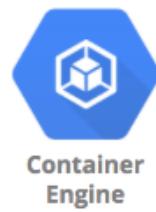
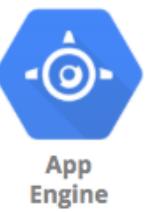
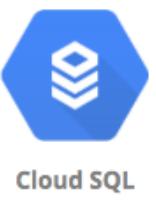
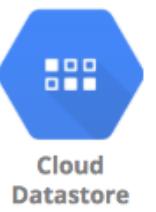
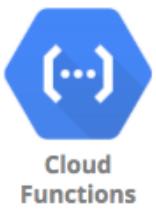


Google Certified Professional Cloud Developer

Getting Started

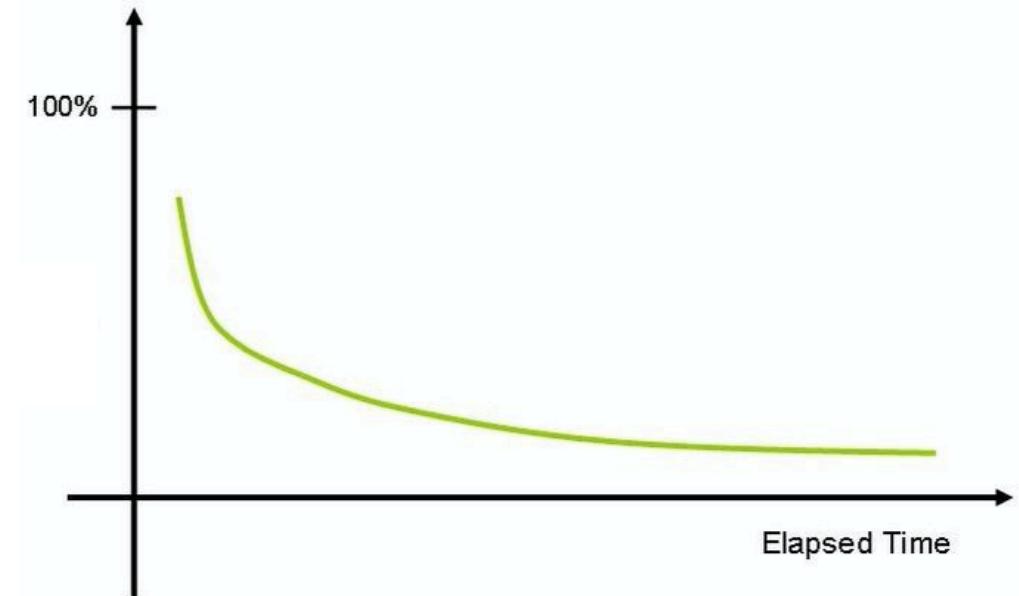
In 28
Minutes



- GCP has 200+ services. This exam expects knowledge of 40+ Services.
- Exam *tests* your **decision making abilities**:
 - Which service do you choose in which situation?
- Exam *tests* your ability to design, build, test and deploy cloud-native applications in Google Cloud!
- This course is **designed** to help you *make these tough choices and build solutions* for GCP
- **Our Goal** : Enable you to develop amazing solutions in GCP

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



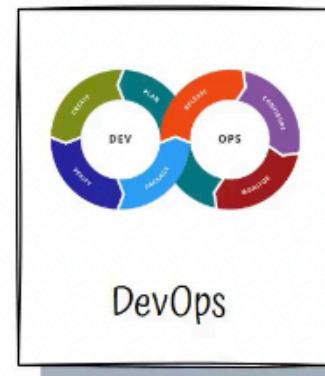
Our Approach

- 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!



FASTEST ROADMAPS

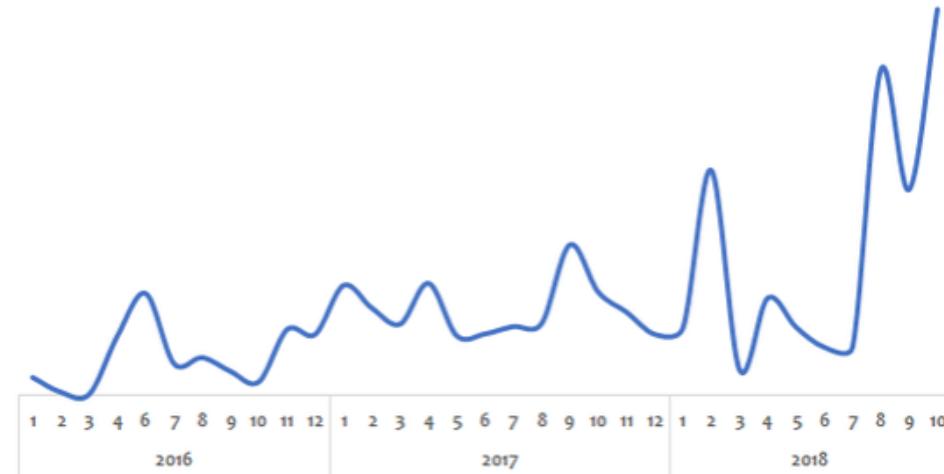
in28minutes.com



GCP - Getting started

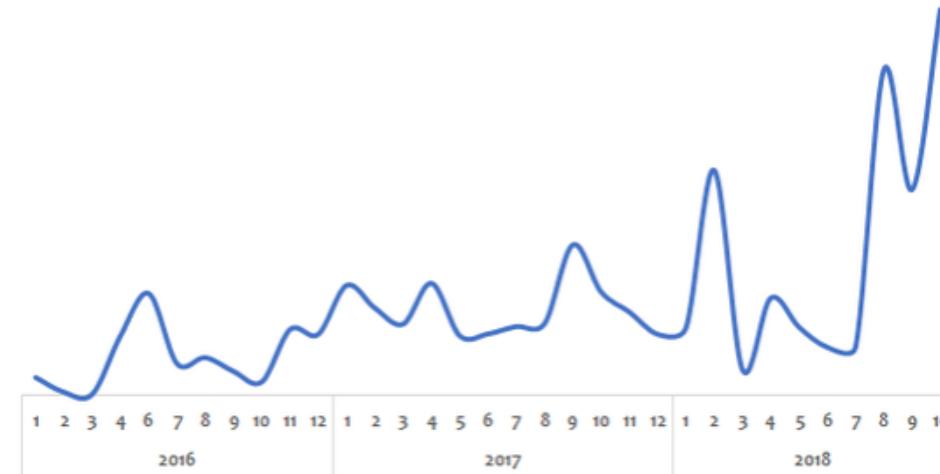
Before the Cloud - Example 1 - Online Shopping App

In 28
Minutes



- 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?

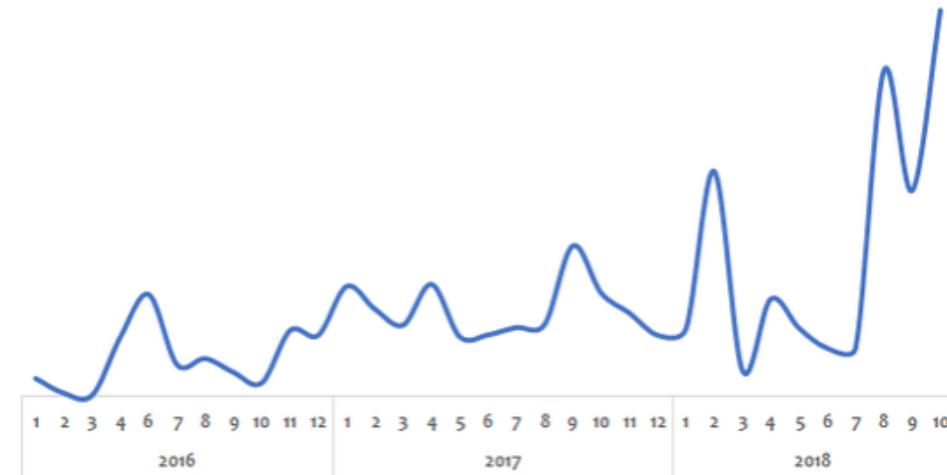
Before the Cloud - Example 2 - Startup



- 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

In 28
Minutes



- 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



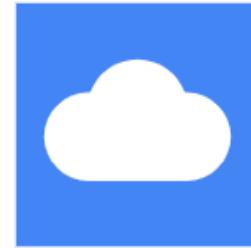
Cloud - Advantages

- 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.



Google Cloud

Best path to learn GCP!



Compute Engine



Cloud Functions



Cloud Datastore



Cloud SQL



App Engine



Container Engine

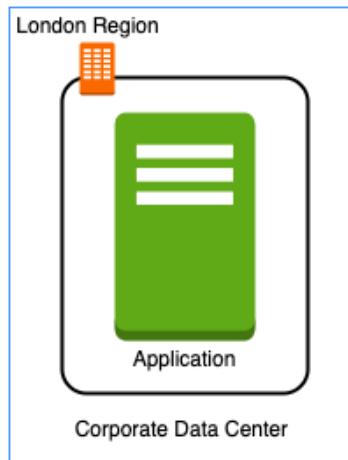
- Cloud applications make use of multiple GCP services
- There is no **single path** to learn these services independently
- **HOWEVER**, we've worked out a simple path!

Setting up GCP Account

- Create GCP Account

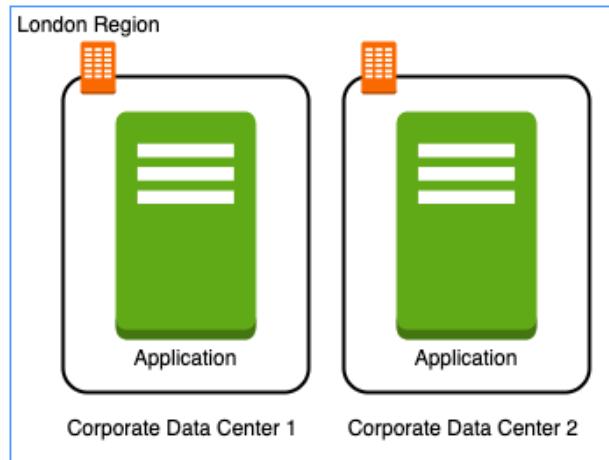
Regions and Zones

Regions and Zones



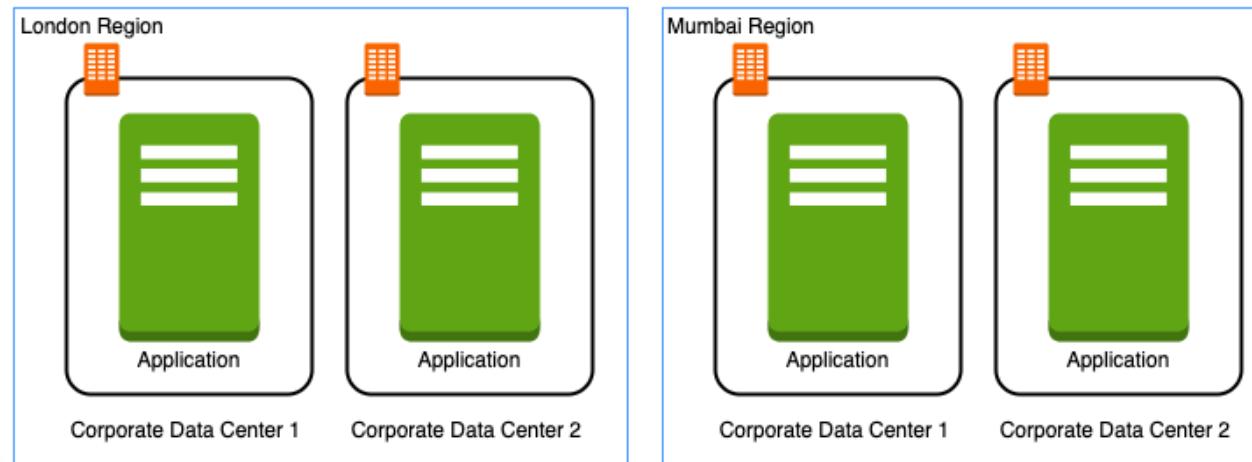
- Imagine that your application is deployed in a data center in London
- What would be the challenges?
 - Challenge 1 : Slow access for users from other parts of the world (**high latency**)
 - Challenge 2 : What if the data center crashes?
 - Your application goes down (**low availability**)

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

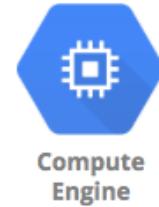
Region Code	Region	Zones	Zones List
us-west1	The Dalles, Oregon, North America	3	us-west1-a us-west1-b us-west1-c
europe-north1	Hamina, Finland, Europe	3	europe-north1-a, europe-north1-b europe-north1-c
asia-south1	Mumbai, India APAC	3	asia-south1-a, asia-south1-b asia-south1-c

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

In 28
Minutes



- 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

Machine name	vCPUs ¹	Memory (GB)	Max number of persistent disks (PDs) ²	Max total PD size (TB)	Local SSD	Maximum egress bandwidth (Gbps) ³
e2-standard-2	2	8	128	257	No	4
e2-standard-4	4	16	128	257	No	8
e2-standard-8	8	32	128	257	No	16
e2-standard-16	16	64	128	257	No	16
e2-standard-32	32	128	128	257	No	16

- 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 software 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



Compute
Engine

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 as well!**
 - 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



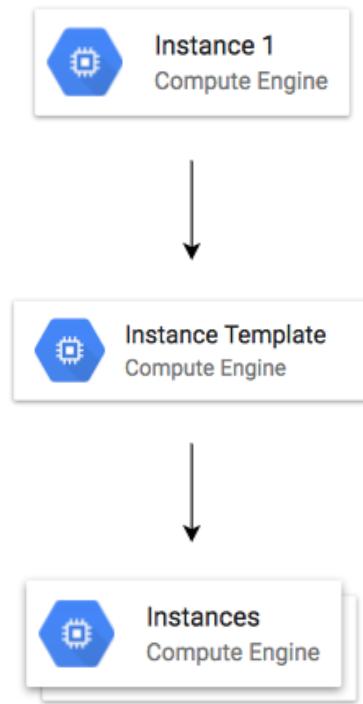
Bootstrapping with Startup script

```
#!/bin/bash
apt update
apt -y install apache2
echo "Hello world from $(hostname) $(hostname -I)" > /var/www/httr
```

- **Bootstrapping:** Install OS patches or software when an VM instance is launched.
- In VM, you can configure **Startup script** to bootstrap
- **DEMO** - Using Startup script

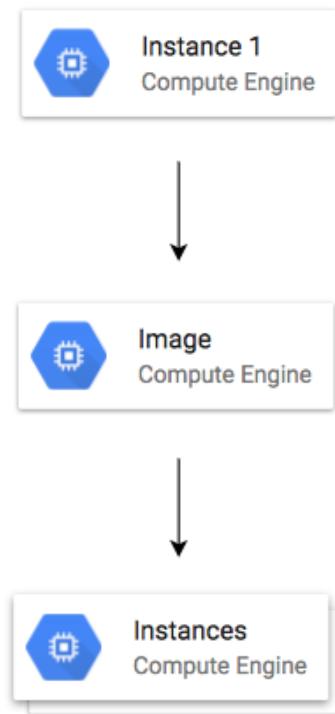
Instance templates

- 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 software at launch of VM instances **increases boot up time**
- How about creating a custom image with OS patches and software **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



Reducing Costs - Compute Engine Virtual Machines

Feature	GCE
Sustained use discounts	Automatic discounts for using resources for long periods of time 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.
Committed use discounts	Reserve compute instances ahead of time Commit for 1 year or 3 years Up to 70% discount based on machine type and GPUs
Preemptible VMs	Cheaper, temporary instances for non critical workloads (Fixed pricing, Max 24 hrs, CHEAPEST)

Compute Engine : Live Migration & Availability Policy

- How do you keep your VM instances running when a host system needs to be updated (a software 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.)

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

In 28
Minutes

Scenario	Solution
What are the pre-requisites to be able to create a VM instance?	<ol style="list-style-type: none">1. Project2. Billing Account3. Compute Engines APIs should be enabled
You want dedicated hardware for your compliance, licensing, and management needs	Sole-tenant nodes
I have 1000s of VM and I want to automate OS patch management, OS inventory management and OS configuration management (manage software installed)	Use "VM Manager"
You want to login to your VM instance to install software	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

Image

- What **operating system** and what **software** 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

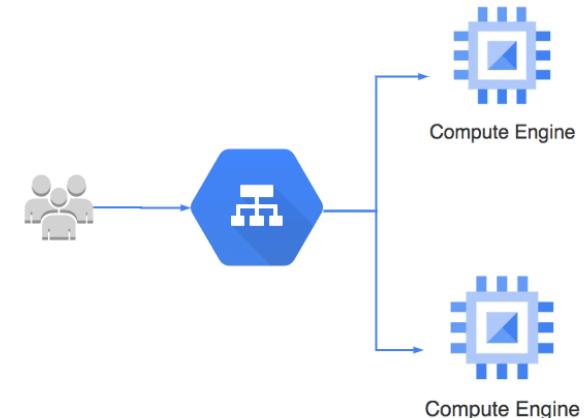
Quick Review

- **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

Instance Groups

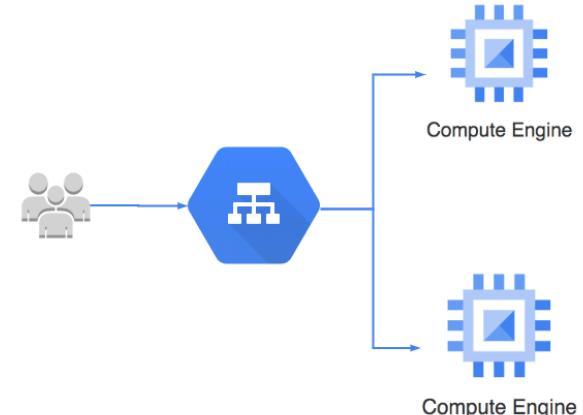
Instance Groups

- 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)



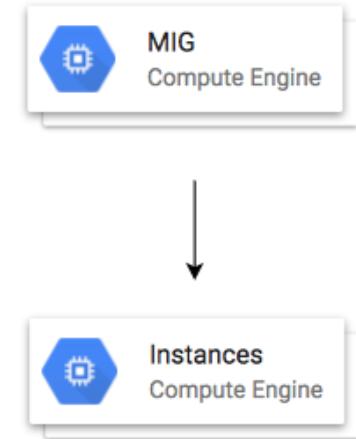
Managed Instance Groups (MIG)

- 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.



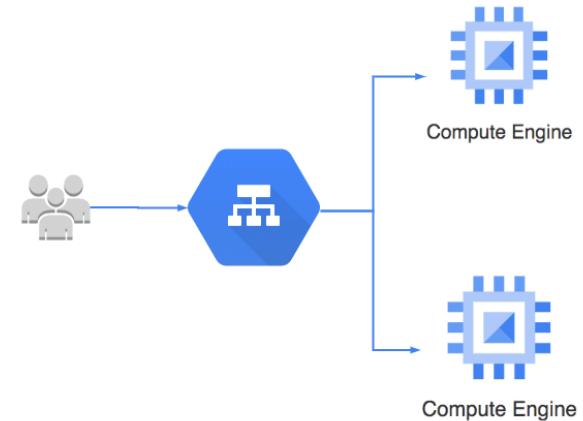
Creating Managed Instance Group (MIG)

- **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)

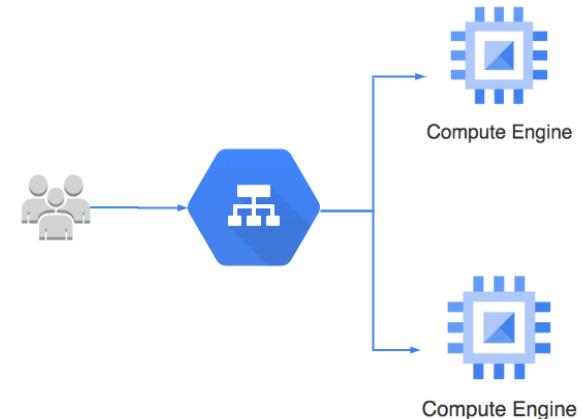
- **Rolling update** - Gradual update of instances in an instance group to the new instance template
 - Specify new template:
 - (OPTIONAL) Specify a template for canary testing
 - Specify how you want the update to be done:
 - When should the update happen?
 - Start the update immediately (Proactive) 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)



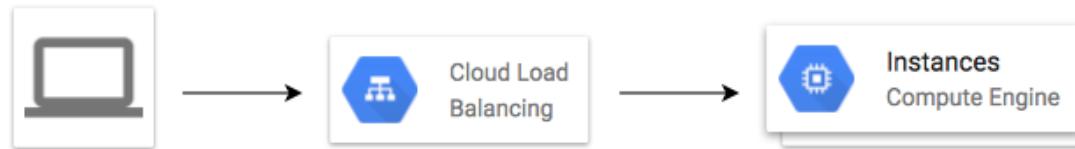
Cloud Load Balancing

Cloud Load Balancing

- Distributes user traffic across instances of an application in single region or multiple regions
 - Fully distributed, software 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

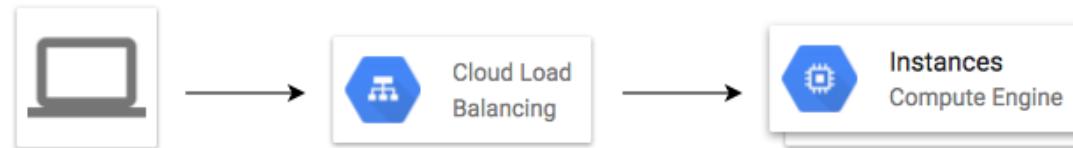


Cloud Load Balancing - Terminology



- **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)

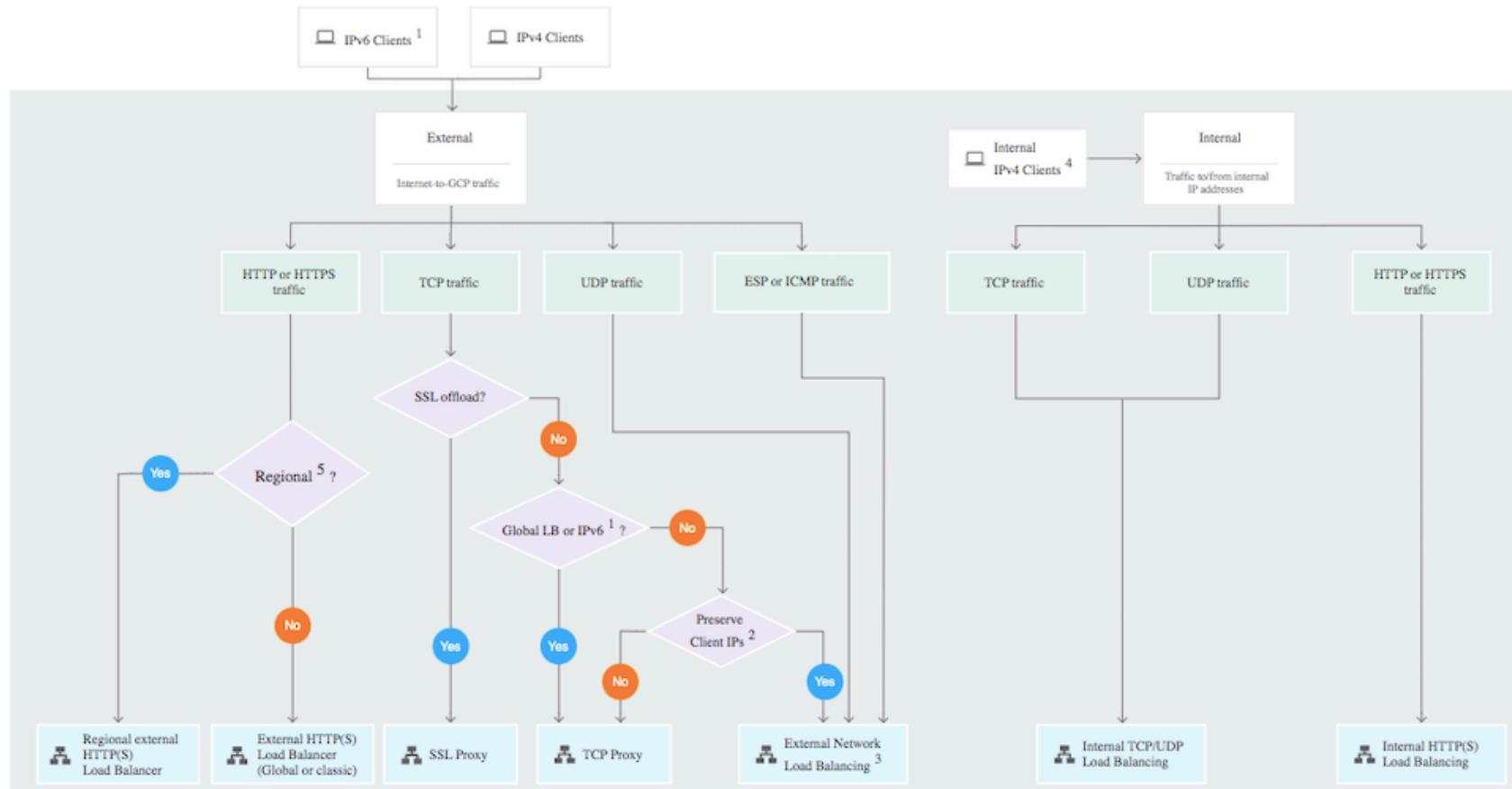
Load Balancing - SSL/TLS Termination/Offloading



- 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>



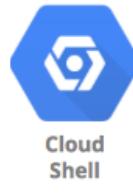
Load Balancer Scenarios

In 28
Minutes

Scenario	Solution
You want only healthy instances to receive traffic	Configure health check
You want high availability for your VM instances	Create Multiple MIGs for your VM instances in multiple regions. Load balance using a Load Balancer.
You want to route requests to multiple microservices using the same load balancer	Create individual MIGs and backends for each microservice. Create Host and path rules to redirect to specific microservice 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 external HTTPS traffic across backend instances, across multiple regions	Choose External HTTP(S) Load Balancer
You want SSL termination for Global non-HTTPS traffic with load balancing	Choose SSL Proxy Load Balancer

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)



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>
- 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

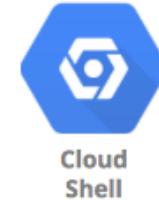
gcloud command structure - Playing with Services

- **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)"

Cloud Shell

In 28
Minutes

- **Important things you need to know about Cloud Shell:**
 - Cloud Shell is backed by a VM instance (automatically provisioned by Google Cloud when you launch Cloud Shell)
 - 5 GB of free persistent disk storage is provided as your \$HOME directory
 - Prepackaged with latest version of Cloud SDK, Docker etc
 - (Remember) Files in your home directory persist between sessions (scripts, user configuration files like .bashrc and .vimrc etc)
 - Instance is terminated if you are inactive for more than 20 minutes
 - Any modifications that you made to it outside your \$HOME will be lost
 - (Remember) After 120 days of inactivity, even your \$HOME directory is deleted
 - Cloud Shell can be used to SSH into virtual machines using their private IP addresses



Managed Services

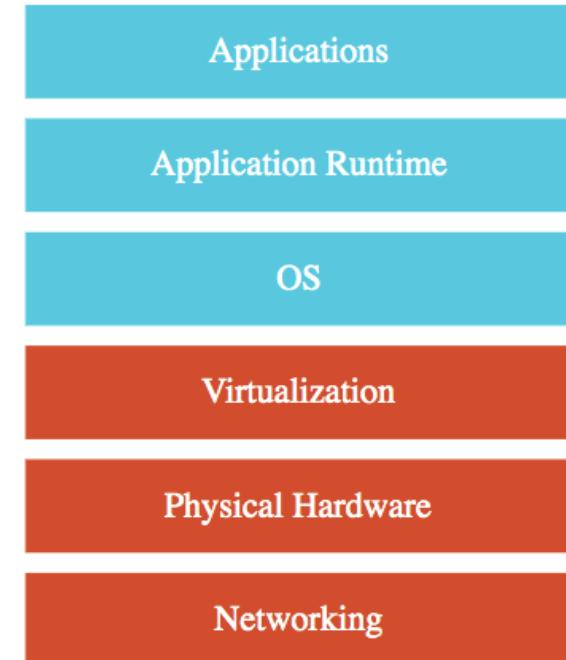
Managed Services

- 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!



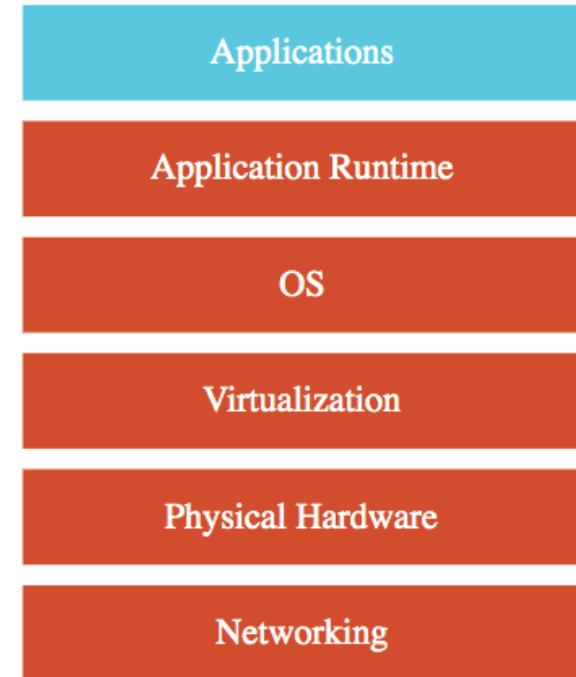
IAAS (Infrastructure as a Service)

- 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!)

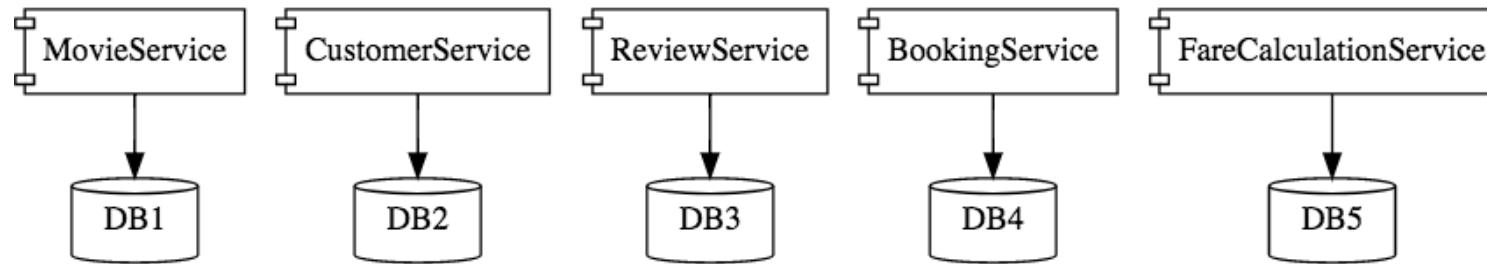


PAAS (Platform as a Service)

- 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!



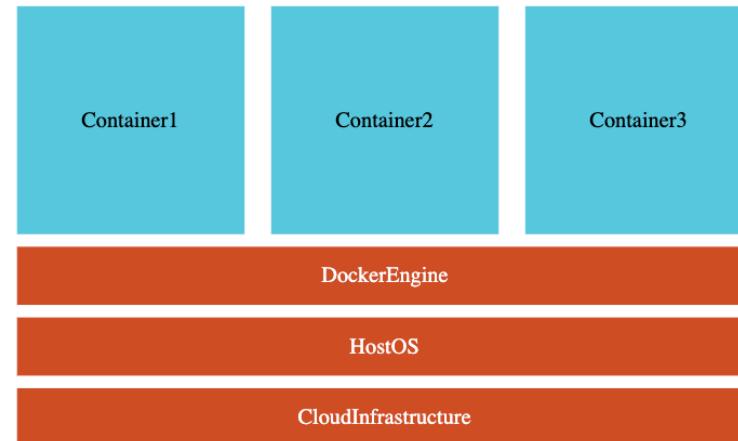
Microservices



- 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!

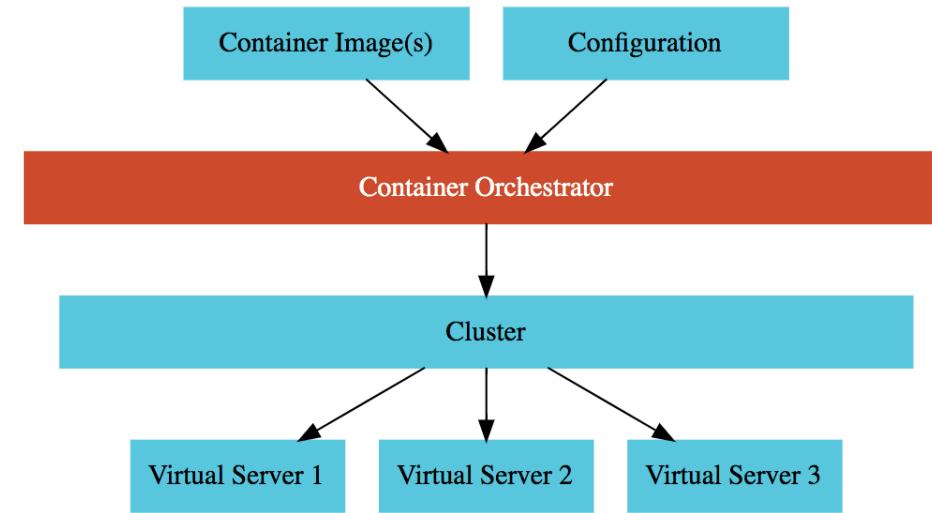
Containers - Docker

- 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



- 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!

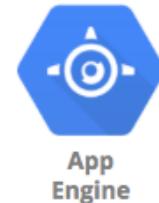
GCP Managed Services for Compute

Service	Details	Category
Compute Engine	High-performance and general purpose VMs that scale globally	IaaS
Google Kubernetes Engine	Orchestrate containerized microservices on Kubernetes Needs advanced cluster configuration and monitoring	CaaS
App Engine	Build highly scalable applications on a fully managed platform using open and familiar languages and tools	PaaS (CaaS, Serverless)
Cloud Functions	Build event driven applications using simple, single-purpose functions	FaaS, Serverless
Cloud Run	Develop and deploy highly scalable containerized applications. Does NOT need a cluster!	CaaS (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 Software
 - Choosing Hardware
 - Fine grained Access/Permissions (Certificates/Firewalls)
 - Availability etc



App
Engine



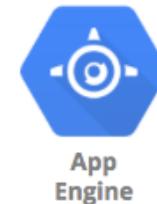
Compute
Engine

- **App Engine**

- PaaS
- Serverless
- LESSER Responsibility
- LOWER Flexibility

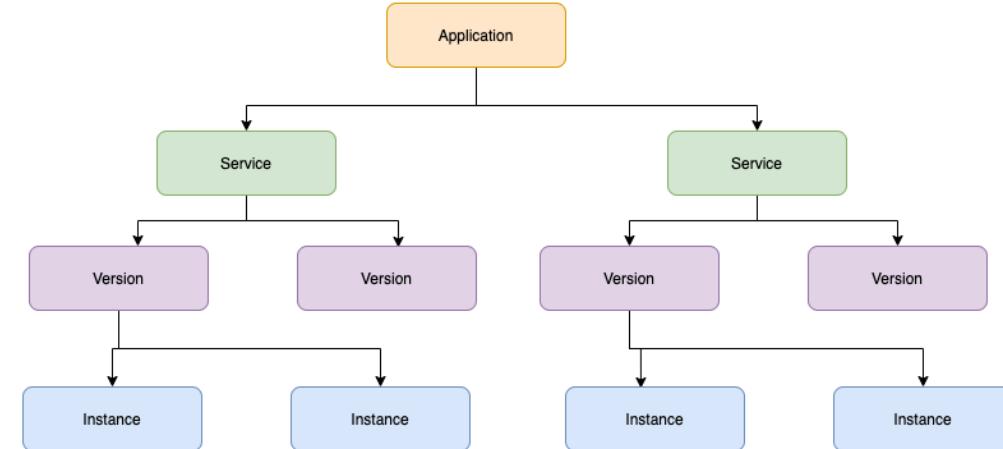
App Engine environments

- **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



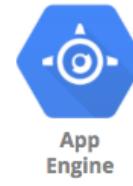
App Engine - Comparison

In 28
Minutes

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 time	Seconds	Minutes
Rapid Scaling	Yes	No
Max. request timeout	1 to 10 minutes	60 minutes
Local disk	Mostly(except for Python, PHP). Can write to /tmp.	Yes. Ephemeral. New Disk on 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
```

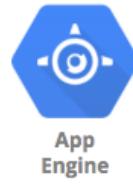
App Engine - Cron Job

```
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

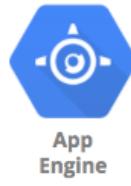
AppEngine - Deploying new versions without downtime

- How do I go from V1 to V2 without downtime?
- **Option 1:** I'm very confident - Deploy & shift all traffic at once:
 - Deploy and shift 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 shifting traffic (*--no-promote*)
 - *gcloud app deploy --no-promote*
 - **STEP 2:** Shift 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 shift 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*)



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

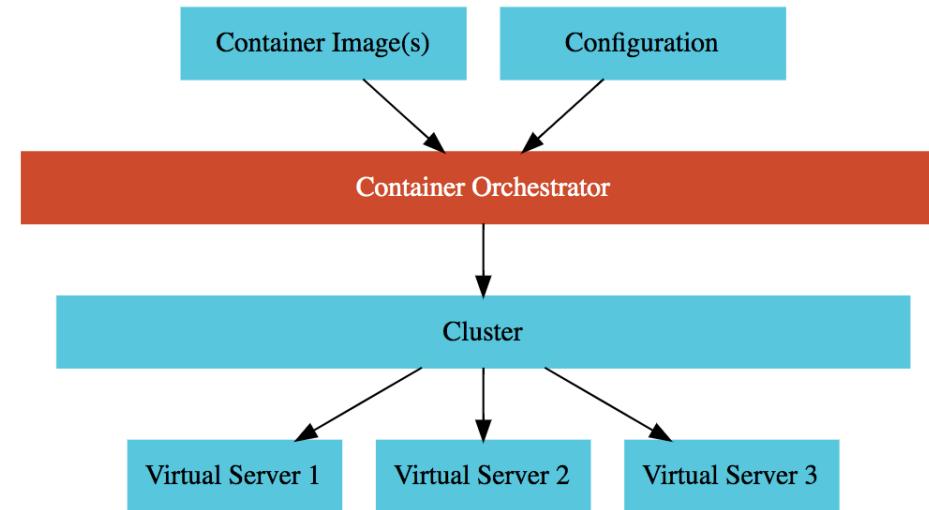
In 28
Minutes

Scenario	Solution
I want to create two Google App Engine Apps in the same project	Not possible. You can only have one App Engine App per project. However you can have multiple services and multiple version for each service.
I want to create two Google App Engine Services inside the same App	Yup. You can create multiple services under the same app. Each service can have multiple versions as well.
I want to move my Google App Engine App to a different region	App Engine App is region specific. You CANNOT move it to different region. Create a new project and create new app engine app in the new region.
I don't want to allow more than 10 instances for an App Engine app	Set max_instances in app.yaml. Example: basic_scaling > max_instances: 10
Deploy new version to App Engine without shifting traffic	gcloud app deploy --no-promote

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 Engine

Autopilot Mode - Google Kubernetes Engine

- New mode of operation for GKE
- Reduce your operational costs in running Kubernetes clusters
- Provides an hands-off experience
 - Do NOT worry about managing the cluster infrastructure (nodes, node pools..)
 - GKE will manage the cluster for you!
 - Pay for CPU, memory, and storage requested by Pods



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 Engine

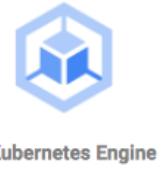
Kubernetes - A Microservice Journey - Deploy Microservice

- 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 --from-literal=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!

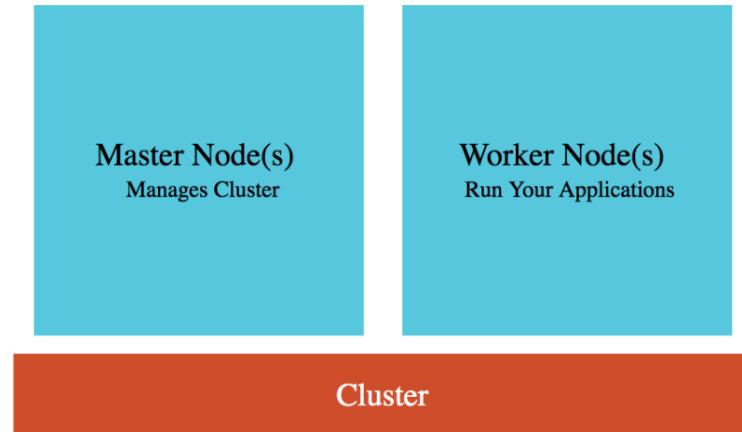
- **11:** Deploy a new microservice which needs nodes with a GPU attached
 - Attach a new node pool with GPU instances to your cluster
 - `gcloud container node-pools create POOL_NAME --cluster CLUSTER_NAME`
 - `gcloud container node-pools list --cluster CLUSTER_NAME`
 - Deploy the new microservice to 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`



Kubernetes Engine

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 & replicaset
 - **etcd** - Distributed database storing the cluster state
- **Worker Node** components:
 - Runs your pods
 - **Kubelet** - Manages communication with master node(s)



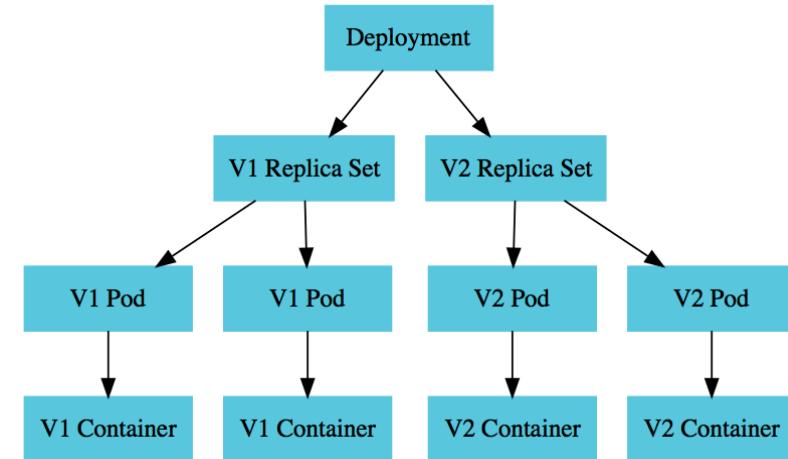
GKE Cluster Types

In 28
Minutes

Type	Description	
Zonal Cluster	Single Zone - Single Control plane. Nodes running in the same zone.	 Kubernetes Engine
	Multi-zonal - Single Control plane but nodes running in multiple zones	
Regional cluster	Replicas of the control plane runs in multiple zones of a given region. Nodes also run in same zones where control plane runs.	
Private cluster	VPC-native cluster. Nodes only have internal IP addresses.	
Alpha cluster	Clusters with alpha APIs - early feature API's. Used to test new K8S features.	

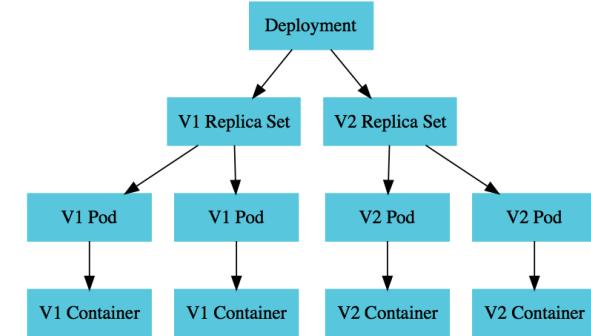
Kubernetes - Pods

- 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

Using Kubernetes Namespaces

- I would want to create **multiple virtual clusters** inside the same physical Kubernetes cluster. How can I do that?
 - Create namespaces
 - Recommended for clusters shared between multiple teams or projects
- List namespaces: `kubectl get namespace`
 - Example namespaces
 - **default** - Used by objects with no namespace
 - **kube-system** - namespace for objects created by the Kubernetes system
- You can include namespace in almost every kubectl command (default is default namespace)
 - `kubectl create namespace my-new-namespace`
 - `kubectl run nginx --image=nginx --namespace=my-new-namespace`
 - `kubectl get pods --namespace=my-new-namespace`
- Namespaces can be used to configure Kubernetes RBAC



Kubernetes Engine

Understanding Service Discovery - Namespaces and DNS



- Microservices need to talk with each other
 - You do NOT want to hardcode the URL of a microservice
 - Enter Service Discovery
- How can a service in a Kubernetes cluster talk to another service in the same Kubernetes cluster without hardcoded the URL?
 - Use Service Discovery provided by Kubernetes
 - **Fully qualified domain name (FQDN)** - {SERVICE-NAME}.{NAMESPACE-NAME}.svc.cluster.local
 - You can use just the {SERVICE-NAME} if you are looking for a service in the same namespace!
 - Example: If you have services with name `microservice-1` and `microservice-2` in the same namespace, they can talk to each other just by using their service names (`http://microservice-1`, `http://microservice-2`)
 - **BONUS:** Automatic Internal Load Balancing by Kubernetes

Troubleshooting Kubernetes Deployment Errors

- Your deployment is failing (*kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.1.RELEASE*)
 - **ImagePullBackOff** or **ErrImagePull**: Problems accessing the container image
 - Is image name correct?
 - Is image tag correct?
 - Does the node pool service account have access to the container registry image?
 - **CrashLoopBackOff**: Container is repeatedly crashing
 - Look at the logs to find out why the container is crashing
 - `kubectl logs POD_NAME`
 - You can also try to connect to the pod and execute commands
 - `kubectl exec -it POD_NAME -- ls`
 - **PodUnschedulable**: Pod cannot be scheduled
 - Are there sufficient resources on the cluster?
- (REMEMBER) `kubectl describe pods POD_NAME` gives you current state of the Pod and recent events

GKE - Remember

In 28
Minutes

- 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

Google Kubernetes Engine - Scenarios - 1

In 28
Minutes

Scenario	Solution
You want to keep your costs low and optimize your GKE implementation	Consider Preemptible VMs, Appropriate region, Committed-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 GKE solution	Configure Horizontal Pod Autoscaler for deployments and Cluster Autoscaler for node pools
You want to execute untrusted third-party code in Kubernetes Cluster	Create a new node pool with GKE Sandbox. Deploy untrusted code to Sandbox node pool.

Google Kubernetes Engine - Scenarios - 2

In 28
Minutes

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

Kubernetes Declarative Configuration

Understanding Kubernetes YAML - Basics

- Each configuration YAML contains:
 - **apiVersion**: Which version of YAML configuration are you making use of?
 - **kind**: What type of configuration is it? (Deployment, Service, ConfigMap, Secret etc)
 - **metadata**: What metadata do you want to associate with a resource?
 - **name**: name of the resource
 - **namespace**: namespace of the resource
 - **labels**: key/value pairs attached with the resource
 - **spec**: Configuration of the resource

```
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
```

Understanding Kubernetes YAML - Deployment

- Commands we executed earlier:
 - `kubectl create deployment hello-world-rest-api --image=in28min/hello-world-rest-api:0.0.3.RELEASE`
 - `kubectl scale deployment hello-world-rest-api --replicas=3`
- Deployment contains definition of Pod & Deployment!
 - What is the template for the pod? (`.spec.template`)
 - How many containers per pod?
 - What is the image to use? (`.spec.template.spec.containers[0].image`)
 - How many pods? (`.spec.replicas`)
 - And a lot more details...

```
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
```

Understanding Kubernetes YAML - Service

- Command we executed earlier:

- `kubectl expose deployment hello-world-rest-api --type=LoadBalancer --port=8080`

- Service configuration:

- **ports**

- How are you exposing your service to end users? (port, protocol)
 - What is the container port? (targetPort)

- **selector**

- What pods does this service map to?

- **type**

- **LoadBalancer**: Exposes Service externally using a cloud provider's load balancer
 - **ClusterIP**: Exposes Service on a cluster-internal IP
 - **NodePort**: Exposes Service on each Node's IP at a static port (the NodePort)

```
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
```

Understanding Kubernetes YAML - Labels

- **Labels:** key/value pairs that are attached to Kubernetes objects
 - Example: Label app=hello-world-rest-api is associated with Deployment
 - Labels can be used to map one Kubernetes object to another
 - Pod can be mapped to Deployment using Labels
 - Deployment can be mapped to a Service using Labels
 - Mapping is defined in **selector**
 - selector > matchLabels (app: hello-world-rest-api)
- Labels play a **key role** in the flexibility that Kubernetes provides
 - Example: Map one Service to two Deployments
 - Demo

```
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
```

```
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
          name: hello-world-rest-api
```

Understanding Liveness and Readiness Probes

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: currency-exchange
spec:
  template:
    spec:
      containers:
        -image: in28min/hello-world-rest-api:0.0.3.RELEASE
        name: hello-world-rest-api
        readinessProbe:
          httpGet:
            port: 8000
            path: /readiness
        livenessProbe:
          httpGet:
            port: 8000
            path: /liveness
```

- Kubernetes uses probes to check the health of a microservice:
 - If readiness probe is not successful, no traffic is sent
 - If liveness probe is not successful, pod is restarted

GKE - Deployment Strategy

```
replicas: 2
minReadySeconds: 45
selector:
  matchLabels:
    app: hello-world-rest-api
    version: v1
strategy:
  rollingUpdate:
    maxSurge: 25%
    maxUnavailable: 25%
  type: RollingUpdate
```

- Default: Rolling Update
 - **maxSurge**: Max no of Pods that can be created over the desired number of Pods
 - **maxUnavailable**: Max no of Pods that can be unavailable during the update process
 - **Example**: 1 new instance at a time, No reduction in capacity:
 - maxSurge set to 1, maxUnavailable set to 0
 - **Example**: Minimum costs, 1 at a time (No increase in no of instances)
 - maxSurge set to 0, maxUnavailable set to 1
- Alternative: Recreate (All existing Pods are killed before new ones are created)

Understanding Kubernetes Ingress

- **Ingress:** API object that manages external access to the services in a cluster
- Recommended Approach for providing external access to services in a cluster
 - Control traffic by defining rules (or routes) on the Ingress resource
 - **Recommendation:** NodePort Service for each microservice. Expose using a Ingress.
 - **Advantage:** One Load Balancer serves multiple microservices!
 - Provides load balancing and SSL termination
 - **An Ingress can handle traffic for multiple hosts:**
 - Same Microservice can be exposed on Different Hosts

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: gateway-ingress
spec:
  rules:
  - http:
    paths:
      - path: /currency-exchange/*
        backend:
          serviceName: currency-exchange
          servicePort: 8000
      - path: /currency-conversion/*
        backend:
          serviceName: currency-conversion
          servicePort: 8100
```

```
spec:
  rules:
  - host: my-first-url.com
    http:
      paths:
        - path: /currency-exchange/*
          backend:
            serviceName: currency-exchange
            servicePort: 8000
  - host: test-another-url.com
    http:
      paths:
        - path: /currency-exchange/*
          backend:
            serviceName: currency-exchange
```

Understanding PersistentVolume & PersistentVolumeClaim

In 28
Minutes



- You want to **Attach a Persistent Disk with your POD**
- **How can you do that?**
 - **PersistentVolume**: Storage in the cluster that has been provisioned by an administrator
 - **PersistentVolumeClaim**: Request for storage by a user
 - Abstracts how storage is provided from how it is consumed
- **Steps:**
 - Create a PersistentVolume connected to the Persistent Disk
 - Create a PersistentVolumeClaim
 - Connect your pod to PersistentVolumeClaim

PersistentVolumeClaim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

- Lifecycle independent of any individual Pod
- **Claim For:**
 - Storage Capacity and
 - Access Mode

Pod to PersistentVolumeClaim

```
kind: Pod
apiVersion: v1
metadata:
  name: mysql-pod
spec:
  volumes:
    - name: mysql-persistent-storage
      persistentVolumeClaim:
        claimName: mysql-pvc
  containers:
    - name: CONTAINER_NAME
      image: mysql:5.6
      ports:
        - containerPort: 3306
          name: mysql
  volumeMounts:
    - mountPath: /var/lib/mysql
      name: mysql-persistent-storage
```

- Map your claim to a Pod

PersistentVolume

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  claimRef:
    namespace: default
    name: mysql-pvc
  gcePersistentDisk:
    pdName: DISK_NAME
    fsType: ext4
```

- Lifecycle independent of any individual Pod
- **Configure:**
 - Type of Storage
 - Storage Capacity and
 - Access Mode

Understanding Kubernetes Network Policies

- How do you decide which pod is allowed to communicate with which pod and what kind of communication is allowed?
 - Network Policies
- Control communication between pods based on:
 - Pod selectors
 - Namespaces
 - IP blocks
- Define rules for ingress:
 - Define the pods that the rule is applicable to in *podSelector*
 - Configure *policyTypes* as ingress
 - Define traffic is allowed *from* and which *ports*

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: my-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      name: microservice-2
  policyTypes:
  - Ingress
  ingress:
  - from:
    - ipBlock:
        cidr: 10.20.0.0/16
    - namespaceSelector:
        matchLabels:
          name: my-namespace
    - podSelector:
        matchLabels:
          name: microservice-1
  ports:
  - protocol: TCP
    port: 8080
```

Exploring Kubernetes Network Policies - Egress Example

- Define rules for egress
 - Define the pods that the rule is applicable to in *podSelector*
 - Configure *policyTypes* as egress
 - Define where traffic is allowed *to* and on which *ports*
- (Remember) You can define egress and ingress in *policyTypes* in the same policy
- (Remember) By default, if no policies exist in a namespace, then all ingress and egress traffic is allowed to and from pods in that namespace.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: my-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      name: microservice-2
  policyTypes:
  - Egress
  egress:
  - to:
    - ipBlock:
        cidr: 10.20.0.0/16
  ports:
  - protocol: TCP
    port: 8080
```

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny-all
spec:
  podSelector: {}
  policyTypes:
  - Ingress
  - Egress
```

Exploring Graceful shutdown - Kubernetes

```
process.on('SIGTERM', () => {
  console.info('Time to clean up and shutdown.');
});

spec:
  containers:
    - name: hello-world-rest-api
      image: in28min/hello-world-rest-api:0.0.3.RELEASE
      lifecycle:
        preStop:
          exec:
            command: ["/bin/sh", "-c", "YOUR_COMMAND_TO_EXECUTE"]
```

- Events when **Kubernetes decides to terminate your pod**:
 - preStop Hook is executed
 - SIGTERM signal is sent to the pod
 - Kubernetes waits for a grace period (default: 30 seconds)
 - (If POD is till running) SIGKILL signal is sent to forcibly remove the pod
 - **Recommended:** Clean up your connections and shutdown properly
 - Define preStop Hook and/or handle SIGTERM signal

Kubernetes - Scenarios

Scenario	Solution
How can you divide one physical Kubernetes cluster into multiple virtual clusters with customized access to different teams?	Use namespaces
How can a service in a Kubernetes cluster talk to another service in the same Kubernetes cluster without hardcoding the URL?	Service Discovery. Fully qualified domain name (FQDN) - {SERVICE-NAME}.{NAMESPACE-NAME}.svc.cluster.local
What does Kubernetes do to ensure that your application has completely started up and is ready to accept traffic?	Use readiness probe (if configured)
You are making use of Kubernetes Deployment Strategy - Rolling Updates. When releasing new version, you want to update 1 instance at a time without reduction in capacity	maxSurge set to 1, maxUnavailable set to 0

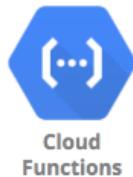
Kubernetes - Scenarios - 2

Scenario	Solution
You want to restrict pods in certain namespace from communicating with pods in another namespace. Both these set of pods are deployed on the same cluster.	Network Policies
You are making a new deployment. You get ImagePullBackOff error	Is image name correct? Is image tag correct? Does the node pool service account have access to the container registry image?
You are making a new deployment. You get PodUnschedulable error	Are there sufficient resources on the cluster?

Cloud Functions and Cloud Run

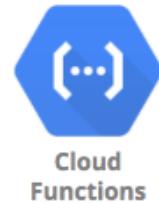
Cloud Functions

- Imagine you want to **execute some code when an event happens?**
 - A file is uploaded in Cloud Storage (OR) An error log is written to Cloud Logging (OR)
 - A message arrives to Cloud Pub/Sub (OR) A http/https invocation is received
- 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
 - Memory and CPU provisioned
 - **Time Bound** - Default 1 min and MAX 60 minutes (3600 seconds)
 - **2 product versions**
 - Cloud Functions (1st gen): First version
 - Cloud Functions (2nd gen): New version built on top of Cloud Run and Eventarc



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');

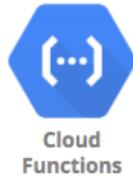
exports.helloHttp = (req, res) => {
  res.send(`Hello ${escapeHtml(req.query.name || req.body.name || 'World')}`);
};
```

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 60 minutes(3600 seconds)



Cloud Run & Cloud Run for Anthos

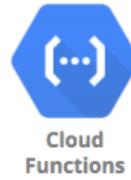
- **Cloud Run - "Container to Production in Seconds"**
 - Built on top of an open standard - Knative
 - **Fully managed** serverless platform for containerized applications
 - ZERO infrastructure management
 - Pay-per-use (For used CPU, Memory, Requests and Networking)
- Fully integrated **end-to-end 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

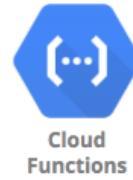
Cloud Functions - Second Generation - What's New?

- **2 Product Versions:**
 - Cloud Functions (1st gen): First version
 - Cloud Functions (2nd gen): New version built on top of Cloud Run and Eventarc
- **Recommended:** Use Cloud Functions (2nd gen)
- **Key Enhancements in 2nd gen:**
 - **Longer Request timeout:** Up to 60 minutes for HTTP-triggered functions
 - **Larger instance sizes:** Up to 16GiB RAM with 4 vCPU (v1: Up to 8GB RAM with 2 vCPU)
 - **Concurrency:** Up to 1000 concurrent requests per function instance (v1: 1 concurrent request per function instance)
 - **Multiple Function Revisions and Traffic splitting supported** (v1: NOT supported)
 - **Support for 90+ event types** - enabled by Eventarc (v1: Only 7)
- **DEMO!**



Cloud Functions - Scaling and Concurrency

- **Typical serverless functions architecture:**
 - **Autoscaling** - As new invocations come in, new function instances are created
 - One function instance handles ONLY ONE request AT A TIME
 - 3 concurrent function invocations => 3 function instances
 - If a 4th function invocation occurs while existing invocations are in progress, a new function instance will be created
 - HOWEVER, a function instance that completed execution may be reused for future requests
 - **(Typical Problem) Cold Start:**
 - New function instance initialization from scratch can take time
 - (Solution) Configure Min number of instances (increases cost)
- **1st Gen** uses the typical serverless functions architecture
- **2nd Gen** adds a very important new feature:
 - One function instance can handle multiple requests AT THE SAME TIME
 - **Concurrency:** How many concurrent invocations can one function instance handle? (Max 1000)
 - (IMPORTANT) Your function code should be safe to execute concurrently



Cloud Functions - Deployment using gcloud

- **gcloud functions deploy [NAME]**

- **--docker-registry** (registry to store the function's Docker images)
 - Default - container - registry
 - Alternative - artifact - registry
- **--docker-repository** (repository to store the function's Docker images)
 - Example: (projects/\${PROJECT}/locations/\${LOCATION}/repositories/\${REPOSITORY})
- **--gen2** (Use 2nd gen. If this option is not present, 1st gen will be used)
- **--runtime** (nodejs, python, java,...)
 - Reference - <https://cloud.google.com/functions/docs/runtime-support>
- **--service-account** (Service account to use)
 - 1 GEN - default - App Engine default service account - PROJECT_ID@appspot.gserviceaccount.com
 - 2 GEN - Default compute service account - PROJECT_N0-compute@developer.gserviceaccount.com
- **--timeout** (function execution timeout)
- **--max-instances** (function execution exceeding max-instances times out)
- **--min-instances** (avoid cold starts at higher cost)

Cloud Functions - Deployment using gcloud - 2

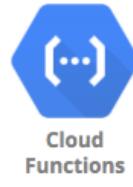
```
//Deploy Pubsub Triggered gen2 function from Cloud Storage Bucket
gcloud functions deploy my-pubsub-function \
    --gen2 \
    --region=europe-west1 \
    --runtime=nodejs16 \
    --source=gs://my-source-bucket/source.zip \
    --trigger-topic=my-pubsub-topic
```

- **gcloud functions deploy [NAME]**

- **--source**
 - Zip file from Google Cloud Storage (gs://my-source-bucket/my_function_source.zip) (OR)
 - Source Repo ([https://URL/projects/\\${PROJECT}/repos/\\${REPO}](https://URL/projects/${PROJECT}/repos/${REPO})) (OR)
 - Local file system
- **--trigger-bucket** (OR) **--trigger-http** (OR) **--trigger-topic** (OR) **--trigger-event-filters** (ONLY in gen2 - Eventarc matching criteria for the trigger)
- **--serve-all-traffic-latest-revision** (ONLY in gen2)

Cloud Functions - Best Practices

- To avoid cold starts, set **min no of instances** (increases cost)
 - Minimize dependencies (loading dependencies increases initialization time)
- Configure **max no of instances** (protect from abnormally high request levels)
- Use Cloud Endpoints (or Apigee or API gateway) for versioning
- Use Cloud Run (& Cloud Functions gen 2) **revisions** for safer releases:
 - Configure which revisions should receive traffic and how much
 - You can rollback to a previous revision, if needed
- Use Secret Manager to securely store secrets (ex: API keys)
- Use **Individual Service Accounts** for each function
 - Grant `roles/cloudfunctions.invoker` role to invoke a cloud function
- Manage dependencies using your language specific tool (npm, pip,..)



Cloud Run and Cloud Functions - Scenarios

Scenario	Solution
With default configuration, what happens when a Cloud Run revision does not receive any traffic?	It will be scaled in.
You want to run a container receiving intermittent requests at minimum cost	Cloud Run (App Engine Flexible will maintain at least one instance and running Kubernetes cluster will be expensive)
You do NOT want to use containers but you want be able to receive intermittent requests at minimum cost	App Engine Standard or Cloud Functions
You want to move a Knative compatible app to Google Cloud with least effort	Consider Cloud Run

Encryption



- **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

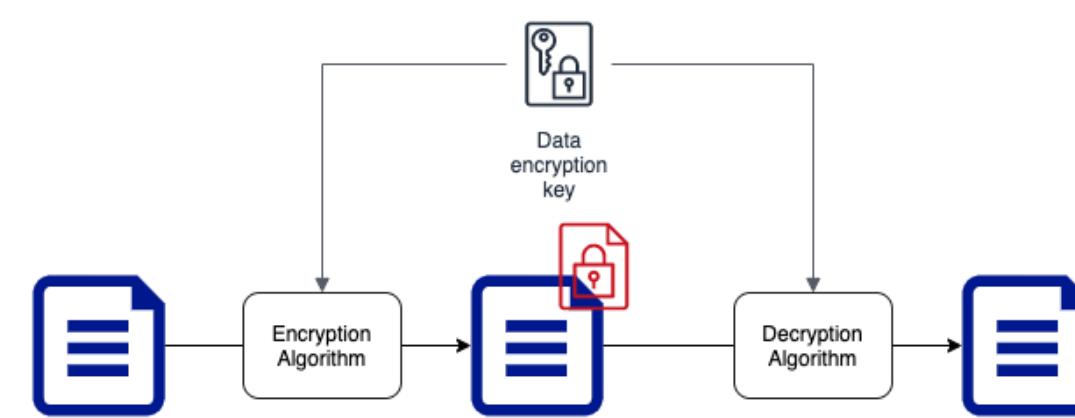
In 28
Minutes



- 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

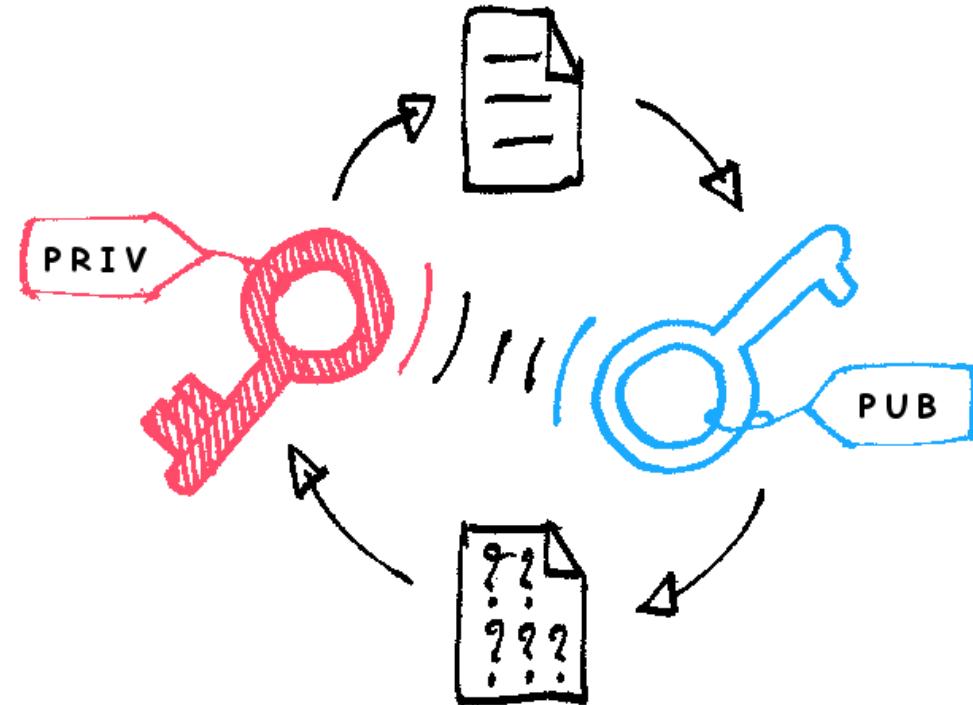
In 28
Minutes



- 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 Cryptography**
- 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_encryption_\(colored\).png](https://commons.wikimedia.org/wiki/File:Asymmetric_encryption_(colored).png)

Cloud KMS

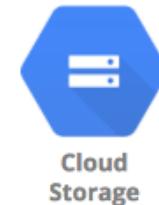
- Create and manage **cryptographic keys** (symmetric and asymmetric)
- **Control their use** in your applications and GCP Services
- Provides an API to encrypt, decrypt, or sign data
- Use existing cryptographic keys created on premises
- **Integrates with almost all GCP services** that need data encryption:
 - Google-managed key: No configuration required
 - Customer-managed key: Use key from KMS
 - Customer-supplied key: Provide your own key



Object Storage - Cloud Storage

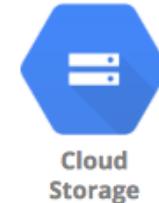
Cloud Storage

- 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



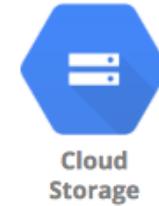
Cloud Storage - Objects and Buckets

- 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)

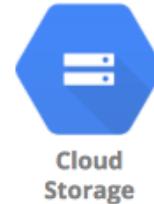


Cloud Storage - Storage Classes - Comparison

Storage Class	Name	Minimum Storage duration	Typical Monthly availability	Use case
Standard	STANDARD	None	> 99.99% in multi region and dual region, 99.99% in regions	Frequently used data/Short period of time
Nearline storage	NEARLINE	30 days	99.95% in multi region and dual region, 99.9% in regions	Read or modify once a month on average
Coldline storage	COLDLINE	90 days	99.95% in multi region and dual region, 99.9% in regions	Read or modify at most once a quarter
Archive storage	ARCHIVE	365 days	99.95% in multi region and dual region, 99.9% in regions	Less than once a year

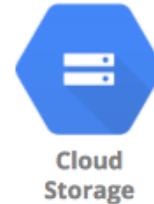
Features across Storage Classes

- High durability (99.99999999% 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



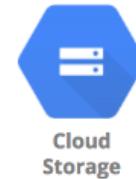
Object Versioning

- 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

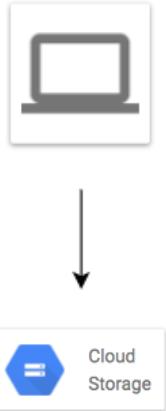


Object Lifecycle Management - Example Rule

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

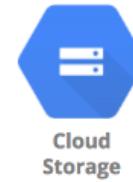
Cloud Storage - Encryption

- (Default) Cloud Storage encrypts data on the server side!
- **Server-side** encryption: Performed by GCS after it receives data
 - **Google-managed** - Default (No configuration needed)
 - **Customer-managed** - Keys managed by customer in **Cloud KMS**:
 - GCS Service Account should have access to customer-managed keys in KMS to be able to encrypt and decrypt files
 - **Customer-supplied** - Customer supplies the keys with every GCS operation
 - Cloud Storage does NOT store the key
 - Customer is responsible for storing and using it when making API calls
 - Use API headers when making API calls
 - x-goog-encryption-algorithm, x-goog-encryption-key (Base 64 encryption key), x-goog-encryption-key-sha256 (encryption key hash)
 - OR when using gsutil: In boto configuration file, configure encryption_key under GSUtil section
- (OPTIONAL) **Client-side** encryption - Encryption performed by customer before upload
 - GCP does NOT know about the keys used
 - GCP is NOT involved in encryption or decryption



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)



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 IAM

Cloud Identity and Access Management (IAM)

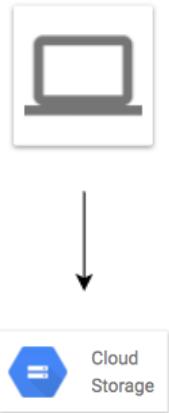
- **Authentication** (is it the right user?) and
- **Authorization** (do they have the right access?)
- **Identities** can be
 - A GCP User (Google Account or Externally Authenticated User)
 - A Group of GCP Users
 - An Application running in GCP
 - An Application running in your data center
 - Unauthenticated 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

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



Cloud IAM

IAM - Predefined Roles - Example Permissions

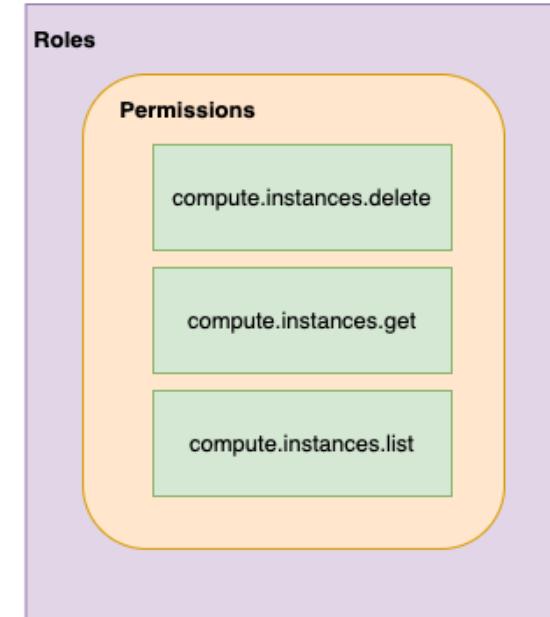
- Important Cloud Storage Roles:
 - **Storage Admin (roles/storage.admin)**
 - storage.buckets.*
 - storage.objects.*
 - **Storage Object Admin (roles/storage.objectAdmin)**
 - storage.objects.*
 - **Storage Object Creator (roles/storage.objectCreator)**
 - storage.objects.create
 - **Storage Object Viewer (roles/storage.objectViewer)**
 - storage.objects.get
 - storage.objects.list
- All four roles have these permissions:
 - resourcemanager.projects.get
 - resourcemanager.projects.list



Cloud IAM

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



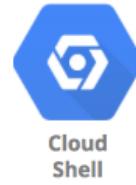
Cloud IAM

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



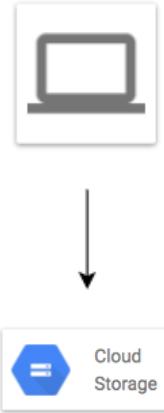
Use case 1 : VM <-> Cloud Storage



- 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!

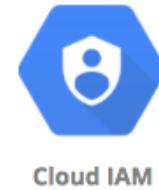
Use case 2 : On Prem <-> Cloud Storage (Long Lived)

- 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 Cloud Storage bucket	Configure the VM to use a Service Account with right permissions
Application on a VM wants to put a message on a Pub Sub Topic	Configure the VM to use a Service Account with right permissions
Is Service Account an identity or a resource?	It is both. You can attach roles with Service Account (identity). 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 A needs to access Cloud Storage bucket in Project B	In project B, add the service account from Project A and assign Storage Object Viewer Permission on the bucket

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

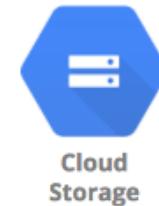
- How do you control access to objects in a Cloud Storage bucket?
- Two types of access controls:
 - Uniform (Recommended) - Uniform bucket level access using IAM
 - Fine-grained - Use IAM and ACLs to control access:
 - Both bucket level and individual object level permissions
- Use Uniform access when 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 IAM

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

In 28
Minutes



- 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**

IAM - Scenarios

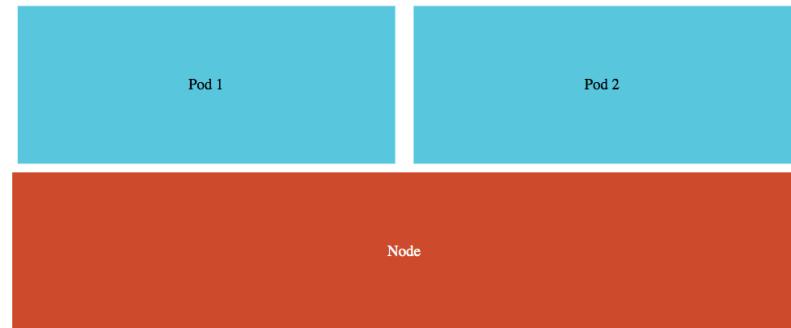
In 28
Minutes

Scenario	Solution
An Application on a GCE VM needs access to cloud storage	Use a Service Account (Google Cloud-managed keys)
An Application on premises needs access to cloud storage	Use Service Account User Managed Key
Allow a user limited time access to your objects	Signed URL
Customize access to a subset of objects in a bucket	Use ACL (Access Control Lists)
Permission is allowed by IAM but NOT by ACL. Will user be able to access the object?	Yes.

Authorization for Kubernetes Workloads

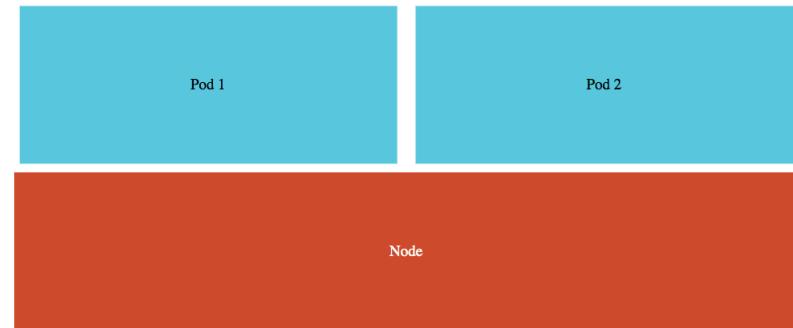
Authorization for Kubernetes Workloads

- Multiple microservices run on the same GKE cluster
- **Different microservices** (pods) need access to different Google Cloud resources
 - Storage, Databases, Pub/Sub etc
- How can you give access to your microservices?
 - **Option 1:** Assign permissions at Node (Compute Engine VM) level
 - **Option 2:** Assign permissions at Pod (individual microservice) level



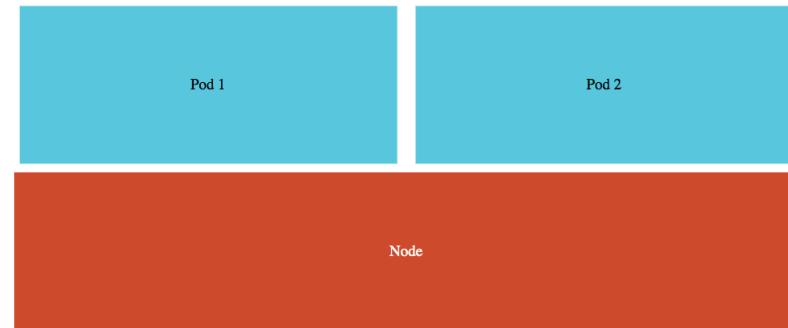
Exploring Kubernetes Node level permissions

- Each Node in a GKE cluster is a **Compute Engine VM**
- Default service account assigned to a node is **Compute Engine default service account**
 - HOWEVER, GCE default service account has wide range of permissions
 - NOT RECOMMENDED
 - OPTION: Create a new service account with customized permissions
 - Service account is shared between microservices => it should have all permissions that all microservices need
 - Violates the principle of least privilege (NOT RECOMMENDED)



Exploring Kubernetes Pod level permissions

- Permissions (and service accounts) are best configured at microservice level
 - Each microservice can have individual service account with customized permissions
- How can you assign a Service Account to a microservice (or a pod)?
 - Option 1: Using Kubernetes Secrets (or secrets management)
 - Option 2: Using Workload Identity



Option 1: Using Kubernetes Secrets

- 1: Create a service account with the right permissions
- 2: Create a JSON key for the service account
- 3: Import service account key as a Secret
 - `kubectl create secret generic my-service-account-key --from-file=key.json=my-service-account-key.json`
- 4: Make your application use the Kubernetes secret by configuring the deployment

```
apiVersion: apps/v1
kind: Deployment
spec:
  template:
    spec:
      volumes:
        - name: google-cloud-key
          secret:
            secretName: my-service-account-key
      containers:
        - name: your-microservice
          image: 'gcr.io/your-microservice:version'
          volumeMounts:
            - name: google-cloud-key
              mountPath: /var/secrets/google
      env:
        - name: GOOGLE_APPLICATION_CREDENTIALS
          value: /var/secrets/google/key.json
```

Option 2: Using Workload Identity (RECOMMENDED)

- Allows you to **configure** a Kubernetes service account to act as a Google service account
 - Kubernetes service accounts are Kubernetes resources
 - Google service accounts are Google Cloud IAM resources
- **Does NOT need** any secrets management!
- Configure **distinct, fine-grained** identities and authorization for each microservice in your cluster
- **PREREQUISITE:** Enable Workload Identity on your GKE cluster
- **RECOMMENDED APPROACH** for accessing Google Cloud services from workloads within GKE



Kubernetes Engine

Workload Identity - Steps

- 1: Enable Workload Identity on GKE cluster
 - At creation: gcloud container clusters create CLUSTER_NAME --workload-pool=PROJECT_ID.svc.id.goog OR
 - At a later point: gcloud container clusters update CLUSTER_NAME --workload-pool=PROJECT_ID.svc.id.goog
 - Enable Workload Identity on Older Node Pools (NOT NEEDED if you created cluster with workload identity enabled)
 - gcloud container node-pools update NODEPOOL_NAME --cluster=CLUSTER_NAME --workload-metadata=GKE_METADATA
- 2: Create a Kubernetes service account
 - kubectl create serviceaccount KUBERNETES_SERVICE_ACCOUNT
- 3: Create a Google service account IAM_SERVICE_ACCOUNT with the permissions that your microservice needs



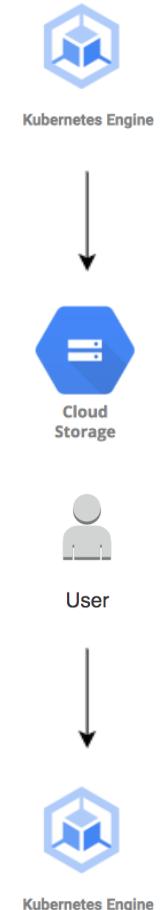
Workload Identity - Steps - 2

```
spec:  
  containers:  
    - image: in28min/hello-world-rest-api:0.0.1.RELEASE  
      name: hello-world-rest-api  
      serviceAccountName: KUBERNETES_SERVICE_ACCOUNT
```

- 4: Configure microservice container to use KUBERNETES_SERVICE_ACCOUNT
- 5: Create an IAM policy binding between Kubernetes service account and the IAM service account
 - *gcloud iam service-accounts add-iam-policy-binding --role roles/iam.workloadIdentityUser --member "serviceAccount:PROJECT_ID.svc.id.goog[KUBERNETES_SERVICE_ACCOUNT]" IAM_SERVICE_ACCOUNT@PROJECT_ID.iam.gserviceaccount.com*
- 6: Add **iam.gke.io/gcp-service-account** annotation to Kubernetes service account
 - *kubectl annotate serviceaccount KUBERNETES_SERVICE_ACCOUNT iam.gke.io/gcp-service-account=IAM_SERVICE_ACCOUNT@PROJECT_ID.iam.gserviceaccount.com*

Authorization for Kubernetes Workloads - Summary

- We focused on authorization for Kubernetes Workloads (with Service Accounts)
 - **Option 1:** Assign permissions at Node (Compute Engine VM) level
 - **Option 2:** Assign permissions at Pod (individual microservice) level
 - 1: Using Kubernetes Secrets
 - 2: Using Workload Identity
 - Recommended option is to use **Workload Identity**
 - Follows principle of least privilege
 - Does NOT need any secrets management!
- **Next Focus Area:** Authorization for users
 - Authorization for Kubernetes cluster operators, Kubernetes workload deployers etc
 - Option 1: Google Cloud IAM
 - Option 2: Kubernetes RBAC



Kubernetes RBAC and Google Cloud IAM

- What things can a **Kubernetes administrator/user** do?
 - Cluster Access
 - Create/Update a Cluster
 - Add/Remove Nodes ...
 - Kubernetes API access
 - Manage Deployments
 - Manage Services ...
- **Google Cloud IAM:** What can a IAM user do across clusters created in a single project, folder or organization?
- **Kubernetes RBAC:** Fine grained access controls at a individual cluster and namespace level
 - Control access at Kubernetes resource levels (deployment, service, pod, secrets, configMap)



Exploring Google Cloud IAM Roles for GKE

- Roles can be assigned at **Project, Folder or Organization** levels
 - They are applicable to all GKE clusters in the Project, Folder or Organization
- **Basic roles:** Owner, Editor, and Viewer
 - NOT RECOMMENDED. Gives global permissions across GCP services.
- **Important Predefined GKE 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 Cluster Viewer** - Read only access to 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



Kubernetes Engine

Exploring Kubernetes Role-based access control (RBAC)

- **Kubernetes RBAC: Kubernetes Feature**
 - Available irrespective of whether you are using GKE, On-prem Kubernetes or AKS (Azure) or EKS (AWS)
 - Fine grained access controls at a individual cluster and namespace level
 - **Control access at Kubernetes resource levels**
 - deployment, service, pod, secrets, configMap etc
- **How do you manage access using Kubernetes RBAC?**
 - **Create Kubernetes RBAC API objects:**
 - **Role:** A set of allowed permissions in a namespace
 - **ClusterRole:** A set of allowed permissions in a cluster
 - **RoleBinding:** Map user (or group or serviceaccount) to a Role or ClusterRole within a namespace
 - **ClusterRoleBinding:** Map a user (or group or service account) to a ClusterRole across all namespaces in a cluster
- **(REMEMBER) These are Kubernetes objects**
 - Just like Service, Deployment, ReplicaSet etc



Understanding Kubernetes RBAC - Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default # namespace is mandatory
  name: all-configmaps-editor-role
rules:
- apiGroups: [""]
  resources: ["configmaps"]
# resourceNames: ["my-configmap"]
  verbs: ["update", "get"]
```

- Represents a set of allowed permissions in a namespace
- **Define:**
 - **namespace**
 - **name**
 - **resources:** Which resource?
 - can be any namespaced resource - "services", "pods" etc
 - **verbs:** What do you want to allow?
 - examples: "get", "list", "create", "update", "delete" etc

Understanding Kubernetes RBAC - ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
    # "namespace" is not configured
    name: all-secrets-viewer-role
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
```

- Represents an allowed set permissions in a cluster
 - **name**
 - Does NOT need a namespace
 - **resources:** Which resource?
 - can be any namespaced resource - "services", "pods" etc (applies to all namespaces in cluster)
 - can be cluster-scoped resources - "nodes"
 - **verbs:** What do you want to allow?
 - examples: "get", "list", "create", "update", "delete" etc

Kubernetes RBAC - RoleBinding and ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding #kind: ClusterRoleBinding
metadata:
  name: edit-configmaps
  namespace: default # NOT needed for ClusterRoleBinding
subjects:
  -kind: User # Group or ServiceAccount. # Google Cloud user account
    name: jane@in28minutes.com
  -kind: ServiceAccount # Kubernetes service account
    name: kubernetes-service-account-name
  -kind: User # IAM service account
    name: iam-service-account@PROJECT-ID.google.com.iam.gserviceaccount.com
roleRef:
  kind: Role # Role or ClusterRole
  name: all-configmaps-editor-role
```

- **Map Subjects to a Role**

- **Subjects:** Google Cloud user account (email address) or
 - Kubernetes service account (name) or
 - IAM service account (service account email address) or
 - Google Group address

Kubernetes RBAC and Google Cloud IAM - Summary

- **Google Cloud IAM:** What can a IAM user do across clusters created in a single project, folder or organization?
 - Use it to configure permissions across multiple clusters
 - You CANNOT use it to configure finegrained permissions to Kubernetes API objects (pods, services, nodes etc)
- **Kubernetes RBAC:** Fine grained access controls at a individual cluster and namespace level
 - Use it to configure fine grained permissions to Kubernetes API objects (pods, services, nodes etc) at a cluster or namespace level
 - Control access at Kubernetes resource levels (deployment, service, pod, secrets, configMap)
- **Final Permissions = IAM permissions + RBAC permissions**
 - Minimum IAM permission needed: container.clusters.get



Kubernetes Engine



Cloud IAM

Understanding Kubernetes Security Best Practices

- Enable Node Auto-Upgrade for GKE nodes
- Use Shielded GKE nodes with secure boot
- Enable Workload Identity
 - (NOT RECOMMENDED) Nodes use Compute Engine default service account by default
 - Configure a separate Service Account with minimum access (logging, monitoring etc) for GKE nodes
 - Use Workload Identity to provide microservice specific accesses
- Use namespaces and RBAC to restrict user access to cluster resources
- Restrict traffic among Pods with a network policy
- Use Kubernetes secrets or Secrets Manager for Secret management
 - Consider HashiCorp Vault if you need multi cloud secrets managements
 - HashiCorp Vault integrates very well with Kubernetes



Kubernetes Engine

Authorization for Kubernetes - Scenarios

Scenario	Solution
Your microservices are going to be deployed to GKE. You want to give each microservice specific access to Google Cloud resources.	Workload Identity
Your microservices are going to be deployed to GKE. You want to give common permissions to Cloud Logging and Cloud Monitoring.	Node Pool Service Account
You are sharing GKE cluster between different teams. You want to ensure that each team has access to deploy only to their specific namespaces.	Kubernetes RBAC
Which role gives you complete access to GKE clusters and the Kubernetes API objects?	Kubernetes Engine Admin (roles/container.admin)
You are assigned permissions using Kubernetes RBAC. But you are unable to access the cluster.	Check if IAM member has the permission - container.clusters.get

Implementing Authentication and Authorization

Getting Started with OAuth

In 28
Minutes

- OAuth is an **Authorization Framework**
- Allows **users** to grant access to **server resources** to another entity **WITHOUT** sharing credentials
- **Example:** Give access to Your Google Drive to Your Photo Editing Application
 - **Resource owner:** You
 - **Resource:** Your Google Drive
 - **Client:** Your Photo Editing application
 - **Authorization Server:** Google OAuth Service
 - **Client IDs and Client Secrets:** How does your Photo Editing application authenticate itself to Google?
 - **Scopes:** What access is allowed? (READ/WRITE)

Playing with OAuth 2.0

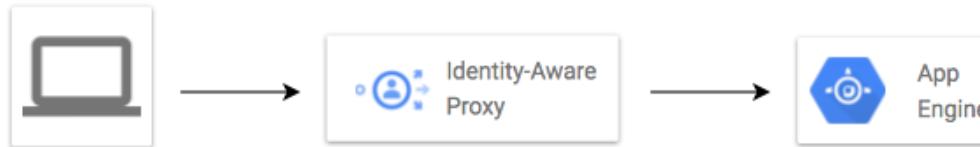
- OAuth 2.0 is the latest and most popular version of OAuth
- OAuth 2.0 Playground
 - <https://developers.google.com/oauthplayground/>
- Example Flows:
 - Authorization Grant: Results in a client receiving an OAuth Access Tokens (Typically JWT tokens)
 - Resource Access: Access the resource using the Access Token
- OAuth Access Tokens allows
 - limited access to specific resources
 - for specific period of time

Getting Started with OpenID Connect (OIDC)

- OAuth is **ONLY about authorization**
 - Does NOT care about how authentication is implemented
- What if you want to allow users to login with Google Id into your application?
 - Use **OpenID Connect (OIDC)**
 - OpenID Connect extends OAuth 2.0
- **OpenID Connect Playground (OIDC)** - <https://openidconnect.net>
 - Access Token - permissions
 - **Id Token** - contains user details
 - Id token is a JSON web token (JWT)
 - JWT contains a few standard fields
 - iss: Issuer
 - sub: Subject
 - aud: Audience
 - iat: Issued at
 - exp: Expiration time
 - And you can add custom fields

Getting Started with Identity-Aware Proxy (IAP)

In 28
Minutes



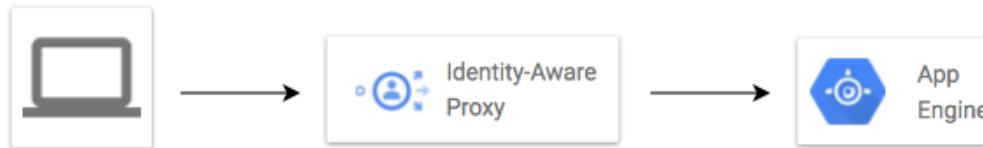
- What is the easiest way to implement authentication and authorization for your apps deployed in App Engine, Cloud Run and GKE?
 - Identity-Aware Proxy (IAP)
- **Identity-Aware Proxy (IAP):** Simplify implementation of authentication and authorization for your cloud-based and on-premises applications
 - Use user **identity** (credentials) and **context** (from where? using what?) to make your authentication/authorization decisions
 - **Simplified integration** with App Engine, Cloud Run and GKE
 - Does NOT need a VPN
 - **(ADDITIONAL FEATURE)** Protect your GCE VMs
 - Does NOT need VPN or Bastion Host or External (Public) IP Address

Exploring Identity-Aware Proxy with App Engine - Demo

- 1: Deploy app to App Engine
- 2: Enable IAP
- 3: Configure OAuth consent screen
- 4: Setup IAP Access: Map members to IAP-secured Web App User role for the project
 - Members can be:
 - Google Account - you@gmail.com (Personal Accounts)
 - Google Group - your-group@googlegroups.com
 - Service account - server@example.gserviceaccount.com
 - G Suite domain - example.com (Corporate Accounts)
 - You can make a web site public by using member - **allUsers**
 - You can provide access to all authenticated users by mapping to member **allAuthenticatedUsers**



Identity-Aware Proxy (IAP) - How does it work?



- **Execution Flow:**
 - 1: When enabled, IAP intercepts calls to applications (sits before App Engine, Cloud Load Balancing)
 - 2: If user is unauthenticated
 - 2A: User is redirected to OAuth 2.0 Google Account sign-in flow
 - 3: If user is authenticated (Valid OAuth token is present with the request)
 - 3A: User authorization is verified (User's IAM role is checked against configured policy for the resource)
 - 3B: If successful, user is allowed access to the resource
- User identity is **passed to backend service** using HTTP headers
 - X-Goog-Authenticated-User-Email - Users email address
 - X-Goog-Authenticated-User-Id - Unique user identifier

Using Identity-Aware Proxy (IAP) with Kubernetes

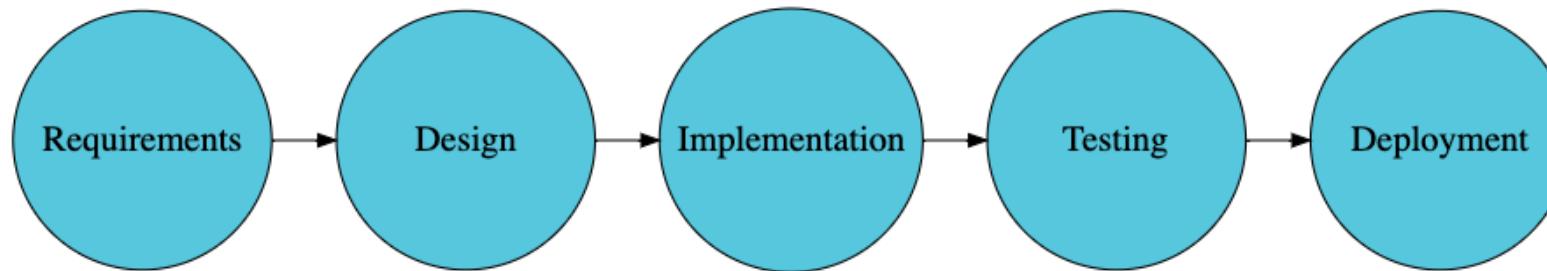
- **GKE cluster:** Traffic comes via HTTP(S) Load Balancing(Typically configured using Ingress)
- **Steps to setup IAP with GKE**
 - 1: Create Client Secret ID and Key
 - kubectl create secret generic my-client-secret --from-literal=client_id=CLIENT_ID_KEY --from-literal=client_secret=CLIENT_SECRET_KEY
 - 2: Configure BackendConfig using Secret
 - BackendConfig is typically used to customize your Kubernetes Ingress
 - 3: Configure K8S Service to use the BackendConfig
 - (ALTERNATIVE) Configure IAP with Anthos Service Mesh (also uses BackendConfig)

```
apiVersion: cloud.google.com/v1
kind: BackendConfig
metadata:
  name: enable-iap
  namespace: my-namespace
spec:
  iap:
    enabled: true
    oauthclientCredentials:
      secretName: my-client-secret
---
metadata:
  annotations:
    beta.cloud.google.com/backend-config: '{"default": "enable-iap"}'
```

Evolution

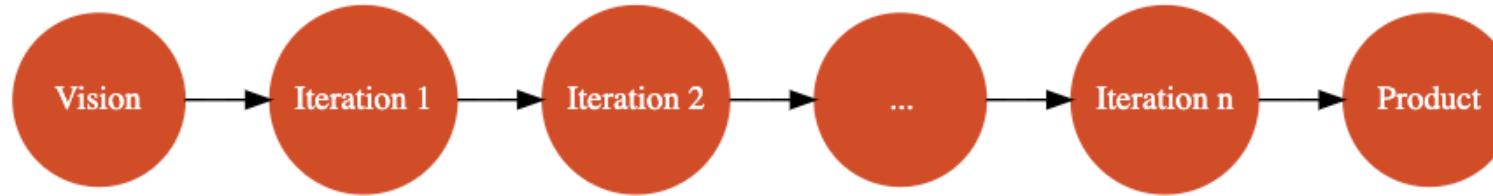
Agile > DevOps > SRE

Software development life cycle (SDLC) - Waterfall



- **Software development in multiple long phases:**
 - Requirements
 - Design
 - Implementation
 - Testing
 - Deployment

Software development life cycle (SDLC) - Spiral

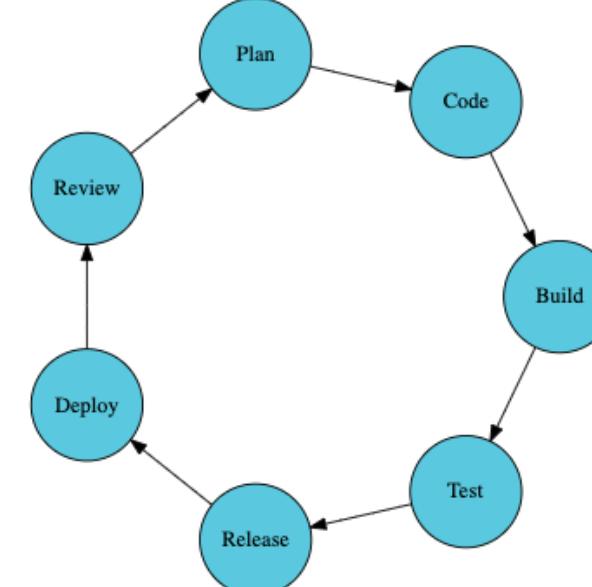


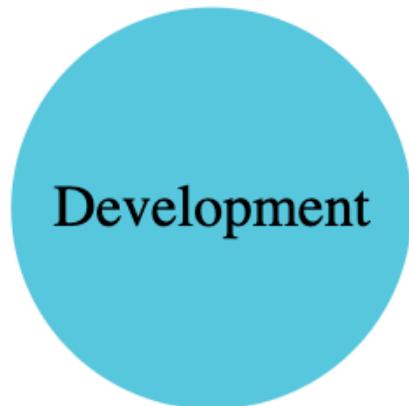
- **Software development in smaller iterations:**

- Start
- Iteration 1
- Iteration 2
- ...
- ...

Software development life cycle (SDLC) - Agile

- **Principles**
 - Individuals and interactions over processes and tools
 - Working software over comprehensive documentation
 - Customer collaboration over contract negotiation
 - Responding to change over following a plan
 - Now there are 12 principles (<https://agilemanifesto.org/principles.html>)
- **Agile is recommended for most software development:**
 - BUT add a bit of rigidity from waterfall model for critical safety software (Flight navigation software, Medical devices software etc)

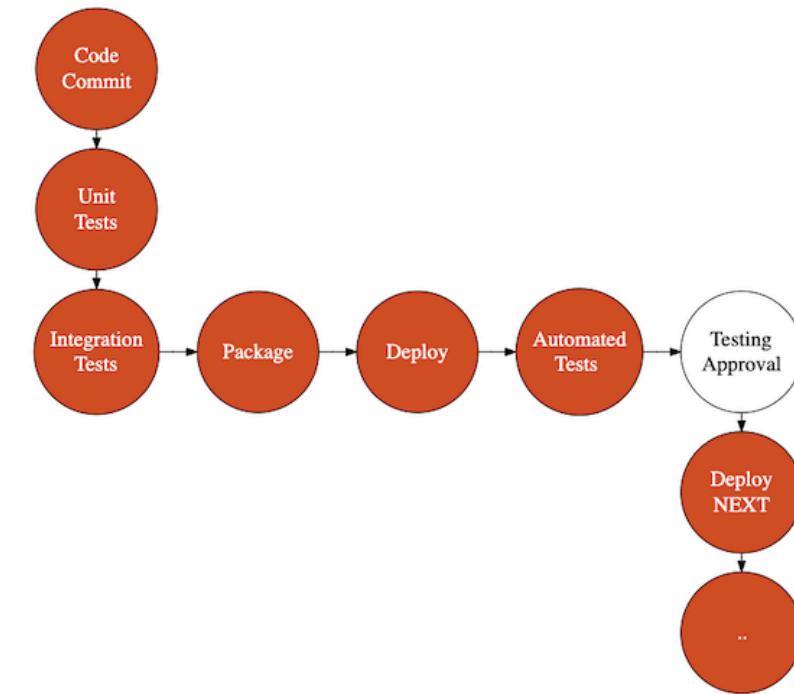




- Getting Better at "**Three Elements of Great Software Teams**"
 - **Communication** - Get teams together
 - **Feedback** - Earlier you find a problem, easier it is to fix
 - **Automation** - Automate testing, infrastructure provisioning, deployment, and monitoring

DevOps - CI, CD

- **Continuous Integration**
 - Continuously run your tests and packaging
- **Continuous Deployment**
 - Continuously deploy to test environments
- **Continuous Delivery**
 - Continuously deploy to production



DevOps - CI CD - Recommended Things to Do

- **Static Code Analysis**
 - Lint, Sonar
 - Including Static Security Checks (Source Code Security Analyzer software like Veracode or Static Code Analyzer)
- **Runtime Checks**
 - Run Vulnerability Scanners (automated tools that scan web applications for security vulnerabilities)
- **Tests**
 - Unit Tests (JUnit, pytest, Jasmine etc)
 - Integration Tests (Selenium, Robot Framework, Cucumber etc)
 - System Tests (Selenium, Robot Framework, Cucumber etc)
 - Sanity and Regression Tests

DevOps - CI, CD Tools

- **Cloud Source Repositories:** Fully-featured, private Git repository
 - Similar to Github
- **Container Registry:** Store your Docker images
- **Jenkins:** Continuous Integration
- **Cloud Build:** Build deployable artifacts (jars or docker images) from your source code and configuration
- **Spinnaker:** Multi-cloud continuous delivery platform
 - Release software changes with high velocity and confidence
 - Supports deployments to Google Compute Engine, Google Kubernetes Engine, Google App Engine and other cloud platforms
 - Supports Multiple Deployment Strategies



Container Registry



Cloud Source Repositories

DevOps - Infrastructure as Code

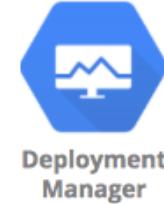
In 28
Minutes



- Treat **infrastructure the same way as application code**
- Track your infrastructure **changes over time** (version control)
- Bring **repeatability** into your infrastructure
- Two Key Parts
 - **Infrastructure Provisioning**
 - Provisioning compute, database, storage and networking
 - Open source cloud neutral - Terraform
 - GCP Service - Google Cloud Deployment Manager
 - **Configuration Management**
 - Install right software and tools on the provisioned resources

Google Cloud Deployment Manager - Introduction

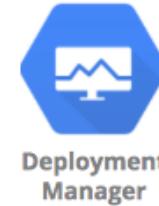
- Lets consider an example:
 - I would want to create a new VPC and a subnet
 - I want to provision a Load balancer, Instance groups with 5 Compute Engine instances and an Cloud SQL database in the subnet
 - I would want to setup the right Firewall
- AND I would want to create 4 environments
 - Dev, QA, Stage and Production!
- Deployment Manager can help you do all these with a simple (actually NOT so simple) script!



Deployment
Manager

Google Cloud Deployment Manager - Advantages

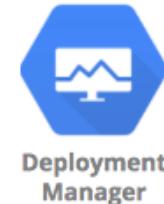
- Automate deployment and modification of Google Cloud resources in a controlled, predictable way
 - Deploy in multiple environments easily!
- Avoid configuration drift
- Avoid mistakes with manual configuration
- Think of it as version control for your environments
- **Important Note** - Always modify the resources created by Deployment Manager using Deployment Manager



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 automatically 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
```

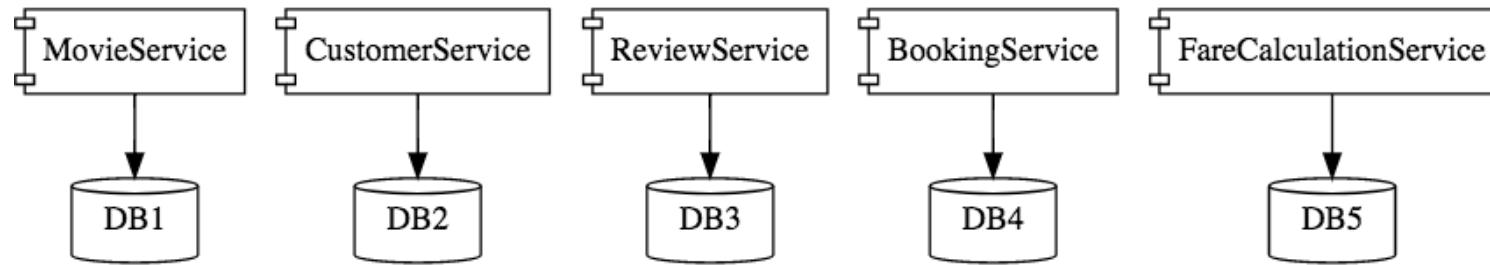
Cloud Deployment Manager - Terminology

- **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 software 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 software 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 - Software, infrastructure needed etc
 - Approximate price

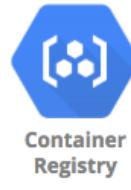
Exploring Container Registry and Artifact Registry



- You've created docker images for your microservices:
 - Where do you store them?
 - Two Options: Container Registry and Artifact Registry
 - Container Registry: Uses GCS bucket to store images
 - Supports Container images only
 - (Alternative) Docker Hub
 - Example: `us.gcr.io/PROJECT-ID/...`
 - Permissions are managed by managing access to GCS buckets
 - Artifact Registry: Evolution of Container Registry
 - Builds upon the capabilities of Container Registry
 - Manage BOTH container images and non-container artifacts
 - Example: `us-central1-docker.pkg.dev/PROJECT-ID/...`

Exploring Artifact Registry

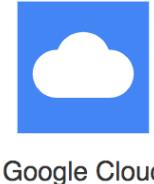
- Supports **multiple artifact formats**:
 - Container images, language packages, and OS packages are supported
- You need to create **separate repository**
 - Does NOT use GCS buckets
 - Repository can be regional or multi-regional
- Example: *us-central1-docker.pkg.dev/PROJECT-ID/...*
- (RECOMMENDED) Control access by using Artifact Registry Roles
 - Artifact Registry Reader
 - Artifact Registry Writer
 - Artifact Registry Administrator etc..
- You can also configure repository specific permissions



Container Registry

Site Reliability Engineering (SRE)

- DevOps++ at Google
- SRE teams **focus on every aspect of an application**
 - availability, latency, performance, efficiency, change management, monitoring, emergency response, and capacity planning
- **Key Principles:**
 - Manage by Service Level Objectives (SLOs)
 - Minimize Toil
 - Move Fast by Reducing Cost of Failure
 - Share Ownership with Developers



Site Reliability Engineering (SRE) - Key Metrics

- **Service Level Indicator(SLI):** Quantitative measure of an aspect of a service
 - Categories: availability, latency, throughput, durability, correctness (error rate)
 - Typically aggregated - "Over 1 minute"
- **Service Level Objective (SLO) - SLI + target**
 - 99.99% Availability, 99.99999999% Durability
 - Response time: 99th percentile - 1 second
 - Choosing an appropriate SLO is complex
- **Service Level Agreement (SLA):** SLO + consequences (contract)
 - What is the consequence of NOT meeting an SLO? (Defined in a contract)
 - Have stricter internal SLOs than external SLAs
- **Error budgets:** $(100\% - \text{SLO})$
 - How well is a team meeting their reliability objectives?
 - Used to manage development velocity

Site Reliability Engineering (SRE) - Best Practices

- **Handling Excess Loads**

- **Load Shedding**

- API Limits
 - Different SLAs for different customers
 - Streaming Data
 - If you are aggregating time series stream data, in some scenarios, you can drop a part of data



Google Cloud

- **Reduced Quality of Service**

- Instead of talking to a recommendations API, return a hardcoded set of products!
 - Not always possible:
 - Example: if you are making a payment

- **Avoiding Cascading Failures**

- **Plan to avoid thrashing**

- Circuit Breaker
 - Reduced Quality of Service

Site Reliability Engineering (SRE) - Best Practices - 2

In 28
Minutes

- **Penetration Testing (Ethical Hacking)**

- Simulate an attack with the objective of finding security vulnerabilities
- Should be authorized by project owners
- No need to inform Google
 - Ensure you are only testing your projects and are in compliance with terms of service!
- Can be white box (Hacker is provided with information about infrastructure and/or applications) or black box (No information is provided)



Google Cloud

- **Load Testing (JMeter, LoadRunner, Locust, Gatling etc)**

- Simulate real world traffic as closely as possible
- Test for spiky traffic - suddenly increases in traffic

Site Reliability Engineering (SRE) - Best Practices - 3

- **Resilience Testing** - "How does an application behaves under stress?"
- **Resilience** - "Ability of system to provide acceptable behavior even when one or more parts of the system fail"
- **Approaches:**
 - **Chaos Testing (Simian Army)** - cause one or more layers to fail
 - "unleashing a wild monkey with a weapon in your data center to randomly shoot down instances and chew through cables"
 - Add huge stress on one of the layers
 - **Include network in your testing (VPN, Cloud Interconnect etc..)**
 - Do we fall back to VPN if direct interconnect fails?
 - What happens when internet is down?
 - **Best Practice: DiRT** - disaster recovery testing at Google
 - Plan and execute outages for a defined period of time
 - Example: Disconnecting complete data center



Google Cloud

Cloud Build

Getting Started with Cloud Build

- Google Cloud's Serverless CI CD Platform
 - Deploy to multiple clouds
- **Typical steps** in a Cloud Build workflow:
 - **Build:** Build a deployable artifact (jar, war or docker image) for your application - Java, Go, Node.js, Python or ..
 - Pickup code from Cloud Source Repositories, GitHub, Bitbucket, ..
 - Push images to Container Registry
 - **Test:** Run automated unit tests or system tests
 - **Static Analysis:** Perform static analysis (coding standards, security standards etc) of your code
 - **Deploy:** Deploy to VMs, Google Kubernetes Engine, App Engine, Cloud Functions, Cloud Run etc



How does Cloud Build work?

- **Trigger** (Can be manual or automated)
 - Push to a branch
 - New Pub/Sub message
 - Manual invocation
- **Configuration:** 2 Options
 - 1: Cloud Build configuration file (YAML or JSON)
 - Build file name: Default - cloudbuild.yaml
 - Or choose your custom build file name in the trigger
 - Also present: Option to write inline YAML
 - 2: Dockerfile
 - Cloud Build will automatically build a Docker Image and push it to Artifact Registry.



Understanding Build Steps

```
steps:  
  - name: first  
    ...  
  - name: second  
    ...  
  - name: third  
    ...  
  - name: fourth  
    ...
```

- **Build configuration:** Define the tasks that should be done in Cloud Build
- Each build is a series of build steps
 - Each build step is run in a separate Docker container
- By default, all steps are executed one after the other!
- You have **flexibility to customize:**
 - Builder image for each step
 - Order of step execution (and parallelization)

Exploring Cloud Build Configuration - Example

```
steps:  
  - name: 'gcr.io/cloud-builders/docker'  
    args: ['build', '-t',  
           'us-central1-docker.pkg.dev/$PROJECT_ID/$REPO_NAME/hello-world-python:$SHORT_SHA', '.']  
  - name: 'gcr.io/cloud-builders/docker'  
    args: ['push', 'us-central1-docker.pkg.dev/$PROJECT_ID/$REPO_NAME/hello-world-python:$SHORT_SHA']  
  - name: 'google/cloud-sdk'  
    args: ['gcloud', 'run', 'deploy', 'hello-world-python',  
           '--image=us-central1-docker.pkg.dev/$PROJECT_ID/$REPO_NAME/hello-world-python:$SHORT_SHA',  
           '--region', 'us-central1', '--platform', 'managed',  
           '--allow-unauthenticated']  
  
images:  
  - us-central1-docker.pkg.dev/$PROJECT_ID/$REPO_NAME/hello-world-python:$SHORT_SHA
```

- **name** - Which builder do you want to use?
 - gcr.io/cloud-builders/docker or gcr.io/cloud-builders/gcloud or ...
- **args** - What commands do you want to execute?
 - Passed as arguments
 - Example: name: 'gcr.io/cloud-builders/mvn' args: ['install']

Controlling Order of Cloud Build Steps

```
steps:  
- name: first  
  id: A  
- name: second  
  id: B  
  waitFor: ['-']  
- name: third  
  waitFor: B  
- name: fourth
```

- **Default:** If `waitFor` is not specified, a step waits for all the previously defined steps to complete
 - Example: fourth step
- **`waitFor: ['-']`:** Start immediately at the start of the build
 - second step starts immediately
- **`waitFor: SPECIFIC-ID`:** Starts after the execution of specific step
 - Step third starts only after second step completes execution

Choosing Builder Image for Each Cloud Build Step

- You can choose between:
 - **Google Cloud Build official builder images:**
 - curl, docker, gcs-fetcher (fetches objects from GCS), git, gke-deploy, go, gradle, gsutil, kubectl, mvn, npm, wget, yarn etc
 - **Google Cloud Build community images:**
 - <https://github.com/GoogleCloudPlatform/cloud-builders-community>
 - To use one of these, you must first build the image and push it to Container Registry in your project
 - **Create your custom builder:**
 - When you want to perform a task (or use a build tool) not supported by official or community builders
 - Create a Dockerfile, build the image and push it to Container Registry in your project



Understanding Flexibility of Cloud Build Configuration

- Important Configuration Options

- **env** - Environment variables
- **dir** - Working directory (/workspace/<dir>)
- **timeout** - Max execution time for the step
- **waitFor** - Which steps should complete before this step is executed?
 - Default (when no value is provided): Executes after all previous steps are run
- **logsBucket** - Which Cloud Storage bucket should logs be written to?
- **machineType** - Default machine type is 1 CPU
 - You can request for high CPU machines (N1_HIGHCPU_8, N1_HIGHCPU_32, E2_HIGHCPU_8, E2_HIGHCPU_32)
- **logging**: Where should your logs go to?
 - Default (when no value is provided): both Cloud Logging and Cloud Storage
 - GCS_ONLY: Store only to Cloud Storage
- **images**: Docker images to be pushed
- **artifacts**: Non-container artifacts to store in Cloud Storage

Exploring Cloud Build Configuration - Substitutions

```
substitutions:  
  _NODE_VERSION_1: v6.9.1 # default value  
  _NODE_VERSION_2: v6.9.2 # default value
```

- Following are some of the supported substitutions:
 - \$PROJECT_ID: ID of your Cloud project
 - \$BUILD_ID: ID of your build
 - For trigger based builds:
 - \$TRIGGER_NAME: the name associated with your trigger
 - \$COMMIT_SHA: the commit ID associated with your build
 - \$REVISION_ID: the revision ID associated with your build
 - \$REPO_NAME: the name of your repository
 - \$BRANCH_NAME: the name of your branch
 - \$TAG_NAME: the name of your tag
 - You can define your own substitutions:
 - node_version=\${_NODE_VERSION_1}
 - (DEMEMBERED) User defined substitutions must begin with an underscore ('')

Sharing Files between Cloud Build Steps

```
steps:  
  - name: first-step-using-a-shared-volume  
    volumes:  
      - name: 'myvolume'  
        path: '/persistent_volume'  
  - name: another-step-using-a-shared-volume  
    volumes:  
      - name: 'myvolume'  
        path: '/persistent_volume'
```

- Containers are discarded after each step
- **What if you want to share files between steps?**
- Source for a build execution is mounted under the directory: `/workspace`
- Use `/workspace` folder to share files between build steps:
 - Files created in `/workspace` are available to next build steps
 - (ALTERNATIVE) Create and use Docker Volumes

Managing Permissions for Cloud Build - Service Account

- Service account used to execute your Cloud Build builds
 - Email: [PROJECT_NUMBER]@cloudbuild.gserviceaccount.com
 - Default permissions:
 - cloudbuild.builds.* - Create, list, get, or cancel builds (and triggers)
 - storage.(objects|buckets).(create|get|list|*) - Store artifacts /images /logs to Cloud Storage/Container Registry
 - artifactregistry.* - Store and get artifacts in Artifact Registry
 - logging.logEntries.create - Write logs to Cloud Logging
 - pubsub.topics.(create|publish) - Push build updates to Pub/Sub
 - resourcemanager.projects.(get|list) - Get project info
 - source.repos.(get|list) - Pull source code and for triggers
- Add additional permissions on the Cloud Build Settings page
 - Give access to Cloud Functions, Cloud Run, App Engine, Kubernetes Engine, Compute Engine, Cloud KMS, Secret Manager etc.



Deploying to Kubernetes - Cloud Build

- 1: Build Container Image for Your App
- 2: Push Container image to Artifact Registry
- 3: Deploy to Kubernetes Cluster:
 - **kubernetes-resource-file:** Kubernetes resource file or folder path with resource files
 - **cluster:** Which cluster do you want to deploy to?
 - **location:** Which zone/region is your cluster in?
 - Ensure that **Kubernetes Engine Developer role** is Enabled in Cloud Build Settings

```
steps:  
- name: 'gcr.io/cloud-builders/docker'  
  args: ['build', '-t',  
        'us-central1-docker.pkg.dev/$PROJECT_ID/hello-world-python:$SHORT_SHA', '.']  
- name: 'gcr.io/cloud-builders/docker'  
  args: ['push', 'us-central1-docker.pkg.dev/$PROJECT_ID/hello-world-python:$SHORT_SHA']  
- name: "gcr.io/cloud-builders/gke-deploy"  
  args:  
    - run  
      - --filename=kubernetes-resource-file  
      - --image=us-central1-docker.pkg.dev/$PROJECT_ID/hello-world-python:$SHORT_SHA  
      - --location=location  
      - --cluster=cluster
```

Exploring Cloud Build Best Practices

- Use/build **lean container images** (builder/application)
 - You need to pay for storage and it takes time to pull and push images
- When a cached image can be used for subsequent builds, use -**-cache-from** argument (only for Docker images)
- **Use Google Cloud Storage to cache files** (other than container images) between builds
 - Recommended when your build takes a long time and produces a small number of files (copying to/from GCS does not take long time)
- **Use custom virtual machine sizes** (`machineType`) to increase speed of your builds
 - Default machine type is 1 CPU. You can request for high CPU machines (`N1_HIGHCPU_8`, `N1_HIGHCPU_32`, `E2_HIGHCPU_8`, `E2_HIGHCPU_32`)



Exploring Cloud Build Best Practices - 2

- Consider **Kaniko cache** for caching build artifacts
 - Stores and indexes intermediate layers within a container image registry
- **Exclude files not needed** for build by creating a **.gcloudignore** file
- **Features to remember:**
 - You can **build custom VM images** using Packer
 - Use Packer community builder image
 - You can **schedule builds**
 - Create a build with manual trigger
 - Schedule it using Cloud Scheduler
 - Run your **build locally** using the `cloud-build-local` tool



Getting Started with Spinnaker

In 28
Minutes

- Cloud agnostic CI/CD Platform developed at Netflix
- Manage your **global** deployments across multiple clouds
- Install it where ever you want (**on-premise or in any of the clouds**)
 - For example: you can install a production-ready instance of Spinnaker on GKE
- **Setup CI/CD pipelines** that can deploy to:
 - Google Cloud Services
 - App Engine
 - Google Compute Engine
 - Google Kubernetes Engine
 - Or other cloud platforms (Azure, AWS, etc..)
- Supports **Advanced Deployment strategies** such as blue/green and canary deployments

Getting Started with Tekton Pipelines

- **Kubernetes style CI/CD pipelines**
 - Runs as an extension on a Kubernetes cluster
- **Introduces five new Kubernetes resources:**
 - Tasks, Pipelines, TaskRuns, PipelineRuns and PipelineResources
 - Pipelines consist of multiple Tasks
 - PipelineRuns execute Pipelines (TaskRuns execute Tasks)
 - PipelineResources are used to share resources between Tasks
- **Can be accessed like any other Kubernetes resources**
 - Through Kubernetes configurations files and kubectl
- **Features:**
 - Deployment patterns like rolling, blue/green, canary deployment
 - Run on hybrid or multi-cloud
- **Installation and Usage:** <https://github.com/tektoncd/pipeline>



Kubernetes Engine

Cloud Build - Scenarios

In 28
Minutes

Scenario	Solution
Use a cached Docker image from one build in the next build	--cache-from
How do you ensure that static analysis (coding standards, security standards etc) of your code is regularly performed?	Integrate it into your CI build
You have created your own custom builder. You are making use of it in your Cloud Build configuration. However you see that the build is unable to access your custom builder image.	Create a Dockerfile, build the image and push your custom builder image to Container Registry in your project
You want to share files between build steps	Create them in /workspace folder (or use shared Docker volumes)

Cloud Build - Scenarios - 2

In 28
Minutes

Scenario

You are using default machine type. You want to increase the speed of your build.

You want to use a cloud agnostic CI/CD Platform that supports Advanced Deployment strategies such as blue/green and canary deployments

Solution

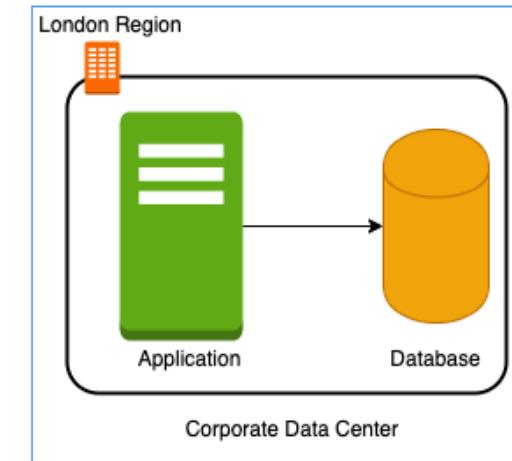
Default machine type is 1 CPU. You can request for high CPU machines (N1_HIGHCPU_8, N1_HIGHCPU_32, E2_HIGHCPU_8, E2_HIGHCPU_32)

Spinnaker

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)

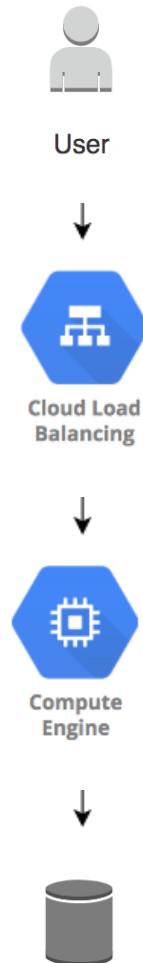
- Your own **isolated network** in GCP cloud
 - Network traffic within a VPC is isolated (not visible) from all other Google Cloud VPCs
- You **control all the traffic** coming in and going outside a VPC
- **(Best Practice)** Create all your GCP resources (compute, storage, databases etc) **within a 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!



Virtual Private
Cloud

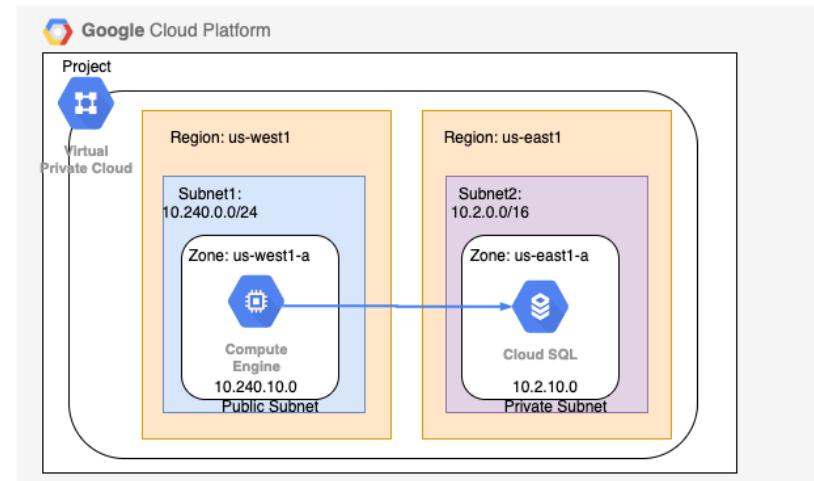
Need for VPC Subnets

- 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



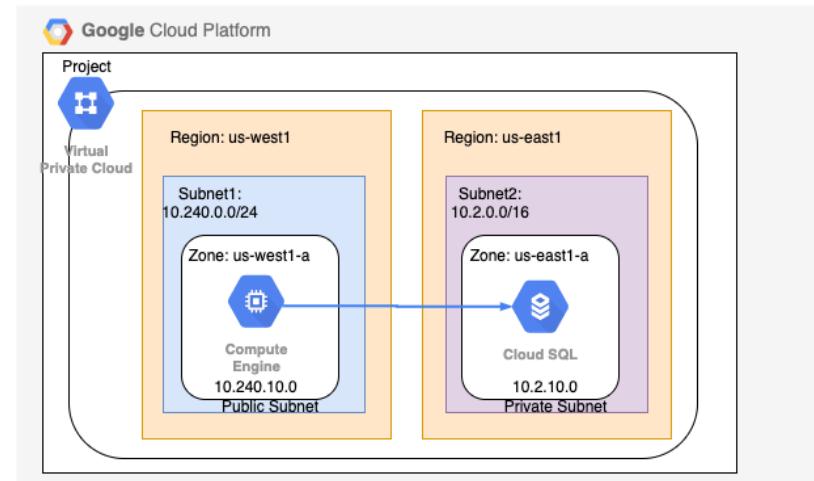
VPC Subnets

- (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 => $2^{24} = 16$ addresses

CIDR Exercises

CIDR	Start Range	End Range	Total addresses	Bits selected in IP address
69.208.0.0/24	69.208.0.0	69.208.0.255	256	01000101.11010000.00000000.*****
69.208.0.0/25	69.208.0.0	69.208.0.127	128	01000101.11010000.00000000.0*****
69.208.0.0/26	69.208.0.0	69.208.0.63	64	01000101.11010000.00000000.00*****
69.208.0.0/27	69.208.0.0	69.208.0.31	32	01000101.11010000.00000000.000****
69.208.0.0/28	69.208.0.0	69.208.0.15	16	01000101.11010000.00000000.0000***
69.208.0.0/29	69.208.0.0	69.208.0.7	8	01000101.11010000.00000000.00000**
69.208.0.0/30	69.208.0.0	69.208.0.3	4	01000101.11010000.00000000.00000**
69.208.0.0/31	69.208.0.0	69.208.0.1	2	01000101.11010000.00000000.000000*
69.208.0.0/32	69.208.0.0	69.208.0.0	1	01000101.11010000.00000000.0000000

- Exercise : How many addresses does **69.208.0.0/26** represent?
 - $2 \text{ to the power } (32-26 = 6) = 64$ addresses from 69.208.0.0 to 69.208.0.63
- Exercise : How many addresses does **69.208.0.0/30** represent?
 - $2 \text{ to the power } (32-30 = 2) = 4$ 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)



Virtual Private
Cloud

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**



Virtual Private
Cloud

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



Virtual Private
Cloud

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



Virtual Private
Cloud

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



Virtual Private
Cloud

Hybrid Cloud

Cloud VPN

In 28
Minutes

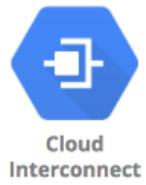
- Cloud VPN - Connect on-premise network to the GCP network
 - Implemented using **IPSec VPN Tunnel**
 - Traffic through internet (public)
 - Traffic encrypted using **Internet Key Exchange** protocol
- Two types of Cloud VPN solutions:
 - HA VPN (SLA of 99.99% service availability with two external 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 VPN

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 Gbps 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



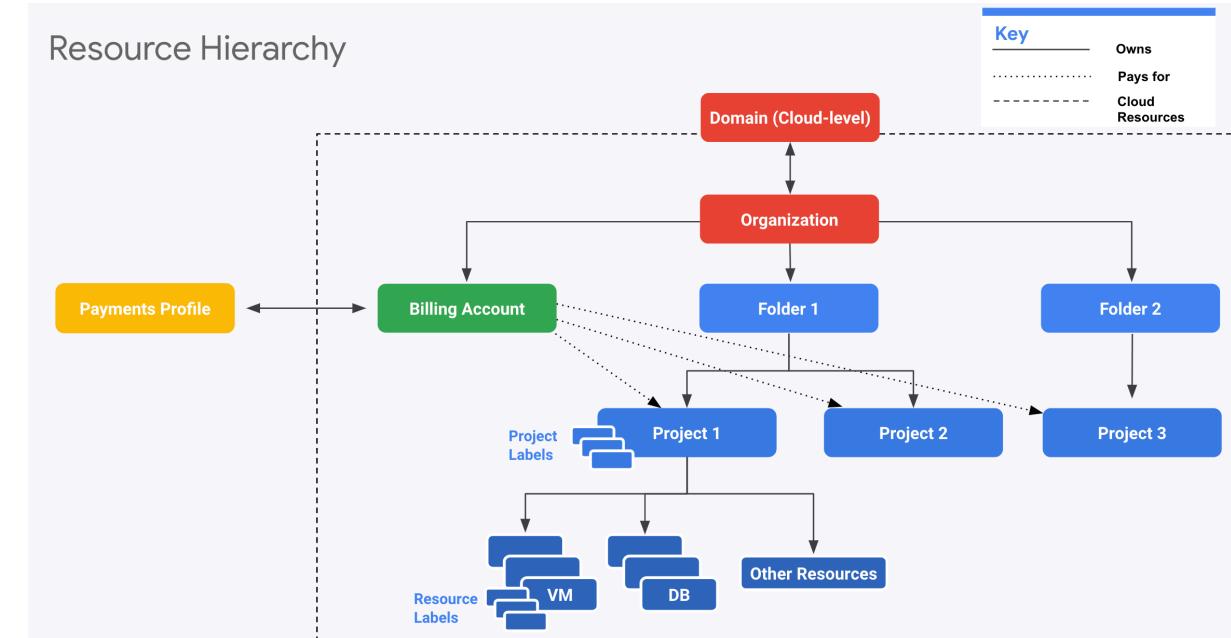
Direct Peering

- 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

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>)

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)

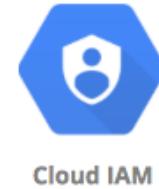
Managing Billing - Budget, Alerts and Exports

- 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)

- **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 shift traffic
 - App Engine Service Admin can shift 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



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

In 28
Minutes

Scenario	Solution
All members in your team have G Suite accounts. You are creating a new production project and would want to provide access to your operations team	Create a Group with all your operations team. Provide access to production project to the Group.
All members in your team have G Suite accounts. You are setting up a new project. You want to provide a one time quick access to a team member.	Assign the necessary role directly to G Suite email address of your 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 all resources in your project BUT he should NOT be able to make any changes	Give them roles/viewer role (Generally basic roles are NOT recommended BUT it is the simplest way to provide view only access to all resources!)
Your application deployed on a GCE VM (Project A) needs to access cloud storage bucket from a different project (Project B)	In Project B, assign the right role to GCE VM service account from Project A

Organization Policy Service

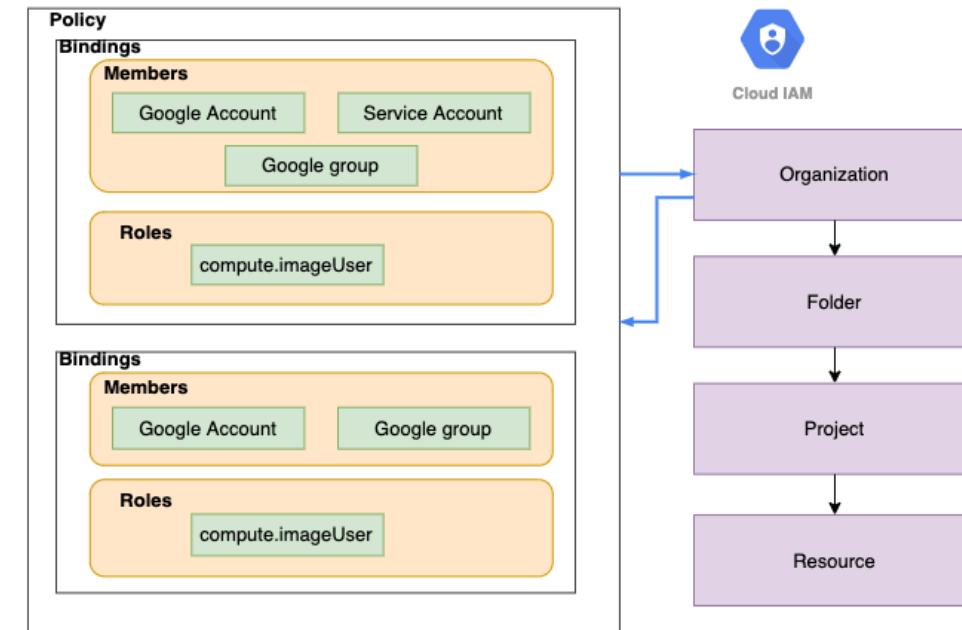
- How to enable **centralized constraints** on all resources created in an Organization?
 - Configure Organization Policy
 - Example: Disable creation of Service Accounts
 - Example: Allow/Deny creation of resources in specific regions
- Needs a Role - Organization Policy Administrator
- (Remember) IAM focuses on **Who**
 - Who can take specific actions on resources?
- (Remember) Organization Policy focuses on **What**
 - What can be done on specific resources?



Cloud IAM

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



Google Cloud APIs and Client Libraries

Designing APIs: gRPC vs REST

- Two Popular Approaches: REST and gRPC
 - REST: POST /todos (username, description) -> todo_URL
 - Addressable entities - **resources**
 - Create, delete and modify resources to create desired behavior
 - gRPC: Based on Remote Procedure Call (RPC) model
 - createTodo(username, description) -> todo_id
 - Addressable entities - **procedures**
 - **Advantage: Hides HTTP details** from developers (uses HTTP/2 in background)
 - Generate gRPC stubs and skeletons
 - Clients use stubs to make gRPC calls
 - Service providers use skeletons to implement services
 - **Advantage: Provides very good performance** (sometime upto 10x better)
 - Use **binary payload** (more efficient) - Protobuf (Protocol Buffers) is most commonly used
 - Efficient management of connections (because of HTTP/2)
 - (REMEMBER) REST can also use binary payload and HTTP/2 **BUT** these features are built into gRPC by default
 - **Disadvantage:** REST API are easier to consume
 - REST API does NOT need any special frameworks/libraries
 - Just do a cURL or execute a call to a URL from your code

Exploring Client Libraries

- GCP provides a number of services
- Services can be accessed via console, gcloud etc
- **How do you make programmatic calls to GCP services?**
 - Google Cloud APIs are provided for most GCP services
 - 1: Directly call them (supports JSON REST and gRPC) or
 - 2: Use client libraries which are provided by google
 - A: Google API Client Libraries
 - B: Google Cloud Client Libraries
- Google API Client Libraries vs Google Cloud Client Libraries
 - **Google API Client Libraries:** OLDER
 - Supports JSON REST interface
 - **Google Cloud Client Libraries:** NEWER and RECOMMENDED
 - Available for MOST Google Cloud APIs
 - Supports JSON REST and gRPC
 - Provides better performance and usability

Google Cloud APIs - Remember

- Enable you to make programmatic calls to GCP services
- REMEMBER: Need to be enabled before use
- Accept only secure requests using TLS encryption
 - Automatically taken care of while using Client Libraries
- Google Cloud Console API Dashboard: Provides information about API usage
 - traffic levels, error rates, and latencies
- API Library: [*https://console.cloud.google.com/apis/library*](https://console.cloud.google.com/apis/library)
- Google APIs Explorer: [*https://developers.google.com/apis-explorer*](https://developers.google.com/apis-explorer)

Using Cloud Client Libraries - Cloud Storage

- 1: Setup Cloud Client Libraries for the Google Cloud Storage API
 - Java: Add to pom.xml or gradle dependencies
 - compile 'com.google.cloud:google-cloud-storage'
 - Node.js: npm install --save @google-cloud/storage
 - Python: pip install --upgrade google-cloud-storage
- 2: Setup Authentication
 - Create service account with the right permissions
 - Make your application use the Service Account
 - (If app running in Google Cloud) Assign the service account to your compute (App Engine, Cloud Run, Compute Engine, Cloud Functions or GKE)
 - OR Create service-account key and make the service account key available as part of your GOOGLE_APPLICATION_CREDENTIALS
 - export GOOGLE_APPLICATION_CREDENTIALS="PATH_TO_SERVICE_ACCOUNT_KEY_FILE"
- 3: Write the code



Using Cloud Client Libraries - Cloud Storage - Code

```
//Java
Storage storage = StorageOptions.getDefaultInstance().getService();
Bucket bucket = storage.create(BucketInfo.of("my-first-bucket-with-unique-name"));
System.out.printf(bucket.getName());

//Node.js
const {Storage} = require('@google-cloud/storage');
const storage = new Storage();
await storage.createBucket("my-first-bucket-with-unique-name");
console.log(`Bucket created`);

//Python
from google.cloud import storage
storage_client = storage.Client()
bucket = storage_client.create_bucket("my-first-bucket-with-unique-name")
print("Bucket {} created.".format(bucket.name))
```

- More API Details
 - Java: <https://googleapis.dev/java/google-cloud-storage/latest/index.html>
 - Nodejs: <https://googleapis.dev/nodejs/storage/latest/Storage.html>
 - Python: <https://googleapis.github.io/google-cloud-python/latest/storage/client.html>

Consuming Google Cloud APIs - Remember

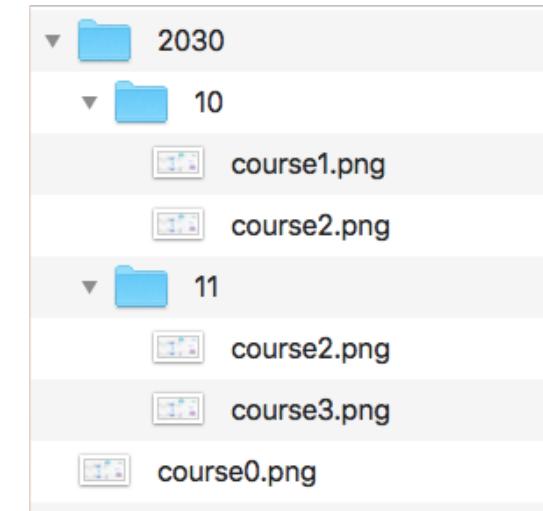
- **How does authentication for Client Libraries work?**
 - 1: Client Libraries look for env variable GOOGLE_APPLICATION_CREDENTIALS
 - If present, the service account tagged is used
 - 2: Use service account attached to the resource where your code is running
 - ELSE: Throw error
 - (RECOMMENDED) If you are using Compute Engine, Google Kubernetes Engine, App Engine, Cloud Run, or Cloud Functions to run your app:
 - Assign Service Account to the resource
- **Be careful when scaling up too fast:**
 - Starting capacity for a new bucket: 1000 object write requests per second, 5000 object read requests per second
 - Take atleast 20 minutes to double request rates
 - Otherwise, you might get higher latency and error rates
 - Retry with Exponential Backoff!

What should you do when an API errors out?

HTTP	gRPC	Description
200	OK	Success
400	INVALID_ARGUMENT	Problem with the client request. Validation failed. Fix the request.
401	UNAUTHENTICATED	Problem with authentication - missing, invalid, or expired OAuth token
403	PERMISSION_DENIED	Client does not have sufficient permission
404	NOT_FOUND	Resource does not exist
429	RESOURCE_EXHAUSTED	Exceeding quota or rate limited. Suggested to retry with exponential backoff - 30s delay
500	INTERNAL	Internal server error
503	UNAVAILABLE	Service unavailable. Suggested to retry with exponential backoff - 1s delay

Exploring Flat Namespace - Cloud Storage

Key	Value
2030/course0.png	image-binary-content
2030/10/course1.png	image-binary-content
2030/10/course2.png	image-binary-content
2030/11/course2.png	image-binary-content
2030/11/course3.png	image-binary-content



Cloud Storage - Flat Namespace - 2

```
from google.cloud import storage

client = storage.Client(project='YOUR_PROJECT')
bucket = client.get_bucket('your-bucket')
blob = bucket.blob('your/folder/name/')

blob.upload_from_string('',
    content_type='application/x-www-form-urlencoded; charset=UTF-8')
```

- Cloud Storage uses a flat namespace
 - Internally there is NO folder structure
 - **How is an empty folder represented?**
 - A zero-byte object is created as a placeholder
 - **Example:** If you create a folder my-folder in a bucket my-bucket, a zero-byte object is created with name gs://my-bucket/my-folder/
 - **How to create empty folder?**
 - In Cloud Console
 - `gsutil cp empty-file gs://your-bucket/empty-folder/`

Exploring Cloud Storage Scenarios

Scenario	Solution
I will frequently access objects in a bucket for 30 days. After that I don't expect to access objects at all. We have compliance requirements to store objects for 4 years. How can I minimize my costs?	Initial Storage Class - Standard Lifecycle policy: Move to Archive class after 30 days. Delete after 4 years.
Customer wants to manage their Keys	Customer-managed - Keys managed by customer in Cloud KMS
Regulatory compliance: Object should not be modified for 2 years	Configure and lock data retention policy

Database Fundamentals

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...



Cloud SQL



Cloud Spanner



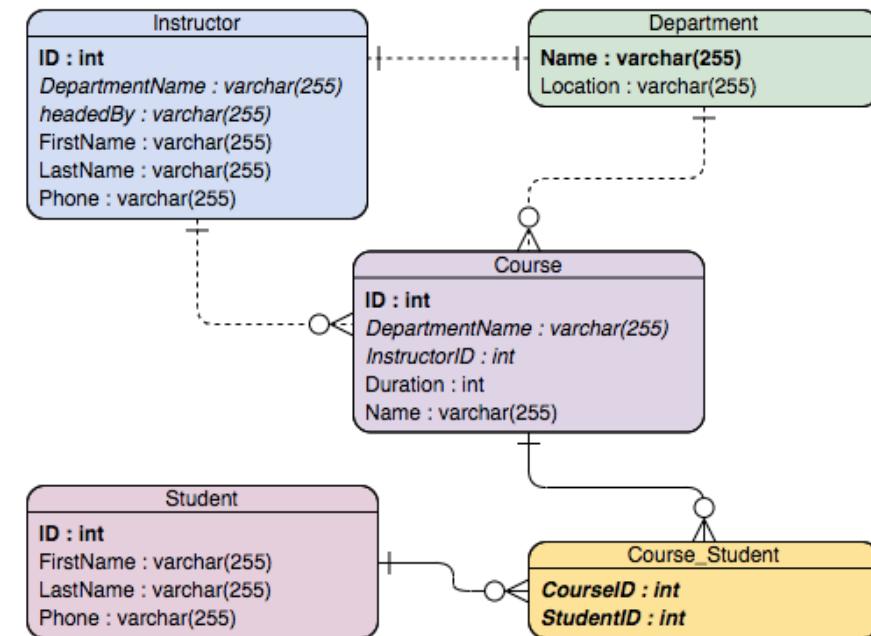
Cloud Datastore



BigQuery

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)

In 28
Minutes

- 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



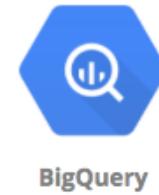
Cloud SQL



Cloud Spanner

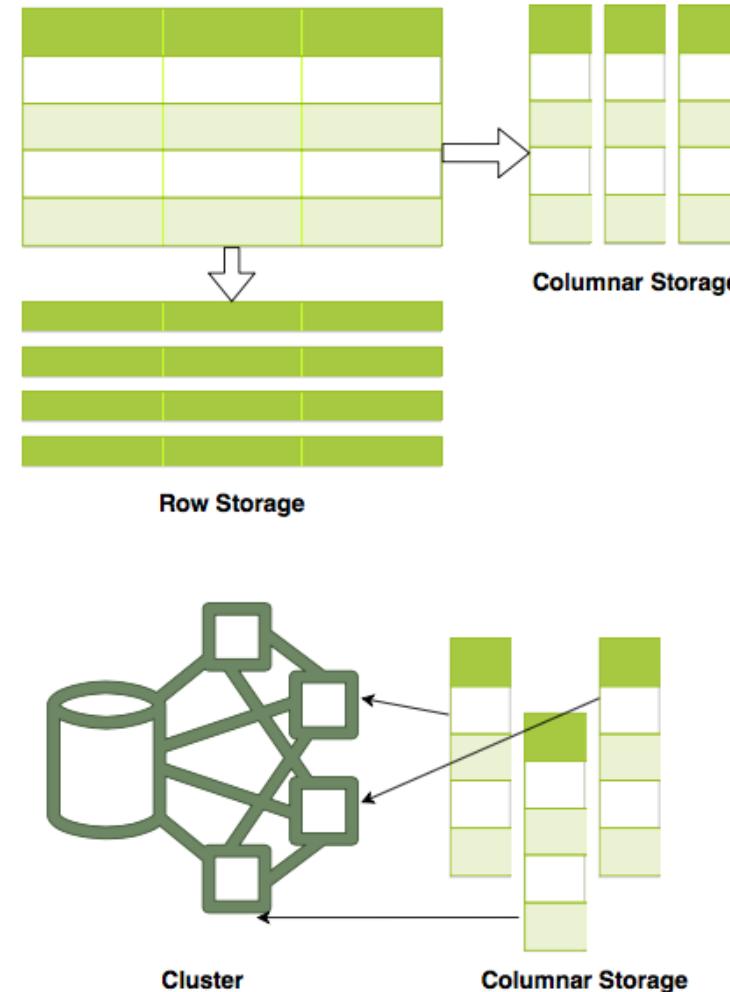
Relational Database - OLAP (Online Analytics Processing)

- 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



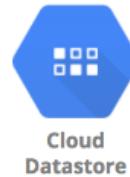
Relational Databases - OLAP vs OLTP

- 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



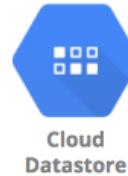
NoSQL Databases

- **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



Memorystore

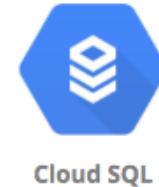
Databases - Summary

Database Type	GCP Services	Description
Relational OLTP databases	Cloud SQL, Cloud Spanner	Transactional usecases needing predefined schema and very 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 databases	BigQuery	Columnar storage with predefined schema. Datawarehousing & BigData workloads
NoSQL Databases	Cloud Firestore (Datastore) , Cloud BigTable	Apps needing quickly evolving structure (schema-less) Cloud Firestore - Serverless transactional document DB 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 - Scenarios

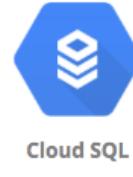
Scenario	Solution
A start up with quickly evolving schema (table structure)	Cloud Datastore/Firestore
Non relational db with less storage (10 GB)	Cloud Datastore
Transactional global database with predefined schema needing to process million of transactions per second	CloudSpanner
Transactional local 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

- **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)
 - Up to 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%)



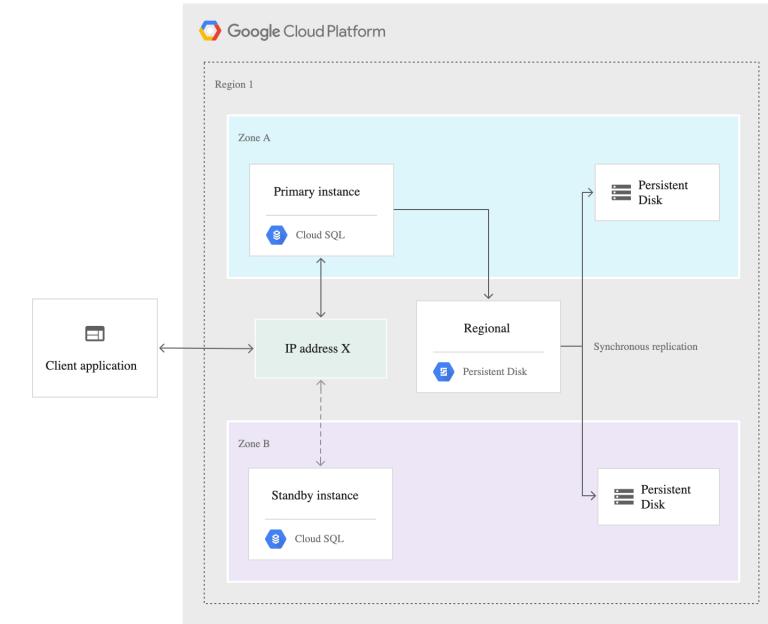
Cloud SQL - Features

- **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

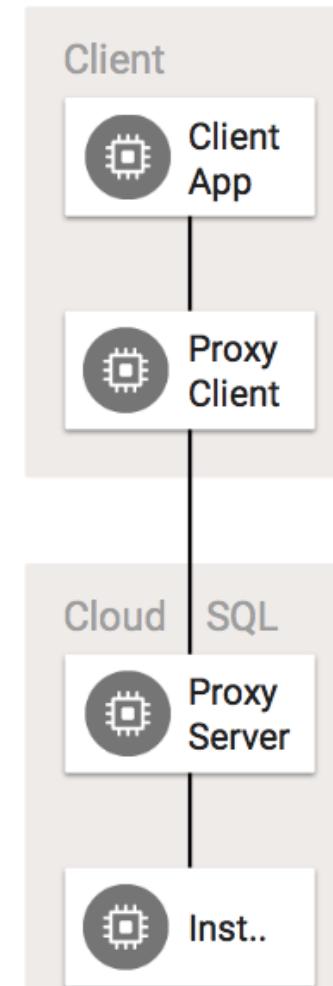


source:cloud.google.com

Connecting with Cloud SQL

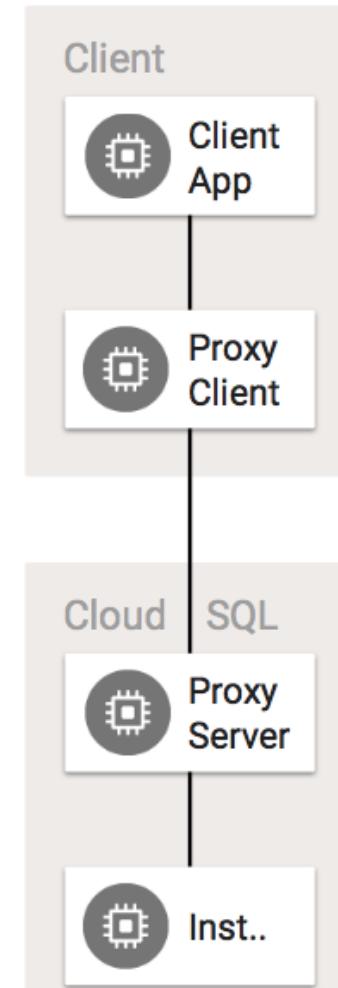
Getting Started with Cloud SQL Auth proxy

- How do you **simplify connection management** and ensure **secure connection (HTTPS)** between clients and Cloud SQL?
 - Cloud SQL Auth proxy
- Communication encrypted with TLS 1.2 with a 128-bit AES cipher
 - Do NOT worry about certificates
- Easier connection management
- Needs installation of Proxy Client on the client machine
 - Can be used to connect from local environment
- Works with both private and public IP addresses
- Cloud SQL Auth proxy is **automatically supported on some GCP services**
 - Example: App Engine Standard, Cloud Run etc



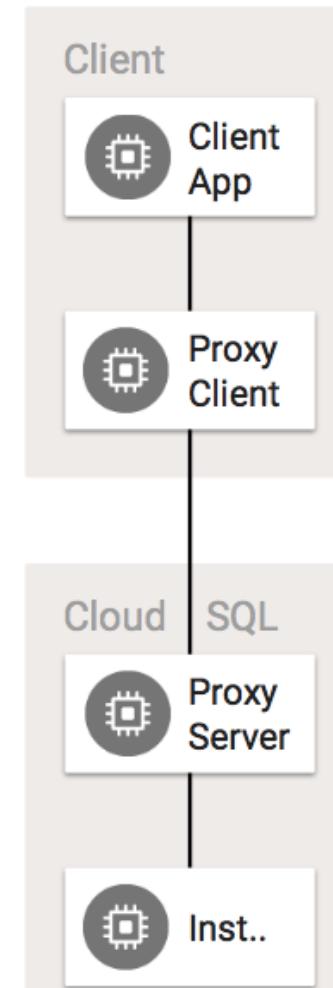
Configuring Cloud SQL Auth proxy - Step By Step

- 1: Enable *Cloud SQL Admin API*
- 2: *Install Cloud SQL Auth Proxy on Client*
 - Ex: wget https://dl.google.com/cloudsql/cloud_sql_proxy.linux.amd64 -O cloud_sql_proxy
- 3: *Start Cloud SQL Auth proxy*
 - chmod +x ./cloud_sql_proxy
 - Ex: ./cloud_sql_proxy -instances=INSTANCE_CONN_NAME=tcp:3306
 - Setup Cloud SQL Auth proxy to connect to Cloud SQL Instance using INSTANCE_CONN_NAME
 - On GKE, you can run Cloud SQL Auth Proxy as a side car container
 - On App Engine and Cloud Run, you can enable it with simple configuration
- 4: *Connect to the Cloud SQL Auth Proxy*
 - Ex: mysql -u USERNAME -p --host 127.0.0.1
 - Cloud SQL Auth proxy is available at **127.0.0.1** on the client



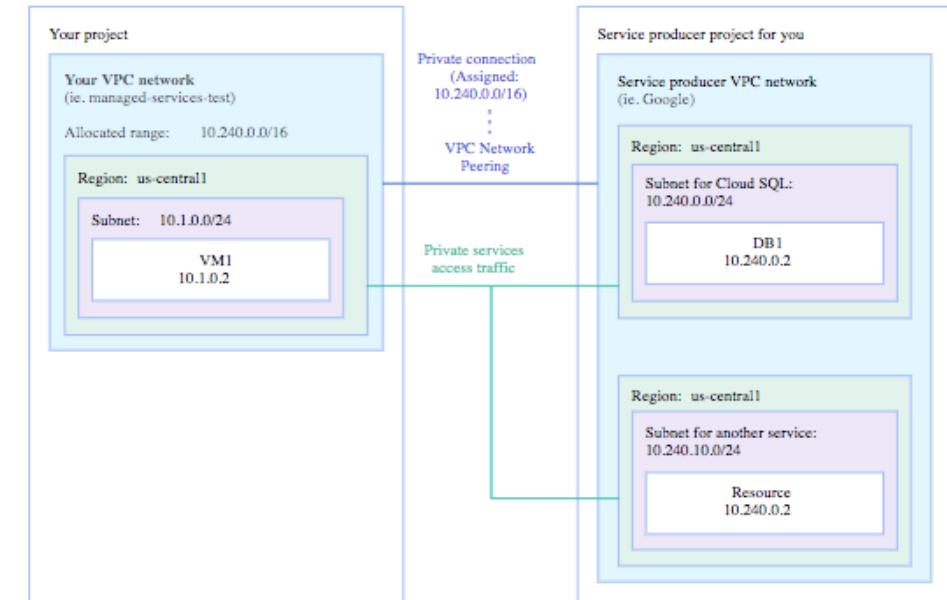
Configuring Authorization for Cloud SQL Auth proxy

- Cloud SQL Auth Proxy needs to connect to Cloud SQL:
 - Recommended Role: Cloud SQL Client
 - Other Roles: Cloud SQL Editor or Cloud SQL Admin
 - OR assign these IAM permissions manually to your role:
 - `cloudsql.instances.connect`, `cloudsql.instances.get`
 - Assign the role to your compute env. service account
 - OR Create a new service account (with key) and use it to launch Cloud SQL Auth Proxy
 - `./cloud_sql_proxy -instances=INSTANCE_CONN_NAME=tcp:3306 -credential_file=PATH_TO_KEY_FILE`
- Your application needs to connect to the database:
 - Configure credentials in application config:
 - `db_user = os.environ["DB_USER"]`
 - `db_pass = os.environ["DB_PASS"]`
 - `db_name = os.environ["DB_NAME"]`
 - In Kubernetes, you can use Kubernetes secrets for storing credentials
 - Or Secrets manager



Getting Started with Private Service Connection

- How to connect from a VM instance **inside a VPC** to a Cloud SQL instance using private address?
 - Create a "Private Service Connection"
 - A **Private Connection** between your VPC and network owned by Google:
 - Enables VM instances in your VPC network to talk to GCP services using internal IP addresses
 - *Example:* Connect from your VM instance to a Cloud SQL instance using private address
 - Examples of supported GCP Services: Memorystore, Tensorflow and Cloud SQL
- **Creating Private Service Connection:**
 - 1: Allocate IP address range for subnet
 - 2: Create private conn. to service producer



Getting Started with Serverless VPC Access

- (REMEMBER) Serverless environments **automatically** use Cloud SQL proxy when using public IP addresses!
 - Example: Cloud Functions, Cloud Run and App Engine standard environment
 - But things are different when using Cloud SQL private IP addresses!
 - How do you connect from your serverless environments to a resource in a VPC network using **private IPs**?
 - *Serverless VPC Access*
- **Serverless VPC Access** allows Cloud Functions, Cloud Run (fully managed) services and App Engine standard environment apps to access resources in a VPC network using those resources' private IPs
 - You can connect to these resources using internal IP addresses
 - GCE VMs
 - Cloud SQL instances
 - Cloud Memorystore instances
 - Kubernetes Pods/Services (on GKE public or private clusters)
 - Internal Load Balancers

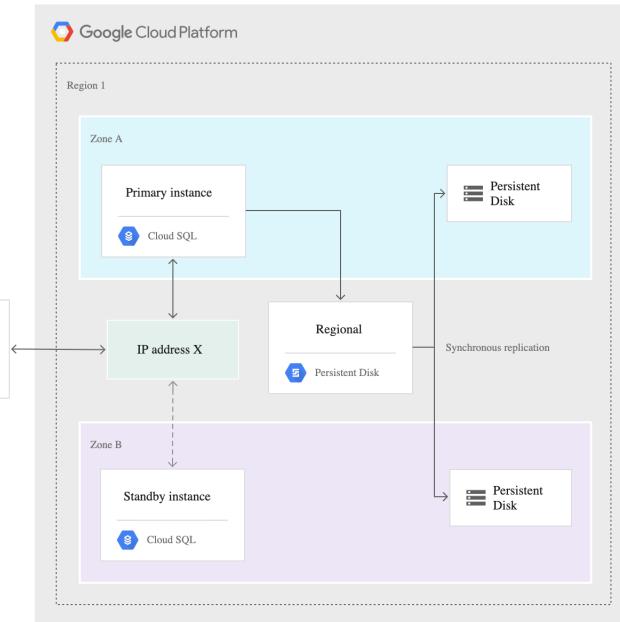
How to connect to Cloud SQL instance?



- Prefer private IP addresses to public IP addresses
- Use Serverless VPC Access for private IP connections from Serverless Environments
 - App Engine, Cloud Functions, Cloud Run
- Use Cloud SQL Auth proxy for these scenarios:
 - Connecting with Public IP from Serverless Environments (App Engine, Cloud Functions, Cloud Run)
 - Connecting with Public / Private IPs from other environments (GCE VMs, GKE etc)
- If you are NOT using Cloud SQL Auth proxy, it is recommended to use self-managed SSL/TLS certificates

Understanding Cloud SQL Best Practices

- Use **Cloud SQL Proxy** when possible
- Use connection pooling & exponential backoff
- Keep transactions short and small
- Use internal IP as much as possible
- Understand **Scalability**:
 - Enable **HA configuration** for high availability
 - Primary instance and a standby instance created in the same Region (Remember - Regional)
 - Read replicas help you **offload read workloads** (reporting, analytics etc)
 - (Remember) Read replicas do NOT increase availability
 - Prefer a **number of small Cloud SQL instances** to having one large instance
 - Cloud SQL cannot scale horizontally for writes



<http://cloud.google.com>

Connect to Cloud SQL instance - Scenarios

Scenario	Solution
How to connect from a VM instance inside a VPC to a Cloud SQL instance using private address?	Create a "Private Service Connection"
Connect to Cloud SQL from Serverless Environments using Private IP address	Use Serverless VPC Access
Connect to Cloud SQL from GCE VMs or GKE using Public IP address	Use Cloud SQL Auth Proxy

Cloud Spanner

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 Spanner

Designing Cloud Spanner Tables - Interleaved Tables

```
CREATE TABLE User (
    UserId INT64 NOT NULL,
    FirstName STRING(1024),
    LastName STRING(1024),
) PRIMARY KEY (UserId);

CREATE TABLE Todo (
    UserId INT64 NOT NULL,
    TodoId INT64 NOT NULL,
    Description STRING(MAX),
) PRIMARY KEY (UserId, TodoId),
INTERLEAVE IN PARENT User ON DELETE CASCADE;
```

- **Scenario:** Every time you access a user, you also access their todos:
 - (DEFAULT) User details and his todos might be stored in different nodes
- How to request Cloud Spanner to store user and his todos together?
 - **Interleaved Tables**
 - User and todo data might be distributed across nodes BUT a specific user and his todo information will be stored on the same node!
 - Provides the best possible performance to access details from multiple tables

Exploring Cloud Spanner Queries - UNNEST

```
CREATE TABLE SomeTable (
    SomeTableId INT64 NOT NULL,
    NumbersArray ARRAY<INT64> NOT NULL
) PRIMARY KEY(SomeTableId);

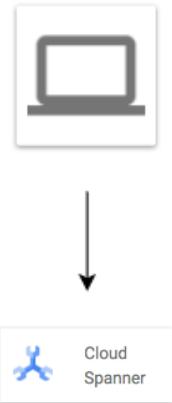
INSERT INTO Todos (UserId, TodoId, Description)
SELECT * FROM UNNEST([(1, 1, 'Learn GCP'),
                      (1, 2, 'Learn AWS'),
                      (1, 3, 'Learn Azure'), ]);

DELETE FROM Todos
WHERE (UserId, TodoId)
IN UNNEST([(1,1),(1,3)]);
```

- **Array:** Ordered list consisting of values of the same data type
 - `SELECT ['AWS', 'Azure', 'GCP'] as platform;`
- Sometimes you would want to convert elements in an array to rows in a table
 - `UNNEST` is used to perform flattening
 - `UNNEST` can be used to execute a single DML statement with an array of data
 - Reduces the need to execute multiple insert or delete statements

Understanding Cloud Spanner Client Libraries

- How do you build applications that talk to Cloud Spanner?
 - Cloud Spanner client libraries (**Cloud Client Libraries Recommended**)
 - Java: Add to pom.xml or gradle dependencies
 - compile 'com.google.cloud:google-cloud-spanner'
 - Node.js: npm install --save @google-cloud/spanner
 - Python: pip install --upgrade google-cloud-spanner
 - **Supported for:** Compute Engine, App Engine flexible environment, Google Kubernetes Engine, and Cloud Functions
 - Java library provides support for App Engine standard environment as well
- **Recommended:** Use Service Account for providing access
- **Couple of things to remember:**
 - Cloud Spanner client library retries a transaction automatically
 - You can divide a query into smaller pieces and execute it in parallel
 - **Example:** txn.partitionQuery(..., Statement.of("SELECT TodoId, Description FROM Todos"))
 - Recommended when reading large amounts of data



Exploring Cloud Spanner Transactions

- **Transaction:** Set of atomic operations
 - Either all operations are successful or none of them are successful
- Cloud Spanner supports **three transaction modes:**
 - **Locking read-write:** When you want to perform write transactions
 - Provide ACID transactional properties of typical relational databases
 - RECOMMENDED: If you are doing single or multiple writes as a result of reads, do all the writes and reads in a Locking read-write transaction mode
 - **Read-only:** Provides consistency across multiple reads
 - Does NOT allow writes
 - **Partitioned DML:** Recommended for Bulk updates (cleanup and backfilling)
 - **Example:** Fill a default value instead of NULL
 - Partitioned DML would partition the key space and execute multiple independent read-write transactions
 - Avoid locking an entire table!



Cloud Spanner

Understanding Cloud Spanner Best Practices

- Use Cloud Spanner **multi-region configuration** when you have users from multiple geographic locations
 - (REMEMBER) Provides faster reads at the **cost of a small increase in write latency**
 - **Available multi-region configurations:**
 - One continent: asia1 (Asia), eur6 (Europe), nam12 (North America) etc
 - Three Continents: nam-eur-asia1 - (North America, Europe and Asia)
- **Design your primary key to distribute data across servers**
 - Avoid monotonically increasing values as the first key part
 - Your inserts and queries might be slow
 - Example: Avoid timestamp-based primary keys
 - If you need to have a timestamp (or a monotonically increasing value) in the primary key
 - Have it as a second part (Have something else as the first part)
 - Example: Use (UserId, LastAccess) as Primary Key instead of (LastAccess, UserId)



Cloud Spanner

Understanding Cloud Spanner Best Practices - 2

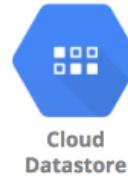
- Accessing **co-located rows** is faster than accessing rows distributed across multiple nodes
 - Make sure that related data is co-located
- Use **stale reads** (as compared to strong read) when your application is latency sensitive but can live with a bit of stale data
- Have Compute Instance(s) and Cloud Spanner instance in the same region:
 - Provides the best performance
- Ensure that high priority total CPU utilization always remains under 65%:
 - Increase number of nodes as needed
- Use Batch DML to send multiple DMLs in a single request



NoSQL Databases in Google Cloud

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



Exploring Cloud Firestore - Native mode vs Datastore mode

- **Firestore in Native mode:**
 - RECOMMENDED for new mobile and web apps
 - Provides Mobile and Web client libraries
 - Real-time updates: Listen to real-time updates on a document
 - Offline features: Important for mobile applications
- **Firestore in Datastore mode:**
 - RECOMMENDED for new server projects
 - Uses Firestore's storage layer
 - Provides Datastore system behavior with a few additions:
 - Strong consistency (by default)
 - Does NOT provide real-time updates and offline features
- **(REMEMBER)** You can use EITHER Native mode or Datastore mode in a single project



Deciding Firestore locations - Regional vs Multi-region

- Firestore supports **Regional** and **Multi-region** locations
- **Regional:**
 - Data is replicated in multiple zones in the same region
 - Used when you want **lower costs and/or lower write latency**
 - Availability SLA: **99.99%**
- **Multi-region:**
 - Data is replicated in multiple regions (and multiple zones with each region)
 - Used to **maximize availability and durability**
 - Can **withstand loss of a complete region**
 - **Examples:** eur3(Europe), nam5(United States)
 - Availability SLA: **99.999%**



Using Indexes with Firestore

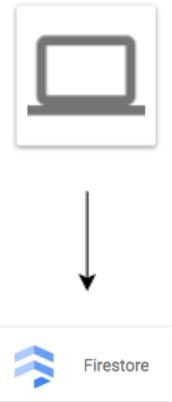
- Single-field indexes automatically created for every field in a document
 - If you do NOT want automatic index on a field, you can create **single-field index exemption**
 - RECOMMENDED: Exempt Large string fields (example: storing descriptions) from indexing
- **Create composite index with multiple fields** as needed by your queries
- **(REMEMBER) Indexes DON'T come FREE!**
 - Indexes increase write latency and storage costs
 - RECOMMENDED: Use indexes judiciously
 - Feel free to create single-field index exemptions if you are NOT going to query on a field



Firestore

Exploring Firestore client libraries

- How to build **server side apps** that talk to Cloud Firestore?
 - Cloud Firestore client libraries (**Cloud Client Libraries Recommended**)
 - Java: Add to pom.xml or gradle dependencies
 - compile 'com.google.cloud:google-cloud-firebase'
 - Node.js: npm install --save @google-cloud/firestore
 - Python: pip install --upgrade google-cloud-firebase
 - **Recommended:** Use Service Account for providing access
- How do you build **mobile applications** that talk to Cloud Firestore?
 - **Firebase** is a Google platform for creating mobile applications
 - **Firebase Admin SDKs** come built-in with Google Cloud client libraries for Firestore
 - Use Firebase Admin SDKs to talk to Firestore
 - Use Firebase Authentication and Firestore Security Rules to secure your data
- (REMEMBER) Firestore SDKs & client libraries **retry failed transactions**
 - If you are using the REST or RPC APIs directly, enable your applications to retry failed transactions



Exploring Firestore client libraries - Node.js Example

```
//insert
const data = { name: 'Ranga', city: 'Hyderabad',
               country: 'India', courses: 45 };
const res = await db.collection('todos').doc('1').set(data);

//update
const todosRef = db.collection('todos');
var documentRef = todosRef.doc('1');
const res = await documentRef.update({state: 'Eindhoven', country: 'Netherlands'});

//delete
const res = await db.collection('todos').doc('1').delete();

//query
const queryRef = todosRef.where('state', '==', 'CA');

//order by
const lastThreeRes = await todosRef.orderBy('city', 'desc').limit(3).get();

//transaction
await db.runTransaction(async (t) => {
  const doc = await t.get(documentRef);
  const newCourses = doc.data().courses + 1;
  t.update(documentRef, {courses: newCourses});
});
```

Exploring Transactions with Firestore

- **Transaction:** Set of atomic operations
 - Either all operations are successful or none of them are successful
- Firestore supports **two types of transactions:**
 - **Read-write transaction:** When you want to perform write transactions
 - If you are doing single or multiple writes as a result of reads:
 - Recommended to do all the writes and reads in a Read-write transaction
 - **Read-only:** Provides consistency across multiple reads
 - Does NOT allow writes
- You can query or lookup any number of entities in a transaction
 - HOWEVER a maximum of 500 entities can be created, updated, or deleted in a transaction



Firestore

Understanding Cloud Firestore Best Practices

- Cloud Firestore/Datastore is a **document store with flexible schema**
 - Recommended for storing things like user profiles
 - Another Use Case: Index for objects stored in Cloud Storage
 - You want to allow users to upload their profile pictures:
 - Store objects (pictures) in Cloud Storage
 - Enable quick search by storing metadata (like ids and cloud storage bucket, object details) in Cloud Datastore
- Design **your keys and indexes** carefully:
 - Distribute operations evenly across key range
 - Avoid monotonically increasing values as keys
 - NOT RECOMMENDED - 1, 2, 3, ..., OR "Customer1", "Customer2", "Customer3", ... or timestamps
 - RECOMMENDED - Use `allocateIds()` for well-distributed numeric IDs



Firestore

Understanding Cloud Firestore Best Practices - 2

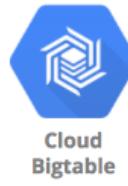
- Create your database instance **in the region nearest to your users and compute resources**
- Prefer **batch operations** (to single read, write or delete operations):
 - More efficient as multiple operations are performed with same overhead as one operation
- Remember **Firestore Soft limits**: 1 write per second per document, 1 million concurrent connections per database
 - Exceeding these limits might affect performance
- **Gradually ramp up traffic** to a new collection: "500/50/5" Rule
 - Start with MAX 500 operations per second
 - Increase by 50% every 5 minutes



Cloud BigTable

In 28
Minutes

- **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`



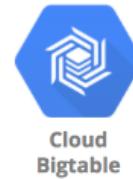
Cloud BigTable - Wide Column Database

Rowid	Column Family 1			Column Family 2			Column Family 3		
	col1	col2	col3	col1	col2	col3	col1	col2	col3
1									
2									
3									

- 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-qualifier(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

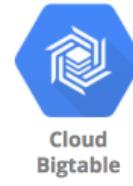
Designing BigTable Tables

- Two things you should know before starting with Bigtable:
 - What data do you want to store? (format, columns etc)
 - What would your frequently used queries look like (ranked by usage)?
- Design your table: Cloud Bigtable is a **key/value store**
 - Each table has **ONLY ONE** index, the row key
 - You cannot create any other secondary indices
 - **Design your row key** based on your frequently used queries
 - Goals when designing row key
 - 1: Avoid full table scan
 - 2: Distribute reads and writes evenly across the row space of a table



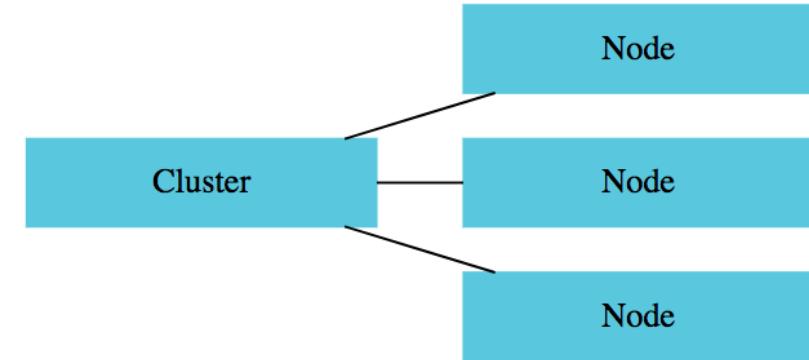
Designing BigTable Tables - Row Key

- You can have multiple row key segments:
 - Separated by a delimiter (**Example:** ranga#123456#abcd)
- **Avoid sequential row keys**
 - Avoid timestamps or sequential numbers as the first key segment
 - IF you plan to retrieve data based on a timestamp
 - Use timestamp as second or third key segment
 - Use reversed timestamp (Ex: Long.MAX_VALUE - timestamp) if you frequently query recent data
 - Records will be ordered from most recent to least recent
- Row keys should **start with a common value and end with a granular value**
 - **Example1:** asia#india#bangalore
 - Allows you to query by continent or country or city
 - **Example2:** customer-name#rest-of-the-key
 - Useful in multi-tenant applications
 - Recommended to store multi-tenant data in the same table when possible



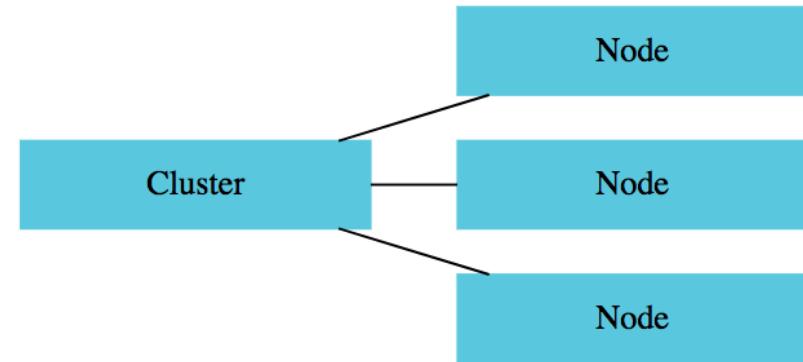
Understanding Cloud BigTable Best Practices

- Recommended for streaming IOT & time series data
- **Automatically shards data into multiple tablets across nodes in cluster:**
 - **Goal 1:** Have same amount of data on each node
 - **Goal 2:** Distribute reads and writes equally across all nodes
 - **(REMEMBER)** Pre-test with heavy load for a few minutes before you run your tests
 - Gives Bigtable a chance to balance data across your nodes
- Cloud Bigtable supports SSD or HDD storage:
 - **SSD** - For most usecases
 - **HDD** - For large non latency-sensitive data sets of size >10 TB with very very few reads



Understanding Cloud BigTable Best Practices - Replication

- You can create a Cloud Bigtable instance with **more than one cluster** to enable **replication** (**Cross Region or Cross Zone**) :
 - Independent copy of data is stored in each cluster (in the zone of the cluster)
 - Bigtable automatically replicates changes
 - Replication **improves durability and availability** of your data
 - Stores separate copies in multiple zones or regions
 - Can automatically failover between clusters if needed
 - Replication helps you to **put data closer to your customers**
 - Configure an application profile, or app profile with routing policy of multi-cluster routing
 - Automatically route to nearest cluster in an instance

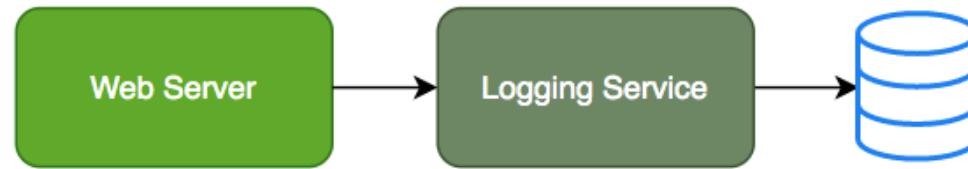


Decoupling Applications with Pub/Sub

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 your applications go down too?
- What if all of sudden, there is high load and there are lots of logs coming in?
 - Log Service is not able to handle the load and goes down very often

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

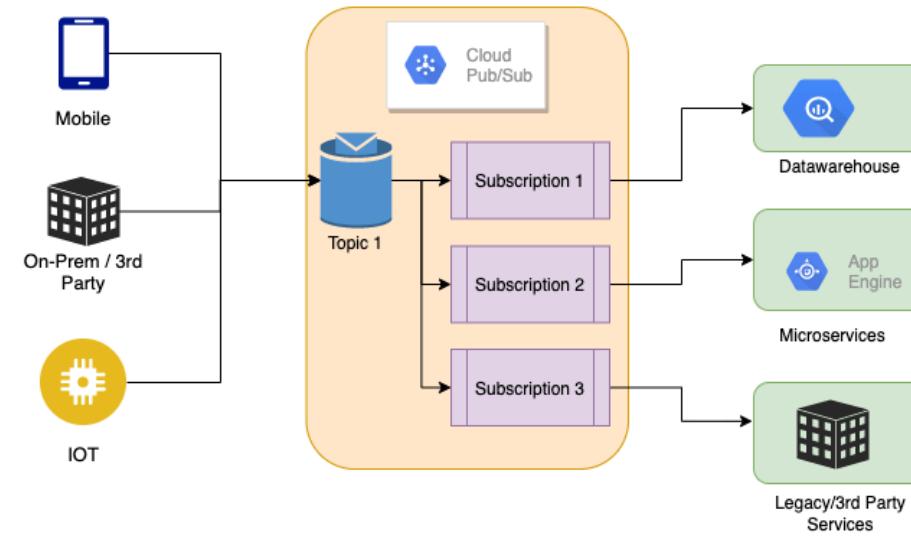
Pub/Sub

- 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)
- Use cases: Event ingestion and delivery for streaming analytics pipelines
- Supports push and pull message deliveries

Cloud
Pub/Sub

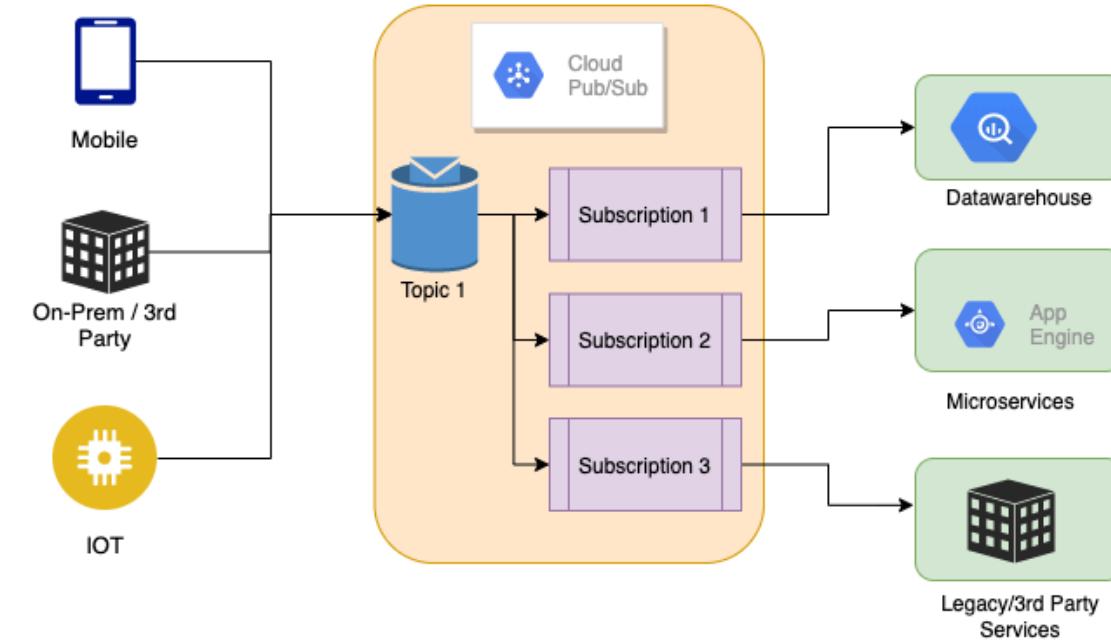
Pub/Sub - How does it work?

- **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



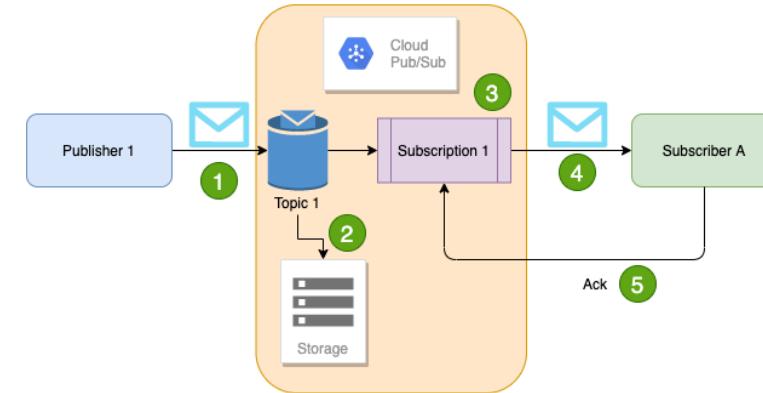
Pub/Sub - Getting Ready with Topic and Subscriptions

- 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



Using Cloud Client Libraries - Pub Sub

- **1: Setup Cloud Client Libraries:**
 - **Java:** Add to pom.xml or gradle dependencies
 - compile 'com.google.cloud:google-cloud-pubsub'
 - **Node.js:** npm install --save @google-cloud/pubsub
 - **Python:** pip install --upgrade google-cloud-pubsub
- **2: Setup Authentication:**
 - Create service account with the right permissions
 - Make your application use the Service Account (assign or use service account key)
- **3: Write the code**

```
//Global code - Reuse these across requests
const {PubSub} = require('@google-cloud/pubsub');
const pubSubClient = new PubSub();

//Publish Message to Topic YOUR_TOPIC_NAME
const messageBuffer = Buffer.from(message);
const messageId = await pubSubClient.topic('YOUR_TOPIC_NAME')
    .publish(messageBuffer);

//Pull a subscription
const subscription = pubSubClient
    .subscription('YOUR_SUBSCRIPTION_NAME');

subscription.on('message', message => {
    console.log(`Message ID - ${message.id}`);
    // Processing Logic

    // "Ack" (Acknowledge receipt of) the message
    message.ack();
});

subscription.on('error', error => {
    console.error('Received error:', error);
    process.exit(1);
});
```

Getting Started with Cloud Tasks

- Manage execution, dispatch, and delivery of a **large number of distributed tasks** asynchronously
- **Supports tasks targeting any HTTP service running on:**
 - Compute Engine
 - Google Kubernetes Engine
 - Cloud Run
 - Cloud Functions
 - On-premises systems
- **Difference between Cloud Task and Pub/Sub:**
 - **Pub/Sub** - Implicit invocation
 - Pull and Push Models
 - **Cloud Tasks** - Explicit invocation
 - Only Push Model

Exploring Cloud Tasks - An Example

```
const {CloudTasksClient} = require('@google-cloud/tasks');
const cloudTasksClient = new CloudTasksClient();
const parent = cloudTasksClient.queuePath('YOUR_PROJECT',
    'YOUR_ZONE', 'YOUR_CLOUD_TASKS_QUEUE_NAME');

const task = {
  httpRequest: {
    httpMethod: 'POST',
    relativeUri: '/URL_YOU_WANT_TO_INVOKE',
  },
};
task.httpRequest.body = Buffer.from('YOUR_PAYLOAD').toString('base64');
task.scheduleTime = {
  seconds: 1800 + Date.now() / 1000, //Schedule after 30 minutes
};

const request = {parent, task};
const [response] = await cloudTasksClient.createTask(request);
console.log(`Created task ${response.name}`);
```

- Example: Schedule a call to '/URL_YOU_WANT_TO_INVOKE' after 30 minutes
 - Creation of Task Queue: *gcloud tasks queues create YOUR_CLOUD_TASKS_QUEUE_NAME*
- Prefer Cloud Tasks to schedule/asynchronously invoke background tasks

Cloud Scheduler

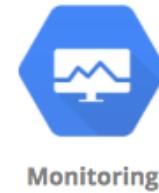
In 28
Minutes

- **Fully managed, enterprise-grade scheduler**
 - Schedule all kinds of jobs
 - Batch, big data jobs, cloud infrastructure operations etc
 - Uses Unix cron format
- Integrates with App Engine, Cloud Pub/Sub, Cloud Logging and any HTTP endpoint
- Manage all your automation tasks from one place
- Provides **automated retries**
- **Use Case:** Schedule tasks across a fleet of Compute Engine instances
 - Use Cloud Scheduler for scheduling a message on Pub/Sub
 - Compute Engine instances can process messages from Pub/Sub
- **(REMEMBER)** Needs an App Engine App in the Project
 - **Earlier Alternative:** App Engine Cron Service

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



Monitoring

Cloud Monitoring - Virtual Machines

In 28
Minutes



- **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



Logging

Cloud Logging - Collection

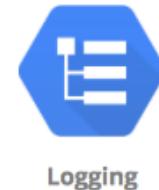
In 28
Minutes



- 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

```
{  
  protoPayload: {  
    @type: "type.googleapis.com/google.cloud.audit.AuditLog",  
    status: {},  
    authenticationInfo: {  
      principalEmail: "user@example.com"  
    },  
    serviceName: "appengine.googleapis.com",  
    methodName: "SetIamPolicy",  
    authorizationInfo: [...],  
    serviceData: {  
      @type: "type.googleapis.com/google.appengine.legacy.AuditData",  
      policyDelta: { bindingDeltas: [  
        action: "ADD",  
        role: "roles/logging.privateLogViewer",  
        member: "user:user@example.com"  
      ], }  
    },  
    request: {  
      resource: "my-gcp-project-id",  
      policy: { bindings: [...] }  
    },  
    response: {  
      bindings: [  
        {  
          role: "roles/logging.privateLogViewer",  
          members: [ "user:user@example.com" ]  
        }  
      ]  
    },  
    insertId: "53179D9A9B559.AD6ACC7.B48604EF",  
    resource: {  
      type: "gae_app",  
      labels: { project_id: "my-gcp-project-id" }  
    },  
    timestamp: "2019-05-27T16:24:56.135Z",  
    severity: "NOTICE",  
    logName: "projects/my-gcp-project-id/logs/cloudaudit.googleapis.com%2Factivity",  
  }  
}
```

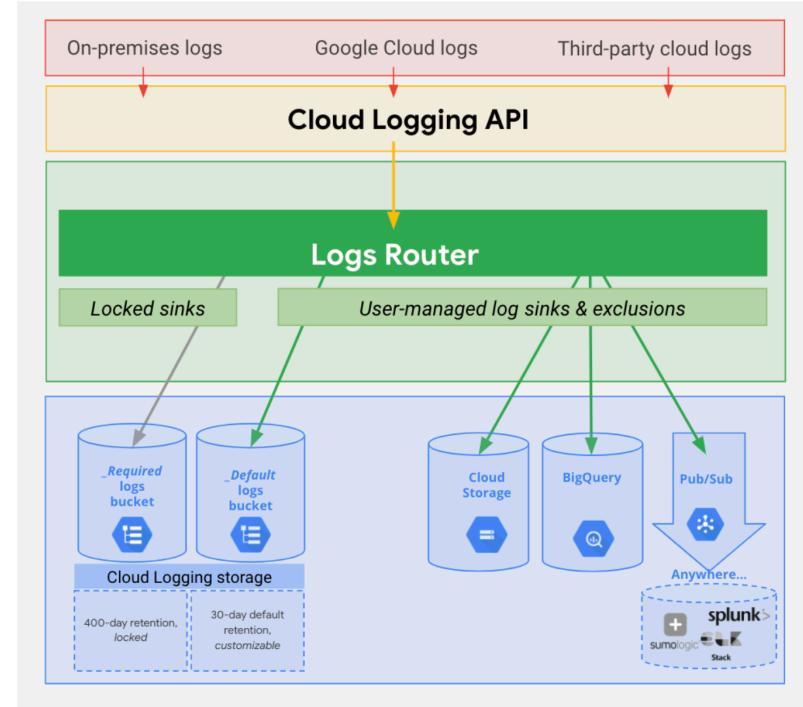
Cloud Audit Logs

In 28
Minutes

Feature	Admin Activity Logs	Data Access Logs	System Event Logs	Policy Denied Logs
Logs for	API calls or other actions that modify the configuration of resources	Reading configuration of resources	Google Cloud administrative actions	When user or service account is denied access
Default Enabled	✓	X	✓	✓
VM Examples	VM Creation, Patching resources, Change in IAM permissions	Listing resources (vms, images etc)	On host maintenance, Instance preemption, Automatic restart	Security policy violation logs
Cloud Storage	Modify bucket or object	Modify/Read bucket or object		
Recommended	Logging/Logs Viewer	Logging/Private	Logging/Logs Viewer	Logging/Logs

Cloud Logging - Controlling & Routing

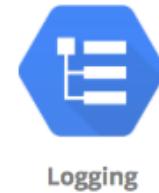
- How do you manage your logs?
 - Logs from various sources reaches **Log Router**
 - Log Router checks against configured rules
 - What to ingest? what to discard?
 - Where to route?
- Two types of Logs buckets:
 - **_Required:** Holds Admin activity, System Events & Access Transparency Logs (retained 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!
 - You can edit retention settings (1 to 3650 days (10 years))



source: (<https://cloud.google.com>)

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)

Setting up Cloud Monitoring for Virtual Machines

- Default metrics monitored include:
 - CPU utilization
 - Some disk traffic metrics
 - Network traffic, and
 - Uptime information
- Install **Cloud Monitoring agent** for more disk, CPU, network, and process metrics:
 - collectd-based daemon
 - Gathers metrics from VM and sends them to Cloud Monitoring
- (NEWER OPTION) **Ops Agent**
 - One agent for (logging + metrics)
 - **Designed for workloads needing higher throughput and/or improved resource-efficiency**



Collecting Logs for Cloud Logging



- Most GCP services send **stdout & stderr logs automatically to Cloud Logging:**
 - GKE
 - App Engine
 - Cloud Run
- Ingest logs from GCE VMs:
 - Install **Logging Agent** (based on fluentd)
 - (Recommended) Run **Logging Agent (Or Ops Agent)** on all VM instances
 - Following logs are automatically picked up and sent to Cloud Logging - Linux syslog, Apache logs, Jetty logs, Mongodb logs, MySQL logs, Nginx logs, RabbitMQ logs, Redis logs, Tomcat logs etc
 - If you want to send custom application logs to Cloud Logging:
 - Configure the path to application logs in a new configuration file (extension conf. Ex:app-logs.conf)
 - Place it in additional configuration directory /etc/google-fluentd/config.d

Creating Custom Metrics - Cloud Monitoring

- You can create your own custom (application-specific) metrics
 - You can write time series data to your custom metrics
 - RECOMMENDED: Use names starting with `custom.googleapis.com/`
- Two options to create custom metrics:
 - RECOMMENDED: Use OpenCensus (open-source monitoring and tracing library)
 - Supports a variety of languages (libraries are present for Go, Java, Node.js, Python,...)
 - Use low-level Cloud Monitoring API (NOT RECOMMENDED)
- Once a metric is sent to Cloud Monitoring, there is no difference between a custom and a built-in metric
 - You can use them in alerts and dashboards



Monitoring

Creating Logs-based metrics for Cloud Monitoring

- **Logs-based metrics:** Metrics generated from log entries
 - Can be used just like other metrics in Cloud Monitoring charts and alerting policies
- **Two Types:**
 - **Counter:** Counts the number of log entries matching a given filter
 - Example: Count the log entries that contain a certain specific error message
 - **Distribution:** Collects numeric data from log entries matching a given filter
 - Example: Average of request latencies



Configuring Cloud Monitoring & Cloud Logging - GKE

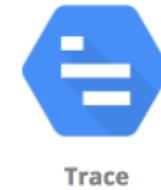
- GKE has **good integration** with Cloud Monitoring and Cloud Logging
 - Cloud Operations for GKE is enabled by default
 - Read logs stream from a pod: `kubectl logs pod-name`
 - **HOWEVER**, if you want to read from multiple clusters:
 - Read directly from Cloud Logging - Cloud console or `gcloud logging read`
- **Prometheus** is a popular monitoring tool often used with Kubernetes
 - You can **run Prometheus server** in the cluster
 - You can **run sidecar containers** beside your workloads to gather and send metrics to Prometheus server
 - Metrics will be visible as external metrics in Cloud Monitoring



Kubernetes Engine

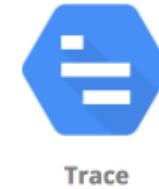
Getting Started with Cloud Trace

- **Distributed tracing system for GCP:** Collect latency data from:
 - Supported Google Cloud Services
 - Instrumented applications
- **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
- **Features:**
 - Automatically identify recent changes to your application's performance (based on continuous monitoring)



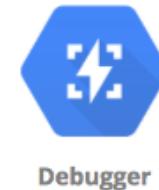
Instrumenting your application - Cloud Trace

- Tracing needs **your application to be instrumented**
- You can send trace data to Cloud Trace using **Cloud Trace API**
 - BUT Cloud Trace APIs are **low-level**
 - RECOMMENDED: Use **OpenTelemetry** and the associated Cloud Trace client library
 - **OpenTelemetry**: Collection of tools, APIs, and SDKs to instrument, generate, collect, and export telemetry data
 - Use **OpenCensus** (An older version of OpenTelemetry) if an OpenTelemetry client library is not available for your programming language
 - ALTERNATIVE: If **your application is using Zipkin** and you want to keep using Zipkin Server:
 - You can forward traces to Cloud Trace for advanced analysis
- By **using open source alternatives**, you can decouple your application from GCP



Getting Started with Cloud Debugger

- How to find root cause for difficult to debug issues that occur only in test or production environments?
 - Cloud Debugger
 - Features:
 - Create Debug snapshots - View the state of local variables and the call stack at specific points in your production code
 - Add Debug logpoints - Inject additional logging in your code without restarting it (without interfering with its normal behavior)
 - Conditional debugging - Use a conditional expression to decide when to capture debug snapshot or add additional logging using logpoint
 - Inspect the state of the application directly in the GCP environment
 - No need to redeploy
 - Very lightweight => Very little impact to users
 - Can be used in any environment: Test, Acceptance, Production
 - Supported for App Engine (automatically enabled), Google Kubernetes Engine and Compute Engine



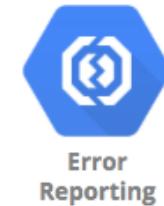
Getting Started with Cloud Profiler

- How do you identify performance bottlenecks in production?
- **Cloud Profiler** - Statistical, low-overhead profiler
 - Continuously gathers CPU and Memory usage from production systems
 - Goal: Identify parts of the application consuming the most resources
 - Connect profiling data with application source code
 - Easily identify performance bottlenecks
 - Supports Java, Go, Node.js, and Python
 - Supported for Compute Engine, Google Kubernetes Engine, App Engine and Dataproc
 - Two major components:
 - Profiling agent (collects profiling information)
 - Profiler interface (visualization)



Getting Started with 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 Logging, Monitoring .. - Stackdriver

Stackdriver Service	New Service Name
Stackdriver Monitoring	Cloud Monitoring
Stackdriver Logging	Cloud Logging
Stackdriver Error Reporting	Error Reporting
Stackdriver Trace	Cloud Trace
Stackdriver Profiler	Cloud Profiler

Exploring Cloud Operations Scenarios

In 28
Minutes

Scenario	Solution
How can you trigger an alert when your application is unavailable?	Configure an Cloud Monitoring uptime check
You want to create a dashboard with how many times a substring occurs in a log	Create a Logs-based counter metric
You want to identify prominent exceptions (or errors) for a specific microservice	Error Reporting
You want to trace a request across multiple microservices	Cloud Trace
You want to debug a problem in production by conditionally adding in a logpoint or capturing a snapshot	Cloud Debugger
You want to look at the logs for a specific request	Cloud Logging
You want to solve performance bottlenecks by identifying parts of the application consuming the most resources	Cloud Profiler

More Security

- **Cloud Armor:** Protect your apps from denial of service and OWASP Top 10 attacks
 - Protects you from **common web attacks** (OWASP Top 10) like XSS (cross-site scripting) and SQL injection
 - Protect applications deployed in Google Cloud, in a hybrid deployment, or in a multi-cloud architecture
 - Integrates very well with Google Cloud Load Balancing
 - Provides **preconfigured security policies** (OWASP Top 10 risks etc)
 - Customize rules as per your needs
 - **Usecases**
 - Enable access for users at specific IP addresses with allowlists
 - Block access for users at specific IP addresses with denylists
 - Protect applications against OWASP Top 10 risks



Secret Manager

```
SecretVersionName secretVersionName = SecretVersionName.of(projectId, secretId, versionId)
AccessSecretVersionResponse response = client.accessSecretVersion(secretVersionName);
String secretValue = response.getPayload().getData().toStringUtf8();
```

- How do you **manage your database passwords, your API keys** securely?
- **Secret Manager** - Store API keys, passwords etc
 - Multiple versions of secrets
 - Automate rotation with Cloud Functions
 - Auditing with Cloud Audit Logs
 - Encrypted by default
- Best Practice: **Do NOT store credentials/passwords in code**
 - Use Secret Manager and access secrets in your application

Getting Started with Container Scanning API

- How do you ensure that the container images created for your applications does NOT have any known vulnerabilities?
 - Use Container Scanning API
 - Service to scan containers for vulnerabilities
- **Can be enabled** on container images in:
 - Container Registry and
 - Artifact Registry
- As soon as you push an image to the registry, a **scan for known security vulnerabilities and exposures** is triggered
- **Can also be triggered manually:**
 - 1: Trigger Scan: outputs a SCAN-ID
 - `gcloud artifacts docker images scan ubuntu:latest`
 - 2: Get the scan results using SCAN-ID
 - `gcloud artifacts docker images list vulnerabilities --SCAN-ID`

Getting Started with Binary Authorization

- How do you ensure that only trusted container images are deployed to Google Cloud?
 - Enable Binary Authorization
 - Ensures that only trusted images are deployed to Google Cloud
 - Supported on:
 - Google Kubernetes Engine (GKE)
 - Cloud Run
 - Anthos Service Mesh
 - Anthos clusters on VMware (on-premises Kubernetes)
- Configure a **set of rules (a Policy)** that are evaluated before a container image is deployed:
 - Example: Verify that an image was built by a specific build system
 - Example: Verify that an image is scanned for vulnerabilities by Container Analysis
 - Example: Verify that an image is manually attested

Getting Started with VPC Service Controls

- We use IAM to give access to resources
 - What if we do NOT want to allow transfer/copy of data beyond a configured boundary even if a user has access through IAM?
- **VPC Service Controls:** Control data exfiltration from GCP service
 - Supported on: BigQuery, GCS
 - Independent of IAM
 - Configure security parameters:
 - Select multiple projects to protect
 - Select services that you want to secure
- Security Best Practice - **Defense in Depth**
 - Use both VPC Service Controls and IAM



Virtual Private
Cloud

Implement Data Security with Cloud Data Loss Prevention

- Discover, classify, & mask sensitive data (like Credit Card numbers, SSNs, clear text passwords & Google Cloud credentials)
 - **Classification** - Identify the type of data that a piece of text contains
 - Example: Given email content "Email address: abc@in28minutes.com SSN:123-456-7890", identify that it contains email address and an SSN
 - **Redaction** - Replace sensitive data with a mask
 - Input: "Email address: abc@in28minutes.com SSN:123-456-7890"
 - Output: "Email address: abc@in28minutes.com SSN:***"
 - **De-identification** - Replace sensitive data while leaving in the possibility of re-identification (by a trusted party in certain situations)
 - Input: "Email address: abc@in28minutes.com SSN:123-456-7890"
 - Output: "Email address: abc@in28minutes.com SSN:SSN134567" and store SSN134567 with the value of SSN
- Integrates with Cloud Storage, BigQuery, and Datastore
- Provides APIs that can be invoked from your applications
- Demo

Exploring Other Google Cloud Security Offerings

Service	Description
Web Security Scanner	Identify vulnerabilities by running security tests. Examples: Cross-site scripting (XSS) MIXED_CONTENT,OUTDATED_LIBRARY, XSS
Anomaly Detection	Uses behavior signals from outside your system to find security anomalies Examples: account_has_leaked_credentials, resource_involved_in_coin_mining, resource_used_for_phishing
Event Threat Detection	Monitors Cloud Logging stream across projects to detect threats. Examples: Data exfiltration, Malware, Brute force SSH, Cryptomining, Outgoing DDoS etc.
Container Threat Detection	Detects container runtime attacks. Examples: Added binary executed, Added library loaded
Security Health Analytics	Automatically detect the highest severity vulnerabilities and misconfigurations Examples: MFA_NOT_ENFORCED, PUBLIC_BUCKET_ACL, PUBLIC_DATASET, PUBLIC_IP_ADDRESS, SSL_NOT_ENFORCED etc Compliance monitoring for industry best practices (PCI DSS v3.2.1, ISO 27001 etc)

Getting Started with Security Command Center

- Google cloud offers a number of services to analyze security of your GCP deployments
 - How do you get a consolidated picture of security in Google Cloud?
 - Security Command Center: Security and data risk database
 - Provides an intelligent risk dashboard and analytics system
- Offers Two Tiers:
 - Standard tier:
 - Security Health Analytics (Basic)
 - Web Security Scanner (for applications with public URLs and IPs)
 - Cloud Data Loss Prevention, Google Cloud Armor, Anomaly Detection
 - Premium tier: (Additional Features)
 - Security Health Analytics (Advanced)
 - Web Security Scanner (All apps)
 - Event Threat Detection, Container Threat Detection

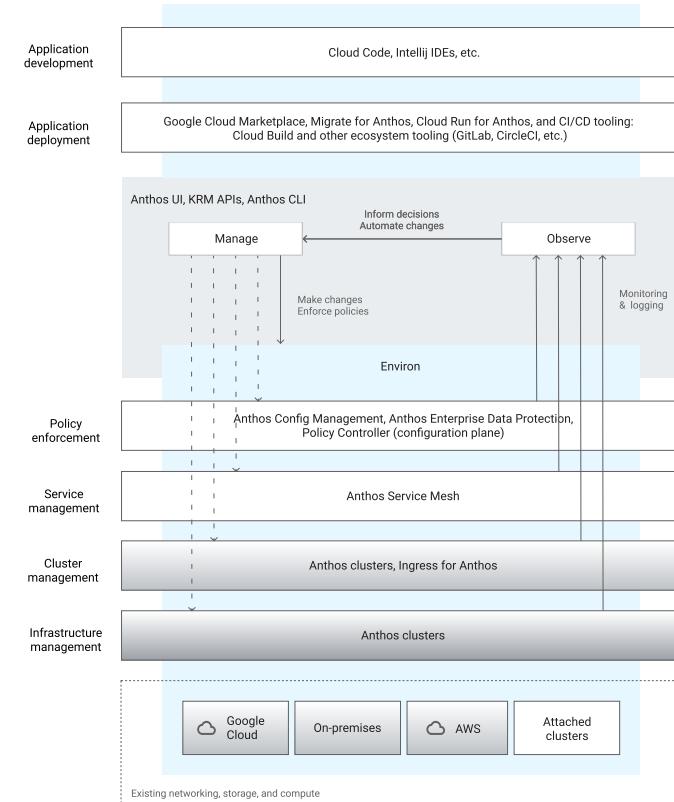
Exploring Google Cloud and Security - Scenarios

Scenario	Solution
Find out which storage buckets contain sensitive data	Cloud DLP
Scan and find common vulnerabilities like cross-site-scripting (XSS) in your App Engine applications	Web Security Scanner
Protect your apps from DDoS and common vulnerabilities	Cloud Armor
Detect if your VM is being used for Bitcoin mining	Anomaly detection
Scan your container images for vulnerabilities	Container Scanning API
Deploy only trusted container images to Google Cloud	Binary Authorization
Detect container threats at runtime	Container Threat Detection
Get an overall security picture of your projects	Security Command Center

Anthos & Anthos Service Mesh

Anthos

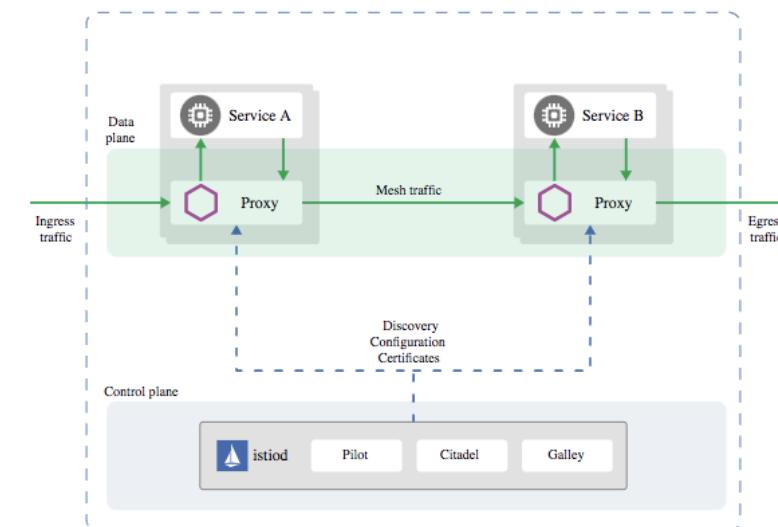
- Run K8S clusters on cloud & on-premises
 - Multi-cluster management: Consistent managed K8S
 - Consistent development and operations experience
- Centralized config management (Git repo)
 - Logically group and normalize clusters as environs
 - Define policies - Kubernetes API, Access control
 - Deploy to clusters on new commits
 - Use Namespaces, labels, and annotations to decide which clusters to apply changes on
- Provides Service Mesh (based on Istio)
 - Sidecar to implement common microservice features
 - Authentication & authorization (service accounts)
 - Distributed Tracing, Automatic metrics, logs & dashboards
 - A/B testing, canary rollouts (even track SLIs & error budgets)
 - Cloud Logging & Cloud Monitoring Support



<https://cloud.google.com>

Getting Started with Service Mesh

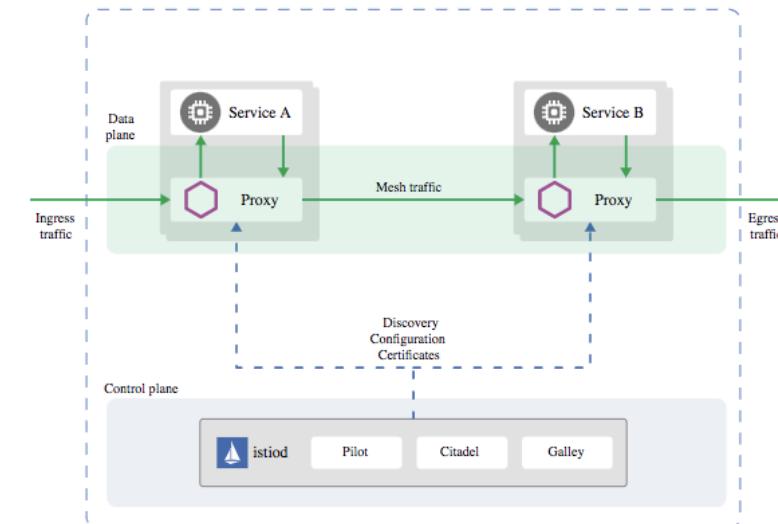
- You are implementing a microservices architecture using Kubernetes.
 - How do you implement common microservice features (security, metrics, logging, tracing, release management)?
 - Enter Service Mesh
- Popular Service Mesh implementation: **Istio**
- **How is Service Mesh implemented?**
 - **Data plane:** proxy deployed beside each container (also called side car)
 - Istio uses a proxy called Envoy
 - **Control plane:** Manage and configure proxies (service discovery, certificate management etc)



<https://istio.io>

Getting Started with Istio

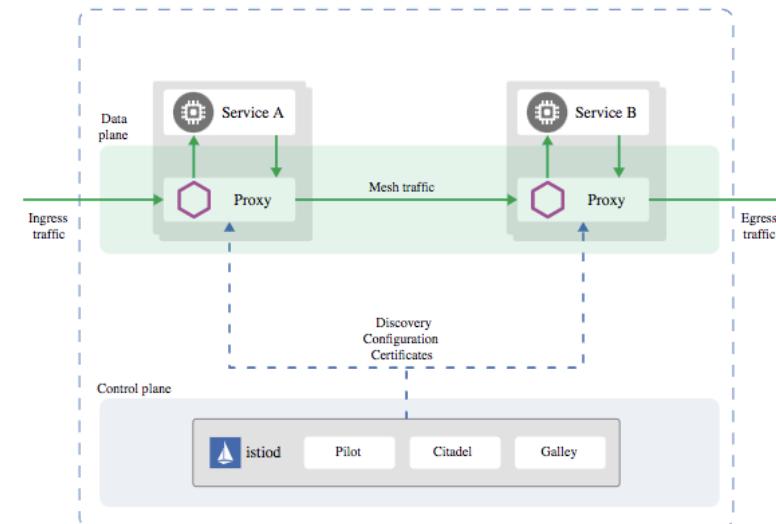
- Popular open source service mesh
- Integrates well with **Kubernetes and GKE**
- Efficiently secure, connect, & monitor services
- **Features:**
 - **Security:** authentication, authorization, mTLS, encryption ..
 - **Observability and Monitoring:** metrics, logging, distributed tracing, ..
 - **Reliability:** rate limiting, health checks, circuit breaker, retry, ..
 - **Release Management:** canary releases, traffic shadowing , A/B testing ..
 - **Automatic load balancing:** Supports HTTP, TCP, gRPC, WebSocket traffic ..



<https://istio.io>

Running Istio on GKE

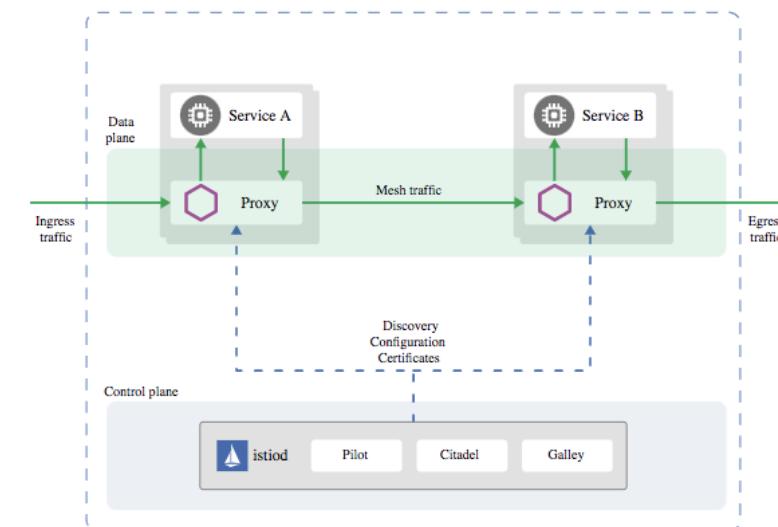
- Two options to use Istio with GKE
 - Istio on GKE (beta)
 - Anthos Service Mesh
- **Istio on GKE:** Automated Istio in your GKE cluster
 - Automatically upgraded with your cluster
 - **NOT RECOMMENDED** for production workloads
 - In beta status
 - Limited configuration options
 - No SLA for Istio components



<https://istio.io>

Exploring Anthos Service Mesh (Istio) - A few use cases

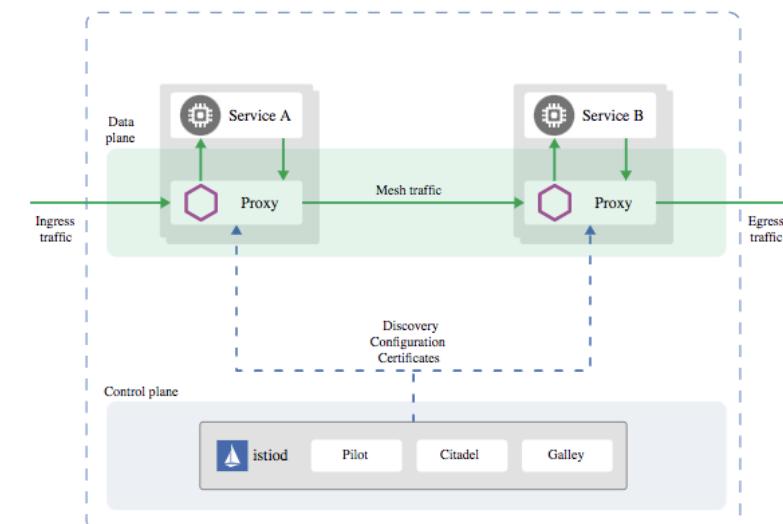
- **Anthos Service Mesh:** Google's fully-supported distribution of Istio
 - RECOMMENDED for production workloads
 - **Features:**
 - Managed private certificate authority (simplifies mTLS)
 - Simplified authentication for application users using Identity-Aware Proxy (IAP) integration
 - Integrates with Cloud Logging, Cloud Monitoring, Cloud Trace
 - SLO monitoring
 - All other typical service mesh features
 - **Use Cases:**
 - Canary deployments (Distribute traffic to two versions of same microservice)
 - Chaos monkey (inject delays and faults to test service reliability)
 - mTLS - Two way TLS between pods
 - SLO monitoring



<https://istio.io>

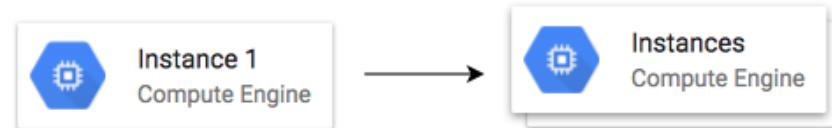
A Quick Review - Putting things into context!

- **Docker** - Create Images and manage Containers
- **Kubernetes** - Container Orchestration
 - **Google Kubernetes Engine** - Fully supported Kubernetes implementation on GCP
- **Istio** - Service Mesh
 - **Use cases:** Cross cutting concerns and testing patterns such as canary, shadow, and A/B testing
 - **Envoy** - Proxy used by Istio
 - **Anthos Service Mesh** - Google's fully-supported distribution of Istio
- **Anthos** - Manage multiple Kubernetes clusters in multi cloud and on-premises



<https://istio.io>

More GCE



- How can VMs in same VPC talk without using IP Addresses?
 - Use internal DNS names
- When we create a VM instance, an **Internal DNS name** is created for it
- **Internal DNS name** resolves to internal IP address of the instance
- **Zonal DNS** is recommended - VM_NAME.ZONE.c.PROJECT_ID.internal
 - **VM_NAME** - What is your VM instance name?
 - **ZONE** - Which zone is it located in?
 - **PROJECT_ID** - Which project is it created in?
- **VM instances in the same VPC** can use internal DNS names to communicate

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
- You can disable Project wide SSH keys on a specific compute instance
 - *gcloud compute instances add-metadata [INSTANCE_NAME] --metadata block-project-ssh-keys=TRUE*



Executing Shutdown Script on a GCE VM

- Execute commands before a GCE VM is stopped, terminated or restarted
 - Perform cleanup or export of logs
 - Applicable for Preemptible and Non Preemptible GCE VMs
- Very similar to startup script
 - Run as:
 - root user in Linux VMs
 - System account in Windows VMs
 - Stored as metadata
 - --metadata-from-file shutdown-script=script.sh
 - You can store startup and shutdown scripts in cloud storage
 - --metadata shutdown-script-url=gs://bucket-in-cloud-storage/file
- Run on best-effort basis
 - Example: WON'T run if you use hard reset (instances.reset)
 - Example: WON'T run if you exceed grace period for preemptible instances



Startup Script Example

Deploy new application version to Compute Engine

```
# Setup logging agent
curl -sS0 https://dl.google.com/cloudagents/install-logging-agent.sh
sudo bash install-logging-agent.sh

# Install GIT
apt-get update
apt-get install -yq git

# Clone Repo
git clone https://github.com/in28minutes/your-app.git /opt/app

# Build and Run app
//...
```

GCE VMs - Project and Instance Custom Metadata

- **Custom metadata** can be configured: Project & Instance levels
 - Metadata applied to project applies to all VM instances in the project
- Examples of VM configuration stored as custom metadata:
Startup and shutdown scripts, SSH keys
- You can **create your custom named attributes** as well
 - *gcloud compute project-info/instances add-metadata --metadata name=value*
- You can use **custom metadata** to store application configuration
- (Remember) Total limit of **512 KB** for all metadata entries
 - If you are nearing the limit, store startup or shutdown scripts in Cloud Storage



Compute
Engine

Troubleshooting VM startup and ...

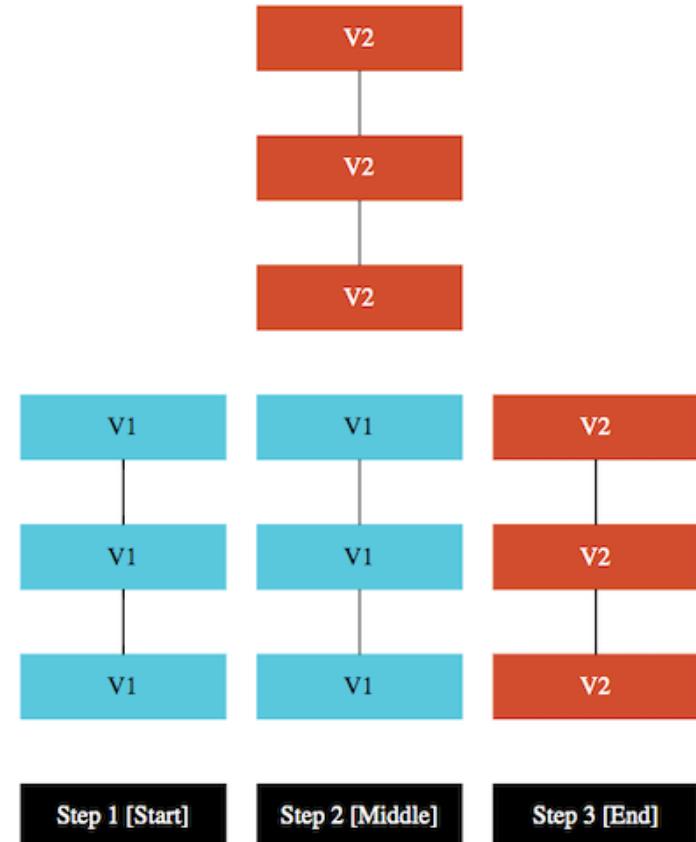
- **Check 1:** Are there Quota errors?
- **Check 2:** Is boot disk full?
- **Check 3: Check serial port output**
 - Each VM instance has 4 virtual serial ports
 - Serial Port Output: OS, BIOS, and other system-level entities write output to serial ports
 - Useful for troubleshooting crashes, failed boots, startup, or shutdown issues
 - Accessible from Cloud Console, the gcloud tool, and the Compute Engine API
 - You can send serial port output to Cloud Logging:
 - `gcloud compute project-info add-metadata --metadata serial-port-logging-enable=true`
 - Interactive access to the serial console allows you to login and debug boot issues (without needing a full boot up)
 - `gcloud compute instances get-serial-port-output`
- **Check 4: Does your disk have a valid file system?**
 - Attach boot disk as a data disk to another VM and check the file system



Release Management

Release Management

- **Goals:** vary from app to app
 - Zero Downtime
 - Only one version live at a time
 - Minimize Costs (and infrastructure needed)
 - Test with production traffic before going live
- **Best Practices:**
 - Small incremental changes
 - Automation (as much as possible)
 - Handling problems with new releases:
 - Analyze logs and metrics from Cloud Monitoring and Logging
 - Rollback to previous release and try replicating the problem in other environments



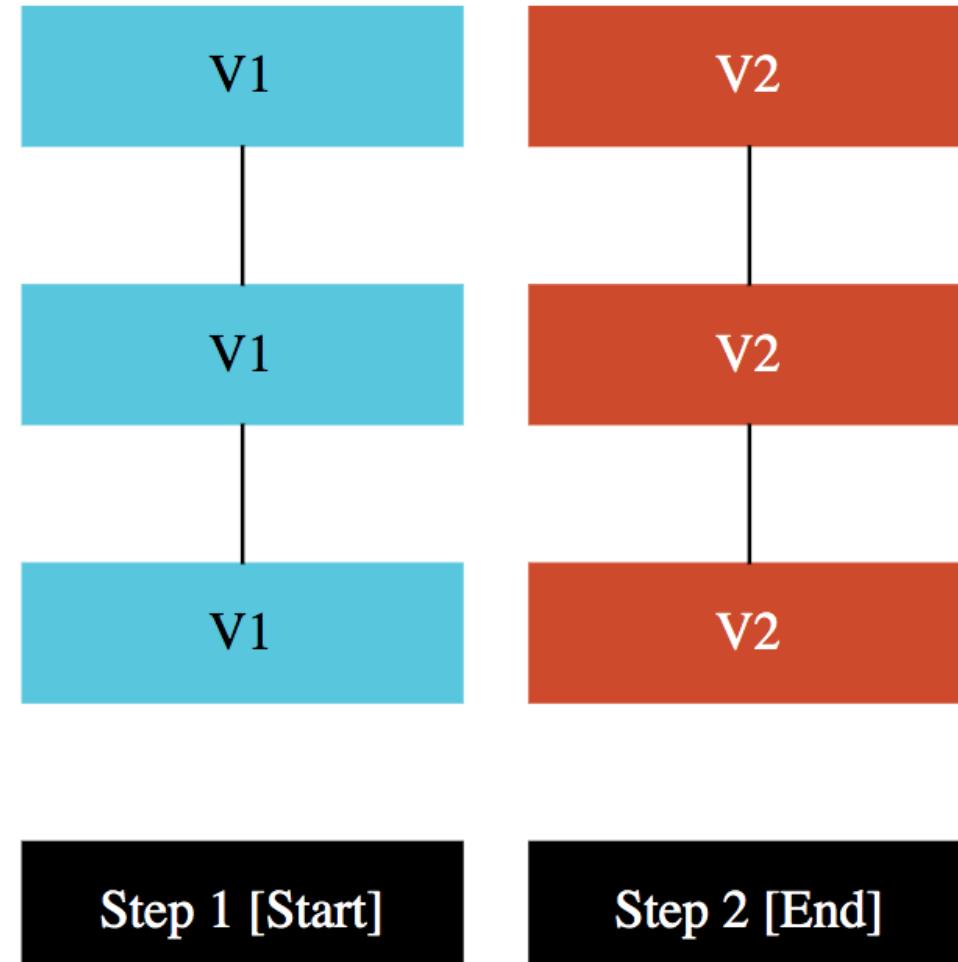
Deployment Approach : Recreate

- **Approach:**

- Terminate Version 1
- Roll out Version 2

- **Characteristics:**

- App down during the release
- Rollback needs redeployment
 - AND more downtime
- Cost effective and Fast
 - BUT disruptive
- Avoid need for backward compatibility (data and applications)



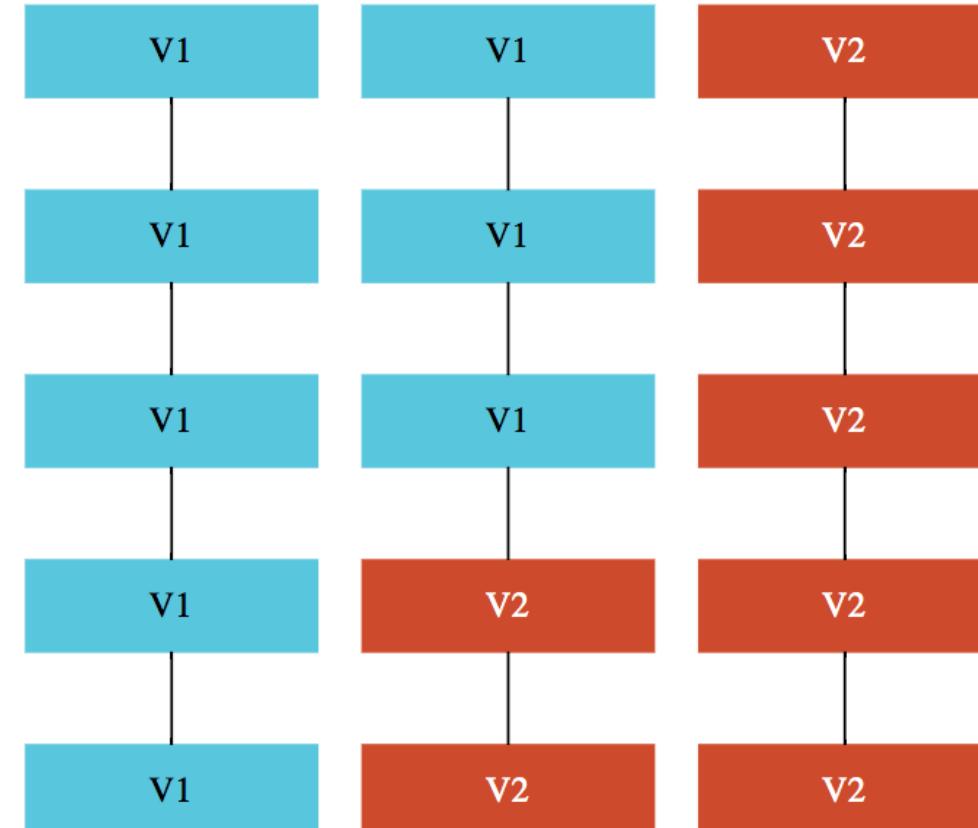
Deployment Approach : Canary

- **Approach:**

- Step 1: V2 rolled out to a subset of instances
- Step 2: On successful testing, V2 rolled out to all instances
 - OR V2 is rolled back in case of failure

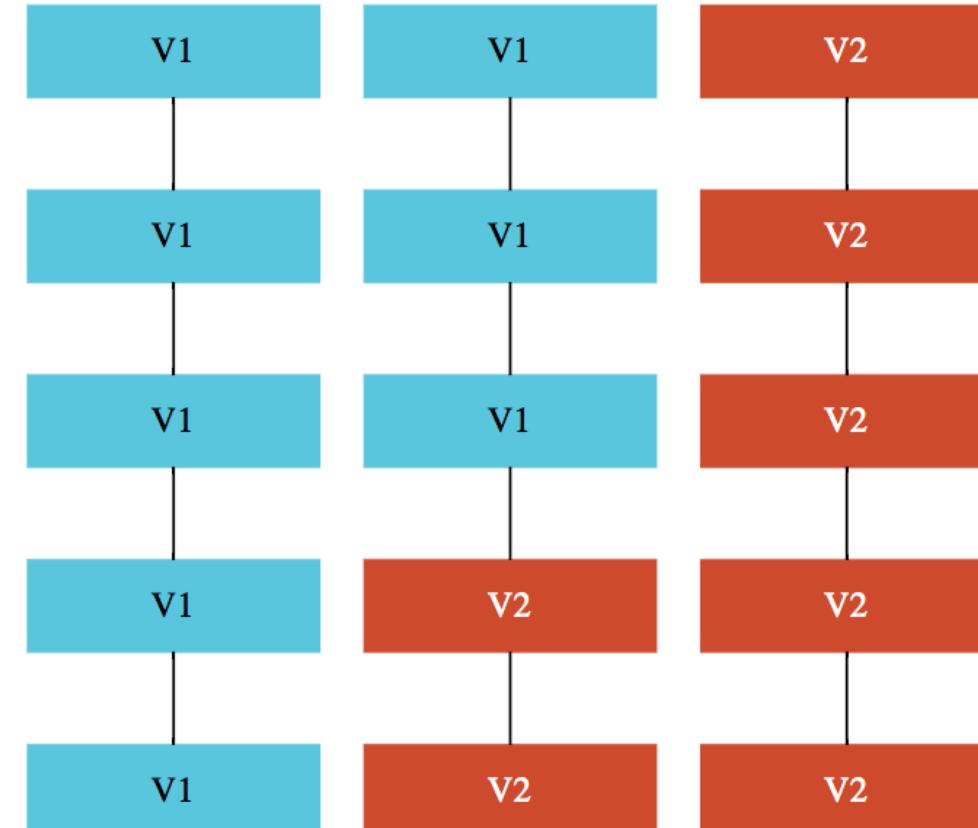
- **Characteristics:**

- Fast
- Zero downtime
- No extra infrastructure
- Minimizes impact to users (in case of release failures)
- Needs Backward compatibility (data and applications)



Testing Approach : A/B Testing

- **Use case:** You want to see if users like a feature!
- **Approach:**
 - **Step 1:** V2 (with new feature) rolled out to a subset of users
 - **Step 2:** On successful testing, V2 rolled out to all users
 - OR we go back to V1 in case users don't like the feature!
- **Characteristics:**
 - Gives the ability to test if your users like a feature



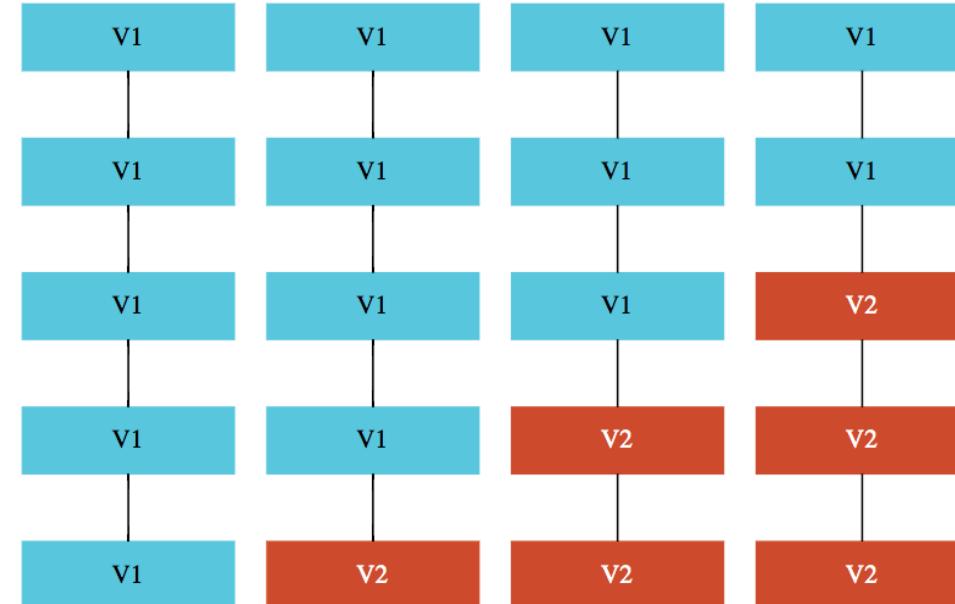
Deployment Approach : Rolling

- **Approach:**

- Step 1: V2 rolled out to a percentage of instances (Example window size: 5%)
- Step 2..N: V2 gradually rolled out to rest of the instances (Example: 5% at a time)

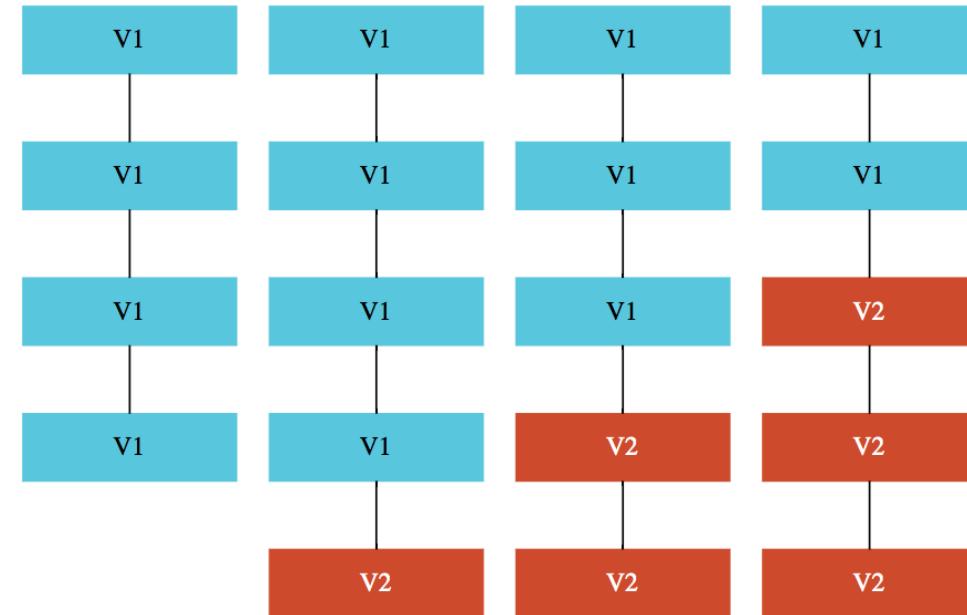
- **Characteristics:**

- Slow
- Zero downtime
- Needs automation and additional setup
- No extra infrastructure
- Minimizes impact to users (in case of release failures)
- Needs Backward compatibility (data and applications)



Deployment Approach : Rolling with Additional Batch

- **Approach:**
 - Step 1: Additional batch of new instances are created with V2 (Example: 5%)
 - Step 2..N: V2 gradually rolled out to the instances batch by batch (Example: 5% at a time)
- **Characteristics:**
 - Same as Rolling Deployment except for:
 - Needs Little bit of extra infrastructure
 - ZERO reduction in number of instances handling user requests



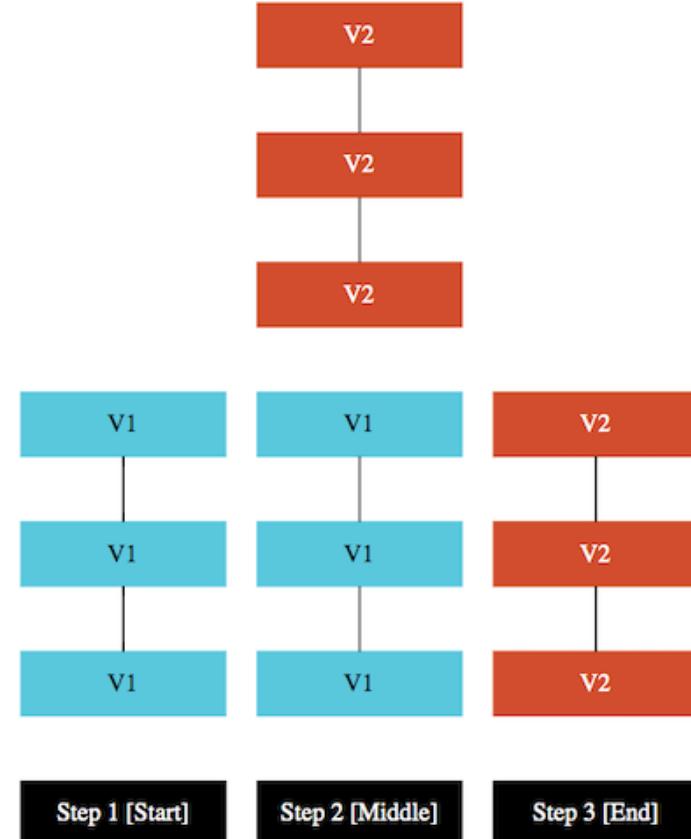
Deployment Approach : Blue Green

- **Approach:**

- Step 1: V1 is Live
- Step 2: Create (or replicate) a parallel environment with V2
- Step 3: Switch all traffic from V1 to V2 (and remove V1 Environment)

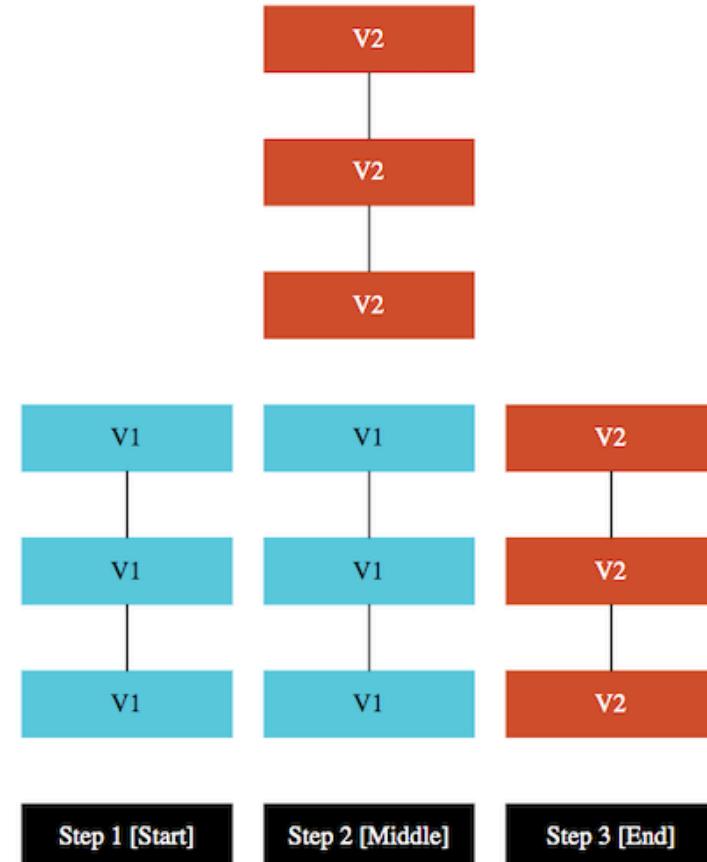
- **Characteristics:**

- Instant
- Zero Downtime
- Easy Rollback
- Needs additional infra (during the release)
- ZERO reduction in available capacity
- Needs Backward compatibility (data and apps)



Testing Approach : Shadow

- **Approach:**
 - Step 1: V1 is Live
 - Step 2: Create (or replicate) a parallel environment with V2
 - Mirror traffic to V1 and V2
 - Step 3: Switch all traffic from V1 to V2 (and remove V1 Environment)
- **Characteristics:**
 - Zero production impact: Test V2 with real production traffic before releasing
 - You can also capture and replay live production traffic
 - Complicated : You don't want double payments (might need stubbing)
 - Needs a lot of additional infrastructure



Deployment Approaches - Managed instance group (MIG)

Option	Details
Rolling Release	<pre>gcloud compute instance-groups managed rolling-action start-update my-mig --version=template=v2-template --max-surge=5 or 10% (Max instances updated at a time) --max-unavailable=5 or 10% (Max instances that can be down during update)</pre>
Canary Release	<pre>gcloud compute instance-groups managed rolling-action start-update my-mig --version=template=v1-template --canary-version=template=v2-template,target-size=10%</pre>
Blue Green Deployment	Create a new MIG and make manual adjustments to Load Balancer backends as needed

App Engine - Releasing New Versions

In 28
Minutes

Option	Details
Deploy & shift all traffic at once	<code>gcloud app deploy</code>
Deploy v2 without shifting traffic	<code>--no-promote</code>
Shift traffic to V2 all at once	<code>gcloud app services set-traffic s1 --splits V2=1</code>
Gradual Migration	Add <code>--migrate</code> option to previous command
A/B testing	<code>gcloud app services set-traffic s1 --splits=v2=.5,v1=.5</code>

GKE - Releasing New Versions

In 28
Minutes

Option	Details
Rolling Update and Recreate	Set strategy > type on Deployment
Blue Green Deployment	Create New Deployment. Control traffic using Ingress (or Service) OR you can use Spinnaker
Canary Deployment	Create New Deployment. Use Service Mesh like Istio (Anthos Service Mesh is GCP service for Istio) OR you can use Spinnaker
Spinnaker vs Cloud Build	Cloud Build or Spinnaker can be used for building Docker image (CI). Cloud Build can perform simple deployments. For advanced automated cross cloud deployments, Spinnaker is recommended

More Development!

Creating Docker Images - Dockerfile

```
FROM node:8.16.1-alpine
WORKDIR /app
COPY . /app
RUN npm install
EXPOSE 5000
CMD node index.js
```

- Dockerfile contains instruction to create Docker images
 - **FROM** - Sets a base image
 - **WORKDIR** - sets the working directory
 - **RUN** - execute a command
 - **EXPOSE** - Informs Docker about the port that the container listens on at runtime
 - **COPY** - Copies new files or directories into image
 - **CMD** - Default command for an executing container

Understanding Best Practices - Docker Image

```
FROM node:8.16.1-alpine
WORKDIR /app
COPY package.json /app
RUN npm install
EXPOSE 5000
COPY . /app
CMD node index.js
```



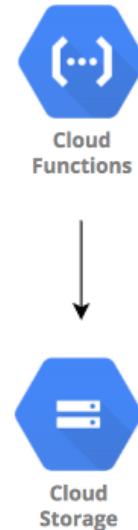
- Use light weight images (Prefer Alpine Linux to Ubuntu)
- Do not copy anything that's unnecessary (node_modules for example)
- Move things that change less often to the top to enable LAYER REUSE
 - Code changes often (index.js) BUT Dependencies change less often (package.json contains libraries that the app needs)
 - So, first build dependencies (npm install - which takes time) and then copy rest of code (COPY . /app)

Understanding Semantic Versioning - Docker Images

- Semantic Versioning Recommended Format: MAJOR.MINOR.PATCH
- How does Semantic Versioning work?
 - Increment:
 - MAJOR version: When you release major features or make incompatible API changes
 - MINOR version: When adding minor features (with backward compatibility)
 - PATCH version: For bug fixes (with backward compatibility)
- Semantic Versioning is recommended for Docker Image Tags
 - Example version: my-docker-image:1.0.0
 - Recommended naming for a future release:
 - Major Feature: my-docker-image:2.0.0
 - Minor Feature: my-docker-image:1.1.0
 - Bug Fix: my-docker-image:1.0.1
- Avoid depending on the latest Docker image tag!

Exploring Function Identity for Cloud Functions

- What are the **best practices** in Authorization for Cloud Function?
 - **Default Service Account:** App Engine default service account: PROJECT_ID@appspot.gserviceaccount.com
 - Has a broad range of permissions
 - **NOT RECOMMENDED**
 - **RECOMMENDED:** Create a **Custom Service Account** with the minimum permissions needed
 - **Example:** If Cloud Function need access to read/write from Cloud Storage, create a Service Account with following roles
 - roles/storage.objectAdmin (full control over objects, including listing, creating, viewing, and deleting objects)
 - roles/logging.logWriter (Write logs to Cloud Logging)
 - roles/monitoring.metricWriter (Write metrics to Cloud Monitoring)



Exploring Google Cloud IDE Integration - Cloud Code

- **Cloud Code:** Simplifies writing, debugging, & deploying your apps
- **Good integration** with GKE, Cloud Run and Cloud Run for Anthos
- **Get continuous feedback on your code** in real time with support for:
 - **Skaffold** (Continuous development for Kubernetes applications)
 - **Jib** (Builds optimized Docker and OCI images for your Java applications - without needing Dockerfile)
 - **kubectl**
- **Integrates with IDEs:**
 - Cloud Code for **VS Code**
 - Cloud Code for **IntelliJ**
 - Cloud Code for **Cloud Shell Editor**
 - Cloud Shell Editor comes with Cloud Code support
 - Use Case: Modify your Google Cloud configuration files directly from the browser



Google Cloud

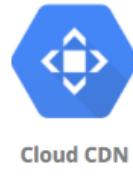
Cloud Emulators

- How do you **develop GCP applications in your local machine without connecting to GCP?**
 - Setup local development environment with Cloud Emulators
- **Supports emulation of:**
 - Cloud Bigtable
 - Cloud Datastore
 - Cloud Firestore
 - Cloud Pub Sub
 - Cloud Spanner
- You can **develop your applications locally using emulators!**

Architecture

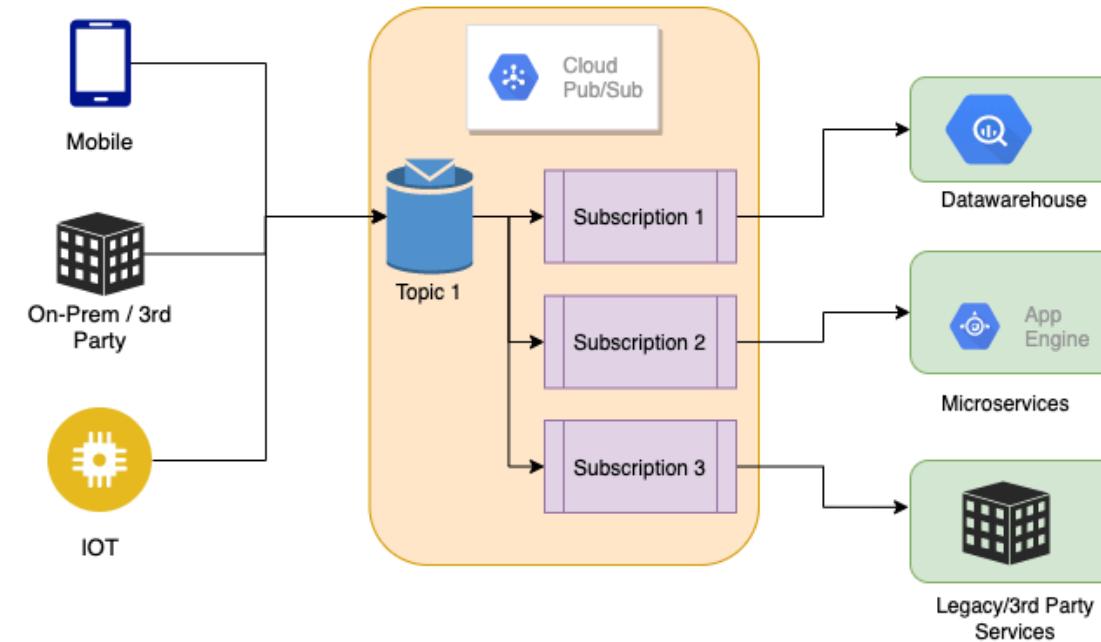
Cloud CDN - Content Delivery Network

- Use Google's global edge network to serve global content with low latency
- Integrates with **External HTTP(S) Load Balancing**
 - LB provides frontend IP addresses and ports
- **Backends can be:**
 - Cloud Storage buckets, Instance groups, App Engine, Cloud Run, or Cloud Functions
 - Endpoints outside of Google Cloud (custom origins)
- **How Cloud CDN works?**
 - External HTTP(S) Load Balancing uses proxies - Google Front Ends (GFEs)
 - Request from user arrives at a Google Front End (GFE)
 - If URL maps to a backend with Cloud CDN configured:
 - If content is found in cache(cache hit), GFE sends cached response
 - If content is NOT found in the cache (cache miss), request is forwarded to backend (origin server)
 - Response is sent to user and cached
 - Using TTL settings to control cache duration



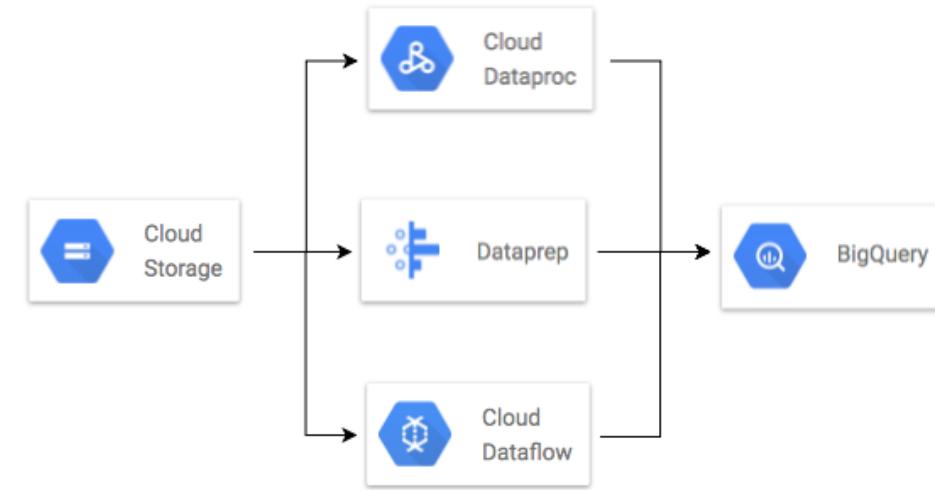
Architecture - Loose Coupling with Pub/Sub

- Whenever you want to decouple a publisher from a subscriber, consider Pub/Sub
- Pub/Sub is used in:
 - Microservices Architectures
 - IOT Architectures
 - Streaming Architectures



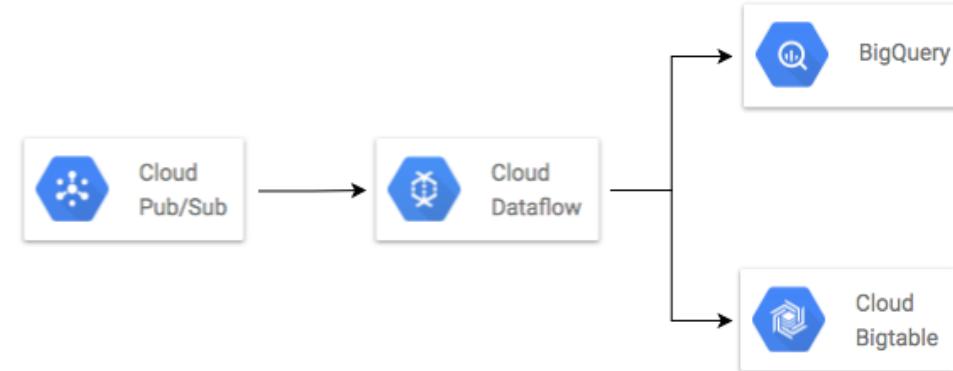
Architecture 1 - Big Data Flow - Batch Ingest

- Use extract, transform, and load (ETL) to load data into BigQuery
 - **Dataprep:** Clean and prepare data
 - **Dataflow:** Create data pipelines (and ETL)
 - **Dataproc:** Complex processing using Spark and Hadoop
- Visualization
 - **Data Studio:** Visualize data in BigQuery



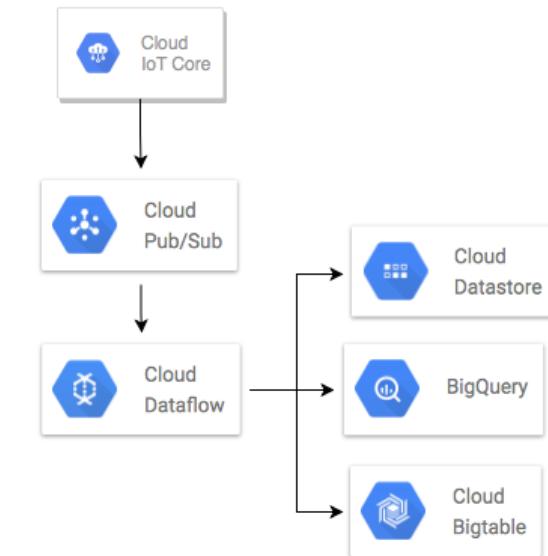
Architecture 2 - Streaming Data - Realtime Querying

- **Query data in Realtime:**
 - Pub/Sub: Receive messages
 - Dataflow: Analyze, aggregate and filter data
 - For pre-defined time series analytics, storing data in Bigtable gives you the ability to perform rapid analysis
 - For ad hoc complex analysis, prefer BigQuery



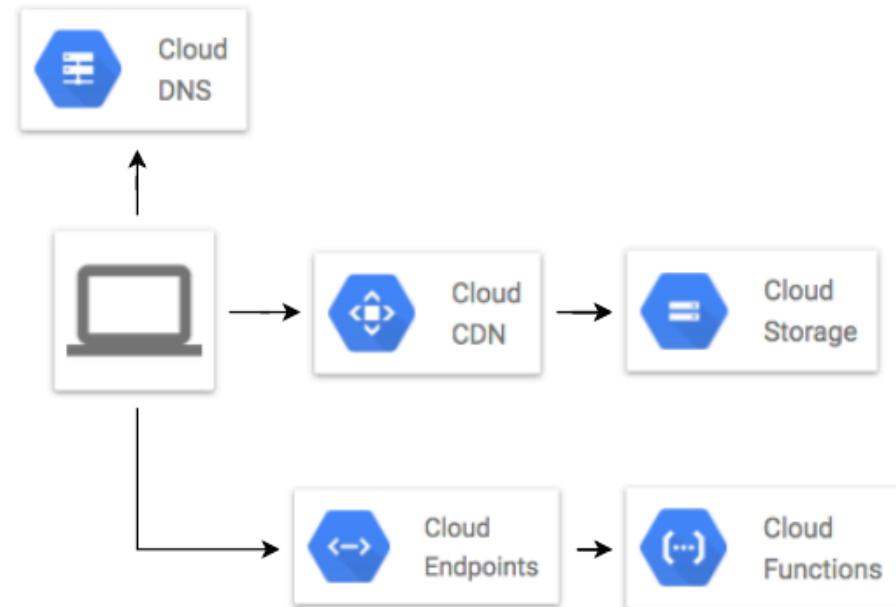
Architecture 3 - IOT

- **IoT Core:** Manage IoT (registration, authentication, and authorization) devices
 - Send/receive messages/real-time telemetry from/to IoT devices
- **Pub/Sub:** Durable message ingestion service (allows buffering)
- **Dataflow:** Processing data (ETL & more..)
 - Alternative: Use Cloud Functions to trigger alerts
- **Data Storage and Analytics:**
 - Make IOT data available to mobile or web apps => **Datastore**
 - Execute pre-defined time series queries => **Bigtable**
 - More complex or ad hoc analytics/analysis => **BigQuery**



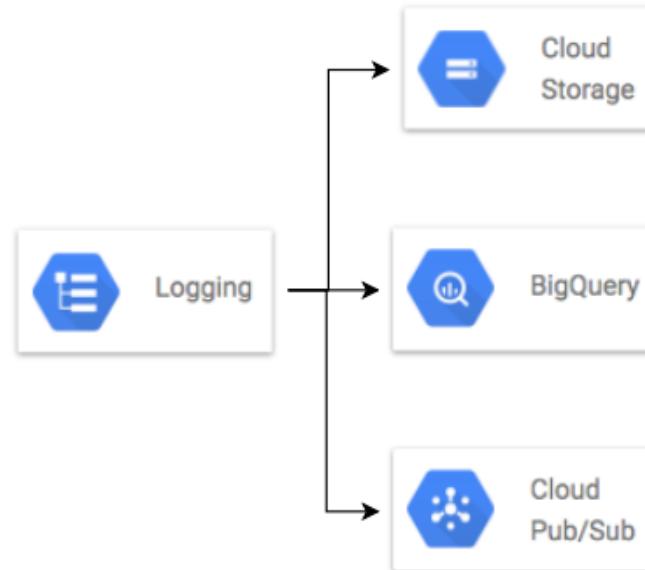
Architecture 4 - Serverless Full Stack

- **Full Stack App:** Has frontend (static) content and backend REST API
- **Static Content:**
 - **Cloud Storage:** Store static content
 - **Cloud CDN:** Content Delivery Network (caching and global distribution)
- **REST API:**
 - **Cloud Functions:** Run Business Logic
 - Alternatives: App Engine, Cloud Run
 - **Cloud Endpoints:** Expose REST API (versioning, rate limiting, ..)
 - Alternatives: Apigee, API gateway



Architecture 5 - Logging

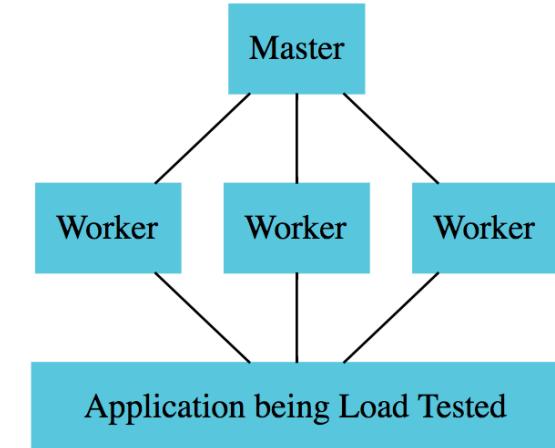
In 28
Minutes



- Send logs from Cloud Logging to different services based on your needs:
 - **Long Term Archival:** Store to Cloud Storage
 - **Real Time Analysis:** Stream to Pub/Sub (and to a real time analysis service)
 - **Long Term Log Analysis:** Send to BigQuery

Exploring Load Testing in Google Cloud

- **Load Testing:** How does an app perform under load?
 - Popular tools: JMeter, LoadRunner, Locust, Gatling etc
 - **Best Practices:**
 - Every new application release should be load tested
 - Simulate real world traffic as closely as possible
 - Test for spiky traffic - suddenly increases in traffic
- Dedicated test infrastructure is **NOT recommended:**
 - Expensive, difficult to maintain, difficult to scale
 - NOT needed continuously
 - **Recommended Architecture:** Containers and Kubernetes
 - Easy to scale
 - Example: Distributed load testing using GKE and Locust
 - Tear down the cluster after the Load Test



REST API Challenges

- Most applications today are built around REST API:
 - Resources (/todos, /todos/{id}, etc.)
 - Actions - HTTP Methods - GET, PUT, POST, DELETE etc.
- Management of REST API is not easy:
 - You've to take care of authentication and authorization
 - You've to be able to set limits (rate limiting, quotas) for your API consumers
 - You've to take care of implementing multiple versions of your API
 - You would want to implement monitoring, caching and a lot of other features..



Exploring API management in Google Cloud

- **Apigee API Management:** Comprehensive API management platform
 - Deployment options: Cloud, on-premises or hybrid
 - Manage Complete API life cycle
 - Design, Secure, Publish, Analyze, Monitor and Monetize APIs
 - Powerful Features
 - On-boarding partners and developers
 - Supports complex integrations (REST, gRPC, Non-gRPC-REST, integrate with GCP, on-premises or hybrid apps)
- **Cloud Endpoints:** Basic API Management for Google Cloud backends
 - Little complicated to setup: You need to build a container and deploy to Cloud Run
 - Supports REST API and gRPC
- **API gateway:** Newer, Simpler API Management for Google Cloud backends
 - Simpler to setup
 - Supports REST API and gRPC

Monolith to Microservices: Application Modernization



- Prefer Managed Services
- Prefer Containers and Container Orchestration
- Prefer Stateless, Horizontally Scalable Microservices
- Use Automation (DevOps, SRE - CI/CD)
- Take a step by step approach:
 - Experiment and design proofs of concept
 - Replace application features with appropriate microservices in phases

Getting Started with Identity Platform

- **Identity Platform:** Customer identity and access management
- **What's the difference:** Cloud IAM vs Identity Platform
 - **Cloud IAM:** Employees and Partners Authorization
 - Control access to Google Cloud Resources
 - Member, Roles, Policy, Service Accounts
 - **Identity Platform:** Customer identity and access management (CIAM)
 - Authentication and Authorization for your applications and services
- **Identity Platform:** Key Features
 - Authentication & authorization for web & mobile apps (iOS, Android, ..)
 - Multiple authentication methods
 - SAML, OIDC, email/password, phone, social - Google/Facebook/Twitter/..
 - Features: User sign-up and sign-in, MFA etc.
 - An upgrade from Firebase Authentication Legacy
 - Integrates well with Identity-Aware Proxy

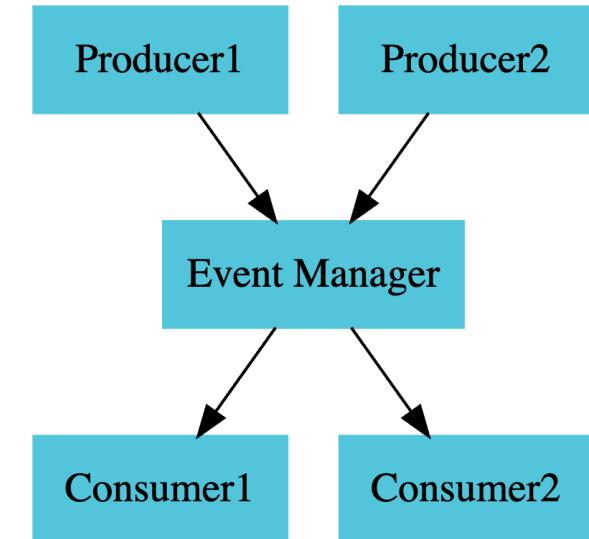


Cloud IAM vs Identity Platform - Scenarios

Scenario	Solution
An Application on a GCE VM needs access to cloud storage	Cloud IAM - Service Account
An enterprise user need access to upload objects to a Cloud Storage bucket	Cloud IAM
I want to manage end users for my application	Identity Platform
I want to enable "Login using facebook/twitter" for my application	Identity Platform
I want to create user sign-up and sign-in workflows for my application	Identity Platform

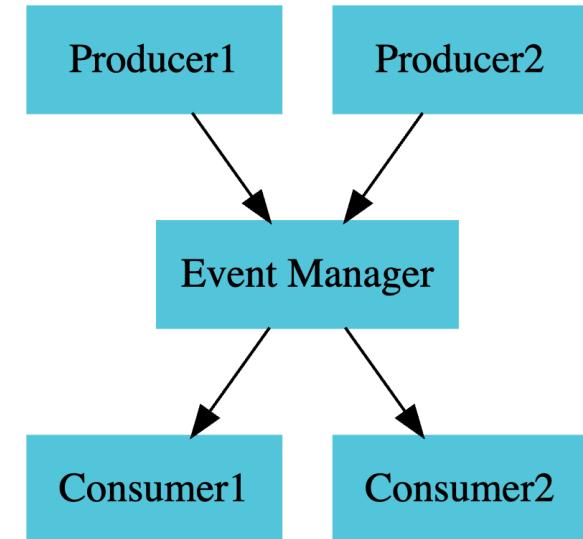
Getting Started with Event Driven Architecture

- Microservices calling each other => Tight Coupling
- **Event driven architectures:** Microservices reacting to changes in state (events)
 - **Example:**
 - 1: Order Service publishes an OrderReceived event
 - 2: Billing Service receives it and publishes an OrderBilled event
 - 3: Warehouse Service receives it & publishes an OrderReadyToShip event
 - 4: Shipping Service receives it and publishes an OrderShipped event
 - 5: Email Service receives it and sends an email to the user
 - **Advantages:**
 - **Loose Coupling:** Microservices do not know about each other
 - **Flexible Orchestration:** Same event can be processed by multiple services
 - **Resiliency:** Events can be easily retried in case of failures
 - **Asynchronous:** A microservice does not need to wait for the consumer to process the event



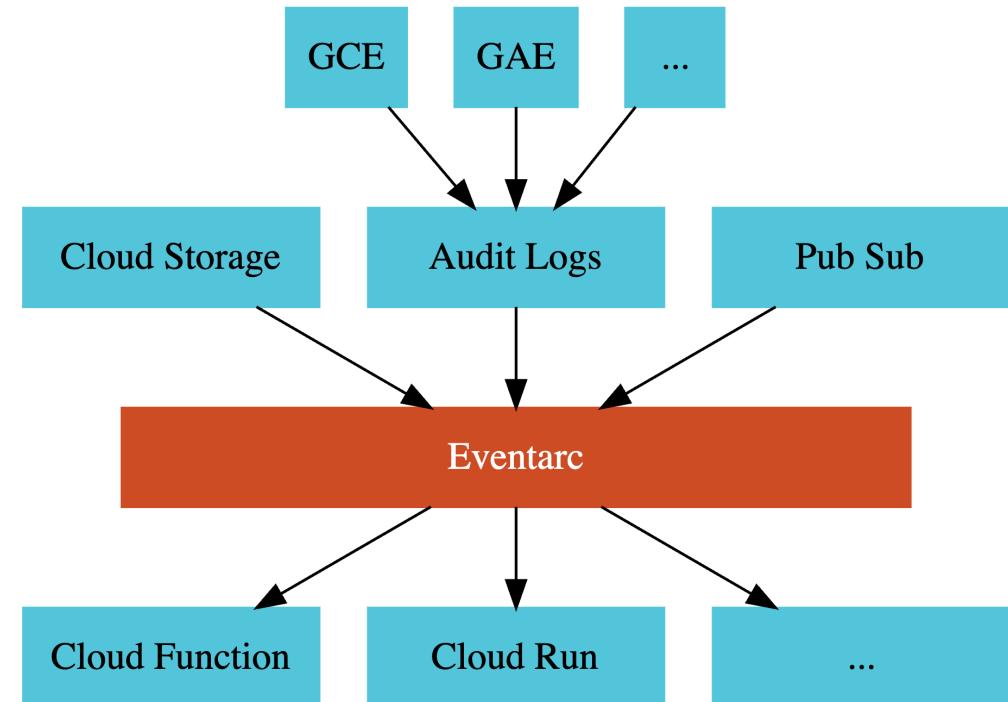
Getting Started with CloudEvents

- **CloudEvents:** Standard specification to describe events
 - Reference: <https://cloudevents.io>
 - **Challenge:** Different publishers use different formats for events
 - **Goal:** Describe event data in a standard way
 - A project under CNCF - Cloud Native Computing Foundation
 - **Advantages:**
 - **Consistency:** All events have the same structure
 - **Standard Libraries and Tooling:** Enables building common libraries and tooling across different types of infrastructure (AWS/Azure/Google Cloud/On-premise/..) and languages (Python, Go, NodeJS, Java, ..)
 - **Portability:** You are no longer tied to specific infrastructure or language
 - **Example Usecases:**
 - Get alerted when changes happen in storage buckets or database tables
 - Send one event to multiple consumers (fan out)



Getting Started with Eventarc

- **Eventarc:** Simplifies event driven architectures in Google Cloud
 - Adheres to the CloudEvents (cloudevents.io) specification
 - **Event provider:** Who can trigger events?
 - **Direct:** From Pub/Sub, Cloud Storage, Cloud Functions, Cloud IoT, Cloud Memorystore ..
 - **In-Direct:** From Cloud Audit Logs Entries
 - Huge variety of Google Cloud services. Ex: GCE, GAE, Artifact Registry, ...
 - **Event destination:** Who can process events?
 - Cloud Functions (2nd gen), Cloud Run & GKE services,..
 - (BACKGROUND) Uses Pub/Sub topics



Exploring Eventarc Event providers

```
gcloud eventarc triggers create my-pub-sub-trigger  
  --destination-run-service=$SERVICE_NAME --destination-run-region=$REGION  
  --event-filters="type=google.cloud.pubsub.topic.v1.messagePublished"

gcloud eventarc triggers create my-audit-log-trigger  
  --destination-run-service=$SERVICE_NAME --destination-run-region=$REGION  
  --event-filters="type=google.cloud.audit.log.v1.written"  
  --event-filters="serviceName=storage.googleapis.com"  
  --event-filters="methodName=storage.objects.create"
```

- **Two Types of Eventarc Event Providers:**

- **1: Directly from Google Cloud Services**
 - Pub/Sub, Cloud Storage, Cloud Functions, Cloud IoT, Cloud Memorystore ..
 - type: google.cloud.storage.object.v1.archived/deleted/finalized
 - type: google.cloud.pubsub.topic.v1.messagePublished
- **2: Indirectly from Cloud Audit Logs Entries**
 - type : google.cloud.audit.log.v1.written
 - serviceName:appengine.googleapis.com, methodName:google.appengine.v1.Applications.CreateApplication
 - serviceName:artifactregistry.googleapis.com, methodName:google.devtools.artifactregistry.v1.ArtifactRegistry.CreateRepository
 - serviceName:compute.googleapis.com, methodName:compute.instances.delete

Getting Started with Observability and OpenTelemetry

- **Observability:** "measure the internal state of a system by examining its outputs"
 - **Goal:** Proactively identify problems and fix them
 - **THREE PILLARS** of observability: logs, metrics and traces
 - Earlier we had different standards for logs, metrics and traces
 - We also had very different approaches across languages
- How about **ONE STANDARD ACROSS PLATFORMS?**
 - **OpenTelemetry:** Collection of technologies (tools, APIs, SDKs) to collect and export telemetry - metrics, traces, and logs (<https://opentelemetry.io>)
 - Open standard
 - A CNCF - Cloud Native Computing Foundation - Project (Kubernetes is another CNCF project)
 - Almost every cloud platform supports OpenTelemetry
 - Steps to use OpenTelemetry:
 - 1: Add OpenTelemetry libraries (for your specific language) to your project
 - 2: Instrument your code to export telemetry



Logging

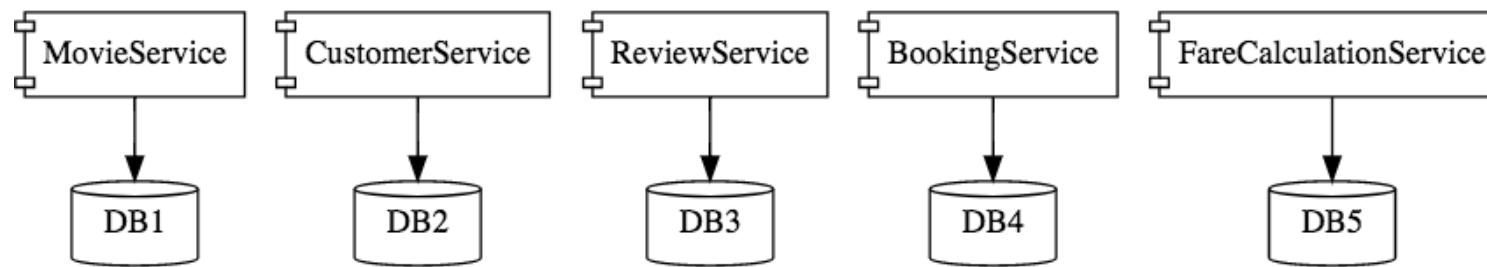


Trace



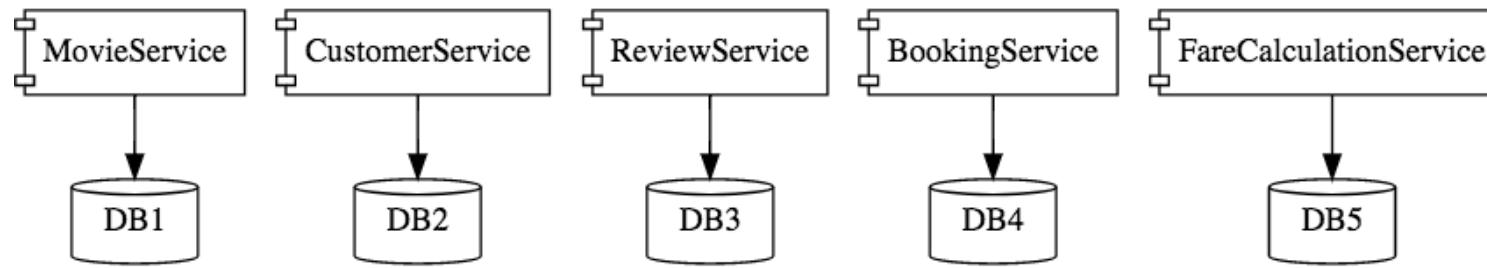
Monitoring

Using Service Directory



- **Service Discovery** - Help microservices find one another
- **Service Directory** - A single place to publish, discover, and connect services
- Your workloads can be running in:
 - Google Cloud
 - (Compute Engine VMs, Google Kubernetes Engine, ..)
 - On-prem
 - Other clouds - AWS, Azure, ..

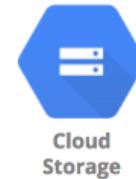
Using Service Directory - Features



- **Managed Service** (highly available and scalable)
- Register/resolve services using **DNS, HTTP, and gRPC**
- Service Directory **client libraries** are available for multiple languages (along with REST/RPC APIs)
 - You can use these libraries to register/resolve service location
- **Audit logging** (“Who did what, where, and when?” - Cloud Logging)
- **Request/response logs** (Cloud Logging)

Cloud Storage - Command Line - gcloud storage

- Earlier, **gsutil** was the recommended CLI for Cloud Storage
 - **GCLOUD STORAGE** is now the recommended CLI for Cloud Storage
 - Advantages:
 - Up to 94% faster storage transfers
 - Better parallel processing
 - Do NOT worry about options/parameters/flags
 - **gcloud storage** will decide the optimal storage transfer approach for you
 - Provides very simple to remember commands (consistent with gcloud):
 - **gcloud storage buckets create gs://BKT_NAME** (Create Cloud Storage bucket)
 - options: --default-encryption-key, --default-storage-class
 - **gcloud storage buckets delete gs://BKT_NAME**
 - **gcloud storage buckets list gs://B***
 - **gcloud storage buckets describe gs://BKT_NAME**
 - **gcloud storage buckets update gs://BKT_NAME**
 - options: --default-encryption-key, --default-storage-class, --[no-]versioning
 - If you have existing scripts that make use of gsutil commands AND
 - You do NOT want to change the scripts AND
 - You want the performance benefits offered by new features in **gcloud storage**
 - Check out **shim** (In boto configuration file, configure use_gcloud_storage=True under GSUtil section)



Case Study

Getting Started with Case Study - Remember

- Case Study is **very important** for the certification exam
 - Download - <https://cloud.google.com/certification/guides/cloud-developer>
 - You will be able to read the case study during the exam:
 - BUT I would NOT recommend depending on it
 - Have a good overview of the case study before you go to the exam
- **Best Attitude:**
 - Test yourselves using the case study:
 - Knowledge of Google Cloud Services
 - Download the case study and spend sometime with it
 - Do your analysis and form your opinions
 - Do NOT treat my recommendations as final! (Think and challenge them!)
- **Exam Tip:**
 - Try to group questions for the case study and answer them together

Case Study - HipLocal - Overview

- **Hyper-local community app:**
 - Facilitate communication between people live near each other
 - Launched recently in Dallas and expanding worldwide
 - Expecting rapid growth
 - **Goal:** Provide a great experience for new local and virtual communities
- **Current Architecture:**
 - **Hybrid Cloud:** on-premises hardware + some infrastructure running in Google Cloud
 - APIs running on Compute Engine virtual machine instances
 - **State** stored in single instance MySQL database in Google Cloud
 - No consistent logging
 - **Manual Deployment** on weekday evenings (periods of slow traffic)

Case Study - HipLocal - Requirements

- **Going Global:**
 - Support 10x concurrent users (Must scale to meet user demand)
 - Expand to new locations while ensuring compliance with local regulations (ex: GDPR)
 - Provide consistent user experience when users travel to different locations
 - Hiring and train a global team to support users in different time zones
- **Reduce infrastructure management time and cost**
 - **Adopt the Google-recommended best practices**
 - Standardize application lifecycle workflows and processes
 - Faster and more accurate validation of new features
 - Define SLIs and SLOs
 - **Analyze and respond to issues quickly**
 - Gather application user activity metrics
 - Actionable Logging and performance metrics (with alerts)
- **Security:** Secure communication between on-premises and cloud
- **Security:** APIs require authentication and authorization

Case Study - HipLocal - Discussion

Service	Discussion
Managed Instance Groups	Simplify management of multiple VMs. Allows auto scaling of VM instances based on load.
Cloud Build	Continuous Integration and Continuous Deployment
Cloud Monitoring agent	Get additional metrics from VMs
Cloud Logging Agent	Gather application logs from VMs
Cloud Monitoring	Alerts.Uptime checks.
Cloud Logging	Centralized Logging. Logs Metrics.
Cloud Endpoints or Apigee	Expose APIs to consumers
OpenCensus (RECOMMENDED) or Cloud Monitoring API	Capture app user activity metrics

Case Study - HipLocal - Discussion - 2

In 28
Minutes

Service	Discussion
Cloud SQL & Cloud SQL Proxy	Managed MySQL database
Memorystore	Can be used as state store. Also possible to use Cloud SQL or Cloud datastore as state store based on the needs.
Cloud Identity-Aware Proxy	Authenticate app users (internal/external)
BigQuery	Store user activity and analyze trends. Store and analyze logs.
Adopt SRE	General recommendation
Cloud Interconnect	High speed, highly available, low-latency private connection into Google Cloud from your company's on-premises network

Get Ready

Google Cloud - Recommended Resources

In 28
Minutes

Title	Link
Cloud Architecture Center	https://cloud.google.com/architecture/
Google Cloud Solutions	https://cloud.google.com/solutions/
Best Practices for Enterprise Applications	https://cloud.google.com/docs/enterprise/best-practices-for-enterprise-organizations
Building Scalable and Resilient Web Applications on Google Cloud Platform	https://cloud.google.com/solutions/scalable-and-resilient-apps

Cloud Developer - Certification Resources

Title	Link
Home Page	https://cloud.google.com/certification/cloud-developer
Exam Guide	https://cloud.google.com/certification/guides/cloud-developer
Case Studies	https://cloud.google.com/certification/guides/cloud-developer
Sample Questions	https://cloud.google.com/certification/sample-questions/cloud-developer
Registering For Exam	https://support.google.com/cloud-certification/#topic=9433215

Cloud Developer - Certification Exam

In 28
Minutes

- 60 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 after exam completion
 - Email (a couple of days later)
- My Recommendations
 - Time Management is critical
 - One of the few exams where I took the complete allotted 120 minutes!
 - Flag questions for future consideration (Review before final submission)
 - Read the entire question
 - Identify and write down the **key parts of the question**
 - Focus on the constraints in the question:
 - Example: Most cost-effective way - Prefer regional to multi-regional if it satisfies all requirements
 - TIP: Answer by Elimination!

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 Professional Cloud Developer
- 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?

FASTEST ROADMAPS

in28minutes.com

