GOOGLE CLOUD PLATFORM FUNDAMENTALS – CORE INFRASTRUCTURE

* Five Fundamental Attributes of Cloud Computing
  + On-Demand and Self Service – Automated Interface, Get resources without any human intervention.
  + Access Over the Network – Access Resources from anywhere in the world
  + Resource Pooling – Provider has a big pool, where he gets economies of scale by buying in bulk.
  + Rapid Elasticity – Easily can scale up and Scale down.
  + Measured Service – Pay for what you use
* Trend Towards Cloud Computing
  + Colocation – In house – User Configured, Managed and Maintained.
  + Virtualized Data Center – User Configured, Provider Managed and Maintained. Data Center.
  + Server Less – Fully Automated.
* Virtualized Data Centers Offer
  + Infrastructure as a Service –
    - Provides raw compute, storage and network organized like physical and virtualized data centers.
    - Pay for what you allocate
  + Platform as a Service –
    - Bind your code to libraries that provide access to infrastructure your application needs.
    - Pay for what you use
* Cloud computing has evolved, and the momentum is shifted towards managed infrastructure and managed service.
* Google Cloud Platform is organized into regions and zones
  + Regions –
    - Regions are independent geographic areas that contain Zones.
    - Location within a region have a network latency of under 5 milliseconds
  + Zones –
    - Zone is s deployment area for GCP resource within a region.
    - Zone is s single failure domain in a region.
    - To deploy fault tolerant applications with high availability, deploy application across multiple zones in the same region.
* GCP Services and Resources can be one of
  + Zonal Resources –
    - Operate in a single zone
    - If zone becomes unavailable all resources become unavailable until service is restarted.
    - Compute Engine VM is a Zonal resource.
  + Regional Resources –
    - These are deployed with redundancy within a region. This gives higher availability than zonal resources.
  + Managed by Google across multiple regions – Multi Regional Resources
    - Few services are redundant within and across regions, these optimize availability, performance and resource efficiency.
    - This protects against loss of an entire region due to natural disaster.
    - App Engine and its features
    - Cloud Data Store
    - Cloud Storage
    - Big Query
* Google was the first major cloud provider to deliver per second billing. Per second billing is offered for users of
  + Compute Engine
  + Kubernetes Engine – Container Infrastructure as a Service
  + Cloud Dataproc – Open Source Big Data System, Hadoop as a Service
* Google Compute Engine Offers automatically applies Sustained Use Discounts, which are automatic discounts applied when you run your VM instance for significant portion of the billing month. Specifically, when you run your instances for more than 25% of a month, compute engine automatically gives you discount for every incremental minute you use.
* Using Open API’s Google services are compatible with open source products. Google Bigtables uses Apache Hbase interface and Google Cloud Dataproc is based on open source big data environment Hadoop.
* Google publishes key elements of its technology, using open source licenses, which provides customers options other than google. Example Tensorflow an open source software library for Machine Learning.
* Security is designed into Google’s technical infrastructure.
  + Hardware Design and Provenance – Server boards and Networking equipment are custom designed by Google. Google also designs custom ships and hardware security chips that are deployed on both servers and peripherals.
  + Secure Boot Stack – Google server machines use variety of technologies to ensure that they are booting the correct software stack, such as cryptographic signatures over BIOS, bootloader, kernel and base operating system image.
  + Premise Security - Google designs and builds its own data centers, which incorporate multiple layers of physical security protections.
  + Encryption of inter-service communication - Google’s infrastructure provides cryptographic privacy and integrity for remote procedure call (“RPC”) data on the network. Google’s services communicate with each other using RPC calls. The infrastructure automatically encrypts all infrastructure RPC traffic which goes between data centers. Google has started to deploy hardware cryptographic accelerators that will allow it to extend this default encryption to all infrastructure RPC traffic inside Google data centers
  + User identity - The service also intelligently challenges users for additional information based on risk factors such as whether they have logged in from the same device or a similar location in the past. Users also have the option of employing second factors when signing in, including devices based on the Universal 2nd Factor (U2F) open standard
  + Encryption at rest: Most applications at Google access physical storage indirectly via storage services, and encryption (using centrally managed keys) is applied at the layer of these storage services. Google also enables hardware encryption support in hard drives and SSDs.
  + Google Front End (“GFE”): Google services that want to make themselves available on the Internet register themselves with an infrastructure service called the Google Front End, which ensures that all TLS connections are terminated using correct certificates and following best practices such as supporting perfect forward secrecy. The GFE additionally applies protections against Denial of Service attacks.
    - A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.
  + Denial of Service (“DoS”) protection: The sheer scale of its infrastructure enables Google to simply absorb many DoS attacks. Google also has multi-tier, multi-layer DoS protections that further reduce the risk of any DoS impact on a service running behind a GFE.
  + Intrusion detection: Rules and machine intelligence give operational security engineers warnings of possible incidents. Google conducts Red Team exercises to measure and improve the effectiveness of its detection and response mechanisms.
  + Reducing insider risk: Google aggressively limits and actively monitors the activities of employees who have been granted administrative access to the infrastructure.
    - Two-factor authentication is an essential way to protect your important accounts. Traditionally, most accounts just need a password to log in—that’s one factor, something you know. Anyone who knows the password can get into your account. Two-factor authentication requires something you know and something you have. Often, this is a message sent to your phone via SMS or a code generated via an app like Google Authenticator or Authy on your phone. Someone needs both your password and access to the physical device to log in.
  + Employee U2F use: To guard against phishing attacks against Google employees, employee accounts require use of U2F-compatible Security Keys.
  + Software development practices: Google employs central source control and requires two-party review of new code. Google also provides its developers libraries that prevent them from introducing certain classes of security bugs. Google also runs a Vulnerability Rewards Program where we pay anyone who can discover and inform us of bugs in our infrastructure or applications.
* GCP Products and Services can be broadly classified as
  + Compute
  + Storage
  + Big Data
  + Machine Learning

Getting Started with GCP

* When you build an application on your On-Premises infrastructure you are responsible for entire stack security.
  + Physical security of the hardware and premise in which they are housed.
  + Encryption of data on disk.
  + Integrity of your network.
  + Securing content stored in your application
* When you move your application to Google Cloud Platform
  + Google handles many of the lower levels of security
  + Google can deliver a higher level of security at these layers than most of its customers could afford to do on their own.
  + The upper layers of the security stack remain the customer’s responsibility. Google provides tools, such as IAM, to help customers implement the policies they choose at these layers.
* GCP Hierarchy
  + Organization Node
  + Folders and Sub Folders
  + Projects
  + Resources
* Policies are inherited downward in the hierarchy
* All GCP Services you use must be associated with a project.
  + Projects are the basis of enabling and using GCP Services. Like enabling API, Billing etc.
  + Each project is a separate compartment, and each resource belongs to exactly one project.
  + Projects can have different owners and users. They’re billed separately, and they’re managed separately.
* The Cloud Resource Manager provides methods that you can use to programmatically manage your projects in Google Cloud Platform.
* Projects have 3 identifying attributes
  + Project ID
    - Globally Unique
    - Chosen by You
    - Immutable
  + Project Name
    - Need not be unique
    - Chosen by You
    - Mutable
  + Project Number
    - Globally Unique
    - Assigned by GCP
    - Immutable
* Cloud IAM Folders
  + Lets you assign policies to folders which are inherited by all resources under that folder.
  + Folders contain other folders, projects or Both.
  + You can use folders to group projects under an organization.
  + To use folders, you need an organization node at the top of the hierarchy.
* Organization Node provides centralized visibility of how resources are being used and you can apply policies centrally.
* There are few special roles which can be assigned at the Organization Level
  + Organization Policy Administrator - People with this privilege can change policies
  + Project Creator Role – People with this privilege can create Projects
* How do you get an organization node?
  + If you have a G Suite Domain, GCP project will automatically belong to your organizational node.
  + You can also use Google Cloud Identity to create one.
* The policies implemented at a higher level can’t take away access that’s granted at lower level. And like wise Policies implemented to take away access at a lower level cannot override access granted at a higher level. The more generous policy takes effect.
* Google Cloud Identity and Access Management define
  + Who – IAM Policies can apply to Google Account, Google Group, Service Account, Entire G Suite or Cloud Identity Domain?
  + Can do what – IAM Roles which are collection of permissions.
  + On which resources – This defines the resources on which these roles and policies are applicable to.
* Types of IAM Roles
  + Primitive Roles – These are broad. IAM primitive roles offer fixed, coarse-grained levels of access
    - Owner - Plus manage roles and permissions on the resource and set up billing.
    - Editor - Plus change its state
    - Viewer - Can examine it but not change its state
    - Billing Administrator - Control the billing for a project without the right to change the resources in the project.
  + Per-Defined – These are roles that apply to specific services in projects. Predefined roles offer more fine-grained permissions on services
  + Custom - Let you define a precise set of permissions.
    - You’ll need to manage the permissions that make them up.
    - Custom roles can only be used at the project or organization levels. They can’t be used at the folder level.
* Least Privilege Model - Each person in your organization gets the minimal amount of privilege needed to do his or her job.
* Service Accounts – These control Server to Server Interactions.
  + Service accounts are named with an email address, but instead of passwords they use cryptographic keys to access resources.
  + Provide an identity for carrying out server-to-server interactions in a project
  + Used to authenticate from one service to another
  + Used to control privileges used by resources, so that applications can perform actions on behalf of authenticated end users.
  + You can assign a predefined or custom IAM role to the service account.
  + Service account is also a resource! So, it can have IAM policies of its own attached to it. For instance, Alice can have the editor role on a service account and Bob can have the viewer role.
  + You can grant different groups of VMs in your project different identities. This makes it easier to manage different permissions for each group.
* Managing Administrative users –
  + Easiest way to access GCP resources is using Google Account and Google Groups. Disadvantage of this is that the identities are not centrally managed. Which causes problem when your have to revoke access to employee who left the company.
  + For GCP customers who are G Suite customers can use Google Admin Console.
  + GCP Customers who are not G Suite customer get the same capabilities using Cloud Identity. Cloud Identity lets you manage users and groups using the Google Admin Console.
  + If you have a different corporate directory, you can use Google Cloud Directory Sync. This tool synchronizes users and groups from your existing Active Directory or LDAP system with the users and groups in your Cloud Identity domain. The synchronization is one-way only; no information in your Active Directory or LDAP map is modified. Google Cloud Directory Sync is designed to run scheduled synchronizations without supervision, after its synchronization rules are set up.
* Interacting with GCP
  + GCP Console
  + Cloud Shell and Cloud SDK
    - A temporary Compute Engine virtual machine instance running a Debian-based Linux operating system
    - Command-line access to the instance from a web browser using terminal windows in the Cloud Platform Console
    - 5 GB of persistent disk storage per user, mounted as your $HOME directory in Cloud Shell sessions across projects and instances
    - Google Cloud SDK and other tools pre-installed on the Compute Engine instance
    - Language support, including SDKs, libraries, runtime environments and compilers for Java, Go, Python, Node.js, PHP and Ruby
    - Web preview functionality, which allows you to preview web applications running on the Cloud Shell instance through a secure proxy
    - Built-in authorization for access to projects and resources
  + Mobile APP
  + REST API
    - Programmatic access to products and services
    - Typically use JSON as an interchange format
    - Use OAuth 2.0 for authentication and authorization
* The APIs Explorer is an interactive tool that lets you easily try Google APIs using a browser.
  + Browse quickly through available APIs and versions.
  + See methods available for each API and what parameters they support along with inline documentation.
  + Execute requests for any method and see responses in real time.
  + Easily make authenticated and authorized API calls.
* Cloud Marketplace gives quick access to solutions. Let’s you quickly deploy functional software packages that run on Google Cloud Platform. You can easily start up a familiar software package without having to manually configure the software, virtual machine instances, storage, or network settings. Google Cloud Platform updates the images for these software packages to fix critical issues and vulnerabilities but doesn't update software that you have already deployed.

Virtual Machines in Cloud

* Virtual Private Cloud
  + VPC networks connect your Google Cloud Platform resources to each other and to the internet.
  + You can segment your networks, use firewall rules to restrict access to instances, and create static routes to forward traffic to specific destinations.
  + VPC networks are Global and Subnets are Regional
    - Google Virtual Private Cloud networks that you define have global scope. They can have subnets in any GCP region worldwide.
    - Subnets can span Zones that make up a region. This architecture makes it easy for you to define a network with global scope.
    - You can have resources in different zones on the same subnet.
    - You can dynamically increase the size of the subnet in your network by expanding the range of IP addresses allocated to it. Doing that does not affect already configured VMs.
* Compute Engines
  + Compute engine lets you create and run virtual machines on Google infrastructure.
  + You can create VMs by using GCP Console or the GCloud Command Line Tool.
  + You configure a VM much like you build out a physical server, by specifying amount of CPU power, memory, storage and Operating system.
  + Compute engine instances can run Linux or windows provided by google or any customized versions of these images. Or you can also build and run images of other operating systems.
  + Compute engines bills by the second for use of the VM, with a 1 min minimum.
  + And discounts apply automatically to virtual machines that run for substantial fractions of a month. You can get up to a 30% net discount for VMs that run the entire month.
  + Compute Engine offers the ability to purchase committed use contracts in return for deeply discounted prices for VM usage. These discounts are known as committed use discounts. If your workload is stable and predictable, you can purchase a specific number of vCPUs and memory for up to a 57% discount off of normal prices in return for committing to a usage term of 1 year or 3 years.
  + Suppose you have a workload that no human being is sitting around waiting to finish. Say, a batch job analyzing a large dataset. You can save money by choosing Preemptible VMs to run the job. A Preemptible VM is different from an ordinary Compute Engine VM in only one respect: you’ve given Compute Engine permission to terminate it if its resources are needed elsewhere. You can save a lot of money with preemptible VMs.
  + Compute Engine has a feature called Autoscaling that lets you add and take away VMs from your application based on load metrics
* VPC capabilities
  + Much like physical networks, VPCs have routing tables. These are used to forward traffic from one instance to another instance within the same network, even across subnetworks and even between GCP zones, without requiring an external IP address. VPCs’ routing tables are built in; you don’t have to provision or manage a router.
  + Another thing you don’t have to provision or manage for GCP: a firewall. VPCs give you a global distributed firewall you can control to restrict access to instances, both incoming and outgoing traffic.
  + You can define firewall rules in terms of metadata tags on Compute Engine instances, which is convenient. For example, you can tag all your web servers with, say, “WEB,” and write a firewall rule saying that traffic on ports 80 or 443 is allowed into all VMs with the “WEB” tag, no matter what their IP address happens to be.
  + VPC belong to Projects, if your company has many VPS’s and the VPC’s need to talk to each other you can use
    - VPC Peering - establish a peering relationship between two VPCs, so that they can exchange traffic
    - Shared VPC - use the full power of IAM to control who and what in one project can interact with a VPC in another.
  + Cloud Load Balancing is a fully distributed, software-defined, managed service for all your traffic. And because the load balancers don’t run in VMs you have to manage, you don’t have to worry about scaling or managing them. You can put Cloud Load Balancing in front of all of your traffic: HTTP(S), other TCP and SSL traffic, and UDP traffic too.
  + With Cloud Load Balancing, a single anycast IP front ends all your backend instances in regions around the world
  + It provides cross-region load balancing, including automatic multi-region failover, which gently moves traffic in a fraction of seconds if backends become unhealthy. Cloud Load Balancing reacts quickly to changes in users, traffic, network, backend health, and other related conditions.
* Cloud DNS
  + DNS is what translates Internet hostnames to addresses.
  + It’s a managed DNS service running on the same infrastructure as Google.
  + It has low latency and high availability, and it’s a cost-effective way to make your applications and services available to your users.
* Cloud CDN (Content Delivery Network)
  + Google has a global system of edge caches. You can use this system to accelerate content delivery in your application using Google Cloud CDN. Your customers will experience lower network latency, the origins of your content will experience reduced load

Storage in Cloud

* GCP Core Storage Options
  + Google Cloud Storage –
    - Durable and Highly Available Object Store
    - No Minimum fee, you only pay for what you use.
    - Storage Classes
      * File Storage – Here you manage your data as a hierarchy of folders
      * Block Storage – Here the OS manages your data as chunks of disk
      * Object Storage – Here you manage and manipulate data storage as distinct objects called Objects. Here the storage lets you address it with a unique key. These unique keys are in the form of urls.
    - Google Cloud Storage always encrypts data in the server side before it is written to disk. And data travelling between customers device and google is also encrypted by default, using HTTPS/TLS and Google was the first major cloud provider to enable HTTPS/TLS by default.
    - Google Cloud Storage is not a file system, although it can be accessed as one via third-party tools such as Cloud Storage FUSE
    - Storage objects are immutable which means you cannot edit them in place, but instead create new versions.
    - Use Cases for Google Cloud Storage
      * Primary when binary large object storage is needed.
      * Online Content Backup and Archiving.
      * Storage of intermediate results in processing workflows
    - Offline Media Import/Export is a third-party solution that allows you to load data into Google Cloud Storage by sending your physical media, such as hard disk drives (HDDs), tapes, and USB flash drives, to a third-party service provider who uploads data on your behalf. Offline Media Import/Export is helpful if you’re limited to a slow, unreliable, or expensive internet connection.
    - Cloud Storage Transfer Service enables you to import large amounts of online data into Google Cloud Storage quickly and cost-effectively. To use Cloud Storage Transfer Service, you set up a transfer from a data source to data sink. Data sources can be an Amazon Simple Storage Service (Amazon S3) bucket, an HTTP/HTTPS location, or another Google Cloud Storage bucket. Data sinks are always a Google Cloud Storage bucket.
    - Your Cloud Storage files are organized into buckets. When you create a bucket, you give it a globally unique name, specify geographic location and choose a default storage class. Pick a location which minimizes latency for your users.
    - There are several ways to control user access to objects and buckets
      * Cloud IAM is a good option, Roles are inherited from project to bucket to object.
      * You can also use Access Control Lists if you need finer control. ACLs define who has access to your buckets and objects, as well as what level of access they have. Each ACL consists of two pieces of information: A scope, which defines who can perform the specified actions (for example, a specific user or group of users). And a permission, which defines what actions can be performed (for example, read or write).
    - If you turn on Object Versioning Cloud Storage keeps a history of modifications--that is, overwrites or deletes--of all objects in the bucket.
      * You can list the archived versions of an object
      * Restore an object to an older state
      * Permanently delete a version as needed.
    - If object versioning is not turned on, New always overwrites the Old.
    - Cloud Storage also offers lifecycle management policies. For example, you could tell Cloud Storage to delete objects older than 365 days, or to delete objects created before January 1, 2013; or to keep only the 3 most recent versions of each object in a bucket that has versioning enabled.
    - Cloud Storage Classes – Regional and Multi Regional are high performance object storage, Nearline and Cold line are backup and archival storage.
      * Regional – Lets you store data in a specific GCP region, Cheaper than Multi Regional, but offers less redundancy. People use regional storage, on the other hand, to store data close to their Compute Engine virtual machines or their Kubernetes Engine clusters. That gives better performance for data-intensive computations.
      * Multi Regional – It is Geo redundant. Cloud Storage stores your data in at least two geographic locations separated by at least 160 kilometers. Multi-Regional Storage is appropriate for storing frequently accessed storing data: website content, interactive workloads, or data that’s part of mobile and gaming applications.
      * Nearline - low-cost, highly durable storage service for storing infrequently accessed data. Great choice if you plan to read or modify your data on average once a month or less. For example, if you want to continuously add files to Cloud Storage and plan to access those files once a month for analysis, Nearline Storage is a great choice.
      * Cold Line - very-low-cost, highly durable storage service for data archiving, online backup, and disaster recovery. Coldline Storage is the best choice for data that you plan to access at most once a year due to its slightly lower availability, 90 days minimum storage duration, costs for data access and higher per operation cost.
    - The availability of these storage classes varies, with multi-regional having the highest availability of 99.95%, followed by regional with 99.9% and nearline and coldline with 99.0%.
    - All storage classes incur a cost per gigabyte of data stored per month, with multi-regional having the highest storage price and coldline the lowest storage price. Egress and data transfer charges may also apply. In addition to these charges Nearline storage also incurs an access fee per gigabyte of data read, and Coldline storage incurs a higher fee per gigabyte of data read.
    - There are several ways to bring data into Cloud Storage.
      * Many customers simply use gsutil, which is the Cloud Storage command from the Cloud SDK.
      * You can also move data in with a drag and drop in the GCP Console, if you use the Google Chrome browser.
      * If you must upload Terra bytes of data Google Cloud Platform offers
        + Online Storage Transfer Service - lets you schedule and manage batch transfers to Cloud Storage from another cloud provider, from a different Cloud Storage region, or from an HTTP(S) endpoint.
        + Offline Transfer Appliance - The Transfer Appliance is a rackable, high-capacity storage server that you lease from Google Cloud. You simply connect it to your network, load it with data, and then ship it to an upload facility where the data is uploaded to Cloud Storage. The service enables you to securely transfer up to a petabyte of data on a single appliance
      * As this storage option is tightly integrated with many of the Google Cloud Platform products and services. you can import and export tables from and to BigQuery, as well as Cloud SQL. You can also store App Engine logs, Cloud Datastore backups, and objects used by App Engine applications like images. Cloud Storage can also store instance startup scripts, Compute Engine images, and objects used by Compute Engine applications. In short, Cloud Storage is often the ingestion point for data being moved into the cloud and is frequently the long-term storage location for data.
  + Google Cloud SQL - Cloud SQL is an easy-to-use service that delivers fully managed relational databases
    - Cloud SQL lets you hand off to Google the mundane, but necessary and often time-consuming tasks—like applying patches and updates, managing backups, and configuring replications—so you can put your focus on building great applications.
    - Every Cloud SQL instance includes a network firewall, allowing you to control network access to your database instance by granting access.
    - Easily scale up to 64 processor cores and more than 100 GB of RAM. Quickly scale out with read replicas.
    - Managed Backups - Cloud SQL takes care of securely storing your backed-up data and makes it easy for you to restore from a backup and perform a point-in-time recovery to a specific state of an instance. Cloud SQL retains up to 7 backups for each instance, which is included in the cost of your instance.
    - Cloud SQL customer data is encrypted when on Google's internal networks and when stored in database tables, temporary files, and backups.
    - Another benefit of Cloud SQL instance is that they are accessible by other GCP services and even external services. You can use Cloud SQL with App Engine using standard drivers like Connector/J for Java or MySQLdb for Python. You can authorize Compute Engine instances to access Cloud SQL instances and configure the Cloud SQL instance to be in the same zone as your virtual machine. Cloud SQL also supports other applications and tools that you might be used to, like SQL Workbench, Toad and other external applications using standard MySQL drivers.
  + Google Cloud Spanner - Cloud Spanner is a horizontally scalable RDBMS
    - Cloud Spanner supports strong consistency, including strongly consistent secondary indexes, SQL, and managed instances with high availability through synchronous and built-in data replication.
    - Cloud Spanner is especially suited for applications requiring
      * A SQL RDBMS, with joins and secondary indexes
      * Built-in high availability
      * Strong global consistency
      * Database sizes exceeding ~2 TB
      * Many IOPS (Tens of thousands of reads/writes per second or more)
  + Cloud Data Store - Cloud Datastore is a highly scalable NoSQL database.
    - Cloud Datastore uses a distributed architecture to automatically manage scaling. Your queries scale with the size of your result set, not the size of your data set.
    - Cloud Datastore features
      * Atomic transactions - Datastore can execute a set of operations where either all succeed, or none occur
      * High availability of reads and writes - Datastore runs in Google data centers, which use redundancy to minimize impact from points of failure.
      * Massive scalability with high performance - Datastore uses a distributed architecture to automatically manage scaling. Datastore uses a mix of indexes and query constraints so your queries scale with the size of your result set, not the size of your data set.
      * Flexible storage and querying of data - Datastore maps naturally to object-oriented and scripting languages and is exposed to applications through multiple clients. It also provides a SQL-like query language.
      * Balance of strong and eventual consistency - Datastore ensures that entity lookups and ancestor queries always receive strongly consistent data. All other queries are eventually consistent. The consistency models allow your application to deliver a great user experience while handling large amounts of data and users.
      * Encryption at rest - Datastore automatically encrypts all data before it is written to disk and automatically decrypts the data when read by an authorized user.
      * Fully managed with no planned downtime - Google handles the administration of the Datastore service so you can focus on your application. Your application can still use Datastore when the service receives a planned upgrade.
  + Google Cloud Bigtable - Cloud Bigtable is Google's NoSQL big data database service.
    - Relational database offers you tables in which every row has the same set of columns, and the database engine enforces that rule, and other rules you specify for each table: the “database schema.” A rigorously enforced, infrequently changing schema helps many applications maintain data integrity.
    - But some applications call for a much more flexible approach. For these applications, not all rows might need to have the same columns, and in fact the database might be designed to take advantage of that by sparsely populating the rows. That’s part of what makes a NoSQL database what it is.
    - Cloud Bigtable is ideal for applications that need very high throughput and scalability for non-structured key/value data, where each value is typically no larger than 10 MB.
    - Cloud Bigtable also excels as a storage engine for batch MapReduce operations, stream processing/analytics, and machine-learning applications.
    - Cloud Bigtable is offered through the same open source API as HBase, the native Hadoop database. This enables portability of applications between HBase and Bigtable.
    - Why choose Big Table
      * BIG - Large Quantities (>10TB) of Semi-structured or Structured Data
      * FAST - Data is high throughput or rapidly changing
      * NoSQL - Transactions, strong relational semantics not required
      * TIME SERIES - Data is time-series or has natural semantic ordering
      * Big Data - You run asynchronous batch or real-time processing on the data
      * Machine Learning - You run machine learning algorithms on the data
    - Bigtable is designed to handle massive workloads at consistent low latency and high throughput, so it's a great choice for both operational and analytical applications, including IoT, user analytics, and financial data analysis
    - From an application API perspective, data can be read from and written to Cloud Bigtable through a data service layer like Managed VMs, the HBase REST Server, or a Java Server using the HBase client. Typically, this will be to serve data to applications, dashboards, and data services.
    - Data can also be streamed in through a variety of popular stream processing frameworks like Cloud Dataflow Streaming, Spark Streaming, and Storm.
    - Data can also be read from and written to Cloud Bigtable through batch processes like Hadoop MapReduce, Dataflow, or Spark. Often, summarized or newly calculated data is written back to Cloud Bigtable or to a downstream database.
* Comparing Storage Options
  + Cloud Storage is best for structured and unstructured binary or object data, like images, large media files and backups.
  + Cloud Datastore is best for semi-structured application data that is used in App Engine applications.
  + Bigtable is best for analytical data with heavy read and write events, like AdTech, financial or IoT data.
  + Cloud SQL is best for web frameworks and existing applications, like storing user credentials and customer orders.
  + Cloud Spanner is best for large-scale database applications that are larger than 2 TB. For example, for financial trading and e-commerce use cases.

Containers in Cloud

* Compute Engine is Infrastructure As A Service offering, with access to servers, file systems and networking.
* App Engine is GCP’s Platform as a service offering.
* Kubernetes Engine is a hybrid which conceptually sits between the two and benefits from both.
* Infrastructure as a Service
  + IAAS allows you to share compute resources with other developers by virtualizing the hardware using virtual machines.
  + Each developer can deploy their own operating system, access the hardware, and build their applications in a self-contained environment with access to RAM, file systems, networking interfaces, and so on.
  + But flexibility comes with a cost. The smallest unit of compute is an app with its VM. The guest OS may be large, even gigabytes in size, and takes minutes to boot.
  + But you have your tools of choice on a configurable system. So, you can install your favorite runtime, web server, database, or middleware, configure the underlying system resources, such as disk space, disk I/O, or networking and build as you like.
  + However, as demand for your application increases, you have to copy an entire VM and boot the guest OS for each instance of your app, which can be slow and costly.
* Platform as a Service
  + App Engine you get access to programming services. Write your code in self-contained workloads that use these services and include any dependent libraries.
  + As demand for your app increases, the platform scales your app seamlessly and independently by workload and infrastructure.
  + This scales rapidly but you won't be able to fine-tune the underlying architecture to save cost.
* Containers - The idea of a container is to give you the independent scalability of workloads in PaaS and an abstraction layer of the OS and hardware in IaaS.
  + What you get is an invisible box around your code and its dependencies, with limited access to its own partition of the file system and hardware.
  + All you need on each host is an OS kernel that supports containers and a container runtime.
  + You are virtualizing the OS. It scales like PaaS, but gives you nearly the same flexibility as IaaS
  + With this abstraction, your code is ultra-portable, and you can treat the OS and hardware as a black box. So, you can go from development, to staging, to production, or from your laptop to the cloud, without changing or rebuilding anything.
  + If you want to scale, for example, a web server, you can do so in seconds and deploy dozens or hundreds of them (depending on the size or your workload) on a single host. Now that's a simple example of scaling one container running the whole application on a single host.
  + You'll likely want to build your applications using lots of containers, each performing their own function like microservices. If you build them this way, and connect them with network connections, you can make them modular, deploy easily, and scale independently across a group of hosts. And the hosts can scale up and down and start and stop containers as demand for your app changes or as hosts fail.
  + A tool that helps you do this well is Kubernetes. Kubernetes makes it easy to orchestrate many containers on many hosts, scale them as microservices, and deploy rollouts and rollbacks.
  + Kubernetes is an open-source orchestrator that abstracts containers at a higher level so you can better manage and scale your applications. At the highest level, Kubernetes is a set of APIs that you can use to deploy containers on a set of nodes called a cluster.
  + The system is divided into a set of master components that run as the control plane and a set of nodes that run containers. In Kubernetes, a node represents a computing instance, like a machine. In Google Cloud, nodes are virtual machines running in Compute Engine. You can describe a set of applications and how they should interact with each other and Kubernetes figures how to make that happen
  + Kubernetes can be configured with many options and add-ons but can be time consuming to bootstrap from the ground up. Instead, you can bootstrap Kubernetes using Kubernetes Engine or (GKE). GKE is a hosted Kubernetes by Google. GKE clusters can be customized and they support different machine types, number of nodes, and network settings.
  + Then you deploy containers on nodes using a wrapper around one or more containers called a Pod. A Pod is the smallest unit in Kubernetes that you create or deploy. A Pod represents a running process on your cluster as either a component of your application or an entire app.
  + Generally, you only have one container per pod, but if you have multiple containers with a hard dependency, you can package them into a single pod and share networking and storage. The Pod provides a unique network IP and set of ports for your containers, and options that govern how containers should run. Containers inside a Pod can communicate with one another using localhost and ports that remain fixed as they're started and stopped on different nodes.
  + One way to run a container in a Pod in Kubernetes is to use the kubectl run command. This starts a Deployment with a container running in a Pod. A Deployment represents a group of replicas of the same Pod and keeps your Pods running even when nodes they run on fail. It could represent a component of an application or an entire app.
  + By default, Pods in a Deployment are only accessible inside your GKE cluster. To make them publicly available, you can connect a load balancer to your Deployment by running the kubectl expose command. Kubernetes creates a Service with a fixed IP for your Pods. Then a network load balancer is attached to the services which also attaches public IP to these services so others outside the cluster can access it. Any client that hits that IP address will be routed to a Pod behind the Service.
  + A Service is an abstraction which defines a logical set of Pods and a policy by which to access them. As Deployments create and destroy Pods, Pods get their own IP address. But those addresses don't remain stable over time. A Service groups a set of Pods and provides a stable endpoint (or fixed IP) for them. For example, if you create two sets of Pods called frontend and backend, and put them behind their own Services, backend Pods may change, but frontend Pods are not aware of this. They simply refer to the backend Service.
  + You can run the kubectl get services command to get the public IP to hit the container remotely.
  + To scale a Deployment, run the kubectl scale command. In this case, three Pods are created in your Deployment and they're placed behind the Service and share one fixed IP. You could also use autoscaling with all kinds of parameters.
  + But the real strength of Kubernetes comes when you work in a declarative way. Instead of issuing commands, you provide a configuration file that tells Kubernetes what you want your desired state to look like, and Kubernetes figures out how to do it.
  + To scale your Deployment using an existing Deployment config file. To get the file, you can run a kubectl get pods command. You'll get a Deployment configuration file. To run five replicas instead of three, all you do is update the Deployment config file. And run the kubectl apply command to use the config file. Then use the kubectl get pods command to watch the pods come online. And check the Deployment to make sure the proper number of replicas are running using either $ kubectl get deployments or $ kubectl describe deployments. And you can still hit your endpoint like before using $ kubectl get services to get the external IP of the Service and hit the public IP from a client.
  + When you want to update a new version of your app. You want to update your container to get new code out in front of users, but it would be risky to roll out all those changes at once. So, you use kubectl rollout or change your deployment configuration file and apply the change using kubectl apply. New Pods will be created according to your update strategy. Configuration will create new version Pods one by one and wait for a new Pod to be available before destroying one of the old Pods.

Applications in Cloud

* App Engine is a platform for building scalable web applications and mobile backends. It allows you to concentrate on innovating your applications by managing the application infrastructure for you. For example, App Engine manages the hardware and networking infrastructure required to run your code.
* App Engine provides you with built-in services and APIs such as NoSQL datastores, memcache, load balancing, health checks, application logging, and a user authentication API, common to most applications.
* App Engine will scale your application automatically in response to the amount of traffic it receives so you only pay for the resources you use. Just upload your code and Google will manage your app's availability. There are no servers for you to provision or maintain.
* Security Scanner automatically scans and detects common web application vulnerabilities. It enables early threat identification and delivers very low false-positive rates. You can easily set up, run, schedule, and manage security scans from the Google Cloud Platform Console.
* App Engine Standard Environment
  + The App Engine standard environment is based on container instances running on Google's infrastructure. Containers are preconfigured with one of several available runtimes (Java 7, Python 2.7, Go and PHP). Each runtime also includes libraries that support App Engine standard APIs.
  + The App Engine standard environment makes it easy to build and deploy an application that runs reliably even under heavy load and with large amounts of data.
  + App Engine Features
    - Persistent storage with queries, sorting, and transactions
    - Automatic scaling and load balancing
    - Asynchronous task queues for performing work outside the scope of a request
    - Scheduled tasks for triggering events at specified times or regular intervals
    - Integration with other Google cloud services and APIs
  + Software Development Kits (SDKs) for App Engine are available in all supported languages. Each SDK includes
    - All the APIs and libraries available to App Engine
    - A simulated, secure sandbox environment that emulates all of the App Engine services on your local computer
    - Deployment tools that allow you to upload your application to the cloud and manage different versions of your application
  + The SDK manages your application locally, and the Google Cloud Platform Console manages your application in production.
  + The Google Cloud Platform Console uses a web-based interface to create new applications, configure domain names, change which version of your application is live, examine access and error logs, and much more.
  + Applications run in a secure, sandboxed environment, allowing the App Engine standard environment to distribute requests across multiple servers, and scaling servers to meet traffic demands. Your application runs within its own secure, reliable environment that is independent of the hardware, operating system, or physical location of the server.
  + You’ll develop your application and run a test version of it locally using the App Engine SDK. Then, when you’re ready, you’ll use the SDK to deploy it. Each App Engine application runs in a GCP project. App Engine automatically provisions server instances and scales and load-balances them. Meanwhile, your application can make calls to a variety of services using dedicated APIs. For example, a NoSQL datastore to make data persistent; caching of that data using memcache; searching; logging; user login; and the ability to launch actions not triggered by direct user requests, like task queues and a task scheduler.
* App Engine Flexible Environment
  + Instead of the sandbox, App Engine Flexible Environment lets you specify the container your application runs in.
  + Your application runs inside Docker containers on Google Compute Engine virtual machines (VMs). App Engine manages these Compute Engine machines for you. They’re health-checked, healed as necessary, and you get to choose what geographical region they run in. And critical, backward-compatible updates to their operating systems are automatically applied.
  + Microservices, authorization, SQL and noSQL databases, traffic splitting, logging, search, versioning, security scanning, memcache, and content delivery networks are all supported natively. In addition, the App Engine flexible environment allows you to customize your runtime and even the operating system of your virtual machine using Dockerfiles.
  + Infrastructure customization: Because VM instances in the flexible environment are Compute Engine virtual machines, you can use SSH to connect to every single VM and Docker container for debugging purposes and further customization.
  + Performance: Take advantage of a wide array of CPU and memory configurations. You can specify how much CPU and memory each instance of your application needs, and the flexible environment will provision the necessary infrastructure for you.
  + App Engine manages your virtual machines, ensuring that:
    - Instances are health-checked, healed as necessary, and co-located with other module instances within the project.
    - Critical, backward-compatible updates are automatically applied to the underlying operating system.
    - VM instances are automatically located by geographical region according to the settings in your project. Google's management services ensure that all of a project's VM instances are co-located for optimal performance.
    - VM instances are restarted on a weekly basis. During restarts, Google's management services will apply any necessary operating system and security updates.
  + App Engine flexible environment apps that use standard runtimes can access App Engine services: Datastore, Memcache, task queues, logging, users, and so on.
  + Flexible Environment lets you ssh into the virtual machines on which your application runs; it lets you use local disk for scratch space; it lets you install third-party software; and it lets your application make calls to the network without going through App Engine. On the other hand, Standard Environment’s billing can drop to zero for a completely idle application.
  + Comparison of App Engine with Kubernetes Engine. App Engine Standard Environment is for people who want the service to take maximum control of their application’s deployment and scaling. Kubernetes Engine gives the application owner the full flexibility of Kubernetes. App Engine Flexible Edition is in between.
  + App Engine Environment treats containers as a means to an end. But for Kubernetes Engine, containers are a fundamental organizing principle.
* Google Cloud Endpoints and Apigee Edge
  + A software service’s implementation can be complex and changeable. If other software services had to be explicitly coded all that detail in order to use that service, the result would be brittle and error prone. So instead, application developers structure the software they write so that it presents a clean, well-defined interface that abstracts away needless detail, and then they document that interface. That’s an Application Programming Interface. The underlying implementation can change, as long as the interface doesn’t, and other pieces of software that use the API don’t have to know or care.
  + Sometimes you do have to change an API, such as to add or deprecate a feature. To make this kind of API change cleanly, developer’s version their APIs. Version 2 of an API might contain calls that version 1 does not; programs that consume the API can specify the API version they want to use in their calls
  + Supporting an API is a very important task, and Google Cloud Platform provides two API management tools.
  + Cloud Endpoint
    - Cloud Endpoints is a distributed API management system. It provides an API console, hosting, logging, monitoring, and other features to help you create, share, maintain, and secure your APIs. You can use Cloud Endpoints with any APIs that support the OpenAPI Specification, formerly known as the Swagger spec.
    - Cloud Endpoints uses the distributed Extensible Service Proxy to provide low latency and high performance for serving even the most demanding APIs. Extensible Service Proxy is a service proxy based on NGINX. It runs in its own Docker container for better isolation and scalability. The proxy is containerized and distributed in the Container Registry and Docker registry, and can be used with App Engine, Kubernetes Engine, Compute Engine or Kubernetes.
    - End Point Features
      * User Authentication - JSON Web Token validation and a streamlined developer experience for Firebase Auth, Google Auth and Auth0.
      * Automated Deployment - With App Engine, the proxy is deployed automatically with your application. On Kubernetes Engine or Compute Engine, use Google’s containerized ESP (Extensible Service Proxy) for simple deployment
      * Logging and monitoring - Monitor traffic, error rates and latency, and review logs in Cloud Logging. Use Cloud Trace to dive into performance and BigQuery for analysis
      * API keys - Generate API keys in Google Cloud Platform Console and validate on every API call. Share your API with other developers to allow them to generate their own keys.
      * Easy integration - Get started quickly by using one of Google’s Cloud Endpoints Frameworks or by simply adding an Open API specification to your deployment.
  + Apigee Edge
    - Apigee Edge is also a platform for developing and managing API proxies. It has a different orientation, though: it has a focus on business problems like rate limiting, quotas, and analytics.
    - Many users of Apigee Edge are providing a software service to other companies, and those features come in handy. Because the backend services for Apigee Edge need not be in GCP.
    - Engineers also often use it when they are working to take a legacy application apart. Instead of replacing a monolithic application in one risky move, they can instead use Apigee Edge to peel off its services one by one, standing up microservices to implement each in turn, until the legacy application can finally be retired.