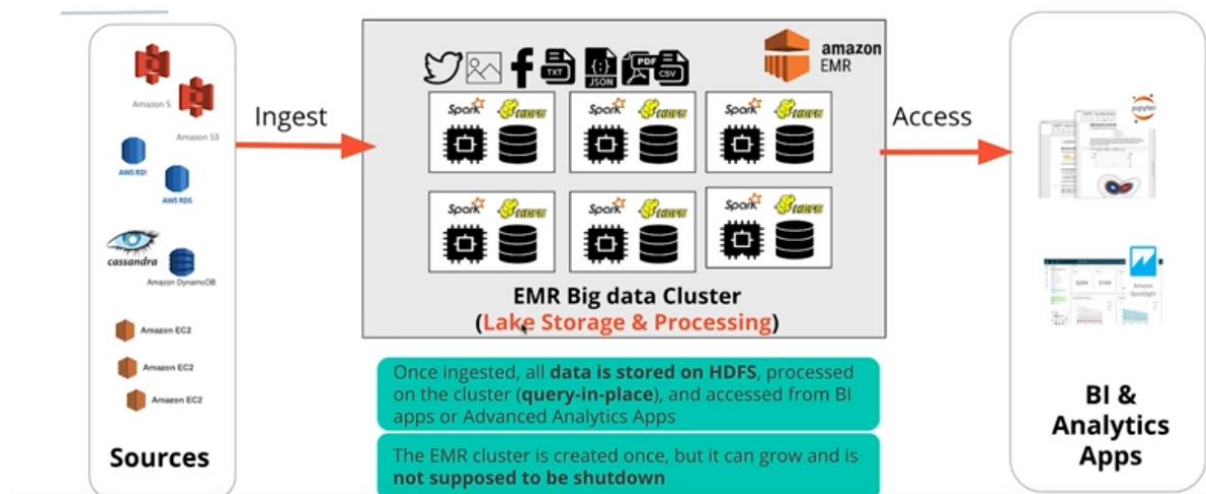
Storage	Processing	AWS-Managed Solution	Vendor-Managed
HDFS	Spark	AWS EMR (HDFS+Spark)	EC2 + Vendor Solution
S3	Spark	AWS EMR (Spark Only)	EC2 + Vendor Solution
S3	Serverless	AWS Athena	Serverless + Vendor Solution

**EMR** is Elastic Map Reduce.

**Vendor Managed** means that there are services from the vendors such as Databricks and Honton Services etc, which provide their services on the top of AWS. These vendors' charges are a bit extra because they make it even easier for the users to use spark.

1. In case of **HDFS + Spark**: HDFS is the distributed Storage and Spark is meant for computation.
2. Had it been **HDFS + Map Reduce** then it would have been the Hadoop styled computation which is 100 times slower than the Spark.
3. In case of **S3 + Spark**: S3 is the Storage and Spark is for computation.

### Data Lake options: AWS EMR (HDFS+Spark)

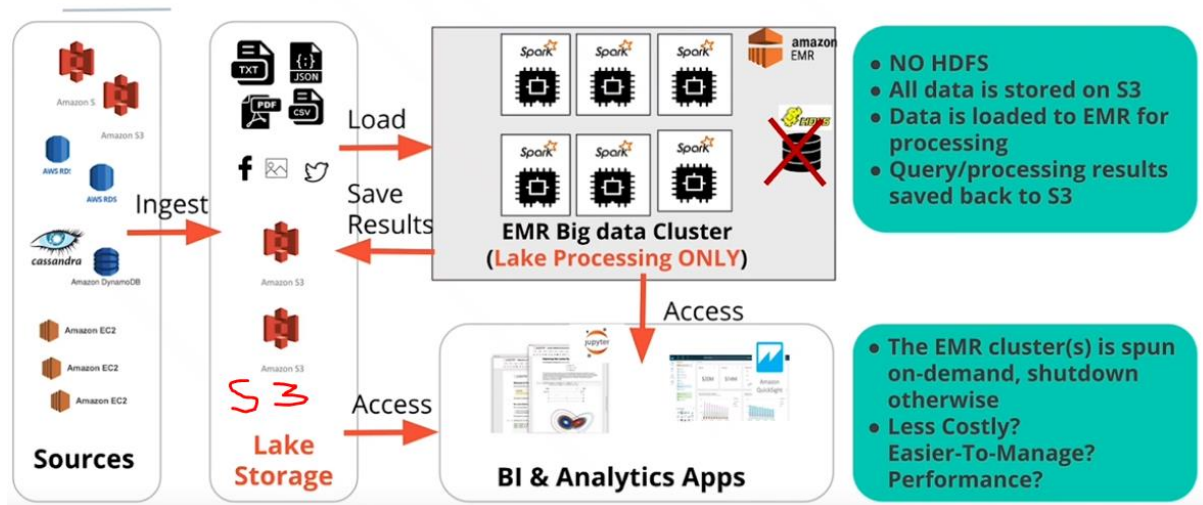


A data lake is created by ingesting the data in their original form such as Json, pdf, csv, structured, text, images etc and is stored in the cluster.

This cluster is made up of multiple nodes or EC2 machines (alias computers or instances). So the data is stored in the cluster for storage as well as computation.

The size of the cluster can be increased based on the requirements, but the data on the cluster is gone once the EMR cluster is shut down. Thus it should run all the time making hdfs costlier. That's why S3 as a storage is preferred as compared to hdfs. This we will learn below.

## Data Lake options: AWS EMR (S3+Spark)



When we use S3 as a storage then we don't use HDFS as a storage, because S3 has several advantages.

The EMR Spark cluster shown above have nodes or EC2 machines meant only for computations or processing, not for storage. Therefore, we can only use them when required, else we can shut them down. This separation of storage and computation from the cluster results in huge saving on account of computation or processing. In this scenario the cluster processing is also called **Lake Processing**.

On ingestion the data is stored in S3 in its original form, and in this case S3 is also known as **Lake Storage**. The data is loaded from S3 to the cluster as and when required for computation or processing. After computation or processing, the data result is saved back into S3 (In case of Batch Processing).

**Cost:** S3+Spark is a bit cheaper than HDFS +Spark; because it is cheaper to store data on S3 as compared to the EC2 machines of the cluster.

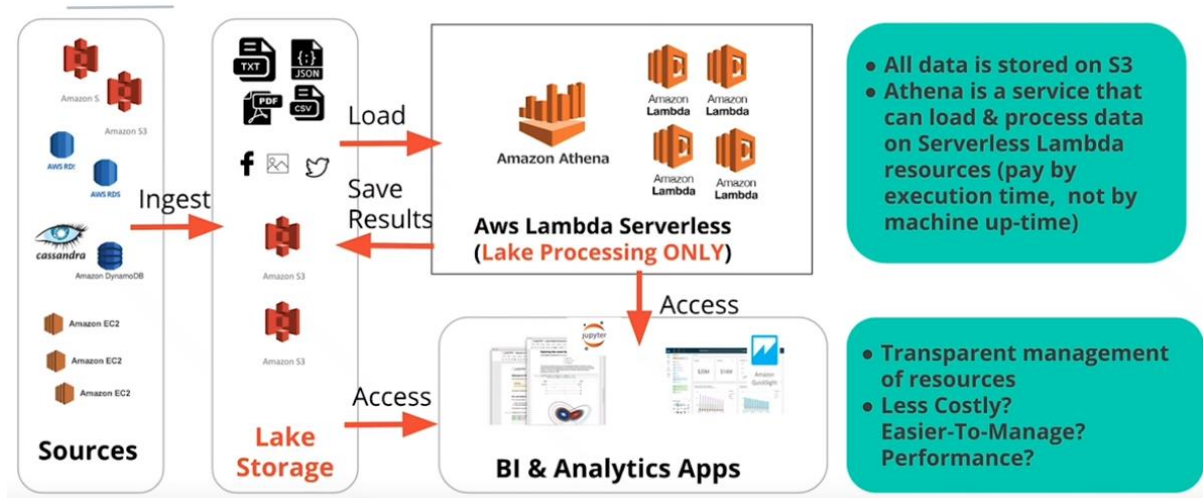
**Ease of Managing:** S3 + Spark is much easier to manage. You can shut down the cluster whenever you want. Decrease the cluster size depending upon the requirement. Hassle free storage on S3, which never shuts down.

**Performance:** The performance is a bit slower in case of S3 + Spark because the data is not stored in cluster; however, since the speed within cloud system (Networking) is improving day by day, thus performance is not that big an issue.

For vendor managed system such as databricks, it can be (S3 + DataBricks) in place of (S3 + EMR Spark Cluster) .

The BI & Analytics application can access the data directly either from S3 or from cluster (in case of streaming)\* . *\*Need more readings for the streaming data access*

## Data Lake options: AWS Athena



In this case, **AWS Lambda** Services is used for computation along with S3 for Storage. Read below for lambda services. **Athena** can also be used to query the data on S3 with schema on read.

For more on AWS lambda <https://lumigo.io/blog/aws-lambda-vs-ec2/>

**History of Serverless Architecture:** AWS Lambda was launched to eliminate infrastructure management of computing. It enables developers to concentrate on writing the function code without having to worry about provisioning infrastructure. We don't need to do any forecasting of the resources (CPU, Memory, Storage, etc.). It can scale resources up and down automatically. It is an epitome of Serverless Architecture. However; for setting up a simple application on EC2, first, we need to forecast how much capacity the application would need. Then, we have to configure it to spin up the Virtual Machine.

In the case of Lambda, you won't need to worry about the provisioning of VMs, software, scaling or load balancing. It is all handled by the Lambda service. We just need to compile the code and deploy to Lambda service. Scaling is automated. We just need to configure how many max concurrent executions we want to allow for a function. Load balancing will be handled by Lambda itself.

So here, we can see **Lambda is a clear winner**.

Unlike EC2, it is charged based on the execution time and memory used. **Thus, it is best suited if we need to use cluster for short period of time in a day.**

### AWS ATHENA

Amazon Athena is defined as "an interactive query service that makes it easy to analyse data directly in Amazon Simple Storage Service (Amazon S3) using standard SQL." So, it's another SQL query engine for large data sets stored in S3. This is very similar to other SQL query engines, such as Apache Drill. But unlike Apache Drill, Athena is limited to data only from Amazon's own S3 storage service. However, Athena is able to query a variety of file formats, including, but not limited to CSV, Parquet, JSON, etc.

## Data Lake issues

---

- Data Lake is prone to being a chaotic **data garbage dump**. Efforts are being made to put measures and practices like detailed metadata to reduce this risk.
  - Since a major feature of the data lake is the wide accessibility of cross-department data and external data of relevance, sometimes **data governance** is not easy to implement. Telling who has access to what is hard.
  - Finally, it is still sometimes unclear, per given case, whether a data lake should **replace, offload or work in parallel** with a data warehouse or data marts. In all cases, dimensional modelling, even in the context of a data lake, continues to remain a valuable practice.
- 

## LAUNCH EMR CLUSTER

### Step 1: Set Up Prerequisites for Your Sample Cluster

<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-prerequisites.html>

### Step 2: Launch Your Sample Amazon EMR Cluster

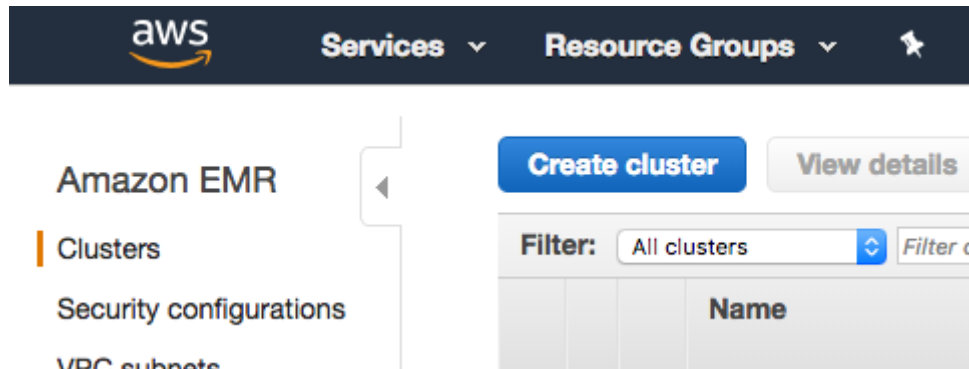
<https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs-launch-sample-cluster.html>

-----

## PRACTICE: Launch EMR Cluster and Notebook

Follow the instructions below to launch your EMR cluster and notebook.

- Go to the [Amazon EMR Console](#)
- Select "Clusters" in the menu on the left, and click the "Create cluster" button.



### Step 1: Configure your cluster with the following settings:


- Release: `emr-5.20.0` or later
- Applications: `Spark`: Spark 2.4.0 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.0
- Instance type: `m3.xlarge`
- Number of instance: `3`
- EC2 key pair: `Proceed without an EC2 key pair` or feel free to use one if you'd like  
You can keep the remaining default setting and click "Create cluster" on the bottom right.

# Create Cluster - Quick Options [Go to advanced options](#)

## General Configuration

**Cluster name**

☒ **Logging** ⓘ

**S3 folder**  

**Launch mode** ☒ **Cluster** ⓘ ☐ **Step execution** ⓘ

## Software configuration

**Release**  ⓘ

**Applications**

- ☐ Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.4, Hue 4.3.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.1
- ☐ HBase: HBase 1.4.8 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.4, Hue 4.3.0, Phoenix 4.14.0, and ZooKeeper 3.4.13
- ☐ Presto: Presto 0.214 with Hadoop 2.8.5 HDFS and Hive 2.3.4 Metastore
- ☒ Spark: Spark 2.4.0 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.0

☐ Use AWS Glue Data Catalog for table metadata ⓘ

## Hardware configuration

**Instance type**

**Number of instances**  (1 master and 2 core nodes)

## Security and access

**EC2 key pair**  ⓘ [Learn how to create an EC2 key pair](#)

**Permissions** ☒ **Default** ☐ **Custom**

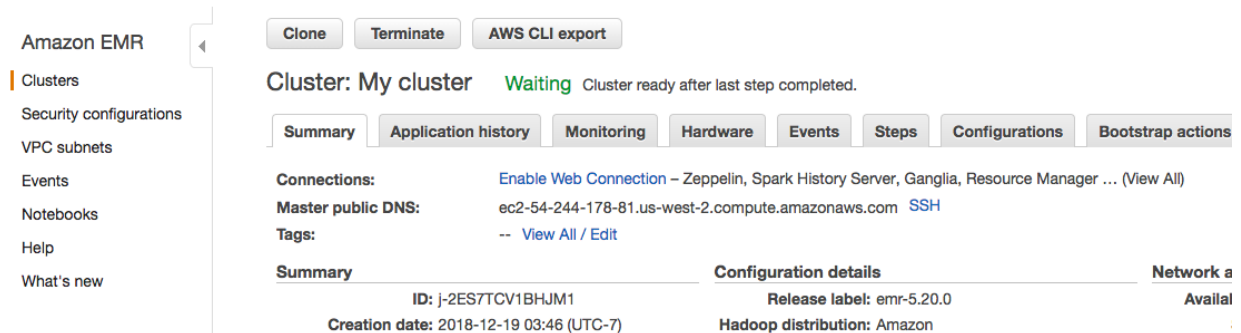
Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

**EMR role** [EMR\\_DefaultRole](#) ⓘ

**EC2 instance profile** [EMR\\_EC2\\_DefaultRole](#) ⓘ

## Step 2: Wait for Cluster "Waiting" Status

Once you create the cluster, you'll see a status next to your cluster name that says *Starting*. Wait a short time for this status to change to *Waiting* before moving on to the next step.



The screenshot shows the Amazon EMR console interface. On the left is a navigation menu with options: Amazon EMR, Clusters, Security configurations, VPC subnets, Events, Notebooks, Help, and What's new. The 'Clusters' section is selected. The main content area displays the details for a cluster named 'My cluster', which is in a 'Waiting' status. At the top, there are buttons for 'Clone', 'Terminate', and 'AWS CLI export'. Below the cluster name, a message states 'Cluster ready after last step completed.' A series of tabs are visible: Summary, Application history, Monitoring, Hardware, Events, Steps, Configurations, and Bootstrap actions. The 'Summary' tab is active, showing the following information:

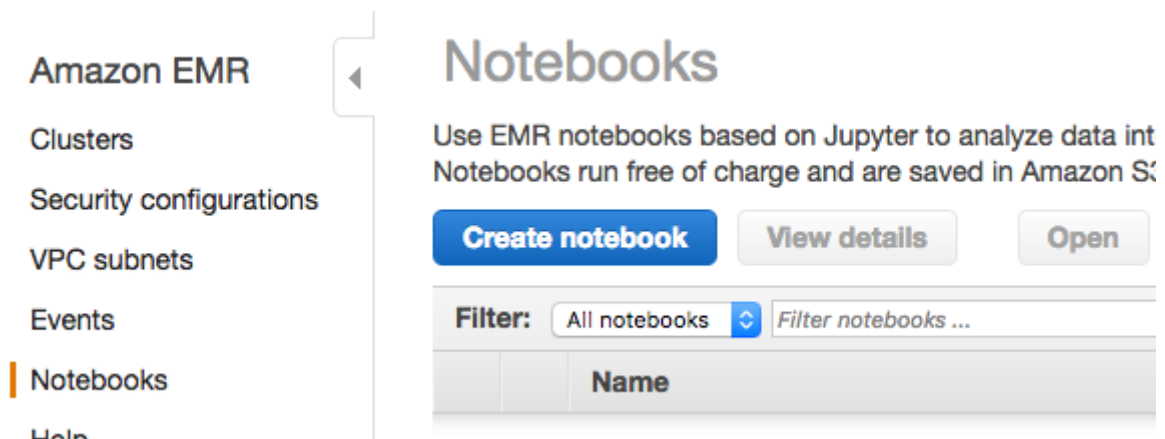
- Connections:** [Enable Web Connection](#) – Zeppelin, Spark History Server, Ganglia, Resource Manager ... (View All)
- Master public DNS:** ec2-54-244-178-81.us-west-2.compute.amazonaws.com [SSH](#)
- Tags:** -- [View All / Edit](#)

Summary	Configuration details	Network a
ID: j-2ES7TCV1BHJM1	Release label: emr-5.20.0	Availal
Creation date: 2018-12-19 03:46 (UTC-7)	Hadoop distribution: Amazon	

## Step 3: Create Notebook

Now that you launched your cluster successfully, let's create a notebook to run Spark on that cluster.

Select "Notebooks" in the menu on the left, and click the "Create notebook" button.



The screenshot shows the Amazon EMR console with the 'Notebooks' section selected in the left-hand navigation menu. The main content area is titled 'Notebooks' and includes a description: 'Use EMR notebooks based on Jupyter to analyze data int' and 'Notebooks run free of charge and are saved in Amazon S3'. Below the description are three buttons: 'Create notebook' (highlighted in blue), 'View details', and 'Open'. A filter section shows 'Filter: All notebooks' with a dropdown arrow and a text input field labeled 'Filter notebooks ...'. Below this is a table header with a single column labeled 'Name'.

## Step 4: Configure your notebook


- Enter a name for your notebook
- Select "Choose an existing cluster" and choose the cluster you just created



- Use the default setting for "AWS service role" - this should be "EMR\_Notebooks\_DefaultRole" or "Create default role" if you haven't done this before.  
You can keep the remaining default settings and click "Create notebook" on the bottom right.


## Create notebook


### Name and configure your notebook


Name your notebook, choose a cluster or create one, and customize configuration options if desired. [Learn more](#) 



**Notebook name\***   
Names may only contain letters (a-z), numbers (0-9), hyphens (-), or underscores (\_).



**Description**   
256 characters max.


**Cluster\*** ☒ Choose an existing cluster  
 My cluster [j-2ES7TCV1BHJM1](#) 

☐ Create a cluster 

**Security groups** ☒ Use default security groups   
☐ Choose security groups (vpc-2cc4274b)

**AWS service role\***   

**Notebook location\*** Choose an S3 location in us-west-2   
 

► **Tags** 

\* Required

[Cancel](#)

[Create notebook](#)

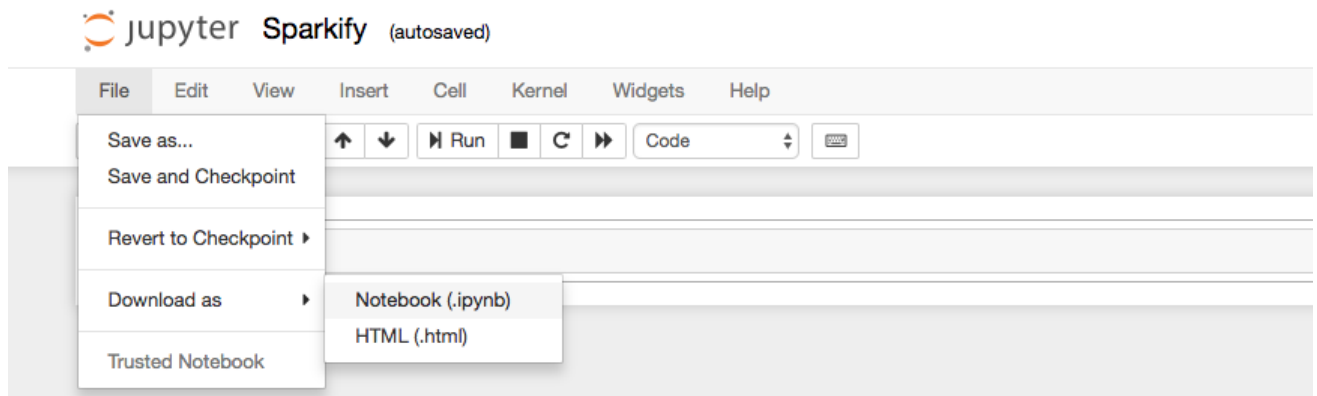
## Step 5: Wait for Notebook "Ready" Status, Then Open

Once you create an EMR notebook, you'll need to wait a short time before the notebook status changes from *Starting* or *Pending* to *Ready*. Once your notebook status is *Ready*, click the "Open" button to open the notebook.



When you are finished with your notebook, click **File** > **Download as** > **Notebook** to download it to your computer. On your local computer, create a git repository including this

notebook and a README file. Submit the URL to your github repository to submit this project. See more details in the [Sparkify Project Overview page](#).



For more information on EMR notebooks, click [here](#).

## Pricing - Be Careful!

From this point on, AWS will charge you for running your EMR cluster. See details on this and how to manage your resources to avoid unexpected costs in the "Managing Resources" section at the end of this lesson.

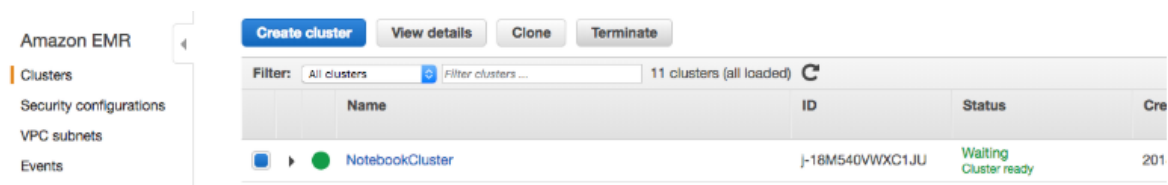
---

## Pricing - Be Careful!

From this point on, AWS will charge you for running your EMR cluster. You can find the details on the [EMR Pricing here](#). If you run your cluster for a week with the settings specified earlier (3 instances of `m3.xlarge`), you should expect costs to be around \$30, which should be covered in the amount of free promotional credits you have received from AWS as a Udacity student. Most importantly, remember to terminate your cluster when you are done. Otherwise, your cluster might run for a day, week, month, or longer without you remembering, and you'll wind up with a large bill!

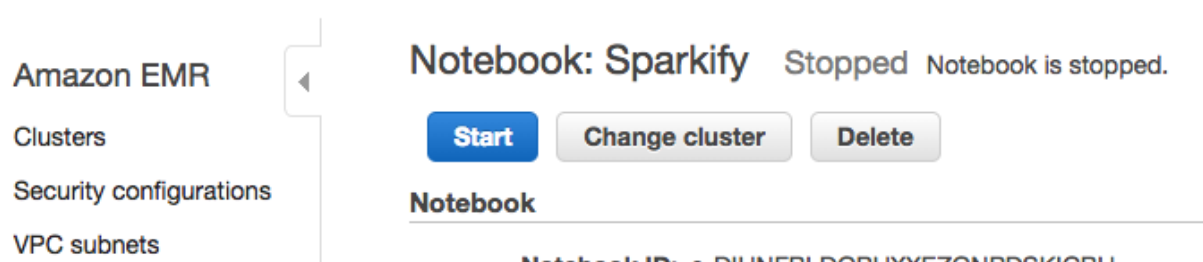
## Terminate Your Cluster

You can terminate your cluster while still keeping the Jupyter notebook you created. In EMR, your EMR cluster and EMR notebook are decoupled (so you can reattach your notebook to a different cluster at any time)! To terminate your cluster, click "Clusters" in the menu on the left, check the box next to your cluster to select, and click the "Terminate" button.



## Change Cluster for Notebook

If you still have a notebook on EMR and terminated the cluster it was connected to, you can still run that notebook at any time by creating another cluster (following the instructions in the previous section). Once you have the new cluster launched and in "waiting" status, click "Notebooks" in the menu on the left and click on the name of your notebook. Then click the "Change cluster" button.



Select your new cluster. Once your notebook reaches "Ready" status, you can now run this notebook on your new cluster.

## Change cluster

Attach your notebook to a new cluster. Choose Security groups if desired. [Learn more](#)

Notebook name Sparkify

Cluster\* ☒ Choose an existing cluster

My cluster j-EM5XHP5UTFAI

☐ Create a cluster

Security groups ☒ Use default security groups

☐ Choose security groups (vpc-2cc4274b)

## Delete Your Notebook

When you've finished with your project and downloaded your notebook, you can delete your notebook from EMR by selecting "Notebooks" in the menu on the left, selecting your notebook, and then clicking "Delete." Make sure to terminate the clusters you launched for this as well.

Amazon EMR

Clusters

Security configurations

VPC subnets

Events

Notebooks

Help

What's new

## Notebooks

Use EMR notebooks based on Jupyter to analyze data interactively with live code, narrative text, visualizations charge and are saved in Amazon S3 independently of clusters. Standard billing for clusters and Amazon S3 ap

Filter: All notebooks Filter notebooks ... 1 notebook (all loaded)

	Name	Status
<input checked="" type="radio"/>	Sparkify	Ready

## Delete S3 buckets

AWS charges primarily for running instances, so most of the charges will cease once you stop the cluster. However, there are smaller storage charges that continue to accrue if you don't delete your buckets. To delete your buckets, go to the [Amazon S3 console](#). Choose the bucket you want to delete from the list, so that the whole bucket row is selected. Choose delete bucket, type the name of the bucket, and click "Confirm."

For more information about deleting folders and buckets, go to [How Do I Delete an S3 Bucket](#) in the *Amazon Simple Storage Service Getting Started Guide*.

You can view your billing information at any time by clicking on your account name on the upper right corner of the console and go to **My Billing Dashboard**.

My Account

My Organization

My Billing Dashboard

My Security Credentials

Sign Out

