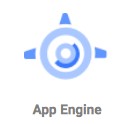
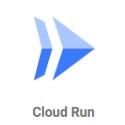
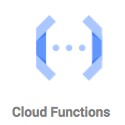
# Google Certified Associate Cloud Engineer

~~1~~



## Getting Started

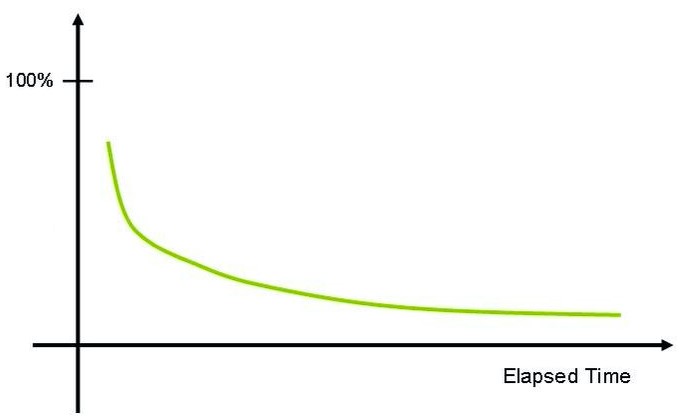


GCP has *200+ services*. This exam expects knowledge of *40+ Services*. Exam *expects* **in-depth knowledge** about these services Exam *tests* your **decision making abilities**: Which service do you choose in which situation?

This course is **designed** to give you *in-depth* knowledge & *make tough choices*

**Our Goal** : Enable you to understand and use GCP in your real world projects!

**How do you put your best foot forward?**

 **Challenging certification** - Expects you to understand and **REMEMBER** a number of services

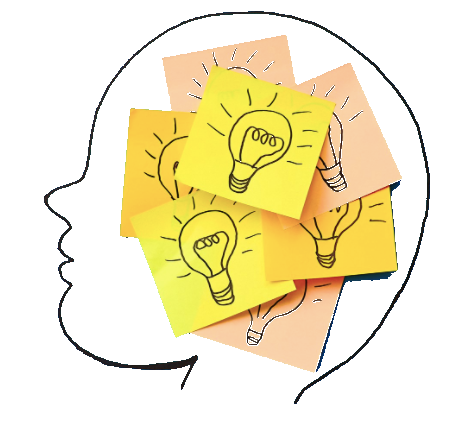
As time passes, humans forget things.

How do you improve your chances of remembering things?

**Active learning** - think and take notes

**Review** the presentation every once in a while

## Our Approach

 Three-pronged approach to reinforce concepts:

Presentations (Video) Demos (Video) **Two kinds of quizzes**:

Text quizzes

Video quizzes

(Recommended) Take your time. Do not hesitate to replay videos!

(Recommended) Have Fun!

# GCP - Getting started

## Before the Cloud - Example 1 - Online Shopping App



Challenge:

Peak usage during holidays and weekends

Less load during rest of the time Solution (before the Cloud):

**PEAK LOAD provisioning** : **Procure** (Buy) infrastructure **for peak load**

What would the infrastructure be doing during periods of low loads?

## Before the Cloud - Example 2 - Startup



Challenge:

Startup suddenly becomes popular

How to handle the **sudden increase** in load?

Solution (before the Cloud):

**Procure** (Buy) infrastructure assuming they would be successful

What if they are not successful?

## Before the Cloud - Challenges



High cost of procuring infrastructure

Needs ahead of time planning (**Can you guess the future?**)

Low infrastructure utilization (**PEAK LOAD** provisioning)

Dedicated infrastructure maintenance team (**Can a startup afford it?**)

## Silver Lining in the Cloud

 How about **provisioning (renting) resources** when you want them and releasing them back when you do not need them?

**On-demand resource provisioning**

Also called **Elasticity**

## Cloud - Advantages

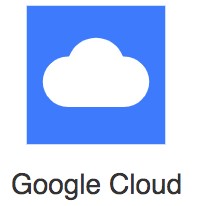
Trade **"capital expense"** for **"variable expense"**

Benefit from massive **economies of scale**

Stop **guessing** capacity

Stop spending money running and maintaining data centers **"Go global"** in minutes

## Google Cloud Platform (GCP)

**One of the Top 3** cloud service providers Provides a number of services (200+) Reliable, secure and highly-performant:

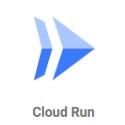
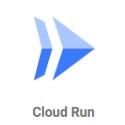
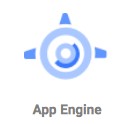
Infrastructure that powers 8 services with over 1 Billion Users: Gmail, Google Search, YouTube etc

One thing I love : **"cleanest cloud"**

Net carbon-neutral cloud (electricity used matched 100% with renewable energy)

The entire course is all about GCP. You will learn it as we go further.

**Best path to learn GCP!**



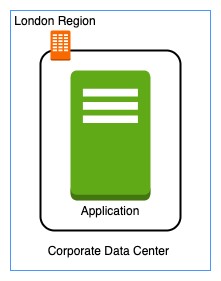
Cloud applications make use of multiple GCP services There is **no single path** to learn these services independently HOWEVER, we've worked out a simple path!

## Setting up GCP Account

Create GCP Account

# Regions and Zones

## Regions and Zones

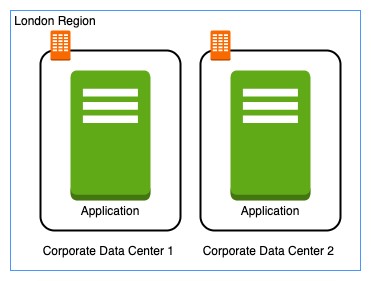


Imagine that your application is deployed in a data center in London What would be the challenges?

Challenge 1 : Slow access for users from other parts of the world (**high latency**) Challenge 2 : What if the data center crashes?

Your application goes down (**low availability**)

### Multiple data centers



Let's **add in one more data center** in London What would be the challenges?

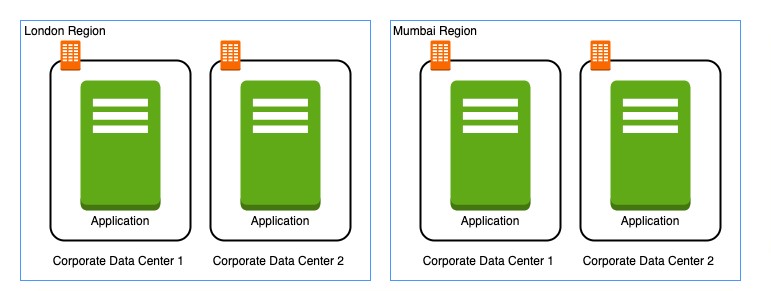
Challenge 1 : Slow access for users from other parts of the world Challenge 2 (**SOLVED**) : What if one data center crashes?

Your application is **still available** from the other data center

Challenge 3 : What if **entire region** of London is unavailable?

Your application goes down

### Multiple regions



Let's add a new region : Mumbai What would be the challenges?

Challenge 1 (**PARTLY SOLVED**) : Slow access for users from other parts of the world

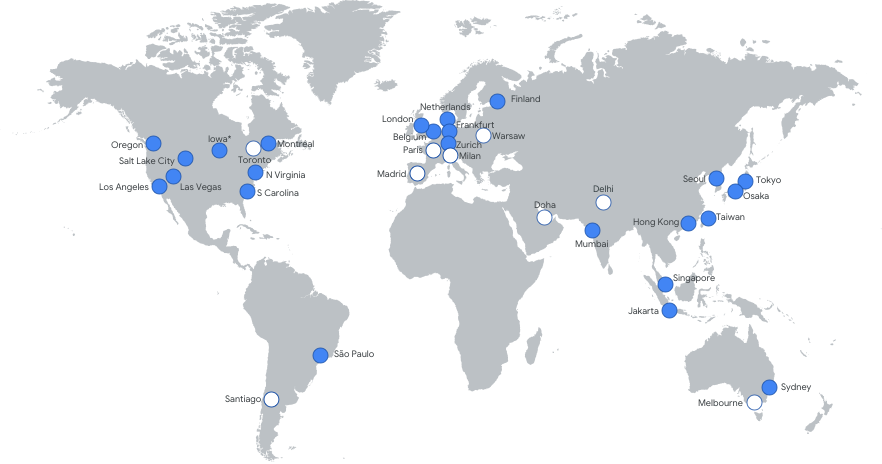
You can solve this by adding deployments for your applications in other regions Challenge 2 (SOLVED) : What if one data center crashes?

Your application is still live from the other data centers

Challenge 3 (**SOLVED**) : What if entire region of London is unavailable?

Your application is served from Mumbai

## Regions

 Imagine setting up data centers in different regions around the world  **Would that be easy?**

(Solution) Google provides **20+ regions** around the world Expanding every year

**Region** : Specific geographical location to host your resources **Advantages**:

High Availability

Low Latency

Global Footprint

Adhere to government **regulations**

## Zones

 How to achieve high availability in the same region (or geographic location)?

Enter **Zones**

Each Region has three or more **zones**

(Advantage) **Increased availability and fault tolerance** within same region

(Remember) Each Zone has **one or more**

**discrete clusters**

**Cluster** : distinct physical infrastructure that is housed in a data center

(Remember) Zones in a region are connected

through **low-latency** links

## Regions and Zones examples

*New Regions and Zones are constantly added*

**Region Code Region Zones Zones List**



**us-west1** The Dalles, Oregon, North America 3 us-west1-a

us-west1-b us-west1-c



**europe-north1** Hamina, Finland, Europe 3 europe-north1-a, europe-north1-b europe-north1-c



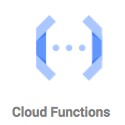
**asia-south1** Mumbai, India APAC 3 asia-south1-a, asia-south1-b

asia-south1-c



# Compute Compute Engine Fundamentals

## Google Compute Engine (GCE)

 In corporate data centers, applications are deployed to physical servers

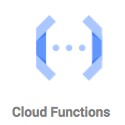
Where do you deploy applications in the cloud?

Rent virtual servers

**Virtual Machines** - Virtual servers in GCP

**Google Compute Engine (GCE)** - Provision & Manage Virtual Machines

## Compute Engine - Features



Create and manage lifecycle of Virtual Machine (VM) instances

**Load balancing** and **auto scaling** for multiple VM instances

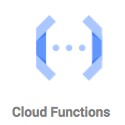
**Attach storage** (& network storage) to your VM instances

Manage **network connectivity and configuration** for your VM instances **Our Goal**:

Setup VM instances as HTTP (Web) Server

Distribute load with Load Balancers

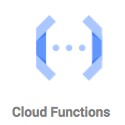
## Compute Engine Hands-on

Let's create a few VM instances and play with them

Let's check out the lifecycle of VM instances

Let's use SSH to connect to VM instances

## Compute Engine Machine Family

What type of hardware do you want to run your workloads on?

Different Machine Families for Different Workloads: **General Purpose (E2, N2, N2D, N1)** : Best price-performance ratio

Web and application servers, Small-medium databases, Dev environments

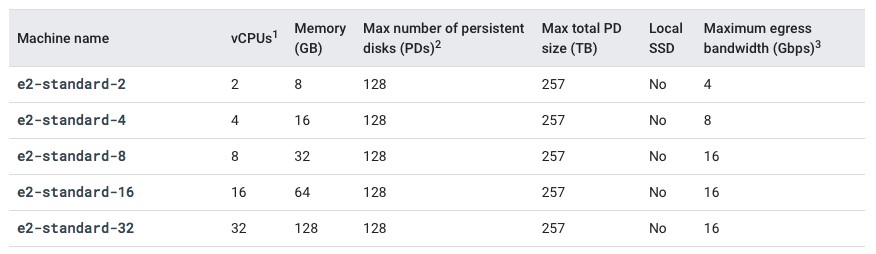
**Memory Optimized (M2, M1)**: Ultra high memory workloads

Large in-memory databases and In-memory analytics

**Compute Optimized (C2)**: Compute intensive workloads

Gaming applications

## Compute Engine Machine Types



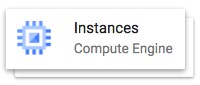
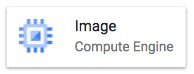
How much CPU, memory or disk do you want?

Variety of machine types are available for each machine family Let's take an example : **e2-standard-2**:

**e2** - Machine Type Family **standard** - Type of workload **2** - Number of CPUs

Memory, disk and networking capabilities increase along with vCPUs

## Image



What operating system and what so ware do you want on the instance?

Type of Images:

**Public Images**: Provided & maintained by Google or Open source communities or third party vendors

**Custom Images**: Created by you for your projects

## Compute Engine Hands-on : Setting up a HTTP server

|  |
| --- |
| #! /bin/bash sudo su apt update  apt -y install apache2 sudo service apache2 start sudo update-rc.d apache2 enable  echo "Hello World" > /var/www/html/index.html  echo "Hello world from $(hostname) $(hostname -I)" > /var/www/html/index.html |

Commands:

sudo su - execute commands as a root user

apt update - Update package index - pull the latest changes from the APT repositories

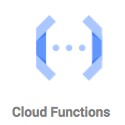
apt -y install apache2 - Install apache 2 web server sudo service apache2 start - Start apache 2 web server

echo "Hello World" > /var/www/html/index.html - Write to index.html

$(hostname) - Get host name

$(hostname -I) - Get host internal IP address

## Internal and External IP Addresses

**External** (Public) IP addresses are **Internet addressable**.

**Internal** (Private) IP addresses are **internal** to a corporate network You CANNOT have two resources with same public (External) IP address.

HOWEVER, two different corporate networks CAN have resources with same Internal (private) IP address

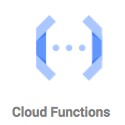
All **VM instances** are assigned at least one Internal IP address

Creation of External IP addresses can be enabled for VM instances

(Remember) When you stop an VM instance, External IP address is lost

**DEMO**: VM instances - Internal and External IPs

## Static IP Addresses

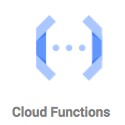
 Scenario : How do you get a constant External IP address for a VM instance?

Quick and dirty way is to assign an Static IP Address to the VM!

**DEMO**: Using Static IP Address with an VM instance

* GCP > VPC N/W > external IP addresses > Reserve static IP

## Static IP Addresses - Remember

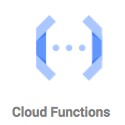
Static IP **can be switched** to another VM instance in same project

Static IP **remains attached** even if you stop the instance. You have to manually detach it.

Remember : You are **billed for** an Static IP when **you are NOT using it**!

Make sure that you explicitly release an Static IP when you are not using it.

## Simplify VM HTTP server setup

 How do we **reduce** the **number of steps** in creating an VM instance and setting up a HTTP Server? Let's explore a few options:

**Startup script**

**Instance Template**

**Custom Image**

## Bootstrapping with Startup script

|  |
| --- |
| **#!/bin/bash apt update**  **apt -y install apache2**  **echo "Hello world from $(hostname) $(hostname -I)" > /var/www/html/index.html** |

**Bootstrapping**: Install OS patches or so ware when an VM instance is launched.

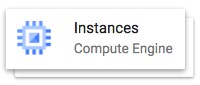
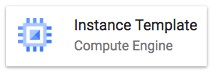
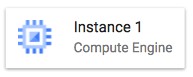
In VM, you can configure **Startup script** to bootstrap

**DEMO** - Using Startup script

* GCP > create VM > under management > startup-script > paste your script

## Instance templates

Why do you need to specify all the VM instance details (Image, instance type etc) **every time** you launch an instance?



How about creating a **Instance template**?

Define **machine type, image, labels, startup script** and other properties

Used to create **VM instances** and **managed instance groups**

Provides a **convenient way** to create similar instances

**CANNOT** be updated

To make a change, copy an existing template and modify it

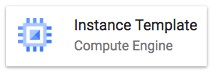
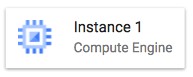
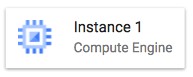
(Optional) Image family can be specified (example - debian-9):

Latest non-deprecated version of the family is used

**DEMO** - Launch VM instances using Instance templates

## Reducing Launch Time with Custom Image

Installing OS patches and so ware at launch of VM instances **increases boot up** time



How about creating a custom image with OS patches and so ware **pre-installed**?

Can be created from an instance, a persistent disk, a snapshot, another image, or a file in Cloud Storage

Can be shared across projects

(Recommendation) Deprecate old images (& specify replacement image)

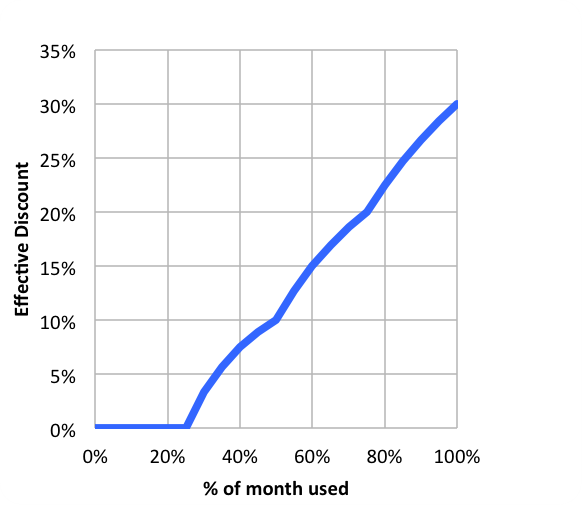
(Recommendation) **Hardening an Image** - Customize images to your corporate security standards

**Prefer** using **Custom Image** to **Startup script**

**DEMO** : Create a Custom Image and using it in an Instance Template

* **GCP > Compute engine > Disk > 3 dot > create image**

## Sustained use discounts

 **Automatic discounts** for running VM instances for significant portion of the billing month Example: If you use N1, N2 machine types for more than 25% of a month, you get a 20% to 50% discount on every incremental minute.

Discount increases with usage (graph) No action required on your part!

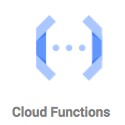
**Applicable** for instances created by **Google Kubernetes Engine** and **Compute Engine**

**RESTRICTION**: Does NOT apply on certain

machine types (example: E2 and A2)

**RESTRICTION**: Does NOT apply to VMs created Source: [***https://cloud.google.com***](https://cloud.google.com/) by App Engine flexible and Dataflow

## Committed use discounts

For workloads with **predictable resource** needs

**Commit** for 1 year or 3 years

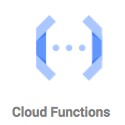
**Up to 70% discount** based on machine type and GPUs

**Applicable** for instances created by **Google Kubernetes Engine** and **Compute Engine**

**RESTRICTION**: Does NOT apply to VMs created by App Engine flexible and Dataflow

* GCP > VM > committed use discount > purchase a commitment

## Preemptible VM

**Short-lived cheaper** (up to 80%) compute instances

Can be stopped by GCP any time (preempted) within 24 hours

Instances get 30 second warning (to save anything they want to save)

**Use Preempt VM's** if:

Your applications are **fault tolerant**

You are very **cost sensitive**

Your workload is **NOT immediate**

Example: Non immediate batch processing jobs

**RESTRICTIONS**:

NOT always available

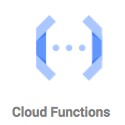
NO SLA and CANNOT be migrated to regular VMs

NO Automatic Restarts

Free Tier credits not applicable

**GCP > VM instance > create new > management > availability policy > pre-emptibility**

## Google Compute Engine - Billing

You are **billed by the second** (after a minimum of 1 minute)

You are NOT billed for compute when a compute instance is stopped

However, you will be billed for any storage attached with it!

(RECOMMENDATION) **Always create Budget alerts** and make use of Budget exports to stay on top of billing!

What are the ways you can save money?

Choose the right machine type and image for your workload

Be aware of the discounts available:

Sustained use discounts

Committed use discounts

Discounts for preemptible VM instances

## Compute Engine : Live Migration & Availability Policy

How do you keep your VM instances running when a host system needs to be updated (a so ware or a hardware update needs to be performed)?

**Live Migration**

Your running instance is migrated to another host in the same zone

Does NOT change any attributes or properties of the VM

SUPPORTED for instances with local SSDs

NOT SUPPORTED for GPUs and preemptible instances

Important Configuration - **Availability Policy**:

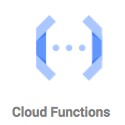
**On host maintenance**: What should happen during periodic infrastructure maintenance?

Migrate (default): Migrate VM instance to other hardware

Terminate: Stop the VM instance

**Automatic restart** - Restart VM instances if they are terminated due to non-user-initiated reasons (maintenance event, hardware failure etc.)

## Compute Engine Features: Custom Machine Types

 What do you do **when predefined VM options are NOT appropriate** for your workload?

Create a machine type customized to your needs (a **Custom Machine Type**)

**Custom Machine Type**: **Adjust** vCPUs, memory and GPUs

Choose between E2, N2, or N1 machine types

Supports a wide variety of Operating Systems: CentOS, CoreOS, Debian, Red Hat, Ubuntu, Windows etc

**Billed per vCPUs, memory provisioned** to each instance

Example Hourly Price: $0.033174 / vCPU + $0.004446 / GB

## Compute Engine Features: GPUs

 How do you accelerate math intensive and graphics-intensive workloads for AI/ML etc?

Add a **GPU** to your virtual machine:

High performance for math intensive and graphics-intensive workloads Higher Cost

(REMEMBER) Use **images with GPU libraries** (Deep Learning) installed

OTHERWISE, GPU will not be used

**GPU restriction**s:

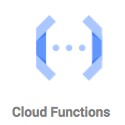
**NOT supported on all machine types** (For example, not supported on shared-core or memory-optimized machine types)

**On host maintenance** can only have the value "Terminate VM instance"

Recommended **Availability policy** for GPUs

Automatic restart - on

## Virtual Machine - Remember

Associated with a **project**

Machine type **availability can vary** from region to regions

You can only change the machine type (adjust the number of vCPUs and memory) of a stopped instance

You CANNOT change the machine type of a running instance

VM's **can be filtered** by various properties

Name, Zone, Machine Type, Internal/External IP, Network, Labels etc

Instances are Zonal (Run in a **specific zone** (in a specific region))

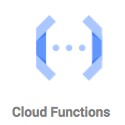
Images are global (You can provide access to other projects - if needed)

Instance templates are global (Unless you use zonal resources in your templates)

**Automatic Basic Monitoring** is enabled

Default Metrics: CPU utilization, Network Bytes (in/out), Disk Throughput/IOPS

For Memory Utilization & Disk Space Utilization - Cloud Monitoring agent is needed

**Virtual Machine - Best Practices**

Choose **Zone and Region** based on:

Cost, Regulations, Availability Needs, Latency and Specific Hardware needs

Distribute instances in multiple zones and regions for high availability

Choose **right machine type** for you needs:

Play with them to find out the right machine type Use **GPUs** for Math and Graphic intensive applications

Reserve for **"committed use discounts"** for constant workloads

Use preemptible instances for fault-tolerant, NON time critical workloads

Use **labels** to indicate environment, team, business unit etc

## Compute Engine Scenarios

**Scenario Solution**



**What are the pre-requisites to be able to create a VM instance?** 1. Project

1. Billing Account
2. Compute Engines APIs should be enabled



**You want dedicated hardware for your compliance,** Sole-tenant nodes

**licensing, and management needs**



**I have 1000s of VM and I want to automate OS patch management,** Use "VM Manager"

**OS inventory management and OS configuration management (manage so ware installed)**



**You want to login to your VM instance to install so ware** You can SSH into it



**You do not want to expose a VM to internet** Do NOT assign an external IP

Address



**You want to allow HTTP traffic to your VM** Configure Firewall Rules



## Quick Review

**Image**

What **operating system** and what **so ware** do you want on the VM instance? Reduce boot time and improve security by creating custom **hardened Images**.

You can share an Image with other projects

**Machine Types**

Optimized combination of compute(CPU, GPU), memory, disk (storage) and networking for specific workloads.

You can **create your own Custom Machine Types** when existing ones don't fit your needs

**Static IP Addresses**: Get a constant IP addresses for VM instances

**Instance Templates**: Pre-configured templates simplifying the creation of VM instances

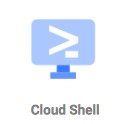
**Sustained use discounts**: **Automatic discounts** for running VM instances for significant portion of the billing month

**Committed use discounts**: 1 year or 3 year **reservations** for workloads with **predictable resource** needs

**Preemptible VM**: Short-lived cheaper (upto 80%) compute instances for non-time-critical fault-tolerant workloads

# Gcloud

## Gcloud

**Command line interface** to interact with Google Cloud Resources Most GCP services can be managed from CLI using Gcloud:

Compute Engine Virtual Machines

Managed Instance Groups

Databases and ... many more

You can create/delete/update/read existing resources and perform actions like deployments as well!

(REMEMBER) SOME GCP services have specific CLI tools:

Cloud Storage - gsutil

Cloud BigQuery - bq

Cloud Bigtable - cbt

Kubernetes - kubectl (in addition to Gcloud which is used to manage clusters)

## Gcloud - Getting Started

**Installation**

Gcloud is part of Google Cloud SDK

Cloud SDK requires Python

Instructions to install Cloud SDK (and Gcloud) => [***https://cloud.google.com/sdk/docs/install***](https://cloud.google.com/sdk/docs/install)

You can also use Gcloud on Cloud Shell

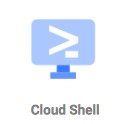
**Connecting to GCP**

* **gcloud init** - initialize or reinitialize gcloud
  + Authorize gcloud to use your user account credentials
  + Setup configuration

Includes current project, default zone etc

**gcloud config list** - lists all properties of the active configuration

## gcloud config set

Sets the specified property in your active configuration

**gcloud config set core/project VALUE**

**gcloud config set compute/region VALUE**

**gcloud config set compute/zone VALUE**

* **gcloud config set core/verbosity VALUE(debug)**
* Syntax - **gcloud config set SECTION/PROPERTY VALUE** 
  + **core**, **compute** - SECTIONS
  + **project**, **region**, **zone** - PROPERTIES
  + Specifying **core** is optional as it is the default SECTION!
    - **gcloud config set project VALUE**
    - **gcloud config set Verbosity VALUE(debug)**

Get more details with **gcloud config set --help**

Look for AVAILABLE PROPERTIES in the content

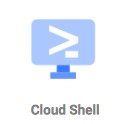
Opposite - **gcloud config unset**

## Playing with gcloud config set

gcloud config set compute/region us-east2 gcloud config set compute/zone us-east1-b gcloud config list

|  |
| --- |
| testing@cloudshell:~ (useful-device-303710)$ gcloud config list  [component\_manager]  disable\_update\_check = True  [compute]  gce\_metadata\_read\_timeout\_sec = 30 region = us-east1 zone = us-east1-b  [core]  account = testing@gmail.com disable\_usage\_reporting = True project = useful-device-303710  verbosity = info [metrics]  environment = devshell |

## Gcloud - Managing Multiple Configurations

 Scenario: You are working on multiple projects from the same machine. You would want to be able to execute commands using different configurations. How do you simplify this?

**gcloud config configurations create/delete/describe/activate/list**

Create new configuration: **gcloud config configurations create NAME(dev)**

Activate specific configuration: **gcloud config configurations activate NAME(dev)**

List Configurations: **gcloud config configurations list**

## gcloud command structure - Playing with Services

**gcloud GROUP SUBGROUP ACTION ...**

GROUP - config or compute or container or dataflow or functions or iam or .. Which service group are you playing with?

SUBGROUP - instances or images or instance-templates or machine-types or regions or zones

Which sub group of the service do you want to play with?

ACTION - create or list or start or stop or describe or ...

What do you want to do?

**Examples**:

gcloud compute instances list gcloud compute zones list gcloud compute regions list gcloud compute machine-types list

gcloud compute machine-types list --filter="zone:us-central1-b"

gcloud compute machine-types list --filter="zone:( us-central1-b europe-west1-d )"

## gcloud compute instances create

Creating Compute Instances **gcloud compute instances create [NAME]**

Options:

--machine-type (default type is n1-standard-1 - gcloud compute machine-types list) --custom-cpu --custom-memory --custom-vm-type(n1/n2) (Custom Machine)

--custom-cpu 6 --custom-memory 3072MB --custom-vm-type n2

--image or --image-family or --source-snapshot or --source-instance-template or --source-machine-image (beta) --service-account or --no-service-account

--zone=us-central1-b

--tags (List of tags - Allow network firewall rules and routes to be applied to VM instances) --preemptible

--restart-on-failure(default) --no-restart-on-failure --maintenance-policy(MIGRATE(default)/TERMINATE)

--boot-disk-size, --boot-disk-type --boot-disk-auto-delete(default) --no-boot-disk-auto-delete

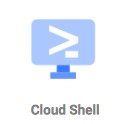
--deletion-protection --no-deletion-protection(default) --metadata/metadata-from-file startup-script/startup-script-url

--metadata-from-file startup-script=/local/path/to/script/startup OR --metadata startup-script="echo 'hello world'" shutdown-script

--network --subnet --network-tier (PREMIUM (default), STANDARD)

--accelerator="type=nvidia-tesla-v100,count=8" --metadata="install-nvidia-driver=True" (GPU)

## Compute Instances - Default Region and Zone

Three Options:

Option 1 (Centralized Configuration): gcloud compute project-info add-metadata

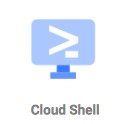
--metadata=[google-compute-default-region=REGION | google-compute-default-zone=ZONE]

Option 2 (Local gcloud configuration): gcloud config set compute/region REGION

Option 3 (Command Specific): --zone or --region in the command

Priority: Option 3 (if exists) overrides Option 2 (if exists) overrides Option 1

## List and Describe commands

Typically list commands are used to list a set of resources gcloud compute RESOURCES list

gcloud compute images/regions/zones/disk-types list

gcloud compute instances/disks/snapshots list

Most list commands support a few common options

--filter="zone:VALUE"

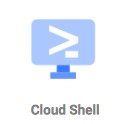
--sort-by (NAME, ~NAME)

[--uri](https://www.googleapis.com/compute/v1/projects/windows-sql-cloud/global/images/sql-2019-web-windows-2019-dc-v20210112)  [(***https://www.googleapis.com/compute/v1/projects/windows-sql-cloud/global/images/sql-2019-webwindows-2019-dc-v20210112***)](https://www.googleapis.com/compute/v1/projects/windows-sql-cloud/global/images/sql-2019-web-windows-2019-dc-v20210112)  gcloud compute images list --sort-by NAME --filter "PROJECT:(windows-cloud ubuntu-os-cloud)"

Typically describe commands are used to describe a specific resource gcloud compute images describe ubuntu-1604-xenial- v20210203 --project ubuntuos-cloud

gcloud compute regions describe us-central1

## Playing with Compute Instances - gcloud

Playing with compute instances

gcloud compute instances list/start/stop/delete/reset/describe/move gcloud compute instances start example-instance

gcloud compute instances stop example-instance-1 example-instance-2 gcloud compute instances delete example-instance

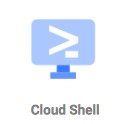
--delete-disks=VALUE (all or data or boot)

--keep-disks=VALUE (all or data or boot)

gcloud compute instances move example-instance-1 --zone us-central1-b --destination-zone uscentral1-f

Move a VM from one zone to another

## Playing with Instance Templates

gcloud compute instance-templates create/delete/describe/list gcloud compute instance-templates create INSTANCE-TEMPLATE

--source-instance=SOURCE\_INSTANCE --source-instance-zone (Which instance to create a template from?)

Supports almost all options supported by **gcloud compute instances create [NAME]**

--image or --image-family or --source-snapshot or --source-instance-template --service-account or --no-service-account

--tags

--preemptible

--restart-on-failure(default) --no-restart-on-failure --maintenance-policy(MIGRATE(default)/TERMINATE)

--boot-disk-size, --boot-disk-type --boot-disk-auto-delete(default) --no-boot-disk-auto-delete

--deletion-protection --no-deletion-protection(default)

--metadata/metadata-from-file startup-script/startup-script-url

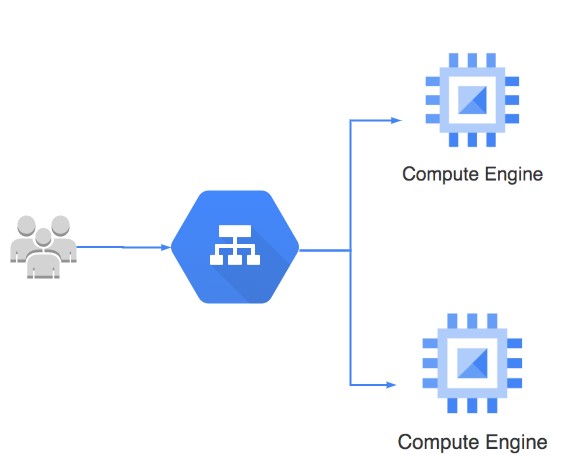
--network --subnet --network-tier (PREMIUM (default), STANDARD)

--accelerator="type=nvidia-tesla-v100,count=8" --metadata="install-nvidia-driver=True" (GPU)

Using Instance Tempate to create an instance gcloud compute instances create my-test-vm --sourc e-instance-template=my-instance-templatewith-custom-image

# Instance Groups

## Instance Groups

How do you create a group of VM instances?

**Instance Group** - Group of VM instances managed as a single entity

**Manage group** of similar VMs having similar lifecycle as **ONE UNIT**

**Two Types of Instance Groups**:

**Managed** : Identical VMs created using a template:

Features: Auto scaling, auto healing and managed releases

**Unmanaged** : Different configuration for VMs in same group:

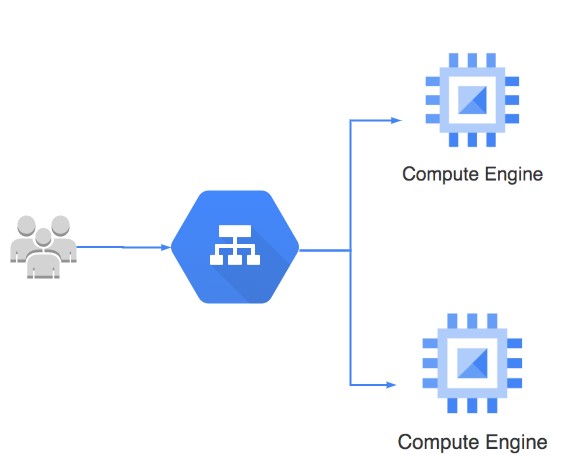
Does NOT offer auto scaling, auto healing & other services

NOT Recommended unless you need different kinds of VMs

**Location** can be Zonal or Regional

Regional gives you higher availability (RECOMMENDED)

## Managed Instance Groups (MIG)

 **Managed Instance Group** - Identical VMs created

using an **instance template** Important Features:

**Maintain** certain number of instances

If an instance crashes, MIG launches another instance

**Detect application failures** using health checks (Self Healing)

Increase and decrease instances based on load **(Auto Scaling)**

Add **Load Balancer** to distribute load

Create instances in multiple zones (regional MIGs) Regional MIGs provide higher availability compared to zonal MIGs

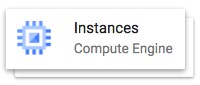
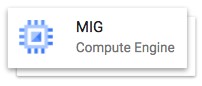
**Release** new application versions without downtime

**Rolling updates**: Release new version step by step (gradually). Update a percentage of instances to the new version at a time.

**Canary Deployment**: Test new version with a group of instances before releasing it across all instances.

## Creating Managed Instance Group (MIG)

**Instance template** is mandatory



Configure **auto-scaling** to automatically adjust number of instances based on load:

**Minimum** number of instances

**Maximum** number of instances

**Autoscaling metrics**: CPU Utilization target or Load Balancer Utilization target or Any other metric from Stack Driver

**Cool-down period**: How long to wait before looking at auto scaling metrics again?

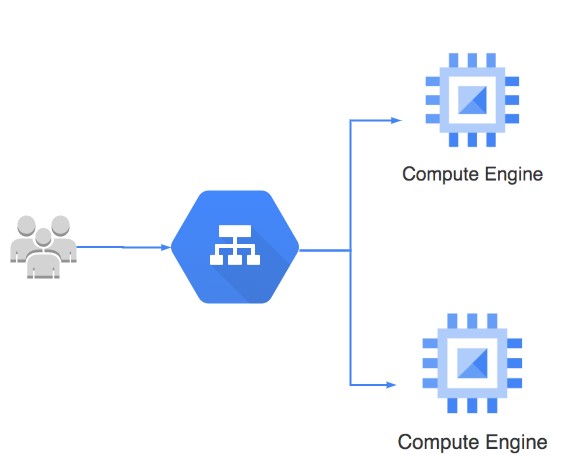
**Scale In Controls**: Prevent a sudden drop in no of VM instances

**Example**: Don't scale in by more than 10% or 3 instances in 5 minutes

**Autohealing**: Configure a Health check with Initial delay (How long should you wait for your app to initialize before running a health check?)

Time for a **Demo**

## Updating a Managed Instance Group (MIG)

 **Rolling update** - Gradual update of instances in an instance group to the new instance template Specify new template:

(OPTIONAL) Specify a template for canary testing

Specify how you want the update to be done: When should the update happen?

Start the update immediately (Proactive) or when instance group is resized later(Opportunistic)

How should the update happen?

**Maximum surge**: How many instances are added at any point in time?

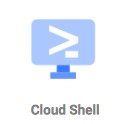
**Maximum unavailable**: How many instances can be offline during the update?

**Rolling Restart/replace**: Gradual restart or replace of all instances in the group

No change in template BUT replace/restart existing VMs

Configure Maximum surge, Maximum unavailable and What you want to do? (Restart/Replace)

## Playing with Managed Instance Groups - Command Line

gcloud compute instance-groups managed

**Create instance group**: create  *gcloud compute instance-groups managed* ***create*** *my-mig --zone us-central1-a --template myinstance-template --size 1*

**--health-check**=HEALTH\_CHECK: How do you decide if an instance is healthy?

**--initial-delay**: How much time should you give to an instance to start?

**Other similar commands** - *gcloud compute instance-groups managed* ***delete/describe/list***

**Setup Autoscaling**: set-autoscaling/stop-autoscaling *gcloud compute instance-groups managed* ***set-autoscaling*** *my-mig* ***--max-num-replicas****=10*

--cool-down-period (default - 60s): How much time should Auto Scaler wait a er initiating an autoscaling action?

--scale-based-on-cpu --target-cpu-utilization --scale-based-on-load-balancing --target-load-balancing-utilization

**--min-num-replicas** --mode (off/on(default)/only-scale-out)

*gcloud compute instance-groups managed* ***stop-autoscaling*** *my-mig*  **Update existing MIG policies** (ex: auto healing policies):

*gcloud compute instance-groups managed* ***update*** *my-mig*

--initial-delay: How much time should you give to the instance to start before marking it as unhealthy?

--health-check: How do you decide if an instance is healthy?

## Managed Instance Group - Command Line - Making Updates

**Resize the group**:

*gcloud compute instance-groups managed* ***resize*** *my-mig* ***--size=5***

**Recreate one or more instances** (delete and recreate instances):

*gcloud compute instance-groups managed* ***recreate-instances*** *my-mig* ***--instances=myinstance-1,my-instance-2***

**Update specific instances**:

*gcloud compute instance-groups managed* ***update-instances*** *my-mig --instances=my-instance-*

*3,my-instance-4 (Update specific instances from the group)*

**--minimal-action**=none(default)/refresh/replace/restart

**--most-disruptive-allowed-action**=none(default)/refresh/replace/restart

**Update instance template**:

*gcloud compute instance-groups managed* ***set-instance-template*** *my-mig* ***--template=v2template***

A er updating instance template, you can trigger roll out of the new template using update-instances, recreateinstances or rolling-action start-update commands

## Managed Instance Groups - Command Line - Rolling Actions

**Scenario**: You want to manage your new release - v1 to v2 - without downtime *gcloud compute instance-groups managed* ***rolling-action***

**Restart(stop & start)**- *gcloud compute instance-groups* ***managed rolling-action restart*** *mymig*

--max-surge=5 or 10% (Max no of instances updated at a time)

**Replace(delete & recreate)**- *gcloud compute instance-groups* ***managed rolling-action replace*** *my-mig*

--max-surge=5 or 10% (Max no of instances updated at a time)

--max-unavailable=5 or 10% (Max no of instances that can be down for the update)

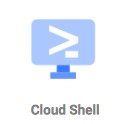
--replacement-method=recreate/substitute (recreate creates instances with new names. substitute reuses names)

**Updates instances** to a new template:

**Basic Version** (Update all instances slowly step by step) - *gcloud compute instance-groups managed* ***rollingaction start-update*** *my-mig --version=template=v1-template*

**Canary Version** (Update a subset of instances to v2) - *gcloud compute instance-groups managed* ***rolling-action start-update*** *my-mig --version=template=v1-template --canary-version=template=v2-template,target-size=10%*  Options: --max-surge, --max-unavailable, --replacement-method

## Playing with Managed Instance Groups - Scenarios

I want to ensure that I have one healthy instance running all the time:

*gcloud compute instance-groups managed* ***set-autoscaling*** *my-group --max-numreplicas=1 --min-num-replicas=1*

I want to make a new release with no reduction in available number of instances. I want to update one instance at a time:

*gcloud compute instance-groups managed* ***rolling-action start-update*** *my-group -version=template=my-v1-template --max-surge 1 --max-unavailable 0*

## Instance Group Scenarios

**Scenario Solution**



**You want MIG managed application** Create multiple zone MIG (or regional MIG) **to survive Zonal Failures**



**You want to create VMs of different** Create Un-managed Instance Group **configurations in the same group**



**You want to preserve VM state in an Stateful MIG** - Preserve VM state (Instance name, attached Persistent

**MIG** disks and Metadata). Recommended for stateful workloads (database,

data processing apps)



**You want high availability in an MIG** Use an instance template with availability policy **even when there are** automatic restart: enabled & on-host maintenance: migrate **hardware/so ware updates** Ensures live migration and automatic restarts



**You want unhealthy instances to be** Configure health check on the MIG (self healing) **automatically replaced**

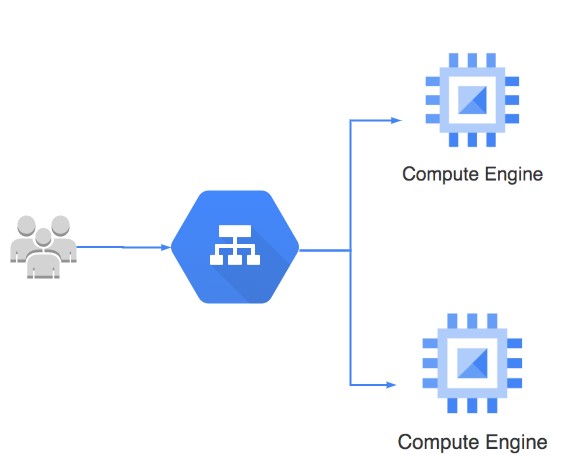


**Avoid frequent scale up & downs** Cool-down period/Initial delay



# Cloud Load Balancing

## Cloud Load Balancing

 Distributes user traffic across instances of an application in single region or multiple regions

**Fully distributed, so ware defined** managed service Important Features:

Health check - Route to healthy instances

Recover from failures

Auto Scaling

Global load balancing with single anycast IP Also supports internal load balancing

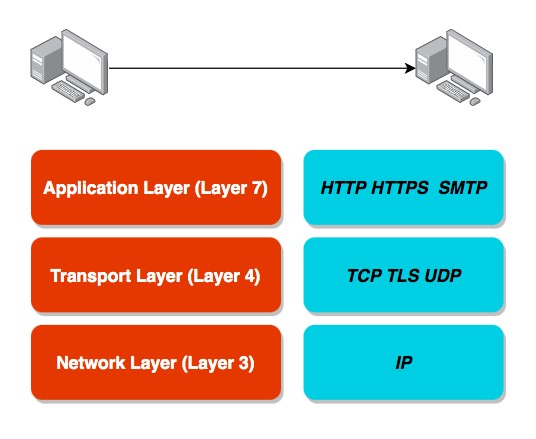
Enables:

High Availability

Auto Scaling

Resiliency

## HTTP vs HTTPS vs TCP vs TLS vs UDP

Computers use protocols to communicate

Multiple layers and multiple protocols

**Network Layer** - Transfer bits and bytes

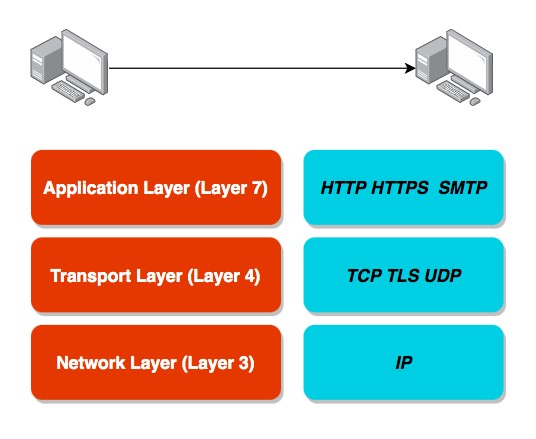
**Transport Layer** - Are the bits and bytes transferred properly?

**Application Layer** - Make REST API calls and Send Emails

(Remember) Each layer makes use of the layers beneath it

(Remember) Most applications talk at application layer. BUT some applications talk at transport layer directly(high performance).

### HTTP vs HTTPS vs TCP vs TLS vs UDP

Network Layer:

IP (Internet Protocol): Transfer bytes. Unreliable.

Transport Layer:

TCP (Transmission Control): Reliability > Performance

TLS (Transport Layer Security): Secure TCP

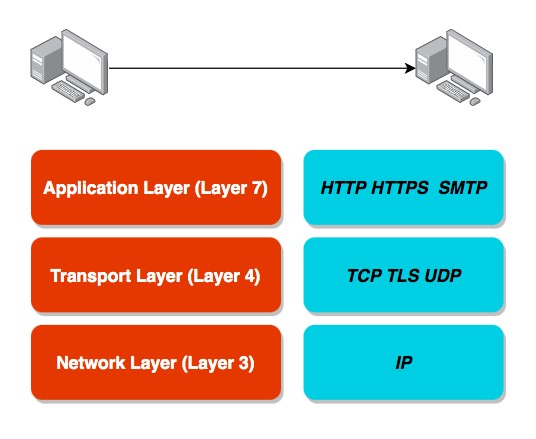
UDP (User Datagram Protocol): Performance > Reliability Application Layer:

HTTP(Hypertext Transfer Protocol): Stateless Request

Response Cycle

HTTPS: Secure HTTP SMTP: Email Transfer Protocol and a lot of others...

### HTTP vs HTTPS vs TCP vs TLS vs UDP

 **Most applications** typically communicate at application layer

Web apps/REST API(HTTP/HTTPS), Email Servers(SMTP), File Transfers(FTP)

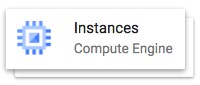
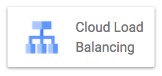
All these applications use TCP/TLS at network layer(for reliability)

**HOWEVER** applications needing high performance **directly** communicate at transport layer: Gaming applications and live video streaming use UDP

(sacrifice reliability for performance)

**Objective**: Understand Big Picture. Its OK if you do not understand all details.

## Cloud Load Balancing - Terminology



**Backend** - Group of endpoints that receive traffic from a Google Cloud load balancer (example: instance groups)

**Frontend** - Specify an IP address, port and protocol. This IP address is the frontend IP for your clients requests. For SSL, a certificate must also be assigned.

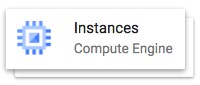
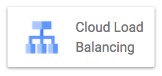
**Host and path rules** (For HTTP(S) Load Balancing) - Define rules redirecting the traffic to different backends:

Based on **path** - in28minutes.com/a vs in28minutes.com/b

Based on **Host** - a.in28minutes.com vs b.in28minutes.com

Based on **HTTP headers** (Authorization header) and methods (POST, GET, etc) etc..

## Load Balancing - SSL/TLS Termination/Offloading



Client to Load Balancer: Over internet

HTTPS recommended

Load Balancer to VM instance: Through Google internal network

HTTP is ok. HTTPS is preferred.

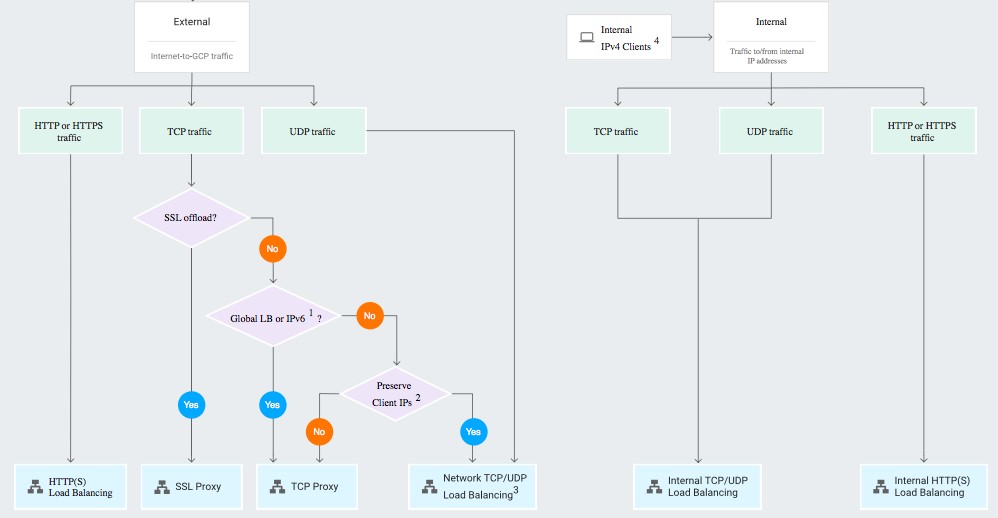
SSL/TLS Termination/Offloading

Client to Load Balancer: HTTPS/TLS

Load Balancer to VM instance: HTTP/TCP

## Cloud Load Balancing - Choosing Load Balancer

[***https://cloud.google.com/load-balancing/images/choose-lb.svg***](https://cloud.google.com/load-balancing/images/choose-lb.svg)



## Cloud Load Balancing - Features

**Load Balancer Type of Traffic Proxy or pass- Destination Ports**

**through**



**External HTTP(S)** Global, External, HTTP or HTTPS Proxy HTTP on 80 or

8080

HTTPS on 443



**Internal HTTP(S)** Regional, Internal, HTTP or HTTPS Proxy HTTP on 80 or

8080

HTTPS on 443



**SSL Proxy** Global, External, TCP with SSL offload Proxy A big list



**TCP Proxy** Global, External, TCP without SSL Proxy A big list

offload



**External Network** Regional, External, TCP or UDP Pass-through any

**TCP/UDP**



**Internal TCP/UDP** Regional, Internal, TCP or UDP Pass-through any



## Load Balancer Scenarios

**Scenario Solution**



**You want only healthy instances to** Configure health check **receive traffic**



**You want high availability for your VM** Create Multiple MIGs for your VM instances in multiple regions. Load **instances** balance using a Load Balancer.



**You want to route requests to** Create individual MIGs and backends for each microservice. **multiple microservices using the** Create Host and path rules to redirect to specific microservice **same load balancer** backend based on the path (/microservice-a, /microservice-b etc).

You can route to a backend Cloud Storage bucket as well.



**You want to load balance Global** Choose External HTTP(S) Load Balancer **external HTTPS traffic across backend instances, across multiple regions**



**You want SSL termination for Global** Choose SSL Proxy Load Balancer **non-HTTPS traffic with load balancing**



# Managed Services

## Managed Services

 Do you want to continue **running applications in the cloud**, the **same way you run them** in your **data center**?

OR **are there OTHER approaches**?

You should **understand some terminology** used with cloud services:

**IaaS** (Infrastructure as a Service)

**PaaS** (Platform as a Service)

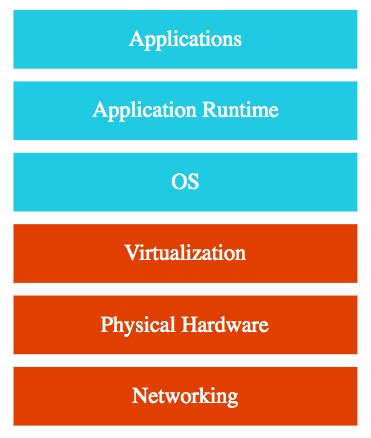
**FaaS** (Function as a Service)

**CaaS** (Container as a Service)

**Serverless**

Let's get on a quick **journey** to understand these!

## IAAS (Infrastructure as a Service)

Use **only infrastructure** from cloud provider

**Example**: Using VM to deploy your applications or databases

You are responsible for:

Application Code and Runtime

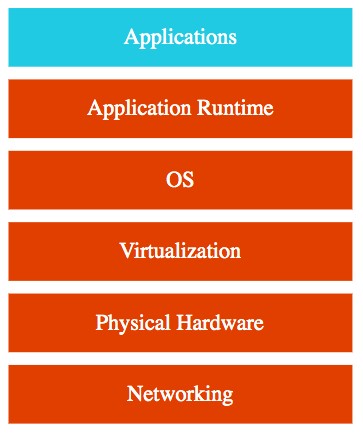
Configuring load balancing

Auto scaling

OS upgrades and patches Availability

etc.. ( and a lot of things!)

## PAAS (Platform as a Service)

Use a platform provided by cloud **Cloud provider** is responsible for:

OS (incl. upgrades and patches)

Application Runtime

Auto scaling, Availability & Load balancing etc.. **You** are responsible for:

Configuration (of Application and Services)

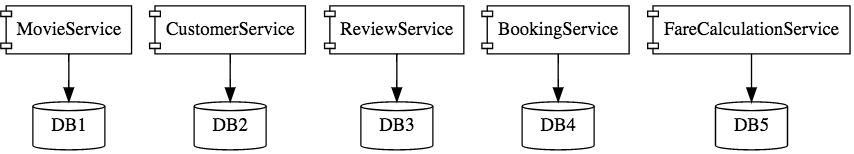
Application code (if needed) Varieties:

**CAAS (Container as a Service)**: Containers instead of Apps

**FAAS (Function as a Service)**: Functions instead of Apps

Databases - Relational & NoSQL (Amazon RDS, Google Cloud SQL, Azure SQL Database etc), Queues, AI, ML, Operations etc!

## Microservices



Enterprises are heading towards microservices architectures

Build small focused microservices

**Flexibility to innovate** and build applications in different programming languages (Go, Java, Python, JavaScript, etc)

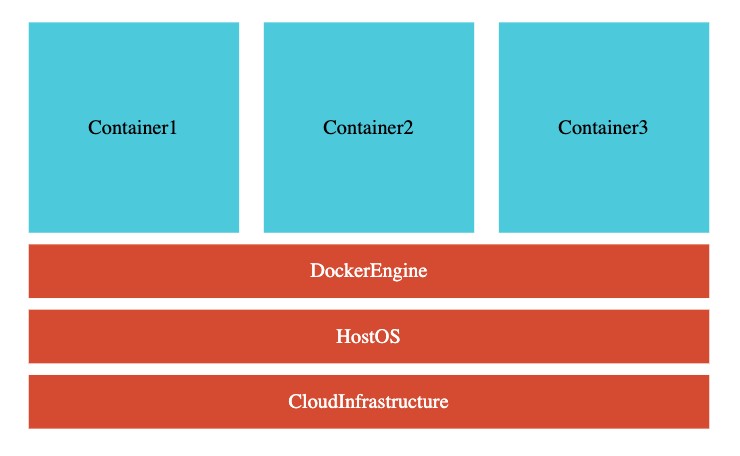
BUT **deployments become complex**!

How can we have **one way of deploying** Go, Java, Python or JavaScript ..

microservices?

Enter **containers**!

## Containers - Docker

Create **Docker images** for each microservice Docker image **has all needs of a microservice**:

Application Runtime (JDK or Python or NodeJS)

Application code and Dependencies

Runs **the same way** on any infrastructure:

Your local machine

Corporate data center

Cloud

Advantages

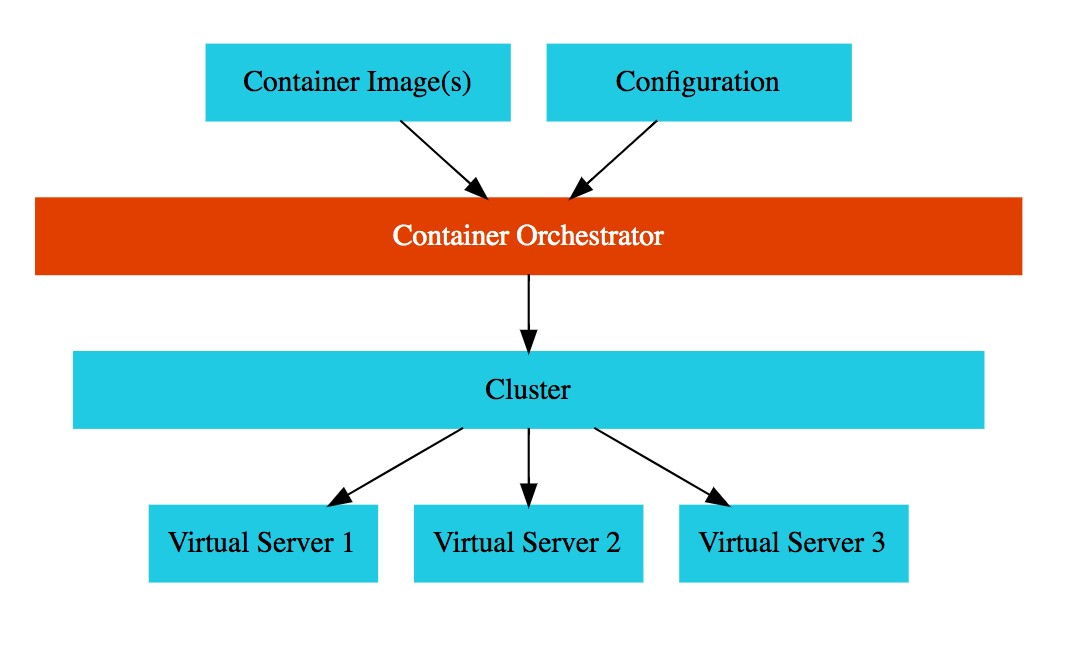
Docker containers are **light weight**

Compared to Virtual Machines as they do not have a Guest OS

Docker provides **isolation** for containers

Docker is **cloud neutral**

## Container Orchestration

 **Requirement** : I want 10 instances of Microservice A container, 15 instances of Microservice B container and ....

Typical Features:

**Auto Scaling** - Scale containers based on demand

**Service Discovery** - Help microservices find one another

**Load Balancer** - Distribute load among multiple instances of a microservice  **Self Healing** - Do health checks and replace failing instances

**Zero Downtime Deployments** - Release new versions without downtime

## Serverless

What do we think about when we develop an application?

Where to deploy? What kind of server? What OS?

How do we take care of scaling and availability of the application?

**What if you don't need to worry about servers and focus on your code?**

Enter **Serverless**

Remember: **Serverless does NOT mean "No Servers"**

**Serverless for me**:

You **don't worry** about infrastructure (ZERO visibility into infrastructure)

Flexible scaling and automated high availability

Most Important: **Pay for use**

Ideally ZERO REQUESTS => ZERO COST

**You focus on code** and the cloud managed service takes care of all that is needed to scale your code to serve millions of requests!

And you pay for requests and NOT servers!

**Serverless - My Perspective!**

Serverless - Important Features:

1: Zero worry about infrastructure, scaling and availability

2: Zero invocations => Zero Cost (Can you scale down to ZERO instances?)

3: Pay for invocations and NOT for instances (or nodes or servers)

Serverless **Level 1**: Features (1 + 2) Serverless **Level 2**: Features (1 + 2 + 3)

When I refer to Serverless, I'm referring to Level 2

HOWEVER cloud providers include managed services at Level 1 and Level 2:

**Level 1**: **Google App Engine** (Google Calls it "App Engine is a fully managed, serverless platform"), **AWS Fargate** (AWS calls it "serverless compute engine for containers") Scale down to ZERO instances when there is no load, **BUT** you pay for number (and type) of instances running!

**Level 2**: **Google Functions, AWS Lambda, Azure Functions** etc

You pay for invocations

## GCP Managed Services for Compute

**Service Details**

**Category**

**Compute** High-performance and general purpose VMs that scale globally IaaS

**Engine**

**Google** Orchestrate containerized microservices on Kubernetes CaaS

**Kubernetes** Needs advanced cluster configuration and monitoring **Engine**

**App Engine** Build highly scalable applications on a fully managed platform PaaS (CaaS,

using open and familiar languages and tools Serverless)

**Cloud** Build event driven applications using simple, single-purpose FaaS,

**Functions** functions Serverless

**Cloud Run** Develop and deploy highly scalable containerized applications. CaaS

Does NOT need a cluster! (Serverless)

# App Engine

## App Engine

**Simplest way** to deploy and scale your applications in GCP

Provides end-to-end application management Supports:

Go, Java, .NET, Node.js, PHP, Python, Ruby using pre-configured runtimes

Use custom run-time and write code in any language

Connect to variety of Google Cloud storage products (Cloud SQL etc) **No usage charges** - Pay for resources provisioned **Features**:

Automatic load balancing & Auto scaling

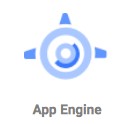
Managed platform updates & Application health monitoring

Application versioning

Traffic splitting

## Compute Engine vs App Engine

**Compute Engine**



IAAS

MORE Flexibility

MORE Responsibility

Choosing Image

Installing So ware

Choosing Hardware

Fine grained Access/Permissions (Certificates/Firewalls) Availability etc

**App Engine**

PaaS

Serverless

LESSER Responsibility

LOWER Flexibility

## App Engine environments

**Standard**: Applications run in language specific sandboxes

Complete isolation from OS/Disk/Other Apps

**V1**: Java, Python, PHP, Go (OLD Versions) ONLY for Python and PHP runtimes:

Restricted network Access

Only white-listed extensions and libraries are allowed

No Restrictions for Java and Go runtimes

**V2**: Java, Python, PHP, Node.js, Ruby, Go (NEWER Versions)

Full Network Access and No restrictions on Language Extensions

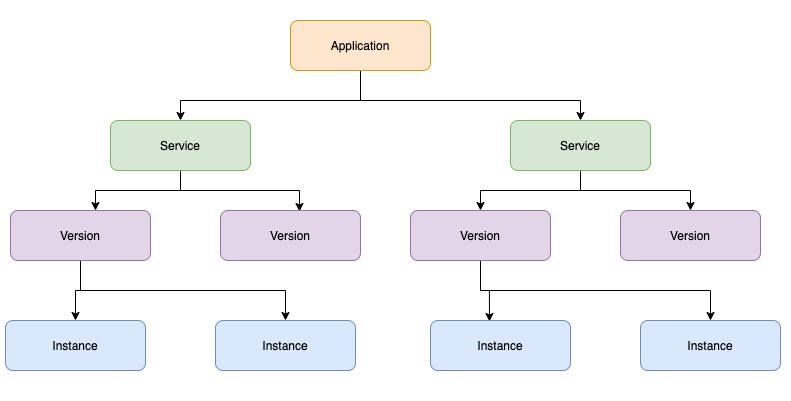
**Flexible** - Application instances run within Docker containers

Makes use of Compute Engine virtual machines

Support ANY runtime (with built-in support for Python, Java, Node.js, Go, Ruby, PHP, or .NET)

Provides access to background processes and local disks

## App Engine - Application Component Hierarchy

**Application**: One App per Project

**Service(s)**: Multiple Microservices or App components

You can have multiple services in a single application

Each **Service** can have different settings

Earlier called Modules

**Version(s)**: Each version associated with code and configuration

Each **Version** can run in one or more instances

Multiple versions can co-exist

Options to rollback and split traffic

## App Engine - Comparison

**Feature Standard Flexible**



**Pricing Factors** Instance hours vCPU, Memory & Persistent Disks



**Scaling** Manual, Basic, Automatic Manual, Automatic



**Scaling to zero** Yes No. Minimum one instance



**Instance startup** Seconds Minutes

**time**



**Rapid Scaling** Yes No



**Max. request** 1 to 10 minutes 60 minutes

**timeout**



**Local disk** Mostly(except for Python, PHP). Can write to Yes. Ephemeral. New Disk on

/tmp. startup.



**SSH for debugging** No Yes

## App Engine - Scaling Instances

**Automatic** - Automatically scale instances based on the load:

Recommended for Continuously Running Workloads

Auto scale based on:

**Target CPU Utilization** - Configure a CPU usage threshold.

**Target Throughput Utilization** - Configure a throughput threshold

**Max Concurrent Requests** - Configure max concurrent requests an instance can receive Configure **Max Instances** and **Min Instances**

**Basic** - Instances are created as and when requests are received:

Recommended for Adhoc Workloads

Instances are shutdown if ZERO requests

Tries to keep costs low

High latency is possible

NOT supported by App Engine Flexible Environment

Configure **Max Instances** and **Idle Timeout**

**Manual** - Configure specific number of instances to run: Adjust number of instances manually over time

## AppEngine Demo

Deploy an application to cloud using App Engine

### app.yaml Reference

|  |
| --- |
| runtime: python28 #The name of the runtime environment that is used by your app api\_version: 1 #RECOMMENDED - Specify here - gcloud app deploy -v [YOUR\_VERSION\_ID] instance\_class: F1 service: service-name  #env: flex  inbound\_services:  - warmup  env\_variables:  ENV\_VARIABLE: "value"  handlers: - url: / script: home.app  automatic\_scaling: target\_cpu\_utilization: 0.65  min\_instances: 5 max\_instances: 100 max\_concurrent\_requests: 50  #basic\_scaling:  #max\_instances: 11  #idle\_timeout: 10m  #manual\_scaling:  #instances: 5 |

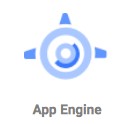
## AppEngine - Request Routing

You can use a **combination** of three approaches: Routing with **URLs**:

[***https://PROJECT\_ID.REGION\_ID.r.appspot.com***](https://project_id.region_id.r.appspot.com/) (default service called) [***https://SERVICE-dot-PROJECT\_ID.REGION\_ID.r.appspot.com***](https://service-dot-project_id.region_id.r.appspot.com/) (specific service) [***https://VERSION-dot-SERVICE-dot-PROJECT\_ID.REGION\_ID.r.appspot.com***](https://version-dot-service-dot-project_id.region_id.r.appspot.com/) (specific version of service) Replace -dot- with . if using custom domain Routing with a **dispatch file**: Configure dispatch.yaml with routes gcloud app deploy dispatch.yaml Routing with **Cloud Load Balancing**:

Configure routes on Load Balancing instance

## AppEngine - Deploying new versions without downtime

How do I go from V1 to V2 without downtime?

**Option 1**: I'm very confident - Deploy & shi all traffic at once:

Deploy and shi all traffic at once from v1 to v2: ***gcloud app deploy***

**Option 2**: I want to manage the migration from v1 to v2 **STEP 1**: Deploy v2 without shi ing traffic (--no-promote)  ***gcloud app deploy --no-promote* STEP 2**: Shi traffic to V2:

**Option 1** (All at once Migration): Migrate all at once to v2

***gcloud app services set-traffic s1 --splits V2=1***

**Option 2 (Gradual Migration)**: Gradually shi traffic to v2. Add --migrate option.

Gradual migration is not supported by App Engine Flexible Environment

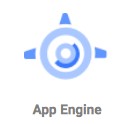
**Option 3 (Splitting)**: Control the pace of migration

***gcloud app services set-traffic s1 --splits=v2=.5,v1=.5***

Useful to perform A/B testing

Ensure that new instances are warmed up before they receive traffic (app.yaml inbound\_services > warmup)

**How do you split traffic between multiple versions?**

How do you decide which version receives which traffic?

**IP Splitting** - Based on request IP address

IP addresses can change causing accuracy issues! (I go from my house to a coffee shop)

If all requests originate from a corporate vpn with single IP, this can cause all requests to go to the same version

**Cookie Splitting** - Based on a cookie (**GOOGAPPUID**)

Cookies can be controlled from your application

Cookie splitting accurately assign users to versions

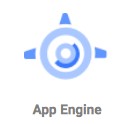
**Random** - Do it randomly How to do it?

Include --split-by option in gcloud app services set-traffic command

Value must be one of: cookie, ip, random

gcloud app services set-traffic s1 --splits=v2=.5,v1=.5 --splitby=cookie

## Playing with App Engine

***gcloud app browse/create/deploy/describe/open-console*** *gcloud app* ***create*** *--region=us-central gcloud app* ***deploy*** *app.yaml*

--image-url: Only for flexible environments. Deploy docker image.

*gcloud app deploy --image-url gcr.io/PROJECT-ID/hello-world-rest-api:0.0.1.RELEASE*

--promote --no-promote (Should new version receive traffic?)

--stop-previous-version --no-stop-previous-version (Should old version be stopped a er new version receives all traffic?)

--version (Assign a version. Otherwise, a version number is generated.)

*gcloud app* ***browse*** *--service="myService" --version="v1"* (open in a web browser) *gcloud app* ***open-console*** *--service="myService" --version="v1" gcloud app* ***open-console --logs***

Other Commands

*gcloud app* ***logs tail*** *gcloud app* ***regions list***

## Playing with App Engine Instances

***gcloud app instances delete/describe/list/scp/ssh*** *gcloud app instances* ***delete*** *i1 --service=s1 --version=v1 gcloud app instances* ***describe*** *--service=s1 --version=v1 i1 gcloud app instances* ***list*** *gcloud app instances* ***scp*** *--service=s1 --version=v1 --recurse local\_dir i1:remote\_dir*

(Copy files to/from App Engine Flexible instances)  *gcloud app instances* ***ssh*** *--service=s1 --version=v1 i1* (SSH into the VM of an App Engine Flexible instance)

## Playing with App Engine Services and Versions

**gcloud app services browse/delete/describe/list/set-traffic** *gcloud app services* ***list***

*gcloud app services* ***browse*** *myService --version="v1" gcloud app services* ***delete*** *service1 service2 gcloud app services* ***describe*** *service1*

*gcloud app services* ***set-traffic*** *APP1 --splits v1=0.9,v2=0.1*

--split\_by (ip, cookie, random)

**gcloud app versions browse/delete/describe/list/migrate/start/stop** *gcloud app versions* ***list***

--hide-no-traffic (Only show versions that are receiving traffic)

*gcloud app versions* ***browse****/****delete****/****describe*** *v1 --service="myService"*

*gcloud app versions* ***migrate*** *v2 --service="myService"* (migrate all traffic to new version) *gcloud app versions* ***start/stop*** *v1*

--service=my-service Only start v1 of service my-service

## App Engine - Cron Job

cron:

- description: "daily summary job" url: /tasks/summary schedule: every 24 hours

Allows to run **scheduled jobs** at pre-defined intervals **Use cases**:

Send a report by email every day Refresh cache data every 30 minutes

Configured using **cron.yaml**

Run this command - ***gcloud app deploy cron.yaml***

Performs a **HTTP GET** request to the configured URL on schedule

## Others Important App Engine yaml files

**dispatch.yaml** - override routing rules

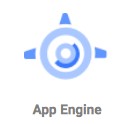
dispatch:

- url: "\*/mobile/\*" service: mobile-frontend - url: "\*/work/\*" service: static-backend

**queue.yaml** - manage task queues

|  |
| --- |
| queue:  - name: fooqueue rate: 1/s retry\_parameters:  task\_retry\_limit: 7 task\_age\_limit: 2d |

## App Engine - Remember

AppEngine is **Regional** (services deployed across zones)

You **CANNOT** change an Application's region

Good option for simple **microservices** (multiple services)

Use **Standard v2** when you are using supported languages

Use **Flexible** if you are building containerized apps

Be aware - **ATLEAST one container** is always running when using **Flexible**:

**Go for Standard** if you want to be able to scale down the number of instances to **zero** when there is NO load

Use a **combination of resident and dynamic** instances

Resident Instances: Run continuously

Dynamic Instances: Added based on load

Use all dynamic instances if you are cost sensitive

If you are not very cost sensitive, keep a set of resident instances running always

## App Engine - Scenarios

**Scenario Solution**



**I want to create two Google App** Not possible. You can only have one App Engine App per project. However **Engine Apps in the same project** you can have multiple services and multiple version for each service.



**I want to create two Google App** Yup. You can create multiple services under the same app. Each service **Engine Services inside the same** can have multiple versions as well. **App**



**I want to move my Google App** App Engine App is region specific. You CANNOT move it to different

**Engine App to a different region** region. Create a new project and create new app engine app in the new region.



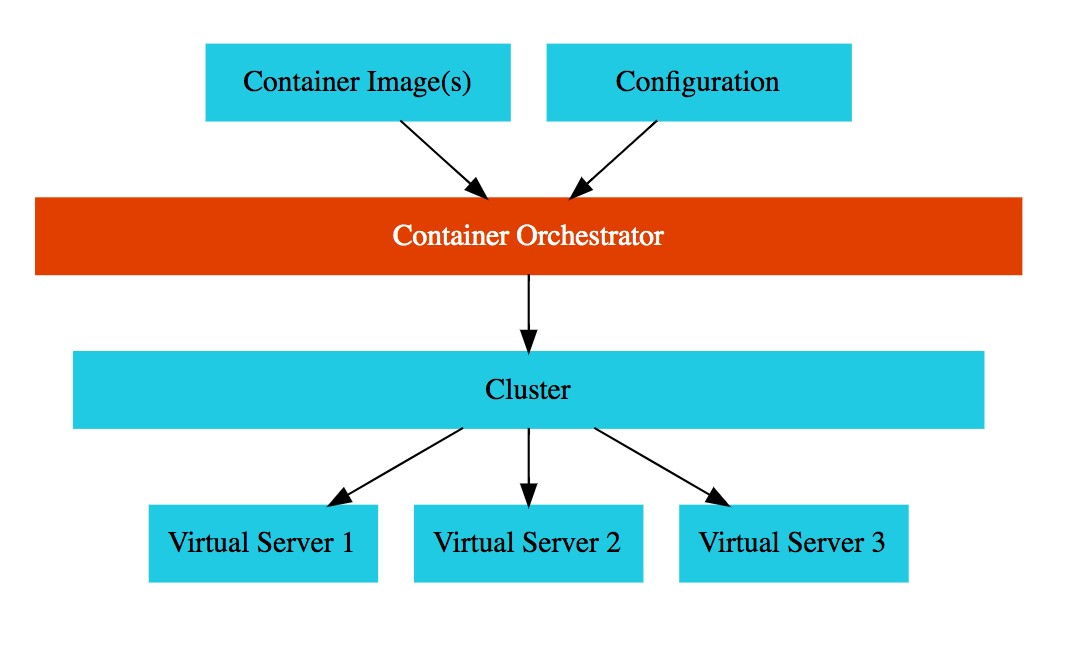
**Perform Canary deployments** Deploy v2 without shi ing traffic (gcloud app deploy --nopromote)

Shi some traffic to V2 (gcloud app services set-traffic s1 --splits v1=0.9,v2=0.1)



**Google Kubernetes Engine (GKE)**

## Kubernetes

 Most popular open source container orchestration solution

Provides Cluster Management

(including upgrades)

Each cluster can have different types of virtual machines

Provides all important container orchestration features:

**Auto Scaling**

**Service Discovery**

**Load Balancer**

**Self Healing**

**Zero Downtime Deployments**

## Google Kubernetes Engine (GKE)

**Managed** Kubernetes service

Minimize operations with **auto-repair** (repair failed nodes) and **auto-upgrade** (use latest version of K8S always) features

Provides **Pod and Cluster Autoscaling**

Enable **Cloud Logging** and **Cloud Monitoring** with simple configuration

Uses **Container-Optimized OS**, a hardened OS built by Google

Provides support for **Persistent disks** and **Local SSD**

## Kubernetes - A Microservice Journey - Getting Started

**Let's Have Some Fun**: Let's get on a journey with Kubernetes:

Let's create a cluster, deploy a microservice and play with it in **13 steps**!

**1:** Create a Kubernetes cluster with the default node pool *gcloud container* ***clusters create*** or use cloud console

**2:** Login to Cloud Shell

**3:** Connect to the Kubernetes Cluster

*gcloud container* ***clusters get-credentials*** *my-cluster --zone us-central1-a --project solid-course-258105*

## Kubernetes - A Microservice Journey - Deploy Microservice

**4:** Deploy Microservice to Kubernetes:

Create deployment & service using kubectl commands

*kubectl* ***create deployment*** *hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE kubectl* ***expose deployment*** *hello-world-rest-api --type=LoadBalancer --port=8080*

**5:** Increase number of instances of your microservice: *kubectl* ***scale deployment*** *hello-world-rest-api --replicas=2*

**6:** Increase number of nodes in your Kubernetes cluster:

*gcloud container clusters resize* ***my-cluster --node-pool my-node-pool --num-nodes 5*** You are NOT happy about manually increasing number of instances and nodes!

**Kubernetes - A Microservice Journey - Auto Scaling and ..**

**7:** Setup auto scaling for your microservice:

*kubectl* ***autoscale deployment*** *hello-world-rest-api --max=10 --cpu-percent=70*

Also called horizontal pod autoscaling - HPA - *kubectl get hpa*

**8:** Setup auto scaling for your Kubernetes Cluster

*gcloud* ***container clusters update*** *cluster-name --enable-autoscaling --min-nodes=1 -max-nodes=10*

**9:** Add some application configuration for your microservice

Config Map - *kubectl* ***create configmap*** *todo-web-application-config --fromliteral=RDS\_DB\_NAME=todos*

**10:** Add password configuration for your microservice

Kubernetes Secrets - *kubectl* ***create secret*** *generic todo-web-application-secrets-1 -from-literal=RDS\_PASSWORD=dummytodos*

## Kubernetes Deployment YAML - Deployment

|  |
| --- |
| apiVersion: apps/v1 kind: Deployment metadata: labels: app: hello-world-rest-api name: hello-world-rest-api namespace: default spec: replicas: 3 selector: matchLabels: app: hello-world-rest-api template: metadata: labels: app: hello-world-rest-api spec: containers:  - image: in28min/hello-world-rest-api:0.0.3.RELEASE  name: hello-world-rest-api |

## Kubernetes Deployment YAML - Service

apiVersion: v1 kind: Service metadata: labels: app: hello-world-rest-api name: hello-world-rest-api namespace: default spec: ports: - port: 8080 protocol: TCP targetPort: 8080 selector: app: hello-world-rest-api sessionAffinity: None type: LoadBalancer

**Kubernetes - A Microservice Journey - The End!**

 **11:** Deploy a new microservice which needs nodes with a GPU attached

Attach a new node pool with GPU instances to your cluster gcloud **container node-pools create** POOL\_NAME --cluster CLUSTER\_NAME gcloud container node-pools list --cluster CLUSTER\_NAME

Deploy the new microservice to the new pool by setting up nodeSelector in the

deployment.yaml

nodeSelector: cloud.google.com/gke-nodepool: POOL\_NAME

**12:** Delete the Microservices

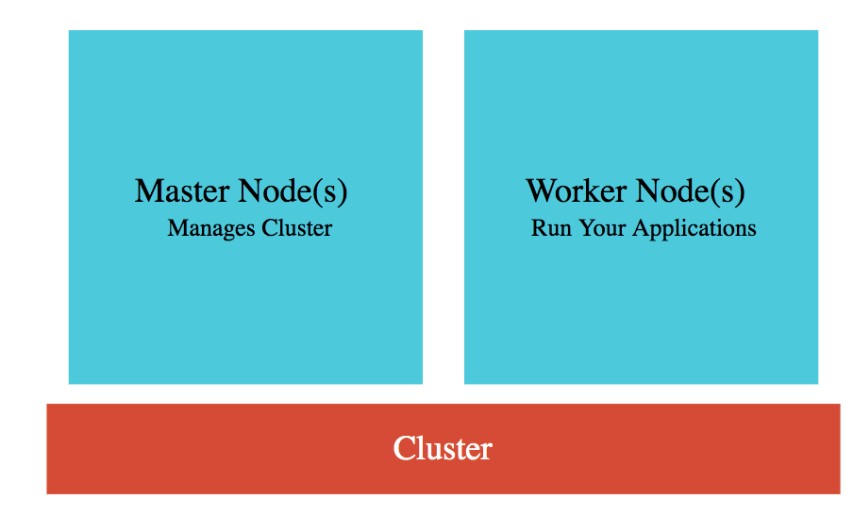
Delete service - *kubectl* ***delete service***

Delete deployment - *kubectl* ***delete deployment***

**13:** Delete the Cluster

*gcloud container* ***clusters delete***

## Google Kubernetes Engine (GKE) Cluster

**Cluster** : Group of Compute Engine instances:

**Master Node(s)** - Manages the cluster

**Worker Node(s)** - Run your workloads (pods)

**Master Node** (Control plane) components:

**API Server** - Handles all communication for a K8S

cluster (from nodes and outside) **Scheduler** - Decides placement of pods

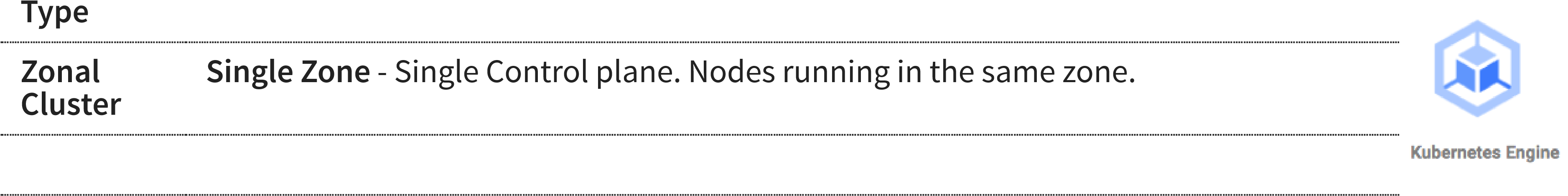
**Control Manager** - Manages deployments & replicasets  **etcd** - Distributed database storing the cluster state

**Worker Node** components:

Runs your pods

**Kubelet** - Manages communication with master node(s)

## GKE Cluster Types

**Description**

**Multi-zonal** - Single Control plane but nodes running in multiple zones

**Regional** Replicas of the control plane runs in multiple zones of a given region. Nodes also run **cluster** in same zones where control plane runs.



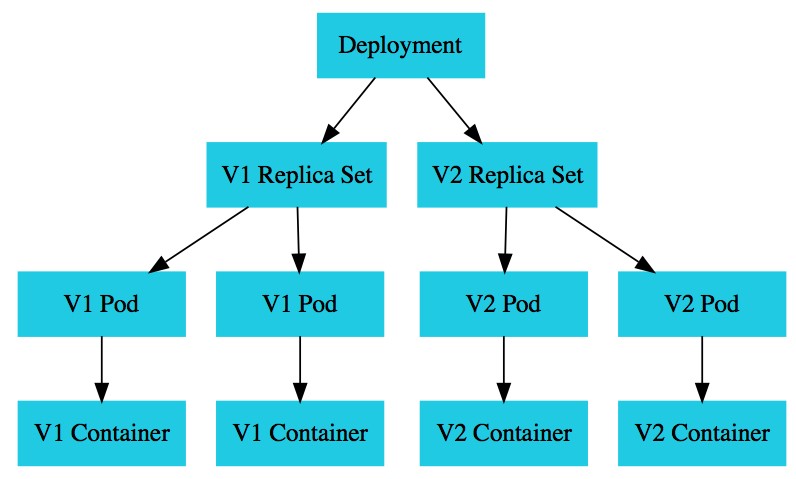
**Private** VPC-native cluster. Nodes only have internal IP addresses. **cluster**



**Alpha** Clusters with alpha APIs - early feature API's. Used to test new K8S features. **cluster**



## Kubernetes - Pods

Smallest deployable unit in Kubernetes

A Pod contains **one or more containers** Each Pod is assigned an ephemeral **IP address** All containers in a pod share:

Network

Storage

IP Address

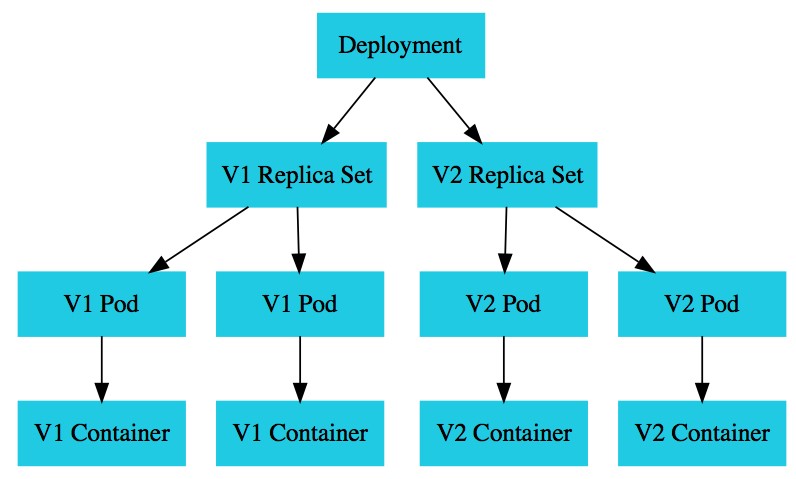
Ports and

Volumes (Shared persistent disks)

POD statuses : Running /Pending /Succeeded

/Failed /Unknown

## Kubernetes - Deployment vs Replica Set

A **deployment** is created for each microservice:

kubectl create deployment m1 --image=m1:v1

Deployment represents a microservice (with all its releases) Deployment manages new releases ensuring zero downtime  **Replica set** ensures that a specific number of pods are running for a specific microservice version

kubectl scale deployment m2 --replicas=2

Even if one of the pods is killed, replica set will launch a new one

Deploy V2 of microservice - Creates a new replica set kubectl set image deployment m1 m1=m1:v2

V2 Replica Set is created

Deployment updates V1 Replica Set and V2 Replica Set based on the release strategies

## Kubernetes - Service

Each Pod has its **own IP address**: How do you ensure that external users are not impacted when:

A pod fails and is replaced by replica set

A new release happens and all existing pods of old release are replaced by ones of new release

Create **Service**

*kubectl expose deployment name --type=LoadBalancer --port=80*

Expose PODs to outside world using a stable IP Address

Ensures that the external world does not get impacted as pods go down and come up

Three Types:

**ClusterIP**: Exposes Service on a cluster-internal IP

Use case: You want your microservice only to be available inside the cluster (Intra cluster communication)

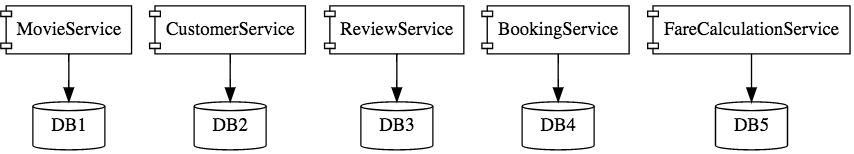
**LoadBalancer**: Exposes Service externally using a cloud provider's load balancer

Use case: You want to create individual Load Balancer's for each microservice

**NodePort**: Exposes Service on each Node's IP at a static port (the NodePort)

Use case: You DO not want to create an external Load Balancer for each microservice (You can create one Ingress component to load balance multiple microservices)

## Container Registry - Image Repository



You've created docker images for your microservices: **Where do you store them?**

**Container Registry** - fully-managed container registry provided by GCP (Alternative) Docker Hub

Can be integrated to CI/CD tools to publish images to registry

You can secure your container images. Analyze for vulnerabilities and enforce deployment policies.

Naming: **HostName/ProjectID/Image:Tag** - **gcr.io/projectname/helloworld:1**

## GKE - Remember

**Replicate master nodes** across multiple zones for high availability (REMEMBER) Some **CPU** on the nodes is **reserved by Control Plane**:

1st core - 6%, 2nd core - 1%, 3rd/4th - 0.5, Rest - 0.25

Creating Docker Image for your microservices(Dockerfile):

Build Image: *docker build -t in28min/hello-world-rest-api:0.0.1.RELEASE .*

Test it Locally: *docker run -d -p 8080:8080 in28min/hello-world-rest-api:0.0.1.RELEASE*

Push it to Container Repository: *docker push in28min/hello-world-rest-api:0.0.1.RELEASE*

Kubernetes supports **Stateful** deployments like Kafka, Redis, ZooKeeper:

**StatefulSet** - Set of Pods with unique, persistent identities and stable hostnames How do we run services on nodes for **log collection or monitoring**?

**DaemonSet** - One pod on every node! (for background services)

(Enabled by default) Integrates with Cloud Monitoring and Cloud Logging

Cloud Logging **System** and **Application Logs** can be exported to **Big Query** or **Pub/Sub**

## GKE - Cluster Management - Command Line

**Description Command**



**Create Cluster** gcloud container clusters **create** my-cluster --zone us-central1-a --node-locations uscentral1-c,us-central1-b



**Resize Cluster** gcloud container clusters **resize** my-cluster --node-pool my-node-pool --num-nodes 10



**Autoscale** gcloud **container clusters update** cluster-name --enable-autoscaling --min-nodes=1 --max-

**Cluster** nodes=10



**Delete Cluster** gcloud container clusters **delete** my-cluster



**Adding Node** gcloud container **node-pools create** new-node-pool-name --cluster my-cluster

**Pool**



**List Images** gcloud container images list



## GKE - Workload Management - Command Line

**Description Command**



**List Pods/Service/Replica** kubectl **get pods/services/replicasets Sets**



**Create Deployment** kubectl **apply -f** deployment.yaml or kubectl **create deployment**



**Create Service** kubectl **expose deployment** hello-world-rest-api --type=LoadBalancer -port=8080



**Scale Deployment** kubectl **scale deployment** hello-world --replicas 5



**Autoscale Deployment** kubectl **autoscale deployment** --max --min --cpu-percent



**Delete Deployment** kubectl **delete deployment** hello-world



**Update Deployment** kubectl **apply -f** deployment.yaml



**Rollback Deployment** kubectl **rollout undo deployment** hello-world --to-revision=1



## Google Kubernetes Engine - Scenarios - 1

**Scenario Solution**



**You want to keep your costs low and optimize** Consider Preemptible VMs, Appropriate region, Committed**your GKE implementation** use discounts.

E2 machine types are cheaper than N1.

Choose right environment to fit your workload type (Use multiple node pools if needed).



**You want an efficient, completely auto scaling** Configure Horizontal Pod Autoscaler for deployments and

**GKE solution** Cluster Autoscaler for node pools



**You want to execute untrusted third-party** Create a new node pool with GKE Sandbox. Deploy **code in Kubernetes Cluster** untrused code to Sandbox node pool.



## Google Kubernetes Engine - Scenarios - 2

**Scenario Solution**



**You want enable ONLY internal communication between your** Create Service of type ClusterIP **microservice deployments in a Kubernetes Cluster**



**My pod stays pending** Most probably Pod cannot be scheduled

onto a node(insufficient resources)

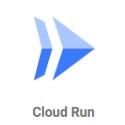


**My pod stays waiting** Most probably failure to pull the image



# Google Cloud Functions

## Cloud Functions

Imagine you want to **execute some code when an event happens**?

A file is uploaded in Cloud Storage

An error log is written to Cloud Logging

A message arrives to Cloud Pub/Sub

Enter **Cloud Functions**

**Run code in response to events**

Write your business logic in Node.js, Python, Go, Java, .NET, and Ruby

**Don't worry** about servers or scaling or availability (only worry about your code)

**Pay only for what you use**

Number of invocations

Compute Time of the invocations

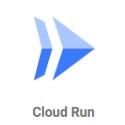
Amount of memory and CPU provisioned

**Time Bound** - Default 1 min and MAX 9 minutes(540 seconds)

**Each execution runs in a separate instance**

No direct sharing between invocations

## Cloud Functions - Concepts

**Event** : Upload object to cloud storage

**Trigger**: Respond to event with a Function call **Trigger** - Which function to trigger when an event happens?

**Functions** - Take event data and perform action?

Events are **triggered from**

Cloud Storage

Cloud Pub/Sub

HTTP POST/GET/DELETE/PUT/OPTIONS

Firebase

Cloud Firestore

Stack driver logging

## Example Cloud Function - HTTP - Node.js

const escapeHtml = require('escape-html');

/\*\*

* HTTP Cloud Function.

\*

* @param {Object} req Cloud Function request context.
* More info: https://expressjs.com/en/api.html#req
* @param {Object} res Cloud Function response context.
* More info: https://expressjs.com/en/api.html#res

\*/

exports.helloHttp = (req, res) => {

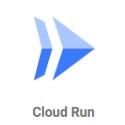
res.send(`Hello ${escapeHtml(req.query.name || req.body.name || 'World')}!`);

};

## Example Cloud Function - Pub/Sub - Node.js

|  |
| --- |
| /\*\*   * Background Cloud Function to be triggered by Pub/Sub. \* This function is exported by index.js, and executed when \* the trigger topic receives a message.   \*   * @param {object} message The Pub/Sub message. \* @param {object} context The event metadata.   \*/  exports.helloPubSub = (message, context) => { const name = message.data  ? Buffer.from(message.data, 'base64').toString()  : 'World';  console.log(`Hello, ${name}!`); }; |

## Cloud Functions - Remember

 No Server Management: You dont need to worry about scaling or availability of your function

Cloud Functions automatically spin up and back down in response to events

They scale horizontally!

Cloud Functions are recommended for responding to events:

Cloud Functions are NOT ideal for long running processes **Time Bound** - Default 1 min and MAX 9 minutes(540 seconds)

## Cloud Run & Cloud Run for Anthos

**Cloud Run** - "Container to Production in Seconds"

Built on top of an open standard - **Knative**

**Fully managed** serverless platform for containerized applications

ZERO infrastructure management

Pay-per-use (For used CPU, Memory, Requests and Networking)

Fully integrated **end-to-end developer experience**:

**No limitations** in languages, binaries and dependencies

Easily portable because of **container** based architecture

Cloud Code, Cloud Build, Cloud Monitoring & Cloud Logging Integrations

**Anthos** - Run Kubernetes clusters anywhere

Cloud, Multi Cloud and On-Premise

**Cloud Run for Anthos**: Deploy your workloads to Anthos clusters running on-premises or on Google Cloud

Leverage your existing Kubernetes investment to quickly run serverless workloads

## Cloud Run - From the Command Line

**Description Command**



**Deploy a new container** *gcloud* ***run deploy*** *SERVICE\_NAME --image IMAGE\_URL --revision-suffix v1*

First deployment creates a service and first revision

Next deployments for the same service create new revisions



**List available revisions** gcloud **run revisions** list

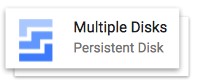
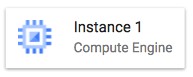


**Adjust traffic assignments** gcloud **run services update-traffic** myservice --to-revisions=v2=10,v1=90



# Encryption

## Data States



**Data at rest**: Stored on a device or a backup

Examples : data on a hard disk, in a database, backups and archives

**Data in motion**: Being transferred across a network

Also called **Data in transit Examples** :

Data copied from on-premise to cloud storage

An application talking to a database

**Two Types**:

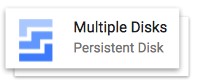
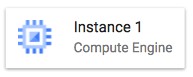
In and out of cloud (from internet)

Within cloud

**Data in use**: Active data processed in a non-persistent state

Example: Data in your RAM

## Encryption



If you store data as is, what would happen if an **unauthorized entity gets access** to it?

Imagine losing an unencrypted hard disk

**First law of security** : Defense in Depth

Typically, enterprises encrypt all data

Data on your hard disks

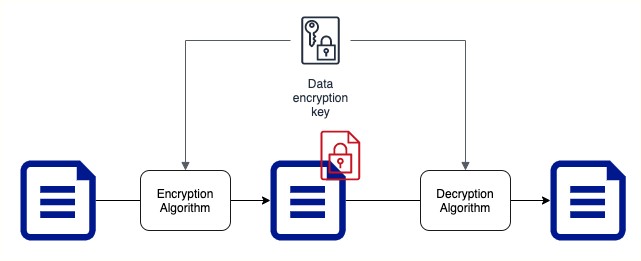
Data in your databases

Data on your file servers

Is it sufficient if you encrypt data at rest?

**No**. **Encrypt data in transit** - between application to database as well.

## Symmetric Key Encryption

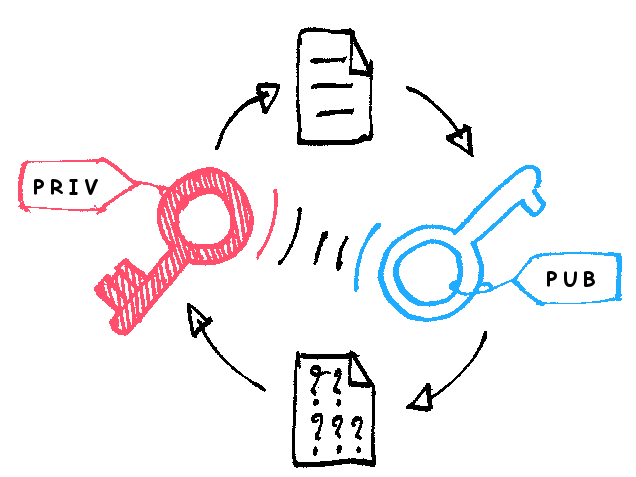


Symmetric encryption algorithms use the **same key for encryption and decryption**

Key Factor 1: Choose the **right encryption algorithm** Key Factor 2: How do we **secure the encryption key**?

Key Factor 3: How do we **share the encryption key**?

## Asymmetric Key Encryption

**Two Keys** : Public Key and Private Key

Also called **Public Key Cyptography**

Encrypt data with Public Key and decrypt with Private Key Share Public Key with everybody and keep the Private Key with you(YEAH, ITS PRIVATE!) No crazy questions:

Will somebody not figure out private key using the public key?

How do you create Asymmetric Keys?

[***https://commons.wikimedia.org/wiki/File:Asymmetric\_encry***](https://commons.wikimedia.org/wiki/File:Asymmetric_encryption_(colored).png)

## Cloud KMS

 Create and manage **cryptographic keys** (symmetric and asymmetric)

**Control their use** in your applications and GCP Services Provides an API to encrypt, decrypt, or sign data

Use existing cryptographic keys created on premises **Integrates with almost all GCP services** that need data encryption:

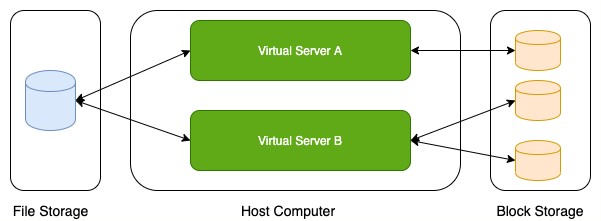
Google-managed key: No configuration required

Customer-managed key: Use key from KMS

Customer-supplied key: Provide your own key

# Storage

## Storage Types - Block Storage and File Storage



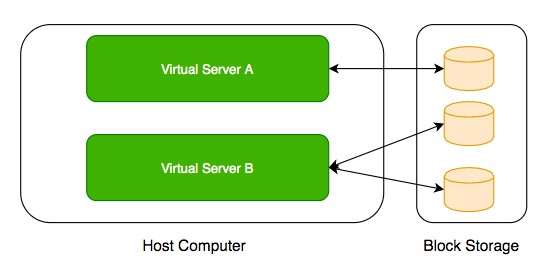
What is the type of storage of your hard disk?

**Block Storage**

You've created a file share to share a set of files with your colleagues in a enterprise. What type of storage are you using?

**File Storage**

## Block Storage

 Use case: Harddisks attached to your computers

Typically, ONE Block Storage device can be connected to ONE virtual server

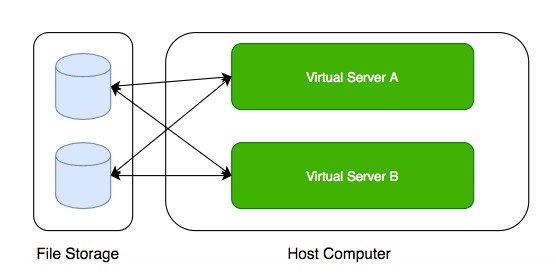
(EXCEPTIONS) You can attach read only block devices with multiple virtual servers and certain cloud providers are exploring multi-writer disks as well!

HOWEVER, you can connect multiple different block storage devices to one virtual server Used as:

**Direct-attached storage (DAS)** - Similar to a hard disk **Storage Area Network (SAN)** - High-speed network connecting a pool of storage devices

Used by Databases - Oracle and Microso SQL Server

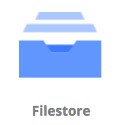
## File Storage

 Media workflows need huge shared storage for supporting processes like video editing

Enterprise users need a quick way to share files in a secure and organized way

These file shares are shared by several virtual servers

## GCP - Block Storage and File Storage



**Block Storage**:

**Persistent Disks**: Network Block Storage

Zonal: Data replicated in one zone Regional: Data replicated in multiple zone

**Local SSDs**: Local Block Storage **File Storage**:

**Filestore**: High performance file storage

## GCP - Block Storage

Two popular types of block storage can be attached to VM instances:

**Local SSDs Persistent Disks**

**Local SSDs** are physically attached to the host of the VM instance

Temporary data

Lifecycle tied to VM instance

**Persistent Disks** are network storage

More durable

Lifecycle NOT tied to VM instance

## Local SSDs

**Physically attached** to the host of VM instance:

Provide very high (IOPS) and very low latency

(BUT) **Ephemeral storage** - Temporary data (Data persists only until instance is running) **Enable live migration** for data to survive maintenance events

Data automatically encrypted HOWEVER, you CANNOT configure encryption keys!

Lifecycle tied to VM instance

ONLY some machine types support Local SSDs

Supports SCSI and NVMe interfaces

Remember:

Choose NVMe-enabled and multi-queue SCSI images for best performance

Larger Local SSDs (more storage), More vCPUs (attached to VM) => Even Better Performance

## Local SSDs - Advantages and Disadvantages

**Advantages**

Very Fast I/O (~ 10-100X compared to PDs)

Higher throughput and lower latency

Ideal for use cases needing high IOPs while storing **temporary information**

Examples: Caches, temporary data, scratch files etc

**Disadvantages**

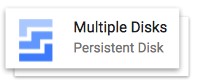
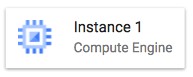
**Ephemeral storage**

Lower durability, lower availability, lower flexibility compared to PDs

You **CANNOT detach and attach** it to another VM instance

## Persistent Disks (PD)

**Network block storage** attached to your VM instance **Provisioned capacity** Very Flexible:



**Increase size when you need it** - when attached to VM instance

Performance scales with size

For higher performance, resize or add more PDs

**Independent lifecycle** from VM instance

Attach/Detach from one VM instance to another

Options: Regional and Zonal

Zonal PDs replicated in single zone. Regional PDs replicated in 2 zones in same Region.

Typically Regional PDs are 2X the cost of Zonal PDs

**Use case** : Run your custom database

## Persistent Disks vs Local SSDs

**Feature Persistent Disks Local SSDs**



**Attachment to VM instance** As a network drive Physically attached



**Lifecycle** Separate from VM instance Tied with VM instance



**I/O Speed** Lower (network latency) 10-100X of PDs



**Snapshots** Supported Not Supported



**Use case** Permanent storage Ephemeral storage



## Persistent Disks - Standard vs Balanced vs SSD

**Feature Standard Balanced SSD**



**Underlying Storage** Hard Disk Drive Solid State Drive Solid State

Drive



**Referred to as** pd-standard pd-balanced pd-ssd



**Performance - Sequential IOPS (Big** Good Good Very Good

**Data/Batch)**



**Performance - Random IOPS** Bad Good Very Good

**(Transactional Apps)**



**Cost** Cheapest In Between Expensive

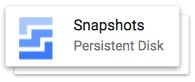
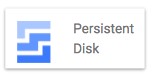


**Use cases** Big Data (cost Balance between cost and High efficient) performance Performance



## Persistent Disks - Snapshots

Take **point-in-time snapshots** of your Persistent Disks You can also schedule snapshots (configure a schedule):



You can also auto-delete snapshots a er X days

Snapshots can be Multi-regional and Regional

You can share snapshots across projects

You can create new disks and instances from snapshots Snapshots are **incremental**:

Deleting a snapshot **only deletes data which is NOT needed** by other snapshots Keep similar data together on a Persistent Disk:

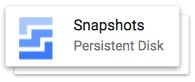
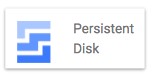
Separate your operating system, volatile data and permanent data

Attach multiple disks if needed

This helps to better organize your snapshots and images

## Persistent Disks - Snapshots - Recommendations

**Avoid** taking snapshots more o en than once an hour



Disk volume is available for use **but Snapshots reduce performance**

(RECOMMENDED) Schedule snapshots during off-peak hours

Creating snapshots from disk is faster than creating from images:

But creating disks from image is faster than creating from snapshots (RECOMMENDED) If you are repeatedly creating disks from a snapshot:

Create an image from snapshot and use the image to create disks

Snapshots are **incremental**:

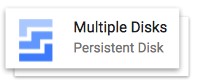
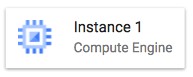
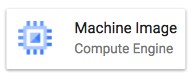
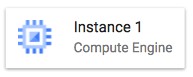
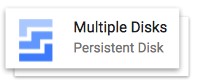
BUT you don't lose data by deleting older snapshots

Deleting a snapshot **only deletes data which is NOT needed** by other snapshots

(RECOMMENDED) Do not hesitate to delete unnecessary snapshots

## Playing with Machine Images

(Remember) **Machine Image** is different from Image **Multiple disks can be attached** with a VM:



One Boot Disk (Your OS runs from Boot Disk) Multiple Data Disks

An image is created from the boot Persistent Disk

HOWEVER, a Machine Image is created from a VM instance:

Machine Image **contains everything you need** to create a VM instance:

Configuration

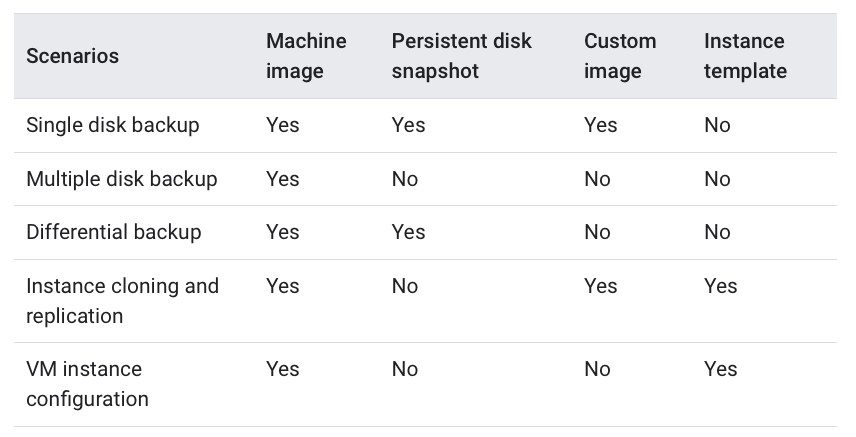
Metadata

Permissions

Data from one or more disks

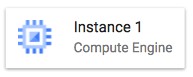
**Recommended for** disk backups, instance cloning and replication

## Let's Compare

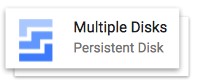


[***https://cloud.google.com/compute/docs/machine-images***](https://cloud.google.com/compute/docs/machine-images)

## Playing with Disks - Command Line

***gcloud compute disks list/create/delete/resize/snapshot*** *gcloud compute disks* ***create*** *my-disk-1 --zone=us-east1-a* What should be the size and type?

*--size=SIZE* (1GB or 2TB)



*--type=TYPE* (default - pd-standard) (gcloud compute disk-types list) What should be on the disk?

*--image --image-family --source-disk --source-snapshot*

How should data on disk be encrypted?

*--kms-key --kms-project*

*gcloud compute disks* ***resize*** *example-disk-1 --size=6TB*

Only increasing disk size is supported

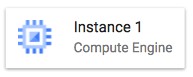
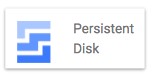
*gcloud compute disks* ***snapshot*** *test --zone=us-central1-a --snapshotnames=snapshot-test*

You can also play with the snapshots which are created:

*gcloud compute snapshots list/describe/delete*

## Playing with Images - Command Line

***gcloud compute images***



Actions: ***create/delete/deprecate/describe/export/import/list/update***

Creating Images

*gcloud compute images* ***create*** *my-image*

From a Disk - *--source-disk=my-disk --source-disk-zone=us-east1-a*

From a Snapshot - *--source-snapshot=source-snapshot*

From another image - *--source-image=source-image --source-image-project=source-image-project*

From latest non deprecated image from a family - *--source-image-family=source-image-family --source-imageproject=source-image-project*

Deprecate Image

*gcloud compute images* ***deprecate*** *IMAGE --state=DEPRECATED*

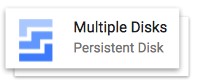
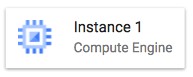
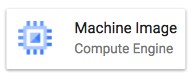
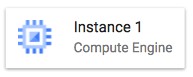
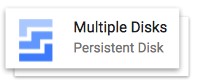
Exports virtual disk images  *gcloud compute images* ***export*** *--image=my-image --destination-uri=gs://my-bucket/my-image.vmdk -export-format=vmdk --project=my-project*

Other Examples:

*gcloud compute images* ***delete*** *my-image1 my-image2 gcloud compute images* ***list*** *--format="value(NAME)"*

## Playing with Machine Images - Command Line

(Remember) gcloud commands for machine images are IN BETA Commands:



Create Machine Image:

***gcloud beta compute machine-images create MACHINE\_IMAGE\_NAME --source-instance SOURCE\_INSTANCE\_NAME***

Create an Instance from the Machine Image:

***gcloud beta compute instances create VM\_NAME --zone ZONE --source-machine-image SOURCE\_MACHINE\_IMAGE\_NAME***

## Storage - Scenarios - Persistent Disks

**Scenario Solution**



**You want to improve performance of Persistent Disks (PD)** Increase size of PD or Add more PDs. Increase vCPUs in your VM.



**You want to increase durability of Persistent Disks (PD)** Go for Regional PDs (2X cost but replicated in 2 zones)



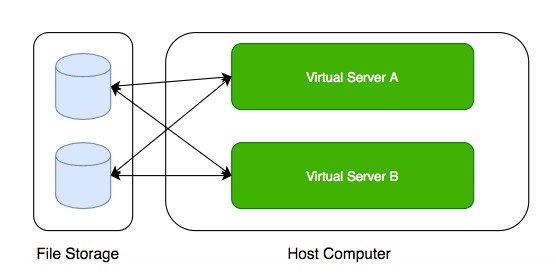
**You want to take hourly backup of Persistent Disks (PD)** Schedule hourly snapshots! **for disaster recovery**



**You want to delete old snapshots created by scheduled** Configure it as part of your snapshot scheduling! **snapshots**



## Cloud Filestore

**Shared cloud file storage**:

Supports NFSv3 protocol

Provisioned Capacity

Suitable for **high performance** workloads:

Up to 320 TB with throughput of 16 GB/s and 480K IOPS

Supports HDD (general purpose) and SSD

(performance-critical workloads)

**Use cases** : file share, media workflows and content

management

## Review - Global, Regional and Zonal Resources

**Global**

Images

Snapshots

Instance templates (Unless you use zonal resources in your templates)

**Regional**

Regional managed instance groups

Regional persistent disks

**Zonal**

Zonal managed instance groups

Instances

Persistent disks

You can attach a disk only to instances in the same zone as the disk

## Storage - Scenarios

**Scenario Solution**



**You want Very High IOPS but your data can be lost without a problem** Local SSDs



**You want to create a high perfomance file sharing system in GCP which can be** Filestore **attached with multiple VMs**



**You want to backup your VM configuration along with all its attached Persistent Disks** Create a Machine

Image



**You want to make it easy to launch VMs with hardened OS and customized so ware** Create a Custom Image



# Object Storage - Cloud Storage

## Cloud Storage

**Most popular, very flexible & inexpensive** storage service

Serverless: Autoscaling and infinite scale

Store large objects using a **key-value** approach:

Treats entire object as a unit (Partial updates not allowed)

Recommended when you operate on entire object most of the time Access Control at Object level

Also called **Object Storage**

Provides REST API to access and modify objects

Also provides CLI (gsutil) & Client Libraries (C++, C#, Java, Node.js, PHP, Python & Ruby)

**Store all file types** - text, binary, backup & archives:

Media files and archives, Application packages and logs

Backups of your databases or storage devices

Staging data during on-premise to cloud database migration

## Cloud Storage - Objects and Buckets

Objects are stored in buckets Bucket names are **globally unique**

Bucket names are used as part of object URLs => Can contain ONLY lower case letters, numbers, hyphens, underscores and periods.

3-63 characters max. Can't start with **goog prefix** or should not contain **google (even misspelled)**  Unlimited objects in a bucket

Each bucket is associated with a project

Each object is identified by a **unique key**

**Key is unique** in a bucket

Max object size is **5 TB**

BUT you can store unlimited number of such objects

## Cloud Storage - Storage Classes - Introduction

**Different kinds of data** can be stored in Cloud Storage

Media files and archives

Application packages and logs

Backups of your databases or storage devices Long term archives

Huge variations in **access patterns**

Can I pay a cheaper price for objects I access less frequently?

**Storage classes** help to optimize your costs based on your access needs

Designed for durability of **99.999999999%(11 9’s)**

## Cloud Storage - Storage Classes - Comparison

**Storage Name Minimum Typical Monthly availability Use case**

**Class Storage**

**duration**



**Standard** STANDARD None > 99.99% in multi region and dual Frequently used

region, 99.99% in regions data/Short period of time



**Nearline** NEARLINE 30 days 99.95% in multi region and dual Read or modify **once a storage** region, 99.9% in regions **month** on average



**Coldline** COLDLINE 90 days 99.95% in multi region and dual Read or modify **at most storage** region, 99.9% in regions **once a quarter**



**Archive** ARCHIVE 365 days 99.95% in multi region and dual **Less than once a year**

**storage** region, 99.9% in regions



## Features across Storage Classes

High durability (99.999999999% annual durability)

**Low** latency (first byte typically in tens of milliseconds)

**Unlimited** storage

Autoscaling (No configuration needed)

**NO minimum** object size

Same APIs across storage classes

**Committed SLA** is 99.95% for multi region and 99.9% for single region for Standard, Nearline and Coldline storage classes

No committed SLA for Archive storage

## Cloud Storage - Uploading and Downloading Objects

**Option Recommended for Scenarios**



**Simple Upload** Small files (that can be re uploaded in case of failures) + NO object metadata



**Multipart upload** Small files (that can be re uploaded in case of failures) + object metadata



**Resumable upload** Larger files. RECOMMENDED for most usecases

(even for small files - costs one additional HTTP request)



**Streaming transfers** Upload an object of unknown size



**Parallel composite uploads** File divided up to 32 chunks and uploaded in parallel.

Significantly faster if network and disk speed are not limiting factors.



**Simple download** Downloading objects to a destination



**Streaming download** Downloading data to a process



**Sliced object download** Slice and download large objects



## Object Versioning

Prevents **accidental deletion** & provides history

Enabled at bucket level

Can be turned on/off at any time **Live version** is the latest version

If you delete live object, it becomes noncurrent object version

If you delete noncurrent object version, it is deleted

Older versions are uniquely identified by (object key + a generation number)

Reduce costs by deleting older (noncurrent) versions!

## Object Lifecycle Management

Files are frequently accessed when they are created

Generally **usage reduces with time**

How do you save costs by **moving files automatically between storage classes**?

Solution: Object Lifecycle Management

Identify objects using conditions based on:

Age, CreatedBefore, IsLive, MatchesStorageClass, NumberOfNewerVersions etc

Set multiple conditions: all conditions must be satisfied for action to happen Two kinds of actions:

**SetStorageClass** actions (change from one storage class to another) **Deletion** actions (delete objects) Allowed Transitions:

(Standard or Multi-Regional or Regional) to (Nearline or Coldline or Archive)

Nearline to (Coldline or Archive)

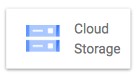
Coldline to Archive

## Object Lifecycle Management - Example Rule

|  |
| --- |
| {  "lifecycle": {  "rule": [  {  "action": {"type": "Delete"},  "condition": {  "age": 30,  "isLive": true  }  }, {  "action": {  "type": "SetStorageClass", "storageClass": "NEARLINE"  },  "condition": {  "age": 365,  "matchesStorageClass": ["STANDARD"]  }  }  ]  }  } |

## Cloud Storage - Encryption

Cloud Storage always encrypts data on the server side!



Configure **Server-side** encryption: Encryption performed by Cloud

Storage

**Google-managed encryption key** - Default (No configuration required)

**Customer-managed** encryption keys - Created using **Cloud Key Management Service (KMS)**. Managed by customer in KMS.

Cloud Storage Service Account should have access to keys in KMS for encrypting and decrypting using the **Customer-Managed** encryption key

(OPTIONAL) **Client-side** encryption - Encryption performed by customer before upload

GCP does NOT know about the keys used

## Cloud Storage - Scenarios

**Scenario Description**



**How do you speed up large uploads (example: 100 GB) to** Use **Parallel composite uploads** (File is

**Cloud Storage?** broken in to small chunks and uploaded)



**You want to permanently store application logs for regulatory** Cloud storage - Archive **reasons. You don't expect to access them at all.**



**Log files stored in Cloud storage. You expect to access them** Cold Line **once in quarter.**



**How do you change storage class of an existing bucket in** Step 1: Change Default Storage Class of the

**Cloud Storage?** bucket.

Step 2: Update the Storage Class of the objects in the bucket.



## Cloud Storage - Command Line - gsutil - 1

(REMEMBER) **gsutil** is the CLI for Cloud Storage (**NOT gcloud**)

Cloud Storage (**gsutil**) *gsutil* ***mb*** *gs://BKT\_NAME* (Create Cloud Storage bucket) *gsutil* ***ls -a*** *gs://BKT\_NAME* (List current and non-current object versions)

*gsutil* ***cp*** *gs://SRC\_BKT/SRC\_OBJ gs://DESTN\_BKT/NAME\_COPY* (Copy objects)

-o '**GSUtil:encryption\_key=ENCRYPTION\_KEY**' - Encrypt Object  *gsutil* ***mv*** (Rename/Move objects)  *gsutil mv gs://BKT\_NAME/OLD\_OBJ\_NAME gs://BKT\_NAME/NEW\_OBJ\_NAME*

*gsutil mv gs://OLD\_BUCKET\_NAME/OLD\_OBJECT\_NAME gs://NEW\_BKT\_NAME/NEW\_OBJ\_NAME*

*gsutil* ***rewrite*** *-s STORAGE\_CLASS gs://BKT\_NAME/OBJ\_PATH* (Ex: Change Storage Class for objects)

*gsutil* ***cp*** : Upload and Download Objects  *gsutil cp LOCAL\_LOCATION gs://DESTINATION\_BKT\_NAME/* (Upload) *gsutil cp gs://BKT\_NAME/OBJ\_PATH LOCAL\_LOCATION* (Download)

## Cloud Storage - Command Line - gsutil - 2

Cloud Storage (gsutil)

*gsutil* ***versioning set on/off*** *gs://BKT\_NAME* (Enable/Disable Versioning) *gsutil* ***uniformbucketlevelaccess set on/off*** *gs://BKT\_NAME*

*gsutil* ***acl ch*** (Set Access Permissions for Specific Objects) *gsutil acl ch -u AllUsers:R gs://BKT\_NAME/OBJ\_PATH* (Make specific object public) *gsutil acl ch -u john.doe@example.com:WRITE gs://BKT\_NAME/OBJ\_PATH*

Permissions - READ (R), WRITE (W), OWNER (O)

Scope - User, allAuthenticatedUsers, allUsers(-u), Group (-g), Project (-p) etc gsutil acl set JSON\_FILE gs://BKT\_NAME

*gsutil* ***iam ch*** *MBR\_TYPE:MBR\_NAME:IAM\_ROLE gs://BKT\_NAME* (Setup IAM role)  *gsutil iam ch user:me@myemail.com:objectCreator gs://BKT\_NAME*

*gsutil iam ch allUsers:objectViewer gs://BKT\_NAME* (make the entire bucket readable)

*gsutil* ***signurl -d 10m*** *YOUR\_KEY gs://BUCKET\_NAME/OBJECT\_PATH* (Signed URL for temporary access)

# IAM

## Typical identity management in the cloud

 You have **resources** in the cloud (examples - a virtual server, a database etc)

You have **identities (human and non-human)** that need to access those resources and perform actions

For example: launch (stop, start or terminate) a virtual server How do you **identify users** in the cloud?

How do you configure resources they can access?

How can you configure what actions to allow?

In GCP: *Identity and Access Management (Cloud IAM)* provides this service

## Cloud Identity and Access Management (IAM)

**Authentication** (is it the right user?) and

**Authorization** (do they have the right access?)

**Identities** can be

A GCP User (Google Account or Externally Authenticated User)

A Group of GCP Users

An Application running in GCP

An Application running in your data center

Unauthenticated users

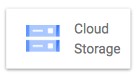
Provides very **granular** control

Limit a single user:

to perform single action on a specific cloud resource from a specific IP address during a specific time window

## Cloud IAM Example

I want to provide access to manage a specific cloud storage bucket to a colleague of mine: Important Generic Concepts:



**Member**: My colleague

**Resource**: Specific cloud storage bucket

**Action**: Upload/Delete Objects In Google Cloud IAM:

**Roles**: A set of permissions (to perform specific actions on specific resources)

**Roles do NOT know about members**. It is all about permissions!

How do you assign permissions to a member?

**Policy**: You assign (or **bind**) a role to a member

1: **Choose a Role** with right permissions (Ex: Storage Object Admin)

2: **Create Policy** binding member (your friend) with role (permissions)

IAM in AWS is very different from GCP (Forget AWS IAM & Start FRESH!)

Example: Role in AWS is NOT the same as Role in GCP

## IAM - Roles

**Roles are Permissions**:

Perform some set of actions on some set of resources Three Types:

**Basic Roles (or Primitive roles)** - Owner/Editor/Viewer

**Viewer(roles.viewer)** - Read-only actions

**Editor(roles.editor)** - Viewer + Edit actions

**Owner(roles.owner)** - Editor + Manage Roles and Permissions + Billing

EARLIEST VERSION: Created before IAM NOT RECOMMENDED: **Don't use in production**

**Predefined Roles** - Fine grained roles predefined and managed by Google

Different roles for different purposes

**Examples**: Storage Admin, Storage Object Admin, Storage Object Viewer, Storage Object Creator  **Custom Roles** - When predefined roles are NOT sufficient, you can create your own custom roles

## IAM - Predefined Roles - Example Permissions

Important **Cloud Storage Roles**: **Storage Admin (roles/storage.admin)** storage.buckets.\*

storage.objects.\*

**Storage Object Admin (roles/storage.objectAdmin)** storage.objects.\*

**Storage Object Creator (roles/storage.objectCreator)** storage.objects.create

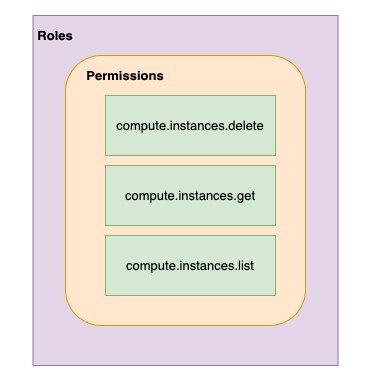
**Storage Object Viewer (roles/storage.objectViewer)** storage.objects.get

storage.objects.list

All four roles have these permissions:

resourcemanager.projects.get resourcemanager.projects.list

## IAM - Most Important Concepts - A Review

**Member** : Who?

**Roles** : Permissions (What Actions? What Resources?)

**Policy** : Assign Permissions to Members Map Roles (What?) , Members (Who?) and Conditions (Which Resources?, When?, From Where?)

Remember: Permissions are NOT directly assigned to Member

Permissions are represented by a Role

Member gets permissions through Role!

A Role can have multiple permissions

You can assign multiple roles to a Member

## IAM policy

Roles are assigned to users through **IAM Policy** documents

Represented by a **policy object**

Policy object has list of bindings

A binding, binds a role to list of members

Member type is identified by **prefix**:

Example: user, serviceaccount, group or domain

## IAM policy - Example

{

"bindings": [

{

"role": "roles/storage.objectAdmin",

"members": [

"user:you@in28minutes.com",

"serviceAccount:myAppName@appspot.gserviceaccount.com",

"group:administrators@in28minutes.com",

"domain:google.com"

]

},

{

"role": "roles/storage.objectViewer",

"members": [

"user:you@in28minutes.com"

],

"condition": {

"title": "Limited time access",

"description": "Only upto Feb 2022",

"expression": "request.time < timestamp('2022-02-01T00:00:00.000Z')",

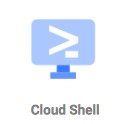
}

}

]

}

## Playing With IAM

**gcloud**: Playing with IAM

gcloud **compute project-info** describe - Describe current project gcloud **auth login** - Access the Cloud Platform with Google user credentials gcloud **auth revoke** - Revoke access credentials for an account

gcloud **auth list** - List active accounts gcloud **projects**

gcloud projects **add-iam-policy-binding** - Add IAM policy binding gcloud projects **get-iam-policy** - Get IAM policy for a project gcloud projects **remove-iam-policy-binding** - Remove IAM policy binding

gcloud projects **set-iam-policy** - Set the IAM policy gcloud projects **delete** - Delete a project

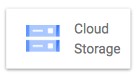
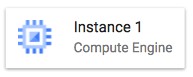
gcloud **iam**

gcloud iam **roles describe** - Describe an IAM role

gcloud iam **roles create** - create an iam role(--project, --permissions, --stage) gcloud iam **roles copy** - Copy IAM Roles

## Service Accounts

Scenario: An Application on a VM needs access to cloud storage



You DONT want to use personal credentials to allow access

(RECOMMENDED) Use **Service Accounts**

Identified by an email address (Ex: id-compute@developer.gserviceaccount.com)

Does NOT have password

Has a **private/public RSA key-pairs**

Can't login via browsers or cookies

Service account types:

**Default service account** - Automatically created when some services are used

(NOT RECOMMENDED) Has **Editor role** by default

**User Managed** - User created

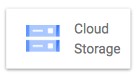
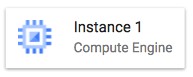
(RECOMMENDED) Provides fine grained access control

**Google-managed service** accounts - Created and managed by Google

Used by GCP to perform operations on user's behalf

In general, we DO NOT need to worry about them

## Use case 1 : VM <-> Cloud Storage



**1:** Create a Service Account Role with the right permissions

**2:** Assign Service Account role to VM instance **Uses Google Cloud-managed keys**:

Key generation and use are automatically handled by IAM when we assign a service account to the instance Automatically rotated

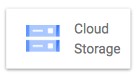
No need to store credentials in config files

**Do NOT delete** service accounts used by running instances:

Applications running on those instances will lose access!

## Use case 2 : On Prem <-> Cloud Storage (Long Lived)

You **CANNOT assign Service Account directly to an On Prem App**



**1:** Create a **Service Account** with right permissions

**2:** Create a **Service Account User Managed Key *gcloud iam service-accounts keys create***

Download the service account key file

Keep it secure (It can be used to impersonate service account)!

**3:** Make the service account key file accessible to your application

Set environment variable GOOGLE\_APPLICATION\_CREDENTIALS  ***export GOOGLE\_APPLICATION\_CREDENTIALS="/PATH\_TO\_KEY\_FILE"***

**4:** Use **Google Cloud Client Libraries**

Google Cloud Client Libraries use a library - Application Default Credentials (ADC) ADC uses the service account key file if env var GOOGLE\_APPLICATION\_CREDENTIALS exists!

## Use case 3 : On Prem <-> Google Cloud APIs (Short Lived)

 **Make calls from outside GCP to Google Cloud APIs** with short lived permissions

Few hours or shorter

**Less risk** compared to sharing service account keys!

**Credential Types**:

OAuth 2.0 access tokens

OpenID Connect ID tokens

Self-signed JSON Web Tokens (JWTs)

**Examples**:

When a member needs elevated permissions, he can assume the service account role (Create OAuth 2.0 access token for service account)

OpenID Connect ID tokens is recommended for service to service authentications:

A service in GCP needs to authenticate itself to a service in other cloud

## Service Account Use case Scenarios

**Scenario Solution**



**Application on a VM wants to talk to a** Configure the VM to use a Service Account with right permissions **Cloud Storage bucket**



**Application on a VM wants to put a** Configure the VM to use a Service Account with right permissions **message on a Pub Sub Topic**



**Is Service Account an identity or a** It is both. You can attach roles with Service Account (identity).

**resource?** You can let other members access a SA by granting them a role on

the Service Account (resource).



**VM instance with default service account** In project B, add the service account from Project A and assign **in Project A needs to access Cloud** Storage Object Viewer Permission on the bucket **Storage bucket in Project B**



## ACL (Access Control Lists)

**ACL**: Define **who** has access to your buckets and objects, as

well as **what level** of access they have **How is this different from IAM?**

IAM permissions apply to all objects within a bucket

ACLs can be used to customized specific accesses to different objects User gets access if he is allowed by either IAM or ACL! (Remember) **Use IAM for common permissions** to all objects in a bucket

(Remember) **Use ACLs** if you need to **customize access to individual objects**

## Access Control - Overview

 How do you control access to objects in a Cloud Storage bucket?

Two types of access controls:

**Uniform** (Recommended) - Uniform bucket level access using IAM **Fine-grained** - Use IAM and ACLs to control access:

Both bucket level and individual object level permissions

Use Uniform access when all users have same level of access across all objects in a bucket

Fine grained access with ACLs can be used when you need to customize the access at an object level

Give a user specific access to edit specific objects in a bucket

## Cloud Storage - Signed URL

 You would want to **allow a user limited time access** to your objects:

Users do NOT need Google accounts

Use **Signed URL** functionality

A URL that gives **permissions for limited time duration** to perform specific actions

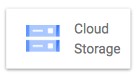
**To create a Signed URL**:

**1:** Create a key (YOUR\_KEY) for the Service Account/User with the desired permissions

**2:** Create Signed URL with the key:

***gsutil signurl -d 10m YOUR\_KEY gs://BUCKET\_NAME/OBJECT\_PATH***

## Cloud Storage - Static website



**1:** Create a bucket with the **same name** as website name (Name of bucket should match DNS name of the website)

**Verify** that the domain is owned by you

**2:** Copy the files to the bucket

Add index and error html files for better user experience

**3:** Add member **allUsers** and grant **Storage Object Viewer** option

Select **Allow Public Access**

# Database Fundamentals

## Databases Primer

 Databases provide **organized** and **persistent** storage for your data

To **choose between different database types**, we would need to understand:

Availability

Durability

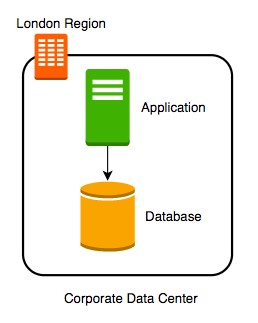
RTO

RPO

Consistency Transactions etc

Let's get started on a **simple journey** to understand these

## Database - Getting Started

 Imagine a database deployed **in a data center in London**

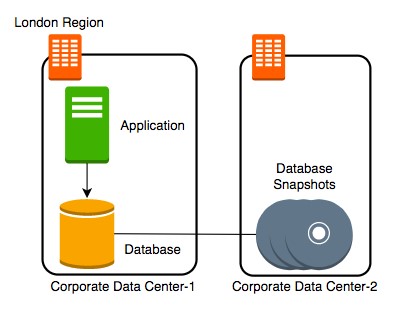
Let's consider some challenges:

**Challenge 1**: Your database will go down if the data center

crashes or the server storage fails

**Challenge 2**: You will lose data if the database crashes

## Database - Snapshots

 Let's automate taking copy of the database (**take a snapshot**) every hour to another data

center in London

Let's consider some challenges:

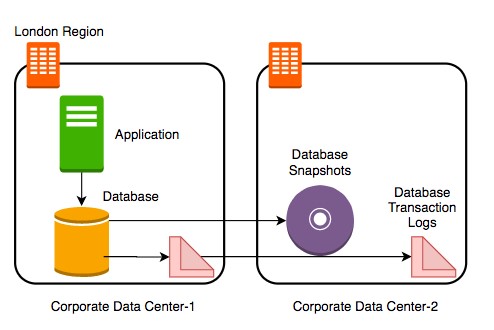
**Challenge 1**: Your database will go down if the data center crashes

**Challenge 2** (PARTIALLY SOLVED): You will lose data if the database crashes

You can setup database from latest snapshot. But depending on when failure occurs you can lose up to an hour of data

**Challenge 3**(NEW): Database will be slow when you take snapshots

## Database - Transaction Logs

 Let's add **transaction logs** to database and create a **process to copy it over** to the second

data center

Let's consider some challenges:

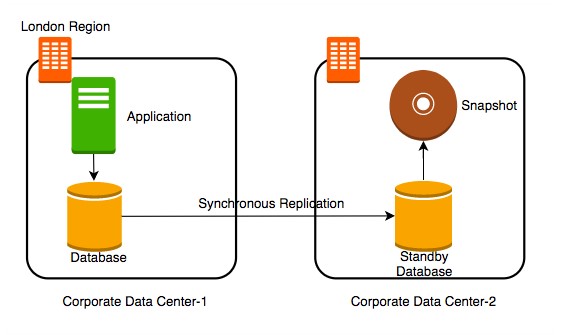
**Challenge 1**: Your database will go down if the data center crashes

**Challenge 2** (SOLVED): You will lose data if the database crashes

You can setup database from latest snapshot and apply transaction logs

**Challenge 3**: Database will be slow when you take snapshots

## Database - Add a Standby

 Let's add a **standby database** in the second data center with replication Let's consider some challenges:

**Challenge 1** (SOLVED): Your database will go down if the data center crashes You can switch to the standby database  **Challenge 2** (SOLVED): You will lose data if the database crashes

**Challenge 3** (SOLVED): Database will be slow when you take snapshots

Take snapshots from standby.

Applications connecting to master will get good performance always

## Availability and Durability

**Availability**

Will I be able to access my data now and when I need it?

Percentage of time an application provides the operations expected of it

**Durability**

Will my data be available a er 10 or 100 or 1000 years?

Examples of measuring availability and durability:

4 9's - 99.99 11 9's - 99.999999999

Typically, an **availability of four 9's** is considered very good

Typically, a **durability of eleven 9's** is considered very good

## Availability

**Availability Downtime (in a month) Comment**



**99.95%** 22 minutes



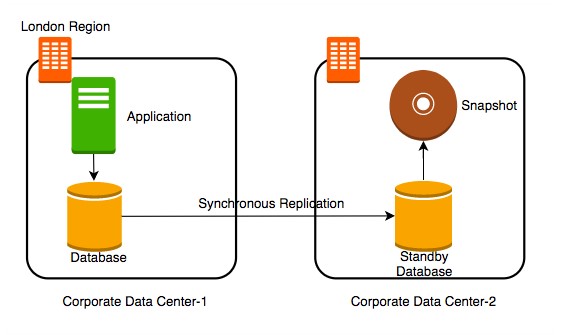
**99.99% (4 9's)** 4 and 1/2 minutes Typically online apps aim for 99.99% (4 9's) availability



**99.999% (5 9's)** 26 seconds Achieving 5 9's availability is tough



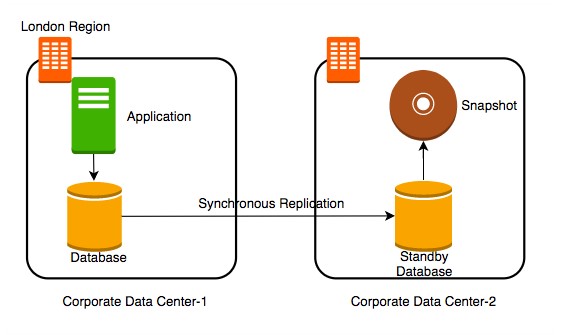
## Durability

 What does a **durability of 11 9's** mean?

If you **store one million files for ten million years**, you would expect to **lose one file** Why should durability be high?

Because **we hate losing data** Once we lose data, it is gone

## Increasing Availability and Durability of Databases

**Increasing Availability**:

Have multiple standbys available OR distribute the database in multiple Zones

in multiple Regions

**Increasing Durability**:

Multiple copies of data (standbys,

snapshots, transaction logs and replicas) in multiple Zones

in multiple Regions

**Replicating data** comes with its own challenges!

We will talk about them a little later

## Database Terminology : RTO and RPO

Imagine a **financial transaction being lost**

Imagine a **trade being lost**

Imagine a **stock exchange going down** for an hour

**Typically** businesses are fine with some downtime but they hate losing data

Availability and Durability are technical measures

How do we measure **how quickly we can recover from failure**?

**RPO (Recovery Point Objective)**: Maximum acceptable period of data loss

**RTO (Recovery Time Objective)**: Maximum acceptable downtime

Achieving **minimum RTO and RPO is expensive**

**Trade-off** based on the criticality of the data

## Question - RTO and RPO

You are running an application in VM instance storing its data on a persistent data storage. You are taking snapshots every 48 hours. If the VM instance crashes, you can manually bring it back up in 45 minutes from the snapshot. What is your RTO and RPO?

RTO - 45 minutes

RPO - 48 hours

## Achieving RTO and RPO - Failover Examples

**Scenario Solution**



**Very small data loss (RPO - 1 minute) Hot standby** - Automatically synchronize data **Very small downtime (RTO - 5 minutes)** Have a standby ready to pick up load

Use automatic failover from master to standby



**Very small data loss (RPO - 1 minute) Warm standby** - Automatically synchronize data

**BUT I can tolerate some downtimes (RTO - 15 minutes)** Have a standby with minimum infrastructure

Scale it up when a failure happens



**Data is critical (RPO - 1 minute) but I can tolerate** Create regular data **snapshots and transaction downtime of a few hours (RTO - few hours) logs**

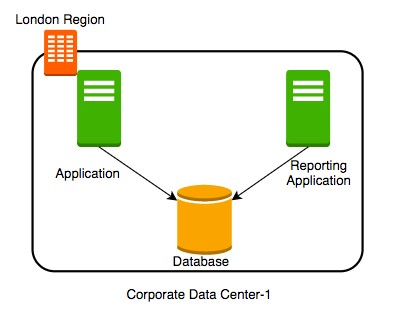
Create database from snapshots and transactions logs when a failure happens



**Data can be lost without a problem (for example: cached** Failover to a completely new server **data)**



## (New Scenario) Reporting and Analytics Applications

 New reporting and analytics applications are being launched using the same database

These applications will ONLY read data

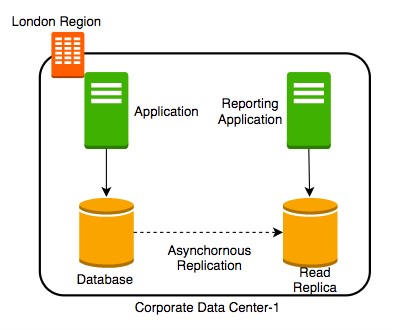
Within a few days you see that the database

performance is impacted How can we fix the problem?

**Vertically scale the database** - increase CPU and memory **Create a database cluster (Distribute the database)** - Typically database clusters are expensive to setup

**Create read replicas** - Run read only applications against read replicas

## Database - Read Replicas

Add **read replica**

Connect reporting and analytics applications

to **read replica**

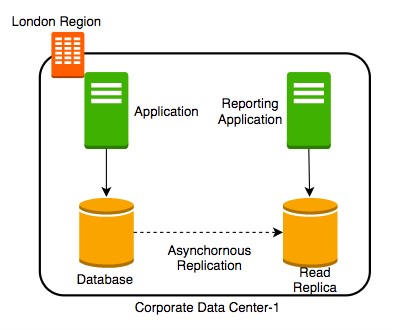
Reduces load on the master databases

Upgrade read replica to master database (supported by some databases)

Create read replicas **in multiple regions**

**Take snapshots** from read replicas

## Consistency

 How do you ensure that data in multiple database instances (standbys and replicas) is updated simultaneously? **Strong consistency** - Synchronous replication to all replicas

Will be slow if you have multiple replicas or standbys

**Eventual consistency** - Asynchronous replication. A little lag few seconds - before the change is available in all replicas In the intermediate period, different replicas might return different values

Used when scalability is more important than data integrity

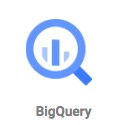
Examples : Social Media Posts - Facebook status messages, Twitter tweets, Linked in posts etc

**Read-a er-Write consistency** - Inserts are immediately available

However, updates would have eventual consistency

## Database Categories

There are **several categories** of databases:



Relational (OLTP and OLAP), Document, Key Value, Graph, In Memory among others  **Choosing type of database** for your use case is not easy. A few factors: Do you want a **fixed schema**?

Do you want flexibility in defining and changing your schema? (schemaless)

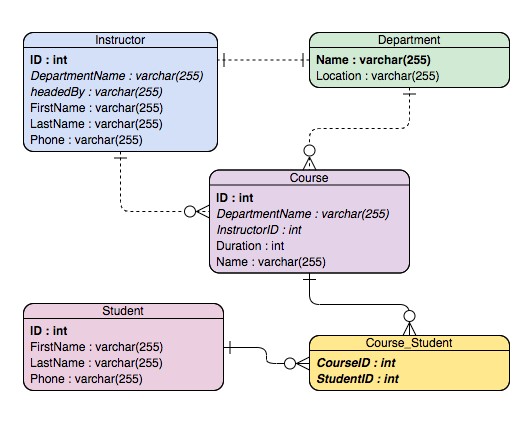
What level of **transaction properties** do you need? (atomicity and consistency)

What kind of **latency** do you want? (seconds, milliseconds or microseconds)

**How many transactions** do you expect? (hundreds or thousands or millions of transactions per second)

**How much data** will be stored? (MBs or GBs or TBs or PBs) and a lot more...

## Relational Databases

 This was the **only option** until a decade back!

Most **popular (or unpopular)** type of databases

**Predefined schema** with tables and relationships

Very **strong transactional** capabilities

Used for

OLTP (Online Transaction Processing) use cases and

OLAP (Online Analytics Processing) use

cases

## Relational Database - OLTP (Online Transaction Processing)

Applications where **large number of users make large number of small transactions**



small data reads, updates and deletes

**Use cases**:

Most traditional applications, ERP, CRM, e-commerce, banking applications

**Popular databases**:

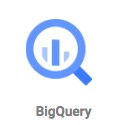
MySQL, Oracle, SQL Server etc

Recommended Google Managed Services:

**Cloud SQL** : Supports PostgreSQL, MySQL, and SQL Server for regional relational databases (upto a few TBs)

**Cloud Spanner**: Unlimited scale (multiple PBs) and 99.999% availability for global applications with horizontal scaling

## Relational Database - OLAP (Online Analytics Processing)

Applications allowing users to **analyze petabytes of data**

**Examples** : Reporting applications, Data ware houses, Business intelligence applications, Analytics systems

**Sample application** : Decide insurance premiums analyzing data from last hundred years

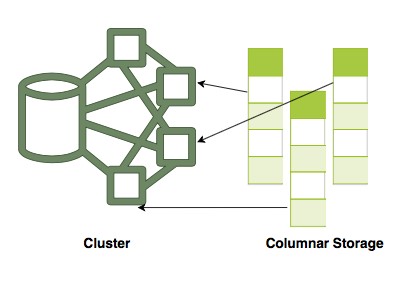
Data is consolidated from multiple (transactional) databases

Recommended GCP Managed Service

**BigQuery**: **Petabyte-scale** distributed data ware house

## Relational Databases - OLAP vs OLTP

OLAP and OLTP use **similar data structures**



BUT **very different approach in how data is stored**

**OLTP databases** use row storage

Each table row is stored together Efficient for processing small transactions

**OLAP databases** use columnar storage

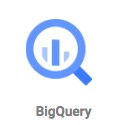
Each table column is stored together

**High compression** - store petabytes of data efficiently

**Distribute data** - one table in multiple cluster nodes **Execute single query across multiple nodes** Complex queries can be executed efficiently

## NoSQL Databases

**New approach** (actually NOT so new!) to building your databases



NoSQL = not only SQL

Flexible schema

Structure data **the way your application needs it** Let the schema evolve with time

Horizontally scale to petabytes of data with millions of TPS

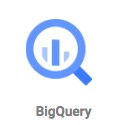
**NOT a 100% accurate generalization** but a great starting point: Typical NoSQL databases trade-off "Strong consistency and SQL features" to achieve "scalability and high-performance"

**Google Managed Services**:

Cloud Firestore (Datastore)

Cloud BigTable

**Cloud Firestore (Datastore) vs Cloud BigTable Cloud Datastore** - Managed serverless NoSQL document database



Provides ACID transactions, SQL-like queries, indexes

Designed for transactional mobile and web applications Firestore (next version of Datastore) adds:

Strong consistency

Mobile and Web client libraries

Recommended for small to medium databases (0 to a few Terabytes) **Cloud BigTable** - Managed, scalable NoSQL wide column database

NOT serverless (You need to create instances)

Recommend for data size > 10 Terabytes to several Petabytes Recommended for large analytical and operational workloads:

NOT recommended for transactional workloads (Does NOT support multi row transactions supports ONLY Single-row transactions)

## In-memory Databases

 **Retrieving data from memory is much faster than retrieving data from disk**

In-memory databases like Redis deliver microsecond latency by storing **persistent data in memory** Recommended GCP Managed Service

**Memory Store**

**Use cases** : Caching, session management, gaming leader boards, geospatial applications

## Databases - Summary

**Database Type GCP Services Description**



**Relational OLTP** Cloud SQL, Cloud Transactional usecases needing **predefined schema** and very **databases** Spanner **strong transactional** capabilities (Row storage)

**Cloud SQL**: MySQL, PostgreSQL, SQL server DBs

**Cloud Spanner**: Unlimited scale and 99.999% availability for global applications with horizontal scaling



**Relational OLAP** BigQuery Columnar storage with predefined schema. Datawarehousing & **databases** BigData workloads



**NoSQL Databases** Cloud Firestore Apps needing **quickly evolving** structure (**schema-less**)

(Datastore) , Cloud **Cloud Firestore** - Serverless transactional document DB

BigTable supporting mobile & web apps. Small to medium DBs (0 - few

TBs)

**Cloud BigTable** - Large databases(10 TB - PBs). Streaming (IOT), analytical & operational workloads. NOT serverless.



**In memory** Cloud Memorystore Applications needing **microsecond** responses **databases/caches**



## Databases - Scenarios

**Scenario Solution**



**A start up with quickly evolving schema (table structure)** Cloud

Datastore/Firestore



**Non relational db with less storage (10 GB)** Cloud Datastore



**Transactional global database with predefined schema needing to process million of** CloudSpanner **transactions per second**



**Transactional local database processing thousands of transactions per second** Cloud SQL



**Cache data (from database) for a web application** MemoryStore



**Database for analytics processing of petabytes of data** BigQuery



**Database for storing huge volumes stream data from IOT devices** BigTable



**Database for storing huge streams of time series data** BigTable



## Cloud SQL

**Fully Managed Relational Database** service

Configure your needs and do NOT worry about managing the database

Supports MySQL, PostgreSQL, and SQL Server

Regional Service providing High Availability (99.95%)

Use SSDs or HDDs (For best performance: use SSDs)

Upto 416 GB of RAM and 30 TB of data storage

**Use Cloud SQL for simple relational use cases**:

To migrate local MySQL, PostgreSQL, and SQL Server databases

To reduce your maintenance cost for a simple relational database

(REMEMBER) Use Cloud Spanner(Expensive $$$$) instead of Cloud SQL if:

You have huge volumes of relational data (TBs) OR

You need infinite scaling for a growing application (to TBs) OR

You need a Global (distributed across multiple regions) Database OR

You need higher availability (99.999%)

## Cloud SQL - Features

**Important Cloud SQL Features**:

Automatic encryption (tables/backups), maintenance and updates High availability and failover:

Create a Standby with automatic failover

Pre requisites: Automated backups and Binary logging Read replicas for read workloads:

Options: Cross-zone, Cross-region and External (NON Cloud SQL DB)

Pre requisites: Automated backups and Binary logging

Automatic storage increase without downtime (for newer versions)

Point-in-time recovery: Enable binary logging

Backups (Automated and on-demand backups)

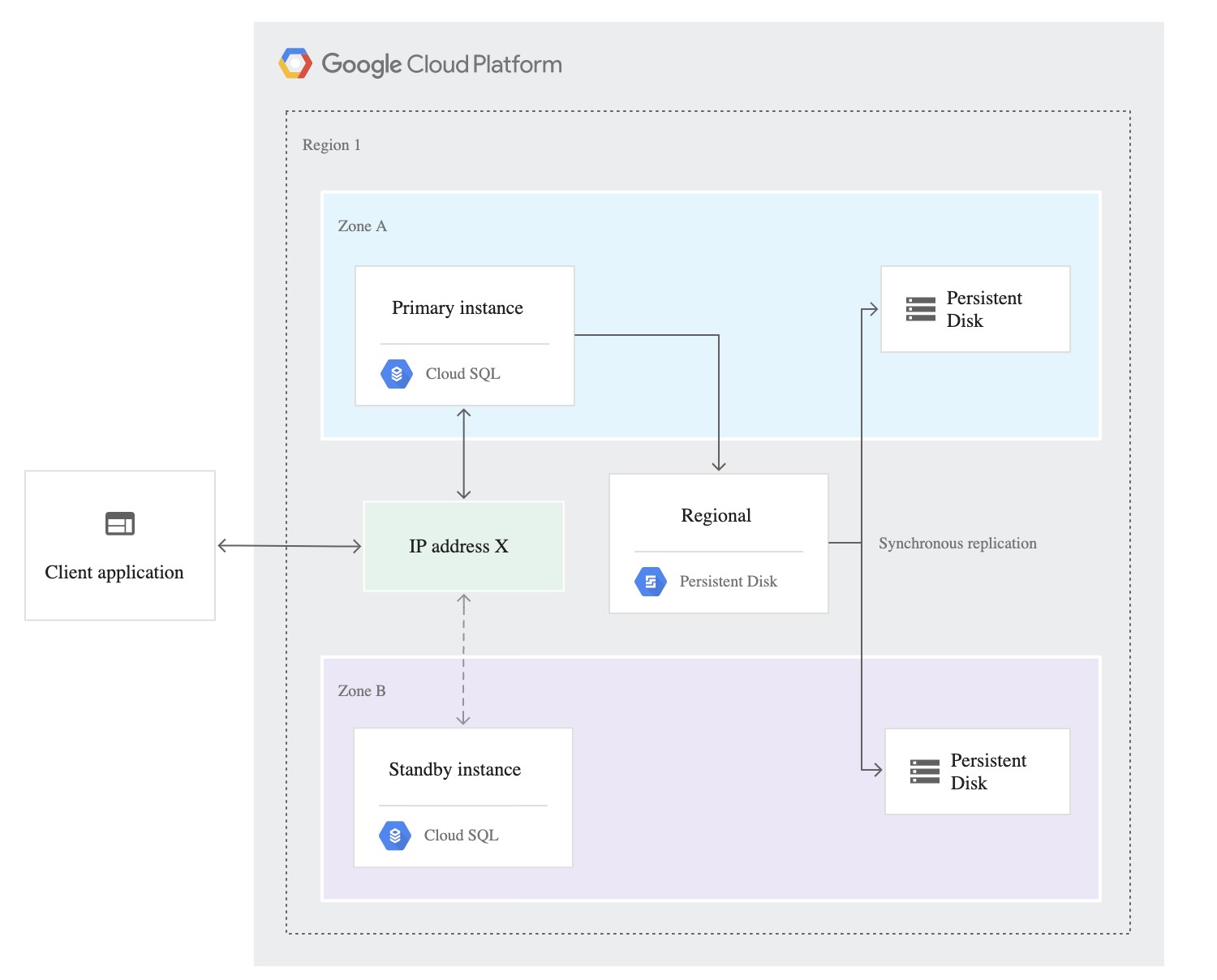
Supports migration from other sources

Use Database Migration Service (DMS)

You can export data from UI (console) or gcloud with formats:

SQL (Recommended if you import data into other databases) and CSV

## Cloud SQL - High Availability

Create a High Availability (HA) Configuration Choose **Primary and Secondary** zones within a region You will have two instances : **Primary** and **Secondary** instances

Changes from primary are replicated **synchronously** to secondary

In case of **Zonal** failure, automatic failover to secondary instance:

If **Primary zone** becomes available, failover does not revert automatically

(Remember) **High Availability** setup CANNOT

be used as a **Read Replica** source:cloud.google.com

## Cloud Spanner

 **Fully managed, mission critical, relational(SQL), globally distributed database** with VERY high availability (99.999%)

Strong transactional consistency at **global scale**

**Scales to PBs of data** with automatic sharding

Cloud Spanner **scales horizontally for reads and writes** Configure no of nodes

(REMEMBER) In comparison, Cloud SQL provides read replicas:

BUT you cannot horizontally scale write operations with Cloud SQL!

Regional and Multi-Regional configurations

**Expensive** (compared to Cloud SQL): Pay for nodes & storage

**Data Export**: Use Cloud Console to export data

Other option is to use Data flow to automate export

No gcloud export option

## Cloud Datastore and Firestore

**Datastore** - Highly scalable NoSQL Document Database

Automatically scales and partitions data as it grows

Recommended for upto a few TBs of data

For bigger volumes, BigTable is recommended

Supports Transactions, Indexes and SQL like queries (GQL)

Does NOT support Joins or Aggregate (sum or count) operations

For use cases needing flexible schema with transactions

Examples: User Profile and Product Catalogs

Structure: Kind > Entity (Use namespaces to group entities)

You can export data ONLY from gcloud (NOT from cloud console)

Export contains a metadata file and a folder with the data

**Firestore** = Datastore++ : Optimized for multi device access

Offline mode and data synchronization across multiple devices - mobile, IOT etc

Provides client side libraries - Web, iOS, Android and more

Offers Datastore and Native modes

## Cloud BigTable

**Petabyte scale, wide column** NoSQL DB (HBase API compatible)

Designed for huge volumes of analytical and operational data

IOT Streams, Analytics, Time Series Data etc

Handle millions of read/write TPS at very low latency

Single row transactions (multi row transactions NOT supported)

**NOT serverless**: You need to create a server instance (Use SSD or

HDD)

Scale horizontally with multiple nodes (No downtime for cluster resizing)

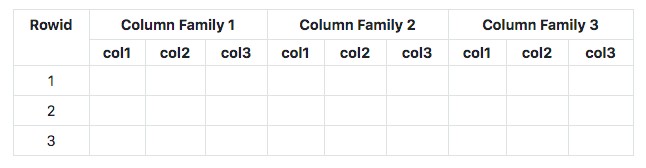
**CANNOT export data using cloud console or gcloud**:

Either use a Java application (java -jar JAR export\import) OR Use HBase commands

Use cbt command line tool to work with BigTable (NOT gcloud)

Ex: **cbt createtable my-table**

## Cloud BigTable - Wide Column Database



At the most basic level, each table is a sorted key/value map

Each value in a row is indexed using a key - **row key**

Related columns are grouped into column families

Each column is identified by using column-family:column-qualifer(or name)

This structure supports high read and write throughput at low latency

**Advantages** : Scalable to **petabytes of data** with **millisecond responses** upto **millions of TPS Use cases** : IOT streams, graph data and real time analytics (time-series data, financial data - transaction histories, stock prices etc)

**Cloud Dataflow** : Used to export data from BigTable to CloudStorage

For ex: Cloud Bigtable to Cloud Storage Avro, Cloud Bigtable to Cloud Storage Parquet

## Memorystore

**In-memory datastore service**: Reduce access times

**Fully managed** (Provisioning, Replication, Failover and

Patching)

Highly available with 99.9% availability SLA

Monitoring can be easily setup using Cloud Monitoring

Support for **Redis and Memcached**:

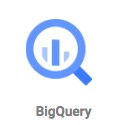
Use Memcached for Caching

Reference data, database query caching, session store etc

Use Redis for low latency access with persistence and high availability

Gaming Leader Boards, Player Profiles, In memory Stream Processing etc

## BigQuery - Datawarehouse

**Exabyte scale modern Datawarehousing** solution from GCP

Relational database (SQL, schema, consistency etc)

Use **SQL-like commands** to query massive datasets

Traditional (Storage + Compute) + Modern (Realtime + Serverless)

When we are talking about a Datawarehouse, **importing and exporting data (and formats) becomes very important**:

Load data from a **variety of sources, incl. streaming data**

Variety of import formats - CSV/JSON/Avro/Parquet/ORC/Datastore backup

Export to Cloud Storage (long term storage) & Data Studio (visualization)

Formats - CSV/JSON (with Gzip compression), Avro (with deflate or snappy compression)

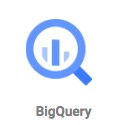
Automatically expire data (**Configurable Table Expiration**)

Query **external data sources** without storing data in BigQuery

Cloud Storage, Cloud SQL, BigTable, Google Drive

Use **Permanent or Temporary** external tables

## BigQuery - Accessing and Querying Data

**Access databases using**: Cloud Console

bq command-line tool (NOT gcloud)

BigQuery Rest API OR

HBase API based libraries (Java, .NET & Python)

(Remember) BigQuery queries **can be expensive** as you are running them on large data sets!

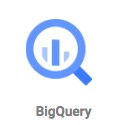
(BEST PRACTICE) **Estimate BigQuery queries before running**:

**1:** Use UI(console)/bq(--dry-run) - Get scanned data volume (estimate)

**2:** Use Pricing Calculator: Find price for scanning 1 MB data. Calculate cost.

## Relational Database - Import and Export

**Cloud SQL** : to/from Cloud Storage (***gcloud sql export/import csv/sql***)



From Console/gcloud/REST API

SQL and CSV formats

For large databases, use serverless mode

Reduces performance impact of export on the live database

**Cloud Spanner**: to/from Cloud Storage

From Console (uses Cloud Data Flow)

**BigQuery**: to/from Cloud Storage and Others (***bq extract/load***)

From Console/bq

Formats - CSV/JSON (with Gzip compression), Avro (with deflate or snappy compression) Variety of options to import data:

Load data from Cloud Storage

Example Use Case: Data Store > Cloud Storage > Big Query

Batch Loading with BigQuery Data Transfer Service

Use Dataflow to setup streaming pipeline

## NoSQL Databases - Import and Export

**Cloud Datastore/Firestore**: to/from Cloud Storage From Console/gcloud/REST API ***gcloud datastore/firestore export/import --kinds --namespaces***



**Cloud BigTable**: to/from Cloud Storage

Create Dataflow jobs

Formats: Avro/Parquet/SequenceFiles

(REMEMBER) Ensure that service accounts have access to Cloud Storage Buckets

ACL (***gsutil acl ch -U SERVICE\_ACCOUNT :W BUCKET***) OR

Roles **Storage Admin** or **Storage Object Admin** or **Storage Object Creator**

## Cloud SQL - CommandLine

***gcloud sql***

*gcloud sql* ***instances create/clone/delete/describe/patch***  *gcloud sql instances* ***create INSTANCE***

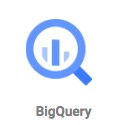
*gcloud sql instances* ***patch --backup-start-time***

*gcloud sql* ***databases create/delete/describe/list/patch***  *gcloud sql databases* ***create DATABASE --instance=INSTANCE***

*gcloud sql* ***connect INSTANCE [--database=DATABASE --user=root]*** gcloud sql **backups create/describe/list**

*gcloud sql* ***backups create*** *--async --instance [INSTANCE]* (one time backup)

## BigQuery - Command Line

BigQuery (**bq**)

*bq* ***show*** *bigquery-public-data:samples.shakespeare bq* ***query*** *'QUERY-STRING'*

**--dry-run** - To estimate the bytes scanned by a query *bq* ***extract*** (export data)

*bq* ***load*** (load data)

(Remember) Use the standard way to set the project *gcloud config set project my-project*

## cbt tool - Cloud Bigtable - CommandLine

**cbt** - CLI for Cloud Bigtable (NOT gcloud)

Installing - *gcloud components install cbt*

Verify Installation - *cbt listinstances*

Create **.cbtrc file** with the configuration *echo project = project-id > ~/.cbtrc echo instance = quickstart-instance >> ~/.cbtrc*

Commands (**cbt**):

cbt **createinstance** - Create an instance cbt **createcluster** - Create a cluster within configured instance!

cbt **createtable/deleteinstance/deletecluster/deletetable** cbt **listinstances/listclusters**

cbt **ls** - list tables and column families

(Remember) You can **configure your project** with cbt in .cbtrc file

## Databases - Remember

(Remember) BigQuery, Datastore, Firebase does NOT need VM configuration whereas Cloud SQL and BigTable need VM configuration

**Relational Databases**

Small Local Databases - Cloud SQL

Highly scalable global databases - Cloud Spanner Datawarehouse - BigQuery

**NoSQL Databases**

Transactional database for a few Terabytes of data - Cloud Datastore

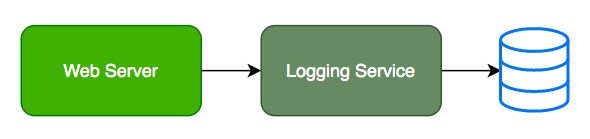
Huge volumes of IOT or streaming analytics data - Cloud BigTable

# Decoupling Applications with Pub/Sub

## Need for Asynchronous Communication

Why do we need asynchronous communication?

## Synchronous Communication



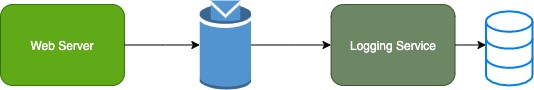
Applications on your web server make synchronous calls to the logging service

What if your logging service goes down?

Will you applications go down too?

What if all of sudden, there is high load and there are lot of logs coming in? Log Service is not able to handle the load and goes down very o en

## Asynchronous Communication - Decoupled



Create a topic and have your applications put log messages on the topic Logging service picks them up for processing when ready Advantages:

Decoupling: Publisher (Apps) don't care about who is listening

Availability: Publisher (Apps) up even if a subscriber (Logging Service) is down

Scalability: Scale consumer instances (Logging Service) under high load

Durability: Message is not lost even if subscriber (Logging Service) is down

## Pub/Sub

 Reliable, scalable, fully-managed asynchronous messaging service

Backbone for **Highly Available** and **Highly Scalable**

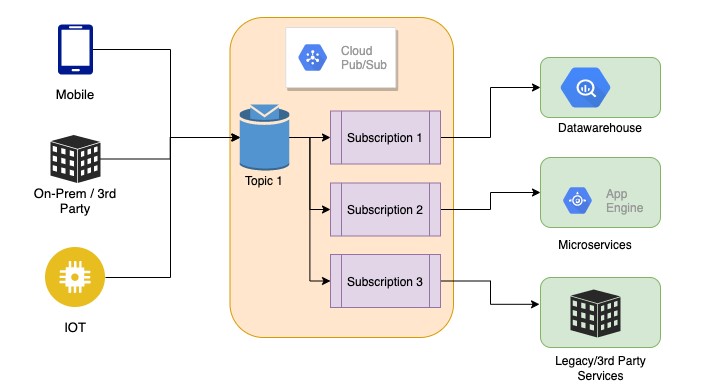
Solutions

Auto scale to process billions of messages per day Low cost (Pay for use)

Usecases: Event ingestion and delivery for streaming analytics pipelines

Supports push and pull message deliveries

**Pub/Sub - How does it work?**

**Publisher** - Sender of a message Publishers send messages by making HTTPS requests to pubsub.googleapis.com

**Subscriber** - Receiver of the message

**Pull** - Subscriber pulls messages when ready

Subscriber makes HTTPS requests to pubsub.googleapis.com

**Push** - Messages are sent to subscribers

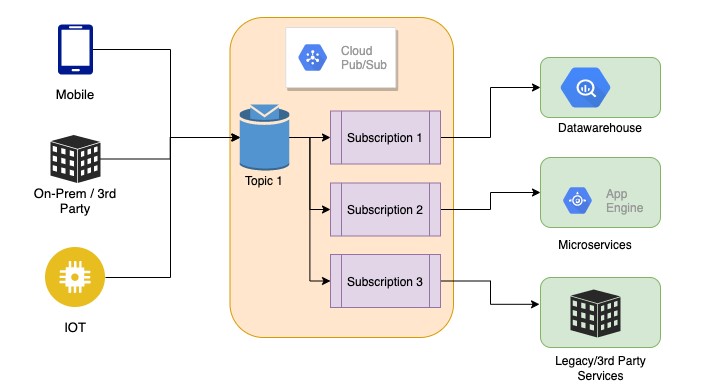
Subscribers provide a web hook endpoint at the time of registration

When a message is received on the topic, A HTTPS POST request is sent to the web hook endpoints

**Very Flexible** Publisher(s) and Subscriber(s) Relationships: One to

Many, Many to One, Many to Many

## Pub/Sub - Getting Ready with Topic and Subscriptions

Step 1 : Topic is created

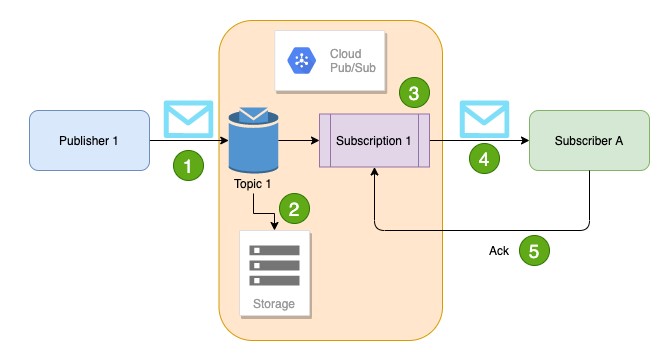
Step 2 : Subscription(s) are created

Subscribers register to the topic

Each Subscription represents discrete pull of messages from a topic:

Multiple clients pull same subscription => messages split between clients Multiple clients create a subscription each => each client will get every message

## Pub/Sub - Sending and Receiving a Message

Publisher sends a message to Topic

Message **individually** delivered to each and every subscription

Subscribers can receive message either by:

Push: Pub/Sub sends the message to Subscriber

Pull: Subscribers poll for messages

Subscribers send acknowledgement(s)

Message(s) are removed from subscriptions message queue

Pub/Sub ensures the message is retained **per**

**subscription** until it is acknowledged

## Managing Pub/Sub

Pub/Sub **pubsub**

*gcloud pubsub* ***topics create*** *my-topic*

*gcloud pubsub* ***subscriptions create*** *my-subscription --topic=my-topic*

**--enable-message-ordering** - ordered message delivery **--ack-deadline** - how long to wait for acknowledgment?

**--message-filter** - criteria to filter messages

*gcloud pubsub* ***subscriptions pull*** *my-subscription*

*--auto-ack --limit*

*gcloud pubsub* ***subscriptions ack*** *my-subscription --ack-ids=[ACK\_ID,…]*

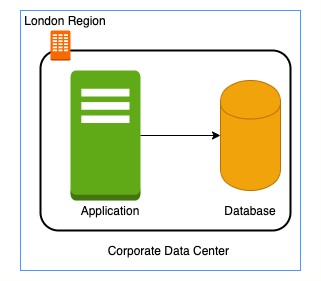
Topic: ***gcloud pubsub topics***

*gcloud pubsub topics* ***delete*** *my-topic gcloud pubsub topics* ***list***

*gcloud pubsub topics* ***list-subscriptions*** *my-topic*

# Networking

## Need for Google Cloud VPC

In a corporate network or an on-premises data center:

Can anyone on the internet **see the data exchange** between the application and the database?

**No**

Can anyone from internet **directly connect to your database**?

Typically **NO**.

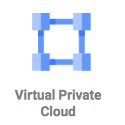
You need to connect to your corporate network and then access your applications or databases.

Corporate network provides a **secure internal network** protecting your resources, data and communication from external users

How do you do create **your own private network** in the cloud?

Enter **Virtual Private Cloud (VPC)**

## Google Cloud VPC (Virtual Private Cloud)

Your **own isolated network** in GCP cloud

Network traffic within a VPC is isolated (not visible) from all other Google Cloud VPCs

You **control all the traffic** coming in and going outside a VPC **(Best Practice)** Create all your GCP resources (compute, storage, databases etc) **within a VPC**  Secure resources from unauthorized access AND

Enable secure communication between your cloud resources

VPC is a **global resource** & contains subnets in one or more region

(REMEMBER) NOT tied to a region or a zone. VPC resources can be in any region or zone!

## Need for VPC Subnets

Different types of resources are created on cloud - databases, compute etc



Each type of resource has **its own access needs**

Load Balancers are accessible from internet (**public** resources)

Databases or VM instances should NOT be accessible from internet

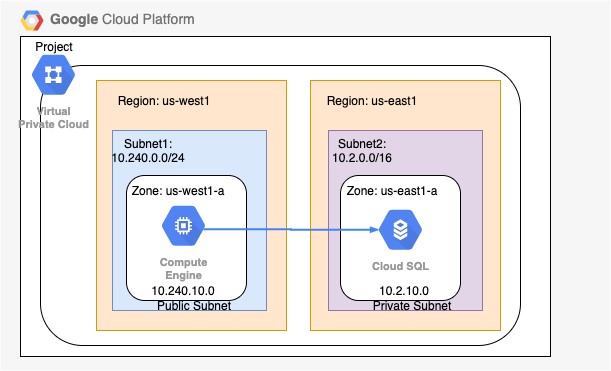
ONLY applications within your network (VPC) should be able to access them(**private** resources)

How do you **separate public resources from private resources** inside a VPC?

Create separate Subnets!

(Additional Reason) You want to distribute resources across multiple regions for high availability

## VPC Subnets

 (Solution) **Create different subnets** for public and private resources

Resources in a public subnet **CAN** be accessed from internet

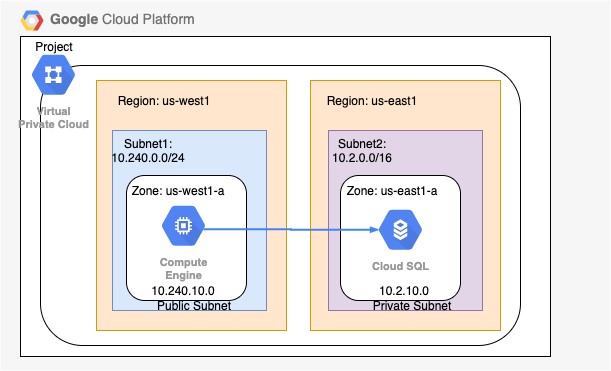
Resources in a private subnet **CANNOT** be accessed from internet

BUT resources in public subnet can talk to resources in private subnet

Each Subnet is created in a region

**Example** : VPC - demo-vpc => Subnets - region us-central1, europe-west1 or us-west1 or ..

## Creating VPCs and Subnets

By default, every project has a default VPC You can create YOUR own VPCs: **OPTION 1**: Auto mode VPC network:

Subnets are automatically created in each region

Default VPC created automatically in the project uses auto mode!

**OPTION 2**: Custom mode VPC network:

No subnets are automatically created

You have complete control over subnets and their IP ranges

Recommended for Production

Options when you create a subnet:

Enable **Private Google Access** - Allows VM's to connect to Google API's using private IP's

Enable **FlowLogs** - To troubleshoot any VPC related network issues

## CIDR (Classless Inter-Domain Routing) Blocks

Resources in a network use continuous IP addresses to make routing easy: Example: Resources inside a specific network can use IP addresses from 69.208.0.0 to

69.208.0.15

How do you express a **range of addresses** that resources in a network can have?

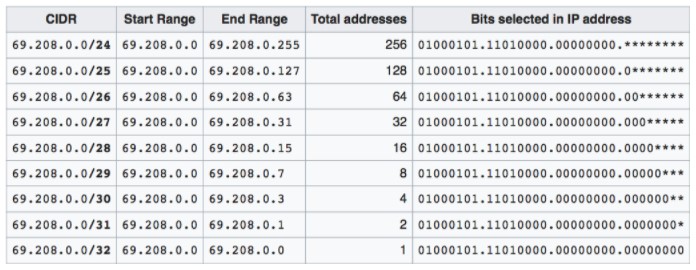
**CIDR block**

A **CIDR block** consists of a **starting IP address(69.208.0.0)** and a **range(/28)** Example: CIDR block 69.208.0.0/28 represents addresses from 69.208.0.0 to 69.208.0.15 - a total of 16 addresses

**Quick Tip**: 69.208.0.0/28 indicates that the first 28 bits (out of 32) are fixed.

Last 4 bits can change => 2 to the power 4 = 16 addresses

## CIDR Exercises



Exercise : How many addresses does **69.208.0.0/26** represent?

2 to the power (32-26 = 6) = 64 addresses from 69.208.0.0 to 69.208.0.63

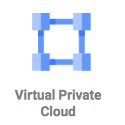
Exercise : How many addresses does **69.208.0.0/30** represent?

2 to the power (32-30 = 2) = 4 addresses from 69.208.0.0 to 69.208.0.3

Exercise : What is the difference between **0.0.0.0/0** and **0.0.0.0/32**?

0.0.0.0/0 represent all IP addresses. 0.0.0.0/32 represents just one IP address 0.0.0.0.

## Examples of Recommended CIDR Blocks - VPC Subnets

**Recommended CIDR Blocks**

Private IP addresses RFC 1918: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16

Shared address space RFC 6598: 100.64.0.0/10 IETF protocol assignments RFC 6890: 192.0.0.0/24

**Restricted Range Examples**

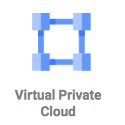
You CANNOT use these as CIDR for VPC Subnets

Private Google Access-specific virtual IP addresses: 199.36.153.4/30, 199.36.153.8/30

Current (local) network RFC 1122: 0.0.0.0/8 Local host RFC 1122: 127.0.0.0/8

(REMEMBER) You **CAN EXTEND** the CIDR Block Range of a Subnet (Secondary CIDR Block)

## Firewall Rules

 Configure Firewall Rules to control traffic going in or out of the network:

Stateful

Each firewall rule has priority (0-65535) assigned to it

0 has highest priority. 65535 has least priority

Default implied rule with lowest priority (65535)

Allow all egress

Deny all ingress

Default rules can't be deleted

You can override default rules by defining new rules with priority 0-65534

Default VPC has 4 additional rules with priority 65534

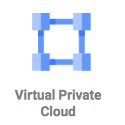
Allow incoming traffic from VM instances in same network (**default-allow-internal**)

Allow Incoming TCP traffic on port 22 (SSH) **default-allow-ssh**

Allow Incoming TCP traffic on port 3389 (RDP) **default-allow-rdp**

Allow Incoming ICMP from any source on the network **default-allow-icmp**

## Firewall Rules - Ingress and Egress Rules

**Ingress Rules**: Incoming traffic from outside to GCP targets

**Target (defines the destination)**: All instances or instances with TAG/SA

**Source (defines where the traffic is coming from)**: CIDR or All instances or instances with TAG/SA

**Egress Rules**: Outgoing traffic to destination from GCP targets

**Target (defines the source)**: All instances or instances with TAG/SA **Destination**: CIDR Block

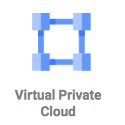
**Along with each rule**, you can also define:

**Priority** - Lower the number, higher the priority

**Action on match** - Allow or Deny traffic **Protocol** - ex. TCP or UDP or ICMP **Port** - Which port?

**Enforcement status** - Enable or Disable the rule

## Shared VPC

 Scenario: Your organization has multiple projects. You want resources in different projects to talk to each other?

How to allow resources in different projects to talk with internal IPs securely and efficiently?

Enter **Shared VPC**

Created at organization or shared folder level (Access Needed: Shared VPC Admin)

Allows VPC network to be shared between projects in same organization Shared VPC contains one host project and multiple service projects:

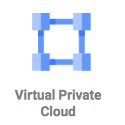
**Host Project** - Contains shared VPC network

**Service Projects** - Attached to host projects

Helps you achieve **separation of concerns**:

Network administrators responsible for Host projects and Resource users use Service Project

## VPC Peering

 Scenario: How to connect VPC networks across different organizations? Enter **VPC Peering**

Networks in same project, different projects and across projects in different organizations can be peered

All communication happens using internal IP addresses

Highly efficient because all communication happens **inside Google network**

Highly secure because **not accessible from Internet**

**No data transfer charges** for data transfer between services

(REMEMBER) Network administration is NOT changed:

Admin of one VPC do not get the role automatically in a peered network

# Hybrid Cloud

## Cloud VPN

Cloud VPN - Connect on-premise network to the GCP network

Implemented using **IPSec VPN Tunnel**

Traffic through internet (public)

Traffic encrypted using **Internet Key Exchange** protocol Two types of Cloud VPN solutions:

HA VPN (SLA of 99.99% service availability with two external IP addresses) Only dynamic routing (BGP) supported

Classic VPN (SLA of 99.9% service availability, a single external IP address)

Supports Static routing (policy-based, route-based) and dynamic routing using BGP

## Cloud Interconnect

 High speed physical connection between on-premise and VPC networks:

Highly available and high throughput

Two types of connections possible

Dedicated Interconnect - 10 Gbps or 100 Gpbs configurations

Partner Interconnect - 50 Mbps to 10 Gbps configurations

Data exchange happens through a private network:

Communicate using VPC network's internal IP addresses from on-premise network

Reduces egress costs

As public internet is NOT used

(Feature) Supported Google API's and services can be privately accessed from on-premise

Use only for high bandwidth needs:

For low bandwidth, Cloud VPN is recommended

## Direct Peering

Connect customer network to google network using network peering

Direct path from on-premises network to Google services

**Not a GCP Service**

Lower level network connection outside of GCP

NOT RECOMMENDED:

Use Cloud Interconnect and Cloud VPN

# Cloud Operations

## Cloud Monitoring

To operate cloud applications effectively, you should know:

Is my application healthy?

Are the users experiencing any issues?

Does my database has enough space?

Are my servers running in an optimum capacity?

**Cloud Monitoring** - Tools to monitor your infrastructure

Measures key aspects of services (Metrics)

Create visualizations (Graphs and Dashboard)

Configure Alerts (when metrics are NOT healthy)

Define Alerting Policies:

Condition

Notifications - Multiple channels

Documentation

## Cloud Monitoring - Workspace

 You can use Cloud Monitoring to monitor one or more GCP projects and one or more AWS accounts

How do you group all the information from multiple GCP projects or AWS Accounts?

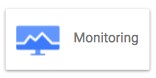
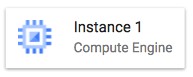
**Create a Workspace**

Workspaces are needed to organize monitoring information A workspace allows you to see monitoring information from multiple projects

Step I: Create workspace in a specific project (Host Project)

Step II: Add other GCP projects (or AWS accounts) to the workspace

## Cloud Monitoring - Virtual Machines



**Default metrics monitored** include:

CPU utilization

Some disk traffic metrics

Network traffic, and

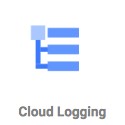
Uptime information

Install **Cloud Monitoring agent** on the VM to get more disk, CPU, network, and process metrics:

collectd-based daemon

Gathers metrics from VM and sends them to Cloud Monitoring

## Cloud Logging

Real time log management and analysis tool

Allows to store, search, analyze and alert on massive volume of data

Exabyte scale, fully managed service

No server provisioning, patching etc Ingest Log data from any source Key Features:

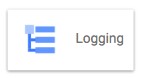
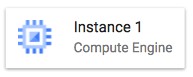
Logs Explorer - Search, sort & analyze using flexible queries

Logs Dashboard - Rich visualization

Logs Metrics - Capture metrics from logs (using queries/matching strings)

Logs Router - Route different log entries to different destinations

## Cloud Logging - Collection



Most **GCP Managed services automatically send logs** to Cloud Logging:

GKE

App Engine

Cloud Run

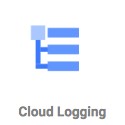
Ingest logs from GCE VMs:

Install **Logging Agent** (based on fluentd)

(Recommended) Run Logging Agent on all VM instances Ingest logs from on-premises:

(Recommended) Use the BindPlane tool from Blue Medora Use the Cloud Logging API

## Cloud Logging - Audit and Security Logs

 **Access Transparency Log**: Captures Actions performed by GCP team on your content (NOT supported by all services):

ONLY for organizations with Gold support level & above

**Cloud Audit Logs**: Answers who did what, when and where:

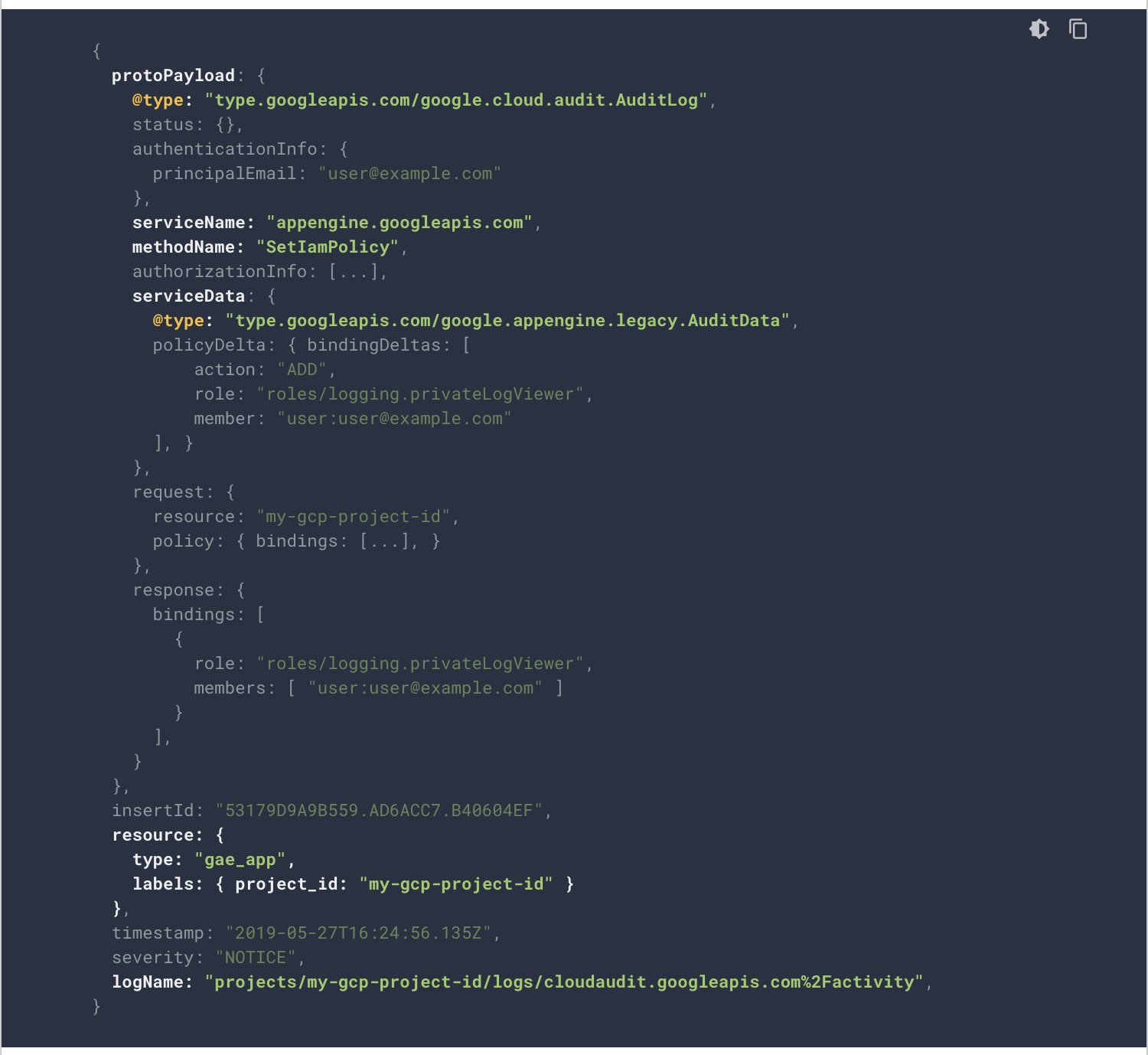
Admin activity Logs

Data Access Logs

System Event Audit Logs

Policy Denied Audit Logs

## Cloud Logging - Audit Logs

Which service? **protoPayload.serviceName** Which operation?

**protoPayload.methodName** What resource is audited? **resource.Type**

Who is making the call?

**authenticationInfo.principalEmail**

## Cloud Audit Logs

**Feature Admin Activity Logs Data Access System Event Logs Policy Denied**

**Logs Logs**



**Logs for** API calls or other actions Reading Google Cloud When user or

that modify the configuration of administrative actions service account configuration of resources is denied access resources



**Default** ✓ X ✓ ✓

**Enabled**



**VM Examples** VM Creation, Patching Listing resources On host maintenance, Security policy

resources, Change in IAM (vms, images Instance preemption, violation logs

permissions etc) Automatic restart



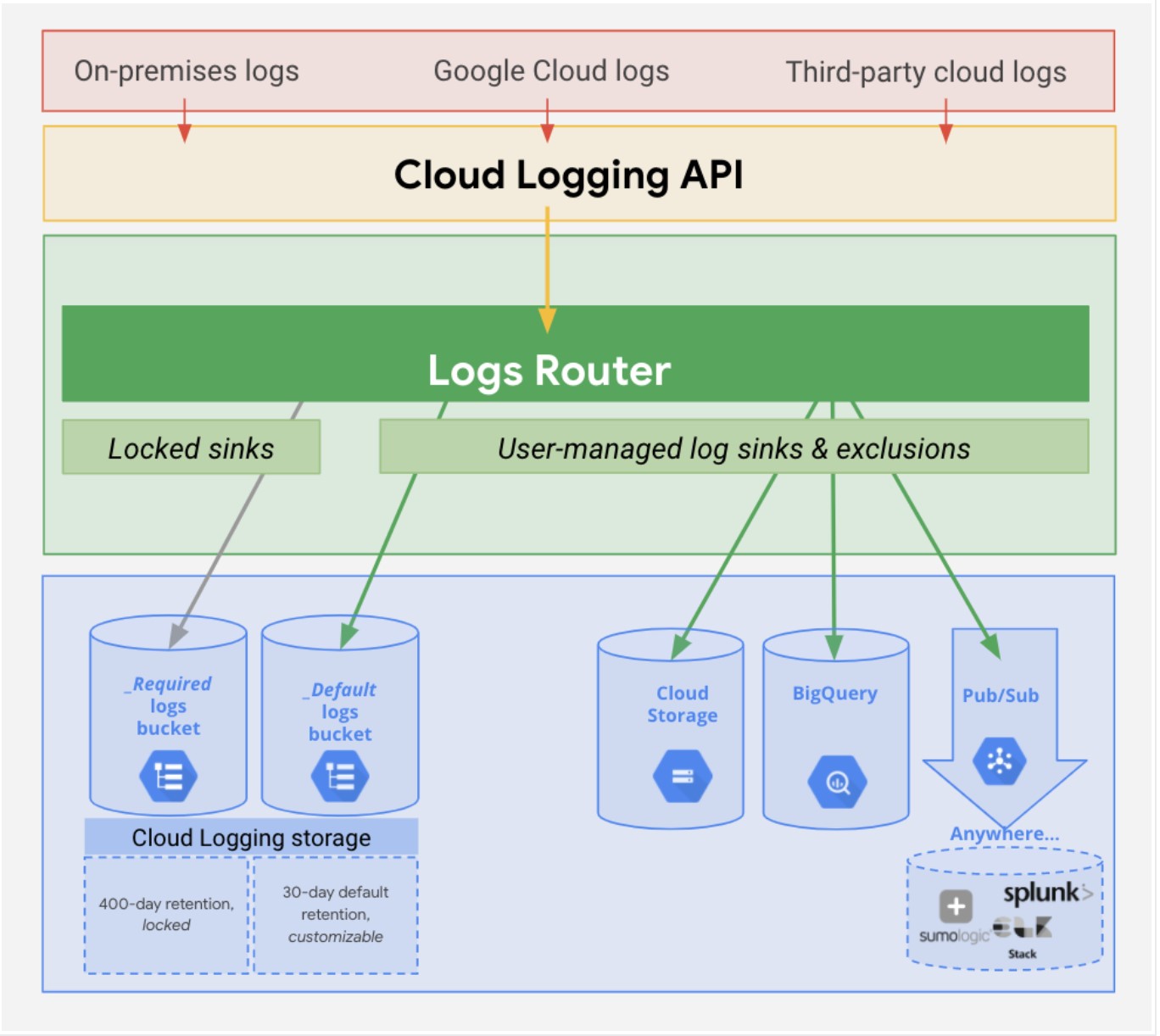
**Cloud Storage** Modify bucket or object Modify/Read bucket or object



**Recommended** Logging/Logs Viewer or Logging/Private Logging/Logs Viewer or Logging/Logs

**Roles** Project/Viewer Logs Viewer or Project/Viewer Viewer or

Project/Owner Project/Viewer

**Cloud Logging - Controlling & Routing** How do you manage your logs?

Logs from various sources reaches **Log Router**

Log Router checks against configured rules

What to ingest? what to discard?

Where to route?

Two types of Logs buckets:

**\_Required**: Holds Admin activity, System Events & Access Transparency Logs (retained for 400 days)

ZERO charge

You cannot delete the bucket

You cannot change retention period

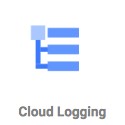
**\_Default**: All other logs (retained for 30 days)

You are billed based on Cloud Logging pricing

You cannot delete the bucket:

But you can disable the **\_Default** log sink route to disable ingestion! source: ([***https://cloud.google.com***)](https://cloud.google.com/)  You can edit retention settings (1 to 3650 days (10 years))

## Cloud Logging - Export

Logs are ideally stored in Cloud Logging for limited period For long term retention (Compliance, Audit) logs can be exported to:

Cloud Storage bucket (ex: bucket/syslog/2025/05/05)

Big Query dataset (ex: tables syslog\_20250505 > columns timestamp, log)

Cloud Pub/Sub topic (base64 encoded log entries)

How do you export logs?

Create **sinks** to these destinations using Log Router:

You can create **include** or **exclude** filters to limit the logs

## Cloud Logging - Export - Use Cases

Use Case 1: Troubleshoot using VM Logs:

Install Cloud logging agent in all VM's and send logs to Cloud Logging Search for logs in Cloud Logging

Use Case 2: Export VM logs to BigQuery for querying using SQL like queries:

Install Cloud logging agent in all VM's and send logs to Cloud Logging

Create a BigQuery dataset for storing the logs

Create an export sink in Cloud Logging with BigQuery dataset as sink destination

Use Case 3: You want to retain audit logs for external auditors at minimum cost

Create an export sink in Cloud Logging with Cloud Storage bucket as sink destination

Provide auditors with Storage Object Viewer role on the bucket

You can use Google Data Studio also (for visualization)

## Cloud Trace

Distributed tracing system for GCP: Collect latency data from:

Supported Google Cloud Services

Instrumented applications (using tracing libraries) using **Cloud Trace API** Find out:

How long does a service take to handle requests?

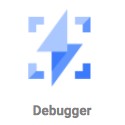
What is the average latency of requests?

How are we doing over time? (increasing/decreasing trend) Supported for:

**Compute Engine, GKE, App Engine (Flexible/Standard)** etc Trace client libraries available for:

**C#, Go, Java, Node.js, PHP, Python & Ruby**

## Cloud Debugger

 How to debug issues that are happening only in test or production environments?

**Cloud Debugger**: Capture state of a running application

Inspect the state of the application directly in the GCP environment

Take snapshots of variables and call stack

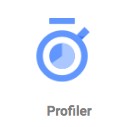
No need to add logging statements

No need to redeploy

Very lightweight => Very little impact to users

Can be used in any environment: Test, Acceptance, Production

## Cloud Profiler

How do you identify performance bottlenecks in production?

**Cloud Profiler** - Statistical, low-overhead profiler

Continuously gathers CPU and Memory usage from production systems

Connect profiling data with application source code

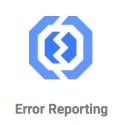
Easily identify performance bottlenecks

Two major components:

Profiling agent (collects profiling information)

Profiler interface (visualization)

## Error Reporting

How do you identify production problems in real time?

Real-time exception monitoring:

Aggregates and displays errors reported from cloud services (using stack traces)

**Centralized Error Management console**:

Identify & manage top errors or recent errors

Use **Firebase Crash Reporting** for errors from Android & iOS client applications

Supported for Go, Java, .NET, Node.js, PHP, Python, and Ruby Errors can be reported by:

Sending them to Cloud Logging OR

By calling Error Reporting API

Error Reporting can be accessed from desktop

Also available in the Cloud Console mobile app for iOS and Android

## Cloud Operations Scenarios

**Scenario Solution**



**You would like to record all operations/requests on all objects in a** Turn on data access audit logging for

**bucket (for auditing)** the bucket



**You want to trace a request across multiple microservices** Cloud Trace



**You want to identify prominent exceptions (or errors) for a specific** Error Reporting **microservice**



**You want to debug a problem in production by executing step by** Cloud Debugger **step**

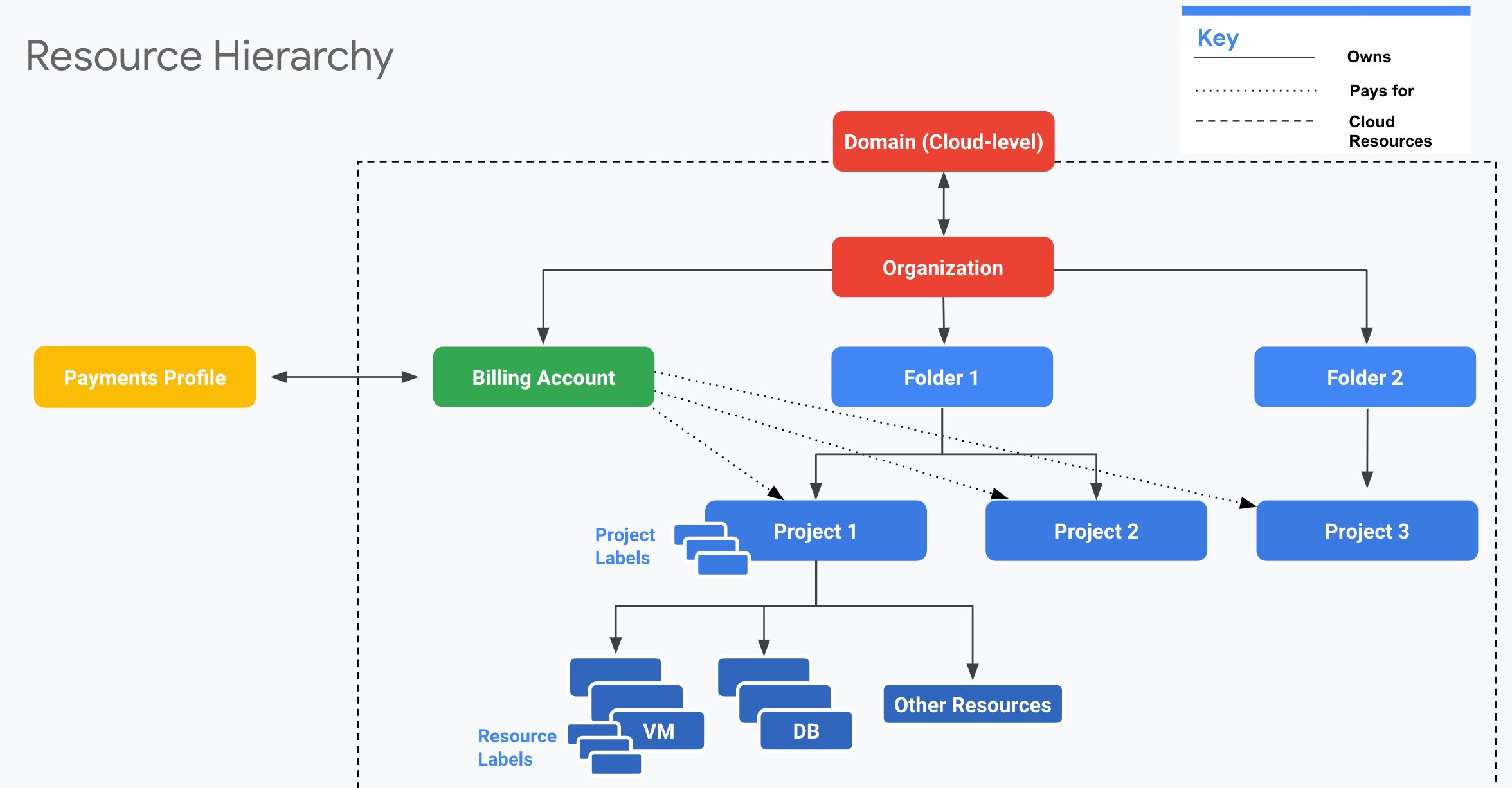


**You want to look at the logs for a specific request** Cloud Logging



# Organizing GCP Resources

## Resource Hierarchy in GCP

**Well defined hierarchy**: Organization > Folder > Project > Resources

**Resources** are created in projects

A **Folder** can contain multiple projects

**Organization** can contain multiple Folders

source: ([***https://cloud.google.com***)](https://cloud.google.com/)

## Resource Hierarchy - Recommendations for Enterprises

Create **separate projects for different environments**: Complete isolation between test and production environments Create **separate folders for each department**:

Isolate production applications of one department from another

We can create a shared folder for shared resources

**One project per application per environment**:

Let's consider two apps: "A1" and "A2"

Let's assume we need two environments: "DEV" and "PROD"

In the ideal world you will create four projects: A1-DEV, A1-PROD, A2-DEV, A2-PROD:

Isolates environments from each other

DEV changes will NOT break PROD

Grant all developers complete access (create, delete, deploy) to DEV Projects Provide production access to operations teams only!

## Billing Accounts

**Billing Account** is mandatory for creating resources in a project:

Billing Account contains the payment details

Every Project with active resources should be associated with a Billing Account

Billing Account can be associated with one or more projects

You can have multiple billing accounts in an Organization

(RECOMMENDATION) Create Billing Accounts representing your organization structure:

A startup can have just one Billing account

A large enterprise can have a separate billing account for each department

Two Types:

**Self Serve** : Billed directly to Credit Card or Bank Account

**Invoiced** : Generate invoices (Used by large enterprises)

## Managing Billing - Budget, Alerts and Exports

Setup a **Cloud Billing Budget** to avoid surprises:

(RECOMMENDED) Configure **Alerts**

Default alert thresholds set at 50%, 90% & 100%

Send alerts to Pub Sub (Optional)

Billing admins and Billing Account users are alerted by e-mail

Billing data can be **exported (on a schedule)** to:

**Big Query** (if you want to query information or visualize it)

**Cloud Storage** (for history/archiving)

## IAM Best Practices

**Principle of Least Privilege** - Give least possible privilege needed for a role!

Basic Roles are NOT recommended

Prefer predefined roles when possible

Use Service Accounts with minimum privileges

Use different Service Accounts for different apps/purposes

**Separation of Duties** - Involve atleast 2 people in sensitive tasks:

Example: Have separate deployer and traffic migrator roles

AppEngine provides App Engine Deployer and App Engine Service Admin roles

App Engine Deployer can deploy new version but cannot shi traffic App Engine Service Admin can shi traffic but cannot deploy new version!

**Constant Monitoring**: Review Cloud Audit Logs to audit changes to IAM policies and access to Service Account keys

Archive Cloud Audit Logs in Cloud Storage buckets for long term retention

Use Groups when possible

Makes it easy to manage users and permissions

## User Identity Management in Google Cloud

Email used to create free trial account => **"Super Admin"**

Access to everything in your GCP organization, folders and projects

Manage access to other users **using their Gmail accounts**

However, this is **NOT recommended** for enterprises **Option 1**: Your Enterprise is using **Google Workspace**

Use Google Workspace to manage users (groups etc) Link Google Cloud Organization with Google Workspace

**Option 2**: Your Enterprise uses an Identity Provider of its own **Federate** Google Cloud with your Identity Provider

## Corporate Directory Federation

 **Federate** Cloud Identity or Google Workspace **with your external identity provider (IdP)** such as Active Directory or Azure Active Directory.

**Enable Single Sign On**:

1: Users are redirected to an external IdP to authenticate

2: When users are authenticated, SAML assertion is sent to Google Sign-In

**Examples**:

Federate Active Directory with Cloud Identity by using Google Cloud

Directory Sync (GCDS) and Active Directory Federation Services (AD FS)

Federating Azure AD with Cloud Identity

## IAM Members/Identities

**Google Account** - Represents a person (an email address)

**Service account** - Represents an application account (Not person)

**Google group** - Collection - Google & Service Accounts

Has an unique email address

Helps to apply access policy to a group

**Google Workspace domain**: Google Workspace (formerly G Suite) provides collaboration services for enterprises:

Tools like Gmail, Calendar, Meet, Chat, Drive, Docs etc are included

If your enterprise is using Google Workspace, you can manage permissions using your Google Workspace domain

**Cloud Identity domain** - Cloud Identity is an Identity as a Service (IDaaS) solution that centrally manages users and groups.

You can use IAM to manage access to resources for each Cloud Identity account

## IAM Members/Identities - Use Cases

**Scenario Solution**



**All members in your team have G Suite accounts. You** Create a Group with all your operations team.

**are creating a new production project and would want** Provide access to production project to the Group. **to provide access to your operations team**



**All members in your team have G Suite accounts. You** Assign the necessary role directly to G Suite email **are setting up a new project. You want to provide a** address of your team member **one time quick access to a team member.** If it is not a one time quick access, the recommended approach would be to create a Group



**You want to provide an external auditor access to view** Give them roles/viewer role (Generally basic roles

**all resources in your project BUT he should NOT be** are NOT recommended BUT it is the simplest way to **able to make any changes** provide view only access to all resources!)



**Your application deployed on a GCE VM (Project A)** In Project B, assign the right role to GCE VM service **needs to access cloud storage bucket from a different** account from Project A **project (Project B)**



## Organization Policy Service

 How to enable **centralized constraints** on all resources created in an Organization?

Configure **Organization Policy**

Example: Disable creation of Service Accounts

Example: Allow/Deny creation of resources in specific regions

Needs a Role - Organization Policy Administrator

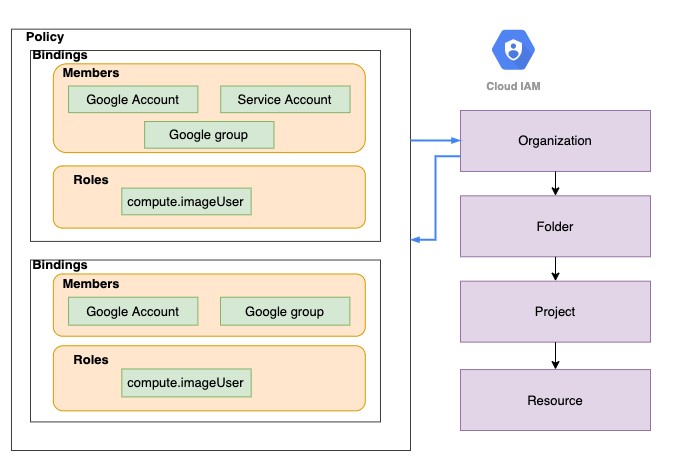
(Remember) **IAM** focuses on **Who**

Who can take specific actions on resources?

(Remember) Organization Policy focuses on **What**

What can be done on specific resources?

## Resource Hierarchy & IAM Policy

 IAM Policy can be set at any level of the hierarchy

Resources inherit the policies of **All parents**

The effective policy for a resource is the union of the policy on that resource and its parents Policy inheritance is transitive:

For example: Organization policies are applied at resource level

You can't restrict policy at lower level if permission is given at an higher level

## Organization, Billing and Project Roles

**Organization Administrator**

Define Resource Hierarchy

Define Access Management Policies

Manage other users and roles

**Billing Account Creator** - Create Billing Accounts

**Billing Account Administrator** - Manage Billing Accounts (payment instruments, billing exports, link and unlink projects, manage roles on billing account)

CANNOT create a Billing Account

**Billing Account User** - Associate Projects with Billing Accounts

Typically used in combination with **Project Creator**

These two roles allow user to create new project and link it with billing account

**Billing Account Viewer** - See all Billing Account details

## Billing Roles - Quick Review

**Roles Description Use Case**



**Billing Account Creator** Permissions to create new billing accounts Finance Team



**Billing Account Administrator** Manages billing account but can't create them Finance Team



**Billing Account User** Assigns projects to billing accounts Project Owner



**Billing Account Viewer** View only access to billing account Auditor



## Organization, Billing and Project Roles - Scenarios

**Scenario 1**: I'm creating a project and I want to associate an existing billing account with the project

Roles needed : Project Creator and Billing Account User (link project to billing account)

**Scenario 2**: I'm a billing auditor

Roles needed : Billing Account Viewer role

## Compute Engine Roles

Compute Engine IAM Roles

**Compute Engine Admin** - Complete control of compute - Instances, Images, Load Balancers, Network, Firewalls etc...

**Compute Instance Admin** - Create, modify, and delete virtual machine instances and disks

**Compute Engine Network Admin** - Complete access to networking resources (routes, networks, health checks, VPN, Gateways etc) and READ ONLY access to (firewall rules and SSL certificates)

**Compute Engine Security Admin** - Complete access to firewall rules and SSL certificates

**Compute Storage Admin** - Complete access to disks, images, snapshots

**Compute Engine Viewer** - Read ONLY access to everything in compute

**Compute OS Admin Login** - Log in to a Compute Engine instance as an administrator user

**Compute OS Login** - Log in to a Compute Engine instance as a standard user

## App Engine Roles

App Engine Roles (CRUD - Create, Read (get/list), Update, Delete)

**App Engine Creator** - applications(CD) (Responsible for creating an application)

**App Engine Admin** - applications(RU), services/instances/versions(CRUD), operations

**App Engine Viewer** - applications/services/instances/versions(R), operations

**App Engine Code Viewer** - appengine.versions.getFileContents (ONLY role that can view code)

**App Engine Deployer** - versions(CRD), applications/services/versions(R)

Deploy a new version of an app (if you also grant the Service Account User role)

**App Engine Service Admin** - versions(RUD), applications(R), services/instances(CRUD), operations: Split or migrate traffic, Start and stop a version

App Engine Roles DO NOT allow you to

View and download application logs

View Monitoring charts in the Cloud Console

Enable and Disable billing

Access configuration or data stored in other services

## Compute Engine and App Engine Roles - Few Scenarios

**Scenario 1**: What is the difference between Compute Engine Admin vs Compute Instance Admin?

Compute Instance Admin can do everything with instances and disks ONLY. Compute Engine Admin is admin for everything in compute - instances, disks, images, network, firewalls etc.

**Scenario 2**: What is a secure way of setting up application deployment?

**Application Deployer** - Roles: App Engine Deployer + Service Account User

Limited to deploying new versions and deleting old versions that are not serving traffic Will NOT be able to configure traffic

**Operations** - Role: **App Engine Service Admin**

CANNOT deploy a new version of an app

Change traffic between versions

## Google Kubernetes Engine (GKE) IAM Roles

**Kubernetes Engine Admin (roles/container.admin)** - Complete Access to Clusters and Kubernetes API objects

**Kubernetes Engine Cluster Admin** - Provides access to management of clusters (Cannot access Kubernetes API objects - Deployments, Pods etc)  **Kubernetes Engine Developer** - Manage Kubernetes API objects (and read cluster info)

**Kubernetes Engine Viewer** - get/list cluster and kubernetes api objects

## Cloud Storage - Roles

**Storage Admin** - storage.buckets.\*, storage.objects.\*

**Storage Object Admin** - storage.objects.\* (DOES NOT HAVE storage.buckets.\*)

**Storage Object Creator** - storage.objects.create

**Storage Object Viewer** - storage.objects.get, storage.objects.list

(REMEMBER) Container Registry stores container images in Cloud Storage buckets

Control access to images in Container Registry using Cloud Storage permissions!

(REMEMBER) **Storage Admin** vs **Storage Object Admin**

Storage Admin can create buckets and play with objects

Storage Object Admin CANNOT create buckets but can play with objects in a bucket!

## Cloud BigQuery Roles

Cloud BigQuery IAM Roles

**BigQuery Admin** - bigquery.\*

**BigQuery Data Owner** - bigquery.datasets.\*, bigquery.models.\*, bigquery.routines.\*, bigquery.tables.\* (**Does NOT have access to Jobs!**)

**BigQuery Data Editor** - bigquery.tables.(create/delete/export/get/getData/getIamPolicy/ list/update/updateData/updateTag), bigquery.models.\*, bigquery.routines.\*, bigquery.datasets.(create/get/getIamPolicy/updateTag)

**BigQuery Data Viewer** - get/list bigquery.(datasets/models/routines/tables)

**BigQuery Job User** - bigquery.jobs.create

**BigQuery User** - BigQuery Data Viewer + get/list (jobs, capacityCommitments, reservations etc)

To see data, you need either BigQuery User or BigQuery Data Viewer roles

You CANNOT see data with BigQuery Job User roles

BigQuery Data Owner or Data Viewer roles do NOT have access to jobs!

## Logging IAM Roles and Service Account Roles

Logging and Audit Logging:

**roles/logging.viewer (Logs Viewer)**: Read all Logs except Access Transparency logs and Data Access audit logs.

**roles/logging.privateLogViewer (Private Logs Viewer)**: Logs Viewer + Read Access Transparency logs and Data Access audit logs

**roles/logging.admin (Logging Admin)**: All permissions related to Logging Service Accounts:

**roles/iam.serviceAccountAdmin**: Create and manage service accounts

**roles/iam.serviceAccountUser**: Run operations as the service account roles/iam.serviceAccountUser => create and manage instances that use a service account. This needs to be added to Admin roles if you want them to attach service accounts with instances.

**roles/iam.serviceAccountTokenCreator** - Impersonate service accounts (create OAuth2 access tokens, sign blobs or JWTs, etc).  **roles/iam.serviceAccountKeyAdmin** - Create and manage (and rotate) service account keys.

## Other Important IAM Roles

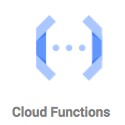
**roles/iam.securityAdmin** - Get and set any IAM policy **roles/iam.securityReviewer** - List all resources & IAM policies

**roles/iam.organizationRoleAdmin** - Administer all custom roles **in the organization and the projects below it**

**roles/iam.organizationRoleViewer** - Read all custom roles **in the organization and the projects below it roles/iam.roleAdmin** - Provides access to all custom roles **in the project roles/iam.roleViewer** - Provides read access to all custom roles **in the project roles/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

## SSHing into Linux VMs - Options

**Compute Engine Linux VMs** uses **key-based SSH** authentication **Two Options**:

**Metadata managed**: Manually create and configure individual SSH keys **OS Login**: Manage SSH access without managing **individual** SSH keys!

Recommended for managing multiple users across instances or projects

Your Linux user account is linked to your Google identity To enable: Set enable-oslogin to true in metadata *gcloud compute* ***project-info/instances add-metadata --metadata enable-oslogin=TRUE***

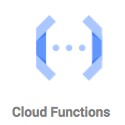
(Advantage) Ability to import existing Linux accounts from on premises AD and LDAP Users need to have roles : roles/compute.osLogin or roles/compute.osAdminLogin

(Windows) **Windows** instances use **password**

authentication(username and password)

Generate using console or gcloud (gcloud compute reset-windows-password)

## SSHing into Linux VMs - Details

**Option 1**: Console - SSH Button

Ephemeral SSH key pair is created by Compute Engine

**Option 2**: Gcloud - *gcloud compute ssh*

A username and persistent SSH key pair are created by Compute Engine SSH key pair reused for future interactions

**Option 3**: Use customized SSH keys

(Metadata managed): Upload the public key to project metadata OR

(OS Login): Upload your public SSH key to your OS Login profile  *gcloud compute os-login ssh-keys add* OR

Use OS Login API : POST [***https://oslogin.googleapis.com/v1/users/ACCOUNT\_EMAIL:importSshPublicKey***](https://oslogin.googleapis.com/v1/users/ACCOUNT_EMAIL:importSshPublicKey)

You can disable Project wide SSH keys on a specific compute instance *gcloud compute* ***instances add-metadata [INSTANCE\_NAME] --metadata blockproject-ssh-keys=TRUE***

## IAM - Scenarios

**Scenario Description**



**You want to give permanent access to a sub set** Use ACLs **of objects in a Cloud Storage bucket**



**You want to give permanent access to the entire** Use IAM **bucket in a Cloud Storage bucket**



**You want to provide time limited access to a** Create a Signed URL **specific object in a Cloud Storage bucket**



**You want to give access to a set of resources to** Create a Group with your development team as member. **your development team** Bind the right Predefined Roles to your Group.



**Which Role? Upload objects to Cloud Storage** Storage Object Creator



**Which Role? Manage Kubernetes API objects** Kubernetes Engine Developer



**Which Role? Manage service accounts** Service Account Admin



**Which Role? View Data in BigQuery** BigQuery Data Viewer



**Other Google Cloud Platform Services**

## Pricing Calculator

**Estimating** the cost of a Google Cloud solution **is NOT easy** You would need to take a **number of factors** into account How do you estimate the cost of your GCP solution?

Use **Google Cloud Pricing Calculator** Estimates for **40+ Services**:

Compute Engine

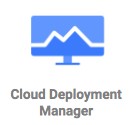
Google Kubernetes Engine

Cloud Run

App Engine Cloud Storage etc

(REMEMBER) **These are Estimates! (NOT binding on GCP)**

## Google Cloud Deployment Manager - Introduction

Lets consider an example:

I would want to create a new VPC and a subnet

I want to provision a Load balancer, Instance groups with 5 Compute

Engine instances and an Cloud SQL database in the subnet

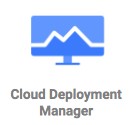
I would want to setup the right Firewall

AND I would want to create 4 environments

Dev, QA, Stage and Production!

Deployment Manager can help you do all these with a simple (actually NOT so simple) script!

## Google Cloud Deployment Manager - Advantages

 Automate deployment and modification of Google Cloud resources in a controlled, predictable way Deploy in multiple environments easily!

Avoid configuration dri

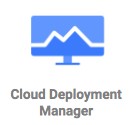
Avoid mistakes with manual configuration

Think of it as version control for your environments

**Important Note** - Always modify the resources created by

Deployment Manager using Deployment Manager

## Google Cloud Deployment Manager

All configuration is defined in a simple text file - YAML I want a VPC, a subnet, a database and ...

Deployment Manager understands dependencies

Creates VPCs first, then subnets and then the database

(Default) Automatic rollbacks on errors (Easier to retry) If creation of database fails, it would automatic delete the subnet and VPC Version control your configuration file and make changes to it over time

Free to use - Pay only for the resources provisioned Get an automated estimate for your configuration

## Cloud Deployment Manager - Example

|  |
| --- |
| * type: compute.v1.instance name: my-first-vm properties: zone: us-central1-a machineType: <<MACHINE\_TYPE>> disks: * deviceName: boot type: PERSISTENT boot: true autoDelete: true initializeParams:   sourceImage: <<SOURCE\_IMAGE>> networkInterfaces: - network: <<NETWORK>>  # Give instance a public IP Address accessConfigs: - name: External NAT type: ONE\_TO\_ONE\_NAT |

## Cloud Deployment Manager - Terminology

**Configuration** file: YAML file with resource definitions for a single deployment

**Templates**: **Reusable resource definitions** that can be used in multiple configuration files Can be defined using:

Python (preferred) OR

JinJa2 (recommended only for very simple scripts)

**Deployment**: Collection of resources that are deployed and managed together

**Manifests**: Read-only object containing original deployment configuration

(including imported templates)

Generated by Deployment Manager

Includes fully-expanded resource list

Helpful for troubleshooting

## Cloud Marketplace (Cloud Launcher)

Installing custom so ware might involve setting up multiple resources:

Example: Installing WordPress needs set up of compute engine and a relational database How do you simplify the set up of custom so ware solutions like Wordpress or even more complex things like SAP HANA suite on GCP?

**Cloud Marketplace**: Central repo of easily deployable apps & datasets

Similar to **App Store/Play Store** for mobile applications

You can search and install a complete stack

Commercial solutions - SAP HANA etc

Open Source Packages - LAMP, WordPress, Cassandra, Jenkins etc

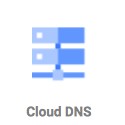
OS Licenses: BYOL, Free, Paid

Categories: Datasets/Developer tools/OS etc

When selecting a solution, you can see:

Components - So ware, infrastructure needed etc Approximate price

## Cloud DNS

 What would be the **steps in setting up a website** with a domain name (for example, in28minutes.com)?

**Step I** : Buy the domain name in28minutes.com (Domain Registrar)

**Step II** : Setup your website content (Website Hosting)

**Step III** : Route requests to in28minutes.com to the my website host server (DNS)

**Cloud DNS = Global Domain Name System** (Step III)

Setup your DNS routing for your website (in28minutes.com)

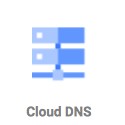
Route api.in28minutes.com to the IP address of api server

Route static.in28minutes.com to the IP address of http server

Route email (ranga@in28minutes.com) to the mail server(mail.in28minutes.com)

Public and private managed DNS zones (container for records)

## Cloud DNS - CLI

*gcloud* ***dns managed-zones*** *create ZONE\_NAME*

--description (REQUIRED - Short description for the managed-zone)

--dns-name (REQUIRED - DNS name suffix that will be managed with the created zone)

--visibility (private/**public**)

--networks (List of networks that the zone should be visible in if the zone visibility is [private])

Three Steps to add records to a managed zone:

Start Transaction for Zone

gcloud dns **record-sets transaction start** --zone

Make Changes gcloud dns **record-sets transaction add** --name=REC\_NAME --ttl --type A/CNAME -zone=ZONE\_NAME

End Transaction for Zone

gcloud dns **record-sets transaction execute** --zone

## Cloud Dataflow

**Cloud Dataflow** is a difficult service to describe: Let's look at a **few example pipelines** you can build:

Pub/Sub > Dataflow > BigQuery (Streaming)

Pub/Sub > Dataflow > Cloud Storage (Streaming - files)

Cloud Storage > Dataflow > Bigtable/CloudSpanner/Datastore/BigQuery (Batch - Load data into databases)

Bulk compress files in Cloud Storage (Batch)

Convert file formats between Avro, Parquet & csv (Batch)

**Streaming and Batch Usecases**

Realtime Fraud Detection, Sensor Data Processing, Log Data Processing, Batch Processing (Load data, convert formats etc)

Use **pre-built** templates

Based on **Apache Beam** (supports Java, Python, Go ...)

Serverless (and Autoscaling)

## Cloud Dataproc

Managed **Spark and Hadoop** service:

Variety of jobs are supported:

Spark, PySpark, SparkR, Hive, SparkSQL, Pig, Hadoop

Perform complex batch processing **Multiple Cluster Modes**:

Single Node / Standard/ High Availability (3 masters) Use regular/preemptible VMs

Use case: Move your Hadoop and Spark clusters to the cloud

Perform your machine learning and AI development using open source frameworks

(REMEMBER) Cloud Dataproc is a data analysis platform

You can export cluster configuration but NOT data

(ALTERNATIVE) BigQuery - When you run SQL queries on Petabytes Go for Cloud Dataproc when you need more than queries (Example: Complex batch processing Machine Learning and AI workloads)

# Get Ready

## Certification Resources

**Title Link**



**Home Page** [***https://cloud.google.com/certification/cloud-engineer***](https://cloud.google.com/certification/cloud-engineer)



**Exam Guide** [***https://cloud.google.com/certification/guides/cloud-engineer***](https://cloud.google.com/certification/guides/cloud-engineer)



**Sample Questions** [***https://cloud.google.com/certification/sample-questions/cloud-engineer***](https://cloud.google.com/certification/sample-questions/cloud-engineer)



**Registering For Exam** [***https://support.google.com/cloud-certification/#topic=9433215***](https://support.google.com/cloud-certification/#topic=9433215)



## Certification Exam

**50 questions** and **Two hours No penalty** for wrong answers **Questions**:

Type 1 : Multiple Choice - 4 options and 1 right answer Type 2 : Multiple Select - 5 options and 2 right answers

Result immediately shown a er exam completion

Email (a couple of days later) **My Recommendations**:

Read the **entire question**

Identify and write down the **key parts of the question**

Read **all answers** at least once

If you do NOT know the answer, **eliminate wrong answers** first

**Flag questions** for future consideration and review them before final submission

**You are all set!**

**Let's clap for you!**

You have a lot of patience! **Congratulations**

You have put your best foot forward to be an Google Cloud Certified Associate

Cloud Engineer

Make sure you prepare well Good Luck!

**Do Not Forget!**

Recommend the course to your friends!

**Do not forget to review**!

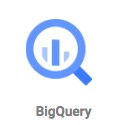
**Your Success = My Success**

Share your success story with me on LinkedIn (Ranga Karanam)

Share your success story and lessons learnt in Q&A with other learners!

**What next?**

Go Deeper into AWS! Three things I would recommend



Serverless (Lambda, API Gateway DynamoDB)

Elastic Beanstalk

ECS

Learn other Cloud Platforms:

Gartner predicts a multi cloud world soon Get certified on AWS, Azure and Google Cloud

Learn DevOps (Containers and Container Orchestration)

Learn Full Stack Development