Google Cloud Summary

A compilation of the following videos:

GCP Fundamentals Smplilearn (35 min)
Top 50+ Google Cloud Services Explained in 7 Minutes
AI-generated summary
Google Cloud in 4 words

Fundamentals

Cloud computing is all about getting things done using someone else's computers (Google's computers). Google Cloud lets you build and host apps, store data, analyze data using highly scalable and reliable computing infrastructure.

Cloud terminology is not settled – PaaS can mean slightly different things with different cloud providers. We will have a brief tour of the **hardware underpinnings**, names that GCP gives to **various abstractions**, e.g. global, regional, or zonal resource. These **abstractions live on top of the hardware**, and there is a distinct difference between the two.

Why choose cloud?

- The **majority of the cost of a computer over its four-year life is energy**. Google is very focused on energy efficiency, and it invested a lot in security and extremely high-speed networking - small companies can't match that investment. Google infra has **well-designed global meshed redundant fiber optic network** that spans the globe + they build new connections. People are realizing that **it's better to rent** instead of buy computers. Google has been **carbon neutral** since 2007, investing heavily in wind and solar. Recently, they've **applied AI** to optimize energy use at data centers.
- **Scalability:** retailer looking to set up a DB for inventory, pricing, demand across thousands of stores. Challenge – handling *massive seasonal / holiday spikes*.
  - **On-premises DB**: expensive and inefficient because of large *upfront costs* for additional DB infra (*not needed* during most of the year) + worry about implementing your own DB *sharding strategy, backups*, meeting *performance requirements*, *securing* the DB.
  - Alternative: **fully managed DB Cloud Spanner** - all this is managed for you. GCP manages scaling - you pay for the data storage you actually need.
- Google had years of experience building GCP for Search, Gmail, YouTube, other offerings. Over a long period of time, they have perfected their technology Now they're offering GCP so that **we could build apps on their platform that scale just as Google apps scale**: no one ideal way to do things => Google offers several choices in computing, storage, and big data + ML.
- Google Cloud allows to **seamlessly integrate multiple products for your unique use case**. Example - social *networking site for dogs*: a) *Cloud Storage* to store profile photos, b) *Cloud Firestore DB* to store other profile info (dog names, locations, and

hobbies), c) *Vision AI* to detect objects in each photo (balls, stuffed animals, and bones), d) *Cloud Run* to deploy your app.

Data centers around the world - where all the **low-level compute power** resides. They **house the compute, storage, and networking capabilities** that power *Google's own products* and services, including Search, Gmail, and YouTube. With Google Cloud, Google is *sharing these resources* with developers so you can build and run apps. **Data center** - a *highly secure facility* with one or more buildings that have their own power substations, high security with perimeter fencing and various codes to get inside. This. **Point of presence** - edge of Google and the rest of the internet. If you need to send packets to Google, they will transit from your computer through the internet, hit a point of presence, and get on the Google fiber to find its way into the data center. Google has over **110 edge points of presence** now in 33 countries, and expanding aggressively. Edge caching makes interactions faster with customers.

Three levels of abstraction.

- **Zone** - roughly equivalent to a data center. It's where you have compute power (VM instances), disks, etc. A zone could fail, so it's best to have compute power in multiple zones and balance the requests across them. A **compute engine instance** is a **zonal resource**.
- Next abstraction - **region** = **geographic area containing multiple zones (**U.S. Central, Asia, or Europe). Network **load balancer** is a regional resource; it sends requests to different zones and be aware of the health of all the computers in them. If one fails, it will reroute requests to other computers in different zones. There are also load balancers across regions.
- Some resources are **global** and span the whole planet (all different regions), example - a **network** where you have a **single IP address** for your web app.

Your zonal compute engine connects to a global network and can talk to other computers connected to the global network. You can have machines on a LAN, but it will be a global network. Things are quite different in the cloud vs the physical world of computers.

Trend - **price of computing is going down**. Google passes the savings to us in four different ways:

1. **Sub-hour billing** - The idea here is that if you ask for a computer, I'll give you a specific example: you can ask for a 32-core, 208-gigabyte computer and run it for only 20 minutes. They don't round up to the next hour. The cost for renting a computer of that size for 20 minutes, maybe you're checking out some code or have a caching algorithm you want to test, will cost you about 50 cents. It's really economical for experimentation and getting things set up.
2. **Sustained use discount** - As you use the computing infrastructure throughout the month, they will give you an increasing discount that goes up to about 25 percent. They're rewarding not just temporary use but also sustained use.
3. **Custom machine type**. You can dial in exactly the amount of CPU power and memory that you need for your application. We're hearing reports that this will save customers about 10 or 15 percent.

4. **Preemptable instances** This is like scavenging for free, essentially unused compute power. If you build a system that anticipates a computer going down and your software can understand that and reschedule it on another computer, you can save fifty percent. The price is half of the same machine in non-preemptable mode.

Google releases innovations as open source – it learned how to run containers very efficiently on its infra and released Kubernetes, container management software. Google Container Engine uses it adding some special Google features. You can also run Kubernetes on your own machine, on Amazon, OpenStack, DigitalOcean, or anywhere. Also, a lot of the low-level libraries for connecting to GCP - helps everyone and moves the industry forward.

Three waves of cloud platforms:

- First wave - **co-location** when you send your equipment to a managed env or rent computers in a managed env, you can connect to your computer sitting in a rack, having the vendor's security and power.
- Next wave – virtual machines and **virtualization** of various parts. We are at the end of the second wave
- The third wave - completely **global elastic cloud**. You just ask for compute power and get it very quickly + scale up and down based on our needs.

In the old days, you had to invest in many servers + software, get them hosted, size them during peak loads. If no peak load later - wasting your investment. With **elastic cloud** you can use / pay for compute resources that you need – **no huge upfront costs**, size things with your usage, much easier to manage.

IaaS vs. PaaS (can be used simultaneously, but different philosophically)

- **IaaS** - *Compute Engine, raw compute power*, storage, networking, etc; closest analog to the physical world (computer w/motherboard, chip, RAM, disks).
- **PaaS** - you supply business logic, and Google takes care of details giving you runtimes in a *sandbox env and scaling things up and down for you automatically*, charging you only for the resources you use (a hundred App Engines or just one).

Compute as an example:

- **Compute Engine = IaaS**, *you build your computers*, decide how big they are and scale them. Google helps with some things, like *moving a VM to different hardware* if they need to do maintenance – still a huge benefit to run things in the cloud.
- **App Engine = PaaS**, most constrained env, not much flexibility, but the benefit - *you really only write the business logic*. Google takes care of everything else: all infra, software updates, patching, scaling.
- **Container Engine = in the middle**, a level above VMs - lightweight Docker containers coming to life very quickly or scaling up or scaling down.

GCP Services

The cloud platform is comprised of many different services a total of 272; everything is communicated through a REST API.

cloud.google.com – access console or search docs for getting started resources, like quickstarts, trainings, certifications, solutions, reference architectures, and code samples.

Google Cloud Console – *manage all GCP resources* anywhere, manage projects, configure billing accounts, manage quotas for product use (to safeguard expenses), interactive quickstarts, marketplace with ready-to-go software stacks to accelerate development. Checklist to complete *common tasks* like creating a project.

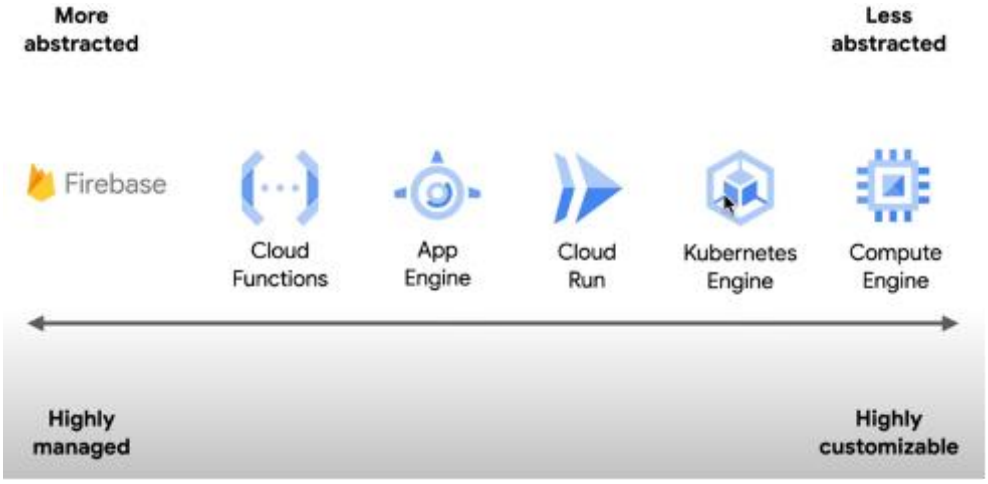
Check out the main **categories of products** Google Cloud has to offer: Compute, Storage, Networking, Operations, Tools, Big Data, and Artificial Intelligence. On the Getting Started page, scroll down to "Put Google Cloud to Work," and click on "Browse Interactive Tutorials."

iOS and Android apps - monitor and manage GCP apps.

Google Cloud SDK contains **gcloud** – scriptable, all-mighty **CLI**. Every action in the console has a gcloud command. You can create resources, change configs, grant authorization. There are other CLIs for BigQuery and GKE.

Gcloud SDK and CLIs come **preinstalled in Cloud Shell** which is a shell env hosted on Google Cloud for managing projects and resources, accessible from web browser, powered by a small VM w/persistent disk space and up-to-date software. Has a web code editor.

Compute (to run code)



The options below are listed in the order from highly managed to highly customizable. **Highly managed** - Google abstracts more of the backend server and infrastructure logic so you can get up and running quickly. **Highly customizable** - you are in charge of more of the backend server and infrastructure logic => more flexibility to customize these options. GKE is more customizable than Cloud Run.

Firebase - *highly managed* option. Google's **app development platform**; includes a DB, storage, functions, authentication, notifications, etc. Firebase may be used for building a **mobile app w/little back-end code** - use cases where most of **processing is done on the client side**. E.g. to build a *quiz game for iOS and Android* - store quiz questions in Cloud Firestore DB, help users log in with Firebase Authentication, and store profile photos in Firebase Storage.

Cloud Functions – if you need some back-end code, but still as a highly managed solution, use **event-driven serverless functions** which respond to cloud events without managing infra (server or runtime env) - triggered by HTTP events like a user visiting a URL or when something happens in another GCP product, like when a DB is updated etc. Example, send a welcome email whenever a new user creates an account.

They scale automatically, so you only pay when your function is executed. Use cases: serverless mobile backends, IoT backends, real-time file processing, and virtual assistants. Languages: Python, Ruby, PHP, Node.js, Java, Go, and .NET.

Google App Engine – if you can't separate code into independent functions, but still want a relatively managed solution => App Engine - higher-level PaaS, **fully managed** env for deploying web apps and backend services to the cloud with **auto-scaling** (easy to scale up and down to meet user demands), developers focus on code only and GCP **manages infra**; choose from several libs & frameworks to develop apps. **Cloud Datastore** – unique data mechanism that scales alongside App Engine.

Example: online retail business - your website receives spikes of traffic during the holiday season, GAE will automatically scale your app up. When traffic returns to normal, GAE will scale back down - you only pay for the computing resources you actually need. Also, App Engine handles **load balancing**, traffic splitting, error reporting, etc.

Cloud Run – CaaS (container as a Service), **container orchestration** solution on the highly managed side, ideal for developers who want to develop and deploy **highly scalable** containerized stateless apps on a **fully managed, serverless platform with minimal configuration and auto-scaling**, ability to move applications across different environments easily. Used for simple and quick deploying of any stateless HTTP containers, for synchronous event-driven apps, microservices and APIs.

Handles tasks like monitoring app health and troubleshooting. It automatically and quickly scales up or down, even to zero. You only pay for the resources that your app uses (by 100th millisecond).

Google Kubernetes Engine (GKE) – CaaS, open-source container orchestration platform on a managed cluster of VMs, suited for users who need **more operational control over the infra** running containers + **customize cluster configurations** using the **full power and flexibility of Kubernetes** for orchestrating complex, large-scale containerized apps. GKE has a set of APIs allowing to control the lifecycle of Kubernetes clusters, incl. exposing them to clients, auto-scaling, running and monitoring, advanced orchestration capabilities, support for stateless and stateful apps.

Additional management layers for Kubernetes – easier *cluster upgrades*, some completely taken care of by Google. Key difference: Cloud Run is priced based on usage vs. *GKE - based on VM*. More fully managed GKE mode - *GKE autopilot*.

Google Compute Engine (GCE) - VMs, GPUs, TPUs, disks to run your code on; launch large compute clusters (instance types and groups?). **Highly customizable** option. While the previous compute options abstract away infra management, here you must **manage the server and infra**, which also means **more flexibility to customize** your computing resources (# cores, RAM, OOTB or custom OS image) => you can easily **run existing apps** on GCE without many changes.

Preemtible VMs – short-lived compute instances (4 words)

Shielded VMs – hardened VMs (4 words)

Sole-tenant nodes – dedicated physical servers (4 words)

Bare metal solutions – hardware for specialized workloads (4 words)

VMware Engine – VMware on Compute Engine (4 words)

Many developers may **use several of these compute products together**: e.g. a game developer may host DB on Firebase, game server on GKE, and email integration on Cloud Functions.

Autoscaling

Standard and Flexible Environments

Stateless Containerized App - treats each client request independently without storing any client data generated during a session or request, making it easier to scale and manage. Each request from a client = independent transaction. Easily scalable horizontally; any instance can handle any request. Simplified load balancing; any request can go to any instance. Instances can be easily replaced or scaled without concern for data loss or recovery. Ideal for services where each request is independent (e.g., RESTful APIs).

Stateful Containerized App - maintains client data across sessions and requests, requiring more complex management for scaling and recovery. Preserves state info = remembers previous interactions. Requests depend on data from previous interactions. Scaling requires managing state across instances, often involving data synchronization. Instances need to restore state information after recovery or scaling. Necessary for applications that need to maintain user or session data (e.g., user authentication services, databases)

Storage

Google Cloud Storage – multi-class multi-region *object storage*: for unstructured data like images, videos, and audio files. You can use cloud storage for a range of scenarios

including serving website content, storing data for archival and disaster recovery, or distributing large data objects to users via direct download.

Google Persistent Disk – *block storage* for VMs.

Local SSDs – *VM locally attached* SSDs; perform faster and have less latency than traditional or SSD persistent drives.

Google Cloud Filestore – *managed NFS server* (Network File System or NFS is a distributed file system protocol originally developed by Sun in 1984, allowing a user on a client computer to access files over a computer network much like local storage is accessed); rapid backups and snapshots to simplify the data preservation.

Databases

Performance and Scalability

High Availability and Replication

Backups and Restores

Bigtable – petabyte-scale, low-latency, high-throughput NoSQL databases for large analytical and operational workloads.

Cloud Firestore - highly *scalable serverless real-time NoSQL document DB* for server, web, mobile, and IoT apps. *JSON-compatible*, Firestore *vector store* for GenAI solutions. Popular for *gaming* – stores most up-to-date version of data in real time. Prev. generation: **Cloud Datastore**: auto-sharding and replication (=fully managed), highly available and durable, scales automatically; myriad of capabilities such as ACID transactions, SQL-like queries, indexes, etc.

MemoryStore – managed Redis and Memcached (caching solutions).

Cloud Spanner – horizontally scalable relational DB; globally distributed and dynamically replicated.

Google Cloud SQL – managed relational DBs: run PostgreSQL, MySQL, SQL Server without the hassles of self-management.

DB Migration Service – migrate to Cloud SQL.

Data Analytics

Blue highlight – familiar OpenText products.

BigQuery - completely managed, serverless big data warehouse / analytics. Extremely *high-performance SQL data processing* system: processes huge volumes of info for *near real-time data analysis*. Example: IoT devices sending info to BigQuery for analysis.

BigQuery BI Engine – in-memory analytics engine.

BigQuery ML (train / serve models), **GIS** (geospatial funcs), **DIS** (data ingestion service).

Cloud Composer - fully managed *workflow orchestrator*: build, plan, monitor, and manage processes that span data centers.

Dataflow – *batch / stream data processing*. Fully managed *streaming analytics* service minimizes latency, processing time, and cost through *auto-scaling and batch processing*. MapReduce v2.

Dataprep – *visual data wrangling*: explore, clean, and prepare data for analysis and ML.

Dataproc - Managed Hadoop and Spark. *Hadoop managed by Google*: it can spin a cluster in 90 sec and resize it to your workload => minimizes cost, *compatible with Hadoop processing outside of the cloud or in other clouds* => bring your Hadoop processing, run it on Dataproc, taking advantage of the Google platform.

Google Data Studio – customizable, informative data *reports and dashboards*.

Cloud Looker – *enterprise BI and Analytics*: minimizes complexity in and speeds up analysis with BigQuery.

Databrick – allows utilizing open source data tools.

Datalab – *Jupyter notebook-like presentation of big data processing*: execute queries, do graphics, and share it in a notebook format. Easy-to-use interactive tool for large-scale data exploration, analysis and visualization using GCP services such as Google BigQuery, Google App Engine Flex and Google Cloud Storage. TensorFlow support, local machine support.

Machine Learning

Helpful for experienced and new ML developers. Google has a ML system that beat the world's expert in the **Go game**.

1. **ML APIs** - uses pre-trained ML models to gain insights from data w/zero prior knowledge of ML - jump straight to prediction. Call all APIs from code w/client libraries in Python, Node.js, Java, Go, C#, PHP, Ruby:

   Vision API: a) *face and emotion detection* (joy, sorrow, and anger), b) *object detection* w/confidence score, c) *harmful content classification* (adult, racist, etc.), d) detect text, logos, and landmarks.

   Natural Language API - analyze text for entities, sentiment, analyze syntax, content categorization (by topic).

   Other APIs: Video Intelligence, Translation, STT, TTS, Cloud Inference API.

2. **AutoML** – custom low-code models - [build upon pretrained ML models to] auto-train on **custom training dataset +** provide a **prediction endpoint**: AutoML Vision, AutoML Natural Language, AutoML Translation, AutoML Video Intelligence, and AutoML Tables accessible via the Vertex AI section of Console.

Example - you're a *meteorologist looking to use ML to classify cloud types* (cirrus, cumulonimbus, or stratus). Vision API can identify if there are clouds in an image, but not what type of clouds. Upload *custom image dataset of clouds*, each one labeled as cirrus, cumulonimbus, or stratus => GCP uses it to train ML model. => you can use a prediction endpoint to classify new images of clouds. You'll be able to see **evaluation metrics** for the model, like precision, recall, and the confusion matrix.

3. **Vertex AI** - all ML services under a single platform: Workbench (Jupyter notebooks), Data Labeling, Preconfigured DL Containers, Matching Engine (vector similarity search), Pipelines (hosted ML workflows), Training (distributed AI training), Predictions (auto-scaled model serving), Explainable AI, AI Feature Store, Model Monitoring (skew / drift).

   Fully managed ML platform to simplify the process of build, deploy, and **scale** ML models faster using **custom training**. Ease of use: requires ~80% fewer lines of code to train a custom ML model on Vertex AI vs. other platforms (unified interface and set of tools). **Handles many aspects of the ML lifecycle**, including data preparation, model training, evaluation, deployment, and monitoring. **MLOps tools**: Model Monitoring for tracking model performance, Vertex AI Pipelines for orchestrating repeatable training and serving pipelines, and Vertex AI Feature Store for managing feature data.

   Scalability - handles scaling and resource management automatically (distributed training with multiple GPUs or TPUs).

   Integrates with other GCP services: BigQuery, Google Cloud Storage, and AutoML. This integration helps in streamlining data ingestion, processing, and analysis. Vertex AI **integrates TensorFlow, PyTorch**, **Scikit-learn** + other ML frameworks via custom containers for training and prediction.

   Interacting w/Vertex AI – *pre-packaged Jupyter notebooks* with commonly used ML frameworks, making it easy to start experimentation and development, GCP Console to manage your ML resources and get access to monitoring and logging, and Cloud Client libs and REST APIs to call Vertex AI from your code.

4. **AI Infra Tools** - preconfigured VMs for deep learning, if you want to handle more of ML process than offered by Vertex AI to build and host ML models – **DL VM Images** = **Google Compute Engine instances** that come pre-installed *w/latest ML frameworks like TensorFlow, PyTorch, and Scikit-learn* + Cloud *GPUs* and *TPUs* to speed up compute.

   More **flexible and customizable** than Vertex AI; you **have to manage the underlying infra** (VM, scaling, etc.), manually handle the entire ML lifecycle

5. **Dialogflow** – create *conversational* [user] interfaces.

6. **Document AI** – *analyze, classify, search documents*. Dashboards, parsers, and other tools.
7. **Recommendations AI** – create *custom recommendations*.

Networking

Virtual Private Cloud (VPC) – software defined networking.

Load Balancing (HTTP(S), TCP/SSL, Network LB) – multi-region load distribution. Fully distributed, software-defined managed service.

Direct Peering – peer w/GCP: link your business network to Google's edge network and exchange total cloud traffic.

Carrier Peering – peer through a carrier.

Dedicated Interconnect – dedicated private network connection.

Cloud Armor – *DDoS protection and Web Application Firewall (WAF)*: security system designed to protect web apps by filtering and monitoring HTTP traffic between a web app and the Internet. Prevents unwanted traffic from accessing VPC resources.

Cloud CDN – content delivery network: uses Google's global edge network to provide content closer to consumers => faster websites and apps.

Cloud DNS – programmable DNS serving: hierarchical, distributed database to store IP addresses and other data and look them up by name.

Cloud NAT - managed network address translation to convert to public IP addresses.

Cloud VPN - connect your on-premises network to your VPC network: a) classic, and b) HA (high-availability) – provides availability SLA (service-level agreement) of 99.99% or 99.9% depending on topology.

Cloud Router – VPC / on-prem network route exchange.

Network Intelligence Center – network monitoring and topology.

Identity & Security

Cloud Identity & Access Management (IAM) – resource access control.

Cloud Identity – manage users, devices, and apps in a centralized way.

Key Management Service (KMS) – hosted key management service: REST API that may encrypt, decrypt, or sign data for storage using a key.

Cloud Audit Logs – *audit trails* for GCP. Audit logs – type of logs that answer "who did what, where, and when?" within GCP resources.

~20 other services.

Cloud Data Loss Prevention (DLP) – *classify and redact sensitive data* by detecting PIIs with raw REST API or language specific client libraries (e.g. Python). DLP API's client

libraries are supported on Compute Engine, App Engine, Google Kubernetes Engine, and Cloud Functions, as well as for additional data sources, custom workloads, and applications on or off cloud. Part of the *Sensitive Data Protection* suite (see below).

Discover and protect sensitive data

Automated sensitive data discovery and classification - continuously *monitor your data* across your entire organization (select organization folders or individual projects). Powerful and easy-to-use UI. 150+ predefined detectors + add your custom types, adjust detection thresholds, create detection rules.

De-identification – *remove identifying information from data* => Detect sensitive data such as PII and then:

- **Mask** it by partially or fully replacing chars w/symbol, e.g. (*) or (#).
- **Delete** it.
- **Obscure** by a) replacing each PII w/token or string, b) encrypting sensitive data w/randomly generated or pre-defined key.

Special Case: Masking of AI/ML Workloads - *classify and de-identify specific sensitive elements within your data* to prepare data for AI model training or protect customer identifiers in chats, feedback, AI prompts, and generated responses to ensure you adhere to regulations and internal policies.

Cloud Management Tools

Cloud Deployment Manager – templated infra deployment by writing flexible templates and configuration files.

Cloud APIs – APIs for cloud services.

Cloud Billing API – programmatically manage GCP billing.

Cloud Billing – billing and cost management tools.

Cloud Mobile App – iOS/Android GCP manager app.

Cloud Console – helps you deploy, scale, and diagnose production issues in a simple *web-based interface*.

Operations & Monitoring

Cloud Monitoring - provides visibility into the *performance, uptime, and overall health of cloud-powered apps*. Collect *metrics*, events, and metadata for apps. Visualize this data on charts and dashboards; create *alerts* when metrics are out of range.

Cloud Logging – *centralized logging*: log ingestion and viewing, log-based metrics, log sinks and exports. **Log Analytics** - run queries to analyze and visualize log data.

Cloud Trace – *app latency insights*: distributed tracing system that collects *latency data* from your apps and displays it in the Google Cloud Console. Capture traces from all of your VMs, containers, or App Engine projects and provides *near real-time performance insights / bottlenecks*.

Cloud Profiler - statistical, low-overhead profiler that continuously *gathers CPU usage and memory-allocation information* from your production apps, attributes that info to app's *source code* to identify the parts of the app consuming the most resources => illuminating the *performance characteristics of the code*.

Error Reporting – app error reporting.

DevOps and Deployment

Cloud Build - CI/CD platform.

Cloud Deploy – deployment pipeline for GKE. Automates distribution of your apps to a set of target environments.

Cloud Source Repositories – hosted private git repos.

Artefact registry – global package manager.

Container registry – private container registry/repository

Google Cloud Deployment Manager – templated infra deployment (see above).

Infrastructure as Code (IaC) – see full chapter below.

API Platform & Ecosystem

API Gateway – fully managed API Gateway: *develop, secure, and monitor APIs*, giving you great speed and scalability.

API Analytics – API metrics.

AppSheet - no-code mobile and web app development.

Cloud IoT Core – manage devices, ingest data.

Developer Tools

Cloud SDK – CLI for GCP.

Cloud Shell – browser-based terminal/CLI.

Cloud Code – *cloud-native IDE extensions* / plugins: make it easier to design, deploy, and connect Google Cloud apps.

Cloud Code for VS Code – VS Code GCP tools.

Cloud Code for IntelliJ – IntelliJ GCP tools (Java and Kotlin).

Cloud Tools for Visual Studio – Visual Studio GCP tools - developers manage and deploy apps to GCP within the IDE.

Cloud Tools for Eclipse – Eclipse GCP tools (Java).

Migration to Google Cloud

Storage Transfer Service – online/on-prem data transfer, e.g. move email and files from your school Google account to another Google account.

Cloud Data Transfer – data migration tools/CLI to transfer data between object and file storage fast and securely across Google Cloud, Amazon, and other platforms.

Google Transfer Appliance – rentable physical data transfer box: send your data securely to Google upload facility, where the data is uploaded to cloud storage.

Migrate for Compute Engine – compute engine migration tools.

Migrate from Amazon Redshift – migrate from Redshift to BigQuery.

Application Integration

Pub/Sub – fully managed, global real-time *messaging* - send messages to specific topics at Google scale (millions per sec) across different apps using the real-time messaging.

Cloud Tasks – asynchronous task execution.

Cloud Workflows – HTTP services orchestration.

Cloud Scheduler – managed cron job service.

Eventarc – event-driven Cloud Run service.

Other groups of services: *Mobile (FireBase – see below)*, *Google Maps*, *Workspace Platform* (Sheets, Slides, Docs, Calendar, Gmail) *Healthcare*, *Retail*, *Gaming*, *Additional Resources* (see links in 4-word Developer Cheat Sheet).

Infrastructure as code (IaC) overview

Configure your infrastructure using code instead of graphical interfaces or command-line scripts:

- **Infrastructure Manager** - managed service automating deploy and management of GCP infra using Terraform; enables management of resources using IaC.
- **Terraform on Google Cloud** - tool to build, change, and version infra safely and efficiently (can be reused and shared) using high-level human-readable config syntax (partial JSON format).
- **Cloud Development Kit (CDK) for Terraform** - configure Terraform using a programming language to define and provision Google Cloud infra (TypeScript, Python, Java, C#, and Go).
- **Google Cloud provider for Pulumi** – configure infra code using programming languages TypeScript, Python, Java, C#, Go, or YAML. Looks like a **Python program** where you create and configure resources.
- **Ansible** - open-source automation IaC tool; allows to automate provision, configure management, application deployment, orchestration and other IT processes.

Recommendations for Infrastructure as Code

Recommender uses ML to provide resizing recommendations for VM instances for more efficient resource utilization and Identity and Access Management (IAM) recommendations to remove unnecessary access to GCP resources ("least privilege" principle).

Recommendations can be applied manually from the Google Cloud console or programmatically as part of your IaC pipeline. Some components:

- **Platform Intelligence Recommenders -** generate security and VM sizing recommendations
- **Cloud Scheduler** – a scheduled job runs the Recommender Parser service which calls the Recommender API to retrieve recommendations for your projects and parses the VM sizing and IAM recommendations; generates a pull request with the changes which you review and merge - a Cloud Build job rolls out the changes to your infra in your Google Cloud organization.
- **Cloud Run service =** recommender-parser service where all the processing logic resides.
- IaC tool can be Hashicorp Terraform or Deployment Manager.
- Google **Cloud Build** invoke Terraform commands automating the deployment of infra based on the changes in the IaC manifests.
- IaC repository uses **GitHub for source control** (integrated with Cloud Build). When commits are made to the master branch, a Cloud Build job is triggered to run a set of preconfigured tasks (generates SSH keys for access to your IaC repo + Personal Access Token to push commits to GitHub).

GCP vs. Firebase

Firebase is best for **quick mobile app development** with **limited server-side experience** with a particular emphasis on **ease of use** and streamlined development workflows. GCP is a comprehensive **cloud computing platform** and is better for **complex large-scale, enterprise applications**, big data needs, and control over infrastructure.

	Firebase	**GCP**
Focus	Mobile & web app development	Cloud computing
Ideal for	Quick mobile app creation with limited server-side experience and growing your user base	Complex enterprise applications, large-scale big data needs, and control over infrastructure
Features	Managed backend services, real-time features.	VMs, DBs, advanced analytics, ML, networking, global load balancers, Google Grade Security, robust data and analytics, big data service
Uses	Used for building a **new mobile app**, augmenting an existing app with **new functionality** and **growing an audience**.	Used for building software leveraging Google's core infrastructure, data analytics, and machine learning
Users	**Client-side app developers** (both web & mobile)	**Backend and server-side developers**

- A Firebase project is also a GCP project
- An existing GCP project can be configured to add Firebase services
- Firebase adds SDKs, tools, and configurations to some Google Cloud products: Cloud Storage, Cloud Firestore, and Cloud Functions.

Within Firebase, the three products work seamlessly with mobile apps by providing additional SDKs for mobile clients, additional tooling with the Firebase CLI, and a way to configure security rules to control access to data through the provided SDKs. Firebase offers additional IAM roles for some Firebase products that give other members of your team granular access to those products, without the risk of them making a dangerous change elsewhere in your project.