Note

This = Correct

This = Doubt

# Descriptive Question Bank

## Explain overview and usage of Google Cloud Platform [10 marks]

Google Cloud Platform enables developers to build, test, and deploy applications on Google’s highly scalable and reliable infrastructure. Google has one of the largest and most advanced networks across the globe. Software infrastructures such as MapReduce, BigTable, and Dremel are the innovations for industrial development. Google Cloud Platform includes virtual machines, block storage, NoSQL datastore, and big data analytics. It provides a range of storage services that allow easy maintenance and quick access of user’s data. The cloud platform offers a fully managed platform as well as flexible virtual machines allowing the user to choose as per the requirements. Google also provides easy integration of user’s application within the cloud platform. Applications hosted on the cloud platform can automatically scale up to handle the most demanding workloads and scale down when traffic subsides. The cloud platform is designed to scale like Google’s own products, even when there is a huge traffic spike. Managed services such as App Engine or Cloud Datastore provide autoscaling that enables application to grow with the users. The user has to pay only for what he or she uses.

GCP differentiates itself from the other public cloud providers in a variety of ways. Google provides a comprehensive security model, a unique billing model — and a strong reliance on data analytics to ensure optimal performance.

### Security

The Google security model is built from more than 15 years of experience focused on keeping customers safe on Google products. Google Cloud Platform allows your applications and data to operate on the same trusted security model that Google built for its own network.

### Pricing

Like most cloud providers, GCP offers a pay-as-you-go monthly plan. This means your bill is based on the usage of its compute engine instances. However, Google takes things a step further by charging per second with a one-minute minimum. This allows you to save even more money when your business isn’t using Compute Engine, especially if you are running short-term workloads, or you’re running a dynamic web application.

### Big Data

Google also stands out because of its data analytics provided by BigQuery. With big data services, you can process data in the cloud to get answers to your most complex questions. You can also create schemas, load data, generate queries — and export data.

### GCP Services

There are several factors that influence why you would choose GCP as your public cloud provider. One of the major reasons is its services and distributed applications model.

This is geared toward businesses interested in building new applications in the cloud, or replacing some of their existing data center applications with cloud-based apps. In other words, moving to the cloud and a services model is not a destination, but rather a transition.

With a focus on services and applications, GCP gives you the power to build applications and blend assets also known as app modernization. By creating a hybridized model with components, you can stage your workloads in a more efficient way using Google's own experience as a pioneer in technology.

## Explain in detail 12 Factors app. [10 marks]

The 12 Factor App is a set of principles that describes a way of making software that, when followed, enables companies to create code that can be released reliably, scaled quickly, and maintained in a consistent and predictable manner.

The following is a brief synopsis of the principles of the 12 Factor App.

### I. Codebase

One codebase tracked in revision control, many deploys

The Codebase principle states that all assets related to an application, everything from source code, the provisioning script, and configuration settings, are stored in a source code repository that is accessible to development, testing, and system administration staff. The source code repository is also accessible to all automation scripts that are part of the Continuous Integration/Continuous Delivery (CI/CD) processes that are part of the enterprise's Software Development Lifecycle. (SDLC).

### II. Dependencies

Explicitly declare and isolate dependencies

The principle of Dependencies asserts that only code that is unique and relevant to the purpose of the application is stored in source control. External artifacts such as Node.js packages, Java .jar files, or .NET DLLs should be referenced in a dependencies manifest loaded into memory at development, testing, and production runtime. You want to avoid storing artifacts along with source code in the source code repository.

### III. Config

Store config in the environment

The Config principle states that configuration information is injected into the runtime environment as environment variables or as settings defined in an independent configuration file. While, in certain cases, it's permissible to store default settings that can be overridden directly in code, settings such as port number, dependency URLs, and state settings such as DEBUG should exist separately and be applied upon deployment. Good examples of external configuration files are a Java properties file, a [Kubernetes manifest file](https://developers.redhat.com/blog/2019/08/15/how-to-use-dekorate-to-create-kubernetes-manifests/), or a [docker-compose.yml](https://docs.docker.com/compose/) file.

The benefit of keeping configuration settings separate from application logic is that you can apply configuration settings according to the deployment path. For example, you can have one set of configuration settings for a deployment intended for a testing environment and a different set for a deployment designed for a production environment.

### IV. Backing Services

Treat backing services as attached resources

The Backing Services principle encourages architects to treat external components such as databases, email servers, message brokers, and independent services that can be provisioned and maintained by systems personnel as attached resources. Treating resources as backing services promotes flexibility and efficiency in the software development lifecycle (SDLC).

### V. Build, Release, Run

Strictly separate build and run stages

The principle of Build, Release, and Run breaks the deployment process down into three replicable stages that can be instantiated at any time. The Build stage is where code is retrieved from the source code management system and built/compiled into artifacts stored in an artifact repository such as Docker Hub or a Maven repository. After the code is built, configuration settings are applied in the Release stage. Then, in the Run stage, a runtime environment is provisioned via scripts using a tool such as Ansible. The application and its dependencies are deployed into the newly provisioned runtime environment.

The key to Build, Release, and Run is that the process is completely ephemeral. Should anything in the pipeline be destroyed, all artifacts and environments can be reconstituted from scratch using assets stored in the source code repository.

### VI. Processes

Execute the app as one or more stateless processes

The principle of Processes, which can be more accurately termed stateless processes, asserts that an application developed under the 12 Factor App structure will run as a collection of stateless processes. This means that no single process keeps track of the state of another process and that no process keeps track of information such as session or workflow status. A stateless process makes scaling easier. When a process is stateless, instances can be added and removed to address a particular load burden at a given point in time. Since each process operates independently, statelessness prevents unintended side effects.

### VII. Port Binding

Export services via port binding

The principle of Port Binding asserts that a service or application is identifiable to the network by port number, not a domain name. The reasoning is that domain names and associated IP addresses can be assigned on-the-fly by manual manipulation and automated service discovery mechanisms. Thus, using them as a point of reference is unreliable. However, exposing a service or application to the network according to port number is more reliable and easier to manage. At the least, potential issues due to a collision between port number assignment private to the network and public use of that same port number by another process publicly can be avoided using [port forwarding](https://en.wikipedia.org/wiki/Port_forwarding).

The essential idea behind the principle of port binding is that the uniform use of a port number is the best way to expose a process to the network. For example, the patterns have emerged in which port 80 is conventional for web servers running under HTTP, port 443 is the default port number for HTTPS, port 22 is for SSH, port 3306 is the default port for MySQL, and port 27017 is the default port for MongoDB.

### VIII. Concurrency

Scale-out via the process mode.

The principle of Concurrency recommends organizing processes according to their purpose and then separating those processes so that they can be scaled up and down according to need. As shown in the illustration above, an application is exposed to the network via web servers that operate behind a load balancer. The group of web servers behind the load balancer, in turn, uses business logic that is in Business Service processes that operate behind their own load balancer. Should the burden on the web servers increase, that group can be scaled up in an isolated manner to meet the demands at hand. However, should a bottleneck occur due to a burden placed on the Business Service, that layer can be scaled up independently.

Supporting concurrency means that different parts of an application can be scaled up to meet the need at hand. Otherwise, when concurrency is not supported, architectures have little choice but to scale up the application in its entirety.

### IX. Disposability

Maximize robustness with fast startup and graceful shutdown

The principle of Disposability asserts that applications should start and stop gracefully. This means doing all the required "housekeeping" before an application is made accessible to consumers. For example, a graceful startup will ensure that all database connections and access to other network resources are operational. Also, any other configuration work that needs to take place has taken place.

In terms of shutdown, disposability advocates ensuring that all database connections and other network resources are terminated properly and that all shutdown activity is logged, as shown in the code example shown above.

### X. Dev/Prod Parity

Keep development, staging, and production as similar as possible

The Dev/Prod Parity principle means all deployment paths are similar yet independent and that no deployment "leapfrogs" into another deployment target.

Rather, the CI/CD process will be adjusted to set the deployment target of V2 to Production. The CI/CD process will follow the expected Build, Release, and Run pattern towards that new target.

As you can see, Dev/Prod Parity is very similar to Build, Release, and Run. The important distinction is that Dev/Prod Parity ensures the same deployment process for Production as it does Development.

### XI. Logs

It treats logs as event streams. The Logs principle advocates sending log data in a stream that a variety of interested consumers can access. The process for routing log data needs to be separate from processing log data. For example, one consumer might only be interested in Error data, while another consumer might be interested in Request/Response data. Another consumer might be interested in storing all log data for event archiving. An added benefit is that even if an app dies, the log data lives on well afterward.

### XII. Admin Processes

It runs admin/management tasks as one-off processes. The principle of Admin Processes states that admin processes are first-class citizens in the software development lifecycle and need to be treated as such.

## Short note on: Spring boot fundamental, APIfication [5 marks each]

### Spring boot fundamental

Spring Boot is an open-source Java-based framework used to create a micro-Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications. This chapter will give you an introduction to Spring Boot and familiarizes you with its basic concepts.

Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring application that you can just run. You can get started with minimum configurations without the need for an entire Spring configuration setup.

#### Advantages

* Reduces the time spent on development and increases the overall efficiency of the development team.
* Helps to autoconfigure all components for a production-grade Spring app.
* Facilitates the creation and testing of Java-based applications by providing a default setup for unit and integration tests.
* Helps to avoid all the manual work of writing boilerplate code, annotations, and complex XML configurations.
* Comes with embedded HTTP servers like Jetty and Tomcat to test web applications.

#### Disadvantages

* If you have never worked with Spring before and want to learn about proxies, dependency injection, and AOP programming, it is not recommended to start with Spring Boot because it doesn’t cover most of these details.
* You really must understand a lot of the underlying Spring systems (and a bit of Spring history too), along with some advanced topics in order to modify and troubleshoot it.
* Spring Boot works well with microservices. The Spring Boot artifacts can be deployed directly into Docker containers. However, some developers don’t recommend the framework for building large and monolithic apps.
* If you are not familiar with other projects of the Spring ecosystem like Spring Security, Spring AMQP, Spring Integration, etc), using them with Spring Boot will make you miss many concepts that you would grasp if you had started using them independently.

### APIfication

Application programming interfaces (APIs) enable to drive new business by offering external and internal consumers seamless engagement with a company’s data and functions. APIs expose critical parts of an application to enable better utilization of technology investments. With technical advancements, APIs are also driving digital transformation initiatives and enabling IoT devices to communicate.

APIfication means offering functionality and data as a service for systems to use. An API hides the actual implementation and reveals only the simplified interactive model to the user. In generalised terms, there are two models for APIfication: proactive and reactive.

#### Benefits

* Outsourcing the data and functionality through a standard interface helps in providing excellent services, making better applications, and improve customer experiences.
* APIs act as a layer of abstraction between data and logic required to run a task at the source.
* APIs are applied to increase Mobility and Developer Productivity by taking an existing feature and using it anywhere instead of re-creating those features. Using existing features helps in minimizing application development time from months to weeks.
* By eliminating barriers, customers can easily access your interface, which allows them to use your product according to their needs.
* Focusing on analyzing the API services rather than application usability, you can utilize Customer Insights in making improvements that matter most to your top customers.
* Automating requests and providing open access to your API is an invaluable way to streamline operations and create and maintain long-term, profitable business relationships.

## What is asymmetric cryptography. Explain any one algorithm in detail. [10 marks]

Asymmetric cryptography, also known as public-key cryptography, is a process that uses a pair of related [keys](https://searchsecurity.techtarget.com/definition/key) -- one public key and one private key -- to [encrypt](https://searchsecurity.techtarget.com/definition/encryption) and decrypt a message and protect it from unauthorized access or use. A public key is a cryptographic key that can be used by any person to encrypt a message so that it can only be deciphered by the intended recipient with their private key. A private key -- also known as a secret key -- is shared only with key's initiator.

Asymmetric encryption uses a mathematically related pair of keys for encryption and decryption: a public key and a private key. If the public key is used for encryption, then the related private key is used for decryption; if the private key is used for encryption, then the related public key is used for decryption.

#### Uses

* Encrypted email
* The SSL/TSL cryptographic protocols
* Bitcoin and other cryptocurrencies

#### Advantages and disadvantages

##### Advantages

* the key distribution problem is eliminated because there's no need for exchanging keys.
* security is increased as the private keys don't ever have to be transmitted or revealed to anyone.

##### Disadvantages

* it's a slow process compared to symmetric cryptography, so it's not appropriate for decrypting bulk messages.
* if an individual loses his private key, he can't decrypt the messages he receives.

### Types of **asymmetric encryption**

* Rivest Shamir Adleman (RSA)
* the Digital Signature Standard (DSS), which incorporates the Digital Signature Algorithm (DSA)
* Elliptical Curve Cryptography (ECC)
* the Diffie-Hellman exchange method
* ElGamal encryption

### Elliptical Curve Cryptography (ECC)

An elliptic curve is an algorithm function for present ECC uses that is a plane and asymmetrical curve, which transverses a finite field comprising the points sustaining the following elliptic curve equation:

y²=x³ ax b.

Concerning the elliptic curve cryptography algorithm, this algebraic function (y²=x³ ax b) will appear like a symmetrical curve that is parallel to the x-axis when plotted.  The elliptic curve method is established on a sole one-way feature in which it simpler to complete a calculation but, at the same time, impracticable to invert or withdraw the outcomes of the calculation to find the initial numbers, unlike other forms of public-key cryptography. This property makes the elliptic curve cryptography algorithm more secure and efficient.

## Explain in detail cloud monitoring and various monitoring tools.

### Cloud Monitoring

Cloud monitoring is primarily part of cloud security and management processes, and it is generally implemented through automated monitoring software that provides central access and control over cloud infrastructure. Cloud administrators can review the operational status and health of any cloud-based device or component.

In addition to monitoring and ensuring cloud infrastructure/solution/service availability, cloud monitoring data also helps in evaluating the performance of the entire infrastructure on a modular level. Properties such as server uptime and response rate report can help in evaluating customer/user experience.

#### Types

The cloud has numerous moving components, and for top performance, it’s critical to safeguard that everything comes together seamlessly. This need has led to a variety of monitoring techniques to fit the type of outcome that a user wants. The main types of cloud monitoring are:

##### Database monitoring

Because most cloud applications rely on databases, this technique reviews processes, queries, availability, and consumption of cloud database resources. This technique can also track queries and data integrity, monitoring connections to show real-time usage data.

##### Website monitoring

A website is a set of files that is stored locally, which, in turn, sends those files to other computers over a network. This monitoring technique tracks processes, traffic, availability, and resource utilization of cloud-hosted sites.

##### Virtual network monitoring

This monitoring type creates software versions of network technology such as firewalls, routers, and load balancers. Because they’re designed with software, these integrated tools can give you a wealth of data about their operation.

##### Cloud storage monitoring

This technique tracks multiple analytics simultaneously, monitoring storage resources and processes that are provisioned to virtual machines, services, databases, and applications. This technique is often used to host infrastructure-as-a-service (IaaS) and software-as-a-service (SaaS) solutions.

##### Virtual machine monitoring

This technique is a simulation of a computer within a computer; that is, virtualization infrastructure and virtual machines. It’s usually scaled out in IaaS as a virtual server that hosts several virtual desktops.

#### Benefits

* Dedicated tools (and hardware) are maintained by the host
* Tools are used across several types of devices, including desktop computers, tablets, and phones, so your organization can monitor apps from any location
* Installation is simple because infrastructure and configurations are already in place

#### Cloud monitoring best practices

##### Observe your cloud service usage and fees.

* Increased costs can be triggered when scaling kicks in to meet demand. Strong monitoring solutions should track how much activity is on the cloud and its associated cost

##### Identify metrics and events that affect your bottom line.

* Not everything that can be measured needs to be reported

##### Use a single platform to report all data.

* You need solutions that can report data from different sources to a single platform. This consolidated information enables you to calculate uniform metrics and results in a complete performance view

##### Trigger rules with data.

* If activity surpasses or drops below certain levels, the right solution should be to add or subtract servers to maintain efficiency and performance

##### Separate your centralized data.

* Your organization must store your monitoring data separately from your proprietary apps, but the information should still be centralized for easy access

##### Monitor the user experience.

* To get the full picture of performance, review metrics such as response times and frequency of use

##### Try failure.

* Test tools to see what happens when an outage or a data breach occurs. This evaluation can create new standards for the alert system

# MCQ QUESTION BANK

## Amazon S3 is which type of storage service?

1. Object
2. Block
3. Simple
4. Secure

## Which of the following AWS tool is used for big data processing and analysis.?

1. S3
2. EC2
3. Elastic MapReduce
4. Snowball

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_enables the user to automatically distribute and balance the incoming application’s traffic

1. Elastic load balancing (ELB)
2. Amazon S3
3. Amazon SQS
4. CloudFront

## Which of the following provides feature of static website hosting, which is ideal for websites with static content?

1. Amazon S3
2. Amazon EC2
3. Amazon SQS
4. Snowbell

## Which of the following Windows Azure tool encapsulates all the information to be aware of before attempting the application migration to Windows Azure?

1. Azure Cloud Services
2. Migration Assessment Tool (MAT)
3. Windows Azure Pricing Calculator
4. Windows Azure Pack

## Which Google Cloud Platform service can be used for serverless file processing and running website backend?

1. Kubernetes Engine
2. App Engine
3. NFS
4. Compute Engine

## When would developers use microservices?

1. When they want to write cell phone applications that run quickly
2. When they work with ephemeral nano technology
3. When they need to create large, enterprise-level applications that are subject to changes on a frequent basis
4. When they create applications specifically for scientific test equipment

## Which of the following microservices architectural pattern defines how clients access the services in a microservice architecture?

1. Decomposition patterns
2. API Gateway pattern
3. Service Discovery pattern
4. Observability pattern

## \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_are used to route requests for a client to an available service instance in a microservice architecture

1. Service Discovery pattern
2. API Gateway pattern
3. Database per Service pattern
4. UI pattern

## Suppose your application is currently using a local PostgreSQL database for its operations and it is afterward replaced with the one hosted on the server of your company by just changing the URL and the database credentials.

### Above is the example of which 12 factor app?

1. Dependencies
2. Backing Services
3. Codebase
4. Build, Release and Run

## In APIfication process, which step defines digital resources and capabilities of the company?

1. Development
2. Design
3. Production
4. Discovery

## Which of API type needs specific access rights to be able to use them and are typically exposed via an API developer portal?

1. Partner API
2. Internal API
3. Open API
4. Closed API

## Which of the following API style of development is a collection of interlinked resources, which is very much how web apps are delivered?

1. Tunnel
2. Query
3. Hypermedia
4. Event

## Which of the following is part of Data Access layer in Spring framework?

1. Beans
2. Aspects
3. JMS
4. Context

## Which of the following is not IaaS related security issue?

1. Multitenancy
2. Identity management and access control
3. Hypervisor security
4. Resource locality

## Stream Ciphers encrypt pseudorandom sequences with bits of plaintext to generate ciphertext, usually with\_\_\_\_\_ operation.

1. OR
2. AND
3. XOR
4. NOR

## Which of the following statement is not correct related to Cipher Block Chaining (CBC)?

1. CBC is a mode of operation for stream ciphers.
2. Initialization vector (IV) is used in CBC in the initial phase.
3. It has better resistive nature towards cryptanalysis than ECB

## Which of the following modes of operations can be followed for both stream ciphers as well as block ciphers?

1. CBC (Cipher Block Chaining)
2. ECB (Electronic Code Book)
3. CFB (Cipher text Feed Back)
4. Counter (CTR) Mode

## Consider the following steps, Substitution bytes, Shift Rows, mix columns, Add round key

### The above steps are performed in each round of which of the following ciphers?

1. Rail fence cipher
2. Data Encryption Standard (DES)
3. Advance Encryption Standard (AES)
4. RSA

## What is the output of a cryptographic hash function means?

1. A variable set of bits
2. A fixed set of bits, derived from one-way mathematical operations
3. An output which may be easily discovered by an adversary
4. Outputs of such functions are of no importance

## Which of the following is strongest encryption technique?

1. DES (Data Encryption Standard)
2. Double DES
3. Triple DES
4. AES (Advance Encryption Standard)

## Which of the following is false for ECB mode of operation?

### i) The Plain text is broken into blocks of size 128 bytes

### ii) Blocks can be swapped, repeated, replaced without recipient noticing

### iii) Good for short data

### iv) Encryption of each block is done separately using a randomly generated key for each block

1. Only i
2. ii) and iii)
3. i) and iv)
4. ii) and iv)

## Which of the following statements are true?

### i) In the CBC mode, the plaintext block is XORed with previous ciphertext block before encryption

### ii) The CTR mode does not require an Initialization Vector

### iii) The last block in the CBC mode uses an Initialization Vector

### iv) In CBC mode repetitions in plaintext do not show up in ciphertext

1. iii)
2. ii) and iv)
3. All the Statements are true
4. i) ii) and iv)

## Which of the following can be classified under advantages and disadvantages of OFB mode?

### i) Transmission errors

### ii) A bit error in a ciphertext segment

### iii) Cannot recover from lost ciphertext segments

### iv) Ciphertext or segment loss

1. Advantages: None; Disadvantages: All
2. Advantages: All; Disadvantages: None
3. Advantages: i); Disadvantages: ii) iii) iv)
4. Advantages: i); ii) Disadvantages: iii) iv)