

# AMAZON ECO-SYSTEM (AWS)

Subject – Big Data Analytics

Group 7

Members:-

*Tanmay Ingle*

*Bhaskar Akula*

## Contents:-

Sr. No.	Title	Page No.
1	AWS	3
2	S3	6
3	EC2	11
4	EMR	13
5	Amazon Machine Learning	20
6	DynamoDB	25
7	Elastic Beanstalk	28
8	References	30

## **Amazon Web Services**

Over last decade, the normal business application architecture has evolved from an old desktop centric installation to loosely coupled web services and service oriented architectures. Virtualization has taken over in an attempt to reduce operating costs and increase the reliability of enterprise wide IT. Grid Computing has made a new class of business intelligence and analytics possible that was costly earlier and time consuming. The speed of innovation in the introduction of new products has changed how market works. Software as a service has been widely accepted.

Amazon has a long history of using a decentralized IT infrastructure. This arrangement has enabled their development teams to access compute and storage resources on demand, and it has increased overall productivity and agility. By 2005, Amazon had spent over a decade and millions of dollars building and managing the large-scale, reliable, and efficient IT infrastructure that powered one of the world's largest online retail platforms. Amazon launched Amazon Web Services (AWS) so that other organizations could benefit from Amazon's experience and investment in running a large-scale distributed, transactional IT infrastructure. AWS has been operating since 2006, and today serves hundreds of thousands of customers worldwide. Today Amazon.com runs a global web platform serving millions of customers and managing billions of dollars' worth of commerce every year. Using AWS, you can requisition compute power, storage, and other services in minutes and have the flexibility to choose the development platform or programming model that makes the most sense for the problems they're trying to solve. You pay only for what you use, with no up-front expenses or long-term commitments, making AWS a cost-effective way to deliver applications.

Here are some of examples of how organizations, from research firms to large enterprises, use AWS today:

- A large enterprise quickly and economically deploys new internal applications, such as HR solutions, payroll applications, inventory management solutions, and online training to its distributed workforce.
- An e-commerce website accommodates sudden demand for a "hot" product caused by viral buzz from Facebook and Twitter without having to upgrade its infrastructure.

- A pharmaceutical research firm executes large-scale simulations using computing power provided by AWS.
- Media companies serve unlimited video, music, and other media to their worldwide customer base.

## What makes AWS UNIQUE?

AWS is readily distinguished from other vendors in the traditional IT computing landscape because it is:

- ***Flexible:*** AWS enables organizations to use the programming models, operating systems, databases, and architectures with which they are already familiar. In addition, this flexibility helps organizations mix and match architectures in order to serve their diverse business needs.
- ***Cost-effective:*** With AWS, organizations pay only for what they use, without up-front or long-term commitments.
- ***Scalable and elastic:*** Organizations can quickly add and subtract AWS resources to their applications in order to meet customer demand and manage costs.
- ***Secure:*** In order to provide end-to-end security and end-to-end privacy, AWS builds services in accordance with security best practices, provides the appropriate security features in those services, and documents how to use those features.
- ***Experienced:*** When using AWS, organizations can leverage Amazon's more than fifteen years of experience delivering large-scale, global infrastructure in a reliable, secure fashion.

AWS consists of various products. However the report focuses on the following products:

- *S3*
- *EC2*
- *EMR*
- *Amazon Machine Learning*
- *Amazon DynamoDB*
- *Amazon Beanstalk*

## **Amazon S3 – Simple Storage Service**

**Problem:** - Storage, distribution and management of all the data nowadays is a very big challenge.

Running applications, delivering content to users, hosting high traffic websites, backing up databases, emails etc. needs a lot of storage and this demand keeps growing every day. Also building your own repository and maintaining it is expensive. First we need to buy racks and racks of dedicated hardware, then we need to hire staff and set complex processes to make sure storage is performing well and backed properly. Adding more capacity i.e. buying more servers, hard drives is expensive and predicting how much capacity you'll need in future is difficult. And if all this is done in an incorrect way, you will either not have adequate storage or you will overspend and have excess capacity which will sit idle.

**Solution:** Amazon's answer to this problem is S3. Amazon Simple Storage Service (Amazon S3), provides developers and IT teams with secure, durable, highly-scalable object storage. Amazon S3 is easy to use, with a simple web services interface to store and retrieve any amount of data from anywhere on the web. With Amazon S3, you pay only for the storage you actually use. There is no minimum fee and no setup cost. You don't need to estimate how much capacity you need in future and can store as much data as you want and access it when you need it.

Amazon S3 can be used alone or together with other AWS services such as Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), and Amazon Glacier, as well as third party storage repositories and gateways. Amazon S3 provides cost-effective object storage for a wide variety of use cases including cloud applications, content distribution, backup and archiving, disaster recovery, and big data analytics.

It makes copies of the data at multiple locations so that loss of data is minimized. There is no minimum cost and thus you just pay for how much you use. And for the data that you don't need very often, S3 helps reduce cost by adding it automatically to Amazon Glacier which is a low cost archive service.

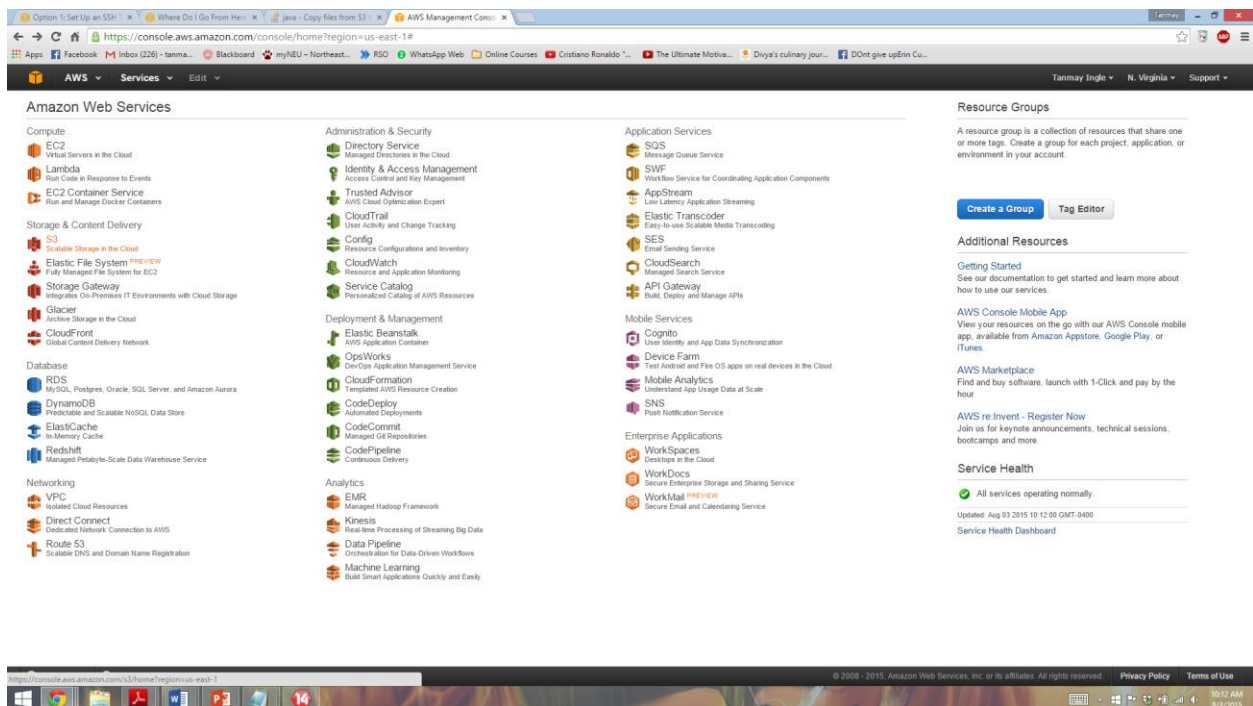
It has flexibility to control who can access the data with identity and access policies, access control, bucket policies etc.

Amazon S3 stores data as objects within resources called "buckets." You can store as many objects as you want within a bucket, and write, read, and delete objects in your bucket. Objects can be up to 5 terabytes in size.

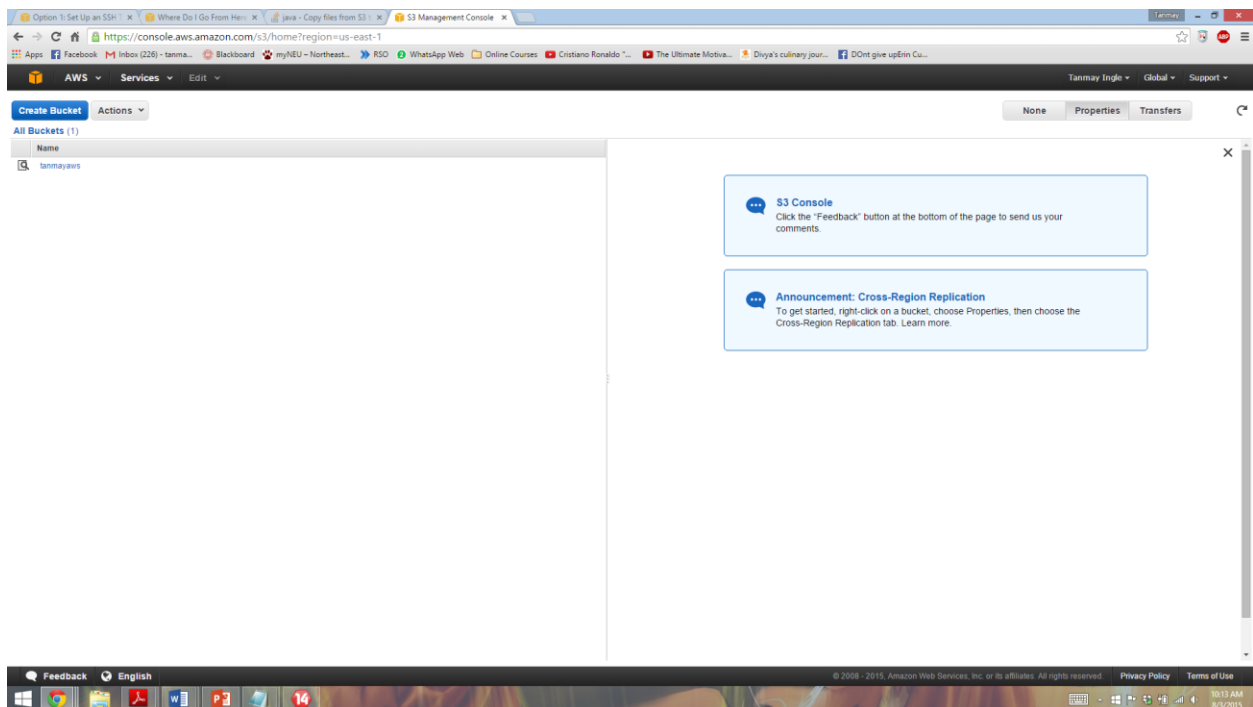
You can control access to the bucket (who can create, delete, and retrieve objects in the bucket for example), view access logs for the bucket and its objects, and choose the AWS region where a bucket is stored to optimize for latency, minimize costs, or address regulatory requirements.

## DEMO

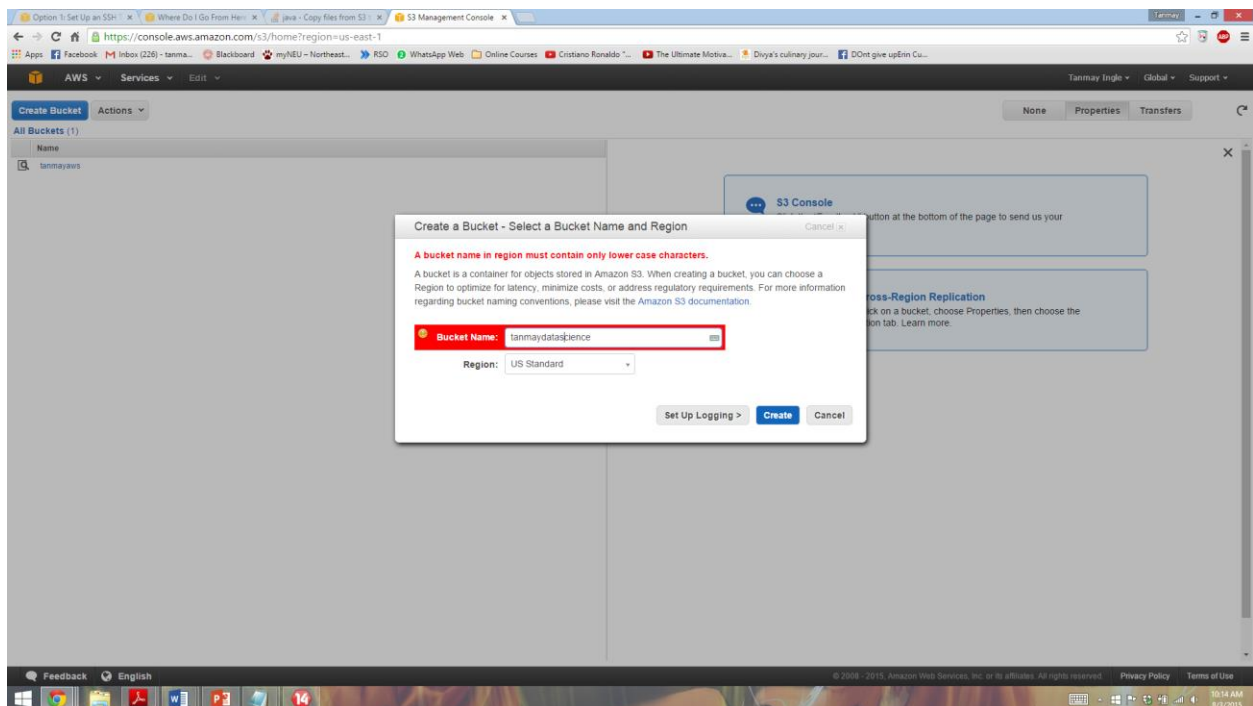
Sign in to your aws console and select S3.



In S3 everything is stored in the form of buckets. So we need to create a bucket first. As you can see I have already created a bucket named 'tanmayaws'.

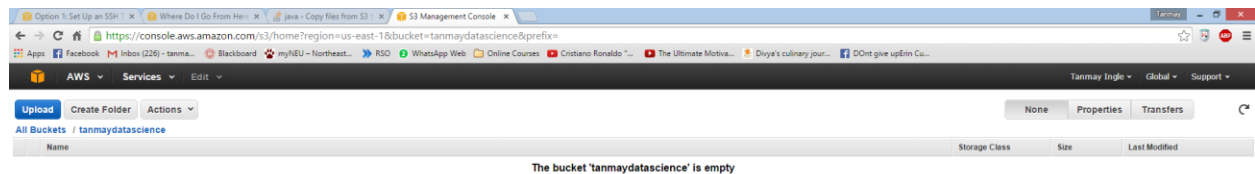
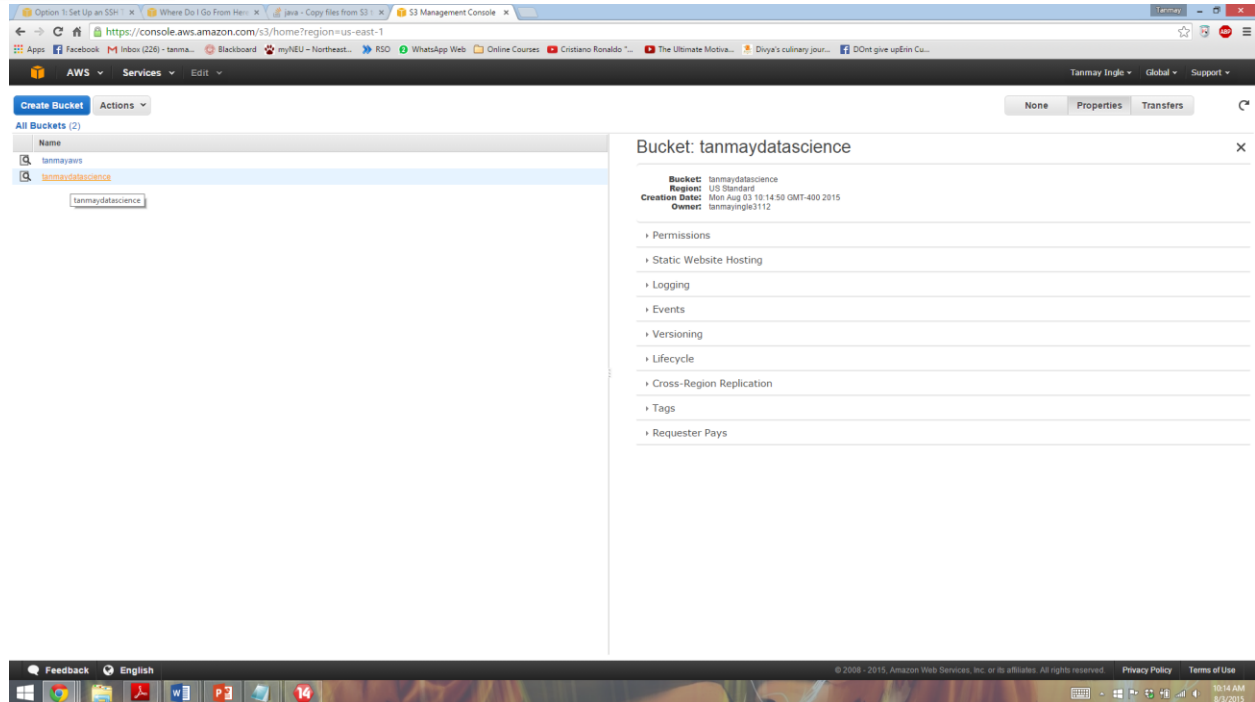


To create a bucket just hit create bucket tab and give a name to your bucket. It should be in lowercase.

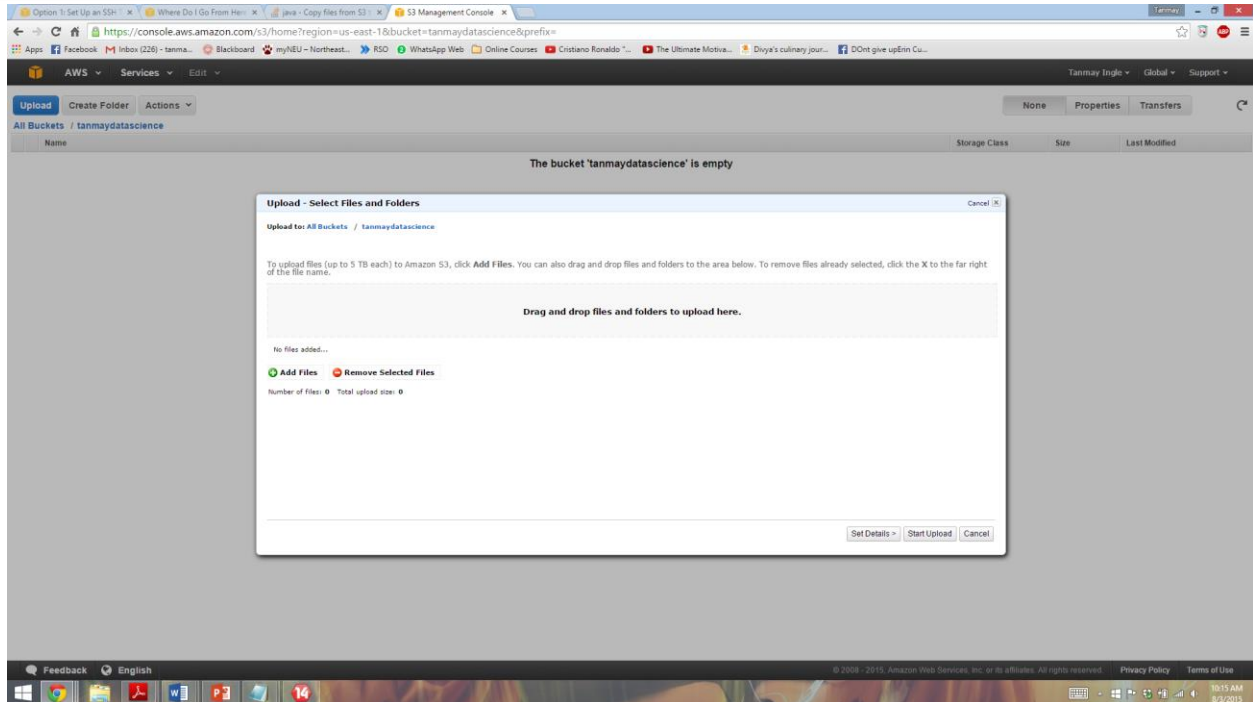




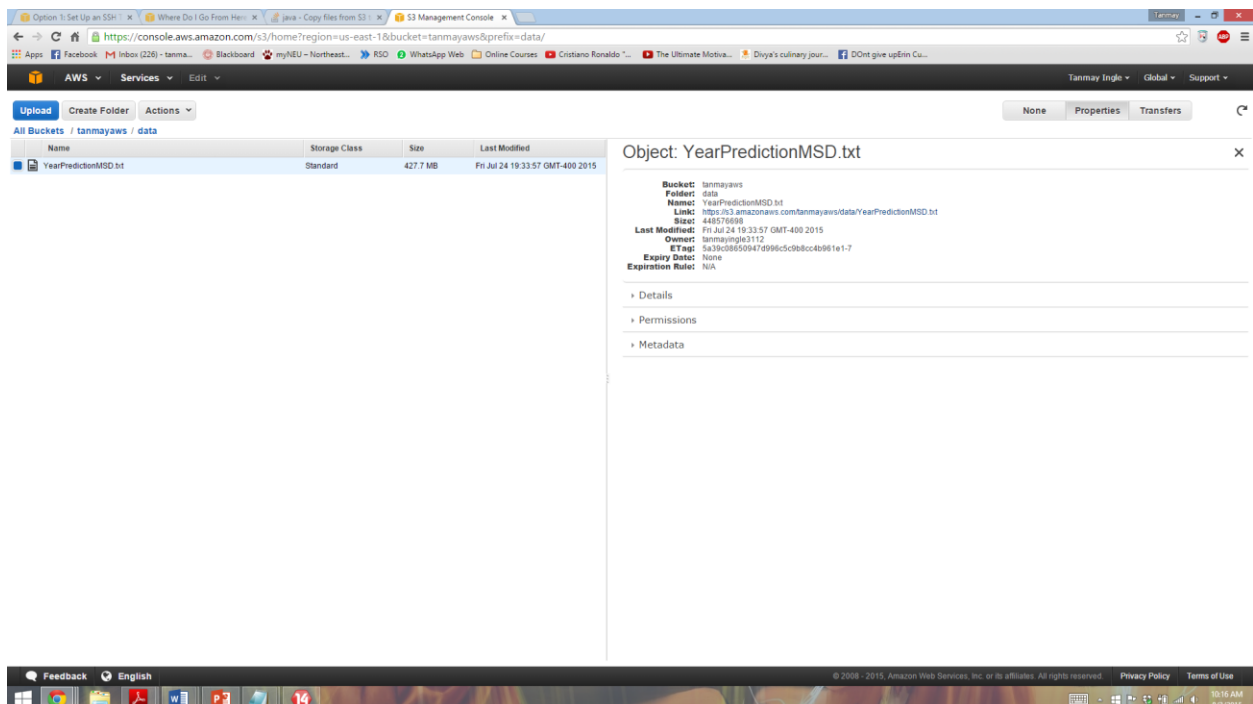
Select the bucket in which you want to upload your data.



Just browse the data in your computed and upload it.



To get the path of your data, go to its properties. You'll find the link to your data. You can also decide the permissions to control who can access that data.



## Amazon EC2 – Elastic Compute Cloud

**Problem:** when we need to buy hardware we need to think about its configuration, come up with a budget which takes time and buy it. And once we do we are stuck with it and we have to use same machines for a long time.

**Solution:** Ec2 makes it easy to have a virtual service known as compute instances in cloud quickly and inexpensively. Simply choose instance type, the template which we want to use and launch the quantity you need. This can be done either in AWS console with few clicks or using sdk in language of your choice and automate the process. Your instances will be running in few minutes. Depending on applications we can choose the type of instances for the best price performance configuration.

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate themselves from common failure scenarios.

It has flexible pricing option as follows:

- On demand pricing - pay for what you use, stop using after it's done. No long term commitments.
- Reserved instances - cheap hourly cost but a one-time fee.
- Spot instance lets you name your price which you want to pay by bidding for it.

### Functionality:

Amazon EC2 presents a true virtual computing environment, allowing you to use web service interfaces to launch instances with a variety of operating systems, load them with your custom application environment, manage your network's access permissions, and run your image using as many or few systems as you desire.

To use Amazon EC2, you simply:

Select a pre-configured, template Amazon Machine Image (AMI) to get up and running immediately. Or create an AMI containing your applications, libraries, data, and associated configuration settings.

- Configure security and network access on your Amazon EC2 instance.
- Choose which instance type(s) you want, then start, terminate, and monitor as many instances of your AMI as needed, using the web service APIs or the variety of management tools provided.
- Determine whether you want to run in multiple locations, utilize static IP endpoints, or attach persistent block storage to your instances.
- Pay only for the resources that you actually consume, like instance-hours or data transfer.

## **Amazon EMR – Elastic Map Reduce**

Amazon Elastic Map Reduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data.

Amazon EMR simplifies big data processing, providing a managed Hadoop framework that makes it easy, fast, and cost-effective for you to distribute and process vast amounts of your data across dynamically scalable Amazon EC2 instances. You can also run other popular distributed frameworks such as Spark and Presto in Amazon EMR, and interact with data in other AWS data stores such as Amazon S3 and Amazon DynamoDB.

Amazon EMR securely and reliably handles your big data use cases, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics.

### Features of EMR:-

Amazon EMR enables you to quickly and easily provision as much capacity as you need and add or remove capacity at any time. This is very useful if you have variable or unpredictable processing requirements. For example, if the bulk of your processing occurs at night, you might need 100 instances during the day and 500 instances at night. Alternatively, you might need a significant amount of capacity for a short period of time. With Amazon EMR you can quickly provision hundreds or thousands of instances, and shut them down when your job is complete (to avoid paying for idle capacity).

Amazon EMR is designed to reduce the cost of processing large amounts of data. Some of the features that make it low cost include low hourly pricing, Amazon EC2 Spot integration, Amazon EC2 Reserved Instance integration, elasticity, and Amazon S3 integration.

With Amazon EMR, you can leverage multiple data stores, including Amazon S3, the Hadoop Distributed File System (HDFS), and Amazon DynamoDB.

Amazon EMR supports powerful and proven Hadoop tools such as Hive, Pig, HBase, and Impala. Additionally, it can run distributed computing frameworks besides Hadoop Map Reduce such as Spark or Presto using bootstrap actions.

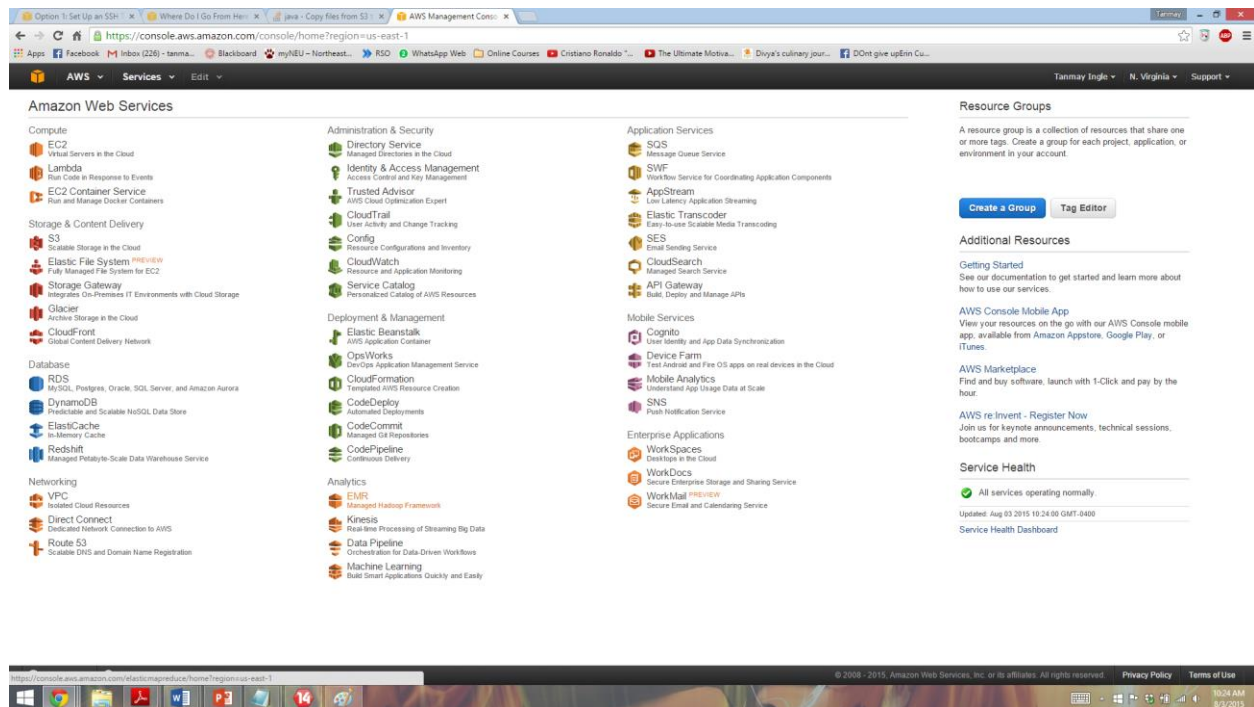
Configuration of EMR way easier than traditional Hadoop.

EMR uses dynamically scalable EC2 instances, thus you have to pay for EMR as well as EC2 instances.

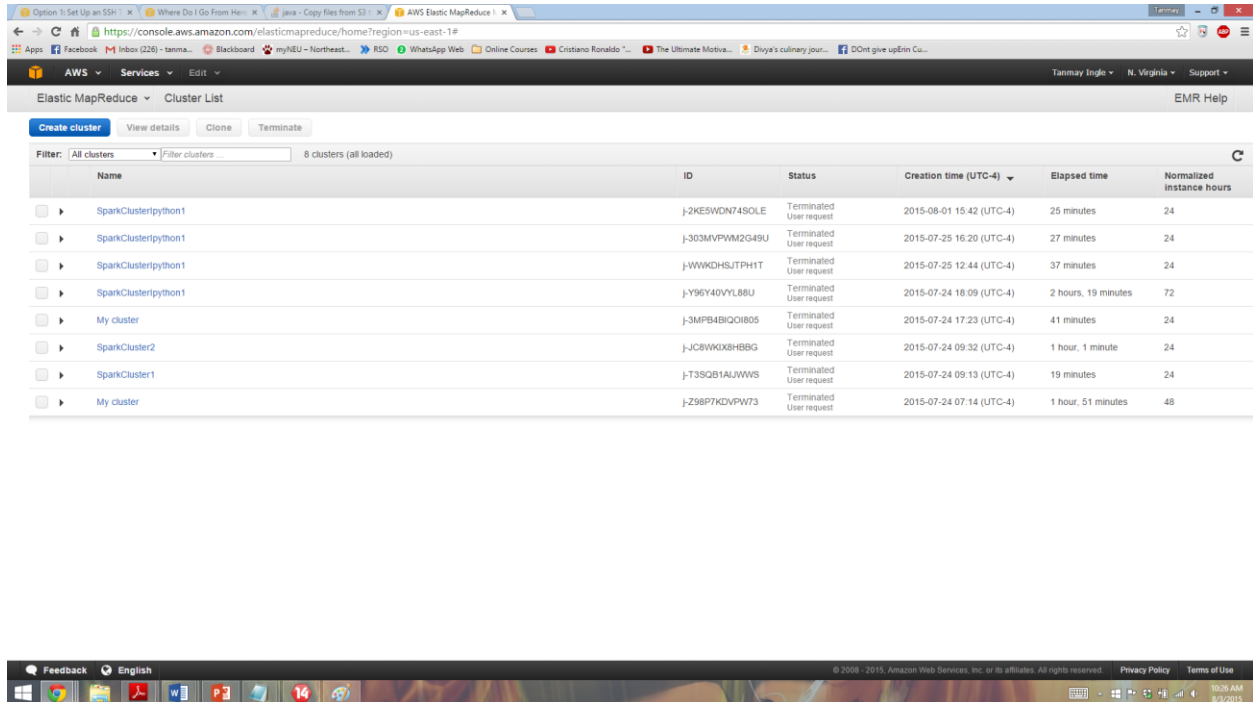
EMR 4.0 has support for APACHE SPARK 1.4.1. To find spark go to following path - /usr/lib/spark.

## DEMO

Sign in to your aws console and select EMR.



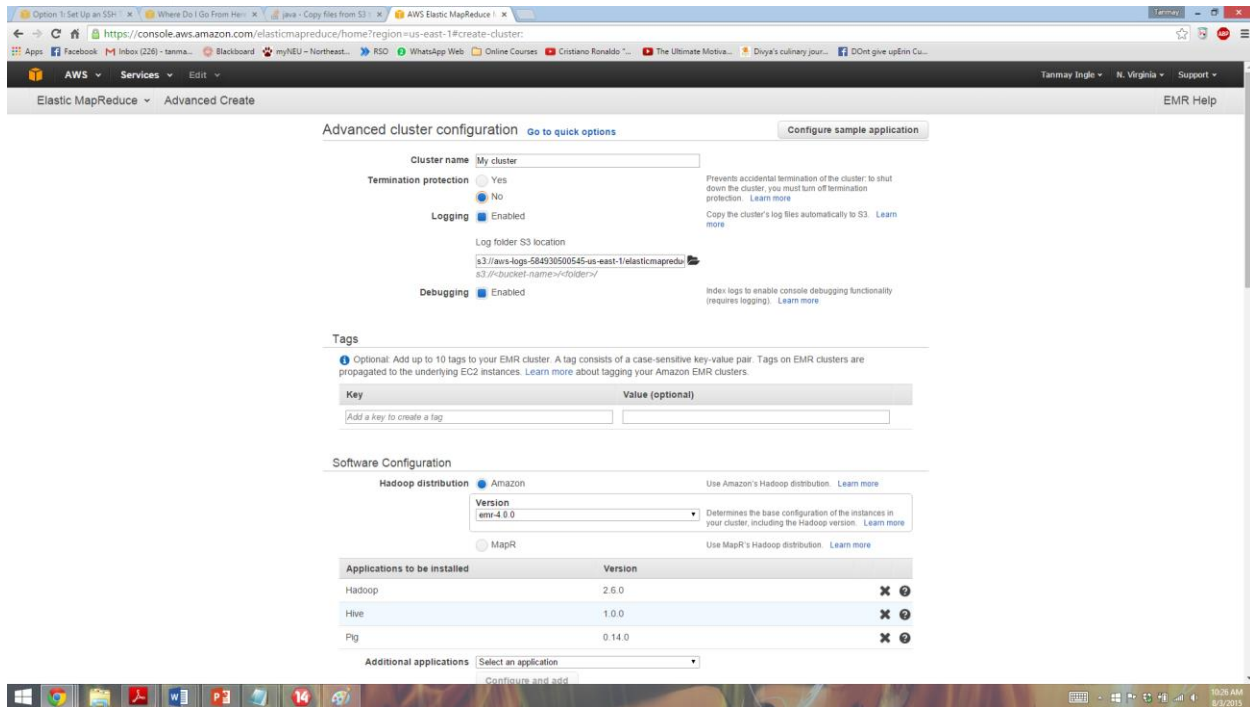
## Hit the tab Create Cluster



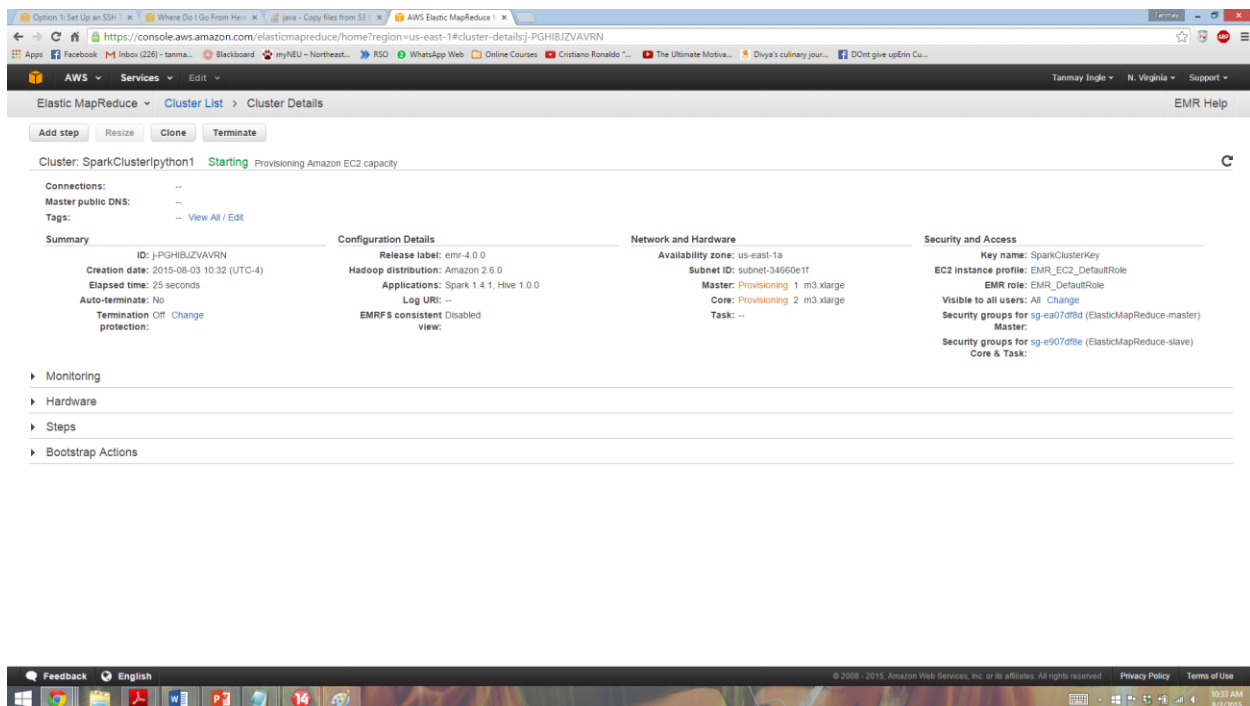
The screenshot shows the AWS Elastic MapReduce console. The 'Create cluster' tab is active. The 'Cluster List' table displays the following data:

Name	ID	Status	Creation time (UTC-4)	Elapsed time	Normalized instance hours
SparkClusterpython1	j-2KE5W0N74SOLE	Terminated User request	2015-08-01 15:42 (UTC-4)	25 minutes	24
SparkClusterpython1	j-303MVPWM2G49U	Terminated User request	2015-07-25 16:20 (UTC-4)	27 minutes	24
SparkClusterpython1	j-WWQDHSJTPH1T	Terminated User request	2015-07-25 12:44 (UTC-4)	37 minutes	24
SparkClusterpython1	j-Y96Y40VYL88U	Terminated User request	2015-07-24 18:09 (UTC-4)	2 hours, 19 minutes	72
My cluster	j-3MPB4BQOI805	Terminated User request	2015-07-24 17:23 (UTC-4)	41 minutes	24
SparkCluster2	j-JC6WKOXBH8BG	Terminated User request	2015-07-24 09:32 (UTC-4)	1 hour, 1 minute	24
SparkCluster1	j-T3SQB1ALWW5	Terminated User request	2015-07-24 09:13 (UTC-4)	19 minutes	24
My cluster	j-Z98P7KDVPT73	Terminated User request	2015-07-24 07:14 (UTC-4)	1 hour, 51 minutes	48

Give cluster name, select software and hardware configuration and select the applications you want to use. Select the EC2 cluster key which you have created. You'll need this to SSH into the master node. You can mention Bootstrap actions if needed. Add following link if you want to install IPYTHON along with pyspark on EMR. ([s3://support.elasticmapreduce/bootstrap-actions/other/install-ipython-notebook-withPySpark](https://s3.amazonaws.com/support.elasticmapreduce/bootstrap-actions/other/install-ipython-notebook-withPySpark)). Once you are done with it, just hit create cluster.



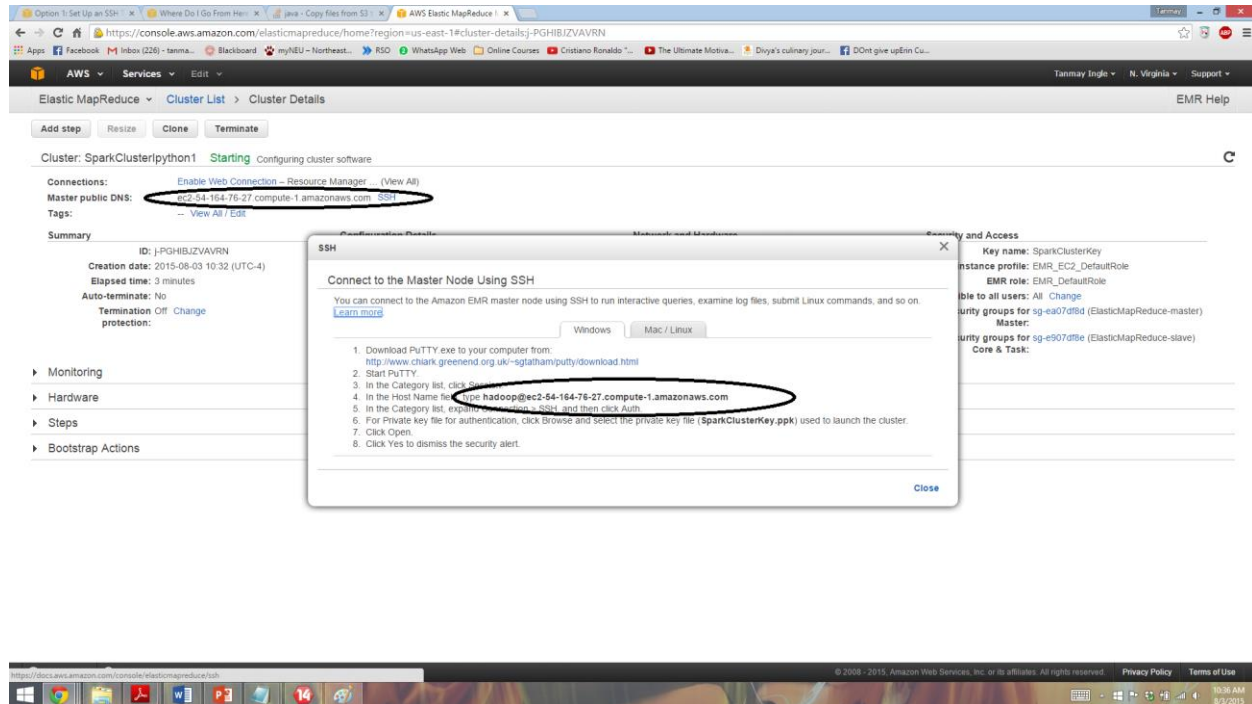
It will take some time to start your instance. Bootstrapping takes a long time. It takes me approximately 8 mins to start the cluster.



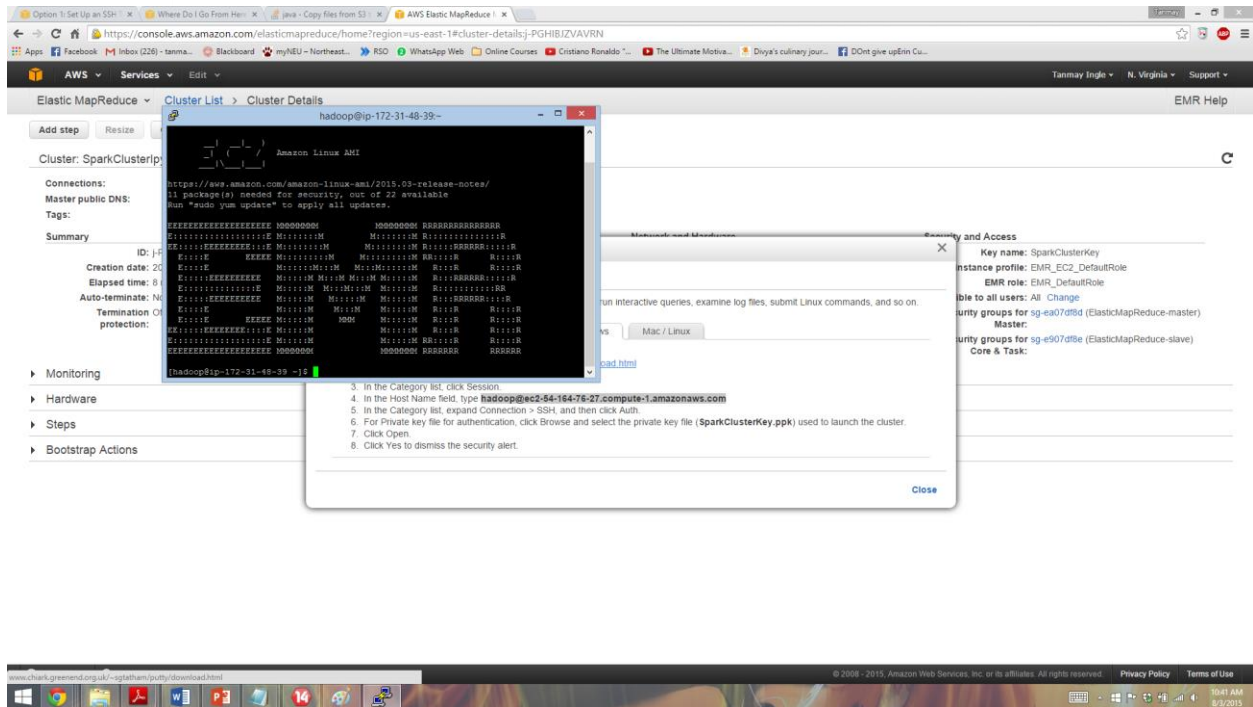


Before we proceed, the EC2 key which we created and downloaded, it is in .pem format. However we need it in .ppk format. Use Putty gen which is a key generator to convert it into .ppk format.

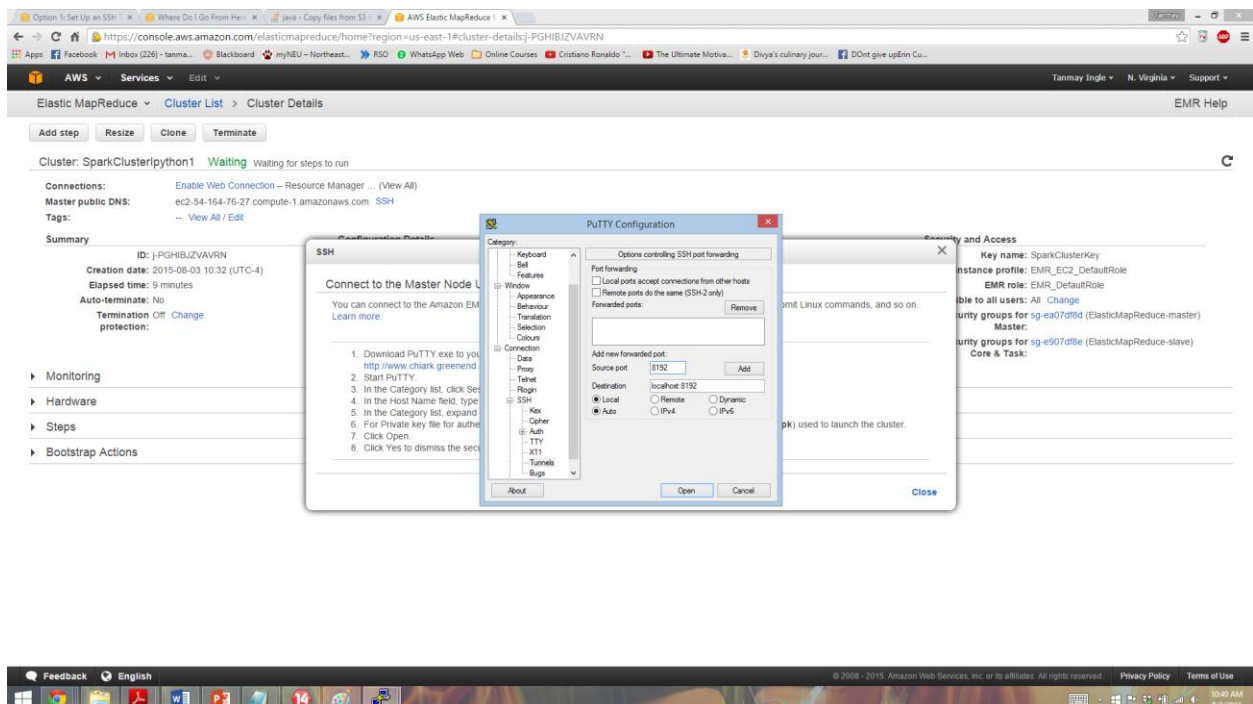
Then Select SSH to get your host name.



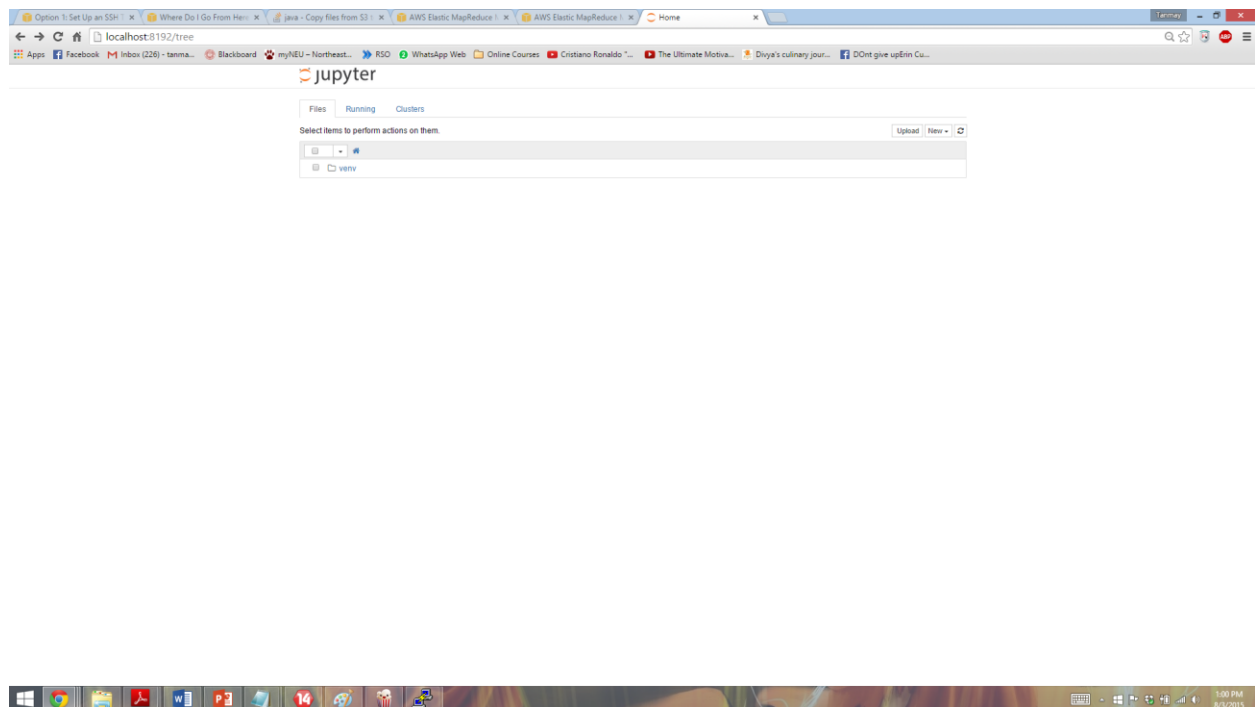
Once the cluster is created, you can start putty to SSH. Add the host name. Then go to SSH and then AUTH and browse for the .ppk key which is stored on the local machine. And hit OK. That's it you will see the terminal in front of you up and running.



Next to start pyspark in ipython notebook start putty again and follow the same procedure of adding host name and cluster key, now create a tunnel by adding source and destination port and hit add.



Now go to your browser and go to localhost:8192. It will start ipython notebook with pyspark. One thing to note is that the data which is used here should be taken from hdfs.



## **Amazon Machine Learning**

**Problem:** Making Machine learning models for developers or in general involves complex mathematics and it is time consuming. Deploying and managing infrastructure to analyze large datasets requires accurate models and then generating predictions. This is expensive and needs continuous maintenance to keep it running.

**Solution:** Amazon Machine learning is a service that allows us to quickly and easily make mobile or any application which uses machine learning. It uses various models and compares and tunes them to an extent to get accurate models and then returns them with new data. It can take data from S3 and store it back in S3.

It can perform batch predictions at once or real time predictions. You can pay as you go so you can start small and scale as you grow.

Amazon Machine Learning is a service that makes it easy for developers of all skill levels to use machine learning technology. Amazon Machine Learning provides visualization tools and wizards that guide you through the process of creating machine learning (ML) models without having to learn complex ML algorithms and technology. Once your models are ready, Amazon Machine Learning makes it easy to obtain predictions for your application using simple APIs, without having to implement custom prediction generation code, or manage any infrastructure.

Amazon Machine Learning is based on the same proven, highly scalable, ML technology used for years by Amazon's internal data scientist community. The service uses powerful algorithms to create ML models by finding patterns in your existing data. Then, Amazon Machine Learning uses these models to process new data and generate predictions for your application.

Amazon Machine Learning is highly scalable and can generate billions of predictions daily, and serve those predictions in real-time and at high throughput. With Amazon Machine Learning, there is no

upfront hardware or software investment, and you pay as you go, so you can start small and scale as your application grows.

## DEMO

Amazon has made usage of Amazon Machine learning very straight forward.

Select Machine Learning from aws console. Create a new machine learning model. The data used should be placed in S3.

The screenshot shows the Amazon Machine Learning console interface. At the top, there's a navigation bar with 'AWS' and 'Services' tabs. Below it, the 'Amazon Machine Learning' section is active. The main area is titled 'Entities' and contains a table listing various machine learning entities. A dropdown menu is open under the 'Create new...' button, showing options like 'Datasource and ML model', 'ML model', 'Evaluation', and 'Batch prediction'. The table has columns for 'Entity name or ID', 'Type', 'ID', 'Status', and 'Creation Time'. The first entity, 'YearPredictionMSD.txt', is in 'Failed' status. Other entities like 'YearPredictionMSD\_bt\_DataSplitting' and 'Banking.csv' are in 'Completed' status. On the right, there's a 'More info' section with links to guides and API reference. At the bottom, there's a 'Machine Learning Concepts' section with a brief description of Amazon ML and a link to 'Getting Started with Amazon Machine Learning'.

Entity name or ID	Type	ID	Status	Creation Time
YearPredictionMSD.txt	Evaluation	ev-S2xPQ0rYZZT	Failed	Jul 25, 2015 12:53:38 PM
YearPredictionMSD_bt_DataSplitting [percentBegin=70, perc...	Datasource	e572e2d5-922b-4740-880c-4c488f4eb36e	Completed	Jul 25, 2015 12:53:37 PM
YearPredictionMSD_bt_DataSplitting [percentBegin=0, perce...	Datasource	cefb787-03ec-4a41-a7ed-59b33d80265	Completed	Jul 25, 2015 12:53:36 PM
YearPredictionMSD.txt	Datasource	ds-WZXH5QPvYfJK	Completed	Jul 25, 2015 12:52:27 PM
Evaluation: ML model: Banking.csv	Evaluation	ev-12VQ4yhn3xT	Completed	Jul 24, 2015 10:32:19 PM
Evaluation: ML model: Banking.csv	Evaluation	ev-gco5yVD6BAB	Completed	Jul 24, 2015 10:22:39 PM
ML model: Banking.csv	ML model	ml-kHtqXEtq2K4	Completed	Jul 24, 2015 10:15:12 PM
Banking.csv_DataSplitting [percentBegin=70, percentEnd=100]	Datasource	bf11e6b3-4d30-41b6-836a-a03a838461a	Completed	Jul 24, 2015 10:15:11 PM
Banking.csv_DataSplitting [percentBegin=0, percentEnd=70]	Datasource	ca435538-4839-4347-93e3-aa53036b2ad1	Completed	Jul 24, 2015 10:15:11 PM
Banking.csv	Datasource	ds-pubhzPkw2SL	Completed	Jul 24, 2015 10:11:21 PM

Machine Learning Concepts

Amazon Machine Learning (Amazon ML) can solve business problems by finding and learning the patterns in your historical data and using the patterns to generate predictions. To get started, you provide Amazon ML with your data. Next, you use Amazon ML to train your ML model and evaluate the model's performance.

[Datasource and ML model](#) [Getting Started with Amazon Machine Learning](#)

Provide the link to the data in S3.

1. Input Data 2. Schema 3. Target 4. Row ID 5. Review

### Input data

Locate the data you want to analyze and process it into a datasource object. You can use a datasource to train (create) and evaluate an ML model.

Where is your data located? ☒ S3 ☐ Redshift

**S3 location**

s3:// tanmayaws/data/YearPredictionMSD.txt

- You can enter the path to a single file or folder in Amazon S3.
- If a file exists at 's3://path.schema', Amazon ML tries to read this as the schema for your datasource. Otherwise, Amazon ML tries to auto-generate a schema from your datasource.
- Learn more about [S3 data layout](#), [supported data formats](#), or [S3 permissions](#)

**Datasource name (optional)**

Provide a name for this datasource

Cancel Verify

Select the variables and continue.

Here either you can specify the recipe for making the model or let Amazon Machine learning make one for you. It automatically split the dataset into 70-30 training test dataset.

1. Input data 2. **ML model settings** 3. Recipe 4. Advanced settings 5. Evaluation 6. Review

### ML model settings

You can use the automatically suggested ML model settings, or you can choose to customize.

**ML model Type** REGRESSION

**ML model target** \_Target\_

**ML model name (Optional)** ML model: YearPredictionMSD.txt

**Training and evaluation settings**

You can select additional settings for your ML model, or use the defaults provided by Amazon ML. Additionally, Amazon ML can reserve a portion of your input data and create an evaluation for your ML model.

☒ **Default (Recommended)**

Select this option to use the recommended ML model parameters and to have an evaluation automatically created for you. Amazon ML will split your input data and use 70 percent for training (training datasource) and 30 percent for evaluating this ML model (evaluation datasource).

**Name this evaluation (Optional)** Evaluation: ML model: YearPredictionMSD.txt

☐ **Custom**

Select this option to specify the available ML model settings and whether you would like to use a portion of your input data to evaluate this ML model.

[Cancel](#) [Previous](#) [Review](#)

Once the model is made we can evaluate the model.

### Evaluation Summary

<b>ID</b>	ev-1zVQ4yhn3xT
<b>Name</b>	Evaluation: ML model: Banking.csv <a href="#">✎</a>
<b>Datasource ID</b>	ca435538-4839-4347-93e3-aad3036b2ad1
<b>Output location</b>	Not available
<b>Created on</b>	07/24/15 22:32:19
<b>Status</b>	Completed
<b>Message</b>	Not available
<b>Log</b>	<a href="#">Download log</a>

### ML model performance metric

On your most recent evaluation, **ev-1zVQ4yhn3xT**, the ML model's quality score is considered **extremely good** for most machine learning applications. ⓘ

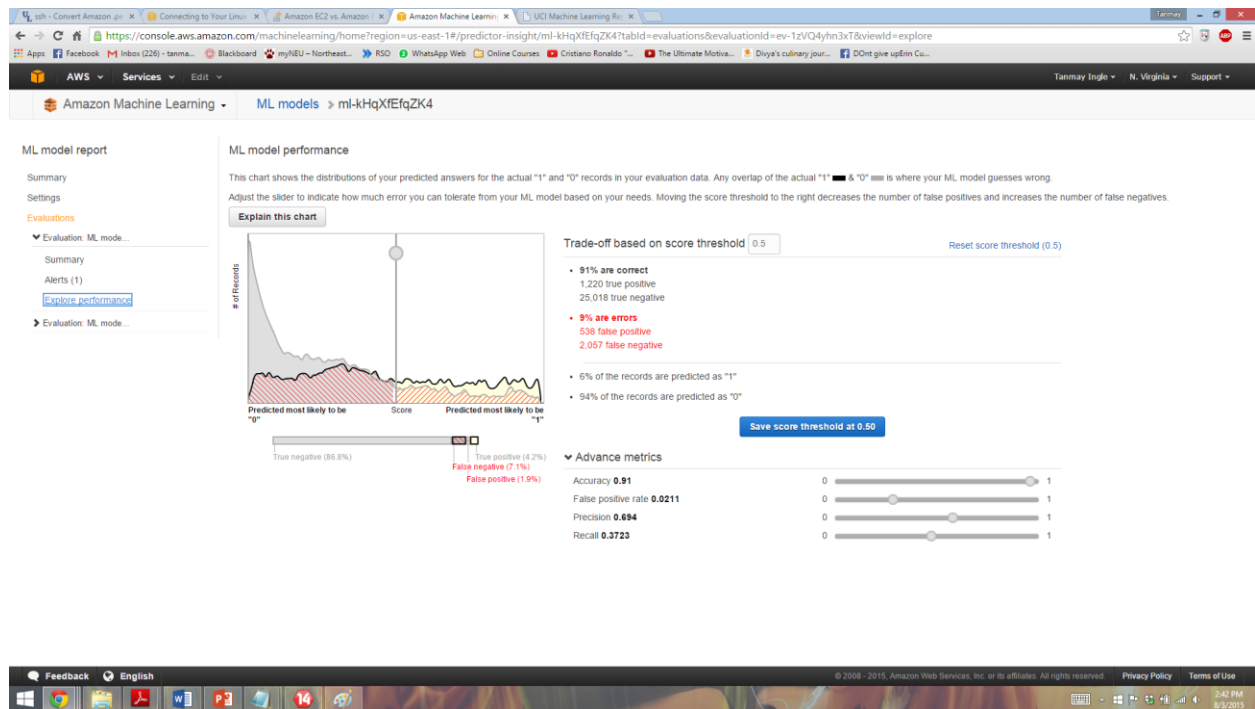
**AUC: 0.94**  
Baseline AUC: 0.50  
Difference: 0.44

**Next step:** If you want to use this ML model to generate predictions, explore trade-offs to optimize the performance of your ML model first. ⓘ

Score threshold: 0.5

[Adjust score threshold](#)

It provides a proper summary of the evaluation of the model





## Amazon DynamoDB

DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.

We can use DynamoDB to create a database table that can store and retrieve any amount of data, and serve any level of request traffic.

All data items are stored on solid state disks (SSDs)

Replicated across multiple Availability Zones in a Region to provide built-in high availability and data durability.'

Data can only be retrieved by the key. However, there are two types of keys.

The first is called a hash. It's a single value and it's what you would normally think of when you are talking about a key

The second is a composite of a hash and a range. This type of key lets you query data by either the hash component or the hash and range component

### **When Dynamo dB and when not?**

If you have a use case that involves storing a large amount of data that you rarely access, then DynamoDB may not be right for you. We recommend that you use S3 for such use cases.

### **Amazon RDS VS DYNAMODB**

While Amazon DynamoDB tackles the core problems of database scalability, management, performance, and reliability, it does not have all the functionality of a relational database. It does not support complex relational queries (e.g. joins) or complex transactions

If your workload requires this functionality, or you are looking for compatibility with an existing relational engine, you may wish to run a relational engine on Amazon RDS or Amazon EC2. While relational database engines provide robust features and functionality, scaling a workload beyond a single relational database instance is highly complex and requires significant time and expertise. As such,

if you anticipate scaling requirements for your new application and do not need relational features, Amazon DynamoDB may be the best choice for you.

### **Amazon DynamoDB differ from Amazon SimpleDB**

Which should I use? Both services are non-relational databases that remove the work of database administration. Amazon DynamoDB focuses on providing seamless scalability and fast, predictable performance. It runs on solid state disks (SSDs) for low-latency response times, and there are no limits on the request capacity or storage size for a given table. This is because Amazon DynamoDB automatically partitions your data and workload over a sufficient number of servers to meet the scale requirements you provide. In contrast, a table in Amazon SimpleDB has a strict storage limitation of 10 GB and is limited in the request capacity it can achieve (typically under 25 writes/second); it is up to you to manage the partitioning and re-partitioning of your data over additional SimpleDB tables if you need additional scale. While SimpleDB has scaling limitations, it may be a good fit for smaller workloads that require query flexibility. Amazon SimpleDB automatically indexes all item attributes and thus supports query flexibility at the cost of performance and scale.

### **When MongoDB?**

DynamoDB is suitable for use cases where data access is by one or two dimensions of data, and usually by applications running on Amazon's EC2 service

A second dimension of data, that must be numeric, can be used to form a composite primary key, e.g. timestamp, this is handy for allowing fast lookup of all orders for user id within a given date range, for example.

But, if your data access patterns require reads on more than two dimensions of data, then MongoDB is a better solution. MongoDB supports any number of indexes (but uses one at most in a single query)

Name	Amazon DynamoDB	MongoDB
------	-----------------	---------

Description	Hosted scalable database service by Amazon with the data stored in Amazons cloud	One of the most popular document stores
Database model	Document store, Key-value store	Document store
Foreign keys	No	No
Replication method	Yes	Master-slave replication
Database as a Service	Yes	No
Server-side script	No	JavaScript

#### **Example Use case of Amazon Dynamo DB:**

We have a single table where you can search by customerId and ordered time range and retrieve the order details for the customer in the last x days/months/years, search by order id and retrieve the customer details, etc.

Script used for example is in the same folder named DynamoDB script.

## **Amazon Elastic BeanStalk**

### **What Is Elastic Beanstalk and Why Do we Need It?**

AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed

Elastic Beanstalk uses Auto Scaling and Elastic Load Balancing to easily support highly variable amounts of traffic

While the variety of services offers flexibility for how you want to manage your AWS infrastructure, it can be challenging to figure out which services to use and how to provision them

### **Elastic Beanstalk Components:**

These components that comprise Elastic Beanstalk work together to enable you to easily deploy and manage your application in the cloud.

#### **Application:**

An Elastic Beanstalk application is a logical collection of Elastic Beanstalk components, including environments, versions, and environment configurations. In Elastic Beanstalk an application is conceptually similar to a folder.

#### **Application Version:**

In Elastic Beanstalk, an application version refers to a specific, labeled iteration of deployable code for a web application. An application version points to an Amazon Simple Storage Service (Amazon S3) object that contains the deployable code such as a Java WAR file. An application version is part of an application. Applications can have many versions and each application version is unique. In a running environment, you can deploy any application version you already uploaded to the application or you can upload and immediately deploy a new application version. You might upload multiple application versions to test differences between one version of your web application and another.

**Environment:**

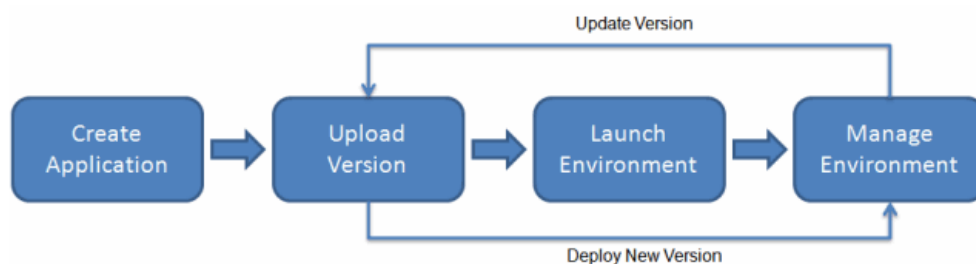
An environment is a version that is deployed onto AWS resources. Each environment runs only a single application version at a time, however you can run the same version or different versions in many environments at the same time. When you create an environment, Elastic Beanstalk provisions the resources needed to run the application version you specified. For more information about the environment and the resources that are created.

**Environment Configuration:**

An environment configuration identifies a collection of parameters and settings that define how an environment and its associated resources behave. When you update an environment's configuration settings, Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources (depending on the type of change).

**Flow:**

To use Elastic Beanstalk, you create an application, upload an application version in the form of an application source bundle (for example, a Java .war file) to Elastic Beanstalk, and then provide some information about the application. Elastic Beanstalk automatically launches an environment and creates and configures the AWS resources needed to run your code. After your environment is launched, you can then manage your environment and deploy new application versions.



## **References:-**

[https://d36cz9buwru1tt.cloudfront.net/AWS\\_Overview.pdf](https://d36cz9buwru1tt.cloudfront.net/AWS_Overview.pdf)

<https://aws.amazon.com/>