


# Essential GCP Infra: Core Services

## 2023 Ivan Vlad S.

### ▼ 1 IAM

- WHO can do WHAT on WHICH resource
- ▼ IAM objects
  - Organization, folders, projects, resources, roles, members
- ▼ IAM resource hierarchy
  - Organization > Folders > Projects > resources
- ▼ Organization node
  - root node for GCP resources
  - ▼ Creating and managing orgs
    - created when goog workspace or Cloud Identity account created a Google Cloud project
  - ▼ Organization roles
    - ▼ Organization admin: control over all cloud resources; useful for auditing
      - Define IAM policies, determine the structure of resource hierarchy, delegate responsibility over critical components: networking, billing, and resource hierarchy through IAM roles
    - project creator: controls project creation; control over who can create projects
- ▼ Roles
  - ▼ Basic: apply accross all GCP services in a project
    - offer fixed, coarse-grained levels of access
    - Owner > editor > viewer | billing admin
  - ▼ Predefined: apply to a particular GCP service in a project, collection of perms
    - offer more fine-grained perms on services
    - compute admin, network admin, storage admin
  - Custom: let you define a precise set of perms
- ▼ Members

- ▼ 5 types: Google accounts, service accounts, google groups, google workspace domains, and cloud identity domains
  - the "who" part of "who can do what on which resource"
- ▼ IAM policies are attached to resources
  - deny policies are made up of deny rules
  - IAM conditions enforce conditional ABAC for GCP resources
- ▼ Organization policies
  - > a configuration of restrictions
    - > defined by configuring a constraint with desired restrictions
    - > applied to the organization node, folder or projects
- ▼ Service Accounts
  - ▼ belongs to application instead of end-user
    - roles can be assigned to groups or users, be cautious when doing this
  - identified by email address
  - ▼ 3 types: user-created, built-in (google-managed), google APIs service account
    - ▼ default compute engine service account
      - automatically created per project with auto-generated name and email
      - > built-in = google managed, store keys
        - > user-created = user managed keys, goog only stored public portion of the key
  - ▼ Scopes determine whether authorization identity is authorized or not
    - can be changed after an instance is created
- ▼  Best Practices
  - ▼ Leverage and understand the resource hierarchy
    - > use projects to group resources that share the same trust boundary
    - > check the policy granted on each resource and make sure you understand the inheritance
    - > use "principle of least priv" when granting roles
    - > audit policies in Cloud Audit Logs: setiampolicy
    - > audit membership of groups used in policies
  - ▼ Grant roles to Google groups instead of individuals

- > update group membership instead of changing IAM policy
- > audit membership of groups used in policies
- > control the ownership of the Google group used in IAM policies

#### ▼ Service Accounts

- > be very careful granting serviceAccountUser role
- > give service account a display name that clearly identifies purpose
- > naming convention for service accounts
- > key rotation policies and methods
- > audit with serviceAccount.keys.list()method

#### ▼ Identity Aware Proxy (IAP)

- ▼ > enforce access control policies for apps and resources
  - identify-based access control
  - central authorization layer for apps accessed by HTTPS
- IAM policy is applied after authentication

### ▼ **2 Data Storage Services**

#### ▼ Object

##### ▼ Cloud Storage

- ▼ binary or object data: images, media serving, backups
  - Classes: standard (0) > nearline (30) > coldline (90) > archive (365)
  - key features: scalable to exabytes, time to first byte in milliseconds, very high availability across all storage classes, single API across storage classes

##### ▼ Features

- customer-supplied encryption key (CSEK)
- object lifecycle mgmt: auto delete or archive objects
- object versioning: maintain multiple versions of objects
- directory sync: syncs VM directory with a bucket
- object change notifications using Pub/Sub

- objects are immutable

##### ▼ Data import services

- Transfer appliance: rack, capture and then ship your data to google
- Storage transfer service: import online data (another bucket, an s3 bucket, or web source)

- offline media import: 3P provider uploads the data from physical media
- ▼ File
  - ▼ Filestore
    - NAS: latency sensitive workloads
      - > Managed file storage service for apps
    - ▼ Use cases
      - application migration, media rendering, electronic design automation (EDA), data analytics, genomics processing, web content mgmt
  - ▼ Relational
    - ▼ Cloud SQL
      - web frameworks: CMS, eComm
      - ▼ fully managed DB service (MySQLm, PostgreSQL, Microsoft SQL Server
        - patches and updates auto applied, you administer MySQL users, Cloud SQL supports many clients
      - ▼ Services
        - HA config, backup service, import/export, scaling: up (machine capacity) or out (read replicas)
    - ▼ Cloud Spanner
      - ▼ RDBMS + scale, HA, HTAP: user metadata, Ad/Fin/MarTech
        - ACID transactions
      - ▼ combines benefits of relational DB structure with non-relational horizontal scale
        - scale to petabytes, strong consistency, HA, used for financial and inventory apps
  - ▼ Non-relational
    - ▼ Firestore
      - hierarchical, mobile, web: user profiles, game state
    - ▼ NoSQL document DB
      - ACID transactions
      - next generation of datastore
      - scales DOWN well
    - ▼ Cloud Bigtable

- heavy read + write, events: AdTech, financial, IoT
- if you don't require transactional consistency
- scales UP well

#### ▼ Cache

- ▼ Memorystore
  - ▼ fully managed Redis service
    - in-memory data store service

#### ▼ Warehouse

- ▼ BigQuery
  - enterprise data warehouse: analytics, dashboards

### ▼ **3 Resource Management**

#### ▼ Resource Manager

- ▼ lets you hierarchically manage resources
  - child policies cannot restrict access granted at the parent level
  - Billing & resource monitoring:
    - > Org: contains all billing accounts
    - > Project: associated with one billing account
    - > Resource: belongs to one and only one project
  - Org node = root node for GCP resources
- ▼ Project accumulates the consumption of all its resources
  - Track resource & quota usage:
    - > enable billing, mng perms and creds, enable services and APIS
  - 3 identifying attributes:
    - > name, number, ID (aka app ID)
- ▼ Resources are global, regional, or zonal
  - > global: images, snapshots, networks
  - > regional: external IP
  - > zonal: instances, disks
  - \*regardless of type, each resource is organized into a project

#### ▼ Quotas

- ▼ all resources are subject to project quotas or limits

- > 15 VPC networks/project
- > rate limits, 5 admin actions/second in Cloud Spanner
- > 24 CPUs region/project
- \*can increase quote in cloud console quota page or support ticket

#### ▼ why use quotas?

- > prevent runaway consumption in case of error or attacks, prevent billing spikes or surprises, forces signing consideration and periodic review

#### ▼ Labels

##### ▼ utility for organizing GCP resources

- attached to resources: VM, disk, snapshot, image through console, gcloud, or API
- example uses: inventory, filter resources, in scripts (help analyze cost, run bulk ops)

##### ▼ use labels for

- > team or cost center
- > components
- > owner or contact
- > state
- > environment or stage

##### ▼ labels =/ tags

- ★ labels organize resources across GCP, tags are applied to instances only (primarily used for networking - applying firewall rules)

#### ▼ Billing

- set budget email alerts
- ▼ label all resources and export to bigquery to analyze spend
  - can visualize spend over time with Looker Studio > billing dashboard

### ▼ 4 Resource Monitoring

#### ▼ GCP Ops Suite (formerly Stackdriver)

- ▼ integrated monitoring, logging, diagnostics
  - > pay for what you use
- ▼ manages across platforms:
  - > GCP and AWS
  - > dynamic discovery of GCP with smart defaults
  - > open-source agents and integration

- Multiple integrated products:
    - > monitoring
    - > logging
    - > error reporting
    - > trace
    - > profiler
  - access to powerful data and analytics tools
  - collab with 3P software
- ▼ Monitoring
- ▼ monitoring is the base of SRE
    - Monitoring > IR > root cause analysis > testing + release procedures > capacity planning > development > product
    - dynamic config and intelligent defaults
  - ▼ platform, system and apps metrics
    - > ingest data
    - > generates insight through dashboards, charts, alerts
    - > can create custom metrics
  - ▼ Metric Scope is the root entity that holds monitoring and config information
    - metric scope is a single pane of glass
      - > can view resources from multiple GCP projects and AWS accounts
  - ▼ dashboards visualize utilization and network traffic
    - alerting policies can notify you of certain conditions
  - ▼ uptime/health checks
    - uptime checks test the availability of your public services
- ▼ Logging
- ▼ collects platform, systems, and app logs
    - > API to write logs
    - > 30-day retention
      - data can be exported to Cloud Storage, BigQuery and Pub/Sub
    - analyze in BQ and visualize in Looker Studio
- ▼ Error Reporting
- ▼ aggregate and display errors for running cloud services
    - error notifications

- error dashboard
- App engine, app script, compute engine, cloud functions, cloud run, GKE, Amazon EC2
- ▼ Tracing
  - ▼ Tracing system
    - displays data in near real-time
    - latency reporting
    - per-URL latency sampling
  - ▼ collects latency data
    - app engine
    - google HTTP(S) load balancers
    - apps instrumented with the cloud trace SDKs
- ▼ Profiling
  - continuously analyze the performance of CPU or memory-intensive functions executed across an app
  - uses statistical techniques and extremely low-impact instrumentation
  - runs across all production instances