What is EMR?

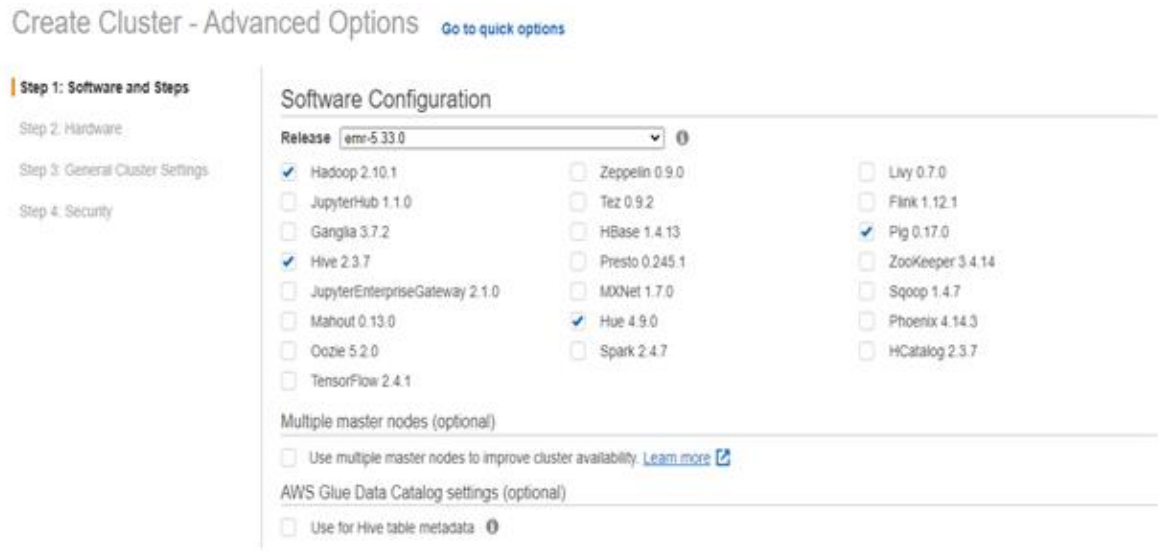
EMR Stands for Elastic Map Reduce and what it really is a managed Hadoop framework that runs on EC2 instances. So basically, Amazon took the Hadoop ecosystem and provided a runtime platform on EC2.

EMR is an AWS Service, but you do have to specify

- **Servers** — Type of Servers e.g. General Purpose, Compute Optimized…
- **Number of Instances** — One master node is compulsory and we can have n number of secondary nodes,
- **Software Configuration** — Spark, Hadoop, Hive, etc…

EMR lets you create managed instances and provides access to Servers to view logs, see configuration, troubleshoot, etc. So, for example, if we want Apache Spark installed on our EMR cluster and if we want to get down and dirty and actually have low-level access to Apache Spark and want to be able to have explicit control over the resources that it has, instead of having this totally opaque system like we can do with services as Glue ETL, where you don't see the servers, then EMR might be for you.

There's a lot of Big data applications and open-source software tools that we can pre-install, or we can install and configure ourselves on EMR by just checking a checkbox. We can include applications such as HBase or Presto or Flink or Hive and more as shown in the below figure.



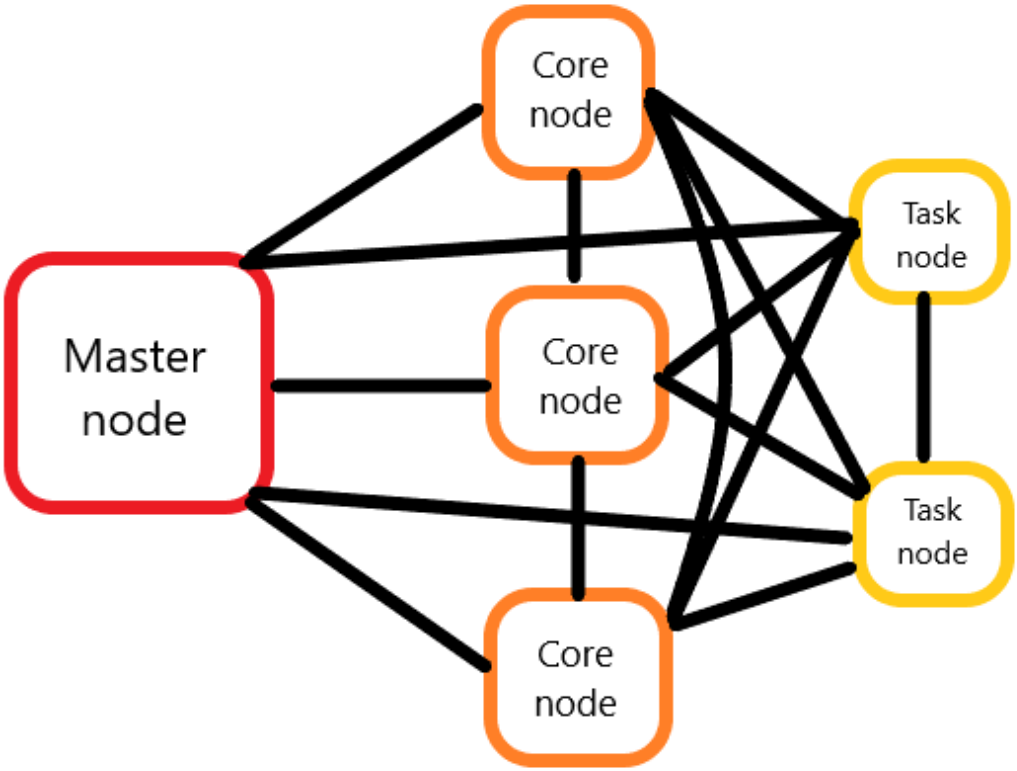
Applications available on EMR

Why do we need EMR?

- Companies have found that Operating Big data frameworks such as Spark and Hadoop are difficult, expensive, and time-consuming.
- Amazon EMR makes deploying spark and Hadoop easy and cost-effective.
- It decouples compute and storage allowing both of them to grow independently leading to better resource utilization.

- EMR allows you to store data in Amazon S3 and run compute as you need to process that data.
- We can launch an EMR cluster in minutes, we don't need to worry about node provisioning, cluster setup, Hadoop configuration, or cluster tuning once the processing is over, we can switch off the clusters.
- We can automatically resize clusters to accommodate Peaks and scale them down.
- We can run multiple clusters in parallel, allowing each of them to share the same data set.
- It monitors your cluster, retries on failed tasks, and automatically replacing poorly performing instances.

Understanding Clusters and Nodes



EMR cluster

The central component of Amazon EMR is the C*luster*. It is a collection of EC2 instances. Each instance within the cluster is named a *node* and every node has certain a role within the cluster, referred to as the *node type*. Amazon EMR also installs different software components on each node type, which provides each node a specific role in a distributed application like Apache Hadoop.

The node types in Amazon EMR are as follows:

Master Node: It manages the clusters, can be referred to as Primary node or Leader Node.

- It manages the cluster resources. It essentially coordinates the distribution of the parallel execution for the various Map-Reduce tasks. We can think about it as the leader that's handing out tasks to its various employees.
- It tracks and directs the HDFS. Therefore, the master node knows the way to lookup files and tracks the info that runs on the core nodes.
- With 5.23.0+ versions we have the ability to select three master nodes. Multiple master nodes are for mitigating the risk of a single point of failure. So, if one master node fails, the cluster uses the other two master nodes to run without any interruptions and what EMR does is automatically replaces the master node and provisions it with any configurations or bootstrap actions that need to happen.
- The master node is also responsible for the YARN resource management. Its job is to centrally manage the cluster resources for multiple data processing frameworks. So, it's the master node's job to allocate to manage all of these data processing frameworks that the cluster uses.
- It also performs monitoring and health on the core and task nodes. So, its job is to make sure that the status of the jobs that are submitted should be in good health, and that the core and tasks nodes are up and running.

Core Nodes: It hosts HDFS data and runs tasks

- They run tasks for the primary node. So, the primary node manages all of the tasks that need to be run on the core nodes and these can be things like Map Reduce tasks, Hive scripts, or Spark applications.
- The core node is also responsible for coordinating data storage. So, it knows about all of the data that's stored on the EMR cluster and it runs the data node Daemon. This means that it breaks apart all of the files within the HDFS file system into blocks and distributes that across the core nodes
- we know that we can have multiple core nodes, but we can only have one core instance group and we'll talk more about what instance groups are or what instance fleets are and just a little while, but just remember, and just keep it in your brain and you can have multiple core nodes, but you can only have one core instance group.

Task Nodes: Runs tasks, but doesn't host data

- These nodes are optional helpers, meaning that you don't have to actually spin up any tasks nodes whenever you spin up your EMR cluster, or whenever you run your EMR jobs, they're optional and they can be used to provide parallel computing power for tasks like Map-Reduce jobs or spark applications or the other job that you simply might run on your EMR cluster.
- It does not store any data in HDFS. So there is no risk of data loss on removing. It's not used as a data store and doesn't run data Node Daemon.
- They are often added or removed on the fly from the cluster. So this will help scale up any extra CPU or memory for compute-intensive applications.
- It can cut down the all-over cost in an effective way if we choose spot instances for extra processing.

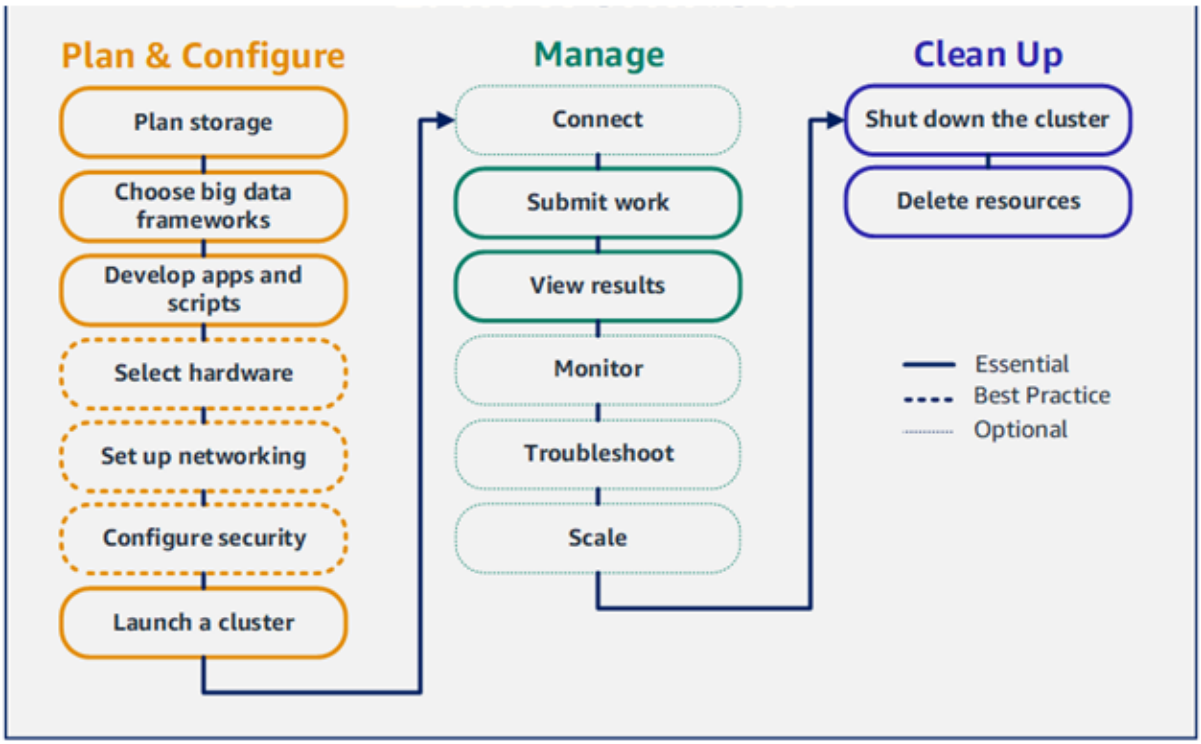
EMR Storage Options

Amazon EMR and Hadoop provide several file systems that you can use when processing cluster steps.

The following table lists the available file systems, Description with recommendations about when it's best to use each one.

File System	Description	Usage
Local File System (Instance Storage)	• Located on disks that are attached to the host machine. • Size/Speed is determined by instance type	• Used for very high I/O performance and high IOPS at a low cost. • Best used for temporary data (caches, buffer).
Local File System (EBS volume)	• Used to Extend HDFS • These volumes are ephemeral	• Used to add more storage to HDFS
Hadoop Distributed File System (HDFS)	• HDFS is a distributed, scalable, and portable file system for Hadoop. • Configurable Replication Factor • Fast, but ephemeral • Distributed storage for EMR	• It is used by the master and core nodes. • Great for jobs with iterative reads on the same dataset or Disk I/O intensive workloads • Used for caching the results produced by intermediate job-flow steps.
Elastic Map Reduce File System (EMRFS)	• It's an implementation of the Hadoop file system used for reading and writing regular files from Amazon EMR directly to Amazon S3 • Feature-rich persistent storage.	• Best used for persistent Store • It adds features like data encryption and Consistency view.

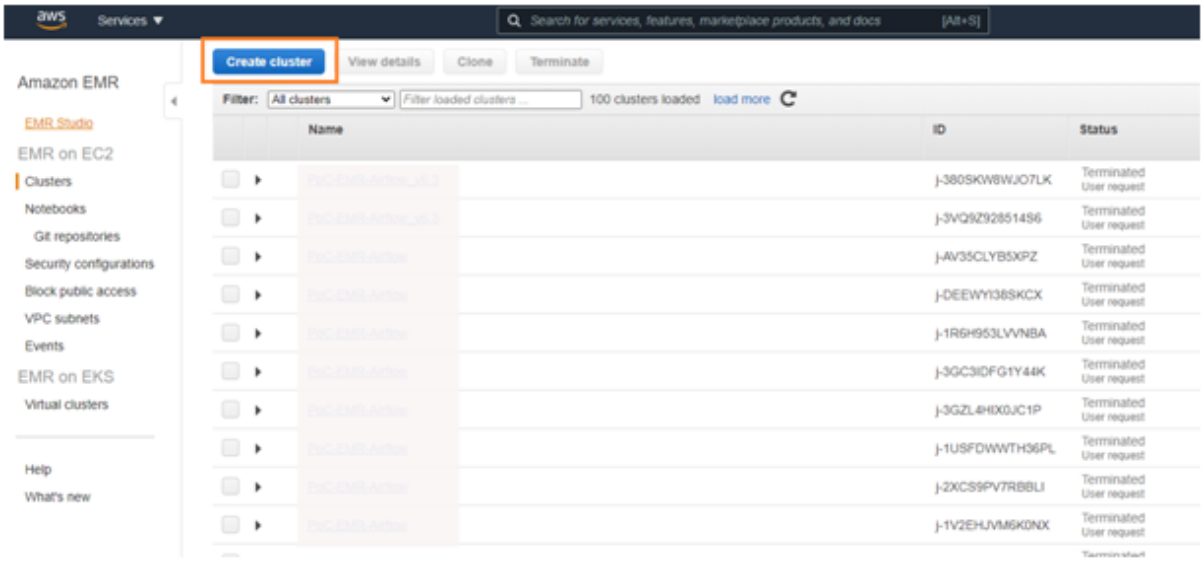
EMR Setup



EMR Workflow

We can quickly set up an EMR cluster in AWS Web Console; then We can deploy the Amazon EMR and all we need is to provide some basic configurations as follows.

1. Before you launch the EMR cluster for SSH, or if you already have an Amazon EC2 key pair that you want to use, or you don't need to authenticate to your cluster. Then skip this step.



EMR console

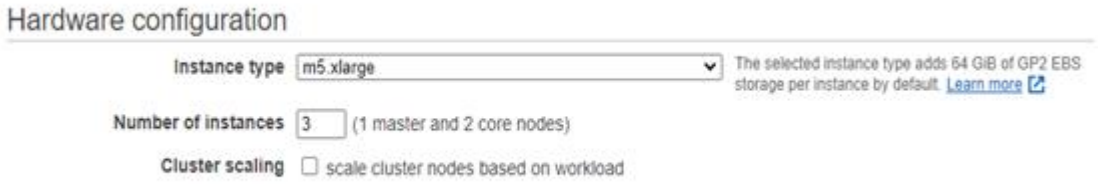
2. We need to give the Cluster name of our choice and we need a point to an S3 folder for storing the logs. If we need to terminate the cluster after steps executions then select the option otherwise leaves default long-running cluster launch mode.

3. We then choose the software configuration for a version of EMR. Amazon is constantly updating them as well as what versions of various software that we want to have on EMR. In the quick option, they provide some applications in bundles or we can customize these bundles in advance UI option. Tick Glue data Catalog when you require a persistent metastore or a metastore shared by different clusters, services, applications, or AWS accounts.

4. Then we tell it how many nodes that we want to have running as well as the size. This is just the quick options and we can configure it to be specific for each type of master node in each type of secondary nodes. We can configure what type of EC2 instance that we want to have running. Refer to the below table to choose the right hardware for your job.

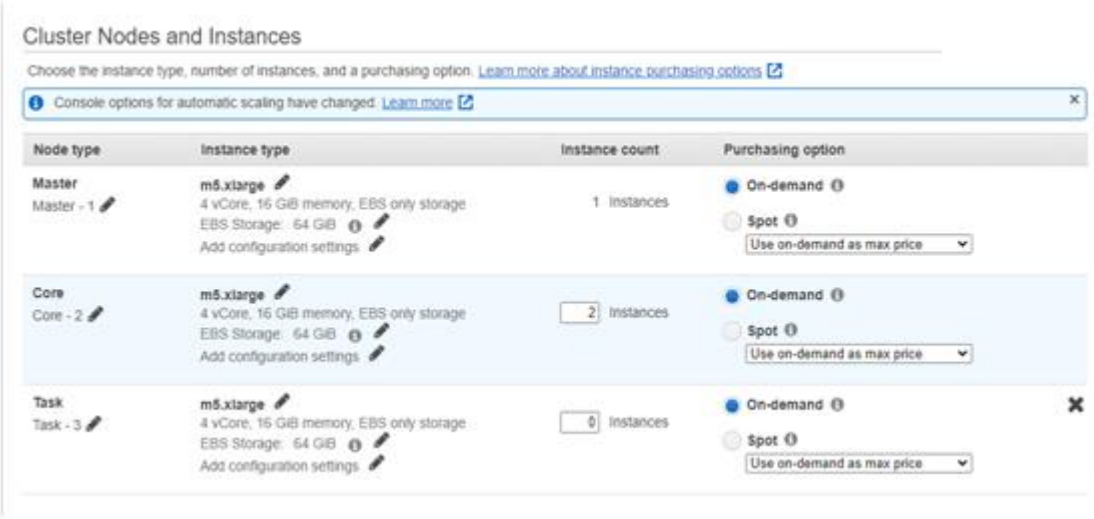
Instance Class	Instance Type Family	Best Used for
General Purpose	M4, M5	Batch Process
Compute-optimized	C5, C4	Machine Learning
Memory-optimized	X1, R4	Interactive Analysis
Storage-optimized	D2, I3	Large HDFS

Hardware Options for EMR



Quick Option Hardware Configuration

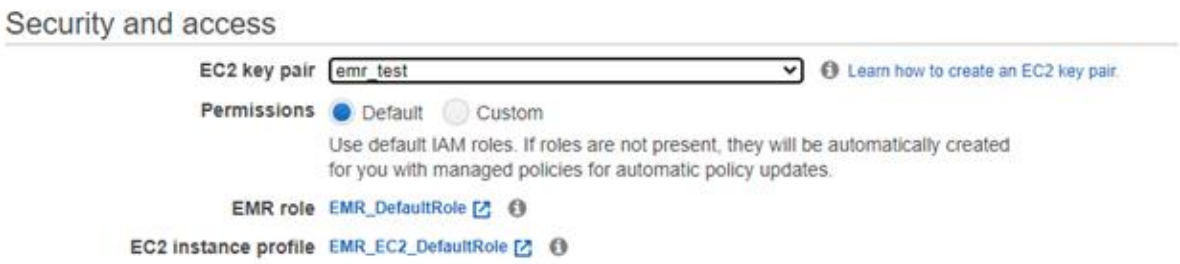
5. EMR will charge you at a per-second rate and pricing varies by region and deployment option. For more pricing information, see Amazon EMR pricing and EC2 instance type pricing granular comparison details please refer to EC2Instances.info.



Advance option Hardware configuration

6. Then, we have security access for the EMR cluster where we just set up an SSH key if we want to SSH into the master node or we can also connect via other types of methods like ForxyProxy or SwitchyOmega. Secondary nodes can only talk to the master node via the security group by default and we can change that if required.

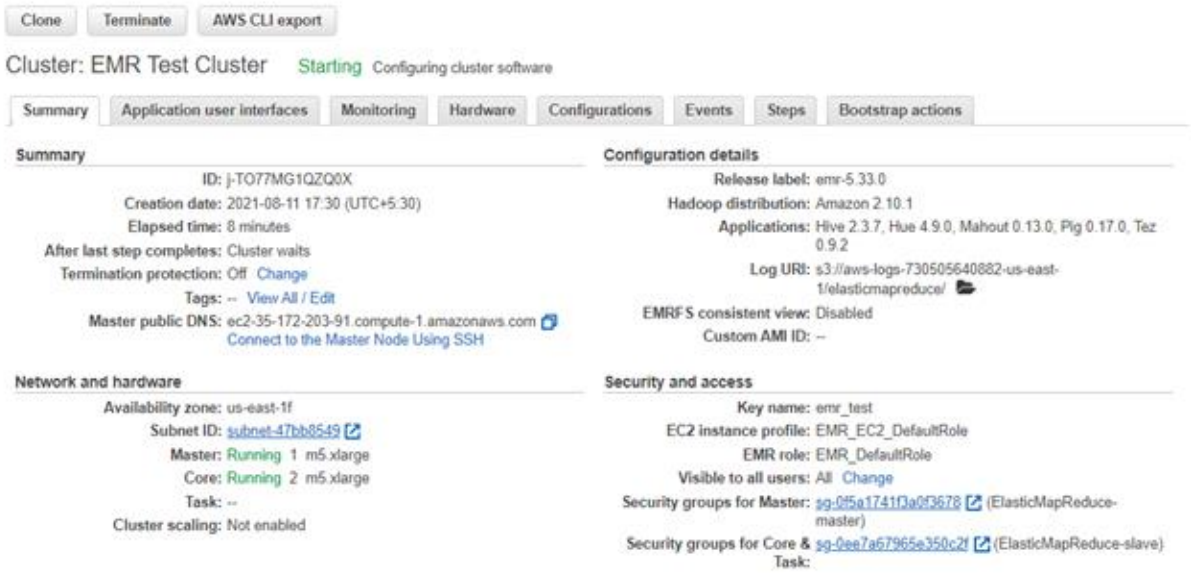
7. We have a couple of pre-defined roles that need to be set up in IAM or we can customize it on our own. It will help us to interact with things like Redshift, S3, DynamoDB, and any of the other services that we want to interact with.



Finally, Node is up and running. We have a summary where we can see the creation date and master node DNS to SSH into the system.

Then we have certain details that will tell us the details about software running under cluster, logs, and features.

We can also see the details about the hardware and security info in the summary section.



There are other options to launch the EMR cluster, like CLI, IaC (Terraform, CloudFormation..) or we can use our favorite SDK to configure. It gives us a way to programmatically Access to Cluster Provisioning using API or SDK. This is how we can build the pipeline. Like when the data arrives, spin up the EMR cluster, process the data, and then just terminate the cluster.

Note: EMR cluster can’t be restored once, they have been terminated without Snapshot.