GCP

- **Compute**
- Storage
- Big Data
- Machine Learning
- Application Services

The cloud computing mostly based on 5 traits.

- Customers get computing resources that are on-demand and self-service.
- Customers get access to those resources over the internet from anywhere they have a connection.
- The cloud provider has a big pool of those resources and allocates them to users out of the pool
- The resources are elastic which means they’re flexible, so customers can be.
- Customers pay only for what they use or reserve as they go.

The evolution of cloud computing has seen several waves, each offering unique benefits and capabilities:

1. **Colocation (First Wave):**
    - Colocation allowed users to rent physical space in data centers rather than invest in building their own infrastructure.
    - This provided financial efficiency and flexibility, laying the foundation for the cloud model.
2. **Virtualized Data Centers (Second Wave):**
    - Virtualized data centers resemble traditional private data centers and colocation facilities but with virtualized components.
    - Users maintain control over the infrastructure, which is user-configured and managed.
3. **Container-Based Architecture (Third Wave):**
    - Google pioneered the shift to a container-based architecture, representing the third wave of cloud computing.
    - This architecture offers a fully automated and elastic cloud environment, consisting of automated services and scalable data.
    - Services automatically provision and configure the infrastructure required to run applications.

Google Cloud now offers this third-wave cloud architecture to its customers, emphasizing the importance of technology and software in driving differentiation and competitiveness. It

highlights the significance of data in building high-quality software and suggests that every company, regardless of size or industry, will eventually become a data company.

Cloud Models

The evolution of cloud computing introduced customers to two primary offerings: Infrastructure as a Service (IaaS) and Platform as a Service (PaaS).

1. **Infrastructure as a Service (IaaS):**
    - IaaS delivers on-demand infrastructure resources via the cloud, including raw compute, storage, and network capabilities.
    - Customers allocate resources similar to physical data centers, paying for the resources they allocate ahead of time.
    - Example: Google Cloud's Compute Engine.
2. **Platform as a Service (PaaS):**
    - PaaS binds code to libraries that provide access to the infrastructure application needs, allowing more focus on application logic.
    - Customers pay for the resources they actually use, making it more cost-effective.
    - Example: Google Cloud's App Engine.

As cloud computing has evolved, the focus has shifted towards managed infrastructure and services. Leveraging managed resources allows companies to concentrate more on their business goals, delivering products and services more quickly and reliably.

Serverless Computing:

- Serverless computing further simplifies development by eliminating the need for any infrastructure management.
- Technologies like Google Cloud Functions and Cloud Run offer event-driven code execution and fully-managed containerized microservices deployment.

Software as a Service (SaaS):

- SaaS provides entire cloud-based applications directly over the internet, eliminating the need for local installation.
- Examples include popular Google applications like Gmail, Docs, and Drive, all part of Google Workspace.

Google Network

Google Cloud operates on its expansive global network, which is the largest of its kind and has been built over many years with significant investments from Google. This network is strategically designed to offer customers the highest throughput and lowest latencies for their applications, leveraging over 100 content caching nodes worldwide.

Key aspects of Google Cloud's network infrastructure include:

1. **Content Caching Nodes:** These nodes cache high-demand content globally, ensuring quicker access for applications and faster response times to user requests.
2. **Global Locations:** Google Cloud's infrastructure spans five major geographic locations: North America, South America, Europe, Asia, and Australia. This global presence ensures services can reach users worldwide, optimizing availability, durability, and latency.
3. **Regions and Zones:** Each location is divided into regions, which represent independent geographic areas, further divided into zones where Google Cloud resources are deployed. For example, London (europe-west2) is a region comprising multiple zones. Deploying resources across multiple regions enhances redundancy and protects against region-wide issues.
4. **Multi-Region Support:** Some Google Cloud services support multi-region configurations, such as Cloud Spanner. These configurations replicate data across multiple zones and regions, enabling low-latency access to data from various locations, enhancing performance and resilience.

Currently, Google Cloud offers 118 zones across 39 regions, with continuous expansion efforts. This extensive network infrastructure ensures reliable and efficient service delivery to customers globally.

Security on Google Cloud

Google prioritizes security across all layers of its infrastructure to ensure the safety of customer data. Here's an overview of the security measures implemented at each layer:

1. **Hardware Infrastructure Layer:**
    - **Hardware Design and Provenance:** Google custom-designs server boards and networking equipment, including custom chips for enhanced security.
    - **Secure Boot Stack:** Servers use cryptographic signatures to ensure they boot the correct software stack securely.
    - **Premises Security:** Google's data centers incorporate multiple layers of physical security, with restricted access limited to a small number of authorized personnel.
2. **Service Deployment Layer:**
    - **Encryption of Inter-Service Communication:** Google's infrastructure encrypts all RPC traffic between services, ensuring cryptographic privacy and integrity for data transmission.
3. **User Identity Layer:**
    - **Central Identity Service:** Google's identity service employs intelligent challenges based on risk factors, and supports secondary factors like U2F-based devices for user authentication.
4. **Storage Services Layer:**
    - **Encryption at Rest:** Data stored on Google's storage services is encrypted using centrally managed keys, with support for hardware encryption in storage devices.

5. **Internet Communication Layer:**
    - **TLS Encryption:** Google Front End terminates TLS connections using public-private key pairs and X.509 certificates, ensuring secure communication over the internet.
    - **DoS Protection:** Google implements multi-tier, multi-layer DoS protections to absorb and mitigate attacks against its services.
6. **Operational Security Layer:**
    - **Intrusion Detection:** Google employs rules and machine intelligence for early warning of possible security incidents, conducting Red Team exercises for continuous improvement.
    - **Insider Risk Reduction:** Google actively monitors employees with administrative access and limits their activities.
    - **Employee U2F Use:** Google requires employees to use U2F-compatible Security Keys to mitigate phishing attacks.
    - **Software Development Practices:** Google enforces stringent development practices, including two-party code review and libraries to prevent security bugs. It also runs a Vulnerability Rewards Program to incentivize bug reporting.

These comprehensive security measures reflect Google's commitment to safeguarding customer data and maintaining the integrity of its infrastructure.

GCP billing Structure

Google Cloud offers a flexible pricing structure designed to accommodate various usage scenarios while providing cost-effective solutions. Here's an overview of some key aspects of Google Cloud's pricing:

1. **Per-Second Billing:** Google Cloud was the first major provider to introduce per-second billing for its compute offerings, including Compute Engine, Google Kubernetes Engine (GKE), Dataproc, and App Engine flexible environment VMs. This ensures that users only pay for the resources they use, down to the second.
2. **Sustained-Use Discounts:** Compute Engine offers automatically applied sustained-use discounts for instances that run for more than 25% of a month. This discount applies to every incremental minute used beyond the threshold.
3. **Custom Virtual Machine Types:** Compute Engine allows users to create custom virtual machine types tailored to their specific workload requirements, optimizing vCPU and memory allocation. This flexibility enables users to optimize pricing based on their application needs.
4. **Pricing Calculator:** Google Cloud provides an online pricing calculator to help estimate costs based on resource usage. Users can adjust parameters to get a clear understanding of potential expenses.
5. **Budgets and Alerts:** Users can set budgets at the billing account or project level, defining fixed limits or tying budgets to specific metrics like a percentage of the previous month's

spend. Alerts can be configured to notify users when expenses approach budget limits, helping to prevent unexpected charges.

6. **Reports:** The Reports tool in the Google Cloud Console allows users to monitor expenditure based on projects or services, providing visual insights into usage patterns and costs.
7. **Quotas:** Google Cloud implements quotas to prevent over-consumption of resources due to errors or malicious attacks. There are two types of quotas: rate quotas, which reset after a specific time, and allocation quotas, which govern the number of resources allowed in a project. Users can request quota increases from Google Cloud Support if needed.

Google Cloud Hierarchy

Google Cloud's resource hierarchy is structured into four levels: resources, projects, folders, and organization nodes.

1. Resources

- Represent virtual machines, Cloud Storage buckets, tables in BigQuery, or any other resources in Google Cloud.
- Form the base level of the hierarchy.

2. Projects

- Organize resources into logical groupings.
- Can be further organized into folders or subfolders.
- Serve as the basis for enabling and utilizing Google Cloud services.

Project ID	Project Name	Project Number
- Globally unique identifier assigned by Google.	- User-created name for the project.	- Unique identifier assigned by Google to the project.
- Immutable; cannot be changed after creation.	- Not required to be unique.	- Used internally by Google Cloud for resource management.
- Used in various contexts to identify the project within Google Cloud.	- Can be changed at any time.	

3. Folders

- Allow assignment of policies and permissions to resources with granular control.
- Contain projects, other folders, or a combination of both.
- Useful for grouping resources based on departmental or team structures.

4. Organization Node

- Top level of the hierarchy, encompassing all projects, folders, and resources.
- Provides centralized management and governance for the entire organization's Google Cloud environment.

- Special roles can be assigned, such as organization policy administrator and project creator.

Key Points

- Policies can be defined and inherited at project, folder, and organization node levels.
- Resource Manager tool helps programmatically manage projects, including creation, updating, and deletion.
- Folder structure facilitates delegation of administrative rights and simplifies permission management.
- Organization node creation depends on Google Workspace domain or can be generated using Cloud Identity.
- Once created, the organization node allows domain members to create projects and billing accounts.

Identity and Access Management (IAM) Overview

To manage access to resources within Google Cloud's complex hierarchy, administrators utilize Identity and Access Management (IAM) policies. IAM enables administrators to define who can perform specific actions on which resources.

Key Concepts:

1. **Principals**: Represent entities granted access, such as Google accounts, Google groups, service accounts, or Cloud Identity domains. Each principal has a unique identifier, typically an email address.
2. **Roles**: Collections of permissions that define what actions a principal can perform. Roles are assigned to principals to grant permissions. Roles are grouped to simplify management and understanding of permissions.
3. **Policy Application**: When a role is assigned to a principal on a specific resource, the policy applies to that resource and all resources below it in the hierarchy. Policies can include both allow and deny rules.
4. **Role Types**:
    - **Basic Roles**: Broad in scope and affect all resources within a Google Cloud project. Includes roles such as owner, editor, viewer, and billing administrator.
        - Owner: Full access to resources, including permission to manage roles and billing.
        - Editor: Access to resources and ability to make changes.
        - Viewer: Access to resources without permission to modify.
        - Billing Administrator: Manages billing without resource modification rights.
    - **Predefined Roles**: Sets of permissions defined by specific Google Cloud services. Offered for services like Compute Engine, allowing granular control over actions within that service.

- **Custom Roles**: Allow organizations to define precise permissions tailored to specific job roles or requirements. Used in "least-privilege" models to grant minimal necessary access.

Considerations:

- Basic roles may be too broad for projects containing sensitive data. Predefined and custom roles offer more granular control.
- Custom roles require management of permissions defining the role, and they can only be applied at the project or organization level, not at the folder level.

IAM provides a flexible and granular approach to access management, ensuring that organizations can enforce security policies while empowering users with appropriate levels of access to resources within Google Cloud.

Service Accounts in Google Cloud

Service accounts are used to grant permissions to applications or virtual machines (VMs) rather than individual users. They are particularly useful when you want to restrict access to specific resources to only certain applications or VMs.

Key Points:

1. **Purpose**: Service accounts authenticate applications or VMs to access Google Cloud resources securely.
2. **Authentication**: Service accounts are identified by email addresses and authenticate using cryptographic keys instead of passwords.
3. **Example Use Case**: Suppose you have an application running on a VM that needs to interact with Cloud Storage. You can create a service account and grant it access to Cloud Storage, ensuring that only that specific VM can access the data.
4. **Permissions**: Service accounts can be assigned roles, just like other Google Cloud resources. For example, granting a service account the Compute Engine's Instance Admin role allows the application running on the associated VM to manage other VMs.
5. **Management**: Service accounts need to be managed, including defining who can create or manage them. IAM policies can be attached to service accounts, allowing granular control over access.
6. **Example Scenario**: Alice might have the editor role on a service account, allowing her to manage its permissions, while Bob might have the viewer role, enabling him to view information about the service accounts.

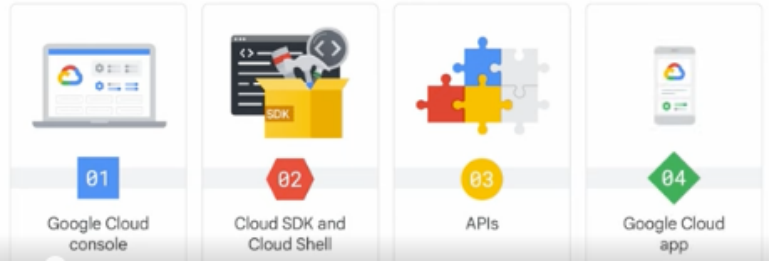
Considerations:

- Service accounts provide a secure way to authenticate applications or VMs to access Google Cloud resources.
- IAM policies can be attached to service accounts to control access permissions.
- Service accounts require management to ensure proper configuration and access control.

Using service accounts, organizations can ensure that only authorized applications or VMs have access to specific resources within Google Cloud, enhancing security and control over data access.

Interaction with Google Cloud

1. **Cloud Console**:
    - Graphical User Interface (GUI) for deploying, scaling, and diagnosing issues.
    - Easily manage resources, check their health, and set budgets.
    - Includes search functionality and SSH access to instances via browser.
2. **Cloud SDK and Cloud Shell**:
    - **Cloud SDK**: Set of tools for managing resources and applications.
    - **Cloud Shell**: Command-line access to cloud resources from a browser.
        - Includes Cloud SDK tools and a 5GB persistent home directory.
3. **Application Programming Interfaces (APIs)**:
    - Services offer APIs for programmatic control.
    - Google APIs Explorer helps discover and test APIs interactively.
    - Cloud Client libraries available in various languages for easier integration.
4. **Google Cloud App**:
    - Mobile app for starting, stopping, and connecting to Compute Engine instances.
    - Provides access to logs, Cloud SQL instance management, and App Engine administration.
    - Offers billing information, customizable metrics graphs, alerts, and incident management.



Understanding Virtual Private Clouds (VPC) in Google Cloud

Google Compute Engine operates within a framework of virtual networking, primarily facilitated by Virtual Private Clouds (VPC). Let's delve into its workings:

1. What is a Virtual Private Cloud (VPC)?

- A VPC is a secure, private cloud-computing environment hosted within a public cloud infrastructure, like Google Cloud.

- It allows users to run code, store data, host websites, and perform other cloud-based activities within a secluded, controlled environment.

2. Key Features of VPC:

- **Data Isolation**: Combines the scalability of public cloud computing with the data isolation of private cloud computing.
- **Connectivity**: VPC networks connect Google Cloud resources to each other and to the internet, enabling seamless communication.
- **Segmentation**: Networks can be segmented using firewall rules to restrict access to instances and create static routes for traffic forwarding.

3. Global Nature of Google VPC Networks:

- Google VPC networks are global in scope, allowing for consistent connectivity across regions worldwide.
- Subnets, segmented pieces of the network, can be deployed in any Google Cloud region.
- Subnets can span zones within a region, facilitating flexible network layouts with global reach.

4. Scalability and Resilience:

- Subnet sizes can be adjusted by expanding the range of IP addresses without affecting existing configurations.
- Compute Engine VMs within the same subnet can be located in different zones, enabling resilient solutions while maintaining network simplicity.

Exploring Google Compute Engine: Infrastructure as a Service (IaaS)

Google Compute Engine offers a robust Infrastructure as a Service (IaaS) solution for creating and running virtual machines (VMs) on Google infrastructure. Let's delve into its features and pricing structure:

1. Key Features of Compute Engine:

- **Flexibility**: Users can create and configure VM instances tailored to their specific requirements, including CPU power, memory, storage, and operating system.
- **Ease of Use**: VM instances can be created via the Google Cloud console, CLI, or API, offering flexibility in deployment.
- **Operating System Support**: Supports Linux and Windows Server images provided by Google, as well as customized versions and images of other operating systems.
- **Cloud Marketplace**: Offers pre-configured solutions from Google and third-party vendors, simplifying deployment without manual configuration.

2. Pricing and Billing Structure:

- **Billing by the Second**: Compute Engine bills usage by the second with a one-minute minimum, providing cost efficiency.
- **Sustained-Use Discounts**: Discounts are automatically applied to VM instances based on usage duration, with higher discounts for longer-running instances.
- **Committed-Use Discounts**: Offers discounts of up to 57% for stable workloads with committed usage terms of one or three years.
- **Preemptible and Spot VMs**: Cost-effective options for non-critical workloads, offering savings of up to 90%. Preemptible VMs can be terminated by Compute Engine, while Spot VMs offer more features and flexibility.
- **Storage**: Default high throughput between processing and persistent disks at no extra cost.
- **Custom Machine Types**: Allows users to tailor machine properties such as virtual CPUs and memory, offering flexibility and cost optimization.

Google Compute Engine provides users with the flexibility to deploy and manage virtual machines according to their specific needs, with a pricing structure that offers cost efficiency and scalability. By leveraging features such as sustained-use discounts and cost-effective VM options, users can optimize their infrastructure costs while benefiting from Google Cloud's powerful computing capabilities.

Auto Scaling and load Balancing

Google Compute Engine offers powerful features for optimizing performance and managing workloads efficiently:

1. Autoscaling:

- With Autoscaling, VM instances can be dynamically added or removed from an application based on predefined load metrics.
- This feature ensures that resources are efficiently utilized, scaling up during periods of high demand and scaling down during low demand.
- Autoscaling helps maintain optimal performance and cost efficiency by automatically adjusting the number of VMs based on workload requirements.

2. Load Balancing:

- Load balancing is crucial for distributing incoming traffic among VM instances to ensure optimal performance and reliability.
- Google's Virtual Private Cloud (VPC) supports various load balancing options tailored to different use cases.
- These load balancing options help evenly distribute traffic, prevent overloading of individual instances, and improve application responsiveness.

3. VM Configuration:

- Compute Engine allows users to configure VM instances with the most appropriate machine properties, such as the number of virtual CPUs and the amount of memory.

- Users can choose from predefined machine types or create custom machine types to tailor resources to their specific needs.
- Large VM configurations are available for specialized workloads like in-memory databases and CPU-intensive analytics, although most users initially opt for scaling out rather than scaling up.

4. Resource Constraints:

- The maximum number of CPUs per VM is determined by its "machine family" and is subject to quota limitations, which vary based on the zone.
- Users can refer to the specifications for currently available VM machine types on the Google Cloud documentation website for detailed information.

By leveraging Autoscaling and Load Balancing features, Google Compute Engine empowers users to efficiently manage workloads, optimize performance, and ensure high availability for their applications. Users can dynamically adjust resources based on demand, scale horizontally to meet growing needs, and distribute traffic effectively to maintain responsiveness and reliability.

Exploring Virtual Private Cloud (VPC) Compatibility Features

Google Cloud's Virtual Private Cloud (VPC) offers a range of compatibility features to streamline network management and enhance security:

1. Routing Tables:

- VPCs come with built-in routing tables, eliminating the need for manual provisioning or management of routers.
- These routing tables facilitate the forwarding of traffic within the same network, across subnetworks, and between Google Cloud zones, without requiring external IP addresses.

2. Distributed Firewall:

- VPCs feature a global distributed firewall, removing the necessity to provision or manage separate firewalls.
- Administrators can control access to instances by configuring firewall rules for both incoming and outgoing traffic.
- Firewall rules can be defined based on network tags assigned to Compute Engine instances, providing convenience and flexibility in access control.

3. Network Tags:

- Firewall rules can be defined using network tags assigned to Compute Engine instances.
- For example, all web servers can be tagged with "WEB," and a firewall rule can be configured to allow traffic on ports 80 or 443 into instances with the "WEB" tag, regardless of their IP addresses.

4. VPC Peering:

- VPC Peering enables the establishment of relationships between two VPCs to exchange traffic.
- It facilitates communication between VPCs within the same or across different Google Cloud projects.

5. Identity Access Management (IAM) Integration:

- IAM allows fine-grained control over who and what can interact with VPC resources.
- Administrators can leverage IAM policies to manage access permissions for resources across multiple Google Cloud projects.

Google Cloud's VPC compatibility features ensure efficient network management, robust security controls, and seamless communication between VPCs within and across projects. By leveraging these features, organizations can establish secure and scalable network architectures tailored to their specific requirements.

Cloud Load Balancing in Google Cloud

Cloud Load Balancing in Google Cloud is a powerful solution for distributing user traffic across multiple instances of an application, ensuring optimal performance and availability. Here's what you need to know:

1. Purpose and Benefits:

- Cloud Load Balancing distributes user traffic to maintain application performance and availability, regardless of fluctuations in load.
- It is a fully distributed, software-defined, managed service, eliminating the need for users to manage and scale load balancers.

2. Supported Protocols:

- Cloud Load Balancing supports various protocols, including HTTP, HTTPS, TCP, SSL, and UDP, making it suitable for a wide range of applications and services.

3. Cross-Region Load Balancing:

- Offers cross-region load balancing with automatic multi-region failover, ensuring continuous service availability even in the event of backend failures.

4. Dynamic Load Balancing:

- Reacts quickly to changes in user traffic, network conditions, and backend health, adjusting load distribution accordingly.

5. Scalability:

- Does not require pre-warming or manual scaling adjustments, making it suitable for handling sudden spikes in demand without the need for user intervention.

6. Load Balancing Options:

- Google Cloud offers a suite of load-balancing options tailored to different use cases:

- Global HTTP(S) load balancing for web applications.
- Global SSL Proxy load balancing for non-HTTP SSL traffic.
- Global TCP Proxy load balancing for other TCP traffic.
- Regional External Passthrough Network load balancing for UDP traffic and traffic on any port number.
- Regional External Application load balancer and Proxy Network load balancer for traffic from the internet.
- Regional Internal load balancer for internal traffic between components within a project.
- Cross-region Internal load balancer for globally distributed backend services with traffic management capabilities.

Cloud Load Balancing in Google Cloud offers a comprehensive set of features and options for effectively managing and optimizing traffic distribution across applications and services, ensuring high performance, scalability, and reliability.

Google Cloud DNS and Cloud CDN

Google Cloud offers robust solutions for optimizing content delivery and managing Domain Name System (DNS) services. Here's an overview of these services:

1. Google Public DNS (8.8.8.8):

- Google Public DNS is a widely used, free DNS resolver service provided by Google.
- It translates internet hostnames to IP addresses and is available for public use.

2. Cloud DNS:

- Cloud DNS is a managed DNS service offered by Google Cloud.
- It runs on Google's infrastructure, ensuring low latency and high availability.
- Cloud DNS allows you to publish and manage DNS zones and records programmatically.
- The DNS information is served from redundant locations globally, ensuring reliability and performance.

3. Edge Caching and Cloud CDN:

- Google operates a global system of edge caches, which store content closer to end users.
- Edge caching accelerates content delivery by reducing network latency and offloading traffic from origin servers.
- Cloud CDN (Content Delivery Network) leverages this infrastructure to deliver content faster to users.
- Enabling Cloud CDN with a single checkbox after setting up HTTP(S) Load Balancing improves user experience, reduces server load, and can lower costs.

4. CDN Interconnect Partner Program:

- Google Cloud's CDN Interconnect partner program allows customers to integrate with other CDN providers seamlessly.
- If you are already using another CDN, it may be part of Google Cloud's CDN Interconnect partner program, allowing you to continue using it alongside Google Cloud services.

Google Cloud DNS and Cloud CDN provide essential tools for optimizing content delivery, improving user experience, and managing DNS services effectively within Google Cloud infrastructure.

Connecting Networks with Google Cloud: Options and Considerations

Connecting Google Virtual Private Cloud (VPC) networks to other networks, such as on-premises networks or networks in other clouds, can be achieved through several effective methods. Here's an overview of these options:

1. Cloud VPN with Cloud Router:

- Establishes a Virtual Private Network (VPN) connection over the internet.
- Cloud Router enables dynamic exchange of route information between networks using Border Gateway Protocol (BGP).
- Automatically propagates route updates, allowing seamless communication between networks.

2. Carrier Peering:

- Provides direct access from on-premises networks through a service provider's network to Google Workspace and Cloud products.
- Offers connectivity to Google's network through a partner's infrastructure.
- Facilitates exchange of traffic between networks via a service provider's network.
- Allows for connectivity to Google's services using one or more public IP addresses.

3. Direct Peering:

- Places a router in the same public data center as a Google point of presence to exchange traffic.
- Google has over 100 points of presence worldwide.
- Customers not in a point of presence can connect through partners in the Carrier Peering program.
- Offers direct access to Google Workspace and Cloud products.

4. Dedicated Interconnect:

- Provides one or more direct, private connections to Google.
- Offers high reliability with an SLA of up to 99.99%.

- Can be backed up by a VPN for added reliability.

5. Partner Interconnect:

- Establishes connectivity between on-premises networks and VPC networks through a supported service provider.
- Useful for locations unable to reach a Dedicated Interconnect colocation facility.
- Offers flexible configurations based on availability needs.

6. Cross-Cloud Interconnect:

- Enables high-bandwidth dedicated connectivity between Google Cloud and other cloud service providers.
- Provisions a dedicated physical connection between networks.
- Supports an integrated multicloud strategy, reducing complexity and enabling site-to-site data transfer.
- Available in 10 Gbps or 100 Gbps sizes.

Each connectivity option has its considerations regarding reliability, availability, scalability, and cost. Customers should choose the option that best aligns with their specific requirements and infrastructure setup. Additionally, it's important to consider factors such as SLAs, service provider partnerships, and network topology when selecting a connectivity solution.

Storage In Cloud

In this section of the course, we will explore Google Cloud's five core storage products: Cloud Storage, Cloud SQL, Cloud Spanner, Firestore, and Cloud Bigtable. These storage solutions cater to various data storage needs, including structured, unstructured, transactional, and relational data. Depending on the specific requirements of your application, you may choose to utilize one or multiple of these services to meet your storage needs.

Cloud Storage Overview

Cloud Storage is a managed object storage service provided by Google Cloud. It offers durable and highly available storage for a wide range of data types, including media files, backups, and archival data. This document provides an overview of Cloud Storage, covering its key features, use cases, and management capabilities.

Object Storage Architecture

Cloud Storage utilizes an object storage architecture, managing data as discrete objects rather than as part of a file hierarchy or disk blocks. Each object consists of:

- **Binary Data**: The actual content of the object.
- **Metadata**: Descriptive information about the object, such as creation date, author, and permissions.

- **Unique Identifier**: A globally unique identifier (URL) for referencing the object.

Key Features

- **Scalability**: Cloud Storage can accommodate any amount of data, making it suitable for both small-scale and large-scale storage needs.
- **Durability**: Data stored in Cloud Storage is replicated across multiple locations, ensuring high durability and availability.
- **Flexibility**: Cloud Storage supports a wide variety of use cases, including serving website content, storing backups, and distributing large data objects.
- **Immutability and Versioning**: Objects in Cloud Storage are immutable, meaning changes result in new versions. Administrators can enable versioning to track modifications and restore previous object states.
- **Access Control**: Cloud Storage provides fine-grained access control using IAM roles and access control lists (ACLs) to ensure data security and privacy.
- **Lifecycle Management**: Administrators can define lifecycle policies to manage object retention and deletion based on specified criteria, optimizing storage costs.

Use Cases

- **Web Content Hosting**: Cloud Storage is commonly used to host static website content, such as images, videos, and HTML files.
- **Backup and Archiving**: Organizations use Cloud Storage to store backups of critical data and archival data that needs to be retained for compliance or historical purposes.
- **Data Distribution**: Cloud Storage is used to distribute large data objects to end users via Direct Download links, enabling efficient content delivery.
- **Processing Workflows**: Cloud Storage serves as a storage layer for intermediate results in data processing workflows, providing scalability and durability.

Management and Administration

- **Buckets**: Data in Cloud Storage is organized into buckets, each requiring a globally unique name and a specified geographic location for storage.
- **Access Control**: Administrators can manage access to buckets and objects using IAM roles and access control lists (ACLs), ensuring that users have appropriate permissions.
- **Lifecycle Policies**: Cloud Storage offers lifecycle management policies to control object retention and deletion based on defined criteria, helping to optimize storage costs.

This structured and organized document provides a comprehensive overview of Cloud Storage, covering its architecture, key features, use cases, and management capabilities.

Primary Storage Classes in Cloud Storage

Cloud Storage offers four primary storage classes tailored to different data access patterns and cost requirements:

1. Standard Storage

- Best for frequently accessed ("hot") data or data stored for short periods.
- Suitable for applications requiring high availability and low latency.

2. Nearline Storage

- Ideal for infrequently accessed data, typically accessed once a month or less.
- Suited for data backups, long-tail multimedia content, and data archiving.

3. Coldline Storage

- Provides a low-cost option for storing infrequently accessed data.
- Intended for data accessed or modified at most once every 90 days.

4. Archive Storage

- Lowest-cost option designed for data archiving, online backup, and disaster recovery.
- Suitable for data accessed less than once a year, with higher costs for data access and operations.

Common Characteristics

- Unlimited storage with no minimum object size requirement.
- Worldwide accessibility and locations.
- Low latency, high durability, and uniform experience in terms of security, tools, and APIs.
- Geo-redundancy in multi-region or dual-region setups for disaster recovery and optimal performance.

Data Transfer Options

- **gcloud storage**: Command-line tool for online transfer.
- **Drag and drop**: Option available in Cloud Console for small-scale transfers.
- **Storage Transfer Service**: Enables quick and cost-effective batch transfers from other providers or endpoints upto TB.
- **Transfer Appliance**: High-capacity storage server leased from Google for transferring large data volumes, up to a petabyte.

Integration with Google Cloud Services

- Import and export tables from BigQuery and Cloud SQL.
- Store App Engine logs, Firestore backups, and application objects.
- Store instance startup scripts, Compute Engine images, and objects used by Compute Engine applications.

This structured format provides a clear overview of the primary storage classes in Cloud Storage, their characteristics, and various data transfer options available.

Cloud SQL: Fully Managed Relational Databases

Cloud SQL is a fully managed relational database service provided by Google Cloud, offering support for MySQL, PostgreSQL, and SQL Server:

- **Managed Service**: Handles tasks like patching, updates, backups, and replication configuration, allowing developers to focus on application development.
- **No Software Maintenance**: Requires no software installation or maintenance, providing a hassle-free experience.
- **Scalability**: Can scale up to 128 processor cores, 864 GB of RAM, and 64 TB of storage to accommodate growing needs.
- **Automatic Replication**: Supports various replication scenarios, including internal and external instances, ensuring data redundancy and high availability.
- **Managed Backups**: Includes managed backups with seven included backups per instance, ensuring data security and accessibility for restores.
- **Data Encryption**: Encrypts customer data on Google's internal networks and in database tables, temporary files, and backups, ensuring data security.
- **Network Firewall**: Includes a network firewall to control network access to each database instance, enhancing security.
- **Integration**: Accessible by other Google Cloud services and external services, facilitating seamless integration into existing workflows.
- **Compatibility**: Compatible with various applications and tools, including App Engine, Compute Engine, and external applications, using standard drivers and connectors.

Integration and Compatibility

- **App Engine**: Accessible using standard drivers like Connector/J for Java or MySQLdb for Python.
- **Compute Engine**: Instances can be authorized to access Cloud SQL instances and configured to reside in the same zone.
- **Other Tools**: Supports applications and tools like SQL Workbench, Toad, and other external applications using standard MySQL drivers.

This structured format provides a comprehensive overview of Cloud SQL's features, benefits, and compatibility, making it easier to understand its capabilities and integration options.

Cloud Spanner: Fully Managed, Horizontally Scalable Relational Database

Cloud Spanner is a fully managed relational database service offered by Google Cloud. Here's a breakdown of its key features:

- **Horizontal Scalability**: Cloud Spanner scales horizontally to handle increasing workloads seamlessly, making it suitable for growing applications.

- **Strong Consistency**: Offers strong global consistency, ensuring that data is always up-to-date and accurate across all replicas.
- **SQL Compatibility**: Supports SQL queries and transactions, making it familiar and easy to use for developers accustomed to relational databases.
- **High Availability**: Built-in high availability ensures that your data is always accessible, minimizing downtime and ensuring business continuity.
- **High Performance**: Capable of handling high numbers of input and output operations per second (IOPS), making it ideal for high-throughput applications.
- **Battle-Tested**: Used by Google's own mission-critical applications and services, demonstrating its reliability and scalability.
- **Suitable Use Cases**: Ideal for applications requiring a SQL relational database management system with joins, secondary indexes, and high levels of performance and consistency.

Cloud Spanner is particularly well-suited for applications with demanding requirements, including:

- **High Throughput**: Capable of handling tens of thousands of reads and writes per second or more.
- **Global Applications**: Applications with a global presence that require strong consistency and low latency across regions.
- **Transaction-Intensive Workloads**: Applications with transaction-heavy workloads that require ACID (Atomicity, Consistency, Isolation, Durability) compliance.

This structured format provides a clear and concise overview of Cloud Spanner's features and suitability for various use cases, making it easier to understand its capabilities and benefits.

Firestore: NoSQL Cloud Database for Flexible Development

Firestore is a versatile NoSQL cloud database provided by Google Cloud, designed to cater to the needs of mobile, web, and server development projects. Below is a structured and summarized overview of Firestore's features and pricing for documentation purposes:

1. **Database Structure**:
    - Data stored in documents organized into collections.
    - Supports complex nested objects and subcollections within documents.
    - Each document contains key-value pairs, allowing for flexible data representation.
2. **Query Capabilities**:
    - Firestore supports powerful NoSQL queries to retrieve specific documents or collections.
    - Queries can include multiple filters and sorting options, providing flexibility in data retrieval.
    - Automatic indexing ensures query performance is proportional to the result set size, not the dataset.

3. **Data Synchronization**:
    - Facilitates seamless data synchronization across connected devices.
    - Enables offline access to data, with automatic sync upon device reconnection.
    - Caches actively used data locally on devices for efficient read, write, listen, and query operations.
4. **Google Cloud Infrastructure**:
    - Leverages Google Cloud's robust infrastructure for automatic multi-region data replication.
    - Offers strong consistency guarantees, atomic batch operations, and real transaction support for data operations.
5. **Usage-Based Pricing**:
    - Firestore charges for document reads, writes, and deletes, as well as storage consumption.
    - Queries incur charges at the rate of one document read per query.
    - Certain network bandwidth usage is also subject to charges.
    - Ingress is free, and egress may be free in many cases.
6. **Free Quotas**:
    - Provides a free daily quota for document reads, writes, deletes, and stored data.
    - Developers can start using Firestore with minimal or no cost, making it accessible for experimentation and development.

Overall, Firestore offers a powerful and cost-effective solution for applications requiring flexible data storage, synchronization, and querying capabilities, backed by Google Cloud's reliable infrastructure.

Cloud Bigtable: Google's NoSQL Big Data Database Service

Cloud Bigtable is a robust NoSQL big data database service provided by Google Cloud, powering many of Google's core services such as Search, Analytics, Maps, and Gmail. Here's a structured and summarized overview of Cloud Bigtable's features and use cases for documentation purposes:

1. **Database Service Overview**:
    - Designed to handle massive workloads with consistent low latency and high throughput.
    - Ideal for operational and analytical applications, including Internet of Things (IoT), user analytics, and financial data analysis.
2. **Key Use Cases**:
    - Suitable for datasets exceeding 1TB of semi-structured or structured data.
    - Efficient for fast data with high throughput or rapidly changing data.

- Designed for NoSQL data operations, particularly transactions where strong relational semantics are not necessary.
- Effective for time-series data or data with natural semantic ordering.
- Well-suited for big data processing, including asynchronous batch or synchronous real-time processing, and running machine learning algorithms.

3. **Integration with Google Cloud Services**:
   - Cloud Bigtable seamlessly interacts with other Google Cloud services and third-party clients.
   - APIs enable reading from and writing to Cloud Bigtable through various data service layers, such as Managed VMs, the HBase REST Server, or Java Servers using the HBase client.
   - Popular stream processing frameworks like Dataflow Streaming, Spark Streaming, and Storm can stream data into Cloud Bigtable.
   - Batch processes like Hadoop MapReduce, Dataflow, or Spark can also read from and write to Cloud Bigtable.
4. **Data Serving and Processing**:
   - Typically used to serve data to applications, dashboards, and data services.
   - Data can be streamed in real-time or processed in batch, with summarized or newly calculated data often written back to Cloud Bigtable or downstream databases.

Cloud Bigtable provides developers and organizations with a scalable and high-performance solution for managing large-scale datasets, enabling efficient data processing and serving for a wide range of applications and use cases.

Comparing Storage Options

Storage Option	Suitable Use Cases	Maximum Capacity	SQL Support	Horizontal Scalability	Best for
Cloud Storage	Immutable blobs larger than 10MB, e.g., images, movies	Petabytes, max object size 5TB	No	No	Storing large, immutable data objects
Cloud SQL	Online transaction processing systems requiring full SQL support	Up to 64TB (Cloud SQL) or petabytes (Cloud Spanner)	Yes	Limited to read replicas (Cloud SQL), Yes (Cloud Spanner)	Storing user credentials, customer orders
Cloud Spanner	Applications needing horizontal scalability and	Petabytes	Yes	Yes	Applications requiring high scalability and strong consistency

Storage Option	Suitable Use Cases	Maximum Capacity	SQL Support	Horizontal Scalability	Best for
	full SQL support				
Firestore	Applications needing massive scaling, real-time query results, and offline query support	Terabytes, max entity size 1MB	No	Yes	Storing, syncing, and querying data for mobile and web apps
Cloud Bigtable	Storing a large number of structured objects with heavy read and write events	Petabytes, max cell size 10MB, max row size 100MB	No	No	Analytical data like AdTech, financial, or IoT data

Introduction to Containers

1. **Introduction to Containers**: Containers provide a lightweight and scalable alternative to traditional virtual machines (VMs). While VMs virtualize hardware resources, containers abstract the operating system (OS) and hardware, allowing for independent scalability of workloads.
2. **Flexibility and Efficiency**: Containers offer flexibility by allowing developers to package their code along with its dependencies, runtime, and configurations into a self-contained environment. Unlike VMs, containers require minimal system resources and start quickly, making them efficient and cost-effective.
3. **Comparison with Virtual Machines**: While VMs deploy entire guest OS instances, which can be large and slow to boot, containers encapsulate code and dependencies with limited access to the file system and hardware. This enables rapid deployment and scalability, as containers can start as quickly as processes.
4. **Portability and Consistency**: Containers facilitate portability, enabling seamless movement of applications across different environments, from development to production or from local machines to the cloud. With containers, developers can ensure consistency in application behavior across various platforms.
5. **Scaling and Modularization**: Containers allow for easy scaling of applications by deploying multiple instances of containers, either on a single host or across multiple hosts. Applications can be built using microservices architecture, where each container performs a specific function, promoting modularity and independent scalability.
6. **Dynamic Resource Management**: Containerized applications can dynamically scale up or down in response to changing demand or host failures. This dynamic resource management ensures optimal utilization of resources and enhances application availability and reliability.

Overall, containers offer a versatile and efficient approach to application development and deployment, providing developers with the agility to build, deploy, and scale applications seamlessly across diverse computing environments.

Kubernetes

1. **Introduction to Kubernetes**: Kubernetes is an open-source platform for orchestrating containerized workloads and services. It simplifies the management of containers by automating deployment, scaling, and operations.
2. **Basic Concepts**:
   - Kubernetes operates using a cluster of nodes, where each node represents a computing instance.
   - Pods are the smallest unit in Kubernetes, representing one or more containers that share networking and storage resources.
   - Deployments manage groups of replica pods, ensuring high availability and fault tolerance.
   - Services provide a stable endpoint for accessing pods, abstracting the dynamic nature of pod IP addresses.
3. **Scaling and Load Balancing**:
   - Kubernetes allows scaling deployments up or down using commands like `kubectl scale` or through autoscaling based on metrics like CPU utilization.
   - Services are associated with load balancers, such as network load balancers in Google Kubernetes Engine (GKE), to distribute traffic among pods.
4. **Declarative Configuration**:
   - While imperative commands like `kubectl run` and `kubectl expose` are useful for learning, Kubernetes emphasizes a declarative approach.
   - Configuration files define the desired state of the application, and Kubernetes handles the implementation details to achieve that state.
5. **Updating and Rollouts**:
   - Kubernetes supports rolling updates, allowing for seamless deployment of new versions without downtime.
   - By updating deployment configuration files and applying changes with `kubectl apply`, Kubernetes orchestrates the creation of new pods while gradually phasing out old ones.

Overall, Kubernetes simplifies container management by providing powerful tools for deploying, scaling, and updating applications, making it a fundamental component of modern cloud-native infrastructure.

Google Kubernetes Engine

Google Kubernetes Engine (GKE) is a managed Kubernetes service provided by Google Cloud, offering the following features and benefits:

1. **Managed Kubernetes Service**: GKE hosts Kubernetes clusters in the cloud, allowing users to deploy and manage containerized applications without worrying about the underlying infrastructure.
2. **Cluster Environment**: GKE clusters are comprised of multiple Compute Engine instances grouped together to form a Kubernetes cluster. Users can customize clusters with various machine types, node counts, and network settings.
3. **Interaction with Kubernetes**: Kubernetes provides the command-line interface (CLI) and API resources for interacting with GKE clusters. Users utilize Kubernetes commands and resources to deploy applications, perform administrative tasks, set policies, and monitor workload health.
4. **Advanced Cluster Management Features**: GKE offers advanced management features, including:
    - Google Cloud's load balancing for Compute Engine instances.
    - Node pools for organizing subsets of nodes within a cluster.
    - Automatic scaling of node instance count to handle changes in workload demand.
    - Automatic upgrades for node software to ensure security and stability.
    - Node auto-repair to maintain node health and availability.
    - Logging and monitoring with Google Cloud's Operations Suite for visibility into cluster performance.
5. **Cluster Creation**: Creating a Kubernetes cluster in GKE is straightforward. Users can use the Google Cloud Console or the `gcloud` command-line tool provided by the Google Cloud SDK. For example, to create a cluster named "k1," users can execute the command: `gcloud container clusters create k1`.

Overall, GKE simplifies Kubernetes cluster management by offering a fully managed solution with advanced features, enabling users to focus on deploying and scaling containerized applications efficiently.

Cloud Run

In this final section of the course, we'll delve into developing applications in the Cloud, starting with Cloud Run, a managed compute platform designed to run stateless containers in response to web requests or Pub/Sub events. Here's what you need to know about Cloud Run:

1. **Serverless Compute Platform**: Cloud Run removes the burden of infrastructure management, allowing developers to focus solely on application development. It leverages Knative, an open API and runtime environment built on Kubernetes, and can be managed on Google Cloud or anywhere Knative runs.
2. **Instantaneous Scaling**: Cloud Run can scale from zero to handle incoming requests almost instantly, ensuring optimal resource utilization. It charges only for resources used, calculated to the nearest 100 milliseconds, eliminating over-provisioning costs.

3. **Developer Workflow**: The Cloud Run developer workflow involves three simple steps:
    - Write your application using your preferred programming language.
    - Build and package your application into a container image.
    - Deploy the container image to Cloud Run, which provides a unique HTTPS URL for accessing your application.
4. **Serverless Benefits**: With Cloud Run, developers can focus on building applications rather than managing infrastructure. It supports both container-based and source-based workflows, allowing flexibility in deployment methods.
5. **Pricing Model**: Cloud Run's pricing model is based on system resource usage while handling web requests, with granularity down to 100 milliseconds. There's no charge for idle containers, and a small fee applies per one million requests served. Container cost increases with CPU and memory usage.
6. **Language Support**: Cloud Run can run applications written in various languages, including popular ones like Java, Python, Node.js, PHP, Go, as well as less common ones like Cobol, Haskell, and Perl, as long as they handle web requests.

Overall, Cloud Run offers developers a seamless and cost-effective solution for deploying and scaling containerized applications, with support for multiple languages and a serverless architecture that simplifies application development and deployment.

Deployment in the cloud

In many applications, certain actions need to be performed in response to specific events, such as uploading an image. Here's how Cloud Functions can help:

1. **Event-Driven Processing**: Cloud Functions allows you to write small, single-purpose functions that respond to cloud events, such as image uploads, without the need to manage servers or runtime environments.
2. **Automatic Invocation**: These functions are automatically triggered whenever a specified event occurs, such as a new image being uploaded. This eliminates the need to provision compute resources continuously, as Cloud Functions execute only in response to events.
3. **Lightweight and Asynchronous**: Cloud Functions is a lightweight, event-based, asynchronous compute solution that enables you to create functions to perform specific tasks in response to cloud events.
4. **Integration with Cloud Services**: Cloud Functions can integrate with other Google Cloud services, allowing you to build application workflows by chaining together individual business logic tasks.
5. **Supported Languages**: You can write Cloud Functions in various programming languages, including Node.js, Python, Go, Java, .NET Core, Ruby, and PHP. Each language runtime has specific versions supported, which can be found in the respective documentation.
6. **Trigger Mechanisms**: Cloud Functions can be triggered asynchronously by events from Cloud Storage and Pub/Sub, or they can be invoked synchronously via HTTP requests.

Overall, Cloud Functions simplifies event-driven processing by providing a serverless compute solution that automatically executes functions in response to specified cloud events, enabling developers to focus on writing code to handle specific tasks without worrying about infrastructure management.