Infrastructure Fundamentals

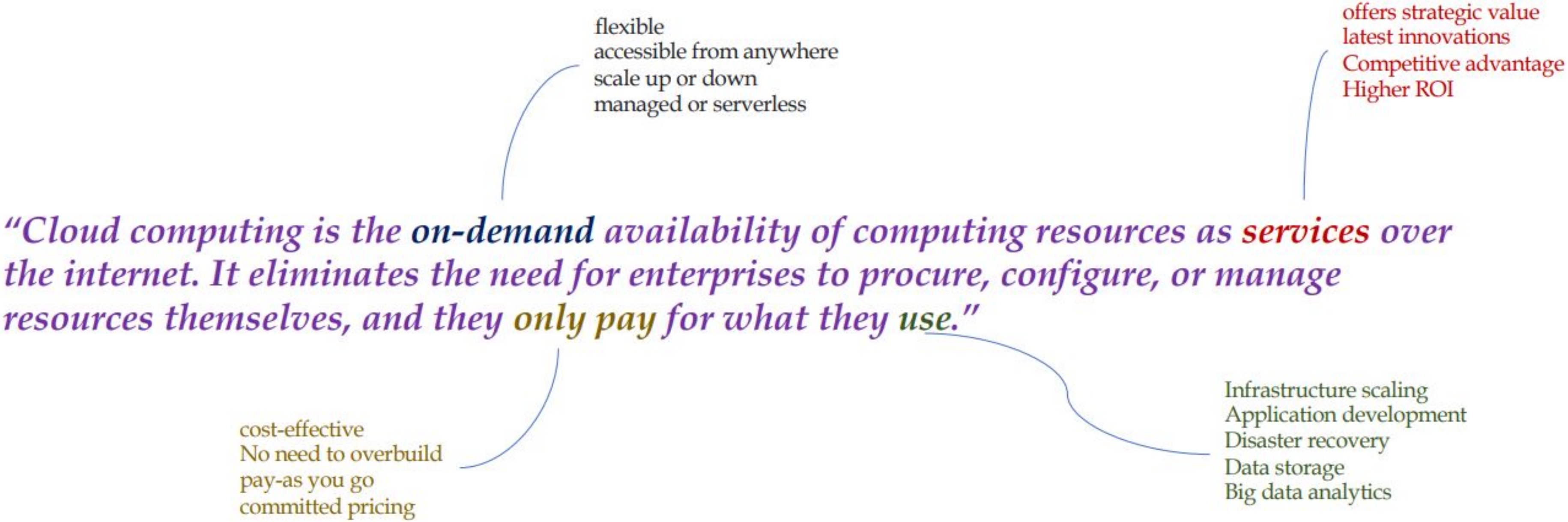
Google Cloud Professional Architect

Agenda

1. INTRODUCING GOOGLE CLOUD
2. GOOGLE COMPUTE ENGINE AND NETWORKING
3. GOOGLE APP ENGINE AND DATABASES
4. GOOGLE CONTAINER ENGINE

INTRODUCING GOOGLE CLOUD

What is cloud computing?



*"Cloud computing is the **on-demand** availability of computing resources as **services** over the internet. It eliminates the need for enterprises to procure, configure, or manage resources themselves, and they only pay for what they use."*

flexible
accessible from anywhere
scale up or down
managed or serverless

offers strategic value
latest innovations
Competitive advantage
Higher ROI

cost-effective
No need to overbuild
pay-as you go
committed pricing

Infrastructure scaling
Application development
Disaster recovery
Data storage
Big data analytics

What is cloud computing?

Three types of cloud computing service models:

- Infrastructure as a service (**IaaS**) offers compute and storage services.
- Platform as a service (**PaaS**) offers a develop-and-deploy environment to build cloud apps.
- Software as a service (**SaaS**) delivers apps as services.

GCP: Services ecosystem

Compute

- App Engine
- Cloud Functions
- Cloud Run
- Compute Engine
- Container-Optimized OS
- GKE On-Prem
- GPU
- Kubernetes Engine

Data Analytics

- BigQuery
- Cloud Composer
- Cloud Data Fusion
- Data Catalog
- Dataflow
- Datalab
- Dataprep
- Dataproc
- Pub/Sub
- Genomics

Networking

- Cloud Armor
- Cloud CDN
- Cloud DNS
- Cloud External IP Addresses
- Cloud Firewall Rules
- Cloud Interconnect
- Cloud Routes
- Cloud Router
- Cloud VPN
- Partner Interconnect
- Premium Network Tier
- Standard Network Tier
- Cloud Load Balancing
- Cloud NAT
- Cloud Network
- Traffic Director
- Virtual Private Cloud

Storage

- Cloud Storage
- Filestore
- Persistent Disk

Databases

- Cloud Bigtable
- Cloud Spanner
- Cloud SQL
- Datastore
- Firestore
- Memorystore

API Management

- API Analytics
- API Monetization
- Apigee API Platform
- Apigee Sense
- Cloud Endpoints
- Developer Portal

Developer Tools

- Cloud Build
- Cloud Code
- Cloud Code for IntelliJ
- Cloud Scheduler
- Cloud SDK
- Cloud Source Repositories
- Cloud Tasks
- Container Registry
- Gradle App Engine Plugin
- IDE plugins
- Maven App Engine Plugin
- Tools for PowerShell
- Tools for Visual Studio
- Cloud Test Lab

AI and Machine Learning

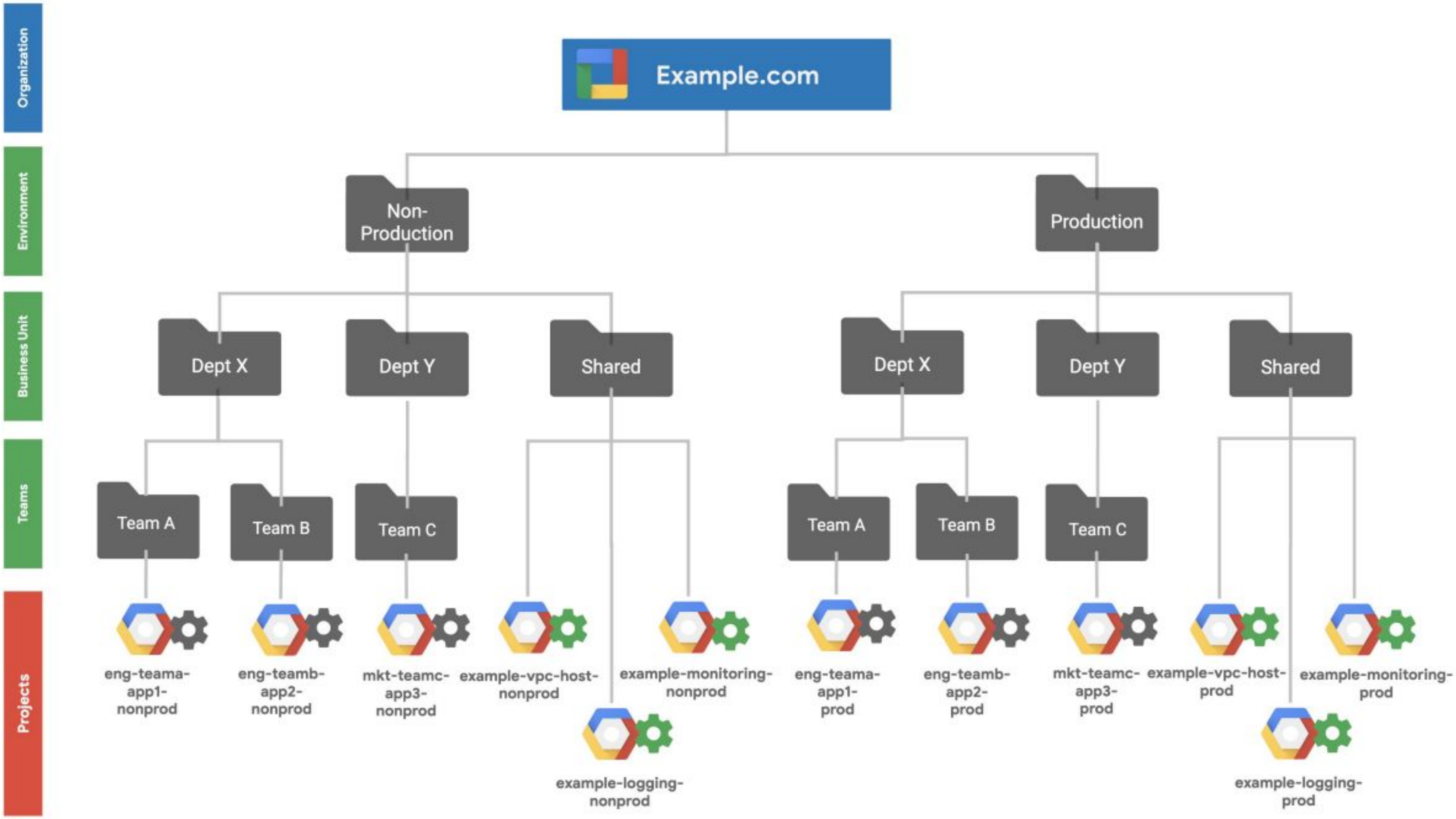
- AI Platform
- AI Hub
- AutoML Video Intelligence
- AutoML
- Dialog Flow Enterprise Edition
- AutoML Tables
- AutoML Translation
- Speech-to-Text
- Text-to-Speech
- Video Intelligence API
- Cloud Natural Language API
- Cloud TPU
- Cloud Translation API
- Cloud Vision API
- Data Labeling
- Recommendations AI

Cloud setup checklist

This checklist helps set up Google Cloud for scalable, production-ready enterprise workloads. The checklist is designed for administrators who are trusted with complete control over the company's Google Cloud resources:

1. **Set up or confirm an identity account**
2. **Add users and groups to your identity account**
3. **Set up administrator access to your organization**
4. **Set up billing**
5. **Set up the resource hierarchy**
6. **Set up access control for your resource hierarchy**
7. **Set up support**
8. **Set up your networking configuration**
9. **Set up logging and monitoring**
10. **Configure organizational security settings**

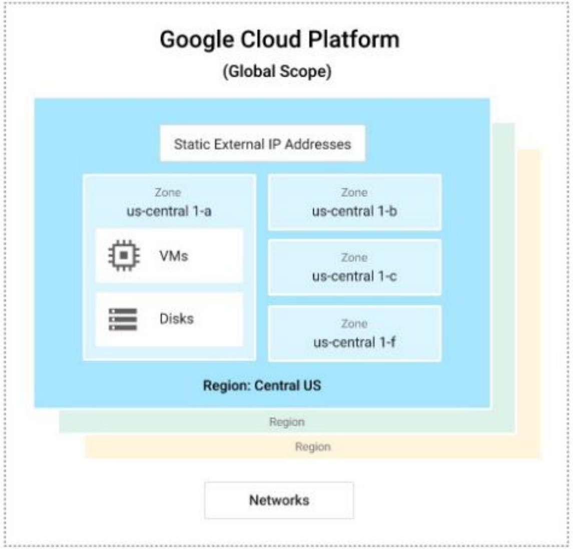
Cloud setup checklist

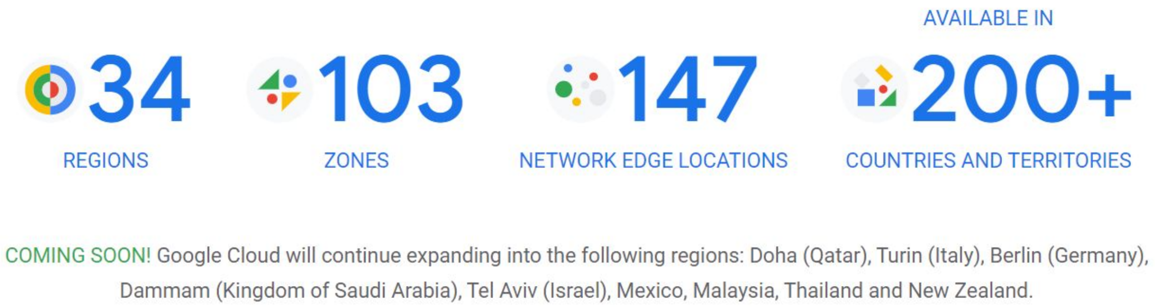


Global Network

Global, regional, and zonal resources:

Google Cloud consists of a set of physical assets, such as computers and hard disk drives, and virtual resources, such as virtual machines (VMs), that are contained in Google's data centers around the globe. The resources are Global, Regional, Zonal or Multiregional depending upon the intended operations.





AVAILABLE IN

34 REGIONS

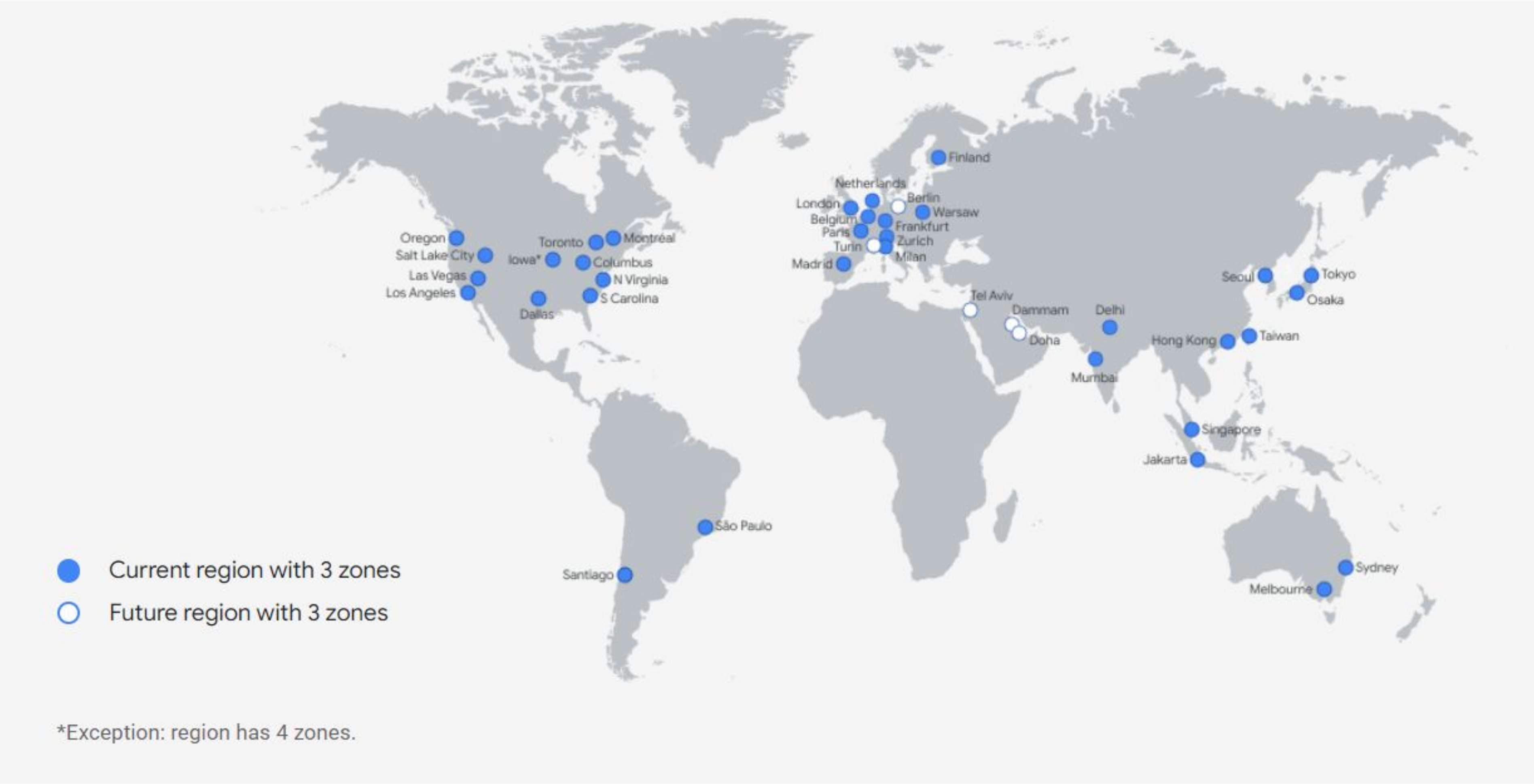
103 ZONES

147 NETWORK EDGE LOCATIONS

200+ COUNTRIES AND TERRITORIES

COMING SOON! Google Cloud will continue expanding into the following regions: Doha (Qatar), Turin (Italy), Berlin (Germany), Dammam (Kingdom of Saudi Arabia), Tel Aviv (Israel), Mexico, Malaysia, Thailand and New Zealand.

Global Network



*Exception: region has 4 zones.

Network Primer

Connect

Hybrid connectivity

Cloud Interconnect, Cloud VPN, Carrier Peering, and Direct Peering provide connectivity solutions for Google Cloud.

Virtual Private Cloud (VPC)

VPC network includes granular IP address range selection, routes, firewall, Cloud VPN (Virtual Private Network), and Cloud Router.

Cloud DNS

Cloud DNS is a scalable, reliable, programmable, and managed authoritative domain naming system (DNS) service running on the same infrastructure as Google

Service Directory

Service Directory helps reduce the complexity of management and operations by providing a single place to publish, discover, and connect all applications services.

Scale

Cloud Load Balancing

Cloud Load Balancing can put resources behind a single anycast IP, scale resources up or down with intelligent autoscaling, and integrate with Cloud CDN.

Cloud CDN

Cloud CDN leverages Google's globally distributed edge caches to accelerate content delivery for websites and applications served out of Compute Engine. Cloud CDN lowers network latency, offloads origins, and reduces serving costs

Secure

Cloud Armor

Google Cloud Armor works with an HTTP(S) load balancer to provide built-in defenses against infrastructure DDoS attacks

Cloud NAT

Cloud NAT enables provisioning application instances without public IP addresses while also allowing access to the internet in a controlled and efficient manner

Network Telemetry

Network Telemetry provides both network and security operations with in-depth, responsive VPC flow logs for Google Cloud networking services.

VPC Service Controls

VPC Service Controls enables enterprises to keep their sensitive data private while leveraging Google Cloud's fully managed storage and data processing capabilities

Optimise

Network Intelligence Center

Network Intelligence Center provides unmatched visibility into your network in the cloud along with proactive network verification.

Network Service Tiers

Improve network experience performance and gain control over network costs with Network Service Tiers.

Authentication Overview

Principals:

A principal is an entity, also known as an identity, that can be granted access to a resource.

- **User accounts** are managed as Google Accounts, and they represent a developer, administrator, or any other person who interacts with Google Cloud.

- **Service accounts** are managed by IAM, and they represent non-human users. They are intended for scenarios where your application needs to access resources or perform actions on its own

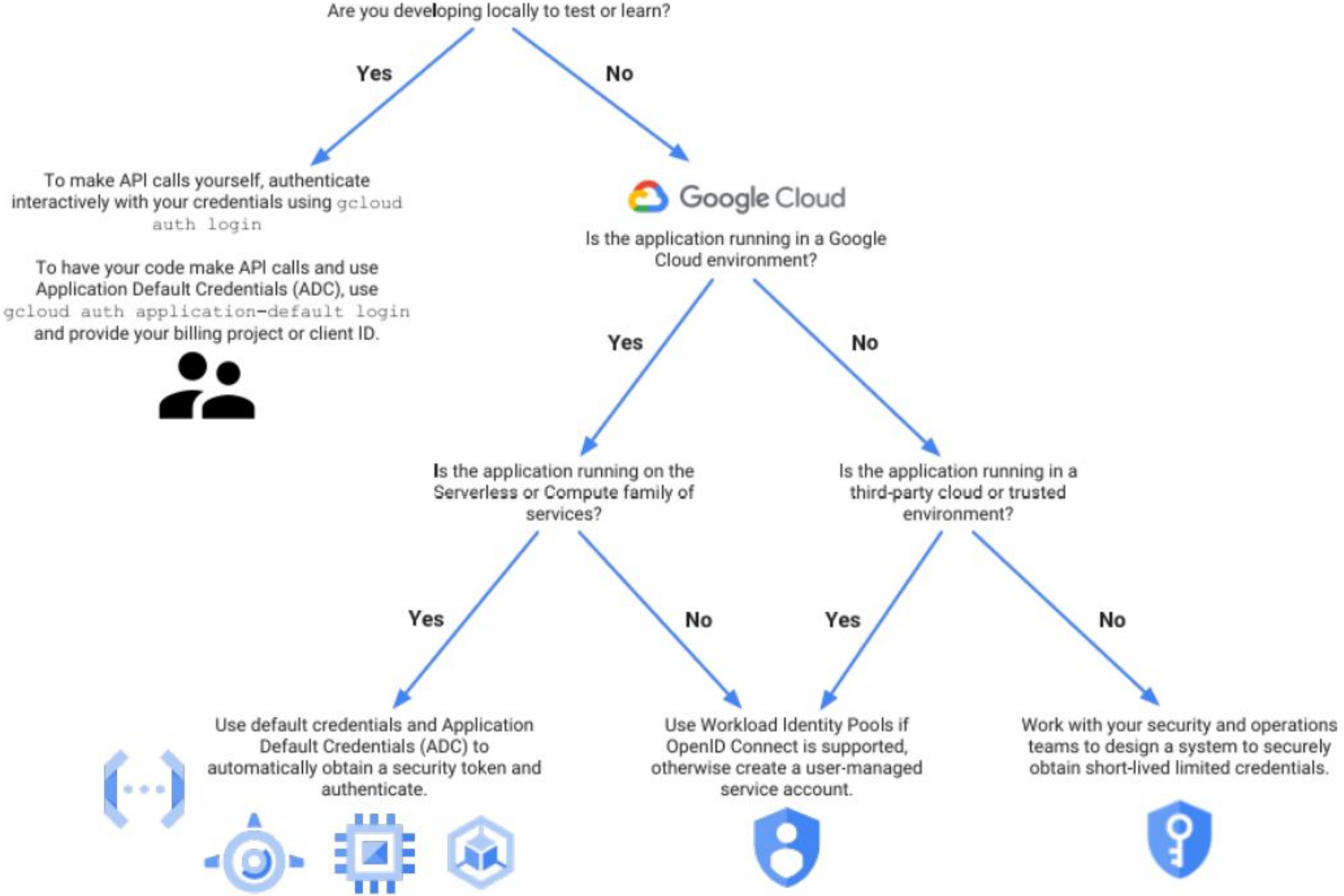
Authentication Overview

Application Default Credentials:

Google auth libraries use a strategy called Application Default Credentials (ADC) to detect and select credentials based on environment or context.

- **Gcloud Credential:** A credential provided by the Gcloud tool that identifies a human user that needs to authenticate to access Google APIs.
- **Service Account Key:** A credential that identifies a non-human user that needs to authenticate to access Google APIs.
- **OAuth Client ID:** A credential that identifies the client application which allows human users to sign-in through 3-legged OAuth flow, which grants the permissions to the application to access Google APIs on behalf of the human user.

Authentication Overview



Are you developing locally to test or learn?

Yes

To make API calls yourself, authenticate interactively with your credentials using `gcloud auth login`

To have your code make API calls and use Application Default Credentials (ADC), use `gcloud auth application-default login` and provide your billing project or client ID.

No

Google Cloud

Is the application running in a Google Cloud environment?

Yes

Is the application running on the Serverless or Compute family of services?

Yes

Use default credentials and Application Default Credentials (ADC) to automatically obtain a security token and authenticate.

No

Use Workload Identity Pools if OpenID Connect is supported, otherwise create a user-managed service account.

No

Is the application running in a third-party cloud or trusted environment?

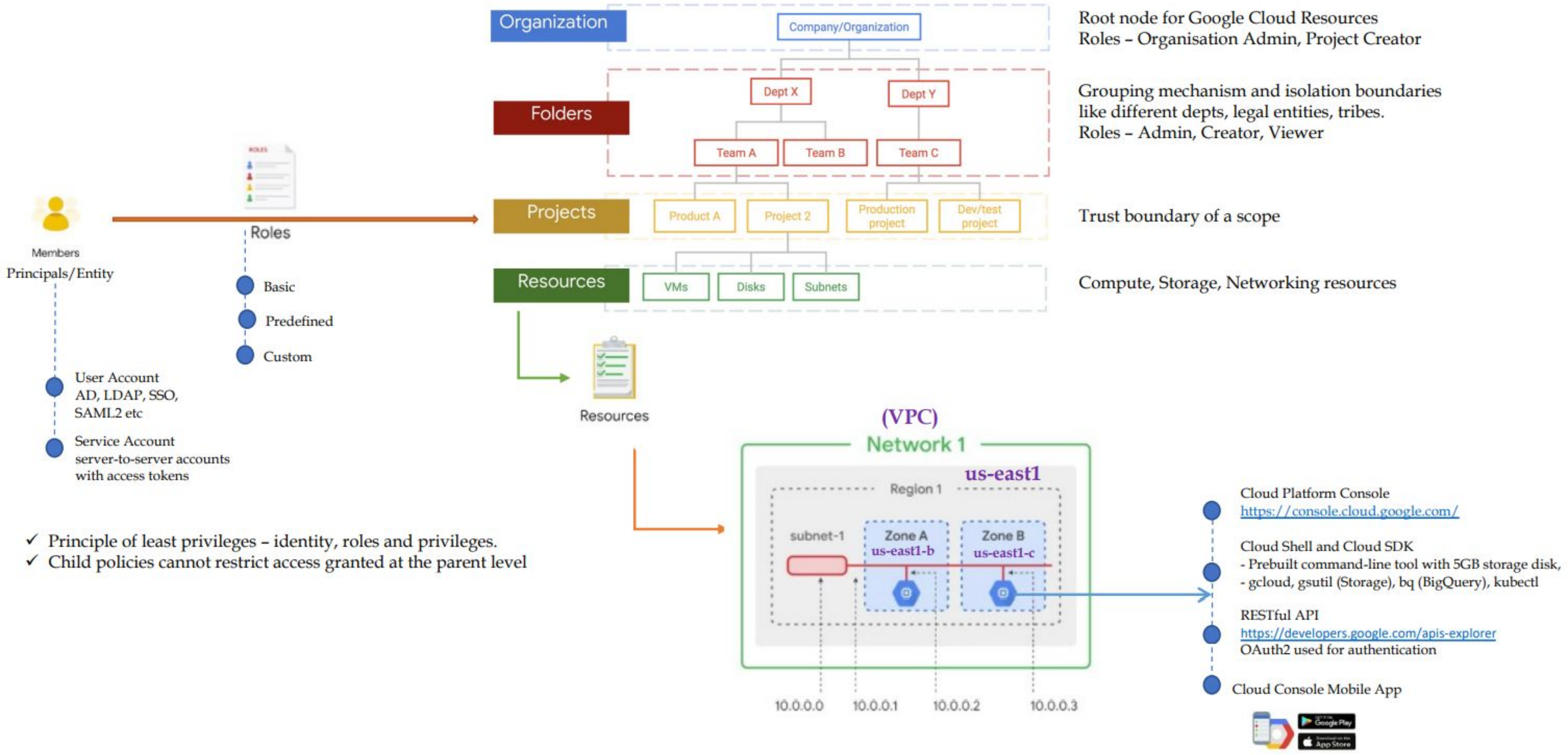
Yes

Use Workload Identity Pools if OpenID Connect is supported, otherwise create a user-managed service account.

No

Work with your security and operations teams to design a system to securely obtain short-lived limited credentials.

The Full Picture



Root node for Google Cloud Resources
Roles – Organisation Admin, Project Creator

Grouping mechanism and isolation boundaries like different depts, legal entities, tribes.
Roles – Admin, Creator, Viewer

Trust boundary of a scope

Compute, Storage, Networking resources

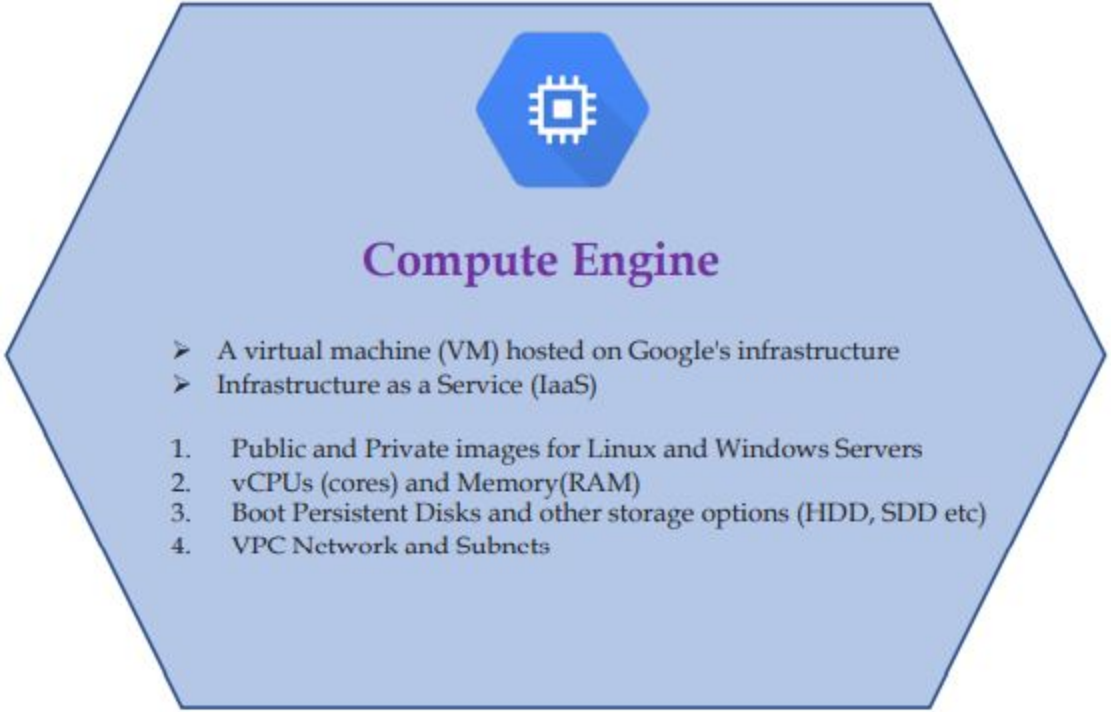
- User Account
  AD, LDAP, SSO,
  SAML2 etc
- Service Account
  server-to-server accounts
  with access tokens

- Cloud Platform Console
  https://console.cloud.google.com/
- Cloud Shell and Cloud SDK
  - Prebuilt command-line tool with 5GB storage disk,
  - gcloud, gsutil (Storage), bq (BigQuery), kubectl
- RESTful API
  https://developers.google.com/apis-explorer
  OAuth2 used for authentication
- Cloud Console Mobile App

✓ Principle of least privileges – identity, roles and privileges.
✓ Child policies cannot restrict access granted at the parent level

GOOGLE COMPUTE ENGINE AND NETWORKING

Compute Engine



Compute Engine

- ➢ A virtual machine (VM) hosted on Google's infrastructure
- ➢ Infrastructure as a Service (IaaS)

1. Public and Private images for Linux and Windows Servers
2. vCPUs (cores) and Memory(RAM)
3. Boot Persistent Disks and other storage options (HDD, SDD etc)
4. VPC Network and Subnets

Compute Engine

Predefined and Custom machine types:

- General purpose (Standard, High-memory, High-CPU)
- Memory-Optimized
- Compute-Optimized

Tau VMs – Optimised for scale-out workloads:

- Web Servers, Containerised microservices, media transcoding Cloud GPUs – For machine learning, scientific computing and 3D visualisation

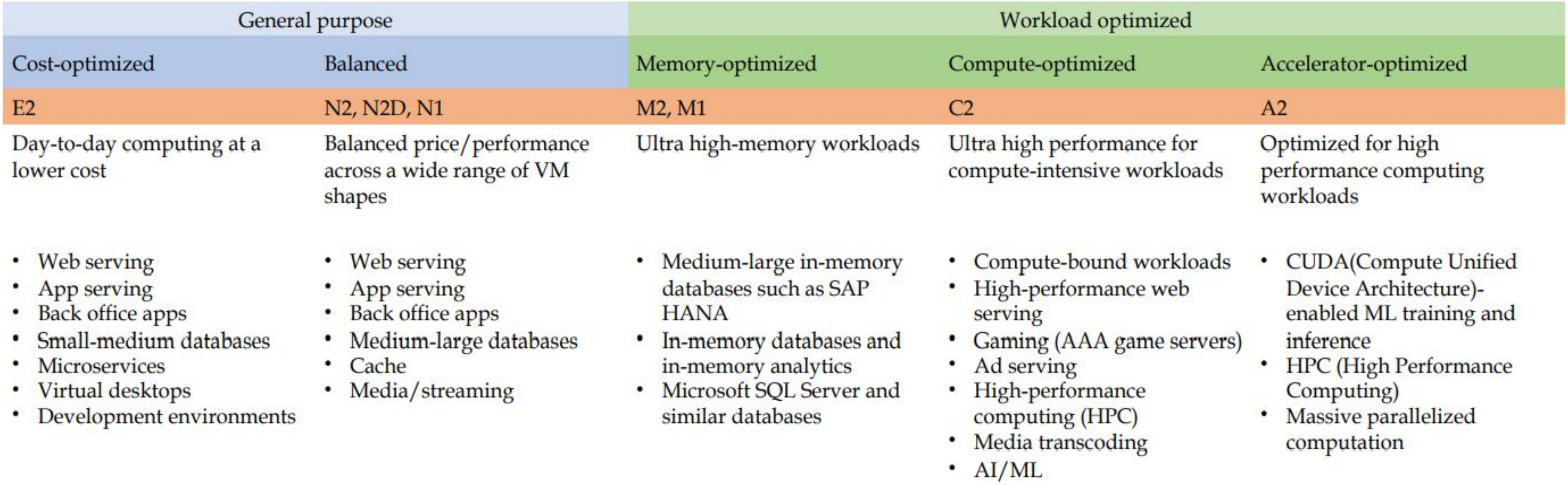
- Machine Learning, Medical Analysis, Video transcoding, Graphic Visualisation

Machine Types

Machine TypesA machine type is a set of virtualized hardware resources available to a virtual machine (VM) instance, including the system memory size, virtual CPU (vCPU) count, and persistent disk limits.

1. General purpose (Standard, High-memory, High-CPU)
2. Memory-Optimized
3. Compute-Optimized
4. Accelerator-optimised

Machine Types



General purpose		Workload optimized		
Cost-optimized	Balanced	Memory-optimized	Compute-optimized	Accelerator-optimized
E2	N2, N2D, N1	M2, M1	C2	A2
Day-to-day computing at a lower cost	Balanced price/performance across a wide range of VM shapes	Ultra high-memory workloads	Ultra high performance for compute-intensive workloads	Optimized for high performance computing workloads
• Web serving • App serving • Back office apps • Small-medium databases • Microservices • Virtual desktops • Development environments	• Web serving • App serving • Back office apps • Medium-large databases • Cache • Media/streaming	• Medium-large in-memory databases such as SAP HANA • In-memory databases and in-memory analytics • Microsoft SQL Server and similar databases	• Compute-bound workloads • High-performance web serving • Gaming (AAA game servers) • Ad serving • High-performance computing (HPC) • Media transcoding • AI/ML	• CUDA(Compute Unified Device Architecture)-enabled ML training and inference • HPC (High Performance Computing) • Massive parallelized computation

Compute Engine

Pricing:

- Per-second billing
- Commitment savings
- Sustained-use savings

Preemptible VM instances:

Much lower price than normal instances but VMs might stop (preempt) based on resource availability:

- Ideal for fault-tolerant applications, batch processing tasks etc
- Always stops after they run for 24 hours
- 30 seconds preemption signal

Compute Engine

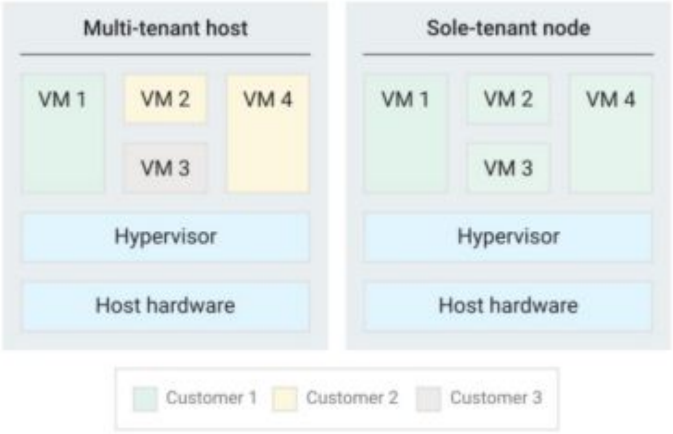
Networking

- Regional HTTPS load balancing
- Network Load Balancing
- Global and multi-regional subnetworks
- Inbound/outbound firewall rules

Storage

- Zonal persistent disk
- Regional persistent disk
- Local SSD
- Cloud Storage buckets
- Filestore

Compute Engine



Sole-tenancy

A sole-tenant node is a physical Compute Engine server that is dedicated to hosting only your project's VMs.

Live migration for VMs

- Live migration to keep VMs running even when a host system event, such as a software or hardware update, occurs.
- Compute Engine live migrates running instances to another host in the same zone instead of requiring VMs to be rebooted.

Instance Templates

Instance templates define the machine type, boot disk image or container image, labels, and other instance properties. It’s used to create Managed Instance Group or to create individual VMs.

Compute Engine

VM access

Linux:

- SSH access – GCP Console, third party client terminal
- CloudShell via Cloud SDK
- Firewall rule tcp:22 allow

Windows:

- RDP access – tcp:3389 allow
- Powershell terminal, third party client

Compute Engine - Instance Groups

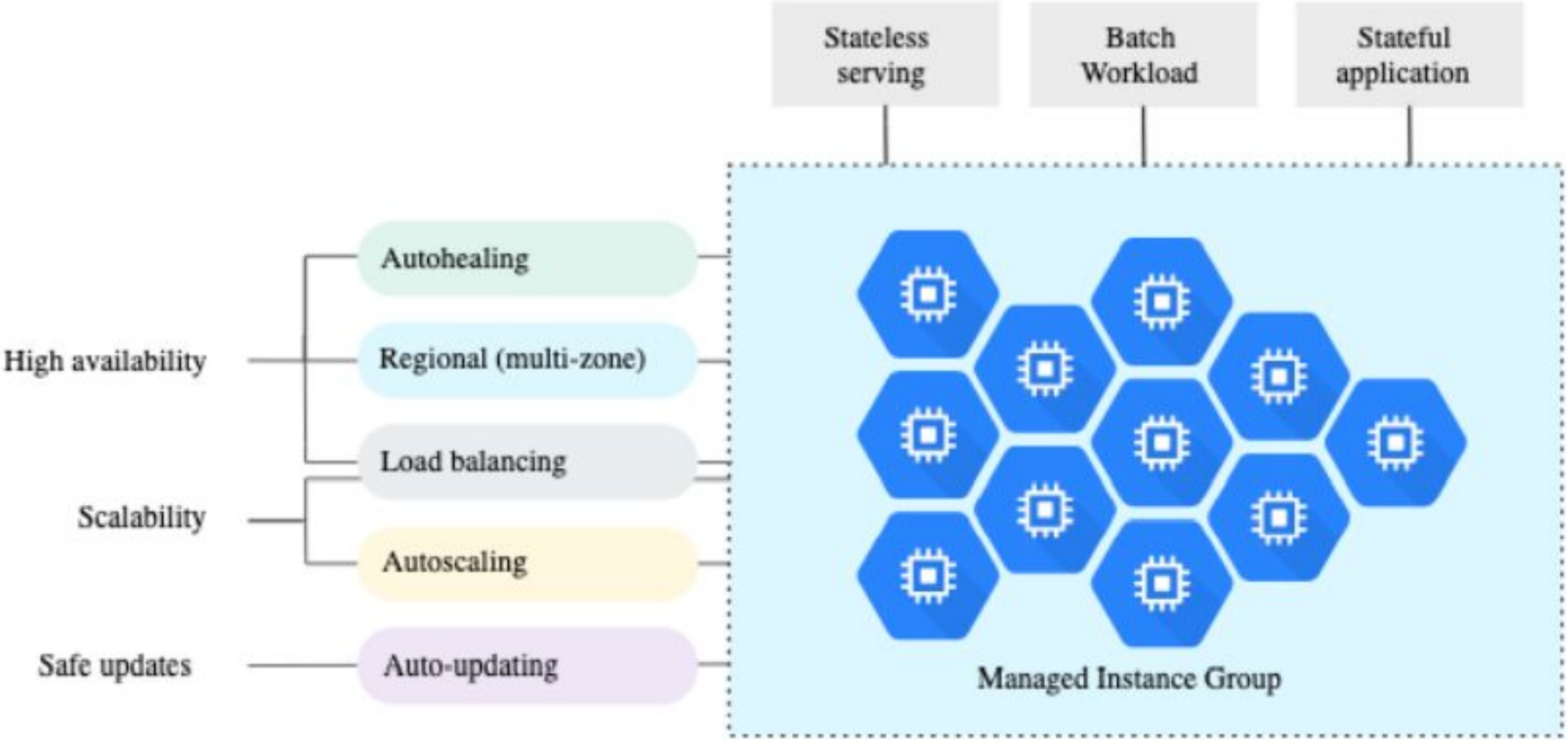
An instance group is a collection of virtual machine (VM) instances that can be managed as a single entity. Compute Engine offers two kinds of VM instance groups, managed and unmanaged:

- Managed instance groups (MIGs).
- Unmanaged instance groups.

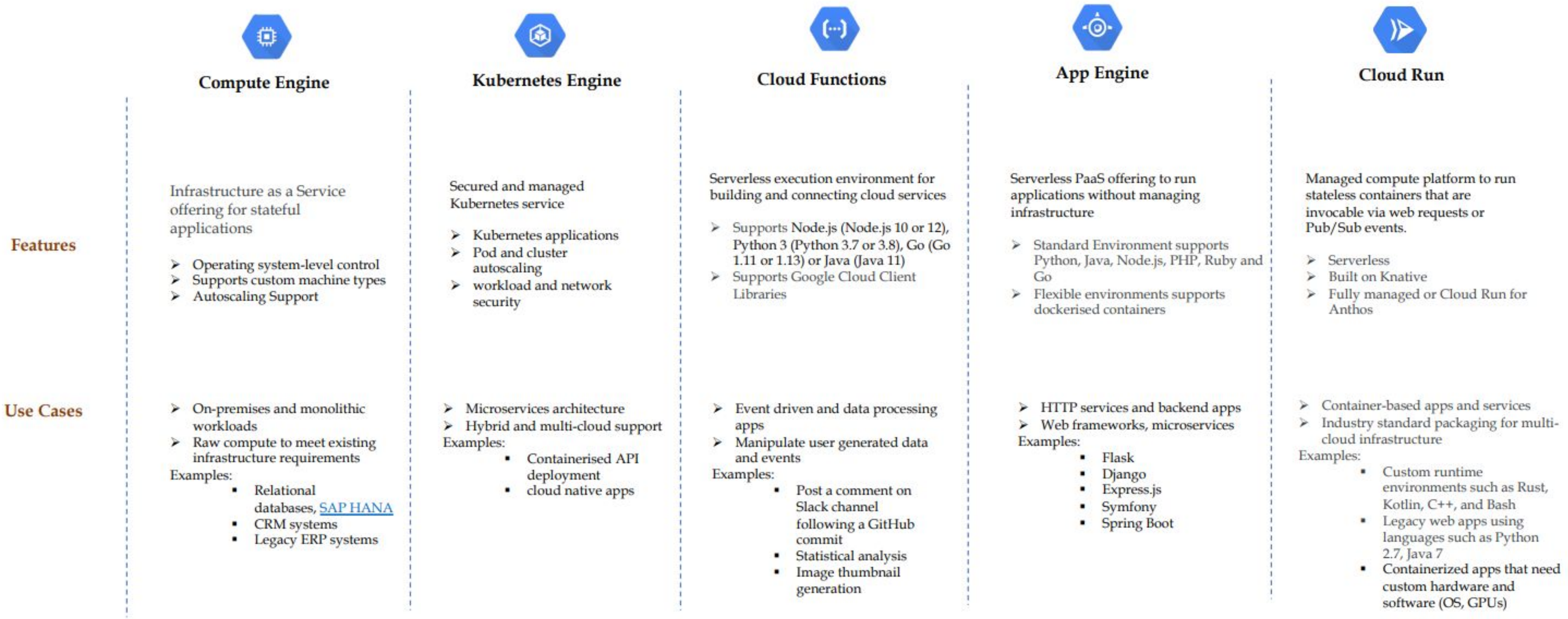
Managed instance groups (MIGs) are suitable for scenarios like these:

- Stateless serving workloads, such as a website frontend.
- Stateless batch, high-performance, or high-throughput compute workloads, such as image processing from a queue.
- Stateful applications, such as databases, legacy applications, and long-running batch computations with checkpointing.

Compute Engine - Instance Groups

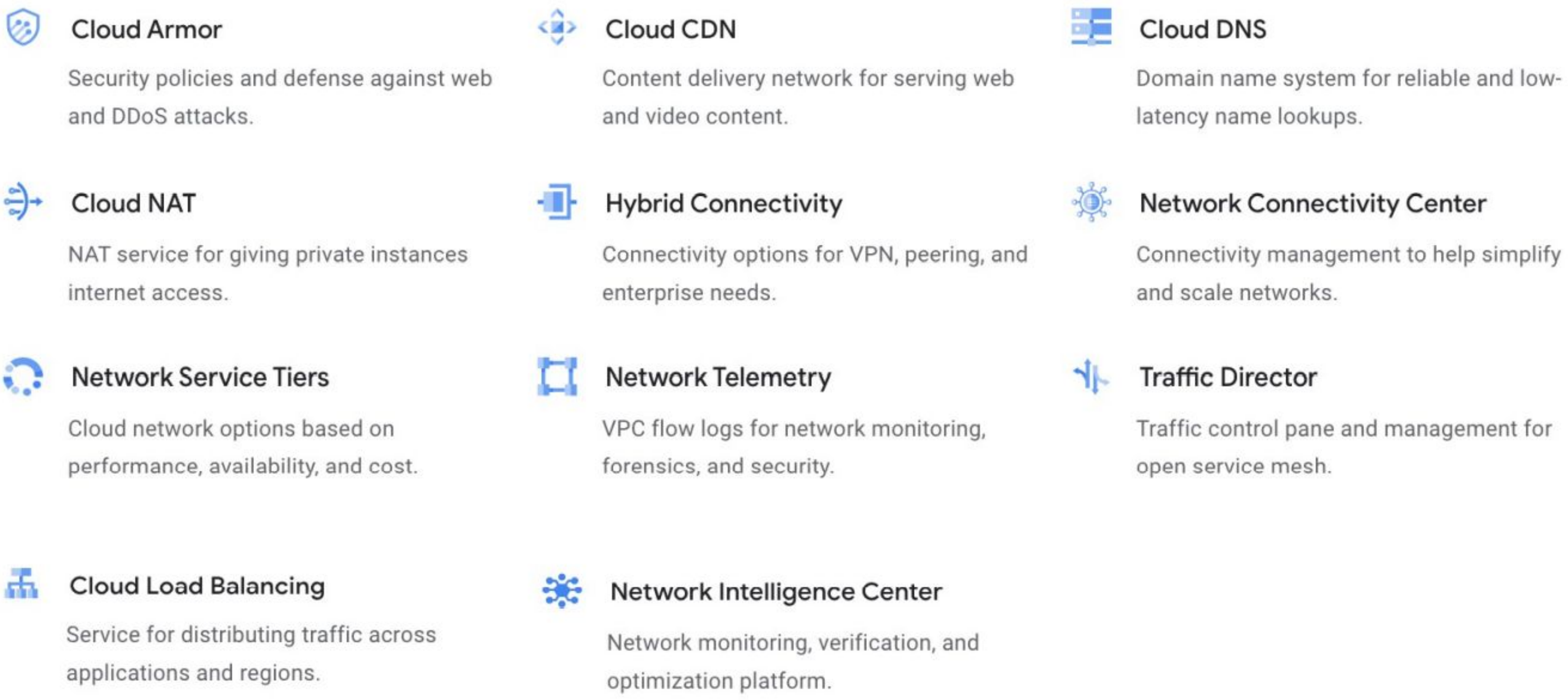


Architecture: Compute Resources



	Compute Engine	Kubernetes Engine	Cloud Functions	App Engine	Cloud Run
Features	Infrastructure as a Service offering for stateful applications ➢ Operating system-level control ➢ Supports custom machine types ➢ Autoscaling Support	Secured and managed Kubernetes service ➢ Kubernetes applications ➢ Pod and cluster autoscaling ➢ workload and network security	Serverless execution environment for building and connecting cloud services ➢ Supports Node.js (Node.js 10 or 12), Python 3 (Python 3.7 or 3.8), Go (Go 1.11 or 1.13) or Java (Java 11) ➢ Supports Google Cloud Client Libraries	Serverless PaaS offering to run applications without managing infrastructure ➢ Standard Environment supports Python, Java, Node.js, PHP, Ruby and Go ➢ Flexible environments supports dockerised containers	Managed compute platform to run stateless containers that are invocable via web requests or Pub/Sub events. ➢ Serverless ➢ Built on Knative ➢ Fully managed or Cloud Run for Anthos
Use Cases	➢ On-premises and monolithic workloads ➢ Raw compute to meet existing infrastructure requirements Examples: ▪ Relational databases, SAP HANA ▪ CRM systems ▪ Legacy ERP systems	➢ Microservices architecture ➢ Hybrid and multi-cloud support Examples: ▪ Containerised API deployment ▪ cloud native apps	➢ Event driven and data processing apps ➢ Manipulate user generated data and events Examples: ▪ Post a comment on Slack channel following a GitHub commit ▪ Statistical analysis ▪ Image thumbnail generation	➢ HTTP services and backend apps ➢ Web frameworks, microservices Examples: ▪ Flask ▪ Django ▪ Express.js ▪ Symfony ▪ Spring Boot	➢ Container-based apps and services ➢ Industry standard packaging for multi-cloud infrastructure Examples: ▪ Custom runtime environments such as Rust, Kotlin, C++, and Bash ▪ Legacy web apps using languages such as Python 2.7, Java 7 ▪ Containerized apps that need custom hardware and software (OS, GPUs)

Networking



Cloud Armor

Security policies and defense against web and DDoS attacks.

Cloud CDN

Content delivery network for serving web and video content.

Cloud DNS

Domain name system for reliable and low-latency name lookups.

Cloud NAT

NAT service for giving private instances internet access.

Hybrid Connectivity

Connectivity options for VPN, peering, and enterprise needs.

Network Connectivity Center

Connectivity management to help simplify and scale networks.

Network Service Tiers

Cloud network options based on performance, availability, and cost.

Network Telemetry

VPC flow logs for network monitoring, forensics, and security.

Traffic Director

Traffic control pane and management for open service mesh.

Cloud Load Balancing

Service for distributing traffic across applications and regions.

Network Intelligence Center

Network monitoring, verification, and optimization platform.

Network – Core Concepts

Virtual Private Cloud

- Region, Zones, Subnet, Routes & Firewall Rules
- VPC Peering
- Shared VPC

Load Balancing

- Global Load Balancer (HTTP(s) Load Balancer, SSL Proxy, TCP Proxy)
- Regional Load Balancer (Network TCP/UDP, Internal TCP/UDP)

Network – IP Addressing & CIDR Block

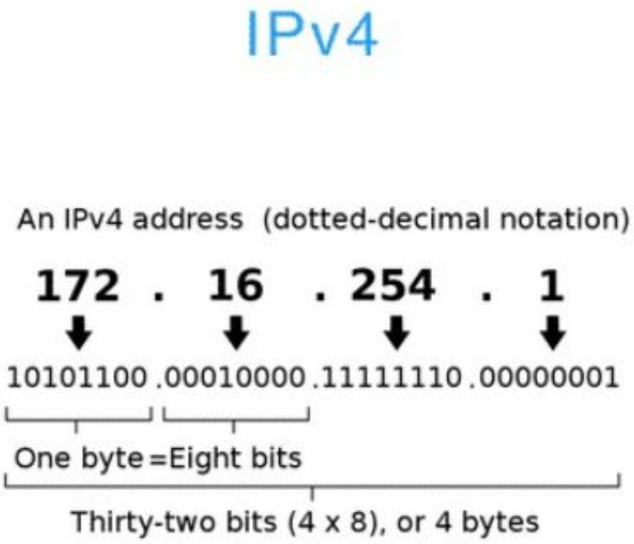
Every location or device on a network must be **addressable**. In the TCP/IP model of network layering, it’s referred as IP address.

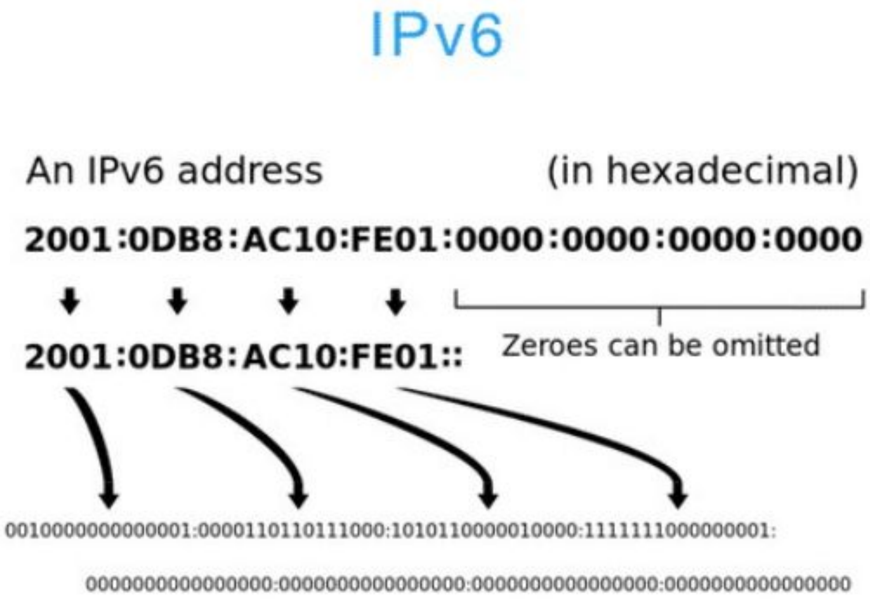
Network Address Translation allows the addresses to be rewritten when packets traverse network borders to allow them to continue to their correct destination.

IPv4 – Four octets, 32 bit address, Each 8-bit segment is divided by a period and typically expressed as a number 0-255. ex- 192.168.20.10

IPv6 – 16 bit blocks, 128 bit address, 8 segments of four hexadecimal digits, double colon (::) is used for leading zeros. ex - 1203:8fe0:fe80:b897:8990:8a7c:99bf:323d

Network – IP Addressing & CIDR Block

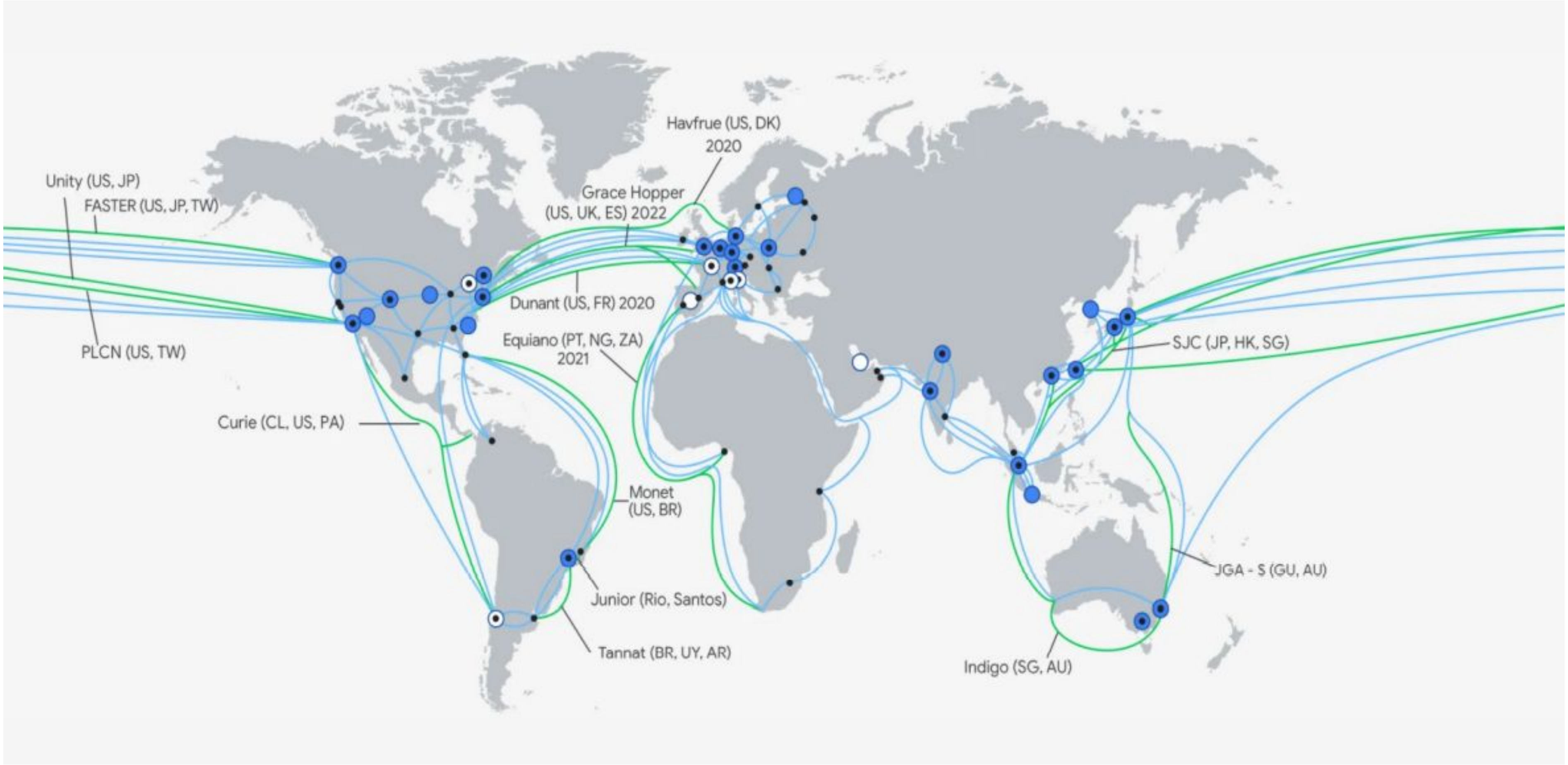




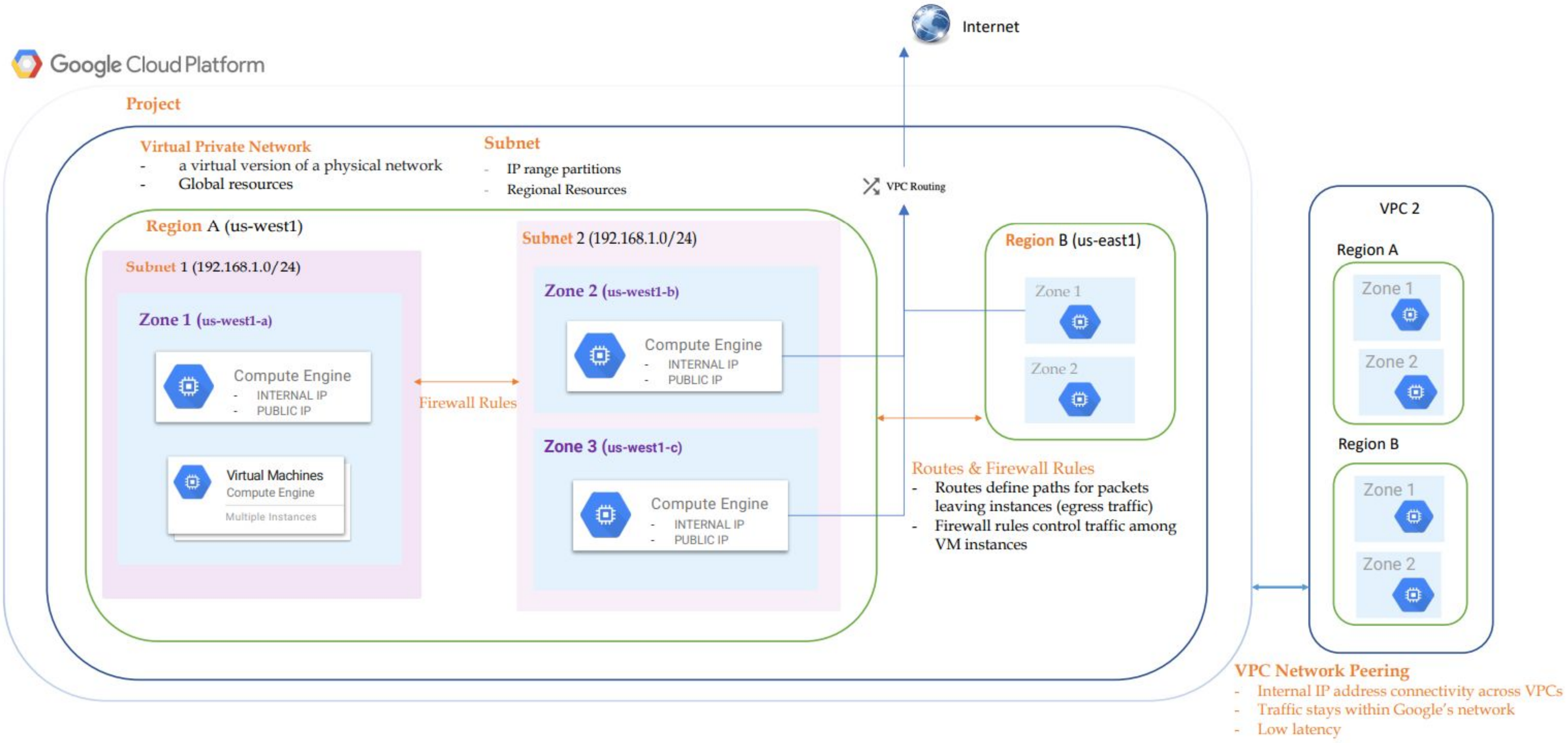
CIDR – Classless inter-domain routing:

CIDR notation of 192.168.0.15/24 means that the first 24 bits of the IP address given are considered significant for the network routing

Network – Global Network Footprint



Network – VPC



Internet

Google Cloud Platform

Project

Virtual Private Network

- a virtual version of a physical network
- Global resources

Subnet

- IP range partitions
- Regional Resources

VPC Routing

Region A (us-west1)

Subnet 1 (192.168.1.0/24)

Zone 1 (us-west1-a)

Compute Engine
- INTERNAL IP
- PUBLIC IP

Virtual Machines
Compute Engine
Multiple Instances

Firewall Rules

Subnet 2 (192.168.1.0/24)

Zone 2 (us-west1-b)

Compute Engine
- INTERNAL IP
- PUBLIC IP

Zone 3 (us-west1-c)

Compute Engine
- INTERNAL IP
- PUBLIC IP

Region B (us-east1)

Zone 1

Zone 2

Routes & Firewall Rules

- Routes define paths for packets leaving instances (egress traffic)
- Firewall rules control traffic among VM instances

VPC 2

Region A

Zone 1

Zone 2

Region B

Zone 1

Zone 2

VPC Network Peering

- Internal IP address connectivity across VPCs
- Traffic stays within Google's network
- Low latency

Network – VPC

- Your 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.
- Many users get started with GCP by defining their own Virtual Private Cloud inside their first GCP project.
- Or they can simply choose the default VPC and get started with that.

Network – VPC

- Google Virtual Private Cloud networks that you define have global scope.
- They can have subnets in any GCP region worldwide.
- Subnets can span the zones that make up a region.
- This architecture makes it easy for you to define your own network layout with global scope.
- You can also have resources in different zones on the same subnet.

Network – VPC

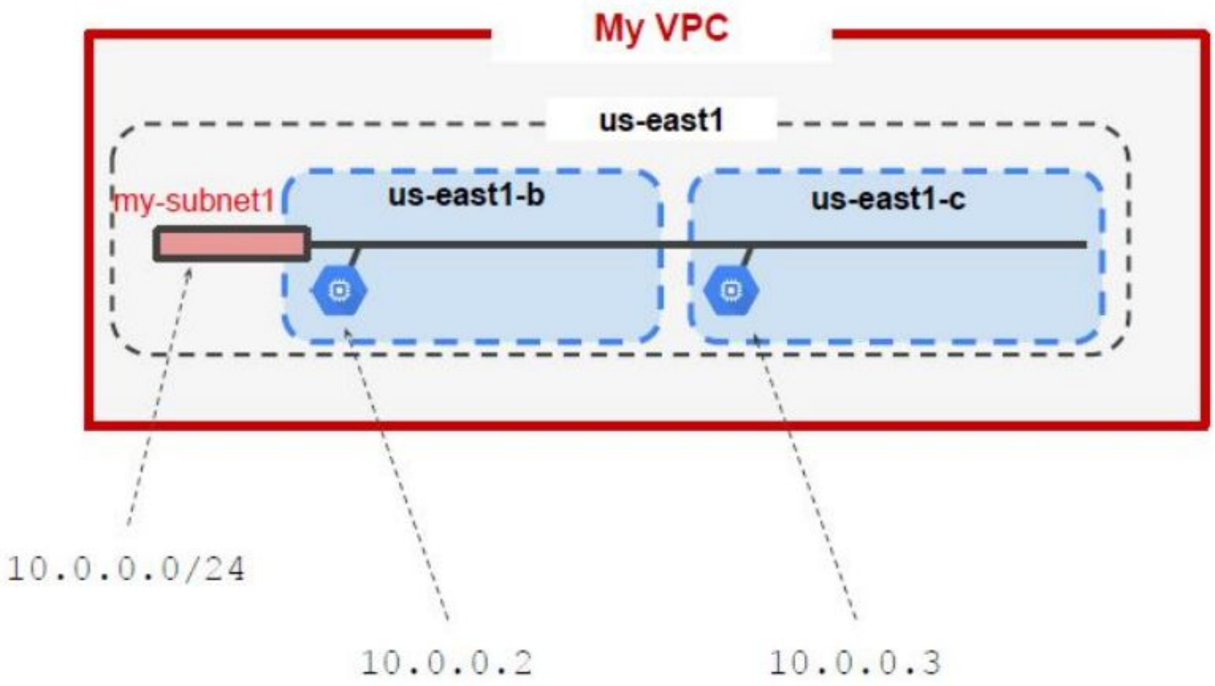
You can dynamically increase the size of a subnet in a custom network by expanding the range of IP addresses allocated to it.

- Doing that doesn’t affect already configured VMs.
- In this example, your VPC has one network. So far, it has one subnet defined, in GCP’s us-east1 region.

  Notice that it has two Compute Engine VMs attached to it.
- They’re **neighbors** on the same subnet even though they are in **different zones**!
- You can use this capability to build solutions that are resilient but still have simple network layouts.

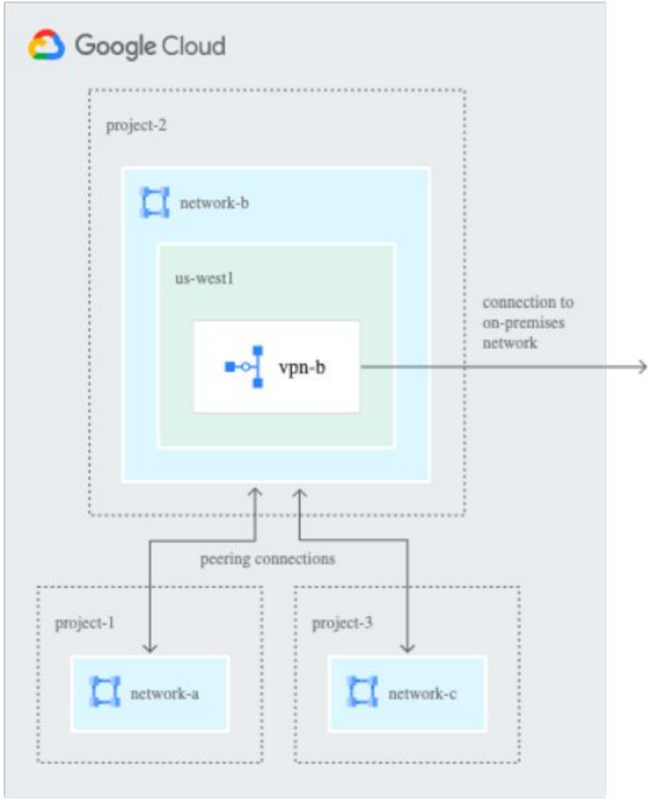
Network – VPC

- Google Cloud VPC networks are global are global; subnets are regional



Network – VPC Peering

VPC VPC Network Peering enables you to connect VPC networks so that workloads in different VPC networks can communicate internally.



Specifications:

- Google Cloud VPC Network Peering allows internal IP address connectivity across two Virtual Private Cloud (VPC) networks regardless of whether they belong to the same project or the same organization.

- Traffic stays within Google's network and doesn't traverse the public internet.

- VPC Network Peering works with Compute Engine, GKE, and App Engine flexible environment.

- Peered VPC networks remain administratively separate. Routes, firewalls, VPNs, and other traffic management tools are administered and applied separately in each of the VPC networks.

- VPC Network Peering is useful in these environments:

1. SaaS (Software-as-a-Service) ecosystems in Google Cloud
2. Organizations with several network administrative domains

- VPC Network Peering gives you several advantages over using external IP addresses or VPNs to connect networks, including:

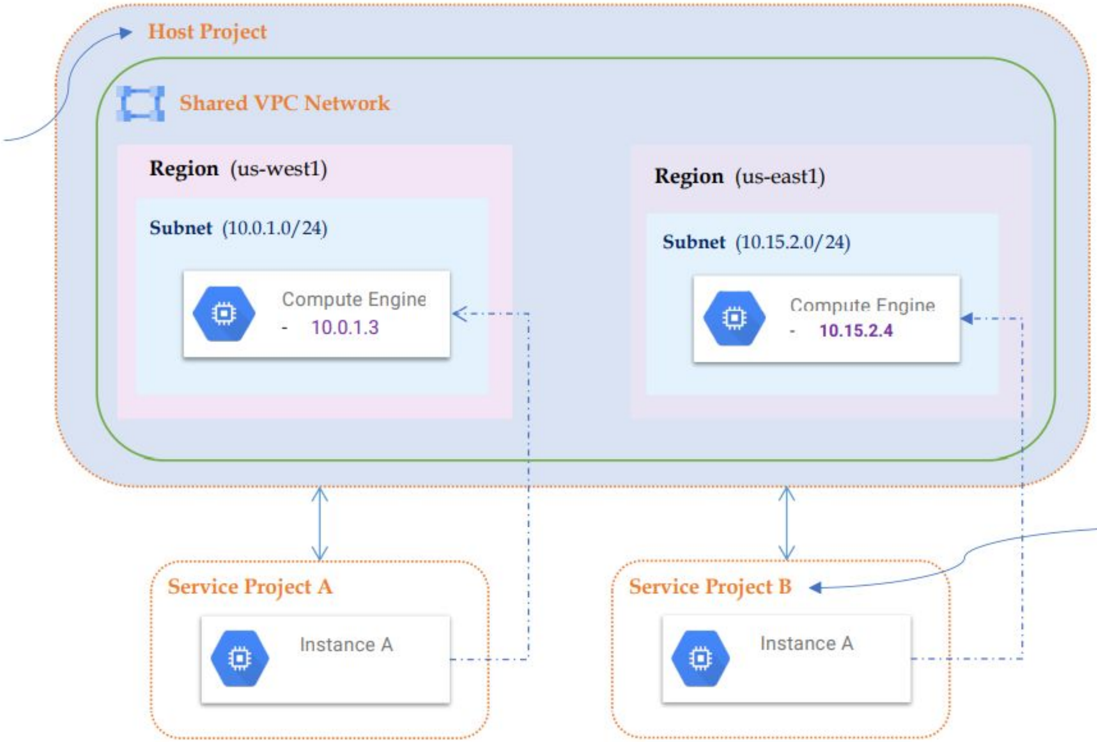
1. Network Latency
2. Network Security
3. Network Cost (egress)

Network – Shared VPC

Shared VPC allows an organization to connect resources from multiple projects to a common Virtual Private Cloud (VPC) network, so that they can communicate with each other securely and efficiently using internal IPs from that network.

*A **host project** contains one or more Shared VPC networks.*

*A **Shared VPC network** is a VPC network defined in a host project and made available as a centrally shared network for eligible resources in service projects.*



Project Requirements:

1. Delegate Administrative responsibilities for Networking
2. Hybrid Cloud & Transit Network

*A **service project** is any project that has been attached to a host project by a Shared VPC Admin.*

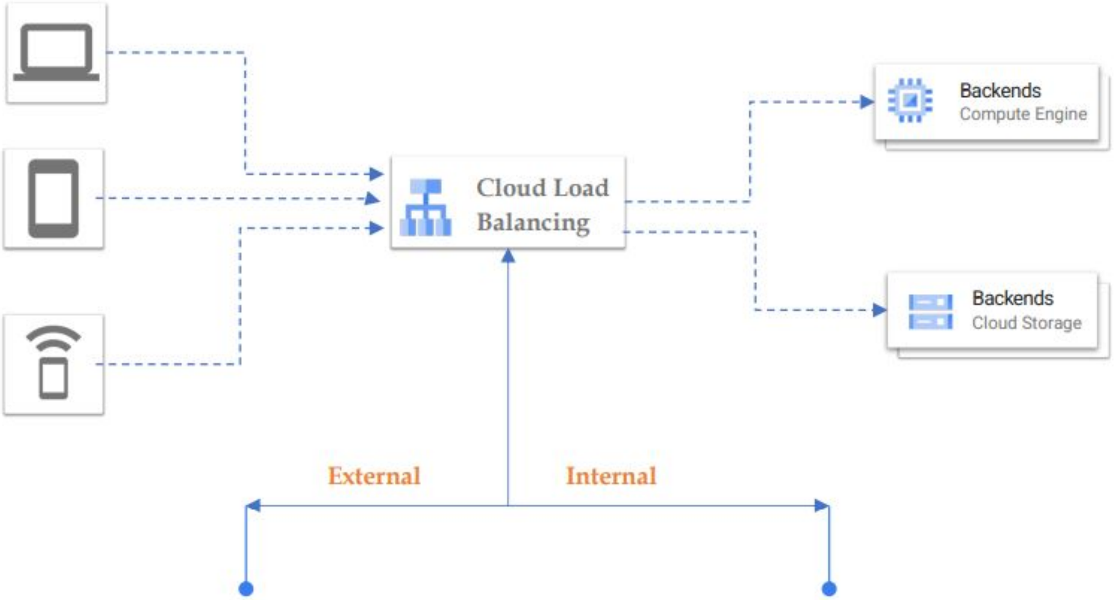
Network – Load Balancer

Cloud Load Balancing is a fully distributed, software-defined managed service, which distributes user traffic across multiple instances of applications.

Google Cloud offers the following load balancing features:

- Single IP address to serve as the frontend
- Automatic intelligent autoscaling of the backends
- Layer 4-based load balancing to direct traffic based on data from network and transport layer protocols, such as IP address and TCP or UDP port
- Layer 7-based load balancing to add content-based routing decisions based on attributes, such as the HTTP header and the uniform resource identifier
- Integration with Cloud CDN for cached content delivery

Network – Load Balancer

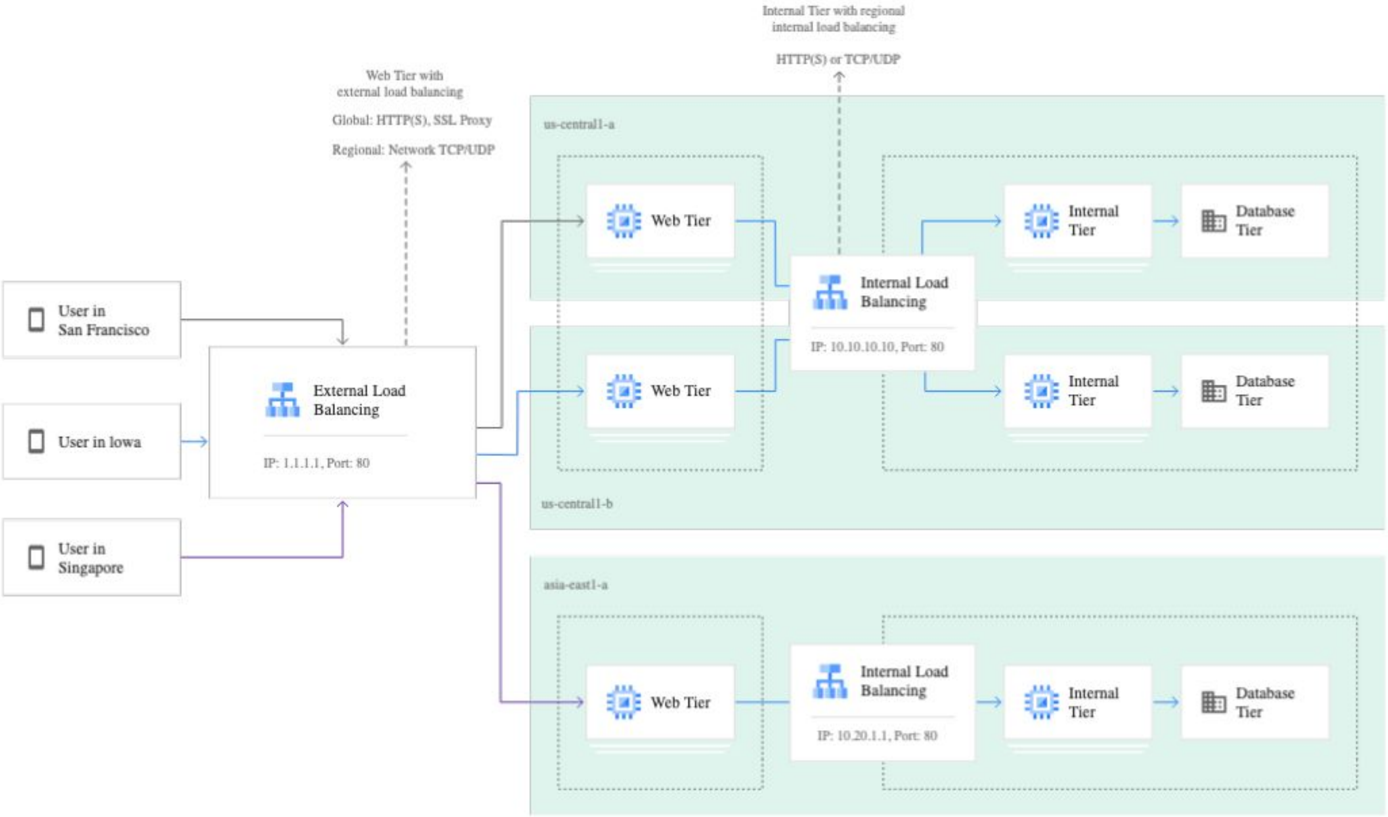


External HTTP(S) Load Balancing
SSL Proxy Load Balancing
TCP Proxy Load Balancing
TCP/UDP Network Load Balancing

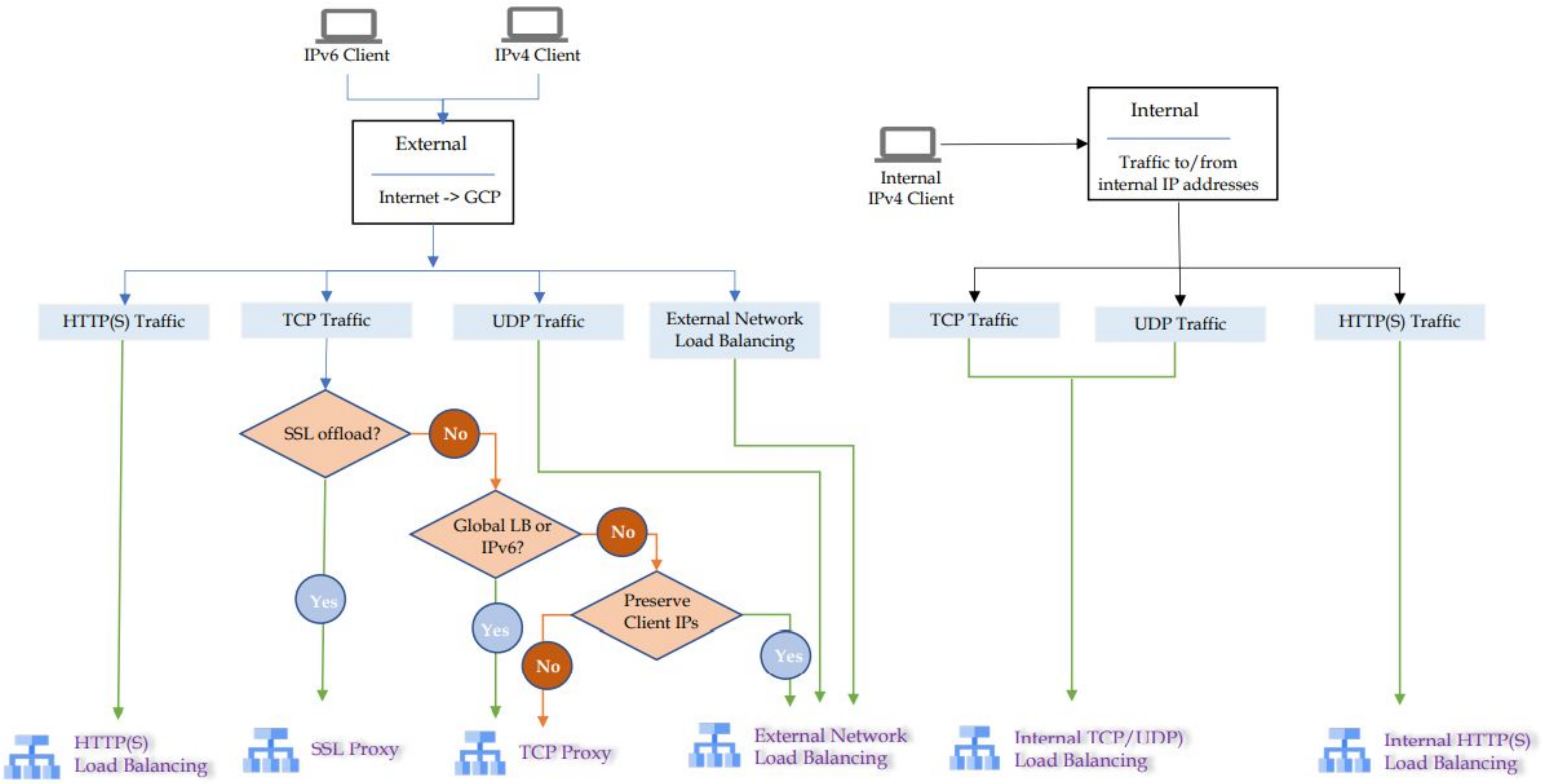
Internal HTTP(S) Load Balancing
Internal TCP/UDP Load Balancing

Network – External vs Internal Load Balancing

The following diagram illustrates a common use case: how to use external and internal load balancing together.



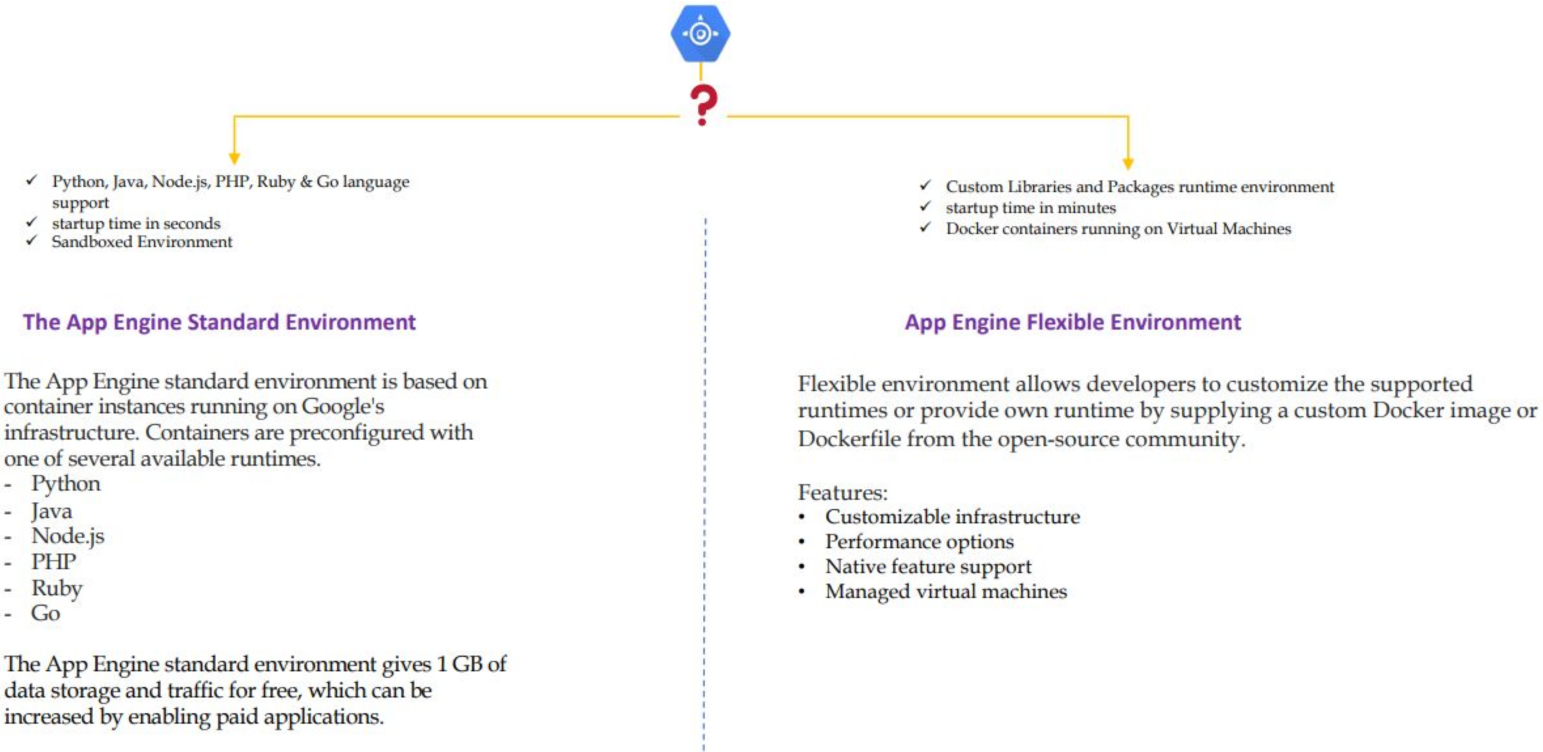
Network – Choosing a Load Balancer



GOOGLE APP ENGINE ET GOOGLE CLOUD DATASTORE

APP ENGINES

App Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, then let App Engine take care of provisioning servers and scaling app instances based on demand.



- ✓ Python, Java, Node.js, PHP, Ruby & Go language support
- ✓ startup time in seconds
- ✓ Sandboxed Environment

The 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.

- Python
- Java
- Node.js
- PHP
- Ruby
- Go

The App Engine standard environment gives 1 GB of data storage and traffic for free, which can be increased by enabling paid applications.

- ✓ Custom Libraries and Packages runtime environment
- ✓ startup time in minutes
- ✓ Docker containers running on Virtual Machines

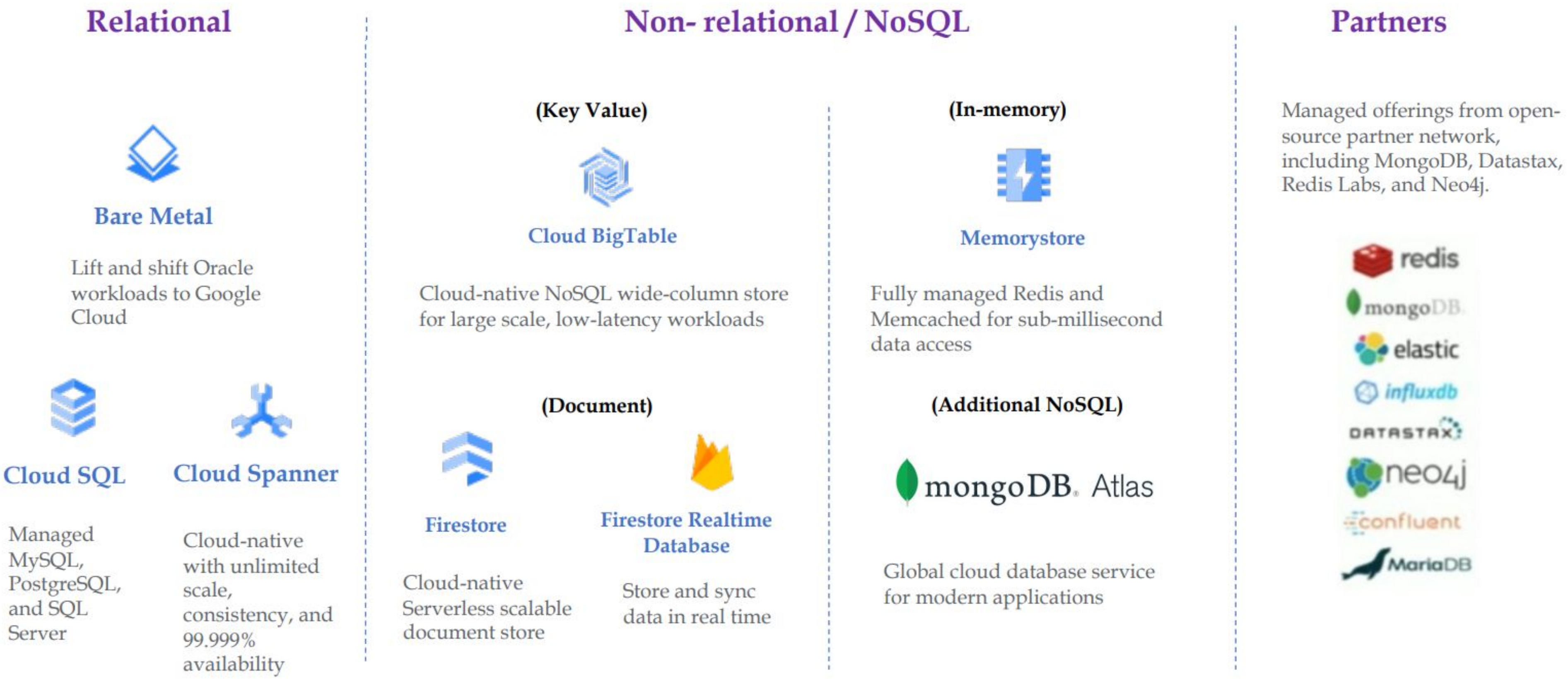
App Engine Flexible Environment

Flexible environment allows developers to customize the supported runtimes or provide own runtime by supplying a custom Docker image or Dockerfile from the open-source community.

Features:

- Customizable infrastructure
- Performance options
- Native feature support
- Managed virtual machines

Architecture: Databases



Relational

Bare Metal

Lift and shift Oracle workloads to Google Cloud

Cloud SQL

Managed MySQL, PostgreSQL, and SQL Server

Cloud Spanner

Cloud-native with unlimited scale, consistency, and 99.999% availability

Non- relational / NoSQL

(Key Value)

Cloud BigTable

Cloud-native NoSQL wide-column store for large scale, low-latency workloads

(Document)

Firestore

Cloud-native Serverless scalable document store

Firestore Realtime Database

Store and sync data in real time

(In-memory)

Memorystore

Fully managed Redis and Memcached for sub-millisecond data access

(Additional NoSQL)

mongoDB. Atlas

Global cloud database service for modern applications

Partners

Managed offerings from open-source partner network, including MongoDB, Datastax, Redis Labs, and Neo4j.

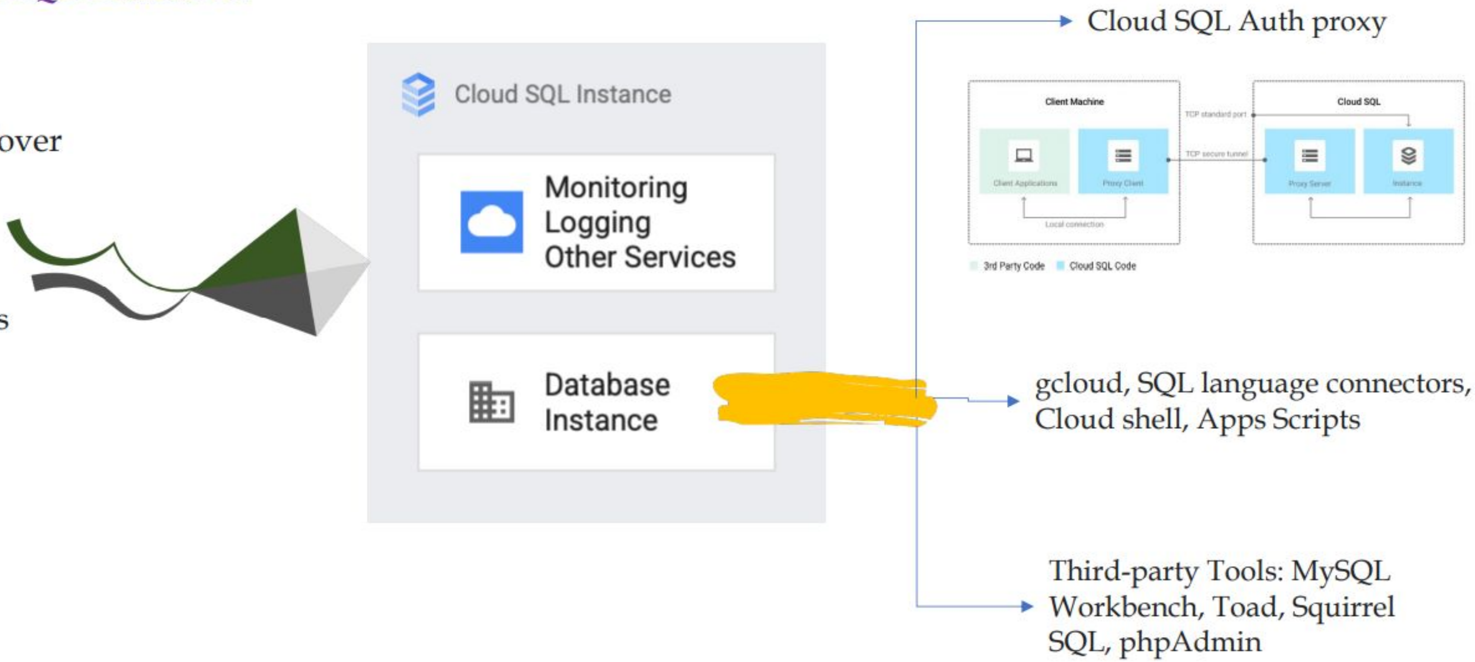
redis
mongoDB
elastic
influxdb
DATASTAX
neo4j
confluent
MariaDB

Databases: Cloud SQL

Fully managed relational database service for MySQL, PostgreSQL, and SQL Server.

Administrative overhead of SQL Databases:

- Backups
- High availability and failover
- Network connectivity
- Export and import
- Maintenance and updates
- Monitoring
- Logging



Cloud SQL Auth proxy

gcloud, SQL language connectors, Cloud shell, Apps Scripts

Third-party Tools: MySQL Workbench, Toad, Squirrel SQL, phpAdmin

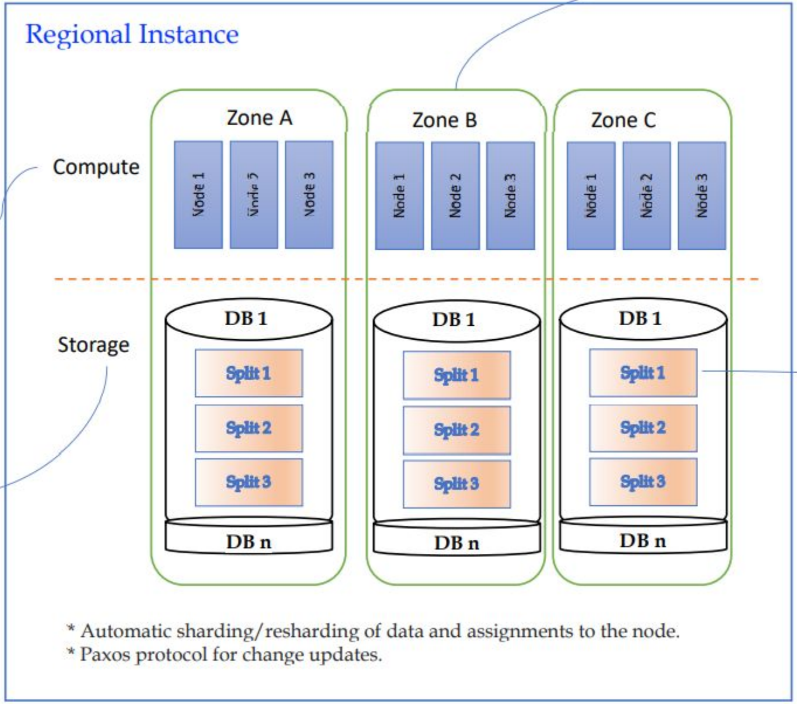
Databases: Cloud Spanner

Fully managed relational database with unlimited scale (horizontal scale across regions), strong consistency, and up to 99.999% availability.Relational Semantics (Schemas, ACID transactions, SQL) + Horizontal Scale (99.999% SLA, fully managed).

Architecture overview:

Node servers serve the read and write/commit transaction requests, but they don't store the data.

The data is stored in Google's underlying Colossus distributed replicated file system.



* Automatic sharding/resharding of data and assignments to the node.
* Paxos protocol for change updates.

Everything is replicated across the three zones – the same goes for split management. Using Paxos consensus protocols, one of the zones is determined to be a leader. The leader is responsible for managing write transactions for that split, while the other replicas can be used for reads.

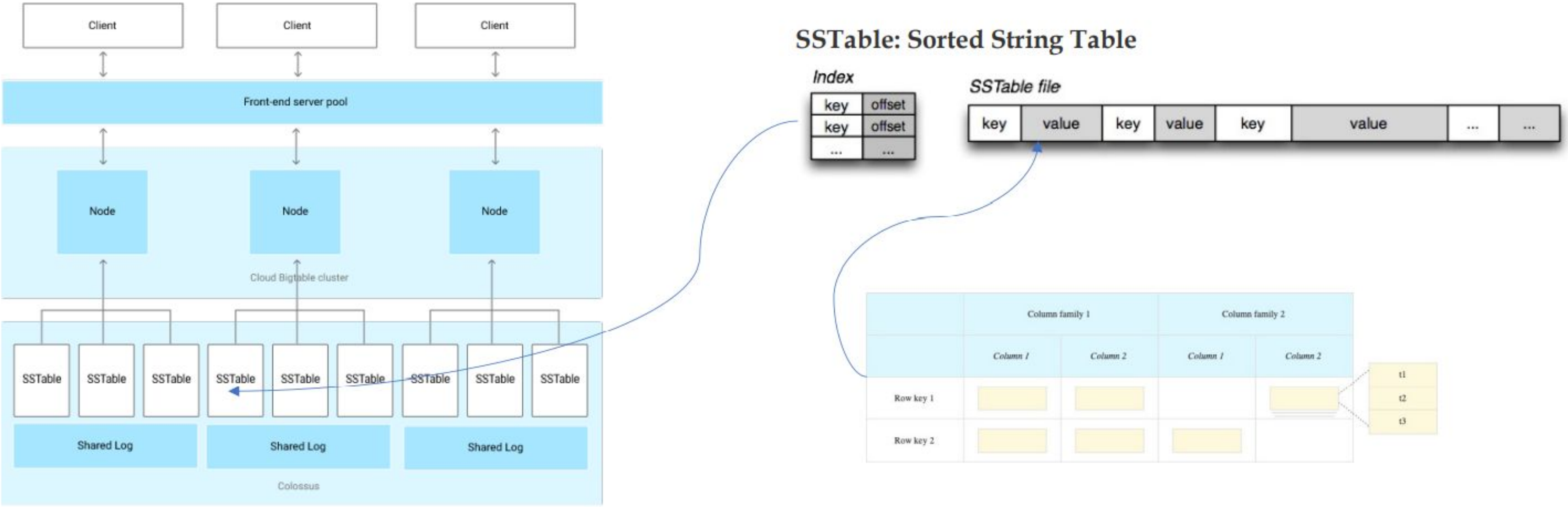
Each table in the database is stored sorted by primary key. Tables are divided by ranges of the primary key and these divisions are known as splits.

Databases: Cloud Bigtable

A fully managed, scalable NoSQL database service for large analytical and operational workloads with up to 99.999% availability.

- Consistent sub-10ms latency—handle millions of requests per second
- Ideal for use cases such as personalization, ad tech, fintech, digital media, and IoT

Bigtable architecture:



SSTable: Sorted String Table

Databases: Cloud Bigtable

Design Principle:

- ✓ RISC (Reduced Instruction Set Computing) - Simplify the operations.
- ✓ Cloud Bigtable is a learning system. - "hot spots"
- ✓ Bigtable is ideal for applications that need very high throughput and scalability for key/value (< 10 MB) data.
- ✓ Bigtable is not a relational database. It does not support SQL queries, joins, or multi-row transactions.

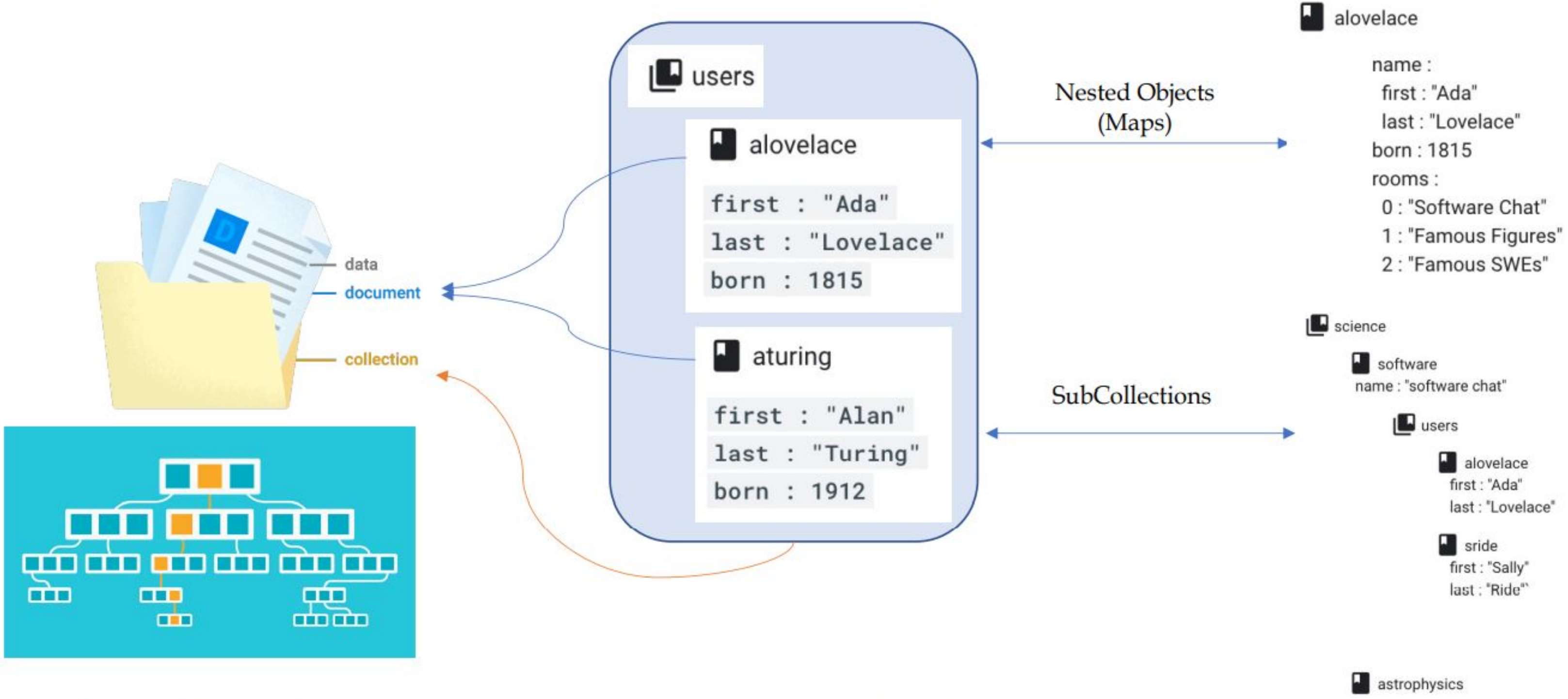
OLTP: OLAP: Document: In-Memory: Real-time Sync:

Bigtable use cases:

- **Time-series data**, such as CPU and memory usage over time for multiple servers.
- **Marketing data**, such as purchase histories and customer preferences.
- **Financial data**, such as transaction histories, stock prices, and currency exchange rates.
- **Internet of Things data**, such as usage reports from energy meters and home appliances.
- **Graph data**, such as information about how users are connected to one another.

Databases: Firestore

Firestore is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows. Instead, data is stored in documents, which are organized into collections.



users

alovelace

first : "Ada"
last : "Lovelace"
born : 1815

aturing

first : "Alan"
last : "Turing"
born : 1912

Nested Objects (Maps)

alovelace

name :
first : "Ada"
last : "Lovelace"
born : 1815
rooms :
0 : "Software Chat"
1 : "Famous Figures"
2 : "Famous SWEs"

SubCollections

science
software
name : "software chat"
users
alovelace
first : "Ada"
last : "Lovelace"
sride
first : "Sally"
last : "Ride"`
astrophysics
...

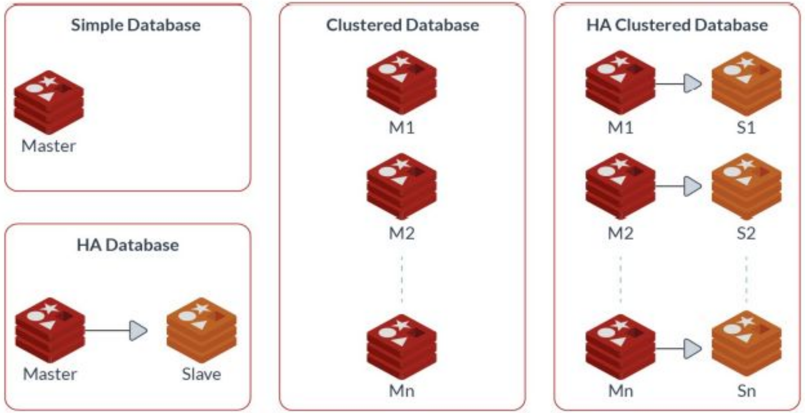
Use Cases: ***User Profiles, Real-time inventories, User session management, Distributed counters***

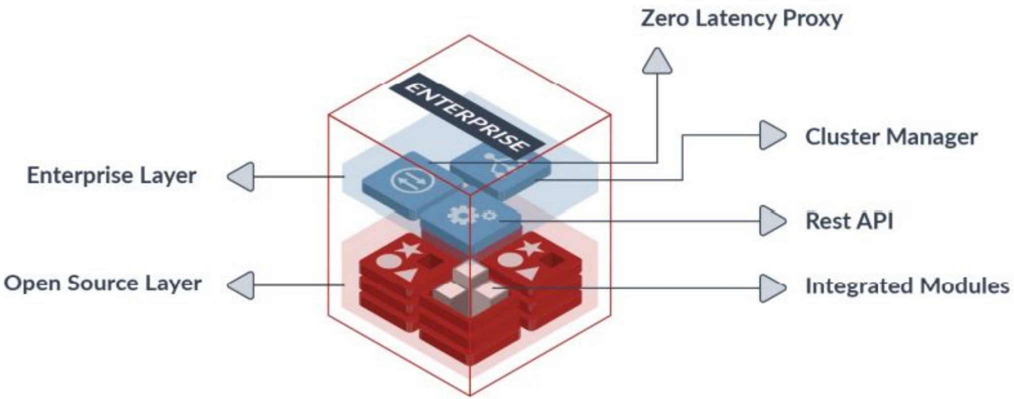
Databases: Memorystore

Reduce latency with scalable, secure, and highly available in-memory service for Redis and Memcached.

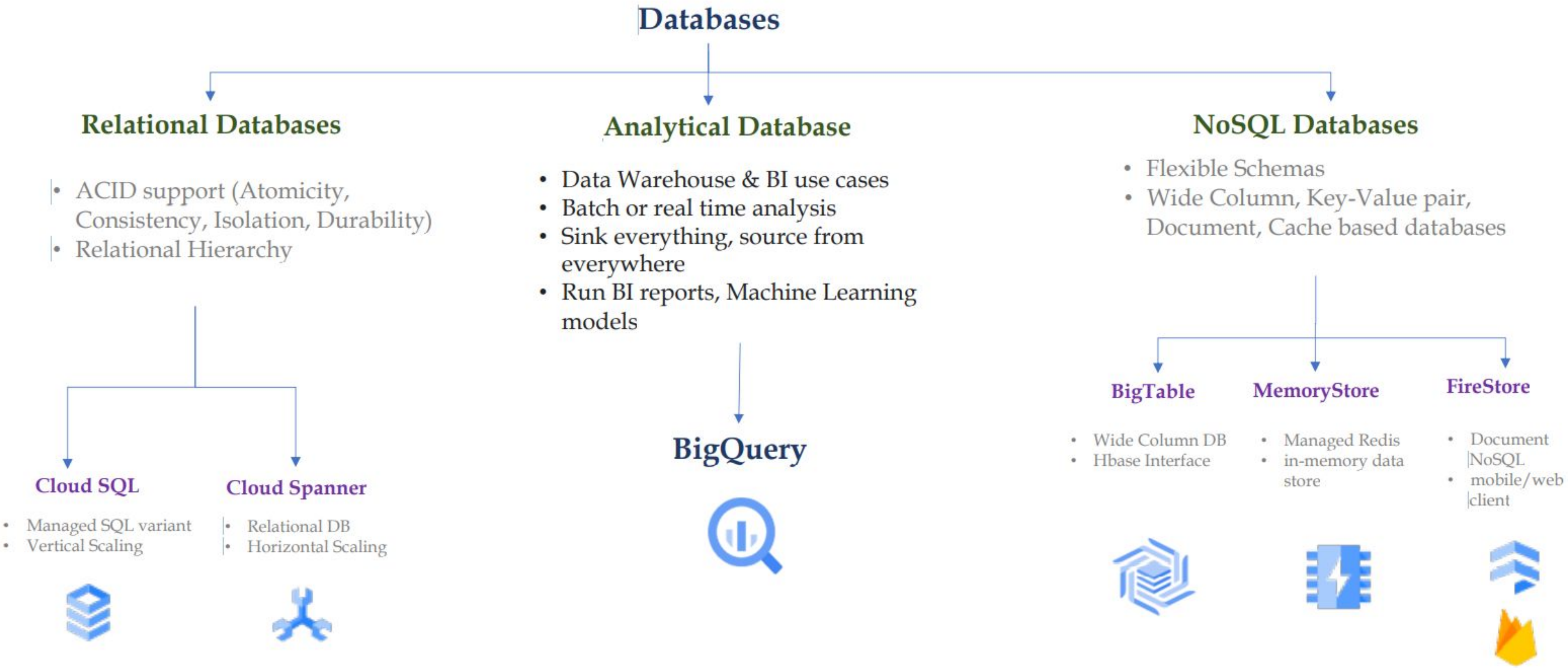
- Build application caches that provide sub-millisecond data access
- 100% compatible with open source Redis and Memcached
- Migrate your caching layer to cloud with zero code change

Redis Enterprise Cluster Architecture (Shared-nothing, linearly scalable, multi-tenant, symmetric architecture





Databases: Analytical Database



Databases

Relational Databases

- ACID support (Atomicity, Consistency, Isolation, Durability)
- Relational Hierarchy

Cloud SQL

- Managed SQL variant
- Vertical Scaling

Cloud Spanner

- Relational DB
- Horizontal Scaling

Analytical Database

- Data Warehouse & BI use cases
- Batch or real time analysis
- Sink everything, source from everywhere
- Run BI reports, Machine Learning models

BigQuery

NoSQL Databases

- Flexible Schemas
- Wide Column, Key-Value pair, Document, Cache based databases

BigTable

- Wide Column DB
- Hbase Interface

MemoryStore

- Managed Redis
- in-memory data store

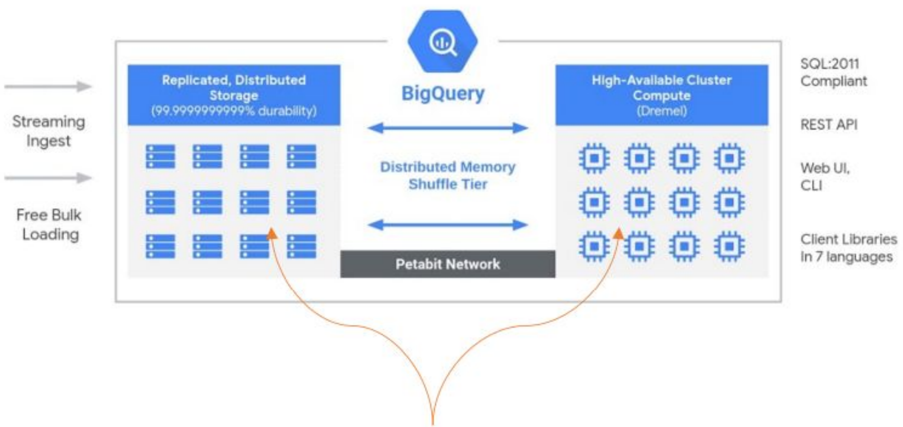
FireStore

- Document NoSQL
- mobile/web client

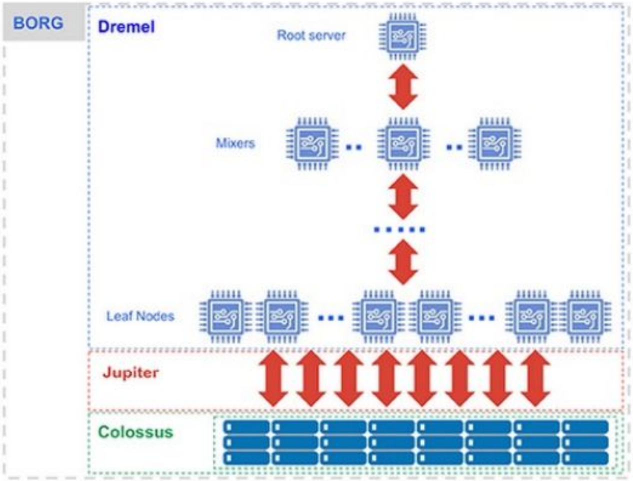
BigQuery: Architecture

BigQuery is Google Cloud's fully managed, petabyte-scale, and cost-effective analytics data warehouse that helps you manage and analyze data with built-in features like machine learning, geospatial analysis, and business intelligence.

BigQuery Architecture:


Streaming Ingest
Free Bulk Loading
Replicated, Distributed Storage (99.9999999999% durability)
BigQuery
Distributed Memory Shuffle Tier
Petabit Network
High-Available Cluster Compute (Dremel)
SQL:2011 Compliant
REST API
Web UI, CLI
Client Libraries In 7 languages

BigQuery's serverless architecture decouples storage and compute and allows them to scale independently on demand.

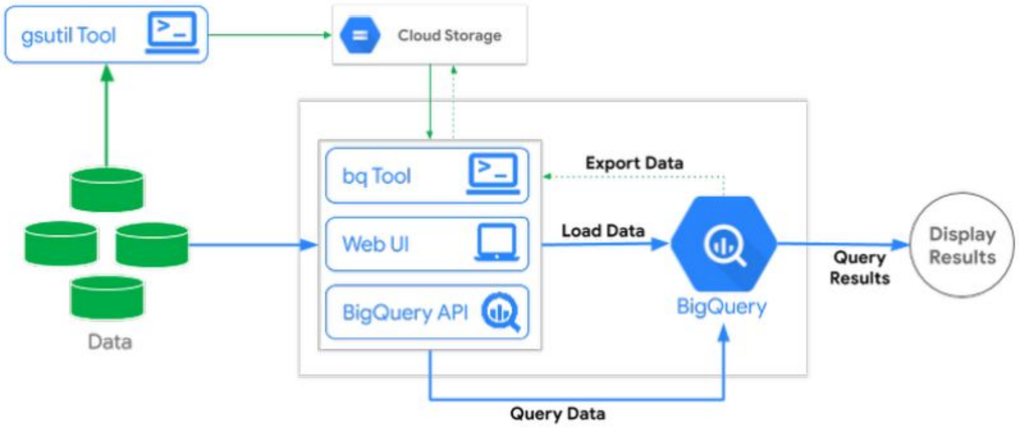


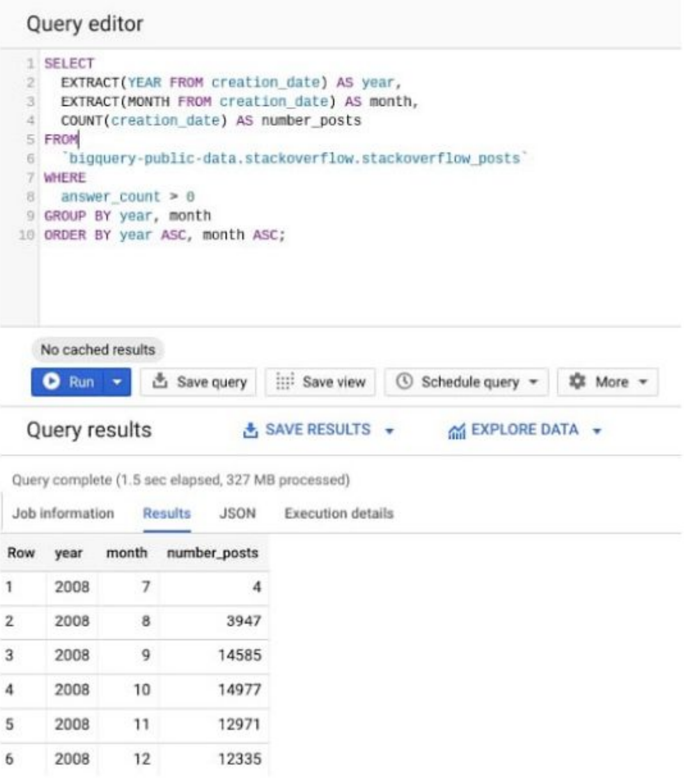
Under the hood, BigQuery employs a vast set of multi-tenant services driven by low-level Google infrastructure technologies like Dremel, Colossus, Jupiter and Borg.

BigQuery: using BigQuery

BigQuery can be accessed in multiple ways:

- Using the GCP console
- Using the command line tool bq
- Making calls to the BigQuery REST API
- Using the variety of client libraries such as Java, .NET or Python





```
SELECT
  EXTRACT(YEAR FROM creation_date) AS year,
  EXTRACT(MONTH FROM creation_date) AS month,
  COUNT(creation_date) AS number_posts
FROM
  `bigquery-public-data.stackoverflow.stackoverflow_posts`
WHERE
  answer_count > 0
GROUP BY year, month
ORDER BY year ASC, month ASC;
```

Query complete (1.5 sec elapsed, 327 MB processed)

Row	year	month	number_posts
1	2008	7	4
2	2008	8	3947
3	2008	9	14585
4	2008	10	14977
5	2008	11	12971
6	2008	12	12335

GOOGLE CONTAINER ENGINE

Why Containers?

- Agile application creation and deployment: increased ease and efficiency of container image creation compared to VM image use.

- Loosely coupled, distributed, elastic, liberated micro-services: applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a monolithic stack running on one big single-purpose machine.

- Continuous development, integration, and deployment

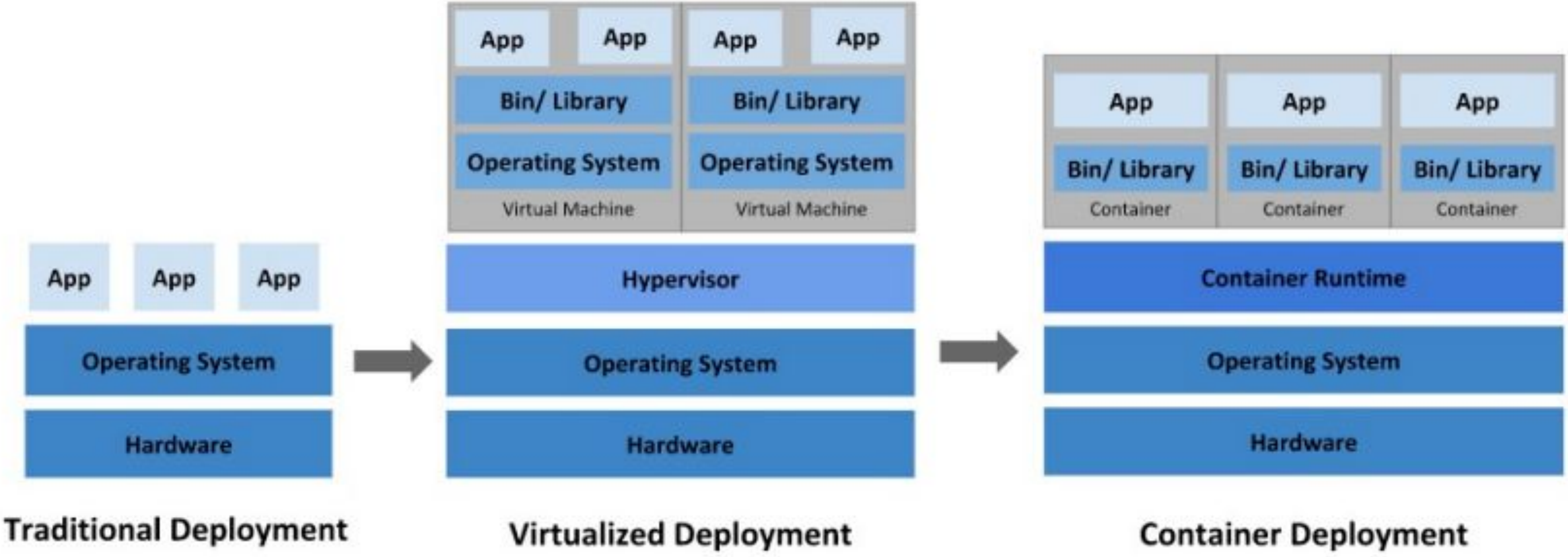
- Environmental consistency across development, testing, and production: Runs the same on a laptop as it does in the cloud.

- Cloud and OS distribution portability: Runs on Ubuntu, RHEL, CoreOS, on-premises, on major public clouds, and anywhere else.

- Application-centric management: Raises the level of abstraction from running an OS on virtual hardware to running an application on an OS using logical resources.

- Resource utilization: high efficiency and density

Why Containers?

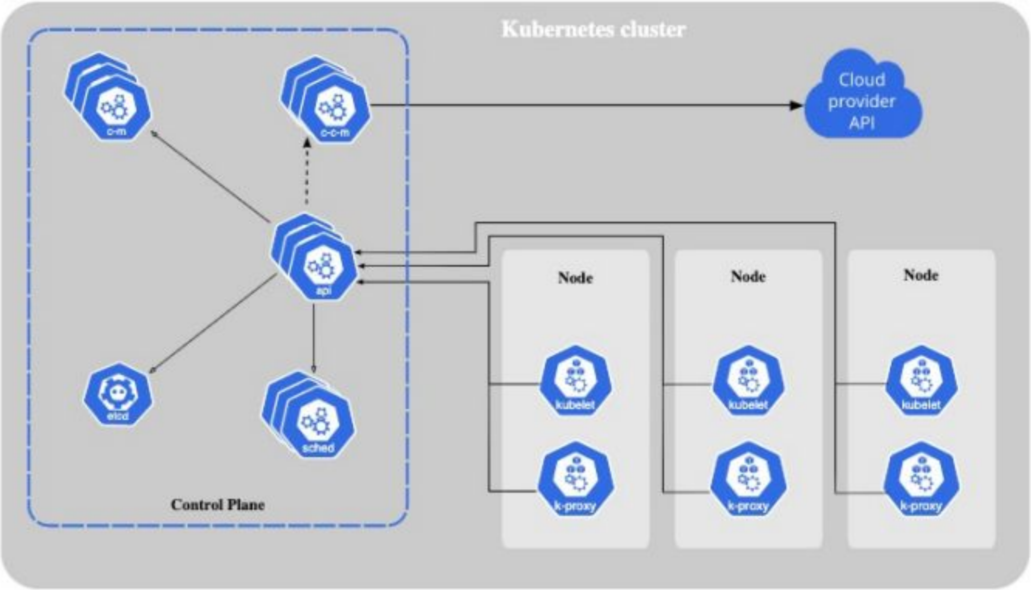


Why Kubernetes?

- Service discovery and load balancing Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.
- Storage orchestration Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.
- Automated rollouts and rollbacks
- Automatic bin packing You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.
- Self-healing Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- Secret and configuration management Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration

Kubernetes: Architecture & Components

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. The worker node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster.



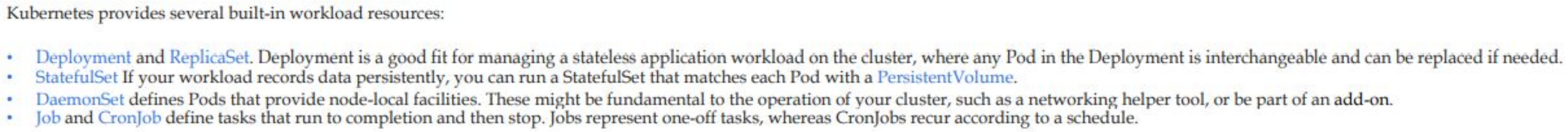
Kubernetes: Architecture & Components

Open Source Kubernetes architecture

- The core of Kubernetes' control plane is the API server. The API server exposes an HTTP API that lets end users communicate with the cluster.

- kubectl - command-line interface or other command line tools, such as kubeadm.

- In Kubernetes, a Pod is the most basic deployable unit within a Kubernetes cluster. A Pod runs one or more containers. Zero or more Pods run on a node. Each node in the cluster is part of a node pool.



Kubernetes provides several built-in workload resources:

- Deployment and ReplicaSet. Deployment is a good fit for managing a stateless application workload on the cluster, where any Pod in the Deployment is interchangeable and can be replaced if needed.
- StatefulSet If your workload records data persistently, you can run a StatefulSet that matches each Pod with a PersistentVolume.
- DaemonSet defines Pods that provide node-local facilities. These might be fundamental to the operation of your cluster, such as a networking helper tool, or be part of an add-on.
- Job and CronJob define tasks that run to completion and then stop. Jobs represent one-off tasks, whereas CronJobs recur according to a schedule.

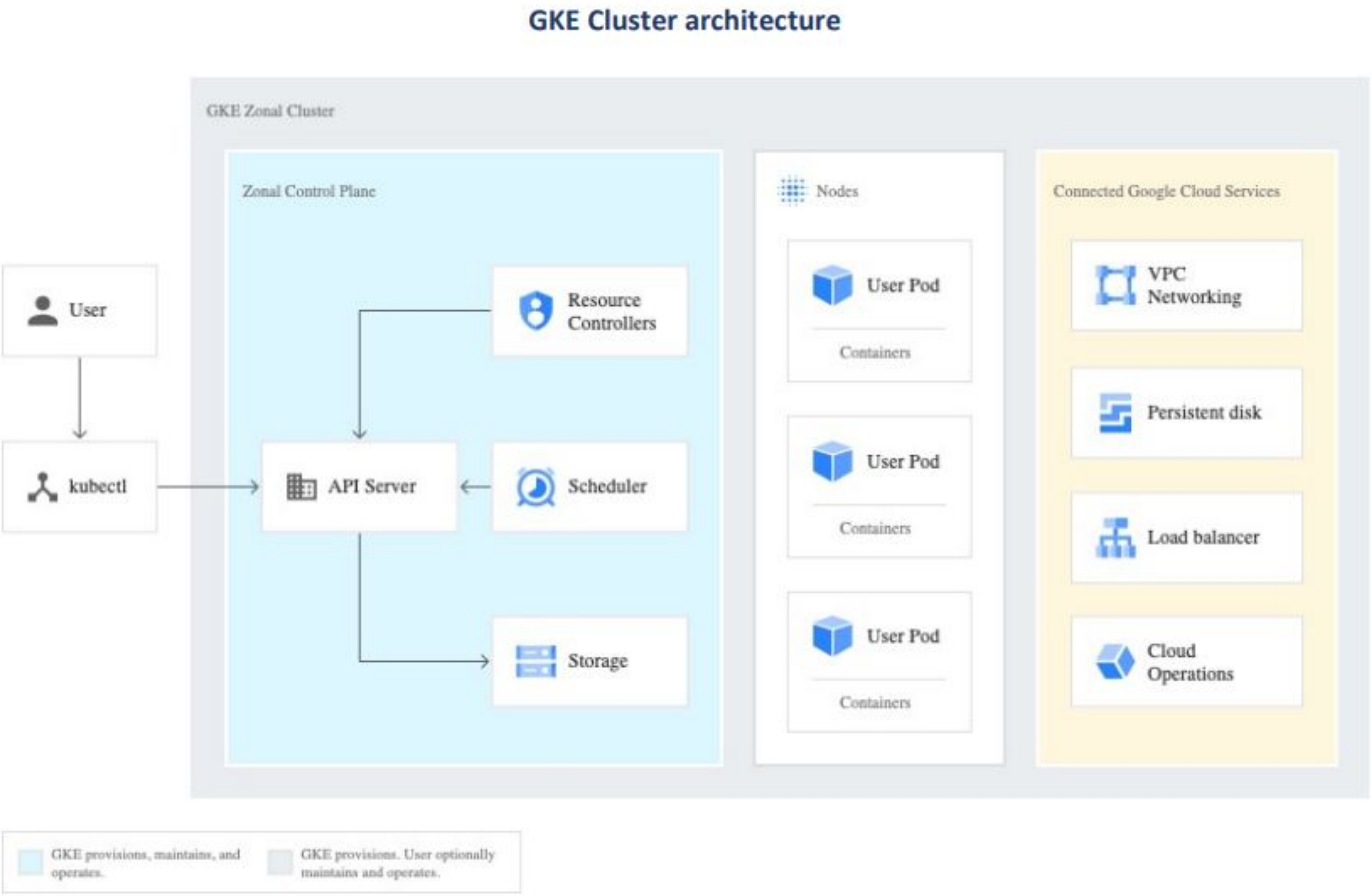
Kubernetes: Google Kubernetes Engine(GKE)

Google Kubernetes Engine (GKE) provides a managed environment for Kubernetes using Google infrastructure.

Kubernetes draws on the same design principles that run popular Google services and provides the same benefits: automatic management, monitoring and liveness probes for application containers, automatic scaling, rolling updates, and more.

GKE cluster control planes are automatically upgraded to run new versions of Kubernetes as those versions become stable to take advantage of newer features from the open source Kubernetes project.

Kubernetes: Google Kubernetes Engine(GKE)



Kubernetes: Google Kubernetes Engine(GKE)

When you run a GKE cluster, you gain the benefit of advanced cluster management features that Google Cloud provides. These include:

- Google Cloud's loadbalancing for Compute Engine instances
- Node pools to designate subsets of nodes within a cluster for additional flexibility
- Automatic scaling of cluster's node instance count
- Automatic upgrades for cluster's node software
- Node auto-repair to maintain node health and availability
- Logging and monitoring with Google Cloud's operations suite for visibility into your cluster

Kubernetes: Google Kubernetes Engine(GKE)

Modes of operation GKE clusters have two modes of operation to choose from:

- **Autopilot:** Manages the entire cluster and node infrastructure for you. Autopilot provides a hands-off Kubernetes experience so that you can focus on your workloads and only pay for the resources required to run your applications. Autopilot clusters are preconfigured with an optimized cluster configuration that is ready for production workloads.

- **Standard:** Provides you with node configuration flexibility and full control over managing your clusters and node infrastructure. For clusters created using the Standard mode, you determine the configurations needed for your production workloads, and you pay for the nodes that you use.

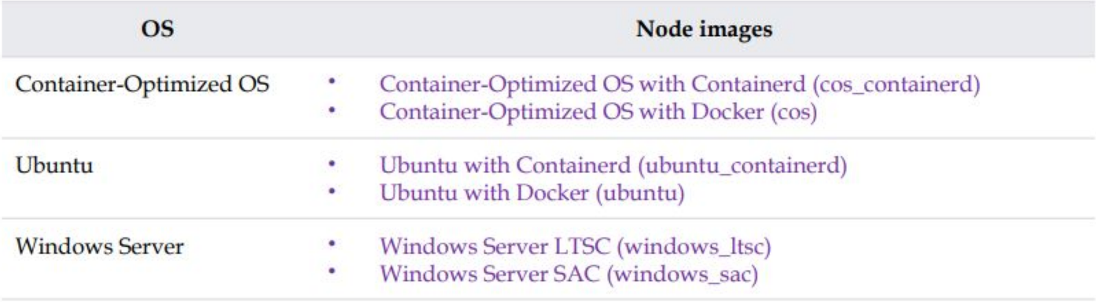
GKE Cluster Configurations

Some of the clusters configuration:

Node pools: A node pool is a group of nodes within a cluster that all have the same configuration. Node pools use a NodeConfig specification. Each node in the pool has a Kubernetes node label, cloud.google.com/gke-nodepool, which has the node pool's name as its value.

Node images: When you create a GKE cluster or node pool, you can choose the operating system image that runs on each node. GKE Autopilot clusters use only the cos_containerd node image.

The Container-Optimized OS from Google node images are based on a recent version of the Linux kernel and are optimized to enhance node security. Container-Optimized OS images are backed by a team at Google that can quickly patch images for security and iterate on features. The Container-Optimized OS images provides better support, security, and stability than other images.



OS	Node images
Container-Optimized OS	• Container-Optimized OS with Containerd (cos_containerd) • Container-Optimized OS with Docker (cos)
Ubuntu	• Ubuntu with Containerd (ubuntu_containerd) • Ubuntu with Docker (ubuntu)
Windows Server	• Windows Server LTSC (windows_ltsc) • Windows Server SAC (windows_sac)

GKE Cluster Configurations

Cluster autoscaler: GKE's cluster autoscaler automatically resizes the number of nodes in a given node pool, based on the demands of workloads. You don't need to manually add or remove nodes or over-provision your node pools. Instead, you specify a minimum and maximum size for the node pool, and the rest is automatic.

- If Pods are unschedulable because there are not enough nodes in the node pool, cluster autoscaler adds nodes, up to the maximum size of the node pool.

- If nodes are under-utilized, and all Pods could be scheduled even with fewer nodes in the node pool, Cluster autoscaler removes nodes, down to the minimum size of the node pool. If the node cannot be drained gracefully after a timeout period (currently 10 minutes), the node is forcibly terminated. The grace period is not configurable for GKE clusters.

Horizontal Pod Autoscaling: HPA changes the shape of your Kubernetes workload by automatically increasing or decreasing the number of Pods in response to the workload's CPU or memory consumption, or in response to custom metrics.

Vertical Pod Autoscaling frees you from having to think about what values to specify for a container's CPU requests and limits and memory requests and limits. The autoscaler can recommend values for CPU and memory requests and limits, or it can automatically update the values.

Kubernetes: GKE Networking

The Kubernetes networking model relies heavily on IP addresses. Services, Pods, containers, and nodes communicate using IP addresses and ports. Kubernetes provides different types of load balancing to direct traffic to the correct Pods.

- ClusterIP: The IP address assigned to a Service. In other documents, it may be called the "Cluster IP". This address is stable for the lifetime of the Service.
- Pod IP: The IP address assigned to a given Pod, which is ephemeral.
- Node IP: The IP address assigned to a given node.

Kubernetes: GKE Networking

IP allocation:

Kubernetes uses various IP ranges to assign IP addresses to nodes, Pods, and Services.

- Each node has an IP address assigned from the cluster's Virtual Private Cloud (VPC) network.
- Each node has a pool of IP addresses that GKE assigns Pods running on that node (a /24 CIDR block by default). Each Pod has a single IP address assigned from the Pod CIDR range of its node. This IP address is shared by all containers running within the Pod, and connects them to other Pods running in the cluster.
- Each Service has an IP address, called the ClusterIP, assigned from the cluster's VPC network.

Services:

In Kubernetes, you can assign arbitrary key-value pairs called labels to any Kubernetes resource. Kubernetes uses labels to group multiple related Pods into a logical unit called a Service. A Service has a stable IP address and ports, and provides load balancing among the set of Pods whose labels match all the labels you define in the label selector when you create the Service.

Kubernetes: GKE Networking

Kube-Proxy:

- Kubernetes manages connectivity among Pods and Services using the kube-proxy component. This is deployed as a static Pod on each node by default.

- kube-proxy, which is not an in-line proxy, but an egress-based load- balancing controller, watches the Kubernetes API server and continually maps the ClusterIP to healthy Pods by adding and removing destination NAT (DNAT) rules to the node's iptables subsystem. When a container running in a Pod sends traffic to a Service's ClusterIP, the node selects a Pod at random and routes the traffic to that Pod.

Networking outside the cluster:

- External load balancers manage traffic coming from outside the cluster and outside your Google Cloud Virtual Private Cloud (VPC) network. They use forwarding rules associated with the Google Cloud network to route traffic to a Kubernetes node.

- Internal load balancers manage traffic coming from within the same VPC network. Like external load balancers, they use forwarding rules associated with the Google Cloud network to route traffic to a Kubernetes node.

- HTTP(S) load balancers are specialized external load balancers used for HTTP(S) traffic. They use an Ingress resource rather than a forwarding rule to route traffic to a Kubernetes node.

Kubernetes: GKE Load Balancing

Service networking is the publishing of applications in a way that abstracts the underlying ownership, implementation, or environment of the application that is being consumed by clients.

Exposing an application to clients involves three key elements of a Service:

1. Frontend

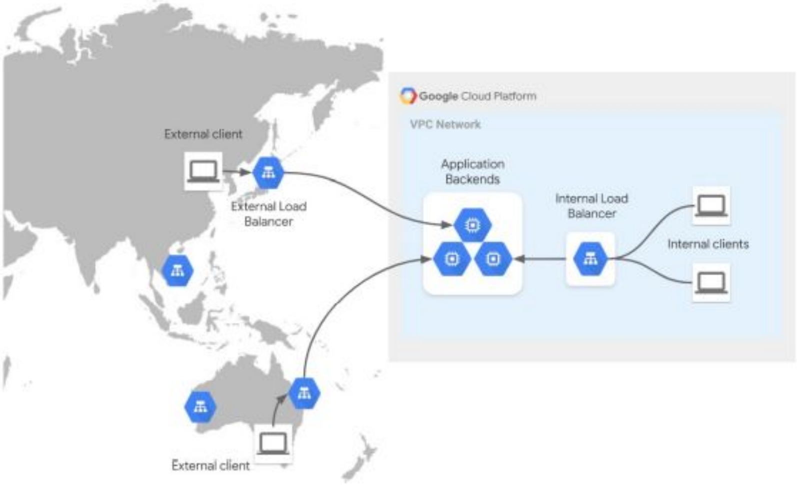
2. Routing and load balancing

3. Backends

Kubernetes: GKE Load Balancing

The following diagram illustrates these concepts for internal and external traffic flows in Google Cloud:

- In this diagram, the External HTTP(S) Load Balancer is listening for traffic on the public internet through hundreds of Google points of presence around the world. This global frontend allows traffic to be terminated at the edge, close to clients, before it load balances the traffic to its backends in a Google data center.



- The Internal HTTP(S) load balancer listens within the scope of your VPC network, allowing private communications to take place internally. These load balancer properties make them suited for different kinds of application use cases

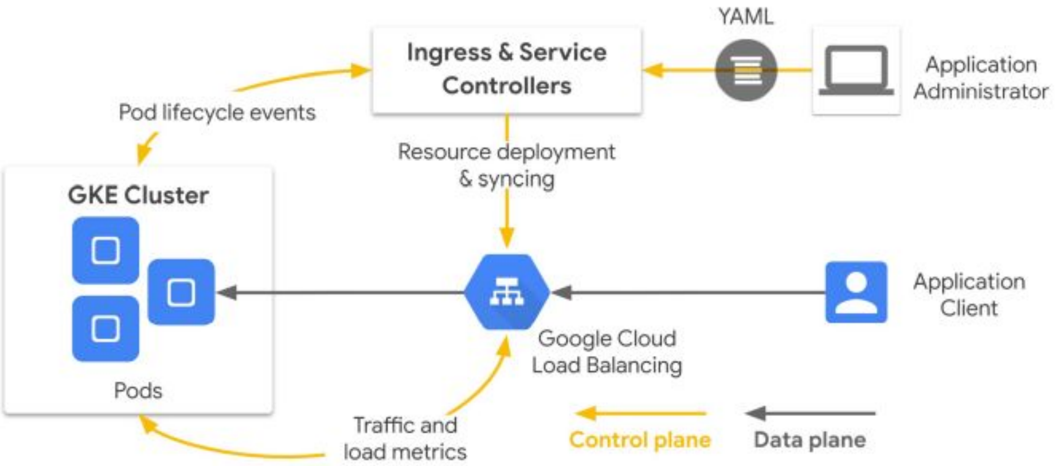
Kubernetes: GKE Load Balancing

The following diagram illustrates how the GKE network controllers automate the creation of load balancers:

As displayed in the diagram, an infrastructure or app admin deploys a declarative manifest against their GKE cluster.

- Ingress and Service controllers watch for GKE networking resources (such as Ingress or MultiClusterIngress objects) and deploy Google Cloud load balancers (plus IP addressing, firewall rules, and so on) based on the manifest.

- The controller continues managing the load balancer and backends based on environmental and traffic changes. Because of this, GKE load balancing becomes a dynamic and self-sustaining load balancer with a simple and developer-oriented interface.


YAML
Ingress & Service Controllers
Application Administrator
Pod lifecycle events
Resource deployment & syncing
GKE Cluster
Pods
Google Cloud Load Balancing
Application Client
Traffic and load metrics
Control plane
Data plane

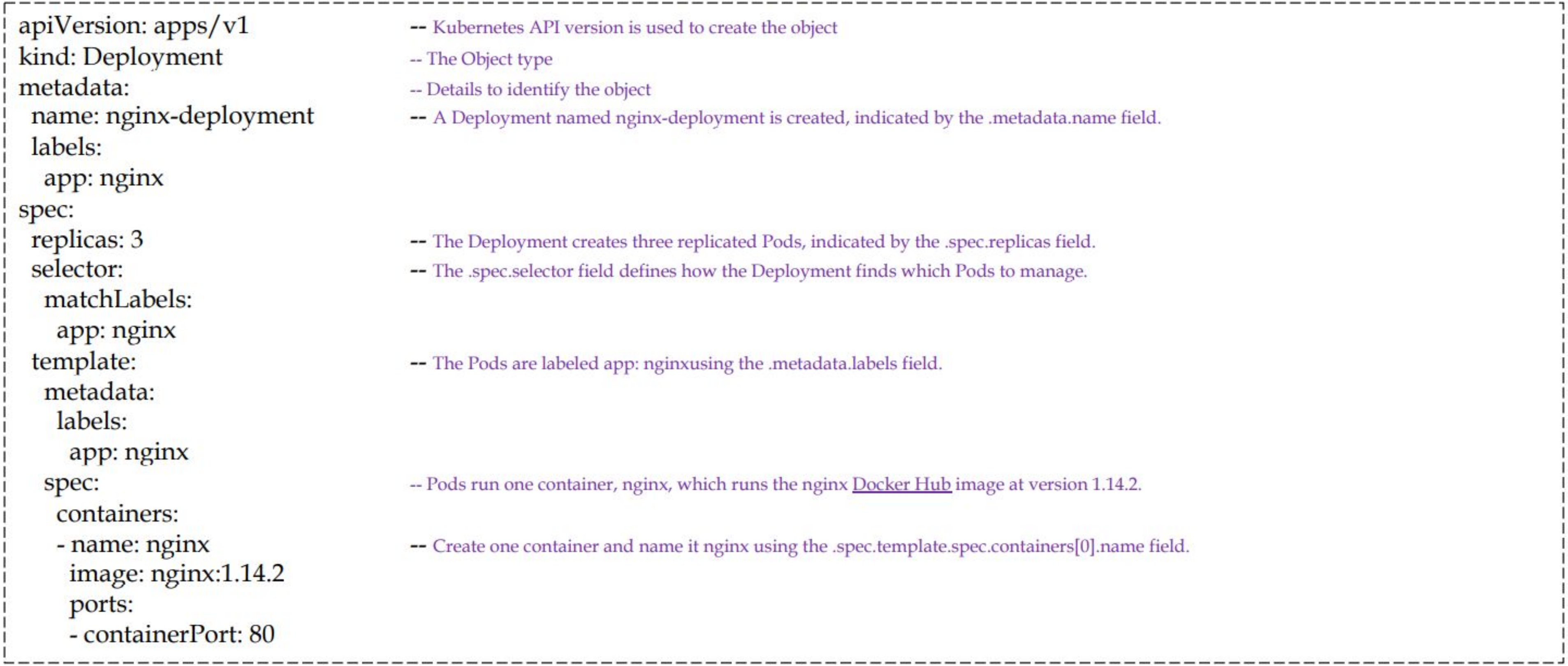
Kubernetes: Deployments

Kubernetes objects can be created, updated, and deleted by storing multiple object configuration files in a directory and using kubectl apply to recursively create and update those objects as needed.

A Deployment provides declarative updates for Pods and ReplicaSets. You describe a desired state in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate.

You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

Kubernetes: Deployments

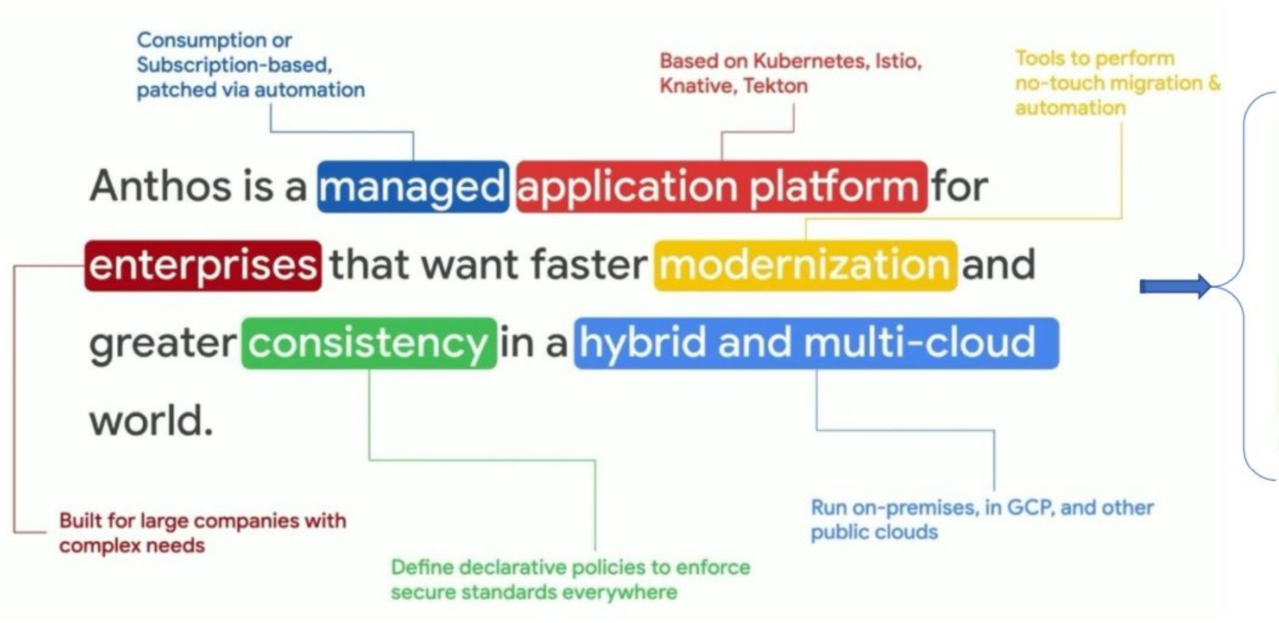


```
apiVersion: apps/v1                -- Kubernetes API version is used to create the object
kind: Deployment                   -- The Object type
metadata:                          -- Details to identify the object
  name: nginx-deployment           -- A Deployment named nginx-deployment is created, indicated by the .metadata.name field.
  labels:
    app: nginx
spec:
  replicas: 3                      -- The Deployment creates three replicated Pods, indicated by the .spec.replicas field.
  selector:                        -- The .spec.selector field defines how the Deployment finds which Pods to manage.
    matchLabels:
      app: nginx
  template:                        -- The Pods are labeled app: nginxusing the .metadata.labels field.
    metadata:
      labels:
        app: nginx
    spec:                          -- Pods run one container, nginx, which runs the nginx Docker Hub image at version 1.14.2.
      containers:
      - name: nginx                -- Create one container and name it nginx using the .spec.template.spec.containers[0].name field.
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Anthos

Anthos: Introduction

Anthos is a modern application management platform that provides a consistent development and operations experience for cloud and on-premises environments.



Consumption or Subscription-based, patched via automation

Based on Kubernetes, Istio, Knative, Tekton

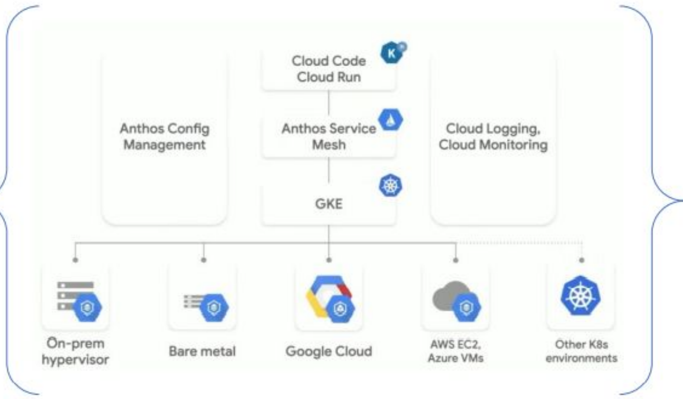
Tools to perform no-touch migration & automation

Anthos is a managed application platform for enterprises that want faster modernization and greater consistency in a hybrid and multi-cloud world.

Built for large companies with complex needs

Define declarative policies to enforce secure standards everywhere

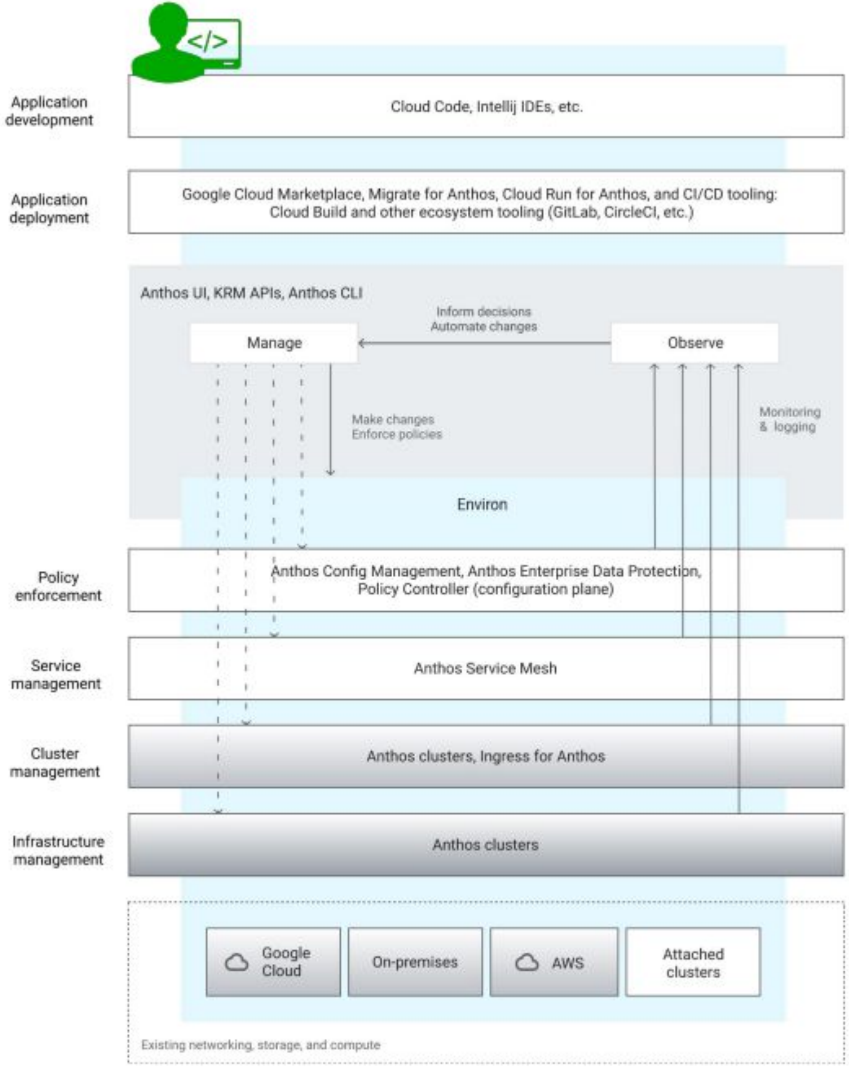
Run on-premises, in GCP, and other public clouds



Core Components of Anthos

Anthos: Technical Overview

The following diagram shows Anthos components and features and how they provide Anthos's functionality across your environments, from infrastructure management to facilitating application development.



</>
Application development
Cloud Code, Intellij IDEs, etc.
Application deployment
Google Cloud Marketplace, Migrate for Anthos, Cloud Run for Anthos, and CI/CD tooling: Cloud Build and other ecosystem tooling (GitLab, CircleCI, etc.)
Anthos UI, KRM APIs, Anthos CLI
Inform decisions
Automate changes
Manage
Observe
Make changes
Enforce policies
Monitoring & logging
Environ
Policy enforcement
Anthos Config Management, Anthos Enterprise Data Protection, Policy Controller (configuration plane)
Service management
Anthos Service Mesh
Cluster management
Anthos clusters, Ingress for Anthos
Infrastructure management
Anthos clusters
Google Cloud
On-premises
AWS
Attached clusters
Existing networking, storage, and compute

Anthos: Technical Overview

Application development

The Kubernetes ecosystem is continually expanding and creating a wealth of functionality that can be enabled on top of existing clusters. In the marketplace solution catalog, diverse range of solutions can be found:

- Storage solutions
- Databases
- Continuous integration and delivery tools
- Monitoring solutions
- Security and compliance tools

Anthos: Technical Overview

- Unified user interface

The Anthos dashboard in the Google Cloud Console provides a secure, unified user interface to view and manage applications, including an out of-the-box structured view of all the Anthos resources.

- Environ

An environ is a domain that groups clusters and infrastructure, manages resources, and keeps a consistent policy across them.

- Consolidated logging and monitoring

Anthos includes Cloud Logging and Cloud Monitoring for system components and Kubernetes Engine Monitoring is enabled by default.

Anthos: Technical Overview

Infrastructure management

- Anthos clusters provide a unified way to work with Kubernetes clusters as part of Anthos, extending GKE to work in multiple environments. You have consistent, unified, and secure infrastructure, cluster, and container management, whether you're using Anthos on Google Cloud (with traditional GKE), hybrid cloud, or multiple public clouds.

- Anthos uses Anthos clusters, which extend GKE for use on Google Cloud, on-premises, or multicloud. These offerings bundle upstream Kubernetes releases and provide management capabilities for creating, scaling, and upgrading conformant Kubernetes clusters.

Anthos: Service Mesh

- A service mesh is an architecture that enables managed, observable, and secure communication across services.

- It factors out all the common concerns of running a service such as monitoring, networking, and security, with consistent, powerful tools, making it easier for service developers and operators to focus on creating and managing great applications for their users.

- Anthos Service Mesh is powered by Istio, a highly configurable and powerful open source service mesh platform.

Anthos: Service Mesh

Istio Architecture:

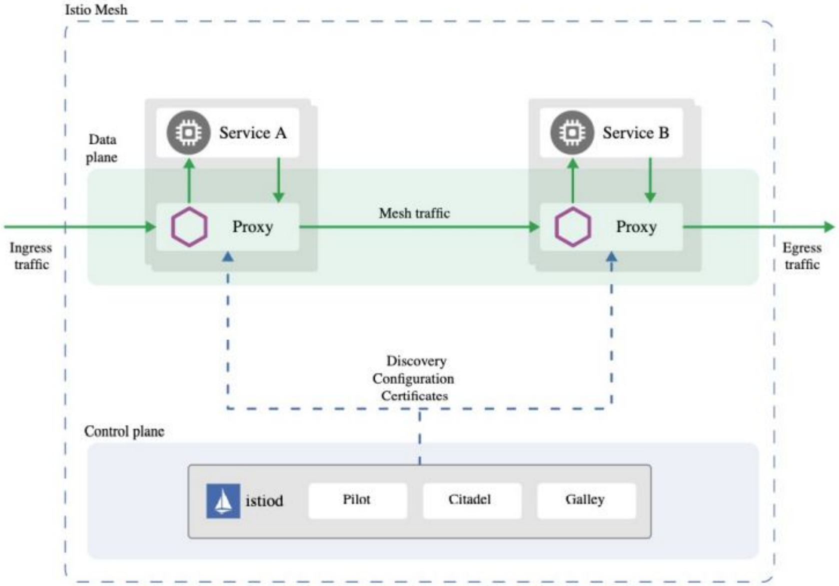
- Architecturally, a service mesh consists of one or more control planes and a data plane. The service mesh monitors all traffic through a proxy.
- On Kubernetes, the proxy is deployed by a sidecar pattern to the microservices in the mesh.
- On Virtual Machines (VMS), the proxy is installed on the VM.
- This pattern decouples application or business logic from network functions, and enables developers to focus on the features that the business needs.

Anthos: Service Mesh

Why use Istio?

Istio makes it easy to create a network of deployed services with load balancing, service-to service authentication, monitoring, and more, with few or no code changes in service code.

Istio is designed for extensibility and meets diverse deployment needs. It does this by intercepting and configuring mesh traffic as shown in the following diagram:



Istio Architecture

Anthos: Service Mesh

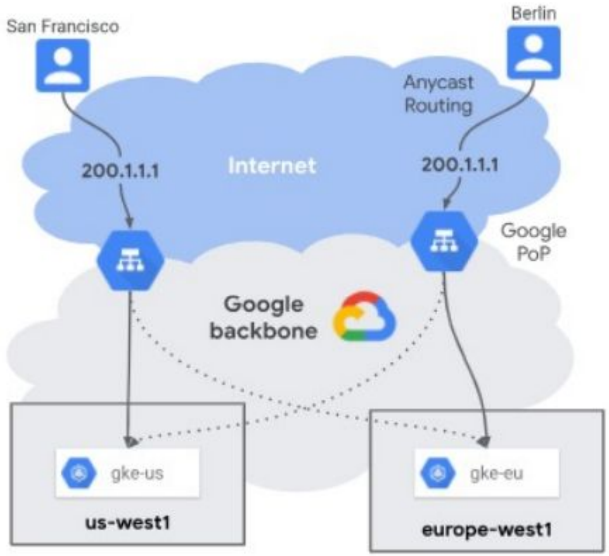
How to use Istio?

You add Istio support to services by deploying a special sidecar proxy throughout your environment that intercepts all network communication between microservices, then configure and manage Istio using its control plane functionality, which includes:

- Automatic load balancing for HTTP, gRPC, WebSocket, and TCP traffic.
- Fine-grained control of traffic behavior with rich routing rules, retries, failovers, and fault injection.
- A pluggable policy layer and configuration API supporting access controls, rate limits and quotas.
- Automatic metrics, logs, and traces for all traffic within a cluster, including cluster ingress and egress.
- Secure service-to-service communication in a cluster with strong identity-based authentication and authorization.

Multi-cluster Ingress

Multi-cluster Ingress (MCI) is a cloud-hosted multi-cluster Ingress controller for Anthos GKE clusters.It's a Google-hosted service that supports deploying shared load balancing resources across clusters and across regions.



Multi-cluster networking:

Many factors drive multi-cluster topologies, including close user proximity for apps, cluster and regional high availability, security and organizational separation, cluster migration, and data locality. Multi-cluster Ingress is designed to meet the load balancing needs of multi-cluster, multi-regional environments. It's a controller for the external HTTP(S) load balancer to provide ingress for traffic coming from the internet across one or more clusters.

Multi-cluster Ingress

Multi-cluster Ingress's multi-cluster support satisfies many use cases including:

- A single, consistent virtual IP (VIP) for an app, independent of where the app is deployed globally.
- Multi-regional, multi-cluster availability through health checking and traffic failover.
- Proximity-based routing through public Anycast VIPs for low client latency.
- Transparent cluster migration for upgrades or cluster rebuilds

Multi-cluster Ingress architecture

Multi-cluster Ingress uses a centralized Kubernetes API server to deploy Ingress across multiple clusters.
This centralized API server is called the config cluster. Any GKE cluster can act as the config cluster.

The config cluster uses two custom resource types: **MultiClusterIngress** and **MultiClusterService**.

By deploying these resources on the config cluster, the Anthos Ingress Controller deploys load balancers across multiple clusters.

The following concepts and components make up Multi-cluster Ingress:

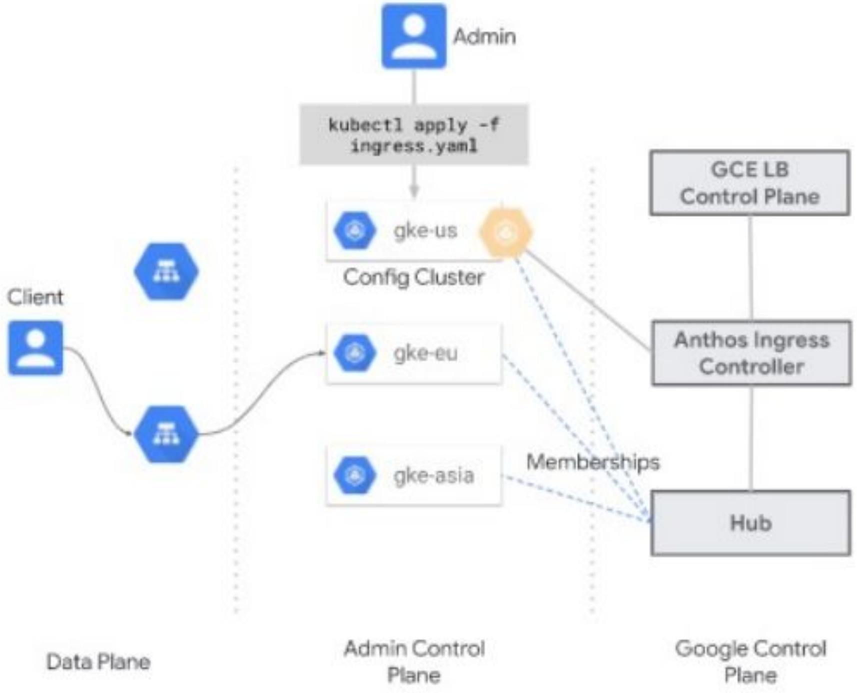
- **Anthos Ingress controller** - This is a globally distributed control plane that runs as a service outside of your clusters. This allows the lifecycle and operations of the controller to be independent of GKE clusters.

- **Config cluster** - This is a chosen GKE cluster running on Google Cloud where the MultiClusterIngress and MultiClusterService resources are deployed.

- **Environ** - An environ is a domain that groups clusters and infrastructure, manages resources, and keeps a consistent policy across them.

- **Member cluster** - Clusters registered to an environ are called member clusters. Member clusters in the environ comprise the full scope of backends that MCI is aware of. The Google Kubernetes Engine cluster management view provides a secure console to view the state of all your registered clusters.

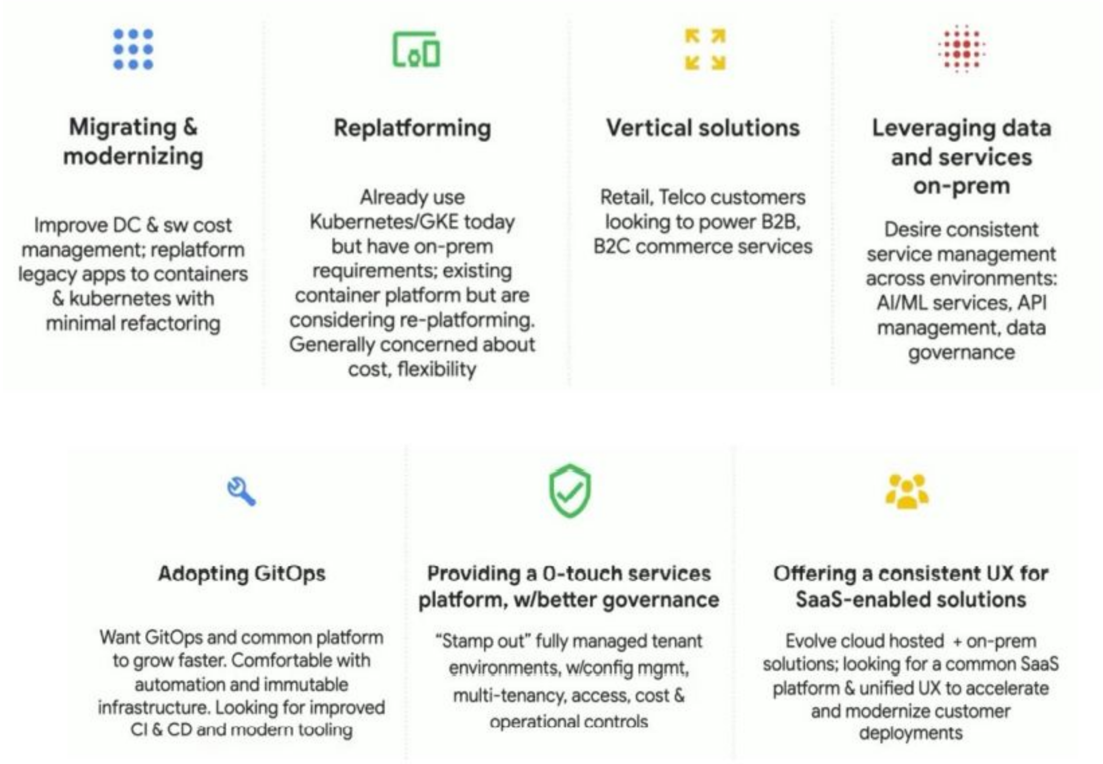
Multi-cluster Ingress architecture



```
kubectl apply -f
ingress.yaml
```

Anthos: Benefits & Use Cases

Anthos is a managed application platform that extends Google Cloud services and engineering practices to other environments so that you can modernize apps faster and establish operational consistency across them.



Migrating & modernizing

Improve DC & sw cost management; replatform legacy apps to containers & kubernetes with minimal refactoring

Replatforming

Already use Kubernetes/GKE today but have on-prem requirements; existing container platform but are considering re-platforming. Generally concerned about cost, flexibility

Vertical solutions

Retail, Telco customers looking to power B2B, B2C commerce services

Leveraging data and services on-prem

Desire consistent service management across environments: AI/ML services, API management, data governance

Adopting GitOps

Want GitOps and common platform to grow faster. Comfortable with automation and immutable infrastructure. Looking for improved CI & CD and modern tooling

Providing a 0-touch services platform, w/better governance

"Stamp out" fully managed tenant environments, w/config mgmt, multi-tenancy, access, cost & operational controls

Offering a consistent UX for SaaS-enabled solutions

Evolve cloud hosted + on-prem solutions; looking for a common SaaS platform & unified UX to accelerate and modernize customer deployments

Anthos: Benefits & Use Cases

Benefits:

- **Manage applications anywhere**

Anthos gives a consistent platform for all application deployments, both legacy as well as cloud native, while offering a service-centric view of all the environments.

- **Deliver software faster**

Build enterprise-grade containerized applications faster with managed Kubernetes on cloud and on-premises environments. Create a fast, scalable software delivery pipeline with cloudnative tooling and guidance.

- **Protect applications and software supply chain**

Leverage a programmatic, outcome-focused approach to managing policies for apps across environments, and enable greater awareness and control with a unified view of services' health and performance.

THANK YOU!