CLOUDYML

40+ DATA ENGINEERING INTERVIEW QNAs

Akash Raj
Data Scientist

1. Can you explain the differences between Hadoop and MapReduce?

Hadoop is a distributed computing framework that provides storage and processing capabilities for large datasets. MapReduce is a programming model used to process large datasets in parallel across a Hadoop cluster. Hadoop includes other components, such as HDFS for distributed storage and YARN for resource management, while MapReduce is a part of the Hadoop ecosystem used to process data.

2. How do you optimize a MapReduce job to improve its performance?

There are several ways to optimize a MapReduce job to improve performance, such as tuning the configuration parameters, optimizing the data input and output formats, and reducing data shuffling. Other optimization techniques include using combiners, compressing intermediate data, and partitioning the input data

Akash Raj
Data Scientist

3. Can you walk me through the process of setting up a Hadoop cluster and configuring it for optimal performance?

Setting up a Hadoop cluster involves configuring several components, such as HDFS, YARN, and MapReduce. The process typically involves selecting appropriate hardware and software, setting up the network, configuring security, and tuning the Hadoop configuration parameters. To optimize performance, you may need to adjust settings such as block size, replication factor, and memory allocation.

4. How do you handle data skew in a MapReduce job?

Data skew can occur when some keys have a disproportionately large amount of data compared to others. One approach to handle data skew is to use a partitioner to distribute data across reducers more evenly. Another approach is to use custom partitioning algorithms that group similar keys together.

Akash Raj
Data Scientist

5.Can you explain how HDFS handles data replication and fault tolerance ?

HDFS stores data in multiple replicas across different nodes in the cluster to ensure fault tolerance. The replication factor determines the number of replicas for each block of data, and HDFS automatically replicates data when a node fails. HDFS also uses a NameNode to manage metadata and provide access control to files.

6. Can you explain the differences between a block and a split in Hadoop?

In Hadoop, a block is the unit of data storage in HDFS, while a split is the unit of data processing in MapReduce. Blocks are typically 128 MB by default and are replicated across nodes in the cluster for fault tolerance. Splits are created by the InputFormat to divide input data into smaller chunks that can be processed in parallel by multiple Map tasks.

7.How do you ensure data consistency and integrity in Hadoop?

Hadoop provides several mechanisms to ensure data consistency and integrity, such as checksums to detect data corruption, replication to ensure data availability, and write-once-read-many (WORM) support to prevent accidental data modification. Additionally, Hadoop provides access controls and auditing features to ensure data security and compliance.

8. Can you explain the concept of speculative execution in MapReduce and how it works?

Speculative execution is a feature in MapReduce that allows redundant tasks to be launched for slow-running tasks on different nodes to complete the job more quickly. When a task is taking longer than expected, Hadoop launches a duplicate task on a different node to complete the same task. The first task to complete the job sends its output to the reducer, while the other task is terminated.

9. Explain The Five Vs of Big Data.

The five Vs of Big Data are –
Volume – Amount of data in the Petabytes and Exabytes
Variety – Includes formats like an videos, audio sources, textual data, etc.
Velocity – Everyday data growth which are includes conversations in forums,blogs,social media posts,etc.
Veracity – Degree of accuracy of data are available
Value – Deriving insights from collected data to the achieve business milestones and new heights

10. What is Hive, how does it work ?

Hive is an open-source data warehousing and SQL-like query language that allows users to perform SQL-like queries on large datasets stored in Hadoop's HDFS (Hadoop Distributed File System). Hive uses a SQL-like language called HiveQL or HQL, which translates queries into MapReduce jobs that can be executed on a Hadoop cluster. Hive was created to make it easier for users familiar with SQL to work with Hadoop's distributed computing framework.

11. What is Hive metadata and how is it used by Hive to manage tables and partitions?

Hive metadata is information about the structure of data stored in Hive tables, such as table schema, partitioning schemes, table locations, and file formats. Hive metadata is managed by the Hive Metastore, which is a service that stores metadata information in a relational database. The metastore serves as a central repository for metadata that can be accessed by different components of Hive, such as the Hive Query Processor or the Hive Execution Engine. Hive uses metadata to manage tables and partitions in a distributed environment.

For example, when a user creates a new table in Hive, the Hive Metastore creates an entry for the table in the metastore database, including information such as the table name, column names, column types, and file format. When a user queries the table, Hive uses the metadata to retrieve the location of the table data, the file format, and the schema of the table, which it then uses to execute the query.

Akash Raj
Data Scientist

12. What is a data warehouse and what are some benefits of using one?

A data warehouse is a centralized repository of data that is used for reporting and data analysis. It is designed to support business intelligence (BI) activities by consolidating data from multiple sources, cleaning and transforming it, and storing it in a way that is optimized for querying and reporting. The goal of a data warehouse is to provide a single, consistent view of data across an organization, so that
decision-makers can gain insights and make informed decisions.

There are several benefits of using a data warehouse, including:

1. Improved data quality: Data in a data warehouse is cleaned and transformed before being loaded, which helps to improve data quality and consistency.

2. Faster query performance: Data is organized and optimized for querying and reporting, which can lead to faster query performance.

3. Scalability: Data warehouses can handle large volumes of data and can be scaled to support growing data volumes and user loads.

4. Better decision-making: Data warehouses provide a single, consistent view of data, which helps decision-makers to make more informed decisions.

Hive is a data warehousing solution that is built on top of Hadoop. It provides a SQL-like language called HiveQL for querying and managing large datasets in Hadoop. Hive is designed to work with big data and provides features such as partitioning, bucketing, and indexing to improve query performance.

13. Can you explain how Hive can be used as a data warehousing solution?

Hive can be used as a data warehousing solution by creating tables in Hive that map to data stored in Hadoop. Data can be loaded into these tables using HiveQL or other tools, and then queried using SQL-like syntax. Hive provides features such as views, subqueries, and window functions to support complex queries and analysis.

Overall, Hive provides a cost-effective and scalable way to build a data warehouse on top of Hadoop, and can be a good choice for organizations that need to store and analyze large volumes of data.

14. Difference between Hive & RDBMS ?

Hive and RDBMS (Relational Database Management System) differ in several ways. RDBMS systems are designed for online transaction processing (OLTP), while Hive is designed for online analytical processing (OLAP). RDBMS systems typically use a schema-on-write approach, where data is structured and defined when it is written to the database. Hive, on the other hand, uses a schema-on-read approach, where the data schema is defined when it is read from the storage system.

Additionally, RDBMS systems are optimized for low-latency, high-throughput operations, while Hive is optimized for large-scale batch processing. Hive is designed to handle very large datasets that may not fit into memory, while RDBMS systems are designed to handle smaller datasets that can fit into memory.

15. Can you explain what a Hive table is and how it is different from a table in a traditional database?

A Hive table is a logical representation of structured data in Hive, which is a data warehousing tool built on top of Hadoop. In Hive, tables are created using a SQL-like language called HiveQL, and the data stored in these tables is stored in Hadoop Distributed File System (HDFS) or other supported storage systems. Hive tables are designed to work with large amounts of data and provide a schema on read capability, allowing users to query the data without knowing its structure beforehand.

The key difference between a Hive table and a table in a traditional database is that Hive tables are built on top of Hadoop and are designed to work with big data. Unlike traditional databases, Hive tables do not enforce constraints on data and do not provide transactions or ACID compliance, which are important features for applications that require high consistency and reliability. Hive tables are also stored in a distributed file system, which means that they can be easily scaled out to handle large datasets.

Another key difference between a Hive table and a traditional database table is that Hive tables are generally used for batch processing of data, rather than real-time processing. This means that Hive tables are typically used for data warehousing and analytics applications, where the focus is on querying large datasets to extract insights and generate reports.
In summary, a Hive table is a logical representation of structured data in Hive, designed to work with big data and provide schema-on-read capabilities. While it is similar to a traditional database table in some ways, it is designed to work with different data structures, at different scales, and for different types of applications.

16. Why does hive doesn’t store metadata in HDFS?

Storing metadata in HDFS can result in high latency/delay due to the sequential access in HDFS for read/write operations. As a result, Hive stores metadata in a separate Metastore, which is usually implemented using a relational database such as MySQL. Storing metadata in a Metastore allows for random access to the metadata, which helps achieve low latency and faster query processing times.

17. How much time it is taking to process the data in hive and spark?

The amount of time it takes to process data in Hive or Spark depends on several factors, including the amount and complexity of the data, the size of the cluster, the hardware configuration, the efficiency of the query or application code, and the resources allocated to the job.

In general, Spark is considered faster than Hive for processing large datasets because it leverages in-memory computation and supports parallel processing across multiple nodes in a cluster. This allows Spark to perform complex computations faster and with less I/O overhead compared to Hive, which relies on MapReduce for processing.

18. What is OFF_HEAP persistence in spark?

One of the most important capabilities in Spark is persisting (or caching) datasets in memory across operations. Each persisted RDD can be stored using a different storage level. One of the possibilities is to store RDDs in serialized format off-heap. Compared to storing data in the Spark JVM, off-heap storage reduces garbage collection overhead and allows executors to be smaller and to share a pool of memory. This makes it attractive in environments with large heaps or multiple concurrent applications.

19. How much time it is taking to process the data in hive and spark?

The amount of time it takes to process data in Hive or Spark depends on several factors, including the amount and complexity of the data, the size of the cluster, the hardware configuration, the efficiency of the query or application code, and the resources allocated to the job.

In general, Spark is considered faster than Hive for processing large datasets because it leverages in-memory computation and supports parallel processing across multiple nodes in a cluster. This allows Spark to perform complex computations faster and with less I/O overhead compared to Hive, which relies on MapReduce for processing.

20. What is an “Accumulator” ?

“Accumulators” are Spark’s offline debuggers. Similar to “Hadoop Counters”, “Accumulators” provide the number of “events” in a program. Accumulators are the variables that can be added through associative operations. Spark natively supports accumulators of numeric value types and standard mutable collections. “AggregrateByKey()” and “combineByKey()” uses accumulators.

21. What is "Lineage Graph" in Spark?

Lineage graph in Spark is a directed acyclic graph (DAG) that represents the entire history of transformations applied to a Resilient Distributed Dataset (RDD). It is a fundamental concept in Spark's fault-tolerance model and enables RDDs to recover lost data partitions in case of node failures. The lineage graph consists of a series of transformations that describe how an RDD was derived from its parent RDDs. Spark uses the lineage graph to recompute lost data partitions by replaying the transformations from the last checkpointed RDD to the lost RDD partition. The lineage graph is a powerful tool for ensuring the reliability and fault-tolerance of distributed computations in Spark.

22. Why is BlinkDB used?

BlinkDB is a query engine for executing interactive SQL queries on huge volumes of data and renders query results marked with meaningful error bars. BlinkDB helps users balance ‘query accuracy’ with response time.

23. What is Pair RDD?

A Pair RDD in Apache Spark is an RDD that consists of key-value pairs, where each key is associated with a corresponding value. Pair RDDs provide a powerful abstraction for working with data that has a key-value structure, such as log files, sensor data, or network traffic. They support a variety of operations that allow data to be grouped, aggregated, and joined based on the keys. Pair RDDs also provide support for advanced operations such as map-reduce-style algorithms, graph processing, and machine learning tasks. Overall, Pair RDDs are a fundamental concept in Spark's programming model and are widely used in many real-world data processing applications.

24. How to start & stop namenode services ?

To start the NameNode service:

1. Ensure that the Hadoop cluster is properly configured and all necessary services are running, including the ResourceManager and DataNodes.
2. Open a terminal or command prompt on the machine where the NameNode service is installed.
3. Navigate to the bin directory of the Hadoop installation.
4. Run the following command to start the NameNode service:
    a. *bash - ./hadoop-daemon.sh start namenode*
5. Verify that the NameNode service is running by checking the log files or by using the 'jps' command to list all Java processes.

To stop the NameNode service:

1. Open a terminal or command prompt on the machine where the NameNode service is running.
2. Navigate to the bin directory of the Hadoop installation.
3. Run the following command to stop the NameNode service:
    a. *arduino - ./hadoop-daemon.sh stop namenode*
4. Verify that the NameNode service has stopped by checking the log files or by using the 'jps' command to list all Java processes

Akash Raj
Data Scientist

25. Explain about the different types of join in Hive?

HiveQL has 4 different types of joins -
JOIN- Similar to Outer Join in SQL
FULL OUTER JOIN - Combines the records of both the left and right outer tables that fulfil the join condition.
LEFT OUTER JOIN- All the rows from the left table are returned even if there are no matches in the right table.
RIGHT OUTER JOIN-All the rows from the right table are returned even if there are no matches in the left table

26. How can I convert a date in string format ("April 22, 2021") to timestamp in hive?

Use the from_unixtime() function in conjunction with the unix_timestamp() function to convert a string date to a Unix timestamp and then format the resulting timestamp into a desired format.

select from_unixtime(unix_timestamp(`date`,'MMM dd, yyyy'),'yyyy-MM-dd')

27. Difference Between RDD, Dataframe, Dataset?

Resilient Distributed Dataset (RDD)

RDD was the primary user-facing API in Spark since its inception. At the core, an RDD is an immutable distributed collection of elements of your data, partitioned across nodes in your cluster that can be operated in parallel with a low-level API that offers transformations and actions.

DataFrames (DF)

Like an RDD, a DataFrame is an immutable distributed collection of data. Unlike an RDD, data is organized into named columns, like a table in a relational database. Designed to make large data sets processing even easier, DataFrame allows developers to impose a structure onto a distributed collection of data, allowing higher-level abstraction; it provides a domain specific language API to manipulate your distributed data.

Datasets (DS)

Starting in Spark 2.0, Dataset takes on two distinct APIs characteristics: a strongly-typed API and an untyped API, as shown in the table below. Conceptually, consider DataFrame as an alias for a collection of generic objects Dataset[Row], where a Row is a generic untyped JVM object.

28. What is the difference between map and flatMap?

In Apache Spark, both 'map()' and 'flatMap()' are transformation operations that are applied on an RDD (Resilient Distributed Dataset). The main difference between them is in their output.

The 'map()' transformation applies a given function to each element of the RDD and returns a new RDD consisting of the results. The resulting RDD has the same number of elements as the input RDD.

The 'flatMap()' transformation applies a given function to each element of the RDD and returns a new RDD consisting of the flattened results. This means that the resulting RDD may have a different number of elements than the input RDD, depending on how the function is applied. The output of the function is expected to be a sequence of values that are concatenated together. For Example:

```
val sentences = sc.parallelize(Seq("Hello world", "How are you"))
val words = sentences.flatMap(_.split(" "))
```

In this example, we use the flatMap() transformation to split each sentence into words and return a new RDD of words. The resulting RDD will have 5 elements (one for each word).

29. How can you trigger automatic clean-ups in Spark to handle accumulated metadata?

You can trigger the clean-ups by setting the parameter "spark.cleaner.ttl" or by dividing the long running jobs into different batches and writing the intermediary results to the disk.

30. What do you understand by Lazy Evaluation?

Spark's approach to processing data is intelligent and efficient. When you instruct Spark to perform operations on a particular dataset, it takes note of the instructions but does not execute them immediately. Instead, Spark waits for a final result to be requested through an action. This means that when a transformation like map() is applied to an RDD, it does not actually execute the operation at that moment, but rather waits until an action is requested to trigger the evaluation of the transformations. By deferring the execution of transformations until an action is called, Spark can optimize the processing workflow and improve overall efficiency.

31. Explain the difference between reduceByKey, groupByKey, aggregateByKey and combineByKey?

In Apache Spark, reduceByKey, groupByKey, aggregateByKey, and combineByKey are all transformations that are used to perform operations on key-value pair RDDs. However, there are some differences in how these transformations work.

- reduceByKey: This transformation groups the values of each key together and applies a reduce function to them to produce a single output value for each key. The reduce function should be associative and commutative, as it may be applied multiple times to combine values across multiple partitions.
- groupByKey: This transformation groups the values of each key together and returns a new RDD where each key is associated with a list of its values. This can be expensive in terms of network and memory usage if the values associated with a key are large, as all the values need to be shuffled over the network to the node where the key is stored.
- aggregateByKey: This transformation is similar to reduceByKey, but allows the user to specify an initial "zero" value for each key, and a separate function to merge the values of each key. This can be useful when the reduce function is not commutative or associative, or when the initial zero value is not the same as the input values.
- combineByKey: This transformation is similar to aggregateByKey, but also allows the user to specify a function to transform the input values before they are combined. This can be useful when the input values are not in the desired format for the reduce operation.

In summary, reduceByKey and aggregateByKey are more efficient than groupByKey because they do not shuffle all the values for a key over the network. combineByKey is a more general transformation that can be used when the input values need to be transformed before the reduce operation.

32. How to find out the different values in between two spark dataframes.?

In Apache Spark, you can use the except transformation to find the different values between two dataframes.

Here's an example:

```
val df1 = Seq((1, "John"), (2, "Alice"), (3, "Bob")).toDF("id", "name")
val df2 = Seq((1, "John"), (2, "Alice"), (4, "Charlie")).toDF("id", "name")

val diff = df1.except(df2)
```

In this example, we have two dataframes df1 and df2, which have some common rows and some different rows. We can use the except transformation to find the rows in df1 that are not present in df2. The resulting dataframe diff will have the row (3, "Bob").

Note that the except transformation returns the rows that are in the first dataframe but not in the second dataframe. If you want to find the rows that are in both dataframes, you can use the intersect transformation instead.

Akash Raj
Data Scientist

33. What are security options in Apache Spark?

Spark currently supports authentication via a shared secret. Authentication can be configured to be on via the spark.authenticate configuration parameter. This parameter controls whether the Spark communication protocols do authentication using the shared secret. This authentication is a basic handshake to make sure both sides have the same shared secret and are allowed to communicate. If the shared secret is not identical they will not be allowed to communicate. The shared secret is created as follows:

For Spark on YARN deployments, configuring spark.authenticate to true will automatically handle generating and distributing the shared secret. Each application will use a unique shared secret. For other types of Spark deployments, the Spark parameter spark.authenticate.secret should be configured on each of the nodes. This secret will be used by all the Master/Workers and applications.

Akash Raj
Data Scientist

34. What is the Difference Between Public Subnet and Private Subnet ?

Public Subnet will have Internet Gateway Attached to its associated Route Table and Subnet, Private Subnet will not have the Internet Gateway Attached to its associated Route Table and Subnet.

Public Subnet will have internet access and Private subnet will not have the internet access directly.

35. What are the Difference Between Route53 and ELB?

Amazon Route 53 will handle DNS servers. Route 53 give you web interface through which the DNS can be managed using Route 53, it is possible to direct and failover traffic. This can be achieved by using DNS Routing Policy.

One more routing policy is Failover Routing policy. we set up a health check to monitor your application endpoints. If one of the endpoints is not available, Route 53 will automatically forward the traffic to other endpoint.

Elastic Load Balancing
ELB automatically scales depends on the demand, so sizing of the load balancers to handle more traffic effectively when it is not required

36. What is the difference between Kafka and flume?

Kafka and Flume are both distributed streaming platforms used for collecting, aggregating, and moving large volumes of data in real-time, but they have some differences.

Kafka is a distributed streaming platform that is designed to be highly scalable, fault-tolerant, and fast. It is a publish-subscribe messaging system that allows producers to publish messages to topics, which are then consumed by consumers. Kafka is known for its ability to handle high volumes of data, support for low-latency processing, and its strong durability guarantees.

Flume, on the other hand, is a distributed data collection system that is used for streaming log data from various sources to a central repository. Flume is designed to be flexible and extensible, and it can collect data from a wide variety of sources, including log files, network traffic, and social media feeds. Flume is known for its simplicity and ease of use, and it is often used for collecting data from legacy systems or applications that don't have native support for Kafka.

In summary, Kafka is a general-purpose distributed streaming platform that is designed to handle high volumes of data and support low-latency processing, while Flume is a specialized distributed data collection system that is designed to be flexible and easy to use.

37. What are the difference between Scala and Java.

Scala and Java are both popular programming languages that run on the Java Virtual Machine (JVM), but they have some differences.

1. Syntax: One of the most obvious differences between Scala and Java is their syntax. Scala has a more concise and expressive syntax than Java, which can result in fewer lines of code and more readable code.
2. Object-oriented vs functional: While both Scala and Java are object-oriented programming languages, Scala has a stronger focus on functional programming. Scala has first-class support for functional programming concepts such as immutability, higher-order functions, and pattern matching, which can make it easier to write code that is more concise and maintainable.
3. Type inference: Scala has a powerful type inference system that can automatically infer the types of variables and expressions, which can make the code shorter and easier to read. Java, on the other hand, requires explicit type declarations for all variables and expressions.
4. Interoperability: Scala is fully interoperable with Java, which means that Scala code can use Java libraries and frameworks and vice versa. This can make it easier to integrate Scala code with existing Java codebases.
5. Performance: Scala can sometimes be faster than Java, particularly when it comes to processing large amounts of data. This is because Scala has better support for concurrency and parallelism, which can take advantage of modern multi-core processors.

In summary, Scala has a more concise and expressive syntax, a stronger focus on functional programming, a powerful type inference system, and better support for concurrency and parallelism than Java. However, Java has a larger ecosystem of libraries and frameworks and is more widely used in enterprise applications.

38. Explain how kerberos authentication happens?

Kerberos is a network authentication protocol that is used to authenticate clients and servers in a distributed environment. Here's how Kerberos authentication happens:

1. Authentication request: A client sends an authentication request to the Kerberos authentication server (AS).
2. Ticket granting ticket (TGT): The AS responds with a ticket granting ticket (TGT), which is encrypted with a secret key that is shared between the client and the AS.
3. Authentication ticket: The client sends the TGT to the ticket granting server (TGS), along with a request for an authentication ticket for a specific service.
4. Service ticket: The TGS responds with an authentication ticket, which is encrypted with a secret key that is shared between the TGS and the service.
5. Service authentication: The client sends the authentication ticket to the service, along with an authenticator that proves the client's identity. The service decrypts the authentication ticket using the secret key shared with the TGS, and verifies the authenticator.
6. Service access: If the authentication is successful, the service grants access to the client.

In summary, Kerberos authentication involves a series of steps that involve the client, the authentication server (AS), the ticket granting server (TGS), and the service. The client requests a TGT from the AS, which is then used to request an authentication ticket from the TGS, which is then used to authenticate the client to the service.

39. What is the difference between Teradata and Oracle

Teradata and Oracle are both popular relational database management systems (RDBMS), but they have some differences:

1. Architecture: Teradata is designed as a massively parallel processing (MPP) system, which allows it to scale out horizontally to handle large amounts of data. Oracle is a more traditional RDBMS that is designed to run on a single server, although it does have some features for distributed computing.
2. SQL support: Both Teradata and Oracle support SQL, but they have some differences in their implementation. Teradata has some unique SQL features, such as support for the OLAP functions, which can make it easier to analyze data. Oracle has some features that are not available in Teradata, such as support for the SQL Model clause.
3. Cost: Teradata is generally considered to be more expensive than Oracle, both in terms of hardware and software costs. However, Teradata is often used for very large data warehouses and analytics applications, where the cost of the system is justified by the performance gains.
4. Market share: Oracle is the more widely used RDBMS, with a larger market share than Teradata. Oracle is used in a wide range of applications, from small business databases to large-scale enterprise applications.

In summary, Teradata is designed for high-performance analytics and data warehousing, and is optimized for large-scale distributed computing. Oracle is a more traditional RDBMS that is widely used in a range of applications, from small databases to large enterprise systems.

40. How to do paswordless SSH in hadoop ?

Here are the steps to set up passwordless SSH in Hadoop:

1. Generate SSH keys: On the client machine, generate an SSH key pair using the ssh-keygen command. This will generate a public key and a private key.
2. Copy the public key to the authorized_keys file: Copy the contents of the public key (usually located in the ~/.ssh/id_rsa.pub file) to the authorized_keys file on the server machine. This file is typically located in the ~/.ssh directory.
3. Test the SSH connection: Try to SSH into the server from the client machine without providing a password. If the SSH connection is successful, then passwordless SSH has been set up.
4. Configure Hadoop to use passwordless SSH: On the client machine, edit the core-site.xml file in the Hadoop configuration directory (usually located in /etc/hadoop/conf/). Set the fs.ssh.hostbased.auth and fs.ssh.authorized.keyfile properties to enable passwordless SSH.
5. Test Hadoop SSH connection: Test the Hadoop SSH connection by running a Hadoop command, such as hdfs dfs -ls /.

If everything is set up correctly, Hadoop should be able to connect to the server machine without requiring a password. Note that it's important to ensure that the correct permissions are set on the authorized_keys file, as allowing unauthorized access to this file could compromise the security of the system.

Akash Raj
Data Scientist

41. What is the maximum file length in S3?

- **u**tf-8 1024 bytes

41. Is EFS a centralised storage service in AWS?

Yes, Amazon Elastic File System (EFS) is a centralized storage service in AWS. EFS provides a scalable and fully-managed file storage service for use with Amazon EC2 instances and on-premises servers. It is designed to provide a shared, elastic file system that multiple instances can use at the same time, allowing you to share files across multiple instances or applications.

EFS is a regional service, meaning that it is deployed within a specific AWS region and is accessible from all availability zones within that region. This makes it a centralized storage service, as all instances within the region can access the same file system and share files.

By contrast, Amazon Elastic Block Store (EBS) provides block-level storage volumes that can only be attached to a single EC2 instance at a time, making it a localized storage service.

42.Which AWS service will you use to collect and process ecommerce data for near real time analysis?

Both Dynamo DB & Redshift

43.Maximum number of bucket which can be crated in AWS.

By default, AWS accounts can create up to 100 buckets per account. However, this limit can be increased by contacting AWS support and requesting a limit increase.

CLOUDYML

WWW.CLOUDYML.COM

CRACK THE CODE TO DATA SCIENCE SUCCESS WITH AKASH RAJ

FOLLOW FOR MORE