Google Cloud Platform Tutorial: From Zero to Hero with GCP

[Sergio Fuentes Navarro]

Do you have the knowledge and skills to design a mobile gaming analytics platform that collects, stores, and analyzes large amounts of bulk and real-time data?

Well, after reading this article, you will.

I aim to take you **from zero to hero in Google Cloud Platform (GCP) in just one article**. I will show you how to:

- Get started with a GCP account for free
- Reduce costs in your GCP infrastructure
- Organize your resources
- Automate the creation and configuration of your resources
- Manage operations: logging, monitoring, tracing, and so on.
- Store your data
- Deploy your applications and services
- Create networks in GCP and connect them with your on-premise networks
- Work with Big Data, AI, and Machine Learning
- Secure your resources

Once I have explained all the topics in this list, I will share with you a solution to the system I described.

If you do not understand some parts of it, you can go back to the relevant sections. And if that is not enough, visit the links to the documentation that I have provided.

Are you up for a challenge? I have selected a few questions from old GCP Professional Certification exams. They will test your understanding of the concepts explained in this article.
I recommend trying to solve both the design and the questions *on your own*, going back to the guide if necessary. Once you have an answer, compare it to the proposed solution.
Try to go beyond what you are reading and ask yourself what would happen if requirement X changed:

- Batch vs streaming data
- Regional vs global solution
- A small number of users vs huge volume of users
- Latency is not a problem vs real-time applications

And any other scenarios you can think of.

At the end of the day, you are not paid just for what you know but for your thought process and the decisions you make. That is why it is vitally important that you exercise this skill.

At the end of the article, I'll provide more resources and next steps if you want to continue learning about GCP.

How to get started with Google Cloud Platform for free

GCP currently offers a 3 month free trial with $300 US dollars of free credit. You can use it to get started, play around with GCP, and run experiments to decide if it is the right option for you.

You will NOT be charged at the end of your trial. You will be notified and your services will stop running unless you decide to upgrade your plan.
I strongly recommend using this trial to practice. To learn, you have to **try things on your own**, face problems, break things, and fix them. It doesn't matter how good this guide is (or the official documentation for that matter) if you do not try things out.

Why would you migrate your services to Google Cloud Platform?

Consuming resources from GCP, like storage or computing power, provides the following benefits:

- No need to spend a lot of money upfront for hardware
- No need to upgrade your hardware and migrate your data and services every few years
- Ability to scale to adjust to the demand, paying only for the resources you consume
- Create proof of concepts quickly since provisioning resources can be done very fast
- Secure and manage your APIs
- Not just infrastructure: data analytics and machine learning services are available in GCP

GCP makes it easy to experiment and use the resources you need in an economical way.

How to optimize your VMs to reduce costs in GCP

In general, you will only be **charged for the time your instances are running**. Google will not charge you for stopped instances. However, if they consume resources, like disks or reserved IPs, you might incur charges.
Here are some ways you can optimize the cost of running your applications in GCP.

Custom Machine Types

GCP provides different machine families with predefined amounts of RAM and CPUs:

- **General-purpose**. Offers the best price-performance ratio for a variety of workloads.
- **Memory-optimized**. Ideal for memory-intensive workloads. They offer more memory per core than other machine types.
- **Compute-optimized**. They offer the highest performance per core and and are optimized for compute-intensive workloads
- **Shared-core**. These machine types timeshare a physical core. This can be a cost-effective method for running small applications. Besides, you can create your custom machine with the amount of RAM and CPUs you need.

Preemptible VM's

You can use preemptible virtual machines to save up to 80% of your costs. They are ideal **for fault-tolerant, non-critical applications**. You can save the progress of your job in a persistent disk using a shut-down script to continue where you left off. Google may stop your instances at any time (with a 30-second warning) and will always stop them after 24 hours.

To reduce the chances of getting your VMs shut down, Google recommends:

- Using **many small instances** and
- Running your jobs during **off-peak times**.
  Note: Start-up and shut-down scripts apply to non-preemptible VMS as well. You can use them the control the behavior of your machine when it starts or stops. For instance, to install software, download data, or backup logs.
  There are two options to define these scripts:

- When you are creating your instance in the Google Console, there is a field to paste your code.

- Using the metadata server URL to point your instance to a script stored in Google Cloud Storage.

  This latter is preferred because it is easier to create many instances and to manage the script.

Sustained Use Discounts

The longer you use your virtual machines (and Cloud SQL instances), the higher the discount - up to 30%. Google does this automatically for you.

Committed Use Discounts

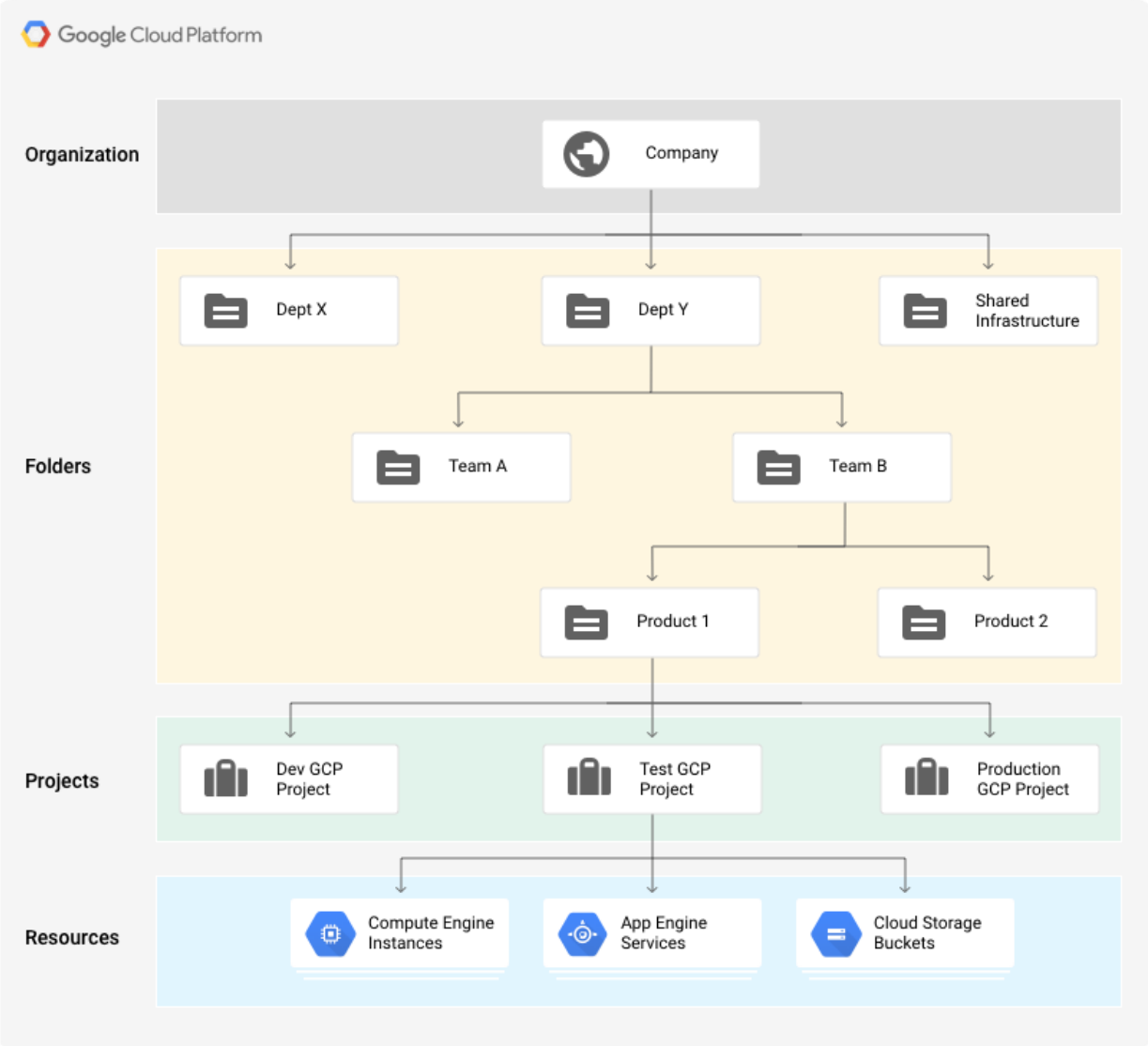
You can get up to 57% discount if you commit to a certain amount of CPU and RAM resources for a period of 1 to 3 years.

To estimate your costs, use the Price Calculator. This helps prevent any surprises with your bills and create budget alerts.

How to manage resources in GCP

In this section, I will explain how you can manage and administer your Google Cloud resources.

Resource Hierarchy



There are four types of resources that can be managed through Resource Manager:

- **The organization resource**. It is the root node in the resource hierarchy. It represents an organization, for example, a company.
- **The projects resource**. For example, to separate projects for production and development environments. They are **required** to create resources.
- **The folder resource**. They provide an extra level of project isolation. For example, creating a folder for each department in a company.
- **Resources**. Virtual machines, database instances, load balancers, and so on.

There are **quotas** that limit the maximum number of resources you can create to prevent unexpected spikes in billing. However, most quotas can be increased by opening a support ticket.
Resources in GCP follow a **hierarchy** via a parent/child relationship, similar to a traditional file system, where:

- **Permissions are inherited as we descend the hierarchy**. For example, permissions granted and the organization level will be propagated to all the folders and projects.
- More permissive parent policies always overrule more restrictive child policies.

This hierarchical organization helps you manage common aspects of your resources, such as access control and configuration settings.

You can create super admin accounts that have access to every resource in your organization. Since they are very powerful, make sure you follow Google's best practices.

Labels

Labels are key-value pairs you can use to organize your resources in GCP. Once you attach a label to a resource (for instance, to a virtual machine), you can filter based on that label. This is useful also to break down your bills by labels.

Some common use cases:

- Environments: prod, test, and so on.
- Team or product owners
- Components: backend, frontend, and so on.
- Resource state: active, archive, and so on.

Labels vs Network tags

These two similar concepts seem to generate some confusion. I have summarized the differences in this table:

LABELS	NETWORK TAGS
Applied to any GCP	Applied only for VPC resources

LABELS	NETWORK TAGS
resource	
Just organize resources	Affect how resources work (ex: through application of firewall rules)

Cloud IAM

Simply put, Cloud IAM controls **who can do what on which resource**. A resource can be a virtual machine, a database instance, a user, and so on.
It is important to notice that permissions are not directly assigned to users. Instead, they are bundled into *roles*, which are assigned to *members*. A *policy* is a collection of one or more bindings of a set of members to a role.

Identities

In a GCP project, identities are represented by Google accounts, created outside of GCP, and defined by an email address (not necessarily @gmail.com). There are different types:

- **Google accounts***. To represent people: engineers, administrators, and so on.
- **Service accounts**. Used to identify non-human users: applications, services, virtual machines, and others. The authentication process is defined by *account keys*, which can be managed by Google or by users (only for user-created service accounts).
- **Google Groups** are a collection of Google and service accounts.
- **G Suite Domain*** is the type of account you can use to identify organizations. If your organization is already using Active Directory, it can be synchronized with Cloud IAM using Cloud Identity.
- **allAuthenticatedUsers**. To represent any authenticated user in GCP.
- **allUsers**. To represent anyone, authenticated or not.

Regarding service accounts, some of Google's best practices include:

- Not using the default service account
- Applying the Principle of Least Privilege. For instance:
  1. Restrict who can act as a service account
  2. Grant only the minimum set of permissions that the account needs
  3. Create service accounts for each service, only with the permissions the account needs

Roles

A role is a **collection of permissions**. There are three types of roles:

- **Primitive**. Original GCP roles that apply to the entire project. There are three concentric roles: **Viewer**, **Editor**, and **Owner**. Editor contains Viewer and Owner contains Editor.
- **Predefined**. Provides access to specific services, for example, storage.admin.
- **Custom**. lets you create your own roles, combining the specific permissions you need.

When assigning roles, follow the principle of least privilege, too. In general, **prefer predefined over primitive roles**.

Cloud Deployment Manager

Cloud Deployment Manager automates repeatable tasks like provisioning, configuration, and deployments for any number of machines.

It is Google's *Infrastructure as Code* service, similar to Terraform - although you can deploy only GCP resources. It is used by GCP Marketplace to create pre-configured deployments.
You define your configuration in YAML files, listing the resources (created through API calls) you want to create and their properties.
Resources are defined by their **name** (VM-1, disk-1), **type** (compute.v1.disk, compute.v1.instance) and **properties** (zone:europe-west4, boot:false).
To increase performance, resources are deployed in parallel. Therefore you need to **specify any dependencies using references.** For instance, do not create virtual machine VM-1 until

the persistent disk disk-1 has been created. In contrast, Terraform would figure out the dependencies on its own.
You can modularize your configuration files using templates so that they can be independently updated and shared. Templates can be defined in Python or Jinja2. The contents of your templates will be inlined in the configuration file that references them.

Deployment Manager will create a manifest containing your original configuration, any templates you have imported, and the expanded list of all the resources you want to create.

Cloud Operations (formerly Stackdriver)

Operations provide a set of tools for monitoring, logging, debugging, error reporting, profiling, and tracing of resources in GCP (AWS and even on-premise).

Cloud Logging

Cloud Logging is GCP's centralized solution for real-time log management. For each of your projects, it allows you to store, search, analyze, monitor, and alert on logging data:

- By default, data will be stored for a certain period of time. The retention period varies depending on the type of log. You cannot retrieve logs after they have passed this retention period.
- Logs can be exported for different purposes. To do this, you create a **sink**, which is composed of a **filter** (to select what you want to log) and a **destination**: Google Cloud Storage (GCS) for long term

retention, BigQuery for analytics, or Pub/Sub to stream it into other applications.

- You can create log-based metrics in Cloud Monitoring and even get alerted when something goes wrong.

Logs are a named collection of **log entries**. Log entries record status or events and includes the name of its log, for example, compute.googleapis.com/activity. There are two main types of logs:
First, User Logs:

- These are generated by your applications and services.
- They are written to Cloud Logging using the Cloud Logging API, client libraries, or logging agents installed on your virtual machines.
- They stream logs from common third-party applications like MySQL, MongoDB, or Tomcat.

Second, Security logs, divided into:

- Audit logs, for administrative changes, system events, and data access to your resources. For example, who created a particular database instance or to log a live migration. Data access logs must be explicitly enabled and may incur additional charges. The rest are enabled by default, cannot be disabled, and are free of charges.
- Access Transparency logs, for actions taken by Google staff when they access your resources for example to investigate an issue you reported to the support team.

VPC Flow Logs

They are specific to VPC networks (which I will introduce later).
VPC flow logs record a **sample of network flows** sent from and received by VM instances, which can be later access in Cloud Logging.
They can be used to monitor network performance, usage, forensics, real-time security analysis, and expense optimization.

Note: you may want to log your billing data for analysis. In this case, you do *not* create a sink. You can directly export your reports to BigQuery.

Cloud Monitoring

Cloud Monitoring lets you monitor the performance of your applications and infrastructure, visualize it in dashboards, create uptime checks to detect resources that are down and alert you based on these checks so that you can fix problems in your environment. You can monitor resources in GCP, AWS, and even on-premise.
It is recommended to create a separate project for Cloud Monitoring since it can keep track of resources across multiple projects.

Also, it is recommended to install a monitoring agent in your virtual machines to send application metrics (including many third-party applications) to Cloud Monitoring. Otherwise, Cloud Monitoring will only display CPU, disk traffic, network traffic, and uptime metrics.

Alerts

To receive alerts, you must declare an **alerting policy**. An alerting policy defines the **conditions** under which a service is considered unhealthy. When the conditions are met, a new incident will be created and notifications will be sent (via email, Slack, SMS, PagerDuty, etc).
A policy belongs to an individual workspace, which can contain a maximum of 500 policies.

Trace

Trace helps **find bottlenecks in your services**. You can use this service to figure out how long it takes to handle a request, which microservice takes the longest to respond, where to focus to reduce the overall latency, and so on.
It is enabled by default for applications running on Google App Engine (GAE) - Standard environment - but can be used for applications running on GCE, GKE, and Google App Engine Flexible.

Error Reporting

Error Reporting will aggregate and display errors produced in services written in Go, Java, Node.js, PHP, Python, Ruby, or .NET. running on GCE, GKE, GAP, Cloud Functions, or Cloud Run.

Debug

Debug lets you inspect the application's state without stopping your service. Currently supported for Java, Go, Node.js and Python. It is automatically integrated with GAE but can be used on GCE, GKE, and Cloud Run.

Profile

Profiler that continuously gathers CPU usage and memory-allocation information from your applications. To use it, you need to install a profiling agent.

How to store data in GCP

In this section, I will cover both Google Cloud Storage (for any type of data, including files, images, video, and so on), the different database services available in GCP, and how to decide which storage option works best for you.

Google Cloud Storage (GCS)

GCS is Google's **storage service for unstructured data**: pictures, videos, files, scripts, database backups, and so on.
Objects are placed in **buckets**, from which they inherit permissions and storage classes.
Storage classes provide different SLAs for storing your data to minimize costs for your use case. A bucket's storage class can be changed (under some restrictions), but it will affect new objects added to the bucket only.
In addition to Google's console, you can interact with GCS from your command line, using gsutil. You can use specify:

- **Multithreaded updates** when you need to upload a large number of small files. The command looks like gsutil -m cp files gs://my-bucket)

- **Parallel updates** when you need to upload large files. For more details and restrictions, visit this link.
  Another option to upload files to GCS is Storage Transfer Service (STS), a service that imports data to a GCS bucket from:
- An AWS S3 bucket
- A resource that can be accessed through HTTP(S)
- Another Google Cloud Storage bucket

  If you need to upload huge amounts of data (from hundreds of terabytes up to one petabyte) consider Data Transfer Appliance: ship your data to a Google facility. Once they have uploaded the data to GCS, the process of data rehydration reconstitutes the files so that they can be accessed again.

Object lifecycle management

You can define rules that determine what will happen to an object (will it be archived or deleted) when a certain condition is met.

For example, you could define a policy to automatically change the storage class of an object from Standard to Nearline after 30 days and to delete it after 180 days.

This is the way a rule can be defined:

```
{
    "lifecycle":{
        "rule":[
            {
                "action":{
                    "type":"Delete"
                },
                "condition":{
                    "age":30,
                    "isLive":true
                }
            },
            {
                "action":{
```

```
                "type":"Delete"
            },
            "condition":{
                "numNewerVersions":2
            }
        },
        {
            "action":{
                "type":"Delete"
            },
            "condition":{
                "age":180,
                "isLive":false
            }
        }
    ]
  }
}
```

It will be applied through gsutils or a REST API call. Rules can be created also through the Google Console.

Permissions in GCS

In addition to IAM roles, you can use Access Control Lists (ACLs) to manage access to the resources in a bucket.

Use IAM roles when possible, but remember that **ACLs** grant access to buckets and **individual objects**, while **IAM roles are project or bucket wide** permissions. Both methods work in tandem.
To grant temporary access to users outside of GCP, use Signed URLs.

Bucket lock

Bucket locks allow you to enforce a **minimum retention period** for objects in a bucket. You may need this for auditing or legal reasons.
Once a bucket is locked, it cannot be unlocked. To remove, you need to first remove all objects in the bucket, which you can only

do after they all have reached the retention period specified by the retention policy. Only then, you can delete the bucket.
You can include the retention policy when you are creating the bucket or add a retention policy to an existing bucket (it retroactively applies to existing objects in the bucket too).

Fun fact: the maximum retention period is 100 years.

Relational Managed Databases in GCP

Cloud SQL and Cloud Spanner are two managed database services available in GCP. If you do not want to deal with all the work necessary to maintain a database online, they are a great option. You can always spin a virtual machine and manage your own database.

Cloud SQL

Cloud SQL provides access to a managed MySQL or PostgreSQL database instance in GCP. Each instance is limited to a **single region** and has a **maximum capacity of 30 TB**.
Google will take care of the installation, backups, scaling, monitoring, failover, and read replicas. For availability reasons, replicas must be defined in the same region but a different zone from the primary instances.

Data can be easily imported (first uploading the data to Google Cloud Storage and then to the instance) and exported using SQL dumps or CSV files format. Data can be compressed to reduce costs (you can directly import .gz files). For "lift and shift" migrations, this is a great option.

If you need global availability or more capacity, consider using Cloud Spanner.

Cloud Spanner

Cloud Spanner is globally available and can scale (horizontally) very well.

These two features make it capable of supporting different use cases than Cloud SQL and more expensive too. Cloud Spanner is not an option for lift and shift migrations.

NoSQL Managed Databases in GCP

Similarly, GCP provides two managed NoSQL databases, Bigtable and Datastore, as well as an in-memory database service, Memorystore.

Datastore

Datastore is a completely no-ops, highly-scalable document database ideal for web and mobile applications: game states, product catalogs, real-time inventory, and so on. It's great for:

- User profiles - mobile apps
- Game save states

By default, Datastore has a built-in **index** that improves performance on simple queries. You can create your own indices, called **composite indexes**, defined in YAML format.
If you need extreme throughput (huge number of reads/writes per second), use Bigtable instead.

Bigtable

Bigtable is a NoSQL database ideal for analytical workloads where you can expect a very high volume of writes, reads in the milliseconds, and the ability to store terabytes to petabytes of information. It's great for:

- Financial analysis
- IoT data
- Marketing data

Bigtable requires the creation and configuration of your nodes (as opposed to the fully-managed Datastore or BigQuery). You can add

or remove nodes to your cluster with zero downtime. The simplest way to interact with Bigtable is the command-line tool cbt. Bigtable's performance will depend on the design of your database schema.

- You can only define one key per row and must keep all the information associated with an entity in the same row. Think of it as a hash table.
- Tables are sparse: if there is no information associated with a column, no space is required.
- To make reads more efficient, try to store related entities in adjacent rows.

  Since this topic is worth an article on its own, I recommend you read the documentation.

Memorystore

It provides a managed version of Redis and Memcache (in-memory databases), resulting in very fast performance. Instances are regional, like Cloud SQL, and have a capacity of up to 300 GB.

How to choose your database

Google loves decision trees. This one will help you choose the right database your your projects. For unstructured data consider GCS or process it using Dataflow (discussed later).

How does networking work in GCP?

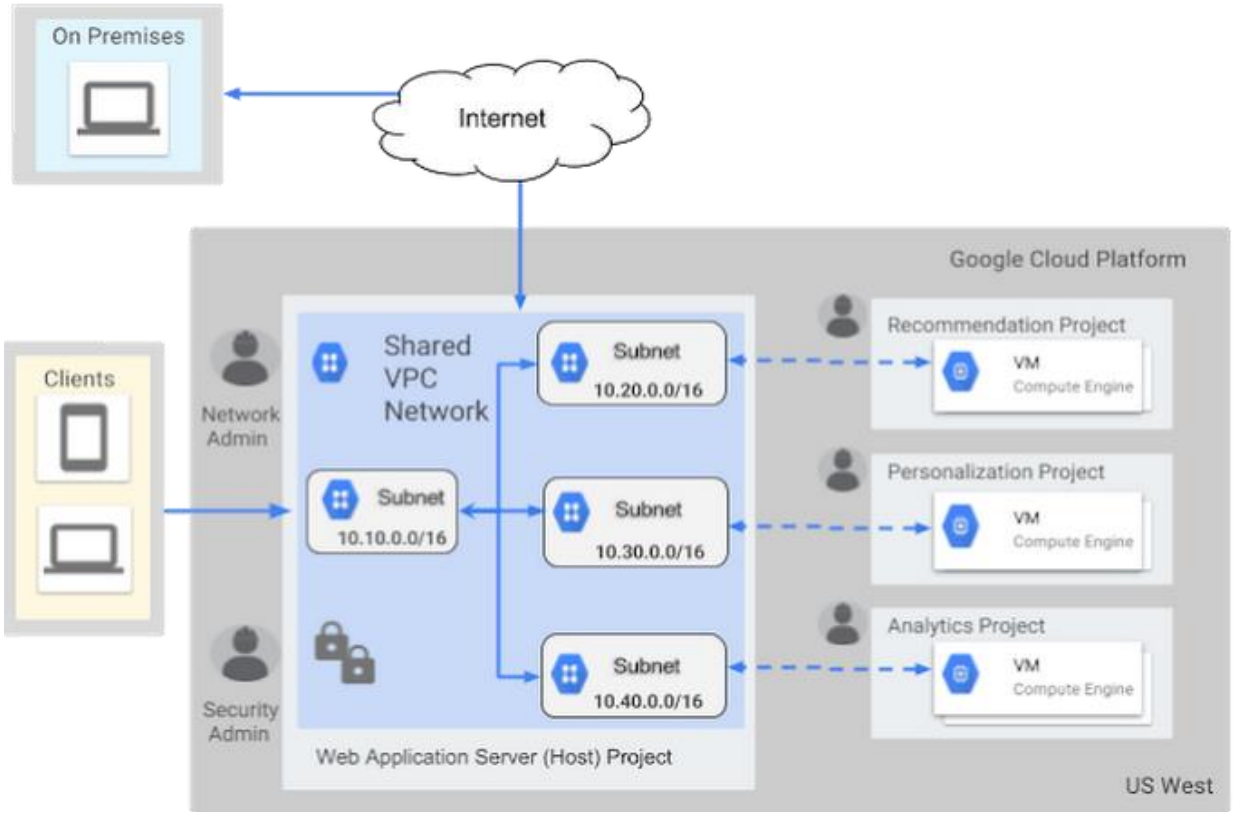
Virtual Private Cloud (VPC) - see the docs here

You can use the same network infrastructure that Google uses to run its services: YouTube, Search, Maps, Gmail, Drive, and so on.

Google infrastructure is divided into:

- **Regions**: Independent geographical areas, at least 100 miles apart from each other, where Google hosts datacenters. It consists of 3 or more zones. For example, us-central1.
- **Zones**: Multiple individual datacenters within a region. For example, us-central1-a.
- **Edge Points of Presence**: points of connection between Google's network and the rest of the internet.
  GCP infrastructure is designed in a way that all traffic between regions travels through a global private network, resulting in better security and performance.

  On top of this infrastructure, you can build networks for your resources, Virtual Private Clouds. They are **software-defined networks**, where all the traditional network concepts apply:
- Subnets. Logical partitions of a network defined using CIDR notation. They belong to one region only but can span multiple zones. If you have multiple subnets (including your on-premise networks if they are connected to GCP), make sure the CIDR ranges do not overlap.
- **IP addresses**. Can be internal (for private communication within GCP) or external (to communicate with the rest of the internet). For external IP addresses, you can use an **ephemeral IP** or pay for a **static IP**. In general, you need an external IP address to connect to GCP services. However, in some cases, you can configure private access for instances that only have an internal IP.
- Firewalls rules, to allow or deny traffic to your virtual machines, both incoming (ingress) and outgoing (egress). By default, all ingress traffic is denied and all egress traffic is allowed. Firewall rules are defined at the VPC level but they **apply to individual instances or groups of instances** using **network tags** or **IP ranges**.
  Common issue: If you know your VMs are working correctly but you cannot access them through HTTP(s) or cannot SSH into them, have a look at your firewall rules.



You can create **hybrid networks** connecting your on-premise infrastructure to your VPC.
When you create a project, a **default network** will be created with subnets in each region (auto mode). You can delete this network, but you need to create at least one network to be able to create virtual machines.
You can also create your **custom networks**, where no subnets are created by default and you have full control over subnet creation (custom mode).
The main goal of a VPC is the **separation of network resource**s. A GCP project is a way to organize resources and manage permissions.
Users of project A need permissions to access resources in project B. All users can access any VPC defined in any project to which they belong. Within the same VPC, resources in subnet 1 need to be granted access to resources in subnet 2.

In terms of IAM roles, there is a distinction between who can create network resources (Network admin, to create subnets, virtual machines, and so on) and who is responsible for the security of the

resources (Security Admin, to create firewall rules, SSL certificates, and so on).

The Compute Instance Admin role combines both roles.

As usual, there are quotas and limits to what you can do in a VPC, amongst them:

- The maximum number of VPCs in a project.
- The maximum number of virtual machines per VPC.
- No broadcast or multicast.
- VPCs cannot use IPv6 to communicate internally, although global load balancers support IPv6 traffic.

How to share resources between multiple VPCs

Shared VPC

Shared VPCs are a way to share resources between different projects within the same organization. This allows you to control billing and manage access to the resources in different projects, following the principle of least privilege. Otherwise you'd have to put all the resources in a single project.

To design a shared VPC, projects fall under three categories:

- **Host project**. It is the project that hosts the common resources. There can only be one host project.
- **Service project**: Projects that can access the resources in the host project. A project cannot be both host and service.
- **Standalone project**. Any project that does not make use of the shared VPC.

You will only be able to communicate between resources created **after** you define your host and service projects. Any existing resources before this will not be part of the shared VPC.

VPC Network Peering

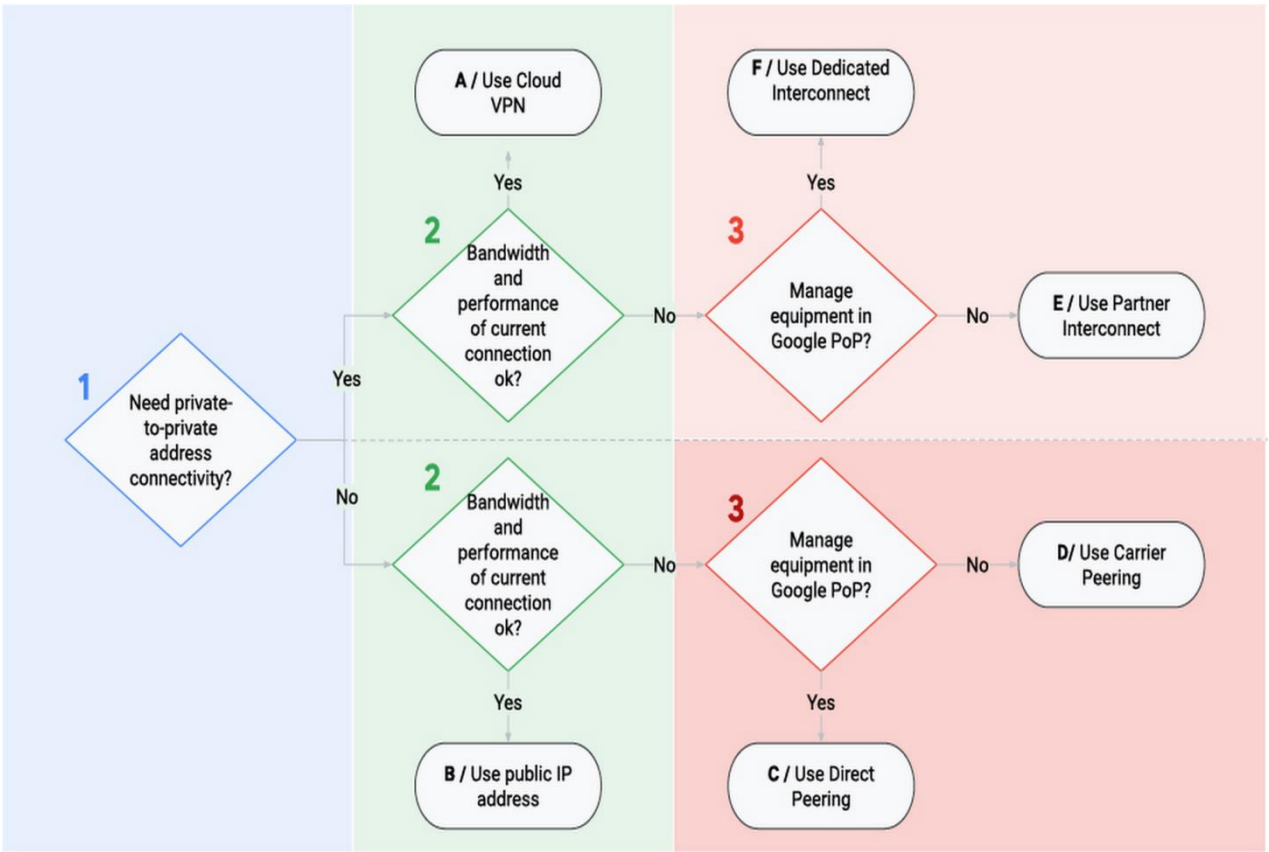
Shared VPCs can be used when all the projects belong to the same organization. However, if:

- You need **private communication** across VPCs.
- The VPCs are in projects that may belong to **different organizations.**
- Want **decentralized** control, that is, no need to define host projects, server projects, and so on.
- Want to reuse existing resources.

VPC Network peering is the right solution.

In the next section, I will discuss how to connect your VPC(s) with networks outside of GCP.

How to connect on-premise and GCP infrastructures



There are three options to connect your on-premise infrastructure to GCP:

- Cloud VPN

- Cloud Interconnect
- Cloud Peering

Each of them with different capabilities, use cases, and prices that I will describe in the following sections.

Cloud VPN

With Cloud VPN, your traffic travels through the public internet over an encrypted tunnel. Each tunnel has a maximum capacity of 3 Gb per second and you can use a maximum of 8 for better performance. These two characteristics make VPN the cheapest option.

You can define two types of routes between your VPC and your on-premise networks:

- **Static routes**. You have to manually define and update them, for example when you add a new subnet. This is not the preferred option.
- **Dynamic routes**. Routes are automatically handled (defined and updated) for you using Cloud Router. This is the preferred option when BGP is available.
  Your traffic gets encrypted and decrypted by VPN Gateways (in GCP, they are regional resources).

  To have a more robust connection, consider using multiple VPN gateways and tunnels. In case of failure, this redundancy guarantees that traffic will still flow.

Cloud Interconnect

With Cloud VPN, traffic travels through the public internet. With Cloud Interconnect, there is a **direct physical connection** between your on-premises network and your VPC. This option will be more expensive but will provide the best performance.
There are two types of interconnect available, depending on how you want your connection to GCP to materialize:

- **Dedicated interconnect**. There is "a direct cable" connecting your infrastructure and GCP. This is the fastest option, with a capacity of 10 to 200 Gb per second. However, it is not available everywhere: at the time of this writing, only in 62 locations in the world.
- **Partner interconnect**. You connect through a service provider. This option is more geographically available, but the not as fast as a dedicated interconnects: from 50 Mb per second to 10 Gb per second.

Cloud Peering

Cloud peering is not a GCP service, but you can use it to connect your network to Google's network and access services like Youtube, Drive, or GCP services.

A common use case is when you need to connect to Google but don't want to do it over the public internet.

Other networking services

Load Balancers (LB)

In GCP, load balancers are pieces of software that distribute user requests among a group of instances.

A load balancer may have multiple backends associated with it, having rules to decide the appropriate backend for a given request.

There are different types of load balancers. They differ in the type of traffic (HTTP vs TCP/UDP - Layer 7 or Layer 4), whether they handle external or internal traffic, and whether their scope is regional or global:

- **HTTP(s)**. Global LB that handles HTTP(s) requests, distributing traffic to multiple regions based on user location (to the closest region with available instances) or URL maps (the LB can be configured to forward requests to *URL/news* to a backend service and *URL/videos* to a different one). It can receive both IPv4 and IPv6 traffic (but this one is terminated at the LB level and proxied as IPv4 to the backends) and has native support for WebSockets.

- **SSL Proxy LB**. **Global** LB that handles encrypted TCP traffic, managing SSL certificates for you.
- **TCP Proxy LB**. **Global** LB that handles unencrypted TCP traffic. Like SSL Proxy LB, by default, it will not preserve the client's IP, but this can be changed.
- **Network Load Balancer**. Regional LB that handles TCP/UDP external traffic, based on IP address and port.
- **Internal Load Balancer**. Like a Network LB, but for internal traffic.
  For the visual learners:

Cloud DNS

Cloud DNS is Google's managed Domain Name System (DNS) host, both for internal and external (public) traffic. It will map URLs like https://www.freecodecamp.org/ to an IP address. It is the only service in GCP with 100% SLA - it is available 100% of the time.

Google Cloud CDN

Cloud DNS is Google's Content Delivery Network. If you have data that does not change often (images, videos, CSS, etc.) it makes sense to cache it close to your users. Cloud CDN provides 90 Edges Point of Presence (POP) to cache the data close to your end-users. After the first request, static data can be stored in a POP, usually much closer to your user than your main servers. Thus, in subsequent requests, you can retrieve the data faster from the POP and reduce the load on your backend servers.

Where can you run your applications in GCP?

I will present 4 places where your code can run in GCP:

- Google Compute Engine
- Google Kubernetes Engine

- App Engine
- Cloud Functions

Note: there is a 5th option: Firebase is Google's mobile platform that helps you quickly develop apps.

Compute Engine (GCE)

Compute engine allows you to spin up virtual machines in GCP. This section will be longer since GCE provides the infrastructure where GKE and GAE run.

In the introduction, I talked about the different types of VMs you can create in GCE. Now, I will cover where to store the data, how to back it up, and how to create instances with all the data and configuration you need.

Where to store your VM's data: disks

Your data can be stored in **Persistent disks**, **Local SSDs**, or in **Cloud Storage**.

Persistent Disk

Persistent disks provide durable and reliable block storage. They are not local to the machine. Rather, they are networked attached, which has its pros and cons:

- Disks can be resized, attached, or detached from a VM even if the instance is in use.
- They have high reliability.
- Disks can survive the instance after its deletion.

- If you need more space, simply attach more disks.
- Larger disks will provide higher performance.
- Being networked attached, they are less performant than local options. SSD persistent disks are also available for more demanding workloads.

  Every instance will need one boot disk and it must be of this type.

Local SSD

Local SSDs are attached to a VM to which they provide high-performance ephemeral storage. As of now, you can attach up to eight 375GB local SSDs to the same instance. However, this data will be lost if the VM is killed.

Local SSDs can only be attached to a machine when it is created, but you can attach both local SSDs and persistent disks to the same machine.

Both types of disks are zonal resources.

Cloud Storage

We have extensively covered GCS in a previous section. GCS is not a filesystem, but you can use GCS-Fuse to mount GCS buckets as filesystems in Linux or macOS systems. You can also let apps download and upload data to GCS using standard filesystem semantics.

How to back up your VM's data: Snapshots

Snapshots are backups of your disks. To reduce space, they are created incrementally:

- Back up 1 contains all your disk content
- Back up 2 only contains the data that has changed since back up 1
- Back up 3 only contains the data that has changed since back up 2, and so on

This is enough to restore the state of your disk.

Even though snapshots can be taken without stopping the instance, it is best practice to at least reduce its activity, stop writing data to disk, and flush buffers. This helps you make sure you get an accurate representation of the content of the disk.

Images

Images refer to the operating system images needed to create boot disks for your instances. There are two types of images:

- **Public images**. They are provided and maintained by Google, open-source communities, and third-party vendors. Ready for you to use as soon as you create your project. Available to anyone
- **Custom images**. Images that you have created.
- They are linked to the project in which you created them but you can share them with other projects.
- You can create images from **persistent disks** and **other images**, both from the same project or shared from another project.
- Related images can be grouped in **image families** to simplify the management of the different image versions.
- For Linux-based images, you can share them also by exporting them to Cloud Storage as a tar.gz file.

You might be asking yourself what is the difference between an image and a snapshot. Mainly, **their purpose**. Snapshots are taken as incremental backups of a disk while images are created to spin up new virtual machines and configure instance templates.

Note on images vs startup scripts:

For simple setups, startup scripts are also an option. They can be used to test changes quickly, but the VMs will take longer to be ready compared to using an image where all the needed software is installed, configured, and so on.

Instance groups

Instance groups let you treat a group of instances as a single unit and they come in two flavors:

- **Unmanaged instance group**. Formed by a heterogeneous group of instances that required individual configuration settings.
- **Managed instance group (MIG).** This is the preferred option when possible. All the machines look the same, making it easy to configure them, create them in multiple zones (high availability), replace them if they become unhealthy (auto-healing), balance the traffic among them, and create new instances if they traffic increases (horizontal scaling).
  To create a MIGs, you need to define an **instance template**, specifying your machine type, zone, OS image, startup and shutdown scripts, and so on. Instance templates are immutable.
  To update a MIG, you need to create a new template and use the **Managed Instance Group Updated** to deploy the new version to every machine in the group.
  This functionality can be used to create canary tests, deploying your changes to a small fraction of your machines first.
  Visit this link to know more about Google's recommendations to ensure an application deployed via a managed instance group can handle the load even if an entire zone fails.

Security best practices for GCE

To increase the security of your infrastructure in GCE, have a look at:

- Shielded VMs
- Prevent instances from being reached from the public internet
- Trusted images to make sure your users can only create disks from images in **specific projects**

App Engine

App Engine is a great choice when you want to focus on the code and let Google handle your infrastructure. You just need to choose the region where your app will be deployed (this cannot be changed once it is set). Amongst its main use cases are websites, mobile apps, and game backends.

You can easily update the version of your app that is running via the command line or the Google Console.

Also, if you need to deploy a risky update to your application, you can split the traffic between the old and the risky versions for a canary deployment. Once you are happy with the results, you can route all the traffic to the new version.

There are two App Engine environments:

- **Standard**. This version can quickly scale up or down (even to zero instances) to adjust to the demand. Currently, only a few programming languages are supported (Go, Java, PHP, and Python) and you do not have access to a VPC (including VPN connections). It can be scaled down to zero instances.
- **Flexible.** Your code runs in Docker containers in GCE, hence more flexible than the Standard environment. However, creating new instances is slower and it cannot be scaled down to zero instances. It is suited for more consistent traffic.
  Regardless of the environment, there are no up-front costs and you only pay for what you use (billed per second).

Memcache is a built-in App Engine, giving you the possibility to choose between a **shared** cache (default, free option) or a **dedicated** cache for better performance.
Visit this link to know more about the best practices you should follow to maximize the performance of your app.

Google Kubernetes Engine (GKE)

Kubernetes is an open-source **container orchestration system**, developed by Google.
Kubernetes is a very extensive topic in itself and I will not cover here. You just need to know that GKE makes it easy to run and manage your Kubernetes clusters on GCP.

Google also provides Container Registry to store your container images - think of it as your private Docker Hub.

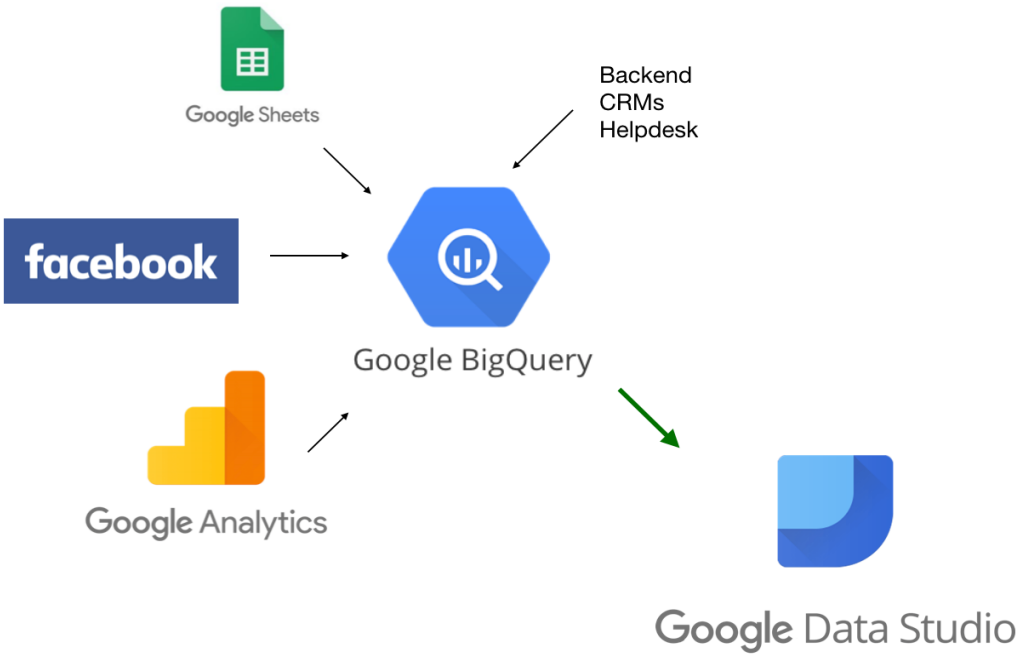
Note: You can use Cloud Build to run your builds in GCP and, among other things, produce Docker images and store them in Container Registry. Cloud Build can import your code from Google Cloud Storage, Cloud Source Repository, GitHub, or Bitbucket.

Cloud Functions

Cloud Functions are the equivalent of Lambda functions in AWS. Cloud functions are **serverless**. They let you focus on the code and not worry about the infrastructure where it is going to run.
With Cloud Functions it is **easy to respond to events** such as uploads to a GCS bucket or messages in a Pub/Sub topic. You are only charged for the time your function is running in response to an event.

How to work with Big Data in GCP

BigQuery



BigQuery is Google's serverless data warehousing and provides analytics capabilities for petabyte-scale databases.

BigQuery automatically backs up your tables, but you can always export them to GCS to be on the safe side - incurring extra costs.

Data can be ingested in batches (for instance, from a GCS bucket) or from a stream in multiple formats: CSV, JSON, Parquet, or Avro (most performant). Also, you can query data that resides in external sources, called federated sources, for example, GCS buckets.

You can interact with your data in BigQuery using SQL via the

- Google Console.
- Command-line, running commands like `bq query 'SELECT field FROM ....`
- REST API.
- Code using client libraries.

User-Defined Functions allow you to combine SQL queries with JavaScript functions to create complex operations.
BigQuery is a columnar data store: records are stored in columns. Tables are collections of columns and datasets are collections of tables.

Jobs are actions to load, export, query, or copy data that BigQuery runs on your behalf.

Views are virtual tables defined by a SQL query and are useful sharing data with others when you want to control exactly what they have access to.

Two important concepts related to tables are:

- **Partitioned tables**. To limit the amount of data that needs to be queried, tables can be divided into partitions. This can be done based on ingest time or including a timestamp or date column or an integer range. This way it is easy to query for certain periods without querying the full table. To reduce costs, you can define an expiration period after which the partition will be deleted.
- **Clustered tables**. Data are clustered by column (for instance, order_id). When you query your table, only the rows associated

with this column will be read. BigQuery will perform this clustering automatically based on one or more columns.
Using IAM roles, you can control access at a project, dataset, or view level, but *not at the table level*. Roles are complex for BigQuery, so I recommend checking the documentation.
For instance, the jobUser role only lets you run jobs while the user role lets you run jobs and create datasets (but not tables).

Your costs depend on how much data you store and stream into BigQuery and how much data you query. To reduce costs, BigQuery automatically caches previous queries (per user). This behavior can be disabled.

When you don't edit data for 90 days, it automatically moves to a cheaper storage class. You pay for what you use, but it is possible to opt for a flat rate (only if you need more than the 2000 slots that are allocated by default).
Check these links to see how to optimize your performance and costs.

Cloud Pub/Sub

Pub/Sub is Google's **fully-managed message queue**, allowing you to decouple publishers (adding messages to the queue) and subscribers (consuming messages from the queue).
Although it is similar to Kafka, Pub/Sub is not a direct substitute. They can be combined in the same pipeline (Kafka deployed on-premise or even in GKE). There are open-source plugins to connect Kafka to GCP, like Kafka Connect.
Pub/Sub guarantees that every message will be delivered at least once but it does not guarantee that messages will be processed in order. It is usually connected to Dataflow to process the data, ensure that the messages are processed in order, and so on.

Pub/Sub support both push and pull modes:

- **Push.** Messages are sent to subscribers, resulting in lower latency.

- **Pull.** Subscribers pull messages from topics, better suited for a large volume of messages.

Cloud Pub/Sub vs Cloud Task

Cloud Tasks is another fully-managed service to execute tasks asynchronously and manage messages between services. However, there are differences between Cloud Tasks and Pub/Sub:

- In Pub/Sub, publishers and subscribers are decoupled. Publishers know nothing about their subscribers. When they publish a message, they implicitly cause one or multiple subscribers to react to a publishing event.
- In Cloud Tasks, the publisher stays in control of the execution. Besides, Cloud Tasks provide other features unavailable for Pub/Sub like scheduling specific delivery times, delivery rate controls, configurable retries, access and management of individual tasks in a queue, task/message creation deduplication.

For more details, check out this link.

Cloud Dataflow

Cloud Dataflow is Google's managed service for **stream and batch data processing**, based on Apache Beam.
You can define pipelines that will transform your data, for example before it is ingested in another service like BigQuery, BigTable, or Cloud ML. The same pipeline can process both stream and batch data.

A common pattern is to stream data into Pub/Sub, let's say from IoT devices, process it in Dataflow, and store it for analysis in BigQuery.
But Pub/Sub does not guarantee that the order in which messages are pushed to the topics will be the order in which the messages are consumed. However, this can be done with Dataflow.

Cloud Dataproc

Cloud Dataproc is Google's managed the Hadoop and Spark ecosystem. It lets you create and manage your clusters easily and turn them off when you are not using them, to reduce costs.

Dataproc can only be used to process batch data, while Dataflow can handle also streaming data.

Google recommends using Dataproc for a lift and leverage migration of your on-premise Hadoop clusters to the cloud:

- Reduce costs turning your cluster off when you are not using it.
- Leverage Google's infrastructure
- Use some preemptible virtual machines to reduce costs
- Add larger (SSD) persistent disks to improve performance
- BigQuery can replace Hive and BigTable can replace HBase
- Cloud Storage replaces HDFS. Just upload your data to GCS and change the prefixes hdfs:// to gs://

  Otherwise, you should choose Cloud Dataflow.

Dataprep

Cloud Dataprep provides you with a **web-based interface to clean and prepare your data** before processing. The input and output formats include, among others, CSV, JSON, and Avro.
After defining the transformations, a Dataflow job will run. The transformed data can be exported to GCS, BigQuery, etc.

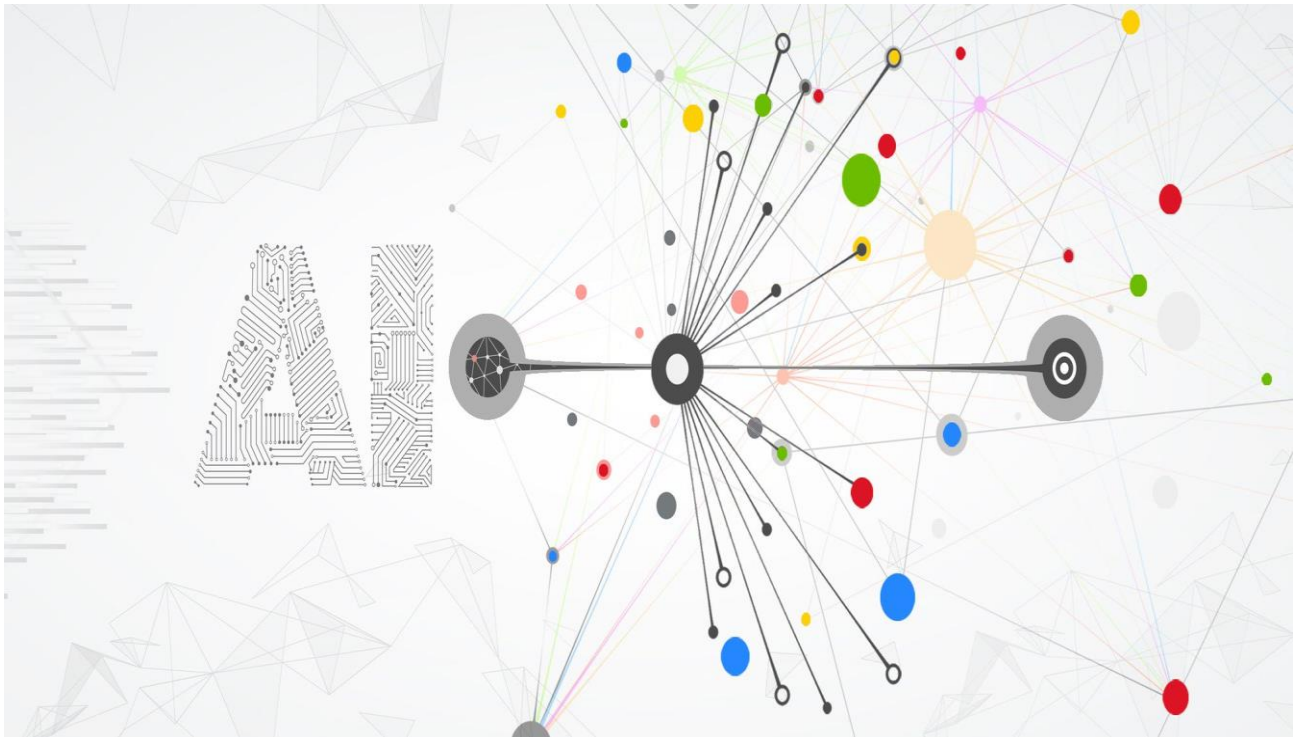
Cloud Composer

Cloud Composer is Google's fully-managed Apache Airflow service to create, schedule, monitor, and manage workflows. It handles all the infrastructure for you so that you can concentrate on combining the services I have described above to create your own workflows.
Under the hood, a GKE cluster will be created with Airflow in it and GCS will be used to store files.

AI and Machine Learning in GCP

Covering the basics of machine learning would take another article. So here, I assume you are familiar with it and will show you how to train and deploy your models in GCP.

We'll also look at what APIs are available to leverage Google's machine learning capabilities in your services, even if you are not an expert in this area.



AI Platform

AI Platform provides you with a **fully-managed platform** to use machine learning libraries like Tensorflow. You just need to focus on your model and Google will handle all the infrastructure needed to train it.
After your model is trained, you can use it to get online and batch predictions.

Cloud AutoML

Google lets you use **your data to train their models**. You can leverage models to build applications that are based on natural language processing (for example, document classification or sentiment analysis applications), speech processing, machine

translation, or video processing (video classification or object detection).

How to explore and visualize your data in GCP

Cloud Data Studio

Data Studio lets you create **visualizations and dashboards** based on data that resides in Google services (YouTube Analytics, Sheets, AdWords, local upload), Google Cloud Platform (BigQuery, Cloud SQL, GCS, Spanner), and many third-party services, storing your reports in Google Drive.
Data Studio is not part of GCP, but **G-Suite**, thus its permissions are not managed using IAM.
There are no additional costs for using Data Studio, other than the storage of the data, queries in BigQuery, and so on. **Caching** can be used to improve performance and reduce costs.

Cloud Datalab

Datalab lets you **explore, analyze, and visualize data** in BigQuery, ML Engine, Compute Engine, Cloud Storage, and Stackdriver.
It is based on Jupyter notebooks and supports Python, SQL, and Javascript code. Your notebooks can be shared via the Cloud Source Repository.

Cloud Datalab itself is free of charge, but it will create a virtual machine in GCE for which you will be billed.

Security in GCP

Encryption on Google Cloud Platform

Google Cloud encrypts data both at rest (data stored on disk) and in transit (data traveling in the network), using AES implemented via Boring SSL.
You can manage the encryption keys yourself (both storing them in GCP or on-premise) or let Google handle them.

Encryption at rest

GCP encrypts data stored at rest **by default**. Your data will be divided into chunks. Each chunk is distributed across different machines and encrypted with a unique key, called a **data encryption key (DEK)**.
Keys are generated and managed by Google but you can also manage the keys yourself, as we will see later in this guide.

Encryption in Transit

To add an extra security layer, all communications between two GCP services or from your infrastructure to GCP are encrypted at one or more network layers. Your data would not be compromised if your messages were to be intercepted.

Cloud Key Management Service (KMS)

As I mentioned earlier, you can let Google manage the keys for you or you can manage them yourself.

Google KMS is the service that allows you to **manage your encryption keys**. You can create, rotate, and destroy symmetric encryption keys. All keys related activity is registered in logs. These keys are referred to as **customer-managed encryption keys**. In GCS, they are used to encrypt:

- The object's data.
- The object's CRC32C checksum.
- The object's MD5 hash.

And Google uses server-side keys to handle the rest of the metadata, including the object's name.

The DEKs used to encrypt your data are also encrypted using key encryption keys (KEKs), in a process called envelope encryption. By default, KEKs are rotated every 90 days.

It is important to note that KMS does not store secrets. KMS is a central repository for KEKs. Only the keys that GCP needs to encrypt secrets that are stored somewhere else, for instance in Secrets management.
Note: For GCE and GCS, you have the possibility of keeping your keys on-premise and let Google retrieve them to encrypt and decrypt your data. These are known as **customer-supplied keys**.

Identity-Aware Proxy (IAP)

Identity-Aware Proxy allows you to **control the access** GCP applications via HTTPs without installing any VPN software or adding extra code in your application to handle login.
Your applications are visible to the public internet, but only accessible to authorized users, implementing a zero-trust security access model.

Furthermore, with TCP forwarding you can prevent services like SSH to be exposed to the public internet.

Cloud Armor

Cloud Armor protects your infrastructure from distributed denial of service (DDoS) attacks. You define rules (for example to whitelist or deny certain IP addresses or CIDR ranges) to create security policies, which are enforced at the Point of Presence level (closer to the source of the attack).
Cloud Armor gives you the option of previewing the effects of your policies before activating them.

Cloud Data Loss Prevention

Data Loss Prevention is a fully-managed service designed to help you discover, classify, and protect sensitive data, like:

- **Personable Identifiable Information (PII)**: name, Social Security number, driver's license number, bank account number, passport number, email address, and so on.
- **Secrets**
- **Credentials**

DLP is integrated with GCS, BigQuery, and Datastore. Also, the source of the data can be outside of GCP.

You can specify what type of data you're interested in, called info type, define your own types (based on dictionaries of words and phrases or based on regex expressions), or let Google use the default which can be time-consuming for large amounts of data.

For each result, DLP will return the likelihood of that piece of data matches a certain info type: LIKELIHOOD_UNSPECIFIED, VERY_UNLIKELY, UNLIKELY, POSSIBLE, LIKELY, VERY_LIKELY.

After detecting a piece of PII, DLP can transform it so that it cannot be mapped back to the user. DLP uses multiple techniques to de-identify your sensitive data like tokenization, bucketing, and date shifting. DLP can detect and redact sensitive data in images too.

VPC Service Control

VPC Service Control helps prevent data exfiltration. It allows you to define a perimeter around resources you want to protect. You can define what services and from what networks these resources can be accessed.

Cloud Web Security Scanner

Cloud Web Security Scanner scanner applications running in Compute Engine, GKE, and App Engine for common vulnerabilities such as passwords in plain text, invalid headers, outdated libraries, and cross-site scripting attacks. It simulates a real user trying to click on your buttons, inputting text in your text fields, and so on.
It is part of Cloud Security Command Center.

More GCP resources

- Google Cloud Solutions Architecture Reference
- GCP solutions by industry
- GCP Youtube channel
- GCP Labs

If you're interested in learning more about GCP, I recommend checking the free practice exams for the different certifications. Whether you are preparing for a GCP or not you can use them to find gaps in your knowledge:

- Professional Cloud Developer
- Professional Cloud Data Engineer
- Professional Cloud Network Engineer
- Professional Cloud Security Engineer
- Professional Cloud DevOps Engineer
- Professional Cloud Machine Learning Engineer
- Professional Cloud Architect

Note: Some questions are based on case studies. Links to the case studies will be provided in the exams so that you have the full context to properly understand and answer the question.

Time to test your knowledge

I've extracted 10 questions from some of the exams above. Some of them are pretty straightforward. Others require deep thought and deciding what is the best solution when more than one option is a viable solution.

Question 1

Your customer is moving their corporate applications to Google Cloud. The security team wants detailed visibility of all resources in the organization. You use the Resource Manager to set yourself up as the Organization Administrator.

Which Cloud Identity and Access Management (Cloud IAM) roles should you give to the security team while following Google's recommended practices?

A. Organization viewer, Project owner

B. Organization viewer, Project viewer

C. Organization administrator, Project browser

D. Project owner, Network administrator

Question 2

Your company wants to try out the cloud with low risk. They want to archive approximately 100 TB of their log data to the cloud and test the serverless analytics features available to them there, while also retaining that data as a long-term disaster recovery backup.

Which two steps should they take? (Choose two)

A. Load logs into BigQuery.

B. Load logs into Cloud SQL.

C. Import logs into Cloud Logging.

D. Insert logs into Cloud Bigtable.

E. Upload log files into Cloud Storage.

Question 3

Your company wants to track whether someone is present in a meeting room reserved for a scheduled meeting.

There are 1000 meeting rooms across 5 offices on 3 continents. Each room is equipped with a motion sensor that reports its status every second.

You want to support the data ingestion needs of this sensor network. The receiving infrastructure needs to account for the possibility that the devices may have inconsistent connectivity.

Which solution should you design?

A. Have each device create a persistent connection to a Compute Engine instance and write messages to a custom application.

B. Have devices poll for connectivity to Cloud SQL and insert the latest messages on a regular interval to a device-specific table.

C. Have devices poll for connectivity to Cloud Pub/Sub and publish the latest messages on a regular interval to a shared topic for all devices.

D. Have devices create a persistent connection to an App Engine application fronted by Cloud Endpoints, which ingest messages and write them to Cloud Datastore.

Question 4

To reduce costs, the Director of Engineering has required all developers to move their development infrastructure resources from on-premises virtual machines (VMs) to Google Cloud.

These resources go through multiple start/stop events during the day and require the state to persist.

You have been asked to design the process of running a development environment in Google Cloud while providing cost visibility to the finance department.

Which two steps should you take? (Choose two)

A. Use persistent disks to store the state. Start and stop the VM as needed.

B. Use the --auto-delete flag on all persistent disks before stopping the VM.

C. Apply the VM CPU utilization label and include it in the BigQuery billing export.

D. Use BigQuery billing export and labels to relate cost to groups.

E. Store all state in a Local SSD, snapshot the persistent disks and terminate the VM.

Question 5

The database administration team has asked you to help them improve the performance of their new database server running on Compute Engine.

The database is used for importing and normalizing the company's performance statistics. It is built with MySQL running on Debian Linux. They have an n1-standard-8 virtual machine with 80 GB of SSD zonal persistent disk which they can't restart until the next maintenance event.

What should they change to get better performance from this system as soon as possible and in a cost-effective manner?

A. Increase the virtual machine's memory to 64 GB.

B. Create a new virtual machine running PostgreSQL.

C. Dynamically resize the SSD persistent disk to 500 GB.

D. Migrate their performance metrics warehouse to BigQuery.

Question 6

Your organization has a 3-tier web application deployed in the same Google Cloud Virtual Private Cloud (VPC).

Each tier (web, API, and database) scales independently of the others. Network traffic should flow through the web to the API tier, and then on to the database tier. Traffic should not flow between the web and the database tier.

How should you configure the network with minimal steps?

A. Add each tier to a different subnetwork.

B. Set up software-based firewalls on individual VMs.

C. Add tags to each tier and set up routes to allow the desired traffic flow.

D. Add tags to each tier and set up firewall rules to allow the desired traffic flow.

Question 7

You are developing an application on Google Cloud that will label famous landmarks in users' photos. You are under competitive pressure to develop a predictive model quickly. You need to keep service costs low.

What should you do?

A. Build an application that calls the Cloud Vision API. Inspect the generated MID values to supply the image labels.

B. Build an application that calls the Cloud Vision API. Pass client image locations as base64-encoded strings.

C. Build and train a classification model with TensorFlow. Deploy the model using the AI Platform Prediction. Pass client image locations as base64-encoded strings.

D. Build and train a classification model with TensorFlow. Deploy the model using the AI Platform Prediction. Inspect the generated MID values to supply the image labels.

Question 8

You set up an autoscaling managed instance group to serve web traffic for an upcoming launch.

After configuring the instance group as a backend service to an HTTP(S) load balancer, you notice that virtual machine (VM) instances are being terminated and re-launched every minute. The instances do not have a public IP address.

You have verified that the appropriate web response is coming from each instance using the curl command. You want to ensure that the backend is configured correctly.

What should you do?

A. Ensure that a firewall rule exists to allow source traffic on HTTP/HTTPS to reach the load balancer.

B. Assign a public IP to each instance and configure a firewall rule to allow the load balancer to reach the instance public IP.

C. Ensure that a firewall rule exists to allow load balancer health checks to reach the instances in the instance group.

D. Create a tag on each instance with the name of the load balancer. Configure a firewall rule with the name of the load balancer as the source and the instance tag as the destination.

Question 9

You created a job that runs daily to import highly sensitive data from an on-premises location to Cloud Storage. You also set up a streaming data insert into Cloud Storage via a Kafka node that is running on a Compute Engine instance.

You need to encrypt the data at rest and supply your own encryption key. Your key should not be stored in the Google Cloud.

What should you do?

A. Create a dedicated service account and use encryption at rest to reference your data stored in Cloud Storage and Compute Engine data as part of your API service calls.

B. Upload your own encryption key to Cloud Key Management Service and use it to encrypt your data in Cloud Storage. Use your uploaded encryption key and reference it as part of your API service calls to encrypt your data in the Kafka node hosted on Compute Engine.

C. Upload your own encryption key to Cloud Key Management Service and use it to encrypt your data in your Kafka node hosted on Compute Engine.

D. Supply your own encryption key, and reference it as part of your API service calls to encrypt your data in Cloud Storage and your Kafka node hosted on Compute Engine.

Question 10

You are designing a relational data repository on Google Cloud to grow as needed. The data will be transactionally consistent and

added from any location in the world. You want to monitor and adjust node count for input traffic, which can spike unpredictably.

What should you do?

A. Use Cloud Spanner for storage. Monitor storage usage and increase node count if more than 70% utilized.

B. Use Cloud Spanner for storage. Monitor CPU utilization and increase node count if more than 70% utilized for your time span.

C. Use Cloud Bigtable for storage. Monitor data stored and increase node count if more than 70% is utilized.

D. Use Cloud Bigtable for storage. Monitor CPU utilization and increase node count if more than 70% utilized for your time span.

Answers

1. B
2. A, E
3. C
4. A, D
5. C
6. D
7. B
8. C
9. D
10. B

Back to the initial proposition

At the beginning of this article, I said you'd learn how to design a mobile gaming analytics platform that collects, stores, and analyzes vast amounts of player-telemetry both from bulks of data and real-time events.

So, do you think you can do it?

Take a pen and a piece of paper and try to come up with your own solution based on the services I have described here. If you get stuck, the following questions might help:

- The platform needs to collect real-time events from the game:
- Where might be the game running?
- How can you ingest streaming data from the game into GCP?
- How can you store it?
- How can you collect and store the uploads of batches of data?
- Can you analyze all the ingested data as it comes? Does it need to be processed?
- What services can you use to analyze the data? How would this change if low-latency was now a new requirement?

I have purposely defined the problem in a very vague way. This is what you can expect when you are facing this sort of challenge: uncertainty. It is part of your job to gather requirements and document your assumptions.

Do not worry if your solution does not look like Google's. This is just one possible solution. Learning to design complex systems is a skill that takes a lifetime to master. Luckily, you're headed in the right direction.

Conclusion

This guide will help you get started on GCP and give you a broad perspective of what you can do with it.

By no means will you be an expert after finishing this guide, or any other guide for that matter. The only way to really learn is by practicing.

You are going to learn infinitely more by doing than by reading or watching. I strongly recommend using your free trial and Code

Labs if you are serious about learning.

You can visit my blog www.yourdevopsguy.com and follow me on Twitter for more high-quality technical content.
Disclaimer: At the time of publishing this article, I don't work or have ever worked for Google. I wanted to organize and summarize the knowledge I have acquired learned via the Google documentation, YouTube videos, the courses that I have taken and most importantly through hands-on practice using GCP daily on my job.
All of this information is free out there. The figures, numbers, and versions that you see here come from the documentation at the time I am publishing this article. To make sure you are using up-to-date data, please visit the official documentation.