## **1. Big Data Processing with Apache Spark**

### **Core Spark Concepts**

#### **Theory**

* Apache Spark APIs overview
* Spark initialization and setup
* Spark session and context
* Resilient Distributed Datasets (RDDs)
  + External datasets integration
  + RDD operations and transformations
  + Key-Value pair operations
  + Shuffle operations
* RDD persistence and data management
* Shared variables
* Cluster deployment strategies
* Spark DataFrames

#### **Practical Components**

* Hadoop Streaming implementation with Python
* Spark installation and configuration
* Basic RDD operations implementation

### **Data Processing and Analysis**

#### **Theory**

* Exploratory Data Analysis (EDA) with PySpark
* ETL (Extract, Transform, Load) operations
* Spark SQL fundamentals
* Database integration capabilities

#### **Practical Components**

* PySpark data exploration exercises
* ETL pipeline implementation
* Spark SQL query execution and analysis
* Election data analysis project
* Database connectivity and manipulation

### **Advanced Analytics**

#### **Theory**

* Machine Learning with Spark's MLlib
* Deep Learning implementation in Spark
* Advanced analytical workflows

#### **Practical Components**

* MLlib exploration and implementation
* Deep Learning models in Spark
* Workflow creation using bash and Python operators

## **2. Stream Processing with Kafka**

### **Kafka Fundamentals**

#### **Theory**

* Apache Kafka architecture
* Producer-Consumer model
* Kafka Connect API
* Stream processing concepts

#### **Practical Components**

* Kafka installation and setup
* Producer-Consumer implementation
* API integration exercises

### **Kafka-Spark Integration**

#### **Theory**

* Spark Streaming with Kafka
* Real-time data processing
* Integration patterns and best practices

#### **Practical Components**

* Spark-Kafka integration setup
* Stream processing implementation
* Real-time data analysis

## **3. DevOps for AI/ML**

### **Version Control**

#### **Theory**

* Version control systems fundamentals
* Git concepts and workflow
* GitHub repository management

#### **Practical Components**

* Git command line operations
* GitHub repository creation and management
* Collaborative development workflows

### **Containerization**

#### **Theory**

* Container technology fundamentals
* Docker architecture and concepts
* Container networking
* Resource management
* Logging and monitoring

#### **Practical Components**

* Docker installation and basic operations
* Custom image creation
* Container deployment and management
* Web server container implementation

### **Kubernetes Orchestration**

#### **Theory**

* Kubernetes architecture
* Cluster components:
  + Management node
  + Worker nodes
  + Pods
  + Deployments
  + Services
* Application lifecycle management
* Rolling updates and scaling

#### **Practical Components**

* Kubernetes cluster setup
* Application deployment
* Service configuration
* Scaling and update management

### **CI/CD Pipeline**

#### **Theory**

* Continuous Integration concepts
* Continuous Deployment practices
* Jenkins architecture and workflow

#### **Practical Components**

* Jenkins pipeline setup
* Integration with Docker and GitHub
* Automated deployment workflows
* Web application CI/CD implementation

## **4. Cloud Computing**

### **Cloud Fundamentals**

#### **Theory**

* Cloud computing concepts
* Service provider comparison (AWS/Azure/GCP)
* Cloud architecture principles
* Service models:
  + Software as a Service (SaaS)
  + Platform as a Service (PaaS)
  + Infrastructure as a Service (IaaS)

#### **Practical Components**

* Cloud platform exploration
* Architecture analysis
* Service model implementation

### **Cloud Operations**

#### **Theory**

* Cloud service administration
* Monitoring and management
* Resource optimization
* Pricing models
* Elastic computing concepts

#### **Practical Components**

* Cloud service deployment
* Resource management exercises
* Cost optimization practices

### **AI/ML in Cloud**

#### **Theory**

* Cloud services for AI/ML workloads
* ML model deployment
* Scalable AI infrastructure

#### **Practical Components**

* AI/ML cloud platform exploration
* Model deployment exercises
* Performance optimization

**PySpark**

**1. An API for using Spark in \_\_\_\_ is PySpark.**

1. Java
2. C
3. C++
4. Python

**Answer:** D) Python

**Explanation:**

An API for using Spark in Python is PySpark.

**2. Using Spark, users can implement big data solutions in an \_\_\_\_-source, cluster computing environment.**

1. Closed
2. Open
3. Hybrid
4. None

**Answer:** B) Open

**Explanation:**

Using Spark, users can implement big data solutions in an open-source, cluster computing environment.

**3. In PySpark, \_\_\_\_ library is provided, which makes integrating Python with Apache Spark easy.**

1. Py5j
2. Py4j
3. Py3j
4. Py2j

**Answer:** B) Py4j

**Explanation:**

In PySpark, Py4j library is provided, which makes integrating Python with Apache Spark easy.

**4. Which of the following is/are the feature(s) of PySpark?**

1. Lazy Evaluation
2. Fault Tolerant
3. Persistence
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The following are the features of PySpark -

1. Lazy Evaluation
2. Fault Tolerant
3. Persistence

**5. In-memory processing of large data makes PySpark ideal for \_\_\_\_ computation.**

1. Virtual
2. Real-time
3. Static
4. Dynamic

**Answer:** B) Real-time

**Explanation:**

In-memory processing of large data makes PySpark ideal for real-time computation.

**6. A variety of programming languages can be used with the PySpark framework, such as \_\_\_\_, and R.**

1. Scala
2. Java
3. Python
4. All of the above

**Answer:** D) All of the above

**Explanation:**

A variety of programming languages can be used with PySpark framework, such as Scala, Java, Python, and R.

**7. In memory, PySpark processes data 100 times faster, and on disk, the speed is \_\_ times faster.**

1. 10
2. 100
3. 1000
4. 10000

**Answer:** B) 100

**Explanation:**

In memory, PySpark processes data 100 times faster, and on disk, the speed is 10 times faster.

**8. When working with \_\_\_\_, Python's dynamic typing comes in handy.**

1. RDD
2. RCD
3. RBD
4. RAD

**Answer:** A) RDD

**Explanation:**

When working with RDD, Python's dynamic typing comes in handy.

**9. The Apache Software Foundation introduced Apache Spark, an open-source \_\_\_\_ framework.**

1. Clustering Calculative
2. Clustering Computing
3. Clustering Concise
4. Clustering Collective

**Answer:** B) Clustering Computing

**Explanation:**

The Apache Software Foundation introduced Apache Spark, an open-source clustering computing framework.

**10. \_\_\_\_ are among the key features of Apache Spark. It is easy to use, provides simplicity, and can run virtually anywhere.**

1. Stream Analysis
2. High Speed
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

Stream analysis and high speed are among the key features of Apache Spark. It is easy to use, provides simplicity, and can run virtually anywhere.

**11. The Apache Spark framework can perform a variety of tasks, such as \_\_\_\_, running Machine Learning algorithms, or working with graphs or streams.**

1. Executing distributed SQL
2. Creating data pipelines
3. Inputting data into databases
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The Apache Spark framework can perform a variety of tasks, such as executing distributed SQL, creating data pipelines, inputting data into databases, running Machine Learning algorithms, or working with graphs or streams.

**12. Programming in \_\_\_\_ is the official language of Apache Spark.**

1. Scala
2. PySpark
3. Spark
4. None

**Answer:** A) Scala

**Explanation:**

Programming in Scala is the official language of Apache Spark.

**13. Scala is a \_\_\_\_ typed language as opposed to Python, which is an interpreted, \_\_\_\_ programming language.**

1. Statically, Dynamic
2. Dynamic, Statically
3. Dynamic, Partially Statically
4. Statically, Partially Dynamic

**Answer:** A) Statically, Dynamic

**Explanation:**

Scala is a statically typed language as opposed to Python, which is an interpreted, dynamic programming language.

**14. A \_\_\_\_ program is written in Object-Oriented Programming (OOP).**

1. Python
2. Scala
3. Both A and B
4. None of the above

**Answer:** A) Python

**Explanation:**

A Python program is written in Object-Oriented Programming (OOP).

**15. \_\_\_\_ must be specified in Scala.**

1. Objects
2. Variables
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

Objects and variables must be specified in Scala.

**16. Python is \_\_ times slower than Scala.**

1. 2
2. 5
3. 10
4. 20

**Answer:** C) 10

**Explanation:**

Python is 10 times slower than Scala.

**17. As part of Netflix's real-time processing, \_\_\_\_ is used to make an online movie or web series more personalized for customers based on their interests.**

1. Scala
2. Dynamic
3. Apache Spark
4. None

**Answer:** C) Apache Spark

**Explanation:**

As part of Netflix's real-time processing, Apache Spark is used to make an online movie or web series more personalized for customers based on their interests.

**18. Targeted advertising is used by top e-commerce sites like \_\_\_\_, among others.**

1. Flipkart
2. Amazon
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

Targeted advertising is used by top e-commerce sites like Flipkart and Amazon, among others.

**19. Java version 1.8.0 or higher is required for PySpark, as is \_\_\_\_ version 3.6 or higher.**

1. Scala
2. Python
3. C
4. C++

**Answer:** B) Python

**Explanation:**

Java version 1.8.0 or higher is required for PySpark, as is Python version 3.6 or higher.

**20. Using Spark\_\_\_\_, we can set some parameters and configurations to run a Spark application on a local cluster or dataset.**

1. Cong
2. Conf
3. Con
4. Cont

**Answer:** B) Conf

**Explanation:**

Using SparkConf, we can set some parameters and configurations to run a Spark application on a local cluster or dataset.

**21. Which of the following is/are the feature(s) of the SparkConf?**

1. set (key, value)
2. setMastervalue (value)
3. setAppName (value)
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The following are the features of the SparkConf -

1. set (key, value)
2. setMastervalue (value)
3. setAppName (value)

**22. Spark programs initially create a Spark\_\_\_\_ object to instruct them how to access the cluster.**

1. Contact
2. Context
3. Content
4. Config

**Answer:** B) Context

**Explanation:**

Spark programs initially create a SparkContext object to instruct them how to access the cluster.

**23. Pyspark provides SparkContext by default as \_\_.**

1. sc
2. st
3. sp
4. se

**Answer:** A) sc

**Explanation:**

Pyspark provides SparkContext by default as sc.

**24. Which of the following parameter(s) is/are accepted by SparkContext?**

1. Master
2. appName
3. SparkHome
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The following parameters are accepted by SparkContext -

1. Master
2. appName
3. SparkHome

**25. The Master \_\_\_ identifies the cluster connected to Spark.**

1. URL
2. Site
3. Page
4. Browser

**Answer:** A) URL

**Explanation:**

The Master URL identifies the cluster connected to Spark.

**26. The \_\_\_\_ directory contains the Spark installation files.**

1. SparkHome
2. pyFiles
3. BatchSize
4. Conf

**Answer:** A) SparkHome

**Explanation:**

The SparkHome directory contains the Spark installation files.

**27. The PYTHONPATH is set by sending \_\_\_\_ files to the cluster.**

1. .zip
2. .py
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

The PYTHONPATH is set by sending .zip or .py files to the cluster.

**28. This number corresponds to the BatchSize of the Python \_\_\_\_.**

1. Objects
2. Arrays
3. Stacks
4. Queues

**Answer:** A) Objects

**Explanation:**

This number corresponds to the BatchSize of the Python objects.

**29. The batching can be disabled by setting it to \_\_\_\_.**

1. 0
2. 1
3. Void
4. Null

**Answer:** B) 1

**Explanation:**

The batching can be disabled by setting it to 1

**30. An integrated \_\_\_\_ programming API is provided by PySpark SQL in Spark.**

1. Relational-to-functional
2. Functional-to-functional
3. Functional-to-relational
4. None of the above

**Answer:** A) Relational-to-functional

**Explanation:**

An integrated relational-to-functional programming API is provided by PySpark SQL in Spark.

**31. What is/are the drawback(s) of Hive?**

1. In other words, if the workflow execution fails in the middle, you cannot recover the position from which it stopped.
2. Changing the trash setting will prevent us from dropping encrypted databases in cascade.
3. MapReduce executes ad-hoc queries, which are launched by Hive, but the performance of the analysis is delayed due to the medium-sized database.
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The drawbacks of Hive are -

1. In other words, if the workflow execution fails in the middle, you cannot recover the position from which it stopped.
2. Changing the trash setting will prevent us from dropping encrypted databases in cascade.
3. MapReduce executes ad-hoc queries, which are launched by Hive, but the performance of the analysis is delayed due to the medium-sized database.

**32. What is/are the feature(s) of PySpark SQL?**

1. Consistence Data Access
2. Incorporation with Spark
3. Standard Connectivity
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The features of PySpark SQL are -

1. Consistence Data Access
2. Incorporation with Spark
3. Standard Connectivity

**33. The Consistent Data Access feature allows SQL to access a variety of data sources, such as \_\_\_\_, JSON, and JDBC, from a single place.**

1. Hive
2. Avro
3. Parquet
4. All of the above

**Answer:** D) All of the above

**Explanation:**

The Consistent Data Access feature allows SQL to access a variety of data sources, such as Hive, Avro, Parquet, JSON, and JDBC, from a single place.

**34. For business intelligence tools, the industry standard is \_\_\_\_ connectivity, which are both used for standard connectivity.**

1. JDBC
2. ODBC
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

For business intelligence tools, the industry standard is JDBC or ODBC connectivity, which are both used for standard connectivity.

**35. What is the full form of UDF?**

1. User-Defined Formula
2. User-Defined Functions
3. User-Defined Fidelity
4. User-Defined Fortray

**Answer:** B) User-Defined Functions

**Explanation:**

The full form of UDF is User-Defined Functions.

**36. A UDF extends Spark SQL's DSL vocabulary for transforming DataFrames by defining a new \_\_\_\_-based function.**

1. Row
2. Column
3. Tuple
4. None

**Answer:** B) Column

**Explanation:**

A UDF extends Spark SQL's DSL vocabulary for transforming DataFrames by defining a new column-based function.

**37. Spark SQL and DataFrames include the following class(es):**

1. pyspark.sql.SparkSession
2. pyspark.sql.DataFrame
3. pyspark.sql.Column
4. All of the above

**Answer:** D) All of the above

**Explanation:**

Spark SQL and DataFrames include the following classes:

1. pyspark.sql.SparkSession
2. pyspark.sql.DataFrame
3. pyspark.sql.Column

**38. DataFrame and SQL functionality is accessed through \_\_\_\_.**

1. pyspark.sql.SparkSession
2. pyspark.sql.DataFrame
3. pyspark.sql.Column
4. pyspark.sql.Row

**Answer:** A) pyspark.sql.SparkSession

**Explanation:**

DataFrame and SQL functionality are accessed through pyspark.sql.SparkSession.

**39. \_\_\_\_ represents a set of named columns and distributed data.**

1. pyspark.sql.GroupedData
2. pyspark.sql.DataFrame
3. pyspark.sql.Column
4. pyspark.sql.Row

**Answer:** B) pyspark.sql.DataFrame

**Explanation:**

pyspark.SQL.DataFrame represents a set of named columns and distributed data.

**40. \_\_\_\_ returns aggregation methods.**

1. DataFrame.groupedBy()
2. Data.groupBy()
3. Data.groupedBy()
4. DataFrame.groupBy()

**Answer:** D) DataFrame.groupBy()

**Explanation:**

DataFrame.groupBy() returns aggregation methods.

**41. Missing data can be handled via \_\_\_\_.**

1. pyspark.sql.DataFrameNaFunctions
2. pyspark.sql.Column
3. pyspark.sql.Row
4. pyspark.sql.functions

**Answer:** A) pyspark.sql.DataFrameNaFunctions

**Explanation:**

Missing data can be handled via pyspark.sql.DataFrameNaFunctions.

**42. A list of built-in functions for DataFrame is stored in \_\_\_\_.**

1. pyspark.sql.functions
2. pyspark.sql.types
3. pyspark.sql.Window
4. All of the above

**Answer:** A) pyspark.sql.functions

**Explanation:**

A list of built-in functions for DataFrame is stored in pyspark.sql.functions.

**43. \_\_\_\_ in PySpark UDF are similar to their functions in Pandas.**

1. map()
2. apply()
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

map() and apply() in PySpark UDF are similar to their functions in Pandas.

**44. Which of the following is/are the common UDF problem(s)?**

1. Py4JJavaError
2. Slowness
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

The following are the common UDF problems -

1. Py4JJavaError
2. Slowness

**45. What is the full form of RDD?**

1. Resilient Distributed Dataset
2. Resilient Distributed Database
3. Resilient Defined Dataset
4. Resilient Defined Database

**Answer:** A) Resilient Distributed Dataset

**Explanation:**

The full form of RDD is Resilient Distributed Dataset.

**46. In terms of schema-less data structures, RDDs are one of the most fundamental, as they can handle both \_\_\_\_ information.**

1. Structured
2. Unstructured
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

In terms of schema-less data structures, RDDs are one of the most fundamental, as they can handle both structured and unstructured information.

**47. A \_\_\_\_ memory abstraction, resilient distributed datasets (RDDs), allows programmers to run in-memory computations on clustered systems.**

1. Compressed
2. Distributed
3. Concentrated
4. Configured

**Answer:** B) Distributed

**Explanation:**

A distributed memory abstraction, resilient distributed datasets (RDDs), allows programmers to run in-memory computations on clustered systems.

**48. The main advantage of RDD is that it is fault \_\_\_\_, which means that if there is a failure, it automatically recovers.**

1. Tolerant
2. Intolerant
3. Manageable
4. None

**Answer:** A) Tolerant

**Explanation:**

The main advantage of RDD is that it is fault-tolerant, which means that if there is a failure, it automatically recovers.

**49. The following type(s) of shared variable(s) are supported by Apache Spark -**

1. Broadcast
2. Accumulator
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

The following types of shared variables are supported by Apache Spark -

1. Broadcast
2. Accumulator

**50. Rather than shipping a copy of a variable with each task, broadcast lets the programmer store a \_\_\_\_-only variable locally.**

1. Read
2. Write
3. Add
4. Update

**Answer:** A) Read

**Explanation:**

Rather than shipping a copy of a variable with each task, broadcast lets the programmer store a read-only variable locally.

**51. \_\_\_ operations are carried out on the accumulator variables to combine the information.**

1. Associative
2. Commutative
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

Associative and commutative operations are carried out on the accumulator variables to combine the information.

**52. Using \_\_\_\_, PySpark allows you to upload your files.**

1. sc.updateFile
2. sc.deleteFile
3. sc.addFile
4. sc.newFile

**Answer:** C) sc.addFile

**Explanation:**

Using sc.addFile, PySpark allows you to upload your files.

**53. With \_\_\_\_, we can obtain the working directory path.**

1. SparkFiles.get
2. SparkFiles.fetch
3. SparkFiles.set
4. SparkFiles.go

**Answer:** A) SparkFiles.get

**Explanation:**

With SparkFiles.get, we can obtain the working directory path.

**54. To decide how RDDs are stored, PySpark has different StorageLevels, such as the following:**

1. DISK\_ONLY
2. DISK\_ONLY\_2
3. MEMORY\_AND\_DISK
4. All of the above

**Answer:** D) All of the above

**Explanation:**

To decide how RDDs are stored, PySpark has different StorageLevels, such as the following:

1. DISK\_ONLY
2. DISK\_ONLY\_2
3. MEMORY\_AND\_DISK

**55. Among the method(s) that need to be defined by the custom profiler is/are:.**

1. Profile
2. Stats
3. Add
4. All of the above

**Answer:** D) All of the above

**Explanation:**

Among the methods that need to be defined by the custom profiler are:

1. Profile
2. Stats
3. Add

**56. class pyspark.BasicProfiler(ctx) implements \_\_\_\_ as a default profiler.**

1. cProfile
2. Accumulator
3. Both A and B
4. None of the above

**Answer:** C) Both A and B

**Explanation:**

class pyspark.BasicProfiler(ctx) implements cProfile and Accumulator as a default profiler.

**57. Job and stage progress can be monitored using PySpark's \_\_\_-level APIs.**

1. Low
2. High
3. Average
4. None

**Answer:** A) Low

**Explanation:**

Job and stage progress can be monitored using PySpark's low-level APIs.

**58. The active stage ids are returned by \_\_\_\_ in an array.**

1. getActiveStageIds()
2. getJobIdsForGroup(jobGroup = None)
3. getJobInfo(jobId)
4. All of the above

**Answer:** A) getActiveStageIds()

**Explanation:**

The active stage ids are returned by getActiveStageIds() in an array.

**59. A tuning procedure on Apache Spark is performed using PySpark \_\_\_\_.**

1. SparkFiles
2. StorageLevel
3. Profiler
4. Serialization

**Answer:** D) Serialization

**Explanation:**

A tuning procedure on Apache Spark is performed using PySpark Serialization.

**60. Serializing another function can be done using the \_\_\_\_ function.**

1. map()
2. data()
3. get()
4. set()

**Answer:** A) map()

**Explanation:**

Serializing another function can be done using the map() function.

**1. Which of the following statements is true about PySpark?**

A) PySpark is a Python library used for Big Data processing.  
 B) PySpark is a standalone data processing system.  
 C) PySpark is used for processing data only in small batches.  
 D) PySpark does not support distributed processing.

**Answer: A**

**Explanation:** PySpark is a Python library used for Big Data processing. It is built on top of Apache Spark, which is a distributed computing system. PySpark provides APIs in Python for data processing, machine learning, and graph processing.

**2. Which of the following is a transformation operation in PySpark?**

A) count()  
 B) filter()  
 C) collect()  
 D) reduce()

**Answer: B**

**Explanation:** filter() is a transformation operation in PySpark. It creates a new RDD by selecting elements from an existing RDD based on a condition. Other transformation operations in PySpark include map(), flatMap(), union(), distinct(), and groupByKey().

**3. Which of the following is an action operation in PySpark?**

A) map()  
 B) filter()  
 C) count()  
 D) flatMap()

**Answer: C**

**Explanation:** count() is an action operation in PySpark. It returns the number of elements in an RDD. Other action operations in PySpark include collect(), reduce(), take(), and foreach().

**4. Which of the following is used to create an RDD in PySpark?**

A) DataFrame  
 B) DataSet  
 C) SQLContext  
 D) SparkContext

**Answer: D**

**Explanation:** SparkContext is used to create an RDD in PySpark. It is the entry point to the Spark computing system and provides APIs to create RDDs, accumulates values, and manipulate data. Other Spark components in PySpark include SQLContext, SparkSession, and DataFrameReader.

**5. Which of the following is an advantage of using PySpark?**

A) It is easy to learn and use.  
 B) It supports only batch processing.  
 C) It can only process structured data.  
 D) It is slower than other Big Data processing systems.

**Answer: A**

**Explanation:** One of the advantages of using PySpark is that it is easy to learn and use. PySpark provides a Python API for data processing, which is familiar to Python developers. PySpark also supports real-time processing, unstructured data processing, and machine learning.

**6. Which of the following is a distributed data processing system?**

A) Pandas  
 B) NumPy  
 C) PySpark  
 D) SciPy

**Answer: C**

**Explanation:** PySpark is a distributed data processing system. It is built on top of Apache Spark, which is a distributed computing system that can process large volumes of data in parallel across a cluster of computers.

**7. Which of the following is used to read data from a file in PySpark?**

A) readTextFile()  
 B) writeTextFile()  
 C) readDataFrame()  
 D) writeDataFrame()

**Answer: A**

**Explanation:** readTextFile() is used to read data from a file in PySpark. It reads the contents of a file and creates an RDD with each line of the file as an element. Other file input/output operations in PySpark include read.csv(), read.json(), write.csv(), and write.json().

**8. Which of the following is used to convert an RDD to a DataFrame in PySpark?**

A) toDataFrame()  
 B) createDataFrame()  
 C) RDDtoDF()  
 D) fromRDD()

**Answer: B**

**Explanation:** createDataFrame() is used to convert an RDD to a DataFrame in PySpark. It creates a DataFrame from an RDD with a specified schema. Other DataFrame operations in PySpark include select(), filter(), groupBy(), and join().

**9. Which of the following is used to cache an RDD in memory in PySpark?**

A) persist()  
 B) cache()  
 C) saveAsTextFile()  
 D) collect()

**Answer: A**

**Explanation:** persist() is used to cache an RDD in memory in PySpark. It stores the RDD in memory and/or on disk so that it can be reused efficiently in subsequent operations. Other RDD operations in PySpark include mapPartitions(), sortByKey(), reduceByKey(), and aggregateByKey().

**10. Which of the following is a transformation operation that shuffles data in PySpark?**

A) map()  
 B) filter()  
 C) groupByKey()  
 D) reduce()

**Answer: C**

**Explanation:** groupByKey() is a transformation operation that shuffles data in PySpark. It groups the values of each key in an RDD and creates a new RDD of (key, value) pairs. Other shuffling operations in PySpark include sortByKey(), reduceByKey(), and aggregateByKey().

**11. Which of the following is used to create a PairRDD in PySpark?**

A) map()  
 B) flatMap()  
 C) groupByKey()  
 D) zip()

**Answer: D**

**Explanation:** zip() is used to create a PairRDD in PySpark. It creates a new RDD by aggregating the elements of two RDDs into pairs. The first element of each RDD becomes the key, and the second element becomes the value. Other PairRDD operations in PySpark include reduceByKey(), groupByKey(), and join().

**12. Which of the following is used to broadcast a read-only variable in PySpark?**

A) sc.broadcast()  
 B) spark.broadcast()  
 C) rdd.broadcast()  
 D) broadcast()

**Answer: A**

**Explanation:** sc.broadcast() is used to broadcast a read-only variable in PySpark. It broadcasts the variable to all nodes in a Spark cluster so that it can be accessed efficiently by tasks. Other broadcasting operations in PySpark include accumulators and counters.

**13. Which of the following is a built-in machine learning algorithm in PySpark?**

A) Linear Regression  
 B) K-Means Clustering  
 C) Random Forest  
 D) All of the above

**Answer: D**

**Explanation:** PySpark provides several built-in machine learning algorithms, including Linear Regression, K-Means Clustering, Random Forest, Decision Trees, Gradient Boosting, and Naive Bayes. These algorithms can be used for regression, classification, clustering, and collaborative filtering.

**14. Which of the following is a method to improve the performance of PySpark jobs?**

A) Partitioning  
 B) Caching  
 C) Shuffling  
 D) None of the above

**Answer: A**

**Explanation:** Partitioning is a method to improve the performance of PySpark jobs. It involves dividing an RDD into smaller partitions, which can be processed in parallel across multiple nodes in a Spark cluster. Other methods to improve PySpark performance include caching, data serialization, and memory management.

**15. Which of the following is a type of join operation in PySpark?**

A) Inner Join  
 B) Outer Join  
 C) Left Join  
 D) All of the above

**Answer: D**

**Explanation:** PySpark supports several types of join operations, including Inner Join, Outer Join, Left Join, Right Join, and Full Join. Join operations are used to combine two RDDs based on a common key.

**16. Which of the following is used to write data to a file in PySpark?**

A) readTextFile()  
 B) writeTextFile()  
 C) readDataFrame()  
 D) writeDataFrame()

**Answer: B**

**Explanation:** writeTextFile() is used to write data to a file in PySpark. It writes the contents of an RDD to a file with each element of the RDD on a separate line. Other file input/output operations in PySpark include write.csv(), write.json(),

**17. Which of the following is used to read data from a CSV file in PySpark?**

A) readCSV()  
 B) readTextFile()  
 C) readJSON()  
 D) read.parquet()

**Answer: A**

**Explanation:** readCSV() is used to read data from a CSV file in PySpark. It reads the contents of a CSV file and creates a DataFrame with each row of the file as a separate row in the DataFrame. Other file input/output operations in PySpark include read.json(), read.parquet(), and read.text().

**18. Which of the following is used to aggregate data in PySpark?**

A) reduce()  
 B) aggregate()  
 C) groupByKey()  
 D) collect()

**Answer: B**

**Explanation:** aggregate() is used to aggregate data in PySpark. It applies a function to each partition of an RDD and then combines the results using another function. Other aggregation operations in PySpark include reduce(), fold(), and combineByKey().

**19. Which of the following is used to sort data in PySpark?**

A) sort()  
 B) sortByKey()  
 C) groupByKey()  
 D) reduceByKey()

**Answer: B**

**Explanation:** sortByKey() is used to sort data in PySpark. It sorts an RDD of (key, value) pairs by the key in ascending or descending order. Other sorting operations in PySpark include sort(), sortBy(), and sortByValue().

**20. Which of the following is used to convert an RDD to a DataFrame in PySpark?**

A) toDF()  
 B) toDataFrame()  
 C) asDF()  
 D) asDataFrame()

**Answer: A**

**Explanation:** toDF() is used to convert an RDD to a DataFrame in PySpark. It creates a DataFrame with columns named \_1, \_2, \_3, etc. based on the number of elements in each row of the RDD. Other DataFrame operations in PySpark include select(), filter(), join(), and groupBy().

**21. Which of the following is used to rename a column in a PySpark DataFrame?**

A) withColumn()  
 B) renameColumn()  
 C) rename()  
 D) column()

**Answer: C**

**Explanation:** rename() is used to rename a column in a PySpark DataFrame. It renames the specified column to a new name. Other DataFrame operations in PySpark include withColumn(), withColumnRenamed(), and drop().

**22. Which of the following is used to filter rows in a PySpark DataFrame?**

A) select()  
 B) filter()  
 C) join()  
 D) groupBy()

**Answer: B**

**Explanation:** filter() is used to filter rows in a PySpark DataFrame. It selects the rows that satisfy a specified condition. Other DataFrame operations in PySpark include select(), join(), groupBy(), and orderBy().

**23. Which of the following is used to aggregate data in a PySpark DataFrame?**

A) groupBy()  
 B) join()  
 C) filter()  
 D) orderBy()

**Answer: A**

**Explanation:** groupBy() is used to aggregate data in a PySpark DataFrame. It groups the rows in the DataFrame based on one or more columns and applies an aggregation function to each group. Other DataFrame operations in PySpark include join(), filter(), and orderBy().

**24. Which of the following is used to write data to a Parquet file in PySpark?**

A) write.parquet()  
 B) write.csv()  
 C) write.json()  
 D) write.text()

**Answer: A**

**Explanation:** write.parquet() is used to write data to a Parquet file in PySpark. Parquet is a columnar storage format that is optimized for query performance. Other file input/output operations in PySpark include write.csv(), write.json(), and write.text().

**25. Which of the following is used to cache a PySpark DataFrame in memory?**

A) cache()  
 B) persist()  
 C) checkpoint()  
 D) repartition()

**Answer: B**

**Explanation:** persist() is used to cache a PySpark DataFrame in memory. It caches the DataFrame in memory or on disk so that subsequent actions can be performed more quickly. Other DataFrame operations in PySpark include cache(), unpersist(), checkpoint(), and repartition().

## **Kafka**

## **1. What is Apache Kafka primarily used for?**

a) Image Processing

b) Real-time streaming and processing

c) Databases

d) Machine Learning

### **Answer:**

b) Real-time streaming and processing

### **Explanation:**

[Apache Kafka](https://www.javaguides.net/p/apache-kafka-tutorial.html) is designed for real-time data streaming and processing.

## **2. Which of the following is NOT a core API in Kafka?**

a) Producer API

b) Consumer API

c) Streaming API

d) Learning API

### **Answer:**

d) Learning API

### **Explanation:**

Kafka does not have a "Learning API". The main APIs are Producer, Consumer, and Streams.

## **3. What is a Kafka broker?**

a) An API

b) A Kafka server

c) A topic

d) A data record

### **Answer:**

b) A Kafka server

### **Explanation:**

A broker is a Kafka server that stores data and serves client requests.

## **4. What is the purpose of a Kafka broker?**

a) To produce messages.

b) To consume messages.

c) To store data and serve client requests.

d) To route messages to different networks.

### **Answer:**

c) To store data and serve client requests.

### **Explanation:**

A Kafka broker is a server that stores data and handles client requests (from producers and consumers). Brokers form the backbone of the Kafka cluster.

## **5. Which of the following best describes Kafka's durability?**

a) Data is stored temporarily

b) Data is never saved

c) Data is stored persistently

d) Data is saved only in memory

### **Answer:**

c) Data is stored persistently

### **Explanation:**

Kafka ensures data persistence by storing records on disk and replicating data across multiple brokers.

## **6. What does the Kafka Consumer API allow you to do?**

a) Send data to topics

b) Process data streams

c) Consume data from topics

d) Monitor Kafka topics

### **Answer:**

c) Consume data from topics

### **Explanation:**

The Consumer API allows applications to read (consume) data from Kafka topics.

## **7. What are Kafka partitions used for?**

a) Data backup

b) Load balancing of data

c) Monitoring

d) Data encryption

### **Answer:**

b) Load balancing of data

### **Explanation:**

Partitions allow Kafka to horizontally scale as each partition can be hosted on a different server.

## **8. What ensures data availability in case a Kafka broker fails?**

a) Checkpoints

b) Replicas

c) Backups

d) Snapshots

### **Answer:**

b) Replicas

### **Explanation:**

Kafka topics are replicated across multiple brokers to ensure data availability in case of a broker failure.

## **9. By default, where does a Kafka consumer start reading messages in a topic?**

a) From the beginning

b) From the last message

c) From the latest offset

d) From a random offset

### **Answer:**

c) From the latest offset

### **Explanation:**

By default, a Kafka consumer starts reading messages from the latest offset, which means it doesn't consume old messages unless configured otherwise.

## **10. In Kafka, a producer...**

a) Consumes data streams

b) Sends messages to topics

c) Manages topic replication

d) Monitors topic offsets

### **Answer:**

b) Sends messages to topics

### **Explanation:**

A producer is responsible for sending data records to Kafka topics.

## **11. What is the importance of an offset in Kafka?**

a) It determines the order of messages

b) It encrypts the messages

c) It compresses the message data

d) It replicates the data

### **Answer:**

a) It determines the order of messages

### **Explanation:**

Each message within a partition has a unique offset which indicates its position in the sequence.

## **12. How does Kafka ensure data integrity?**

a) By using data checksums

b) By replicating data once

c) By encrypting all data

d) By avoiding persistent storage

### **Answer:**

a) By using data checksums

### **Explanation:**

Kafka uses checksums to validate the integrity of data.

## **13. Which of the following ensures message order in Kafka?**

a) Broker

b) Consumer

c) Partition

d) Replica

### **Answer:**

c) Partition

### **Explanation:**

Within a Kafka partition, the order of messages is maintained. However, across different partitions, the order isn't guaranteed.

## **14. Which of the following best describes a Kafka Cluster?**

a) A collection of Kafka topics

b) A type of Kafka API

c) A collection of Kafka brokers working together

d) A method to process data in Kafka

### **Answer:**

c) A collection of Kafka brokers working together

### **Explanation:**

A Kafka cluster consists of multiple brokers that work together to manage and maintain data records.

## **15. If a Kafka Broker goes down, what ensures the data is not lost?**

a) Data is backed up in cloud storage

b) Data is replicated across multiple brokers in the cluster

c) Data is saved in external databases

d) Kafka uses failover servers

### **Answer:**

b) Data is replicated across multiple brokers in the cluster

### **Explanation:**

Replication in Kafka ensures that even if a broker (or multiple brokers) fails, data will not be lost.

## **16. Which role does the Kafka Producer primarily play?**

a) Consumes data from the Kafka topic

b) Coordinates the brokers in the cluster

c) Sends data to the Kafka topic

d) Ensures data replication

### **Answer:**

c) Sends data to the Kafka topic

### **Explanation:**

The primary role of a Kafka producer is to publish or send data records to topics.

## **17. What is the function of a Kafka Consumer?**

a) Producing data for topics

b) Managing the Kafka cluster

c) Reading data from a topic

d) Storing data in partitions

### **Answer:**

c) Reading data from a topic

### **Explanation:**

A Kafka consumer subscribes to one or more topics and reads (consumes) the data from them.

## **18. How is a Kafka Topic best described?**

a) A replication factor

b) A Kafka API

c) A queue for storing data records

d) A method of consuming data

### **Answer:**

c) A queue for storing data records

### **Explanation:**

A Kafka topic is a distinct category or feed to which data records are published.

## **19. Why is Kafka Partitions important?**

a) They ensure data encryption

b) They replicate data across clusters

c) They allow for horizontal scalability and parallel processing

d) They coordinate broker activities

### **Answer:**

c) They allow for horizontal scalability and parallel processing

### **Explanation:**

Partitions enable Kafka topics to scale by splitting the data across multiple nodes in the cluster.

## **20. In the context of Kafka, what are Offsets?**

a) Encryption keys

b) Data replication factors

c) Unique IDs for brokers

d) Sequence IDs for messages within a partition

### **Answer:**

d) Sequence IDs for messages within a partition

### **Explanation:**

An offset is a unique identifier for a record within a Kafka partition, indicating its position in the sequence.

## **21. If you have multiple consumers reading from the same topic, what allows them to keep track of messages they have already read?**

a) Partitions

b) Brokers

c) Offsets

d) Producer IDs

### **Answer:**

c) Offsets

### **Explanation:**

Each consumer tracks its offset, signifying up to where it has read, so it knows where to continue from.

## **22. What is a Consumer Group in Kafka?**

a) A group of topics

b) A collection of producers

c) A set of consumers sharing a common group identifier

d) A cluster of brokers

### **Answer:**

c) A set of consumers sharing a common group identifier

### **Explanation:**

A Consumer Group consists of multiple consumers that share a common identifier. They work together to consume data, ensuring each record is processed once.

## **23. Why would you use multiple consumers in a Consumer Group?**

a) To produce data on multiple topics

b) To consume data from multiple clusters

c) To achieve parallel processing of data and improve consumption speed

d) To backup data in Kafka

### **Answer:**

c) To achieve parallel processing of data and improve consumption speed

### **Explanation:**

Having multiple consumers in a consumer group allows them to read from different partitions in parallel, speeding up data consumption.

## **24. What is the primary role of ZooKeeper in a Kafka cluster?**

a) Storing actual message data.

b) Balancing load between Kafka brokers.

c) Managing topic and partition metadata.

d) Compressing data for faster transmission.

### **Answer:**

c) Managing topic and partition metadata.

### **Explanation:**

In the Kafka ecosystem, ZooKeeper's main role is to manage broker metadata, such as topic and partition information. It doesn't store the actual message data; that's handled by the Kafka brokers. ZooKeeper ensures all broker nodes have consistent metadata, making the cluster robust and fault-tolerant.

## **25. If ZooKeeper fails in a Kafka cluster, what is the most likely immediate impact?**

a) Message data will be lost.

b) New topics cannot be created, but existing topics will continue to function.

c) The entire Kafka cluster will go offline.

d) Kafka will start using another tool automatically.

### **Answer:**

b) New topics cannot be created, but existing topics will continue to function.

### **Explanation:**

While ZooKeeper is vital for the management of metadata within a Kafka cluster, its failure doesn't imply the loss of message data or the entire Kafka cluster going offline. Existing topics will continue to operate since the brokers have the information they need for ongoing operations. However, operations that require coordination, such as creating new topics, will not be possible until ZooKeeper is restored.

**Git**

## **1. What is Git?**

a) A distributed version control system

b) A centralized version control system

c) A text editor

d) A compiler

### **Answer:**

a) A distributed version control system

### **Explanation:**

Git is a distributed version control system that allows multiple developers to collaborate on a project. Each developer has a full copy of the project history locally. It is designed to track changes in source code and manage project versions.

Unlike centralized systems, Git allows developers to work offline and merge changes efficiently when they're back online. This makes it ideal for large projects and remote collaboration.

Git is widely used in software development because of its flexibility, speed, and distributed architecture.

## **2. What is the primary purpose of GitHub?**

a) To host Git repositories

b) To write code

c) To debug programs

d) To compile code

### **Answer:**

a) To host Git repositories

### **Explanation:**

GitHub is a web-based platform used for hosting Git repositories. It enables collaboration by providing tools for version control, issue tracking, and project management, making it easier for teams to work together.

GitHub also integrates with various development tools and offers features like pull requests and code reviews, which streamline the development workflow.

GitHub is widely used in both open-source and enterprise projects, allowing developers to share, contribute to, and manage code efficiently.

## **3. What is the command to initialize a new Git repository?**

a) git init

b) git start

c) git create

d) git begin

### **Answer:**

a) git init

### **Explanation:**

The git init command initializes a new Git repository in the current directory. It creates a hidden .git folder that stores the repository’s metadata, including history and configuration.

This command is used when starting a new project or converting an existing directory into a Git repository. Once the repository is initialized, you can start tracking files and committing changes.

Understanding git init is fundamental for using Git, as it is the starting point for any new repository.

## **4. What is a commit in Git?**

a) A snapshot of changes

b) A branch

c) A remote repository

d) A bug fix

### **Answer:**

a) A snapshot of changes

### **Explanation:**

A commit in Git is a snapshot of the project at a specific point in time. It captures the changes made to files and directories, storing them in the repository's history. Each commit has a unique identifier (SHA) and contains metadata such as the author, date, and commit message.

Commits form the foundation of Git's version control system, allowing you to track progress, revert to previous states, and collaborate with others by sharing these snapshots.

Effective use of commits helps maintain a clean and understandable project history, enabling better collaboration and project management.

## **5. How do you check the status of your Git repository?**

a) git check

b) git status

c) git info

d) git log

### **Answer:**

b) git status

### **Explanation:**

The git status command displays the state of your working directory and staging area. It shows which files are staged for the next commit, which are modified but not yet staged, and which are untracked by Git.

This command is crucial for keeping track of changes in your project and ensuring that you know what will be committed or what still needs to be added to the staging area.

Regular use of git status helps prevent mistakes by giving you a clear picture of the current state of your repository.

## **6. How do you add changes to the staging area in Git?**

a) git add

b) git commit

c) git stage

d) git push

### **Answer:**

a) git add

### **Explanation:**

The git add command is used to add changes to the staging area in Git. This prepares the files for the next commit. You can add individual files, multiple files, or all changes using commands like git add filename or git add . to stage everything.

Staging allows you to control which changes will be committed. It's a useful feature for managing and organizing changes before creating a commit.

Using git add effectively ensures that only the intended changes are included in each commit, maintaining a clean project history.

## **7. How do you commit changes in Git?**

a) git add

b) git commit

c) git push

d) git save

### **Answer:**

b) git commit

### **Explanation:**

The git commit command is used to save changes to the local repository. After staging changes with git add, you can use git commit -m "message" to create a commit with a message describing the changes.

Each commit is a snapshot of the project’s state at a particular point in time, allowing you to track changes and revert to earlier versions if needed.

Frequent, descriptive commits make it easier to manage and collaborate on projects, providing a clear history of development.

## **8. What is the command to view the commit history in Git?**

a) git history

b) git log

c) git commit

d) git show

### **Answer:**

b) git log

### **Explanation:**

The git log command displays the commit history of the repository. It shows details like the commit hash, author, date, and the commit message. By default, it lists all the commits in reverse chronological order.

You can use options like git log --oneline to view a condensed version or git log -p to show the differences introduced by each commit.

Understanding the commit history is important for tracking changes, debugging issues, and reviewing the project’s development over time.

## **9. How do you remove a file from the staging area in Git?**

a) git reset

b) git delete

c) git remove

d) git rm --cached

### **Answer:**

d) git rm --cached

### **Explanation:**

The git rm --cached command is used to remove a file from the staging area without deleting it from the working directory. This command only unstages the file, so it will not be included in the next commit, but it remains in the directory.

For example, if you’ve added a file by mistake, you can use git rm --cached filename to unstage it. The file will still exist locally but won't be committed until it is re-added.

This command is useful when you want to exclude files from a commit or undo a mistaken git add operation without losing the file entirely.

## **10. What is a branch in Git?**

a) A pointer to a specific commit

b) A remote repository

c) A bug fix

d) A tag

### **Answer:**

a) A pointer to a specific commit

### **Explanation:**

A branch in Git is a pointer to a specific commit in the repository. It allows developers to work on different features, bug fixes, or experiments independently without affecting the main project.

For example, when you create a new branch with git branch new-feature, it starts from the current commit. Any changes you make on this branch will not affect the main branch until they are merged.

Branches provide a flexible way to manage multiple streams of work in parallel, making Git a powerful tool for collaborative development.

## **11. What is the command to create a new branch in Git?**

a) git branch new-branch

b) git new-branch

c) git checkout branch

d) git switch branch

### **Answer:**

a) git branch new-branch

### **Explanation:**

The git branch new-branch command creates a new branch named "new-branch" in the current repository. This branch is based on the current commit and allows you to work on changes independently from other branches.

Once the new branch is created, you can switch to it using git checkout new-branch or git switch new-branch to start working on it. This separation allows multiple developers to work on different features without affecting each other's work.

Branches are fundamental to collaborative workflows, enabling teams to develop features in parallel and later merge them into the main project.

## **12. What is the command to switch to another branch in Git?**

a) git checkout branch-name

b) git switch branch-name

c) Both a and b

d) git change branch-name

### **Answer:**

c) Both a and b

### **Explanation:**

In Git, you can switch to another branch using either git checkout branch-name or git switch branch-name. The checkout command was traditionally used for this purpose, but newer versions of Git introduced the switch command for clarity.

Both commands achieve the same result, but switch is more intuitive for switching branches, whereas checkout can be used for other operations like switching to a specific commit.

Switching branches allows you to move between different streams of development and work on multiple features without losing progress on any of them.

## **13. How do you merge a branch into the current branch in Git?**

a) git merge branch-name

b) git merge commit

c) git combine branch-name

d) git rebase branch-name

### **Answer:**

a) git merge branch-name

### **Explanation:**

The git merge branch-name command is used to merge another branch into the current branch. For example, if you're on the main branch and want to merge a feature branch, you would run git merge feature-branch.

Merging combines the changes from both branches, allowing you to integrate new features or updates into the main branch. If there are conflicts between the branches, Git will prompt you to resolve them before completing the merge.

Merging is a common operation in Git, as it allows developers to bring their independent work together into a single, unified project.

## **14. What is a conflict in Git?**

a) When two branches have changes to the same lines of code

b) When a file is deleted

c) When a file is added

d) When the repository is corrupted

### **Answer:**

a) When two branches have changes to the same lines of code

### **Explanation:**

A conflict in Git occurs when two branches have made changes to the same lines of code or the same file, and Git is unable to automatically merge them. This typically happens when different developers make changes to the same part of the codebase.

When a conflict occurs, Git will mark the conflicting sections in the file, and it's up to the developer to manually resolve the conflict by choosing which changes to keep.

Understanding how to resolve conflicts is crucial for collaborative development, ensuring that all team members’ changes are properly integrated into the project.

## **15. What is a pull request in GitHub?**

a) A request to merge changes from one branch to another

b) A request to delete a branch

c) A request to clone a repository

d) A request to reset a branch

### **Answer:**

a) A request to merge changes from one branch to another

### **Explanation:**

A pull request in GitHub is a request to merge changes from one branch into another, typically from a feature branch to the main branch. It allows developers to review, discuss, and approve changes before they are merged into the main project.

Pull requests facilitate collaboration by providing a formal process for reviewing code changes and ensuring that all team members have an opportunity to give feedback or suggest improvements.

Using pull requests in GitHub is an essential part of modern development workflows, promoting better code quality and collaboration.

## **16. How do you clone a Git repository?**

a) git clone

b) git pull

c) git fetch

d) git copy

### **Answer:**

a) git clone

### **Explanation:**

The git clone command is used to copy a Git repository from a remote server to your local machine. It downloads the entire history of the repository and creates a working directory for you to start working on the project.

For example, running git clone https://github.com/user/repo.git will clone the repository to your local system. This command is typically used to start working on an existing project.

Understanding how to clone repositories is essential for collaborative development, as it allows you to work on projects hosted on GitHub or other Git hosting services.

## **17. What is the command to fetch changes from a remote repository without merging them?**

a) git pull

b) git fetch

c) git merge

d) git update

### **Answer:**

b) git fetch

### **Explanation:**

The git fetch command retrieves updates from a remote repository but does not automatically merge them into your working directory. This allows you to review the changes before deciding whether to merge them into your branch.

After fetching changes, you can use git merge to integrate the fetched changes into your local branch. This is useful for safely incorporating updates without automatically applying them.

Using git fetch gives you more control over the update process, especially when working on critical code that requires careful review before merging.

## **18. How do you push your changes to a remote repository?**

a) git commit

b) git push

c) git send

d) git upload

### **Answer:**

b) git push

### **Explanation:**

The git push command is used to upload local repository changes to a remote repository. It transfers commits from your local branch to the corresponding branch on the remote server, making your changes available to other collaborators.

For example, git push origin main will push your local changes to the main branch on the remote repository named origin.

Understanding git push is crucial for sharing your work with others and collaborating on projects hosted on platforms like GitHub.

## **19. What is a fork in GitHub?**

a) A copy of a repository in your GitHub account

b) A branch of a repository

c) A pull request

d) A reset of a branch

### **Answer:**

a) A copy of a repository in your GitHub account

### **Explanation:**

A fork in GitHub is a personal copy of someone else's repository that you create in your GitHub account. Forking a repository allows you to experiment with changes without affecting the original project.

After forking a repository, you can make changes to it and later submit a pull request to the original repository if you want to contribute your changes. This is commonly used in open-source projects to encourage collaboration.

Forking is an essential feature of GitHub, enabling developers to work on independent copies of projects and contribute back to the main repository.

## **20. How do you pull changes from a remote repository and merge them into your local branch?**

a) git fetch

b) git pull

c) git merge

d) git checkout

### **Answer:**

b) git pull

### **Explanation:**

The git pull command is used to fetch changes from a remote repository and automatically merge them into your local branch. It combines the functionality of git fetch and git merge in a single step.

For example, git pull origin main will pull the latest changes from the main branch of the origin remote repository and merge them into your local branch.

Using git pull ensures that your local branch stays up-to-date with the remote repository, especially when working collaboratively on shared projects.

## **21. How do you delete a branch locally in Git?**

a) git branch -d branch-name

b) git remove branch-name

c) git delete branch-name

d) git erase branch-name

### **Answer:**

a) git branch -d branch-name

### **Explanation:**

The git branch -d branch-name command is used to delete a branch locally in Git. It removes the specified branch from your local repository but does not affect any remote branches.

For example, git branch -d feature-branch deletes the feature-branch from your local environment. If you want to force-delete a branch, you can use -D instead of -d.

Deleting branches is important for keeping your repository organized, especially after a feature branch has been merged or is no longer needed.

## **22. What does git stash do in Git?**

a) Temporarily saves uncommitted changes

b) Merges branches

c) Deletes a branch

d) Resets the repository

### **Answer:**

a) Temporarily saves uncommitted changes

### **Explanation:**

The git stash command temporarily saves uncommitted changes in your working directory without committing them. It allows you to clean your working directory without losing changes and later restore them when needed.

For example, if you are working on some code but need to switch branches, you can run git stash to save your changes. After switching branches, you can use git stash pop to restore the stashed changes.

git stash is useful for temporarily shelving work when you need to focus on something else or switch branches without committing unfinished changes.

## **23. How do you list all the branches in a Git repository?**

a) git list

b) git branches

c) git show-branches

d) git branch

### **Answer:**

d) git branch

### **Explanation:**

The git branch command lists all the branches in your repository. It shows the currently checked-out branch with an asterisk (\*) next to it.

For example, running git branch displays a list of all local branches. To view both local and remote branches, you can use git branch -a.

Listing branches helps you keep track of different streams of work in your repository and easily switch between them when necessary.

## **24. How do you reset the current branch to a specific commit?**

a) git revert commit-hash

b) git reset commit-hash

c) git delete commit-hash

d) git undo commit-hash

### **Answer:**

b) git reset commit-hash

### **Explanation:**

The git reset commit-hash command resets the current branch to a specific commit. It modifies the commit history by moving the current branch pointer to the specified commit and optionally changing the working directory and index.

For example, running git reset --hard abc123 will reset the branch to commit abc123 and remove any changes made after that commit.

git reset is useful when you need to undo recent changes or return the repository to a previous state.

## **25. What is the command to undo the last commit in Git?**

a) git reset --hard HEAD^

b) git revert HEAD

c) Both a and b

d) git delete HEAD

### **Answer:**

c) Both a and b

### **Explanation:**

In Git, you can undo the last commit using either git reset --hard HEAD^ or git revert HEAD. The git reset command removes the commit and changes, while git revert creates a new commit that undoes the changes made by the last commit.

For example, git reset --hard HEAD^ moves the HEAD pointer to the previous commit and discards all changes. On the other hand, git revert HEAD is safer because it doesn't alter the commit history, making it ideal for shared branches.

Understanding both methods is essential for correcting mistakes in Git while preserving the integrity of your project’s commit history.

## **26. What is the purpose of a GitHub repository's README.md file?**

a) To describe the project and provide instructions

b) To store sensitive information

c) To commit code changes

d) To manage branches

### **Answer:**

a) To describe the project and provide instructions

### **Explanation:**

A README.md file in a GitHub repository is used to describe the project, provide installation instructions, usage details, and any other relevant information. It is written in Markdown and typically serves as the first document users see when they visit the repository.

For example, a README.md file might contain instructions on how to install and use the project, as well as contribution guidelines.

Having a well-written README.md is crucial for making your repository more accessible to contributors and users, improving the overall project documentation.

## **27. How do you remove a file from both the working directory and Git index?**

a) git delete file

b) git rm file

c) git remove file

d) git reset file

### **Answer:**

b) git rm file

### **Explanation:**

The git rm file command removes a file from both the working directory and the Git index (staging area). This means the file will no longer be tracked by Git, and it will also be deleted from your local file system.

For example, running git rm filename will stage the file for deletion in the next commit and remove it from your local directory.

Use this command when you want to permanently remove a file from the project and stop tracking it in future commits.

## **28. What is a Git tag?**

a) A label for a specific commit

b) A branch of a repository

c) A commit message

d) A pull request

### **Answer:**

a) A label for a specific commit

### **Explanation:**

A Git tag is a label used to mark a specific commit in the repository's history. Tags are typically used to indicate important milestones, such as software releases (e.g., v1.0).

You can create a tag using git tag tagname, which points to the current commit. Tags are helpful for referencing a specific point in the repository’s history without worrying about changes to branches.

Using tags makes it easier to manage and reference specific versions of a project, particularly when releasing software to users.

## **29. How do you create an annotated tag in Git?**

a) git tag -a tagname

b) git create tagname

c) git annotate tagname

d) git label tagname

### **Answer:**

a) git tag -a tagname

### **Explanation:**

The git tag -a tagname command creates an annotated tag in Git. Annotated tags include additional metadata such as the author’s name, date, and a message, making them more informative than lightweight tags.

For example, running git tag -a v1.0 -m "Version 1.0 release" creates an annotated tag for version 1.0 with a message describing the tag.

Annotated tags are useful for marking releases and milestones, providing more context and history than lightweight tags.

## **30. How do you view the history of tags in Git?**

a) git show-tags

b) git tag

c) git log --tags

d) git history

### **Answer:**

b) git tag

### **Explanation:**

The git tag command lists all tags in the repository. Tags are useful for marking important commits, such as release points, and the git tag command helps you see all available tags in one place.

You can use options like git tag -l "v1.\*" to list tags matching a specific pattern (e.g., all version 1 tags). This is helpful when navigating through different versions of the project.

Tags are crucial for version control and release management, providing a way to reference specific points in the project’s history.

## **31. What is a Git remote?**

a) A repository hosted on a server

b) A local branch

c) A tag

d) A commit

### **Answer:**

a) A repository hosted on a server

### **Explanation:**

A Git remote is a reference to a repository hosted on a server, such as GitHub, GitLab, or a private server. It allows you to collaborate with others by pushing and pulling changes between your local repository and the remote one.

For example, origin is a common name for the default remote repository when you clone a repository. You can add additional remotes using git remote add.

Understanding remotes is key to collaborating with other developers and managing distributed workflows in Git.

## **32. How do you add a remote repository in Git?**

a) git add remote

b) git remote add

c) git remote new

d) git push remote

### **Answer:**

b) git remote add

### **Explanation:**

The git remote add command is used to add a new remote repository to your Git configuration. It links your local repository to the remote, allowing you to push and pull changes between them.

For example, git remote add origin https://github.com/user/repo.git adds the origin remote, which points to the specified GitHub repository.

Adding a remote is essential for collaborating on projects, as it enables you to synchronize your work with a central repository.

## **33. What is the purpose of the git diff command?**

a) To display the differences between two commits

b) To switch branches

c) To reset the repository

d) To delete a branch

### **Answer:**

a) To display the differences between two commits

### **Explanation:**

The git diff command is used to display the differences between commits, branches, or the working directory and staging area. It shows what has changed between two states, helping you review code before committing or merging changes.

For example, git diff compares the working directory with the staging area, while git diff commit1 commit2 compares two specific commits.

Understanding git diff is important for reviewing changes, spotting errors, and ensuring that only the intended modifications are committed.

## **34. How do you rename a Git branch?**

a) git rename branch new-name

b) git branch -m old-name new-name

c) git branch new-name

d) git switch branch new-name

### **Answer:**

b) git branch -m old-name new-name

### **Explanation:**

The git branch -m old-name new-name command is used to rename a branch in Git. It changes the name of the branch both locally and remotely if you push the updated branch to the remote repository.

For example, running git branch -m feature-branch awesome-feature will rename feature-branch to awesome-feature.

Renaming branches helps keep your repository organized, especially when the branch name no longer reflects the current work or feature.

## **35. What does the git log --oneline command do?**

a) Displays a condensed view of the commit history

b) Shows the current branch

c) Displays the differences between commits

d) Displays the working directory status

### **Answer:**

a) Displays a condensed view of the commit history

### **Explanation:**

The git log --oneline command displays the commit history in a condensed format, showing only the commit hash and message in a single line for each commit. This provides a more readable and compact view of the repository’s history.

For example, running git log --oneline outputs a simplified list of all commits, making it easier to quickly review past changes.

This command is useful when you want a quick overview of the repository’s history without detailed commit information.

## **36. How do you rebase a branch in Git?**

a) git merge branch

b) git rebase branch

c) git pull branch

d) git sync branch

### **Answer:**

b) git rebase branch

### **Explanation:**

The git rebase branch command is used to apply the commits from one branch on top of another. It rewrites the commit history to make it appear as though the branch was created from the latest commit in the base branch.

For example, git rebase main will reapply your branch’s commits on top of the latest commits in the main branch.

Rebasing is useful for keeping a clean and linear project history, but it should be used carefully, especially on shared branches.

## **37. What is the difference between git pull and git fetch?**

a) git pull fetches and merges changes, while git fetch only downloads changes

b) git fetch deletes remote changes, while git pull updates your repository

c) git fetch resets the repository, while git pull clones a new repository

d) Both commands do the same thing

### **Answer:**

a) git pull fetches and merges changes, while git fetch only downloads changes

### **Explanation:**

The difference between git pull and git fetch is that git pull fetches changes from a remote repository and merges them into your local branch, while git fetch only downloads the changes without merging them.

After fetching changes, you can manually merge them into your local branch. Pulling automatically merges the changes, saving you a step.

Knowing when to use each command helps you control the update process and avoid unintended merges.

## **38. How do you compare two commits in Git?**

a) git diff commit1 commit2

b) git compare commit1 commit2

c) git log commit1 commit2

d) git show commit1 commit2

### **Answer:**

a) git diff commit1 commit2

### **Explanation:**

The git diff commit1 commit2 command compares the differences between two commits. It shows the changes made in each commit, helping you understand what has been modified between two points in the project’s history.

For example, git diff abc123 def456 compares the changes between the commits with hashes abc123 and def456.

Understanding how to compare commits is essential for reviewing code, debugging, and tracking the evolution of your project.

## **39. What is the purpose of git blame?**

a) To show who made changes to each line of a file

b) To delete a commit

c) To reset the repository

d) To merge branches

### **Answer:**

a) To show who made changes to each line of a file

### **Explanation:**

The git blame command shows who made changes to each line of a file, along with the commit hash and timestamp for each change. This is useful for identifying when and by whom specific changes were introduced.

For example, running git blame filename will display each line of the file, along with the author and commit details responsible for that line.

git blame is helpful for tracking down the source of bugs or understanding the history of changes to a particular file.

## **40. What is the purpose of .gitignore?**

a) To specify files and directories that Git should ignore

b) To reset the repository

c) To clone a repository

d) To view the commit history

### **Answer:**

a) To specify files and directories that Git should ignore

### **Explanation:**

The .gitignore file is used to specify files and directories that Git should ignore. This prevents certain files (such as temporary files, build outputs, or sensitive information) from being tracked or included in commits.

For example, a typical .gitignore might include entries like \*.log to ignore all log files or node\_modules/ to ignore the node\_modules directory in a JavaScript project.

Using .gitignore helps keep your repository clean by excluding unnecessary or sensitive files from version control.

## **41. What does the command git remote -v do?**

a) Lists all remote repositories with URLs

b) Verifies the connection to the remote repository

c) Deletes a remote repository

d) Displays commit history from the remote repository

### **Answer:**

a) Lists all remote repositories with URLs

### **Explanation:**

The git remote -v command lists all remote repositories associated with your local Git repository, along with their URLs. The -v flag stands for "verbose," and it displays both the fetch and push URLs for each remote.

This command is useful for checking the configuration of your remotes, especially when working with multiple remotes or verifying that your repository is correctly linked to its GitHub counterpart.

It’s an essential tool for managing remote repositories in collaborative workflows.

## **42. How do you undo changes in the working directory that haven’t been committed yet?**

a) git revert

b) git reset --hard

c) git checkout -- file

d) git delete

### **Answer:**

c) git checkout -- file

### **Explanation:**

The command git checkout -- file is used to undo changes in the working directory for a specific file, reverting it to the state from the last commit. It does not affect staged changes, only changes in the working directory.

For example, if you’ve made changes to file.txt but haven’t committed them yet, running git checkout -- file.txt will discard the modifications.

This is useful when you want to revert uncommitted changes and reset the file to its previous state.

## **43. What does the command git clean -f do?**

a) Deletes untracked files from the working directory

b) Resets all staged changes

c) Resets the repository to its last commit

d) Deletes remote branches

### **Answer:**

a) Deletes untracked files from the working directory

### **Explanation:**

The git clean -f command removes untracked files from the working directory. These are files that are not part of the version control system. The -f flag stands for "force" and is required to perform this operation.

For example, running git clean -f will delete any untracked files, helping to clean up your working directory when you no longer need those files.

This command is useful for tidying up your workspace, but it should be used with caution, as deleted files cannot be recovered from Git.

## **44. How do you discard changes from the staging area in Git?**

a) git remove file

b) git reset HEAD file

c) git revert HEAD file

d) git clean -f file

### **Answer:**

b) git reset HEAD file

### **Explanation:**

The git reset HEAD file command removes a file from the staging area without deleting the changes in the working directory. This allows you to unstage the file but keep the changes intact locally.

For example, if you’ve staged file.txt and decide you don’t want to include it in the next commit, you can run git reset HEAD file.txt to remove it from the staging area.

This command is useful for managing the content that will be included in the next commit without losing any changes in the file.

## **45. What does git reflog display?**

a) A history of all reference updates

b) A list of untracked files

c) A history of all pull requests

d) A list of all merged branches

### **Answer:**

a) A history of all reference updates

### **Explanation:**

The git reflog command displays a history of all reference updates in your Git repository, including branch checkouts, commits, and resets. It shows all movements of the HEAD pointer, even those that are not recorded in the commit history.

For example, running git reflog will list every change made to HEAD, allowing you to recover lost commits or branches that were reset or modified.

git reflog is especially useful for undoing mistakes and tracking the repository's history more comprehensively than the regular git log.

## **46. How do you amend the last commit in Git?**

a) git commit --amend

b) git reset --amend

c) git change HEAD

d) git fix commit

### **Answer:**

a) git commit --amend

### **Explanation:**

The git commit --amend command is used to modify the last commit. This allows you to change the commit message or include additional changes without creating a new commit.

For example, if you forgot to include a file in your last commit, you can stage the file and run git commit --amend to add it to the previous commit.

This command is useful for making minor corrections to your last commit without cluttering the commit history.

## **47. How do you merge changes from the remote repository into your local branch?**

a) git pull

b) git fetch

c) git merge

d) git commit

### **Answer:**

a) git pull

### **Explanation:**

The git pull command is used to fetch changes from a remote repository and automatically merge them into your local branch. It combines the functionality of git fetch and git merge into a single command.

For example, git pull origin main will fetch changes from the main branch of the origin remote and merge them into your local branch.

This command is essential for keeping your local repository synchronized with the remote repository, especially when working collaboratively on a project.

## **48. What does git cherry-pick do?**

a) Applies a specific commit from one branch to another

b) Deletes a specific commit

c) Creates a new branch

d) Reverts changes made in a commit

### **Answer:**

a) Applies a specific commit from one branch to another

### **Explanation:**

The git cherry-pick command applies the changes from a specific commit to the current branch. This allows you to take one or more commits from another branch and apply them individually, without merging the entire branch.

For example, running git cherry-pick abc123 will apply the commit with hash abc123 to the current branch.

This command is useful for selectively bringing changes into a branch, especially when you only need certain updates or fixes from another branch.

## **49. What does the command git bisect do?**

a) Helps find the commit that introduced a bug

b) Splits a commit into multiple parts

c) Creates a new commit

d) Shows the differences between two commits

### **Answer:**

a) Helps find the commit that introduced a bug

### **Explanation:**

The git bisect command helps you find the commit that introduced a bug by performing a binary search through the commit history. It repeatedly checks for bad commits by asking you to mark them as "good" or "bad" until the problematic commit is identified.

For example, running git bisect start initiates the bisect process, and Git will guide you through checking each commit in the search range.

This is a powerful debugging tool that allows you to quickly locate the commit where an issue was introduced.

## **50. How do you list all tags in a Git repository?**

a) git tags

b) git tag --list

c) git tag -a

d) git list-tags

### **Answer:**

b) git tag --list

### **Explanation:**

The git tag --list command lists all tags in the repository. Tags are typically used to mark specific points in the repository's history, such as releases or important milestones.

You can also use git tag without options to list all tags, or apply filters like git tag -l "v1.\*" to list specific versions.

Tags are an important part of version control, allowing you to reference key points in your project's history.

## **51. What does the git shortlog command do?**

a) Summarizes the commit history by author

b) Displays the full commit history

c) Shows a brief log of remote changes

d) Displays file-level changes

### **Answer:**

a) Summarizes the commit history by author

### **Explanation:**

The git shortlog command provides a summary of the commit history grouped by author. It displays the number of commits each contributor has made, along with their commit messages.

This command is useful for getting an overview of who contributed to the project and how much work they’ve done without going through the entire log.

Understanding git shortlog is helpful for assessing contributions, especially in collaborative projects.

## **52. What is a “detached HEAD” in Git?**

a) A state where HEAD points to a commit instead of a branch

b) A corrupted branch

c) A repository without a HEAD

d) A branch without commits

### **Answer:**

a) A state where HEAD points to a commit instead of a branch

### **Explanation:**

A “detached HEAD” state in Git occurs when HEAD points directly to a commit rather than a branch. This can happen when you checkout a specific commit instead of a branch, meaning any new commits you make won’t be associated with a branch.

While in a detached HEAD state, you can still make changes, but they are not linked to a branch unless you explicitly create one.

Understanding the detached HEAD state is important for avoiding confusion when working with commits that are not linked to branches.

## **53. How do you reapply commits on top of another base branch in Git?**

a) git merge

b) git rebase

c) git pull

d) git cherry-pick

### **Answer:**

b) git rebase

### **Explanation:**

The git rebase command re-applies commits from the current branch on top of another base branch. It allows you to integrate changes from the base branch and rewrite commit history as if the branch was originally created from the updated base.

For example, git rebase main will rebase the current branch on top of the main branch, applying your changes after the latest main commits.

Rebasing creates a linear history, making it cleaner but should be used cautiously to avoid conflicts in shared branches.

## **54. What does the git stash apply command do?**

a) Applies the latest stashed changes without deleting them

b) Deletes stashed changes

c) Commits stashed changes

d) Creates a new stash

### **Answer:**

a) Applies the latest stashed changes without deleting them

### **Explanation:**

The git stash apply command applies the latest stashed changes back into your working directory without deleting them from the stash. This allows you to restore the saved state while keeping the stash intact for future use.

For example, if you have stashed changes and want to reapply them later, git stash apply will restore those changes while keeping the stash available.

Using git stash apply is helpful when you need to temporarily save and later restore changes during context switching.

## **55. What does the command git pull --rebase do?**

a) Fetches and rebases changes from the remote branch

b) Fetches and creates a new branch

c) Deletes remote changes

d) Creates a new commit on the remote repository

### **Answer:**

a) Fetches and rebases changes from the remote branch

### **Explanation:**

The git pull --rebase command fetches changes from the remote repository and rebases them onto your local branch instead of merging them. This ensures a linear commit history by applying your local commits on top of the fetched changes.

This is useful when you want to avoid merge commits and keep the project history cleaner.

Rebasing during a pull avoids unnecessary merge conflicts and ensures that your local changes are reapplied after the latest remote updates.

## **56. What does the git archive command do?**

a) Creates a zip or tar archive of the repository

b) Archives old commits

c) Deletes branches

d) Shows archived commits

### **Answer:**

a) Creates a zip or tar archive of the repository

### **Explanation:**

The git archive command is used to create an archive (zip, tar, etc.) of the contents of a Git repository. It doesn’t include the Git metadata but packages the working directory files at a particular point in time.

This is useful for generating distribution-ready archives of a specific version of the project without including the entire repository history.

You can also use git archive to export a particular branch or tag.

## **57. How do you view all stashes in a Git repository?**

a) git stash list

b) git log --stash

c) git stash show

d) git reflog stash

### **Answer:**

a) git stash list

### **Explanation:**

The git stash list command shows all the stashes saved in the repository. Each stash is listed with a unique identifier, making it easier to reference and apply later.

For example, running git stash list will output a list of all saved stashes, including the latest one and any previously created stashes.

This command is useful for managing multiple stashes and reapplying or dropping them as needed.

## **58. What does the git diff --cached command show?**

a) Differences between the index (staging area) and the last commit

b) Differences between working directory and the index

c) Differences between two commits

d) Differences between local and remote branches

### **Answer:**

a) Differences between the index (staging area) and the last commit

### **Explanation:**

The git diff --cached command shows the differences between the index (staging area) and the last commit. It helps you review the changes that are about to be committed, allowing you to see what has been staged.

For example, running git diff --cached before committing will show you exactly what changes have been staged for the next commit.

This command is crucial for ensuring that only the intended changes are included in a commit.

## **59. How do you tag a specific commit in Git?**

a) git tag tagname commit-hash

b) git tag -m commit-hash

c) git annotate commit-hash

d) git commit -t

### **Answer:**

a) git tag tagname commit-hash

### **Explanation:**

The git tag tagname commit-hash command tags a specific commit in Git by providing the tag name and commit hash. This allows you to mark a specific commit with a tag, even if it’s not the latest commit on a branch.

For example, running git tag v1.0 abc123 tags the commit with hash abc123 as version 1.0.

Tagging specific commits is useful for marking important milestones, such as software releases, in your project.

## **60. What does git push origin --tags do?**

a) Pushes all tags to the remote repository

b) Pushes only the latest tag

c) Deletes remote tags

d) Pushes tags and removes untracked files

### **Answer:**

a) Pushes all tags to the remote repository

### **Explanation:**

The git push origin --tags command pushes all tags from your local repository to the remote repository. Tags are not automatically pushed with commits, so this command is necessary to make your tags available on the remote server.

For example, if you’ve created several tags locally and want to share them on GitHub or another remote, you can run git push origin --tags to push them all at once.

This is important when managing versioned releases in a project where tags mark key points like software versions.

## **61. What does the git stash drop command do?**

a) Deletes the latest stash

b) Applies the latest stash

c) Lists all stashes

d) Commits the latest stash

### **Answer:**

a) Deletes the latest stash

### **Explanation:**

The git stash drop command deletes the latest stash from the stash list. This is useful when you no longer need a stash and want to remove it to keep the stash list clean.

For example, running git stash drop will remove the most recent stash, while specifying a stash with its identifier (e.g., git stash drop stash@{1}) will remove a specific stash.

Keeping your stash list organized is essential for managing multiple sets of saved changes effectively.

## **62. What is the command to view the commit history of a file?**

a) git log filename

b) git history filename

c) git show-log filename

d) git commit-log filename

### **Answer:**

a) git log filename

### **Explanation:**

The git log filename command shows the commit history of a specific file. It lists all commits that modified the file, allowing you to trace the evolution of that file through the project’s history.

This is useful for tracking changes made to a particular file, identifying when a bug was introduced, or understanding the file’s development history.

Using git log filename helps you analyze how and when specific changes were made to the file.

## **63. What does the git show command do?**

a) Displays details about a specific commit

b) Shows a list of all branches

c) Shows the current branch status

d) Displays a list of stashes

### **Answer:**

a) Displays details about a specific commit

### **Explanation:**

The git show command displays details about a specific commit, including the commit message, changes made, and file differences. By default, it shows the most recent commit, but you can specify a commit hash to view any commit in the history.

For example, git show abc123 shows the details of the commit with hash abc123, including the author, date, and changes made.

This command is useful for reviewing individual commits and understanding their impact on the project.

## **64. How do you rename a file in Git?**

a) git mv oldname newname

b) git rename oldname newname

c) git move oldname newname

d) git update-file oldname newname

### **Answer:**

a) git mv oldname newname

### **Explanation:**

Renaming the file is very important when we are working on a project. Git Rename is used to rename a file in the working directory. Suppose we are working in a team and any fellow developer created the file and we want to rename it using Git or Github so this command help us to do so. We may change the name of the file even though we created the file due to some reason. To rename any file or folder, use git mv command which takes two arguments. The first argument is the source and the second is the destination. We can easily rename any file using the git command and the new name will be assigned to that file. We can rename the file using GitHub or the command line.

Step 1: Open Git Bash.

Step 2: Open the repository.

Step 3: Rename the file using the command:

git mv old\_filename new\_filename

Step 4: Use the “git status” command to check the changes.

Step 5: Commit the renamed file.

git commit -m "Renamed\_file"

Step 6: Push the changes using git push origin branch\_name

## **65. What does git rm --cached do?**

a) Removes a file from the staging area but keeps it in the working directory

b) Deletes a file from both the staging area and working directory

c) Moves a file to a new location

d) Reverts the last commit

### **Answer:**

a) Removes a file from the staging area but keeps it in the working directory

### **Explanation:**

The git rm --cached command removes a file from the staging area but keeps it in the working directory. This unstages the file, preventing it from being included in the next commit, but the file itself remains in your local directory.

For example, running git rm --cached filename will untrack the file but keep it in the project directory.

This command is useful when you want to remove files from version control without deleting them locally.

## **66. What is a Git hook?**

a) A script that runs automatically before or after Git events

b) A tool for resolving merge conflicts

c) A command to check remote branches

d) A tag for annotating commits

### **Answer:**

a) A script that runs automatically before or after Git events

### **Explanation:**

A Git hook is a script that runs automatically before or after specific Git events, such as commits, merges, or pushes. Hooks allow you to automate tasks, such as running tests, formatting code, or enforcing coding standards.

For example, a pre-commit hook can be used to check for code formatting issues before a commit is made.

Hooks are powerful tools for ensuring code quality and enforcing development workflows.

## **67. How do you resolve merge conflicts in Git?**

a) Manually edit the conflicting files and commit the changes

b) Run git resolve

c) Delete the conflicting files

d) Reset the repository to a previous state

### **Answer:**

a) Manually edit the conflicting files and commit the changes

### **Explanation:**

To resolve merge conflicts in Git, you must manually edit the conflicting files to reconcile the changes, then stage and commit the resolved files. Git marks the conflicting sections in the files, and it's up to you to decide which changes to keep.

After resolving the conflicts, you can commit the changes with a new commit.

This process is essential for integrating code from different branches when changes have been made to the same parts of a file.

## **68. What is the command to list all untracked files in Git?**

a) git status

b) git ls-files --others

c) git show untracked

d) git log --untracked

### **Answer:**

b) git ls-files --others

### **Explanation:**

The command git ls-files --others lists all untracked files in your working directory. These are files that have not been added to the index (staging area) and are not being tracked by Git.

Alternatively, you can use git status to see untracked files along with other status information.

Knowing which files are untracked is important when deciding which files to include in your repository.

## **69. How do you squash commits in Git?**

a) By using git rebase -i (interactive rebase)

b) By using git squash

c) By using git commit --squash

d) By using git combine

### **Answer:**

a) By using git rebase -i (interactive rebase)

### **Explanation:**

To squash commits in Git, you use the git rebase -i (interactive rebase) command. Squashing commits combines multiple commits into a single one, simplifying the commit history.

During the rebase process, you can choose which commits to squash, which is useful for cleaning up your project history before merging it into the main branch.

This technique is often used to maintain a more concise and understandable commit history.

## **70. What does git revert do?**

a) Creates a new commit that undoes the changes of a previous commit

b) Deletes the most recent commit

c) Resets the repository to a specific commit

d) Moves the HEAD pointer to the previous commit

### **Answer:**

a) Creates a new commit that undoes the changes of a previous commit

### **Explanation:**

The git revert command creates a new commit that undoes the changes introduced by a previous commit. It leaves the commit history intact, unlike git reset, which modifies the history.

For example, running git revert abc123 will create a new commit that reverts the changes made in the commit with hash abc123.

This command is ideal when you need to undo changes but still want to preserve the commit history for future reference.

## **71. What does the command git checkout -b branchname do?**

a) Creates and switches to a new branch

b) Merges a branch

c) Deletes a branch

d) Checks out the latest commit on the main branch

### **Answer:**

a) Creates and switches to a new branch

### **Explanation:**

The git checkout -b branchname command creates a new branch with the specified name and immediately switches to it. This is a shortcut for creating a branch with git branch branchname followed by git checkout branchname.

For example, git checkout -b feature-branch creates and switches to a branch named feature-branch.

## **72. How do you rename a Git remote?**

a) git remote rename oldname newname

b) git rename remote oldname newname

c) git remote move oldname newname

d) git rename remote newname

### **Answer:**

a) git remote rename oldname newname

### **Explanation:**

The git remote rename oldname newname command renames a remote in your repository. For example, if you want to rename origin to upstream, you would run git remote rename origin upstream.

This command is useful when you need to change the name of the remote without affecting its configuration or URL.

Renaming remotes can help better organize multiple remote repositories, especially in collaborative projects.

## **73. What does the git fetch --prune command do?**

a) Removes references to branches that no longer exist on the remote

b) Deletes all branches on the remote

c) Fetches all branches from the remote

d) Updates the current branch to match the remote

### **Answer:**

a) Removes references to branches that no longer exist on the remote

### **Explanation:**

The git fetch --prune command removes local references to remote branches that have been deleted from the remote repository. It ensures that your local representation of the remote branches is up-to-date and free of stale references.

This command is useful for cleaning up local branches that no longer exist on the remote and maintaining a clean repository.

Using git fetch --prune regularly ensures that your local repository reflects the state of the remote repository accurately.

## **74. What does the git diff command show?**

a) The differences between the working directory and the staging area

b) The commit history

c) The list of branches

d) The differences between two commits

### **Answer:**

a) The differences between the working directory and the staging area

### **Explanation:**

The git diff command shows the differences between the working directory and the staging area. It helps you see the changes that have been made but not yet staged for commit.

For example, if you've edited files but haven't run git add yet, git diff will display the modifications you've made.

This command is essential for reviewing changes before staging or committing them to ensure only the intended changes are included.

## **75. How do you reset your working directory to the last commit in Git?**

a) git reset --hard

b) git revert HEAD

c) git checkout --last

d) git reset HEAD

### **Answer:**

a) git reset --hard

### **Explanation:**

The git reset --hard command resets your working directory, index (staging area), and HEAD pointer to the state of the last commit. It discards all changes in the working directory and the staging area, effectively reverting the repository to the last committed state.

This command is useful when you want to discard all uncommitted changes and return to a clean working directory.

Be cautious when using git reset --hard as it permanently deletes uncommitted changes.

## **76. How do you check out a specific commit in Git?**

a) git checkout commit-hash

b) git revert commit-hash

c) git checkout branch-name

d) git pull commit-hash

### **Answer:**

a) git checkout commit-hash

### **Explanation:**

The git checkout commit-hash command checks out a specific commit by its hash. This places your repository in a detached HEAD state, allowing you to view or work with the project at the state of that commit.

For example, running git checkout abc123 will check out the commit with hash abc123.

Be aware that if you make new commits in this state, they won’t be attached to any branch unless you create a new one.

## **77. What does git blame display?**

a) Shows who made changes to each line of a file

b) Shows a list of uncommitted changes

c) Shows the commit history

d) Shows all remote branches

### **Answer:**

a) Shows who made changes to each line of a file

### **Explanation:**

The git blame command shows who last modified each line of a file, along with the commit hash and author details. It’s useful for tracing the origin of a particular change, especially when debugging or reviewing code history.

For example, running git blame filename will display a breakdown of each line of the file, with information about the commit that last modified it.

This command is crucial for understanding the history of changes made to specific files in a project.

## **78. What does the git describe command do?**

a) Describes the most recent tag reachable from a commit

b) Shows the description of the repository

c) Adds a description to a commit

d) Lists the differences between commits

### **Answer:**

a) Describes the most recent tag reachable from a commit

### **Explanation:**

The git describe command provides a human-readable description of a commit by finding the most recent tag that is reachable from the commit. It uses the tag, the number of commits since the tag, and a shortened commit hash to describe the commit.

For example, running git describe might return v1.2-3-gabc1234, meaning the commit is three commits after tag v1.2, and its short hash is gabc1234.

This command is useful for summarizing the state of the project at a particular commit.

## **79. How do you initialize a new Git repository?**

a) git init

b) git start

c) git new

d) git create

### **Answer:**

a) git init

### **Explanation:**

The git init command initializes a new Git repository in the current directory. It creates a hidden .git folder that contains all the metadata and version control history for the project.

This is the first step when starting a new project with Git, as it allows you to begin tracking changes and commits in your project.

Once a repository is initialized, you can start adding files and making commits to track your project history.

## **80. What does the git submodule command do?**

a) Manages external repositories within a Git repository

b) Creates branches for external modules

c) Merges multiple repositories into one

d) Deletes external modules

### **Answer:**

a) Manages external repositories within a Git repository

### **Explanation:**

The git submodule command allows you to manage external repositories within a Git project. It adds other repositories as submodules, enabling you to track and update external code dependencies separately from your main project.

For example, using git submodule add URL adds an external repository as a submodule to your project.

Submodules are useful for projects that rely on external libraries or components that need to be versioned independently.

## **81. What does the git show-ref command do?**

a) Lists references (branches or tags) in a Git repository

b) Displays detailed information about a commit

c) Shows the current status of the working directory

d) Displays a summary of the commit history

### **Answer:**

a) Lists references (branches or tags) in a Git repository

### **Explanation:**

The git show-ref command lists references in a Git repository, such as branches or tags. It shows the reference names along with their corresponding commit hashes, making it useful for tracking references to specific commits.

This command helps you see all the reference points (tags, branches) in your repository and their associated commits.

It’s particularly helpful for identifying where different branches or tags point to in the repository history.

## **82. How do you create an alias for a Git command?**

a) git config --global alias.alias-name 'git-command'

b) git alias create alias-name

c) git alias git-command alias-name

d) git create alias alias-name

### **Answer:**

a) git config --global alias.alias-name 'git-command'

### **Explanation:**

The git config --global alias.alias-name 'git-command' command creates an alias for a Git command, allowing you to use a shorter or custom command instead of the full command.

For example, you can create an alias for git status by running git config --global alias.st 'status', and then use git st instead of typing git status.

Using aliases can make frequent or complex commands easier to remember and faster to execute.

## **83. How do you ignore a file in Git without deleting it?**

a) Add the file to .gitignore

b) Use git rm --cached filename

c) Delete the file locally

d) Use git reset filename

### **Answer:**

a) Add the file to .gitignore

### **Explanation:**

To ignore a file in Git without deleting it, you can add the file to the .gitignore file. This prevents Git from tracking changes to the file in future commits but leaves the file intact in your working directory.

For example, adding config.php to .gitignore will prevent Git from tracking changes to config.php, while keeping it available for use locally.

This is useful for ignoring sensitive files or configuration files that should not be committed to the repository.

## **84. What does git push --force do?**

a) Forces the remote repository to accept changes, overwriting conflicts

b) Deletes the remote branch

c) Merges a remote branch with the local branch

d) Pushes tags to the remote repository

### **Answer:**

a) Forces the remote repository to accept changes, overwriting conflicts

### **Explanation:**

The git push --force command forces the remote repository to accept your changes, even if it would overwrite commits on the remote branch. This is often used after rebasing or resetting a branch locally.

For example, after using git reset to modify the commit history, you may need to use git push --force to push those changes to the remote repository, overwriting the previous history.

Be cautious when using --force as it can overwrite others' work on shared branches.

## **85. How do you delete a remote branch in Git?**

a) git push origin --delete branch-name

b) git remove branch-name

c) git branch -d branch-name

d) git reset origin branch-name

### **Answer:**

a) git push origin --delete branch-name

### **Explanation:**

To delete a remote branch in Git, you use the git push origin --delete branch-name command. This removes the specified branch from the remote repository (e.g., GitHub).

For example, running git push origin --delete feature-branch will delete the feature-branch from the remote repository.

Deleting remote branches is useful for cleaning up after merging a feature branch or when a branch is no longer needed.

## **86. What does the git config --list command show?**

a) Displays all configuration settings

b) Shows the commit history

c) Lists all local branches

d) Displays the differences between commits

### **Answer:**

a) Displays all configuration settings

### **Explanation:**

The git config --list command displays all Git configuration settings for the repository, including user information, aliases, remote repository URLs, and other settings defined in the Git configuration files.

For example, running git config --list will show settings like your user email, the default branch name, and any custom aliases you’ve set up.

This command is useful for reviewing and troubleshooting Git configuration issues.

## **87. What does git reset --soft do?**

a) Moves the HEAD pointer to a specific commit but keeps changes in the working directory

b) Deletes uncommitted changes

c) Deletes the commit history

d) Stashes changes

### **Answer:**

a) Moves the HEAD pointer to a specific commit but keeps changes in the working directory

### **Explanation:**

The git reset --soft command moves the HEAD pointer to a specific commit but leaves changes in the index (staging area) and working directory untouched. This allows you to re-commit changes with a new commit message or make further modifications.

It’s useful when you want to amend a commit without losing any changes in your working directory or staging area.

## **88. How do you view the commit history of a repository in a graph format?**

a) git log --graph

b) git graph

c) git log --history

d) git show --graph

### **Answer:**

a) git log --graph

### **Explanation:**

The git log --graph command displays the commit history in a graphical representation, showing branch structures and merges. It helps visualize the relationships between commits and branches.

This command is useful for reviewing the project’s commit history and understanding how branches have diverged and merged over time.

## **89. What does the git cherry-pick command do?**

a) Applies a specific commit from one branch to another

b) Deletes a commit from the history

c) Resets the current branch to a specific commit

d) Combines multiple branches

### **Answer:**

a) Applies a specific commit from one branch to another

### **Explanation:**

The git cherry-pick command applies a specific commit from one branch to another without merging the entire branch. This is useful when you want to bring specific changes (commits) into your branch without pulling in unrelated work.

For example, git cherry-pick abc123 will apply the commit with hash abc123 to the current branch.

This command is useful for selectively incorporating changes from other branches.

## **90. How do you configure Git to use a specific text editor?**

a) git config --global core.editor "editor-name"

b) git set editor editor-name

c) git edit config editor-name

d) git commit --editor editor-name

### **Answer:**

a) git config --global core.editor "editor-name"

### **Explanation:**

You can configure Git to use a specific text editor by running the command git config --global core.editor "editor-name". This sets the default text editor for Git commands like git commit that open an editor.

For example, running git config --global core.editor "vim" sets Vim as the default text editor for Git.

This command ensures that Git uses the editor of your choice for editing commit messages and other text files.

## **91. How do you undo the last local commit without removing the changes?**

a) git reset --soft HEAD^

b) git revert HEAD^

c) git reset --hard HEAD^

d) git checkout --last

### **Answer:**

a) git reset --soft HEAD^

### **Explanation:**

The git reset --soft HEAD^ command undoes the last commit while keeping the changes in the staging area. This allows you to modify the commit or make additional changes before committing again.

For example, running this command moves the HEAD pointer back to the previous commit but leaves all changes intact.

This is useful when you want to amend or adjust your last commit without losing any work.

## **92. What does the git stash pop command do?**

a) Applies the latest stashed changes and removes them from the stash list

b) Removes all stashed changes

c) Displays the list of stashes

d) Saves uncommitted changes without removing them

### **Answer:**

a) Applies the latest stashed changes and removes them from the stash list

### **Explanation:**

The git stash pop command applies the latest stashed changes to your working directory and removes them from the stash list. It’s similar to git stash apply but with the added step of removing the stash once applied.

This is useful when you want to retrieve stashed changes and no longer need to keep them in the stash list.

Be cautious, as any conflicts when applying the stash will need to be resolved manually.

## **93. What does the git gc command do?**

a) Optimizes the Git repository by cleaning up unnecessary files and compressing objects

b) Generates a commit graph

c) Commits all changes automatically

d) Deletes all untracked files

### **Answer:**

a) Optimizes the Git repository by cleaning up unnecessary files and compressing objects

### **Explanation:**

The git gc command performs garbage collection in a Git repository, optimizing it by cleaning up unnecessary files and compressing loose objects. It helps reduce the size of the repository and improves performance.

For example, Git might run git gc automatically after a certain number of commits, but you can also run it manually to clean up the repository.

This is important for maintaining a healthy repository, especially after many changes or merges.

## **94. How do you squash commits during a rebase?**

a) Use git rebase -i and mark the commits you want to squash as "squash"

b) Use git squash after rebasing

c) Use git merge --squash

d) Use git squash rebase

### **Answer:**

a) Use git rebase -i and mark the commits you want to squash as "squash"

### **Explanation:**

To squash commits during a rebase, you use the git rebase -i (interactive rebase) command and mark the commits you want to combine with "squash." This combines multiple commits into one, keeping the commit history cleaner.

During the interactive rebase process, you’ll be prompted to squash or reword commits as necessary.

Squashing commits is helpful for simplifying commit history before merging branches.

## **95. How do you create a new branch from an existing branch in Git?**

a) git checkout -b new-branch

b) git branch create new-branch

c) git switch --create new-branch

d) git merge --create new-branch

### **Answer:**

a) git checkout -b new-branch

### **Explanation:**

The git checkout -b new-branch command creates a new branch and immediately switches to it. This is a commonly used command when you want to start work on a new feature or task.

This command is equivalent to running git branch new-branch followed by git checkout new-branch in a single step.

Creating new branches is essential for isolating work on features, bug fixes, or experiments.

## **96. How do you fetch all remote branches without merging them into your local branches?**

a) git fetch

b) git pull --no-merge

c) git fetch --merge

d) git clone --all

### **Answer:**

a) git fetch

### **Explanation:**

The git fetch command retrieves all remote branches, tags, and commits without merging them into your local branches. It updates your local repository with new information from the remote repository but doesn’t affect your working directory.

For example, running git fetch will download all changes from the remote, allowing you to review or merge them later.

This command is important for synchronizing your local repository with the remote without making any immediate changes to your local branches.

## **97. What does git reset --hard HEAD^ do?**

a) Resets the current branch to the previous commit and deletes all uncommitted changes

b) Deletes the last commit but keeps the changes in the working directory

c) Reverts the last commit and stages the changes

d) Moves the HEAD pointer to the first commit

### **Answer:**

a) Resets the current branch to the previous commit and deletes all uncommitted changes

### **Explanation:**

The git reset --hard HEAD^ command resets the current branch to the previous commit and deletes all uncommitted changes in both the working directory and staging area. It’s a forceful reset that discards all local changes.

This is useful when you want to completely undo the last commit and any modifications in the working directory, returning to a clean state.

Be careful when using this command, as it permanently deletes changes that aren’t committed.

## **98. How do you view the commit history with only the commit messages?**

a) git log --oneline

b) git history --short

c) git log --brief

d) git log --message

### **Answer:**

a) git log --oneline

### **Explanation:**

The git log --oneline command shows the commit history in a compact format, displaying only the commit hash and commit message on a single line for each commit. This makes it easier to quickly browse through the commit history.

For example, running git log --oneline gives a concise overview of the project’s commit history without overwhelming detail.

This command is helpful for reviewing commit messages and identifying relevant commits at a glance.

## **99. What does git clone do?**

a) Copies a remote Git repository to your local machine

b) Deletes a remote repository

c) Merges a remote repository into the current branch

d) Pushes changes to the remote repository

### **Answer:**

a) Copies a remote Git repository to your local machine

### **Explanation:**

The git clone command creates a copy of a remote Git repository on your local machine. It downloads the repository’s history, branches, and files, allowing you to work on the project locally.

For example, running git clone https://github.com/user/repo.git will download the repository located at that URL to your machine.

This is the first step when you want to contribute to an existing Git project.

## **100. What does git config --global user.name do?**

a) Sets the user’s name for all Git repositories on the system

b) Displays the current user name

c) Adds a username to the local repository

d) Configures the remote repository username

### **Answer:**

a) Sets the user’s name for all Git repositories on the system

### **Explanation:**

The git config --global user.name command sets the user’s name for all Git repositories on the system. This name will be associated with your commits and is used to identify you as the author of those commits.

For example, running git config --global user.name "John Doe" sets your name globally, so it will appear in the author field of all future commits.

Setting the username is a crucial step in configuring Git to ensure that your commits are properly attributed to you.

**Docker**

**1. Which of the following view displays all of your containers and apps in real-time in Docker?**

1. Container
2. Hub
3. Images

**Answer:** A) Container

**Explanation:**

The Containers view displays all of your containers and apps in real-time.

**2. \_\_\_\_ is a cloud registry service that allows you to obtain Docker images created by other communities.**

1. Container
2. Hub
3. Images

**Answer:** B) Hub

**Explanation:**

Docker Hub is a cloud registry service that allows you to obtain Docker images created by other communities.

**3. Which of the following command displays all of the pictures that are presently installed on the system?**

1. See images
2. Docker see images
3. Docker images
4. Docker pictures

**Answer:** C) Docker images

**Explanation:**

Docker Images command displays all of the pictures that are presently installed on the system.

**4. A Container \_\_\_\_.**

1. Can be run on local machines, virtual machines, or deployed to the cloud.
2. Is portable
3. Is a runnable instance of an image
4. All of the above

**Answer:** D) All of the above

**Explanation:**

Container is a:

* Can be run on local machines, virtual machines, or deployed to the cloud.
* Is portable
* Is a runnable instance of an image

**5. Which of the following command can run the application in a container?**

1. Docker run
2. Run
3. Docker start
4. Start

**Answer:** A) Docker run

**Explanation:**

The docker run command can run the application in a container.

**6. Which of the following command you will use to list your containers?**

1. Docker list
2. Docker Ps
3. List
4. Docker\_container\_list

**Answer:** B) Docker Ps

**Explanation:**

Docker ps command will be used to list your containers.

**7. Which of the following statement is correct?**

1. To remove a container, you first need to stop it
2. You can directly remove a container, without stopping it.

**Answer:** A) To remove a container, you first need to stop it

**Explanation:**

Statement (A) is correct, to remove a container, you first need to stop it. Once it has stopped, you can remove it.

**8. To stop the container, which of the following command is used?**

1. Stop
2. Docker end
3. Docker stop
4. Docker finish

**Answer:** C) Docker stop

**Explanation:**

To stop the container, use the docker stop command.

**9. Once the container has stopped, which of the following command you will use to remove a container?**

1. Docker remove
2. Docker Destroy
3. Docker rm
4. Docker del

**Answer:** C) Docker rm

**Explanation:**

When the container has stopped, use the docker rm command to remove it.

**10. Which of the following is the default registry in docker?**

1. Docker images
2. Docker hub
3. Docker container

**Answer:** B) Docker hub

**Explanation:**

Docker hub is the default registry.

**11. \_\_\_\_ allows you to connect specified filesystem paths of the container back to the host computer.**

1. Volumes
2. Images
3. Containers

**Answer:** A) Volumes

**Explanation:**

Volumes allow you to connect specified filesystem paths of the container back to the host computer.

**12. How many types of volumes are there in Docker?**

1. 3
2. 4
3. 5
4. 2

**Answer:** D) 2

**Explanation:**

There are two types of volumes in docker:

* Named volumes
* Bind Mounts

**13. The \_\_\_\_ is the actual location on the disk where the data is stored.**

1. Disk point
2. Container memory
3. Mountpoint

**Answer:** C) Mountpoint

**Explanation:**

The mountpoint is the actual location on the disk where the data is stored.

**14. Which of the following volume type allows you to share a directory from the host's filesystem into the container?**

1. Named volumes
2. Bind Mounts

**Answer:** B) Bind Mounts

**Explanation:**

The bind mount volume type allows you to share a directory from the host's filesystem into the container.

**15. In which of the following volume type docker chooses the host location?**

1. Named volumes
2. Bind Mounts

**Answer:** A) Named volumes

**Explanation:**

In the Named volume type docker chooses the host location, whereas you decide the host location in the bind mount type of volume.

**16. Which of the following is a tool that was built to assist define and distribute multi-container applications?**

1. Docker setup
2. Docker compose
3. Docker notify

**Answer:** B) Docker compose

**Explanation:**

Docker Compose is a tool that was built to assist define and distribute multi-container applications.

**17. You can examine the command that was used to construct each layer within an image by using which of the following command?**

1. docker image layer
2. docker image
3. docker image history
4. docker image history layer

**Answer:** C) docker image history

**Explanation:**

You can examine the command that was used to construct each layer within an image by using the docker image history command.

**18. Which of the following command is used to display the statistics of a running container?**

1. Docker statistics
2. Stats
3. Docker statics
4. Docker stats

**Answer:** D) Docker stats

**Explanation:**

The stats command is used to display the statistics of a running container.

**19. Which of the following command is used to pause processes in a running container?**

1. Docker hold
2. Docker halt
3. Docker wait
4. Docker pause

**Answer:** D) Docker pause

**Explanation:**

Docker pause command is used to pause processes in a running container.

**20. Which command is used to terminate processes in a running container?**

1. Docker kill
2. Docker terminate
3. Docker Suspend
4. Docker delete

**Answer:** A) Docker kill

**Explanation:**

The docker kill command is used to terminate processes in a running container.

**21. Which command is used to terminate the Docker daemon process?**

1. Service kill
2. Docker kill
3. service docker stop
4. service docker Terminate

**Answer:** C) service docker stop

**Explanation:**

service docker stop command is used to terminate the Docker daemon process.

**22. Daemon level of logging has how many levels?**

1. 2
2. 3
3. 4
4. 5

**Answer:** C) 4

**Explanation:**

Daemon level of logging has 4 levels:

* Debug
* info
* Error
* Fatal

**23. How many components are there in Docker?**

1. 3
2. 4
3. 5
4. 6

**Answer:** A) 3

**Explanation:**

Docker has three components:

* Docker client
* Docker Host
* Docker Registry

**24. Which of the following component stores the docker images?**

1. Docker client
2. Docker Host
3. Docker Registry

**Answer:** C) Docker Registry

**Explanation:**

Docker registry component stores the docker images.

**25. Among named volumes and bind mounts which is easier to back up or migrate?**

1. Named volumes (volume)
2. Bind mounts

**Answer:** A) Named volumes (volume)

**Explanation:**

Volumes are easier to back up or migrate than bind mounts.

**26. To save up space and eliminate any unnecessary volumes which of the command is used?**

1. docker volume delete
2. docker volume remove\_all
3. docker volume rm
4. docker volume prune

**Answer:** D) docker volume prune

**Explanation:**

docker volume prune command is used to save up space and eliminate any unnecessary volumes.

**27. Which of the following networks are typically used when your apps operate in separate containers and need to interact with one another?**

1. Bridge
2. Host
3. Overlay
4. ipvlan

**Answer:** A) Bridge

**Explanation:**

Bridge networks are typically used when your apps operate in separate containers and need to interact with one another.

**28. Which of the following is the default network driver?**

1. Bridge
2. Host
3. Overlay
4. ipvlan

**Answer:** A) Bridge

**Explanation:**

Bridge is the default network driver.

**29. Which of the following networks are ideal when containers operating on various Docker hosts need to connect, or when numerous apps use swarm services to collaborate?**

1. Bridge
2. Host
3. Overlay
4. ipvlan

**Answer:** C) Overlay

**Explanation:**

Overlay networks are ideal when containers operating on various Docker hosts need to connect, or when numerous apps use swarm services to collaborate.

**30. Which of the following network driver gives users total control over both IPv4 and IPv6 addresses?**

1. Bridge
2. Host
3. Overlay
4. Ipvlan

**Answer:** D) Ipvlan

**Explanation:**

The IPvlan driver gives users total control over both IPv4 and IPv6 addressing.

**31. Which of the following concatenates a collection of strings into a single string and inserts a divider between each list item?**

1. Append
2. Join
3. Merge
4. Concat

**Answer:** B) Join

**Explanation:**

Join concatenates a collection of strings into a single string and inserts a divider between each list item.

**32. Each value is printed on a new line using \_\_\_\_.**

1. Print\_new
2. Print
3. Prinline
4. println

**Answer:** D) println

**Explanation:**

Each value is printed on a new line using println.

**33. A \_\_\_\_ is a string-based key-value pair?**

1. Map
2. Set
3. Label
4. List

**Answer:** C) Label

**Explanation:**

A label is a string-based key-value pair.

**34. Label keys should begin and end with a \_\_\_\_ letter.**

1. lower-case
2. upper-case
3. any case

**Answer:** A) lower-case

**Explanation:**

Label keys should begin and end with a lowercase letter.

**35. Docker provides how many modes for delivering messages from the container to the log driver?**

1. 2
2. 3
3. 4
4. 5

**Answer:** A) 2

**Explanation:**

Docker provides two modes for delivering messages from the container to the log driver:

* (default) direct, blocking delivery
* non-blocking delivery

**36. To list all the networks linked with Docker on the host, the \_\_\_\_ command is used.**

1. docker network list
2. docker network ls
3. docker ls
4. network ls

**Answer:** B) docker network ls

**Explanation:**

To list all the networks linked with Docker on the host, the docker network ls command is used.

**37. The \_\_\_\_ command is used to obtain all client and server version information.**

1. Version
2. Info
3. docker version

**Answer:** C) docker version

**Explanation:**

The docker version command is used to obtain all client and server version information.

**38. Containerization is an abstract version of an application.**

1. True
2. False

**Answer:** A) True

**Explanation:**

Containerization is an abstract version of an application.

**39. A \_\_\_\_ is a piece of software that enables virtualization.**

1. Emulator
2. Hypervisor
3. Invertor

**Answer:** B) Hypervisor

**Explanation:**

A hypervisor is a piece of software that enables virtualization.

**40. \_\_\_\_ is an abstract version of a real computer.**

1. Virtualization
2. containerization

**Answer:** A) Virtualization

**Explanation:**

virtualization is an abstract version of a real computer.

**41. How many types of hypervisors are there?**

1. 5
2. 4
3. 3
4. 2

**Answer:** D) 2

**Explanation:**

There are two types of hypervisors:

* Native hypervisor
* Hosted hypervisor

**42. Docker is a platform for \_\_\_\_.**

1. Developing an application
2. Shipping an application
3. Running an application
4. All of the above

**Answer:** D) All of the above

**Explanation:**

Docker is a platform for :

* Developing an application
* Shipping an application
* Running an application

**43. Which of the following architecture does Docker uses?**

1. Client-server
2. Master-slave
3. Layered

**Answer:** A) Client-server

**Explanation:**

Docker uses a client-server architecture.

**44. Docker is written in which of the following programming language?**

1. Ruby
2. Go
3. Kotlin
4. Swift

**Answer:** B) Go

**Explanation:**

Docker is written in the Go programming language.

**Kubernetes**

1. What is Kubernetes primarily used for?

a) Managing databases

b) Orchestrating containerized applications

c) Running virtual machines

d) Hosting static websites

Answer:

b) Orchestrating containerized applications

Explanation:

Kubernetes is an open-source platform that automates the management, scaling, and deployment of containerized applications across clusters of machines.

2. What is a Kubernetes Pod?

a) A virtual machine

b) The smallest deployable unit in Kubernetes

c) A network protocol

d) A configuration file

Answer:

b) The smallest deployable unit in Kubernetes

Explanation:

A Pod is the smallest and simplest Kubernetes object that represents a single instance of a running process in a cluster, and can contain one or more containers.

3. Which component of Kubernetes is responsible for managing the state of the cluster?

a) Kubelet

b) etcd

c) Kube-scheduler

d) Kube-apiserver

Answer:

d) Kube-apiserver

Explanation:

The Kube-apiserver is the control plane component that exposes the Kubernetes API and manages the state of the Kubernetes cluster.

4. What is etcd used for in Kubernetes?

a) Storage for container images

b) Storing cluster data

c) Load balancing

d) Monitoring logs

Answer:

b) Storing cluster data

Explanation:

etcd is a distributed key-value store that stores all the cluster data for Kubernetes, including configuration and state information.

5. What is a Kubernetes Service?

a) A Docker container

b) A static IP address for external access

c) An abstraction that defines a logical set of Pods and a policy to access them

d) A load balancer for physical servers

Answer:

c) An abstraction that defines a logical set of Pods and a policy to access them

Explanation:

A Kubernetes Service defines a logical set of Pods and a network policy for accessing them. It ensures that your application is accessible consistently even as Pods are dynamically created and destroyed.

6. What is the role of the Kube-scheduler in Kubernetes?

a) Schedules tasks on nodes

b) Monitors application performance

c) Configures networking between Pods

d) Stores application logs

Answer:

a) Schedules tasks on nodes

Explanation:

The Kube-scheduler is responsible for assigning Pods to available nodes in the Kubernetes cluster based on resource availability and other constraints.

7. What is a ReplicaSet in Kubernetes?

a) A tool to clone Pods

b) Ensures a specified number of Pod replicas are running

c) A backup service for nodes

d) A command to restart Kubernetes

Answer:

b) Ensures a specified number of Pod replicas are running

Explanation:

A ReplicaSet ensures that a specified number of identical Pods are always running in a Kubernetes cluster. If a Pod crashes, the ReplicaSet will create a new Pod to replace it.

8. Which command is used to deploy an application in Kubernetes?

a) kubectl create deployment

b) kubectl deploy app

c) kubectl deploy pod

d) kubectl build app

Answer:

a) kubectl create deployment

Explanation:

The kubectl create deployment command is used to create a new deployment for an application in Kubernetes.

9. What is the purpose of a StatefulSet in Kubernetes?

a) To create stateless applications

b) To manage stateful applications

c) To store logs of the cluster

d) To monitor network traffic

Answer:

b) To manage stateful applications

Explanation:

A StatefulSet is used in Kubernetes to manage stateful applications, such as databases, where each Pod needs a persistent identity and storage.

10. How do you scale a deployment in Kubernetes?

a) kubectl scale deployment <deployment-name> --replicas=<number>

b) kubectl create scale <deployment-name>

c) kubectl replicas set --scale <number>

d) kubectl deploy replicas <number>

Answer:

a) kubectl scale deployment <deployment-name> --replicas=<number>

Explanation:

To scale a Kubernetes deployment, you use the kubectl scale deployment command followed by the desired number of replicas.

11. What is the role of Kubelet in Kubernetes?

a) To manage containers on a node

b) To store configuration files

c) To monitor network traffic

d) To schedule Pods

Answer:

a) To manage containers on a node

Explanation:

The Kubelet is an agent that runs on each node in the Kubernetes cluster. It ensures that containers are running in a Pod as expected.

12. Which Kubernetes component ensures communication between different nodes in a cluster?

a) Kube-proxy

b) Kube-scheduler

c) Kube-apiserver

d) etcd

Answer:

a) Kube-proxy

Explanation:

Kube-proxy is a network component in Kubernetes that ensures that services can communicate with each other across different nodes within the cluster.

13. What is a DaemonSet in Kubernetes?

a) A type of database for storing logs

b) A deployment strategy

c) Ensures that a copy of a Pod runs on all nodes

d) A way to create stateful applications

Answer:

c) Ensures that a copy of a Pod runs on all nodes

Explanation:

A DaemonSet ensures that a copy of a specific Pod is running on all (or some) nodes in a Kubernetes cluster.

14. What is Helm in the Kubernetes ecosystem?

a) A tool for scaling Pods

b) A package manager for Kubernetes applications

c) A logging tool

d) A Kubernetes API wrapper

Answer:

b) A package manager for Kubernetes applications

Explanation:

Helm is a package manager for Kubernetes, which simplifies the deployment and management of applications through reusable "Helm charts."

15. What is the primary purpose of a Kubernetes Ingress?

a) To scale applications

b) To provide external access to services

c) To manage storage

d) To deploy applications

Answer:

b) To provide external access to services

Explanation:

An Ingress is an API object in Kubernetes that manages external access to services, typically HTTP and HTTPS routes.

16. How can you expose a Kubernetes deployment to external traffic?

a) By using a DaemonSet

b) By creating a Service of type LoadBalancer or NodePort

c) By creating a StatefulSet

d) By using etcd

Answer:

b) By creating a Service of type LoadBalancer or NodePort

Explanation:

You can expose a Kubernetes deployment to external traffic by creating a Service of type LoadBalancer or NodePort to allow access from outside the cluster.

17. What is a PersistentVolume (PV) in Kubernetes?

a) A network protocol

b) A storage resource provisioned by the administrator

c) A type of Service

d) A way to scale containers

Answer:

b) A storage resource provisioned by the administrator

Explanation:

A PersistentVolume (PV) is a piece of storage in a Kubernetes cluster that is provisioned by the administrator for use by Pods that require persistent storage.

18. What does the kubectl command do in Kubernetes?

a) Automates testing

b) It is the command-line interface to communicate with Kubernetes clusters

c) Monitors application performance

d) Handles backups of the cluster

Answer:

b) It is the command-line interface to communicate with Kubernetes clusters

Explanation:

kubectl is the command-line tool that allows users to interact with Kubernetes clusters, performing tasks like deploying applications and managing resources.

19. What is the main purpose of a ConfigMap in Kubernetes?

a) To store application logs

b) To store non-confidential configuration data

c) To manage container images

d) To update the Kubernetes API

Answer:

b) To store non-confidential configuration data

Explanation:

A ConfigMap in Kubernetes is used to store non-confidential configuration data such as environment variables, command-line arguments, or configuration files.

20. What is the purpose of Kubernetes Secrets?

a) To store encrypted credentials and sensitive information

b) To track changes to configurations

c) To define network policies

d) To backup cluster data

Answer:

a) To store encrypted credentials and sensitive information

Explanation:

Secrets in Kubernetes are used to store sensitive information such as passwords, OAuth tokens, and SSH keys, ensuring they are stored securely and used by Pods when required.

21. Which Kubernetes object is used to automatically scale Pods based on resource utilization?

a) ReplicaSet

b) Horizontal Pod Autoscaler (HPA)

c) StatefulSet

d) DaemonSet

Answer:

b) Horizontal Pod Autoscaler (HPA)

Explanation:

The Horizontal Pod Autoscaler (HPA) automatically scales the number of Pods in a deployment based on observed CPU utilization or other select metrics.

22. What is a Node in Kubernetes?

a) A container image

b) A server or VM that runs applications in containers

c) A network switch

d) A load balancer

Answer:

b) A server or VM that runs applications in containers

Explanation:

A Node in Kubernetes is a worker machine, either a physical server or virtual machine, where Pods are deployed and run.

23. What is the purpose of kubeadm in Kubernetes?

a) To schedule Pods

b) To provide a simple way to set up a Kubernetes cluster

c) To manage cloud resources

d) To store cluster logs

Answer:

b) To provide a simple way to set up a Kubernetes cluster

Explanation:

kubeadm is a tool that helps users to easily set up a Kubernetes cluster by automating the initialization process and managing configurations.

24. What is the purpose of Kubernetes Labels?

a) To store metadata in key-value pairs for objects

b) To generate Pod logs

c) To define storage classes

d) To create configuration files

Answer:

a) To store metadata in key-value pairs for objects

Explanation:

Labels are key-value pairs attached to objects such as Pods or Services, and are used for identifying and selecting objects within the cluster based on their metadata.

25. What is a Kubernetes Namespace?

a) A physical node in a cluster

b) A virtual cluster within a physical cluster

c) A load balancer for services

d) A monitoring tool

Answer:

b) A virtual cluster within a physical cluster

Explanation:

A Namespace is a virtual cluster within a Kubernetes cluster that helps organize and manage resources, such as Pods and Services, by separating them into logical groups.

These questions cover various concepts and components within Kubernetes, helping you understand how Kubernetes operates and its key functionalities. Keep practicing to strengthen your knowledge of Kubernetes and container orchestration.

**Jenkins**

**1. What is Jenkins?**

a. A continuous integration and continuous delivery tool  
 b. A software development platform  
 c. A project management tool  
 d. A bug tracking tool

**Answer: a**

**Explanation:** Jenkins is a popular open-source automation server that helps in continuous integration and continuous delivery (CI/CD) of software.

**2. Which programming language is used to write Jenkins?**

a. Java  
 b. Python  
 c. Ruby  
 d. JavaScript

**Answer: a**

**Explanation:** Jenkins is written in Java and is cross-platform, meaning it can run on any operating system that supports Java.

**3. What is the purpose of Jenkins?**

a. To automate the building, testing, and deployment of software  
 b. To manage project timelines and resources  
 c. To track and manage bugs  
 d. To provide a platform for software development

**Answer: a**

**Explanation:** The main purpose of Jenkins is to automate the building, testing, and deployment of software, thus facilitating continuous integration and delivery.

**4. Which of the following is not a core component of Jenkins?**

a. Jenkins Master  
 b. Jenkins Slave  
 c. Jenkins Node  
 d. Jenkins Client

**Answer: d**

**Explanation:** Jenkins Client is not a core component of Jenkins. The core components of Jenkins include Jenkins Master, Jenkins Slave, and Jenkins Node.

**5. What is a Jenkinsfile?**

a. A file that defines the build process in Jenkins  
 b. A file that contains the code for a Jenkins plugin  
 c. A file that contains the Jenkins configuration  
 d. A file that contains the results of a Jenkins build

**Answer: a**

**Explanation:** A Jenkinsfile is a text file that defines the build process in Jenkins. It is written in Groovy and is used to automate the pipeline.

**6. What is a pipeline in Jenkins?**

a. A set of plugins that provide additional functionality to Jenkins  
 b. A series of automated steps that build, test, and deploy software  
 c. A graphical user interface for managing Jenkins  
 d. A feature that allows users to create and manage Jenkins jobs

**Answer: b**

**Explanation:** A pipeline in Jenkins is a series of automated steps that build, test, and deploy software. It is created using a Jenkinsfile and can be visualized using the Blue Ocean plugin.

**7. What is a Jenkins plugin?**

a. A set of scripts that automate the building of software  
 b. A piece of software that provides additional functionality to Jenkins  
 c. A configuration file that defines the build process in Jenkins  
 d. A report generated by Jenkins after a build is complete

**Answer: b**

**Explanation:** A Jenkins plugin is a piece of software that provides additional functionality to Jenkins. There are thousands of plugins available for Jenkins, including plugins for source control, build tools, and notification systems.

**8. What is the difference between Jenkins Master and Jenkins Slave?**

a. Jenkins Master is the central server, while Jenkins Slave is a remote agent  
 b. Jenkins Master is responsible for building the software, while Jenkins Slave is responsible for testing it  
 c. Jenkins Master is a component of Jenkins, while Jenkins Slave is a separate tool  
 d. Jenkins Master is used for local builds, while Jenkins Slave is used for cloud-based builds

**Answer: a**

**Explanation:** Jenkins Master is the central server that manages and schedules builds, while Jenkins Slave is a remote agent that performs the actual building and testing of the software.

**9. What is the role of Jenkins Node?**

a. To provide a graphical user interface for Jenkins  
 b. To manage the distribution of work to Jenkins Slaves  
 c. To store the configuration data for Jenkins  
 d. To manage the build process in Jenkins

**Answer: b**

**Explanation:** Jenkins Node is used to manage the distribution of work to Jenkins Slaves.

**10. What is the purpose of the Jenkins CLI?**

a. To provide a command-line interface for Jenkins  
 b. To enable Jenkins to communicate with external systems  
 c. To manage the configuration of Jenkins  
 d. To generate reports on the status of Jenkins jobs

**Answer: a**

**Explanation:** The Jenkins CLI (Command Line Interface) is used to provide a command-line interface for Jenkins. It allows users to interact with Jenkins from a terminal or command prompt.

**11. What is the Jenkins workspace?**

a. The directory where Jenkins is installed  
 b. The directory where Jenkins stores configuration data  
 c. The directory where Jenkins stores build artifacts  
 d. The directory where Jenkins stores log files

**Answer: c**

**Explanation:** The Jenkins workspace is the directory where Jenkins stores build artifacts, such as compiled code, test results, and documentation.

**12. Which of the following is not a Jenkins plugin?**

a. Git  
 b. Maven  
 c. Jenkinsfile  
 d. Slack

**Answer: c**

**Explanation:** Jenkinsfile is not a Jenkins plugin. It is a text file that defines the build process in Jenkins.

**13. What is the purpose of the Jenkins Dashboard?**

a. To provide an overview of the status of Jenkins jobs  
 b. To manage the configuration of Jenkins  
 c. To generate reports on the status of Jenkins jobs  
 d. To enable Jenkins to communicate with external systems

**Answer: a**

**Explanation:** The Jenkins Dashboard provides an overview of the status of Jenkins jobs. It allows users to monitor the progress of builds and view the results of tests.

**14. What is the Jenkins Global Tool Configuration?**

a. A feature that allows users to configure global settings in Jenkins  
 b. A plugin that provides additional functionality to Jenkins  
 c. A tool used to manage the installation of software on Jenkins nodes  
 d. A tool used to manage the distribution of work to Jenkins slaves

**Answer: c**

**Explanation:** The Jenkins Global Tool Configuration is a tool used to manage the installation of software on Jenkins nodes. It allows users to specify which software tools are required for builds and ensures that they are installed on the nodes.

**15. What is the purpose of the Jenkins Build History?**

a. To provide a log of the builds that have been performed in Jenkins  
 b. To manage the configuration of Jenkins  
 c. To generate reports on the status of Jenkins jobs  
 d. To enable Jenkins to communicate with external systems

**Answer: a**

**Explanation:** The Jenkins Build History provides a log of the builds that have been performed in Jenkins. It allows users to view the status of previous builds and identify any issues that occurred.

**16. What is the Jenkins Pipeline Syntax?**

a. A syntax used to define the build process in Jenkins  
 b. A syntax used to define the configuration of Jenkins  
 c. A syntax used to define the structure of Jenkins plugins  
 d. A syntax used to define the communication protocols used by Jenkins

**Answer: a**

**Explanation:** The Jenkins Pipeline Syntax is a syntax used to define the build process in Jenkins. It is used to create pipelines that automate the building, testing, and deployment of software.

**17. What is the purpose of the Jenkinsfile Validator?**

a. To validate the syntax of Jenkinsfiles  
 b. To validate the configuration of Jenkins  
 c. To validate the functionality of Jenkins plugins  
 d. To validate the results of Jenkins builds

**Answer: a**

**Explanation:** The Jenkinsfile Validator is used to validate the syntax of Jenkinsfiles. It ensures that the syntax is correct and identifies any errors or issues.

**18. What is the Jenkinsfile Sandbox?**

a. A feature that allows users to test their Jenkinsfiles in a secure environment  
 b. A plugin that provides additional functionality to Jenkins  
 c. A tool used to manage the installation of software on Jenkins nodes  
 d. A tool used to manage the distribution of work to Jenkins slaves

**Answer: a**

**Explanation:** The Jenkinsfile Sandbox is a feature that allows users to test their Jenkinsfiles in a secure environment. It allows users to validate their Jenkinsfiles and ensure that they will run correctly when executed in the production environment.

**19. What is the Jenkins Pipeline?**

a. A plugin that provides additional functionality to Jenkins  
 b. A feature that allows users to define the build process in Jenkins  
 c. A tool used to manage the installation of software on Jenkins nodes  
 d. A tool used to manage the distribution of work to Jenkins slaves

**Answer: b**

**Explanation:** The Jenkins Pipeline is a feature that allows users to define the build process in Jenkins. It is a script-based approach to defining the build process and allows users to create complex workflows that automate the entire build process.

**20. What is the Jenkins Job?**

a. A configuration file used to define the build process in Jenkins  
 b. A script used to define the build process in Jenkins  
 c. A plugin that provides additional functionality to Jenkins  
 d. A unit of work in Jenkins that performs a specific task

**Answer: d**

**Explanation:** The Jenkins Job is a unit of work in Jenkins that performs a specific task. Jobs can be used to build, test, or deploy software, and can be configured to run on specific nodes or at specific times.

**21. What is the Jenkins Node?**

a. A unit of work in Jenkins that performs a specific task  
 b. A plugin that provides additional functionality to Jenkins  
 c. A tool used to manage the installation of software on Jenkins nodes  
 d. A machine that is configured to execute Jenkins jobs

**Answer: d**

**Explanation:** The Jenkins Node is a machine that is configured to execute Jenkins jobs. Nodes can be configured to run on different operating systems, and can be used to distribute work across multiple machines.

**22. What is the purpose of the Jenkins Build Executor?**

a. To manage the configuration of Jenkins  
 b. To execute Jenkins jobs on Jenkins Nodes  
 c. To generate reports on the status of Jenkins jobs  
 d. To enable Jenkins to communicate with external systems

**Answer: b**

**Explanation:** The Jenkins Build Executor is used to execute Jenkins jobs on Jenkins Nodes. It is responsible for running the build process and executing the commands specified in the job configuration.

**23. What is the purpose of the Jenkins Build Queue?**

a. To manage the configuration of Jenkins  
 b. To execute Jenkins jobs on Jenkins Nodes  
 c. To generate reports on the status of Jenkins jobs  
 d. To queue Jenkins jobs for execution by the Build Executor

**Answer: d**

**Explanation:** The Jenkins Build Queue is used to queue Jenkins jobs for execution by the Build Executor. Jobs are placed in the queue when there are no Build Executors available to execute them, and are executed in the order in which they were added to the queue.

**24. What is the purpose of the Jenkins Artifacts?**

a. To provide a log of the builds that have been performed in Jenkins  
 b. To manage the configuration of Jenkins  
 c. To store build artifacts generated by the build process  
 d. To enable Jenkins to communicate with external systems

**Answer: c**

**Explanation:** Jenkins Artifacts are used to store build artifacts generated by the build process. Artifacts can include compiled code, test results, and documentation, and are stored in the Jenkins workspace.

**25. What is the Jenkins Plugin Manager?**

a. A feature that allows users to configure global settings in Jenkins  
 b. A tool used to manage the installation of plugins in Jenkins  
 c. A tool used to manage the installation of software on Jenkins nodes  
 d. A tool used to manage the distribution of work to Jenkins slaves

**Answer: b**

**Explanation:** The Jenkins Plugin Manager is a tool used to manage the installation of plugins in Jenkins. It allows users to search for and install new plugins, update existing plugins, and configure plugin settings.

**26. What is the purpose of the Jenkins Global Tool Configuration?**

a. To manage the configuration of Jenkins  
 b. To manage the installation of software on Jenkins nodes  
 c. To configure global settings for tools used in Jenkins jobs  
 d. To enable Jenkins to communicate with external systems

**Answer: c**

**Explanation:** The Jenkins Global Tool Configuration is used to configure global settings for tools used in Jenkins jobs. It allows users to configure the location of tools, such as compilers and test frameworks, that are used by Jenkins jobs.

**27. What is the Jenkins Authentication Mechanism?**

a. A tool used to manage the installation of software on Jenkins nodes  
 b. A feature that allows users to configure global settings in Jenkins  
 c. A tool used to manage the distribution of work to Jenkins slaves  
 d. A mechanism used to authenticate users in Jenkins

**Answer: d**

**Explanation:** The Jenkins Authentication Mechanism is used to authenticate users in Jenkins. It can be configured to use a variety of authentication methods, including LDAP and Active Directory.

**28. What is the purpose of the Jenkins Authorization Mechanism?**

a. A mechanism used to authenticate users in Jenkins  
 b. A tool used to manage the installation of software on Jenkins nodes  
 c. A feature that allows users to configure global settings in Jenkins  
 d. A mechanism used to control user access to Jenkins resources

**Answer: d**

**Explanation:** The Jenkins Authorization Mechanism is used to control user access to Jenkins resources. It can be used to define user roles and permissions, and to control access to specific jobs and build artifacts.

**29. What is the Jenkins User Interface?**

a. A tool used to manage the installation of software on Jenkins nodes  
 b. A feature that allows users to configure global settings in Jenkins  
 c. The graphical user interface used to interact with Jenkins  
 d. A mechanism used to authenticate users in Jenkins

**Answer: c**

**Explanation:** The Jenkins User Interface is the graphical user interface used to interact with Jenkins. It allows users to configure jobs, view build results, and manage Jenkins resources.

**30. What is the purpose of the Jenkins REST API?**

a. To manage the configuration of Jenkins  
 b. To enable Jenkins to communicate with external systems  
 c. To configure global settings for tools used in Jenkins jobs  
 d. To manage the installation of plugins in Jenkins

**Answer: b**

**Explanation:** The Jenkins REST API is used to enable Jenkins to communicate with external systems. It allows users to access Jenkins resources and perform actions, such as triggering builds and retrieving build results, from external systems.

**31. What is the Jenkins Continuous Integration Server?**

a. A tool used to manage the installation of software on Jenkins nodes  
 b. A feature that allows users to configure global settings in Jenkins  
 c. A tool used to manage the distribution of work to Jenkins slaves  
 d. A tool used to automate the build and testing process in software development

**Answer: d**

**Explanation:** The Jenkins Continuous Integration Server is a tool used to automate the build and testing process in software development. It is designed to support the continuous integration and delivery of software, and can be used to automate the entire build process.

**32. What is the Jenkins Master?**

a. The primary node in a Jenkins cluster  
 b. A tool used to manage the installation of software on Jenkins nodes  
 c. A tool used to manage the distribution of work to Jenkins slaves  
 d. The machine on which the Jenkins server is installed

**Answer: d**

**Explanation:** The Jenkins Master is the machine on which the Jenkins server is installed. It is responsible for managing the Jenkins configuration, scheduling builds, and distributing work to Jenkins slaves.

**33. What is a Jenkins Slave?**

a. A tool used to manage the installation of software on Jenkins nodes  
 b. A tool used to manage the distribution of work to Jenkins slaves  
 c. A machine that is configured to execute builds for a Jenkins Master  
 d. The primary node in a Jenkins cluster

**Answer: c**

**Explanation:** A Jenkins Slave is a machine that is configured to execute builds for a Jenkins Master. It receives work from the Jenkins Master and executes it in a separate process or container.

**34. What is the purpose of the Jenkins Pipeline?**

a. To manage the configuration of Jenkins  
 b. To manage the installation of plugins in Jenkins  
 c. To enable Jenkins to communicate with external systems  
 d. To define the entire build process in code

**Answer: d**

**Explanation:** The Jenkins Pipeline is used to define the entire build process in code. It allows users to define build stages, dependencies, and triggers, and to specify the order in which builds should be executed.

**35. What is the Jenkinsfile?**

a. A file used to configure Jenkins plugins  
 b. A file used to define a Jenkins Pipeline  
 c. A file used to configure Jenkins nodes  
 d. A file used to define Jenkins user roles

**Answer: b**

**Explanation:** The Jenkinsfile is a file used to define a Jenkins Pipeline. It is written in Groovy and contains the script that defines the build process.

**36. What is the purpose of the Jenkins Job DSL?**

a. To manage the installation of plugins in Jenkins  
 b. To enable Jenkins to communicate with external systems  
 c. To define Jenkins jobs in code  
 d. To manage the configuration of Jenkins

**Answer: c**

**Explanation:** The Jenkins Job DSL is used to define Jenkins jobs in code. It allows users to define job configurations in a script, which can be versioned and shared like any other code.

**37. What is the purpose of the Jenkins Environment Variables?**

a. To manage the installation of plugins in Jenkins  
 b. To enable Jenkins to communicate with external systems  
 c. To define global settings for tools used in Jenkins jobs  
 d. To define parameters that can be passed to Jenkins jobs

**Answer: c**

**Explanation:** The Jenkins Environment Variables are used to define global settings for tools used in Jenkins jobs. They allow users to define common settings, such as the location of a compiler or the version of a tool, that can be reused across multiple jobs.

**38. What is the purpose of the Jenkins Plugins?**

a. To manage the configuration of Jenkins  
 b. To manage the installation of software on Jenkins nodes  
 c. To manage the distribution of work to Jenkins slaves  
 d. To extend the functionality of Jenkins

**Answer: d**

**Explanation:** The Jenkins Plugins are used to extend the functionality of Jenkins. They allow users to add new features, such as support for different version control systems, or to integrate with external tools, such as issue trackers or deployment systems.

**39. What is the Jenkins CLI?**

a. A command-line interface used to interact with a Jenkins server  
 b. A graphical user interface used to interact with a Jenkins server  
 c. A tool used to manage the installation of software on Jenkins nodes  
 d. A tool used to manage the distribution of work to Jenkins slaves

**Answer: a**

**Explanation:** The Jenkins CLI is a command-line interface used to interact with a Jenkins server. It allows users to perform common tasks, such as creating jobs or triggering builds, from the command line.

**40. What is the Jenkins REST API?**

a. An interface used to interact with Jenkins via HTTP requests  
 b. A tool used to manage the installation of software on Jenkins nodes  
 c. A tool used to manage the distribution of work to Jenkins slaves  
 d. A graphical user interface used to interact with a Jenkins server

**Answer: a**

**Explanation:** The Jenkins REST API is an interface used to interact with Jenkins via HTTP requests. It allows users to perform common tasks, such as triggering builds or retrieving build results, using simple HTTP calls.

**41. What is the purpose of the Jenkins Notification Plugins?**

a. To manage the installation of plugins in Jenkins  
 b. To enable Jenkins to communicate with external systems  
 c. To define Jenkins jobs in code  
 d. To extend the functionality of Jenkins

**Answer: b**

**Explanation:** The Jenkins Notification Plugins are used to enable Jenkins to communicate with external systems. They allow users to send notifications, such as build results or deployment status, to other systems, such as chat services or email.

**42. What is the purpose of the Jenkins Distributed Builds?**

a. To manage the distribution of work to Jenkins slaves  
 b. To manage the installation of plugins in Jenkins  
 c. To enable Jenkins to communicate with external systems  
 d. To define Jenkins jobs in code

**Answer: a**

**Explanation:** The Jenkins Distributed Builds are used to manage the distribution of work to Jenkins slaves. They allow users to distribute builds across multiple nodes, reducing the load on any single machine and increasing the speed of builds.

**43. What is the purpose of the Jenkins Backup Plugin?**

a. To manage the installation of plugins in Jenkins  
 b. To manage the distribution of work to Jenkins slaves  
 c. To enable Jenkins to communicate with external systems  
 d. To create backups of Jenkins configurations and jobs

**Answer: d**

**Explanation:** The Jenkins Backup Plugin is used to create backups of Jenkins configurations and jobs. It allows users to easily restore a Jenkins server in the event of a failure or to migrate to a new server.

**44. What is the purpose of the Jenkins Security Plugin?**

a. To manage the installation of plugins in Jenkins  
 b. To manage the distribution of work to Jenkins slaves  
 c. To enable Jenkins to communicate with external systems  
 d. To manage security settings for Jenkins

**Answer: d**

**Explanation:** The Jenkins Security Plugin is used to manage security settings for Jenkins. It allows users to define users, groups, and permissions for accessing Jenkins resources, ensuring that only authorized users can perform actions in Jenkins.

**45. What is the purpose of the Jenkins Parameterized Builds?**

a. To define parameters that can be passed to Jenkins jobs  
 b. To manage the distribution of work to Jenkins slaves  
 c. To manage the installation of software on Jenkins nodes  
 d. To enable Jenkins to communicate with external systems

**Answer: a**

**Explanation:** The Jenkins Parameterized Builds are used to define parameters that can be passed to Jenkins jobs. They allow users to customize the behavior of Jenkins jobs, such as specifying the version of software to build or the target environment for deployment.

**Cloud**

**1. What is the full form of AWS?**

1. Amazon web-based service
2. Amazon web-store service
3. Amazon web service
4. Amazon web-data service

**Answer:** C) Amazon web service

**Explanation:**

AWS stands for amazon web service.

**2. How many types of cloud computing are there?**

1. 2
2. 3
3. 4
4. 5

**Answer:** B) 3

**Explanation:**

There are three types of cloud computing, IAAS(Infrastructure as a service), PAAS(Platform as a service), and SAAS (Software as a service).

**3. Which of the following are the advantages of AWS?**

1. Flexibility
2. Cost-effectiveness
3. Scalability
4. Security
5. All of the above

**Answer:** E) All of the above

**Explanation:**

AWS provides flexibility, cost-effectiveness, scalability, and security.

**4. Do traditional IT models provide flexibility?**

1. YES
2. NO

**Answer:** B) NO

**Explanation:**

No, the traditional IT model was unable to provide flexibility as they require huge investments, on the other hand, AWS provides flexibility.

**5. What is the region in AWS?**

1. A region is a geographical area or collection of data centers.
2. A region is an isolated logical data center
3. A region is the end-points for AWS.

**Answer:** A) A region is a geographical area or collection of data centers.

**Explanation:**

A region is a geographical area or collection of data centers. A region contains more than 1 availability zone.

**6. What is the Availability zone in AWS?**

1. An Availability zone is a geographical area or collection of data centers.
2. An Availability zone is an isolated logical data center in a region
3. An Availability zone is the end-points for AWS.

**Answer:** B) An Availability zone is an isolated logical data center in a region

**Explanation:**

An Availability zone is an isolated logical data center in a region, these are multiplied within each region.

**7. What are edge locations in AWS?**

1. The edge location is a geographical area or collection of data centers.
2. The edge location is an isolated logical data center in a region
3. The edge locations are the end-points for AWS, used to deliver fast content to users.

**Answer:** C) The edge locations are the end-points for AWS, used to deliver fast content to users.

**Explanation:**

The edge locations are the end-points for AWS, used to deliver fast content to users.

**8. Which of the following are the components of AWS infrastructure?**

1. Edge location
2. Regions
3. Availability zone
4. Regional Edge caches
5. All of the above

**Answer:** E) All of the above

**Explanation:**

The following are the components of AWS infrastructure:

1. Edge location
2. Regions
3. Availability zone
4. Regional Edge caches

**9. What do you mean by AWS account ID?**

1. AWS account ID is a 12-digit number that is used to construct Amazon Resource Names (ARNs).
2. AWS account ID is 64-digit hexadecimal used in an Amazon S3 bucket policy.

**Answer:** A) AWS account ID is a 12-digit number that is used to construct Amazon Resource Names (ARNs).

**Explanation:**

AWS account ID is a 12-digit number that is used to construct Amazon Resource Names (ARNs).

**10. What do you mean by canonical user ID?**

1. Canonical user ID is a 12-digit number that is used to construct Amazon Resource Names (ARNs).
2. Canonical user ID is 64-digit hexadecimal used in an Amazon S3 bucket policy.

**Answer:** B) Canonical user ID is 64-digit hexadecimal used in an Amazon S3 bucket policy.

**Explanation:**

Canonical user ID is 64-digit hexadecimal used in an Amazon S3 bucket policy.

**11. What Does IAM stands for in AWS?**

1. Identity access manager
2. Identity access management
3. Identify user-access management

**Answer:** B) Identity access management

**Explanation:**

In AWS, IAM stands for identity access management.

**12. What is the role of IAM?**

1. Allows you to set permission, roles, and users.
2. Allows you to manage users and their access.
3. It allows only specific users to use specific services.
4. All of the above.

**Answer:** D) All of the above.

**Explanation:**

In AWS, the IAM role is that it allows us to set permissions, roles, and users and also allows us to manage different users and their access. It helps us to assign a specific user to use some specific services.

**13. Is the IAM service in AWS free to use?**

1. Yes
2. No

**Answer:** A) YES

**Explanation:**

In AWS, the IAM service is free to use.

**14. Does IAM provide us with networking control?**

1. Yes
2. No

**Answer:** A) YES

**Explanation:**

In AWS, IAM provides us to control the network as it ensures that the user only accesses the organization's corporate network.

**15. In AWS, IAM identities are categorized into how many parts?**

1. 1
2. 2
3. 3
4. 4

**Answer:** C) 3

**Explanation:**

In AWS, IAM identities are categorized into three parts. They are as follows- IAM users, IAM roles, and IAM group.

**16. What is an AWS account root user?**

1. AWS account root user has complete access to all the services and resources in the account.
2. Group of many users in IAM AWS is generally called an AWS account root user.

**Answer:** A) AWS account root user has complete access to all the services and resources in the account.

**Explanation:**

AWS account root user has complete access to all the services and resources in the account.

**17. Can an AWS account root user access the billing information?**

1. Yes
2. No

**Answer:** A) YES

**Explanation:**

Yes, AWS account root users can access the billing information.

**18. \_\_\_\_ are used to grant permissions to your IAM Users to access AWS resources within your own or different account.**

1. IAM root user
2. IAM groups
3. IAM roles

**Answer:** C) IAM roles

**Explanation:**

AWS Roles permit users to access resources belonging to your own or someone else's account via your IAM Roles.

**19. Can you create a role for an AWS service using AWS CLI?**

1. Yes
2. No

**Answer:** A) YES

**Explanation:**

Yes, you can easily create roles for an AWS service using AWS CLI.

**20. What does S3 stand for in AWS?**

1. Simple storage service
2. Standard storage service
3. Simple standard storage
4. Simple standard service

**Answer:** A) Simple storage service

**Explanation:**

In AWS, S3 stands for simple storage service.

**21. S3 is a \_\_\_\_?**

1. Block-based storage
2. Object-based storage

**Answer:** B) Object-based storage

**Explanation:**

S3 is object-based storage.

**22. What kind of data can you store in S3?**

1. Images
2. PDF files
3. Word files
4. Documents
5. All of the above

**Answer:** E) All of the above

**Explanation:**

In AWS, S3 can store images, pdf files, word files, etc.

**23. What capacity of files can you store in S3?**

1. 0 bytes- 100 TB
2. 0 bytes -50 TB
3. 0 bytes -5 TB
4. 0 bytes -25 TB

**Answer:** C) 0 bytes -5 TB

**Explanation:**

In S3, the individual size of the file should be from 0 bytes to 5 TB.

**24. Does s3 have unlimited data storage?**

1. Yes
2. No

**Answer:** A) YES

**Explanation:**

Yes, S3 has unlimited data storage.

**25. What are buckets in S3?**

1. Buckets are simply a container that store files.
2. Buckets are the entities that you want to store in S3.
3. Buckets are the unique identifier in S3.

**Answer:** A) Buckets are simply a container that store files.

**Explanation:**

Buckets are simply a container that store files.

**26. What are objects in S3?**

1. objects are simply a container that store files.
2. objects are the entities that are stored in S3.
3. objects are the unique identifier in S3.

**Answer:** B) objects are the entities that are stored in S3.

**Explanation:**

Objects are the entities that are stored in S3.

**27. Every object in a bucket is associated with one unique\_\_\_\_?**

1. Value
2. Region
3. Key
4. Zone

**Answer:** C) Key

**Explanation:**

Every object in a bucket is associated with one unique key.

**28. Is it important to keep the bucket name unique?**

1. Yes
2. No

**Answer:** A) YES

**Explanation:**

Yes, buckets should have unique names, as they are universal namespaces.

**29. How many types of storage classes does the S3 contain?**

1. 1
2. 2
3. 3
4. 4

**Answer:** D) 4

**Explanation:**

S3 contains four storage classes. They are as follows:

* S3 standard
* S3 standard IA
* S3 one zone-infrequent access
* S3 glacier

**30. Among the four s3 storage classes, which storage class stores the data redundantly across multiple devices?**

1. S3 standard
2. S3 standard IA
3. S3 one zone-infrequent access
4. S3 glacier

**Answer:** A) S3 standard

**Explanation:**

Among the four s3 storage classes S3 standard storage class stores the data redundantly across multiple devices.

**31. What does IA stand for in S3 standard IA?**

1. Identifier access
2. Identity Access
3. Infrequent accessed
4. Identify access

**Answer:** C) Infrequent accessed

**Explanation:**

IA stands for Infrequent access in S3 standard IA.

**32. Among the four s3 storage classes, which storage class is used for the data that is need to be accessed rarely Nevertheless, a rapid response is required when needed.**

1. S3 standard
2. S3 standard IA
3. S3 one zone-infrequent access
4. S3 glacier

**Answer:** B) S3 standard IA

**Explanation:**

Among the four S3 storage classes S3 standard IA storage class is used for the data that need to be accessed rarely Nevertheless, a rapid response is required when needed.

**33. Which of the following is the cheapest s3 storage class?**

1. S3 standard
2. S3 standard IA
3. S3 one zone-infrequent access
4. S3 glacier

**Answer:** D) S3 glacier

**Explanation:**

S3 glacier is the cheapest s3 storage class.

**34. What is cross-region replication?**

1. Data is replicated across buckets in a different region using Cross Region Replication.
2. Data is replicated across single buckets in a different availability zone using Cross Region Replication.
3. Data is stored across buckets in a region using Cross Region Replication.

**Answer:** A) Data is replicated across buckets in a different region using Cross Region Replication.

**Explanation:**

Data is replicated across buckets in a different region using Cross Region Replication.

**35. What do you mean by On-premises in AWS?**

1. The concept of on-premises refers to keeping the IT environment on-site while the cloud is housed off-site under the care of someone else.
2. The concept of on-premises refers to keeping the IT environment off-site under the care of someone else while the cloud is kept on-site.
3. It means storing the cloud activity on your device and IT environment on the cloud.

**Answer:** A) The concept of on-premises refers to keeping the IT environment on-site while the cloud is housed off-site under the care of someone else.

**Explanation:**

The concept of on-premises refers to keeping the IT environment on-site while the cloud is housed off-site under the care of someone else.

**36. What is a storage gateway?**

1. Storage gateway is a service in AWS that connects on-premise cloud-based storage to off-site software appliances.
2. Storage gateway is a service in AWS that connects an on-premises software appliance with cloud-based storage.

**Answer:** B) Storage gateway is a service in AWS that connects an on-premises software appliance with cloud-based storage.

**Explanation:**

Storage gateway is a service in AWS that connects an on-premises software appliance with cloud-based storage.

**37. How many types of storage gateways are there?**

1. 1
2. 2
3. 3
4. 4

**Answer:** C) 3

**Explanation:**

There are three types of storage gateway: -

1. File Gateway (NFS)
2. Volume Gateway (iSCSI)
3. Tape Gateway (VTL)

**38. Which type of storage gateway is used to store flat files like videos, pictures, and pdf?**

1. File Gateway
2. Volume Gateway
3. Tape Gateway

**Answer:** A) File Gateway

**Explanation:**

File gateway is used to store flat files like videos, pictures, pdf, etc.

**39. Which type of storage gateway uses the network file system technique?**

1. File Gateway
2. Volume Gateway
3. Tape Gateway

**Answer:** A) File Gateway

**Explanation:**

File gateway is a type of storage gateway in AWS which uses the NFS technique commonly known as the network file system.

**40. Which type of storage gateway presents your on-premise application with the iSCSI block storage instead?**

1. File Gateway
2. Volume Gateway
3. Tape Gateway

**Answer:** B) Volume Gateway

**Explanation:**

Volume Gateways present your on-premise application with the iSCSI block storage instead.

**1. When did Google cloud platform launch?**

1. 2006
2. 2005
3. 2007
4. 2008

**Answer:** D) 2008

**Explanation:**

GCP (google cloud platform) was launched in 2008.

**2. Among AWS and GCP, which of the cloud services came first?**

1. AWS (amazon web service)
2. GCP (Google cloud platform)

**Answer:** A) AWS (amazon web service)

**Explanation:**

AWS (Amazon web service) was launched in 2006.

**3. Among AWS and GCP, which of the cloud services is more in demand and offers more services?**

1. AWS (amazon web service)
2. GCP (Google cloud platform)

**Answer:** A) AWS (amazon web service)

**Explanation:**

AWS (Amazon web service) is demanded more as it offers more services than GCP.

**4. What do you mean by cloud computing? Select the best answer.**

1. A term referring to storing data over the internet.
2. A platform where you can access huge amounts of data without having a hard disk.
3. A platform where you can access the data from a remote server.
4. All of the above.

**Answer:** D) All of the above.

**Explanation:**

Cloud computing is a general term that is used to store a huge amount of data over the internet. This is a platform where you can access your data from any remote server.

**5. Cloud computing offers how many service models?**

1. 3
2. 4
3. 2
4. 5

**Answer:** A) 3

**Explanation:**

Cloud computing offers three service models: IaaS, PaaS, and SaaS.

**6. Developing and deploying web-based software in data centers is managed by which Google cloud platform service?**

1. Compute engine
2. Google App engine
3. Kubernetes Engines

**Answer:** B) Google App engine

**Explanation:**

Developing and deploying web-based software in data centers is managed by the Google App engine service of GCP.

**7. Google App Engine follows which cloud computing model?**

1. IAAS
2. SAAS
3. PAAS

**Answer:** C) PAAS

**Explanation:**

Google app engine follows the PAAS i.e., platform as a service cloud computing model.

**8. Google App Engine comes under which kind of service?**

1. Compute service
2. Networking service
3. Storage service
4. Big data service

**Answer:** A) Compute service

**Explanation:**

Google app engine is a compute service in GCP.

**9. Can you test your application in the Google app engine?**

1. Yes
2. No

**Answer:** A) Yes

**Explanation:**

Google app engine offers the feature to test your application.

**10. Which of the following languages are supported by the Google app engine?**

1. Python
2. Java
3. Ruby
4. C++
5. All of the above

**Answer:** E) All of the above

**Explanation:**

Google app engine supports a variety of languages including GO, PHP, Java, Python, NodeJS, .NET, and Ruby.

**11. Google app engine follows which type of architecture?**

1. Layered architecture
2. Client-Server architecture.
3. Event-Driven architecture.
4. Microkernel architecture.
5. Microservices architecture.

**Answer:** E) Microservices architecture.

**Explanation:**

Google app engine follows the microservices type of architecture.

**12. Which of the following GCP Services aid us to run Windows and Linux-based virtual machines?**

1. Compute engine
2. Google App engine
3. Kubernetes Engines

**Answer:** A) Compute engine

**Explanation:**

Compute engine is a type of GCP Service that aids us to run Windows and Linux-based virtual machines.

**13. Google compute engine follows which cloud computing model?**

1. IAAS
2. SAAS
3. PAAS

**Answer:** A) IAAS

**Explanation:**

Google compute engine follows the IAAS i.e., infrastructure as a service cloud computing model.

**14. Google Compute Engine comes under which kind of service?**

1. Compute service
2. Networking service
3. Storage service
4. Big data service

**Answer:** A) Compute service

**Explanation:**

Google compute engine is a compute service in GCP and both the google app engine and google compute engine works together but their process is different.

**15. What is GKE in the google cloud platform?**

1. Google kit environment
2. Google Kubernetes environment
3. Google Kubernetes engine

**Answer:** C) Google Kubernetes engine

**Explanation:**

GKE stands for Google Kubernetes engine.

**16. Containerized applications can be deployed, managed, and scaled on Google using which service?**

1. Compute engine
2. Google App engine
3. Kubernetes Engines

**Answer:** C) Kubernetes Engines

**Explanation:**

Containerized applications can be deployed, managed, and scaled on the Google Kubernetes engine.

**17. Google Kubernetes Engines follows which cloud computing model?**

1. IAAS
2. SAAS
3. PAAS

**Answer:** C) PAAS

**Explanation:**

Google Kubernetes Engines follows the PAAS i.e., platform as a service cloud computing model.

**18. Google Kubernetes Engines comes under which kind of service?**

1. Compute service
2. Networking service
3. Storage service
4. Big data service

**Answer:** A) Compute service

**Explanation:**

Google Kubernetes Engines is a compute service in GCP.

**19. What does VPC stand for?**

1. Virtual private computing
2. Virtual private cloud
3. Virtual public computing

**Answer:** B) Virtual private cloud

**Explanation:**

VPC stands for virtual private cloud.

**20. VPC comes under which kind of service?**

1. Compute service
2. Networking service
3. Storage service
4. Big data service

**Answer:** B) Networking service

**Explanation:**

VPC is a networking service in GCP.

**21. Which of the following GCP services helps you to create a secure environment for the application deployments?**

1. Compute engine
2. Google App engine
3. Cloud load balancing
4. VPC

**Answer:** D) VPC

**Explanation:**

VPC is a kind of GCP service that helps you to create a secure environment for application deployments.

**22. Suppose you are asked to distribute workload across different computing resources then which of the following GCP service you would use?**

1. Compute engine
2. Google App engine
3. Cloud load balancing
4. VPC

**Answer:** C) Cloud load balancing

**Explanation:**

Cloud load balancing is a type of GCP service which distributes workload across different computing resources.

**23. How many types of google cloud load balancers are there?**

1. 3
2. 4
3. 5
4. 6

**Answer:** D) 6

**Explanation:**

Google cloud load balancers are of 6 types, which are again divided into two types: global type and regional type.

**24. How many types of global load balancers are there in GCP?**

1. 4
2. 5
3. 3
4. 6

**Answer:** C) 3

**Explanation:**

Global load balancers are: HTTP Load Balancer, the SSL Proxy, and the TCP Proxy.

**25. How many types of regional load balancers are there in GCP?**

1. 4
2. 5
3. 3
4. 6

**Answer:** C) 3

**Explanation:**

Regional load balancers are: Internal HTTP(S), External network TCP/UDP, and Internal TCP/UDP.

**26. Google load balancing comes under which kind of service?**

1. Compute service
2. Networking service
3. Storage service
4. Big data service

**Answer:** B) Networking service

**Explanation:**

Google load balancing comes under networking service.

**27. What is CDN?**

1. Compute delivery net
2. Cloud delivery network
3. Content delivery network
4. Content domain name

**Answer:** C) Content delivery network

**Explanation:**

CDN stands for content delivery network.

**28. In which of the following GCP service data centers and proxy servers are distributed geographically?**

1. Google App engine
2. Cloud load balancing
3. VPC
4. Content delivery network

**Answer:** D) Content delivery network

**Explanation:**

CDN is a type of GCP service in which data centers and proxy servers are distributed geographically.

**29. CDN comes under which kind of service?**

1. Compute service
2. Networking service
3. Storage service
4. Big data service

**Answer:** B) Networking service

**Explanation:**

CDN comes under networking service.

**30. How many types of storage classes does GCP provide?**

1. 2
2. 3
3. 4
4. 5

**Answer:** C) 4

**Explanation:**

There are four types of storage classes in GCP: Standard storage, nearline storage, coldline storage, and archive storage.

**31. Which type of storage class in GCP is commonly used for hot data?**

1. Standard storage
2. Nearline storage
3. Coldline storage
4. Archive storage.

**Answer:** A) Standard storage

**Explanation:**

Standard storage is a type of storage class in GCP which is commonly used for hot data.

**1. When Microsoft Azure came into existence?**

1. 2004
2. 2002
3. 2010
4. 2001

**Answer:** C) 2010

**Explanation:**

Microsoft Azure came into existence in the year 2010.

**2. What do you mean by cloud computing? Select the best answer.**

1. A term referring to storing data over the internet
2. A platform where you can access huge amounts of data without having a hard disk.
3. A platform where you can access the data from a remote server.
4. All of the above

**Answer:** D) All of the above

**Explanation:**

Cloud computing is a general term that is used to store a huge amount of data over the internet. This is a platform where you can access your data from any remote server.

**3. What is Microsoft Azure? Select the best answer.**

1. An open platform that aids in development, service hosting
2. A cloud computing service created by Microsoft which helps in testing, deploying, and managing the application.
3. A platform that offers a variety of technologies that are used for cloud service, mobile apps, etc.
4. All of the above

**Answer:** D) All of the above

**Explanation:**

Microsoft Azure is an open cloud computing service which is created by Microsoft that helps us in the development of an application, hosting an application, testing, managing, and deploying an applications. It provides a variety of services and technologies which can greatly be used for cloud service, mobile app development, etc.

**4. How many types of clouds are there in Microsoft Azure?**

1. 3
2. 2
3. 4
4. 5

**Answer:** A) 3

**Explanation:**

There are three types of cloud in Azure: PAAS, SAAS, and IASS.

**5. What is the Azure active directory?**

1. A directory where you can manage services, and app users from a distributed location.
2. A directory where you can manage services, and apps users from a centralized location.

**Answer:** B) A directory where you can manage services, and apps users from a centralized location.

**Explanation:**

Azure active directory is also known as AD. It is a directory that allows you to manage your files, users, services, etc from one central location.

**6. Among Windows active directory and Azure active directory which is known to be the oldest version?**

1. Windows Active directory
2. Azure active directory

**Answer:** B) Azure active directory

**Explanation:**

Windows active directory is the previous version of Azure active directory.

**7. How many types of audiences are there in the Azure active directory?**

1. 3
2. 4
3. 5
4. 6

**Answer:** A) 3

**Explanation:**

There are three types of audiences in the Azure active directory: IT administrators, Application developers, and Online customers.

**8. Which type of audience takes care of all the sign-in procedures in the Azure active directory?**

1. IT administrators
2. Application developers
3. Online customers.

**Answer:** A) IT administrators

**Explanation:**

IT administrators take care of all the sign-in procedures in the Azure active directory and all the work/issues related to authentications.

**9. Which is the foundational cloud platform layer?**

1. IAAS
2. PAAS
3. SAAS

**Answer:** A) IAAS

**Explanation:**

IAAS (infrastructure as a service) is the foundational cloud platform layer mainly used by IT administrators for processing, storage, and any fundamental computer operation.

**10. Who are the main users of IAAS?**

1. Developers
2. End users
3. Network architect

**Answer:** C) Network architect

**Explanation:**

IAAS is used by network architects.

**11. Which of the following are the advantages of using IAAS?**

1. IAAS provides dynamic scaling.
2. IAAS enables instant recovery from outages.
3. IAAS provides us with a monthly operational expense.
4. All of the above

**Answer:** D) All of the above

**Explanation:**

There are several advantages of IAAS:

* IAAS provides dynamic scaling.
* IAAS enables instant recovery from outages.
* IAAS provides us the monthly operational expense
* IAAS provides pay for what you use metrics
* It can be accessed via a simple internet connection.

**12. Which of the following platforms is used by clients to develop and deploy the application?**

1. IAAS
2. PAAS
3. SAAS

**Answer:** B) PAAS

**Explanation:**

PAAS is known as a platform as a service and clients use this to develop and deploy the application.

**13. Who are the main users of PAAS?**

1. Developers
2. End users
3. Network architect

**Answer:** A) Developers

**Explanation:**

PAAS is used by developers and application providers.

**14. Who are the main users of SAAS?**

1. Developers
2. End users
3. Network architect

**Answer:** B) End users

**Explanation:**

SAAS is used by end users.

**15. In which cloud computing service model software is hosted on the cloud and made available to clients?**

1. IAAS
2. PAAS
3. SAAS

**Answer:** C) SAAS

**Explanation:**

SAAS (software as a service) is a cloud computing service model where software is hosted on the cloud and made available to clients.

**16. The term \_\_\_\_refers to a geographical location or area?**

1. Regions
2. Data center
3. Resources

**Answer:** A) Regions

**Explanation:**

The term region refers to a geographical location or area.

**17. To reduce the latency in network requests, you should select a region that is at \_\_\_\_to most of the customers.**

1. Far distance
2. Closer distance
3. Moderate distance

**Answer:** B) Closer distance

**Explanation:**

It is suggested to choose a region that is closer to the customer, as it helps us to reduce the latency in network requests.

**18. \_\_\_\_ are physical buildings located in specific geographical regions?**

1. Resources
2. Regions
3. Data center
4. Resource groups

**Answer:** C) Data center

**Explanation:**

Data centers are physical buildings located in specific geographical regions.

**19. Creating, managing, and removing Azure resources and services is done through the \_\_\_\_, which is a web-based application?**

1. Resources
2. Azure portal
3. Resource manager templates
4. Resource groups

**Answer:** B) Azure portal

**Explanation:**

Creating, managing, and removing Azure resources and services is done through the Azure portal, which is a web-based application.

**20. Are Azure resources and Azure services the same?**

1. Yes
2. No

**Answer:** B) NO

**Explanation:**

Azure resources are billable objects whereas the services are something that is needed to be performed.

**21. An Azure \_\_\_\_ is a container that holds related resources for an Azure solution.**

1. Resources
2. Resource Group
3. Azure portal

**Answer:** B) Resource Group

**Explanation:**

An Azure resource group is a container that holds related resources for an Azure solution.

**22. Is Azure CLI and Azure PowerShell the same thing?**

1. Yes
2. No

**Answer:** B) NO

**Explanation:**

No, Azure CLI and Azure PowerShell are two different tools, and according to the task appropriate tool is decided.

**23. \_\_\_\_ is a set of modules that offer cmdlets to manage Azure.**

1. Azure PowerShell
2. Azure CLI

**Answer:** A) Azure PowerShell

**Explanation:**

PowerShell is a set of modules that offer cmdlets to manage Azure.

**24. Which of the following act as the kernel of Azure Service Platform in managing the hardware resources?**

1. Azure PowerShell
2. Azure CLI
3. Azure fabric controller
4. Azure orchestrator

**Answer:** C) Azure fabric controller

**Explanation:**

Azure fabric controllers act as the kernel of Azure Service Platform in managing the hardware resources. It mainly manages and monitors the allocation of computing resources.

**25. Azure Fabric controller in Azure is connected to which software?**

1. Azure PowerShell
2. REST API
3. Authentication servers
4. Azure orchestrator

**Answer:** D) Azure orchestrator

**Explanation:**

Azure fabric controller is connected to Azure orchestrator, which includes web services and REST API.

**26. How many certification categories does Microsoft Azure provide?**

1. 3
2. 4
3. 5
4. 6

**Answer:** A) 3

**Explanation:**

There are three categories of Microsoft certifications: Azure administrator, Azure developer, and Azure solution architect.

**27. How many types of cloud computing are there?**

1. 8
2. 4
3. 3
4. 6

**Answer:** C) 3

**Explanation:**

There are three types of cloud computing: Private cloud, public cloud and Hybrid cloud.

**28. In which of the following cloud computing resources are owned and operated by a third-party cloud service provider?**

1. Public cloud
2. Private Cloud
3. Hybrid cloud

**Answer:** A) Public cloud

**Explanation:**

In the public cloud resources are owned and operated by a third-party cloud service provider.

**29. A \_\_\_\_ is a collection of resources that are used exclusively within a business or organization.**

1. Public cloud
2. Private Cloud
3. Hybrid cloud

**Answer:** B) Private Cloud

**Explanation:**

A private cloud is a collection of resources that are used exclusively within a business or organization.

**30. Which storage is used to store huge amounts of unstructured data in Azure?**

1. Azure queue storage
2. Azure File storage
3. Azure blob storage
4. Azure table storage
5. Azure Archive storage

**Answer:** C) Azure table storage

**Explanation:**

Azure blob storage is used to store huge amounts of unstructured data in Azure, here you can store any type of text or binary data, such as a document, or media file.