

Databases play a crucial role in the functioning of an application. Also, performance of an application is directly dependent on how the underlying database performs for the application. AWS Database Services is a set of databases offered by AWS on the cloud.

With AWS Database Services, you just have to focus on your application code and business expansion. The rest will be managed and taken care of by AWS, i.e., all the hardware up-gradations, security patches, and even the software up-gradations are taken care of by AWS. You just have to pay AWS for the time you used their database services, nothing extra for the hardware up-gradations, etc.

Advantages of Amazon AWS Databases

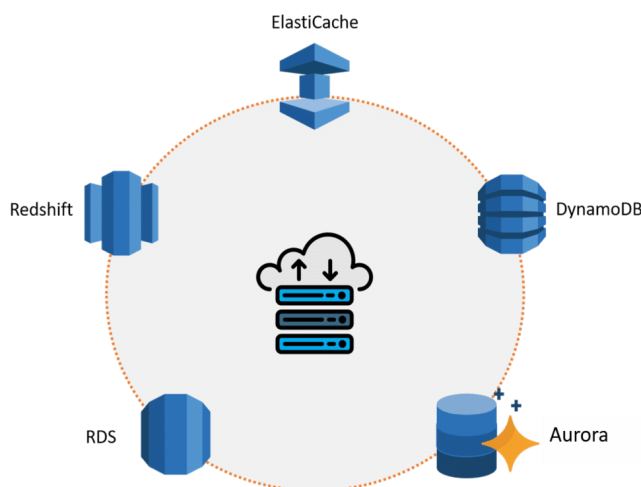
- **Highly Scalable:** You can scale your database as your application grows without any downtime
- **Fully Managed:** Everything, from maintenance to hardware upgrades, is managed by AWS.
- **Enterprise Class:** You get the same world-class infrastructure used by Amazon's giant ecommerce platform.
- **Distributed:** Now that your application and database exist on separate machines, your application becomes highly fault-tolerant.
- **Workforce Reduction:** Since everything is managed by AWS, you don't need a Database Maintenance team in your organization.

Types of AWS Database Services

AWS provides a wide range of fully managed, purpose-built and both relational and non-relational database services specially designed to handle any kind of application requirements. From fully managed database services, a data warehouse for analytics, to an in-memory data store for caching, AWS has got it all.

You will find an AWS Database Service for just about any kind of database requirements. One can import an existing MySQL, Oracle, or Microsoft SQL database into Amazon's databases or even build their own relational or NoSQL databases from scratch.

Following are different types of database services provided by AWS:



1. **Relational Database:** In relational databases, the data is usually stored in a tabular format. Relational databases particularly use structured query language (SQL) to run queries to perform operations such as insertion, updating, deletion, and more. AWS provides following relational database services.
 - Amazon RDS
 - Amazon Redshift
 - Amazon Aurora
2. **Key-Value Database:** The key-value database is a type of NoSQL database where the method of having a value attached to a key is used to store data. Meaning that the data is composed of two elements, keys and values.
 - Amazon DynamoDB
3. **In-memory Database:** This type of database is primarily based on the main memory for computer data storage. Basically, an in-memory database keeps the whole data in the RAM. Meaning that each time you access the data, you only access the main memory and not any disk. And the reason that the main memory is faster than any disk is why in-memory databases are so popular.
 - Amazon ElastiCache

What Is AWS RDS?



One of the most commonly used database services provided by AWS that falls under the category of relational databases is [Amazon RDS](#). Amazon RDS is a service that supports various open-source relational database products including the database products provided by AWS itself. RDS is used to set up, operate, and scale a relational database in the cloud. It automates administrative tasks such as hardware provisioning, database setup, backups, and more.

Following are some of the benefits of using RDS:

- It provides high performance and is fast to scale.
- It provides high availability as a result of two distinct replication features, namely, Multi-AZ deployments and Read Replicas.
- It automatically takes care of Backup and Restore by patching up the AWS database software.
- It also takes care of Maintenance and Upgrades, automatically.

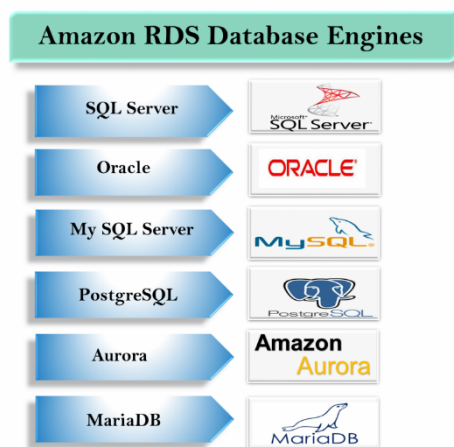
Amazon Relational database service

Amazon RDS is the service provided by the AWS which makes it easy to set up, deploy, scale, and operate relational databases in the cloud. It provides you with six familiar database engines to choose and that includes Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle Database, and SQL Server.

What is a Relational Database?

- Relational Databases are the databases that what most of us are all used to. It has been around since the 70's.
- Relational Database is like a spreadsheet such as Excel, etc.
- A Database consists of tables. For example, Excel is a spreadsheet that consists of a workbook, and inside the workbook, you have different sheets, and these sheets are made up of rows and columns.

Relational Database Types



What is a NonRelational Database?

- **NonRelational Database** is a database that does not follow the relational database model provided by the relational database management system.
- It is a NoSQL database, we have seen a steady growth in Non Relational Database with the rise in Big data applications.
- We have a NonRelational Database. Inside the database, we have a collection. Inside a collection, we have got a document, and inside the document, we have key-value pairs. If we talk in a relational sense then Collection is a table, the document is a row, and row consists of key-value pairs.

How does a Non Relational database work?

A Non Relational database model uses a variety of different data models such as key-value, document, Graph, in-memory, and search.

Let's understand through an example.

- In a relational database, a book record consists of separate tables, and the relationship between tables are defined by primary and foreign constraints. For example, **Book** table has three columns, i.e., **Book id**, **Book title** and **Edition Number**, **Author** table has three columns, i.e., **Author id**, **Author name**, and **Book id**. The relationship model is designed so that the database can enforce referential integrity between the tables to reduce redundancy.
- In NonRelational database, records are stored in the form of json format. Each book item such as Book id, Book title, Edition Number, Author id, Author name is stored as attributes in a document.

```

1. JSON/NoSQL
2. {
3.   "Book id": "1",
4.   "Book title": "Computer",
5.   "Edition Number": "121",
6.   "Author id": "23",
7.   "Author name": "Ankita"
8. }
```

Why to use Non Relational database

Non Relational database is used because of the following features:

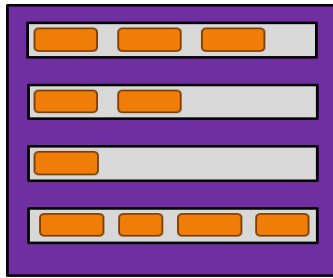
- **Flexibility:** It has a very flexible data model which provides faster and iterative development. The flexible model of Non Relational database makes an ideal for structured, semi-structured and unstructured data.
- **Scalability:** Non Relational databases provide scaling out by using distributed clusters of hardware rather than scaling up by adding expensive servers.
- **High-performance:** Non Relational databases use some specific data models such as key-value, document, etc that provides higher performance than relational databases.
- **Highly functional:** Non Relational databases provide highly functional APIs and data types for their respective data models.

What Is Amazon AWS DynamoDB?

Amazon DynamoDB is a fast, fully managed, and flexible [NoSQL database](#). It also supports document-based data. AWS affirms that DynamoDB delivers single-digit millisecond performance at any scale. DynamoDB comes with built-in Security, Backup, and Restore features.

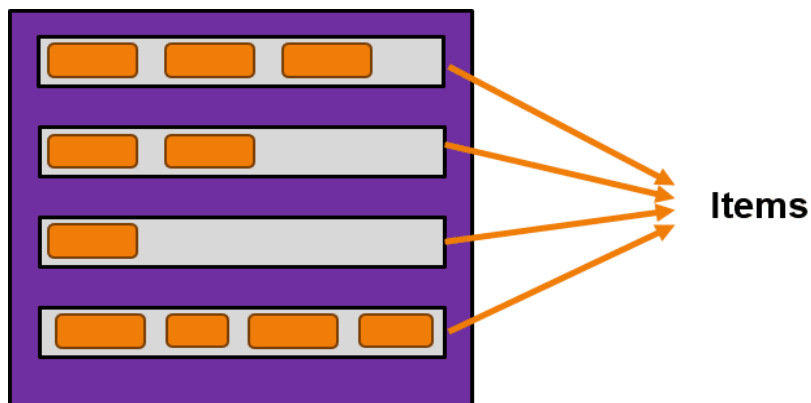
Since DynamoDB is a NoSQL database, it doesn't require any schema. In DynamoDB, there are basically three core components:

1. **Tables:** The collection of data is called a table in DynamoDB. It's not a structured table with a fixed number of rows and columns.



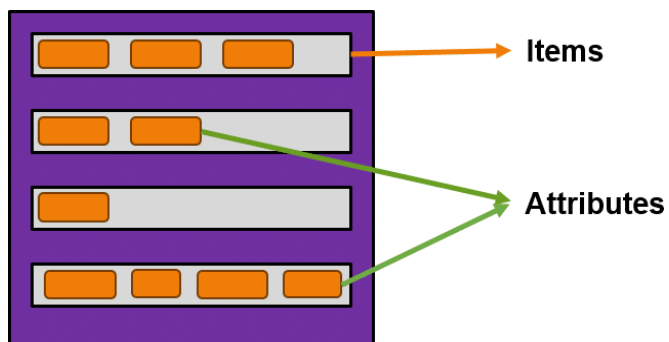
Table

2. Items: Tables in [DynamoDB](#) contain one or more items. Items are made up of a group of uniquely identifiable attributes.



Table

3. Attributes: Attributes are the data elements or values that reside in each item. They are equivalent to data values in a relational database that reside in a particular cell of a table.



Table

Following are some of the benefits of using Amazon DynamoDB:

- Easy to set up and manage
- Data is automatically replicated across multiple Availability Zones
- Supports both key-value and document-based data models

Amazon DynamoDB:

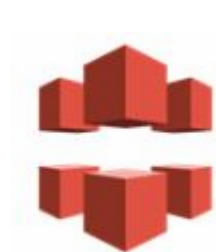
Amazon DynamoDB is the NoSQL database service provided by amazon which is fully managed and automated. NoSQL means that you don't have to write queries to create a table or retrieve the

data you can do so by some clicks to create a dynamic table which means you can add any amount of attributes, columns and store the data. The main advantage of using this is that it is fully managed and it automatically handles the traffic of data on the multiple servers and gives the optimum performance and you don't have to lookup for the underlying hardware, setup, configuration, scaling is all managed by the amazon. It also automatically backup and restore that provides the security of data. The feature provided by the amazon which is commendable is that according to your need of data or traffic it automatically scale up and scale down you don't have to look upon the underlying servers or their maintenance and according to the usage you have you used it charges accordingly and also there are no minimum charges for the usage. According to the AWS, DynamoDB can handle 20 trillion of the request per day and also can handle the peak of the traffic up to 20 million per second which is huge and commendable.

What is DynamoDB?

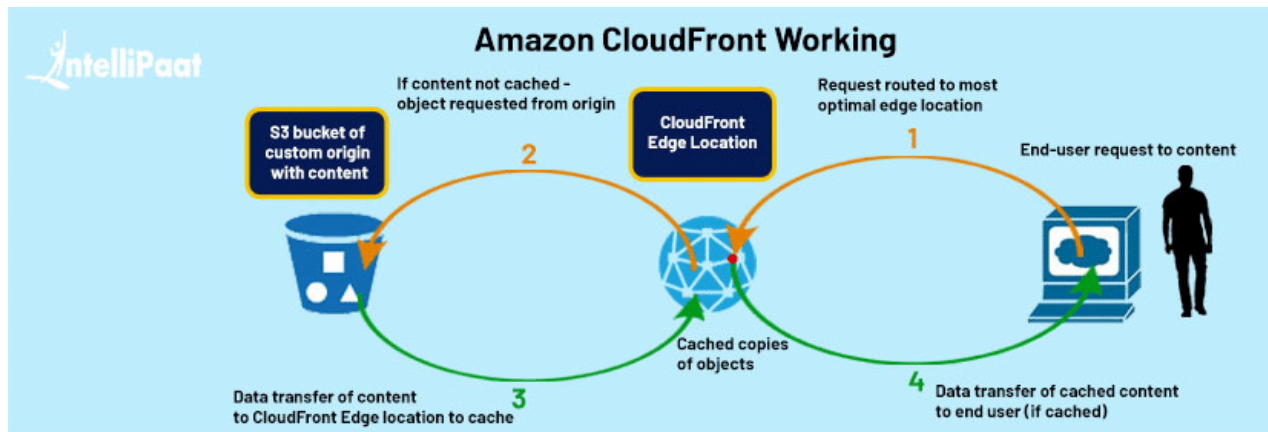
- Amazon DynamoDB is a fast and flexible NoSQL database service for all applications that require consistent single-digit millisecond latency at any scale.
- It is a fully managed database that supports both document and key-value data models.
- Its flexible data model and performance makes it a great fit for mobile, web, gaming, ad-tech, IOT, and many other applications.
- It is stored in SSD storage.
- It is spread across three geographically data centres.

CloudFront CDN



Amazon Cloudfront is a content delivery network (AWS CDN) that retrieves data stored in the Amazon S3 bucket and distributes it to numerous edge locations across the world. Edge locations are the network of data centers distributed worldwide through which content is delivered. CloudFront CDN is a system of distributed servers that deliver web pages and other web content to a user based on the geographic locations of the user, the origin of the webpage and a content delivery server. The nearest edge location is routed when the user requests for data, resulting in lowest latency, low network traffic, fast access to data, etc. There was a time when having an own Content Delivery Network was extremely rare for companies due to its high costs and complicated IT infrastructure. But now, AWS CloudFront has helped users to request data resulting in low latency, low network traffic, and quick data access with minimal cost. Thus, making it a very popular network.

What is AWS Cloudfront?



When a user requests content that is being served with CloudFront, the request is routed to the nearest edge location that gives the lowest latency. This helps the user to access content with the best possible performance. The Amazon cloud architecture functions as follows.

- If the content is already cached in the edge location, CloudFront delivers it immediately with the lowest latency possible.
- If the content is not present in the edge location, CloudFront retrieves it from the origin (like Amazon S3 bucket, a MediaPackage channel, or an HTTP server) that has been identified for your content.

How does Amazon CloudFront deliver content?

Once an Amazon S3 Bucket or HTTP Server is set up, a CloudFront distribution is created to tell CloudFront where you want the content to be delivered and details about tracking and managing content delivery. CloudFront then uses edge servers that are close to your viewers to deliver content quickly when someone accesses it.



Steps to configure CloudFront to deliver content :

1. When the origin servers such as the Amazon S3 bucket or an own HTTP server are specified, CloudFront gets the required files which are then distributed from CloudFront edge locations all over the world.
2. Files are uploaded to the origin servers. The files also known as objects, typically include web pages, images, and media files. It can include anything that can be served over HTTP.

3. The CloudFront distribution tells CloudFront which origins to get the files from when users request the files through the website or application. CloudFront also logs all the requests and distribution can be enabled as soon as it's created.
4. A domain is assigned to the new distribution that can be found in the CloudFront console or an alternate domain can be used instead.
5. The distribution is sent across to all the edge locations or points of presence collections that are collections of servers in geographically dispersed data centers where CloudFront caches a copy of the data.

Key Terminology of CloudFront CDN

- **Edge Location:** Edge location is the location where the content will be cached. It is a separate to an AWS Region or AWS availability zone.
- **Origin:** It defines the origin of all the files that CDN will distribute. Origin can be either an S3 bucket, an EC2 instance or an Elastic Load Balancer.
- **Distribution:** It is the name given to the CDN which consists of a collection of edge locations. When we create a new CDN in a network with aws means that we are creating a Distribution.

The distribution can be of two types:

- **Web Distribution:** It is typically used for websites.
- **RTMP:** It is used for Media Streaming.

How CloudFront CDN works



- Edge locations spread all around the world and currently, there are 50 edge locations.
- When the first user requests to get the content, and the request goes to the nearest edge location. The nearest edge will be read first to determine whether it contains the cached data or not. If an edge does not contain the cached data, the edge location pulls the data from the S3 bucket. Suppose the S3 bucket is in Ireland. But this process is not quicker for the first user. However, when the second user accesses the same file, this file is already cached to the edge location, so it pulls the data from its edge location. It speeds up the delivery of the data.

Creating a CloudFront CDN

Step 1: Create a bucket and upload content in a bucket.

Choose a region for your bucket. By default, Amazon S3 bucket stores the object in the US East (Ohio) region. Enable public access to the object that has been uploaded successfully. After uploading a file, you can navigate to the object by using a URL given below:

<https://s3.us-east-2.amazonaws.com/jtpbucket/jtp.jpg>

Step 2: Create a CloudFront Distribution

- Open the CloudFront Console by using the link <https://console.aws.amazon.com/cloudfront/>.
- Click on the **Create Distribution**
- Select the delivery method for your content, in the **Web Distribution**, click on the **Get Started** button.

Origin Settings

Create Distribution

Origin Settings

Origin Domain Name	<input type="text" value="jtpbucket.s3.amazonaws.com"/>					
Origin Path	<input type="text"/>					
Origin ID	<input type="text" value="S3-jtpbucket"/>					
Restrict Bucket Access	<input checked="" type="radio"/> Yes <input type="radio"/> No					
Origin Access Identity	<input checked="" type="radio"/> Create a New Identity <input type="radio"/> Use an Existing Identity					
Comment	<input type="text" value="access-identity-"/>					
Grant Read Permissions on Bucket	<input checked="" type="radio"/> Yes, Update Bucket Policy <input type="radio"/> No, I Will Update Permissions					
Origin Custom Headers	<table><thead><tr><th>Header Name</th><th>Value</th></tr></thead><tbody><tr><td><input type="text"/></td><td><input type="text"/></td></tr></tbody></table>	Header Name	Value	<input type="text"/>	<input type="text"/>	
Header Name	Value					
<input type="text"/>	<input type="text"/>					

Where,

Origin Domain Name: It defines from where the origin is coming from. Origin domain name is **jtpbucket.s3.amazonaws.com** in which jtpbucket is a bucket that we have created in S3.

Origin Path: There can be multiple origins in a distribution. Origin path is a folder in S3 bucket. You can add the folders in S3 bucket and put it in the Origin Path, means that the origin is coming from the different folders not from the bucket itself. I leave the Origin Path with a default value.

Origin ID: It is the name of the origin. In our case, the name of the origin is **S3-jtpbucket**.

Restrict Bucket Access: If you don't want the bucket to be publicly accessible by the S3 URL and you want that all requests must go through CloudFront, then enable the **Restrict Bucket Access** condition.

Origin Access Identity: We do not have any existing identity, so we click on the **Create a new identity**.

Grant Read Permissions on Bucket: Either you can manually update the permissions or you want the permissions to be updated automatically. So, we click on the **Yes, Update Bucket Policy**.

How AWS CloudFront Delivers the Content?

AWS CloudFront delivers the content in the following steps.

Step 1 – The user accesses a website and requests an object to download like an image file.

Step 2 – DNS routes your request to the nearest CloudFront edge location to serve the user request.

Step 3 – At edge location, CloudFront checks its cache for the requested files. If found, then returns it to the user otherwise does the following –

- First CloudFront compares the request with the specifications and forwards it to the applicable origin server for the corresponding file type.
- The origin servers send the files back to the CloudFront edge location.
- As soon as the first byte arrives from the origin, CloudFront starts forwarding it to the user and adds the files to the cache in the edge location for the next time when someone again requests for the same file.

Step 4 – The object is now in an edge cache for 24 hours or for the provided duration in file headers. CloudFront does the following –

- CloudFront forwards the next request for the object to the user's origin to check the edge location version is updated or not.
- If the edge location version is updated, then CloudFront delivers it to the user.
- If the edge location version is not updated, then origin sends the latest version to CloudFront. CloudFront delivers the object to the user and stores the latest version in the cache at that edge location.

Benefits of AWS CloudFront

- **Global** There are 216 edge locations across the globe. With the number of edge locations, it enables the end-user to use your content without any latency.
- **Fast** Amazon CloudFront provides a high data transfer rate since the content is already cached in the nearest edge location, it gives the end-user a lightning-fast speed delivery of the content.
- **Dynamic Transfer** It provides an option for both static and dynamic content delivery. As soon as one byte is loaded into the cache, it is immediately transferred to the client working as a Live Stream.
- **Encryption** A highly secure application is developed without extra cost. A CloudFront inherits the features of AWS Shield Standard. The edge locations which are spread across the globe feature a 7 Layer Protection named AWS Web Application Firewall.
- **AWS Integration** The whole of AWS-supported services for your content or application can be integrated into the CloudFront. Amazon CloudFront is designed in such a way that it can be easily integrated with other AWS services, like Amazon S3, Amazon EC2.

- **Cost-effective** – Using Amazon CloudFront, we pay only for the content that you deliver through the network, without any hidden charges and no up-front fees.
- **Elastic** – Using Amazon CloudFront, we need not worry about maintenance. The service automatically responds if any action is needed, in case the demand increases or decreases.
- **Reliable** – Amazon CloudFront is built on Amazon's highly reliable infrastructure, i.e. its edge locations will automatically re-route the end users to the next nearest location, if required in some situations.

Companies Using CloudFront

Spotify
Jio Saavan
Sky news
TV1-EU

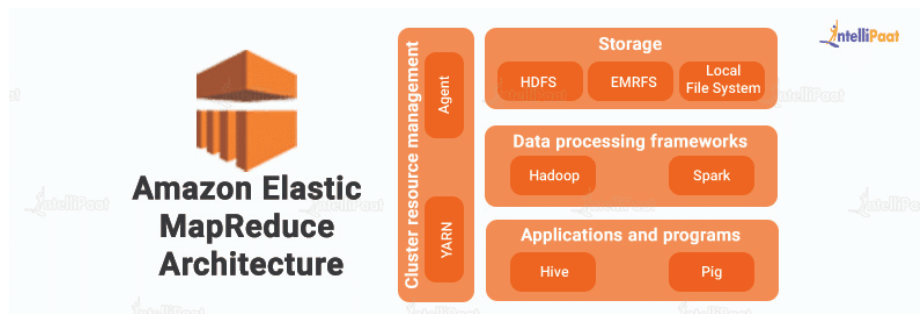
Amazon Elastic MapReduce (EMR)



EMR is a massive data processing and analysis service from [AWS](#). It is a web service that provides a managed framework to run data processing frameworks in an easy, cost-effective, and secure manner. It is used for data analysis, web indexing, data warehousing, financial analysis, scientific simulation, etc. AWS EMR is one of the most popular clouds and big data-based platforms that provides a supervised architecture for easily, cost-effectively, and securely running data processing frameworks. It is used for processing large volumes of data with open source technologies. Elastic MapReduce provides a simple and comprehensible solution to handle the processing of big data sets. EMR has a remarkable pricing list that appeals to businesses and the wider public. You may utilize it only over an hour base and the number of units in your clusters because it has an on-demand charging option.

Architecture of AWS EMR

The AWS EMR service architecture is made up of multiple layers, each offering clusters with specific features and functions.



Storage

The storage layer contains the various system files which a cluster uses. There are a variety of storage choices available, as shown below.

Cluster Resource Management

Then comes the next layer, Cluster Resource Management. This layer is in charge of cluster resource management and data processing scheduling tasks.

Data Processing Frameworks

The third layer of the AWS architecture is data processing frameworks. It is an engine that processes and analyses data.

Applications and Programs

The fourth layer contains the applications and programs which aid in the processing and management of big data sets, such as HIVE, PIG, streaming libraries, and machine learning algorithms.

Features of AWS EMR

Moving on, it's time to see some features of AWS EMR:

1.Adaptability

AWS EMR makes it easier to create and manage large data platforms and apps. Easy provision, controlled scaling, and cluster reconfiguration are among the EMR characteristics, as is EMR Studio for cohesive development.

2.Elasticity

AWS EMR allows you to supply as much capacity as you require fast and efficiently, and to add multiple capacities manually or automatically. This is especially beneficial if your processing requirements are changeable or unexpected.

3.Flexibility

AWS EMR is highly flexible. You may use several data stores with AWS EMR, including Amazon S3, Hadoop Distributed File System (HDFS), and [Amazon DynamoDB](#).

4.Tools for Big Data

Apache Spark, Apache Hive, Presto, and Apache HBase are among the Hadoop technologies supported by AWS EMR. Data scientists use EMR to execute deep learning and its technologies like TensorFlow and Apache MXNet, as well as scenario tools and frameworks, utilizing bootstrap operations.

Components of AWS EMR

The AWS EMR service consists of a few components as follows:

Clusters: Clusters are groups of EC2 instances. You can build two sorts of clusters which are temporary clusters and long-running clusters.

- A temporary cluster that ends when the steps are completed
- A permanent cluster is a long-running cluster that keeps operating unless you explicitly stop it.

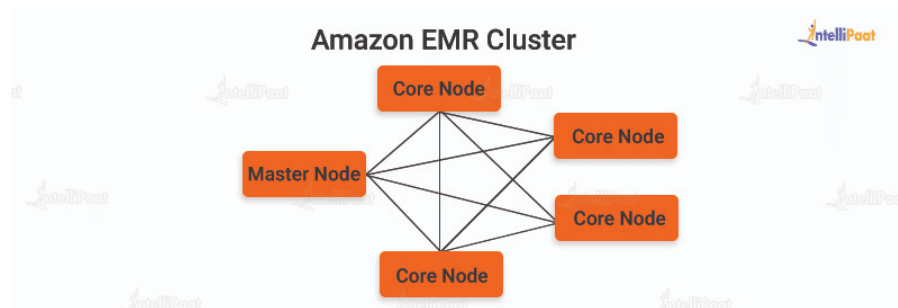
Node: Every EC2 instance in a cluster is referred to as a node. The node type refers to the role that each node plays inside the cluster. The different sorts of nodes are the Master node, Core node, and Task node.

- Every cluster has a master node that oversees data and job distribution among all the other nodes. The master node keeps track of project status and oversees the cluster's stability.
- The Core Node is in charge of performing the job and storing the data in the cluster's HDFS. All processing is handled by the core Node, and the data is then written to the chosen HDFS location.
- As the Task Node is optional, it simply has the job of completing the task. The data is not stored in HDFS in this case.

Working of AWS EMR

We can also use a method of connecting the master node to other nodes through a secure connection and use the interfaces and tools provided for the software that runs straight on your cluster. Using this method, you can submit work and connect with the software deployed in your AWS EMR cluster instantly.

The cluster distribution in EMR is depicted in the diagram below. Let's take a closer look at that:



When you use AWS EMR to process data, the data is saved as files underneath your file system of choices, such as Amazon S3 or HDFS. In the process, this data moves from one stage to the next. (EMR clusters can accept one or more ordered steps.)

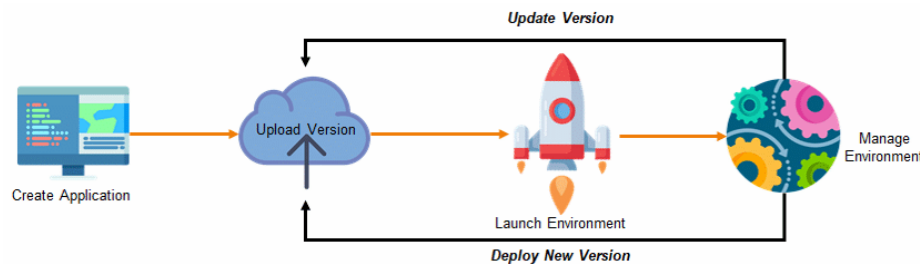
The resulting data is written in a specified place, such as an Amazon S3 bucket, in the last step. To run the data, the steps are performed in the following order:

1. To begin the procedural processes, a request is filed.
2. All steps' states are set to PENDING.
3. The state of the sequence changes to RUNNING when the first step begins. The other stages are still shown as PENDING.
4. When the first step is finished, the status of the step switches to COMPLETED.
5. The next step in the series begins, and the status of the sequence is changed to RUNNING. Its status switches to COMPLETED after it's finished.
6. This procedure is repeated for each stage until they are all finished and the processing is finished.

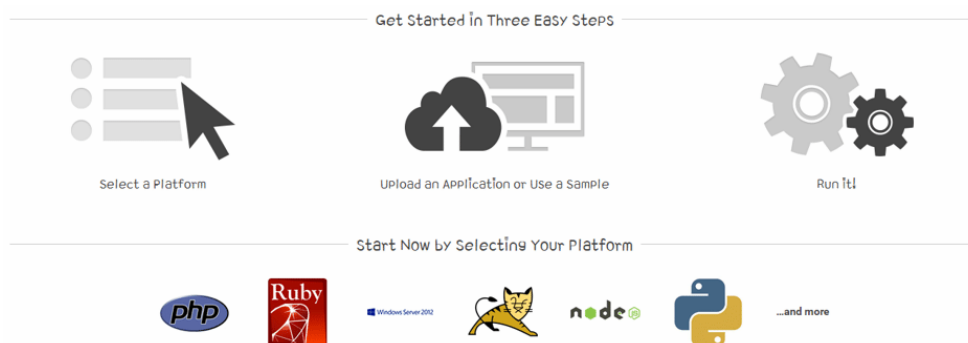
What is Elastic Beanstalk in AWS?

AWS Elastic Beanstalk is a [compute](#) service that makes it easier for the developers to quickly deploy and manage applications that you upload to the AWS cloud. Developers simply upload their application to the AWS cloud, and then let the AWS Beanstalk provision and handle the configuration for you. Your application will be provided with Capacity Provisioning, Load Balancing, [AWS Auto-Scaling](#), and Health monitoring.

How does Elastic Beanstalk in AWS work?



Before using Amazon elastic beanstalk service, you have to create a local application of any platform. It can be [Python](#), PHP, Node.js, etc. After that, you have to create an application in Elastic Beanstalk with an environment where you can upload your local application. Then you deploy it and use the URL provided for it to launch it.



AWS Elastic Beanstalk Benefits

Now that we understood AWS Elastic Beanstalk and how it works, let us now understand its benefits. So, Elastic Beanstalk provides the user with several benefits and they are:

- Easy to start with
- Autoscaling options
- Developer productivity
- Customization
- Cost-effective
- Management and updates

Easy to start with

- The fastest and easy way to upload your application and keep it running is by uploading it to Elastic Beanstalk.
- You need not worry about the platform of your application, you can create it on your local system and upload it.

Autoscaling options

- Beanstalk takes care of scaling up or down whenever required. If your application's traffic increases or decreases, beanstalk automatically scales it accordingly.

Developer productivity

- Developers don't need to think much about uploading their application online, they only have to concentrate on keeping their application more secure and user friendly.

Customization

- AWS Elastic Beanstalk allows you to select the configuration of the [AWS services](#) that you have used with your application. For example, consider [Amazon EC2](#), you can change the [AWS EC2 instance types](#) which is optimal for your application. Also, if you want to take control of some services manually, you can change the settings according to it.

Cost-Effective

- There is no cost involved in creating a Beanstalk environment. When there is a need for making it into the production of the application, then you can create your application bigger.

Management and Updates

- You don't need to worry about updating your application according to the change in the platform. The software patches, platform updates, and infrastructure management are taken care of by the AWS professionals.