O'REILLY

Google Cloud Professional Cloud Architect Crash Course

Prerequisites

- Familiarity with cloud platforms (AWS, Azure)
- Basic familiarity with the GCP

- This training focuses on breadth
- Concepts, fundamentals, and applications

Day 1: Course Schedule

- Big Picture: Terms and concepts
- Resource Hierarchy on Google Cloud
- Infrastructure-as-a-Service
  - Google Compute Engine
  - Managed Instance Groups
- Serverless Compute
  - Cloud Run
  - Cloud Run Functions
  - App Engine
- Google Kubernetes Engine (GKE)
- Load Balancing

Day 2: Course Schedule

- Networking on the Google Cloud
- Interconnecting Networks
- Storage Solutions
- Identity and Access Management
    - IAM Best Practices
- Security Features Overview
- Logging and Monitoring

Introductions

I have experience with the Google Cloud Platform:

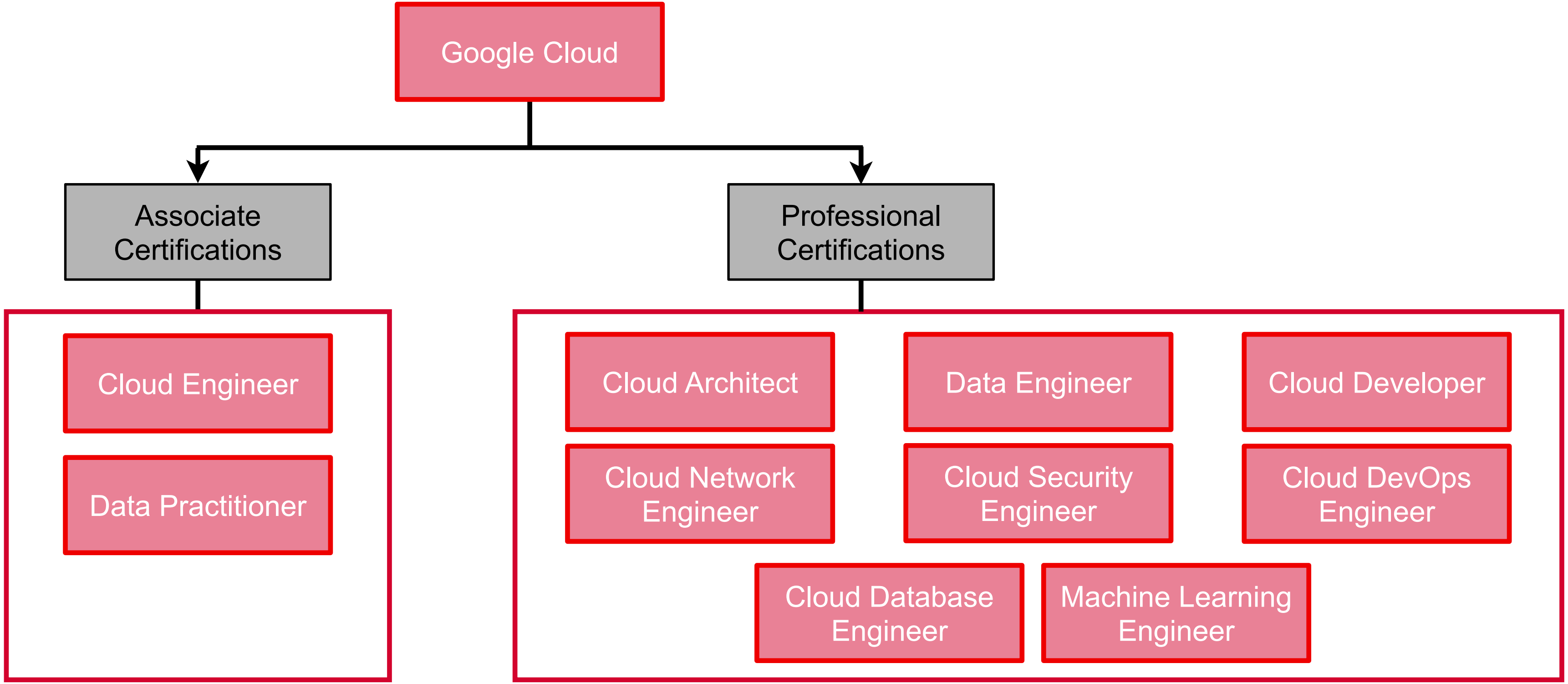
1. No experience at all
2. 0-1 years of experience
3. 2-3 years of experience
4. 3+ years of experience

Introductions

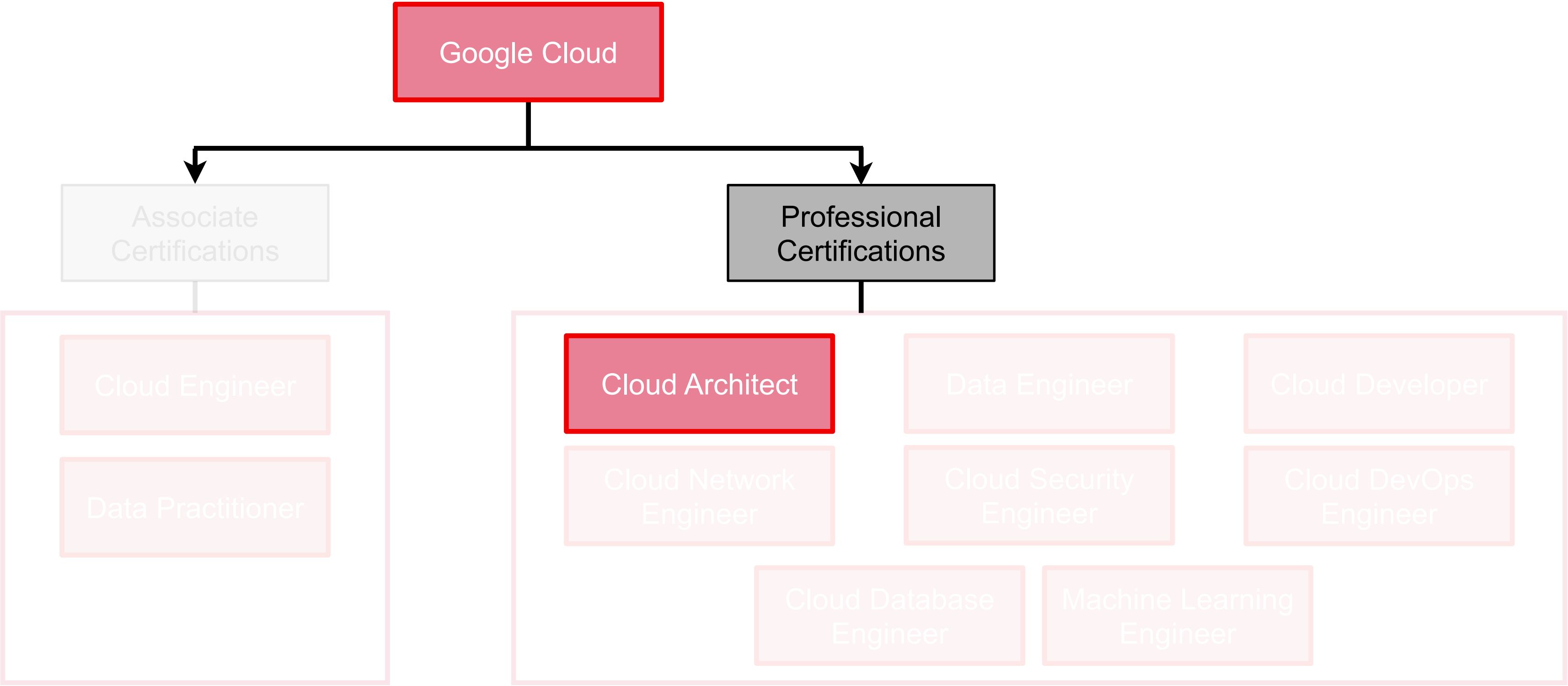
I have worked on other cloud platforms:

1. Mostly AWS
2. Mostly Azure
3. Mostly Oracle
4. Mostly IBM
5. Other cloud platforms

Google Cloud Certifications



Google Cloud Certifications



Professional Cloud Architect

- Test duration: 2 hours
- Registration fee: $200 + taxes
- Languages: English, Japanese
- Exam format: 50-60 multiple choice and multiple select questions
- Case Studies: 2 case studies in each exam - make up 20-30% of the exam
- Recommended: 3+ years industry experience, 1+ year designing and managing solutions on GCP

Professional Cloud Architect

- Vast array of services for a wide variety of use cases
- A good understanding of the specialized strengths of each service
- Main exam link:
  - https://cloud.google.com/certification/cloud-architect

Professional Cloud Architect

- Cloud Architect Certification training path:
  - https://www.cloudskillsboost.google/paths/12
- Case studies link here:
  - https://cloud.google.com/certification/guides/professional-cloud-architect/
- Extensive labs for hands-on practice:
  - https://codelabs.developers.google.com/?cat=Cloud
- Sample test:
  - https://docs.google.com/forms/d/e/1FAIpQLSf54f7FbtSJcXUY6-DUHfBG31jZ3pujgb8-a5io_9biJsNpqg/viewform?usp=sf_link

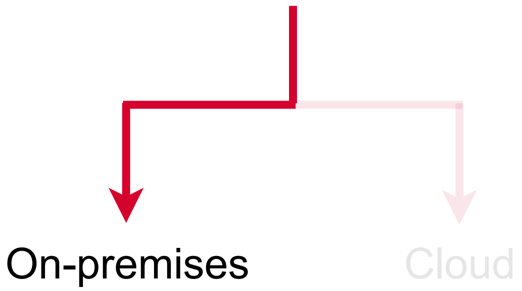
Big Picture Basics: Concepts and Terms

Building Software

On-premises

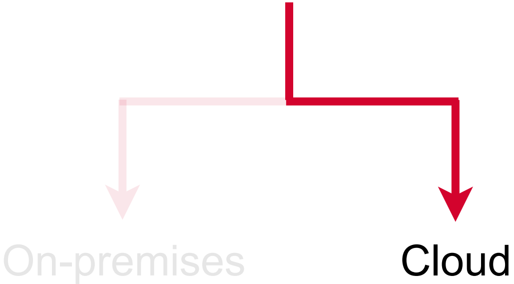
Cloud

On-premises



- Infrastructure (servers, networks) located at and managed by the organization
- Complete control over hardware, software, data
- Scaling requires purchasing new hardware

Cloud

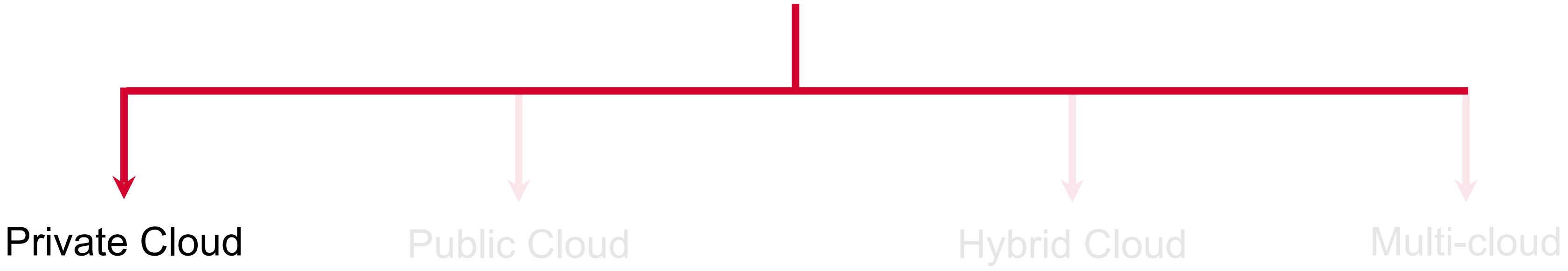


- Infrastructure hosted by a third-party provider and accessed over the internet
- Provider manages infrastructure, while customer manages applications and data.
- Resources can be scaled up or down on demand, providing elasticity and agility.

Types of Cloud

- Private Cloud
- Public Cloud
- Hybrid Cloud
- Multi-cloud

Private Cloud



- Cloud infrastructure dedicated to a single organization.
- Hosted on-premises or by a third-party provider.
- **Best for:** Organizations with strict security, compliance, or data sovereignty requirements

On-prem: Location of the infrastructure

Private Cloud: Architecture and model of service delivery

Public Cloud

Private Cloud | Public Cloud | Hybrid Cloud | Multi-cloud

- Services delivered over the public internet, anyone can access and use
- Amazon Web Services (AWS), Microsoft Azure, Google Cloud.
- **Best for:** Scalability, cost-effectiveness, and a wide range of services

Hybrid Cloud

Private Cloud | Public Cloud | Hybrid Cloud | Multi-cloud

- Combines public cloud + on-premises
- **Best for:** Leveraging existing on-premises investments, security for certain apps, scalability for others

Multi-cloud

Private Cloud | Public Cloud | Hybrid Cloud | **Multi-cloud**

- Multiple public cloud providers
- Hybrid cloud involves on-premises
- **Best for:** Avoiding vendor lock-in, leveraging the best services from different providers

CapEx and OpEx

CapEx

OpEx

CapEx: Associated with On-prem Deployments

CapEx

OpEx

Funds used by a company to acquire, upgrade, and maintain servers and data centers. Upfront expenditure depreciated over time

OpEx: Associated with Cloud Deployments

CapEx

OpEx

The ongoing costs for a company to run its day-to-day operations. Pay-as-you-go cloud computing costs. Regular, predictable, ongoing expenses

Trade-offs: On-prem vs. Cloud

Factor	On-Premises	Cloud
Cost	High upfront CapEx, predictable ongoing costs.	Low upfront cost, pay-as-you-go OpEx.
Control	Full control over all aspects of infrastructure.	Shared control with the cloud provider.
Security	Full responsibility for security.	Shared responsibility with the provider.
Scalability	Limited and requires manual intervention.	Elastic and on-demand.
Performance	Dependent on internal hardware and expertise.	High-performance options are available, but latency can be a factor depending on internet connectivity.

Cloud Migration Terms

Lift-and-Shift (Rehost)	Lift-and-Refresh (Re-platform)
Refactor (Re-architect)	Repurchase and Retire

Lift-and-shift

Lift-and-Shift (Rehost)	Lift-and-Refresh (Re-platform)
Refactor (Re-architect)	Repurchase and Retire

Move existing applications unchanged into the cloud (quickest path, minimal refactoring, but may not leverage cloud-native benefits).

Lift-and-Refresh

Lift-and-Shift (Rehost)	**Lift-and-Refresh (Re-platform)**
Refactor (Re-architect)	Repurchase and Retire

Make minimal changes such as moving to a managed database to optimize for cost or operations without full redesign

Cloud Migration Terms

Lift-and-Shift (Rehost)	Lift-and-Refresh (Re-platform)
Refactor (Re-architect)	Repurchase and Retire

Redesign applications to be cloud-native (microservices, containerization, serverless), unlocking full complete benefits but at higher upfront effort

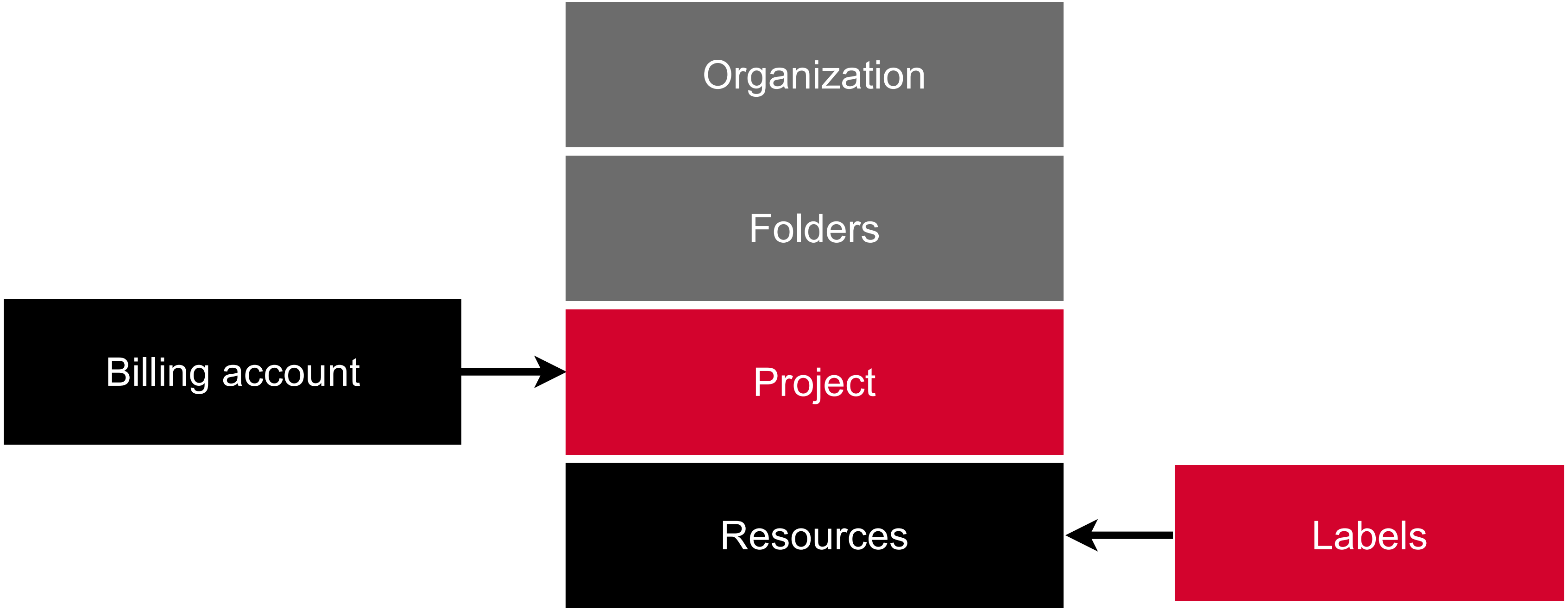
Cloud Migration Terms

Lift-and-Shift (Rehost)	Lift-and-Refresh (Re-platform)
Refactor (Re-architect)	Repurchase and Retire

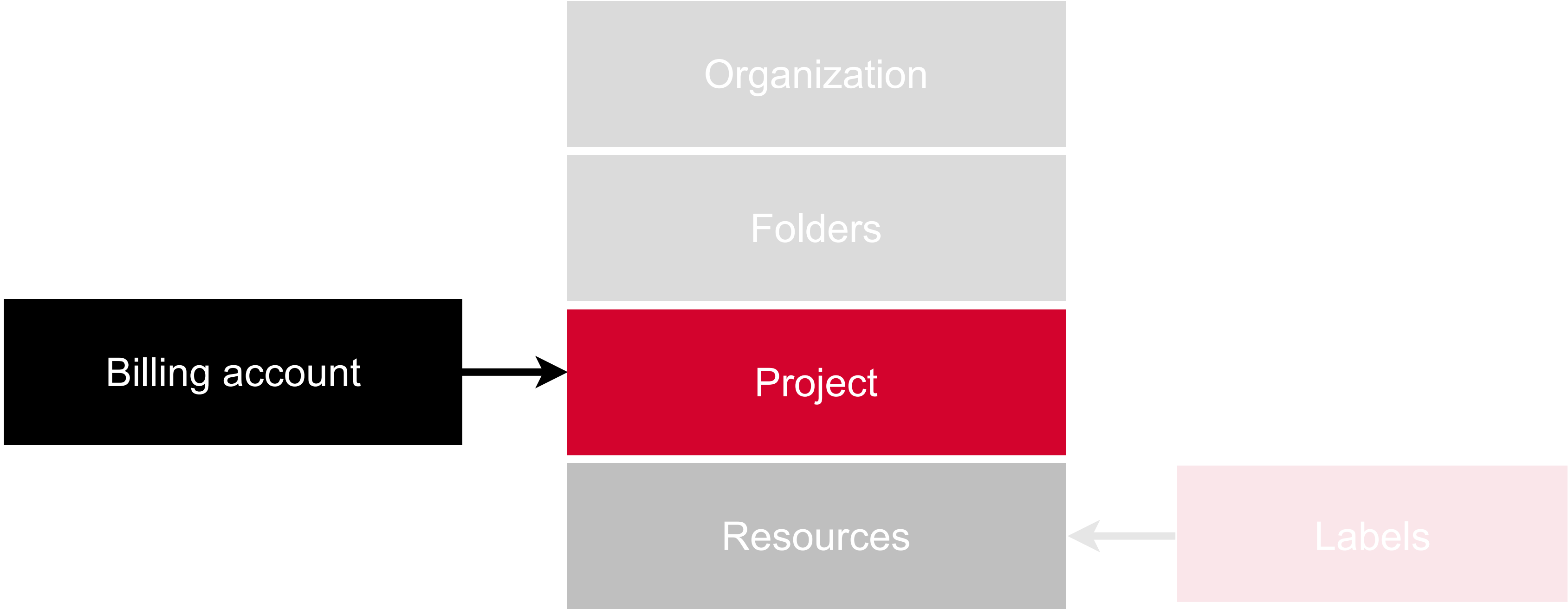
replace legacy systems with SaaS solutions where appropriate or decommission applications no longer needed

Google Cloud Platform Basics

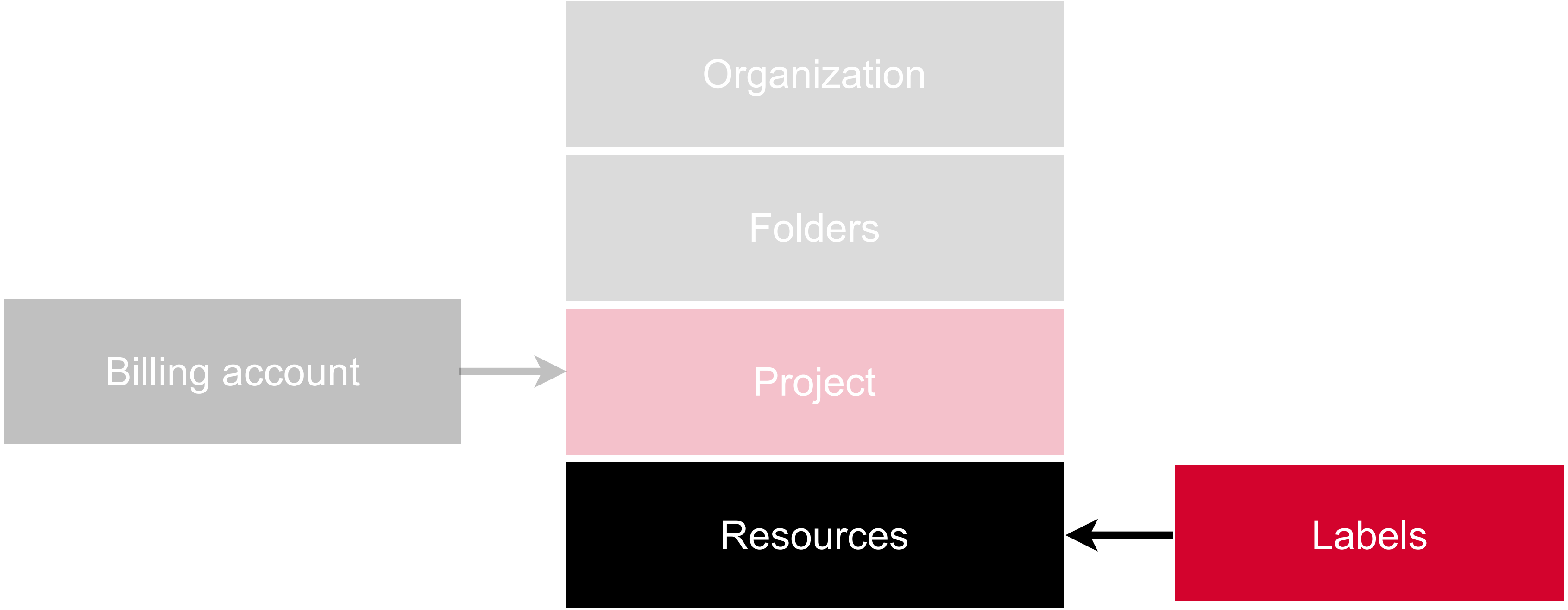
Resource Hierarchy of Components



Billing Accounts Are Associated with Projects



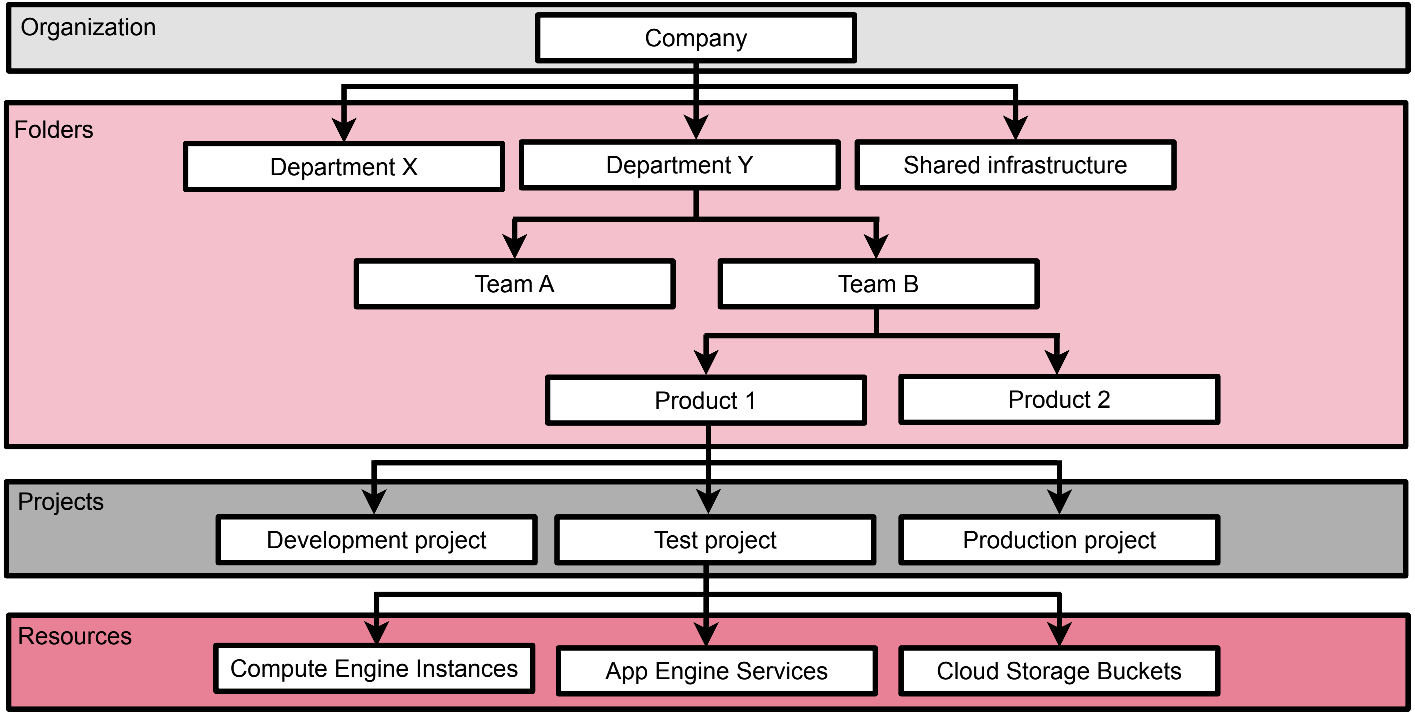
Labels Are Applied to Resources



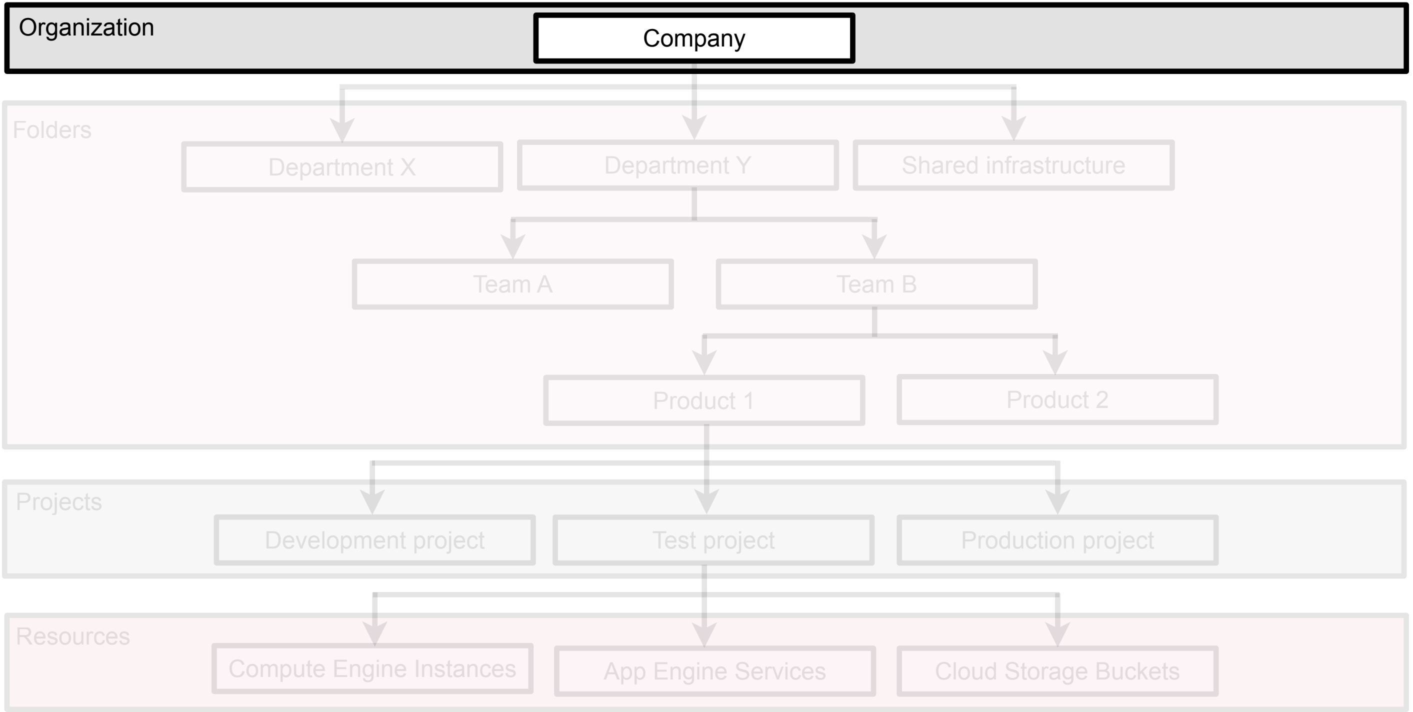
Labels Help in Allocating Costs

- Categorize resources
  - Different environments
  - Different projects
- Label resources accordingly
  - *env=dev, env=prod*
  - *service=search, service=catalog*
- Can export billing to BigQuery and analyze costs using labels

Google Cloud Hierarchy

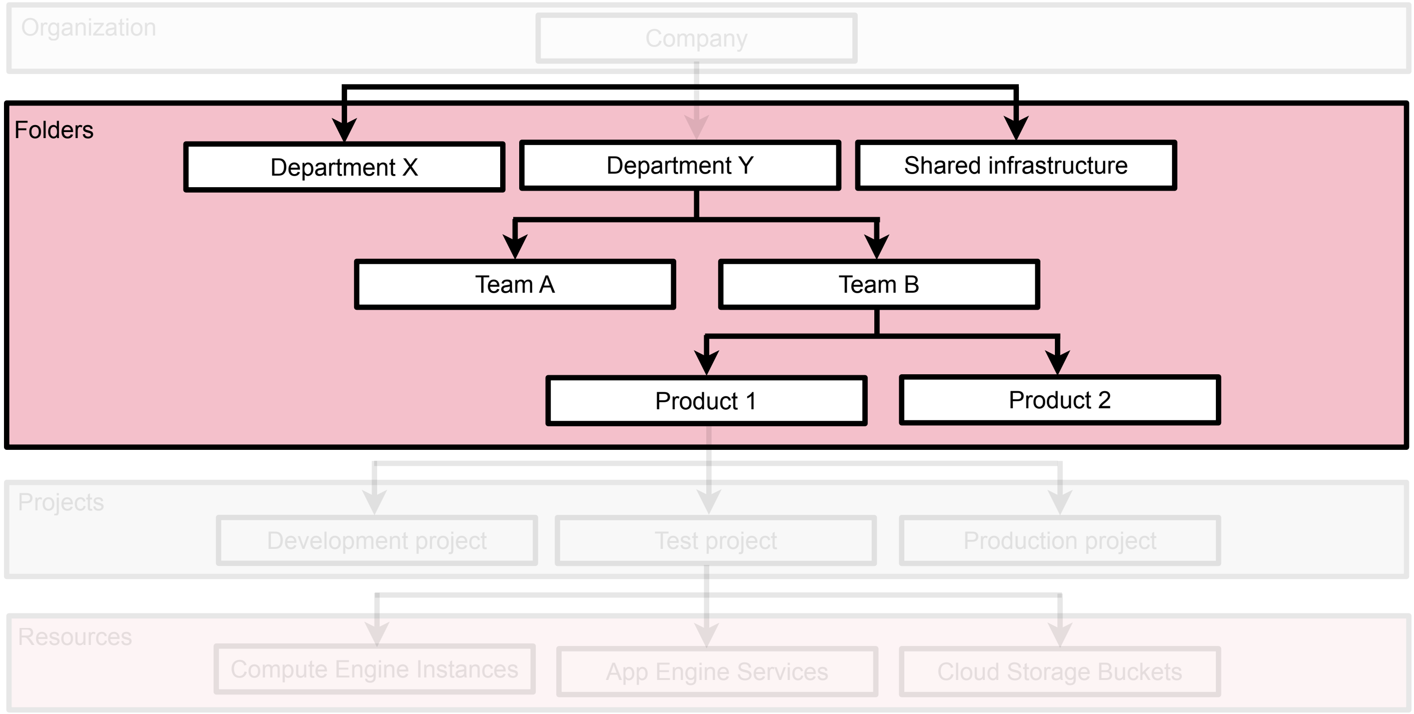


Organization



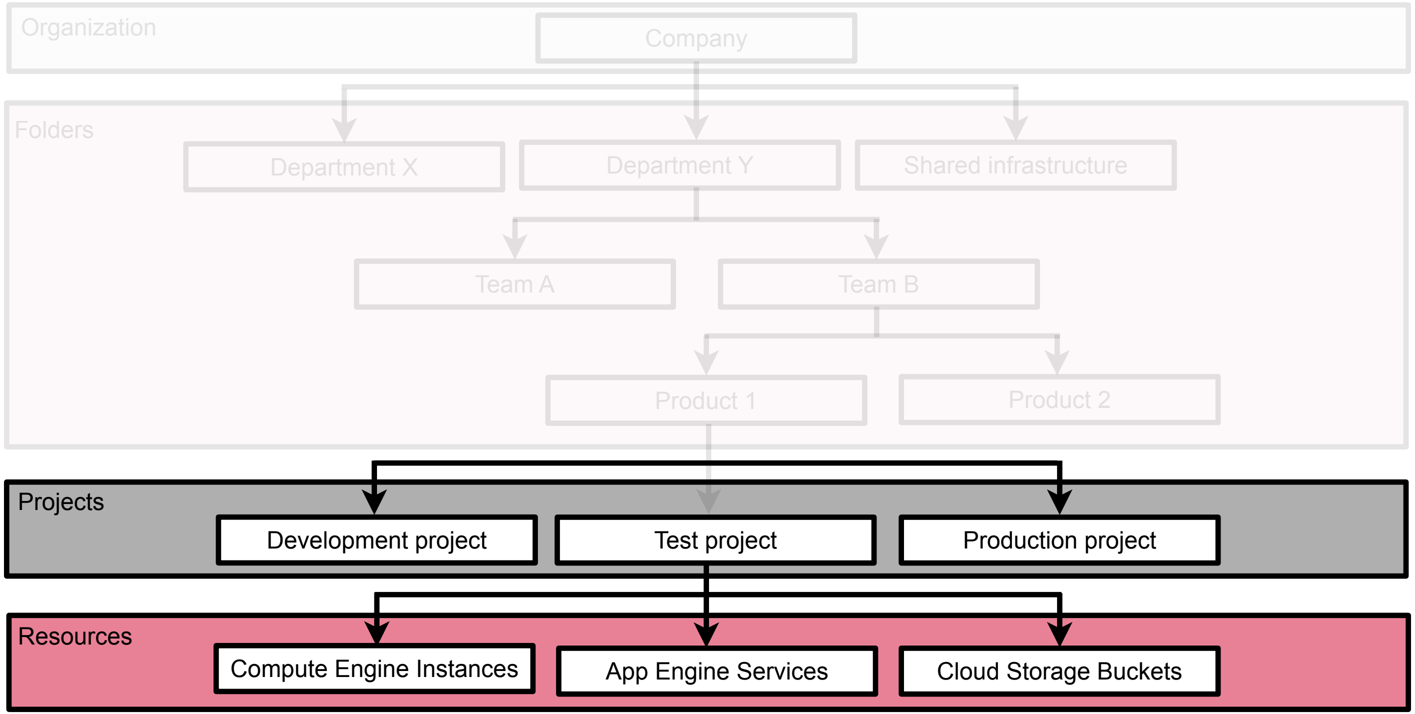
Typically one organization at the root of the hierarchy as a best practice.
Mergers and acquisitions may result in multiple organizations

Folders



Model one folder per department especially if the permissions for department members apply to all projects in a folder

Projects



Environments should always be separated into different projects to be able to manage permissions and resource capacities

Using Google Cloud Resources

Cloud Console

Cloud Shell

Command-line Tools

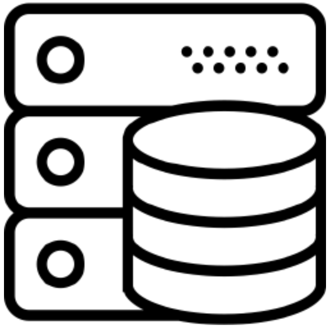
gsutil, bq

APIs and Client Libraries

Choices in Computing

Compute

Where is code executed and how?



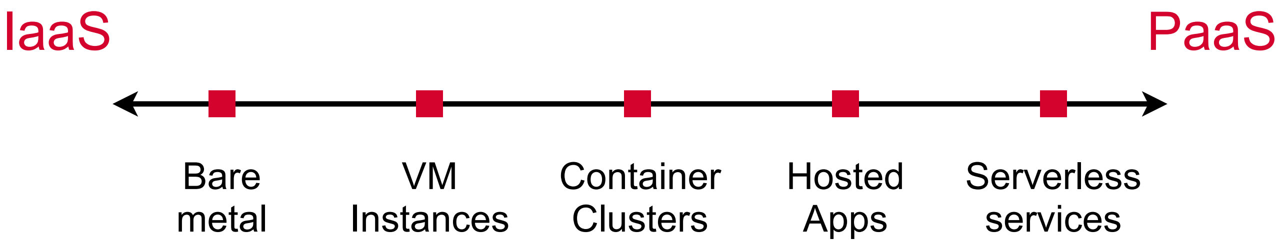
Storage

Where is data stored?

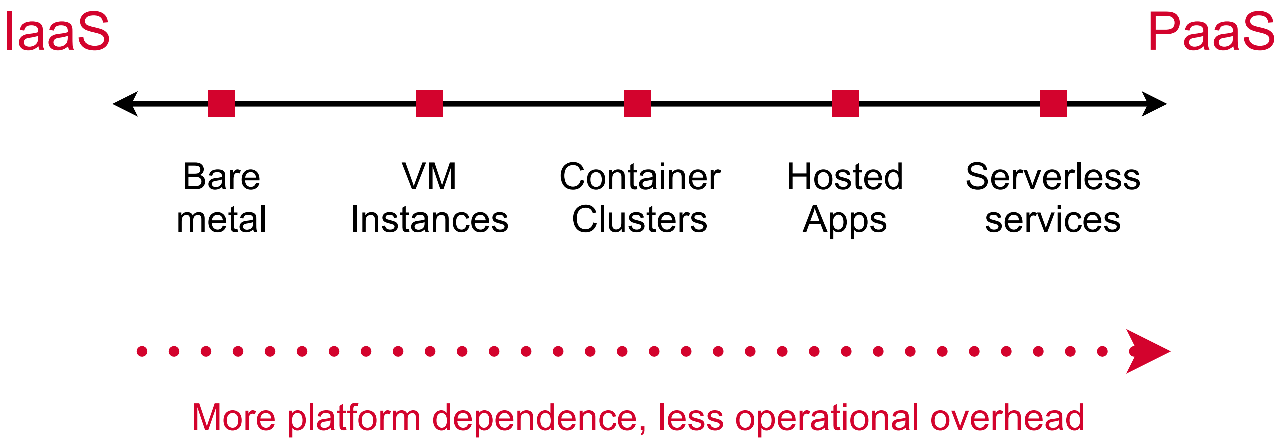
Networking, logging, are choices made after this fundamental decision

Compute Choices

Compute Choices



Compute Choices



Compute Choices

IaaS

PaaS

Bare metal

VM Instances

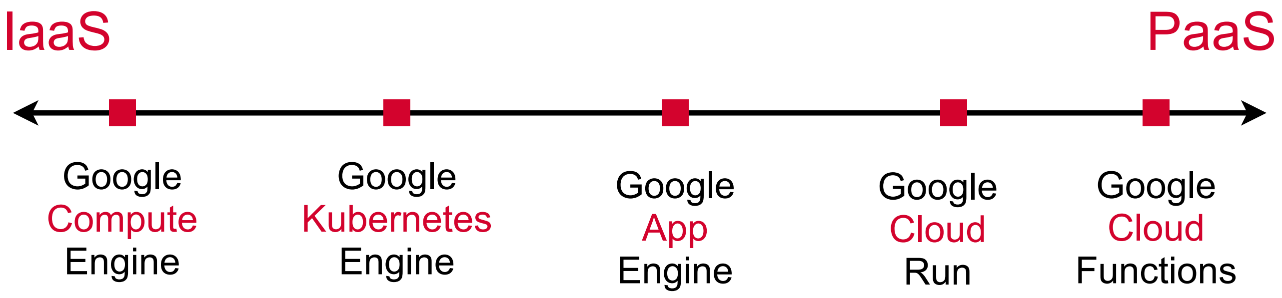
Container Clusters

Hosted Apps

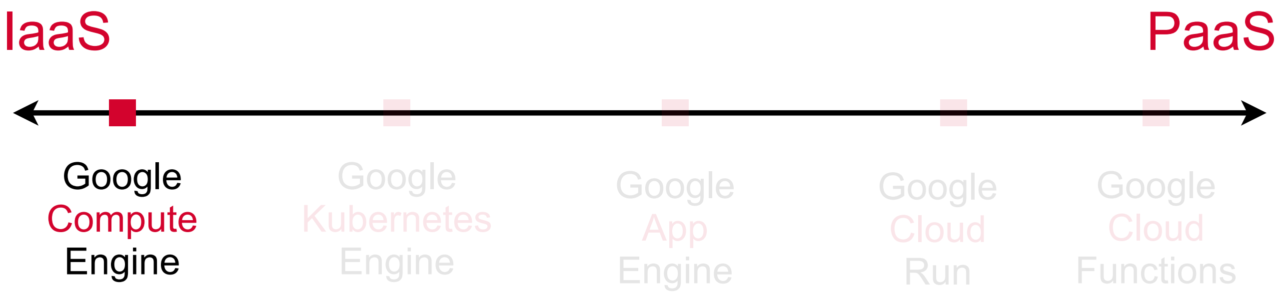
Serverless services

More control, granular access, more administrative overhead

Google Cloud Compute Choices



Google Cloud Compute Choices



Introduction to Google Cloud

As a cloud engineer frequently moving between different computers and locations, you require consistent, daily access to your Google Cloud project's resources, specifically BigQuery, Bigtable, and Kubernetes Engine. Your priority is to have a ready-to-use command-line environment with the gcloud tool without the overhead of installing, configuring, and maintaining it on every machine you use. Which of the following provides the most efficient and hassle-free solution?

A. Install the gcloud CLI on each workstation you use and create a script to automate the update process across all machines.
B. Utilize a system package manager, such as apt or yum, to streamline the gcloud CLI installation process on each of your different workstations.
C. Leverage the browser-accessible, pre-configured Cloud Shell environment provided within the Google Cloud Console for all your tasks.
D. Utilize a system package manager, such as apt or yum, to streamline the gcloud CLI installation process on each of your different workstations.

Introduction to Google Cloud

As a cloud engineer frequently moving between different computers and locations, you require consistent, daily access to your Google Cloud project's resources, specifically BigQuery, Bigtable, and Kubernetes Engine. Your priority is to have a ready-to-use command-line environment with the gcloud tool without the overhead of installing, configuring, and maintaining it on every machine you use. Which of the following provides the most efficient and hassle-free solution?

A. Install the gcloud CLI on each workstation you use and create a script to automate the update process across all machines.
B. Utilize a system package manager, such as apt or yum, to streamline the gcloud CLI installation process on each of your different workstations.
C. Leverage the browser-accessible, pre-configured Cloud Shell environment provided within the Google Cloud Console for all your tasks.
D. Utilize a system package manager, such as apt or yum, to streamline the gcloud CLI installation process on each of your different workstations.

Introduction to Google Cloud

Your organization has several departments, each with development teams requiring similar access permissions within their department. Each department also maintains separate dev, test, and prod environments to isolate workloads. You need to design a Google Cloud resource hierarchy that ensures clear separation of environments, consistent permission management, and ease of administration across departments. Which approach best meets these requirements?

A. Create a single project for each department and manage dev, test, and prod environments by using different VPC networks or subnets within that project to isolate workloads.

B. Use folders to represent each department, and create separate projects under each folder for dev, test, and prod environments. Assign permissions at the folder and project levels for consistent and isolated access control.

C. Have all environments (dev, test, prod) for all departments inside a single project, using labels and resource tagging to distinguish between them and control permissions accordingly.

D. Design separate VPCs for each environment (dev, test, prod) across all departments, while keeping all projects centralized under one folder representing the entire organization.

Introduction to Google Cloud

Your organization has several departments, each with development teams requiring similar access permissions within their department. Each department also maintains separate dev, test, and prod environments to isolate workloads. You need to design a Google Cloud resource hierarchy that ensures clear separation of environments, consistent permission management, and ease of administration across departments. Which approach best meets these requirements?

A. Create a single project for each department and manage dev, test, and prod environments by using different VPC networks or subnets within that project to isolate workloads.

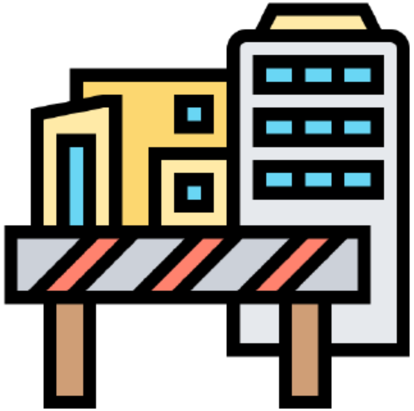
B. Use folders to represent each department, and create separate projects under each folder for dev, test, and prod environments. Assign permissions at the folder and project levels for consistent and isolated access control.

C. Have all environments (dev, test, prod) for all departments inside a single project, using labels and resource tagging to distinguish between them and control permissions accordingly.

D. Design separate VPCs for each environment (dev, test, prod) across all departments, while keeping all projects centralized under one folder representing the entire organization.

Google Compute Engine (GCE)

Zones and Regions



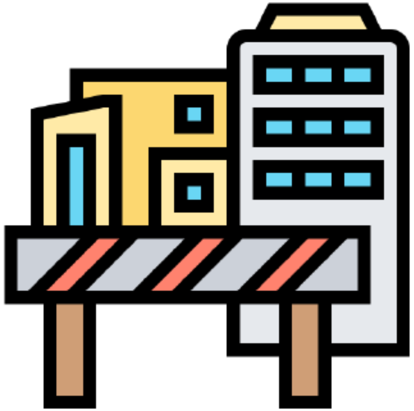
Zone

Availability zone (similar to a datacenter)

Region

Set of zones with high-speed network links

Zones and Regions



Zone

“asia-south1-a”

Region

“asia-south1”

Networks are Global Resources

Network

User-controlled IP
addresses, subnets
and firewalls

Networks are Global Resources

Network

default

Global, Regional, and Zonal Compute Resources

Global

Regional

Zonal

Global, Regional, and Zonal Compute Resources

Global

Regional

Zonal

Global resources accessible from any location. Manage infrastructure that has a global scope e.g. networks, global static IPs, images and snapshots

Global, Regional, and Zonal Compute Resources

Global

Regional

Zonal

Span multiple zones in one region. High availability, can survive zonal failures e.g. regional persistent disks, subnets, regional static IPs

Regional persistent disks allow you to have high availability for your VMs while guarding against zonal failure

Global, Regional, and Zonal Compute Resources

Global

Regional

Zonal

Resources tied to a single zone e.g VM instances, persistent disks, zonal MIGs

Configuration Choices

Machine Family

General purpose, compute optimized, memory optimized, accelerator-optimized

Machine Series

Machines have generation numbers where higher generations have newer features

Machine Type

vCPUs count, memory capacity, and storage capacity

Base Image

Public (free or premium), custom, snapshots from boot disks

Machine Type

Predefined

Custom

Machine Type

Predefined

Custom

Fixed set of types with fixed ratios of memory to vCPU count

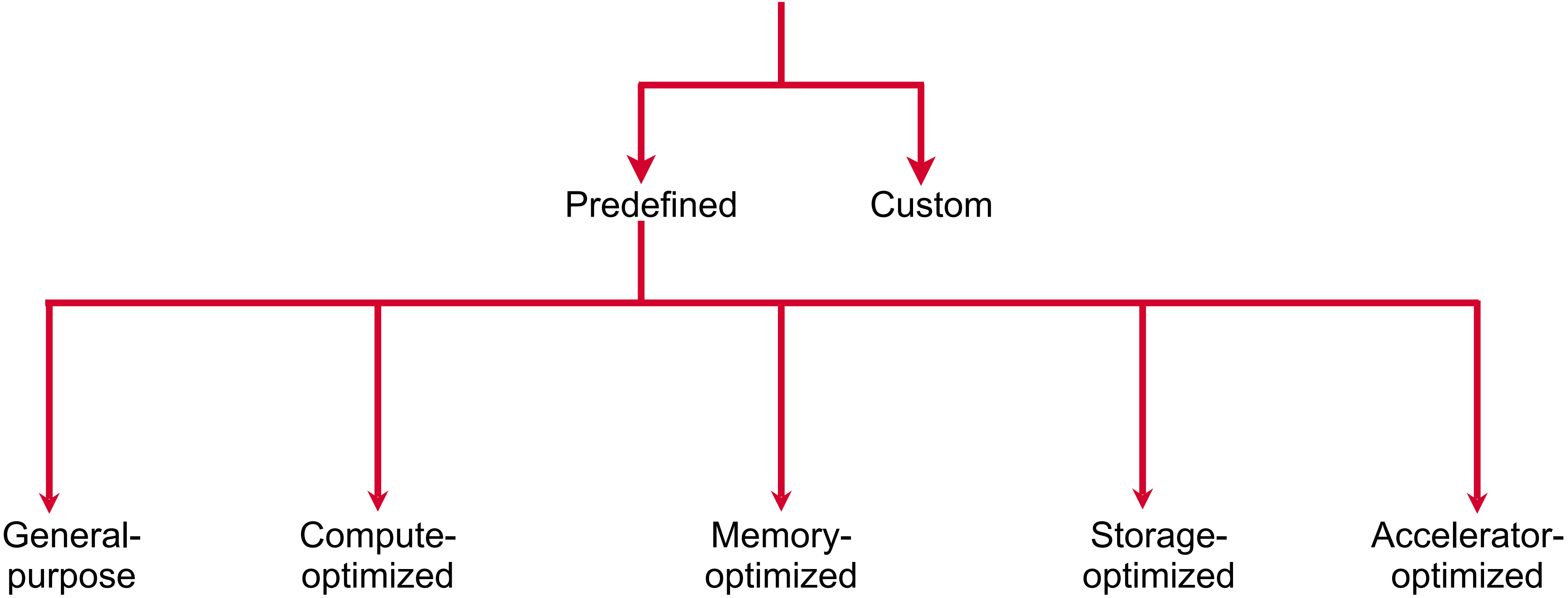
Machine Type

Predefined

Custom

Can independently
specify vCPU count
and amount of memory

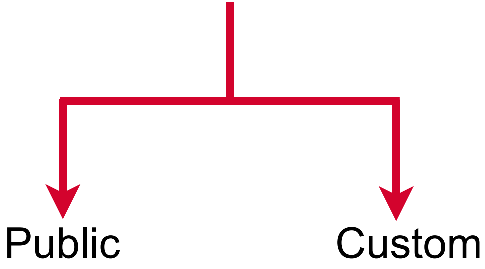
Machine Type



Shared-core Machines

- Cost-effective for running non-resource intensive operations
- A single vCPU run for a time period on single hardware
- Offer micro-bursting capabilities for spikes
- Instance will use additional physical CPUs during spikes

Base Images



Base Images

Public

Custom

Provided and maintained by Google, open-source communities, and third-party vendors

All projects have access to these images and can use them to create instances

Linux, Windows, Container-optimized OS, SQL Server

Base Images

Public

Custom

Available only to your project

First, create a custom image from boot disks and other images; then, use the custom image to create an instance

Spot Instances

An instance that you can create and run at a much lower price than normal instances. However, **GCE might terminate (preempt)** these instances if it requires access to those resources for other tasks.

May not always be available. Not covered by SLAs

- Batch processing and data analysis
- CI/CD pipelines

Preemptible Instances

Similar to Spot VMs (older product and will have fewer features than Spot VMs)

Will definitely be preempted every 24 hours

May not always be available. Not covered by SLAs

- Batch processing and data analysis
- CI/CD pipelines

Spot VMs and Preemptible VMs are used to reduce infrastructure costs in fault-tolerant workloads

Sole-tenant Nodes

A sole-tenant node is a physical Compute Engine server that is **dedicated to hosting VM instances** only for your specific **project**

Keeps your instances physically separated from instances in other projects.
Group instances on the same hardware

- Compliance requirements
- Performance-sensitive applications
- Data isolation

Sole-tenant Nodes - Affinity Labels

Affinity

Anti-affinity

Sole-tenant Nodes - Affinity Labels

Affinity

Anti-affinity

Affinity labels are used to keep VMs (and hence applications) together on the same node or node group

Sole-tenant Nodes - Affinity Labels

Affinity

Anti-affinity

Anti-affinity labels (repulsion) ensure that certain VMs are schedule apart from each other on sole tenant nodes or node groups

Bring-Your-Own-License (BYOL) in Windows Server machines requires the use of a sole tenant node

Shielded VMs

Shielded VMs provide enhanced security features to protect virtual machines from rootkits, bootkits, and other advanced persistent threats.

Ensures VMs firmware and boot loader not tampered with.

- Secure boot
- Virtual Trusted Platform Module (vTPM)

Confidential VMs

Confidential VMs are designed to provide advanced security and privacy for your workloads by encrypting data in use. Ensures that data processed within the VM is encrypted

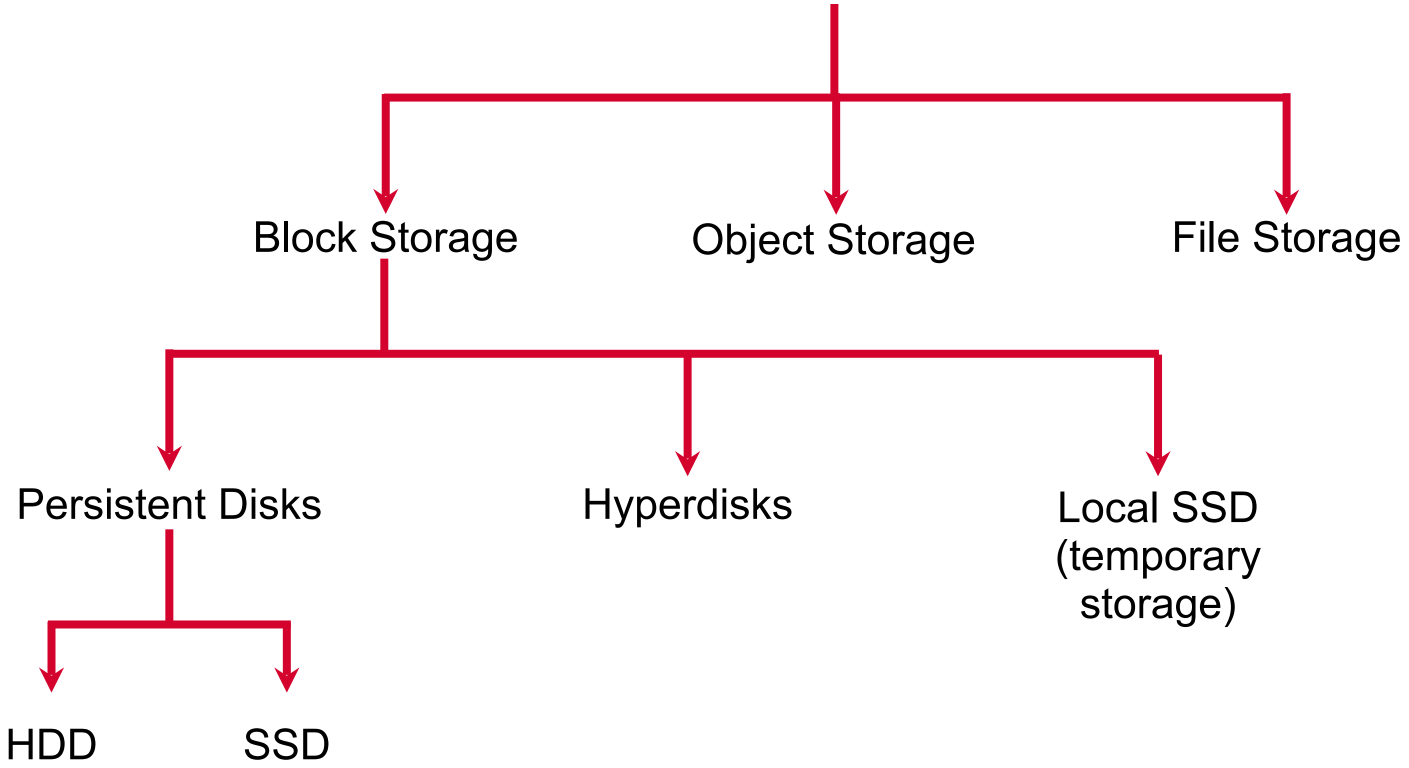
Accessing Storage from VMs

Block Storage

Object Storage

File Storage

Accessing Storage from VMs



Accessing Storage from VMs

Block Storage

Object Storage

GCS Buckets

Pay-what-you-use
blob storage

File Storage

Cloud Filestore

File-system interface
with a managed
shared file storage
service

Migrate to Virtual Machines

- Lift-and-shift tool to migrate from different virtual environments to Google Cloud
- On-prem (VMware vSphere), AWS, Azure
- Minimal downtime with streaming migration
- Disk data is streamed to Google VMs and can use these VMs in a few minutes (streaming continues in the background)
- No software agents to be installed on source VMs
- Can group VMs into “waves” for phased migration

Snapshots and Images

Image

- Binary file used to instantiate VM root disk
- Usually based off OS image
- Also contains boot loader
- Can also contain customizations
- Managed by GCP image service

Snapshot

- Binary file with exact contents of persistent disk
- “Point-in-time” snapshot
- Managed by GCP snapshot service
- Incremental backups possible too
- Used to back up data from persistent disks

Compute Instances

Your company runs batch processing workloads on Google Cloud Platform, some of which are not time-sensitive. Additionally, your organization must comply with strict financial data regulations requiring the use of certified GCP services. You also want to manage infrastructure costs effectively while ensuring compliance. How should you design your solution following Google best practices?

A. Use preemptible VMs to minimize costs and immediately stop using all GCP services that do not meet the financial compliance requirements.

B. Deploy non-preemptible standard VMs in appropriate regions to ensure workload stability and compliance, while carefully selecting only certified GCP services for processing sensitive data.

C. Consolidate all workloads into a single region using preemptible VMs and apply labels to track compliance, without limiting service usage based on certification status.

D. Use standard VMs with auto-scaling in multiple regions to reduce costs, but continue using all GCP services regardless of compliance certification to maintain flexibility.

Compute Instances

Your company runs batch processing workloads on Google Cloud Platform, some of which are not time-sensitive. Additionally, your organization must comply with strict financial data regulations requiring the use of certified GCP services. You also want to manage infrastructure costs effectively while ensuring compliance. How should you design your solution following Google best practices?

A. Use preemptible VMs to minimize costs and immediately stop using all GCP services that do not meet the financial compliance requirements.

B. Deploy non-preemptible standard VMs in appropriate regions to ensure workload stability and compliance, while carefully selecting only certified GCP services for processing sensitive data.

C. Consolidate all workloads into a single region using preemptible VMs and apply labels to track compliance, without limiting service usage based on certification status.

D. Use standard VMs with auto-scaling in multiple regions to reduce costs, but continue using all GCP services regardless of compliance certification to maintain flexibility.

Compute Instances

Your company plans to migrate several Linux virtual machines running on an on-premises VMware infrastructure to Google Cloud Compute Engine. You want to follow Google-recommended best practices for a reliable and efficient migration. What approach should you take?

A. 1. Inventory the applications and their dependencies on each VM. 2. Export the VMs as disk images, upload to Google Cloud Storage, and create Compute Engine instances using these images.

B. 1. Assess the existing VMware virtual machines and their workloads. 2. Prepare a detailed migration plan, create a migration runbook for Migrate for Compute Engine, and perform the migration using this service.

C. 1. Install third-party migration software agents on all Linux VMs. 2. Use these agents to replicate and migrate VMs manually to Compute Engine 3. Use Migrate for Compute Engine for verification.

D. 1. Perform a quick assessment and shut down all VMs at once. 2. Create snapshots and manually import them as boot disks using Migrate for Compute Engine

Compute Instances

Your company plans to migrate several Linux virtual machines running on an on-premises VMware infrastructure to Google Cloud Compute Engine. You want to follow Google-recommended best practices for a reliable and efficient migration. What approach should you take?

A. 1. Inventory the applications and their dependencies on each VM. 2. Export the VMs as disk images, upload to Google Cloud Storage, and create Compute Engine instances using these images.

B. 1. Assess the existing VMware virtual machines and their workloads. 2. Prepare a detailed migration plan, create a migration runbook for Migrate for Compute Engine, and perform the migration using this service.

C. 1. Install third-party migration software agents on all Linux VMs. 2. Use these agents to replicate and migrate VMs manually to Compute Engine 3. Use Migrate for Compute Engine for verification.

D. 1. Perform a quick assessment and shut down all VMs at once. 2. Create snapshots and manually import them as boot disks using Migrate for Compute Engine

Compute Instances

A media company runs a video processing application on Google Compute Engine. The application stores its working data on attached disks and must resume operation quickly in another zone if the current zone experiences an outage. The company wants to minimize downtime and avoid data loss during such events while keeping the recovery process automated and efficient. How should they design this setup?

A. Schedule frequent backups of the VM’s boot and data disks to Cloud Storage, and after a zone failure, restore the backups manually and recreate the VM in the same zone.

B. Deploy the application using an instance template and attach a regional persistent disk to the VM. If the current zone goes down, automatically launch a new instance in another zone in the same region with the attached regional persistent disk.

C. Use zonal persistent disks for the application’s data and replicate data asynchronously to a secondary zone. In case of failure, manually detach and attach the disk to a new VM in the other zone.

D. Set up the application on multiple VMs across zones using local SSDs and rely on application-level replication for data recovery in case of a zone outage.

Compute Instances

A media company runs a video processing application on Google Compute Engine. The application stores its working data on attached disks and must resume operation quickly in another zone if the current zone experiences an outage. The company wants to minimize downtime and avoid data loss during such events while keeping the recovery process automated and efficient. How should they design this setup?

A. Schedule frequent backups of the VM's boot and data disks to Cloud Storage, and after a zone failure, restore the backups manually and recreate the VM in the same zone.

B. Deploy the application using an instance template and attach a regional persistent disk to the VM. If the current zone goes down, automatically launch a new instance in another zone in the same region with the attached regional persistent disk.

C. Use zonal persistent disks for the application's data and replicate data asynchronously to a secondary zone. In case of failure, manually detach and attach the disk to a new VM in the other zone.

D. Set up the application on multiple VMs across zones using local SSDs and rely on application-level replication for data recovery in case of a zone outage.

Compute Instances

Your company is migrating a legacy financial application to Google Cloud. This application has a strict, per-physical-server licensing model. To maintain compliance, the Virtual Machines running this application must be confined to a specific, pre-paid set of physical hosts.

You have created a single sole-tenant node group for the finance department. Within this group, you have provisioned two specific nodes (`licensed-host-01` and `licensed-host-02`) that are designated solely for this licensed application, while other nodes in the same group are for general use.

When you create a new VM instance for this licensed application, how do you ensure it is scheduled *only* on one of the designated licensed hosts and not on any other nodes in the group?

A. When creating the VM, apply a network tag that matches the name of the sole-tenant node group.

B. When creating the VM, configure a custom metadata key named `gce-schedule-on-host` with the value `licensed-host-01`.

C. When creating the VM, define a node affinity label that requires the VM to be scheduled on any node within the finance department's node group.

D. When creating the VM, define a node affinity label that specifically targets the node names `licensed-host-01` or `licensed-host-02`.

Compute Instances

Your company is migrating a legacy financial application to Google Cloud. This application has a strict, per-physical-server licensing model. To maintain compliance, the Virtual Machines running this application must be confined to a specific, pre-paid set of physical hosts.

You have created a single sole-tenant node group for the finance department. Within this group, you have provisioned two specific nodes (`licensed-host-01` and `licensed-host-02`) that are designated solely for this licensed application, while other nodes in the same group are for general use.

When you create a new VM instance for this licensed application, how do you ensure it is scheduled *only* on one of the designated licensed hosts and not on any other nodes in the group?

A. When creating the VM, apply a network tag that matches the name of the sole-tenant node group.

B. When creating the VM, configure a custom metadata key named `gce-schedule-on-host` with the value `licensed-host-01` or `licensed-host-02`.

C. When creating the VM, define a node affinity label that requires the VM to be scheduled on any node within the finance department's node group.

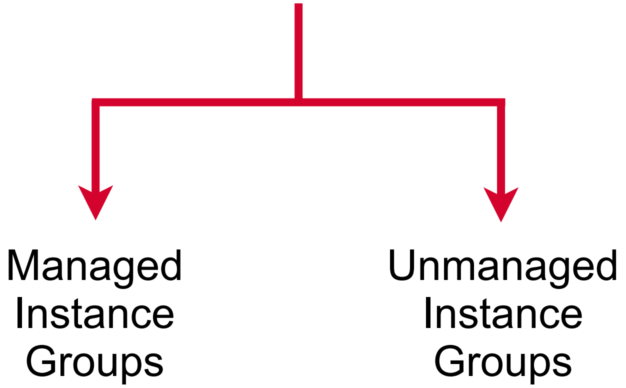
D. When creating the VM, define a node affinity label that specifically targets the node names `licensed-host-01` or `licensed-host-02`.

Instance Groups

Instance Groups

A collection of virtual machines that you can manage as a single entity

Instance Groups



Instance Groups

Managed Instance Groups

Unmanaged Instance Groups

Managed Instance Group

Group of identical GCE VM instances, created from the same instance template that are managed by the platform

Managed Instance Group

Group of identical GCE VM instances, created from the same instance template that are managed by the platform

Instances have the exact same configuration

Managed Instance Group

Group of identical GCE VM instances, created from the same instance template that are managed by the platform

The configuration is specified in an instance template

Instance Template

A specification of machine type, boot disk (or image), zone, labels and other instance properties that can be used to instantiate either individual VM instances or a Managed Instance Group

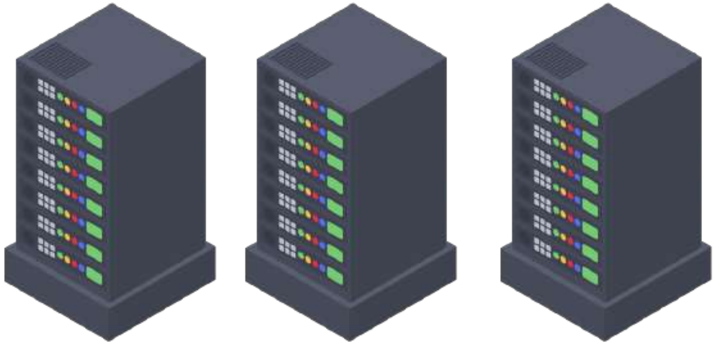
Instance Template

A specification of machine type, boot disk (or image), zone, labels and other instance properties that can be used to instantiate either individual VM instances or a Managed Instance Group

Instance templates can reference images (public or custom) to use in instantiated VMs

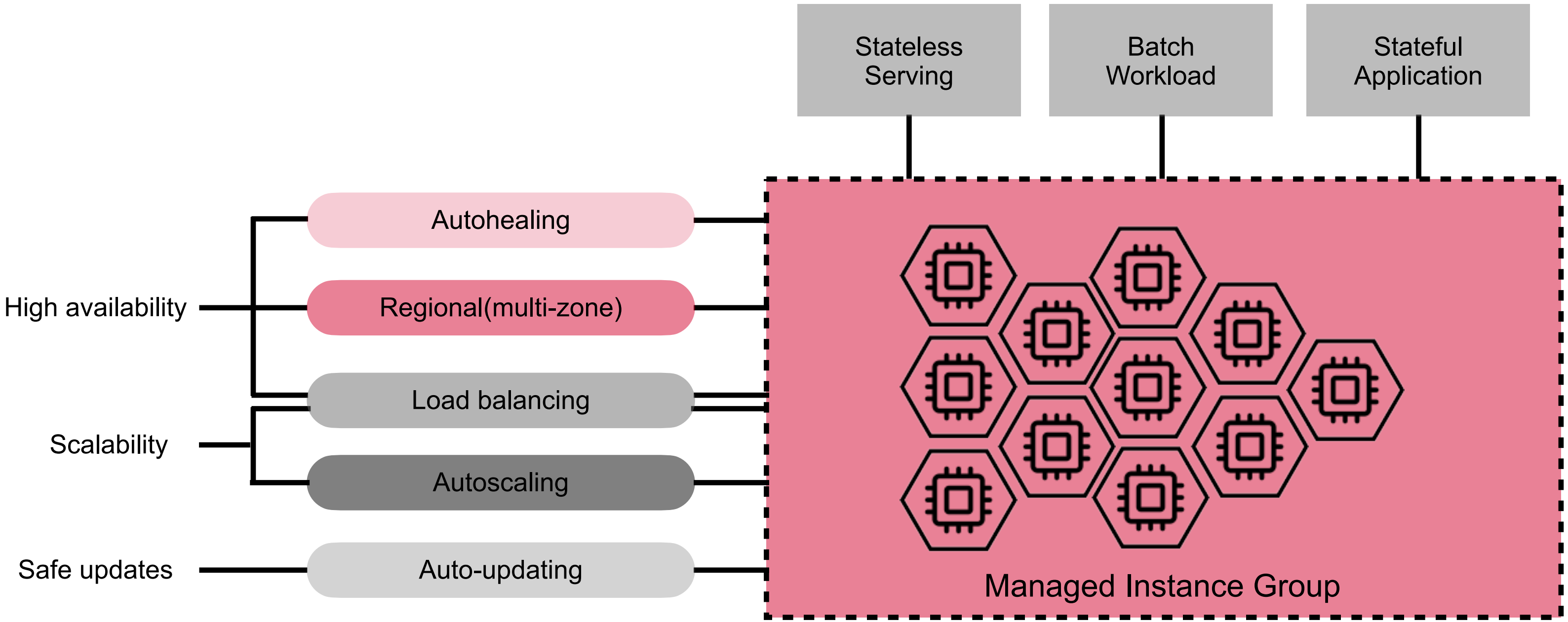
Instance Template to Create Instances

Instance Template

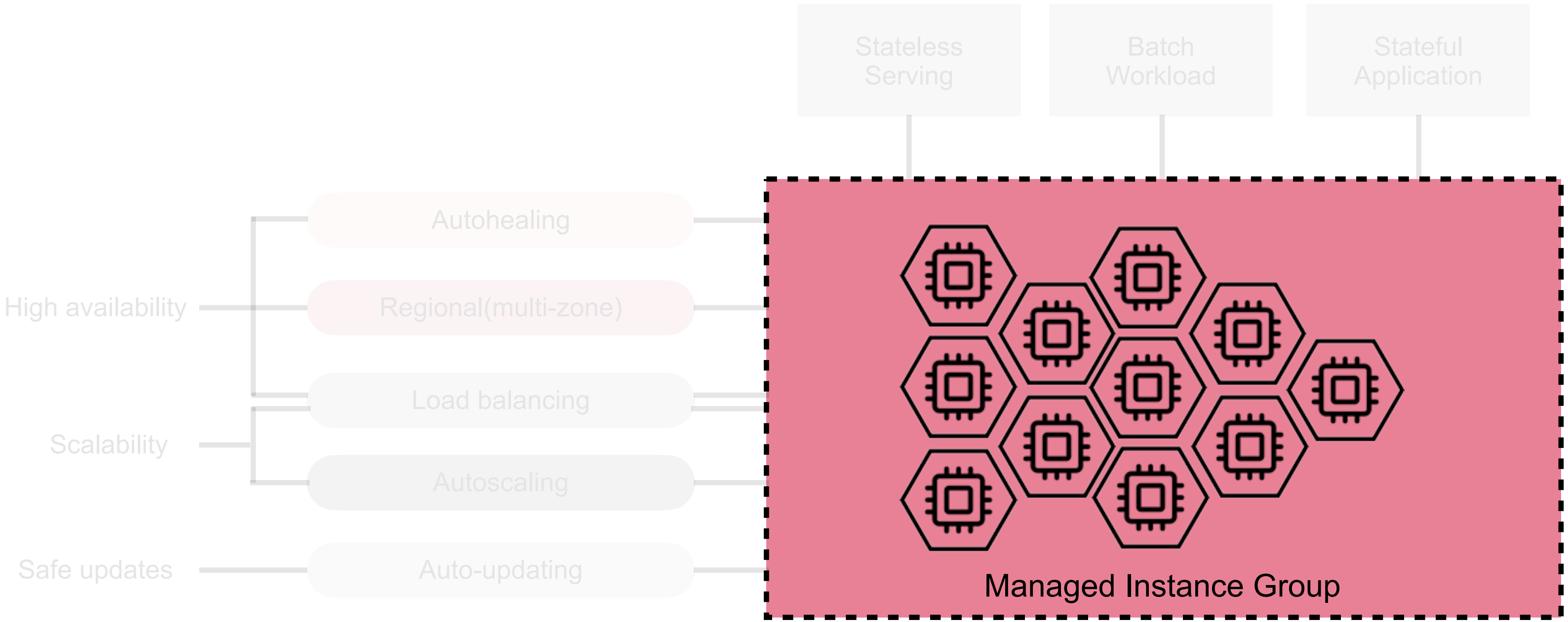


Managed Instance
Group

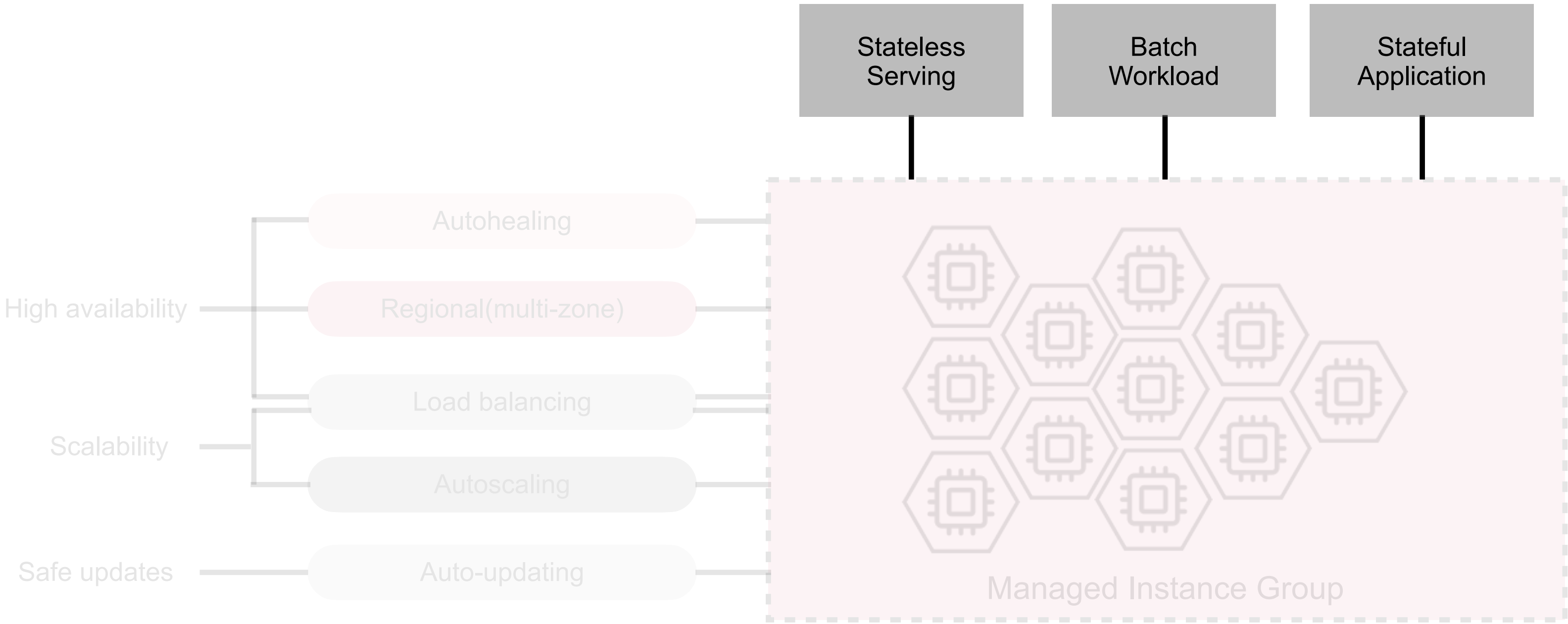
Managed Instance Groups



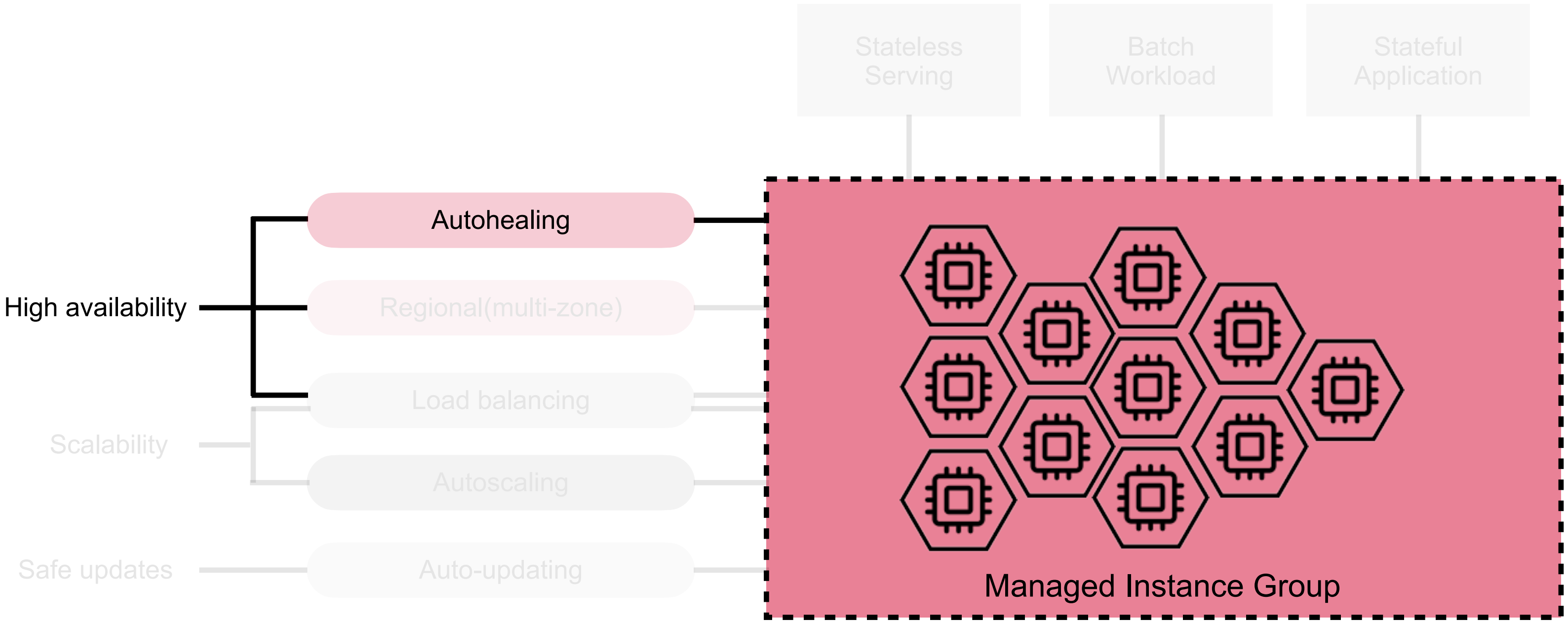
Multiple Instances Created from the Same Template



Managed Instance Groups



Autohealing



Autohealing

Automatically repair failed VMs

Application-based auto healing

Autohealing

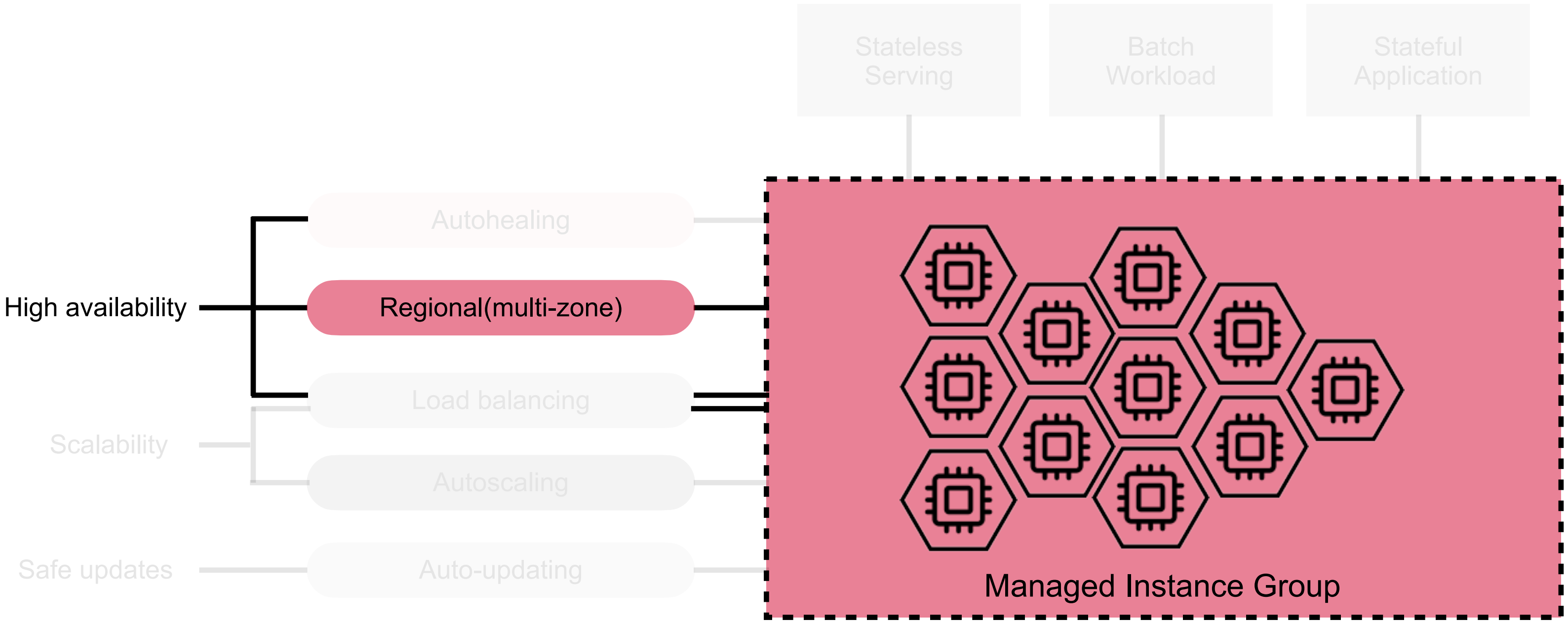
Automatically repair failed VMs

Application-based auto healing

If VM stops, crashes, or is pre-empted (spot VMs) it is automatically recreated based on the template

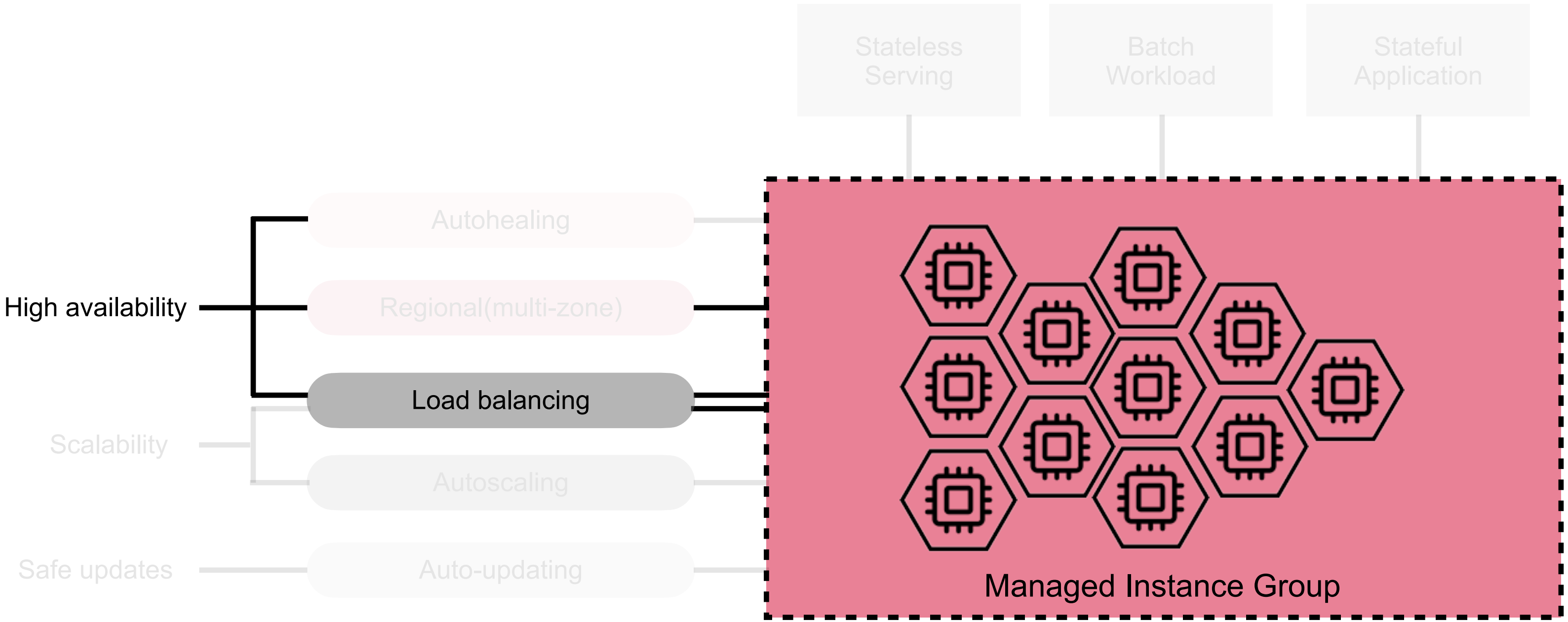
Application-based health checks to check whether the application is responding as expected. If not the VM is recreated

Regional (Multi-zone) Clusters



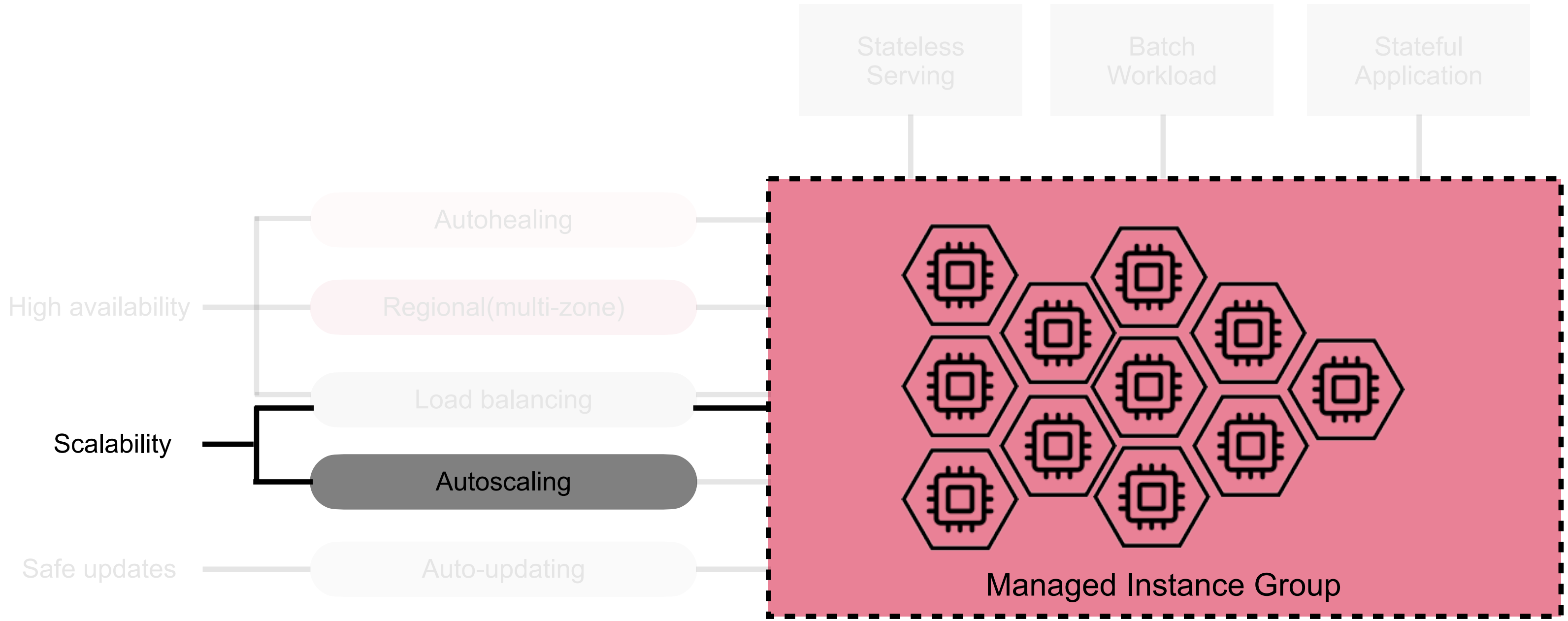
This replication protects against zonal failures. If that happens, your app can continue serving traffic from instances running in the remaining available zones in the same region.

Load Balancing



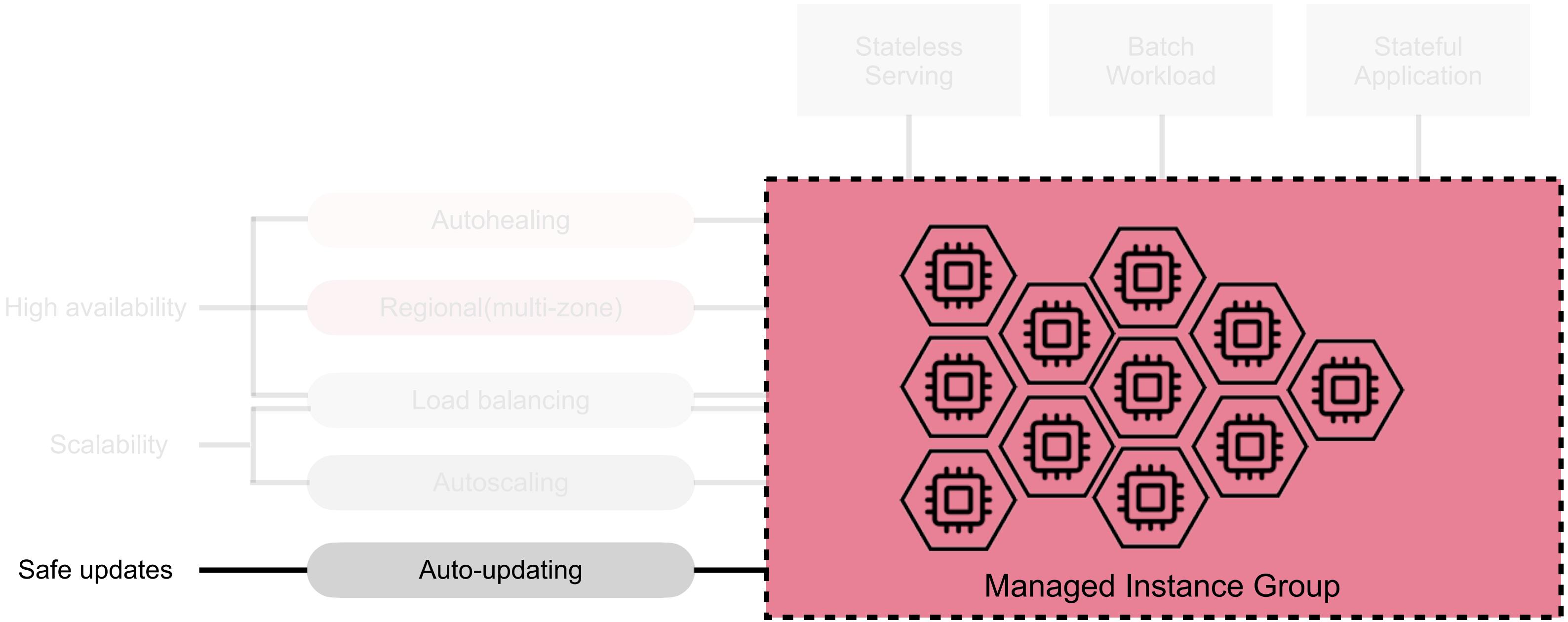
Can work with load balancing services to distribute traffic across the instances of the group

Scalability



The number of instances can grow automatically to meet demand and will shrink automatically when demand drops

Updates



Can deploy new versions of software to instances in your MIG - supports rolling updates and canary updates (partial rollouts)

Update Mode for Managed Instance Groups

Proactive

Opportunistic

Proactive for Required or Critical Updates

Proactive

Opportunistic

The MIG systematically replaces old instances with new ones created from the updated template. It follows user-defined rules, such as `maxUnavailable` and `maxSurge`, to perform a controlled rolling update.

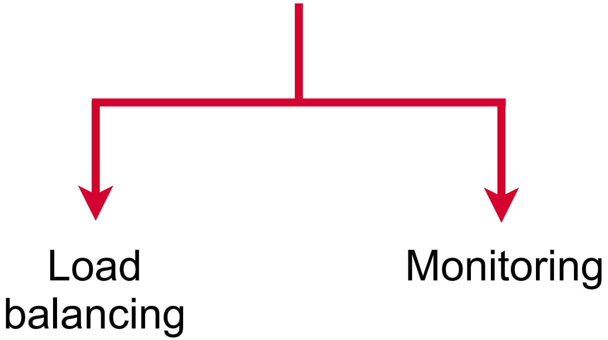
Opportunistic for Non-critical Updates

Proactive

Opportunistic

The MIG waits for an "opportunity" to apply the update. An update is only applied to an instance when that instance is naturally recreated for another reason (manually creating a new VM or autoscaling)

Health Checks



Health Checks

Load balancing

Monitoring

Direct traffic from non-responsive instances towards healthy instances

Signal to the MIG to proactively delete and recreate instances that become unhealthy

Use separate health checks for load balancing and monitoring - do not rely on the same health check

Instance Groups

Managed Instance Groups

Unmanaged Instance Groups

Unmanaged instance groups can contain heterogeneous instances that you can arbitrarily add and remove from the group.

Do not offer autoscaling, auto healing - can be used with a load balancer

If you want load balancing WITHOUT auto-scaling use unmanaged instance groups

Managed Instance Groups

Your team has developed a minor enhancement for an application running on a managed instance group in Google Compute Engine. You've created a new instance template that includes this enhancement. Since the change is non-critical, you want to ensure no currently running instances are interrupted or restarted. Instead, you want only future instances created by the managed instance group to include the update. What should you do?

A. Initiate a rolling restart to apply the update to all existing instances in sequence.

B. Trigger a rolling replacement, which will systematically delete and recreate instances with the new template.

C. Perform a rolling update using an opportunistic strategy

D. Perform a rolling update using a proactive strategy

Managed Instance Groups

Your team has developed a minor enhancement for an application running on a managed instance group in Google Compute Engine. You've created a new instance template that includes this enhancement. Since the change is non-critical, you want to ensure no currently running instances are interrupted or restarted. Instead, you want only future instances created by the managed instance group to include the update. What should you do?

A. Initiate a rolling restart to apply the update to all existing instances in sequence.

B. Trigger a rolling replacement, which will systematically delete and recreate instances with the new template.

C. Perform a rolling update using an opportunistic strategy

D. Perform a rolling update using a proactive strategy

Managed Instance Groups

Your team manages a Compute Engine managed instance group hosting a critical web application. During high traffic periods, users report slow response times and occasional timeouts. Investigation shows that the application process on each instance has reached the max CPU limit and is causing delays, but all other system processes (e.g., OS services, monitoring agents) have normal CPU and memory usage. Autoscaling has already hit its maximum number of instances configured. Downstream systems like the database are operating normally. It's important that user delays be resolved right away. What would you do?

A. Modify the autoscaling metric to trigger scaling based on memory consumption instead of CPU.

B. Disable health checks temporarily to reduce load on the instances during peak traffic.

C. Increase the CPU quota in the project to allow larger VM types to be used in the managed instance group.

D. Increase the maximum instance limit in the autoscaling policy so more instances can be provisioned under load.

Managed Instance Groups

Your team manages a Compute Engine managed instance group hosting a critical web application. During high traffic periods, users report slow response times and occasional timeouts. Investigation shows that the application process on each instance has reached the max CPU limit and is causing delays, but all other system processes (e.g., OS services, monitoring agents) have normal CPU and memory usage. Autoscaling has already hit its maximum number of instances configured. Downstream systems like the database are operating normally. It’s important that user delays be resolved right away. What would you do?

A. Modify the autoscaling metric to trigger scaling based on memory consumption instead of CPU.

B. Disable health checks temporarily to reduce load on the instances during peak traffic.

C. Increase the CPU quota in the project to allow larger VM types to be used in the managed instance group.

D. Increase the maximum instance limit in the autoscaling policy so more instances can be provisioned under load.

Managed Instance Groups

Your team maintains a web service running on a single Compute Engine virtual machine. The workload is highly variable, with heavy usage during business hours and low activity otherwise. To improve responsiveness and handle traffic spikes automatically, you want to implement a scalable solution that ensures new instances are launched consistently with the current application state. What is the best approach to achieve this?

A. Create a custom image from the existing VM's disk, create an instance template using that image, and deploy a managed instance group with autoscaling enabled based on this template.

B. Create an instance template directly from the running VM without creating a custom image, then manually scale the managed instance group during peak hours.

C. Use an instance template that references a default public image instead of the current VM's image, then enable autoscaling on the managed instance group.

D. Take a snapshot of the VM's disk and use it directly to launch additional instances as needed without using instance templates.

Managed Instance Groups

Your team maintains a web service running on a single Compute Engine virtual machine. The workload is highly variable, with heavy usage during business hours and low activity otherwise. To improve responsiveness and handle traffic spikes automatically, you want to implement a scalable solution that ensures new instances are launched consistently with the current application state. What is the best approach to achieve this?

A. Create a custom image from the existing VM's disk, create an instance template using that image, and deploy a managed instance group with autoscaling enabled based on this template.

B. Create an instance template directly from the running VM without creating a custom image, then manually scale the managed instance group during peak hours.

C. Use an instance template that references a default public image instead of the current VM's image, then enable autoscaling on the managed instance group.

D. Take a snapshot of the VM's disk and use it directly to launch additional instances as needed without using instance templates.

Google App Engine

Google Cloud Compute Choices

IaaS

PaaS

Google Compute Engine

Google Kubernetes Engine

Google App Engine

Google Cloud Run

Google Cloud Functions

Google App Engine

Web framework and platform for hosting web applications on the Google Cloud

Support for Go, PHP, Java, Python, Node.js, .NET, Ruby and other languages

Google App Engine

Web framework and platform for hosting web applications on the Google Cloud

Support for Go, PHP, Java, Python, Node.js, .NET, Ruby and other languages

Focus on development and code

Infrastructure and scaling taken care of by the platform

App Engine Environments

Standard Environment	Flexible Environment

App Engine Environments

Standard

- App runs in a proprietary sandbox
- Instances start up in seconds
- Code in few languages/versions only
- No other runtimes possible
- Apps cannot access Compute Engine resources
- Can install 3rd party binaries only for selected runtimes
- Scales to zero if no traffic (potentially free for very low traffic)

App Engine Environments

Standard

- App runs in a proprietary sandbox
- Instances start up in seconds
- Code in few languages/versions only
- No other runtimes possible
- Apps cannot access Compute Engine resources
- Can install 3rd party binaries only for selected runtimes
- Scales to zero if no traffic (potentially free for very low traffic)

Flexible

- Runs in Docker container on GCE VM
- Instance start up in minutes
- Code in far more languages/versions
- Custom runtimes possible
- Apps can access Compute Engine resources, some OS packages
- Can install and access third-party binaries
- Does not scale to zero - minimum of one instance running incurs baseline cost

App Engine Environments

Standard

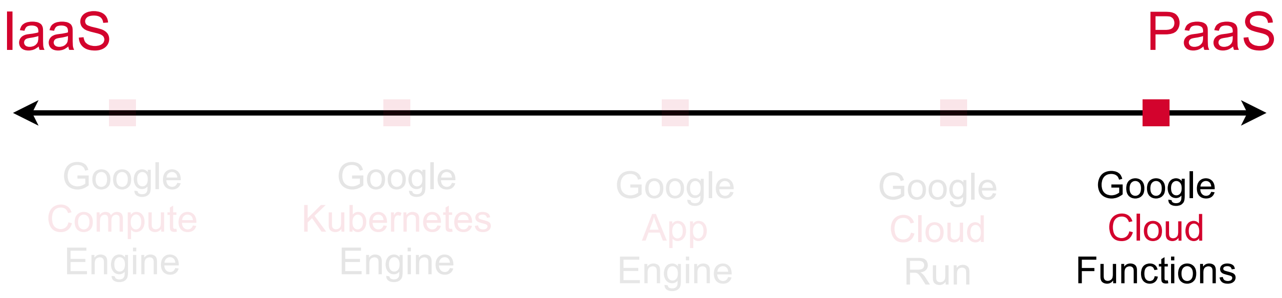
- Apps that experience traffic spikes
- Usually stateless HTTP web apps

Flexible

- Apps that experience consistent traffic
- General purpose apps

Google Cloud Functions

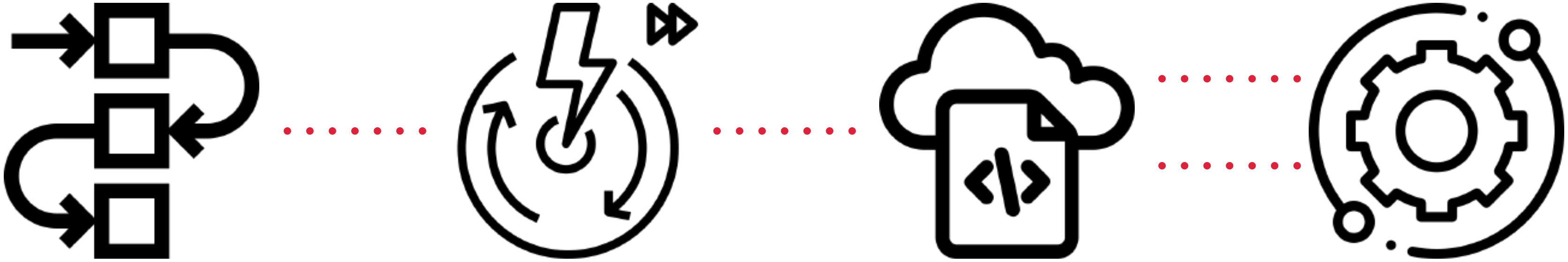
Google Cloud Compute Choices



Cloud Functions

Event-driven serverless compute platform

Event-driven Serverless Compute



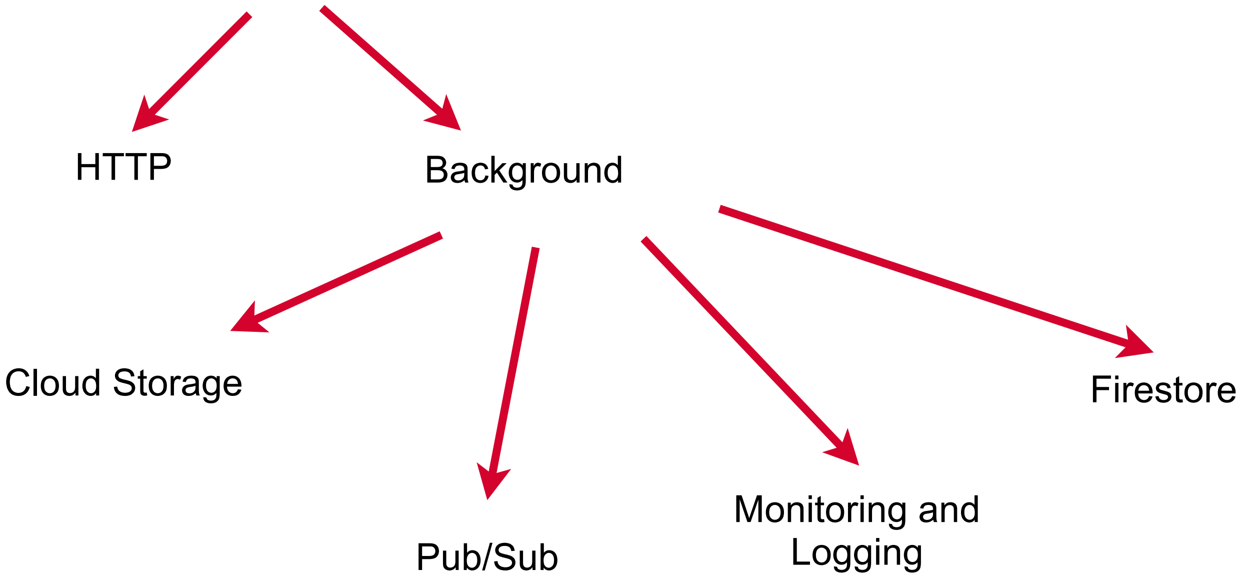
Event occurs

Platform triggers execution

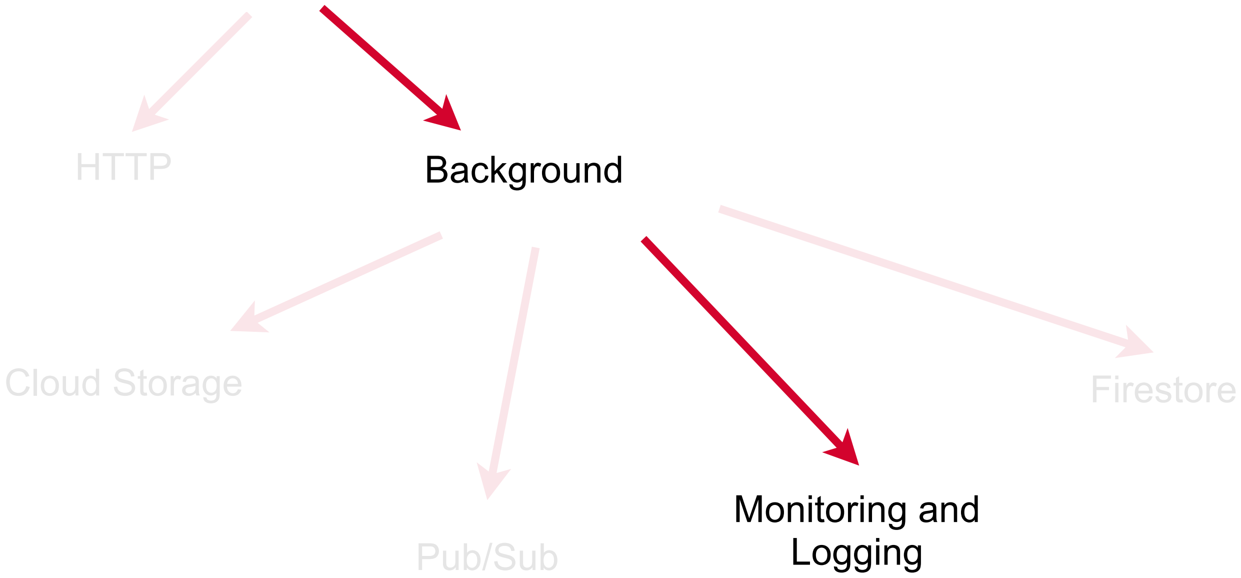
Cloud Function code runs

Invokes other Google Cloud services

Types of Events



Types of Events



Indirect process that makes use of an intermediary such as Pub/Sub

Concurrency and Scale

- Spin up function instances based on current load
- Functions receive event parameters from platform
- Functions do not share memory or variables
- An instance processes a single request (generation 1)
- Function concurrency supported (generation 2)
- Functions should be stateless
- Scales to zero when no request being processed

Google Cloud Run

Google Cloud Compute Choices

IaaS

PaaS

Google Compute Engine

Google Kubernetes Engine

Google App Engine

Google Cloud Run

Google Cloud Functions

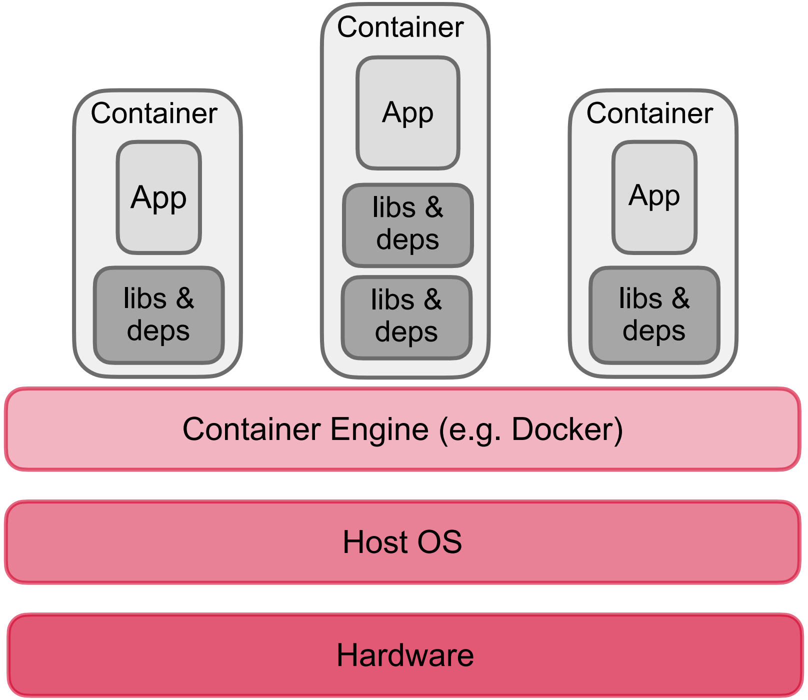
Container

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it; code, runtime, system tools, system libraries, settings

Container

- Contains applications
- And all of the application’s dependencies
- Platform independent
- Runs on layer of abstraction
- Docker Runtime (for Docker containers)

Modern Workloads on Containers



Cloud Run

Serverless, managed platform that lets you run containers directly on top of Google's scalable architecture

Cloud Run

- Write your code in any programming language
- Create a container image (or use source-based deployment option - Google Cloud will build container image for you)
- Register the container with the artifact registry
- Deploy your container directly using Cloud Run
- No cluster creation no infrastructure management

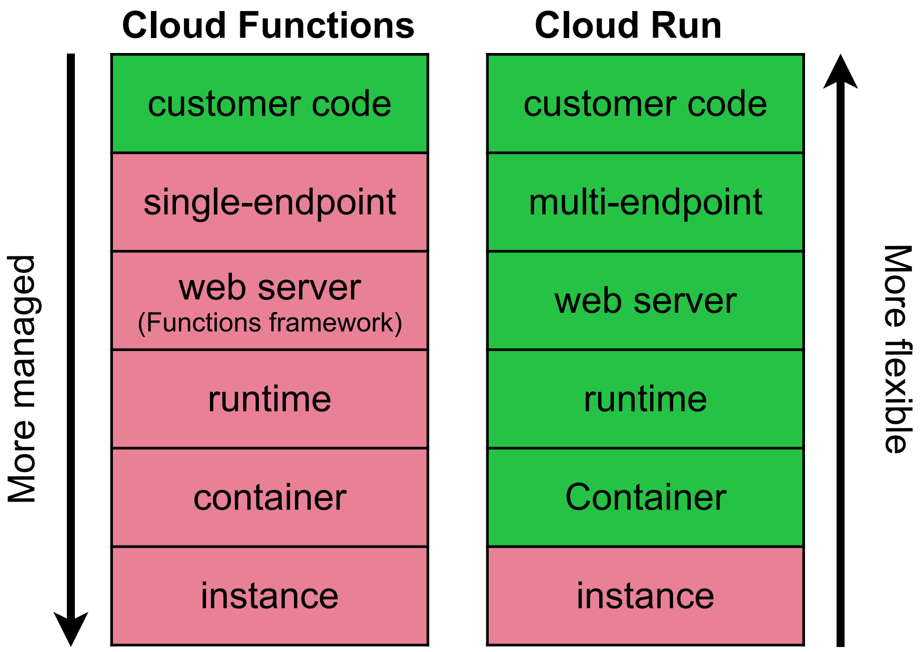
Running Code Using Cloud Run

Cloud Run Services

Cloud Run Jobs

Both use the same environment and have the same integrations with other Google Cloud services

Cloud Functions vs. Cloud Run



How managed do you want to be?

Cloud Functions vs. Cloud Run

Cloud Functions

- Specific limited runtimes supported
- Can be triggered based on platform events
- No support for running jobs
- 2nd generation functions support concurrency

Cloud Run

- All runtimes that can be run using containers
- Expose endpoints and invoked using HTTP requests
- Support for running jobs
- Great support for concurrent requests

Cloud Functions vs. Cloud Run

Cloud Functions

- Choose Cloud Functions if you primarily want to connect to other cloud services on Google Cloud

Cloud Run

- Choose Cloud Run if you want a simple way to scale and maintain services using containers

Servless Compute Options

Your digital marketing agency rapidly develops and tests multiple versions of landing pages for client campaigns. Developers need a way to deploy new page designs quickly. A key requirement is to empower non-technical Account Managers to instantly switch the live landing page to a new version using a simple interface. The solution must not require the agency to manage servers, clusters, or any underlying infrastructure.

Which Google Cloud product is best suited for these requirements?

A.App Engine

B.GKE On-Prem

C.Compute Engine

D.Google Kubernetes Engine

Servless Compute Options

Your digital marketing agency rapidly develops and tests multiple versions of landing pages for client campaigns. Developers need a way to deploy new page designs quickly. A key requirement is to empower non-technical Account Managers to instantly switch the live landing page to a new version using a simple interface. The solution must not require the agency to manage servers, clusters, or any underlying infrastructure.

Which Google Cloud product is best suited for these requirements?

A.App Engine

B.GKE On-Prem

C.Compute Engine

D.Google Kubernetes Engine

Serverless Compute Options

Your team is building a new media processing application. The core function of the application is to accept user-uploaded videos and apply a series of complex transformations using a specific, third-party Linux binary (e.g., a proprietary version of FFmpeg) that is not included in standard cloud platform runtimes.

You need a solution that scales automatically from zero to handle unpredictable workloads, while completely abstracting away the underlying infrastructure so your team can focus on the application logic. The deployment method must allow you to package your custom binaries along with your code.

Which Google Cloud service should you use?

A. App Engine Standard

B. Cloud Run

C. Compute Engine

D. Compute Engine with Managed Instance Groups

E. Google Kubernetes Engine (GKE)

Serverless Compute Options

Your team is building a new media processing application. The core function of the application is to accept user-uploaded videos and apply a series of complex transformations using a specific, third-party Linux binary (e.g., a proprietary version of FFmpeg) that is not included in standard cloud platform runtimes.

You need a solution that scales automatically from zero to handle unpredictable workloads, while completely abstracting away the underlying infrastructure so your team can focus on the application logic. The deployment method must allow you to package your custom binaries along with your code.

Which Google Cloud service should you use?

A. App Engine Standard

B. Cloud Run

C. Compute Engine

D. Compute Engine with Managed Instance Groups

E. Google Kubernetes Engine (GKE)

Serverless Compute Options

Your e-commerce platform runs as a service on Cloud Run. You have developed a new version of the service that includes an experimental recommendation engine. You need to perform a canary release, directing 5% of live production traffic to this new version to monitor its performance and error rates, while the other 95% of users continue to use the stable version.

You want to use the most direct and idiomatic feature of the platform to accomplish this with minimal configuration. What should you do?

A. Deploy the new revision. Use the Cloud Run traffic splitting feature to direct a fixed portion to the new version and the majority to the old version.

B. Create a second Cloud Run service for the experimental version. Place an external HTTPS Load Balancer in front of both the stable and experimental services and configure weighted backend routing.

C. Deploy the new version with a specific label, like version: experimental. Configure a Cloud Monitoring alert that will automatically increase traffic if the error rate is low.

D. Integrate your Cloud Run service with a service mesh like Istio. Define routing rules within the service mesh to split traffic between the stable and experimental versions of your service.

Serverless Compute Options

Your e-commerce platform runs as a service on Cloud Run. You have developed a new version of the service that includes an experimental recommendation engine. You need to perform a canary release, directing 5% of live production traffic to this new version to monitor its performance and error rates, while the other 95% of users continue to use the stable version.

You want to use the most direct and idiomatic feature of the platform to accomplish this with minimal configuration. What should you do?

A. **Deploy the new revision. Use the Cloud Run traffic splitting feature to direct a fixed portion to the new version and the majority to the old version.**

B. Create a second Cloud Run service for the experimental version. Place an external HTTPS Load Balancer in front of both the stable and experimental services and configure weighted backend routing.

C. Deploy the new version with a specific label, like version: experimental. Configure a Cloud Monitoring alert that will automatically increase traffic if the error rate is low.

D. Integrate your Cloud Run service with a service mesh like Istio. Define routing rules within the service mesh to split traffic between the stable and experimental versions of your service.

Cloud Functions and Cloud Run

Your company has an application that uploads images to a Cloud Storage bucket, and you need to process each image as soon as it becomes available. The processing should start automatically when a new image is uploaded, and you want to minimize costs by only paying for the compute resources when the processing happens. Additionally, your team does not want to manage any infrastructure. What should you do?

A. Deploy a managed instance group (MIG) to continuously monitor the bucket and process the images.
B. Set up a Kubernetes cluster and configure autoscaling to process images when they are uploaded.
C. Deploy your image processing code in a Cloud Function triggered by the Cloud Storage bucket.
D. Use a Cloud Run service to handle image processing, scaling based on incoming requests.

Cloud Functions and Cloud Run

Your company has an application that uploads images to a Cloud Storage bucket, and you need to process each image as soon as it becomes available. The processing should start automatically when a new image is uploaded, and you want to minimize costs by only paying for the compute resources when the processing happens. Additionally, your team does not want to manage any infrastructure. What should you do?

A. Deploy a managed instance group (MIG) to continuously monitor the bucket and process the images.

B. Set up a Kubernetes cluster and configure autoscaling to process images when they are uploaded.

C. Deploy your image processing code in a Cloud Function triggered by the Cloud Storage bucket.

D. Use a Cloud Run service to handle image processing, scaling based on incoming requests.

Containers and Kubernetes

Google Cloud Compute Choices

IaaS

PaaS

Google Compute Engine

Google Kubernetes Engine

Google App Engine

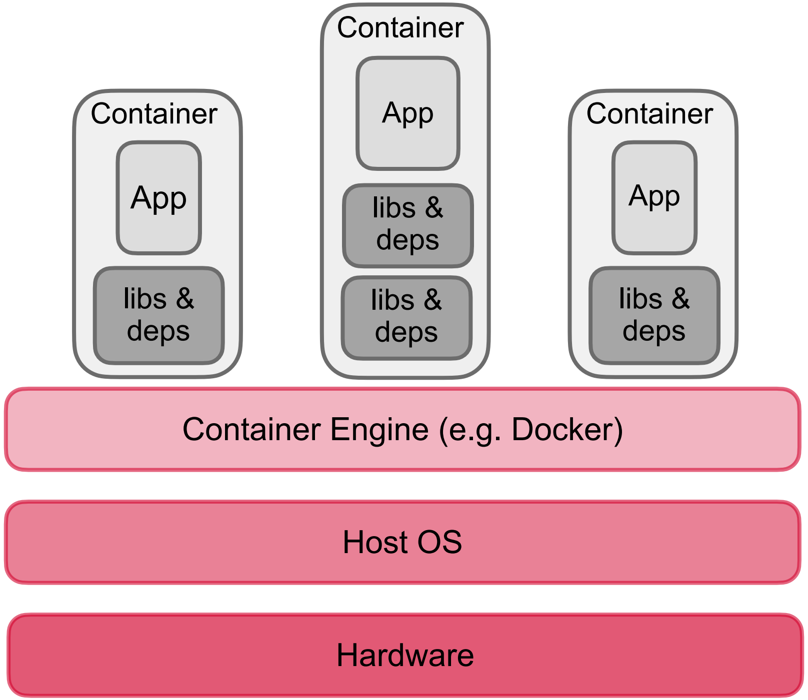
Google Cloud Run

Google Cloud Functions

Container

A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it; code, runtime, system tools, system libraries, settings

Modern Workloads on Containers



Attractions of Containers

- No guest OS
  - Platform independent
  - Considerably smaller than VM images
- Lightweight
  - Small and fast
  - Quick to start
  - Speeds up autoscaling
- Hybrid, multi-cloud
  - Hybrid: Work on-premise and on cloud
  - Multi-cloud: Not tied to any specific cloud platform

Kubernetes

Orchestration technology for containers - convert isolated containers running on different hardware into a cluster

Kubernetes is the middle-ground between IaaS and PaaS in a hybrid, multi-cloud world

IaaS vs. PaaS

Infrastructure-as-a-Service

- Heavy operational burden
- Migration is hard

Platform-as-a-Service

- Provider lock-in
- Migration is very hard

Compute Choices

IAAS

PAAS

Containers

↓

Containers Clusters

↓

Kubernetes

Kubernetes as Orchestrator

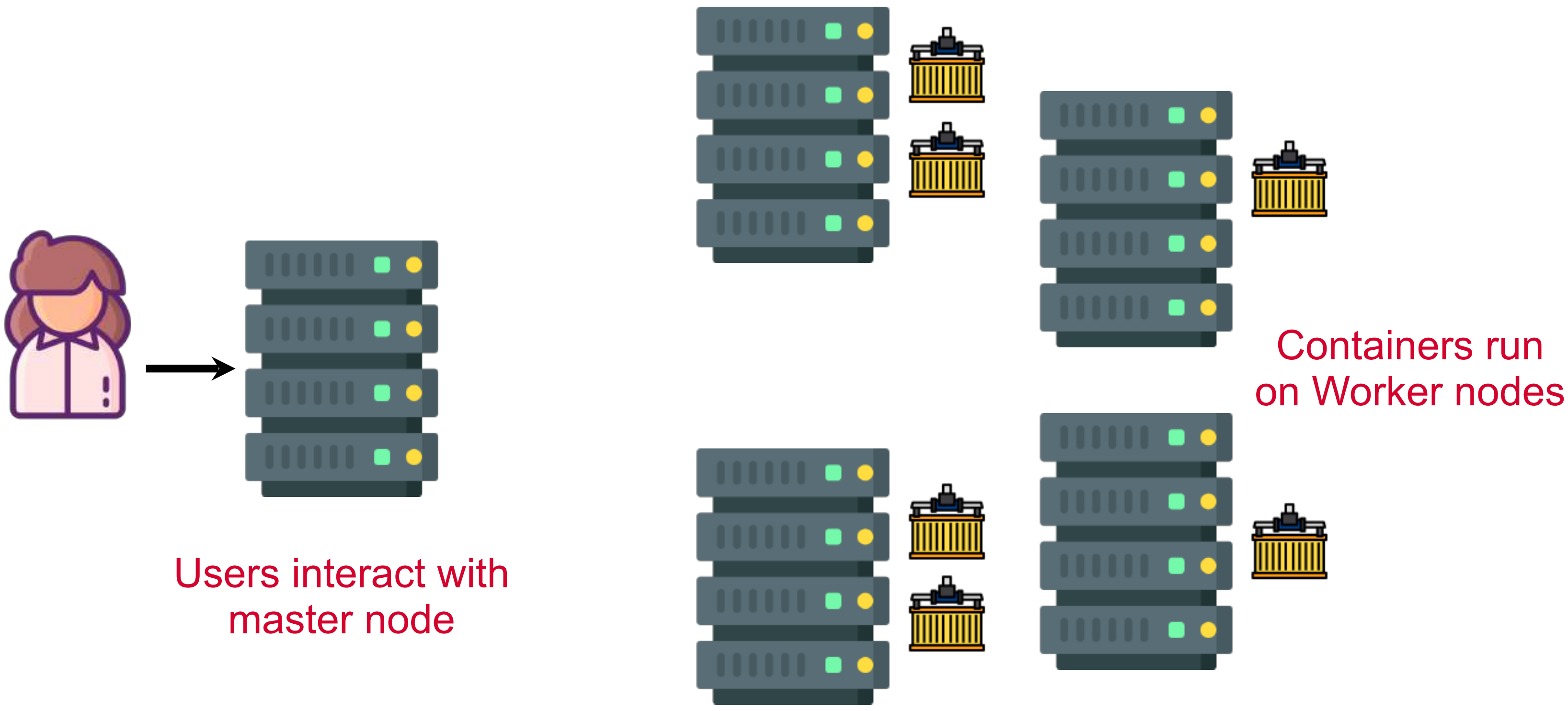
- Fault-tolerance
- Autohealing
- Isolation
- Scaling
- Autoscaling
- Load balancing

Google Kubernetes Engine (GKE)

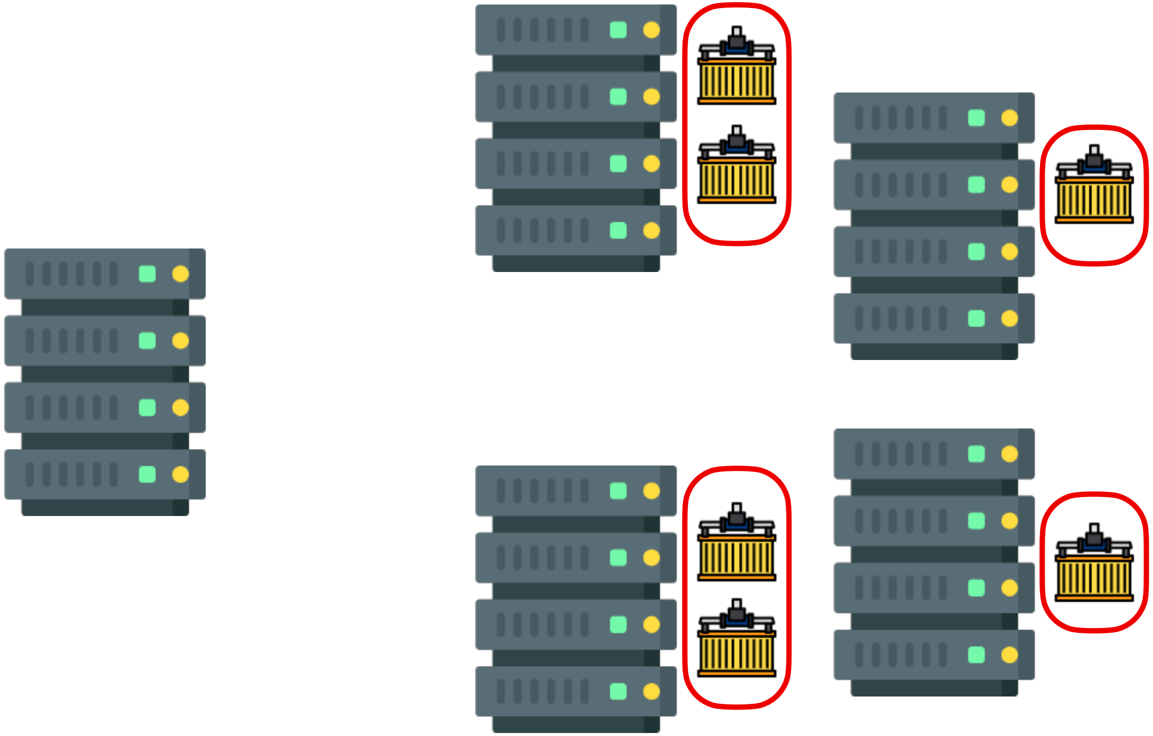
- Service for working with Kubernetes clusters on GCP
- Runs Kubernetes on GCE VM instances
- Many more abstractions and a lot more support than using plain Kubernetes on-premises

Kubernetes Clusters

Kubernetes Clusters

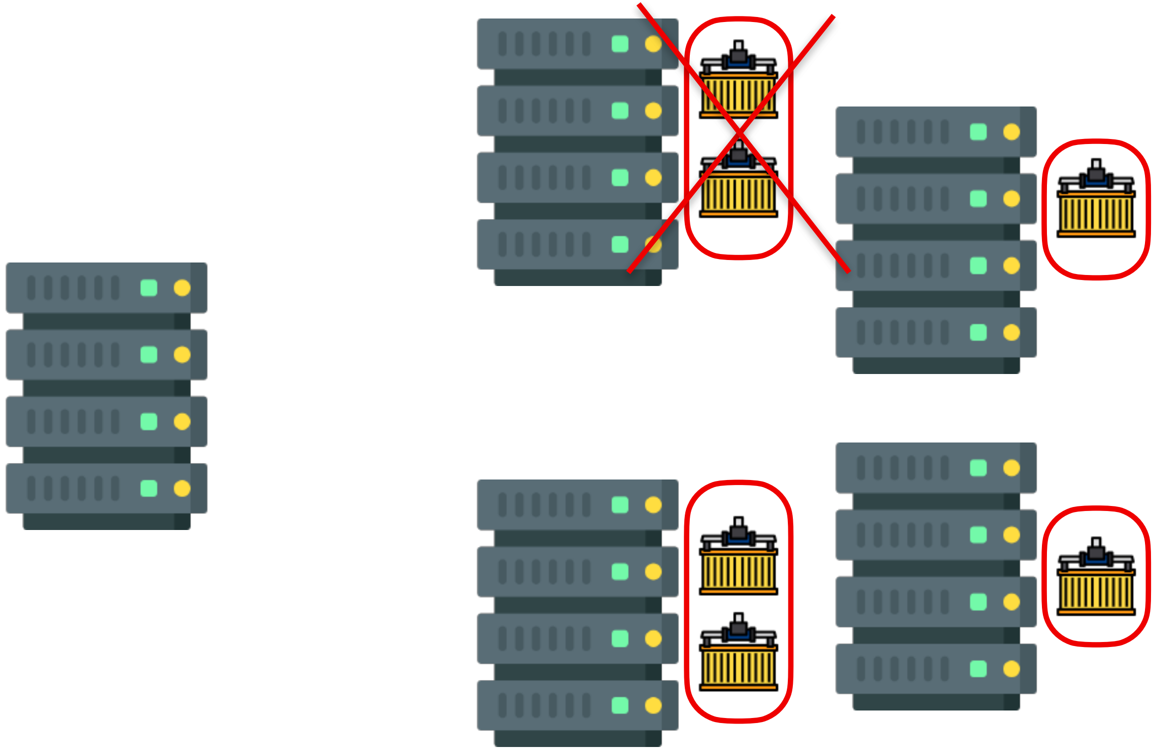


Containers Deployed in Pods



A pod is the smallest and simplest unit of deployment in Kubernetes

Pods are Ephemeral



Kubernetes replaces pods if they fail, and they may be recreated or moved across nodes. Each pod gets a unique IP, so new pod instances don't retain the same identity as previous ones.

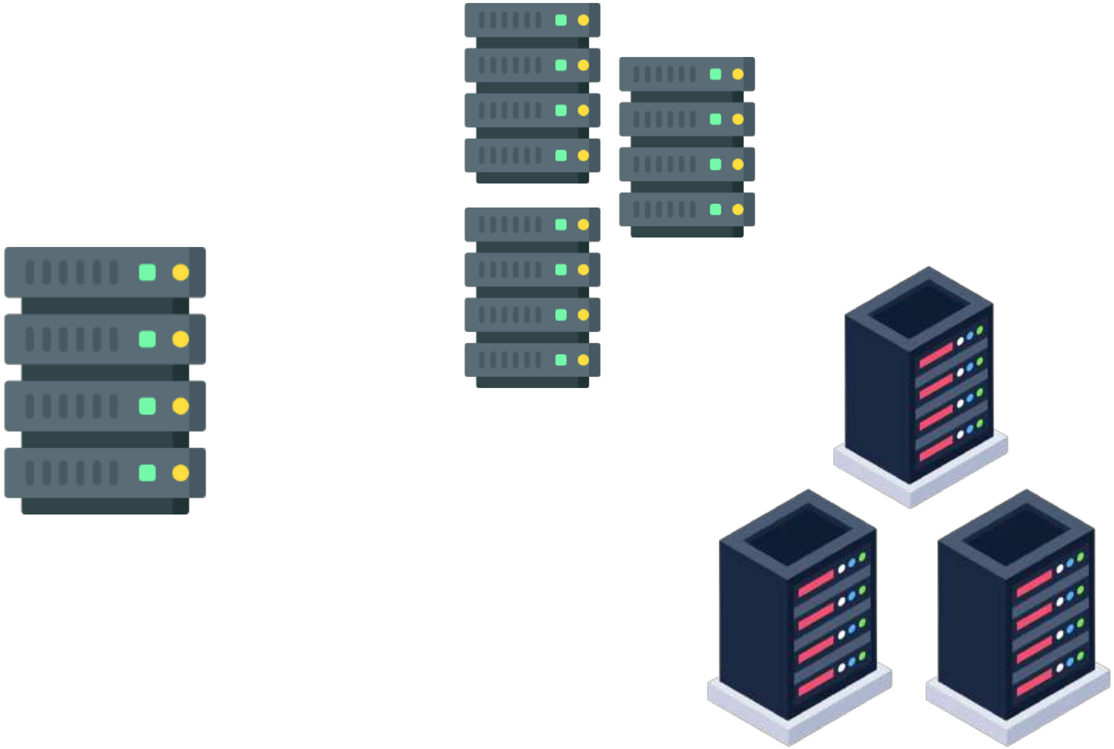
Nodes

On-premises or cloud VMs on which the the containers are run

Nodes

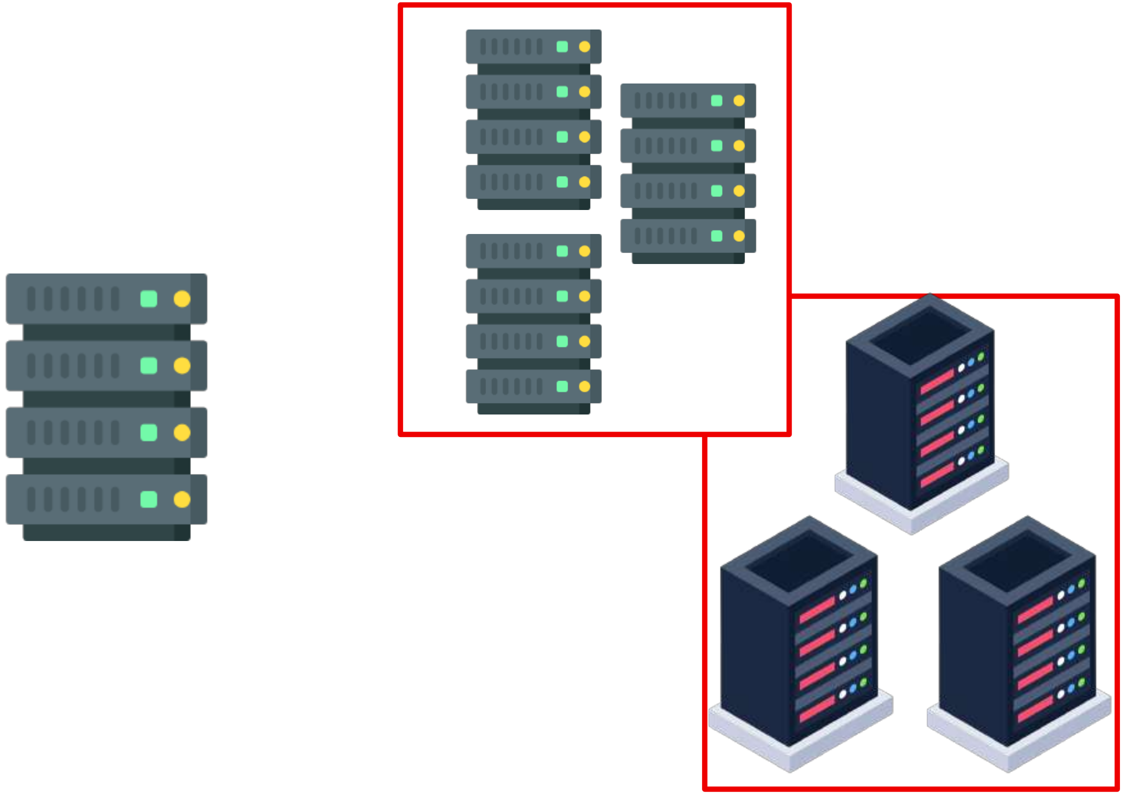
Run services to run containers e.g. Docker containers, communicate with the master node

Node Pools



Groups of nodes in your cluster that have the same configuration settings such as machine type, disk size, and labels

GKE Clusters Can Have Multiple Node Pools



Each node pool can be optimized for different workloads or operational needs

Each Node Pool Can be Configured to Autoscale

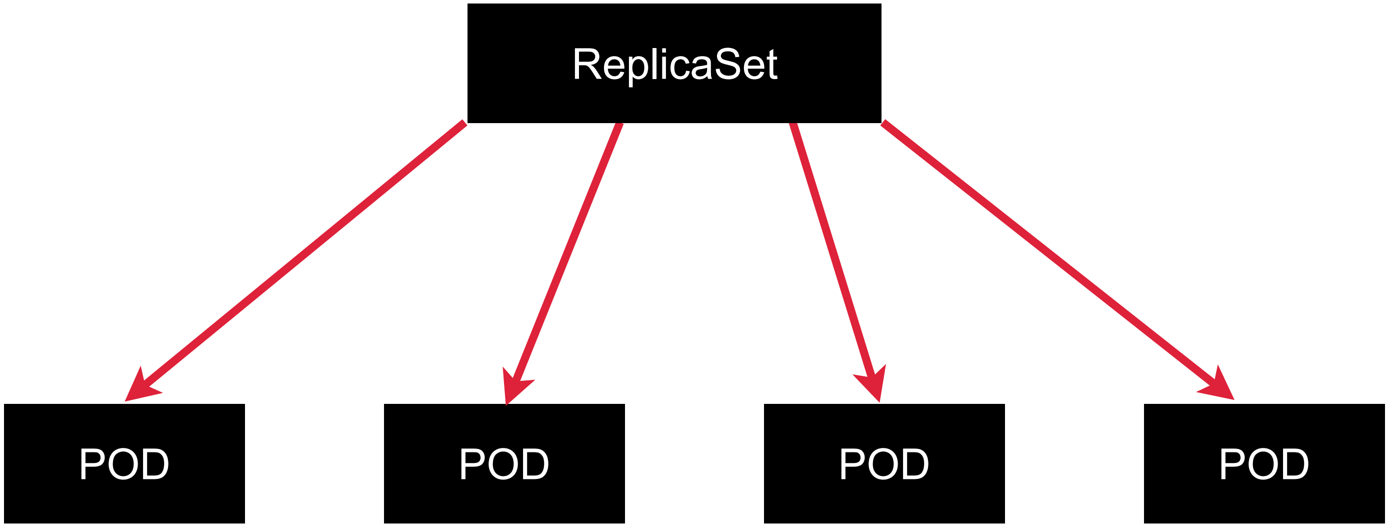
Optimize resource allocation in a workload specific manner

Uses of Node Pools

- Each node pool has its own **custom configuration**
  - e.g. GPUs, Spot VMs
  - Tailor different node pools for different workloads
- **Workload segregation** using node pools
  - Separate workloads that need high computing power, fault tolerant workloads, critical workloads

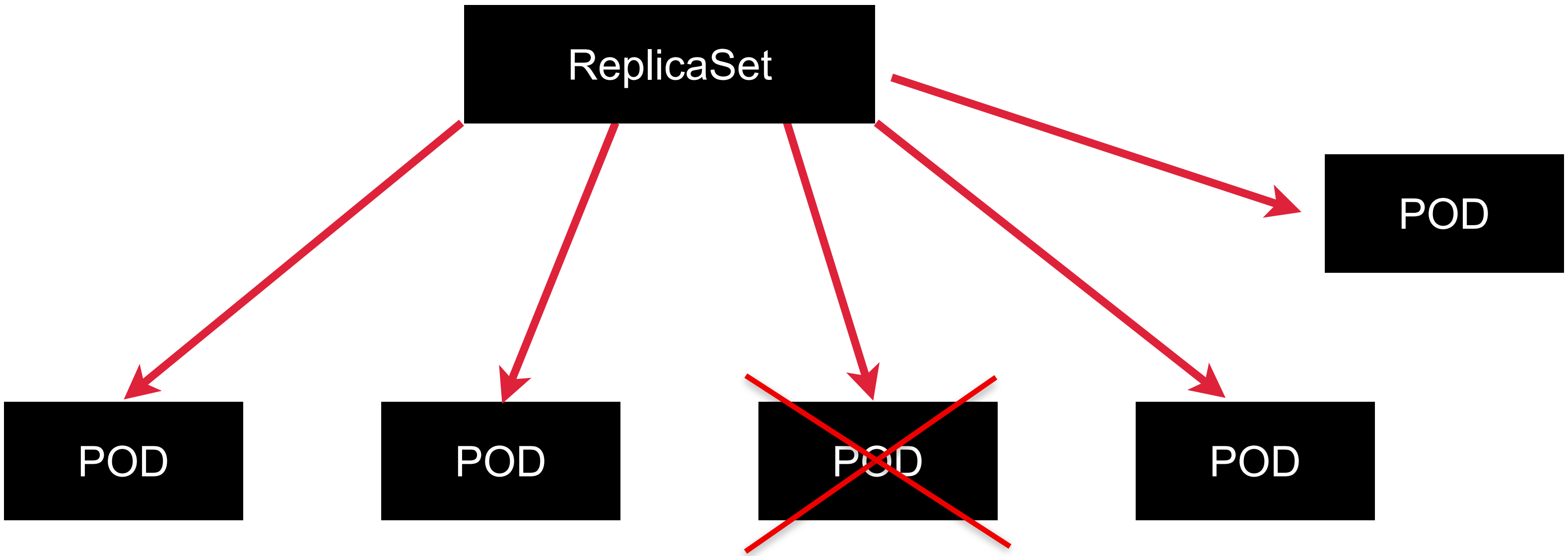
Kubernetes uses higher level abstractions to deal with containers running in the cluster

ReplicaSets



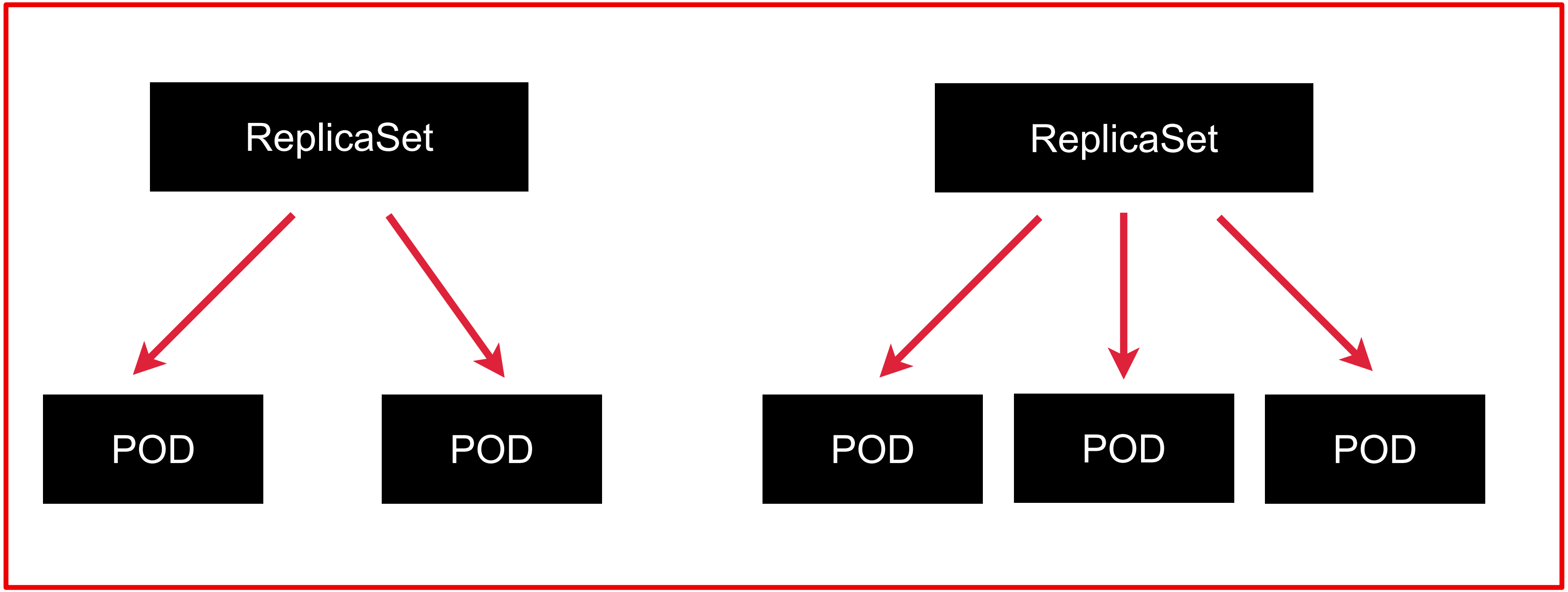
Kubernetes abstraction responsible for maintaining a specified number of identical pod instances running at all times

ReplicaSets



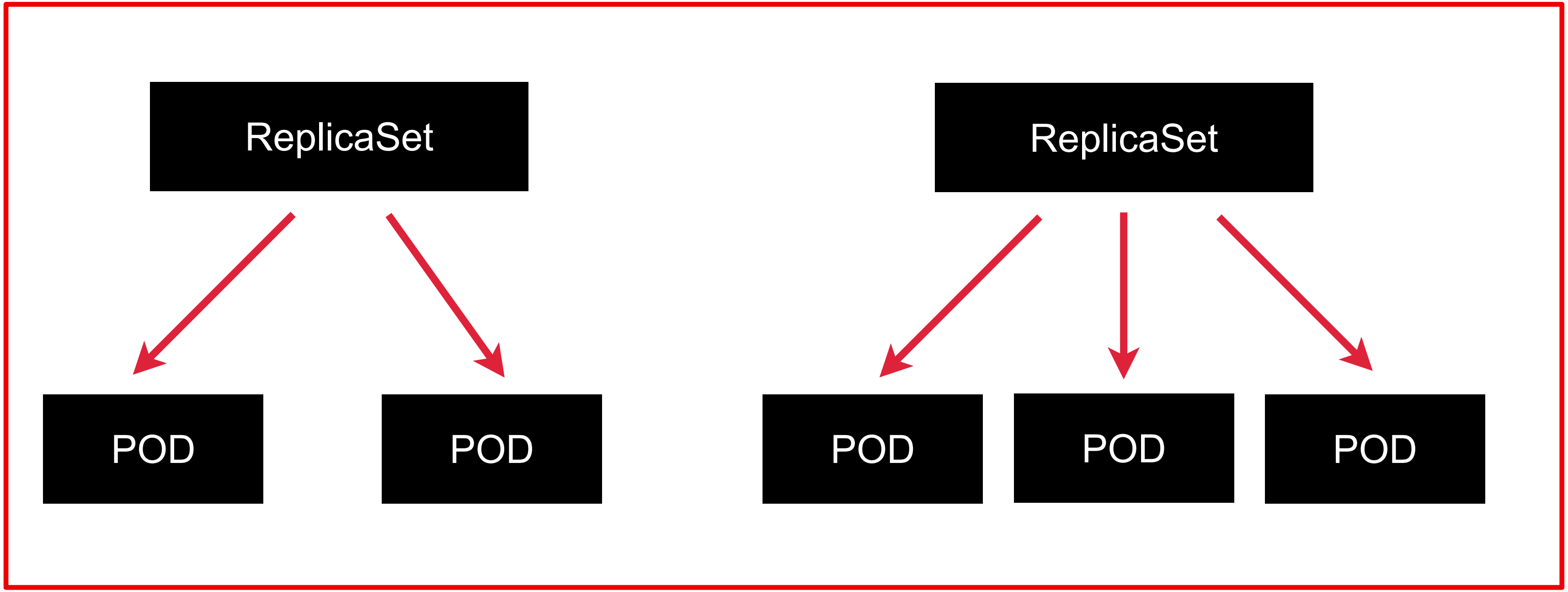
If a pod crashes the ReplicaSet will create a new pod to replace the crashed pod - this maintains the high availability and resilience of the application

Deployment



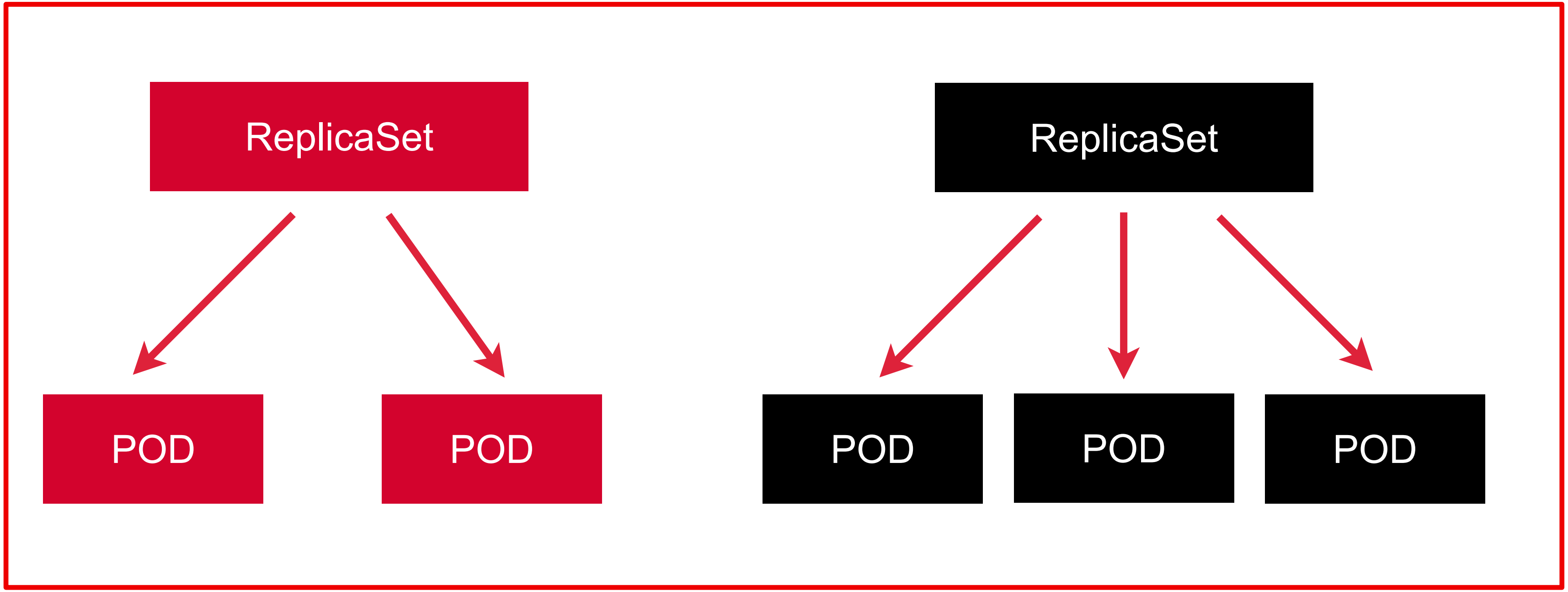
Higher level abstraction that manages ReplicaSets

Deployment - Versioning and Rollback



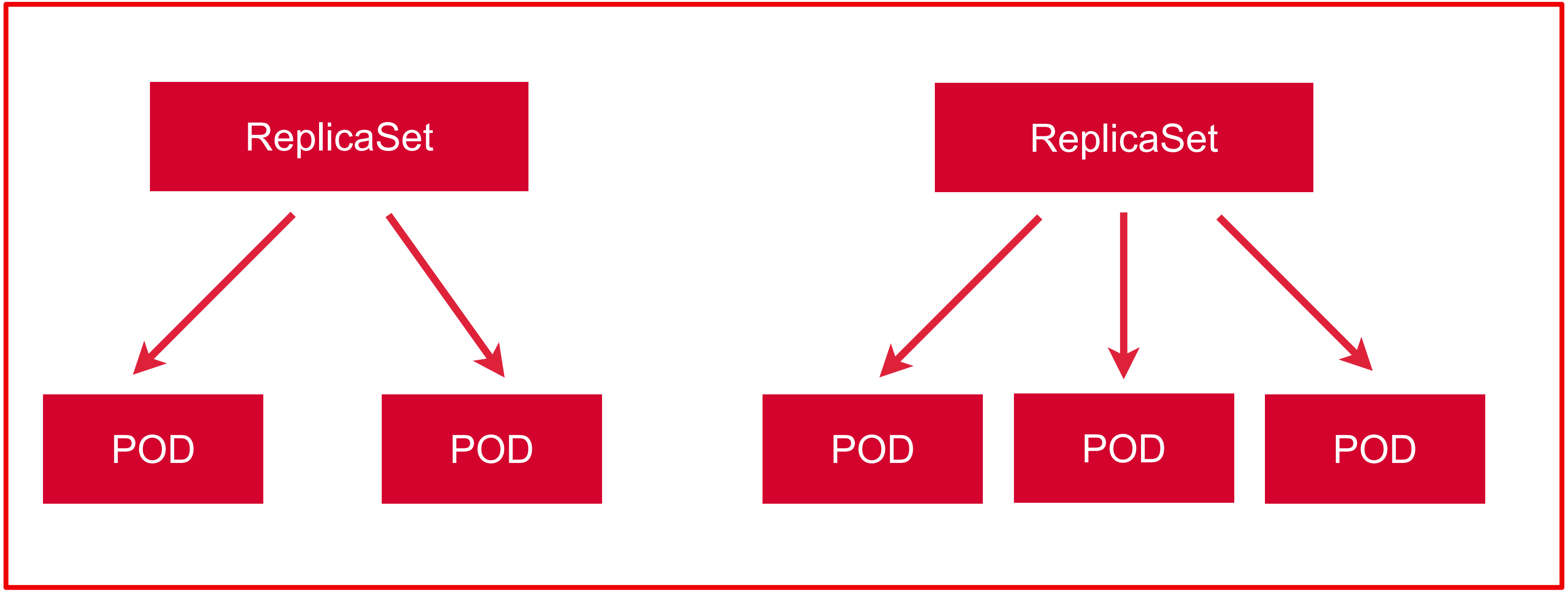
Adds versioning and rollback functionality to applications

Deployment - Rolling Updates



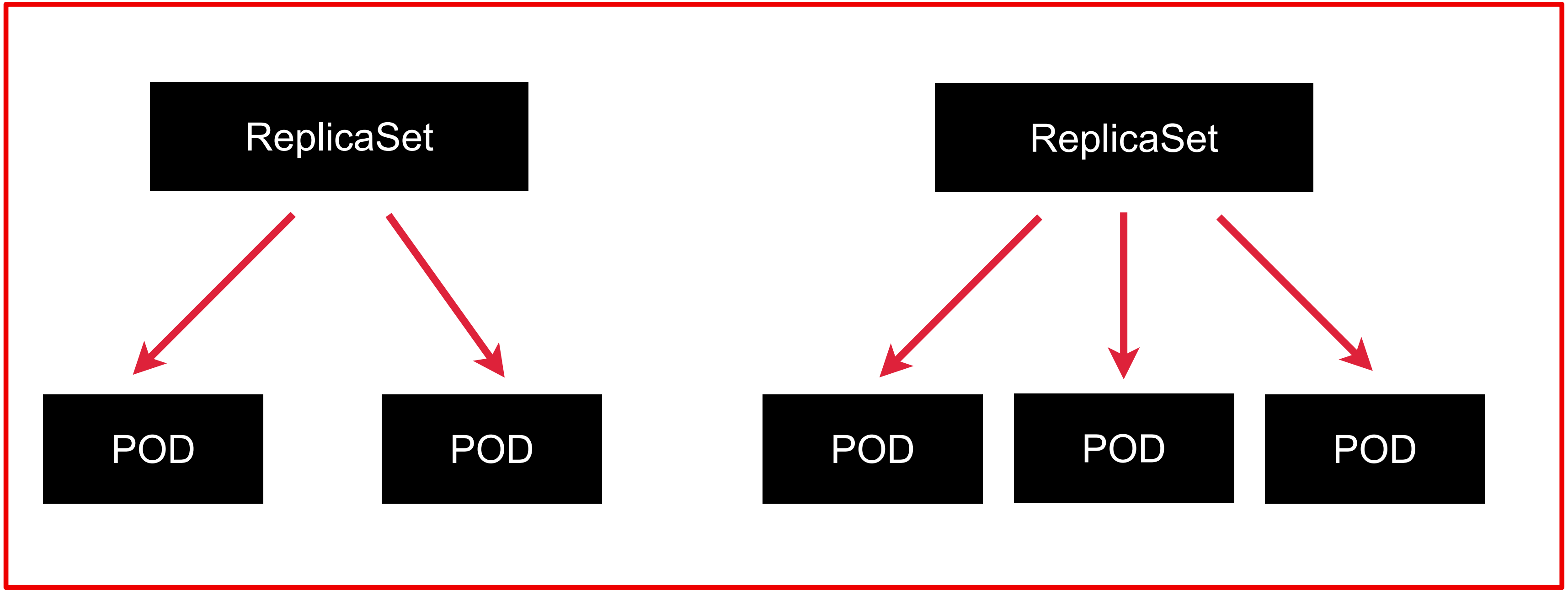
When deploying a new version of an application, Kubernetes performs a rolling update, gradually replacing old pods with new ones. This ensures zero downtime by keeping the application available during the update process.

Deployment - Rolling Updates



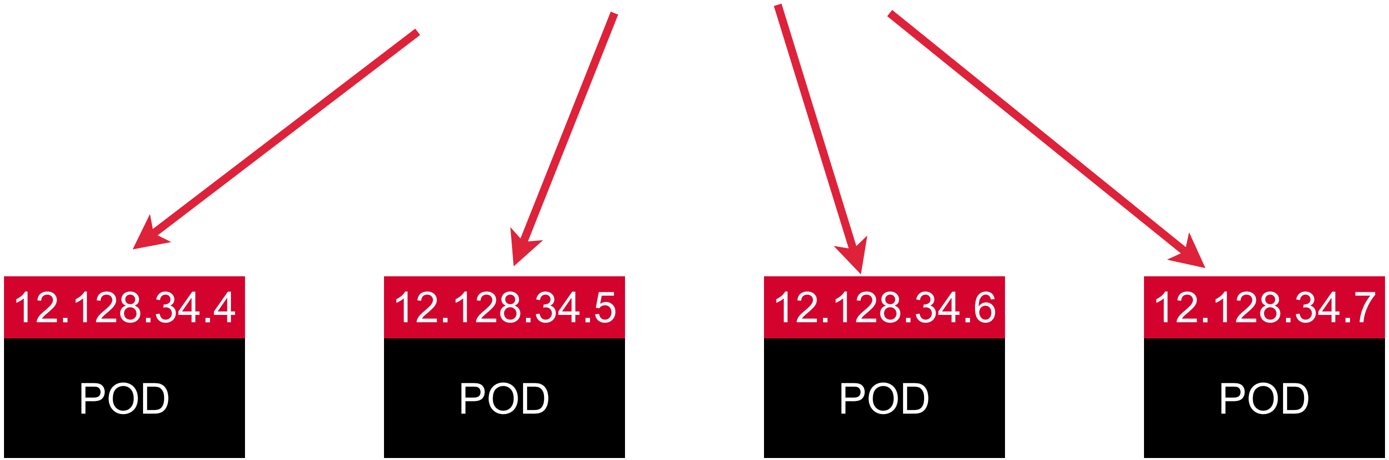
If updates cause problems Kubernetes can quickly rollback the Deployment to a previous version

Deployment - Rollbacks



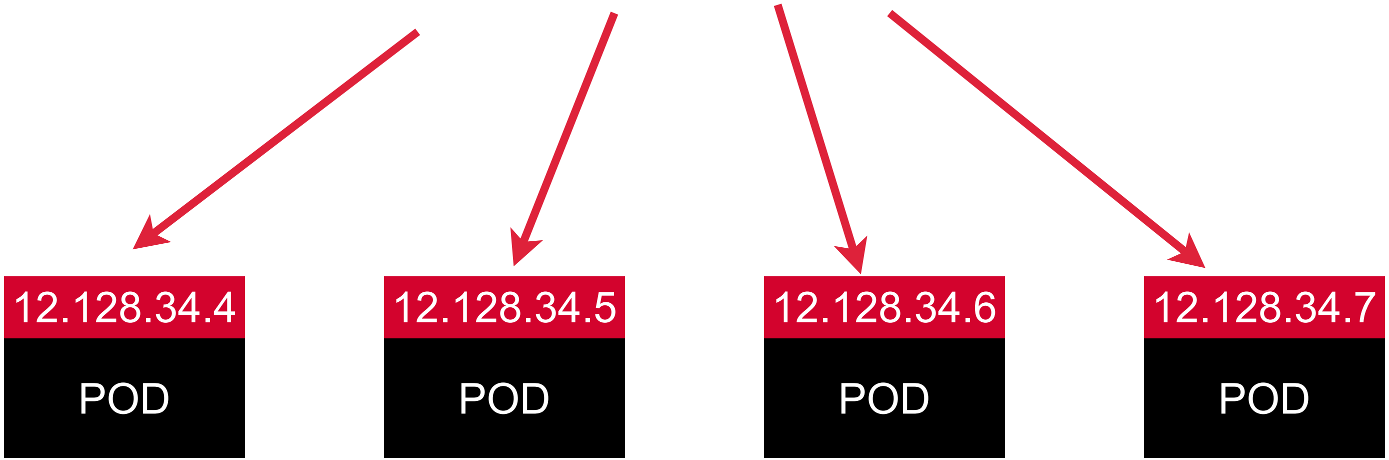
If updates cause problems Kubernetes can quickly rollback the Deployment to a previous version

Pod IP Addresses are Ephemeral



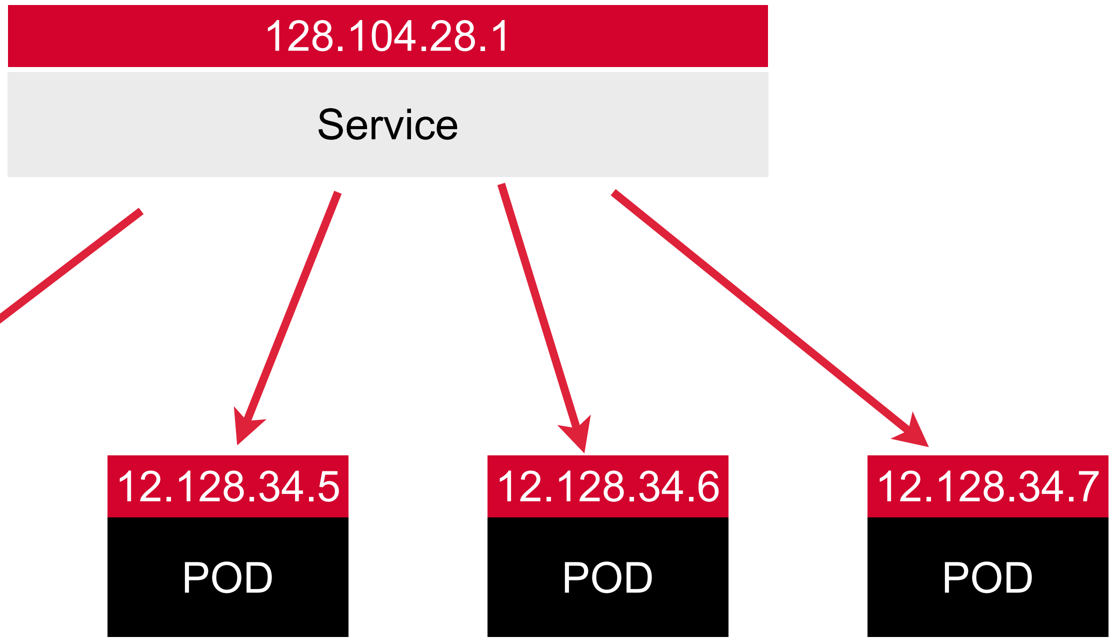
Each time a pod is recreated it will be assigned a new IP address

Pod IP Addresses are Ephemeral



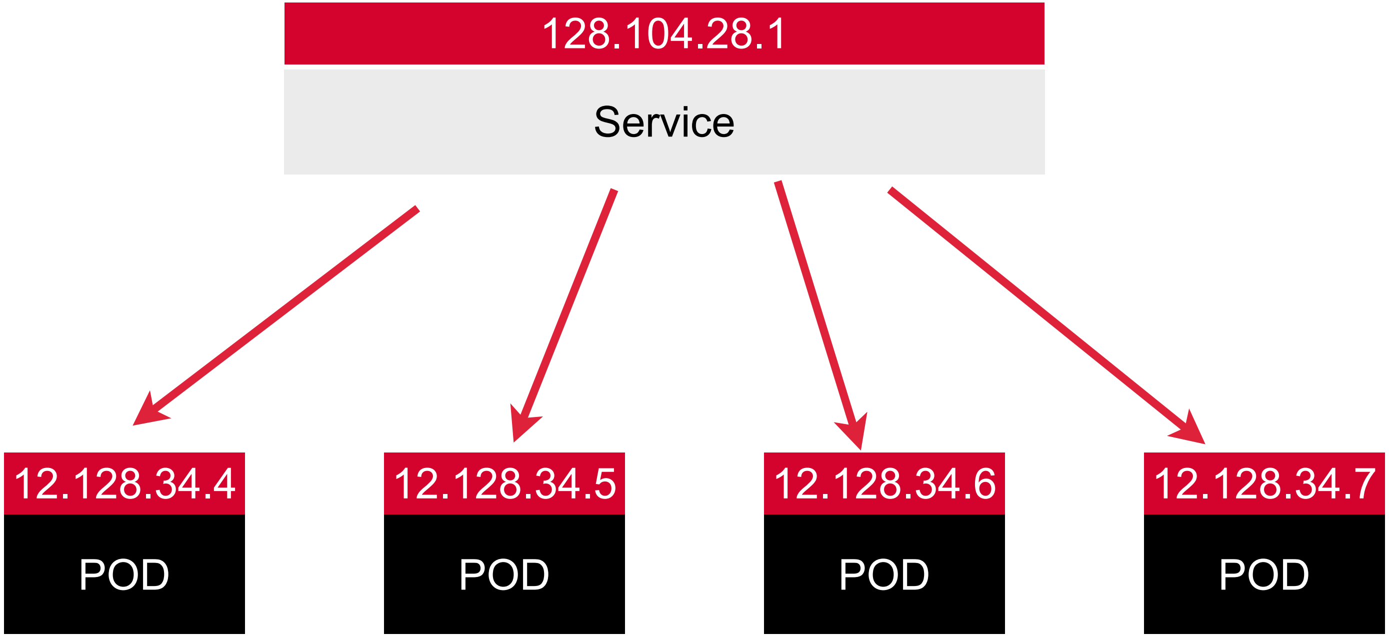
Cannot route traffic to pods using their IP addresses

Service



Provides a consistent IP address or DNS name that remains constant even as the underlying pods are created or destroyed

Service

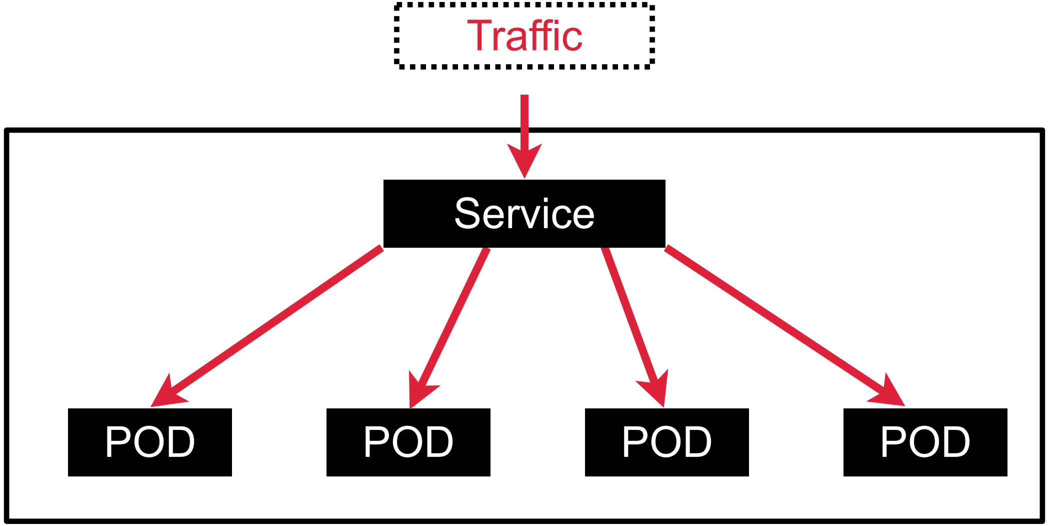


Makes it easy for users, other services or pods to connect to the application running in pods

Types of Services

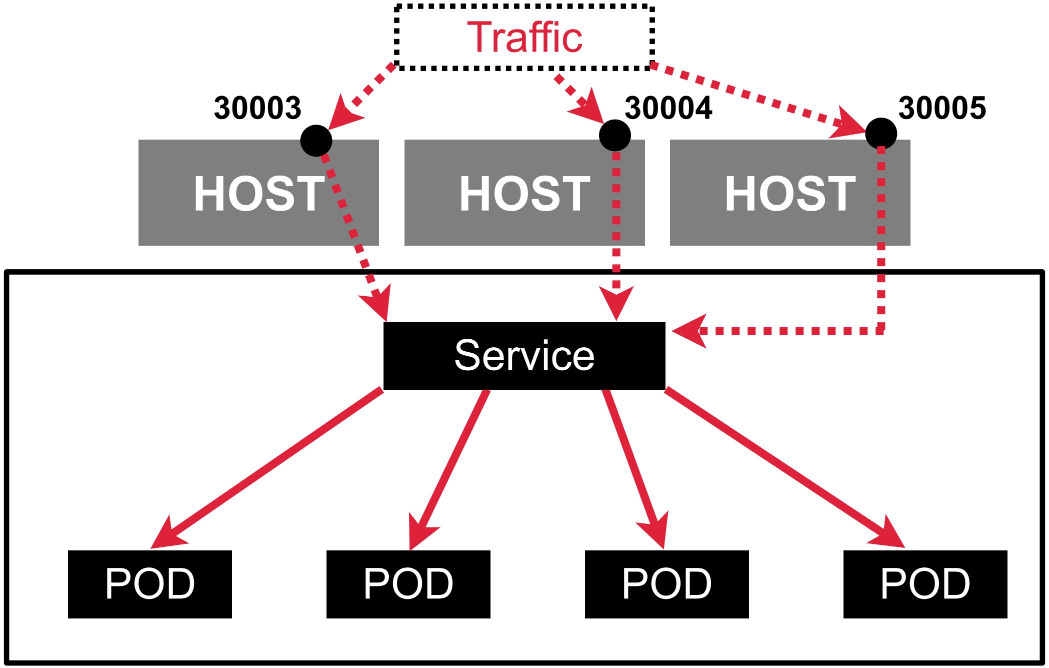
- **Cluster IP**
  - Exposes the service only inside the cluster accessible to other pods in the cluster
- **NodePort**
  - Exposes the service on a specific port of each node
- **LoadBalancer**
  - Integrates with cloud providers to provision a load balancer making the services accessible outside the cluster

ClusterIP



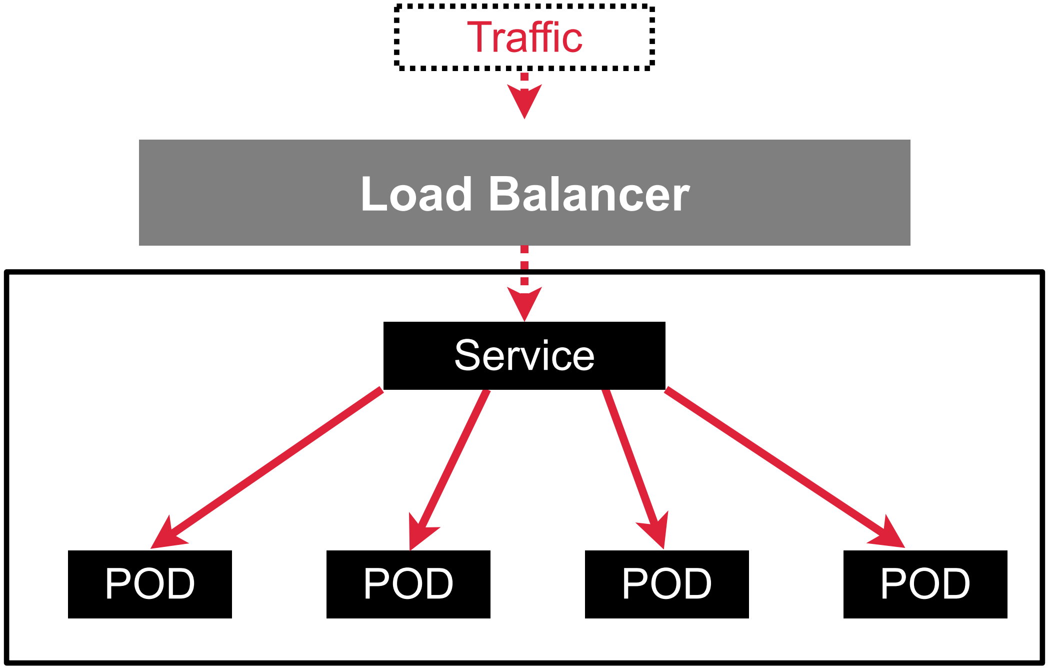
Great for internal connectivity between different services in an application that don't need to be exposed to the outside world

NodePort



Opens up a designated port on each node that forwards traffic to the service - ideal for applications that need to be accessed outside the cluster like web apps or APIs

LoadBalancer



Allows external connections to the service and distributes traffic to the different nodes running the service

Kubernetes Workloads

- Stateless workloads
- Stateful workloads
- Batch and job workloads
- Monitoring and logging workloads

Kubernetes Workloads

Stateless workloads

Stateful workloads

Batch and job workloads

Monitoring and logging workloads

Stateless Workloads

- Most commonly run using the **Deployment** object
  - Each pod operates independently
  - Does not retain data between sessions
- Web applications, microservices, and APIs that do not store data locally in the pod

Kubernetes Workloads

- Stateless workloads
- **Stateful workloads**
- Batch and job workloads
- Monitoring and logging workloads

Stateful Workloads

- **StatefulSet** object used to run stateful applications
    - Apps require persistent storage and unique identities for each pod
- Databases, message queues, other applications that need stable storage and consistent naming

Kubernetes Workloads

- Stateless workloads
- Stateful workloads
- Batch and job workloads
- Monitoring and logging workloads

Batch and Job Workloads

- The **Job** object managed one-off task or batch processing
  - A single job that needs to run to completion
- The **CronJob** is a specialized job that runs on a scheduled bases
  - Useful for tasks to be executed at regular intervals

Kubernetes Workloads

- Stateless workloads
- Stateful workloads
- Batch and job workloads
- Monitoring and logging workloads

Monitoring and Logging Workloads

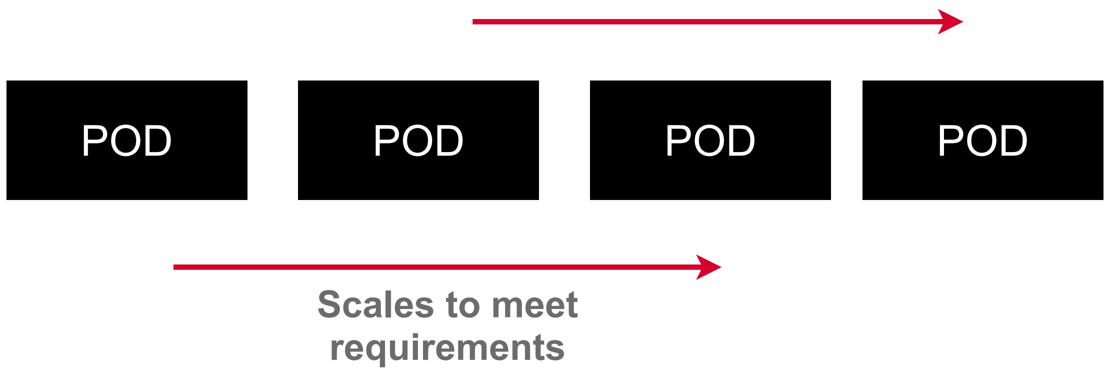
- A **DaemonSet** ensures that a copy of a pod is running on every node in the cluster
  - Used for system-level services running on every node
  - Logging agents, monitoring agents, or node-level proxies

Autoscaling

Horizontal Pod Autoscaler

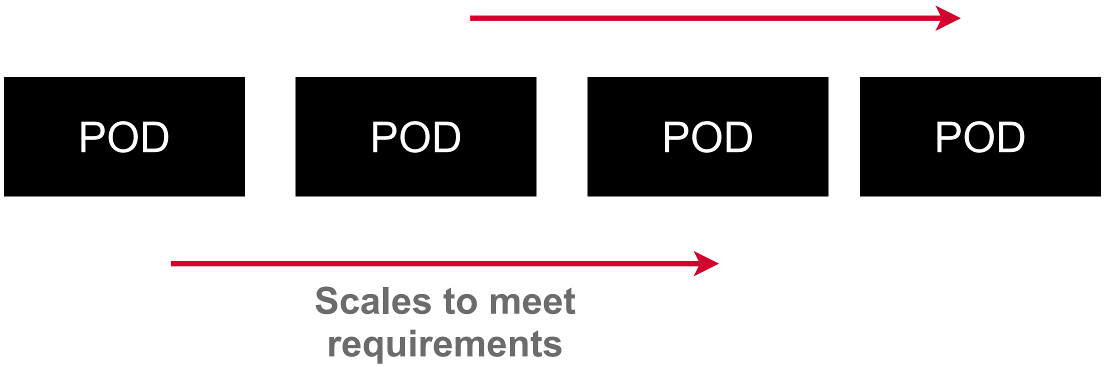
Vertical Pod Autoscaler

Horizontal Pod Autoscaler



Adjusts the number of pods based on metrics such as CPU utilisation, memory usage, or other custom metrics

Horizontal Pod Autoscaler

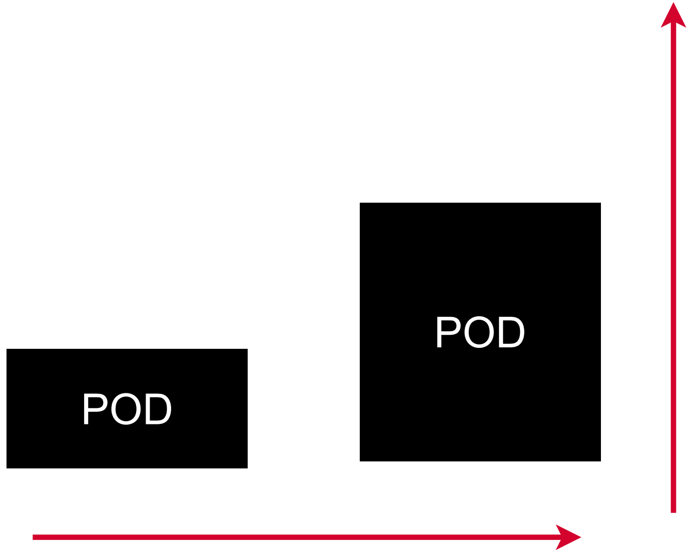


Monitors the performance metrics of pods and adds pods to match load - useful for stateless applications that handle fluctuating traffic

The HPA only scales pods on a cluster

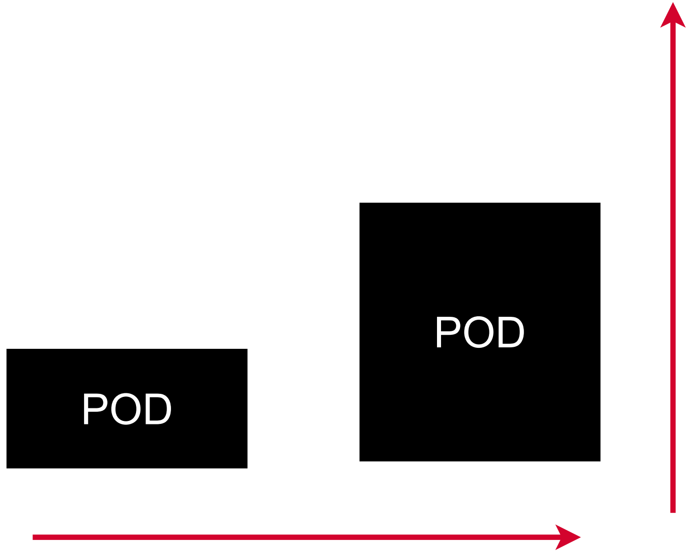
Scaling cluster nodes requires us to enable cluster autoscaling separately on the cluster

Vertical Pod Autoscaler



Adjusts resource requests and CPU and memory limits of existing pods based on real-time usage

Vertical Pod Autoscaler



Scales pods vertically by increasing or decreasing the resources over time - useful for stateful applications that need more compute or memory as the application scales

GKE Mode of Operation

Autopilot Mode

Standard Mode

Autopilot Mode

- More managed GKE experience
- GKE manages the underlying infrastructure - get a more serverless experience
- Node configuration, autoscaling, auto-upgrades, baseline security and networking configurations
- Implements best practices for security, scalability, and cost optimization by default

Autopilot Mode

- **Cost effective**: Only pay for compute resources that your workloads use while running
- **Automation**: Google creates and manages nodes, scales nodes and workloads based on traffic
- **Security**: Enables many security settings and automatically applies security patches

Standard Mode

- Complete control over all GKE configuration settings
- Manage configurations for node pools, security, scheduling, scaling, resource management, version management and software upgrades

Use Standard Mode If:

- You want granular control over your configuration settings
- You want to install or modify software running on the nodes themselves i.e. change node OS
- Use certain features that are only available in the Standard Mode (GKE Sandbox, Cloud TPU)
- Test alpha features in open source Kubernetes

Google recommends that you use an Autopilot cluster unless you have a specific need for a Standard cluster

Public and Private Clusters

Public clusters	Private clusters

Public and Private Clusters

Public clusters

Private clusters

Both the control plane (the Kubernetes API server) and the nodes have public IP addresses by default, making them reachable from the internet.

Public and Private Clusters

Public clusters

Private clusters

Can use `kubectl` from external networks (the internet) to manage the cluster

Public and Private Clusters

Public clusters

Private clusters

The nodes are stripped of their public IP addresses, providing a significant layer of network isolation. This drastically reduces the cluster's attack surface.

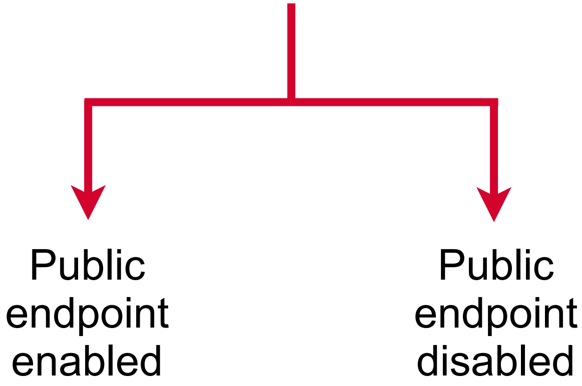
Public and Private Clusters

Public clusters

Private clusters

To allow workloads on private nodes to access the internet (e.g., for pulling external images or calling external APIs), you must configure **Cloud NAT** for the subnet.

Private Clusters



Private Clusters

Public endpoint enabled

Public endpoint disabled

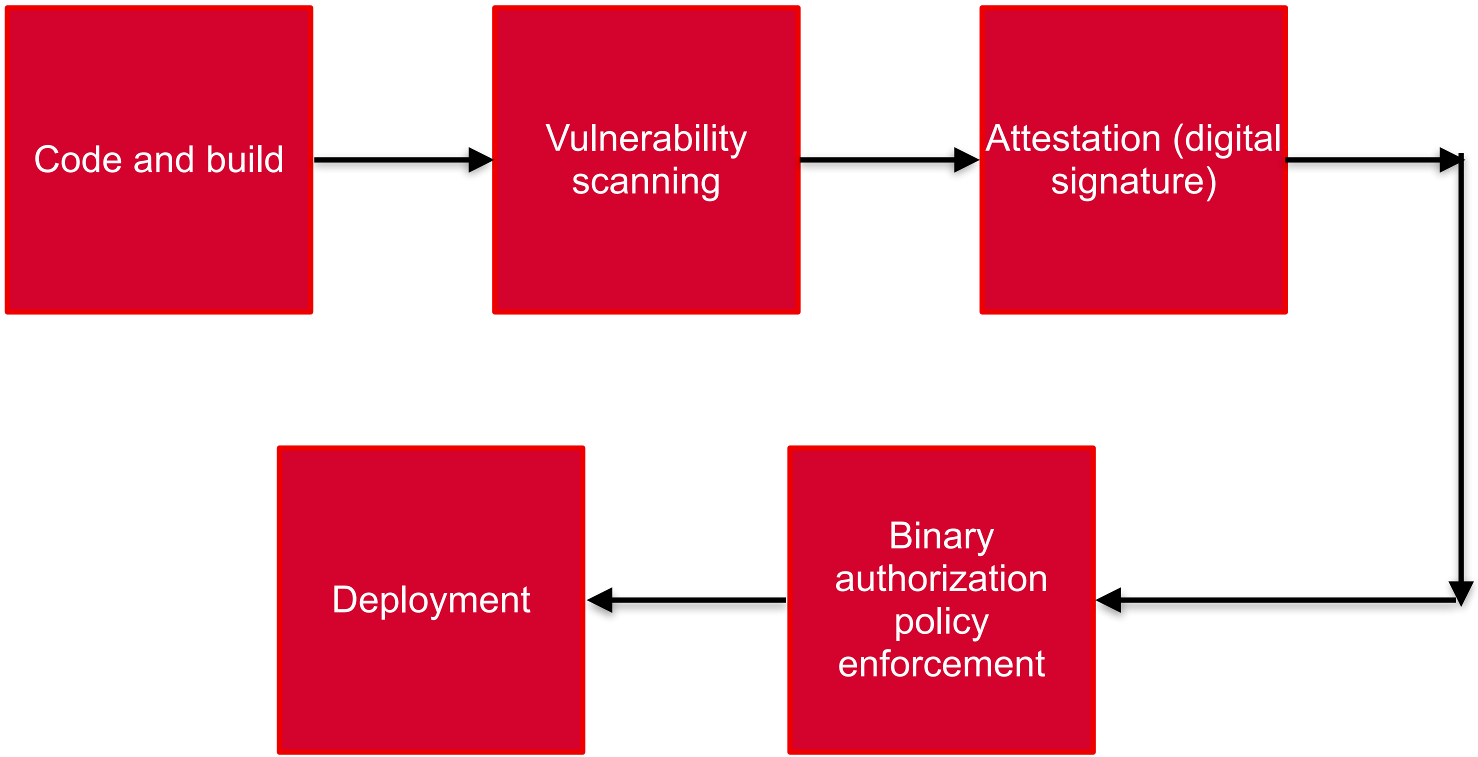
The control plane has both a public and a private endpoint. You can still manage the cluster from the internet (ideally restricted with authorized networks), while nodes communicate with it over the private endpoint.

This is the most secure option. It is completely inaccessible from the public internet - can connect only from the same network or interconnected network

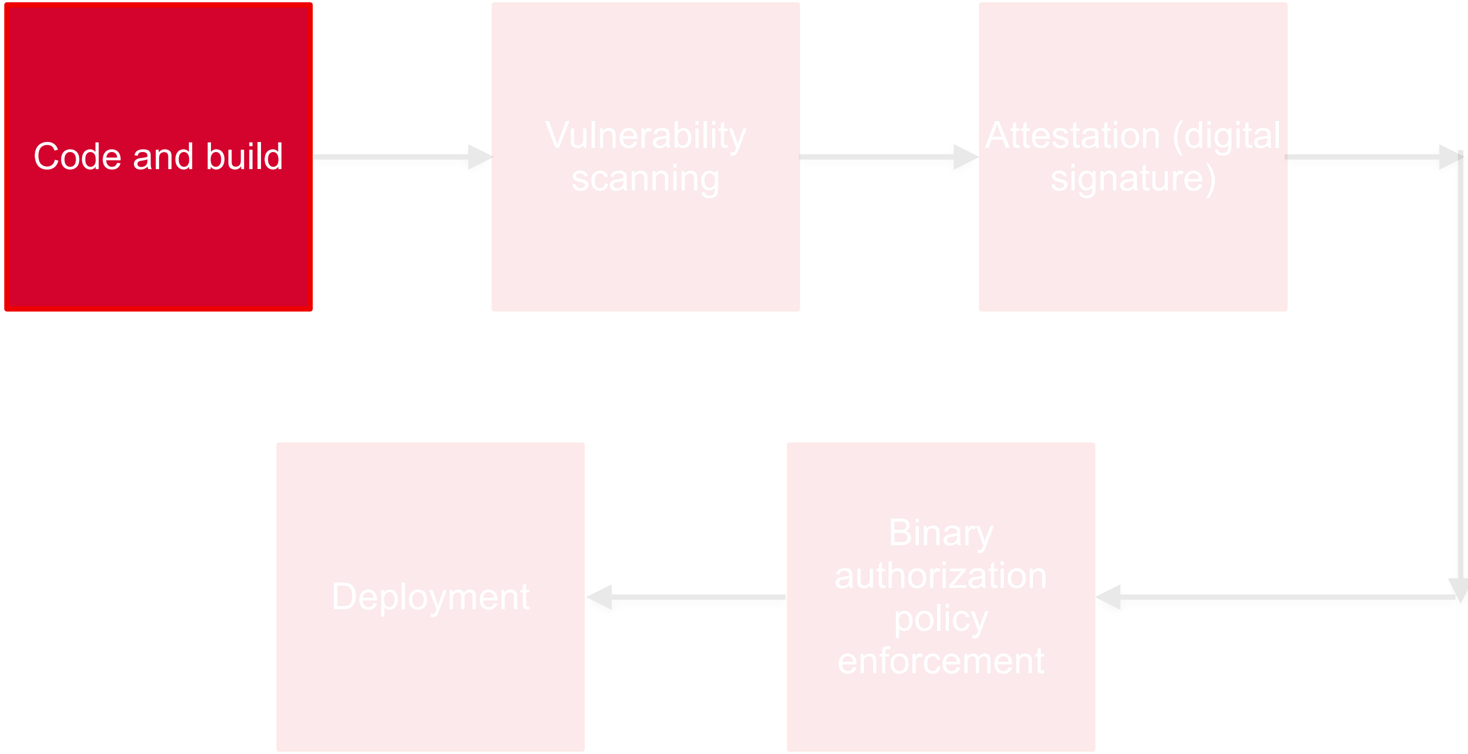
Binary Authorization

Security service for Google Kubernetes Engine (GKE) that enforces a strict policy at deployment time, ensuring that only **trusted and verified container images** can be deployed into your environment.

Binary Authorization



Binary Authorization



Commit code and build a container image (e.g. using Cloud Build)

Binary Authorization

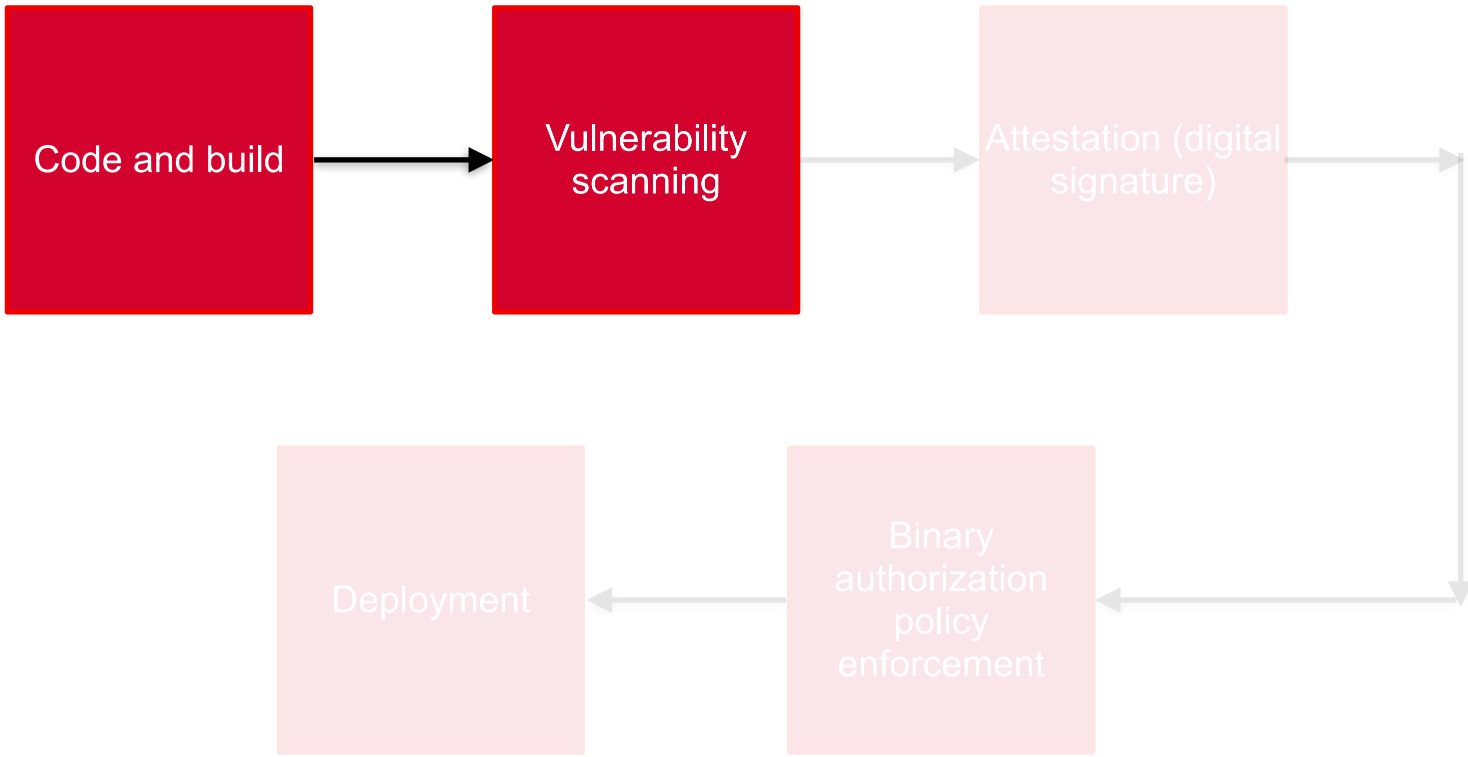
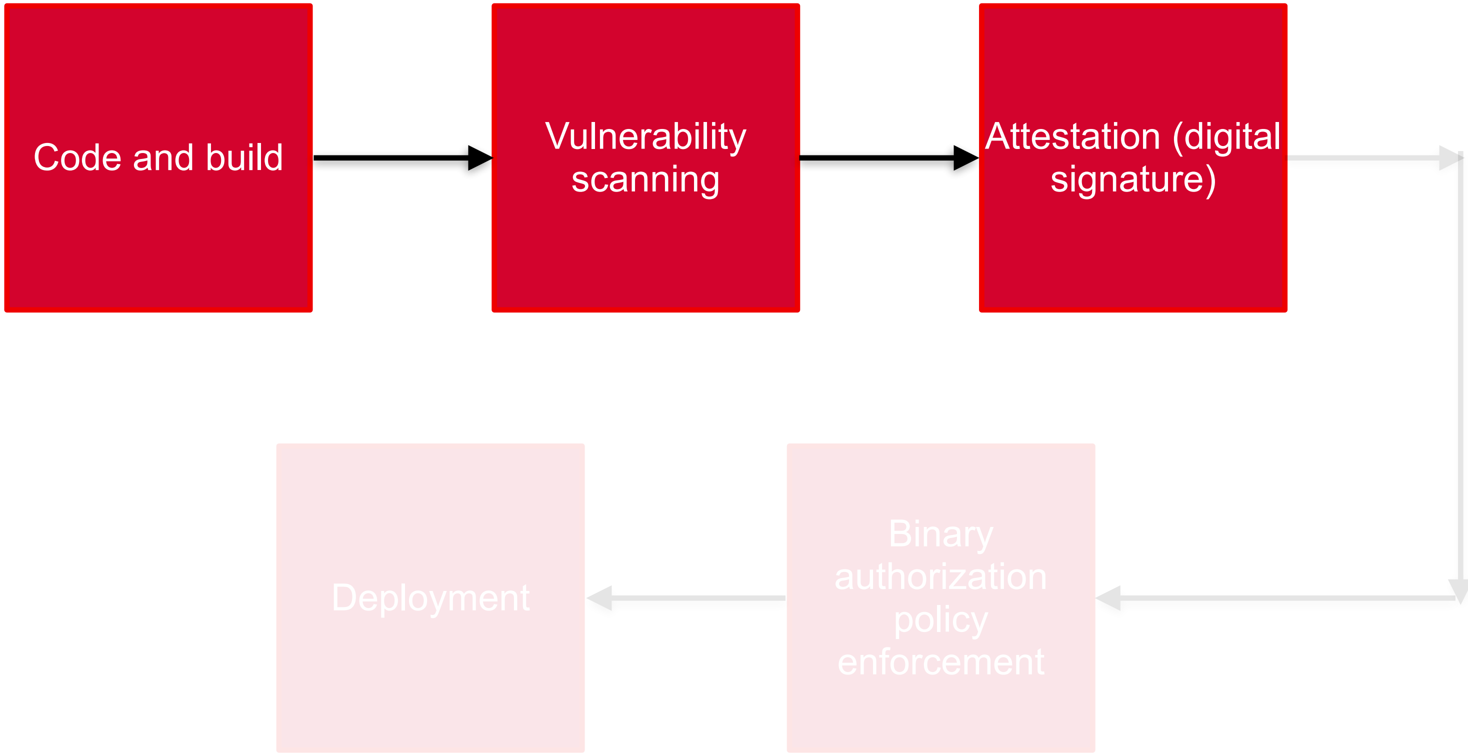


Image pushed to a registry (Artifact Registry) and automatically scanned for common vulnerabilities

Binary Authorization

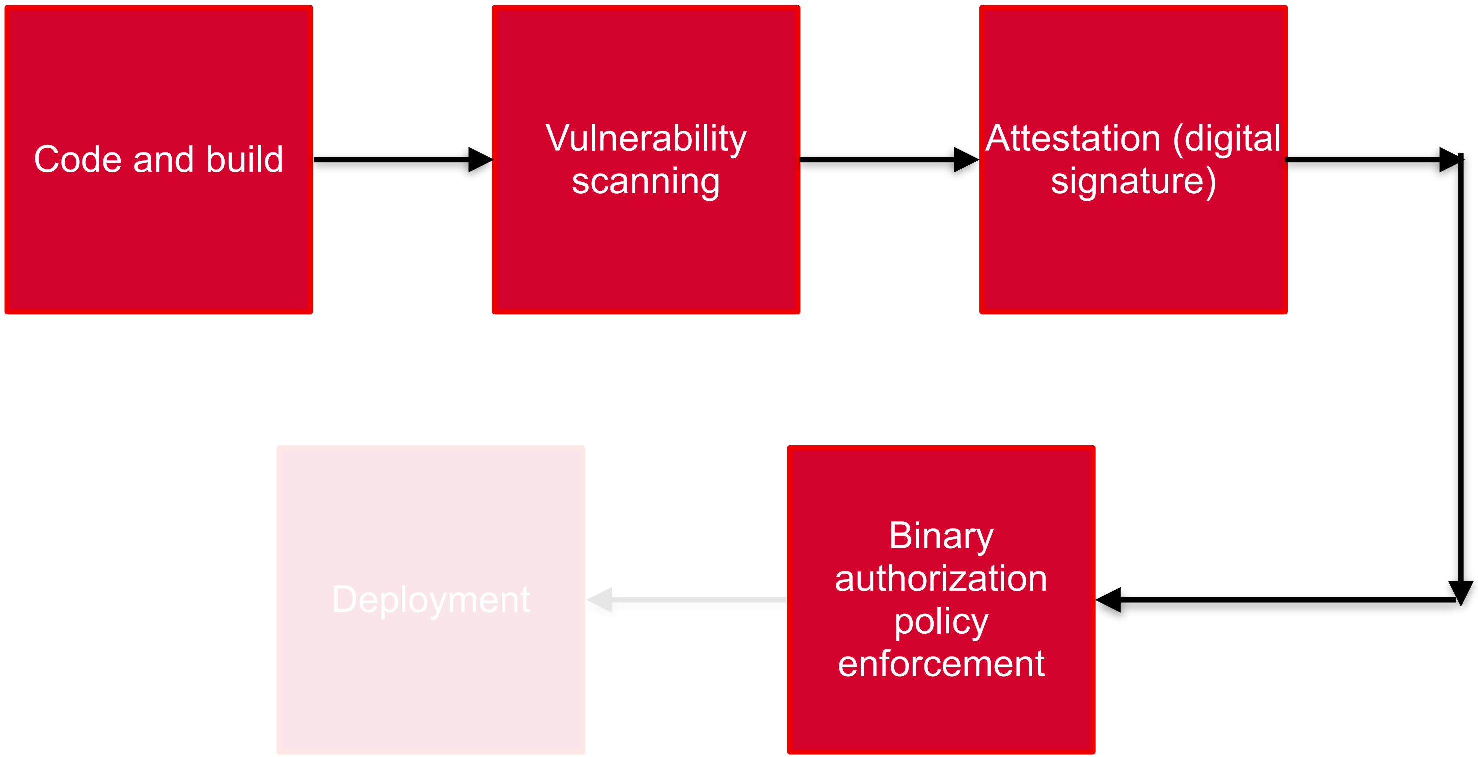


A trusted authority called attestor digitally signs image

Attestor and Attestation

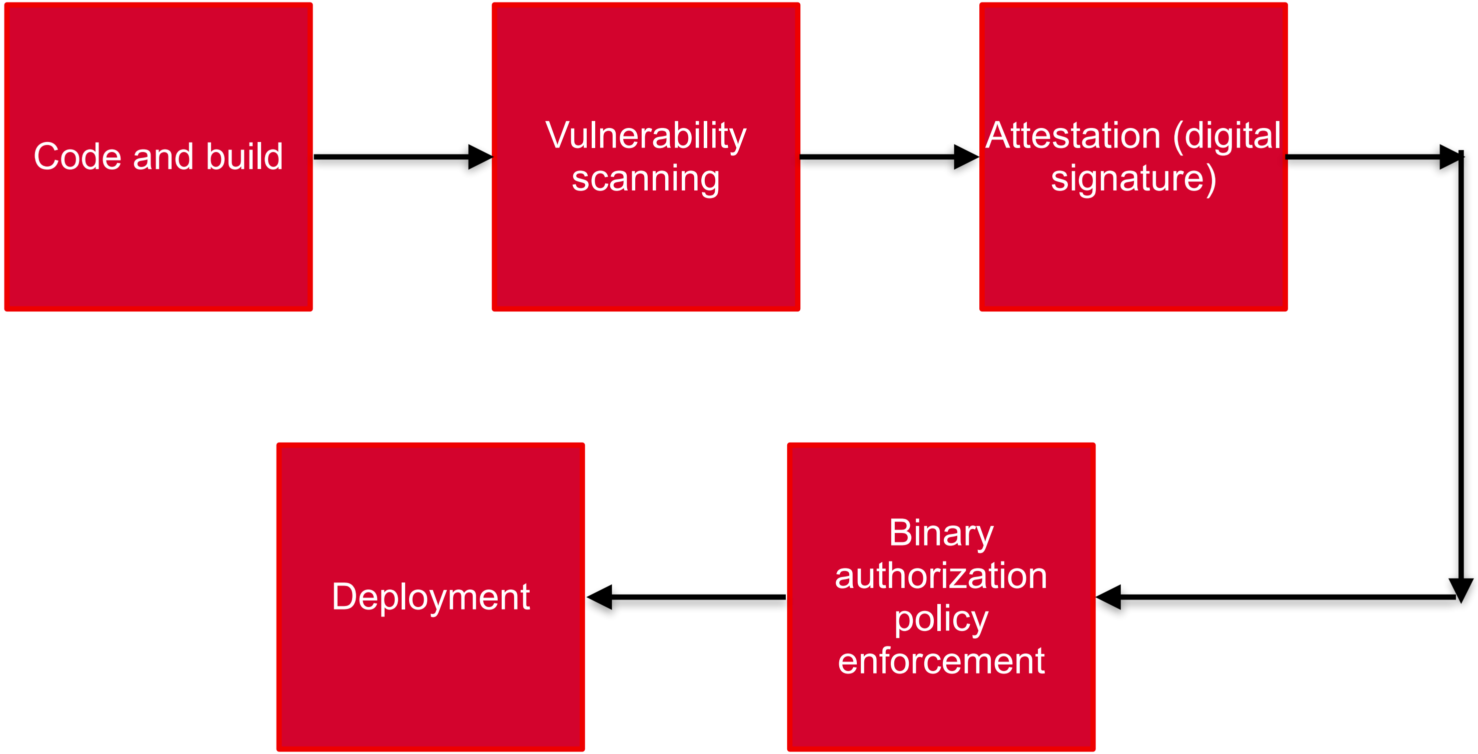
- Attestor: Cloud resource that uses cryptographic keys to sign an image
  - Different attestors for different stages `(qa-passed, no-vulnerabilities)`
- Attestation: Signing creates an attestation

Binary Authorization



A binary authorization policy enforces rules e.g `qa-passed, no-vulnerabilities` should both have passed

Binary Authorization



Enforcer checks if all attestations are present and only then allows the deployment to proceed

Google Kubernetes Engine

An online retail company runs its entire backend on a large-scale Google Kubernetes Engine (GKE) environment. The environment consists of a single cluster with a large, statically-sized node pool that runs 24/7. This cluster supports everything from the public-facing web storefront (a stateless application) and user session databases (stateful services), to end-of-day inventory processing jobs (batch workloads).

The finance department has mandated a reduction in cloud infrastructure spending, but the engineering team must not allow this to impact website uptime or responsiveness for customers. What is the most effective strategy to meet these requirements?

A. Implement a two-tiered autoscaling strategy: Use the Horizontal Pod Autoscaler (HPA) and enable the GKE Cluster Autoscaler to dynamically resize the node pool as needed.
B. Re-architect the environment by provisioning a second, smaller GKE cluster exclusively for the inventory processing jobs, and manually re-allocate some nodes from the main cluster to the new one.
C. Focus on workload stability by defining strict CPU and memory requests and limits for every deployment in the cluster, ensuring no single pod can consume excessive resources.
D. Achieve maximum cost savings by converting the entire node pool to use Spot VMs, forcing all workloads including the stateful databases and web storefront to run on nodes that can be terminated at any time.

Google Kubernetes Engine

An online retail company runs its entire backend on a large-scale Google Kubernetes Engine (GKE) environment. The environment consists of a single cluster with a large, statically-sized node pool that runs 24/7. This cluster supports everything from the public-facing web storefront (a stateless application) and user session databases (stateful services), to end-of-day inventory processing jobs (batch workloads).

The finance department has mandated a reduction in cloud infrastructure spending, but the engineering team must not allow this to impact website uptime or responsiveness for customers. What is the most effective strategy to meet these requirements?

A. **Implement a two-tiered autoscaling strategy: Use the Horizontal Pod Autoscaler (HPA) and enable the GKE Cluster Autoscaler to dynamically resize the node pool as needed**.
B. Re-architect the environment by provisioning a second, smaller GKE cluster exclusively for the inventory processing jobs, and manually re-allocate some nodes from the main cluster to the new one.
C. Focus on workload stability by defining strict CPU and memory requests and limits for every deployment in the cluster, ensuring no single pod can consume excessive resources.
D. Achieve maximum cost savings by converting the entire node pool to use Spot VMs, forcing all workloads including the stateful databases and web storefront to run on nodes that can be terminated at any time.

Google Kubernetes Engine

You are designing a backend order-processing system on Google Kubernetes Engine. The system consists of multiple internal components, including an order-intake service and a payment-processing service. The order-intake service needs to reliably communicate with the payment-processing service.

1. The payment-processing service is resilient and can be scaled up or down by simply changing a replica count.
2. The order-intake service can use a single, consistent address to reach the payment service, and Kubernetes will automatically load-balance requests across all available payment service replicas.
3. Communication between these services remains private and internal to the cluster.

A. For the payment-processing service, create a **Deployment**. To make it accessible, create an **Ingress** resource to expose it to the public internet, and have the order-intake service communicate via the Ingress's public IP address.
B. For the payment-processing service, manually create several individual **Pods**. Then, create a **Service** that uses a label selector to group these pods and provide a single DNS name for communication.
C. For the payment-processing service, create a **StatefulSet** to manage its pods. Then, create a **Gateway** resource to manage traffic routing from the order-intake service.
D. For the payment-processing service, create a **Deployment**. Then, create a **Service** (of type ClusterIP) that targets the Deployment's pods. The order-intake service will use the auto-generated DNS name of this Service.

Google Kubernetes Engine

You are designing a backend order-processing system on Google Kubernetes Engine. The system consists of multiple internal components, including an order-intake service and a payment-processing service. The order-intake service needs to reliably communicate with the payment-processing service.

1. The payment-processing service is resilient and can be scaled up or down by simply changing a replica count.
2. The order-intake service can use a single, consistent address to reach the payment service, and Kubernetes will automatically load-balance requests across all available payment service replicas.
3. Communication between these services remains private and internal to the cluster.

A. For the payment-processing service, create a **Deployment**. To make it accessible, create an **Ingress** resource to expose it to the public internet, and have the order-intake service communicate via the Ingress's public IP address.
B. For the payment-processing service, manually create several individual **Pods**. Then, create a **Service** that uses a label selector to group these pods and provide a single DNS name for communication.
C. For the payment-processing service, create a **StatefulSet** to manage its pods. Then, create a **Gateway** resource to manage traffic routing from the order-intake service.
D. For the payment-processing service, create a Deployment. Then, create a Service (of type ClusterIP) that targets the Deployment's pods. The order-intake service will use the auto-generated DNS name of this Service.

Google Kubernetes Engine

A financial services company runs its critical payment application on a production Google Kubernetes Engine (GKE) cluster. For strict compliance, the company must enforce a policy that only allows container images to be deployed if they have passed two specific quality gates:

1. A successful vulnerability scan from an automated security tool.
2. A manual sign-off from the Quality Assurance (QA) team.

How do you prevent the deployment of images who have not passed these checks?

A. In the application's Deployment manifest, add a postStart lifecycle hook that calls an external script to verify the image's approval status and stops the container if it fails.

B. Use Cloud Source Repositories to host the container images and configure IAM permissions so that only the QA team lead has rights to deploy to the production cluster.

C. Use Binary Authorization to define a policy for the production cluster that attestations for each quality gate before an image can be deployed. Integrate the creation of these attestations into your CI/CD process.

D. Write and deploy a custom validating admission webhook. The webhook's logic will query an external database of approved images to either admit or deny each pod creation request.

Google Kubernetes Engine

A financial services company runs its critical payment application on a production Google Kubernetes Engine (GKE) cluster. For strict compliance, the company must enforce a policy that only allows container images to be deployed if they have passed two specific quality gates:

1. A successful vulnerability scan from an automated security tool.
2. A manual sign-off from the Quality Assurance (QA) team.

How do you prevent the deployment of images who have not passed these checks?

A. In the application's Deployment manifest, add a postStart lifecycle hook that calls an external script to verify the image's approval status and stops the container if it fails.

B. Use Cloud Source Repositories to host the container images and configure IAM permissions so that only the QA team lead has rights to deploy to the production cluster.

C. Use Binary Authorization to define a policy for the production cluster that attestations for each quality gate before an image can be deployed. Integrate the creation of these attestations into your CI/CD process.

D. Write and deploy a custom validating admission webhook. The webhook's logic will query an external database of approved images to either admit or deny each pod creation request.

O'REILLY

GKE Enterprise (formerly Anthos)

Hybrid and Multicloud Environments

- **Workloads on-premises**
  - Data sovereignty and compliance
  - Low latency and performance needs
  - Already existing investment in infra
- **Workloads on another cloud**
  - Mitigating vendor lock-in
  - Building resilience

Multi-cluster management

- Organizations might deploy multiple clusters to meet technical and business needs
  - Separate production and non-production environments
  - Adhere to regulatory requirements
  - Organize services by tiers, locations, or teams

GKE Enterprise

Advanced version of Google Kubernetes Engine designed to meet the needs of large organizations with complex, large-scale Kubernetes deployments

Makes it easier to implement hybrid and multicloud strategies

GKE Enterprise Fleets

- A way to logically group and normalize Kubernetes resources
  - Manage groups of clusters rather than individual clusters
- Resources in a fleet generally related to one another
  - Resources with large cross-service communication benefit from being part of the same fleet

Benefits of Fleets

- Unified management of clusters
- Consistent operations across clusters
- Enhanced visibility over the entire system

If you want load balancing across multiple clusters configure a Multi-cluster Ingress not a Load Balancer

Load Balancer vs. Multi-cluster Ingress

Load Balancer

- Load balancing a service in a single GKE cluster
- Layer 4 (network load balancer) - does not understand HTTP

Multi-cluster Ingress

- Load balancing a service that runs on multiple clusters
- Layer 7 (HTTP/S load balancer)

Google Kubernetes Engine

A gaming company hosts the backend for its popular real-time multiplayer game on a GKE cluster in europe-west1. Following a successful launch in North America, players there are experiencing high latency (lag), which negatively impacts gameplay.

You need to create a low-latency experience for players in both regions by directing them to the nearest healthy cluster. The game backend is a stateless application.

What is the best way to architect this multi-region deployment on Google Cloud?

A. Create a new GKE cluster in a North American region. In each cluster, expose the game server via a Service of type LoadBalancer. In Cloud DNS, create A records with the same hostname for both public IPs and rely on DNS-based routing.

B. Create a new GKE cluster in a North American region. Register both clusters to a GKE Fleet and configure Multi-Cluster Ingress to deploy a single global load balancer that intelligently routes player traffic to the nearest available regional cluster.

C. Keep the single GKE cluster in europe-west1 but place a Global External HTTP/S Load Balancer in front of it and enable Cloud CDN to cache static game assets closer to North American players.

D. Keep the single GKE cluster in europe-west1 and enable Horizontal Pod Autoscaling to ensure there are enough game server pods to handle the increased player load from both continents.

Google Kubernetes Engine

A gaming company hosts the backend for its popular real-time multiplayer game on a GKE cluster in europe-west1. Following a successful launch in North America, players there are experiencing high latency (lag), which negatively impacts gameplay.

You need to create a low-latency experience for players in both regions by directing them to the nearest healthy cluster. The game backend is a stateless application.

What is the best way to architect this multi-region deployment on Google Cloud?

A. Create a new GKE cluster in a North American region. In each cluster, expose the game server via a Service of type LoadBalancer. In Cloud DNS, create A records with the same hostname for both public IPs and rely on DNS-based routing.

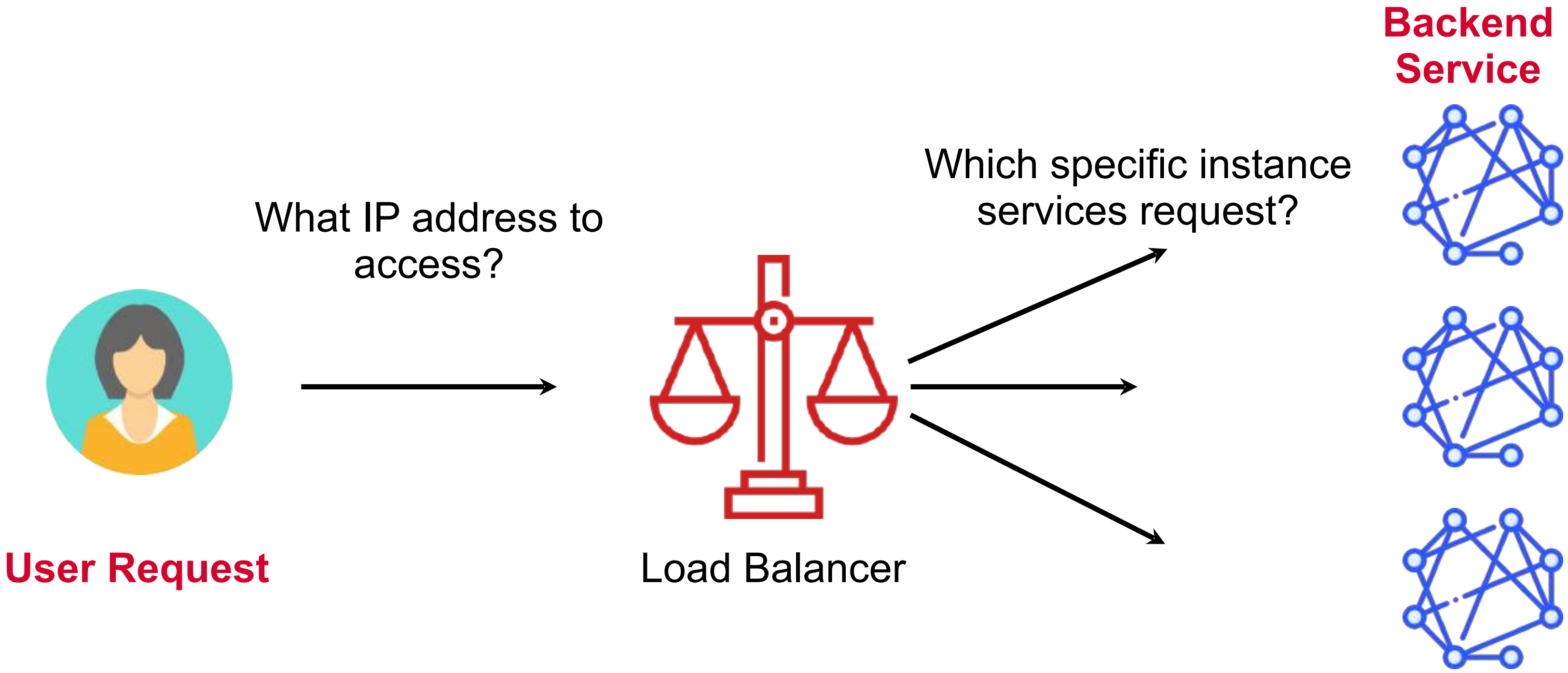
B. Create a new GKE cluster in a North American region. Register both clusters to a GKE Fleet and configure Multi-Cluster Ingress to deploy a single global load balancer that intelligently routes player traffic to the nearest available regional cluster.

C. Keep the single GKE cluster in europe-west1 but place a Global External HTTP/S Load Balancer in front of it and enable Cloud CDN to cache static game assets closer to North American players.

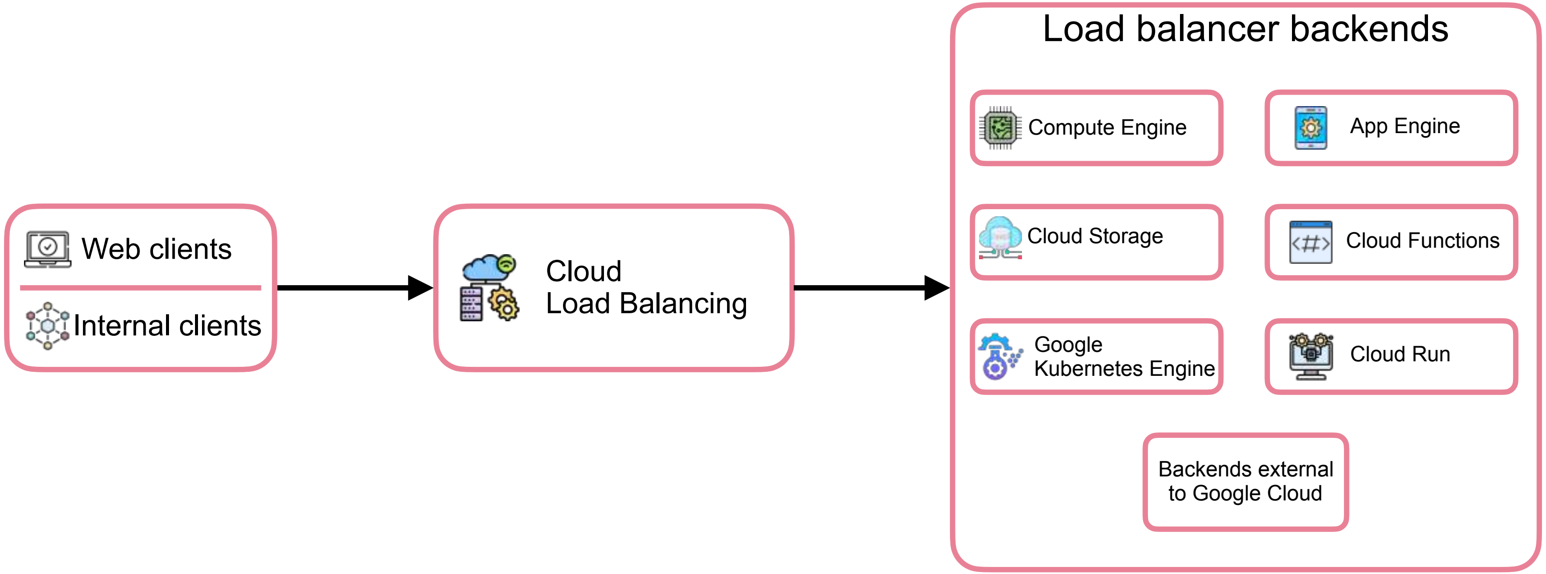
D. Keep the single GKE cluster in europe-west1 and enable Horizontal Pod Autoscaling to ensure there are enough game server pods to handle the increased player load from both continents.

Load Balancing

Load Balancers



Load Balancers Used with Multiple Backends



Load Balancers

- Complex service
- Many moving parts
- Basic idea
  - Stable front-end IP
  - Forwarding rules to funnel traffic
  - Connect to backend service
  - Distribute load intelligently
  - Health checks to avoid unhealthy instances

Load balancers **distribute** traffic to resources close to users and meet **high-availability** requirements

Load Balancers on the GCP

- Fully managed, software-defined, redundant and highly available
- Supports > 1 million queries per second with high performance and low latency
- Autoscaling to meet increased traffic

Load Balancing Categorization

Global

Use when your users and instances are globally distributed, load balanced resources lie across multiple regions

Regional

Use when instances and users are concentrated in one region

Load Balancing Categorization

External

Distributes traffic from the internet to the Google Cloud

Internal

Distributes traffic only within the Google Cloud, all clients are inside of the Google Cloud

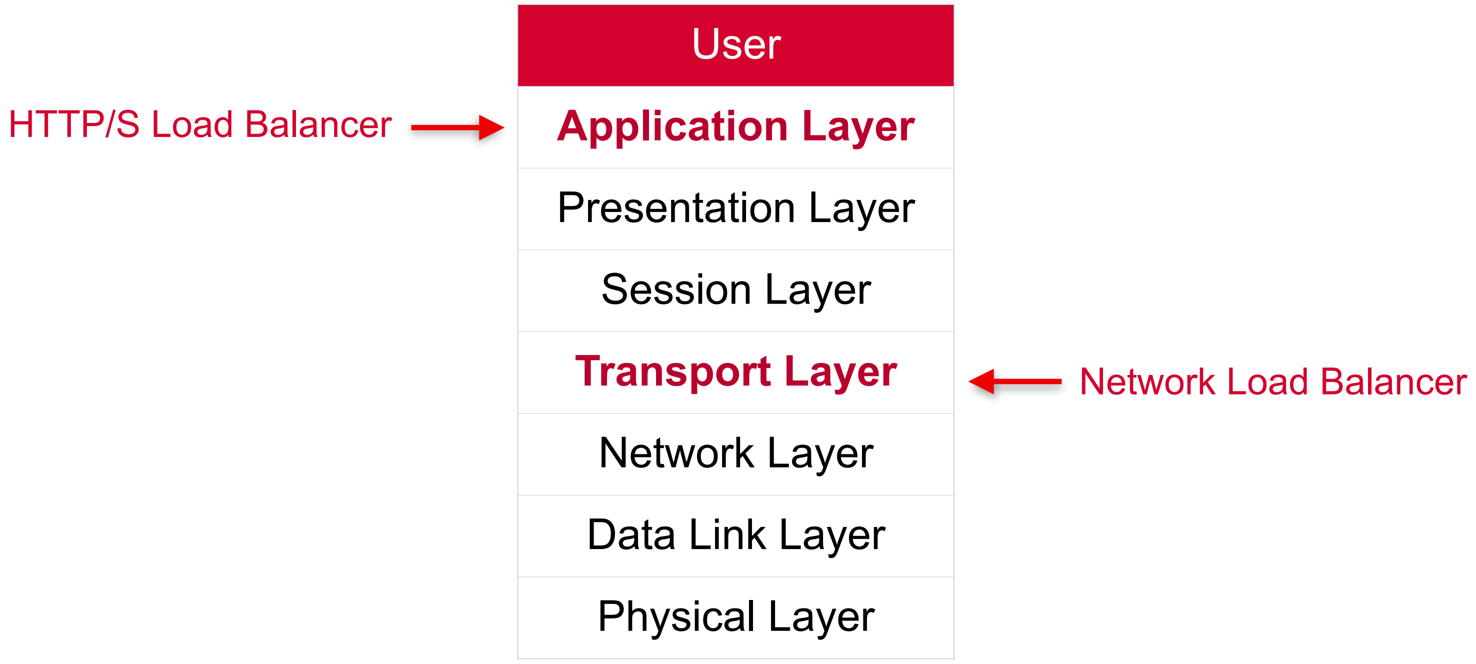
7 Layer OSI Network Stack

Routing decisions based on attributes of the request i.e. HTTP headers and the URL

User
Application Layer
Presentation Layer
Session Layer
Transport Layer
Network Layer
Data Link Layer
Physical Layer

Direct traffic based on data from network and transport layer protocols such as TCP, UDP, ESP, GRE, ICMP, and ICMPv6

7 Layer OSI Network Stack



Two Types of Load Balancers

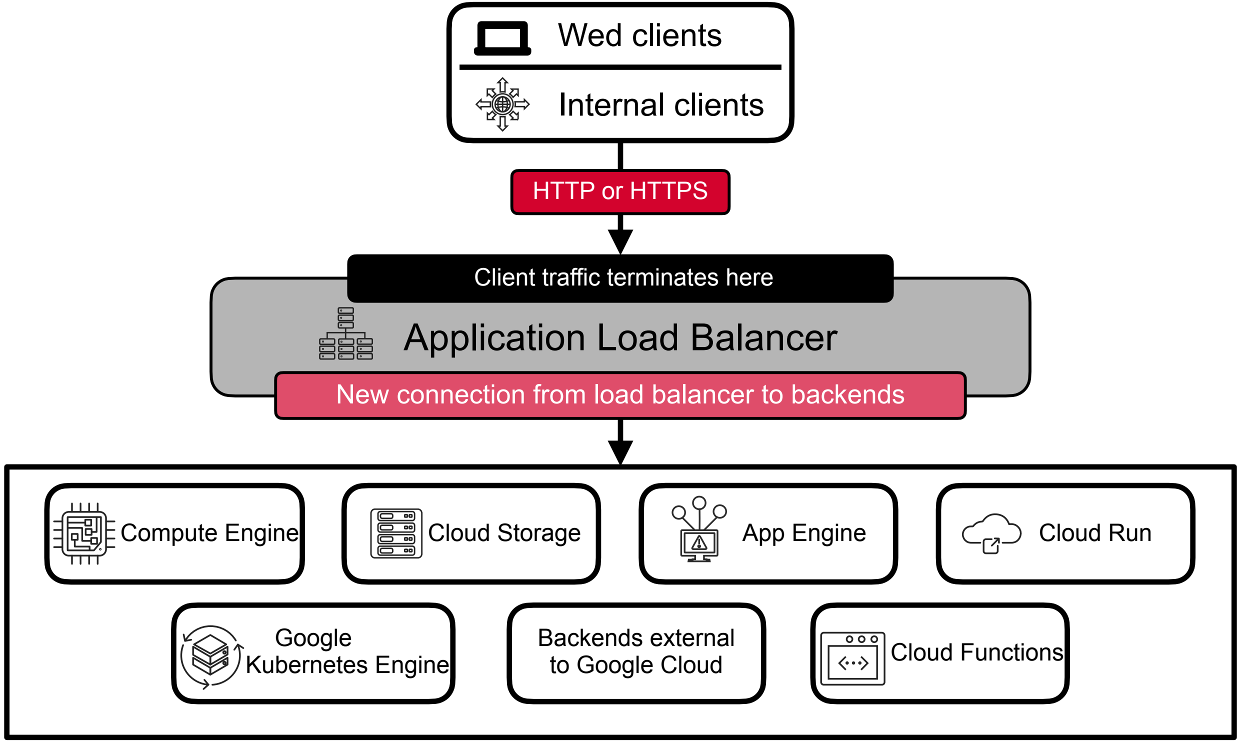
Application Load Balancers

Network Load Balancers

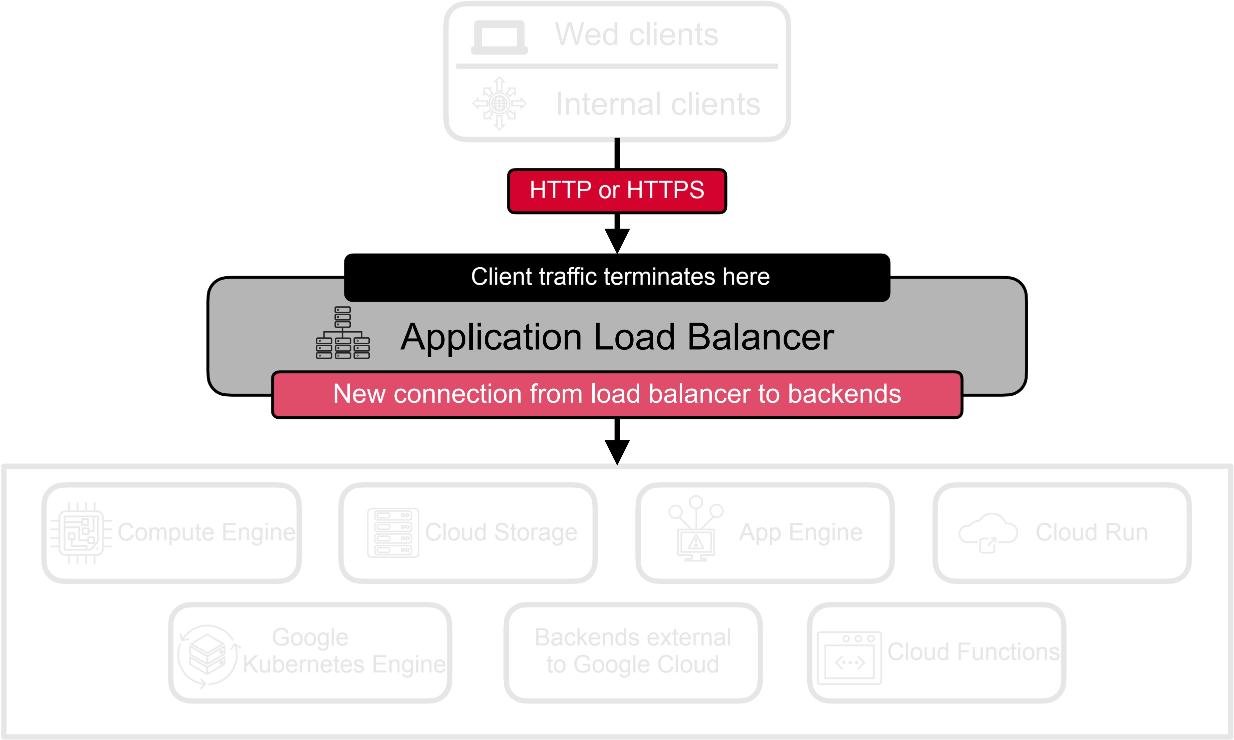
Application Load Balancers

- **Proxy-based** layer 7 load balancers
- Allow you to scale your services behind a single IP
- Distributes HTTP and HTTPS traffic to Google backends and external backends
  - Compute Engine, GKE, Cloud Run

Application Load Balancers



Proxy Load Balancing



URL-based Routing

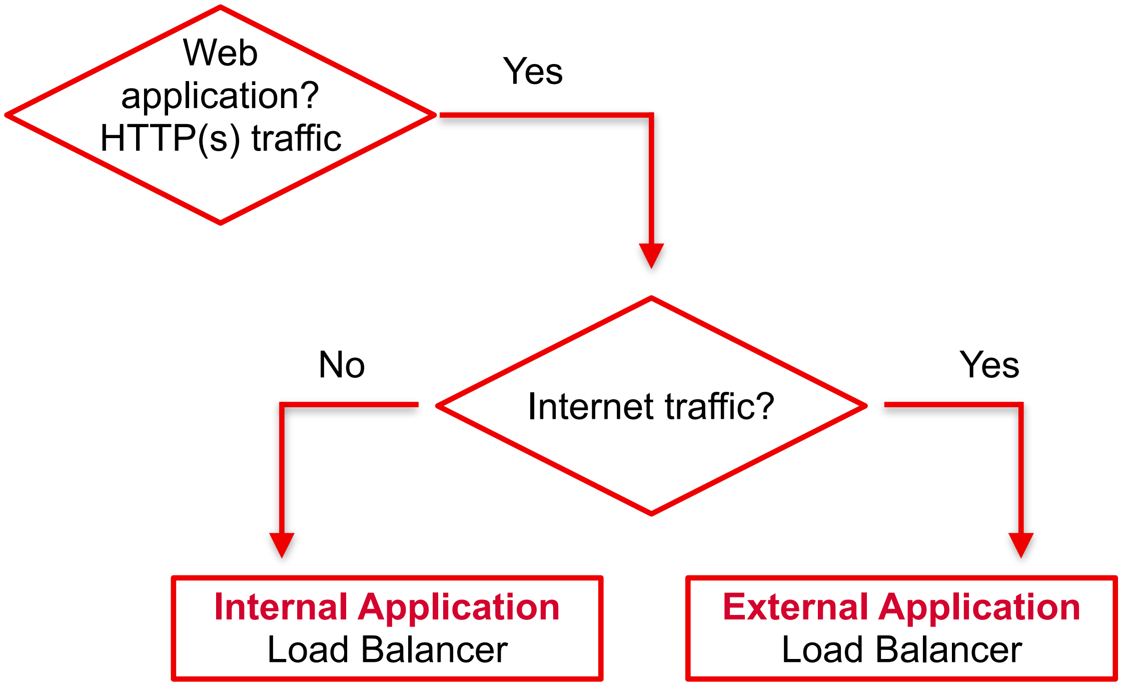
The **HTTP(S) Load Balancer** can split traffic based on content using **URL-based routing rules**

The load balancer inspects the incoming request's URL path or hostname and direct the traffic to different backend services or instances based on predefined conditions

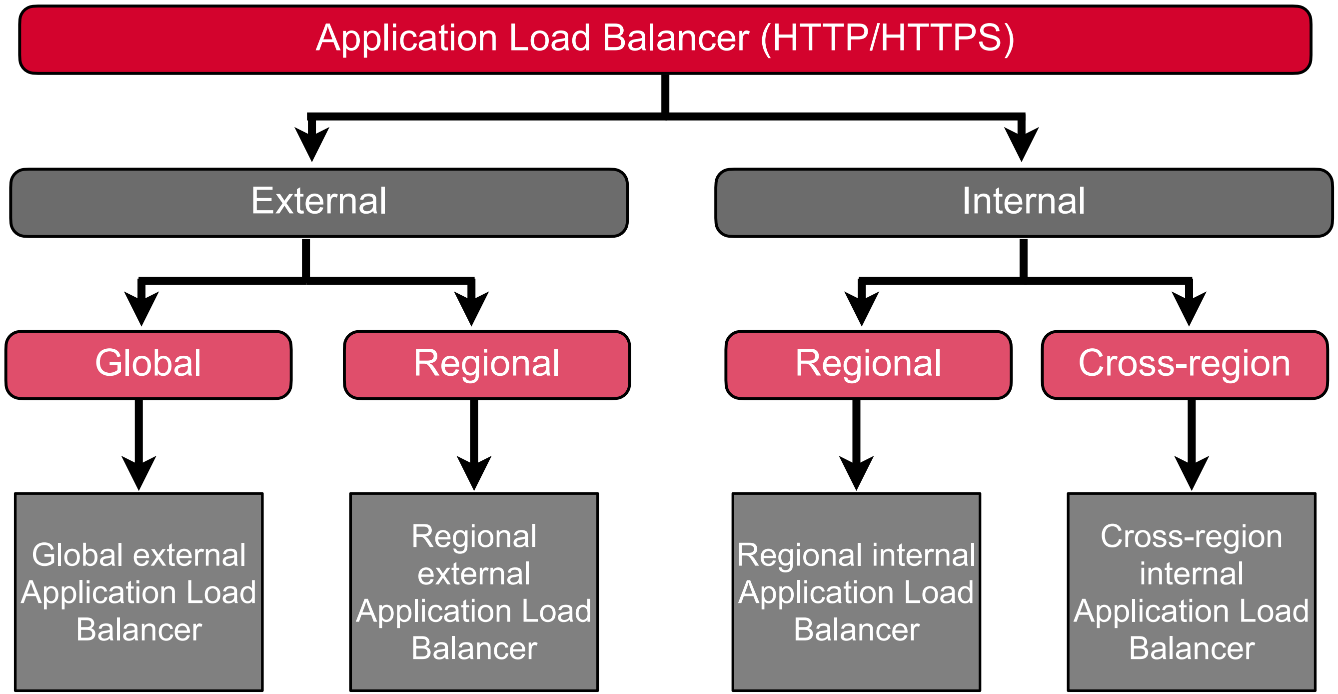
URL-based Routing

- Path-based routing
  - example.com/images/* routed to Backend Service 1
  - example.com/videos/* routed to Backend Service 2
- Host-based routing
  - app.example.com routed to Backend Service 3
  - blog.example.com routed to Backend Service 4

Choosing Load Balancers



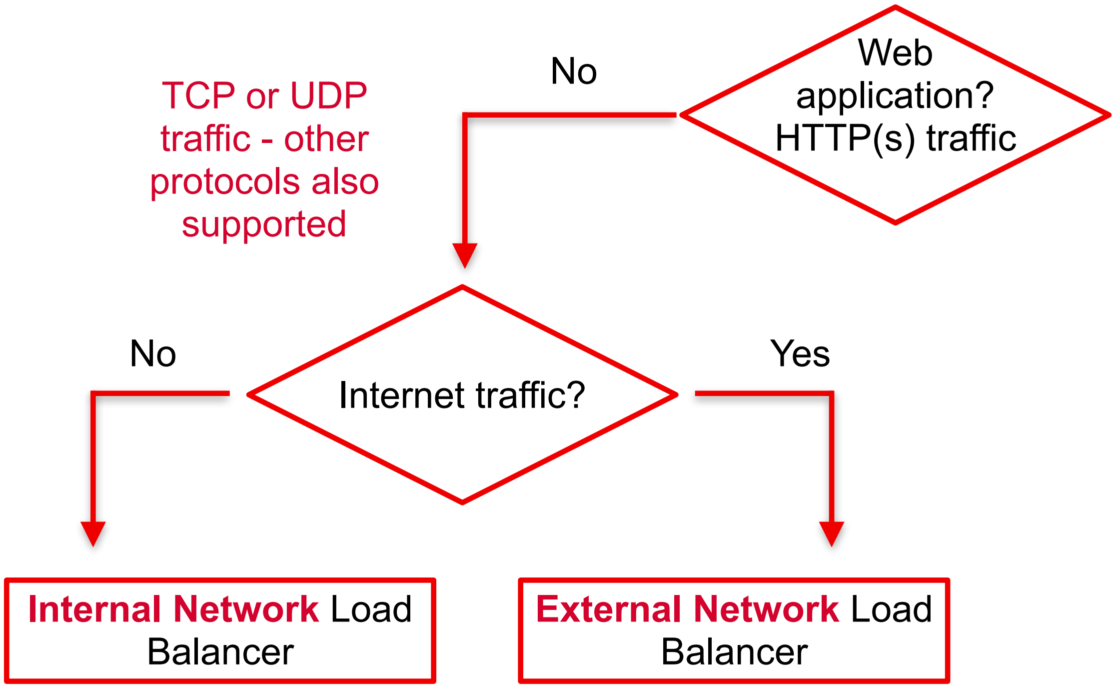
Application Load Balancers



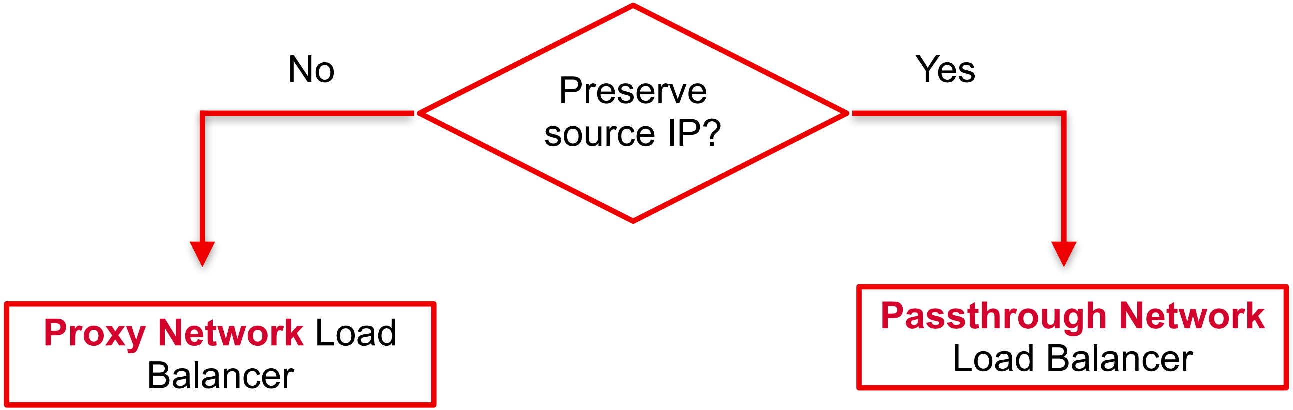
Network Load Balancers

- Layer 4 load balancers
- Handle TCP, UDP, or other IP protocol traffic
- Can be of two types
    - Proxy
    - Passthrough

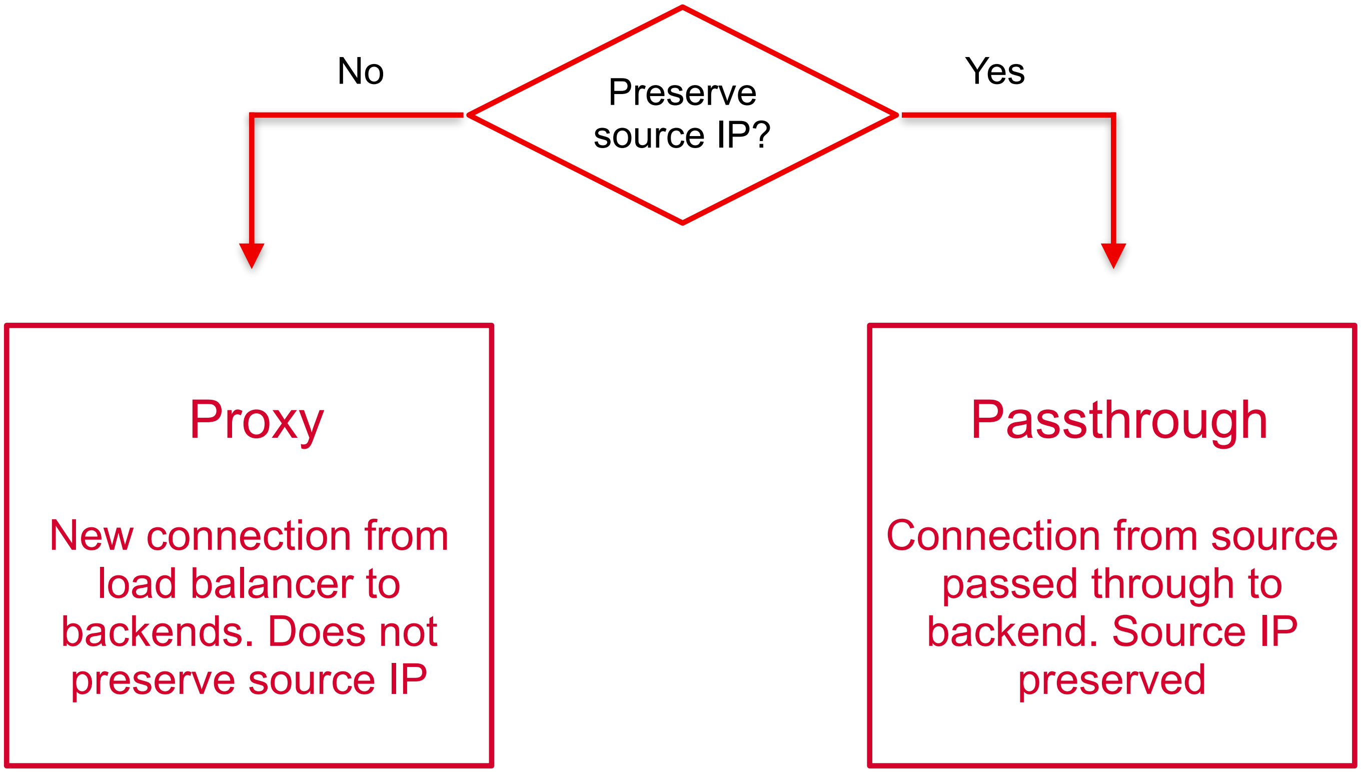
Choosing Load Balancers



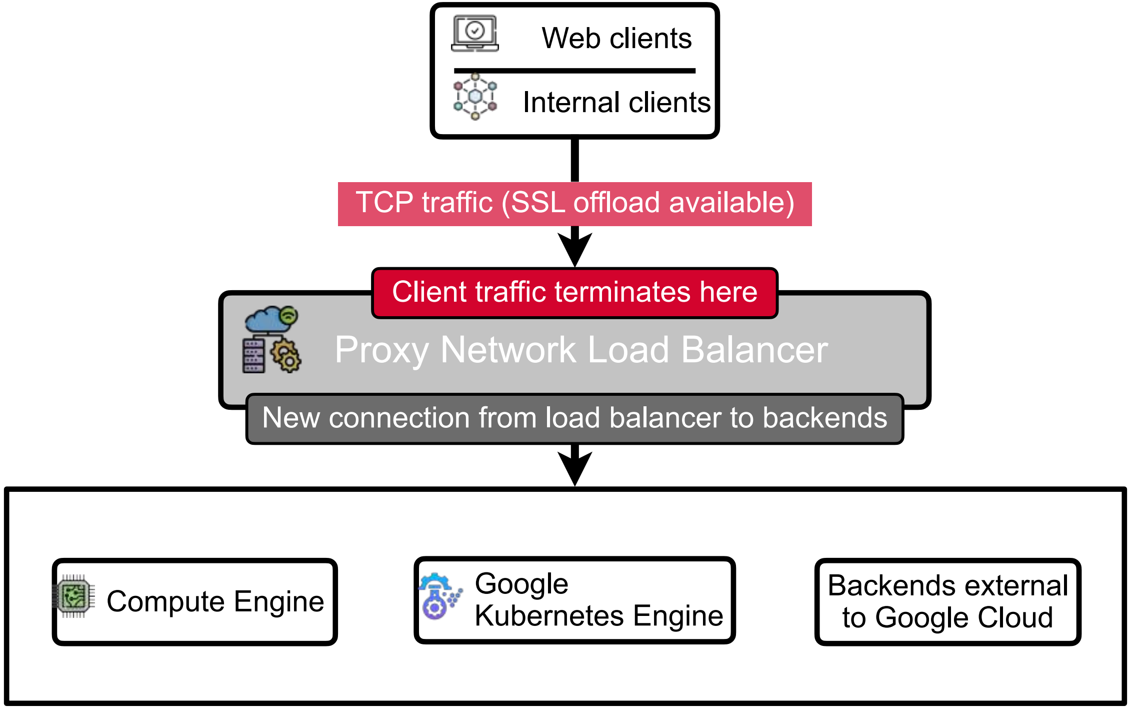
Choosing Network Load Balancers



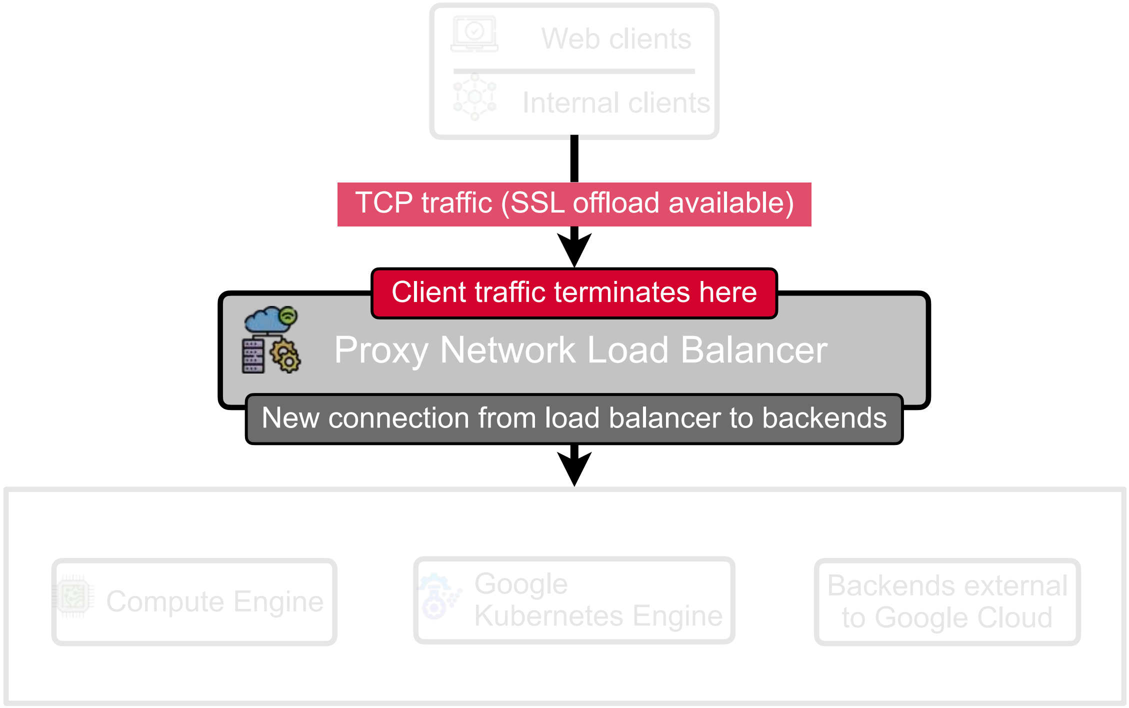
Choosing Network Load Balancers



Network Proxy Load Balancers

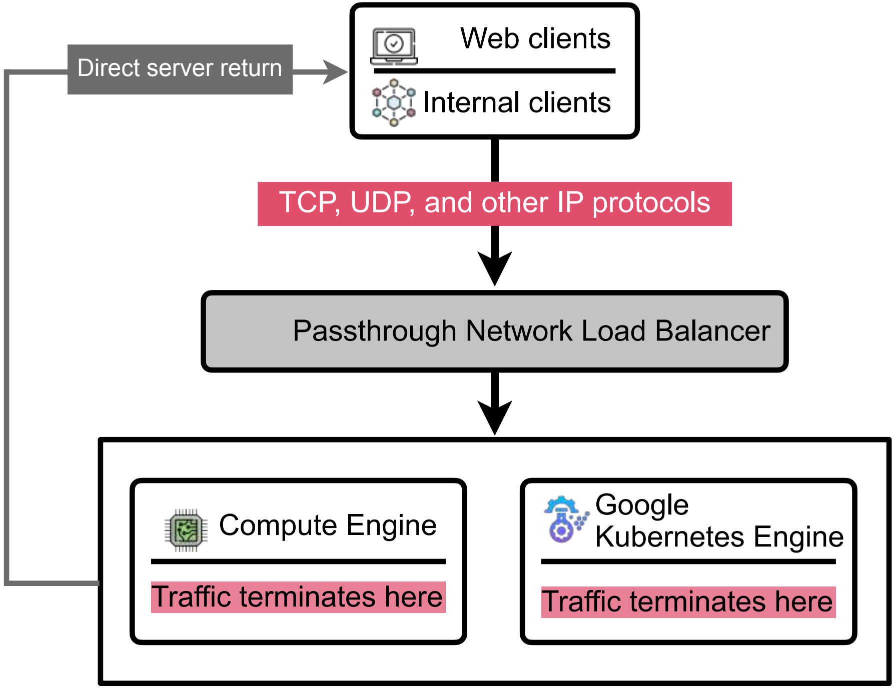


Network Proxy Load Balancers - SSL Offload

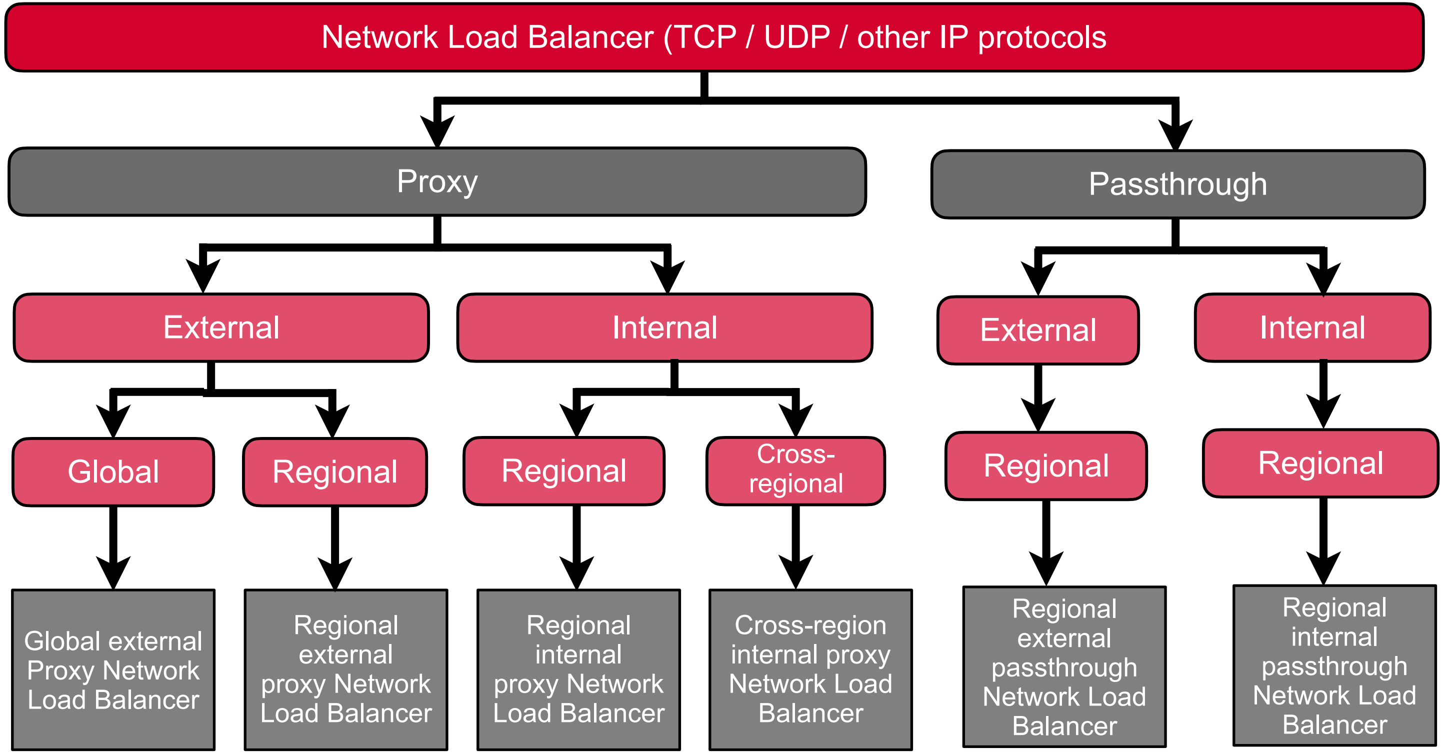


The load balancer terminates the secure SSL/TLS connection, decrypts the data, and forwards the plain HTTP traffic to the backend servers.

Network Passthrough Load Balancers



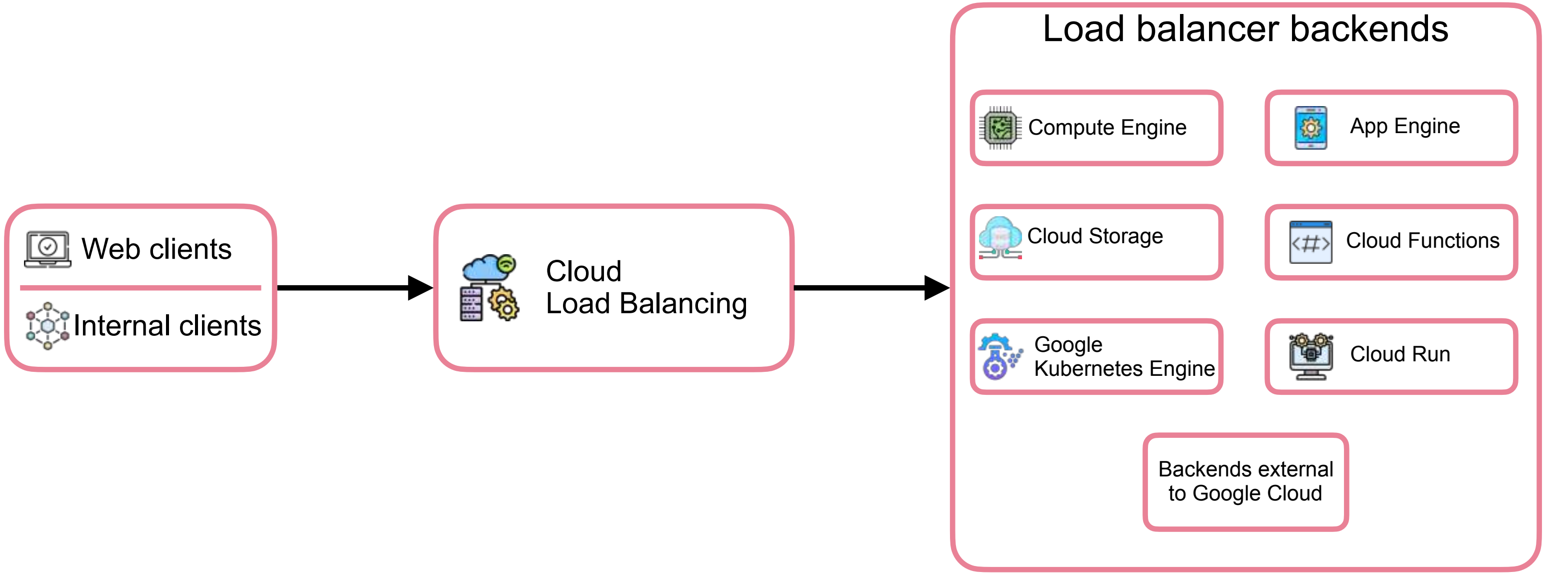
Network Load Balancers



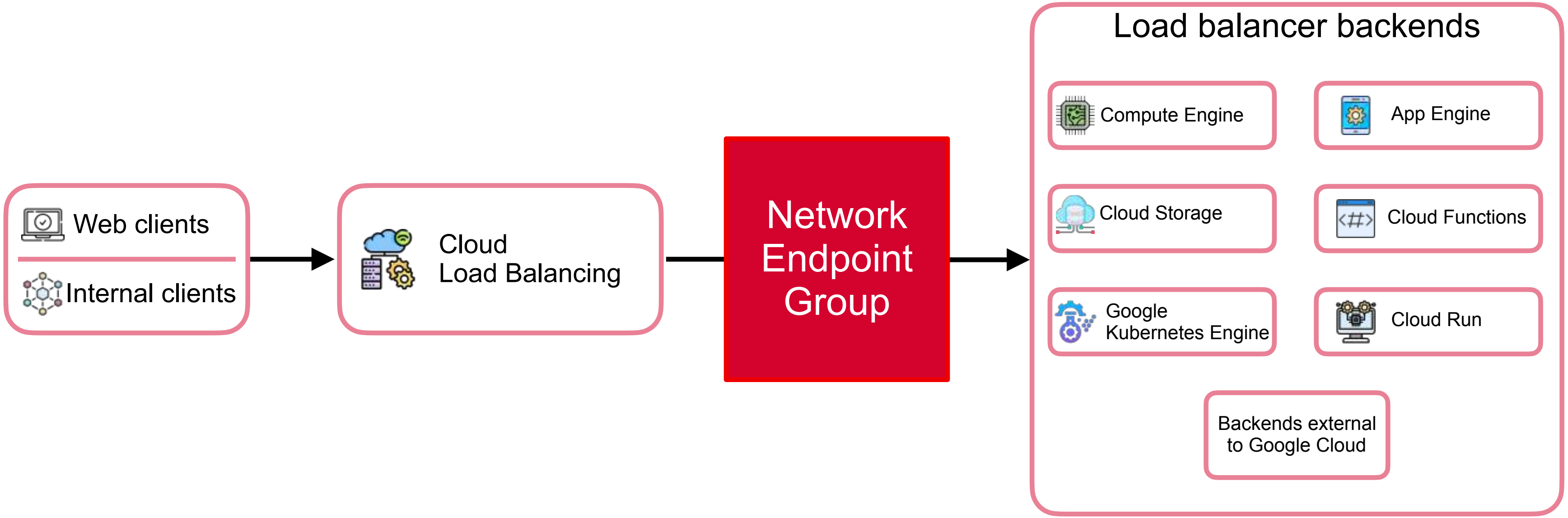
Network Endpoint Groups

Logical grouping of network endpoints, where an endpoint is an `IP_address:Port` combination. They provide a flexible and scalable way to group backend endpoints for your load balancers.

Load Balancers Used with Multiple Backends



Decouple Load Balancer from Backend



Types of NEGs

Zonal

Serverless

Internet

Types of NEGs

Zonal

Serverless

Internet

A group of endpoints that are located within a single zone. Used for container-native load balancing on zonal GKE clusters

Types of NEGs

Zonal

Serverless

Internet

Points to a single serverless Google Cloud service, such as Cloud Run, App Engine, or Cloud Functions.

Types of NEGs

Zonal

Serverless

Internet

A group of endpoints that live **outside of Google Cloud**, on the public internet. You can specify endpoints by either their IP address or Fully Qualified Domain Name (FQDN).

When you configure the different types of Cloud Load Balancers for your services the right type of NEG is automatically created and used

Load Balancing

Your team needs to deploy a legacy financial application on Google Cloud. The application communicates over TCP and relies on direct access to a local filesystem to maintain data integrity. It cannot be scaled horizontally because it does not have synchronization and accessing the file system concurrently causes problems that cannot be resolved easily. Although brief downtime is acceptable during failover, the application must remain highly available to support continuous business operations. How should you architect this deployment?

A. Deploy the application on a managed instance group across multiple zones, use Cloud Filestore for shared storage, and place an HTTP(S) load balancer in front to distribute traffic.

B. Deploy the application on a managed instance group across multiple zones, use Cloud Filestore for shared storage, and use a network load balancer to balance traffic.

C. Create an unmanaged instance group with one active and one standby VM in different zones, attach a regional persistent disk for storage, and place an HTTP(S) load balancer in front to route requests.

D. Create an unmanaged instance group with an active VM and a standby VM in separate zones, use a regional persistent disk for data storage, and place a network load balancer in front to route client connections.

Load Balancing

Your team needs to deploy a legacy financial application on Google Cloud. The application communicates over TCP and relies on direct access to a local filesystem to maintain data integrity. It cannot be scaled horizontally because it does not have synchronization and accessing the file system concurrently causes problems that cannot be resolved easily. Although brief downtime is acceptable during failover, the application must remain highly available to support continuous business operations. How should you architect this deployment?

A. Deploy the application on a managed instance group across multiple zones, use Cloud Filestore for shared storage, and place an HTTP(S) load balancer in front to distribute traffic.

B. Deploy the application on a managed instance group across multiple zones, use Cloud Filestore for shared storage, and use a network load balancer to balance traffic.

C. Create an unmanaged instance group with one active and one standby VM in different zones, attach a regional persistent disk for storage, and place an HTTP(S) load balancer in front to route requests.

D. Create an unmanaged instance group with an active VM and a standby VM in separate zones, use a regional persistent disk for data storage, and place a network load balancer in front to route client connections.

Load Balancing

A media company is building a global video streaming platform on Google Cloud using dozens of microservices. They require a CI/CD pipeline for frequent, independent updates using immutable artifacts. The architecture must provide low-latency access to users in both North America and Asia via a single global entry point, while keeping core backend services private from the internet. Which set of Google Cloud services is most suitable for this architecture?

A. Artifact Registry to store container images, Google Kubernetes Engine (GKE) clusters in an Asian and North American region, and a Global External HTTP/S Load Balancer.

B. App Engine to host the services in multiple regions, Cloud Spanner for a global database, and a regional Internal TCP/UDP Load Balancer.

C. Cloud Storage to store application binaries, Managed Instance Groups (MIGs) in each region, and a Regional External Network Load Balancer for each region.

D. Cloud Functions for the microservice logic and a Global External HTTP/S Load Balancer configured with Serverless Network Endpoint Groups (NEGs).

Load Balancing

A media company is building a global video streaming platform on Google Cloud using dozens of microservices. They require a CI/CD pipeline for frequent, independent updates using immutable artifacts. The architecture must provide low-latency access to users in both North America and Asia via a single global entry point, while keeping core backend services private from the internet. Which set of Google Cloud services is most suitable for this architecture?

A. **Artifact Registry to store container images, Google Kubernetes Engine (GKE) clusters in an Asian and North American region, and a Global External HTTP/S Load Balancer.**

B. App Engine to host the services in multiple regions, Cloud Spanner for a global database, and a regional Internal TCP/UDP Load Balancer.

C. Cloud Storage to store application binaries, Managed Instance Groups (MIGs) in each region, and a Regional External Network Load Balancer for each region.

D. Cloud Functions for the microservice logic and a Global External HTTP/S Load Balancer configured with Serverless Network Endpoint Groups (NEGs).

Load Balancing

An international e-commerce company has deployed its containerized shopping cart API on Cloud Run in two regions: europe-west1 and australia-southeast1. They need to provide a single global endpoint (api.shopping.com) for their frontend applications that automatically routes customers to the closest healthy region, ensuring a fast and resilient shopping experience.

What is the recommended Google Cloud native approach to achieve this?

A. Create a Regional External HTTP/S Load Balancer in both regions. Use Cloud DNS to create weighted A records to distribute traffic between the two load balancer IPs.

B. For each regional Cloud Run service, create a Serverless Network Endpoint Group (NEG). Create a single Global External HTTP/S Load Balancer and configure its backend service to use these two Serverless NEGs.

C. Use API Gateway in front of each Cloud Run service. Configure the two API Gateway instances under a single custom domain in Cloud DNS to handle routing.

D. Create a Global External HTTP/S Load Balancer. Manually add the default .run.app URLs of the two Cloud Run services as Internet Network Endpoint Group backends to the load balancer.

Load Balancing

An international e-commerce company has deployed its containerized shopping cart API on Cloud Run in two regions: europe-west1 and australia-southeast1. They need to provide a single global endpoint (api.shopping.com) for their frontend applications that automatically routes customers to the closest healthy region, ensuring a fast and resilient shopping experience.

What is the recommended Google Cloud native approach to achieve this?

A. Create a Regional External HTTP/S Load Balancer in both regions. Use Cloud DNS to create weighted A records to distribute traffic between the two load balancer IPs.

B. For each regional Cloud Run service, create a Serverless Network Endpoint Group (NEG). Create a single Global External HTTP/S Load Balancer and configure its backend service to use these two Serverless NEGs.

C. Use API Gateway in front of each Cloud Run service. Configure the two API Gateway instances under a single custom domain in Cloud DNS to handle routing.

D. Create a Global External HTTP/S Load Balancer. Manually add the default .run.app URLs of the two Cloud Run services as Internet Network Endpoint Group backends to the load balancer.