Course layout

Tuesday, August 25, 2020 3:52 PM

- Section 1: Setting up a cloud solution environment
- Section 2: Planning and configuring a cloud solution
- Section 3: Deploying and implementing a cloud solution
- Section 4: Ensuring successful operation of a cloud solution
- Section 5: Configuring access and security

§1: Setting up a cloud solution environment

- 1.1 Setting up cloud projects and accounts
- 1.2 Managing billing configuration
- 1.3 Installing and configuring the command line interface (CLI)

§2: Planning and configuring a cloud solution

- 2.1 Planning and estimating GCP product use using the Pricing Calculator
- 2.2 Planning and configuring compute resources
- 2.3 Planning and configuring data storage options
- 2.4 Planning and configuring network resources

§3: Deploying and implementing a cloud solution

A CLOUD GUR

- 3.1 Deploying and implementing Compute Engine resources
- 3.2 Deploying and implementing Kubernetes Engine resources
- 3.3 Deploying and implementing App Engine and Cloud Functions resources
- 3.4 Deploying and implementing data solutions
- 3.5 Deploying and implementing networking resources
- 3.6 Deploying a Solution using Cloud Launcher
- 3.7 Deploying an Application using Deployment Manager

§4: Ensuring successful operation of a cloud solution

- 4.1 Managing Compute Engine resources
- 4.2 Managing Kubernetes Engine resources
- 4.3 Managing App Engine resources
- 4.4 Managing data solutions
- 4.5 Managing networking resources
- 4.6 Monitoring and logging

§5: Configuring access and security

- 5.1 Managing Identity and Access Management (IAM)
- 5.2 Managing service accounts
- 5.3 Viewing audit logs for project and managed services

Different services breadth

Monday, February 1, 2021 10:49 PM

- **Compute**
  a. Compute Engine (GCE):
     i. Is a Zonal service
     ii. Provides fast booting VMs on demand for rent
     iii. Normal instances are expensive but become cheaper with the 'sustained use discount' if used long enough
     iv. Pre-emptive instances are even cheaper, but can be deleted at any time by Google
  b. Kubernetes Engine (GKE):
     i. Used to be called Google cloud engine
     ii. Provides container orchestration with autoscaling
     iii. Is a Regional resource
     iv. Doesn't integrate with GCP IAM
  c. App engine (GAE):
     i. Platform as a Service
     ii. Is a Regional resource
     iii. Auto scales based on load
  d. Cloud functions (GCF):
     i. Runs code in response to an event in :Node.js, python, java, go
     ii. Function as a Service (or serverless)
     iii. Is a Regional resource
     iv. Pay for CPU and RAM assigned to a function per 100ms (minimum)
     v. Each function automatically gets an HTTP endpoint
     vi. Can be triggered by GCS, PUB/SUB, etc
     vii. Auto horizontal scaling-Runs as many copies as needed to handle the load
     viii. Use cases are typical apps who's usage can vary depending on requirement ex: Chat boxes, message processors, Iot, automation etc
- **Storage**
  a. Local SSD:
     i. Is Zonal resource I.e attached to any instance
     ii. Ephemeral in general
     iii. Pay for per GB
  b. Persistent disk:
     i. Is Zonal resource but can be regional too in certain cases
     ii. Flexible, block based, network attached storage
     iii. Can be resized while still in use
     iv. Can be shared among different VMs, but only in read only mode
  c. Cloud filestore:
     i. Fully managed file based storage
     ii. Comparable to a NAS
     iii. Is zonal is reachable to every instance in that zone's VPC
     iv. Primary used for application migration to cloud
  d. Google cloud storage:
     i. Can be regional or multi-regional
     ii. Versioning can also be done wherein GCS stores different versions of the same object
     iii. Has integrated site hosting and CDN functionality too

iv. Lifecycle includes: multi-regional, regional, nearline, coldline
v. Pay for data operations and GB stored
vi. Nearline and coldline: Also pay for GBs retrieved, plus early deletion fee if <30/90 days respectively

- **Databases**
  a. Cloud SQL:
    i. Reliable mysql, sql server and postgres db
    ii. Is a Regional resource
  b. Cloud spanner:
    i. Horizontally scalable, strongly consistent, relational db
    ii. Scales from 1 to 1000 nodes
    iii. Regional, Multi-regional and Global resource
    iv. Used mostly for big production uses and cost can go to $1000s with more nodes
  c. BigQuery:
    i. Serverless column store data warehouse for analytics using SQL
    ii. Scales internally to handle petabyte scale data
    iii. Caches results for 24hrs for free, if needs to be re-used
    iv. Gets cheaper if the table is not modified for 90days, with read only operations
    v. Can streaming inserts from streaming data as well
  d. BigTable:
    i. Low latency and high throughput NoSQL db for large operational analytics apps
    ii. Supports open source HBase api
    iii. Is zonal resource
    iv. Scales seamlessly and unlimitedly
    v. Made for huge workloads
    vi. Can be cheaper if using hdd instead of ssd
  e. Datastore:
    i. Managed and autoscaled NoSql with indexed, queries and acid transactions
    ii. Has some in-built indexes and supports manual custom indexes
    iii. Regional and multi-regional resource
    iv. Pay for gb of data used, and data IOs done on the db
  f. Firebase realtime db and cloud firestore:
    i. Firebase is Zonal resource. Firestore is Multi-regional resource
    ii. NoSql document store with ~real time client updates via managed web sockets (which can complicated in AWS)
    iii. Firestore has collections, documents and contained data. It revolves around 1 huge DB is central america, and its data revolves around it
- **Data transfer**
  a. Data transfer appliance:
    i. Rackable, high-capacity server to physically ship data to GCS
    ii. 100TB/400TB viz faster than network transfer
    iii. Is global resource
  b. Storage transfer service:
    i. Copies objects so you need to setup machines
    ii. Source could be anything-S3, http endpoints, any other GCS bucket. Destination is always a GCS bucket
    iii. Free to use but have to pay for actions
    iv. Is global resource
- **External networking**
  a. Google domains
    i. Google's registrar for getting domain names

ii. Global resource. Gives private whois records. Supports DNSSEC

b. Cloud DNS
   i. Scalable, reliable, managed DNS service
   ii. Build and use custom nameserver service
   iii. 100% uptime guarantee. Mostly for private and public managed zones-> so its only accessible for a particular VPC if required or split to internet.
   iv. Pay fixed fee per zone or to distribute DNS around world.

c. Static IP:
   i. Reserve static ip addr to assign to resources
   ii. Regional IPs used for GCE and network balancers
   iii. DNS is preferred over static
   iv. Pay if hoard static ip after a while

d. Load balancer
   i. High perf, scalable traffic distributed integrated with autoscaling and cloud cdn
   ii. Build on andromeda sdn. Handles spikes without pre-warming.
   iii. No need to build instances or devices
   iv. Can be regional or global
   v. Pay by making ingress traffic billable

e. Cloud CDN
   i. Low latency content delivery based on HTTP(s) CLB integrated with GCE and GCS
   ii. Supports HTTP/2 and HTTPS but no custom origins, only supports GCP
   iii. Simple to integrate with load balancer. Caches results to improve latency
   iv. Pay per http(s) request volume. You also pay if you to have cache invalidation request, to handle updating caches automatically. Hence good cache strategy can help

- **<u>Internal Networking</u>**

a. VPC
   i. Global IPv4 SDN for GCP
   ii. Automatic mode is easy, custom gives control
   iii. VPC is global but subnets are global

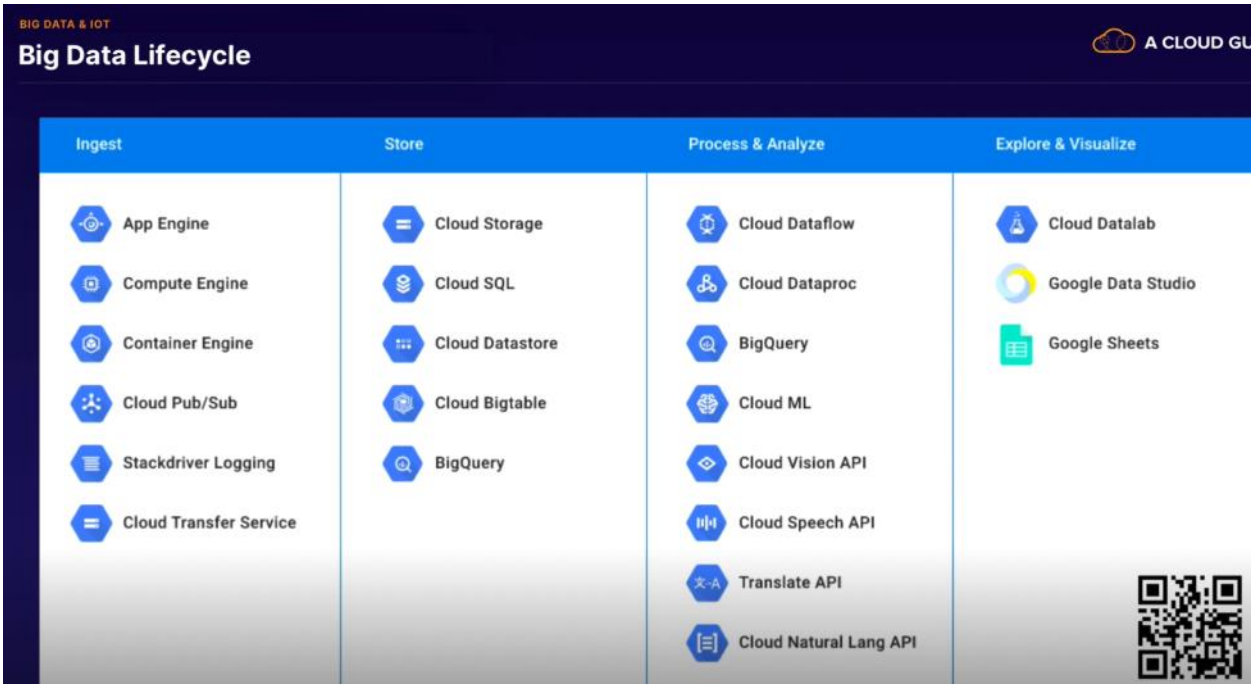
b. Cloud Interconnect
   i. Connect external networks to google's
   ii. Private connection vpc via vpn or dedicated/partner interconnect. It also gives some SLAs
   iii. Dedicated interconnect doesn't encrypt the data as opposed to vpn
   iv. Public google services accessible via External peering (no SLAs!)
      1. Direct peering for high volume
      2. Carrier peering via a partner for lower volume
   v. Significantly low ingress fee

c. Cloud VPN
   - Ipsec vpn using public internet for low-data volume
   - For persistent static connections
   - Pay per hour for tunnel usage

d. Dedicated interconnect
   i. Direct physical link from vpc to on-prem
   ii. VLAN is private connection to vpc in 1 region, no public gcp apis
   iii. Private links but not encrypted, have to add your own
   iv. Pay for per 10Gbps link vlan
   v. You however pay reduced egress rates through this

e. Cloud Router
   i. Dynamic routing for hybrid linking vpc to external networks

    ii. Free to setup but vpc egress charges
  f. CDN Interconnect
    i. Connect vpc to external CDNs not GCP's cdn
    ii. Free to enable but pay for egress
- **Machine learning**
  a. Cloud ML engine
    i. ML models and making predictions
    ii. Based on tensorflow
    iii. Enables apps/dev to use tensorflow on any size of dataset
    iv. Integrates with gcs, datalab, dataflow
    v. HyperTune automatically tunes models to avoid manual tweaking
    vi. Pay per hour for training
    vii. Pay per provisioned node-hour plus prediction request volume made
  b. Cloud vision
    i. Image processing
    ii. Already pre-trained model
    iii. Pay per image, based on detection features requested
  c. Cloud speech
    i. Auto speech recog. with pre-trained models for 110 languages
    ii. Pay per 15sec audio that is processed
  d. CloudNatural language
    i. Analyze text for sentiment, intent, content classification and extracts info
    ii. Good to use with speech api, vision and translation
    iii. cost depends on what analysis type and is charged per 1000 characters
  e. Cloud translation
    i. Translate semantics, auto detect source language
    ii. Helps support non-native language
    iii. Cost is per character processed
  f. Diagflow
    i. Build conversational interfaces for websites, mobile apps, messaging apps , iot
    ii. Has pre-trained model and service for accepting, parsing, lexing input and responding
    iii. Add custom code to handle chat bots or use pre-built agents
    iv. Free plan has unlimited chat but limited voice responses. Paid is premium and has non limits
  g. Cloud video intelligence
    i. Annotates videos in gcs with info about what they contain ex. Offensive content
    ii. Pay per minute video is processed
  h. Cloud job discovery
    i. Helps career sites, company job boards, to improve engagement and conversion
    ii. Integrates with job/hiring systems
- **Big data and IoT**



a. Cloud IoT core:
    i. Managed service to connect, manage, and ingest data from devices globally
    ii. Device manager handles device identity, auth, config or control
    iii. Connect securely using MQTT or HTTPS
    iv. Can publish telemetry to cloud pub/sub for processing
    v. 2-way device communication handles configuration and updates
    vi. Pay per MB of data exchanged
b. Cloud Pub/Sub
    i. Infinitely-scalable messaging for ingestion, decoupling
    ii. Allows subscribers to read based on Topics
    iii. Is global resource, and subscribers can be load balanced worldwide
    iv. Push mode delivers to https endpoints and succeeds on http success status code
        1. Slow-start algo ramps up on success and back off & retries, on failures
    v. Pull mode delivers messages to requesting clients and waits for ack to delete
        1. lets client set rate of consumption and supports batching and long-polling
c. Cloud data prep
    i. Visually explore, clean and prepare data for analysis without running servers
    ii. Data wrangling for business analytics not IT pros
    iii. managed version of trifacta wrangler and manager by it not Google
    iv. Source data from GCS, BQ or file upload-formatted in json,csv
    v. Pay for underlying dataflow job, plus management overhead
d. Data proc
    i. Batch map reduce processing via configurable, managed spark and hadoop cluster
    ii. Handles scale even while running jobs
    iii. Switches between versions of spark, hadoop and others
    iv. Best for moving existing spark/hadoop setup to gcp, but you should use Data flow for processing new data pipelines
e. Data flow
    i. Smartly autoscaled and fully managed batch or stream map reduce like processing
    ii. Autoscales and dynamically redistributes lagging work, mid-job to optimize run

time
iii. Data flow shuffle service for batch offload. Shuffle ops from workers for big gains
iv. Pay for per second vCPUs, ram, ssd and is extra for data flow shuffle mode

f. Datalab
i. Tool for data exploration, analysis, visualization and ML
ii. Similar to jupyter notebooks
iii. Pay for gce, gae instance storing your notebook

g. Data studio
i. Big data visualization tool for dashboards
ii. No charge for this

h. Genomics
i. Store and process genomes
ii. Query complete genomic info or large projects in secs
iii. Requester pays sharing model for pricing

- **Identity and access (Core Security)**
  a. Roles
     i. Collections or permissions to use or manage gcp resources
  b. IAM
     i. Controls access to gcp resources: authorization but not authentication really
     ii. Member is user, group, domain, service account, or the public (I.e all users)
     iii. Policies bind members to roles at hierarchy level
  c. Service accounts
     i. Account that represents apps not users
     ii. Always use service accounts rather than user accounts or api keys, for most development work
     iii. Use cloud platform managed keys for most gcp resource usage
  d. Identity
     i. Gsuite or gcloud type service
  e. Security key enforcement
     i. Usb or bluetooth 2-step verification
  f. Resource manager
     i. Centrally manage and secure organization's project
     ii. Is root node in hierarchy
  g. Cloud IAP
     i. Guards apps running on gcp via identity verification, not VPN access
     ii. Based on clb and iam
     iii. Grant access to any iam, group or service accounts
  h. Cloud audit logging
     i. Maintains non-tamperable audit logs for each project and organization
     ii. Admin activity-400 days retention
     iii. Access transperancy-400 day
        1. shows actions by google staff
     iv. data access 30 day retention
        1. for gcp visible services
- **Monitoring and responding**
  a. Cloud armor
     i. Edge level protection from DDoS and other on global https load balancer
     ii. Manage IPs with cidr based allow/block lists
     iii. You can preview effect of changes before turning on
     iv. Pay monthly charge per policy and rule
  b. Security scanner

i. Free but limited GAE app vulnerability scanner
ii. Can identify cross-site-scripting, flash injection, mixed content, outdate libraries

c. Cloud DLP
i. Finds and optionally redacts sensitive info in unstructured data streams
ii. Helps minimize what you collect, expose or copy to other systems
iii. 50+ sensitive data collectors including card number, ssn, paspport# etc
iv. Can scan text and images
v. Pay for amount of data processed per GB and get cheaper with volume

d. Event threat detection
i. Automatically scans stackdriver logs for suspicious activity
ii. Uses threat intelligence including safe browsing
iii. Can detect malware, cryptominig, outgoing DDoS attacks, brute-force ssh
iv. Can monitor access to gcp resources via abusive IAM access, or look for patterns out of the normal
v. Integrates with Stackdriver, BQ, Pub/sub, etc for more analysis

e. Cloud SCC
i. Security info and event mgmt (SIEM)
ii. Can integrate with other vendor SIEM products
iii. Helps you prevent, detect, and respond to threats from a single pane of glass
iv. Integrates with ETD, scanner, DLP
v. Free to use. Can alert when needed

- **Encryption key mgmt**
  a. Cloud KMS
  i. Manage and use crypto keys
  ii. Supports symmetric (AES) and asymm (RSA)
  iii. Move secrets out of code and into env
  iv. Integrated with IAM and Audit logging
  v. Rotate keys for new encryption either automatically or on demand
  vi. Keeps older keys to decrypt older keys'

  b. Cloud HSM
  i. Cloud KMS keys managed by FIPS 140-2
  ii. Device hosts encryption leus and performs crypto ops
  iii. KMS uses HSM to store keys
  iv. Same apis and features as KMS

- **Operations and Management Services-Stackdriver**
  a. Stackdriver
  i. Family of services for monitoring, logging and diagnosing apps
  ii. Can work on GCP, AWS, Hybrid etc
  iii. Simple usage based pricing

  b. SD Logging
  i. Store, search, analyze, monitor and alert on log data and events
  ii. Debug issues via integration with SD monitoring, trace and error reporting
  iii. Create real time metrics from log data

  c. SD Error reporting
  i. Counts, analyzes, aggregates errors and crashes
  ii. Groups errors in groups automatically
  iii. Links notifications to errors and gives time charts, occurrences, affects user counts etc
  iv. Exception stack trace parser knows java, python, js, ruby, c#, php, go

  d. SD Trace
  i. Tracks and displays tree and timings across distributed systems

ii. Automatically captures from GCE
iii. Detects app latency shift (degradation) over time by evaluating perf results

e. SD Debugger
   i. Grabs program state in live deploys with low deploy
   ii. Logpoints repeat for upto 24hr, fuller snapshots run once but can be conditional
   iii. Source view supports cloud repos, but you can also upload
   iv. Supports java and python on gce, gae,gke. Go only on gcp,gke
   v. Auto enabled for gce, but agents can be installed
   vi. Is free to use

f. SD Profiler
   i. Continuous cpu and memory profiling to improve perf and reduce cost
   ii. Low overhead Max<5% typical overhead
   iii. Is Agent based, which need to be installed

g. Deployment manager
   i. Create/manage resources via declarative templates
   ii. Supports yaml, python, jinja2
   iii. Supports input/output parameters with json schema

h. Cloud billing api
   i. Programmatically manage billing for projects and pricing
   ii. List billing accs, enable/disable accs, list billable SKUs, get public pricing
   iii. Wont show current bill. You would have to export it manually.

- **Development and APIs**

a. Cloud source repositories
   i. Similar to github
   ii. No enhances support like PULL requests like github
   iii. Integrates with SD debugger
   iv. Pay per month per usage per user

b. Cloud Build
   i. Takes source code to build, test and deploy it-CI/CD service
   ii. Similar to Jenkis
   iii. Free and runs in parallel (upto 10 at a time)
   iv. Plus scans for package vulnerabilities
   v. Docker: simple build and push functions
   vi. Json and Yaml supported
   vii. Pay per min of build time->after you go over 120mins in a day

c. Container Registry
   i. Fast, private and docker image storage with docker v2 api
   ii. Creates and manages multi-regional gcs bucket, then translates gcr calls to gcs
   iii. Integrates with cloud build and sd logs

d. Cloud Endpoints
   i. Handles authorization, monitoring, logging and API keys like nginx
   ii. Proxy instances are distributed and hook into cloud load balancer
   iii. Super fast <1ms
   iv. Handles auth and logging
   v. Handles gRPC and can transcode HTTP
   vi. Pay per call to your API

e. Apigee
   i. Enterprise API management platform for whole API lifecycle
   ii. Transform calls between protocols SOAP, REST, XMS, Binary
   iii. Authenticates via OAuth or role based
   iv. Throttle traffic with quotas, manage API versions etc

v. Enterprise grade and bit expensive

f. Test lab for Android
    i. Cloud infra for running test matrix across REAL android devices
    ii. Production grade devices flashed with real Android versions
    iii. Automatic, can add custom scripts, monitor progress

GCP commands (highlighted are only in gsutil cli not gui)

Thursday, December 12, 2019 8:36 PM

https://github.com/dennyzhang/cheatsheet-gcp-A4/blob/master/cheatsheet-gcp-A4.pdf gcloud cheat sheet
https://cloud.google.com/sdk/gcloud/reference : Reference

- gcloud init performs the following setup steps:
  - Authorizes Cloud SDK tools to use your user account credentials to access Google Cloud, or lets you select an account if you have previously authorized access
  - Sets up a Cloud SDK configuration and sets a base set of properties, including the active account from the step above, the current project, and if applicable, the default Google Compute Engine region and zone
- You can run the following as alternatives to gcloud init:

Command	Description
gcloud auth login	Authorize with a user account without setting up a configuration.
gcloud auth activate-service-account	Authorize with a service account instead of a user account. Useful for authorizing non-interactively and without a web browser.
gcloud config [COMMAND] gcloud config configurations [COMMAND]	Create and manage Cloud SDK configurations and properties.

- When you install the SDK, the following components are installed by default: [ref]

ID	Name	Description
gcloud	Default gcloud CLI Commands	Tool for interacting with Google Cloud. Only commands at the General Availability and Preview release levels are installed with this component. You must separately install the gcloud alpha Commands and/or gcloud beta Commands components if you want to use commands at other release levels.
bq	BigQuery Command-Line Tool	Tool for working with data in Google BigQuery
gsutil	Cloud Storage Command-Line Tool	Tool for performing tasks related to Google Cloud Storage.
core	Cloud SDK Core Libraries	Libraries used internally by the SDK tools.

GCLOUD Commands:

Basic syntax for gcloud follows: gcloud <global flags> <service/product> <group/area> <command> <flags> <parameters>

Global flags: For all commands in the gcloud CLI, users can specify a number of global flags to modify command behavior. The --account <account>, --project <project id>, --billing-project <billing acc> and --configuration flags allow you to override the current defaults for your environment. These are useful to avoid modifying or switching configurations to run a few quick commands. Other global flags including --quiet, --flatten, --format (ex: json,yaml,csv etc), and --verbosity allow users to modify the output from running commands—often useful when running scripts or debugging operations.

1) Gcloud config list: what account and what project currently in
   a. Gcloud config get-value project: List project name
   b. Gcloud auth list: Show **active** account details too
2) Gcloud compute instances create <vm name>: create vm instance. Either specify zone or select from cli prompts
   a. Gcloud compute instances delete <vm name>: delete vm instance
3) gcloud services list: List enable service APIs

a. gcloud services enable <api name > : Enable api

4) gcloud compute ssh --project=<PROJECT_ID> --zone=<ZONE VM_NAME> : SSH into vm from gcloud
5) gcloud config set compute/region REGION : Set region for gcloud commands
6) gcloud compute regions list: A list of regions can be fetched
7) gcloud config unset compute/region: To unset the property
8) gcloud compute project-info describe : gcloud compute project-info describe displays all data associated with the Compute Engine project resource. The project resource contains data such as global quotas, common instance metadata, and the project's creation time.

- gcloud config set : sets the specified property in your active configuration only. A property governs the behavior of a specific aspect of Cloud SDK such as the service account to use or the verbosity level of logs. To set the property across all configurations, use the `--installation` flag. For more information regarding creating and using
- Cloud Shell provisions 5 GB of free persistent disk storage mounted as your $HOMEdirectory on the virtual machine instance. This storage is on a per-user basis and is available across projects. Unlike the instance itself, this storage does not time out on inactivity. All files you store in your home directory, including installed software, scripts and user configuration files like .bashrc and .vimrc, persist between sessions. Your $HOME directory is private to you and cannot be accessed by other users.

- Most gcloud commands follow the following format:

  gcloud + release level (optional) + component + entity + operation + positional args + flags

  - For example: gcloud + compute + instances + create + example-instance-1 + --zone=us-central1-a
  - Release level: Release Level refers to the command's release status.

    Example: alpha for alpha commands, beta for beta commands, no release level needed for GA commands.
  - Component: Component refers to the different Google Cloud services.
    - Example: compute for Compute Engine, app for App Engine, etc.
  - Entity: Entity refers to the plural form of an element or collection of elements under a component.
    - Example: disks, firewalls, images, instances, regions, zones for compute
  - Operation: Operation refers to the imperative verb form of the operation to be performed on the entity.
    - Example: Common operations are describe, list, create/update, delete/clear, import, export, copy, remove, add, reset, restart, restore, run, and deploy.
  - Positional args: Positional args refer to the required, order-specific arguments needed to execute the command.
    - Example: <INSTANCE_NAMES> is the required positional argument for gcloud compute instances create.
  - Flags: Flags refer to the additional arguments, --flag-name(=value), passed in to the command after positional args.
    - Example: --machine-type=<MACHINE_TYPE> and --preemptible are optional flags for gcloud compute instances create.

Gcloud help compute instances create : To get help filling out command flags

GSUTIL Commands:

gsutil uses the prefix gs:// to indicate a resource in Cloud Storage:

Ex: gs://BUCKET_NAME/OBJECT_NAME

1) Gsutil -> used to connect to gcp storage
2) Gsutil mb -l northamerica-northeast1 gs://<bucket name>: To create bucket in northeast region
3) Gsutil label get gs://<storage name>: Get bucket labels in json format

    a. Gsutil label get gs://<storage name> > test.json : Save bucket labels to json file
    b. Gsutil label set test.json gs://<storage name>: Set labels for bucket from json file
4) Gsutil label ch -l "extralabel:extravalue" <bucket storage name>: Change label. While you can do a 'get' to pull the bucket labels in json format, edit it and set it back to the bucket with 'set', changing/adding individual labels is better this way.
5) Gsutil versioning get <bucket name>: get if versioning is enabled or not
    a. Gsutil versioning set on <bucket name>: enable versioning. This can be done only through the cli or api. Object versioning will make sure that the bucket doesn't lose any objects due to running the delete command or if we add a new version of the object
6) Gsutil ls -a <bucket name>: list all files and archive files too
7) Only way to delete object is to remove it by name
8) Gsutil cp <bucket name>/** <dest>: copy all files. Use -r to copy AND maintain file structure
9) Gsutil acl ch -u AllUsers:R <file location inside bucket>: Make objects available to all users in public internet. Ch says change, -u says users, :R says Read only access.

In general functionality => rest api>cli>console

Gcloud Configurations:

Configurations are a group of settings or properties. 'Gcloud config init' initializes the initial base configuration of a project and is helpful across multiple projects. You can create configuration commands using 'gcloud config create' and to set common properties (this is different from gcloud config configurations list, which shows all configurations). But you need to activate the configuration to switch to it. 'Gcloud config list' will list config properties of the current config. IS_ACTIVE column in the configuration list shows which configuration is currently being used.

1) Gcloud config configurations create <name>: Create a gcloud config
    a. Gcloud config configurations activate <name>: Activate that configuration
2) Gcloud --configuration=<newconfig> config list: Show configs of another configuration file
3) gcloud config set compute/zone us-west2-b : set default region for creating instances
4) gcloud compute ssh <vm name>: Connect securely directly with secure keys to your vm
5) Settings stored by gcloud cli tool is DIFFERENT than that used by the gcp console. Once created, you cannot change name and region for a vm. You can stop it and change its machine-type. Non-Preemptibility allows google to maintain vm instance UP at all times. By turning it ON such systems can be turned OFF at any time by google due to system issues or requirements.. By allowing service account access to the vm, you can delete the vm from within itself. This action is done by the service account holder on the vm.

Stackdriver, Operations and mgmt

Tuesday, January 14, 2020 11:00 PM

Stackdriver agents can be used for monitoring and logging and diagnosing on gcp or even aws.

The stackdriver agent pushes logs to stackdriver. To authenticate the transactions, the SD agent needs to use an authentication token that the SD console can accept.
Stackdriver never pulls logs from instances, instances push data to stackdriver. The **host** project stores Stackdriver's metadata.

Can be used for:
1) Uptime monitoring
2) Performance metrics
3) Alerting: Runs conditions I.e tests to keep check. Can also send documentations along with the notifications.
4) AWS Monitoring too
5) Debug: Stack driver debug can debug a stateful application without affecting it.
6) Trace: Can trace issues across all services ex: latency issues on any instance
7) Logging: Pulls and aggregate all logs
8) Error: Pulls error from the logs pulled

Stackdriver always requires a host project to run on. If you use a single project you can make it the host project. However if you want to run stackdriver on multiple projects, you should create an empty project just to run stackdriver.

You can export logs by creating one or more sinks that include a logs query and an export destination. Supported destinations for exported log entries are Cloud Storage, BigQuery, and Pub/Sub.

1 SD account can track multiple accounts across gcp,aws etc

- **SD logging (Google Cloud logging):**
  - Store, search, analyze and monitor/alert on log data and events. Pay per gb with 1st 50gb is free
  - Requires installing a logging agent based on fluentd, to stream logs from common third-party applications and system software to Logging.
  - The following VM instances support Logging using their own software, so manually installing the Logging agent on them is not supported:
    - App Engine standard environment instances. App Engine includes built-in support for Logging.
    - App Engine flexible environment instances. Apps running in the App Engine flexible environment can write logs that are in addition to what is included in the App Engine standard environment.
    - Google Kubernetes Engine node instances. You can enable Cloud Operations for GKE, an integrated monitoring and logging solution, for your new or existing container clusters.
    - For instances running on Anthos clusters on VMware, the agent collects system logs but doesn't collect application logs.
    - Cloud Run container instances. Cloud Run includes built-in support for Logging.
    - Cloud Functions HTTP and background functions. Cloud Functions includes built-in support for Logging
  - You can install agents on VMs using console by Monitoring->Select Workspace->Dashboards->VM Instances->Inventory tab->Not detected VMs->Install agent
    - Else you can install it manually on each VM by getting the source code repo and installing it
  - To give custom retention values to logs stored in buckets:
    - `gcloud logging buckets update _Default --location=global --retention-days=[RETENTION_DAYS]`

- You can refine the scope of the logs displayed in the Logs Explorer through the **Refine scope** panel. You have the option to only search logs within the current project or to search logs based on one or more storage views.
- Logs-based metrics are Cloud Monitoring metrics that are based on the content of log entries. For example, the metrics can record the number of log entries containing particular messages, or they can extract latency information reported in log entries. You can use logs-based metrics in Cloud Monitoring charts and alerting policies. Logs-based metrics are calculated from both included and excluded logs.
- Logs-based metrics apply only to a single Google Cloud project. You cannot create them for logs buckets or for other Google Cloud resources such as Cloud Billing accounts or organizations.
- System (logs-based) metrics are predefined by Logging. These metrics record the number of logging events that occurred within a specific time period. These metrics are free of charge and available to all Google Cloud projects.
- User-defined (logs-based) metrics are created by a user on a Google Cloud project. They count the number of log entries that match a given filter, or keep track of particular values within the matching log entries. When you create a filter for the log entries that you want to count in your metric, you can use regular expressions.
- Metric types include: Counter (All system based are. Counts number of entries matching your filter) and Distribution (gives count, mean, std dev)

- **SD monitoring (now Google Cloud Monitoring):**
  - Performance, uptime and health of cloud apps. Can log and alert
  - By default monitors: CPU utilization, basic disk traffic information, uptime info and network traffic. Additional metrics require the collectd based daemon agent to be installed.
  - Cloud Monitoring uses Workspaces to organize and manage its information. A Workspace can manage the monitoring data for a single Google Cloud project, or it can manage the data for multiple Google Cloud projects and AWS accounts. However, a Google Cloud project or an AWS account can only be associated with one Workspace at a time.
  - After installing the Monitoring agent, you can monitor supported third-party applications by adding application-specific collectd configurations.
  - The following services have the agent pre-installed: App engine, Dataflow, Dataproc, GKE uses cloud operations for GKE.
  - Edit the Monitoring agent configuration file /etc/stackdriver/collectd.conf and then restart it (to make any changes).
  - Custom metrics. Collectd metrics that have the metadata key stackdriver_metric_type and a single data source are handled as custom metrics and sent to Monitoring using the projects.timeSeries.create method in the Monitoring API.
- **Sd error reporting (Cloud Error Reporting):**
  - Counts, analyzes, aggregates, and tracks crashes in a central interface. Alerts when new error cannot be grouped with existing ones. Can understand java,python,go,c#,php and ruby
  - You can report errors from your application by sending them directly to Cloud Logging with proper formatting or by calling an Error Reporting API endpoint that sends them for you.
  - You need the google-fluentd library to call logger api from your code.
  - Modify your application so that it logs exceptions and their stack traces to Logging. Choose a log name for your error information to keep it separate from other logged information.
  - Error logging is automatically available for app engine, cloud functions and cloud run. GKE needs agent installed.
- **SD trace (or Cloud trace):**
  - Tracks and displays call tree and timings across systems. Can capture traces from apps on java, nodejs, ruby and go. Can detect latency issues over time by evaluating performance reports on apps.
  - Cloud Trace provides distributed tracing data for your applications.
  - Trace allows us to analyze calls made from any source ex: http requests from anywhere, can be better

analyzed for latency using trace.

- After instrumenting your application, you can inspect latency data for a single request and view the aggregate latency for an entire application in the Cloud Trace console.
- Cloud Trace recommends using OpenTelemetry. OpenTelemetry is an open-source product from the merger between OpenCensus and OpenTracing.
- For your application to submit traces to Cloud Trace, it must be instrumented. You can instrument your code by using the Google client libraries (except for app engine):
- Use OpenTelemetry and the associated Cloud Trace client library. This is the recommended way to instrument your applications.
- Use OpenCensus if an OpenTelemetry client library is not available for your language.
- Use the Cloud Trace API and write custom methods to send tracing data to Cloud Trace.
- Cloud Trace doesn't sample every request, this is customizable. However you can force it to trace a curl you make, by adding the header "X-Cloud-Trace-Context:TRACE_ID/SPAN_ID;o=TRACE_TRUE" to your curl command

- **SD debugger (now Cloud debugger):**
  - Grabs program state in live deploys. Can be used to debug issues on apps at any specific times. Source view supports logs from cloud source repos, github, bitbucket, local machine traces being uploaded.
  - Cloud Debugger is a feature of Google Cloud that lets you inspect the state of an application at any code location without using logging statements and without stopping or slowing down your applications.
  - The gcloud debug command group provides interaction with Cloud Debugger, allowing you to list and manipulate debugging targets, snapshots and logpoints.
  - Import the 'google-python-cloud-debugger' library in your python code to add snapshots and logpoints to your code.
    - Debug Snapshots: After you have deployed or started your app, you can open Cloud Debugger in the Google Cloud Console. Debugger allows you to capture and inspect the call stack and local variables in your app without stopping or slowing it down. After you set a snapshot, the Debugger agent tests the snapshot on a subset of your instances. After the Debugger agent verifies the snapshot can execute successfully, the snapshot is applied to all your instances. This takes about 40 seconds.
    - Debug Logpoints: Logpoints allow you to inject logging into running services without restarting or interfering with the normal function of the service. Every time any instance executes code at the logpoint location, Cloud Debugger logs a message. Output is sent to the appropriate log for the target's environment.
- **SD profiler (now Cloud Profiler):**
  - Watch app cpu and memory usage realtime to improve perf of apps. Agent based wherein the agent send data to profiler to explore for 30 days, and is free
  - Needs the agent installed: 'google-cloud-profiler'
  - Import the googlecloudprofiler module and call the googlecloudprofiler.start function as early as possible in your initialization code.
  - The information provided shows which consumes the most CPU time and Wall Time (state-end time)
  - Cloud Trace is not supported on Google Cloud Storage. Stackdriver Trace runs on Linux in the following environments: Compute Engine, Google Kubernetes Engine (GKE), App Engine flexible environment, App Engine standard environment.

- Cloud billing api: Manage billing for gcp projects. List of accounts, details on each project, change account for each project.

- **VPC Flow logs:**
  - VPC Flow Logs record a sample of network flows sent from and received by VM instances. These logs can be used

for network monitoring, forensics, real-time security analysis, and expense optimization.
Flow logs are aggregated by connection, at 5-second intervals, from Compute Engine VMs and exported in real time.
By subscribing to Cloud Pub/Sub, you can analyze flow logs using real-time streaming APIs.

- If you want to adjust log sampling and aggregation, click Configure logs and adjust any of the following:
    - Aggregation interval: whether or not to Include metadata in the final log entries
    - By default, Include metadata only includes certain fields. Refer to Customizing metadata fields for details. To customize metadata fields, you must use the gcloud command-line interface or the API.
    - the Sample rate. 100% means that all entries are kept.

Logging and cloud audit

Tuesday, February 9, 2021 2:45 PM

Write log entries by using the gcloud tool

- Logging supports log entries with structured and unstructured data. Structured data consists of a JSON data structure; for example {"weather": "partly cloudy"}. Unstructured data is a string of characters; for example, "A simple entry".
- In the next steps, you use the gcloud tool to write a log entry with unstructured data and a log entry with structured data:
  - Write a log entry with unstructured data to the log my-test-log:
    - gcloud logging write my-test-log "A simple entry."
  - After the command completes, you see the message: Created log entry.
  - Write a log entry with structured data to my-test-log:
    - gcloud logging write --payload-type=json my-test-log '{ "message": "My second entry", "weather": "partly cloudy"}'
  - When you write a log entry with structured data, you must include --payload-type=json. If you omit this field, Logging interprets the payload as unstructured text.
- If the log my-test-log doesn't exist, then Logging creates the log when the log entry is received.

- **List log entries by using the gcloud tool**
- You can retrieve log entries from Logging and display them by using the gcloud tool. For example, to retrieve and display the log entries with a resource type of global, run the following command:
  - gcloud logging read "resource.type=global"

Query log entries in the Logs Explorer

You can query log entries by using a text field, and with structured logs, by the key and value. For example, to display all log entries that contain the text simple, do the following:

- In the search-query box, enter the string simple. The logs display shows only the log entry A simple entry.
- After you have viewed your log, remove the query string you added and click refresh (refresh). Both log entries reappear in the display.

To display all log entries with structured data that have a key of weather where the value field contains partly, do the following:

Switch to the advanced query mode by clicking the drop-down menu arrow_drop_down in the query box and then selecting Convert to advanced filter.

The query box contains the line resource.type="global". Under that line, enter this line:

jsonPayload.weather:partly

Click Submit filter.

About the Logging agent

In its default configuration, the Logging agent streams logs from common third-party applications and system software to Logging; review the list of default logs. You can configure the agent to stream additional logs; go to Configuring the Logging agent for details on agent configuration and operation.

It is a best practice to run the Logging agent on all your VM instances. The agent runs under both Linux and Windows. To install the Logging agent, go to Installing the agent.

If you are running specialized logging workloads that require higher throughput and/or improved resource-efficiency compared to the standard Cloud Logging agent, consider using the Ops agent.

The logging agent works on both GCP instances and AWS EC2 instances. You must allow internal routing in your gcp instance if the no external routing is supported.

The following VM instances support Logging using their own software, so manually installing the Logging agent on them is not supported:

- App Engine standard environment instances. App Engine includes built-in support for Logging. For more information, go to Writing application logs.
- App Engine flexible environment instances. Apps running in the App Engine flexible environment can write logs that are in addition to what is included in the App Engine standard environment. For more information, go to Writing application logs.
- Google Kubernetes Engine node instances. You can enable Cloud Operations for GKE, an integrated monitoring and logging solution, for your new or existing container clusters.
- For instances running on Anthos clusters on VMware, the agent collects system logs but doesn't collect application logs.
- Cloud Run container instances. Cloud Run includes built-in support for Logging. For more information, go to Logging and viewing logs.
- Cloud Functions HTTP and background functions. Cloud Functions includes built-in support for Logging. For more information, go to Writing, Viewing, and Responding to Logs.

Overview of logs exports

You can export some or all of your logs to various sink destinations. You might want to export logs for the following reasons:

- To store logs that are unlikely to be read but that must be retained for compliance purposes.
- To use big-data analysis tools on your logs.
- To stream your logs to other applications, other repositories, or third parties.

All logs, including audit logs, platform logs, and user logs, are sent to the Cloud Logging API where they pass through the Logs Router. The Logs Router checks each log entry against existing rules to determine which log entries to ingest (store), which log entries to include in exports, and which log entries to discard.

Exporting involves writing a **filter** that selects the log entries you want to export, and choosing a **destination** from the following options:

- Cloud Storage: JSON files stored in Cloud Storage buckets.
- BigQuery: Tables created in BigQuery datasets.
- Pub/Sub: JSON messages delivered to Pub/Sub topics. Supports third-party integrations, such as Splunk, with Logging.
- Another Google Cloud Cloud project: Log entries held in Cloud Logging logs buckets.

The filter and destination are held in an object called a **sink**. Sinks can be created in Google Cloud project, organizations, folders, and billing accounts.

Access control

To create or modify a sink, you must have the Identity and Access Management roles **Owner** or **Logging/Logs Configuration Writer** in the sink's parent resource. To view existing sinks, you must have the IAM roles **Viewer** or **Logging/Logs Viewer** in the sink's parent resource. For more information, go to Access control.

To export logs to a destination, the sink's writer service account must be permitted to write to the destination.

Cloud Audit Logs

Cloud Audit Logs provides the following audit logs for each Cloud project, folder, and organization:

- Admin Activity audit logs
- Data Access audit logs
- System Event audit logs
- Policy Denied audit logs

Google Cloud services write audit log entries to these logs to help you answer the questions of "who did what, where, and when?" within your Google Cloud resources.

Publicly available resources that have the Identity and Access Management policies allAuthenticatedUsers or allUsers don't generate audit logs. This helps protect end-user identities and information.

1. Admin Activity audit logs

- Admin Activity audit logs contain log entries for API calls or other actions that modify the configuration or metadata of resources. For example, these logs record when users create VM instances or change Identity and Access Management permissions.
- To view these logs, you must have the IAM role **Logging/Logs Viewer** or **Project/Viewer**.
- Admin Activity audit logs are always written; you can't configure or disable them. There is no charge for your Admin Activity audit logs.

2. Data Access audit logs

- Data Access audit logs contain API calls that read the configuration or metadata of resources, as well as user-driven API calls that create, modify, or read user-provided resource data.
- Data Access audit logs do not record the data-access operations on resources that are publicly shared (available to All Users or All Authenticated Users) or that can be accessed

without logging into Google Cloud.

- To view these logs, you must have the IAM roles **Logging/Private Logs Viewer** or **Project/Owner**.
- Data Access audit logs-- except for BigQuery Data Access audit logs-- are disabled by default because audit logs can be quite large. If you want Data Access audit logs to be written for Google Cloud services other than BigQuery, you must explicitly enable them. Enabling the logs might result in your Cloud project being charged for the additional logs usage.

3. System Event audit logs

- System Event audit logs contain log entries for Google Cloud actions that modify the configuration of resources. System Event audit logs are generated by Google systems; they are not driven by direct user action.
- To view these logs, you must have the IAM role **Logging/Logs Viewer** or **Project/Viewer**.
- System Event audit logs are always written; you can't configure or disable them. There is no charge for your System Event audit logs.

4. Policy Denied audit logs

- Cloud Logging records Policy Denied audit logs when a Google Cloud service denies access to a user or service account because of a security policy violation.
- To view these logs, you must have the IAM role **Logging/Logs Viewer** or **Project/Viewer**.
- Policy Denied audit logs are generated by default and your Cloud project is charged for the logs storage.
- You can use Logs exclusions to exclude Policy Denied logs from ingestion into Cloud Logging.

Audit log entry structure

Every audit log entry in Cloud Logging is an object of type LogEntry. What distinguishes an audit log entry from other log entries is the protoPayload field; this field contains an AuditLog object that stores the audit logging data.

Exporting audit logs

You can export audit log entries to Cloud Logging or to certain Google Cloud services.

To export audit log entries outside of Logging, create a logs sink. Give the sink a query that specifies the audit log types you want to export; for query examples, go to Security logging queries.

If you want to export audit log entries for a Google Cloud organization, folder, or billing account, review Aggregated sinks.

- Without the aggregated sink feature, sinks are limited to exporting log entries from the exact resource in which the sink was created: a Google Cloud project, organization, folder, or billing account.
- To use the aggregated sink feature, create a sink in a Google Cloud organization or folder

and set the sink's `includeChildren` parameter to `True`. That sink can then export log entries from the organization or folder, plus (recursively) from any contained folders, billing accounts, or projects. You can use the sink's filter to specify log entries from projects, resource types, or named logs.

The supported destinations for sinks are the following:

- A Cloud Storage bucket
- A Pub/Sub topic
- A BigQuery table
- A Cloud Logging bucket

Logging scenarios and solutions:

1. Scenario – Export for compliance requirements
    a. In this scenario, the exported logs are delivered to a Cloud Storage bucket that you configure. You grant permissions to limit access to the logs as appropriate. In order to reduce long-term storage costs, you can use the object lifecycle management feature in Cloud Storage to move logs to Nearline or Coldline storage classes and delete them after the required retention period has passed.
    b. Moving logs to Nearline or Coldline and then deleting them helps you manage the ongoing operational cost of maintaining the logs.
2. Scenario – Export for security and access analytics
    a. In this scenario, the exported logs are delivered to a dataset in BigQuery that you configure as a part of the export. You grant permissions to limit access to the logs as appropriate. You can simplify managing and querying the data by organizing the exported data into date-partitioned tables. This approach can help reduce query costs by reducing the amount of data scanned as a part of queries. One benefit of partitioning is that you can set an expiration date for the partitions so that you maintain the logging data only as long as you find it useful. For example, you might keep audit logging data for 3 years and then delete it.
3. Scenario – Export to Splunk
    a. This document gives you an overview of two different supported methods for log export to Splunk: either by pushing or pulling logs from Google Cloud. As explained in the following section, the cloud-native push-based approach is recommended in most cases.
4. Scenario – Export to Elasticsearch
5. Scenario – Export to Datadog

- ## How Google Cloud's operations suite routes logs
- In Cloud Logging, all logs, including audit logs, platform logs, and user logs, are sent to the Cloud Logging API where they pass through the Logs Router.
- The Logs Router checks each log entry against existing rules to determine which log entries to discard, which log entries to ingest (store) in Cloud Logging, and which log entries to route to supported destinations using log sinks.
- Cloud Logging compares each log entry it receives against the Google Cloud project, organization, or the folder's log sinks, which act independently of each other:

- **Sinks**. Cloud Logging compares the log entry against a sink's filter to determine whether to route the log entry to the sink's destination. Matching log entries are then compared against the sink's exclusion filters to determine whether to discard the log entry or to route it to the sink's destination. Logs sinks can be used to route log entries to supported destinations.
- **Exclusions**. By default, every project has a `_Default` logs sink that routes all logs to be stored in a `_Default` logs bucket in Cloud Logging. Logs exclusions control the exclusion filters for the `_Default` log sink and can be used to prevent matching logs from being stored in Cloud Logging by default.

Billing budgets and alerts

Tuesday, August 18, 2020 5:26 PM

Top 3 billing details to know:

- When creating a project through gcloud, it does not automatically link the project to your billing account; you can use gcloud beta billing to do that.

1) Billing accounts:
A billing acct is a type of resource that stays at the organization level, not project level. Hence billing accounts don't affect projects directly.
Consists of payment methods to link to a project. Each project will have 1 billing acc. You can change billing accounts. 2 types of billing accounts
A) Self served: Charged when you hit your acc or 30 days
B) Invoiced: For organizations

Role	Purpose	Level	Use Case
Billing Account Creator (`roles/billing.creator`)	Create new self-serve (online) billing accounts.	Organization	Use this role for initial billing setup or to allow creation of additional billing accounts. Users must have this role to sign up for Google Cloud with a credit card using their corporate identity. Tip: Minimize the number of users who have this role to help prevent proliferation of untracked cloud spend in your organization.
Billing Account Administrator (`roles/billing.admin`)	Manage billing accounts (but not create them).	Organization or billing account.	This role is an owner role for a billing account. Use it to manage payment instruments, configure billing exports, view cost information, link and unlink projects and manage other user roles on the billing account.
Billing Account User (`roles/billing.user`)	Link projects to billing accounts.	Organization or billing account.	This role has very restricted permissions, so you can grant it broadly, typically in combination with Project Creator. These two roles allow a user to create new projects linked to the billing account on which the role is granted.
Billing Account Viewer (`roles/billing.viewer`)	View billing account cost information and transactions.	Organization or billing account.	Billing Account Viewer access would usually be granted to finance teams, it provides access to spend information, but does not confer the right to link or unlink projects or otherwise manage the properties of the billing account.
Project Billing Manager (`roles/billing.projectManager`)	Link/unlink the project to/from a billing account.	Organization, folder, or project.	This role allows a user to attach the project to the billing account, but does not grant any rights over resources. Project Owners can use this role to allow someone else to manage the billing for the project without granting them resource access.

Billing account user can only link projects to billing accounts (from the org or billing acc perspective), but project billing manager can link and unlink projects from billing accounts (from org or project level perspective).
To change the billing account of a project: Use Project Owner or Project Billing Manager on the project, ***AND*** Billing Account Administrator or Billing Account User for the target Cloud Billing account.
Both the "Owner" role and the "Project Billing Manager" role each grant the same billing permissions--and these are exactly the `resourcemanager.projects.createBillingAssignment` permission

You can also create custom billing roles.

2) Budgets and alerts: Can link budget alerts to individual projects or individual billing accounts. If attached to a billing account, it will alert when the budget for the total account is reached, not each project. Can create alerts based on billing history too (previous month bill). Billing alerts are sent to the billing admin when thresholds are hit. You can use pub/sub to interact with these alerts.

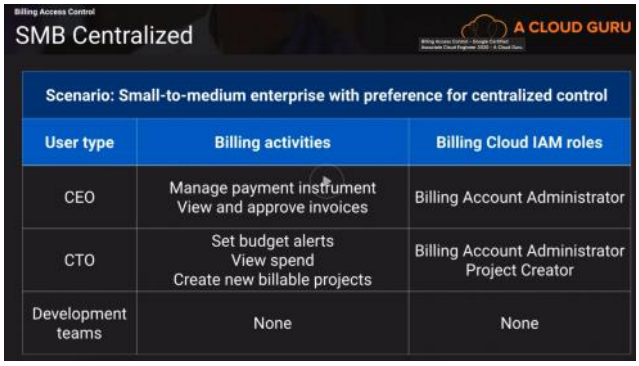
3) Billing exports: You can export billing data for better understanding using either BigQuery export or File Export. You can file store it in csv or json format. You can also automatically have it store a file per day into the storage bucket, so everyday new files would be added. However billing exports are NOT real time updated, so there can be a delay of even few hours between action and export.
Billing export has to be setup for each billing acount.
Billing data in BigQuery can be done using BQ in the project or even better a BQ in another project. This way the resources are separate but better organized. Billing export is not real-time, there is a delay of few hours. You should also tag different resources being tracked for better understanding.

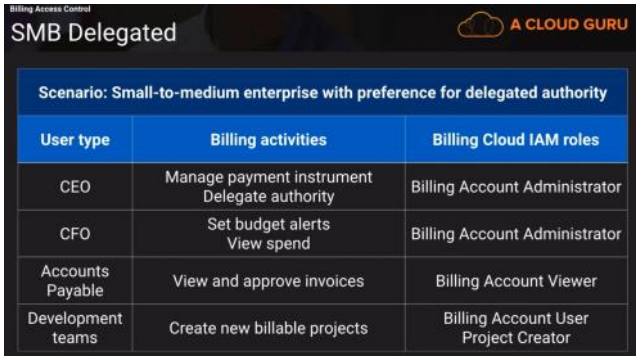
Move a billing account between Organizations

- A billing account can be moved from one Organization to another, although this isn't often a necessary step. Most existing Organizations will already have a billing account that should be

used instead. If you need to migrate an existing billing account:

- Get the necessary permissions for migration:
  - roles/billing.admin on the source Organization.
  - roles/billing.creator on the destination Organization.
  - Go to the Billing page in the Cloud Console.
  - Go to the Billing page
  - Click on the name of the billing account you want to move.
  - At the top of the Overview page, click Change organization.
  - Select the destination Organization, and then click Ok.
- The billing account is now associated with the specified Organization.





Metadata and its use with startup and stop scripts, MIG

Tuesday, December 17, 2019 9:09 PM

https://cloud.google.com/compute/docs/storing-retrieving-metadata

Every instance stores its metadata on a metadata server. You can query this metadata server programmatically, from within the instance and from the Compute Engine API. You can query for information about the instance, such as the instance's host name, instance ID, startup and shutdown scripts, custom metadata, and service account information. Your instance automatically has access to the metadata server API without any additional authorization.

The metadata server is particularly useful when used in combination with startup and shutdown scripts because you can use the metadata server to programmatically get unique information about an instance, without additional authorization. For example, you can write a startup script that gets the metadata key-value pair for an instance's external IP and use that IP in your script to set up a database. Because the default metadata keys are the same on every instance, you can reuse your script without having to update it for each instance. This helps you create less brittle code for your applications.

Metadata is stored in the format key:value. There is a default set of metadata entries that every instance has access to. You can also set custom metadata. Metadata ALWAYS uses http and not https to get data.

To access the metadata server, you can query the metadata URL.

Getting metadata:

You can query the contents of the metadata server by making a request to the following root URLs from within a virtual machine instance. Use the `http://metadata.google.internal/computeMetadata/v1/` URL to make requests to the metadata server.

Note: When you make a request to get information from the metadata server, your request and the subsequent metadata response never leave the physical host that is running the virtual machine instance. Since instance is connected to the metadata server internally over the VM machine, you can just use http instead of https

All metadata values are defined as sub-paths below these root URLs.

You can query for default metadata values only from within the associated instance. You cannot query an instance's default metadata from another instance or directly from your local computer. You can use standard tools like `curl` or `wget` from the instance to its metadata server.

When you query for metadata, you must provide the following header in all of your requests:

```
Metadata-Flavor: Google (this is required only for GCE not app engine. In app engine you need to use OAuth or service token to authenticate)
```

This header indicates that the request was sent with the intention of retrieving metadata values, rather than unintentionally from an insecure source, and lets the metadata server return the data you requested. If you don't provide this header, the metadata server denies your request.

Ex: curl -H Metadata-Flavor:Google metadata.google.internal/computeMetadata/v1/

For automating startup: you need to add key and value of bucket storage to allow the instance to store data directly to the bucket.

Some default metadata queries:

1. Relative to `http://metadata.google.internal/computeMetadata/v1/project/`
   a. `Additional attributes include: disable-legacy-endpoints, enable-oslogin, vmdnssetting, ssh-keys`
2. `Relative to http://metadata.google.internal/computeMetadata/v1/instance/`
   a. `Additional attributes include:enable-oslogin, vmdnssetting, ssh-keys, cpu-platform, description, disks/, guest-attributes/, hostname, id, machine-type, name, network-interfaces/, network-interfaces/<index>/forwarded-ips/, scheduling/ (preemptible, automatice-restart, on-host-maintenance), service-accounts/(name, token, identity), tags, zone`

Setting custom metadata

You can set custom metadata for an instance or project from the Google Cloud Console , the `gcloud` command-line tool, or the Compute Engine API. Custom metadata is useful for passing in arbitrary values to your project or instance, and for setting startup and shutdown scripts. There is a limit on key (128B) and values (256KB), hence keep in mind while setting ssh-keys or startup/shutdown scripts as they

count towards that limit.

CLI ex: `gcloud compute instances create example-instance \ --metadata foo=bar`

Compute Engine enforces a combined total limit of 512 KB for all metadata entries. Maximum size limits are also applied to each key and value as follows:

- Each metadata key has a maximum limit of 128 bytes
- Each metadata value has a maximum limit of 256 KB

In particular, SSH keys are stored as custom metadata under the ssh-keys key. If your metadata content for this key exceeds the 256 KB limit, you won't be able add more SSH keys. If you run into this limit, consider removing unused keys to free up metadata space for new keys.

Startup and shutdown script contents might also be stored as custom metadata and count toward these size limitations, if you provide the startup or shutdown script contents directly. To avoid this, store your startup or shutdown script as a file hosted at an external location, such as Cloud Storage, and provide the startup script URL when creating an instance. These files are downloaded onto the VM instance, rather than stored in the metadata server.

Some gcloud commands:

- gcloud compute project-info add-metadata \
  --metadata foo=bar,baz=bat
  : Add metadata at project level
- gcloud compute project-info describe \
  --flatten="commonInstanceMetadata[]"
  : Query project metadata
- gcloud compute project-info describe: Check metadata

- gcloud compute instances add-metadata instance-name \
  --metadata bread=mayo,cheese=cheddar,lettuce=romaine
  : Add metadata at instance level
- gcloud compute instances remove-metadata instance-name \
  --keys lettuce : Remove particular metadata
- gcloud compute instances describe example-instance \
  --flatten="metadata[]"
  : Query instance metadata

Guest attributes are a specific type of custom metadata that your applications can write to while running on your instance. Any application or user on your instance can both read and write data to these guest attribute metadata values.
Use guest attributes only for use cases that require small amounts of data that don't change frequently. The best use cases for guest attributes have the following characteristics:

- Startup scripts that can signal successful initialization by setting a custom status value in guest attributes.
- Configuration management agents that can publish a guest OS name and version to guest attributes.
- Inventory management agents that can publish list of packages installed in the VM instance to guest attributes.
- Workload orchestration software that can signal completion of an operation in the guest to the software control plane by setting a custom status value in guest attributes.

IAM, Service accounts

Thursday, December 12, 2019 9:29 PM

The GCP IAM model for access management has three main parts:

- **Member**. A *member* can be a Google Account (for end users), a service account (for apps and virtual machines), a Google group, or a Google Workspace or Cloud Identity domain that can access a resource. The identity of a member is an email address associated with a user, service account, or Google group; or a domain name associated with Google Workspace or Cloud Identity domains.
- **Role**. A *role* is a collection of permissions. Permissions determine what operations are allowed on a resource. When you grant a role to a member, you grant all the permissions that the role contains.
- **Policy**. The *IAM policy* binds one or more members to a role. When you want to define who (member) has what type of access (role) on a resource, you create a policy and attach it to the resource. Policies are only to Allow access not Deny.

Cloud identity is just the identity aspect of G-Suite without the rest of the apps. It is possible to use a non-google provider account if it is supported by google cloud directory sync (viz AD sync for google).

Roles:

There are more than 200+ roles.
Types of roles:

1. Primitive: Roles: Project Viewer, editor, owner. Owner can do everything an editor does but also handle billing and acct services. Predates Google IAM but still exists. Too generic for general use.
2. Predefined: More granular. For specific uses. IAM for different services.
3. Custom roles: Like predefined but custom made. Can be made as specific as needed but better to use least privilege policy. Custom roles allow us to group permissions to different services and assign members any of those roles. You can also segregate roles based on services and billing accounts. However when you create a custom role, you add permissions based on what google has defined at that time, incase google updates their permissions your custom role won't be updated so you would have to update it.

Each permission follows service.resource.verb, and any user who gets access is allowed access to that resource, for example, pubsub.subscriptions.consume.

Get-iam-policy: To get policy in json/yaml format. Set-iam-policy to set policy

```
gcloud projects get/set-iam-policy file.json
```

```
Delete iam role: gcloud iam roles delete ROLE_ID (--organization=ORGANIZATION|--project=PROJECT_ID)
[GCLOUD_WIDE_FLAG …]
```

You can use get-iam-policy in json/yaml format, edit the json/yaml file and can then set the new policy using set-iam-policy.
A Policy is a collection of bindings. A binding binds one or more members to a single role.

```
gcloud iam list-testable-permissions project-id : List applicable roles
```

JSON Format:

```
{
  "bindings": [
    {
      "role": "roles/resourcemanager.organizationAdmin",

            "members": [
                    "user:mike@example.com",
                    "group:admins@example.com",
                    "domain:google.com",
                    "serviceAccount:my-project-id@appspot.gserviceaccount.com"
                  ]

                } ] }
```

- Bindings can be multiple so it's a list, Role is singular so its just a dict key:value, members can be multiple so it's a list.

Instead, prefer to use :*"gcloud [group] add-iam-policy-binding [resource name] --role=[role-id-grant] --member=user: [user email]"*
"gcloud [group] remove-iam-policy-binding [resource name] --role=[role-id-grant] --member=user: [user email]"
since they are simpler, less work and less error prone. This also helps avoid race condition, wherein 2 user edit permissions at the same time, resulting in only 1 getting used. Race condition creates issue with *get-iam* and *set-iam* policy (where gcloud will try to add multiple policies at once, and some might interfere with other when not done sequentially), however in *gcloud add-iam policy* commands only the targeted command is sent to user not the whole set of commands.

https://cloud.google.com/iam/docs/using-iam-securely

- For projects level:
  - gcloud projects add-iam-policy-binding PROJECT_ID --member=MEMBER --role=ROLE [--condition=[KEY=VALUE,…] |--condition-from-file=CONDITION_FROM_FILE] [GCLOUD_WIDE_FLAG …]

    --condition : Condition expression that evaluates to True or False. Includes values like timestamp, resource type, ip addr, url path. Can be taken from a json or yaml file too.
- For organization level:
  - gcloud organizations add-iam-policy-binding ORGANIZATION --member=MEMBER --role=ROLE [--condition=[KEY=VALUE,…] | --condition-from-file=CONDITION_FROM_FILE] [GCLOUD_WIDE_FLAG …]

Folders are nodes in the Cloud Platform Resource Hierarchy. A folder can contain projects, other folders, or a combination of both. Organizations can use folders to group projects under the organization node in a hierarchy. For example, your organization might contain multiple departments, each with its own set of Google Cloud resources. Folders allow you to group these resources on a per-department basis. Folders are used to group resources that share common IAM policies. While a folder can contain multiple folders or resources, a given folder or resource can have exactly one parent.

- To create a folder under the Organization resource using the gcloud command-line tool, run the following command.
  - gcloud alpha resource-manager folders create \

    --display-name=[DISPLAY_NAME] \

    --organization=[ORGANIZATION_ID]
- To create a folder whose parent is another folder:
  - gcloud alpha resource-manager folders create \

    --display-name=[DISPLAY_NAME] \

    --folder=[FOLDER_ID]
- gcloud iam roles copy [--dest-organization=DEST_ORGANIZATION] [--dest-project=DEST_PROJECT] [--destination=DESTINATION] [--source=SOURCE] [--source-organization=SOURCE_ORGANIZATION] [--source-project=SOURCE_PROJECT] [GCLOUD_WIDE_FLAG …] : Copy IAM roles from 1 project/org to another
  - --dest-organization=DEST_ORGANIZATION
    - The organization of the destination role.
  - --dest-project=DEST_PROJECT
    - The project of the destination role.
  - --destination=DESTINATION
    - The destination role ID for the new custom role. For example: viewer.
  - --source=SOURCE
    - The source role ID. For predefined roles, for example: roles/viewer. For custom roles, for example: myCompanyAdmin.
  - --source-organization=SOURCE_ORGANIZATION
    - The organization of the source role if it is an custom role.
  - --source-project=SOURCE_PROJECT
    - The project of the source role if it is an custom role.
- `gcloud group remove-iam-policy-binding resource \ --member=member --role=role-id : To revoke or remove an IAM role. Group can be project or organization`
- Testing permissions
  - Most Google Cloud resources expose the testIamPermissions() method, which allows you to programmatically check whether the currently authenticated caller has been granted one or more specific IAM permissions on the resource. The testIamPermissions() method takes a resource identifier and a set of permissions as input parameters, and returns the set of permissions that the caller is allowed.

- Access scopes only apply to the default service account and not custom service accounts, since google assumes we have the correct settings in the custom account.

- Google kubernetes engine doesn't integrate with GCP-IAMs, hence you need to manually add IAMs for GKE.
- Google recommends predefined roles instead of primitive roles like Project Editor.

Members:

In IAM, you grant access to *members*. Members can be of the following types:

- Google Account: A Google Account represents a developer, an administrator, or any other person who interacts with Google Cloud. Any email address that's associated with a Google Account can be an identity, including gmail.com or other domains. New users can sign up for a Google Account by going to the Google Account signup page.
- Service account: A service account is an account for an application instead of an individual end user. When you run code that's hosted on Google Cloud, the code runs as the account you specify. You can create as many service accounts as needed to represent the different logical components of your application.

- Google group: A Google group is a named collection of Google Accounts and service accounts. Every Google group has a unique email address that's associated with the group. You can find the email address that's associated with a Google group by clicking **About** on the homepage of any Google group.
- Google Workspace domain: A Google Workspace domain represents a virtual group of all the Google Accounts that have been created in an organization's Google Workspace account. Google Workspace domains represent your organization's internet domain name (such as `example.com`), and when you add a user to your Google Workspace domain, a new Google Account is created for the user inside this virtual group (such as `username@example.com`). Like Google Groups, Google Workspace domains cannot be used to establish identity, but they enable convenient permission management.
- Cloud Identity domain: A Cloud Identity domain is like a Google Workspace domain because it represents a virtual group of all Google Accounts in an organization. However, Cloud Identity domain users don't have access to Google Workspace applications and features.
- All authenticated users: The value `allAuthenticatedUsers` is a special identifier that represents all service accounts and all users on the internet who have authenticated with a Google Account. This identifier includes accounts that aren't connected to a Google Workspace or Cloud Identity domain, such as personal Gmail accounts. Users who aren't authenticated, such as anonymous visitors, aren't included. Some resource types do not support this member type.
- All users: The value `allUsers` is a special identifier that represents anyone who is on the internet, including authenticated and unauthenticated users. Some resource types do not support this member type.

- **Service accounts:** Google says "For almost all cases whether you are developing locally or in a production, you should use service accounts rather than user accounts or api keys". Service accounts are meant to be used by services and code to leverage gcp api in an authorized manner. Some are maintained by google, others can be built by us. You need to specify to your code explicitly where your service account keys are, else it will choose default viz google's own service acc.
  - Service accounts are associated with private/public RSA key-pairs that are used for authentication to Google.
  - Service accounts do not have passwords
  - Note that service accounts can be thought of as both a resource and as an identity.
    - When thinking of the service account as an identity, you can grant a role to a service account, allowing it to access a resource (such as a project).
    - When thinking of a service account as a resource, you can grant roles to other users to access or manage that service account.

Types of service accounts

User-managed service accounts

You can create user-managed service accounts in your project using the IAM API, the Cloud Console, or the `gcloud` command-line tool. You are responsible for managing and securing these accounts.

By default, you can create up to 100 user-managed service accounts in a project. If this quota does not meet your needs, you can use the Cloud Console to request a quota increase. The default service accounts described on this page do not count towards this quota.

When you create a user-managed service account in your project, you choose a name for the service account. This name appears in the email address that identifies the service account, which uses the following format:

`service-account-name@project-id.iam.gserviceaccount.com`

Default service accounts

When you use some Google Cloud services, they create user-managed service accounts that enable the service to deploy jobs that access other Google Cloud resources. These accounts are known as *default service accounts*.

The default service accounts help you get started with Google Cloud services. For production workloads, we strongly recommend that you create your own user-managed service accounts and grant the appropriate roles to each service account.

When a default service account is created, it is automatically granted the Editor role (`roles/editor`) on your project. This role includes a very large number of permissions. To follow the principle of least privilege, we strongly recommend that you disable the automatic role grant by adding a constraint to your organization policy, or by revoking the Editor role manually. If you disable or revoke the role grant, you must decide which roles to grant to the default service accounts.

The following table lists the services that create default service accounts:

Service	Service account name	Email address
App Engine, and any Google Cloud service that uses App Engine	App Engine default service account	`project-id@appspot.gserviceaccount.com`
Compute Engine, and any Google Cloud service that uses Compute Engine	Compute Engine default service account	`project-number-compute@developer.gserviceaccount.com`

User impersonating service accounts:

- There are several predefined roles that allow a member to impersonate a service account:

- **Service Account User (roles/iam.serviceAccountUser)**: Allows members to indirectly access all the resources that the service account can access. For example, if a member has the Service Account User role on a service account, and the service account has the Cloud SQL Admin role (roles/cloudsql.admin) on the project, then the member can impersonate the service account to create a Cloud SQL instance.
- When granted together with roles/compute.instanceAdmin.v1, roles/iam.serviceAccountUser gives members the ability to create and manage instances that use a service account. Specifically, granting roles/iam.serviceAccountUser and roles/compute.instanceAdmin.v1 together gives members permission to:
  - Create an instance that runs as a service account.
  - Attach a persistent disk to an instance that runs as a service account.
  - Set instance metadata on an instance that runs as a service account.
  - Use SSH to connect to an instance that runs as a service account.
  - Reconfigure an instance to run as a service account.
- You can grant roles/iam.serviceAccountUser one of two ways:
  - **Recommended**. Grant the role to a member on a specific service account. This gives a member access to the service account for which they are an iam.serviceAccountUser but prevents access to other service accounts for which the member is not an iam.serviceAccountUser.
  - Grant the role to a member on the project level. The member has access to all service accounts in the project, including service accounts that are created in the future.

Service accounts and service account keys

- Rotate your service account keys using the IAM service account API. You can rotate a key by creating a new key, switching applications to use the new key and then deleting old key. Use the serviceAccount.keys.create() method and serviceAccount.keys.delete() method together to automate the rotation.
- Implement processes to manage user-managed service account keys.
- Service accounts do not have passwords
- Be careful not to confuse encryption keys with service account keys. Encryption keys are typically used to encrypt data and service account keys are used for secure access to Google Cloud APIs.
- Do not delete service accounts that are in use by running instances. This could result in all or parts of your application to fail if you have not transitioned to using an alternative service account first.
- Use the display name of a service account to keep track of what they are used for and what permissions they should have.
- Don't check in the service account keys into the source code or leave them in the Downloads directory.

- **Authenticating an application as a service account [REF}**
- Application credentials provide the required information about the caller making a request to a Google Cloud API. Valid credential types include API keys, OAuth 2.0 client credentials, or service account keys.
- Follow this guide to decide how to authenticate your app : [ref]

Requirement	Recommendation	Comment
Accessing public data anonymously	API key	An API key only identifies the application and doesn't require user authentication. It is sufficient for accessing public data.
Accessing private data on behalf of an end user	OAuth 2.0 client	An OAuth 2.0 client identifies the application and lets end users authenticate your application with Google. It allows your application to access Google Cloud APIs on behalf of the end user.
Accessing private data on behalf of a service account inside Google Cloud environments	Environment-provided service account	If your application runs inside a Google Cloud environment, such as Compute Engine, App Engine, GKE, Cloud Run, or Cloud Functions, your application should use the service account provided by the environment. Google Cloud Client Libraries will automatically find and use the service account credentials.
Accessing private data on behalf of a service account outside Google Cloud environments	Service account key	You need to create a service account, and download its private key as a JSON file. You need to pass the file to Google Cloud Client Libraries, so they can generate the service account credentials at runtime. Google Cloud Client Libraries will automatically find and use the service account credentials by using the GOOGLE_APPLICATION_CREDENTIALS environment variable.

- Automatic:
  - If your application runs inside a Google Cloud environment that has a default service account, your application can retrieve the service account credentials to call Google Cloud APIs.
  - We recommend using this strategy because it is more convenient and secure than manually passing credentials.
  - Additionally, we recommend you use Google Cloud Client Libraries for your application. Google Cloud Client Libraries use a library called Application Default Credentials (ADC) to automatically find your service account credentials.
  - ADC looks for the credentials in the following order:

- If GOOGLE_APPLICATION_CREDENTIALS is set as an env variable, use it.
- If GOOGLE_APPLICATION_CREDENTIALS is NOT set as an env variable, use service acc of the resource running your code
- If not either, use the default service acc for that service.

- Manual:
  - Create a service account using 'gcloud iam service-accounts create <acc-name>'
  - Grant permissions using 'gcloud projects add-iam-policy <project id> --member="serviceAccount=<acc-name>@<project id>" --role="roles/owner"'
  - Generate the key file (in json) using 'gcloud iam service-accounts keys create <file name>.json --iam-account=<acc-name>@<project id>.iam.gserviceaccount.com'
  - Pass this json file to env variable GOOGLE_APPLICATION_CREDENTIALS using export i.e export GOOGLE_APPLICATION_CREDENTIALS="<path to file>/key.json"
- Create JSON keys for the service account and execute gcloud auth activate-service-account --key-file [KEY_FILE] : To create a service acc and verify if it works.

User-managed keys and other options:

User-managed key pairs imply that you own both the public and private portions of a key pair. You can create one or more user-managed key pairs (also known as "external" keys) that can be used from outside of Google Cloud. Google only stores the public portion of a user-managed key.

Additionally, you can create a public key in the appropriate format and upload it to Google, where it is permanently associated with the specified service account. When you need to perform signing operations on behalf of that service account, such as when creating service account keys, the uploaded public key is used.

The private portion of a user-managed key pair is most commonly used with Application Default Credentials. The private key is then used to authenticate server-to-server applications. For user-managed keys, you are responsible for security of the private key and other management operations such as key rotation. User-managed keys can be managed by the IAM API, `gcloud` command-line tool, or the service accounts page in the Google Cloud Console. You can create up to 10 service account keys per service account to facilitate key rotation.

Consider using Cloud Key Management Service (Cloud KMS) to help securely manage your keys.

Preventing user-managed keys: User-managed keys are extremely powerful credentials, and they can represent a security risk if they are not managed correctly. You can limit their use by applying the `constraints/iam.disableServiceAccountKeyCreation` Organization Policy Constraint to projects, folders, or even your entire organization. After applying the constraint, you can enable user-managed keys in well-controlled locations to minimize the potential risk caused by unmanaged keys.

Different types of **predefined** roles:

Project Level Roles

Role	Description	Permissions	
roles/resourcemanager.projectCreator	Project Creator	Provides access to create new projects. Once a user creates a project, they're automatically granted the owner role for that project.	resourcemanager.organizations.get resourcemanager.projects.create
roles/resourcemanager.projectDeleter	Project Deleter	Provides access to delete Google Cloud projects.	resourcemanager.projects.delete
roles/resourcemanager.projectMover	Project Mover	Provides access to update and move projects.	resourcemanager.projects.get resourcemanager.projects.move resourcemanager.projects.update
roles/resourcemanager.projectIamAdmin	Project IAM Admin	Provides permissions to administer IAM policies on projects.	resourcemanager.projects.get resourcemanager.projects.getIamPolicy resourcemanager.projects.setIamPolicy
roles/browser	Browser	Read access to browse the hierarchy for a project, including the folder, organization, and IAM policy. This role doesn't include permission to view resources in the project.	resourcemanager.folders.get resourcemanager.folders.list resourcemanager.organizations.get resourcemanager.projects.get resourcemanager.projects.getIamPolicy resourcemanager.projects.list

APP Engine

Roles/appengine.appAdmin	App Engine Admin	Read/Write/Modify access to all application configuration and settings.	Project
Roles/appengine.appViewer	App engine viewer	Read-only access to all application configuration and settings. Not deployed source code	Project
Roles/appengine	App engine	Read-only access to all application configuration, settings, and deployed source code. This is the only app engine predefined	Project

.codeViewer	code viewer	role that grants access to view the Source code (not even admin)	
Roles/appengine.deployer	App engine deployer	Read-only access to all application configuration and settings. Write access only to create a new version; cannot modify existing versions other than deleting versions that are not receiving traffic. Note: The App Engine Deployer (roles/appengine.deployer) role alone grants adequate permission to deploy using the App Engine Admin API. To use other App Engine tooling, like gcloud commands, you must also have the Compute Storage Admin (roles/compute.storageAdmin) and Cloud Build Editor (cloudbuild.builds.editor) roles.	Project
Roles/appengine.serviceAdmin	App engine service admin	Read-only access to all application configuration and settings. Write access to module-level and version-level settings. Cannot deploy a new version.	Project
`roles/appengine.appCreator`	App Engine Creator	Ability to create the App Engine resource for the project.	Project

All the above roles on their own can use app-engine using the API, if you want to use the gcloud tool to deploy, you must add the Compute Storage Admin (roles/compute.storageAdmin) and Cloud Build Editor (roles/cloudbuild.builds.editor) roles.

Compute Engine		
Roles/compute.admin	Compute admin	Full control of all Compute Engine resources. If the user will be managing virtual machine instances that are configured to run as a service account, you must also grant the `roles/iam.serviceAccountUser` role.
Roles/compute.imageUser	Compute image user	Permission to list and read images without having other permissions on the image. Granting the `compute.imageUser` role at the project level gives users the ability to list all images in the project and create resources, such as instances and persistent disks, based on images in the project.
Roles/compute.instanceAdmin	Compute instance admin (beta)	Permissions to create, modify, and delete virtual machine instances. This includes permissions to create, modify, and delete disks, and also to configure Shielded VMBETA settings. If the user will be managing virtual machine instances that are configured to run as a service account, you must also grant the `roles/iam.serviceAccountUser` role. For example, if your company has someone who manages groups of virtual machine instances but does not manage network or security settings and does not manage instances that run as service accounts, you can grant this role on the organization, folder, or project that contains the instances, or you can grant it on individual instances.
Roles/compute.instanceAdmin.v1	Compute instance admin (v1)	Full control of Compute Engine instances, instance groups, disks, snapshots, and images. Read access to all Compute Engine networking resources. If you grant a user this role only at an instance level, then that user cannot create new instances.
Roles/compute.viewer	Compute viewer	Read-only access to get and list Compute Engine resources, without being able to read the data stored on them. For example, an account with this role could inventory all of the disks in a project, but it could not read any of the data on those disks.
LoadBalancer Roles		
Roles/compute.loadBalancerAdmin	Compute load balancer admin	Permissions to create, modify, and delete load balancers and associate resources. For example, if your company has a load balancing team that manages load balancers, SSL certificates for load balancers, SSL policies, and other load balancing resources, and a separate networking team that manages the rest of the networking resources, then grant the load balancing team's group the `loadBalancerAdmin` role.
Networking roles		
Roles/compute.networkAdmin	Compute network admin	Permissions to create, modify, and delete networking resources, except for firewall rules and SSL certificates. The network admin role allows read-only access to firewall rules, SSL certificates, and instances (to view their ephemeral IP addresses). The network admin role does not allow a user to create, start, stop, or delete instances. For example, if your company has a security team that manages firewalls and SSL certificates and a networking team that manages the rest of the networking resources, then grant the networking team's group the `networkAdmin` role.
Roles/compute.networkUser	Compute network user	Provides access to a shared VPC network Once granted, service owners can use VPC networks and subnets that belong to the host project. For example, a network user can create a VM instance that belongs to a host project network but they cannot delete or create new networks in the host project.
Roles/compute.networkViewer	Compute network viewer	Read-only access to all networking resources For example, if you have software that inspects your network configuration, you could grant that software's service account the `networkViewer` role.
Security Roles		
Roles/compute.orgSecurityPolicyAdmin	Compute org security policy admin	Full control of Compute Engine Organization Security Policies.
Roles/compute.orgSecurityPolicyUser	Compute org security policy User	View or use Compute Engine Security Policies to associate with the organization or folders.
SSH Login roles		
Roles/compute.osAdminLogin	Compute os admin login	Access to log in to a Compute Engine instance as an administrator user.

Roles/compute.osLogin	Compute os login	Access to log in to a Compute Engine instance as a standard user.
Roles/compute.osLoginExternalUser	Compute os login external user	Available only at the organization level. Access for an external user to set OS Login information associated with this organization. This role does not grant access to instances. External users must be granted one of the required OS Login roles in order to allow access to instances using SSH.
Packet mirroring		
Roles/compute.packetMirroringAdmin	Compute packet mirroring admin	Specify resources to be mirrored.
Roles/compute.packetMirroringUser	Compute packet mirror user	Use Compute Engine packet mirrorings
Firewall rules		
Roles/compute.securityAdmin	Compute Security Admin	Permissions to create, modify, and delete firewall rules and SSL certificates, and also to configure Shielded VMBETA settings. For example, if your company has a security team that manages firewalls and SSL certificates and a networking team that manages the rest of the networking resources, then grant the security team's group the `securityAdmin` role.
Shared VPC role		
Roles/compute.xpnAdmin	Compute shared vpc admin	Permissions to administer shared VPC host projects, specifically enabling the host projects and associating shared VPC service projects to the host project's network. This role can only be granted on the organization by an organization admin. Google Cloud recommends that the Shared VPC Admin be the owner of the shared VPC host project. The Shared VPC Admin is responsible for granting the `compute.networkUser` role to service owners, and the shared VPC host project owner controls the project itself. Managing the project is easier if a single principal (individual or group) can fulfill both roles.

BigQuery predefined IAM roles

The following table lists the predefined BigQuery IAM roles with a corresponding list of all the permissions each role includes. Note that every permission is applicable to a particular resource type. You can grant access at the following BigQuery resource levels: organization or Google Cloud project level, dataset level, table or view level.

Role	Title	Description	Permissions	Lowest resource
`roles/bigquery.admin`	BigQuery Admin	Provides permissions to manage all resources within the project. Can manage all data within the project, and can cancel jobs from other users running within the project.	bigquery.* resourcemanager.projects.get resourcemanager.projects.list	Project
`roles/bigquery.connectionAdmin`	BigQuery Connection Admin		bigquery.connections.*	
`roles/bigquery.connectionUser`	BigQuery Connection User		bigquery.connections.get bigquery.connections.getIamPolicy bigquery.connections.list bigquery.connections.use	
`roles/bigquery.dataEditor`	BigQuery Data Editor	When applied to a table or view, this role provides permissions to: • Read and update data and metadata for the table or view. • Delete the table	bigquery.datasets.create bigquery.datasets.get bigquery.datasets.getIamPolicy bigquery.datasets.updateTag bigquery.models.* bigquery.routines.* bigquery.tables.create bigquery.tables.delete bigquery.tables.export bigquery.tables.get bigquery.tables.getData bigquery.tables.getIamPolicy bigquery.tables.list bigquery.tables.update	Table or view

		or view. This role cannot be applied to individual models or routines. When applied to a dataset, this role provides permissions to: • Read the dataset's metadata and list tables in the dataset. • Create, update, get, and delete the dataset's tables. When applied at the project or organization level, this role can also create new datasets.	bigquery.tables.updateData bigquery.tables.updateTag resourcemanager.projects.get resourcemanager.projects.list	
`roles/bigquery.dataOwner`	BigQuery Data Owner	When applied to a table or view, this role provides permissions to: • Read and update data and metadata for the table or view. • Share the table or view. • Delete the table or view. This role cannot be applied to individual models or routines. When applied to a dataset, this role provides permissions to: • Read, update, and delete the dataset. • Create, update, get, and delete the	bigquery.datasets.* bigquery.models.* bigquery.routines.* bigquery.tables.* resourcemanager.projects.get resourcemanager.projects.list	Table or view

		dataset's tables. When applied at the project or organization level, this role can also create new datasets.		
roles/bigquery.dataViewer	BigQuery Data Viewer	When applied to a table or view, this role provides permissions to: • Read data and metadata from the table or view. This role cannot be applied to individual models or routines. When applied to a dataset, this role provides permissions to: • Read the dataset's metadata and list tables in the dataset. • Read data and metadata from the dataset's tables. When applied at the project or organization level, this role can also enumerate all datasets in the project. Additional roles, however, are necessary to allow the running of jobs.	bigquery.datasets.get bigquery.datasets.getIamPolicy bigquery.models.export bigquery.models.getData bigquery.models.getMetadata bigquery.models.list bigquery.routines.get bigquery.routines.list bigquery.tables.export bigquery.tables.get bigquery.tables.getData bigquery.tables.getIamPolicy bigquery.tables.list resourcemanager.projects.get resourcemanager.projects.list	Table or view
roles/bigquery.jobUser	BigQuery Job User	Provides permissions to run jobs, including queries, within the project.	bigquery.jobs.create resourcemanager.projects.get resourcemanager.projects.list	Project
roles/bigquery.metadataViewer	BigQuery Metadata Viewer	When applied to a table or view, this role provides permissions to: • Read metadata from the table or	bigquery.datasets.get bigquery.datasets.getIamPolicy bigquery.models.getMetadata bigquery.models.list bigquery.routines.get bigquery.routines.list bigquery.tables.get bigquery.tables.getIamPolicy bigquery.tables.list	Table or view

		view. This role cannot be applied to individual models or routines. When applied to a dataset, this role provides permissions to: • List tables and views in the dataset. • Read metadata from the dataset's tables and views. When applied at the project or organization level, this role provides permissions to: • List all datasets and read metadata for all datasets in the project. • List all tables and views and read metadata for all tables and views in the project. Additional roles are necessary to allow the running of jobs.	resourcemanager.projects.get resourcemanager.projects.list	
`roles/bigquery.readSessionUser`	BigQuery Read Session User	Access to create and use read sessions	bigquery.readsessions.* resourcemanager.projects.get resourcemanager.projects.list	
`roles/bigquery.resourceAdmin`	BigQuery Resource Admin	Administer all BigQuery resources.	bigquery.bireservations.* bigquery.capacityCommitments.* bigquery.jobs.get bigquery.jobs.list bigquery.jobs.listAll bigquery.reservationAssignments.* bigquery.reservations.* resourcemanager.projects.get resourcemanager.projects.list	
`roles/bigquery.resourceEditor`	BigQuery Resource Editor	Manage all BigQuery resources, but cannot make	bigquery.bireservations.get bigquery.capacityCommitments.get bigquery.capacityCommitments.list bigquery.jobs.get	

		purchasing decisions.	bigquery.jobs.list bigquery.jobs.listAll bigquery.reservationAssignments.* bigquery.reservations.* resourcemanager.projects.get resourcemanager.projects.list	
roles/bigquery.resourceViewer	BigQuery Resource Viewer	View all BigQuery resources but cannot make changes or purchasing decisions.	bigquery.bireservations.get bigquery.capacityCommitments.get bigquery.capacityCommitments.list bigquery.jobs.get bigquery.jobs.list bigquery.jobs.listAll bigquery.reservationAssignments.list bigquery.reservationAssignments.search bigquery.reservations.get bigquery.reservations.list resourcemanager.projects.get resourcemanager.projects.list	
roles/bigquery.user	BigQuery User	When applied to a dataset, this role provides the ability to read the dataset's metadata and list tables in the dataset. When applied to a project, this role also provides the ability to run jobs, including queries, within the project. A member with this role can enumerate their own jobs, cancel their own jobs, and enumerate datasets within a project. Additionally, allows the creation of new datasets within the project; the creator is granted the BigQuery Data Owner role (roles/bigquery.dataOwner) on these new datasets.	bigquery.bireservations.get bigquery.capacityCommitments.get bigquery.capacityCommitments.list bigquery.config.get bigquery.datasets.create bigquery.datasets.get bigquery.datasets.getIamPolicy bigquery.jobs.create bigquery.jobs.list bigquery.models.list bigquery.readsessions.* bigquery.reservationAssignments.list bigquery.reservationAssignments.search bigquery.reservations.get bigquery.reservations.list bigquery.routines.list bigquery.savedqueries.get bigquery.savedqueries.list bigquery.tables.list bigquery.transfers.get resourcemanager.projects.get resourcemanager.projects.list	Dataset

Cloud storage roles

Role	Description	Permissions
Storage Object Creator (roles/storage.objectCreator)	Allows users to create objects. Does not give permission to view, delete, or replace objects.	resourcemanager.projects.get resourcemanager.projects.list storage.objects.create
Storage Object Viewer (roles/storage.objectViewer)	Grants access to view objects and their metadata, excluding ACLs. Can also list the objects in a bucket.	resourcemanager.projects.get resourcemanager.projects.list storage.objects.get storage.objects.list
Storage Object Admin (roles/storage.objectAdmin)	Grants full control over objects, including listing, creating, viewing, and deleting objects.	resourcemanager.projects.get resourcemanager.projects.list storage.objects.*

Storage HMAC Key Admin (roles/storage.hmacKeyAdmin)	Full control over HMAC keys in a project. This role can only be applied to a project.	storage.hmacKeys.*
Storage Admin (roles/storage.admin)	Grants full control of buckets and objects. When applied to an individual **bucket**, control applies only to the specified bucket and objects within the bucket.	firebase.projects.get resourcemanager.projects.get resourcemanager.projects.list storage.buckets.* storage.objects.*

Kubernetes engine roles

Roles/container.admin	Kubernetes engine admin	Provides access to full management of Container Clusters and their Kubernetes API objects.	Project
Roles/container.clusterAdmin	Kub engine cluster admin	Provides access to management of Container Clusters.	Project
Roles/container.clusterViewer	Kub engine cluster viewer	Read-only access to Kubernetes Clusters.	Project
Roles/container.developer	Kub engine developer	Provides full access to Kubernetes API objects inside Container Clusters.	Project
Roles/container.hostServiceAgentUser	Kub eng host service agent user	Use access of the Kubernetes Engine Host Service Agent.	
Roles/container.viewer	Kub eng viewer	Provides read-only access to GKE resources	Project

IAM Best practices and scenarios

Tuesday, February 9, 2021 5:52 PM

- Gcloud projects/org get-iam-policy file.json
- Gcoud projects/orgs add/remove-iam-policy-binding –role=role –member=member
- Gcloud iam roles delete
- Gcloud iam roles create –project/org=projectid/orgid --file=file.json/yaml or (--permissions=permissions --)
- Gcloud iam list-testable-roles project
- Gcloud iam roles copy –dest=dest –src=src
- Gcloud iam service-accounts create
- 'gcloud iam service-accounts keys create <file name>.json --iam-account=<acc-name>@<project id>.iam.gserviceaccount.com' : Generate credentials file for service account

Best practices

- Mirror your Google Cloud resource hierarchy structure to your organization structure. The Google Cloud resource hierarchy should reflect how your company is organized, whether it's a startup, a SME, or a large corporation. A startup may start out with a flat resource hierarchy with no organization resource. When more people start collaborating on projects and the number of projects increase, getting an organization resource might make sense. An organization resource is recommended for larger companies with multiple departments and teams where each team is responsible for their own set of applications and services.
- Use projects to group resources that share the same trust boundary. For example, resources for the same product or microservice can belong to the same project.
- Set policies at the organization level and at the project level rather than at the resource level. As new resources are added, you may want them to automatically inherit policies from their parent resource. For example, as new virtual machines are added to the project through auto-scaling, they automatically inherit the policy on the project.
  For more information about how to set policies, see Granting, changing, and revoking access.
- Grant roles to a Google group instead of to individual users when possible. It is easier to manage members in a Google group than to update an IAM policy. Make sure to control the ownership of the Google group used in IAM policies.
  For more information about how to manage Google groups, see Google Groups help.
- Use the security principle of least privilege to grant IAM roles; that is, only give the least amount of access necessary to your resources.
  To find the appropriate predefined role, see the predefined roles reference. If there are no appropriate predefined roles, you can also create your own custom roles.
- Grant roles at the smallest scope needed. For example, if a user only needs access to publish messages to a Pub/Sub topic, grant the Publisher role to the user for that topic.
- Remember that the policies for child resources inherit from the policies for their parent resources. For example, if the policy for a project grants a user the ability to administer Compute Engine virtual machine (VM) instances, then the user can administer any Compute Engine VM in that project, regardless of the policy you set on each VM.
- If you need to grant a role to a user or group that spans across multiple projects, set that role at the folder level instead of setting it at the project level.

- Use labels to annotate, group, and filter resources.
- Audit your policies to ensure compliance. Audit logs contain all `setIamPolicy()` calls, so you can trace when a policy has been created or modified.
- Audit the ownership and the membership of the Google groups used in policies.
- If you want to limit project creation in your organization, change the organization access policy to grant the Project Creator role to a group that you manage.

IAM roles for Billing-related Job Functions

This topic shows you how to configure Identity and Access Management (IAM) permissions for a set of sample billing scenarios. It provides guidance on which IAM roles to grant to the billing-related functional roles in your company for the scenarios. These examples are mainly targeted at billing administrators and employees who manage billing tasks for an organization.

- Small company configuring billing permissions:
  - In this scenario a small company is trying to configure and use Google billing accounts. They have a handful of engineers who develop and maintain their applications, but none of them manage their billing. They have an office manager, who is responsible for matching payments to invoices, but for compliance reasons the office manager is not permitted to have access to Google Cloud resources in the projects. The CEO also holds and manages the credit card details.
  - CEO: As just user, give him Organization administrator role. "role": "roles/resourcemanager.organizationAdmin"
  - CEO and Office manager: Create a group with them in it. Grant the group the Billing Admin role. "role": "roles/billing.admin"

- Finance teams managing budgets:
  - In this scenario, a large organization wants the finance team in each division to be able to set budgets and view team spending in the division, but not have access to the Google Cloud resources. They don't mind if the developers see the spend for their own projects, but a broad view of expenses should not be allowed to the developers.
  - Finance manager for each division: Grant the finance group billing account administrator role on the billing account that has been setup. "role": "roles/billing.admin". This allows them to set budgets and view spending for billing accounts in their division, but not to view project contents.
  - Developers: Grant them the Viewer role for the billing account. This allows them to view the expenses for their project, but not edit them.

- Customer self-service portal, Developers cannot adjust billing:
  - In this scenario, a customer's central IT team provides Google Cloud resources to their developers as part of their self service portal. Developers request access to Google Cloud projects and other approved cloud services via the portal. The cost center of the developer pays the central IT team for the cloud resources consumed.

- The central IT team must be able to:
  - Associate projects with billing accounts.
  - Turn off billing for projects.
  - View the credit card information.
  - They must not have permissions to view the project contents.
  - Developers should be able to view the actual costs of the Google Cloud resources being consumed, but shouldn't be able to turn billing off, associate billing with projects, and view the credit card information.
- IT Dept: Grant them the billing account administrator role. They can associate projects with billing, turn of billing on projects, view credit card info, but cannot view the projects.
- Service account: Create a Service account that is used for automating project creation. Geant it the Billing account user role to enable billing on projects.

  "role": "roles/billing.user",

  "members": [

  serviceAccount:my-project-creator@shared-resources-proj.iam.gserviceaccount.com]
- Developers of project: Grant them the Viewer role to allow the developers to view the expenses for the projects they own.

- Developers creating billed projects:
  - A large digital native wants to allow all their developers to create billed projects on their organization's invoiced account without giving them Billing Account Administrator rights.
  - A project needs to have billing enabled to ensure that APIs beyond the default can be enabled. Thus if a developer creates a project, they need to associate it with a billing account to enable the APIs.
  - Developers. Grant the developers group Billing account user role to attach the billing account to new projects.

- Cost aggregation
  - In this scenario, a company wants to calculate and keep track of how much each team, department, service, or project is costing them. For example, keep track of how much does a test deployment cost them each month.
  - This can be tracked by using the following practices:
  - Use projects to organize resources. Cost is shown per project and project IDs are included in billing export.
  - Annotate projects with labels that represent additional grouping information. For example, environment=test. Labels are included in billing export to allow you to slice and dice further. However, labels on a project are permissioned the same way as the rest of the project's metadata which means a project owner can change labels. You can educate your employees about what not to change and then monitor (through audit logs), or grant them only granular permissions so they can't change project metadata.

- You can export to JSON and CSV, but exporting directly to BigQuery is the solution we recommend. This is easily configurable from the billing export section of the billing console.
- If each cost center must pay a separate invoice or pay in a separate currency for some workloads, then a separate billing account for each cost center is required. However this approach would require an affiliate agreement signed for each billing account

IAM roles for Networking-related Job Functions

This topic shows how to configure Identity and Access Management (IAM) permissions for networking scenarios. It provides guidance on what IAM roles to grant to the networking-related functional roles in your company for the scenarios. This content is mainly targeted at network administrators and employees who manage networking tasks for an organization. The scenarios described below all assume that a Google Cloud organization is configured.

- Single team manages security & network for organization
  - In this scenario, a large organization has a central team that manages security and networking controls for the entire organization. Developers do not have permissions to make changes to any network or security settings defined by the security and networking team but they are granted permission to create resources such as virtual machines in shared subnets.
  - To facilitate this the organization makes use of a shared VPC (Virtual Private Cloud). A shared VPC allows creation of a VPC network of RFC 1918 IP spaces that associated projects (service projects) can then use. Developers using the associated projects can create VM instances in the shared VPC network spaces. The organization's network and security admins can create subnets, VPNs, and firewall rules usable by all the projects in the VPC network.
  - For this scenario you need three separate IAM policies: one for the organization, one for the host project, and one for the service projects.
  - The first IAM policy, which needs to be attached at the organization level, grants the network and security team the roles they need to administer shared VPC host projects. This includes the ability to associate service projects with the host project. It also grants the network and security team the ability to manage all network and security resources in all projects in the organization.
  - Security & network admin team: Grant them 3 roles at the organization level. The xpn.Admin role (xpn is short for shared vpc) allows vpc sharing, the network and security admin allows them to manage all network and security resource in all projects.
    - Group: sec@example.com,
    - Role: roles/compute.xpnAdmin, roles/compute.networkAdmin, roles/compute.securityAdmin
  - Developers: They need 2 iam policies: 1 to allow them to use subnets of the shared vpc (at the host project level) and the other to manage instances in the service project and use shared vpc in host project (at each service project level)
    - Group: dev@exaple.com
    - Role: roles/compute.networkUser (host project level), roles/compute.networkUser,

roles/compute.instanceAdmin (at service project level)

Resource:	Organization	
Roles:	Shared VPC Admin Network Admin Security Admin	
Member:	Security & network admin team	

Resource:	Host Project	This role grants permission to use subnets that the shared VPC has shared.
Role:	Network user	
Member:	Developers	

Resource:	Service project	Note this role allows the permission to use External IP addresses. See the note below for guidance on how to prevent this action.
Role:	compute.instanceAdmin	
Member:	Developers	

- Separate network & security teams
  - In this scenario, a large organization has two central teams: one that manages security controls, and another that manages all other networking resources for the entire organization. Developers do not have permissions to make changes to any network or security settings defined by the security and networking team, but they are granted permission to create resources such as virtual machines in shared subnets.
  - As with the first scenario, a shared VPC will be used and the appropriate permissions configured for the three groups network, security, and developers.

Resource:	Organization	
Roles:	Shared VPC Admin Network Admin	
Member:	Network Admin team	

Resource:	Organization	
Roles:	Security Admin Organization Admin	
Member:	Security team	

Resource:	Host Project	This role grants permission to use subnets that the shared VPC has shared.
Role:	Network user	
Member:	Developers	

Resource:	Service project	Note this role allows the permission to use External IP addresses. See the note below for guidance on how to prevent this action.
Role:	compute.instanceAdmin	
Member:	Developers	

- For this scenario you need three separate IAM policies: one for the organization, one for the host project, and one for the service projects.
- The first IAM policy, which needs to be attached at the organization level, grants the network team the roles they need to administer shared VPC host projects and to manage all network resources. This includes the ability to associate service projects with the host project. The network admin role also grants the network team the ability to view but not modify firewall rules. It also grants the security team the ability to set IAM policies and manage firewall rules and SSL certificates in all projects in the organization.
    - Network Team: Grant them the xpn (share vpc) admin role, network admin role to create and share vpc

      role: roles/compute.xpnAdmin, roles/compute.networkAdmin
    - Security Team: Grant security admin and resource manager roles to set iam policies and manage firewall rules and SSL certss

      role: roles/compute.securityAdmin, roles/resourcemanager.organization
- The second IAM policy is at the host level, and enable the developers to use the shared networks and shared vpc
    - Developers: roles/compute.networkUser
- The 3rd IAM policy is at each service project level, which allows developers using the project to manage instances in the service project and to use shared subnets in the host project.
- You could place all service projects in a folder and set this particular policy at that level of the hierarchy. This would allow all projects created in that folder to inherit the permissions set at the folder within which the service project is created.
    - Developers: roles/compute.networkUser, roles/compute.instanceAdmin

- Each team can manage its own network
    - A digital native wants to give their development teams the ability to work in an autonomous manner. They have no central IT admin teams and trust their teams to manage all aspects of their projects.
    - Despite this, they equally want to be able to put in place some loose controls to allow them to adopt a more formal set-up as they grow and their product goes GA.
    - To implement this scenario, each team of developers is assigned its own folder. This structure ensures that individual projects created under the folder inherit the appropriate permissions, while allowing each team to work independently. Each team should still follow the principle of least privilege when it sets IAM policies for its own resources.
    - Even though it will initially be the same team members who will be managing the network resources and the actual resources in the projects, creating separate groups for the logical duties is best practice.
    - This approach facilitates limiting access to those resources that temporary staff need or maybe new staff that need training up before they can modify network resources. It also allows the ability to change who has access to what resources without having to modify the IAM policy every time a personnel change occurs.

Resource:	Folder	A service account can be used to create and own projects.
Roles:	Project creator Folder Admin	
Member:	Dev Teamleads Service account	

Note: Refer to IAM roles for billing-related job functions for the IAM settings to allow a service account or user to associate a project with a billing account.

Resource:	Folder	
Roles:	Network Admin Security Admin	
Member:	Network & security team	

Resource:	Folder	These roles allow the developers to manage all aspects of BigQuery and Compute engine.
Roles:	Instance Admin BigQuery Admin	
Member:	Developers	

- This requires an IAM policy bound at each team's allocated folder.
  - DevteamLeads01: roles/resourcemanager.foldersAdmin, roles/resourcemanager.projectCreator
  - Security team: roles/compute.securityAdmin, roles/compute.networkAdmin, roles/compute.instanceAdmin
  - Devteam: roles/compute.instanceAdmin, roles/bigquery.admin

SSH to Compute engine

Thursday, January 28, 2021 2:18 AM

SSH into VM from gcloud shell:

- If you try to ssh into a vm from gcloud using `gcloud compute ssh --project=PROJECT_ID --zone=ZONE VM_NAME` or `gcloud compute ssh <instance name>`
  , gcloud will add it to its know hosts file, but wont be able to access the vm, as its key has not been added to the vm. It looks like:
  - srinivassr_mail@cloudshell:~ (firstproject-303021)$ ssh 34.75.214.113
  - The authenticity of host '34.75.214.113 (34.75.214.113)' can't be established.
  - ECDSA key fingerprint is SHA256:pmlMhxaCoHdAtmngLNXPvIH/x9m5KylmKNZSOwDM2J0.
  - Are you sure you want to continue connecting (yes/no)? yes
  - Warning: Permanently added '34.75.214.113' (ECDSA) to the list of known hosts.
  - srinivassr_mail@34.75.214.113: Permission denied (publickey).
- The first time you attempt ssh into vm, it will see that the gcloud doesn't have a private and public key for that instance, so it will run key-gen and show:
  - WARNING: The private SSH key file for gcloud does not exist.
  - WARNING: The public SSH key file for gcloud does not exist.
  - WARNING: You do not have an SSH key for gcloud.
  - WARNING: SSH keygen will be executed to generate a key.
  - Generating public/private rsa key pair.
- Once the public key (<key name>.pub) and private key (only <key name>) have been generated for gcloud, it will update the project metadata with the public key. This is similar to going inside the console->Compute engine->Metadata->ssh keys->edit->add keys and adding the public key
  The output will look like
  - Updating project ssh metadata...::Updated [https://www.googleapis.com/compute/v1/projects/firstproject-303021].
  - Updating project ssh metadata...done.
- Once the public key has been added to the project metadata, it will send this key to the compute instance to authenticate ssh request. Its output looks like:
  - Waiting for SSH key to propagate.

To check ssh-keys in metadata for any project or instance you can run : curl -H Metadata-Flavor:Google metadata.google.internal/computeMetadata/v1/instances/ssh-keys

SSH into VM from external machine/PC:

There are 2 ways to SSH into your gcp instance from an external pc. Of these 2, OS login is the preferred method in scenarios when you want to handle ssh authentication for multiple users on many projects. Maintaining and saving ssh-keys manually is not a good practice, hence manual ssh key handling is good only for individual projects. OS Login is not currently supported in Google Kubernetes Engine (GKE)/Microsoft servers/SQL servers. GKE cluster nodes continue to use metadata SSH keys when OS Login is enabled. RDP for Microsoft servers.

To perform this task, you must have the following permissions:

- `compute.instances.setMetadata` on the instance if setting metadata on the instance
- `compute.projects.setCommonInstanceMetadata` on the project if setting project-wide metadata

- `iam.serviceAccounts.actAs` on the project if setting project-wide metadata

1. **OS login method:**
    a. Enabling/disabling os login:
        i. Gcloud: Enable os-login either at the project level to all instances or at each instance level using either gcloud compute project-info add-metadata \ --metadata enable-oslogin=TRUE or gcloud compute instances add-metadata <vm name> \ --metadata enable-oslogin=TRUE
        ii. Console: In compute engine->metadata->edit/custom metadata-> set key enable-oslogin value True to enable False to disable. For entire project level do this at the organization level and choose project, or for instance level do it from inside the project the instance lies in.
        iii. Alternatively you can also create instances with enable-oslogin true, by adding it in the create vm page or by gcloud compute instances create <vm name> \ --metadata enable-oslogin=TRUE
    b. Add IAM roles to user you want oslogin to accept, else they wont get the correct ssh access
        i. Grant one of the following instance access roles in IAM->User name.
            1. roles/compute.osLogin, which doesn't grant administrator permissions
            2. roles/compute.osAdminLogin, which grants administrator permissions
        ii. If your VM instance uses a service account, then each user must be configured to have the roles/iam.serviceAccountUser role on the service account. This is useful if your app (in GCE) requires ssh access. For outside apps manually configure the service account for it.
        iii. For users that are outside of your organization to access your VMs, in addition to granting an instance access role, grant the roles/compute.osLoginExternalUser role. This role must be granted at the organization level by an organization administrator.
    c. Create ssh keys associated with that username: You need to create and provide your ssh public keys to gcloud so that it can use those to allow your username to login
        i. Open a terminal on your workstation and use the ssh-keygen command to generate a new key. Specify the -C flag to add a comment with your username.
            - ssh-keygen -t rsa -f ~/.ssh/<KEY_FILENAME> -C <USERNAME>
            - where: <KEY_FILENAME> is the name that you want to use for your SSH key files. For example, a filename of my-ssh-key generates a private key file named my-ssh-key and a public key file named my-ssh-key.pub. <USERNAME> is the username for the user connecting to the instance.
        ii. This command generates a private SSH key file and a matching public SSH key (at ~/.ssh/<KEY_FILENAME>.pub) with the following structure: ssh-rsa <KEY_VALUE> <USERNAME> where: <KEY_VALUE> is the key value generated by ssh-keygen. It is a long string of characters. <USERNAME> is the user that created the key. You can modify this value to be more descriptive.
        iii. Restrict access to your private key so that only you can read it and nobody can write to it.
            - chmod 400 ~/.ssh/[KEY_FILENAME> where <KEY_FILENAME> is the name that you used for your SSH key files.
        iv. Repeat this process for every user who needs a new key. Before uploading to gcp, make Then, locate the public SSH keys that you made and any existing public SSH keys that you want to add to a project or instance.
        v. Add the public key to your gcloud using gcloud compute os-login ssh-keys add --key-file .ssh/id_rsa.pub and do the same for all usernames
    d. SSH from external pc using command: ssh -i <path to private key> username@external_ip
        1. Username can be google identity that they use to login to gcp

1. **Manual SSH key handling:**
    a. Create ssh public and private keys on client PC by running ssh-keygen -t rsa -f ~/.ssh/<KEY_FILENAME> -C

<USERNAME>

b. Inside GCP add your ssh public key by going inside the console->Compute engine->Metadata->ssh keys->edit->add keys and adding the public key
c. This can be done at the project or instance level
d. SSH from external pc using command: ssh -i <path to private key> username@external_ip

Adding or removing project-wide public SSH keys

Use project-wide public SSH keys to give users general access to a Linux instance. Project-wide public SSH keys give users access to all of the Linux instances in a project that allow project-wide public SSH keys. If an instance blocks project-wide public SSH keys, a user can't use their project-wide public SSH key to connect to the instance unless the same public SSH key is also added to instance metadata.

Note: Google Cloud creates a project-wide SSH key by default.

Note: To block/allow project wide ssh keys from accessing the instance run, `gcloud compute instances add-metadata [INSTANCE_NAME] --metadata block-project-ssh-keys=TRUE/False`

- **To add or remove instance-level public SSH keys with the gcloud tool:**
  - If your instance already has instance-level public SSH keys, get those public SSH keys from metadata:
  - Get the existing metadata for the instance:
    - gcloud compute instances describe [INSTANCE_NAME]
  - From the output, find the ssh-keys metadata value:
    - Note: The ssh-keys metadata value doesn't appear if your instance doesn't have existing instance-level public SSH keys.
    - ...
    - metadata:
    - fingerprint: QCofVTHlggs=
    - items:
    - ...
    - - key: ssh-keys
    - value: |-
    - [USERNAME_1]:ssh-rsa [EXISTING_KEY_VALUE_1] [USERNAME_1]
    - [USERNAME_2]:ssh-rsa [EXISTING_KEY_VALUE_2] [USERNAME_2]
    - ...
    - where:
    - [USERNAME_1] and [USERNAME_2] are the usernames for your existing keys.
    - [EXISTING_KEY_VALUE_1] and [EXISTING_KEY_VALUE_2] are public key values that are already applied to your instance.
    - Note: If a public SSH key has an expiration time, that key has a slightly different format than the keys in this example.
  - Copy the public SSH keys under the ssh-keys metadata value.
  - Create and open a new text file on your local workstation.
  - In the file, create a list of all of the public SSH keys that you want to add or keep in instance-level metadata. If you currently have public SSH keys in instance-level metadata, any keys that you don't include in your list are removed.
    - For example, the sample list below removes the key for [USERNAME_1] because their SSH key is omitted. It also keeps the SSH key for [USERNAME_2] and adds the SSH key for [USERNAME_3] because their SSH keys are included in the list.
    - [USERNAME_2]:ssh-rsa [EXISTING_KEY_VALUE_2] [USERNAME_2]
    - [USERNAME_3]:ssh-rsa [NEW_KEY_VALUE] [USERNAME_3]
    - where:

- [USERNAME_1],[USERNAME_2], and [USERNAME_3] are the usernames of the public SSH keys.
- [EXISTING_KEY_VALUE_1] is a public key value for an SSH key that you want to remove.
- [EXISTING_KEY_VALUE_2] is a public key value for an SSH key that you want to keep.
- [NEW_KEY_VALUE] is a public key value for an SSH key that you want to add.

- Save and close the file.
- In the command prompt, use the compute instances add-metadata command to set the instance-only ssh-key value. Include the --metadata-from-file flag and specify the path to the public key file list that you made.
  - gcloud compute instances add-metadata [INSTANCE_NAME] --metadata-from-file ssh-keys=[LIST_PATH]
  - where:
  - [INSTANCE_NAME] is the name of the instance where you want to apply the public SSH key file.
  - [LIST_PATH] is the path to your list of public SSH keys.

SSH auth process:

Authentication using SSH key pairs begins after the symmetric encryption has been established as described in the last section. The procedure happens like this:

1. The client begins by sending an ID for the key pair it would like to authenticate with to the server.
2. The server check's the authorized_keys file of the account that the client is attempting to log into for the key ID.
3. If a public key with matching ID is found in the file, the server generates a random number and uses the public key to encrypt the number.
4. The server sends the client this encrypted message.
5. If the client actually has the associated private key, it will be able to decrypt the message using that key, revealing the original number.
6. The client combines the decrypted number with the shared session key that is being used to encrypt the communication, and calculates the MD5 hash of this value.
7. The client then sends this MD5 hash back to the server as an answer to the encrypted number message.
8. The server uses the same shared session key and the original number that it sent to the client to calculate the MD5 value on its own. It compares its own calculation to the one that the client sent back. If these two values match, it proves that the client was in possession of the private key and the client is authenticated.

Compute engine

Friday, February 12, 2021 11:42 AM

- When choosing boot disk for a compute engine, you can choose from multiple options what the image attached to the disk is:
Scenarios	Machine image	Persistent disk snapshot	Custom image	Instance template
Single disk backup	Yes	Yes	Yes	No
Multiple disk backup	Yes	No	No	No
Differential backup	Yes	Yes	No	No
Instance cloning and replication	Yes	No	Yes	Yes
VM instance configuration	Yes	No	No	Yes

  - Machine Image: A machine image is a Compute Engine resource that stores all the configuration, metadata, permissions, and data from one or more disks required to create a virtual machine (VM) instance. You can use a machine image in many system maintenance scenarios, such as instance creation, backup and recovery, and instance cloning. You have to create the instance from within this image, not the GCE menu.
  - Persistent disk snapshot: Create snapshots to periodically back up data from your zonal persistent disks or regional persistent disks. You can create snapshots from disks even while they are attached to running instances. Snapshots are global resources, so you can use them to restore data to a new disk or instance within the same project. You can also share snapshots across projects.
  - Instance templates: Instance templates define the machine type, boot disk image or container image, labels, and other instance properties. You can then use an instance template to create a MIG or to create individual VMs. Instance templates are a convenient way to save a VM instance's configuration so you can use it later to create VMs or groups of VMs.
  - Custom Images: A custom image is a boot disk image that you own and control access to. You can clone images from other GCE images, your own on-prem images. You can copy an image into another, by using the first image as the source for the 2nd image.
  - Public images: Standard linux and windows OS images provided by gcp to everyone.
- A VM instance's availability policy determines how it behaves when an event occurs that requires Google to move your VM to a different host machine. For example, you can choose to keep your VM instances running while Compute Engine live migrates them to another host or you can choose to terminate your instances instead.
  - You can update an instance's availability policy at any time to control how you want your VM instances to behave.
  - You can change an instance's availability policy by configuring the following two settings:
    - The VM instance's maintenance behavior, which determines whether the instance is live migrated or terminated when there is a maintenance event.
    - The instance's restart behavior, which determines whether the instance automatically restarts if it crashes or gets terminated.
  - The default maintenance behavior for instances is to live migrate, but you can change the behavior to terminate your instance during maintenance events instead.
  - Configure an instance's maintenance behavior and automatic restart setting using the onHostMaintenance and automaticRestart properties.
  - All instances are configured with default values unless you explicitly specify otherwise.
    - onHostMaintenance: Determines the behavior when a maintenance event occurs that might cause your instance to reboot.
      - [Default] migrate, which causes Compute Engine to live migrate an instance when there is a maintenance event.
      - terminate, which terminates an instance instead of migrating it.

        - automaticRestart: Determines the behavior when an instance crashes or is terminated by the system.
          [Default] true, so Compute Engine restarts an instance if the instance crashes or is terminated.
          false, so Compute Engine does not restart an instance if the instance crashes or is terminated.

- **Snapshots and images:**
    - An image is a complete backup of your server including all volumes (and OS).
        - A snapshot can be done from a specific volume (for example you have a server with a volume containing the OS and another one containing the application data, and you want to use different snapshot strategies on both volumes) but mostly not OS.
    - You need to have a persistent made before you can create a snapshot or an image.
    - Create snapshots to periodically back up data from your zonal persistent disks or regional persistent disks.
    - You can create snapshots from disks even while they are attached to running instances. Snapshots are global resources, so you can use them to restore data to a new disk or instance within the same project. You can also share snapshots across projects.
    - If you create a snapshot of your persistent disk while your application is running, the snapshot might not capture pending writes that are in transit from memory to disk. Because of these inconsistencies, the snapshot might not reflect the exact state of your application at the time you captured the snapshot.
    - You cannot edit a snapshot schedule. To change a schedule that is already attached to a disk, you must first detach the schedule from the disk and delete it. Then you can create a new schedule, and attach it to the disk.
    - If you delete a snapshot schedule, all auto-generated snapshots associated with the snapshot schedule are kept permanently. However, after the schedule is deleted, it can no longer generate snapshots.
    - You can create disk images from the following sources:
        - A persistent disk, even while that disk is attached to an instance
        - A snapshot of a persistent disk
        - Another image in your project
        - An image that is shared from another project
        - A compressed RAW image in Cloud Storage

- Hence creation hierarchy is Image created-->Can create snapshot-->Can create image

- **Deprecating an image**
    - Compute Engine lets you deprecate a custom image that you own by setting the deprecation status on the image.
    - Each deprecation status causes a different response from the server, helping you transition users away from unsupported images in a manageable way.
    - Use the Google Cloud Console, the gcloud command-line tool, or the Compute Engine API method to deprecate an image.
        - gcloud compute images deprecate <IMAGE_NAME> \ --state <STATE> \ --replacement <REPLACEMENT image>
    - Deprecation states: The following deprecation states are supported:
        - ACTIVE: The image is active and can be used as normal. Image families point to the most recent and active image in a family.
        - DEPRECATED: The image is marked deprecated but can still be used. If you use an image that is deprecated, the request succeeds with a warning. New links to this image are allowed. Image families no longer point to this image even if it is the most recent image in the family.
        - OBSOLETE: The image is marked obsolete and is no longer available for use. An error message is returned if you try to use this image in a request. Existing links to this image are still allowed.
        - DELETED: This image is deleted. An error message is returned if you try to use a deleted image.

- You can revert a deprecation (make an image active again), by changing the deprecation state to ACTIVE.
- You can only delete custom images that you, or someone who has access to the project, have added. Use the Google Cloud Console, `gcloud` command-line tool, or the Compute Engine API method to delete the image.
  - gcloud compute images delete <IMAGE_NAME>

- **Setting image versions in an image family**
  - Use image families to simplify image versioning. Add an image to an image family to set it as the most recent image version.
  - If you determine that you must roll back the image family to a previous image version, deprecate the most recent image in the family.
  - Optionally, you can specify the image's storage location by using the Google Cloud Console, the gcloud compute images create command with the --storage-location flag, or the images().insert method.
  - gcloud compute images create image-v1 \

    --source-disk disk-1 \

    --source-disk-zone us-central1-f \

    --family my-image-family

- **Managing access to custom images**
  - Allows users/projects to create, delete, use or share custom images, but you need to grant them the correct IAM role
  - To grant ability to create custom images in org or project, use the predefined 'Storage admin role'. If you want to grant only image creation access you need to create a custom role but it must include the compute.images.<create/list>: create or list images or compute.disks.<use/list>: create image from disk or list disks
    - If someone has granted you access to one or more images in another project, you can access these images in the project by specifying the image project in your requests.
    - gcloud compute instances create test-instance \ --image database-image-a --image-project database-images
  - To grant ability to delete images add the compute.images.<delete,get,list> role
  - To grant image sharing access you use 2 predefined roles:
    - roles/compute.imageUser: Permissions to list, read and use images without any other permissions
    - roles/compute.storageAdmin: Permissions to create, modify and delete disk and snapshots
  - You can't grant a user the role of allUsers on images.
  - You can only share resources, such as images, with all authenticated users; you cannot share projects or organizations with all authenticated users. This restriction, and the resource hierarchy, helps prevent an organization from inadvertently sharing their entire project with all authenticated Compute Engine users.
  - To restrict what users can do with the shared images, you can create an yaml file with the restrictions and add it to the IAM role constraints/compute.storageResourceUseRestrictions which would apply to your users, at the project level.
  - Note: You can set the constraints/compute.storageResourceUseRestrictions constraint only at the organization level.
    - gcloud beta resource-manager org-policies set-policy \ --organization organization-id org-policy.yaml

- When to use a machine image: The following table compares the use of machine images, persistent disk snapshots, instance templates, and custom images.

Scenarios	Machine image	Persistent disk snapshot	Custom image	Instance template
Single disk backup	Yes	Yes	Yes	No
Multiple disk backup	Yes	No	No	No

Differential backup	Yes	Yes	No	No
Instance cloning and replication	Yes	No	Yes	Yes
VM instance configuration	Yes	No	No	Yes

- From the preceding table, you can see that machine images are the most ideal resources for the following use cases:
  - Disk backups
  - Instance cloning and replication

Managed Instance Group (MIG):

- Instance templates are immutable i.e you can't edit an existing template. You can duplicate/copy it into another template and then edit that.

- Instance groups: Are used to add multiple instance created from a instance template, into 1 system.
  - Managed groups allow autoscaling and is used for multiple similar instances.
  - Unmanaged groups is for dissimilar instances and doesn't support autoscaling though it does load balancing. Unmanaged instance group cannot be multi-zone.

- Note: With load balancing alone, you'll have to know ahead of time how much capacity you need so you can keep additional instances running and registered with the load balancer to serve higher loads. Or you could manually stop worrying about it and auto scale based on say CPU usage so that instances increase or decrease dynamically based on the load.

- Deleting a managed instance group will delete all the instances it had created, since it owns them. However deleting an unmanaged instance group wont delete any instances.

- maxSurge specifies the maximum number of instances that can be created over the desired number of instances. If maxSurge is set to 0, the rolling update cannot create additional instances and is forced to update existing instances resulting in a reduction in capacity. Therefore, it does not satisfy our requirement to ensure that the available capacity does not decrease during the deployment.
- maxUnavailable - specifies the maximum number of instances that can be unavailable during the update process. When maxUnavailable is set to 1, the rolling update updates 1 instance at a time. i.e. it takes 1 instance out of service, updates it, and puts it back into service. This option results in a reduction in capacity while the instance is out of service. Example - if we have 10 instances in service, this combination of setting results in 1 instance at a time taken out of service for an upgrade while the remaining 9 continue to serve live traffic. That's a reduction of 10% in available capacity and does not satisfy ou r requirement to ensure that the available capacity does not decrease during the deployment.
- Manually setting the size of a MIG
  - If a managed instance group is not already set to automatically scale, you can resize the group manually to change the number of instances.
  - If you increase the size, the managed instance group uses the current instance template to add new instances.
  - If you decrease the size, the managed instance group deletes VMs from the group. The group deletes instances with a `currentAction` of `DELETING`, `CREATING`, and `RECREATING` before it deletes instances that are running with no scheduled actions.
- To remove an instance from the instance group *without deleting the instance*, use the `abandon-instances` command. Once instances have been abandoned, the currentSize of the group is automatically reduced as well to reflect the change.
  - Abandoning an instance does not reboot or delete the underlying virtual machine instances, but just removes the instances from the instance group. If you would like the delete the underlying instances, use the `delete-instances` command instead.

Scaling a managed instance:

- Each autoscaling policy can scale according to a target utilization metric or a specified schedule.

- Target based autoscaling:
  - Scaling based on **CPU** utilization: You can autoscale based on the average CPU utilization of a managed instance group (MIG).
    - gcloud compute instance-groups managed set-autoscaling example-managed-instance-group \

      --max-num-replicas 20 \

      --target-cpu-utilization 0.60 \

      --cool-down-period 90
  - Scaling based on the serving capacity of an external **HTTP(S)** load balancer
    - This is autoscaling in conjunction with loadbalancing.
    - This means that autoscaling adds or removes VM instances in the group when the load balancer indicates that the group has reached a configurable fraction of its fullness, where fullness is defined by the target capacity of the selected balancing mode of the backend instance group.
    - The loadbalancer offers 2 balancing mode: Utilization and Rate
      - Utilization specifies the max target for avg backend utilization. Rate specifies target number of requests per second on a per-instance or per-group basis
    - Autoscaling does not work with maximum requests per group because this setting is independent of the number of instances in the instance group. The load balancer continuously sends the maximum number of requests per group to the instance group, regardless of how many instances are in the group.
  - Scaling based on **Cloud Monitoring metrics**
    - Scale based on monitored metrics provided on a per-instance basis or a per-group basis, and handled in Cloud Monitoring (but not detected by cloud monitoring logs)
    - These can be cloud monitoring based or custom built metrics.
    - Per-instance metrics: Per-instance metrics provide data for each VM in a MIG separately, indicating resource utilization for each instance. When using per-instance metrics, the MIG cannot scale below a size of 1 VM because the autoscaler requires metrics about at least one running VM in order to operate.
    - Valid utilization metric for scaling meets the following criteria:
      - The standard metric must contain data for a gce_instance monitored resource. You can use the timeSeries.list API call to verify whether a specific metric exports data for this resource.
      - The standard metric describes how busy an instance is, and the metric value increases or decreases proportionally to the number of VMs in the group.
    - Per-group metrics: Per-group metrics allow autoscaling with a standard or custom metric that does not export per-instance utilization data. Instead, the group scales based on a value that applies to the whole group and corresponds to how much work is available for the group or how busy the group is. The group scales based on the fluctuation of that group metric value and the configuration that you define.
- Scaling based on schedules:
  - You can use schedule-based autoscaling to allocate capacity for anticipated loads.
  - For each scaling schedule, specify the following:
    - Capacity: minimum required VM instances
    - Schedule: start time, duration, and recurrence (for example, once, daily, weekly, or monthly)
  - Each scaling schedule is active from its start time and for the configured duration. During this time, autoscaler scales the group to have at least as many instances as defined by the scaling schedule.
- Cool down period:
  - The cool down period is also known as the application initialization period. Compute Engine uses the cool down period for scaling decisions in two ways:

- To omit unusual usage data after a VM is created and while its application is initializing.
- If predictive autoscaling is enabled, to inform the autoscaler how much time in advance to scale out ahead of anticipated load, so that applications are initialized when the load arrives.
- Specify a cool down period to let your instances finish initializing before the autoscaler begins collecting usage information from them. By default, the cool down period is 60 seconds.

- Stabilization period
  - For the purposes of scaling in, the autoscaler calculates the group's recommended target size based on peak load over the last 10 minutes. These last 10 minutes are referred to as the stabilization period.
  - Using the stabilization period, the autoscaler ensures that the recommended size for your managed instance group is always sufficient to serve the peak load observed during the previous 10 minutes.
  - This 10-minute stabilization period might appear as a delay in scaling in, but it is actually a built-in feature of autoscaling. The delay ensures that the smaller group size is enough to support peak load from the last 10 minutes
- Predictive autoscaling
  - If you enable predictive autoscaling to optimize your MIG for availability, the autoscaler forecasts future load based on historical data and scales out a MIG in advance of predicted load, so that new instances are ready to serve when the load arrives.
  - Predictive autoscaling works best if your workload meets the following criteria:
  - Your application takes a long time to initialize—for example, if you configure a cool down period of more than 2 minutes.
  - Your workload varies predictably with daily or weekly cycles.
- Scale-in controls
  - If your workloads take many minutes to initialize (for example, due to lengthy installation tasks), you can reduce the risk of response latency caused by abrupt scale-in events by configuring scale-in controls. Specifically, if you expect load spikes to follow soon after declines, you can limit the scale-in rate to prevent autoscaling from reducing a MIG's size by more VM instances than your workload can tolerate.
  - To configure scale-in controls, set the following properties in your autoscaling policy.
  - Maximum allowed reduction: The number of VM instances that your workload can afford to lose (from its peak size) within the specified trailing time window.
  - Trailing time window: Define how long the autoscaler should wait before removing instances, as defined by the maximum allowed reduction. With a longer trailing time window, the autoscaler considers more historical peaks, making scale-in more conservative and stable.

Updating a managed instance group:

- A managed instance group contains one or more virtual machine instances that are controlled using an instance template.
- To update instances in a managed instance group, you can make update requests to the group as a whole, using the Managed Instance Group Updater feature.
  - The Managed Instance Group Updater allows you to easily deploy new versions of software to instances in your managed instance groups, while controlling the speed of deployment, the level of disruption to your service, and the scope of the update.
  - The Updater offers two primary advantages:
    - The rollout of an update happens automatically to your specifications, without the need for additional user input after the initial request.
    - You can perform partial rollouts which allows for canary testing.
  - By allowing new software to be deployed inside an existing managed instance group, there is no need for you to reconfigure the instance group or reconnect load balancing, autoscaling, or autohealing each time new version of software is rolled out.

- Without the Updater, new software versions must be deployed either by creating a new managed instance group with a new software version, requiring additional set up each time, or through a manual, user-initiated, instance-by-instance recreate.
  - Both of these approaches require significant manual steps throughout the process.
- A rolling update is an update that is gradually applied to all instances in an instance group until all instances have been updated. You can control various aspects of a rolling update, such as how many instances can be taken offline for the update, how long to wait between updating instances, whether the update affects all or just a portion of instances, and so on.
- There is no explicit command for rolling back an update to a previous version, but if you decide to roll back an update (either a fully committed update or a canary update), you can do so by making a new update request and passing in the instance template that you want to roll back to.

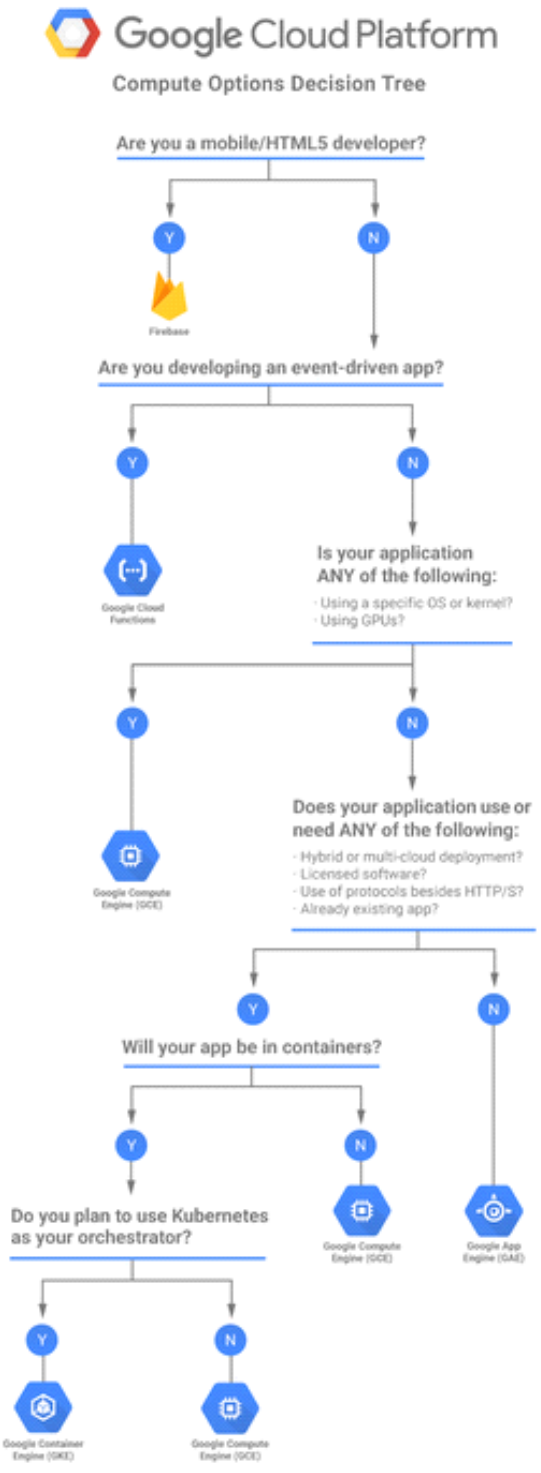
Machine types

- E2 machine types are cost-optimized VMs that offer up to 32 vCPUs with up to 128 GB of memory with a maximum of 8 GB per vCPU. E2 machines have a predefined CPU platform running either an Intel or the second generation AMD EPYC Rome processor. E2 VMs provide a variety of compute resources for the lowest price on Compute Engine, especially when paired with committed-use discounts.
- N2 machine types offer up to 80 vCPUs, 8 GB of memory per vCPU, and are available on the Intel Cascade Lake CPU platforms.
  - N2D machine types offer up to 224 vCPUs, 8 GB of memory per vCPU, and are available on second generation AMD EPYC Rome platforms.
  - N1 machine types offer up to 96 vCPUs, 6.5 GB of memory per vCPU, and are available on Intel Sandy Bridge, Ivy Bridge, Haswell, Broadwell, and Skylake CPU platforms.
- Shared-core machine types are available in the E2 and N1 families. These machine types timeshare a physical core. This can be a cost-effective method for running small, non-resource intensive applications.
  - E2: e2-micro, e2-small, and e2-medium shared-core machine types have 2 vCPUs available for short periods of bursting.
  - N1: f1-micro and g1-small shared-core machine types have up to 1 vCPU available for short periods of bursting.

E2 **General purpose**	**N2, N2D, N1** **General purpose**	**M2, M1** **Memory-Optimized**	**C2** **Compute-Optimized**	**A2** **Accelerator-Optimized**
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 applications • Small-medium databases • Microservices • Virtual desktops • Development environments	• Web serving • App serving • Back office applications • Medium-large databases • Cache • Media/streaming	• Large in-memory databases like SAP HANA • In-memory analytics	• HPC • Electronic Design Automation (EDA) • Gaming • Single-threaded applications	• High performance computing (HPC) • Machine learning (ML) • Massive parallelized computation
• Start with e2-std-2 with 2 vCPUs and 8gb ram. Scales upto e2-std-32 with 32 vCPUs and 128gb ram. • e2-highcpu-2 starts with 2vCPUs and 2gb ram and goes upto e2-highcpu-32 with 32 vCPUs and 32gb	• N2-std-2 starts with 2 vCPU and 8gb ram, goes upto n2-std-80 (just after std-64) with 80 vCPU and 320gb ram. • N2-highmem-2 starts with 2 vCPU and 16gb ram and goes upto n2-std-80 with	• M1-megamem-96 is 96 vCPU and 1.4TB ram • M1-ultramem-40 starts with 40 vCPU and 961gb ram and goes upto 160 vCPU and 3.75TB ram	• C2-std-4 starts with 4 vCPU and 16gb ram and goes upto 60 vCPU and 240gb ram (vCPU becomes 30 instead of 32 and ram become 120 instead of 128)	• A2-highcpu-1g starts with 12vCPU and 85gb ram and goes upto 96vCPU and 680gb ram • A2-megacpu-16g is 96 vCPU and 1.33TB ram

ram.	80 vCPU and 640gb ram • N2-highcpu-2 starts with 2 vCPU and 2gb ram and goes upto 80 vCPU and 80gb ram • N1-std-1 goes from 1 vCPU and 3.75 gb ram to n1-std-96 with 96 vCPU and 360gb ram (goes 360 from 240gb) • N1-highmem-2 goes from 2 vCPU and 13gb ram to n1-std-96 with 96 vCPU and 624gb ram (goes 624 after 416gb ram) • N1-highcpu-2 starts with 2vCPU and 1.8gb ram and goes upto n1-std-96 with 96 vCPU and 86.4gb ram			
Custom type: vCPU has to be multiples of 2 (starting with 2-32) . Ram has to be atleast anywhere from 0.5 to 8gb per vCPU (max 128gb)	For N2: Multiples of 2 upto 30 vCPU and for 32 and higher divisible by 4. Ram is 0.5-8gb per vCPU For N2D: Multiples of 2 upto 16 vCPUs after which increment by 16 upto 96. Ram is the same. For N1: The number of vCPUs must be a multiple of 2, up to 96 vCPUs for Skylake platform, or up to 64 vCPUs for Broadwell, Haswell, or Ivy Bridge CPU platforms. Ram is 0.9-6.5gb per vCPU			

- **<u>Disks, Images and Snapshots:</u>**
-


Google Cloud Platform
Compute Options Decision Tree
Are you a mobile/HTML5 developer?
Y
N
Firebase
Are you developing an event-driven app?
Y
N
Google Cloud Functions
Is your application ANY of the following:
· Using a specific OS or kernel?
· Using GPUs?
Y
N
Google Compute Engine (GCE)
Does your application use or need ANY of the following:
· Hybrid or multi-cloud deployment?
· Licensed software?
· Use of protocols besides HTTP/S?
· Already existing app?
Y
N
Will your app be in containers?
Y
N
Google Compute Engine (GCE)
Google App Engine (GAE)
Do you plan to use Kubernetes as your orchestrator?
Y
N
Google Container Engine (GKE)
Google Compute Engine (GCE)

Networking, VPC and Firewalls

Monday, January 6, 2020 1:48 AM

Networking:

Access/Scope Level of VPC ->

- VPC networks - **Global**
- Routes - **Global**
- Firewall rules - **Global**
- Subnets - **Regional**

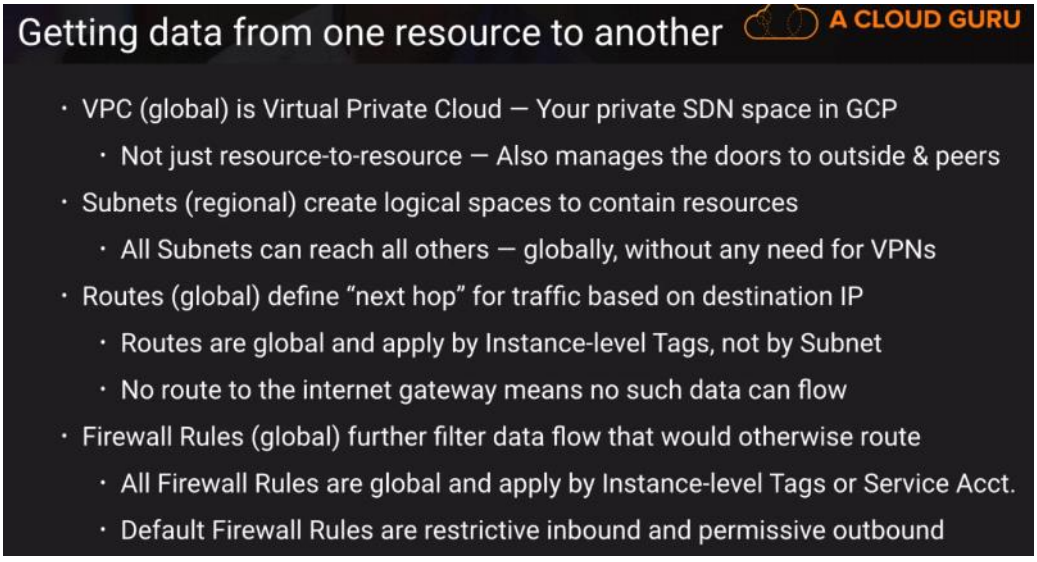
- gcloud compute networks create testvpc --project=<project name> --subnet-mode=custom --mtu=1460 --bgp-routing-mode=regional
- gcloud compute networks subnets create subnet1 --project=<project name> --range=192.168.0.0/24 --network=testvpc --region=us-east1

- gcloud compute networks subnets expand-ip-range NAME --prefix-length=PREFIX_LENGTH [--region=REGION] [GCLOUD_WIDE_FLAG …]

- Google provides premium and standard routing to its services. Premium routing or cold potato, is used when you want your app to reach google cloud services directly on egress. Standard routing or hot potato is used when you traverse the general internet before reaching googles services.
- Reserved IP addresses in a subnet
- Every subnet has four reserved IP addresses in its primary IP range. There are no reserved IP addresses in the secondary IP ranges.

Reserved IP address	Description	Example
Network	First address in the primary IP range for the subnet	`10.1.2.0` in `10.1.2.0/24`
Default gateway	Second address in the primary IP range for the subnet	`10.1.2.1` in `10.1.2.0/24`
Second-to-last address	Second-to-last address in the primary IP range for the subnet that is reserved by Google Cloud for potential future use	`10.1.2.254` in `10.1.2.0/24`
Broadcast	Last address in the primary IP range for the subnet	`10.1.2.255` in `10.1.2.0/24`

- Static IP: Reserve static Ips in projects to assign in resources. You pay for reserved Ips and not used Ips.
- You can increase the default CIDR subnet at any time, but the new subnet must contain the older one i.e the new one should have the older as a subset.
  - gcloud compute networks subnets expand-ip-range NAME --prefix-length=PREFIX_LENGTH [--region=REGION] [GCLOUD_WIDE_FLAG …]
    - Name: Name of subnetwork
    - Prefix-Length: The new prefix length of the subnet
    - `REGION:` Region of the subnetwork to operate on. Is optional if already set in gcloud init



Google Cloud offers two types of VPC networks, determined by their *subnet creation mode*:

- When an *auto mode* VPC network is created, one subnet from each region is automatically created within it. These automatically created subnets use a set of predefined IP ranges that fit within the `10.128.0.0/9` CIDR block. As new Google Cloud regions become available, new subnets in those regions are automatically added to auto mode VPC networks by using an IP range from that block. In addition to the automatically created subnets, you can add more subnets manually to auto mode VPC networks in regions that you choose by using IP ranges outside of `10.128.0.0/9`.
- When a *custom mode* VPC network is created, no subnets are automatically created. This type of network provides you with complete control over its subnets and IP ranges. You decide which subnets to create in regions that you choose by using IP ranges that you specify.

  Gcloud commands to create a custom vpc and adding 2 subnets in different regions:
  - gcloud compute networks create testvpc --project=<project name> --subnet-mode=custom --mtu=1460 --bgp-routing-mode=regional
  - gcloud compute networks subnets create subnet1 --project=<project name> --range=192.168.0.0/24 --network=testvpc --region=us-east1
  - gcloud compute networks subnets create subnet2 --project=<project name> --range=192.168.1.0/24 --network=testvpc --region=us-west1

- Which Dynamic Routing scope you should choose?
  - Each VPC network has an associated dynamic routing mode that controls the behavior of all of its Cloud Routers.
  - A Cloud Router is associated with a specific VPC network and region
  - In the Regional mode, routes learned by the Cloud Router only applies to the subnets existing in the same region in which Cloud Router exists.
  - For Global mode, Let's try to understand with the help of an example: You have your On-prem data center in the Mumbai region and your VM and Cloud Router are in the us-west1 region in GCP and both are able to communicate with each other and Dynamic Routing is regional by default and Dynamic Routing is configured at VPC level, not at Subnet level.
  - Now, you have a requirement to add a subnet in the us-central1 region and a VM in us-central1 region, if you try to communicate with your On-prem Data center, you will not be able to do so. The reason is you have chosen Region Dynamic Routing which means, if you create any new subnet in us-west1, it automatically adds routes to the On-prem data center but if you create a subnet in the us-central1 region, it won't add any routes in the On-prem data center.
  - Now you have to edit your VPC and change the dynamic routing to switch to Global. Your VM in us-central1 will now be able to communicate with the On-prem data center.
- Why you should choose Private Google Access?
  - VM instances that only have internal IP addresses and no external IP address, can use Private Google Access to reach the external IP address of Google services.

Enabling Private Google Access helps you save your egress traffic costs.

- In case you have disabled the Private Google Access, now VM instances can no longer access Google services, they will be able to send traffic within the VPC network, if you still want to access Google services then you have to configure external IP Address.
- Private Google Access has no impact on instances that have external IP addresses too. Instances with an external IP address will access the web. They do not need any explicit setup to send requests to the external IP address of Google Apis and services.
- You can enable Private Google Access on a subnet by subnet basis, not on the whole network, it is a setting for subnets in a VPC network.

- Always blocked traffic: Google Cloud always blocks the traffic that is described in the following table. Your firewall rules *cannot* be used to allow any of this traffic

Always blocked traffic	Applies to
Certain GRE traffic (beta)	• Traffic in Cloud VPN tunnels • Traffic on Cloud Interconnect attachments (VLANs) • Traffic for forwarding rules (load balancing or protocol forwarding) GRE is allowed *within* a VPC network
Protocols other than TCP, UDP, ICMP, AH, ESP, SCTP, and GRE to external IP addresses of Google Cloud resources	The type of resource further limits the protocol. For example, Network TCP/UDP Load Balancing supports only TCP and UDP. Also, a forwarding rule for protocol forwarding only processes a single protocol. Refer to the protocol forwarding documentation for a list of supported protocols.
Egress traffic to TCP destination port 25 (SMTP)	Traffic from: • instances to external IP addresses on the internet • instances to external IP addresses of instances

You can switch a VPC network from auto mode to custom mode. This is a one-way conversion; custom mode VPC networks cannot be changed to auto mode VPC networks.

Shared VPC:

- In any organization you can share VPC among projects. HOST project owns the shared vpc while the projects are granted access.
- 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. Shared VPC networks can be either auto or custom mode, but legacy networks are not supported.
- When a host project is enabled, you have two options for sharing networks:
  - You can share all host project subnets. If you select this option, then any new subnets created in the host project, including subnets in new networks, will also be shared.
  - You can specify individual subnets to share. If you share subnets individually, then only those subnets are shared unless you manually change the list.
- Organization policies and IAM permissions work together to provide different levels of access control. Organization policies enable you to set controls at the organization, folder, or project level.
- If you are an organization policy administrator, you can specify the following Shared VPC constraints in an organization policy:
  - You can limit the set of host projects to which a non-host project or non-host projects in a folder or organization can be attached. The constraint applies when a Shared VPC Admin attaches a service project with a host project. The constraint doesn't affect existing attachments. Existing attachments remain intact even if a policy denies new ones. For more information, see the constraints/compute.restrictSharedVpcHostProject constraint.
  - You can specify the Shared VPC subnets that a service project can access at the project, folder, or organization level. The constraint applies when you create new resources in the specified subnets and doesn't affect existing resources. Existing resources continue to operate normally in their subnets even if a policy prevents new resources from being added. For more information, see the constraints/compute.restrictSharedVpcSubnetworks constraint.

- When defining each Service Project Admin, a Shared VPC Admin can grant permission to use the whole host project or just some subnets:
  - Project-level permissions: A Service Project Admin can be defined to have permission to use all subnets in the host project if the Shared VPC Admin grants the role of compute.networkUser for the whole host project to the Service Project Admin. The result is that the Service Project Admin has permission to use all subnets in all VPC networks of the host project, including subnets and VPC networks added to the host project in the future.
  - Subnet-level permissions: Alternatively, a Service Project Admin can be granted a more restrictive set of permissions to use only some subnets if the Shared VPC Admin grants the role of compute.networkUser for those selected subnets to the Service Project Admin. A Service Project Admin who only has subnet-level permissions is restricted to using only those subnets. After new Shared VPC networks or new subnets are added to the host project, a Shared VPC Admin should review the permission bindings for the compute.networkUser role to ensure that the subnet-level permissions for all Service Project Admins match the intended configuration.
- Shared VPC Admins have full control over the resources in the host project, including administration of the Shared VPC network. They can optionally delegate certain network administrative tasks to other IAM members:
  -

Administrator	Purpose
Network Admin • IAM member in the host project, or • IAM member in the organization	Shared VPC Admin defines a Network Admin by granting an IAM member the *Network Admin* (`compute.networkAdmin`) role to the host project. Network Admins have full control over all network resources except for firewall rules and SSL certificates.
Security Admin • IAM member in the host project, or • IAM member in the organization	A Shared VPC Admin can define a Security Admin by granting an IAM member the *Security Admin* (`compute.securityAdmin`) role to the host project. Security Admins manage firewall rules and SSL certificates.

- The following resources can used a shared vpc:
  - Compute: instances, instance templates, instance groups
  - Kubernetes cluster
  - Cloud run: fully managed service
  - Cloud functions
  - App engine: Standard, flex
  - Internal IP, DNS, Cloud DNS, Load balancers

Implied rules Firewall rules:

- Every VPC network has two implied firewall rules. These rules exist, but are not shown in the Cloud Console:
  - Implied allow egress rule. An egress rule whose action is allow, destination is 0.0.0.0/0, and priority is the lowest possible (65535) lets any instance send traffic to any destination, except for traffic blocked by Google Cloud. A higher priority firewall rule may restrict outbound access. Internet access is allowed if no other firewall rules deny outbound traffic and if the instance has an external IP address or uses a Cloud NAT instance. For more information, see Internet access requirements.
  - Implied deny ingress rule. An ingress rule whose action is deny, source is 0.0.0.0/0, and priority is the lowest possible (65535) protects all instances by blocking incoming connections to them. A higher priority rule might allow incoming access. The default network includes some additional rules that override this one, allowing certain types of incoming connections.

    - The implied rules cannot be removed, but they have the lowest possible priorities. You can create rules that override them as long as your rules have higher priorities (priority numbers less than 65535). Because deny rules take precedence over allow rules of the same priority, an ingress allow rule with a priority of 65535 never takes effect.
- Source can be of type ip addr subnet or source tags or service accounts:
- Target can be of either All instances, tags or service accounts (not multiple)
    - If Target is type All instance: You can choose IP addr. You can choose either tag or service acc
    - If target is type Tag: You can choose IP addr or tag
    - If target is type Service acc: You can choose ip or service

- **Pre-populated rules:**
    - The default network is pre-populated with firewall rules that allow incoming connections to instances. These rules can be deleted or modified as necessary:
    - default-allow-internal
        - Allows ingress connections for all protocols and ports among instances in the network. This rule has the second-to-lowest priority of 65534, and it effectively permits incoming connections to VM instances from others in the same network. This rule allows traffic in 10.128.0.0/9 (from 10.128.0.1 to 10.255.255.254), a range that covers all subnets in the network.
    - default-allow-ssh
        - Allows ingress connections on TCP destination port 22 from any source to any instance in the network. This rule has a priority of 65534.
    - default-allow-rdp
        - Allows ingress connections on TCP destination port 3389 from any source to any instance in the network. This rule has a priority of 65534, and it enables connections to instances running the Microsoft Remote Desktop Protocol (RDP).
    - default-allow-icmp
        - Allows ingress ICMP traffic from any source to any instance in the network. This rule has a priority of 65534, and it enables tools such as ping.
- ***Hierarchical firewall policies*** let you create and enforce a consistent firewall policy across your organization (across multiple VPCs).
    - You can assign hierarchical firewall policies to the organization as a whole or to individual folders.
    - These policies contain rules that can explicitly deny or allow connections, as do Virtual Private Cloud (VPC) firewall rules. In addition, hierarchical firewall policy rules can delegate evaluation to lower-level policies or VPC network firewall rules with a goto_next action.
    - Hierarchical firewall policy rules are defined in a firewall policy resource that acts as a container for firewall rules. The rules defined in a firewall policy are not enforced until the policy is associated with a node (an organization or a folder).
    - VPC firewall rules are evaluated. VPC firewall rules either allow or deny connections.
    - Lower-level rules cannot override a rule from a higher place in the resource hierarchy. This lets organization-wide admins manage critical firewall rules in one place.
    - Target networks
        - You can restrict a hierarchical firewall policy rule to VMs in only specified networks. Specifying the target network in the rule gives you control over which VPC networks are configured with that rule. Combined with goto_next or allow, it lets you create exceptions for specific networks when you want to define an otherwise restrictive policy.
    - Target service accounts
        - You can specify a target service account for a rule. Such rules are applied only to VMs owned by the specified service account. Hierarchical firewall policy rules do not support targeting by instance tags.

Connecting On-prem to GCP:

- **Cloud interconnect:** Connect external networks to gcp. Private to VPC via cloud vpn or dedicated/partner interconnect.
    - Cloud Interconnect provides low latency, highly available connections that enable you to reliably transfer data between your on-premises and Google Cloud Virtual Private Cloud (VPC) networks.
    - Cloud Interconnect connections also provide internal IP address communication, which means internal IP addresses are directly accessible from both networks.
    - Cloud interconnect contains the following:
      Dedicated Interconnect, Partner Interconnect, Direct Peering, and Carrier Peering can all help you optimize egress traffic from your VPC network and can help you reduce your egress costs.
    - Cloud VPN by itself does not reduce egress costs.
    - **Dedicated interconnect:** Direct link between vpc and on-prem system. VLAN is private to vpc in 1 region no public gcp. Link is private but not encrypted.
- **Cloud vpn:** Ipsec vpn to connect to vpc via public internet. Supports static or dynamic routing.
    - "You must configure routes so that Google API traffic is forwarded through your Cloud VPN or Cloud Interconnect connection, firewall rules on your on-premises firewall to allow the outgoing traffic, and DNS so that traffic to Google APIs resolves to the IP range you've added to your routes."
    - "You can use Cloud Router Custom Route Advertisement to announce the Restricted Google APIs IP addresses through Cloud Router to your on-premises network.
    - The Restricted Google APIs IP range is 199.36.153.4/30. While this is technically a public IP range, Google does not announce it publicly. This IP range is only accessible to hosts that can reach your Google Cloud projects through internal IP ranges, such as through a Cloud VPN or Cloud Interconnect connection."
- Public to gcp via external peering i.e **carrier peering (or direct peering)** via partner for lower volume (no SLAs).
    - Similar to Interconnect, except it provides connection to entire google suite I.e gcp, google workspace applications
    - Unless you need to access Google Workspace applications as described in the preceding use case, Partner Interconnect is the recommended way to connect to Google through a service provider.
- **CDN interconnect:** Direct low latency connection to certain CDN. But NOT to gcp CDN.
    - Work with your supported CDN provider to learn what locations are supported and how to correctly configure your deployment to use intra-region egress routes.
    - Typical use cases for CDN Interconnect
        - High-volume egress traffic. If you're populating your CDN with large data files from Google Cloud, you can use the CDN Interconnect links between Google Cloud and selected providers to automatically optimize this traffic and save money.
        - Frequent content updates. Cloud workloads that frequently update data stored in CDN locations benefit from using CDN Interconnect because the direct link to the CDN provider reduces latency for these CDN destinations.
        - For example, if you have frequently updated data served by the CDN originally hosted on Google Cloud, you might consider using CDN Interconnect.

- **Cloud DNS:**
    - Cloud DNS: Useful for public and private DNS service with 100% uptime.
    - Reduces latency. Supports DNSSEC. Fixed fee for service and fee per lookup.
    - Cloud DNS offers both public zones and private managed DNS zones. A public zone is visible to the public internet, while a private zone is visible only from one or more Virtual Private Cloud (VPC) networks that you specify.
    -

Highly Available Global DNS Network

Google Cloud DNS is a **scalable, reliable and managed authoritative Domain Name System (DNS)** service running on the same infrastructure as Google. It has **low latency, high availability and is a cost-effective way to make your applications and services available to your users**. Cloud DNS translates requests for domain names like www.google.com into IP addresses like 74.125.29.101. Cloud DNS is programmable. You can easily publish and manage millions of DNS zones and records using our simple user interface, command-line interface or API.

Load Balancers

Wednesday, February 17, 2021 2:09 PM

- 'gcloud compute backend-services add-backend' adds a backend to a backend service. A backend is a group of VMs or network endpoints that can handle requests sent to a load balancer.

- gcloud compute backend-services add-backend <backend-service-name> ([instance group, region and zone], [network endpoint group region and zone])
  - --balancing mode=<balancing mode>
  - -- capacity-scaler=<scales down target capacity i.e max util, rate or connections>
  - --description=<optional desc>
  - --failover<optional not used if global flag set>
  - --max-utilization=<max utilization target>
  - --global <required if backend is global service>
  - --region <is not global and not mentioned in config>
  - Balancing mode Determines how to measure whether backend can handle additional traffic or not
  - BALANCING_MODE must be one of:
    - CONNECTION: Spreads load based on how many concurrent connections the backend can handle.
    - RATE: Spreads load based on how many HTTP requests per second (RPS) the backend can handle.
    - UTILIZATION: Spreads load based on the backend utilization of instances in a backend instance group.
  - A network endpoint group (NEG) is a configuration object that specifies a group of backend endpoints or services. A common use case for this configuration is deploying services in containers. You can also distribute traffic in a granular fashion to applications running on your backend instances.

	Zonal NEG	Internet NEG	Serverless NEG
Purpose	One or more internal IP address endpoints that resolve to either Compute Engine VM instances or GKE Pods. Is zonal	A single internet-routable endpoint that is hosted outside of Google Cloud. Is global	A single endpoint within Google's network that resolves to an Google Cloud, Cloud Functions or Cloud Run (fully managed) service. Is Regional

- **Load balancer:**
  The type of traffic that you need your load balancer to handle is another factor in determining which load balancer to use:
  - For **HTTP and HTTPS traffic**, use:
    - External HTTP(S) Load Balancing
    - Internal HTTP(S) Load Balancing
    - Can be single or multi-regional
  - For **TCP traffic**, use:
    - TCP Proxy Load Balancing
    - Network Load Balancing

    - Internal TCP/UDP Load Balancing
    - Can be single or multi-regional
  - For **UDP traffic**, use:
    - Network Load Balancing
    - Internal TCP/UDP Load Balancing
    - Is only single regional

- SSL Proxy Load Balancing is intended for non-HTTP(S) traffic. Although SSL Proxy Load Balancing can handle HTTPS traffic, we don't recommend this. You should instead use HTTP(S) Load Balancing for HTTPS traffic. HTTP(S) Load Balancing also does the following, which makes it a better choice in most cases:
  - Negotiates HTTP/2 and SPDY/3.1.
  - Rejects invalid HTTP requests or responses.
  - Forwards requests to different VMs based on URL host and path.
  - Integrates with Cloud CDN.
  - Spreads the request load more evenly among backend instances, providing better backend utilization. HTTPS load balances each request separately, whereas SSL Proxy Load Balancing sends all bytes from the same SSL or TCP connection to the same backend instance.

- The backend instances must allow connections from the load balancer GFE (google front end)/health check ranges. This means that you must create a firewall rule that allows traffic from 130.211.0.0/22 and 35.191.0.0/16 to reach your backend instances or endpoints. These IP address ranges are used as sources for health check packets and for all load-balanced packets sent to your backends.
- All load balancers forward traffic to a backend service which can include backend instances (normal or managed instance groups) or a network endpoint group.
- Health checks ensure that Compute Engine forwards new connections only to instances that are up and ready to receive them. Compute Engine sends health check requests to each instance at the specified frequency; once an instance exceeds its allowed number of health check failures, it is no longer considered an eligible instance for receiving new traffic. Existing connections will not be actively terminated which allows instances to shut down gracefully and to close TCP connections.
  - The health checker continues to query unhealthy instances, and returns an instance to the pool when the specified number of successful checks is met. If all instances are marked as UNHEALTHY, the load balancer directs new traffic to all existing instances.
  - Network Load Balancing relies on legacy HTTP Health checks for determining instance health. Even if your service does not use HTTP, you'll need to at least run a basic web server on each instance that the health check system can query.

- Internal HTTP(S) Load Balancing operates at a regional level.
  - Internal HTTP(S) Load Balancing isn't compatible with the following features:
    - Cloud CDN
    - Google Cloud Armor
    - Cloud Storage buckets
    - Google-managed SSL certificates
    - SSL policies
  - Internal TCP/UDP Load Balancing distributes traffic among VM instances in the same region in a Virtual Private Cloud (VPC) network by using an internal IP address.

- Google Cloud External HTTP(S) Load Balancing is a global, proxy-based Layer 7 load balancer that enables you to run and scale your services worldwide behind a single external IP address. External HTTP(S) Load Balancing distributes HTTP and HTTPS traffic to backends hosted on Compute Engine and Google Kubernetes Engine (GKE).
  - Support only ports 80 and 8080 for http and port 443 for https
  - When you configure an external HTTP(S) load balancer in Premium Tier, it uses a global

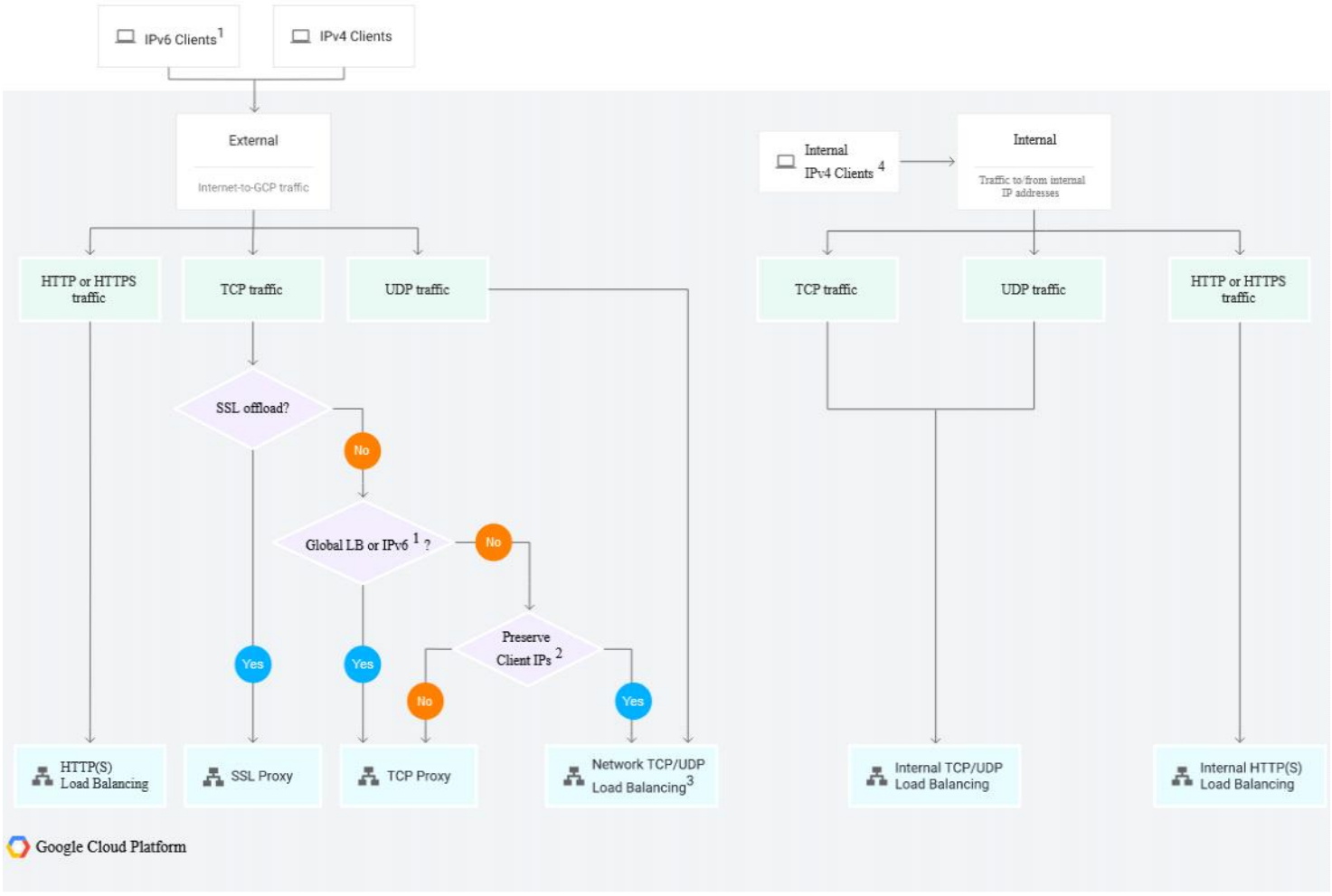
external IP address and can intelligently route requests from users to the closest backend instance group or NEG, based on proximity.
    - External HTTP(S) Load Balancing supports the following backend types:
        - Instance groups
        - Zonal network endpoint groups (NEGs)
        - Serverless NEGs: One or more App Engine, Cloud Run, or Cloud Functions services
        - Internet NEGs, for endpoints that are outside of Google Cloud (also known as custom origins)
        - Buckets in Cloud Storage
    - HTTP(S) Load Balancing supports content-based load balancing using URL maps to select a backend service based on the requested host name, request path, or both. For example, you can use a set of instance groups or NEGs to handle your video content and another set to handle everything else.

- Internal TCP/UDP Load Balancing distributes traffic among VM instances in the same region in a Virtual Private Cloud (VPC) network by using an internal IP address.
    - Google Cloud Internal TCP/UDP Load Balancing is a regional load balancer that enables you to run and scale your services behind an internal load balancing IP address that is accessible only to your internal virtual machine (VM) instances.
    - You can access an internal TCP/UDP load balancer in your VPC network from a connected network by using the following:
        - VPC Network Peering
        - Cloud VPN and Cloud Interconnect
    - Unlike a proxy load balancer, an internal TCP/UDP load balancer doesn't terminate connections from clients and then open new connections to backends. Instead, an internal TCP/UDP load balancer routes original connections directly from clients to the healthy backends, without any interruption.

- External TCP/UDP Network Load Balancing overview:
    - Google Cloud external TCP/UDP Network Load Balancing (after this referred to as Network Load Balancing) is a regional, pass-through load balancer. A network load balancer distributes TCP or UDP traffic among virtual machine (VM) instances in the same region.
    - Use Network Load Balancing in the following circumstances:
        - You need to load balance non-TCP traffic, or you need to load balance a TCP port that isn't supported by other load balancers.
        - It is acceptable to have SSL traffic decrypted by your backends instead of by the load balancer. The network load balancer cannot perform this task. When the backends decrypt SSL traffic, there is a greater CPU burden on the VMs.
        - Self-managing the backend VM's SSL certificates is acceptable to you. Google-managed SSL certificates are only available for HTTP(S) Load Balancing and SSL Proxy Load Balancing.
        - You need to forward the original packets unproxied. For example, if you need the client source IP to be preserved.
        - You have an existing setup that uses a pass-through load balancer, and you want to migrate it without changes.

- SSL Proxy Load Balancing overview:
    - SSL Proxy Load Balancing is a reverse proxy load balancer that distributes SSL traffic coming from the internet to virtual machine (VM) instances in your Google Cloud VPC network.
    - When using SSL Proxy Load Balancing for your SSL traffic, user SSL (TLS) connections are terminated at the load balancing layer, and then proxied to the closest available backend instances by using either SSL (recommended) or TCP.
    - With the Premium Tier, SSL Proxy Load Balancing can be configured as a global load balancing service. With Standard Tier, the SSL proxy load balancer handles load balancing regionally.

- **Support for the following well-known ports:** 25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 3389, 5222, 5432, 5671, 5672, 5900, 5901, 6379, 8085, 8099, 9092, 9200, and 9300. When you use Google-managed SSL certificates with SSL Proxy Load Balancing, the frontend port for traffic must be 443 to enable the Google-managed SSL certificates to be provisioned and renewed.
- By default, the original client IP address and port information is not preserved. You can preserve this information by using the PROXY protocol.



Load balancer	IP Address	Forwarding Rule	Other Frontend Components	Backend Components
Internal TCP/UDP Load Balancing	An internal IP address must be defined in the same project as the backend instances.	An internal forwarding rule must be defined in the same project as the backend instances (the service project).	Not applicable.	A regional backend service must be defined in the same project as the backend instances. Health checks associated with backend services must be defined in the same project as well.
External TCP/UDP Load Balancing	A regional external IP address must be	A regional external	Not applicable.	The target pool must be defined in the same project and same

	defined in the same project as the instances being load balanced.	forwarding rule must be defined in the same project as the instances in the target pool (the service project).		region where the instances in the target pool exist. Health checks associated with the target pool must be defined in the same project as well.
Internal HTTP(S) Load Balancing	An internal IP address must be defined in the same project as the load balancer.	An internal forwarding rule must be defined in the same project as the backend instances (the service project).	A regional target HTTP(S) proxy and associated regional URL map must be defined in the same project as the backend instances.	A regional backend service must be defined in the same project as the backend instances. Health checks associated with backend services must be defined in the same project as well.
External HTTP(S) Load Balancing	An external IP address must be defined in the same project as the instances being load balanced (the service project).	The external forwarding rule must be defined in the same project as the backend instances (the service project).	The target HTTP proxy or target HTTPS proxy and associated URL map must be defined in the same project as the backend instances.	A global backend service must be defined in the same project as the backend instances. These instances must be in instance groups attached to the backend service as backends. Health checks associated with backend services must be defined in the same project as the backend service as well.
SSL Proxy Load Balancing			The target SSL proxy must be defined in the same project as the backend instances.	
TCP Proxy Load Balancing			The target TCP proxy must be defined in the same project as the backend instances.	

The following table provides some specific information about each load balancer.

Load balancer type	Traffic type	Preserve Client IP	Global or regional	Load balancing scheme	Load balancer destination ports	Proxy or pass-through
External HTTP(S)	HTTP or HTTPS	No	Global*	EXTERNAL	HTTP on 80 or 8080; HTTPS on 443	Proxy
Internal HTTP(S)	HTTP or HTTPS	No	Regional	INTERNAL_MANAGED	HTTP on 80 or 8080; HTTPS on 443	Proxy
SSL Proxy	TCP with SSL offload	No	Global*	EXTERNAL	25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 3389, 5222, 5432, 5671, 5672, 5900, 5901, 6379, 8085, 8099, 9092, 9200, and 9300	Proxy

TCP Proxy	TCP without SSL offload	No	Global*	EXTERNAL	25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 3389, 5222, 5432, 5671, 5672, 5900, 5901, 6379, 8085, 8099, 9092, 9200, and 9300	Proxy
External Network TCP/UDP	TCP or UDP	Yes	Regional	EXTERNAL	Any	Pass-through
Internal TCP/UDP	TCP or UDP	Yes	Regional backends, regional frontends (global access supported)	INTERNAL	Any	Pass-through

*Global in Premium Tier. Regional in Standard Tier.

- ## Load balancer features

You can select the appropriate load balancer based on your application needs.

In the following tables, a checkmark indicates that a feature is supported. For more information about a feature, click the *info* link.

Type of load balancer

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Proxy	info Yes	Yes info			SSL Proxy info TCP proxy info
Pass-through			infoYes	info Yes	

Protocols from the load balancer to the backends

For links to reference information, see Backend services.

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy	
One of: HTTP (HTTP/1.1) HTTPS (HTTP/1.1) HTTP/2 (requires TLS), including gRPC						
One of: TCP or UDP						
One of: SSL (TLS) or TCP						
WebSockets	info Yes	info Yes				

Protocols from the clients to the load balancer

For links to reference information, see Forwarding rules.

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy	
One of:						

HTTP/1.1, HTTP/2, or HTTPS		(includes QUIC)				
One of: TCP or UDP						
SSL or TCP						
WebSockets	info	info				

Backends

Filter this table:

Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy	
Backends can be in multiple regions		(Premium Tier)			(Premium Tier)
Backends must be in one region		(Standard Tier)			(Standard Tier)
Cloud Storage in backend buckets		info			
External endpoints in internet NEGs as custom origins for Cloud CDN		info (Premium Tier)			
Load balancer can have multiple backend services and a URL map					
Self-managed Kubernetes and GKE					
Serverless backends: • Cloud Run (fully managed) • App Engine • Cloud Functions		info			
Virtual machine backends on Compute Engine					
Zonal NEGs	Using GCE_VM_IP_PORT type endpoints with GKE: • Use standalone zonal NEGs • Use Ingress for Internal HTTP(S) Load	Using GCE_VM_IP_PORT type endpoints with GKE: • Use standalone zonal NEGs • Use Ingress for external HTTP(S) Load	Using GCE_VM_IP type endpoints with GKE: • Use zonal NEGs for Internal TCP/UDP Load Balancing		Use standalone zonal NEGs

	Balancing	Balancing • Use Ingress for Internal HTTP(S) Load Balancing			

Health checks

For links to reference information, see Health checks.

Filter this table:

Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy	
Configurable expected response string				[1]	
Configurable health checks: • Port • Check intervals • Timeouts • Healthy and unhealthy thresholds					
Configurable request path (HTTP, HTTPS, HTTP/2)				[1]	
Configurable request string or path (TCP or SSL)				[1]	
HTTP health checks				[1]	
HTTP/2 health checks				[1]	
HTTPS health checks				[1]	
SSL health checks				[1]	
TCP health checks				[1]	

[1] This table documents health checks supported by backend service-based network load balancers (currently in Preview). Target pool-based network load balancers only support legacy HTTP health checks.

IP addresses

For links to reference information, see Addresses.

Filter this table:

Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy	
Client source IP address preservation	X-Forwarded-For header	X-Forwarded-For header			In TCP Proxy header
Internal IP address, accessible in your Virtual Private Cloud (VPC) network					
Internet accessible (including by clients that are in Google Cloud and have internet access)					

IPv6 termination					
Multiple forwarding rules with the same IP address, each having a unique protocol and port combination					
Privately accessible	• From same VPC network • From connected network • From same region		• From same VPC network • From connected network • From any region (with global access)		
Public IP address (global anycast)		(Premium Tier)			(Premium Tier)
Public IP address (regional)		(Standard Tier)			(Standard Tier)

Network topologies

Filter this table:

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Relationships between VPC networks and load balancer backends					
Backends must be in the same VPC network					
Backends can be located in multiple VPC networks in the same project (the networks do not have to be connected)					
Backends can use a Shared VPC network					
Client access to load balancers					
Google Cloud or on-premises clients must access the load balancer privately by being either in the same VPC network, in a peered VPC network, or in another network connected using Cloud VPN tunnels or Cloud Interconnect attachments (VLANs)					
Google Cloud client VMs require external IP addresses or a NAT solution like Cloud NAT to access the load balancer					
On-premises client VMs require internet access to access the load balancer					
Google Cloud client VMs can be located in any region			If global access is enabled		
Google Cloud client VMs can be located in any project (subject to other requirements in this table)					

Failover

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Automatic failover to healthy backends within same region					
Automatic failover to healthy backends in other regions		(Premium Tier)			(Premium Tier)
Behavior when all backends are unhealthy	Returns HTTP 503	Returns HTTP 502	Configurable behavior	Traffic distributed among all backends	Traffic dropped
Configurable standby backends			(with failover backends)	(with failover backends[1])	
Connection draining on failover and failback			info (configurable)	(configurable[2])	

This table documents failover as supported by backend service-based network load balancers (currently in Preview).

[1] Target pool-based network load balancers use backup pools to support failover.

[2] Target pool-based network load balancers do not support configuration of connection draining on failover/failback.

Logging and monitoring

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Byte count metrics	info	info	info	info	info
Packet count metrics			info	info	info
Round trip time or latency metrics	info	info	info	info	info
Connection count metrics					info
HTTP request count metrics	info	info			
HTTP request and response attribute logs	info	info			

Session affinity

For detailed information, see Session affinity.

For links to reference information, see Backend services.

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Client IP address (2-tuple hash of packet's source and destination IP addresses)			(TCP only)	(TCP only)	
Headers					
HTTP cookie					
Generated cookie					

Client IP address, protocol (3-tuple hash of packet's source IP address, packet's destination IP address, and protocol)			(TCP only)	(TCP only)	
Client IP address, port, protocol			(TCP only)	(TCP only)	
None (5-tuple hash)					

Load balancing methods

For detailed information, see the Backend services overview.

For links to reference information, see Backend services.

Filter this table:

Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy	
Balancing mode: backend utilization (instance group backends only)					
Balancing mode: connection					
Balancing mode: rate (requests per second)					
Circuit breaking					
Configurable maximum capacity per backend instance group or NEG					
Percent of traffic/weight-based					
Prefers region closest to client on the internet When the closest region is at capacity or isn't healthy, prefers next closest region		(Premium Tier)			(Premium Tier)
Within zone/region load balancing policy	Load balancing locality policy	Round robin in a zone	Round robin among all backends in the active pool when failover is configured, or among all backends in the region	Round robin among all backends in the active pool when failover is configured, or among all backends in the region [1]	Round robin in a zone

[1] This table documents load balancing methods supported by backend service-based network load balancers (currently in Preview). Target pool-based network load balancers round robin among all instances in the target pool or backup pool.

Routing and traffic management

For internal HTTP(S) load balancers, see the following links:

- Traffic management overview for internal HTTP(S) load balancers
- Setting up traffic management for internal HTTP(S) load balancers

For external HTTP(S) load balancers, see the following links:

- Traffic management overview for external HTTP(S) load balancers
- Setting up traffic management for external HTTP(S) load balancers

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
HTTP/Layer 7 request routing	Suffix, prefix, and match on: • host name and path • headers • method • cookies • request parameters	Suffix, prefix, and match on: • host name and path • headers • cookies • request parameters			
Fault injection	info				
Configurable timeouts	info	info			info
Retries	info	info			
Redirects	info	info			
URI rewrites	info	info			
Request/response header transformations	info				
Traffic splitting	info				
Traffic mirroring	info				
Outlier detection	info				
Retry failed requests	info				

Autoscaling and autohealing

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Managed instance group autoscaling based on load balancer serving capacity					
Autohealing (native to managed instance groups and GKE)					
Connection draining				[1]	

[1] This table documents autoscaling and autohealing features supported by backend service-based network load balancers (currently in Preview). Target pool-based network load balancers do not support connection draining.

Security

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Managed certificates		info			info (SSL proxy only)
CORS	info				
Identity-Aware Proxy (IAP)	info	info			
Google Cloud Armor		info			
SSL offload					(SSL proxy only)
SSL policies (TLS version and cipher suites)		info			info (SSL proxy only)

Special features

Feature	Internal HTTP(S)	External HTTP(S)	Internal TCP/UDP	External TCP/UDP Network	External SSL Proxy and TCP Proxy
Cloud CDN		info (Premium Tier)			
External endpoints in internet NEGs as custom origins for Cloud CDN		info (Premium Tier)			
Internal DNS names	info		info		
Load balancer as next hop			info		
Specify network interface of a backend VM (Multi-NIC load balancing)			info		
Custom request and response headers		info Geo-location information, Smoothed RTT, Client latency			
Automatic Service Directory registration (Preview)	info		info		

Storage services and additional disk options

Monday, January 6, 2020 7:28 PM

Additional Disk Options:

- **Local SSD:** Fast 375GB SSDs attached to a server. Can stripe across 24 (total 9TB) for better performance, but depends on machine type Stays ON till instance is on.
- You can create an instance with:
  - 16 local SSD partitions for 6 TB of Local SSD space and performance of 1.6 million read IOPS
  - 24 local SSD partitions for 9 TB of Local SSD space and performance of 2.4 million read IOPS

  This is available on instances with N1, N2, N2D, and custom machine types. To achieve maximum performance on N1 machines, select a machine type with 32 or more vCPUs. To achieve maximum performance on N2 and N2D machines, select a machine type with 24 or more vCPUs.
- Local SSD: High performance, transient, local block storage.
  - Data persists only if user reboots, live migration or host error (if recovered within 60mins)
  - Stop or shut down will loose data
  - If you disable project billing. The instance will stop and your data will be lost.

- **Persistent disk:** Flexible block based attached to every GCE boot. Slower than SSD but more durable. Stay on even when instance is shutdown. Can resize upto 64TB. Can also store instance snapshots for full backup. Mostly zonal resources. You can mount 1 disk to multiple instances if all are read-only.
- When you configure a zonal or regional persistent disk, you can select one of the following disk types.
  - **Standard persistent disks** (`pd-standard`) are backed by standard hard disk drives (HDD).
  - **Balanced persistent disks** (`pd-balanced`) are backed by solid-state drives (SSD). They are an alternative to SSD persistent disks that balance performance and cost.
  - **SSD persistent disks** (`pd-ssd`) are backed by solid-state drives (SSD).
- If you create a disk in the Cloud Console, the default disk type is `pd-balanced`. If you create a disk using the `gcloud` tool or the Compute Engine API, the default disk type is `pd-standard`.
  - Zonal persistent disk: Efficient, reliable block storage.
    - You can create zonal disks from existing pd, images or snapshots
    - You can only resize a zonal persistent disk to increase its size. You cannot reduce the size of a zonal persistent disk.
    - The zone, region, and disk type of the clone must be the same as that of the source disk.
    - You cannot create a zonal disk clone from a regional disk. You cannot create a regional disk clone from a zonal disk.
    - You can create at most 1000 **total** disk clones of a given source disk. Exceeding this limit returns an `internalError`.
  - Regional persistent disk:
    - Regional block storage replicated in two zones.
    - You cannot create regional disks from images
    - Regional persistent disks can't be used as boot disks.
    - When resizing a regional persistent disk, you can only increase its size. You cannot reduce the size of a persistent disk.
    - To convert your existing zonal persistent disks to regional persistent disks, snapshot the current persistent disk and create a regional persistent disk from the snapshot.
    - The minimum size of a regional standard persistent disk is 200 GB.

  - R/W IOPS and Throughput comparison:
    - Persistent disk: ~7500-15000 (in regional) & 240-1200 MB/s
    - SSD Persistent disk: ~15000-100,000 & 240-1200 MB/s
    - Local SSD-SCSI: 800,000/900,000 (R/W) & 4680-9360 MB/s
    - Local SSD-NVMe: ~1,200,000/2,400,000(R/W) & 4680-9360 MB/s

- Cloud Storage buckets: Affordable object storage.
- Filestore: High performance file storage for Google Cloud users.

- **Cloud filestore:** Multiple users for the same file. Fully manages files but not backups. Comparable to a NAS. Is Zonal resource.
  - Google Cloud Filestore provides managed NFS file servers as a fully managed service on GCP. It is meant to provide high-performance file storage capabilities to applications running on Compute Engine and Kubernetes Engine instances.

- **Data storage service:** Service where you can physically send you stored data to google storage, where they will upload it directly to GCS. Is useful for very high amount of data ~400TB.
- **Cloud storage transfer:** Storage Transfer Service allows you to quickly import online data (ex: AWS data) into Cloud Storage. You can also set up a repeating schedule for transferring data, as well as transfer data within Cloud Storage, from one bucket to another. You can use a signed URL to allow people temporary access to read/write to your bucket. It can be made valid for limited time or more.
- **Cloud Transfer appliance:** Transfer Appliance is a hardware appliance you can use to securely migrate large volumes of data (from hundreds of terabytes up to 1 petabyte) to Google Cloud Platform without disrupting business operations.
  - To get help with general Transfer Appliance billing questions, email Transfer Appliance support.

- **Cloud storage (GCS)**: Infinitely scalable, managed, versioned and durable object storage.
  - **Google Cloud Storage** has some specific features that differentiate it from a proper file system:
    - It doesn't actually provide directories/folders, it only implements buckets and objects, i.e there's no concept of folders, nested directories, etc... See doc here for more details about that.
    - It doesn't implement file modification. When you upload an update to an existing object, it actually replaces that object altogether with the new version (versioning is available though).

  - Has integrated site hosting and CDN functionality too. Can be transitioned from region to multi-regional etc.
  - The gsutil utility can also automatically use object composition to perform uploads in parallel for large, local files that you want to upload to Cloud Storage. It splits a large file into component pieces, uploads them in parallel and then recomposes them once they're in the cloud (and deletes the temporary components it created locally).
    - gsutil -o GSUtil:parallel_composite_upload_threshold=150M cp ./localbigfile gs://your-bucket
      Where "localbigfile" is a file larger than 150MB. This divides up your data into chunks ~150MB and uploads them in parallel, increasing upload performance.
  - You can host static (http) webpages on GCS for high-availability but limited budget.
  - To host a static site in Cloud Storage, you need to create a Cloud Storage bucket, upload the content, and test your new site.
  - You can serve your data directly from storage.googleapis.com, or you can verify that you own your domain and use your domain name. Either way, you'll get consistent, fast delivery from global edge caches.
  - You can create your static web pages however you choose. For example, you could hand-author pages by using HTML and CSS. You can use a static-site generator, such as Jekyll, Ghost, or Hugo, to create the content. Static-site generators make it easier for you to create a static website by letting you author in markdown, and providing templates and tools. Site generators generally provide a local web server that you can use to preview your content.
  - After your static site is working, you can update the static pages by using any process you like. That process could be as straightforward as hand-copying an updated page to the bucket. You might choose to use a more automated approach, such as storing your content on GitHub and then using a webhook to run a script that updates the bucket. An even more advanced system might use a continuous-integration /continuous-delivery (CI/CD) tool, such as Jenkins, to update the content in the bucket. Jenkins has a Cloud Storage plugin that provides a Google Cloud Storage Uploader post-build step to publish build artifacts to Cloud Storage.; If you have a web application that needs to serve static content or user-uploaded static media, using Cloud Storage can be a cost-effective and efficient way to host and serve this content, while reducing the amount of dynamic requests to your web application.
  - Signed URLs: Use to give time-limited resource access to anyone in possession of the URL, regardless of whether they have a Google account.
    - Signed URLs can only be used to access resources in Cloud Storage through XML API endpoints
    - Creating a signed URL to download an object:
      - Generate a new private key, or use an existing private key for a service account. The key can be in either JSON or PKCS12 format.
      - gsutil signurl -d 10m Desktop/private-key.json gs://example-bucket/cat.jpeg
    - Creating a signed URL to upload an object
      - Generate a new private key, or use an existing private key for a service account. The key can be in either JSON or PKCS12

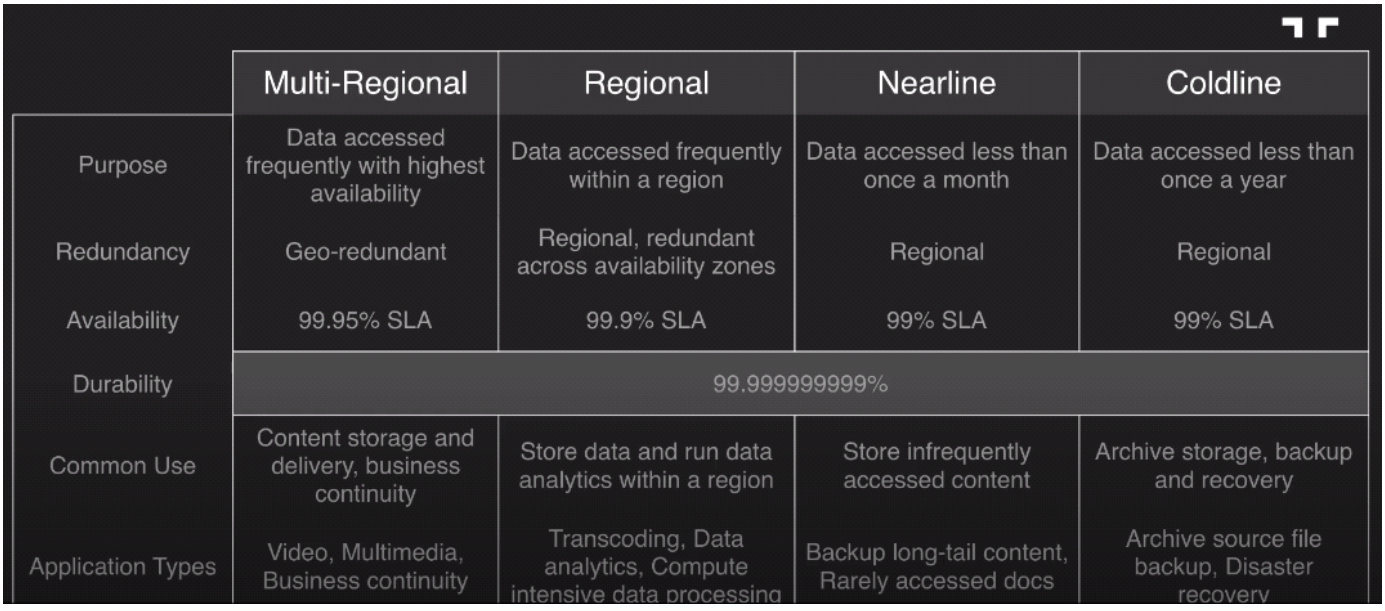
format.

- Use gcloud auth activate-service-account to authenticate with the service account:
  - gcloud auth activate-service-account --key-file KEY_FILE_LOCATION/KEY_FILE_NAME
- gsutil signurl -m PUT -d 1h -c CONTENT_TYPE -u gs://BUCKET_NAME/OBJECT_NAME

  -c: Specifies the content type for which the signed url is valid for ex:-c text/plain

- Cloud object Metadata: Objects stored in Cloud Storage have metadata associated with them.
  - Metadata identifies properties of the object, as well as specifies how the object should be handled when it's accessed. Metadata exists as key:value pairs. For example, the storage class of an object is represented by the metadata entry storageClass:STANDARD.
  - The mutability of metadata varies: some metadata you can edit at any time, some metadata you can only set at the time the object is created, and some metadata you can only view. For example, you can edit the value of the Cache-Control metadata at any time, but you can only assign the storageClass metadata when the object is created or rewritten, and you cannot directly edit the value for the generation metadata, though the generation value changes when the object is replaced.
  - Content-Disposition: The Content-Disposition metadata specifies presentation information about the data being transmitted. Setting Content-Disposition allows you to control presentation style of the content, for example determining whether an attachment should be automatically displayed or whether some form of action from the user should be required to open it
  - Content-Type: The most commonly set metadata is Content-Type (also known as media type), which lets browsers render the object properly. All objects have a value specified in their Content-Type metadata, but this value does not have to match the underlying type of the object. For example, if the Content-Type is not specified by the uploader and cannot be determined, it is set to application/octet-stream or application/x-www-form-urlencoded, depending on how you uploaded the object.
  - Object holds: Use metadata flags to place object holds, which prevent objects from being deleted or replaced.
- Best practices:
  - If you are concerned that your application software or users might erroneously delete or replace objects at some point, Cloud Storage has features that help you protect your data:
  - A retention policy that specifies a retention period can be placed on a bucket. An object in the bucket cannot be deleted or replaced until it reaches the specified age.
  - An object hold can be placed on individual objects to prevent anyone from deleting or replacing the object until the hold is removed.
  - Object versioning can be enabled on a bucket in order to retain older versions of objects. When the live version of an object is deleted or replaced, it becomes noncurrent if versioning is enabled on the bucket. If you accidentally delete a live object version, you can copy the noncurrent version of it back to the live version.
  - If you want to bulk delete a hundred thousand or more objects, avoid using gsutil, as the process takes a long time to complete. Instead, use one of the following options:
    - The Cloud Console can bulk delete up to several million objects and does so in the background. The Cloud Console can also be used to bulk delete only those objects that share a common prefix, which appear as part of a folder when using the Cloud Console.
  - Object Lifecycle Management can bulk delete any number of objects. To bulk delete objects in your bucket, set a lifecycle configuration rule on your bucket where the condition has Age set to 0 days, and the action is set to Delete. Be aware that during the deletion process, object listing for the affected bucket may be impacted.

Available storage classes



	Multi-Regional	Regional	Nearline	Coldline
Purpose	Data accessed frequently with highest availability	Data accessed frequently within a region	Data accessed less than once a month	Data accessed less than once a year
Redundancy	Geo-redundant	Regional, redundant across availability zones	Regional	Regional
Availability	99.95% SLA	99.9% SLA	99% SLA	99% SLA
Durability	99.999999999%			
Common Use	Content storage and delivery, business continuity	Store data and run data analytics within a region	Store infrequently accessed content	Archive storage, backup and recovery
Application Types	Video, Multimedia, Business continuity	Transcoding, Data analytics, Compute intensive data processing	Backup long-tail content, Rarely accessed docs	Archive source file backup, Disaster recovery

Storage Class	Characteristics	Use Cases	Price (per GB per month)**
Multi-Regional Storage	• >99.99% typical monthly availability • 99.95% availability SLA* • Geo-redundant	Storing data that is frequently accessed ("hot" objects) around the world, such as serving website content, streaming videos, or gaming and mobile applications. For Multi-Regional Storage data stored in dual-regional locations, you also get optimized performance when accessing Google Cloud Platform products that are located in one of the associated regions.	$0.026
Regional Storage	• 99.99% typical monthly availability • 99.9% availability SLA* • Lower cost per GB stored • Data stored in a narrow geographic region • Redundant across availability zones	Storing frequently accessed data in the same region as your Google Cloud DataProc or Google Compute Engine instances that use it, such as for data analytics.	$0.020
Nearline Storage	• 99.95% typical monthly availability in multi-regional locations; 99.9% typical monthly availability in regional locations. • 99.9% availability SLA* in multi-regional locations; 99.0% availability SLA* in regional locations. • Very low cost per GB stored • Data retrieval costs • Higher per-operation costs • 30-day minimum storage duration	Data you do not expect to access frequently (i.e., no more than once per month). Ideal for back-up and serving long-tail multimedia content.	$0.010
Coldline Storage	• 99.95% typical monthly availability in multi-regional locations; 99.9% typical monthly availability in regional locations. • 99.9% availability SLA* in multi-regional locations; 99.0% availability SLA* in regional locations. • Lowest cost per GB stored • Data retrieval costs • Higher per-operation costs • 90-day minimum storage duration	Data you expect to access infrequently (i.e., no more than once per year). Typically this is for disaster recovery, or data that is archived and may or may not be needed at some future time.	$0.007

The following table summarizes the primary storage classes offered by Cloud Storage. See class descriptions for a complete discussion.

Storage Class	Name for APIs and gsutil	Minimum storage duration	Typical monthly availability[1]
Standard Storage	STANDARD	None	• >99.99% in multi-regions and dual-regions • 99.99% in regions
Nearline Storage	NEARLINE	30 days	• 99.95% in multi-regions and dual-regions • 99.9% in regions
Coldline Storage	COLDLINE	90 days	• 99.95% in multi-regions and dual-regions • 99.9% in regions
Archive Storage	ARCHIVE	365 days	• 99.95% in multi-regions and dual-regions • 99.9% in regions

Cloud storage and gsutil

Saturday, February 6, 2021 12:09 AM

- The `gsutil config` command applies to users who have installed gsutil as a standalone tool.
  - If you installed gsutil via the Cloud SDK, `gsutil config` fails unless you are specifically using the `-a` flag or have configured gcloud to not pass its managed credentials to gsutil (via the command `gcloud config set pass_credentials_to_gsutil false`). For all other use cases, Cloud SDK users should use the `gcloud auth` group of commands instead, which configures OAuth2 credentials that gcloud implicitly passes to gsutil at runtime. To check if you are using gsutil from the Cloud SDK or as a stand-alone, use `gsutil version -l` and in the output look for "using cloud sdk".
  - The `gsutil config` command obtains access credentials for Cloud Storage and writes a boto/gsutil configuration file containing the obtained credentials along with a number of other configuration-controllable values.
  - Unless specified otherwise (see OPTIONS), the configuration file is written to ~/.boto (i.e., the file .boto under the user's home directory). If the default file already exists, an attempt is made to rename the existing file to ~/.boto.bak; if that attempt fails the command exits. A different destination file can be specified with the `-o` option (see OPTIONS).
  - Because the boto configuration file contains your credentials you should keep its file permissions set so no one but you has read access. (The file is created read-only when you run `gsutil config`.)

- `gsutil mb [-b (on|off)] [-c <class>] [-l <location>] [-p <proj_id>] [--retention <time>] gs://<bucket_name>...`
  - The mb command creates a new bucket. Cloud Storage has a single namespace, so you are not allowed to create a bucket with a name already in use by another user.
  - The -c and -l options specify the storage class and location, respectively, for the bucket. Once a bucket is created in a given location and with a given storage class, it cannot be moved to a different location, and the storage class cannot be changed. Instead, you would need to create a new bucket and move the data over and then delete the original bucket.
    - Use `gsutil rewrite -s nearline gs://bucket/foo` or set new default storage call in new bucket to change storage class of objects.
  - The --retention option specifies the retention period for the bucket.
    You can specify retention period in one of the following formats:
    - --retention <number>s
    - Specifies retention period of <number> seconds for objects in this bucket.
    - --retention <number>d
    - Specifies retention period of <number> days for objects in this bucket.
    - --retention <number>m
    - Specifies retention period of <number> months for objects in this bucket.
    - --retention <number>y
    - Specifies retention period of <number> years for objects in this bucket.
- rb - Remove buckets
  - gsutil rb [-f] gs://<bucket_name>...
  - The rb command deletes a bucket. Buckets must be empty before you can delete them.
  - Be certain you want to delete a bucket before you do so, as once it is deleted the name becomes available and another user may create a bucket with that name.

- `gsutil -o GSUtil:parallel_composite_upload_threshold=150M cp ./localbigfile gs://your-bucket`
  Where "localbigfile" is a file larger than 150MB. This divides up your data into chunks ~150MB and uploads them in parallel, increasing upload performance.

- **Access to buckets:** Cloud Storage offers two systems for granting users permission to access your buckets and objects: IAM and Access Control Lists (ACLs). These systems act in parallel - in order for a user to access a Cloud Storage resource, only one of the systems needs to grant the user permission. IAM is used throughout Google Cloud and allows you to grant a variety of permissions at the bucket and project levels. ACLs are used only by Cloud Storage and have limited permission options, but they allow you to grant permissions on a per-object basis

  In order to support a uniform permissioning system, Cloud Storage has uniform bucket-level access. Using this feature disables ACLs for all Cloud Storage resources: access to Cloud Storage resources then is granted exclusively through IAM. After you enable uniform bucket-level access, you can reverse your decision only within 90 days.

  - **iam** - Get, set, or change bucket and/or object IAM permissions.
    - Cloud Identity and Access Management (Cloud IAM) allows you to control who has access to the resources in your Google Cloud project.
    - The iam command has three sub-commands:
    - Get: The iam get command gets the Cloud IAM policy for a bucket or object, which you can save and edit for use with the iam set command. The output is in json similar to the iam-policy-binding output
    - The following examples save the bucket or object's Cloud IAM policy to a text file:
      - gsutil iam get gs://example > bucket_iam.txt
    - Set: The iam set command sets a Cloud IAM policy on one or more buckets or objects, replacing the existing policy on those buckets or objects.
      - `gsutil -m iam set -r iam.txt gs://dogs`
    - The set sub-command has the following options:
      - -R, -r: Performs iam set recursively on all objects under the specified bucket. This flag can only be set if the policy exclusively uses roles/storage.legacyObjectReader or roles/storage.legacyObjectOwner. This flag cannot be used if the bucket is configured for uniform bucket-level access.
      - -a: Performs iam set on all object versions.
      - -e <etag>: Performs the precondition check on each object with the specified etag before setting the policy. You can retrieve the policy's etag using iam get.
      - -f The default gsutil error-handling mode is fail-fast. This flag changes the request to fail-silent mode. This option is implicitly set when you use the gsutil -m option.
    - Ch: The iam ch command incrementally updates Cloud IAM policies. You can specify multiple access grants or removals in a single command. The access changes are applied as a batch to each url in the order in which they appear in the command line arguments. Each access change specifies a member and a role that is either granted or revoked.
    - You can use gsutil -m to handle object-level operations in parallel.
    - The ch sub-command has the following options:
      - -d Removes roles granted to the specified member.
      - -R, -r: Performs iam ch recursively to all objects under the specified bucket.
      - This flag can only be set if the policy exclusively uses roles/storage.legacyObjectReader or roles/storage.legacyObjectOwner. This flag cannot be used if the bucket is configured for uniform bucket-level access.
      - -f : The default gsutil error-handling mode is fail-fast. This flag changes the request to fail-silent mode. This is implicitly set when you invoke the gsutil -m option.
  - **acl** - Get, set, or change bucket and/or object ACLs
    - The acl command has three sub-commands:
      - gsutil acl set [-f] [-r] [-a] <file-or-canned_acl_name> url...
        - `R: READ`

W: WRITE
O: OWNER

- gsutil acl get url
- Use gsutil acl ch to change access sequentially of some members:
- gsutil acl ch [-f] [-r] <grant>... url...

  where each <grant> is one of the following forms:

  - -u <id>|<email>:<permission> ex –u AllUsers:R I.e grant allusers read only permission
  - -g <id>|<email>|<domain>|All|AllAuth:<perm> ex: -g admins@example.com:O gs://example-bucket/**.jpg I.e allow admin group owner access to all jpeg files
  - -p (viewers|editors|owners)-<project number>:<perm> ex: -p owners-project12445:W gs://example-bucket i.e grant all project owners write access
  - -d <id>|<email>|<domain>|All|AllAuth|(viewers|editors|owners)-<project number> ex: -d viewers-12345 gs://example-bucket i.e delete all viewers for project number 12345

- Note that you can set an ACL on multiple buckets or objects at once. For example, to set ACLs on all .jpg files found in a bucket:
  - gsutil acl set acl.txt gs://bucket/**.jpg
- If you have a large number of ACLs to update you might want to use the gsutil -m option, to perform a parallel (multi-threaded/multi-processing) update:
  - gsutil -m acl set acl.txt gs://bucket/**.jpg
  - One strategy for uploading large files is called *parallel composite uploads*. In such an upload, a single file is divided into up to 32 chunks, the chunks are uploaded in parallel to temporary objects, the final object is recreated using the temporary objects, and the temporary objects are deleted.
  - Parallel composite uploads can be significantly faster if network and disk speed are not limiting factors; however, the final object stored in your bucket is a *composite object*, which only has a crc32c hash and not an MD5 hash. As a result, you must use crcmod to perform integrity checks when downloading the object with gsutil or other Python applications.
- Note that multi-threading/multi-processing is only done when the named URLs refer to objects, which happens either if you name specific objects or if you enumerate objects by using an object wildcard or specifying the acl -r flag.

- **bucketpolicyonly** - Configure uniform bucket-level access
  - When you enable uniform bucket-level access on a bucket, Access Control Lists (ACLs) are disabled, and only bucket-level Identity and Access Management (IAM) permissions grant access to that bucket and the objects it contains. You revoke all access granted by object ACLs and the ability to administrate permissions using bucket ACLs.
    - You might not want to use uniform bucket-level access and instead retain fine-grained ACLs if you want to control access to specific objects in a bucket via legacy ACLs.
  - gsutil bucketpolicyonly set (on|off) gs://<bucket_name>...
  - gsutil bucketpolicyonly get gs://<bucket_name>...
  - The bucketpolicyonly command is used to retrieve or configure the uniform bucket-level access setting of Cloud Storage buckets. This command has two sub-commands, get and set.
  - The bucketpolicyonly get command shows whether uniform bucket-level access is enabled for the specified Cloud Storage bucket.
  - The bucketpolicyonly set command enables or disables the uniform bucket-level access feature on Cloud Storage buckets.
  - The Bucket Policy Only feature is now known as uniform bucket-level access. The bucketpolicyonly

command is still supported, but we recommend using the equivalent ubla command.

- **defacl** - Get, set, or change default ACL on buckets
  - gsutil defacl set <file-or-canned_acl_name> gs://<bucket_name>...

    Allows 3 categories:
  - Set: The "defacl set" command sets default object ACLs for the specified buckets. If you specify a default object ACL for a certain bucket, Cloud Storage applies the default object ACL to all new objects uploaded to that bucket, unless an ACL for that object is separately specified during upload.
  - Similar to the "acl set" command, the file-or-canned_acl_name names either a canned ACL or the path to a file that contains ACL text.
  - Setting a default object ACL on a bucket provides a convenient way to ensure newly uploaded objects have a specific ACL. If you don't set the bucket's default object ACL, it will default to project-private. If you then upload objects that need a different ACL, you will need to perform a separate ACL update operation for each object. Depending on how many objects require updates, this could be very time-consuming.
  - Get:Gets the default ACL text for a bucket, which you can save and edit for use with the "defacl set" command.
  - Ch:
    - The "defacl ch" (or "defacl change") command updates the default object access control list for a bucket. The syntax is shared with the "acl ch" command, so see the "CH" section of gsutil help acl for the full help description.
    - Grant anyone on the internet READ access by default to any object created in the bucket example-bucket:
    - gsutil defacl ch -u AllUsers:R gs://example-bucket
    - Ch Options:
    - The "ch" sub-command has the following options
      - -d Remove all roles associated with the matching entity.
      - -f Normally gsutil stops at the first error. The -f option causes it to continue when it encounters errors. With this option the gsutil exit status will be 0 even if some ACLs couldn't be changed.
      - -g Add or modify a group entity's role.
      - -p Add or modify a project viewers/editors/owners role.
      - -u Add or modify a user entity's role.

- compose - Concatenate a sequence of objects into a new composite object.
  - gsutil compose gs://bucket/obj1 [gs://bucket/obj2 ...] gs://bucket/composite
  - The compose command creates a new object whose content is the concatenation of a given sequence of source objects under the same bucket. gsutil uses the content type of the first source object to determine the destination object's content type.
- cp command - Allows you to copy data between your local file system and the cloud, within the cloud, and between cloud storage providers.
  - gsutil cp [OPTION]... src_url dst_url
  - For example, to upload all text files from the local directory to a bucket, you can run:
    - gsutil cp *.txt gs://my-bucket
  - You can also download text files from a bucket:
    - gsutil cp gs://my-bucket/*.txt .
  - Use the -r option to copy an entire directory tree. For example, to upload the directory tree dir:

        - gsutil cp -r dir gs://my-bucket
    - If you have a large number of small files to transfer, you can perform a parallel multi-threaded/multi-processing copy using the top-level gsutil -m option:
        - gsutil -m cp -r dir gs://my-bucket
    - You can use the -I option with stdin to specify a list of URLs to copy, one per line. This allows you to use gsutil in a pipeline to upload or download objects as generated by a program:
        - cat filelist | gsutil -m cp -I gs://my-bucket

          or:
        - cat filelist | gsutil -m cp -I ./download_dir
    - where the output of cat filelist is a list of files, cloud URLs, and wildcards of files and cloud URLs.
- defstorageclass - Get or set the default storage class on buckets
    - gsutil defstorageclass set <storage-class> gs://<bucket_name>...
    - gsutil defstorageclass get gs://<bucket_name>...
    - The defstorageclass command has two sub-commands:
    - Set: The "defstorageclass set" command sets the default storage class for the specified bucket(s). If you specify a default storage class for a certain bucket, Cloud Storage applies the default storage class to all new objects uploaded to that bucket, except when the storage class is overridden by individual upload requests.
        - Setting a default storage class on a bucket provides a convenient way to ensure newly uploaded objects have a specific storage class. If you don't set the bucket's default storage class, it will default to Standard.
    - Get: Gets the default storage class for a bucket.
- du - Display object size usage
    - gsutil du url...
    - The du command displays the amount of space in bytes used up by the objects in a bucket, subdirectory, or project.
- label - Get, set, or change the label configuration of a bucket.
    - gsutil label set <label-json-file> gs://<bucket_name>...
    - gsutil label get gs://<bucket_name>
    - gsutil label ch <label_modifier>... gs://<bucket_name>...
        - where each <label_modifier> is one of the following forms:
        - -l <key>:<value> : To add/change a label
        - -d <key> : To delete a label with key <key>
- lifecycle - Get or set lifecycle configuration for a bucket
    - gsutil lifecycle get gs://<bucket_name>
    - gsutil lifecycle set <config-json-file> gs://<bucket_name>..
        - To delete object lifecycle rules, either use console and delete them, or create an empty lifecycle config and set it:
            - ```
              {
                "lifecycle": {
                  "rule": []
                }
              }
              ```
    - The following lifecycle configuration defines three rules. Note that the second and third rules are applicable only when using Object Versioning on the bucket:
              ```

- Delete live versions (isLive:True) of objects older than 30 days.
  - If the bucket uses Object Versioning, such objects become noncurrent and are subject to the other two rules.
  - If the bucket does not use Object Versioning, such objects are permanently deleted and cannot be recovered.
- Delete noncurrent versions of objects if there are 2 newer versions (`numNewerVersions`) of the object in the bucket. Objects subject to this rule are permanently deleted and cannot be recovered.
- Delete noncurrent versions (isLive:False) of objects older than 35 days. Objects subject to this rule are permanently deleted and cannot be recovered.

```
{
"lifecycle":
{

"rule": [

{   "action": {"type": "Delete"},

        "condition": {
          "age": 30,
          "isLive": true
        }
      },

{
    "action": {"type": "Delete"},

        "condition": {
          "numNewerVersions": 2
        }
      },

{
    "action": {"type": "Delete"},

        "condition": {
          "age": 35,
          "isLive": false
        }
      }

    ]

}

}
```

- logging - Configure or retrieve logging on buckets
  - Cloud Storage offers usage logs and storage data in the form of CSV files that you can download and view.
  - The logs and storage data files are automatically created as new objects in a bucket that you specify, in 24 hour intervals.
  - Usage logs provide information for all of the requests made on a specified bucket in the last 24 hours, while the storage logs provide information about the storage consumption of that bucket for the last 24 hour period. The logs and storage data files are automatically created as new objects in a bucket that you specify, in 24 hour intervals.

- The logging command has two sub-commands:
- Set: The set sub-command has two sub-commands:
  - The "gsutil logging set on" command will enable usage logging of the buckets named by the specified URLs, outputting log files in the specified logging_bucket.
  - logging_bucket must already exist, and all URLs must name buckets (e.g., gs://bucket). The required bucket parameter specifies the bucket to which the logs are written, and the optional log_object_prefix parameter specifies the prefix for log object names. The default prefix is the bucket name. For example, the command:
  - gsutil logging set on -b gs://my_logging_bucket -o UsageLog \ gs://my_bucket1 gs://my_bucket2
    - will cause all read and write activity to objects in gs://mybucket1 and gs://mybucket2 to be logged to objects prefixed with the name "UsageLog", with those log objects written to the bucket gs://my_logging_bucket.
- In addition to enabling logging on your bucket(s), you will also need to grant cloud-storage-analytics@google.com write access to the log bucket, using this command:
  - gsutil acl ch -g cloud-storage-analytics@google.com:W gs://my_logging_bucket
  - Note that log data may contain sensitive information, so you should make sure to set an appropriate default bucket ACL to protect that data.
- Off: This command will disable usage logging of the buckets named by the specified URLs. All URLs must name buckets (e.g., gs://bucket).
  - No logging data is removed from the log buckets when you disable logging, but Cloud Storage will stop delivering new logs once you have run this command.
- Get: If logging is enabled for the specified bucket url, the server responds with a JSON document that looks something like this:

```
{
  "logBucket": "my_logging_bucket",
  "logObjectPrefix": "UsageLog"
}
```

  - You can download log data from your log bucket using the gsutil cp command.

- rsync - Synchronize content of two buckets/directories
  - gsutil rsync [OPTION]... src_url dst_url
  - The gsutil rsync command makes the contents under dst_url the same as the contents under src_url, by copying any missing files/objects (or those whose data has changed), and (if the -d option is specified) deleting any extra files/objects. src_url must specify a directory, bucket, or bucket subdirectory
- setmeta - Set metadata on already uploaded objects
  - gsutil setmeta -h [header:value|header] ... url...
  - The gsutil setmeta command allows you to set or remove the metadata on one or more objects. It takes one or more header arguments followed by one or more URLs, where each header argument is in one of two forms:
  - If you specify header:value, it sets the provided value for the given header on all applicable objects.
  - If you specify header (with no value), it removes the given header from all applicable objects.
  - For example, the following command sets the Content-Type and Cache-Control headers while also removing the Content-Disposition header on the specified objects:
    - gsutil setmeta -h "Content-Type:text/html" \ -h "Cache-Control:public, max-age=3600" \ -h "Content-Disposition" gs://bucket/*.html
  - You can also use the setmeta command to set custom metadata on an object:

        - gsutil setmeta -h "x-goog-meta-icecreamflavor:vanilla" gs://bucket/object
        - Custom metadata is always prefixed in gsutil with x-goog-meta-. This distinguishes it from standard request headers. Other tools that send and receive object metadata by using the request body do not use this prefix.
    - Stat : Can be used to display current metadata
- versioning - Enable or suspend versioning for one or more buckets
        - gsutil versioning set (on|off) gs://<bucket_name>...
        - gsutil versioning get gs://<bucket_name>...
    - The Versioning Configuration feature enables you to configure a Cloud Storage bucket to keep old versions of objects.
    - **Caution:** Object Versioning **does not** protect your data if you delete the entire bucket.
    - The gsutil versioning command has two sub-commands:
    - Set: The "set" sub-command requires an additional sub-command, either "on" or "off", which, respectively, will enable or disable versioning for the specified bucket(s).
    - Get: The "get" sub-command gets the versioning configuration for a bucket and displays whether or not it is enabled.
- signurl - Create a signed url
    - gsutil signurl
        - [-c <content_type for which valid>]
        - [-d <duration>] : Max is 7 days
        - [-m <http_method>] : Specifies the HTTP method to be authorized for use with the signed url, default is GET (download). PUT (upload) can also be use or RESUMABLE to allow resumable upload
        - [-p <password>] : Specify the private key password instead of prompting.
        - [-r <region>]
        - [-b <project>] : Allows you to specify a user project that will be billed for requests that use the signed URL.
        - (-u | <private-key-file>) : Use service account credentials instead of a private key file to sign the url.
        - (gs://<bucket_name> | gs://<bucket_name>/<object_name>)...
    - The signurl command will generate a signed URL that embeds authentication data so the URL can be used by someone who does not have a Google account.
    - Multiple gs:// urls may be provided and may contain wildcards. A signed url will be produced for each provided url, authorized for the specified HTTP method and valid for the given duration.

Create a signed url for downloading an object valid for 10 minutes:

```
gsutil signurl -d 10m <private-key-file> gs://<bucket>/<object>
```

Create a signed url without a private key, using a service account's credentials:

```
gsutil signurl -d 10m -u gs://<bucket>/<object>
```

Create a signed url by impersonating a service account:

```
gsutil -i <service account email> signurl -d 10m -u gs://<bucket>/<object>
```

Create a signed url, valid for one hour, for uploading a plain text file via HTTP PUT:

```
gsutil signurl -m PUT -d 1h -c text/plain <private-key-file> \
    gs://<bucket>/<obj>
```

To construct a signed URL that allows anyone in possession of the URL to PUT to the specified bucket for one day, creating an object of Content-Type image/jpg, run:

```
gsutil signurl -m PUT -d 1d -c image/jpg <private-key-file> \
    gs://<bucket>/<obj>
```

To construct a signed URL that allows anyone in possession of the URL to POST a resumable upload to the specified bucket for one day, creating an object of Content-Type image/jpg, run:

```
gsutil signurl -m RESUMABLE -d 1d -c image/jpg <private-key-file> \
    gs://bucket/<obj>
```

Cloud storage: Object lifecycle management

Saturday, February 6, 2021 11:16 PM

Covered Service	Monthly Uptime Percentage
Standard storage class in a multi-region or dual-region location of Cloud Storage	>= 99.95%
Standard storage class in a regional location of Cloud Storage; Nearline or Coldline storage class in a multi-region or dual-region location of Cloud Storage	>= 99.9%
Nearline or Coldline storage class in a regional location of Cloud Storage; Durable Reduced Availability storage class in any location of Cloud Storage	>= 99.0%

Standard Storage (per GB per Month)	Nearline Storage (per GB per Month)	Coldline Storage (per GB per Month)	Archive Storage (per GB per Month)
$0.020	$0.010	$0.004	$0.0012

- To support common use cases like setting a Time to Live (TTL) for objects, retaining noncurrent versions of objects, or "downgrading" storage classes of objects to help manage costs, Cloud Storage offers the Object Lifecycle Management feature.
- You can assign a lifecycle management configuration to a bucket. The configuration contains a set of rules which apply to current and future objects in the bucket. When an object meets the criteria of one of the rules, Cloud Storage automatically performs a specified action on the object. Here are some example use cases:
    - Downgrade the storage class of objects older than 365 days to Coldline Storage.
    - Delete objects created before January 1, 2013.
    - Keep only the 3 most recent versions of each object in a bucket with versioning enabled.
- Lifecycle configuration
    - Each lifecycle management configuration contains a set of rules. When defining a rule, you can specify any set of conditions for any action. If you specify multiple conditions in a rule, an object has to match all of the conditions for the action to be taken. If you specify multiple rules that contain the same action, the action is taken when an object matches the condition(s) in any of the rules. Each rule should contain only one action.
    - If multiple rules have their conditions satisfied simultaneously for a single object, Cloud Storage performs the action associated with only one of the rules, based on the following considerations:
        - The Delete action takes precedence over any SetStorageClass action.

        - The SetStorageClass action that switches the object to the storage class with the lowest at-rest storage pricing takes precedence.
        - Once an action occurs, the object is re-evaluated before any additional actions are taken. So, for example, if you have one rule that deletes an object and another rule that changes the object's storage class, but both rules use the exact same condition, the delete action always occurs when the condition is met. If you have one rule that changes the object's class to Nearline Storage and another rule that changes the object's class to Coldline Storage, but both rules use the exact same condition, the object's class always changes to Coldline Storage when the condition is met.
    - Delete: The Delete action deletes an object when the object meets all conditions specified in the lifecycle rule.
        - Exception: In buckets with Object Versioning enabled, deleting the live version of an object causes it to become a noncurrent version, while deleting a noncurrent version deletes that version permanently.
    - SetStorageClass: The SetStorageClass action changes the storage class of an object when the object meets all conditions specified in the lifecycle rule.
    - SetStorageClass supports the following storage class transitions:
    -

Original storage class	New storage class
Durable Reduced Availability (DRA) Storage	Nearline Storage Coldline Storage Archive Storage Multi-Regional Storage/Regional Storage[1]
Standard Storage, Multi-Regional Storage, or Regional Storage	Nearline Storage Coldline Storage Archive Storage
Nearline Storage	Coldline Storage Archive Storage
Coldline Storage	Archive Storage

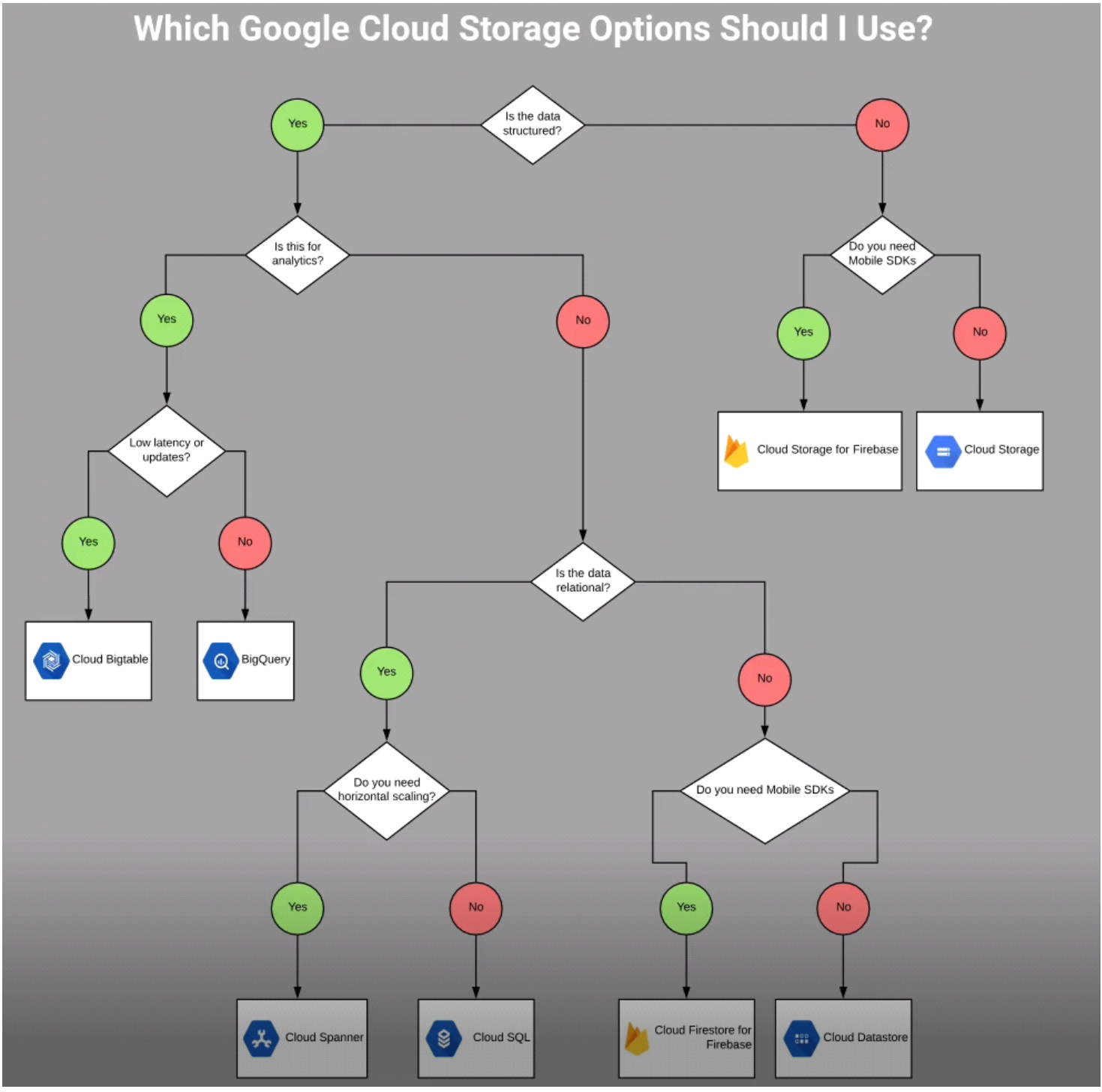
    - For buckets in a region, the new storage class cannot be Multi-Regional Storage. For buckets in a multi-region or dual-region, the new storage class cannot be Regional Storage.
- Lifecycle conditions: A lifecycle rule includes conditions which an object must meet before the action defined in the rule occurs on the object. Lifecycle rules support the following conditions:
    - Age
    - CreatedBefore
    - CustomTimeBefore
    - DaysSinceCustomTime

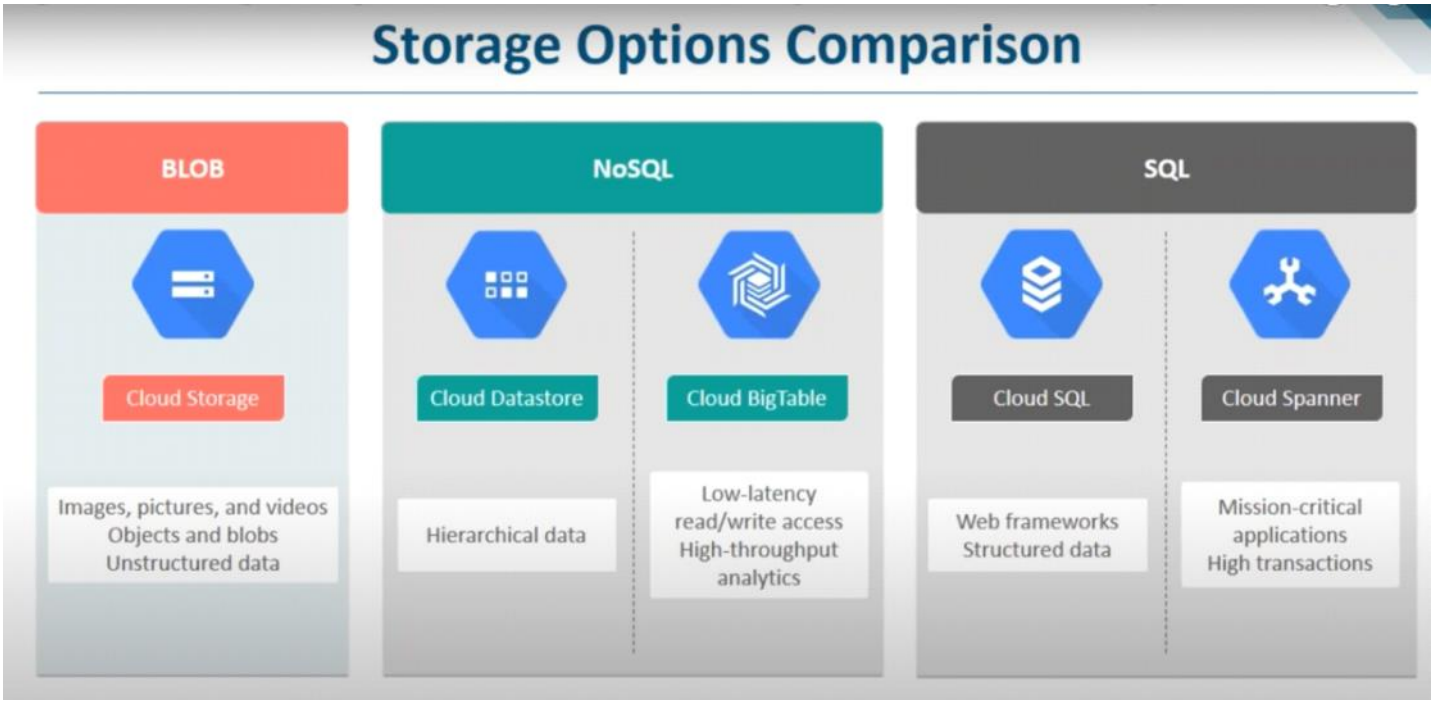
- DaysSinceNoncurrentTime
- IsLive
- MatchesStorageClass
- NoncurrentTimeBefore
- NumberOfNewerVersions

- Options for tracking Lifecycle actions: To track the lifecycle management actions that Cloud Storage takes, use one of the following options:
  - Use Cloud Storage usage logs. This feature logs both the action and who performed the action. A value of GCS Lifecycle Management in the cs_user_agent field of the log entry indicates the action was taken by Cloud Storage in accordance with a lifecycle configuration.
  - Enable Pub/Sub Notifications for Cloud Storage for your bucket. This feature sends notifications to a Pub/Sub topic of your choice when specified actions occur

Databases

Monday, January 6, 2020 7:40 PM





ACID DBs Properties:

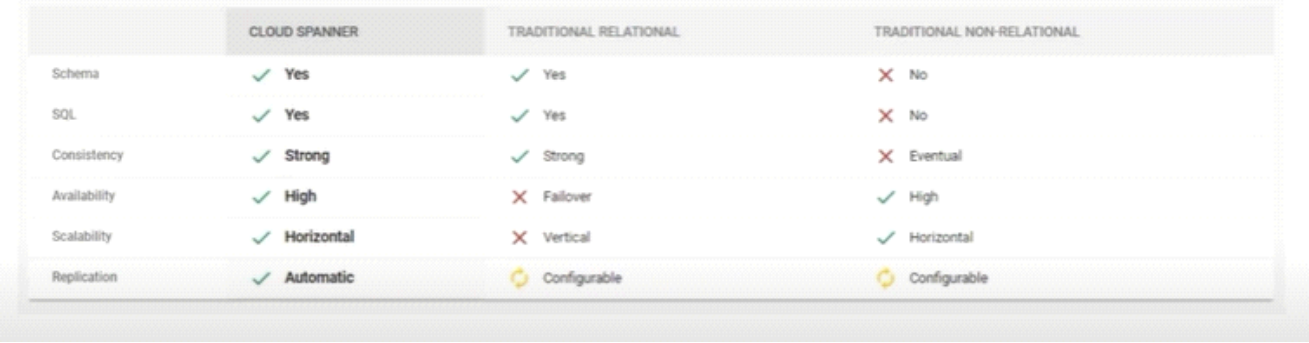
- **A**tomic transactions: Executes many queries as a single entity, such that if 1 fails, entire query should fail.
- **C**onsistency: Data that doesn't follow the DB's rules, should not be added
- **I**solation: Multiple transactions shouldn't affect each other
- **D**urability and scalability: High performance and redundancy

Cloud SQL: Managed MySQL, PostGRESQL and SQLServer databases. Auto replication, failover and backup but manual scaling. Can do automatic db backups when traffic is low or can be manually triggered.

- Failover replicas: Are used to provide high-availability for your db. Automated backups and point-in-time recovery must be enabled for high availability (point-in-time recovery uses binary logging). This can include multiple read replicas.
  - The high-availability feature is not a scaling solution for read-only scenarios; you cannot use a standby replica to serve read traffic.
- Read replicas: You use a read replica to offload work from a Cloud SQL instance. The read replica is an exact copy of the primary instance. Data and other changes on the primary instance are updated in almost real time on the read replica. Read replicas are read-only; you cannot write to them.
- You must enable point-in-time recovery to enable binary logging on the primary instance to support read replicas. Binary logging is supported on read replica instances (MySQL 5.7 and 8.0 only). You enable binary logging on a replica with the same API commands as on the primary, using the replica's instance name instead of the primary's instance name.

Cloud spanner: 1st horizontally scalable, consistent, RDBMS. Has scaling from 1-1000s of nodes. Provides consistency over cloud sql. Not based on failover, but rather any system can reply to any query. It is a transactional and relational db. Basically its like cloud SQL, but more modern wherein you don't have to worry about scaling a sql db, as it spins up more nodes based on your requirements.

Spanner Advantages



	CLOUD SPANNER	TRADITIONAL RELATIONAL	TRADITIONAL NON-RELATIONAL
Schema	✓ Yes	✓ Yes	✗ No
SQL	✓ Yes	✓ Yes	✗ No
Consistency	✓ Strong	✓ Strong	✗ Eventual
Availability	✓ High	✗ Failover	✓ High
Scalability	✓ Horizontal	✗ Vertical	✓ Horizontal
Replication	✓ Automatic	○ Configurable	○ Configurable

BigQuery: Serverless column store data warehouse for analytics using SQL Scales internally. Pay for GBs actually scanned during queries. Can caches query results for upto 24hrs. You also pay for per GB for streaming inserts. Its Is not a transactional db. Best practices: https://cloud.google.com/bigquery/docs/best-practices-costs#price_your_queries_before_running_them

External tables:

- An external data source (also known as a federated data source) is a data source that you can query directly even though the data is not stored in BigQuery. Instead of loading or streaming the data, you create a table that references the external data source.
- BigQuery offers support for querying data directly from:
  - Cloud Bigtable
  - Cloud Storage
  - Google Drive
  - Cloud SQL
- Supported formats are:
  - Avro, CSV, JSON (newline delimited only), ORC, Parquet
- Use cases for external data sources include:
  - Loading and cleaning your data in one pass by querying the data from an external data source (a location external to BigQuery) and writing the cleaned result into BigQuery storage.
  - Having a small amount of frequently changing data that you join with other tables. As an external data source, the frequently changing data does not need to be reloaded every time it is updated.
- Query performance for external data sources may not be as high as querying data in a native BigQuery table. If query speed is a priority, load the data into BigQuery instead of setting up an external data source.

Cloud Bigtable: Low latency, high throughput **NoSQL db** for large operational and analytical apps. Scales auto but processing nodes must be scaled manually. Cloud Bigtable is the most performant storage option to work with IoT and time series data. Google Cloud Bigtable is a fast, fully managed, highly-scalable NoSQL database service. It is designed for the collection and retention of data from 1TB to hundreds of PB.

- BigTable does not autoscale. BigTable does not store its data in GCS.

Bigtable is not an ideal storage option for state management. It allow use to lookup the data using a single row key.
BigTable is mutable and has fast key-based lookup whereas BigQuery is immutable and has slow key-based lookup.
You can use 'cbt' cli tool to control instances in bigtable. The ~/.cbtrc file stores the project id and instance id. The cbt createfamily creates a family of columns in the table. Cbt set <table name> <key>:<value> to store data in each column.
Cloud Bigtable has a limit of 1,000 tables per instance, but in most cases, you should have far fewer tables than that.

Cloud datastore (now called cloud firestore): Similar to cloud bigtable (is NoSql), except it provides a SQL like syntax to work with data. Must be indexed. **Has built-in indexe**s for simple fileting and sorting. Pay for IO operations (read, write and deletes) and not data stored. Cloud Datastore is not the most performant product for frequent writes or timestamp-based queries. Datastore has much better functionality around transactions and queries (since secondary indexes exist). Is ACID compliant

- Datastore can autoscale. Hence is preferred db for app engine
- It also provides a SQL-like query language.

Datastore emulator:

- The Datastore emulator provides local emulation of the production Datastore environment. You can use the emulator to develop and test your application locally. In addition, the emulator can help you generate indexes for your production Firestore in Datastore mode instance and delete unneeded indexes.
- The Datastore emulator is a component of the Google Cloud SDK's gcloud tool. Use the gcloud components install command to install the Datastore emulator:
  - gcloud components install cloud-datastore-emulator
- Start the emulator by executing datastore start from a command prompt:
  - gcloud beta emulators datastore start [flags]
  - where [flags] are optional command-line arguments supplied to the gcloud tool. For example:

    --data-dir=[DATA_DIR] changes the emulator's data directory. The emulator creates the /WEB-INF/appengine-generated/local_db.bin file inside [DATA_DIR] or, if available, uses an existing file.

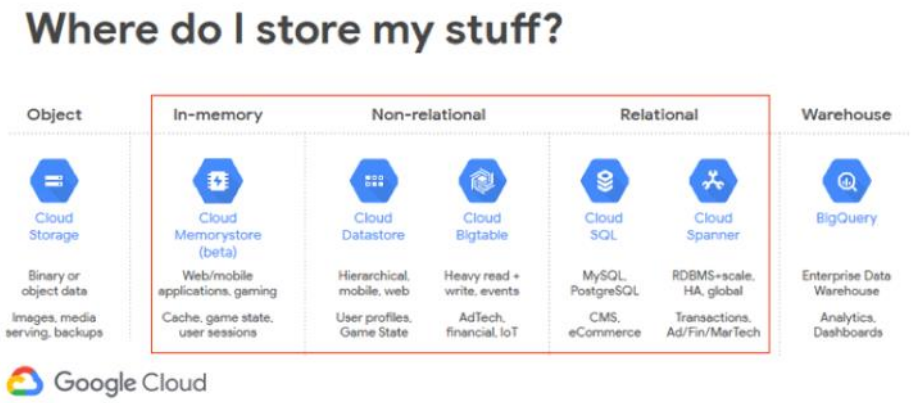
    --no-store-on-disk configures the emulator not to persist any data to disk for the emulator session.
- After you start the emulator, you need to set environment variables so that your application connects to the emulator instead of your production Datastore mode database. Set these environment variables on the same machine that you use to run your application.
  - You need to set the environment variables each time you start the emulator. The environment variables depend on dynamically assigned port numbers that could change when you restart the emulator.
  - $(gcloud beta emulators datastore env-init)
- By running your application using the emulator, you can generate indexes for your production Datastore mode database, as well as delete unneeded indexes.
- If your application and the emulator run on the same machine, you can remove the environment variables automatically:
  - Run env-unset using command substitution:
  - $(gcloud beta emulators datastore env-unset)

- Your application will now connect to your production Datastore mode database.

Firebase db: NoSQL doc store with **real-time client update**s via managed websockets. Revolves around a single JSON doc located in central US. Mostly created for app development. Provides a login interface for new and existing clients to login to the db. Data is synced across each client and remains available even if the app is offline. Using the Blaze plan you can scale across multiple locations.

Cloud memorystore: Fully managed in-memory data store to build app caches and sub-millisecond data access. You can use redis or memcache. Access is limited mostly to apps in the same location/subnet.
Both redis and memcache are powerful solutions but there are few aspects that redis does that memcache doesn't and vice-versa:

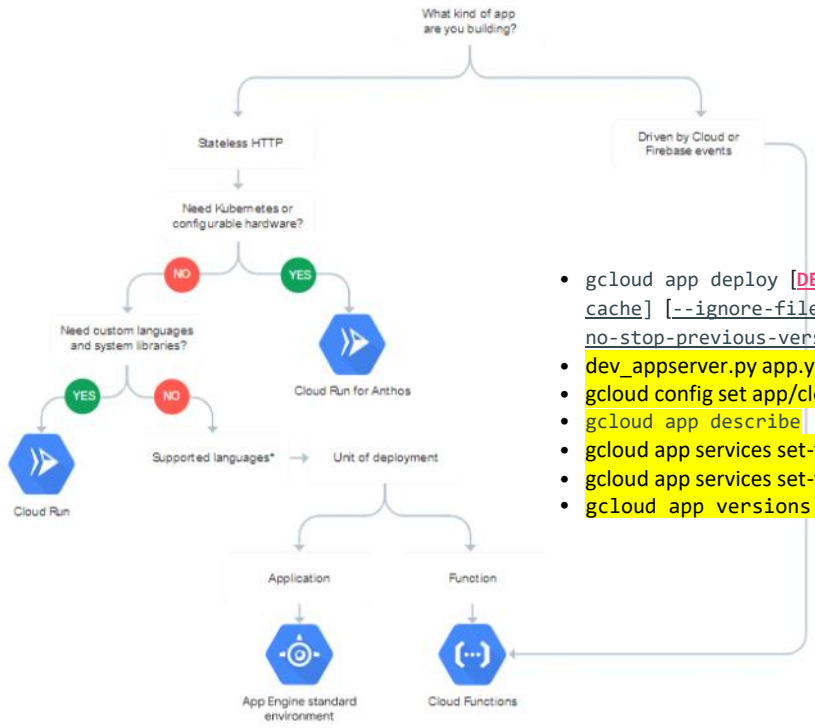
- Redis Only: Snapshots , replication, transactions, pub/sub, advanced data structures
- Memcache Only: Multi-threaded architecture and is Easier to use



Service	Transactional	Relational
BigQuery	No	Yes
Cloud SQL	Yes	Yes
Cloud spanner	Yes	Yes
Cloud bigtable	Yes	No
Cloud datastore	No	No

APP engine and cloud functions and cloud run

Monday, January 6, 2020 7:25 PM



- gcloud app deploy [DEPLOYABLES …] [--appyaml=APPYAML] [--bucket=BUCKET] [--no-cache] [--ignore-file=IGNORE_FILE] [--image-url=IMAGE_URL] [--no-promote] [--no-stop-previous-version] [--version=VERSION, -v VERSION] [GCLOUD_WIDE_FLAG …]
- dev_appserver.py app.yaml
- gcloud config set app/cloud_build_timeout 600s
- gcloud app describe
- gcloud app services set-traffic [MY_SERVICE] --splits [MY_VERSION]=1
- gcloud app services set-traffic [MY_SERVICE] --splits [MY_VERSION]=1 --migrate
- gcloud app versions list –hide-no-traffic

Differences:

- App engine is Platform as a service whereas cloud functions is Functions as a service.
- Cloud Functions server instances handle requests in a serial manner, which is not configurable whereas Cloud Run instances handle requests in a parallel manner, and the level of parallelism is configurable.
- App Engine is more suitable for applications, which have numerous functionalities inter-related even unrelated with each other e.g. microservices, while cloud functions are more events based functions and perform some single-purpose action.
- FaaS applications scale better than PaaS applications and have a number of benefits that help enterprises reduce costs. These include:
  - Reduced need for coding capabilities, fewer glitches and bugs
  - Reduced cloud vendor lock-in
  - Highly available
  - More features for fewer management resources
- Cloud run utilizes stateless applications inside Docker containers. The service can scale up and down automatically and will scale to 0 when not in use whereas the App Engine cannot scale down to 0. In Google App Engine, There is always running at least 1 instance at all times, even if your application is not getting any requests, you will be still paying for that instance.
- You can use WebSockets to create a persistent connection from a client (such as a mobile device or a computer) to an App Engine Flex instance or quite recently Cloud Run but not Cloud functions. The open connection allows two-way data exchange between the client and the server at any time, resulting in lower latency and better use of resources.
- In Cloud Run, you are **only** paying when you are processing requests, and the billing granularity is 0.1 seconds.
- Cloud Run deployments are faster as they are not running on VMs. App engine runs on VMs
- GAE Flexible and Cloud Run are very similar. They both accept container images as deployment input, they both auto-scale, and manage the infrastructure your code runs on for you. However:
  - GAE Flexible is built on VMs, therefore is slower to deploy and scale.
  - GAE Flexible does not scale to zero, at least 1 instance must be running.
  - GAE Flexible billing has 1 minute granularity, Cloud Run in 0.1 second.
- Cloud Run accepts container images built with any tool capable of building container images, as long as they respect the container contract. In particular, your code must listen for HTTP requests on the port defined by the PORT environment variable. This PORT environment variable is automatically injected by Cloud Run into your container.
- This page describes several ways to build container images:
  - Using a Dockerfile: docker build . --tag IMAGE_URL
  - Using Buildpacks (cloud build) : gcloud builds submit --tag IMAGE_URL (<gcr.io/>)
-

App Engine

- Serves as a platform for application development i.e runs your code directly with automatic integration to SQL etc but mostly for HTTP/HTTPS code.
- Natively supports python, php, nodejs, java, go, .net ruby and **app engine flex** supports any container

- App Engine standard does not support C++ application and the testing application needs to be dockerized to be used with flexible engine. A major advantage of using the App Engine flexible environment is the ability to customize the runtime.
- **Note:** Gradual traffic migration traffic between versions running in the flexible environment is not supported. You must migrate traffic immediately to versions that are running in the flexible environment.

- **Google App Engine - Standard is like a** read-only folder in which you upload your code. Read-only means there are a fixed set of libraries installed for you and you cannot deploy third-party libraries at all). DNS / Sub-domains etc are so much easier to map.
- **Google App Engine - Flexible** is like a real file-system where you have more control as compared to the Standard App engine, you have write permissions, but less as compared to GCP Compute Engine. In Flexible App Engine, you can use whatever library your app depends on.
- Google App Engine Standard cannot directly use Cloud VPN but Flexible can.

App engine Commands:

- dev_appserver.py app.yaml : The Cloud SDK includes a local development server (dev_appserver.py) that you can run locally to simulate your application running in production App Engine. The simulated environment enforces some sandbox restrictions, such as restricted system functions and Python 2 module imports, but not others, like request time-outs or quotas.
  - This is slowly being migrated to Datastore emulator
- gcloud app deploy : Deploys the app if its in the same dir. When you run it, it will ask the region where you want to deploy it.
  - `gcloud app deploy [DEPLOYABLES …] [--appyaml=APPYAML] [--bucket=BUCKET] [--no-cache] [--ignore-file=IGNORE_FILE] [--image-url=IMAGE_URL] [--no-promote] [--no-stop-previous-version] [--version=VERSION, -v VERSION] [GCLOUD_WIDE_FLAG …]`
  - `--appyaml=APPYAML`
    - Deploy with a specific app.yaml that will replace the one defined in the DEPLOYABLE.
  - `--bucket=BUCKET`
    - The Google Cloud Storage bucket used to stage files associated with the deployment. If this argument is not specified, the application's default code bucket is used.
  - `--no-cache`
    - Skip caching mechanisms involved in the deployment process, in particular do not use cached dependencies during the build step.
  - `--ignore-file=IGNORE_FILE`
    - Override the .gcloudignore file and use the specified file instead.
  - `--image-url=IMAGE_URL`
    - (App Engine flexible environment only.) Deploy with a specific Docker image. Docker url must be from one of the valid Container Registry hostnames.
  - `--promote`
    - Promote the deployed version to receive all traffic. Overrides the default `app/promote_by_default` property value for this command invocation. Use `--no-promote` to disable.
  - `--stop-previous-version`
    - Stop the previously running version when deploying a new version that receives all traffic.
    - Note that if the version is running on an instance of an auto-scaled service in the App Engine Standard environment, using `--stop-previous-version` will not work and the previous version will continue to run because auto-scaled service instances are always running.
    - Overrides the default `app/stop_previous_version` property value for this command invocation. Use `--no-stop-previous-version` to disable.
  - `--version=VERSION, -v VERSION`
    - The version of the app that will be created or replaced by this deployment. If you do not specify a version, one will be generated for you.
- gcloud config set app/cloud_build_timeout 600s: You can configure timeouts as described here only when using the App Engine flexible environment. The App Engine standard environment does not allow the build timeout to be configured.

App Engine is *regional*, which means the infrastructure that runs your apps is located in a specific region and is managed by Google to be redundantly available across all the zones within that region.

Meeting your latency, availability, or durability requirements are primary factors for selecting the region where your apps are run. You can generally select the region nearest to your app's users but you should consider the location of the other Google Cloud products and services that are used by your app. Using services across multiple locations can affect your app's latency as well as pricing.

App Engine is available in the following regions:

- `northamerica-northeast1` (Montréal)

- `us-central` (Iowa)
- `us-west2` (Los Angeles)
- `us-east1` (South Carolina)
- `us-east4` (Northern Virginia)
- `southamerica-east1` (São Paulo)
- `europe-west` (Belgium)
- `europe-west2` (London)
- `europe-west3` (Frankfurt)
- `europe-west6` (Zürich)
- `asia-northeast1` (Tokyo)
- `asia-northeast2` (Osaka)
- `asia-east2` (Hong Kong)
- `asia-south1` (Mumbai)
- `australia-southeast1` (Sydney)

You cannot change an app's region after you set it.

If you already created an App Engine application, you can view the region by running the `gcloud app describe` command or opening the App Engine Dashboard in the Cloud Console. The region of your App Engine application is listed under `http://[YOUR_PROJECT_ID].appspot.com`.

You can not use compute engine instance templates to deploy applications to Google Cloud App Engine. Google App Engine lets you deploy applications quickly by providing run time environments for many of the popular languages like Java, PHP, Node.js, Python, Ruby, and Go. You have an option of using custom runtimes but using compute engine instance templates is not an option.

To select a region, you create an App Engine application in your Google Cloud project. See *Managing Projects, Applications, and Billing* for details in your language for either the standard or flexible environment.

Instance scaling and management: https://cloud.google.com/appengine/docs/standard/python/how-instances-are-managed

- Automatic scaling
  - Automatic scaling creates instances based on request rate, response latencies, and other application metrics. You can specify thresholds for each of these metrics, as well as a minimum number instances to keep running at all times.
  - Instances are created on demand to handle requests and automatically turned down when idle.
- Basic Scaling
  - Basic scaling creates instances when your application receives requests. Each instance will be shut down when the application becomes idle. Basic scaling is ideal for work that is intermittent or driven by user activity.
  - Instances are created on demand to handle requests and automatically shut down when idle, based on the `idle_timeout` configuration parameter. An instance that is manually stopped has 30 seconds to finish handling requests before it is forcibly terminated.
- Manual scaling
  - Manual scaling specifies the number of instances that continuously run regardless of the load level. This allows tasks such as complex initializations and applications that rely on the state of the memory over time.
  - Instances are sent a start request automatically by App Engine in the form of an empty GET request to `/_ah/start`. As with basic scaling, an instance that is manually stopped has 30 seconds to finish handling requests before it is forcibly terminated.
- Manual, basic, and automatically scaling instances startup differently. When you start a manual scaling instance, App Engine immediately sends a `/_ah/start` request to each instance. When you start an instance of a basic scaling service, App Engine allows it to accept traffic, but the `/_ah/start` request is not sent to an instance until it receives its first user request. Multiple basic scaling instances are only started as necessary, in order to handle increased traffic. Automatically scaling instances do not receive any `/_ah/start` request.

Google cloud functions

- Google cloud functions is basically serverless. Runs functions NOT images/builds
- Doesn't give much room for customizing the machine
- Does autoscaling and load balancing on its own (whereas app engine needs to have autoscaling configured). Can be triggered by HTTP requests, GCS objects, PUB/SUB, IoT devices, automation etc

App Engine Deployment Files:

- YAML files act as a deployment descriptor and defines the scaling type and the runtime, handlers, and other resource settings for a specific version of a service. If you are deploying several versions of a service, you can create multiple YAML files in the same directory to represent the configuration for each of your versions.
- For a simple app you just use app.yaml, but with more complex deployments you can use others such as:
  - Dispatch.yaml : You can create a dispatch file to override App Engine's URL-based routing rules and define your own custom routing rules. With a dispatch file, you can send incoming requests to a specific service based on the path or host name in the request URL.
  - Index.yaml : If using datastore db, it needs to index every query, and needs to know in advance which queries are going to be used. Use index.yaml to specify that.

    - Cron.yaml : A cron job can trigger events either if scheduled or if the frequency is specified. When the timings are met, it sends an http get request to an url. This url and the schedule is in the cron.yaml
- For each service, you can create separate directories in the root of your app when you are developing locally. If you host your app out of a version control system (VCS), for example GitHub, you can also structure your app to use separate directories in a repository, or use separate repositories for each service.
- Each directory or repository should represent a single service and contain that service's `app.yaml` file along with the associated source code.
- Dependencies for Python applications are declared in a standard requirements.txt file. For example:

    Flask==0.10.1

    google-cloud-storage

    - When you deploy to App Engine, the dependencies specified in the requirements.txt file will be installed automatically with your deployed app.
    - By default, App Engine caches fetched dependencies to reduce build times, to install uncached use 'gcloud beta app deploy --no-cache'
    - To import private dependencies, you need to add a separate module alongside your app
        - You can use pip install -t lib priv_module to copy it to the lib dir
        - Add an empty __init__.py file to the lib dir
        - Keep the lib directory alongside your app

- To migrate traffic from 1 version to another:
    - To migrate traffic immediately: gcloud app services set-traffic [MY_SERVICE] --splits [MY_VERSION]=1
    - To gradually migrate traffic, you include the optional --migrate flag: gcloud app services set-traffic [MY_SERVICE] --splits [MY_VERSION]=1 --migrate
    - When warmup requests are enabled, traffic is migrated gradually by first sending a *warmup request* to new instances before those instances receive any user requests. Warmup requests improve user response time by allowing the version currently receiving traffic to handle those requests but the traffic migration to the new version can take a short amount of time while the new instances are created.
    - When warmup requests are not enabled, user requests are sent to those new instances before they have been created. Due to the delay caused by creating the new instances and loading application code, latency can occur for those user responses.

Splitting Traffic

You can use traffic splitting to specify a percentage distribution of traffic across two or more of the versions within a service. Splitting traffic allows you to conduct A/B testing between your versions and provides control over the pace when rolling out features.

Traffic splitting is applied to URLs that do not explicitly target a version. For example, the following URLs split traffic because they target all the available versions within the specified service:

https://PROJECT_ID.REGION_ID.r.appspot.com - Distributes traffic to versions of the default service.
https://SERVICE_ID-dot-PROJECT_ID.REGION_ID.r.appspot.com - Distributes traffic to versions of the [SERVICE_ID] service.

Splitting traffic across multiple versions:
When you have specified two or more versions for splitting, you must choose whether to split traffic by using either an IP address or HTTP cookie. It's easier to set up an IP address split, but a cookie split is more precise.

- `gcloud app services set-traffic [MY_SERVICE] --splits [MY_VERSION1]=[VERSION1_WEIGHT],[MY_VERSION2]=[VERSION2_WEIGHT] --split-by [IP_OR_COOKIE]`
    - `[VERSION_WEIGHTS] : Key-value pairs describing what proportion of traffic should go to each version. The split values are added together and used as weights. The exact values do not matter, only their relation to each other. For example, v1=2,v2=2 is equivalent to v1=.5,v2=.5 or if only v2=1 then all traffic is sent to V2 only.`

IP address splitting
If you choose to split traffic to your application by IP address, when the application receives a request, it hashes the IP address to a value between 0–999, and uses that number to route the request.

IP address splitting has some significant limitations:

1) IP addresses are reasonably sticky, but are not permanent. Users connecting from cell phones might have a shifting IP address throughout a single session. Similarly, a user on a laptop might be moving from home to a cafe to work, and will also shifting through IP addresses. As a result, the user might have an inconsistent experience with your app as their IP address changes.
2) Because IP addresses are independently assigned to versions, the resulting traffic split will differ somewhat from what you specify. Although, as your application receives more traffic, the closer the actual split gets to your target. For example, if you ask

for 5% of traffic to be delivered to an alternate version, the initial percent of traffic to the version might actually be between 3–7% but eventually averages closer to your target 5%.

3) If you need to send internal requests between apps, you should use cookie splitting instead. Requests that are sent between apps running on Google's cloud infrastructure, originate from a small number of IP addresses which are likely all assigned to the same version. Therefore, all internal requests might behave similar to requests sent from a single IP address, meaning that those requests are all routed to the same version. As a result, internal requests do not closely respect the percentages that you set for your IP-based traffic splits. For example, if you set a version to receive 1% of all the traffic to your app and the Google cloud infrastructure addresses were coincidently assigned to that version, then the actual result might far exceed 1% because all the internal requests are always routed to the assigned version. Requests sent to your app from outside of Google's cloud infrastructure will work as expected since they originate from a varied distribution of IP addresses.

Cookie splitting

If you choose to split traffic to your application by cookies, the application looks in the HTTP request header for a cookie named GOOGAPPUID, which contains a value between 0–999:

1) If the cookie exists, the value is used to route the request.
2) If there is no such cookie, the request is routed randomly.

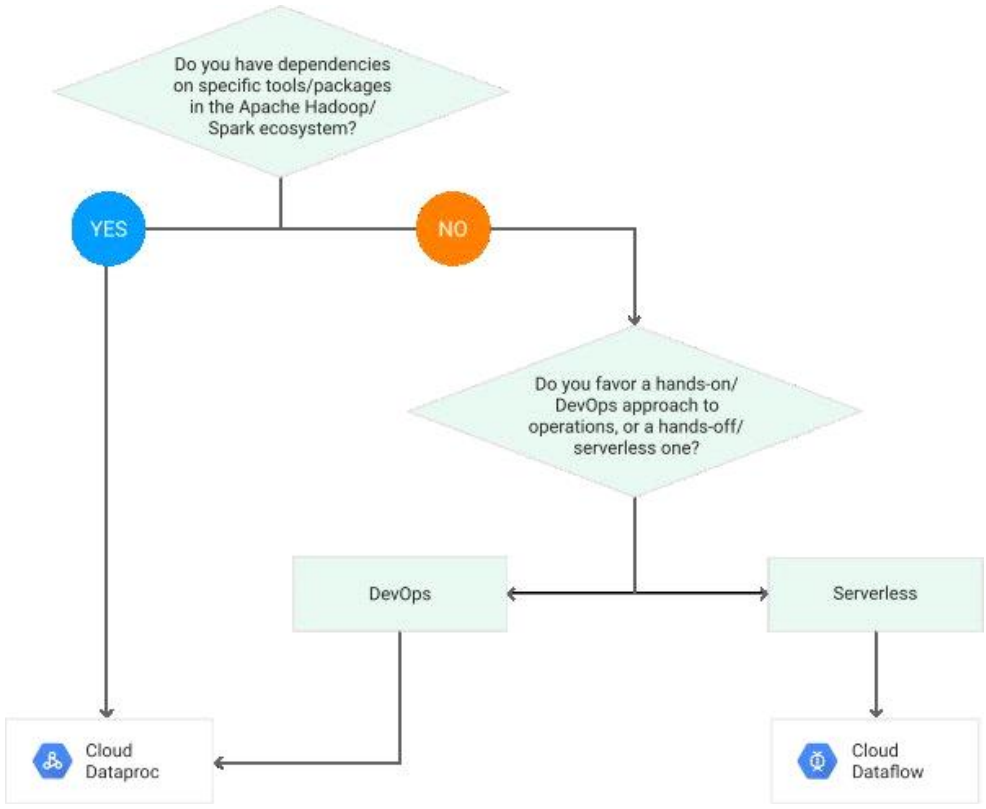
If the response does not contain the GOOGAPPUID cookie, the app first adds the GOOGAPPUID cookie with a random value between 0–999 before it is sent.

Using cookies to split traffic makes it easier to accurately assign users to versions. The precision for traffic routing can reach as close as 0.1% to the target split. Although, cookie splitting has the following limitations:

1) If you are writing a mobile app or running a desktop client, it needs to manage the GOOGAPPUID cookies. For example, when a Set-Cookie response header is used, you must store the cookie and include it with each subsequent request. Browser-based apps already manage cookies in this way automatically.
2) Splitting internal requests requires extra work. All user requests that are sent from within Google's cloud infrastructure, require that you forward the user's cookie with each request. For example, you must forward the user's cookie in requests sent from your app to another app, or to itself. Note that it is not recommended to send internal requests if those requests don't originate from a user.

Dataflow and dataproc

Wednesday, January 6, 2021 1:23 AM



- Cloud Dataproc is a fast, easy-to-use, fully managed cloud service for running Apache Spark, Apache Hadoop, HIVE, Pig clusters in a simpler, more cost-efficient way. While Dataproc is very efficient at processing ETL and Big Data pipelines, it is not as suitable for running a ruby application that runs tests each day.
- Dataflow/Beam provides a clear separation between processing logic and the underlying execution engine. This helps with portability across different execution engines that support the Beam runtime, i.e. the same pipeline code can run seamlessly on either Dataflow, Spark or Flink.

Using Dataflow SQL:

- Data flow can run SQL jobs on incoming data similar to an Apache beam pipeline machine. The data is ingested, queried using sql queries and the results can be stored in big query.
- To create a Dataflow SQL job, write and run a Dataflow SQL query.
- To configure this you need to use the console:
  - Inside BigQuery web ui->Query settings->Dataflow Engine
  - Enable Cloud dataflow APIs and Data catalog
  - You can also use Dataflow monitoring interface->Create job from SQL
- Dataflow queries are similar to BigQuery standard sql queries.
- You can Dataflow sql streaming extensions, to aggregate data from continuously updating dataflow sources:
  - Pub/Sub topics
  - Cloud storage filesets
  - BigQuery tables
- Dataflow SQL can write query results to the following destinations:
  - Pub/Sub topics
  - BigQuery tables

- Querying Pub/Sub topics: To query a Pub/Sub topic with Dataflow SQL, complete the following steps:
    - Add the Pub/Sub topic as Dataflow source.
    - Assign a schema to the Pub/Sub topic.
    - Use the Pub/Sub topic in a Dataflow SQL query.
- Querying Cloud Storage filesets: To query a Cloud Storage fileset with Dataflow SQL, complete the following steps:
    - Create a Data Catalog fileset for Dataflow SQL
    - Add the Cloud Storage fileset as Dataflow source.
    - Use the Cloud Storage fileset in a Dataflow SQL query.

Apigee and cloud endpoints

Tuesday, February 9, 2021 2:20 AM

- Endpoints is API management gateway which helps you develop, deploy, and manage APIs on any Google Cloud backend. It runs on GCP and leverages a lot of Google's underlying infrastructure. It also has native hooks to integrate with other products in GCP suite. If you have applications running on GCP, then Endpoints can enable you quick build APIs around them.
    - Apigee on the other hand is a comprehensive API management platform built for Enterprises, with deployment options on cloud, on-premises or hybrid. The feature set includes a full-fledged API gateway, customizable portal for on-boarding partners and developers, monetization, and deep analytics around your APIs. It also out of the box configurations to support traffic management, mediation, security, packaging APIs, developer key management etc. You can use Apigee for any http/https backend, no matter where they are running (on-premises, any public cloud etc.).
    - If you are using GCP, then there are use-cases for you to use both in your architecture. But if you have not GCP backend, you only use Apigee.
- Basically, both product do the same thing. But they are very different.
    - First, Endpoint, is integrated to App Engine and can be deployed elsewhere, like on Cloud Run. Endpoint has the basic features of an Endpoint Proxy: authentication, API key validation, JSON to gRPC transcoding, API monitoring, tracing and logging. Endpoint is free (or you pay only the Cloud Run when you deploy on it)
    - Apigee do the same things, but with more advance features, like quota, billing, request pre and post processing,... In addition, it has the capability to connect APIs that differ than REST and gRPC and thus can be integrated with a legacy application and allow it to expose API even if it hasn't designed for. Apigee is EXPENSIVE, but POWERFUL!

- https://cloud.google.com/endpoints/docs/openapi/about-cloud-endpoints
- https://cloud.google.com/apigee/docs/api-platform/get-started/what-apigee

Deployment manager-Google marketplace (formerly google launcher)

Thursday, February 20, 2020 11:46 PM

Google Cloud Marketplace lets you quickly deploy functional software packages that run on Google Cloud Platform. Even if you are unfamiliar with services like Compute Engine or Cloud Storage, you can easily start up a familiar software package without having to manually configure the software, virtual machine instances, storage, or network settings. Deploy a software package now, and scale that deployment later when your applications require additional capacity. Google Cloud Platform updates the images for these software packages to fix critical issues and vulnerabilities, but doesn't update software that you have already deployed.

Deploying a software package

You can select and deploy software packages from the Google Cloud Marketplace page of the Cloud Console. Google Cloud Marketplace offers many different solutions, and in some cases, offers a few variations of the same solution; for example, Google Cloud Marketplace has multiple packages for WordPress. To ensure a solution meets your needs, each solution has a details page that describes the VM type, operating system, estimated costs, and more.

Search for a package and select one that meets your business needs. When you launch the deployment, you can use the default configuration or customize the configuration to use more virtual CPUs or storage resources. Some packages allow you to specify the number of virtual machine instances to use in a cluster.

- **Cloud deployment Manager:**
  - Deployment Manager uses three different file types to describe a deployment. Each type is described below:
    - **Configurations** are YAML files that describe resources and their properties. For Identity and Access Management (IAM) custom roles, each YAML file specifies all of the properties for one or more custom roles, such as its name, description, and permissions.
    - **Schemas** specify all possible properties for a given resource and their default values. For IAM custom roles, there are schemas for project-level and organization-level custom roles.
    - **Templates**, which are Jinja or Python files that enhance configurations by allowing you to break down a configuration into reusable parts. For IAM custom roles, there are two provided templates: one for project-level custom roles, and one for organization-level custom roles.
  - Create and manage resources via templates. Show how you want to see and app. Supports json and yaml. Free to use
  - Deployment manager supports yaml, python and jinja2 format
  - A configuration is a file written in YAML syntax that lists each of the resources you want to create and its respective resource properties.
    - A configuration can contain templates, which are essentially parts of the configuration file that has been abstracted into individual building blocks. A template file is written in either Python or Jinja2.
      - Templates contain Resources. A resource represents a single API resource. This can be an API resource provided by a Google-managed base type or an API resource provided by a Type Provider. For example, a Compute Engine instance is a single resource, a Cloud SQL instance is a single resource, and so on.
  - Resources->Templates->Configuration->Deployment-Manager
  - A composite type contains one or more templates that are preconfigured to work together. These templates expand to a set of base types when deployed in a deployment. Composite types are essentially hosted templates that you can add to Deployment Manager. You can create composite types for common solutions so that the solution is easily reusable, or create complex setups that you can reuse in the future.
  - Manifest: A manifest is a read-only object that contains the original configuration you provided, including any imported templates, and also contains the fully-expanded resource list, created by Deployment Manager. Each time you update a deployment, Deployment Manager generates a new manifest file to reflect the new state of the deployment.

Deployment guide:

- With the gcloud command-line tool, use the deployments create command:
  - gcloud deployment-manager deployments create my-first-deployment \ --config vm.yaml
  - The --config flag is a relative path to your YAML configuration file.
- By default, if your configuration includes resources that are already in your project, those resources are acquired by the deployment, and can be managed using the deployment. If you don't want to acquire a resource, you must use the --create-policy option, as in the following gcloud beta command:
  - gcloud beta deployment-manager deployments create my-first-deployment \ --config vm.yaml --create-policy CREATE
- If your deployment is successfully created, you can get a description of the deployment:
  - gcloud deployment-manager deployments describe my-first-deployment
- You can use the following policies for creating your resources:
  - CREATE - Deployment Manager creates resources that do not exist. If any of the resources in your configuration already exist in the project, the deployment fails.
  - ACQUIRE - Deployment Manager acquires resources that already exist, using the same criteria as CREATE_OR_ACQUIRE.
  - Use the ACQUIRE policy if you have a number of resources already in your project, and want to manage them together, as a single deployment

- gcloud deployment-manager deployments create my-first-deployment \ --config vm.yaml
- gcloud deployment-manager deployments describe my-first-deployment
- gcloud deployment-manager deployments list

- The default policy for removing resources is DELETE.
- The default policy for updating resources is UPDATE

- With the gcloud command-line tool, use the deployments list subcommand:
  - gcloud deployment-manager deployments list
- For each deployment, you can apply one update at a time. If an update is already in progress, you must stop the current update before starting a new update.
- If you modified a resource in a deployment without using Deployment Manager, such as in the Cloud Console or gcloud, you might see errors or unexpected issues when you try to modify the resource in an update.
- You can preview the update you want to make before committing any changes, with the gcloud command -line tool or the API. The Deployment Manager service previews the configuration by expanding the full configuration and creating "shell" resources.
- Deployment Manager does not instantiate any actual resources when you preview a configuration, giving you the opportunity to see the deployment before committing to it.
  - gcloud deployment-manager deployments update example-deployment \ --config configuration-file.yaml \ --preview
- After previewing a deployment, you can fully deploy the configuration by making the same PUT() request, omitting both the configuration and the preview query parameter. Deployment Manager uses your last preview to perform the update. For example:
  - gcloud deployment-manager deployments update example-deployment
- After you preview an update, you must decide if you want to continue with the update. If you do not want to continue, or if you want to use a different configuration file to update the deployment, cancel the current preview.
  - gcloud deployment-manager deployments cancel-preview my-first-deployment

- **What is Runtime Configurator?**
- The Runtime Configurator feature lets you define and store data as a hierarchy of key value pairs in Google Cloud Platform. You can use these key value pairs as a way to:
  - Dynamically configure services
  - Communicate service states
  - Send notification of changes to data
  - Share information between multiple tiers of services
- For example, imagine a scenario where you have a cluster of nodes that run a startup procedure. During startup, you can configure your nodes to report their status to the Runtime Configurator, and then have another application query the Runtime Configurator and run specific tasks based on the status of the nodes.
- The Runtime Configurator also offers a Watcher service and a Waiter service. The Watcher service watches a specific key pair and returns when the value of the key pair changes, while the Waiter service waits for a specific end condition and returns a response once that end condition has been met.
- You use Runtime Configurator through using the gcloud tool, or as a standalone API.
- Concepts:
  - Config resource: A Config resource contains a hierarchical list of variables. You can create different configurations for different purposes. For example, you can separate configurations based on environment (prod, dev, test), based on different tiers of applications or services (back end, front end), or based on entities (one configuration per user of your application).
  - Variables: Variables are key value pairs that belong to a RuntimeConfig resource. Variable keys have the following format:
    - projects/[project_id]/configs/[CONFIG_ID]/variables/[VARIABLE_NAME]
    - You can set, get, and watch variable values to communicate information to your applications, to signal a completed state, to send notification of data changes, and more. Variables are hierarchical, so your variable key could have several levels.
    - For example, sample variable keys could be:
    - webserver-1/users/name
    - webserver-1/users/favorite_color
- Watchers: You can use the watch() method to watch a variable and return when the variable changes, the watcher times out, or the watcher is deleted. Use the watch() functionality to dynamically configure your applications based on changes in your data.
- Waiters: If you create a Waiter resource to watch a specific path prefix, the waiter returns once the number of variables under the prefix reaches a particular amount. This is referred to as a Cardinality condition.
  - For example, if you set a condition for the path /foo and the number of paths is set to 2, the following setup would meet the condition:
  - /foo/variable1 = "value1"
  - /foo/variable2 = "value2"
  - /bar/variable3 = "value3" # Not /foo path
  - A waiter has both a failure and success condition that you can set.
  - After you create a Waiter, the service returns an operation object that you poll for completion. The operation is complete when the one of the following occurs:

- The success condition is met.
- The failure condition is met.
- The Waiter reached the timeout deadline specified in the initial request.
- Using a waiter is ideal for startup scenarios, where you might need to pause a deployment until a certain number of services is running.

Encryption

Monday, February 22, 2021 2:26 AM

- Data encryption options

  Cloud Storage always encrypts your data on the server side, before it is written to disk, at no additional charge. Besides this standard, Google-managed behavior, there are additional ways to encrypt your data when using Cloud Storage. Below is a summary of the encryption options available to you:

  - *Server-side encryption*: encryption that occurs after Cloud Storage receives your data, but before the data is written to disk and stored.
    - *Customer-supplied encryption keys: You can create and manage your own encryption keys. These keys act as an additional encryption layer on top of the standard Cloud Storage encryption.*
    - *Customer-managed encryption keys: You can manage your encryption keys, which are generated for you by Cloud Key Management Service. These keys act as an additional encryption layer on top of the standard Cloud Storage encryption.*
  - *Client-side encryption: encryption that occurs before data is sent to Cloud Storage. Such data arrives at Cloud Storage already encrypted but also undergoes server-side encryption.*
  - If an object is encrypted with a customer-supplied encryption key, the key must be used to perform operations on the object such as downloading or moving it.
  - When a requester wants to read an object encrypted with a customer-managed encryption key, they simply access the object as they normally would. During such a request, the service agent automatically decrypts the requested object as long as:

- **Key replacement**
  - Use the following guidelines when replacing the key you use to encrypt Cloud Storage objects with a new key:
    a. Check your buckets to see which use the key as their default encryption key. For these buckets, replace the old key with a new key.
    b. This ensures that all objects written to the bucket use the new key going forward.
    c. Inspect your source code to understand which requests use the key in ongoing operations, such as setting bucket configurations and uploading, copying, or rewriting objects. Update these instances to use the new key.
    d. Check for objects, in all of your buckets, encrypted with the old key. Use the Rewrite Object method to re-encrypt each object with the new key.
    e. Disable all versions of the old key. After disabling old key versions, monitor client and service logs for operations that fail due to a version becoming unavailable.

ML and AI and Big data and IoT

Tuesday, January 7, 2020 12:09 AM

Cloud ML engine: Enable apps to use Tensor flow on datasets of any size.

Cloud vision api: Classifies images and is pre-trained.

Cloud speech api: Automatic speech recognition.

Cloud natural language api: analyzes sentiment, intent, content classification and extracts info. Can extract toekn, parts of speech, and dependency trees. Has more than 700+ categories.

Cloud Iot Core: Manage, connect and ingest data from devices globally.
Cloud pub/sub: Scalable for msg ingestion and decoupling.
Cloud data prep: Visually explore, create and prepare data for analysis. For business analytics, cleans data and finds missing data using ML.
Data proc: Batch map reduce via spark and hadoop clusters. Best for moving existing spark and hadoop clusters to gcp. Use cloud data flow if starting fresh in gcp.
Data flow: Smart autoscaled managed batch or stream map reduce like processing. Not completely similar to map reduce.
Cloud data lab: Tool for data exploration, analysis and visualization and ML. Uses jupyter notebooks.
Cloud data studio: Big data visualization tool for dashboards and reporting.
Cloud genomics: Store and process genomes. Similar to big query for large research experiments. Can process many experiments in parallel.

Labs commands [ref]

Wednesday, February 17, 2021 11:24 PM

- **Networking and firewall**
  - gcloud compute networks create whizlabs-pc --description=whizlabs-pc --bgp-routing-mode=regional --subnet-mode=auto
  - gcloud compute networks list - list Google Compute Engine networks
  - gcloud compute firewall-rules create allow-ssh --network=whizlabs-pc --direction=ingress --source-ranges=0.0.0.0/0 --rules=tcp:22 --action=allow --priority=65534 –no-disabled
    - Specify target-tags or target-service-accounts if needed, else applies to All instances by default
  - gcloud compute ssh whizlabs-pc
  - Gcloud compute instances whizlabs-pc describe: Describe all parameters of the instance
  - gcloud compute firewall-rules list --filter network=NETWORK_NAME : List firewall rules for a network
- **Compute engine**
  - gcloud compute instances create example-instance \ --custom-cpu 4 --custom-memory 5 --custom-vm-type n1: create an instance running an N1 machine type with 4 vCPUs and 5 GB of total memory
- **App engine**
  - Is a regional resource i.e. when deploying you can only specify region not zone
  - Gcloud app deploy
  - Read doc on usage of app.yaml and requirements.txt files
  - gcloud app versions list –hide-no-traffic : This command correctly lists just the versions that are receiving traffic by hiding versions that do not receive traffic.

    --hide-no-traffic : Only show versions that are receiving traffic.

    --service=SERVICE, -s SERVICE : Only show versions from this service.
- **Cloud run**
  - gcloud builds submit –tag gcr.io/<project_id>/<folder_name> : Build image from app and deployment code
  - gloud run deploy –image=gcr.io/<project_id>/<folder_name> --platform=<managed> –-region=us-central1 –-allow-unauthenticated <folder_name>
    - --platform: Platform to run image on either cloud run managed, or gke managed or kubernetes knative managed
    - --allow-unauthenticed: Enable unauthenticated users access. Disable with –no-allow-unauthenticated
  - Read doc on usage of app.yaml and Dockerfile
- **Cloud functions**
  - Similarly has different files for deploying functions.
  - You edit main.py for python3 to set as responding source code and its def main(): function as the entry point
  - gcloud functions deploy Name –region=region --allow-unauthenticated

        - --entry-point=entry_point : Main function inside the responding source code
        - --runtime=runtime : Runtime ex: python, nodejs etc
        - --source=source: Location of source code. .zip for GCS, any source repo, local file system
        - --trigger-<bucket=bucket/http/topic> : Different trigger types need to be mentioned as either from a gcs bucket, http or pubsub topic
        - --trigger-event=event_type : Specifies which action should trigger the function. Gcloud functions event-types list shows all event types ex:pubsub/firestore etc
- **Autoscaling**
    - Created instance and added firewall rule for ingress and egress
    - Created instance group from this instance
        - Select autoscaling mode: Autoscale, don't autoscale and autoscale only up
        - Select metric to monitor: CPU, HTTP utilization, Stackdriver monitoring metric
        - Set cooldown period
        - Min and Max instances
- **TCP Loadbalancing**
    - Follow the same steps as above
    - Reserve an external static ip address to be used on public side
    - Backend config: Select the existing instance group
        - Check http as heath check type
    - Frontend: Select the reserved public ip and add 80 as port number
- **CloudSQL**
    - Create instance and select sql type as mysql
    - Customize machine type and disk size if needed
    - Create a db inside the instance and provide root user pwd
    - Gcloud connect <sql instance name> --user=root : To connect to the sql instance from cloud shell
        - This command also adds your IP to the allowed list of IPs in sql
- **Disks and Snapshots**
    - Create a disk of 1 of the types: standard persistent, balanced persistent and ssd persistent
    - Gcloud compute images list: List all gcloud images image families
    - Gcloud compute disks list: List all disks in your project
    - Gcloud compute disks create <disk name>
        - --type=type : Type of disk to create
        - --size=size : Size of disk in GB or TB
        - --region=region, --zone=zone : Region and zone
    - Once a disk has been created, you can create either a manual snapshot or a scheduled snapshot
    - Manual snapshot:
        - Manual method to take a snapshot
        - Gcloud compute disks snapshot <disk name>
            - --snapshot-names=snapshot name : Name of snapshot
            - --storage-location=Location : Cloud storage location of the disk. Can be regional or multi-regional, but is multi-regional by default
            - --region=Region or –zone=Zone : Either region or zone needs to be selected.
    - Scheduled snapshots:
        - Gcloud compute resource-policies create snapshot-schedule [schedule name]:
            - --max-retention-days <#days> : No. Of days to retain snapshot
            - --hoursly-schedule <hourly interval in hrs>, --daily-schedule <time in 24hr format>, --weekly-schedule <actual days of the week, not case sensitive>

        - --on-source-disk-deletion [deletion option] : Action to take if source disk is deleted. Either keep-auto-snapshots or apply-retention-policy
      - Once you make the schedule, attach it to a disk using
      - Gcloud compute-disks add-resource-policies <disk name> --resource-policies <scehdule name> --zone <zone>
    - To view your snapshot schedules run: gcloud compute resource-policies list
    - You cannot edit an existing schedule. You would need to detach it from the disk, delete it and create a new disk. Deleting schedule wont delete existing snapshots
    - Each scheduled snapshot is an incremental increase from the previous one, since it will only add the blocks that have changed from the previous, not a new snapshot.
- **HTTPS loadbalancer**
  - Create instance template, instance group and add firewall rule for port 80
  - Loadbalancer create HTTP(s) loadbalancer
    - Backend service->Add the instance group created
      - Create health checks for TCP port 80
      - Host and path rules leave as is. If required you can use url based load balancing or url rewrite based
      - Frontend->Add a Reserved Static ip if created
- **Use cloud functions to load GCS data into mysql**
  - Create bucket and upload csv file
  - Create mysql instance and db
  - Create cloud functions
    - Add script in main.py and python requirements in requirement.txt
    - Deploy code
  - Your csv tables should come up in the mysql db
- **BigQuery**
  - `bq show bigquery-public-data:samples.shakespeare`
    `: Display schema of table (shakesphere) in dataset (samples) inside project (bigquery-public-data)`
  - bq query –use_legacy_sql=false : Make standard sql the default query syntax
  - `bq ls <project>:`
    `:List datasets in project. Semi-colon required at the end of the command`
    - `Bq ls : Shows just your datasets info`
    - `Bq ls <dataset name> : Shows your tables info inside that dataset`
  - `Bq show <dataset>.<table name> : List table info like scehama, rows and bytes in that dataset`
  - Bq mk <dataset> : Make a new dataset
  - Bq ls : List dataset in your project
  - bq load <datasetname>.<tables name> <txt/csv file> <schema ex: name:string,count:integer etc> : Create and load a table in your dataset with the schema you provided from file you provide
  - Bq query <query> : Run and return query results
  - bq query –dry_run <query> : Show estimate data consumption on running query

Pricing info

Thursday, February 25, 2021 8:09 PM

For pricing purposes, all units such as MB and GB represent *binary* measures. For example, 1 MB is 2^{20} bytes. 1 GB is 2^{30} bytes. These binary units are also known as mebibyte (MiB) and gibibyte (GiB), respectively. Note also that MB and MiB, and GB and GiB, are used interchangeably.

- Cloud run and cloud functions charged similarly
- Cloud spanner and bigtable charged similarly
- Data proc and data flow are close

Cloud run: Cloud Run (fully managed) charges you only for the resources you use, rounded up to the nearest 100 millisecond.

Tier	CPU	Memory	Requests	Networking
Free	First 180,000 vCPU-seconds free per month	First 360,000 GiB-seconds free per month	2 million requests free per month	1 GiB free egress within North America per month
1	$0.00002400 / vCPU-second beyond free quota ($0.00000250 if idle*)	$0.00000250 / GiB-second beyond free quota ($0.00000250 if idle*)	$0.40 / million requests beyond free quota	Google Cloud Network Premium tier pricing beyond free quota.

BigQuery:

BigQuery pricing has two main components:

- Analysis pricing is the cost to process queries, including SQL queries, user-defined functions, scripts, and certain data manipulation language (DML) and data definition language (DDL) statements that scan tables.
    - BigQuery offers a choice of two pricing models for running queries:
        - On-demand pricing. With this pricing model, you are charged for the number of bytes processed by each query.
            - The first 1 TB of query data processed per month is free.
            - $5 for every Tb after that
        - Flat-rate pricing. With this pricing model, you purchase slots, which are virtual CPUs. When you buy slots, you are buying dedicated processing capacity that you can use to run queries. Slots are available in the following commitment plans:
            - Flex slots: You commit to an initial 60 seconds.
            - Monthly: You commit to an initial 30 days.
            - Annual: You commit to 365 days.

        With monthly and annual plans, you receive a lower price in exchange for a longer-term capacity commitment.
- Storage pricing is the cost to store data that you load into BigQuery.

Operation	Pricing	Details
Active storage (1st 90days)	$0.020 per GB	The first 10 GB is free each month.
Long-term storage	$0.010 per GB	The first 10 GB is free each month.

Cloud storage: Cloud Storage pricing is based on the following components:

- Data storage: the amount of data stored in your buckets. Storage rates vary depending on the storage class of your data and location of your buckets.
- Network usage: the amount of data read from or moved between your buckets.
- Operations usage: the actions you take in Cloud Storage, such as listing the objects in your buckets.
- Retrieval and early deletion fees: applicable for data stored in the Nearline Storage, Coldline Storage, and Archive Storage classes.

Standard Storage (per GB per Month)	Nearline Storage (per GB per Month)	Coldline Storage (per GB per Month)	Archive Storage (per GB per Month)
$0.020	$0.010	$0.004	$0.0012

Compute Engine:

- All vCPUs, GPUs, and GB of memory are charged a minimum of 1 minute. For example, if you run your virtual machine for 30 seconds, you will be billed for 1 minute of usage. After 1 minute, instances are charged in 1 second increments.
- vCPU and memory usage for each machine type can receive one of the following discounts:
  - Sustained use discounts
  - Committed use discounts
  - Discounts for preemptible VM instances
- N1 machine types can receive a sustained use discount up to 30%.
- N2 machine types can receive a sustained use discount up to 20%.
- E2 machine types do not offer sustained use discounts but provide larger savings directly through the on-demand and committed-use prices.

Datastore:This is similar with Firestore too

	Free quota
Entity reads	50,000 per day
Entity writes	20,000 per day
Entity deletes	20,000 per day
Stored data	1 GB
Small operations	50,000 per day
Network egress	10 GiB per month

Dataproc pricing:

Dataproc pricing is based on the size of Dataproc clusters and the duration of time that they run. The size of a cluster is based on the aggregate number of virtual CPUs (vCPUs) across the entire cluster, including the master and worker nodes. The duration of a cluster is the length of time between cluster creation and cluster deletion.

The Dataproc pricing formula is: $0.010 * # of vCPUs * hourly duration.

Although the pricing formula is expressed as an hourly rate, Dataproc is billed by the second, and all Dataproc clusters are billed in one-second clock-time increments, subject to a 1-minute minimum billing.

Dataflow: Each Dataflow job uses at least one Dataflow worker. The Dataflow service provides two worker types: batch and streaming. There are separate service charges for batch and streaming workers.

Dataflow workers consume the following resources, each billed on a per second basis.

VCPU, Memory, Storage: Persistent Disk, GPU (optional)

Cloud SQL: Pricing for Cloud SQL depends on your instance type:

- SQL Server: Charged for compute and memory, storage, network and licensing (Microsoft).
- MySQL: Charges for Instances and Networking. Cloud SQL charges per second for instances and rounds-off to the nearest integer (0.49s->0s). Egress network charges depend on the destination.
- PostgreSQL: Charged for compute and memory, storage, network and instances.

Cloud Bigtable: Are charged for the following:

- The type of Cloud Bigtable instance and the total number of nodes in your instance's clusters.
- The amount of storage that your tables use.
- The amount of network bandwidth that you use.
- Storage and bandwidth usage are measured in binary gigabytes (GB), where 1 GB is 2^30 bytes. This unit of measurement is also known as a gibibyte (GiB).

Pub/Sub pricing is based on:

- Message ingestion and delivery:
    - Charged for volume of data transmitted in a month. First 10gb is free but then $40 per TiB
    - A minimum of 1000 bytes per publish, push or pull request is assessed regardless of message size. This means that for messages smaller than 1000 bytes, it is cheaper to batch multiple messages per request.
    - Ingestion and delivery charges apply only to publish requests and data delivered using pull, streamingPull or push operations. Other operations are free.
    - Data egress fee is also applied similar to GCE, but there is zone fees and egress to Google products is not exempt.
- Seek-related message storage: snapshots and retained acknowledged messages
    - Message storage fees, at a rate of $0.27 per GiB-month, are charged in these cases:
    - A subscription is configured to retain acknowledged messages to make it possible to re-process them using seek. In this case, storage fees are charged for retained acknowledged messages.
    - A snapshot of a subscription is created. In this case, message storage fees are charged for storing the snapshot's unacknowledged messages. In addition, if the subscription has a backlog of unacknowledged messages when the snapshot is created, a one-time fee equivalent to storing that backlog for seven days is charged.
    - Snapshots can be a more economical way to retain messages for replay than retaining all acknowledged messages because a single snapshot can be used across multiple subscriptions. Snapshots generally have a small billable data volume that increases gradually through the age of the snapshots.
    - Subscriptions configured to retain acknowledged messages maintain a fixed time window of message data (at steady state), but may be more convenient.

Cloud operations: Monitoring, Logging and Trace

Logging:

Feature	Price[1]	Free allotment per month	Effective date
Logging data	$0.50/GiB	First 50 GiB/project	July 1, 2018

Monitoring:

Feature	Price[1]	Free allotment per month	Effective date
Monitoring data	$0.2580/MiB: 150–100,000 MiB $0.1510/MiB: 100,000–250,000 MiB $0.0610/MiB: >250,000 MiB	All GCP metrics[2] First 150 MiB per billing account for chargeable metrics (Metric data from Google Cloud, Anthos, and Knative isn't chargeable.)	July 1, 2018
Monitoring API calls	$0.01/1,000 Read API calls (Write API calls are free)	First 1 million Read API calls included per billing account	July 1, 2018

Trace:

FEATURE	PRICE	FREE ALLOTMENT PER MONTH	EFFECTIVE DATE
Trace Ingestion	$0.20/million spans	First 2.5 million spans	November 1, 2018

Cloud Spanner: Charges for No. Of Nodes in your instance, Amount of storage, Amont of backup storage and Network bandwidth.

Cloud functions: Cloud Functions are priced according to how long your function runs, how many times it's invoked and how many resources you provision for the function. If your function makes an outbound network request, there are also additional data transfer fees.

- Invocations charges:

Invocations per Month	Price/Million
First 2 million	Free
Beyond 2 million	$0.40

- Compute resources used: From time it receives a request till its completed

Memory	CPU[1]	Price/100ms (Tier 1 Price)
128MB	200MHz	$0.000000231
256MB	400MHz	$0.000000463
512MB	800MHz	$0.000000925
1024MB	1.4 GHz	$0.000001650
2048MB	2.4 GHz	$0.000002900
4096MB	4.8 GHz	$0.000005800

- Networking: Outbound data transfer (that is, data transferred from your function out to somewhere else) is measured in GB and charged at a flat rate. Outbound data to other Google APIs in the same region is free, as is inbound data. Google APIs that are global (i.e. not region-specific) are considered to be the same region.

Type	Price/GB
Outbound Data (Egress)	$0.12
Outbound Data per month	5GB Free

Inbound Data (Ingress)	Free
Outbound Data to Google APIs in the same region	Free

- Local Disk: Cloud Functions provides access to a local disk mount point (/tmp) which is known as a "tmpfs" volume in which data written to the volume is stored in memory. There is no specific fee associated with this however writing data to the /tmp mountpoint will consume memory resources provisioned for the function.

GKE is charges in 2 ways:

- Autopilot: Autopilot clusters accrue a flat fee of $0.10/h per cluster for each cluster after the free tier, plus the CPU, memory and ephemeral storage compute resources that are provisioned for your Pods. The amount of pod resources provisioned is based on the resource requests of the Kubernetes PodSpec. You are not charged for system Pods, the operating system overhead, or unallocated space. All Autopilot resources are charged in 1 second increments, there is no minimum charge.
- Standard: Clusters created in Standard mode accrue a management fee of $0.10 per cluster per hour, irrespective of cluster size or topology, after the free tier. GKE cluster management fees do not apply to Anthos clusters. In Standard mode, GKE uses Compute Engine instances worker nodes in the cluster. You are billed for each of those instances according to Compute Engine's pricing, until the nodes are deleted. Compute Engine resources are billed on a per-second basis with a one-minute minimum usage cost.

Some best practices

Sunday, January 19, 2020 8:25 PM

- Object lifecycle in GCS: https://cloud.google.com/storage/docs/lifecycle

***Hands-on practice*

Get extra practice with Google Cloud through self-paced exercises covering a single topic or theme offered via Qwiklabs.

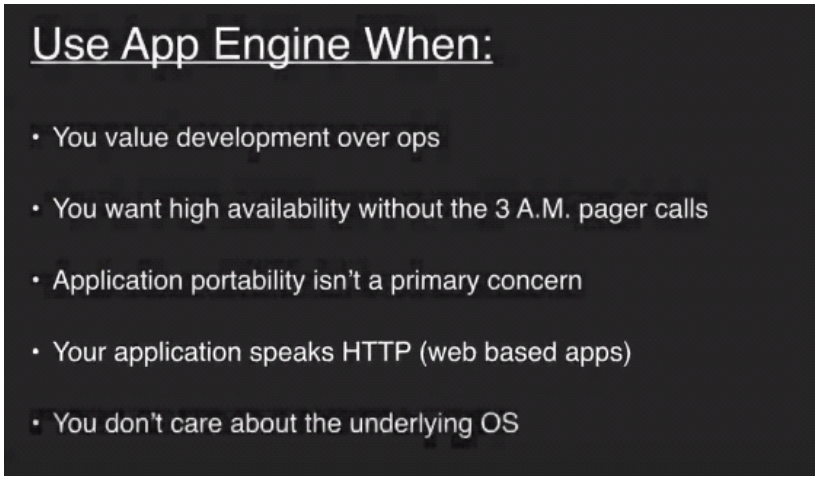
Complete the recommended quests and labs:

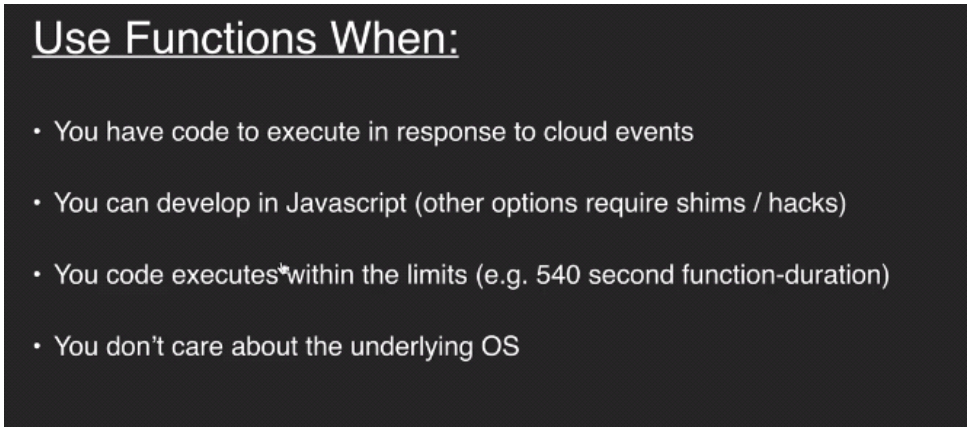
- Quest: Getting Started: Create & Manage Resources
- Quest: Perform Foundational Infrastructure Tasks in Google Cloud
- Quest: Setup and Configure a Cloud Environment in Google Cloud
- Quest: Build and Secure Networks in Google Cloud
- Quest: Deploy to Kubernetes in Google Cloud
- Hands-on lab: Cloud Run - Hello Cloud Run

- The binary log contains "events" that describe database changes such as table creation operations or changes to table data. It also contains events for statements that potentially could have made changes (for example, a DELETE which matched no rows), unless row-based logging is used. The binary log also contains information about how long each statement took that updated data. The binary log is not used for statements such as SELECT or SHOW that do not modify data.

3 pillars of any cloud service: Compute, Network, Storage

1) Compute resources and when to use them:





Use Kubernetes Engine When:

- You're running containerized workloads
- You care about containers rather than the underlying OS
- You care about application portability
- You want to focus on containers as a unit of deployment

Use Compute Engine When:

- You need control over the OS
- You need control over CPU, GPU, SSDs, RAM, etc…
- You need to lift-and-shift an existing application
- You're workload isn't containerized
- You're performing bulk processing and require preemptible instances

App Engine is used to develop highly scalable web based applications

Cloud Functions are used to run JavaScript code in response to cloud events

Kubernetes Engine is used to run containerized applications

Compute Engine is used for anything, and everything else that you'd run on a VM

When Would You Use Kubernetes Engine?

Reasons to use containers

High level of application portability

Avoid dev vs. prod discrepancies

Help migrate from on-premises to the cloud