Lecture
on the
Introduction to Amazon Elastic MapReduce (EMR)

Natalia Myronova
Scientific Researcher,
University of Applied Science and Arts Dortmund, Germany,
Assoc. Prof. of Department of Information Technologies of Electronic Devices,
National University “Zaporizhzhia Polytechnic”, Ukraine
natali.myronova@gmail.com

Winter School on Advanced Perspectives on Data Science, 15.11.2021 – 19.11.2021

Content Overview

What is Amazon EMR?

Amazon Elastic Map Reduce (Amazon EMR) is a managed cluster platform that simplifies running big data frameworks, such as **Apache Hadoop** and **Apache Spark**, on **AWS** to process and analyze vast amounts of data. Using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads.

Amazon EMR also lets you transform and move large amounts of data into and out of other **AWS** data stores and databases, such as **Amazon Simple Storage Service (Amazon S3)** and **Amazon DynamoDB.**

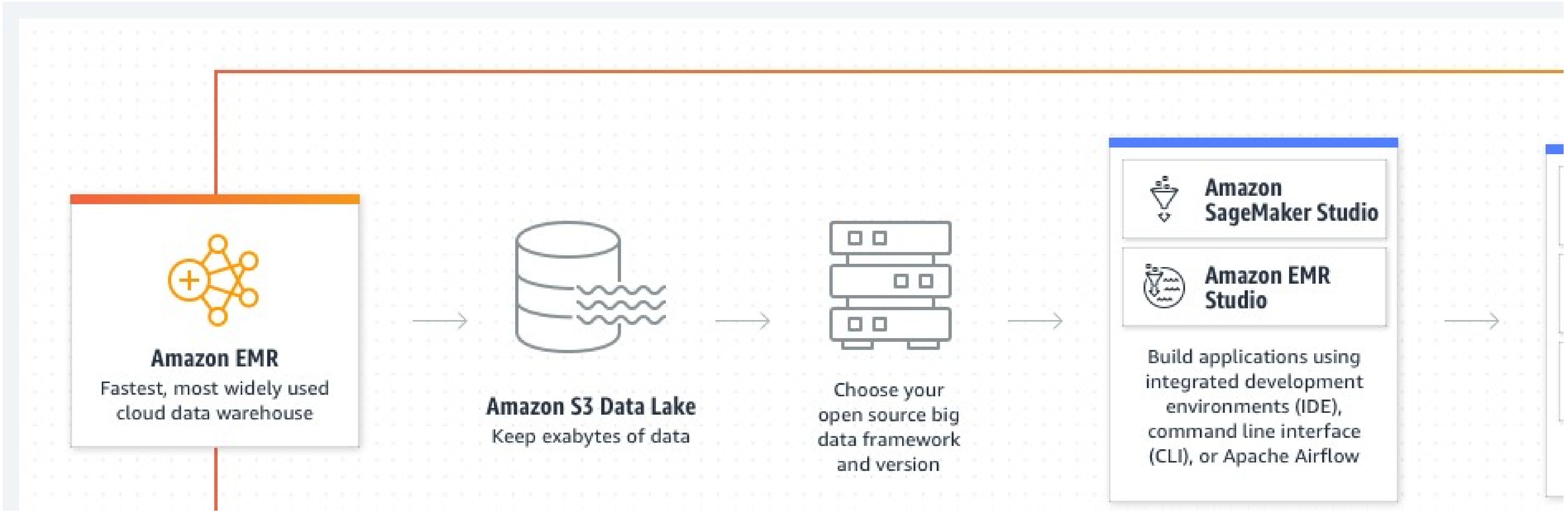
Amazon EMR allows:

•easily run and scale Apache Spark, Hive, Presto, and other big data workloads;
•simplify management with rapid cluster provisioning, managed scaling, and automated software installation;
•secure your data and resources through customizable permissions in AWS Identity and Access Management (IAM), AWS Lake Formation, and Apache Ranger;
•run big data applications and petabyte-scale analysis faster, and at less than half the cost of on-premises solutions.

Source: https://www.javatpoint.com/what-is-amazon-emr, https://aws.amazon.com/emr/?nc1=h_ls

How it works?

Amazon EMR is a platform for rapidly processing, analyzing, and applying machine learning (ML) to big data using open-source frameworks.


Amazon EMR
Fastest, most widely used cloud data warehouse
Amazon S3 Data Lake
Keep exabytes of data
Choose your open source big data framework and version
Amazon SageMaker Studio
Amazon EMR Studio
Build applications using integrated development environments (IDE), command line interface (CLI), or Apache Airflow

Use cases

•**Build scalable data pipelines**: extract data from a variety of sources, process it at scale, and make it available for both applications and users. Leverage Apache Hudi features for petabyte scale data lakes.

•**Accelerate data science with ML**: use open-source ML frameworks such as Apache Spark MLlib, TensorFlow, and Apache MXNet built into EMR. Connect to SageMaker Studio for analysis, reporting, and model training.

•**Process real-time data streams**: analyze events from streaming data sources in real time to create long-running, highly available, and fault-tolerant streaming data pipelines.

•**Query any dataset**: query datasets from different data stores including object storage, relational databases, NoSQL databases, and more using open-source SQL tools.

Benefits of using Amazon EMR

There are many benefits to using Amazon EMR:

- Cost savings
- AWS integration
- Deployment
- Scalability and flexibility
- Reliability
- Security
- Monitoring
- Management interfaces

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Cost savings

Amazon EMR pricing depends on the instance type and number of Amazon EC2 instances that you deploy and the Region in which you launch your cluster.

On-demand pricing offers low rates, but you can reduce the cost even further by purchasing Reserved Instances or Spot Instances. Spot Instances can offer significant savings — as low as a tenth of on-demand pricing in some cases.

If you use Amazon S3, Amazon Kinesis, or DynamoDB with your EMR cluster, there are additional charges for those services that are billed separately from your Amazon EMR usage.

For more information about pricing options and details, see Amazon EMR pricing.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: AWS integration

Amazon EMR integrates with other AWS services to provide capabilities and functionality related to networking, storage, security, and so on, for your cluster. The following list provides several examples of this integration:

- Amazon EC2 for the instances that comprise the nodes in the cluster
- Amazon Virtual Private Cloud (Amazon VPC) to configure the virtual network in which you launch your instances
- Amazon S3 to store input and output data
- Amazon CloudWatch to monitor cluster performance and configure alarms
- AWS Identity and Access Management (IAM) to configure permissions
- AWS CloudTrail to audit requests made to the service
- AWS Data Pipeline to schedule and start your clusters
- AWS Lake Formation to discover, catalog, and secure data in an Amazon S3 data lake

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Deployment

EMR cluster consists of EC2 instances, which perform the work that you submit to your cluster. When you launch your cluster, Amazon EMR configures the instances with the applications that you choose, such as Apache Hadoop or Spark. Choose the instance size and type that best suits the processing needs for your cluster: batch processing, low-latency queries, streaming data, or large data storage. For more information about the instance types available for Amazon EMR, see Configure cluster hardware and networking.

Amazon EMR offers a variety of ways to configure software on your cluster. For example, you can install an Amazon EMR release with a chosen set of applications that can include versatile frameworks, such as Hadoop, and applications, such as Hive, Pig, or Spark. You can also install one of several MapR distributions. Amazon EMR uses Amazon Linux, so you can also install software on your cluster manually using the yum package manager or from the source. For more information, see Configure cluster software.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Scalability and flexibility

- Amazon EMR provides flexibility to scale your cluster up or down as your computing needs change. You can resize your cluster to add instances for peak workloads and remove instances to control costs when peak workloads subside.

- Amazon EMR also provides the option to run multiple instance groups so that you can use On-Demand Instances in one group for guaranteed processing power together with Spot Instances in another group to have your jobs completed faster and at lower costs. You can also mix different instance types to take advantage of better pricing for one Spot Instance type over another.

- Amazon EMR provides the flexibility to use several file systems for your input, output, and intermediate data. For example, you might choose the Hadoop Distributed File System (HDFS) which runs on the master and core nodes of your cluster for processing data that you do not need to store beyond your cluster's lifecycle. You might choose the EMR File System (EMRFS) to use Amazon S3 as a data layer for applications running on your cluster so that you can separate your compute and storage, and persist data outside of the lifecycle of your cluster. EMRFS provides the added benefit of allowing you to scale up or down for your compute and storage needs independently. You can scale your compute needs by resizing your cluster and you can scale your storage needs by using Amazon S3.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Reliability

- **Amazon EMR** monitors nodes in your cluster and automatically terminates and replaces an instance in case of failure.

- **Amazon EMR** provides configuration options that control if your cluster is terminated automatically or manually. If you configure your cluster to be automatically terminated, it is terminated after all the steps complete. This is referred to as a transient cluster. However, you can configure the cluster to continue running after processing completes so that you can choose to terminate it manually when you no longer need it. Or, you can create a cluster, interact with the installed applications directly, and then manually terminate the cluster when you no longer need it. The clusters in these examples are referred to as long-running clusters.

- Additionally, you can configure termination protection to prevent instances in your cluster from being terminated due to errors or issues during processing. When termination protection is enabled, you can recover data from instances before termination. The default settings for these options differ depending on whether you launch your cluster by using the console, CLI, or API.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Security (1/2)

Amazon EMR leverages other AWS services, such as Identity and Access Management(IAM) and Amazon VPC, and features such as Amazon EC2 key pairs, to help you secure your clusters and data.

•**IAM**: Amazon EMR integrates with IAM to manage permissions. You define permissions using IAM policies, which you attach to IAM users or IAM groups. The permissions that you define in the policy determine the actions that those users or members of the group can perform and the resources that they can access.

•**Security groups**: Amazon EMR uses security groups to control inbound and outbound traffic to your EC2 instances.

•**Encryption:** Amazon EMR supports optional Amazon S3 server-side and client-side encryption with EMRFS to help protect the data that you store in Amazon S3. With server-side encryption, Amazon S3 encrypts your data after you upload it. With client-side encryption, the encryption and decryption process occurs in the EMRFS client on your EMR cluster. You manage the master key for client-side encryption using either the AWS Key Management Service (AWS KMS) or your own key management system.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Security (2/2)

•**Amazon VPC**: Amazon EMR supports launching clusters in a virtual private cloud (VPC) in Amazon VPC. A VPC is an isolated, virtual network in AWS that provides the ability to control advanced aspects of network configuration and access

•**AWS CloudTrail**: Amazon EMR integrates with CloudTrail to log information about requests made by or on behalf of your AWS account. With this information, you can track who is accessing your cluster when, and the IP address from which they made the request.

•**Amazon EC2 key pairs**: You can monitor and interact with your cluster by forming a secure connection between your remote computer and the master node. You use the Secure Shell (SSH) network protocol for this connection or use Kerberos for authentication. If you use SSH, an Amazon EC2 key pair is required.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Management interfaces (1/2)

There are several ways you can interact with Amazon EMR:

Console - A graphical user interface that you can use to launch and manage clusters. With it, you fill out web forms to specify the details of clusters to launch, view the details of existing clusters, debug, and terminate clusters. Using the console is the easiest way to get started with Amazon EMR; no programming knowledge is required. The console is available online at https://console.aws.amazon.com/elasticmapreduce/home.

AWS Command Line Interface (AWS CLI) - A client application you run on your local machine to connect to Amazon EMR and create and manage clusters. The AWS CLI contains a feature-rich set of commands specific to Amazon EMR. With it, you can write scripts that automate the process of launching and managing clusters. If you prefer working from a command line, using the AWS CLI is the best option. For more information, see Amazon EMR in the AWS CLI Command Reference.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Benefits of using Amazon EMR: Management interfaces (2/2)

Software Development Kit (SDK) - SDKs provide functions that call Amazon EMR to create and manage clusters. With them, you can write applications that automate the process of creating and managing clusters. Using the SDK is the best option to extend or customize the functionality of Amazon EMR. Amazon EMR is currently available in the following SDKs: Go, Java, .NET (C# and VB.NET), Node.js, PHP, Python, and Ruby. For more information about these SDKs, see Tools for AWS and Amazon EMR sample code & libraries.

Web Service API - A low-level interface that you can use to call the web service directly, using JSON. Using the API is the best option to create a custom SDK that calls Amazon EMR. For more information, see the Amazon EMR API Reference..

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-benefits.html

Understanding clusters and nodes

The central component of **Amazon EMR** is the **cluster**. A cluster is a collection of **Amazon Elastic Compute Cloud (Amazon EC2)** instances. Each instance in the cluster is called a **node**. Each node has a role within the cluster, referred to as the **node type**. **Amazon EMR** also installs different software components on each node type, giving each node a role in a distributed application like **Apache Hadoop**.

The **node types** in **Amazon EMR** are as follows:

• **Master node**: A node that manages the cluster by running software components to coordinate the distribution of data and tasks among other nodes for processing. The master node tracks the status of tasks and monitors the health of the cluster. Every cluster has a master node, and it's possible to create a single-node cluster with only the master node.

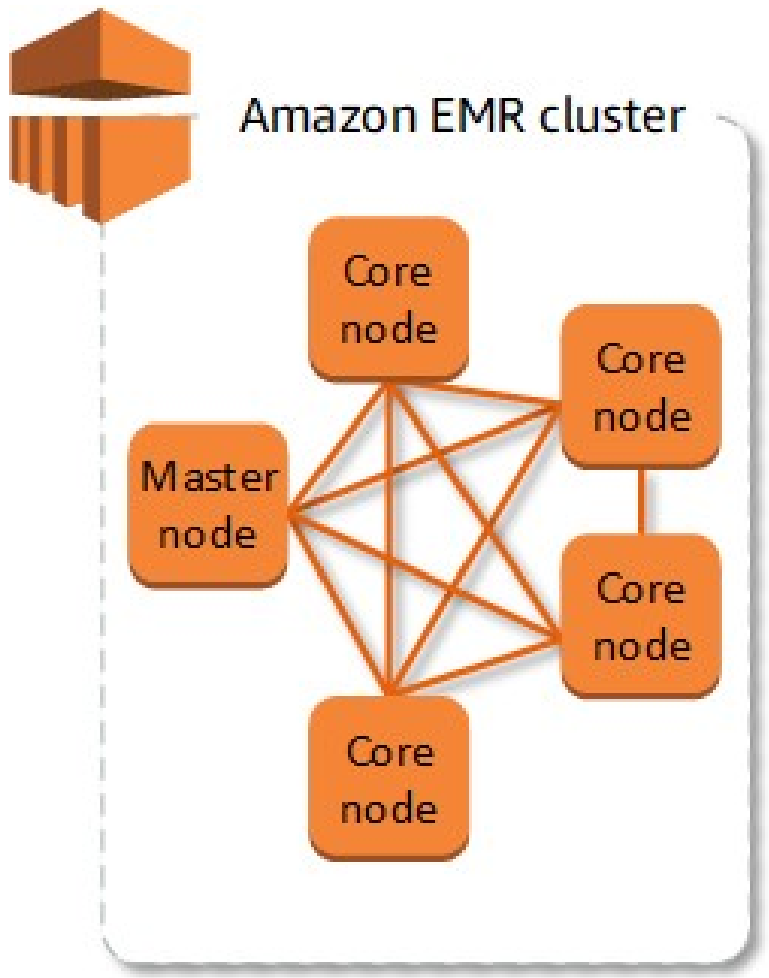
• **Core node**: A node with software components that run tasks and store data in the Hadoop Distributed File System (HDFS) on your cluster. Multi-node clusters have at least one core node.

•**Task node**: A node with software components that only runs tasks and does not store data in HDFS. Task nodes are optional.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html#emr-overview-clusters

Example of Amazon EMR cluster

The following diagram represents a cluster with one master node and four core nodes.



Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html#emr-overview-clusters

Overview of Amazon EMR architecture

Amazon EMR service architecture consists of several layers, each of which provides certain capabilities and functionality to the cluster:

- **Storage**: Hadoop Distributed File System (HDFS) , EMR File System (EMRFS), Local file system;
- **Cluster resource management:** YARN(Yet Another Resource Negotiator);
- **Data processing frameworks**: Hadoop MapReduce and Spark;
- **Applications and programs:** Hive, Pig, and the Spark Streaming

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-arch.html

Storage

The storage layer includes the different file systems that are used with your cluster. There are several different types of storage options as follows.

- **HDFS** is a distributed, scalable file system for Hadoop. HDFS distributes the data it stores across instances in the cluster, storing multiple copies of data on different instances to ensure that no data is lost if an individual instance fails. HDFS is ephemeral storage that is reclaimed when you terminate a cluster. HDFS is useful for caching intermediate results during MapReduce processing or for workloads that have significant random I/O.

- **EMRFS** – using this file system, **Amazon EMR** extends Hadoop to add the ability to directly access data stored in Amazon S3 (Amazon Simple Storage Service) as if it were a file system like HDFS. You can use either HDFS or Amazon S3 as the file system in your cluster. Most often, Amazon S3 is used to store input and output data and intermediate results are stored in HDFS.

- **Local file system:** The local file system refers to a locally connected disk. When you create a Hadoop cluster, each node is created from an Amazon EC2 instance that comes with a preconfigured block of pre-attached disk storage called an instance store. Data on instance store volumes persists only during the lifecycle of its Amazon EC2 instance.

Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-arch.html

Cluster resource management

The **resource management layer** is responsible for managing cluster resources and scheduling the jobs for processing data.

- By default, **Amazon EMR** uses **YARN (Yet Another Resource Negotiator),** which is a component introduced in **Apache Hadoop 2.0** to centrally manage cluster resources for multiple data-processing frameworks. However, there are other frameworks and applications that are offered in Amazon EMR that do not use YARN as a resource manager. Amazon EMR also has an agent on each node that administers YARN components, keeps the cluster healthy, and communicates with Amazon EMR.

- Because Spot Instances are often used to run task nodes, Amazon EMR has default functionality for scheduling YARN jobs so that running jobs do not fail when task nodes running on Spot Instances are terminated. Amazon EMR does this by allowing application master processes to run only on core nodes. The application master process controls running jobs and needs to stay alive for the life of the job.

Data processing frameworks

The **data processing framework layer** is the engine used to process and analyze data. The main processing frameworks available for **Amazon EMR** are **Hadoop MapReduce** and **Spark**.

- **Hadoop MapReduce** is an open-source programming model for distributed computing. It simplifies the process of writing parallel distributed applications by handling all of the logic, while you provide the **Map** and **Reduce** functions. The **Map** function maps data to sets of key-value pairs called intermediate results. The **Reduce** function combines the intermediate results, applies additional algorithms, and produces the final output. There are multiple frameworks available for **MapReduce**, such as **Hive**, which automatically generates **Map** and **Reduce** programs.

- **Spark** is a cluster framework and programming model for processing big data workloads. Like **Hadoop MapReduce**, **Spark** is an open-source, distributed processing system but uses directed acyclic graphs for execution plans and in-memory caching for datasets. When you run Spark on **Amazon EMR**, you can use **EMRFS** to directly access your data in Amazon S3. Spark supports multiple interactive query modules such as **SparkSQL**.

How to use Amazon EMR(1/2)

•**Develop your data processing application**. You can use Java, Hive (a SQL-like language), Pig (a data processing language), Cascading, Ruby, Perl, Python, R, PHP, C++, or Node.js. Amazon EMR provides code samples and tutorials to get you up and running quickly.

•**Upload your application and data to Amazon S3**. If you have a large amount of data to upload, you may want to consider using AWS Import/Export Snowball, to upload data using physical storage devices; or AWS Direct Connect, to establish a dedicated network connection from your data center to AWS. If you prefer, you can also write your data directly to a running cluster.

•**Configure and launch your cluster**. Using the AWS Management Console, the AWS CLI, SDKs, or APIs, specify the number of Amazon EC2 instances to provision in your cluster, the types of instances to use (standard, high memory, high CPU, high I/O, etc.), the applications to install (Apache Spark, Apache Hive, Apache HBase, Presto, etc.), and the location of your application and data. You can use Bootstrap Actions to install additional software or change default settings.

Source: https://aws.amazon.com/emr/getting-started/?pg=ln&sec=hs

How to use Amazon EMR(2/2)

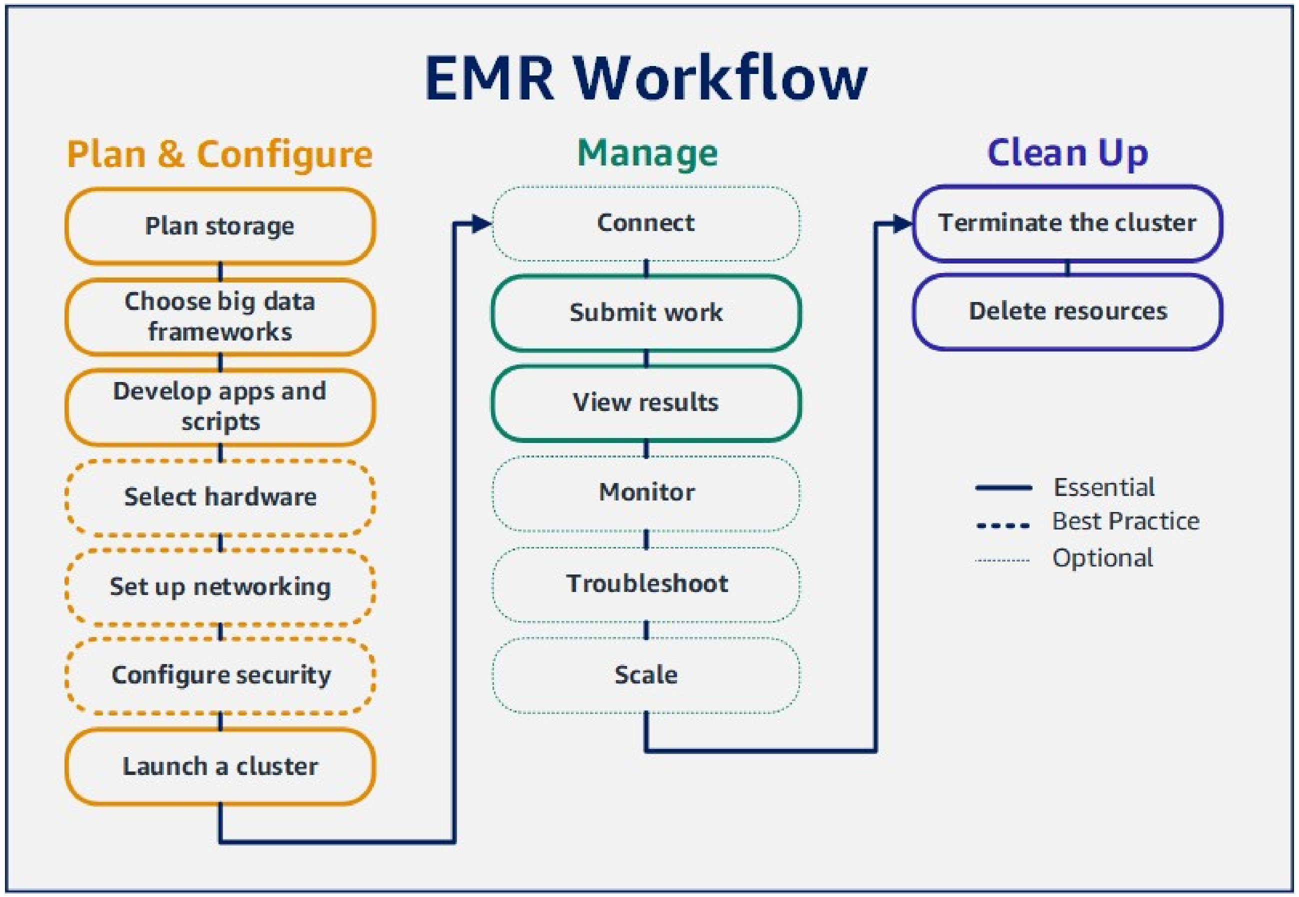
•**Monitor the cluster**. You can monitor the cluster's health and progress using the Management Console, Command Line Interface, SDKs, or APIs. EMR integrates with Amazon CloudWatch for monitoring/alarming and supports popular monitoring tools like Ganglia. You can add/remove capacity to the cluster at any time to handle more or less data. For troubleshooting, you can use the console's simple debugging GUI.

•**Retrieve the output.** Retrieve the output from Amazon S3 or HDFS on the cluster. Visualize the data with tools like Amazon QuickSight, Tableau and MicroStrategy. Amazon EMR will automatically terminate the cluster when processing is complete. Alternatively you can leave the cluster running and give it more work to do.

Source: https://aws.amazon.com/emr/getting-started/?pg=ln&sec=hs

Getting started with Amazon EMR: Overview

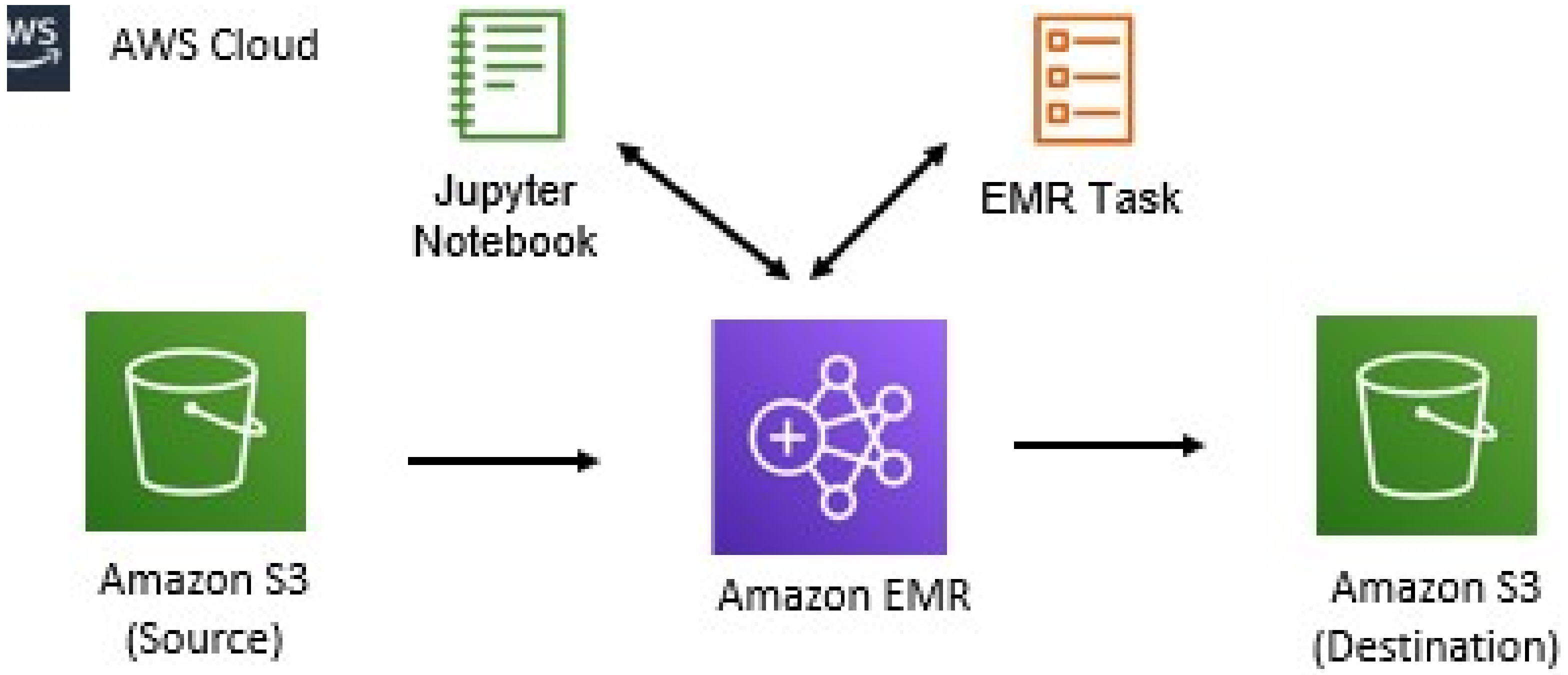
With Amazon EMR you can set up a cluster to process and analyze data with big data frameworks in just a few minutes. This tutorial shows you how to launch a sample cluster using Spark, and how to run a simple PySpark script stored in an Amazon S3 bucket. It covers essential Amazon EMR tasks in three main workflow categories: Plan and Configure, Manage, and Clean Up.



Source: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html 24

Getting started with Amazon EMR: Step-by-Step

Amazon EMR is a big data platform for processing large scale data using open source tools such as **Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi,** and **Presto**. **Amazon EMR** is easy to set up, operate, and scale for the big data requirement by automating time-consuming tasks like provisioning capacity and tuning clusters. In this workshop, you launch an **EMR cluster**. You then use Jupyter Notebook to do PySpark based programming with EMR Cluster. Finally, you launch a data processing task using EMR Cluster. The following diagram shows the scenario you are going to build.



Getting started with Amazon EMR: Pre-requisite

You need to have an **AWS account** with administrative access to complete the workshop. If you don't have an AWS account, you have to create free trial account for AWS.

Getting started with Amazon EMR: Create S3 Bucket

You create **S3 bucket** and **folder** which is used as the source and destination data location when you process data using **EMR cluster**.

Step 1. Login to **AWS Management Console** and change the region.

Step 2. Goto S3 Management Console. Create a bucket with the name **workshop-data**. If the bucket name is not available, create bucket with a name which is available. In this bucket, create three folders - **input**, **output** and **script**.

Step 3. The **input** folder is used as the input location for the data to be processed by the **EMR cluster**. The **output folder** is the destination location where **EMR cluster** will write the processed data output. The script folder is used to store the script when creating a task within the **EMR cluster**.

Step 4. Download data.csv file. Upload the data.csv file to the input folder.

Step 5. Open the data.csv file to get familiar with the data you will work with. It is a sample data. The S3 configuration is ready. In the next step, you launch EMR cluster.

Getting started with Amazon EMR: Launch EMR Cluster

You launch EMR cluster which is used to process data using PySpark based scripting.

Step 1. Goto the EMR Management console and click on the **Create cluster** button

Step 2. On the next screen, click on the **Go to advanced** options link.

Step 3. Use the default selection for the **Release** field. For the software configuration, make additional selection for JupyterEnterpriseGateway 2.1.0 and Spark 2.4.7 along with the default selection. Keep rest of the configuration to the default and click on the Next button.

Step 4. Select **Uniform instance groups** for the Instance group configuration. Select **default VPC** for the network and select one the default subnet. Keep rest of the configuration to the default and click on the Next button.

Step 5. Type in **workshop-cluste**r for the cluster name. Keep rest of the configuration to the default and click on the Next button.

Step 6. Keep the configuration to the default and click on the Create cluster button.

Step 7. The cluster creation will start. It will take some time to finish. Wait till the cluster status changes to Cluster ready.

Step 8. The cluster is ready. You launch Jupyter Notebook in the next step.

Getting started with Amazon EMR: Launch Jupyter Notebook

The **EMR cluster** is ready. You now launch **Jupyter Notebook** and associate with the **EMR cluster.**

Step 1. Goto the **EMR Management console**, click on the Notebooks menu in the left and click on the Create notebook button.

Step 2. Type in **workshop** for the notebook name. For the cluster, select **workshop-cluster** . Keep rest of the configuration to the default and click on the Create notebook button.

Step 3. It will start notebook creation. Wait till the status changes to Ready.

Step 4. The notebook is ready. You can now try **PySpark** code to process S3 based data in **Jupyter Notebook** using **EMR Cluster**.

Getting started with Amazon EMR: Coding in Notebook

You now run PySpark code to process S3 based data in the EMR Cluster.

Step 1. In the EMR Management console, on the dojocluster notebook page, click on the Open in Jupyter button.

Step 2. It will open Notebook environment in a new browser tab or window. In Jypyter environment, click on PySpark option under the New menu.

Step 3. It will open Jypyter notebook IDE. It the cell, copy-paste the programming code and run it. The code imports libraries for the PySpark.

Step 4. You copy-paste and run the following code to get the spark session.

Step 5. You copy-paste and run the code to read data.csv data from the s3 bucket and populate into df dataframe. If you created bucket with a different name, then use that bucket name. When reading the data, you are considering to infer the schema and also treating the first row as the header.

Step 6. You copy-paste and run the code to check the schema of the df dataframe.

Step 7. The code has written the output in the S3 bucket. You can verify it by navigating to the output location in the S3 bucket.

Step 8. This was an example to see how you can run PySpark code in Jypyter Notebook to perform data transformation. The notebook is mostly used for development purpose. In the next step, you run the same code using EMR Task. EMR Task is a method used in the production.

Getting started with Amazon EMR: Running Task

You learnt how to use Jupyter Notebook for PySpark coding. In this step, you submit the entire code as task in Jupyter notebook.

Step 1. Create a local file with the name **emrtask.p**y and copy-paste the following code into it. If you created bucket with a different name, replace the bucket name in this code with that name. It is the same code which you tried in the Jypyter Notebook but with little change. The input and output locations have been parameterized in order to avoid hard-coding.

Step 2. Upload the **emrtask.py** file to the script folder under **workshop-data** bucket. If you created bucket with a different name then use that bucket.

Step 3. Goto the **EMR Management console** and open the **workshop-cluster** EMR cluster details. Click on the Steps tab.

Step 4. Click on the Add step button.

Step 5. select Custom JAR for the step type. Type in emrtask for the name. Type in command-runner.jar for the JAR location. Copy-Paste spark-submit s3://workshop-data/script/dojoemrtask.py s3://workshop-data/input/customers.csv s3://workshop-data/output/taskoutput/ for the Arguments. If you created bucket with different name, use that one. You are providing three arguments - command to submit spark task, input file location and output folder location. The script above uses the input and output location in the code. Finally, click on the Add button.

Step 6. The step will be added and the execution will start in a while. Wait till the status of the task changes to Completed. The task has completed. It has written output to the output location in the S3 bucket. You can verify it.

Getting started with Amazon EMR: Clean up

Delete the resources used for the workshop to avoid any further cost.

- Delete workshop-cluster EMR Cluster.
- Delete workshop-cluster Jupyter Notebook.
- Delete S3 bucket workshop-data.

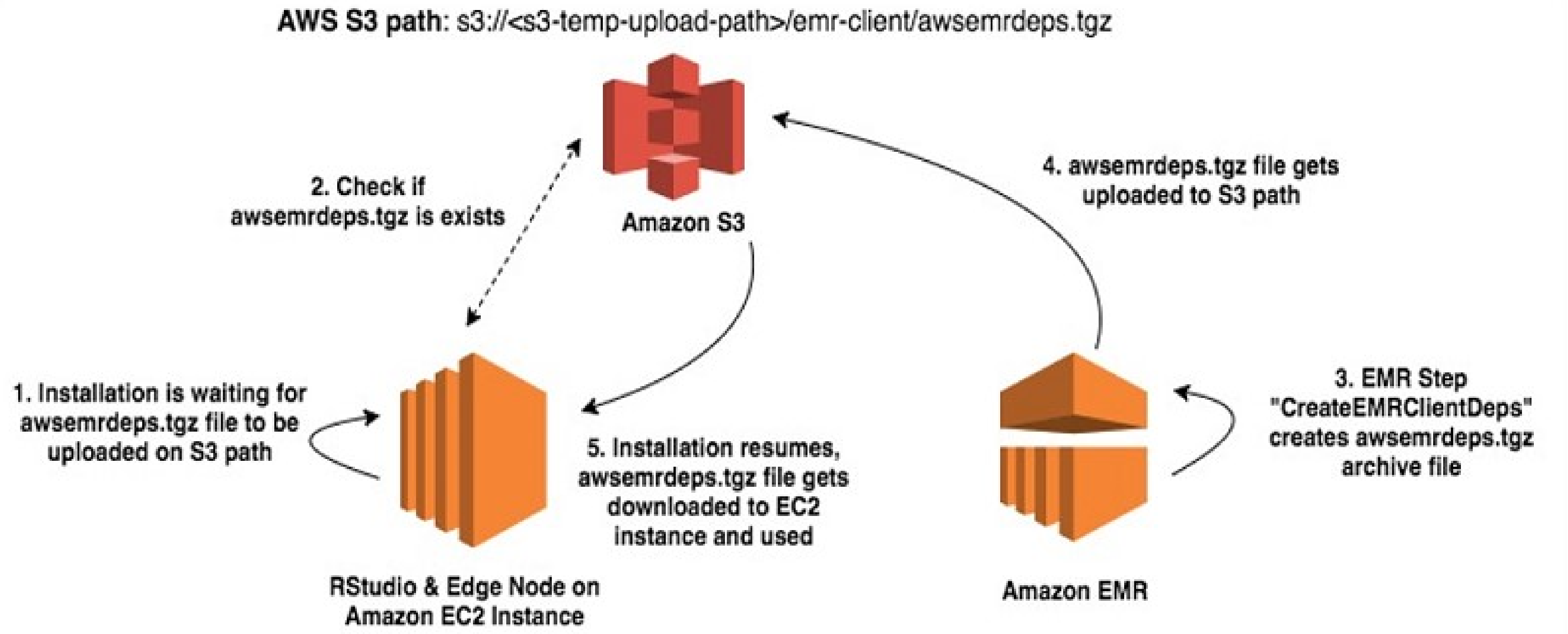
  If you created a bucket with different name, delete that one.

Summary

Now you…

- … can explain assignment of Amazon EMR and describe the Ecosystem of Amazon EMR
- … understand clusters and nodes
- … can explain EMR workflow
- … know about AWS Management Console: EC2, S3, Analytics: EMR
- … can running task in EMR cluster using Jupyter Notebook

Thank you for your attention!


AWS S3 path: s3://<s3-temp-upload-path>/emr-client/awsemrdeps.tgz
Amazon S3
2. Check if awsemrdeps.tgz is exists
4. awsemrdeps.tgz file gets uploaded to S3 path
1. Installation is waiting for awsemrdeps.tgz file to be uploaded on S3 path
5. Installation resumes, awsemrdeps.tgz file gets downloaded to EC2 instance and used
3. EMR Step "CreateEMRClientDeps" creates awsemrdeps.tgz archive file
RStudio & Edge Node on Amazon EC2 Instance
Amazon EMR

Books and References

Books

- K. Schmidt, Ch. Phillips, “Programming Elastic MapReduce: Using AWS Services to Build an End-to-End Application”, O'Reilly Media, 2014

Other References:

- Overview of Amazon EMR https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview.html
- E-books https://aws.amazon.com/events/aws-innovate/modern-apps/resources/

Online course

- Big Data on AWS https://aws.amazon.com/training/classroom/big-data-on-aws
- Big Data on Amazon web services (AWS) https://www.udemy.com/course/big-data-on-amazon-web-services-aws-cloud-2018