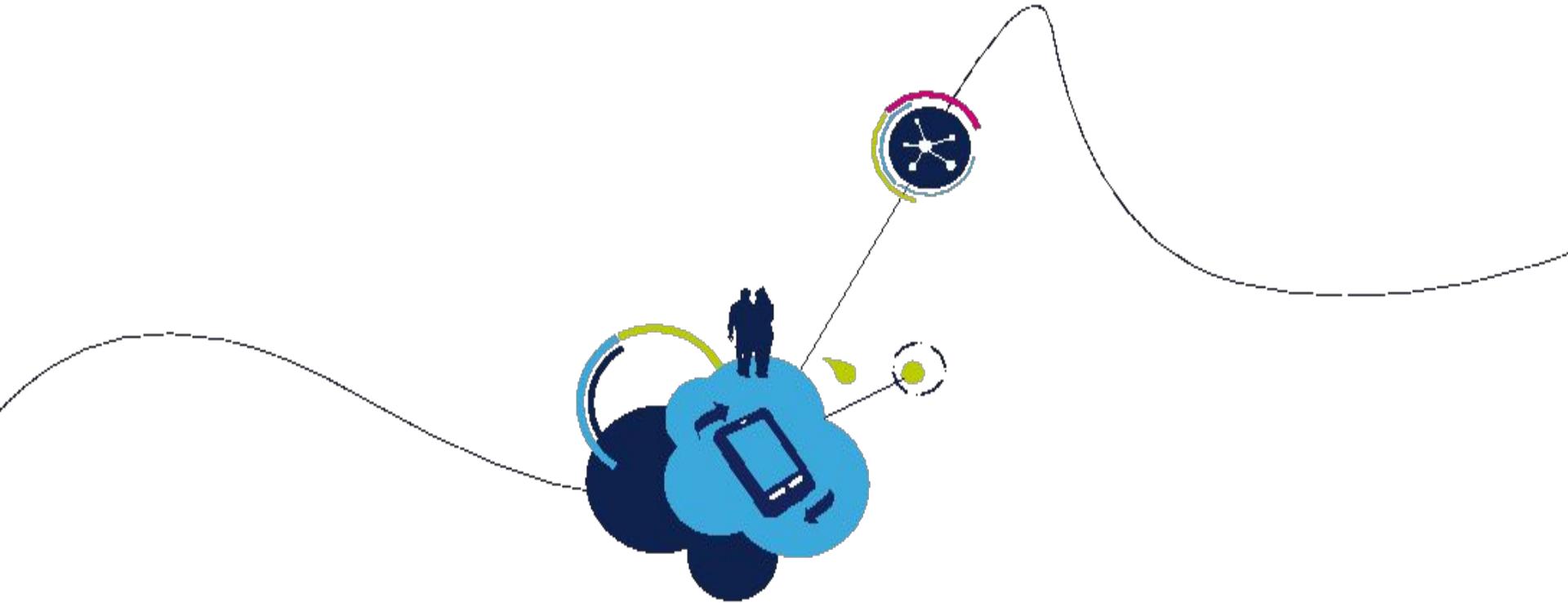


Google Cloud Platform Fundamentals

Agenda

- 1- Introduction to Google Cloud Platform
- 2- Compute
- 3- Storage
- 4- Networking
- 5- Building for “Cloud 3.0”
- 6- Machine Learning

- 7- Virtual Private Cloud (VPC) Network
- 8- Compute Engine
- 9- Compute Engine
- 10- Important VPC capabilities
- 11- Google Cloud Platform resource hierarchy
- 12- Identity and Access Management (IAM)
- 13- Interacting with Google Cloud Platform



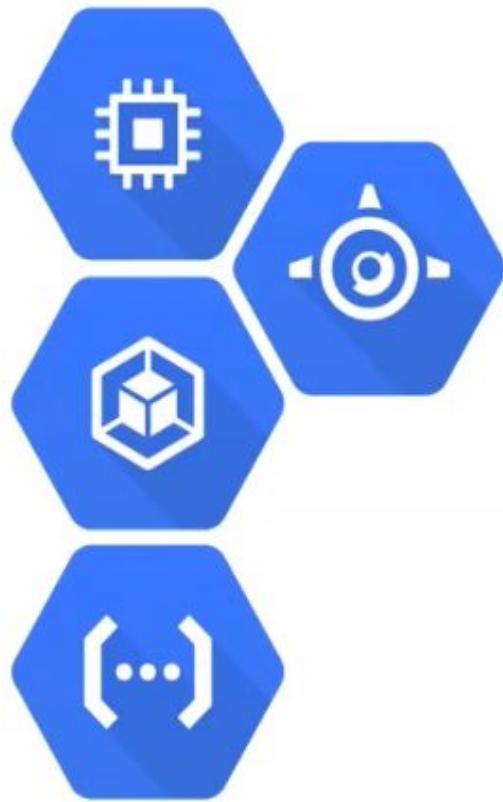
Introduction to Google Cloud Platform

Google Cloud Platform



Public cloud service provider
Started in 2008
Leverages Google global infrastructure

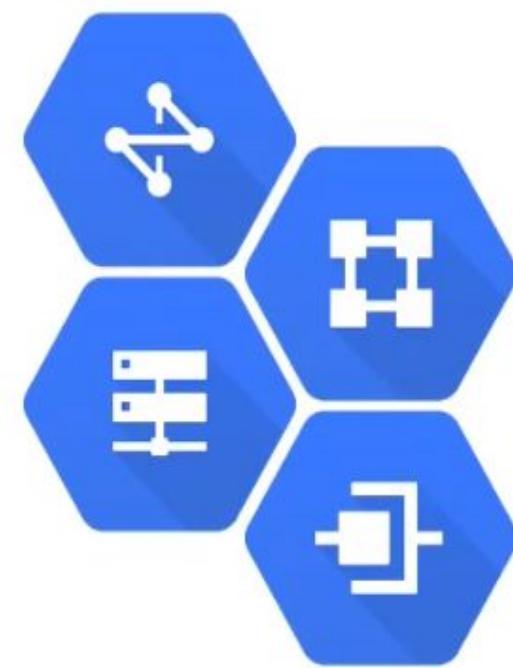
Processing



Storage



Networking



Building Out the Stack

Development Tools

Management

Big Data

Machine Learning

Processing

Storage

Networking

Global Presence



Calendars | Google Cloud Platform X Howard

Secure | https://cloud.google.com/pricing/calculators

Google Cloud Platform

Why Google Products Solutions Launcher Pricing Customers Documentation Support Partners TRY IT FREE CONTACT SALES

Pricing

Overview Principles Innovation Price Leader Price List Calculators

[Calculators](#)

CALCULATE YOUR CLOUD SAVINGS

Get a quick estimate of what your usage will cost on Google Cloud Platform and compare prices with AWS and other public clouds.

A graphic illustration on the right side of the page features a dark blue calculator, a line graph with three data points, a green pie chart with a dark blue slice, and an orange pencil. A yellow speech bubble contains a black percentage sign (%).

- Sustained use discounts
- Committed use discounts
- Preemptible VMs
- Flexible machine configurations

Mar 1 – 31, 2017

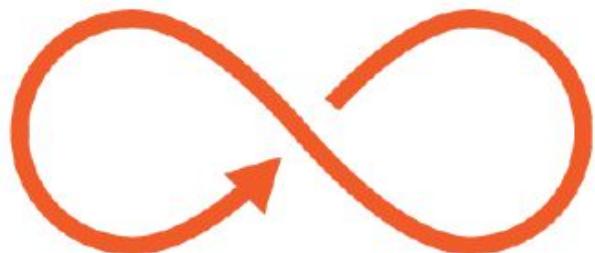
Documents (1)

Monthly Invoice (1)

[Monthly Invoice ...201703](#)

Date	Description
Mar 1 – 31, 2017	Compute Engine Sustained Usage Discount (Source:Project Phoenix [project-phoenix-prod])
Mar 1 – 31, 2017	Compute Engine Standard Intel N1 1 vCPU running in Americas: 743 Hours (Source:Project Phoenix [project-phoenix-prod])
Mar 1 – 31, 2017	Compute Engine Network Load Balancing: Forwarding Rule Minimum Service Charge Americas: 743 Hours (Source:Project Phoenix [project-phoenix-prod])
Mar 1 – 31, 2017	Compute Engine Storage PD Capacity: 99.866 Gibibyte-months (Source:Project Phoenix [project-phoenix-prod])
Mar 28, 2017	Automatic payment: Visa ****2165
Mar 1 – 22, 2017	Cloud Storage Multi-Regional Storage US: 0.204 Gibibyte-months (Source:Project Phoenix [project-phoenix-prod])
Mar 1 – 22, 2017	Compute Engine Network Inter Zone Egress: 1.507 Gibibytes (Source:Project Phoenix [project-phoenix-prod])
Mar 1 – 15, 2017	Compute Engine Network Internet Egress from Americas to China: 0.023 Gibibytes (Source:Project Phoenix [project-phoenix-prod])
Mar 1, 2017	Sales tax (on \$21.18)
Mar 1, 2017	Sales tax (on \$108.76)

Sustained Use Discounts



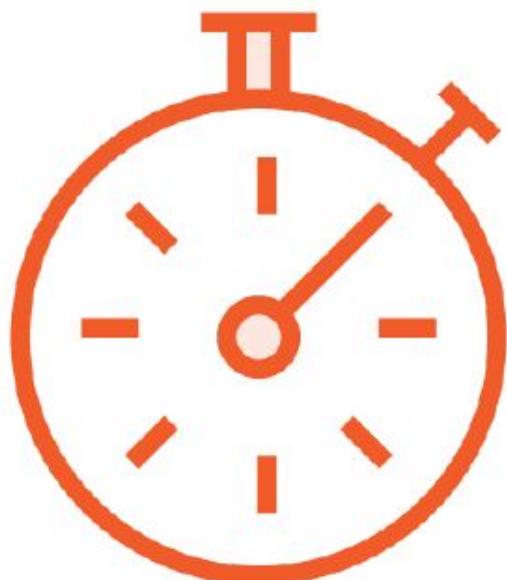
For long-running compute workloads
Given automatically
Savings up to 30%

Committed Use Discounts



For well-understood workloads
Pre-purchased computing resources for a period of time
Savings up to 70%

Preemptible Virtual Machines

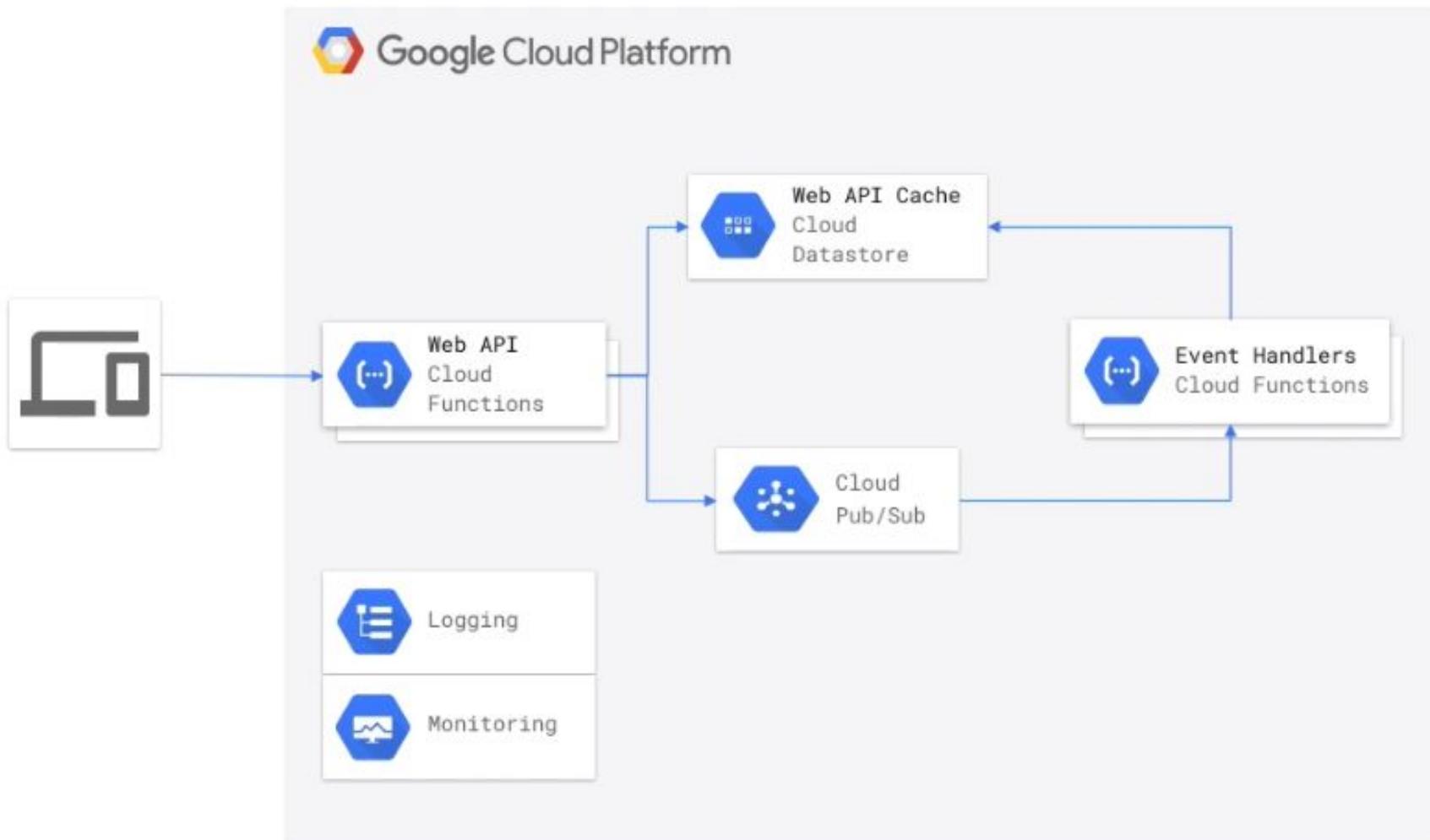


Useful for time-insensitive, batch workloads
Result of excess compute capacity
Can be terminated by the platform when resources are needed

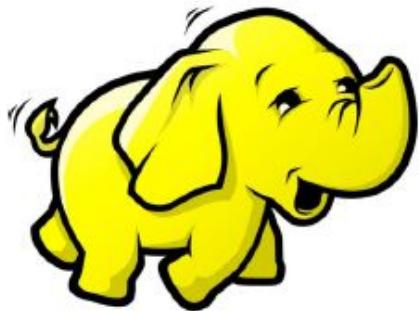
Flexible Machine Configurations



Enables tailoring of machine to workload
Includes number and type of CPUs, GPUs,
and memory



Open Source



Google Map
Reduce



Google “Borg”



Kubernetes

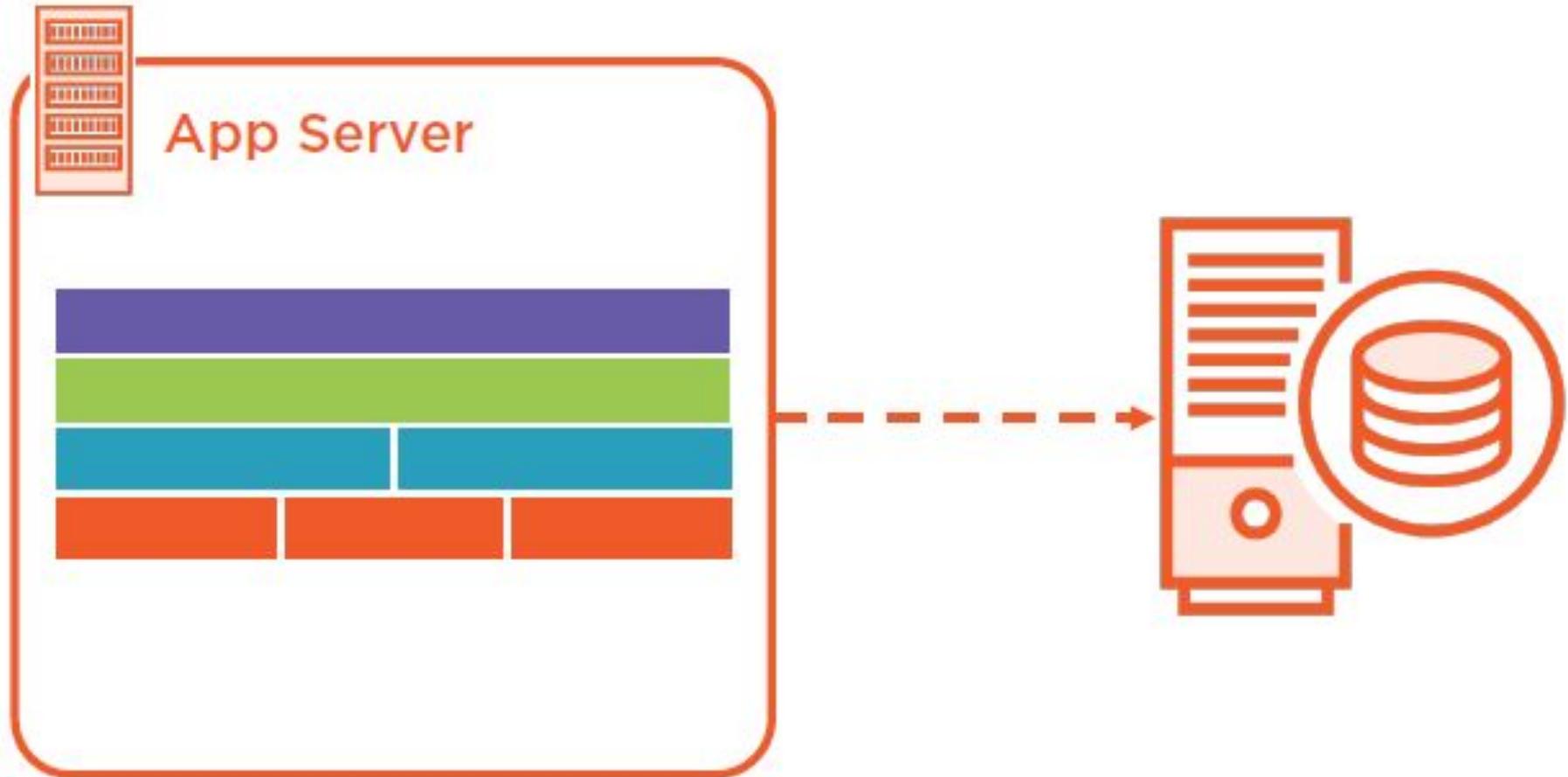


Tensorflow

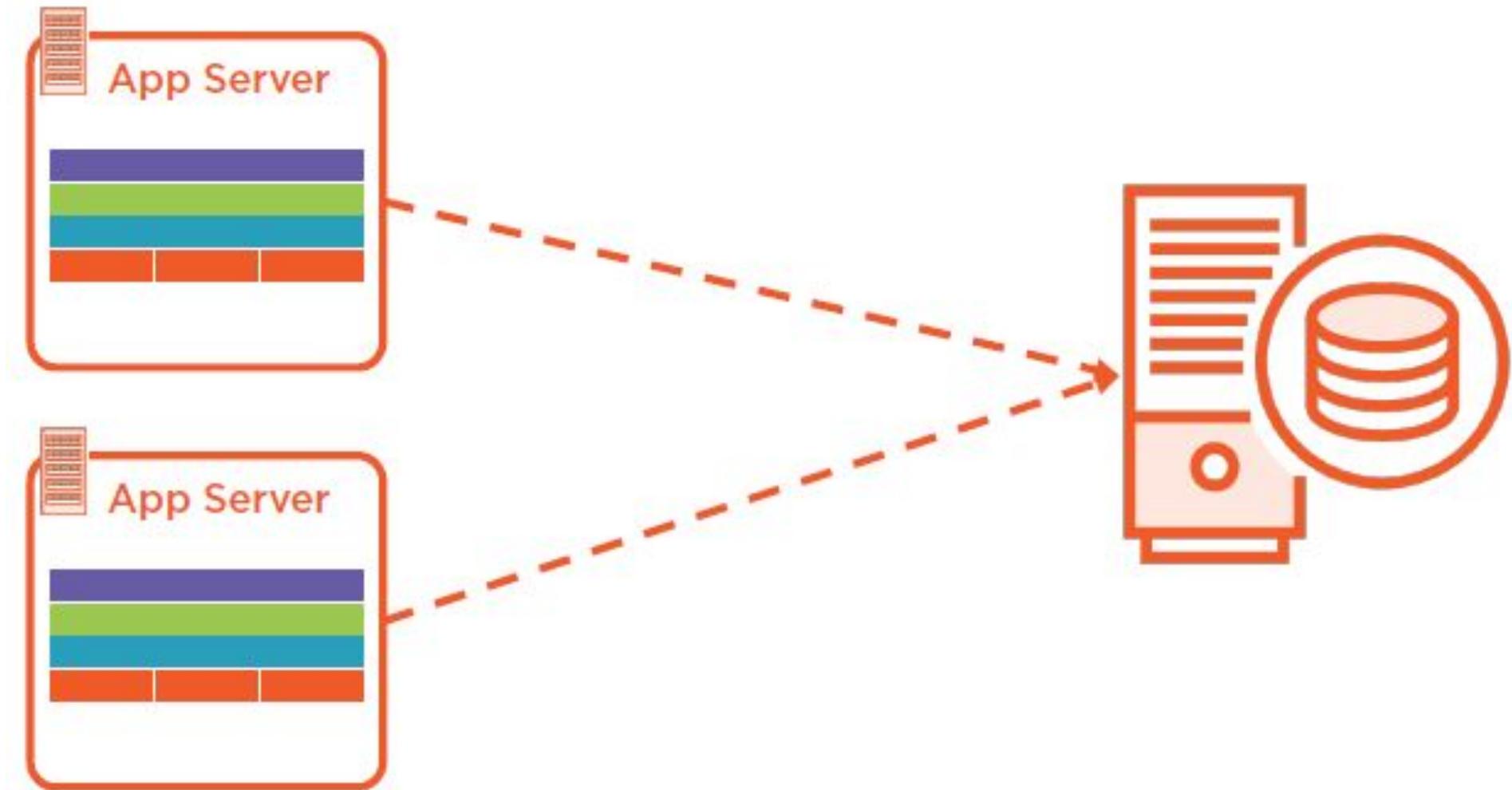
Compute services



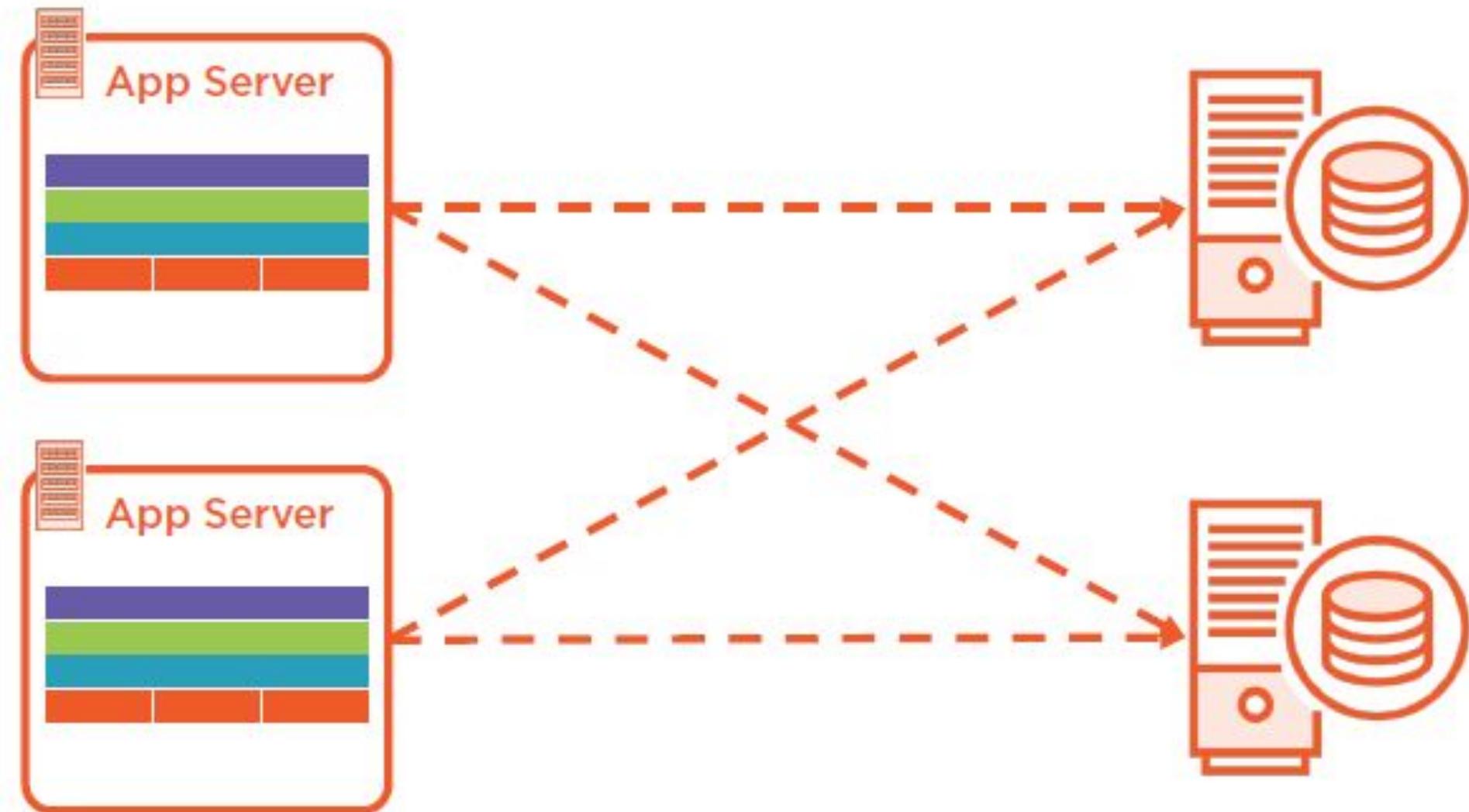
From IaaS to PaaS



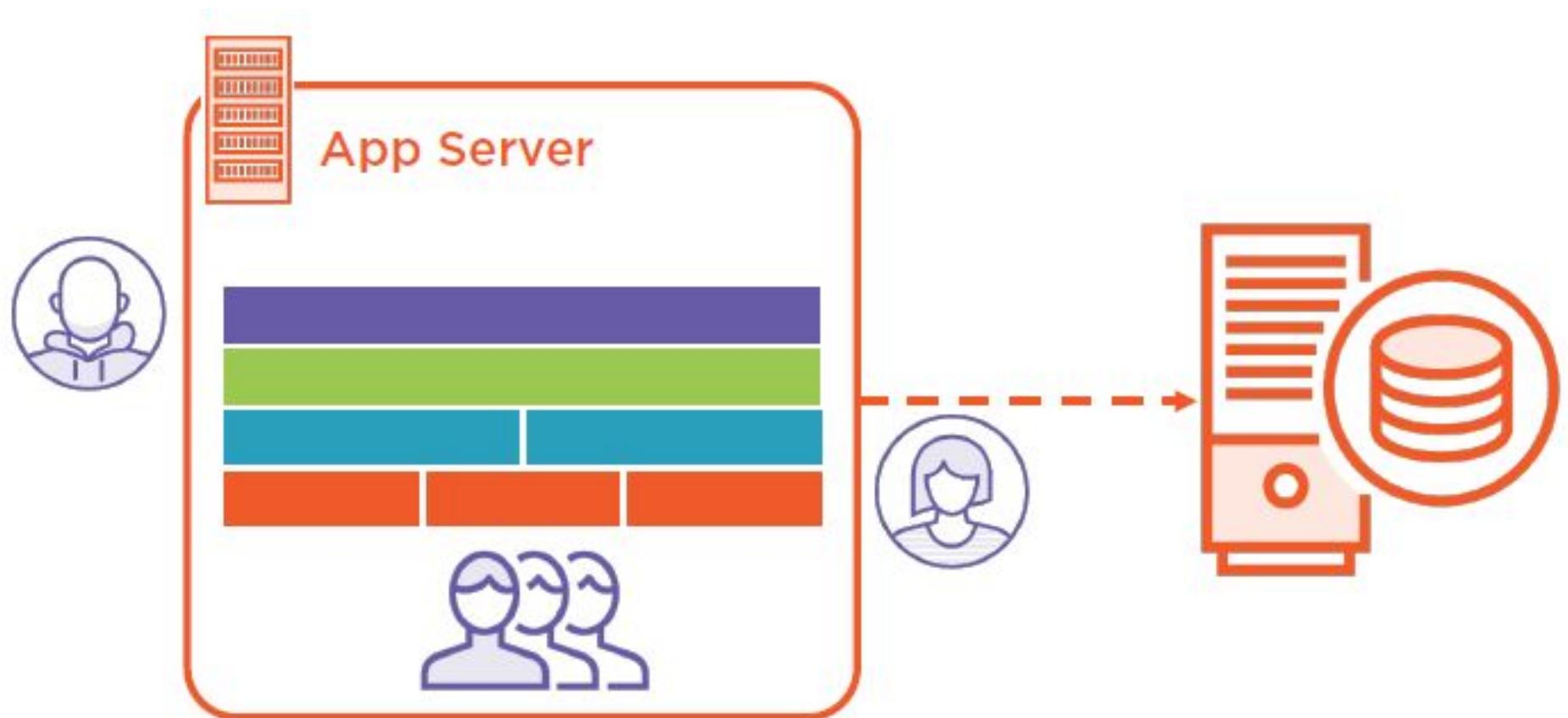
From IaaS to PaaS

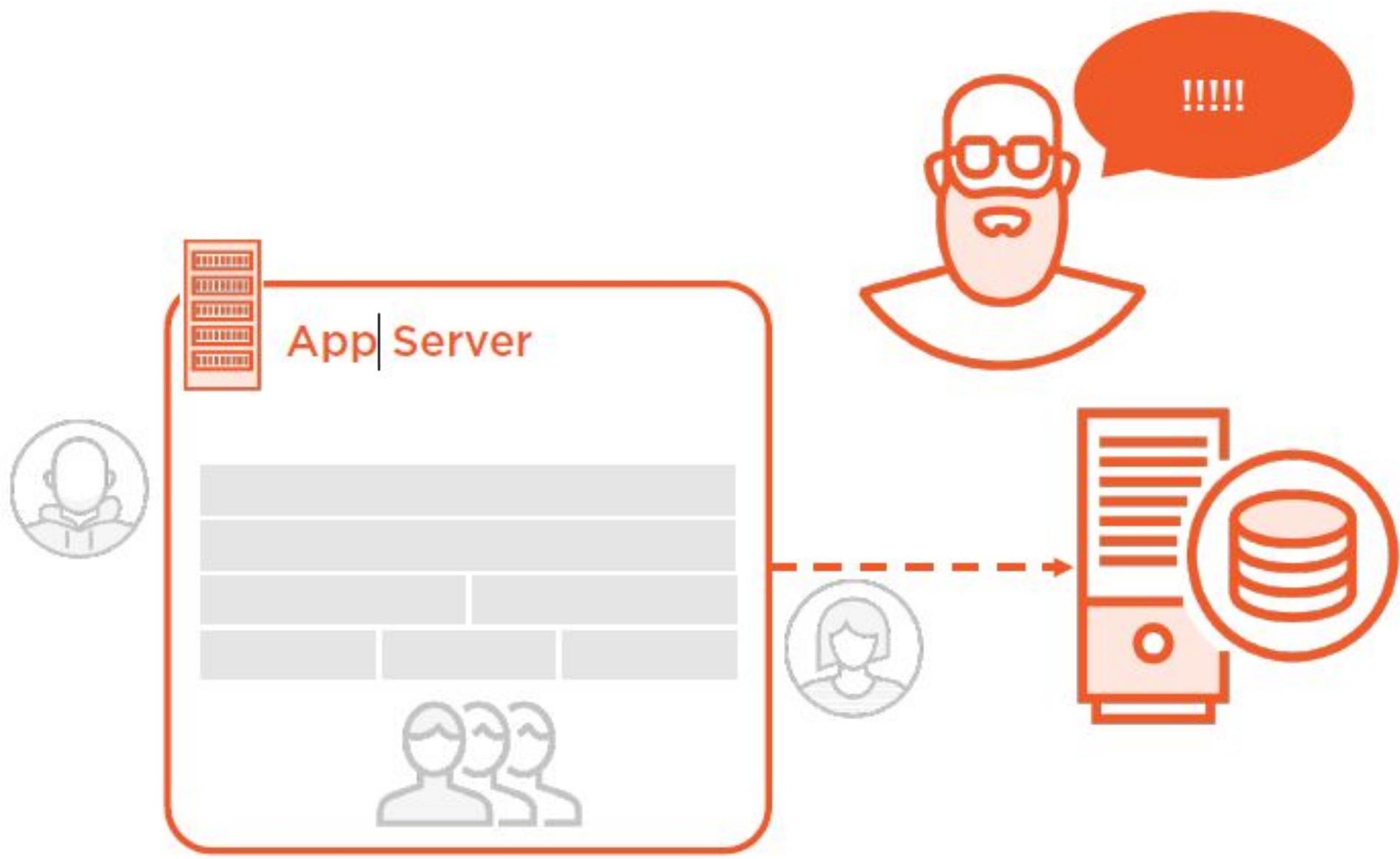


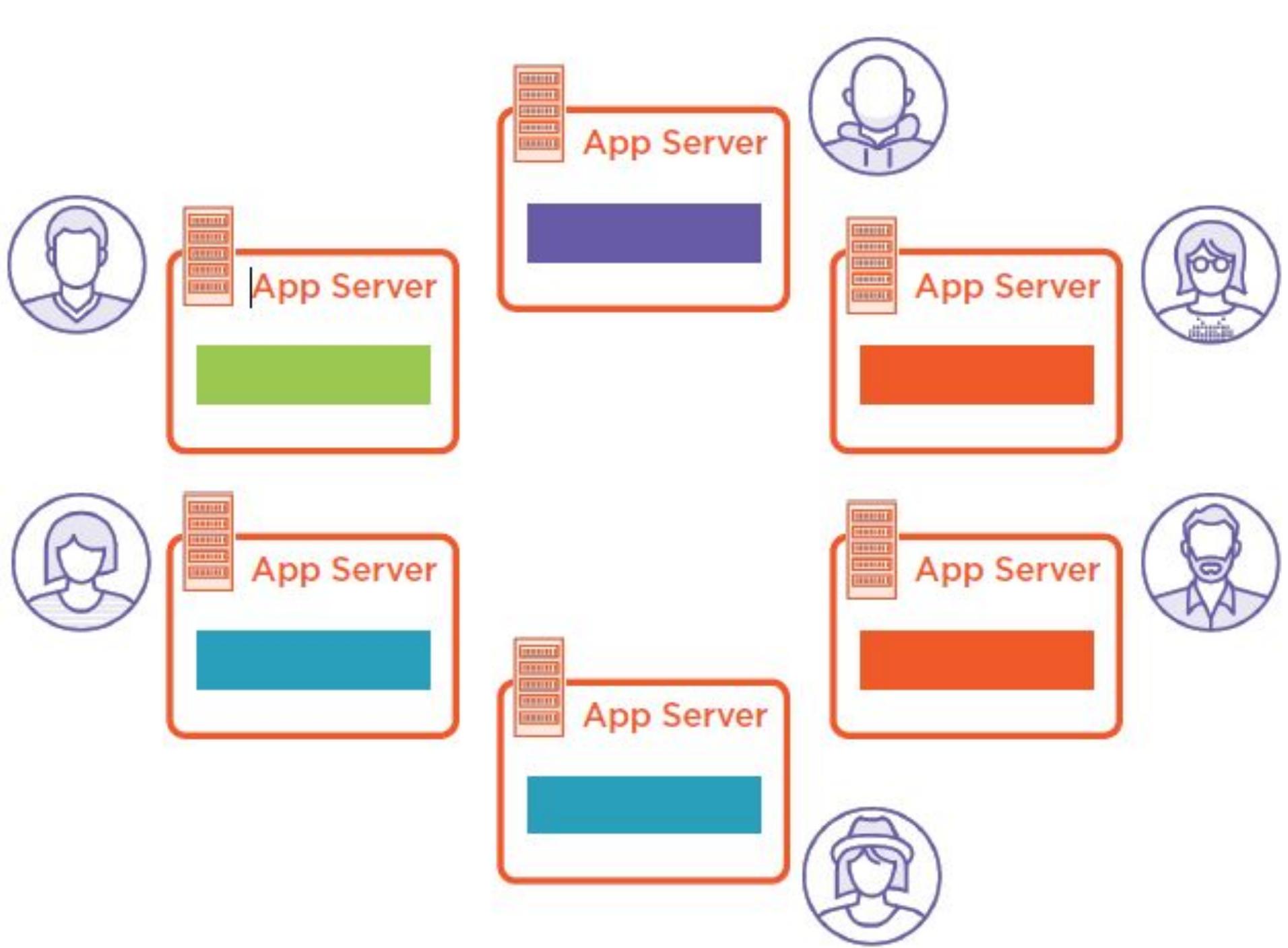
From IaaS to PaaS

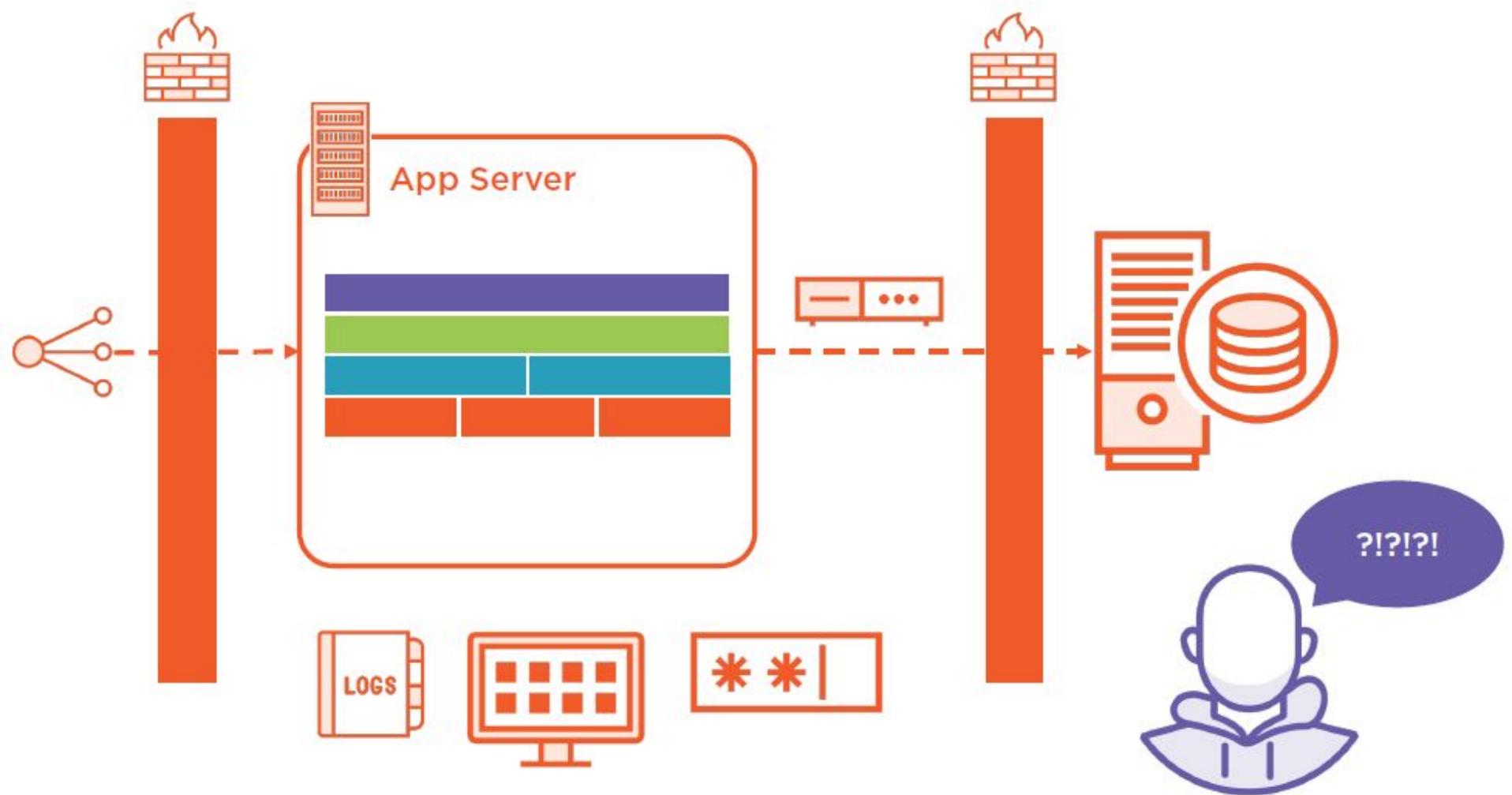


From IaaS to PaaS









Compute Services



Compute Engine



Kubernetes
Engine



App Engine



Cloud Functions

Serverless computing

Limited language support

Integrated lifecycle management

Ideal for reactive architecture

Cloud Functions



Managed code runtime

Integrated lifecycle management

Application stack

Multiple language support

App Engine Flexible

App Engine



**Container
orchestration**

Hosted Kubernetes

**Multiple cluster
configuration options**

Kubernetes Engine



Cluster Templates

Create a Kubernetes cluster

Cluster templates
Select a template with preconfigured setting, or customize a template to suit your needs

- Clone an existing cluster**
Select one of your existing clusters to populate fields
- Standard cluster**
Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.
- Your first cluster**
Experimenting with Kubernetes Engine, deploying your first application. Affordable choice to get started.
- CPU intensive applications**
Web crawling or anything else that requires more CPU.
- Memory intensive applications**
Databases, analytics, things like Hadoop

'Standard cluster' template
Continuous integration, web serving, backends. Best choice for further customization or if you are not sure what to choose.

Some fields can't be changed after the cluster is created. Hover over the help icons to learn more. Dismiss

Name standard-cluster-1

Location type Zonal Regional

Zone us-central1-a

Master version 1.9.7-gke.11 (default)

Node pools
Node pools are separate instance groups running Kubernetes in a cluster. You may add node pools in different zones for higher

Create **Cancel** Equivalent REST or command line

**Hosted container
registry service**

**Requires some
infrastructure
knowledge**

Kubernetes Engine



**Highly configurable
virtual machines**

**Predefined and
custom machine types**

**Cost optimization
options**

Sole tenant option

**Instance groups and
autoscaling**

Compute Engine

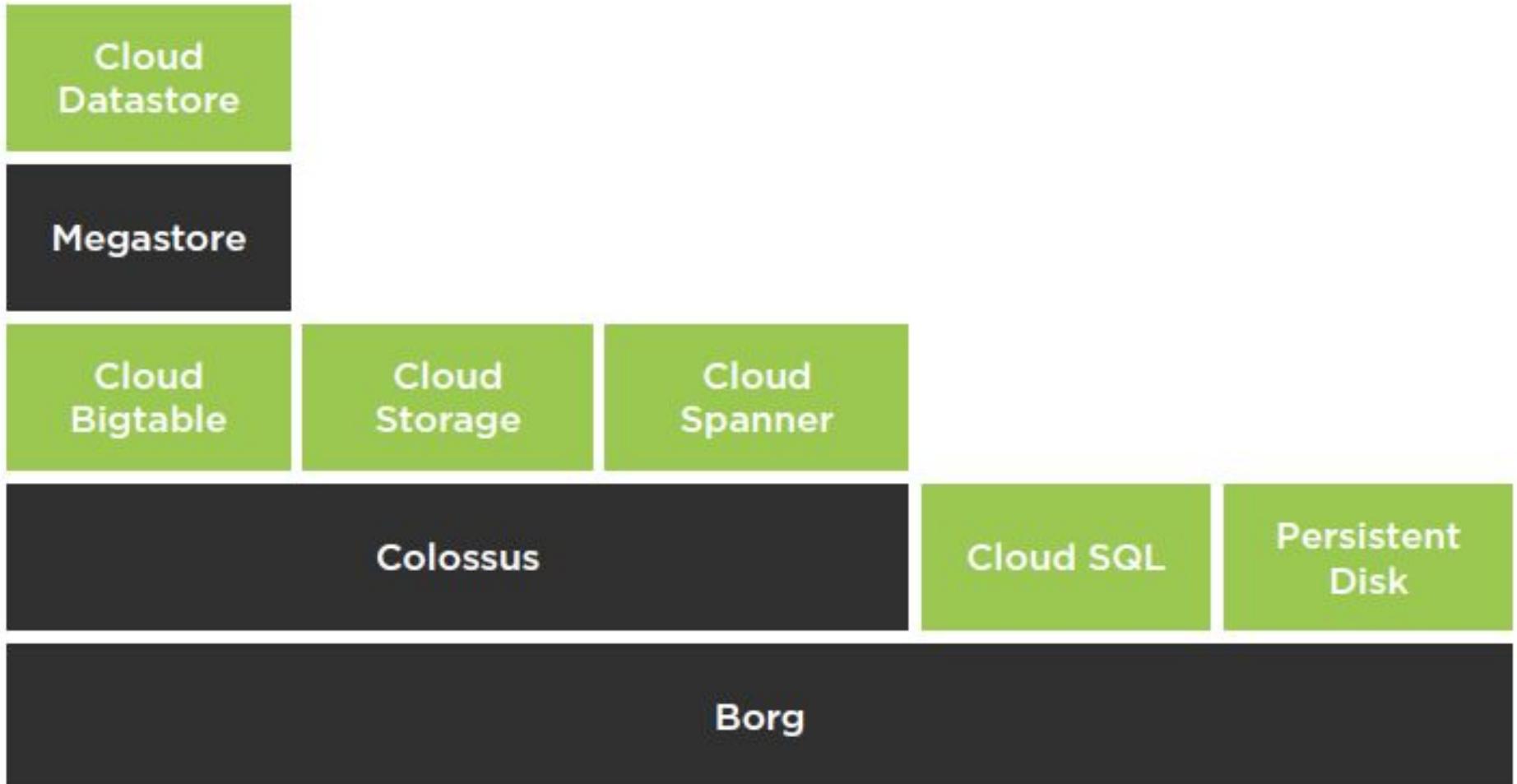


Storage services

Cloud SQL
Cloud Storage
Persistent Disk
Cloud Filestore
Cloud Bigtable
Cloud Spanner
Cloud Datastore
(Firestore)
Cloud Memorystore



Technology Layering



Cloud SQL



- Managed Postgres and MySQL
- Scalable for most OLTP workloads
- Benefits from Compute Engine cost optimizations
- Runtime and management configuration

Persistent Disks



Block storage for use with compute VMs
Independent of VMs
Highly configurable
Managed and optimized infrastructure

Cloud Filestore

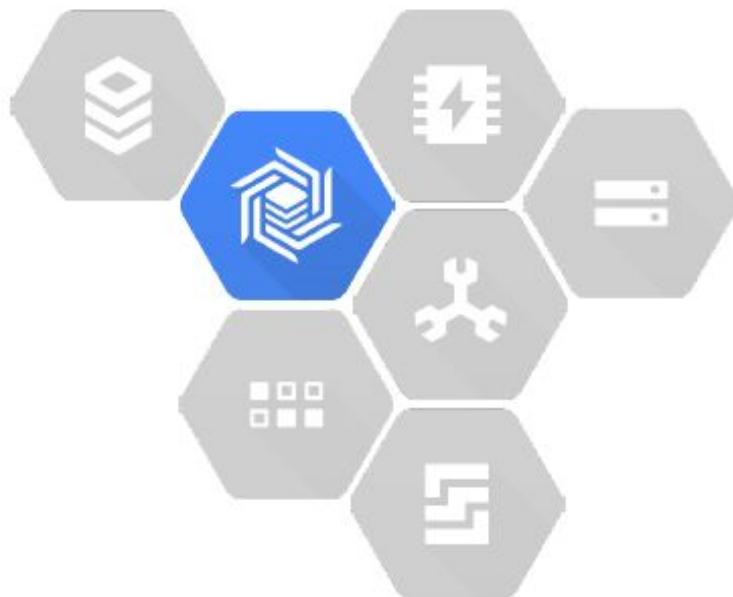


Managed Network Attached Storage (NAS)

Shared block storage for compute VMs or containers

Low latency, low maintenance

Cloud Bigtable



- High performance at scale
- Managed infrastructure
- Library support for multiple languages
- All entity information modeled as a row with a single index
- No transactional guarantees beyond a row operation
- Optimized for sparsely-populated rows

Cloud Storage



Global blob storage

Automatic edge caching

Different storage classes for different data types

Object lifecycle management

Cloud Spanner



Horizontally-scalable, managed RDBMS
Distributed transaction support
Built for Google's own applications

Cloud Datastore



Used for managing structured data
Limited transaction support
Scales based on size of query results

Cloud Firestore



- Brings together Cloud Datastore and the Firebase real-time database
- Changes data model and storage model
- Full transaction support
- 3 modes for transitioning from Cloud Datastore

Cloud Memorystore

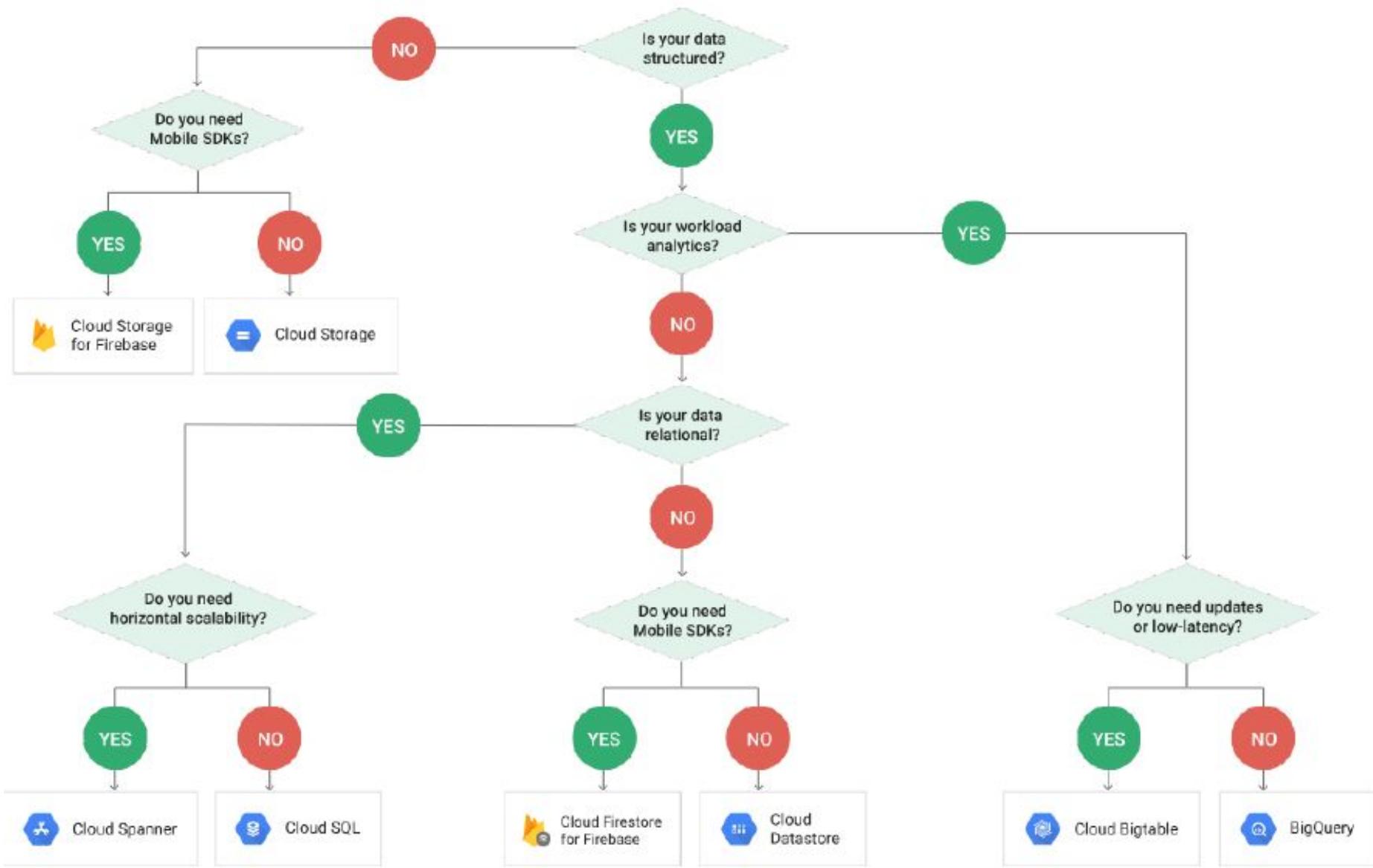


In-memory database

Ideal for caching

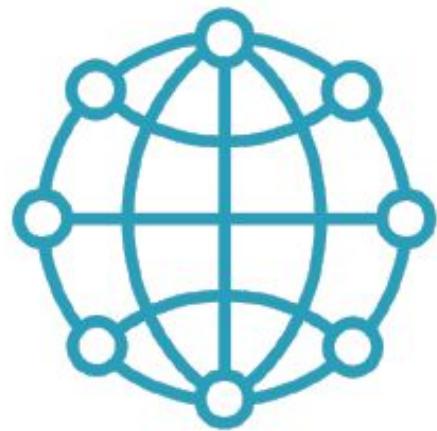
Supports Redis application protocol

Managed infrastructure

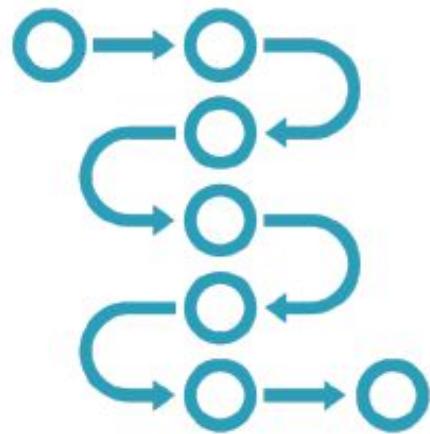


Networking services

What Makes Google's Network Special?



Size and scale

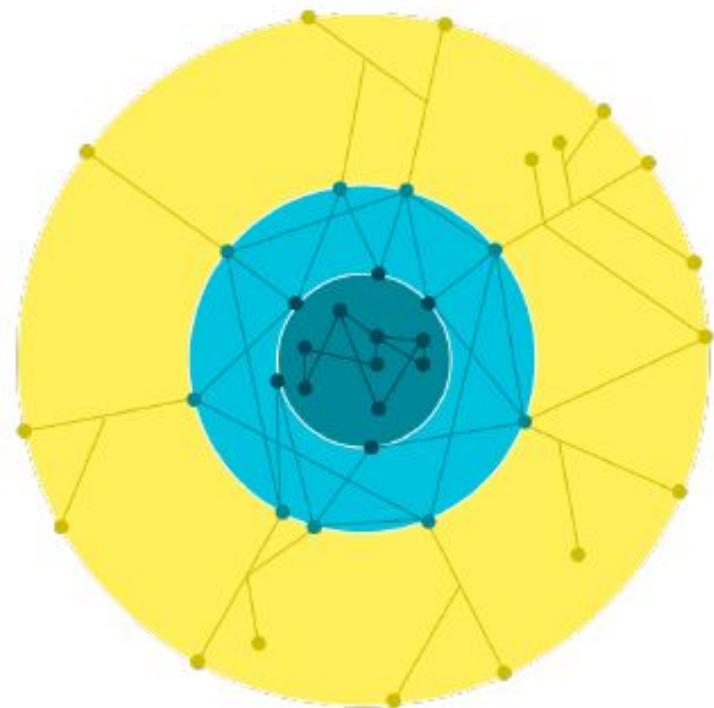


Agility



Performance





Data center network

Edge Points of Presence (PoPs)

Edge caching nodes (GGC)

B4 (2013)

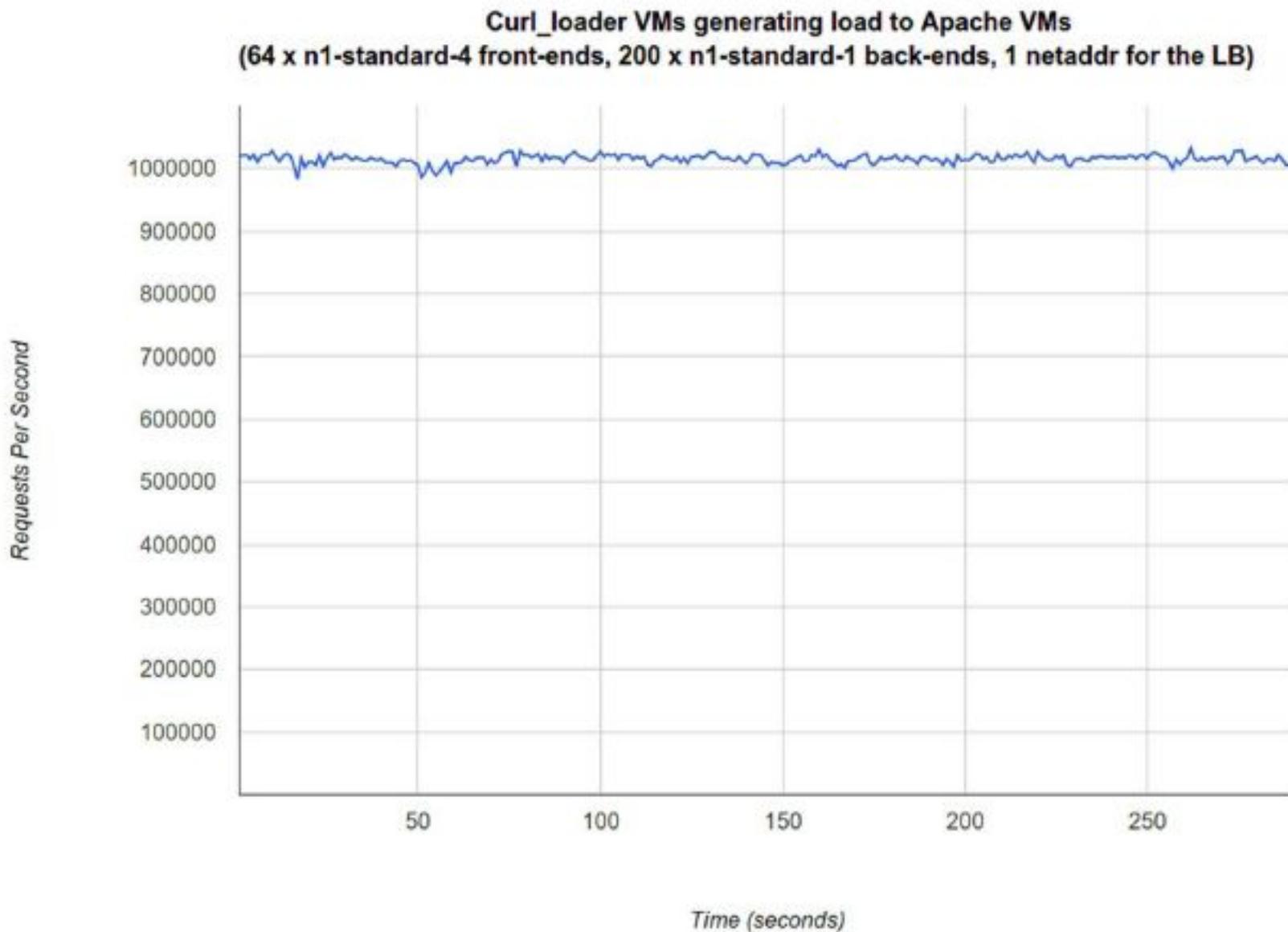
Andromeda (2014)

Jupiter (2015)

Espresso (2017)

Software Defined Networking (SDN)

Performance



Virtual Private Cloud

Cloud Load Balancing

Cloud Armor and Telemetry

Content Delivery Network

Cloud DNS

Cloud Interconnect

Network Service Tiers



Virtual Private Cloud (VPC)



Private network space

Metadata-driven approach to policy

Shared VPCs for large, federated systems

Cloud Load Balancing



Single, load-balanced IP address
Uses anycast IP addresses

Cloud Armor and Telemetry



Protection against DDoS attack

Applies policy on top of Cloud Load Balancer

Telemetry provides detailed inspection of all VPC ingress and egress

Content Delivery Network (CDN)



Extends caching beyond peering edge
Caches content on Google Global Cache nodes

Cloud DNS



Leverages existing Google DNS infrastructure

Flexible DNS configuration management

Cloud Interconnect



- Connects existing network infrastructure to Google network
- Includes both VPN and peering connections
- Supports direct and partner-mediated connections

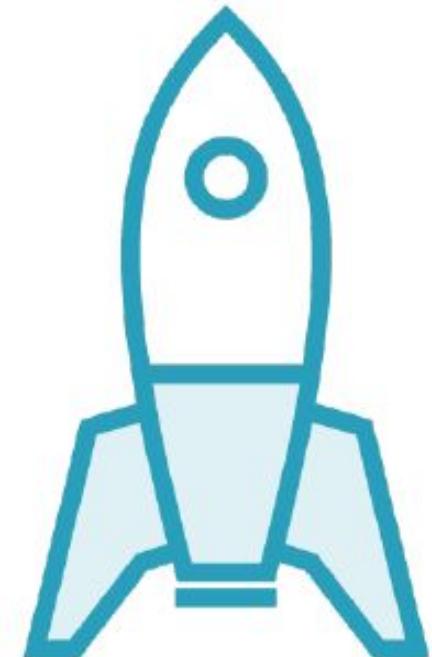
Network Service Tiers



Base level and premium tiers

Difference based on how long traffic stays
in Google's network

Enabling “Cloud 3.0”



- Applications and functions, not VMs
- Storage disaggregation, not disks
- SLAs, not load balancing and scheduling
- Intelligence, not data processing
- Policy, not “middle boxes”

Summary



Core building blocks include compute, storage, and networking

Layering provides choice and agility

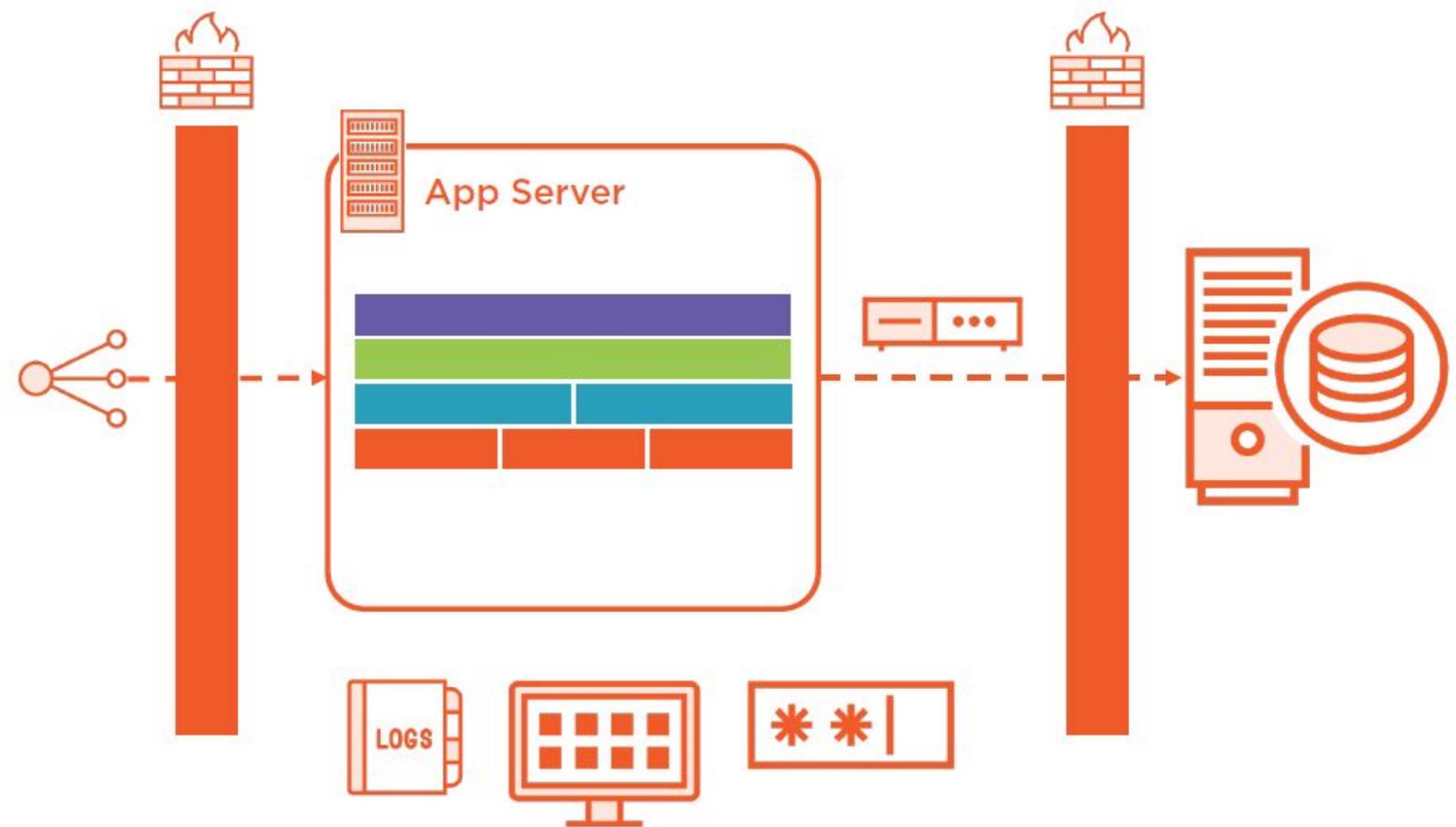
Common infrastructure across all of Google

Building for “Cloud 3.0”

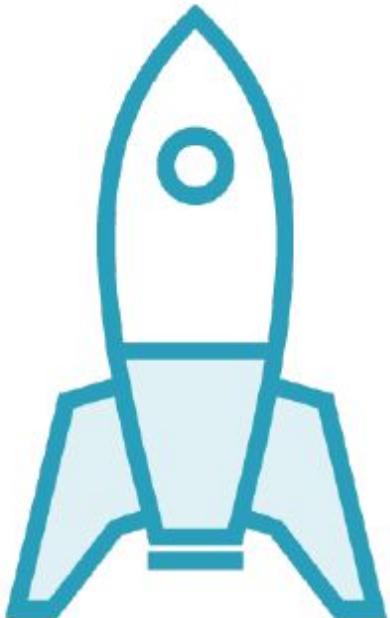
“Cloud 2.0”



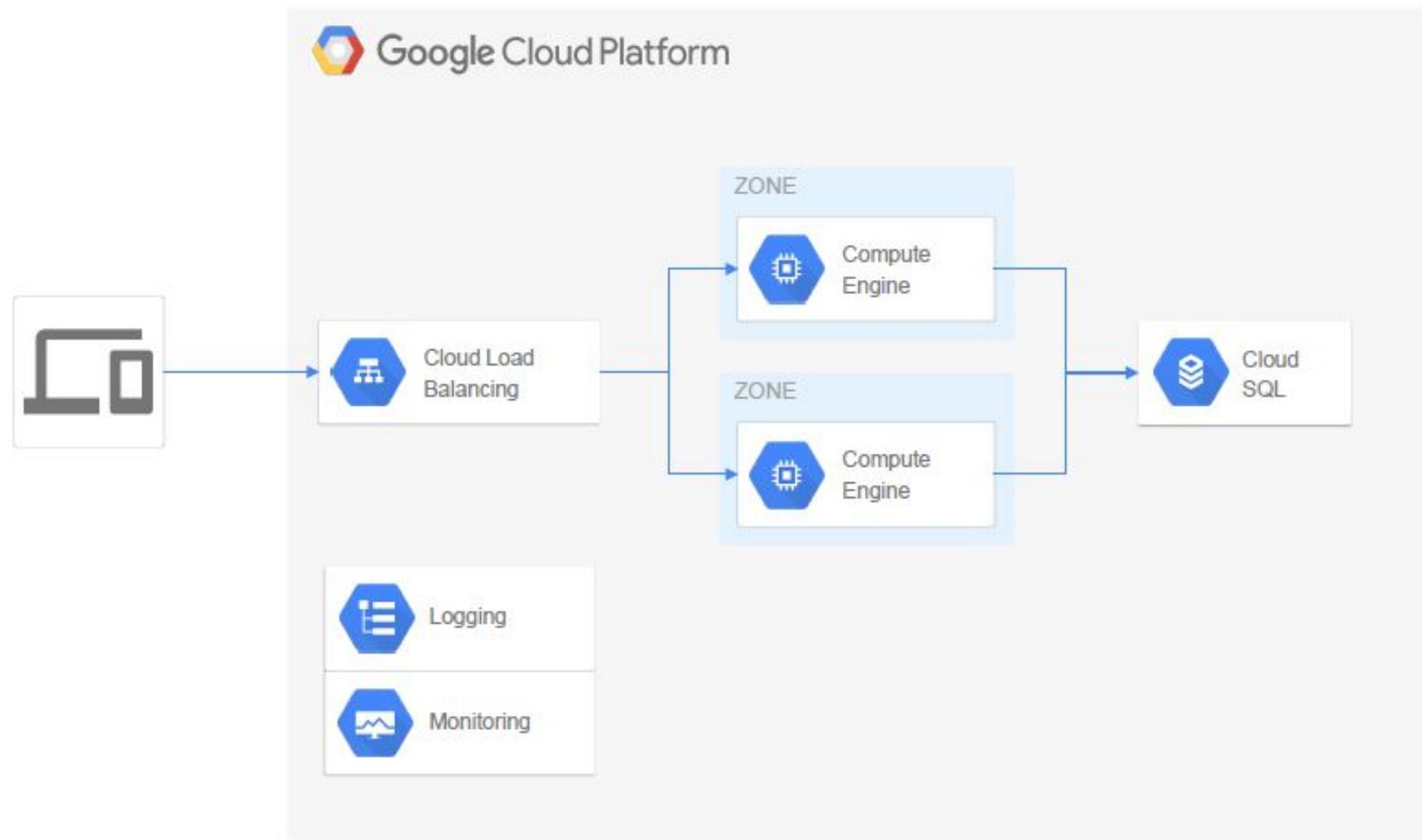
On-demand computing resources
Projection of on-premises architecture



“Cloud 3.0”



- Applications and functions, not VMs
- Storage disaggregation, not disks
- SLAs, not load balancing and scheduling
- Policy, not “middle boxes”
- Intelligence, not data processing





Managed logging and monitoring
Declarative network policy
Task automation



Google Cloud Platform



Cloud Load
Balancing



Cloud
SQL



Logging



Monitoring



Google Cloud Platform



Cloud Load
Balancing



Kubernetes
Engine



Cloud
SQL



Logging



Monitoring

Benefits of Container Orchestration



- Technical: Increase density
- Architectural: Decomposition
- Organizational: End-to-end teams

Processing Events



Cloud PubSub



Cloud Dataflow

Cloud PubSub



- Fully managed, message broker
- Follows the publish/subscribe messaging pattern
- “At least once” message delivery
- Ordering is not guaranteed

Cloud Dataflow



Unified data processing platform for batch
and streaming workloads

Programming model is Apache Beam

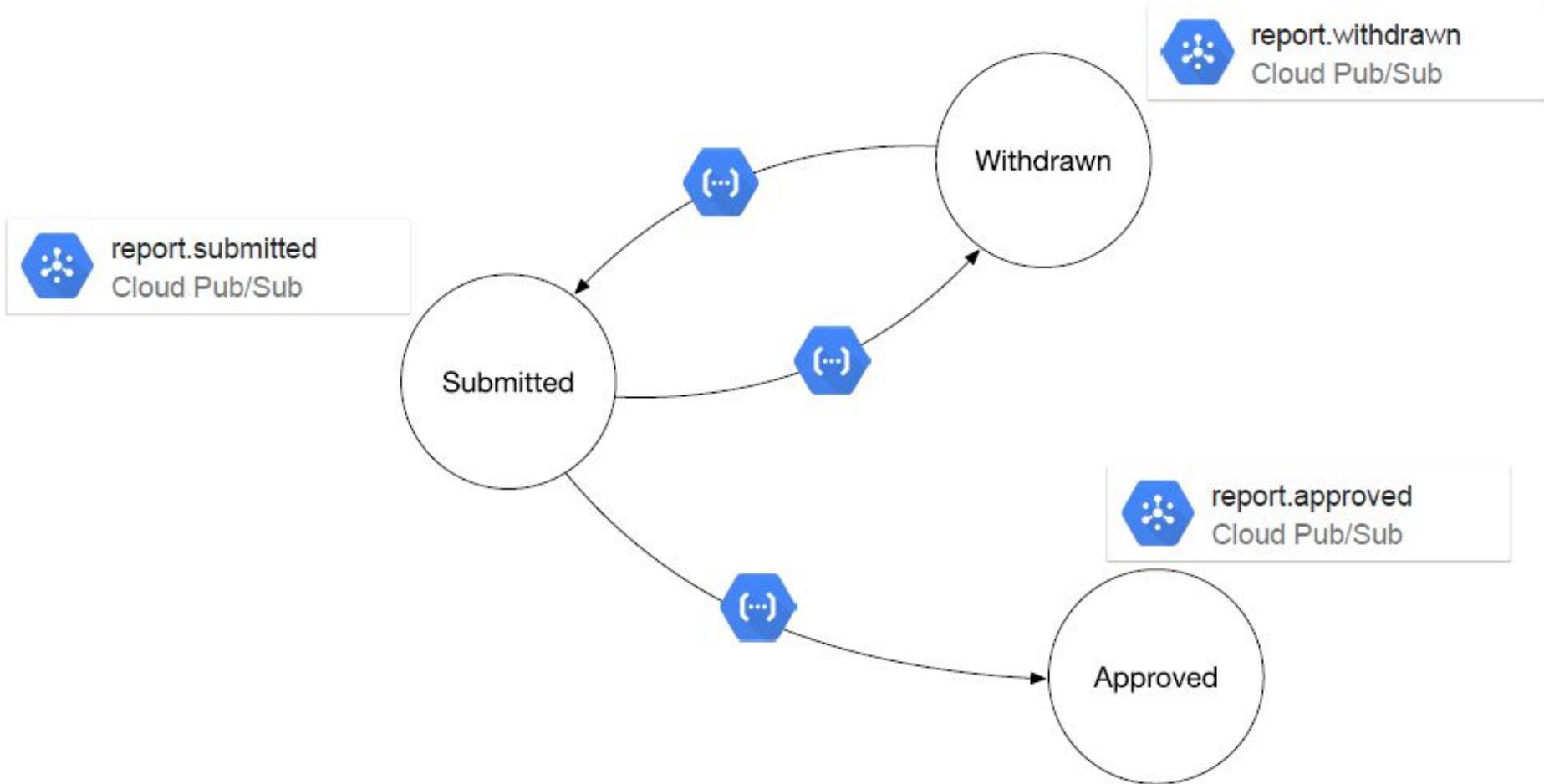
Towards a New Architecture

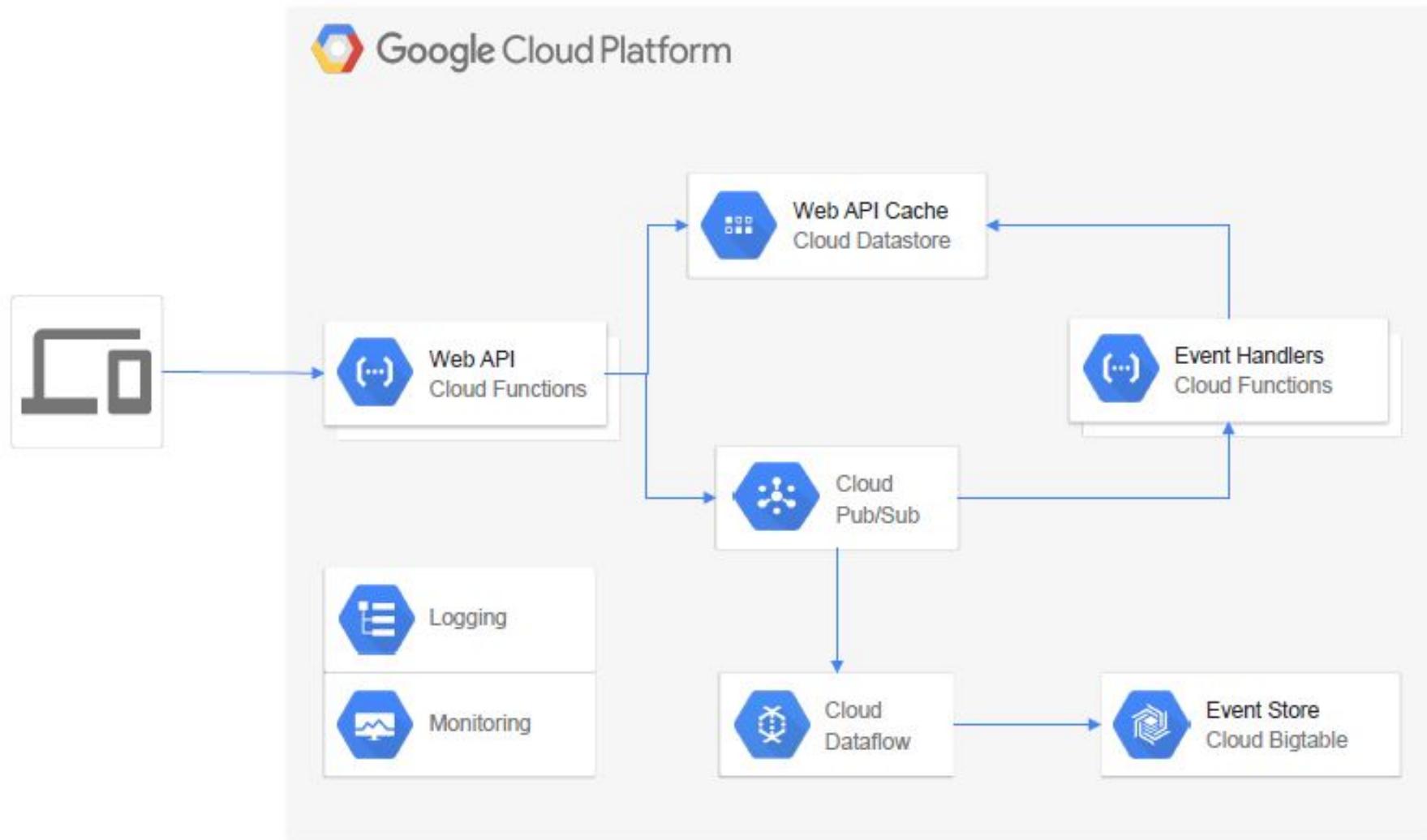


Related terms: Event sourcing, CQRS, Kappa

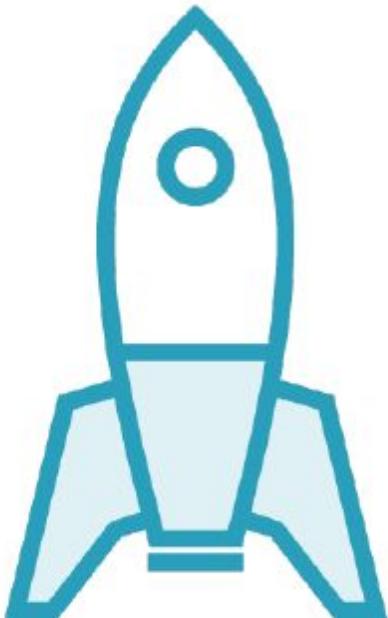
System is a single state machine

State machine is implemented using PubSub topics and subscriptions



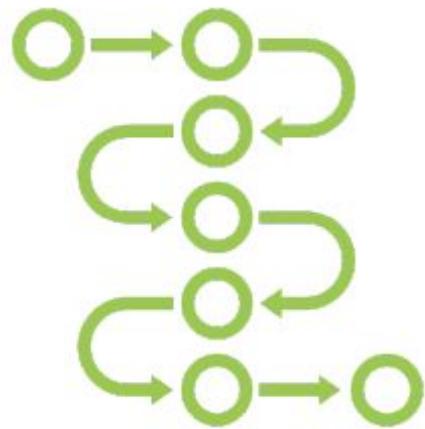


“Cloud 3.0”

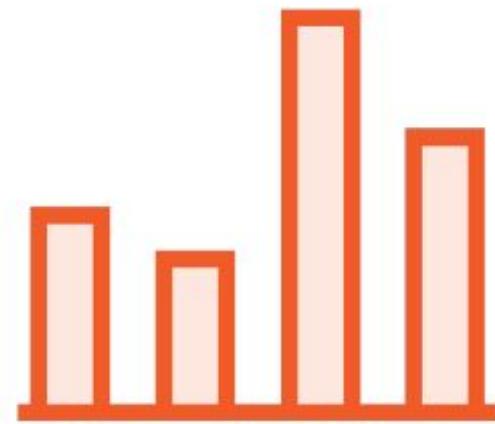


- Applications and functions, not VMs
- Storage disaggregation, not disks
- SLAs, not load balancing and scheduling
- Policy, not “middle boxes”
- Intelligence, not data processing

Traditional Business Applications



Transactional Applications



Data Applications

Intelligence,
not data processing.

Analytics Services



Big Query



Cloud Dataproc



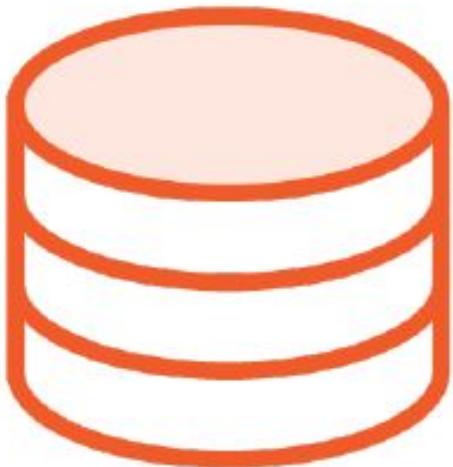
Cloud Datalab



Cloud Dataprep

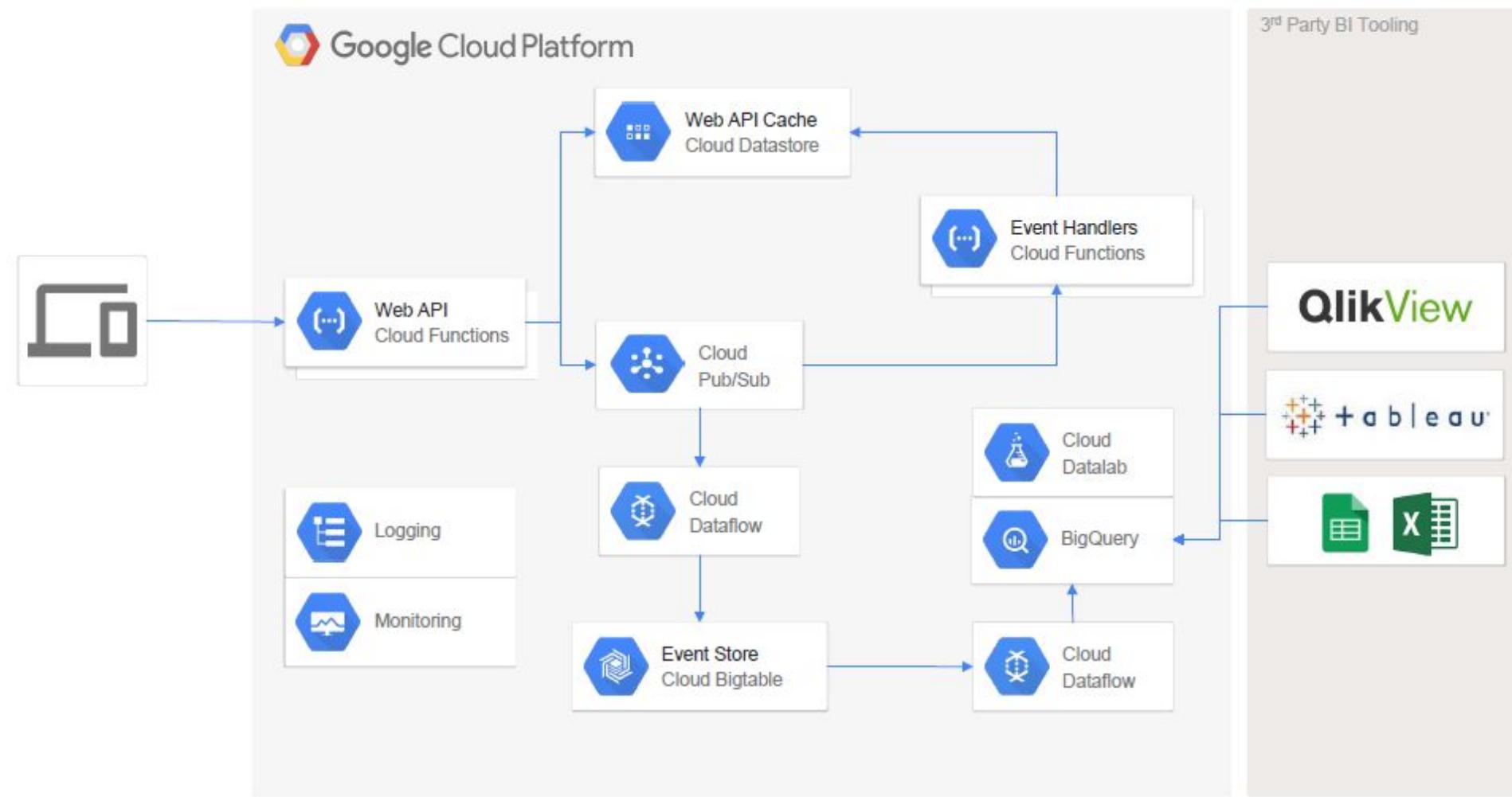


Cloud Data Studio



cart_ID	product_ID	qty	unit_price
1	432	1	15.95
1	654	3	3.50
1	126	1	25.00

2011-12-17T02:44:48.481Z	cart.item-added	{...}
2011-12-17T02:46:15.310Z	cart.item-added	{...}
2011-12-17T02:51:12.052Z	cart.item-removed	{...}
2011-12-17T02:57:21.000Z	cart.item-added	{...}



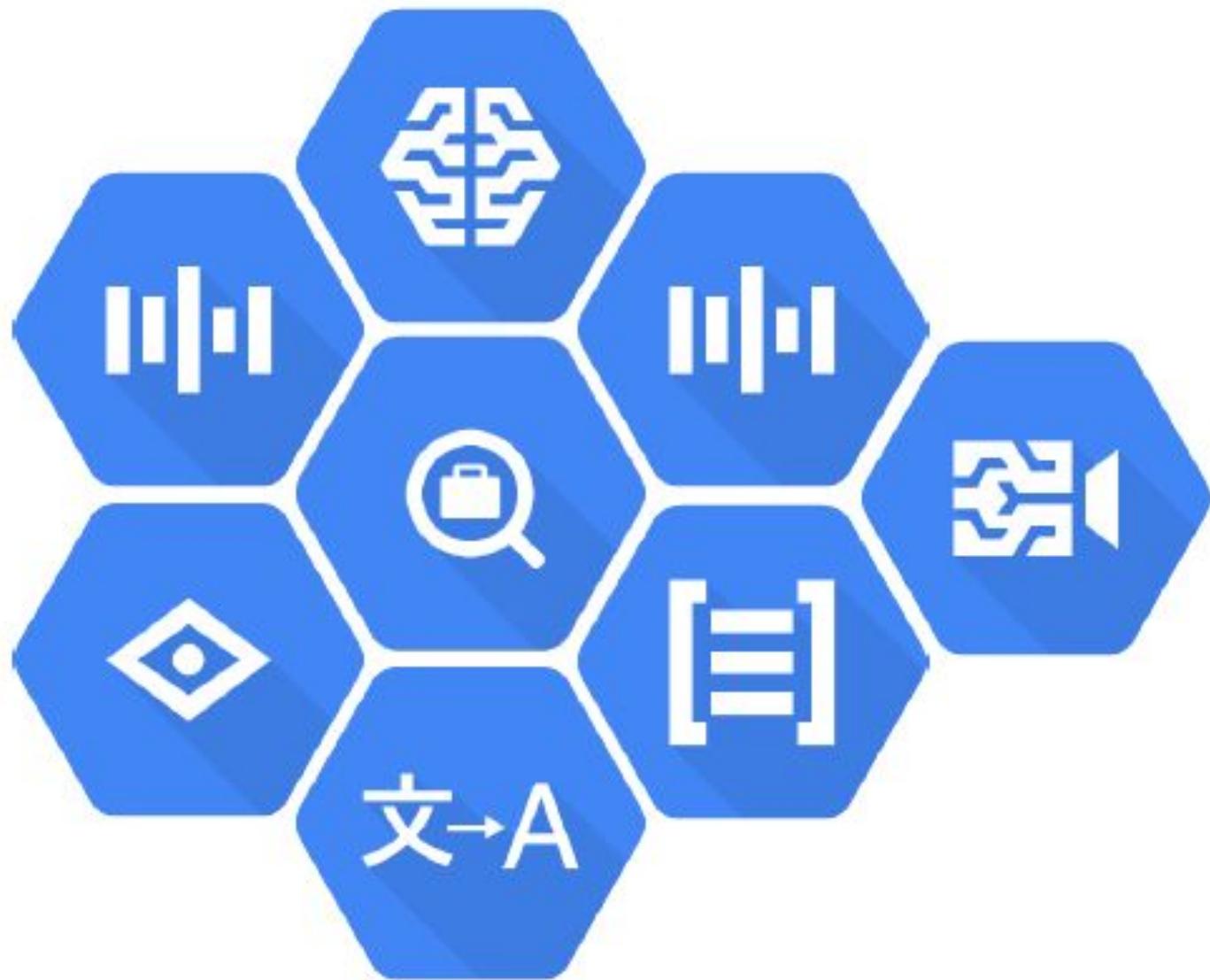
Machine Learning

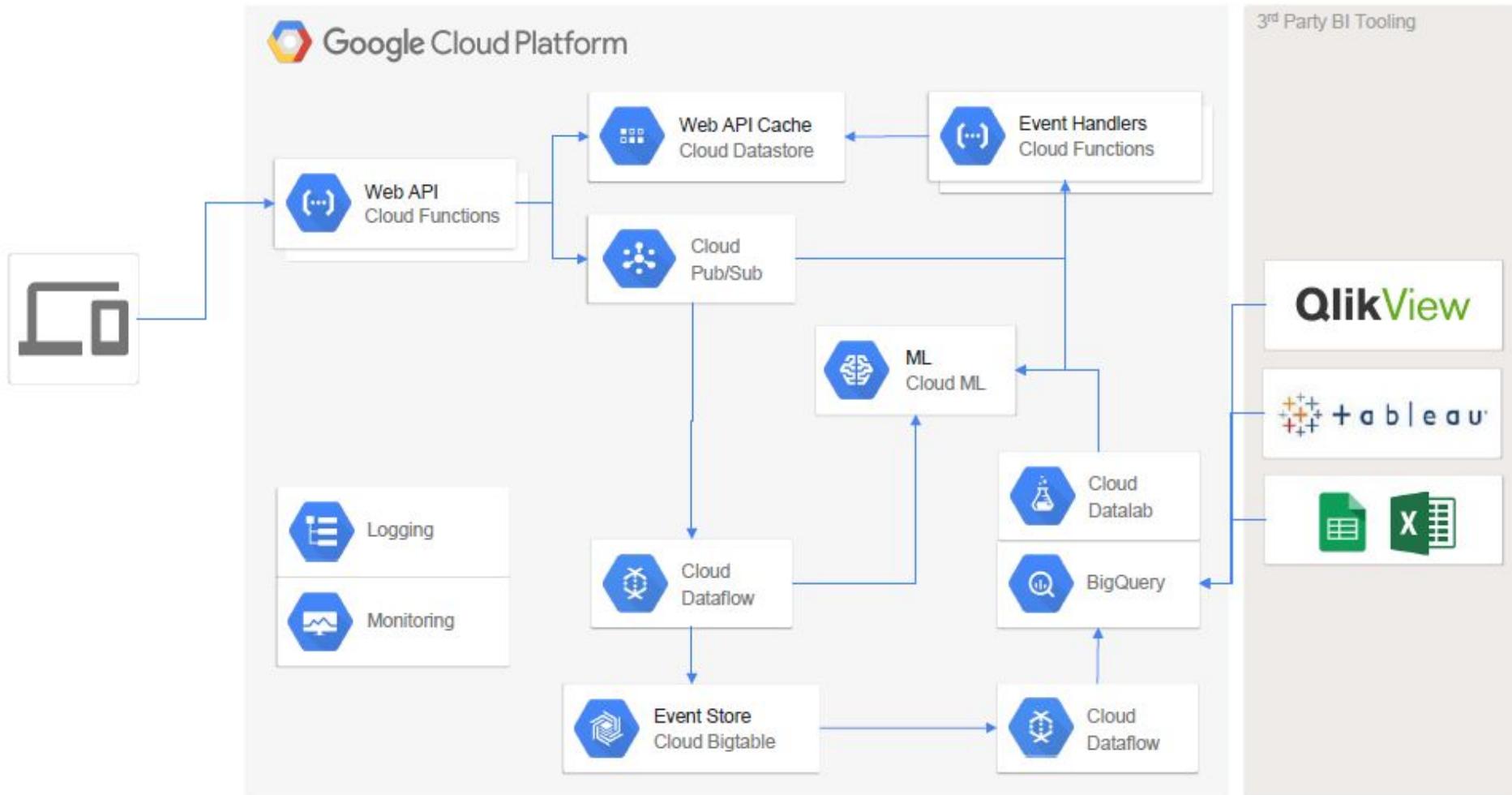
Machine Learning at Google

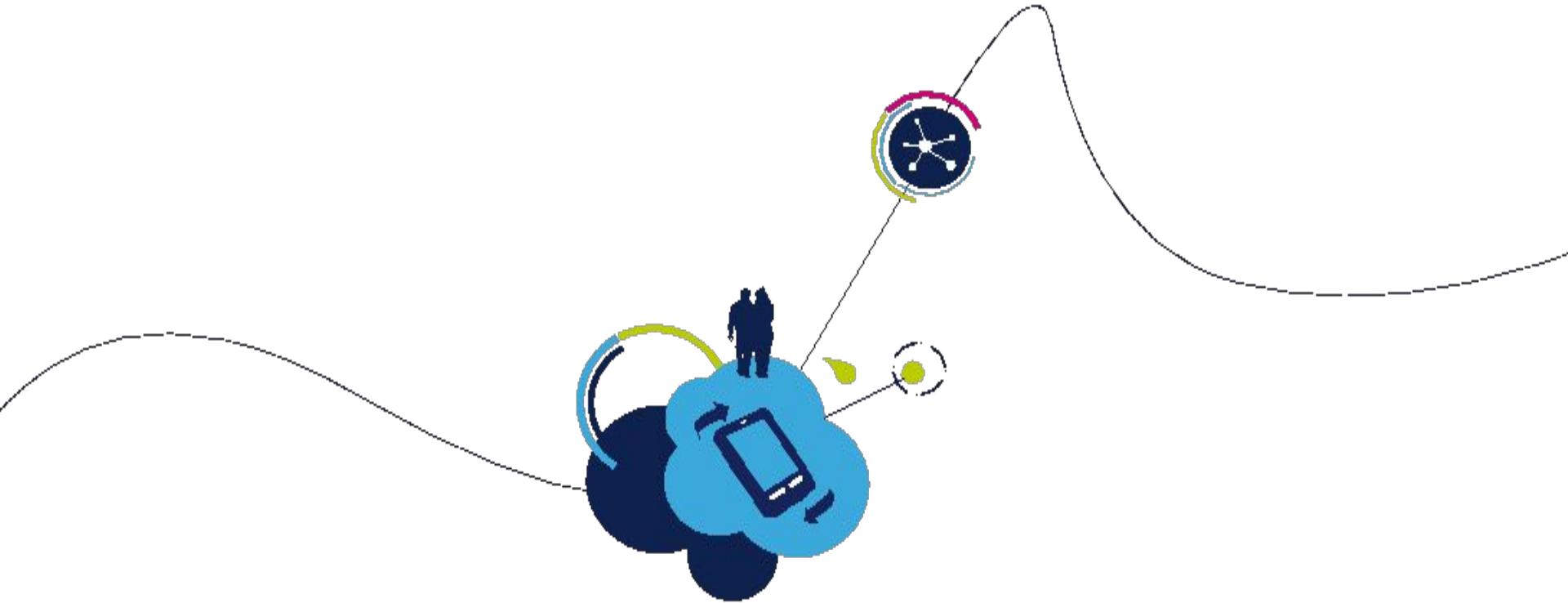


- Built from internal efforts of the “Google Brain” team
- Released as TensorFlow project
- Fully-managed
- Supported by custom hardware
- Integrated

Machine Learning Services



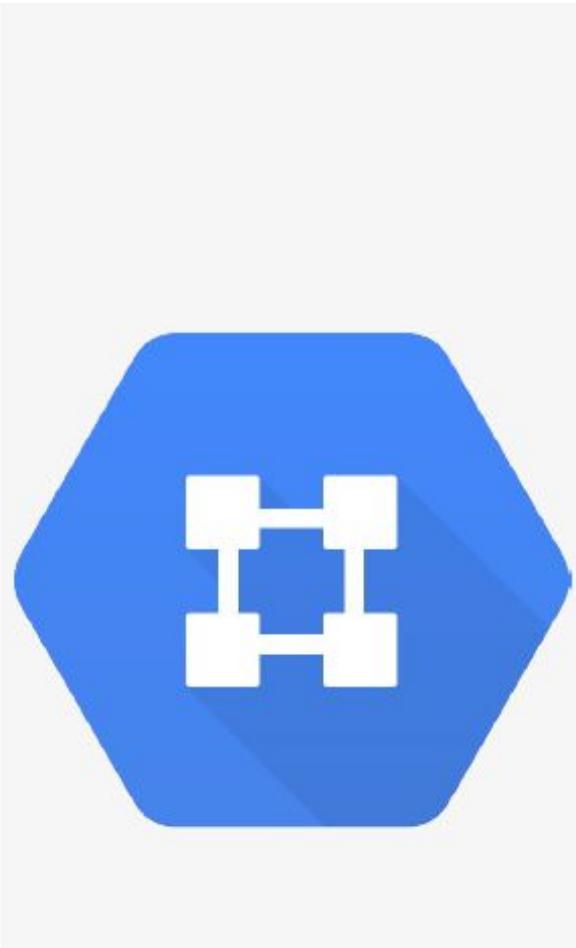




Virtual Private Cloud (VPC) Network

Virtual Private Cloud Networking

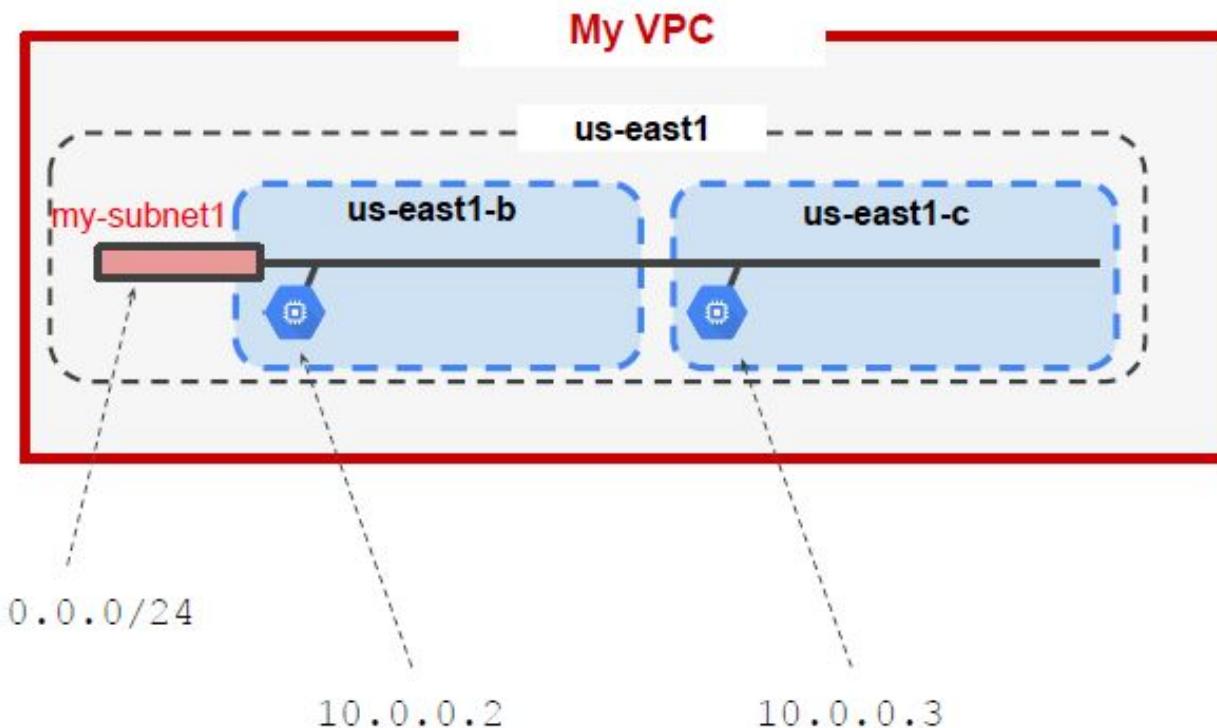
- Each VPC network is contained in a GCP project.
- You can provision Cloud Platform resources, connect them to each other, and isolate them from one another.



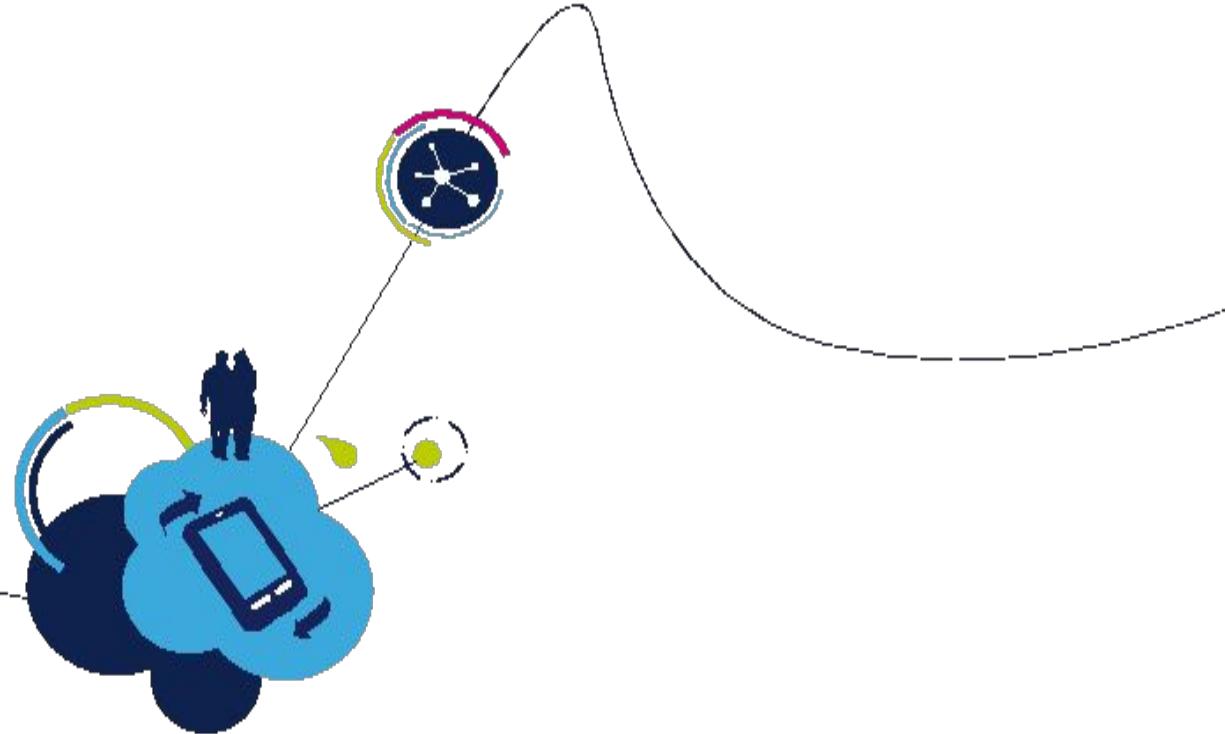
- Your VPC networks connect your Google Cloud Platform resources to each other and to the internet.
- You can segment your networks, use firewall rules to restrict access to instances, and create static routes to forward traffic to specific destinations.
- Many users get started with GCP by defining their own Virtual Private Cloud inside their first GCP project.
- Or they can simply choose the default VPC and get started with that.

- Google Virtual Private Cloud networks that you define have global scope.
- They can have subnets in any GCP region worldwide.
- Subnets can span the zones that make up a region.
- This architecture makes it easy for you to define your own network layout with global scope.
- You can also have resources in different zones on the same subnet.

Google Cloud VPC networks are global; subnets are regional



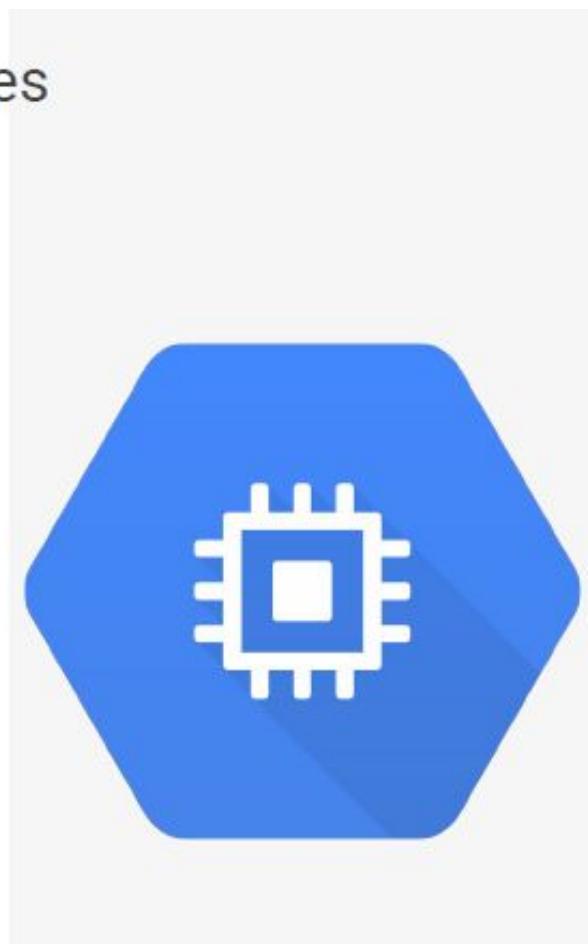
- You can dynamically increase the size of a subnet in a custom network by expanding the range of IP addresses allocated to it.
- Doing that doesn't affect already configured VMs.
- In this example, your VPC has one network. So far, it has one subnet defined, in GCP's **us-east1** region.
Notice that it has two Compute Engine VMs attached to it.
- *They're **neighbors** on the same subnet even though they are in different zones!*
- You can use this capability to build solutions that are resilient but still have simple network layouts.



Compute Engine

Compute Engine offers managed virtual machines

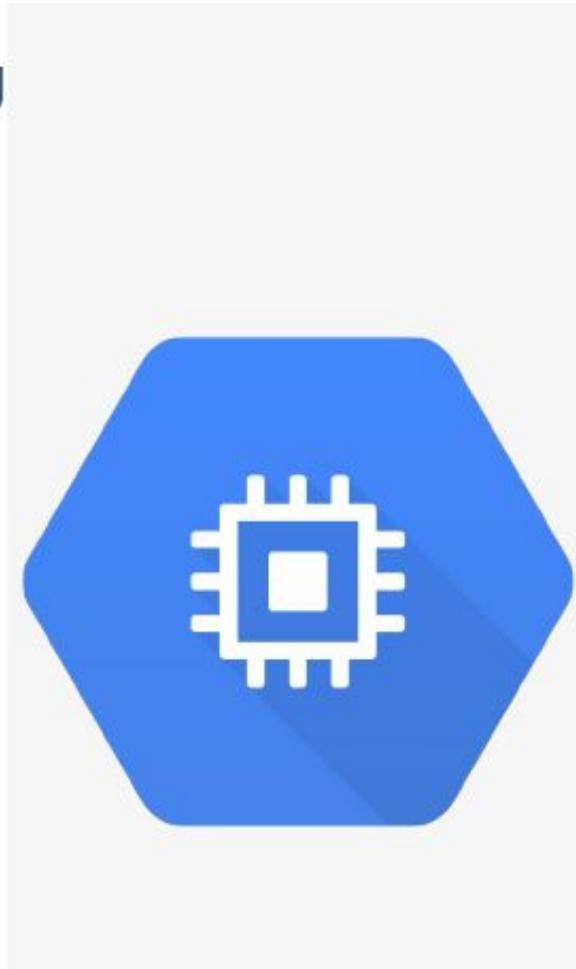
- High CPU, high memory, standard and shared-core machine types
- Persistent disks
 - Standard, SSD, local SSD
 - Snapshots
- Resize disks with no downtime
- Instance metadata and startup scripts



- Virtual machines have the power and generality of a full-fledged operating system in each.
- You configure a virtual machine much like you build out a physical server:
=> By specifying its amounts of CPU power and memory, its amounts and types of storage, and its operating system.
- Compute Engine lets you create and run **virtual machines** on Google infrastructure.
- There are no upfront investments, and you can run thousands of virtual CPUs on a system that is designed to be fast and to offer consistent performance.

Compute Engine offers customer friendly pricing

- Per-second billing, sustained use discounts, committed use discounts
- Preemptible instances
- High throughput to storage at no extra cost
- Custom machine types: Only pay for the hardware you need



- You can create a virtual machine instance by using the Google Cloud Platform **Console** or the **gcloud** command-line tool.
- A Compute Engine can run Linux and Windows Server **images** provided by Google or any instance customized versions of these images.
- You can also build and run images of other operating systems.

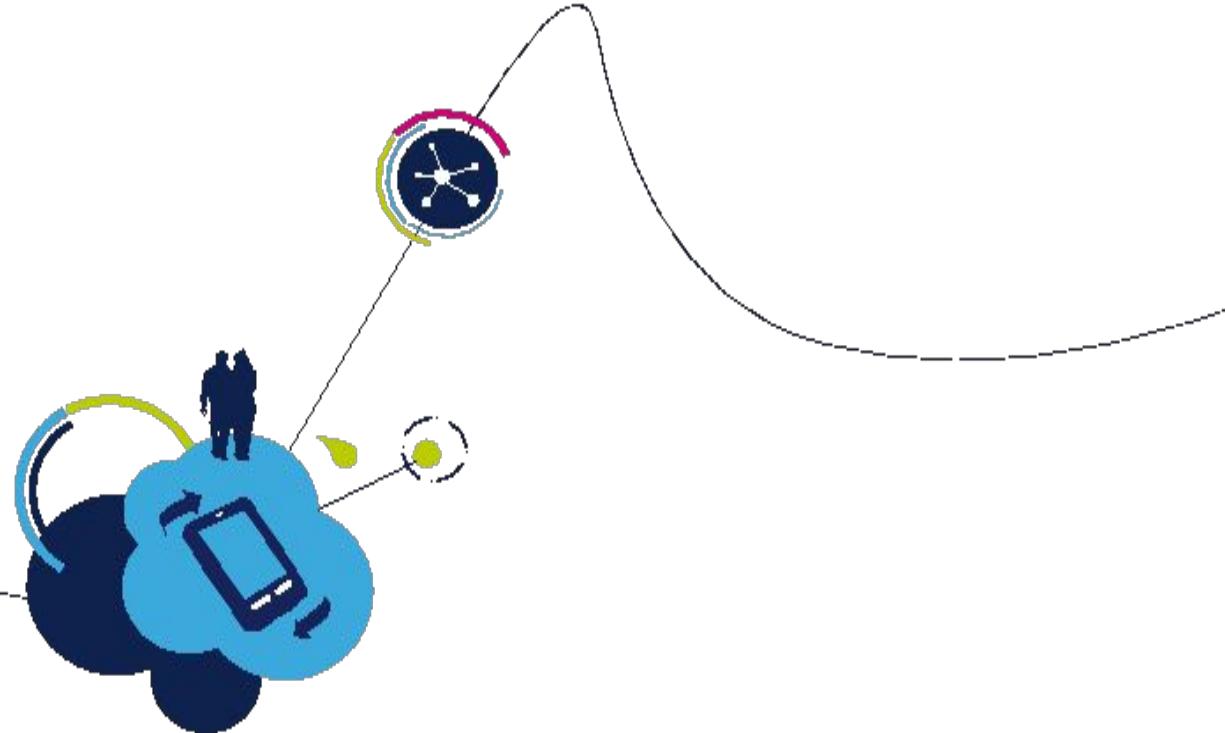
Scale up or scale out with Compute Engine



Use big VMs for memory- and compute-intensive applications



Use Autoscaling for resilient, scalable applications



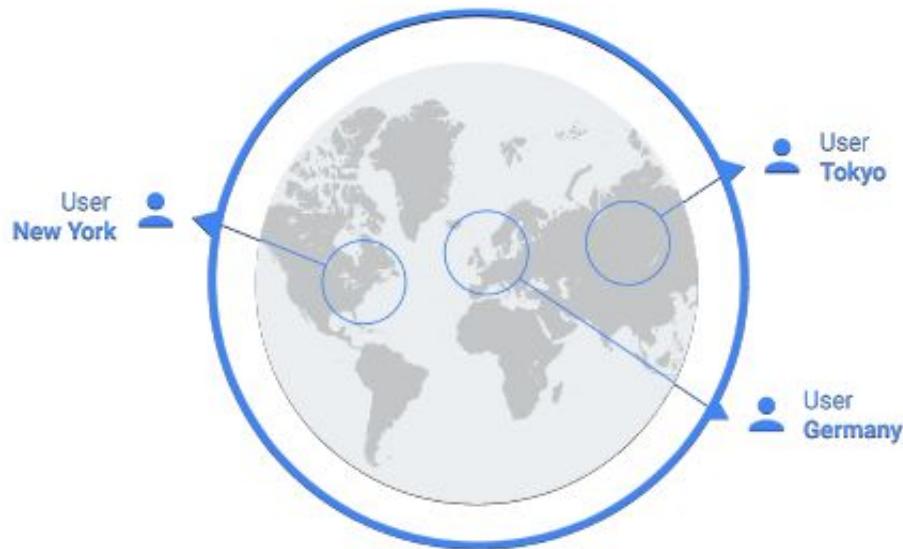
Important VPC capabilities

You control the topology of your VPC network

- Use its route table to forward traffic within the network, even across subnets.
- Use its firewall to control what network traffic is allowed.
- Use Shared VPC to share a network, or individual subnets, with other GCP projects.
- Use VPC Peering to interconnect networks in GCP projects.



With global Cloud Load Balancing, your application presents a single front-end to the world



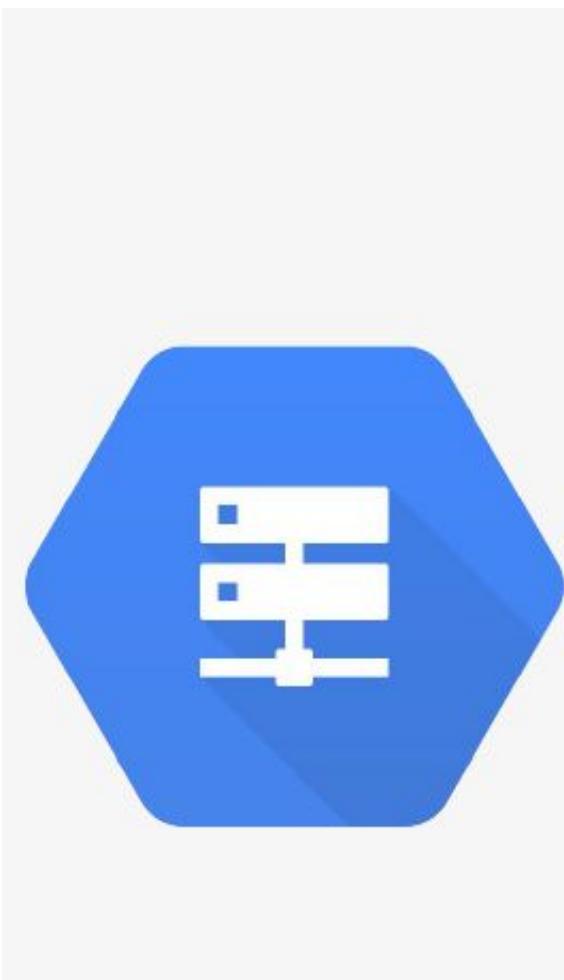
- Users get a single, global anycast IP address.
- Traffic goes over the Google backbone from the closest point-of-presence to the user.
- Backends are selected based on load.
- Only healthy backends receive traffic.
- No pre-warming is required.

Google VPC offers a suite of load-balancing options

Global HTTP(S)	Global SSL Proxy	Global TCP Proxy	Regional	Regional internal
Layer 7 load balancing based on load	Layer 4 load balancing of non-HTTPS SSL traffic based on load	Layer 4 load balancing of non-SSL TCP traffic	Load balancing of any traffic (TCP, UDP)	Load balancing of traffic inside a VPC
Can route different URLs to different back ends	Supported on specific port numbers	Supported on specific port numbers	Supported on any port number	Use for the internal tiers of multi-tier applications

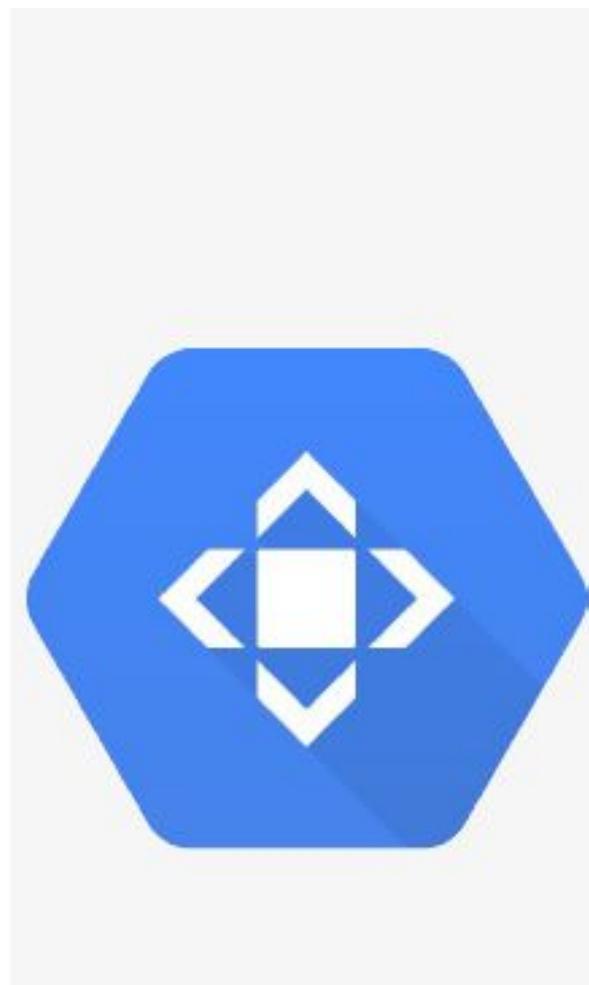
Cloud DNS is highly available and scalable

- Create managed zones, then add, edit, delete DNS records
- Programmatically manage zones and records using RESTful API or command-line interface



Cloud CDN (Content Delivery Network)

- Use Google's globally distributed edge caches to cache content close to your users
- Or use CDN Interconnect if you'd prefer to use a different CDN



Google Cloud Platform offers many interconnect options



VPN

Secure multi-Gbps connection over VPN tunnels



Direct Peering

Private connection between you and Google for your hybrid cloud workloads



Dedicated Interconnect

Connect N X 10G transport circuits for private cloud traffic to Google Cloud at Google POPs
SLAs available



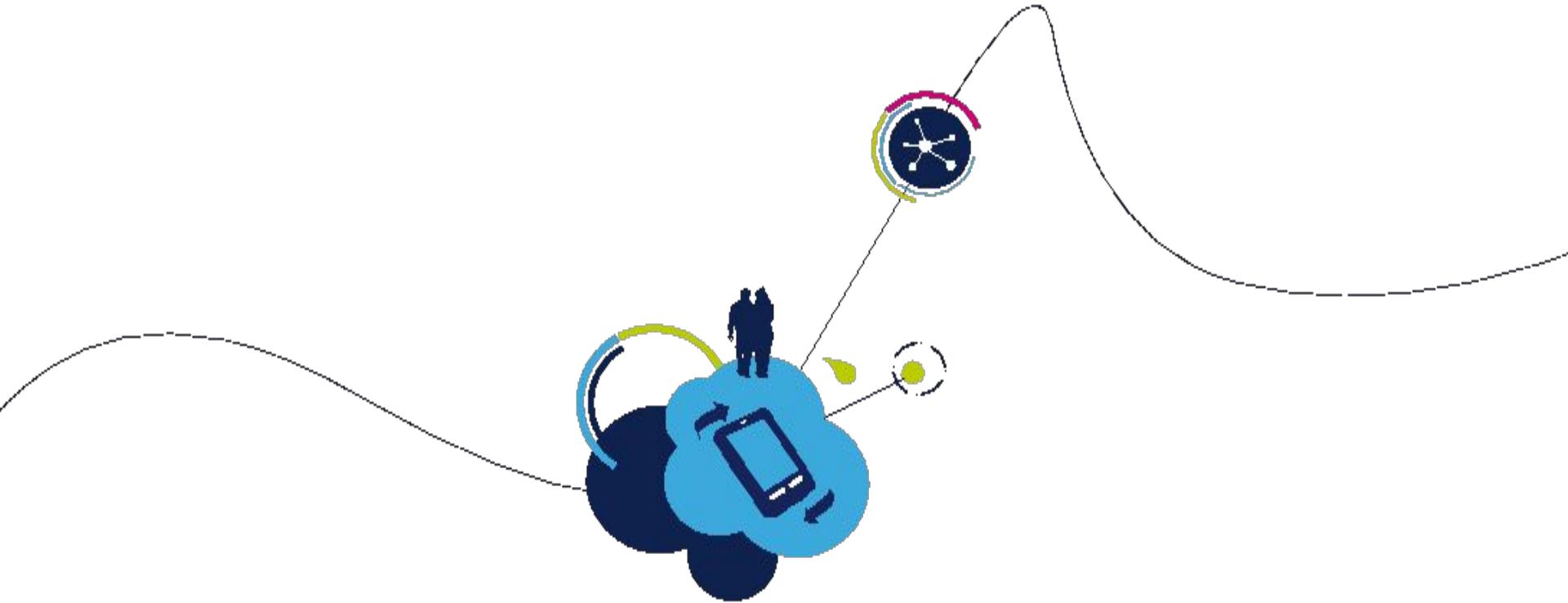
Carrier Peering

Connection through the largest partner network of service providers



Partner Interconnect

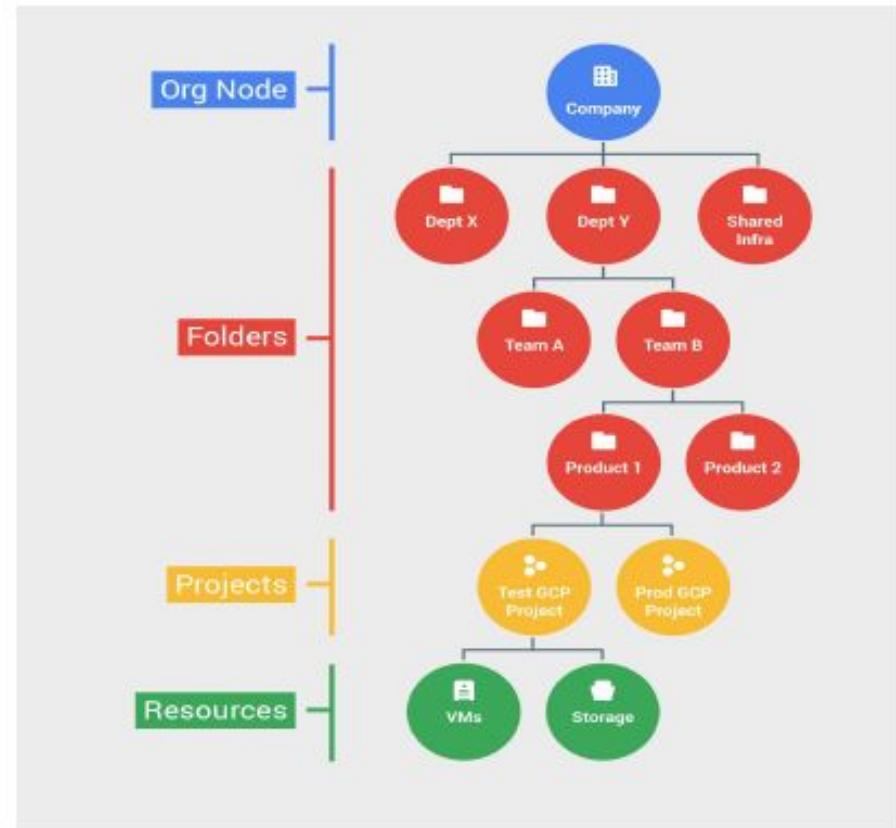
Connectivity between your on-premises network and your VPC network through a supported service provider
SLAs available



Google Cloud Platform resource hierarchy

Resource hierarchy levels define trust boundaries

- Group your resources according to your organization structure.
- Levels of the hierarchy provide trust boundaries and resource isolation.



All GCP services you use are associated with a project

- Track resource and quota usage.
- Enable billing.
- Manage permissions and credentials.
- Enable services and APIs.

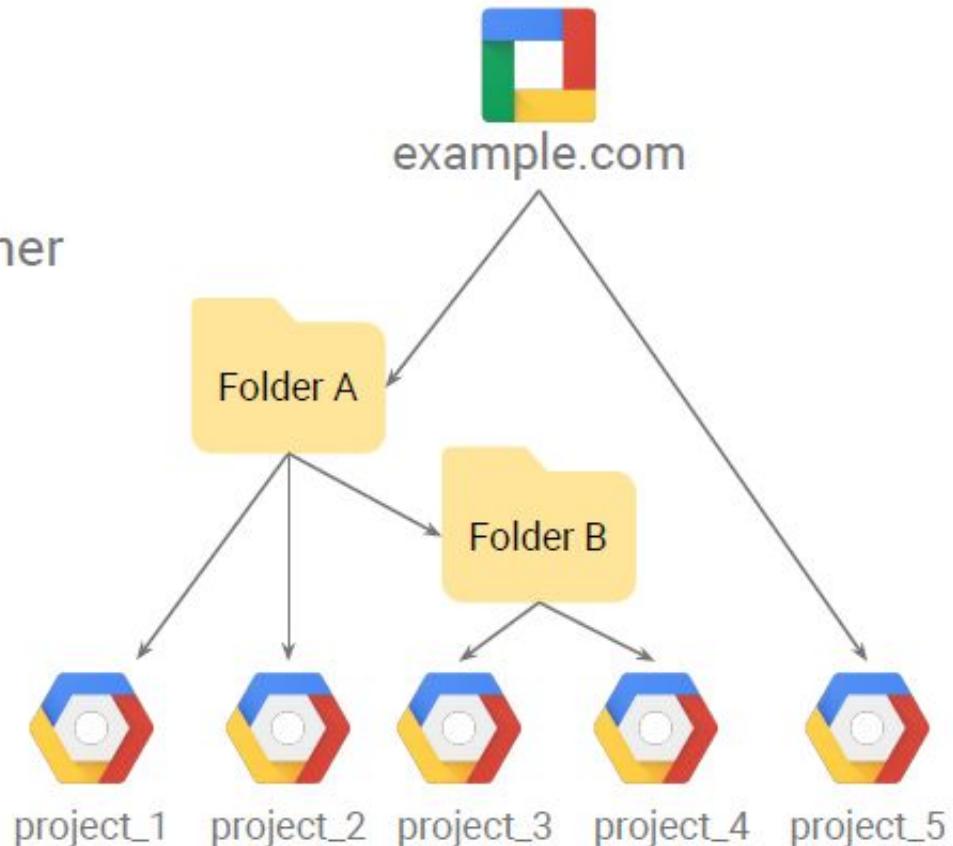


Projects have three identifying attributes

Project ID	Globally unique	Chosen by you	Immutable
Project name	Need not be unique	Chosen by you	Mutable
Project number	Globally unique	Assigned by GCP	Immutable

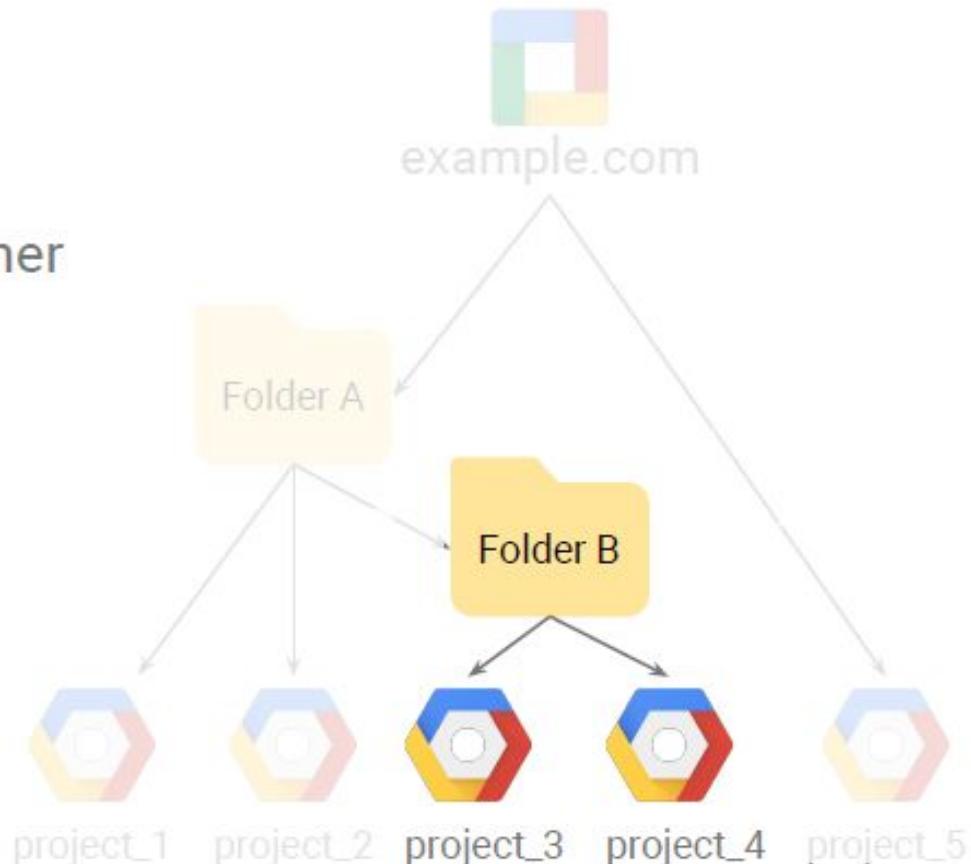
Folders offer flexible management

- Folders group projects under an organization.
- Folders can contain projects, other folders, or both.
- Use folders to assign policies.

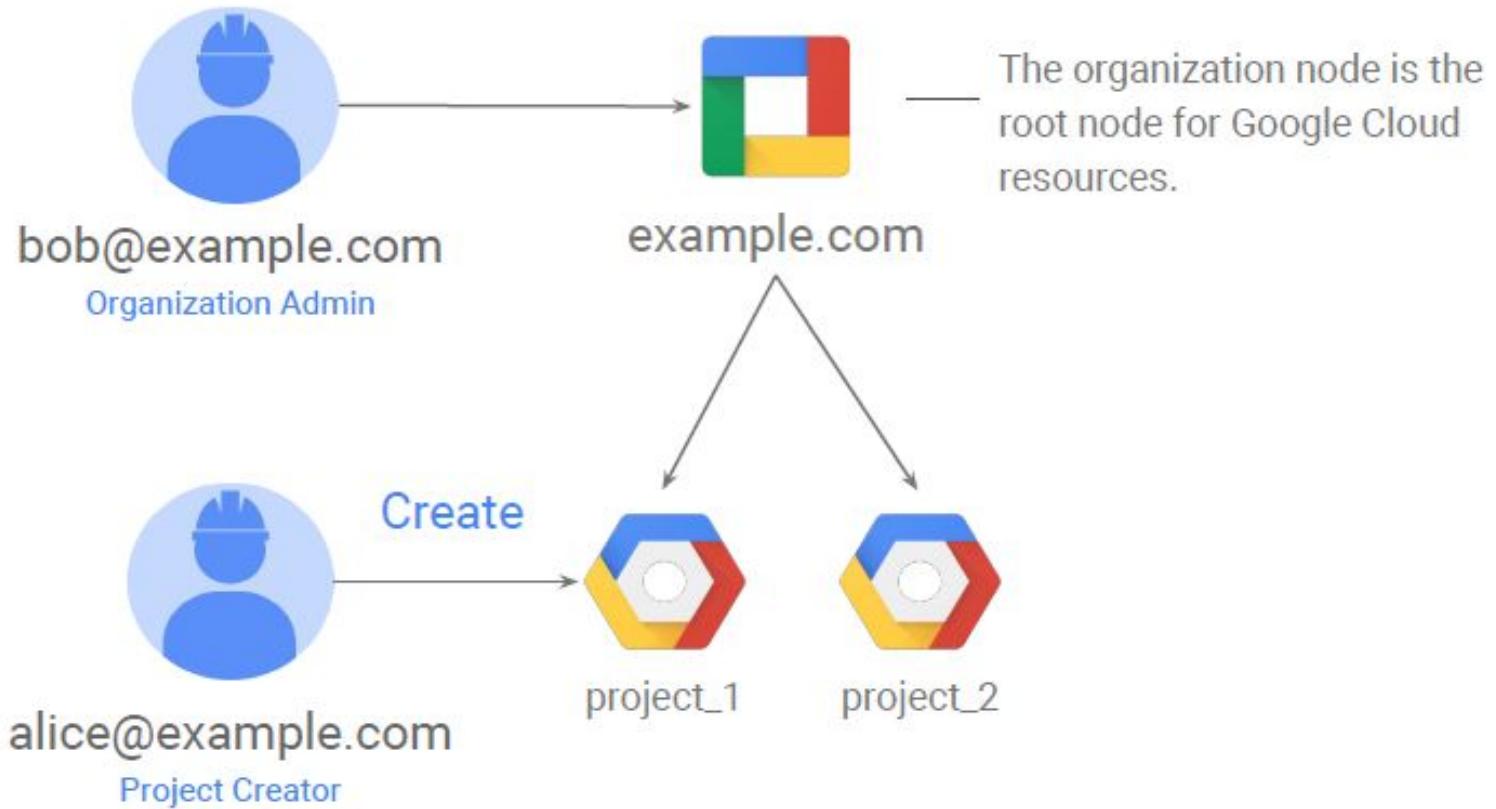


Folders offer flexible management

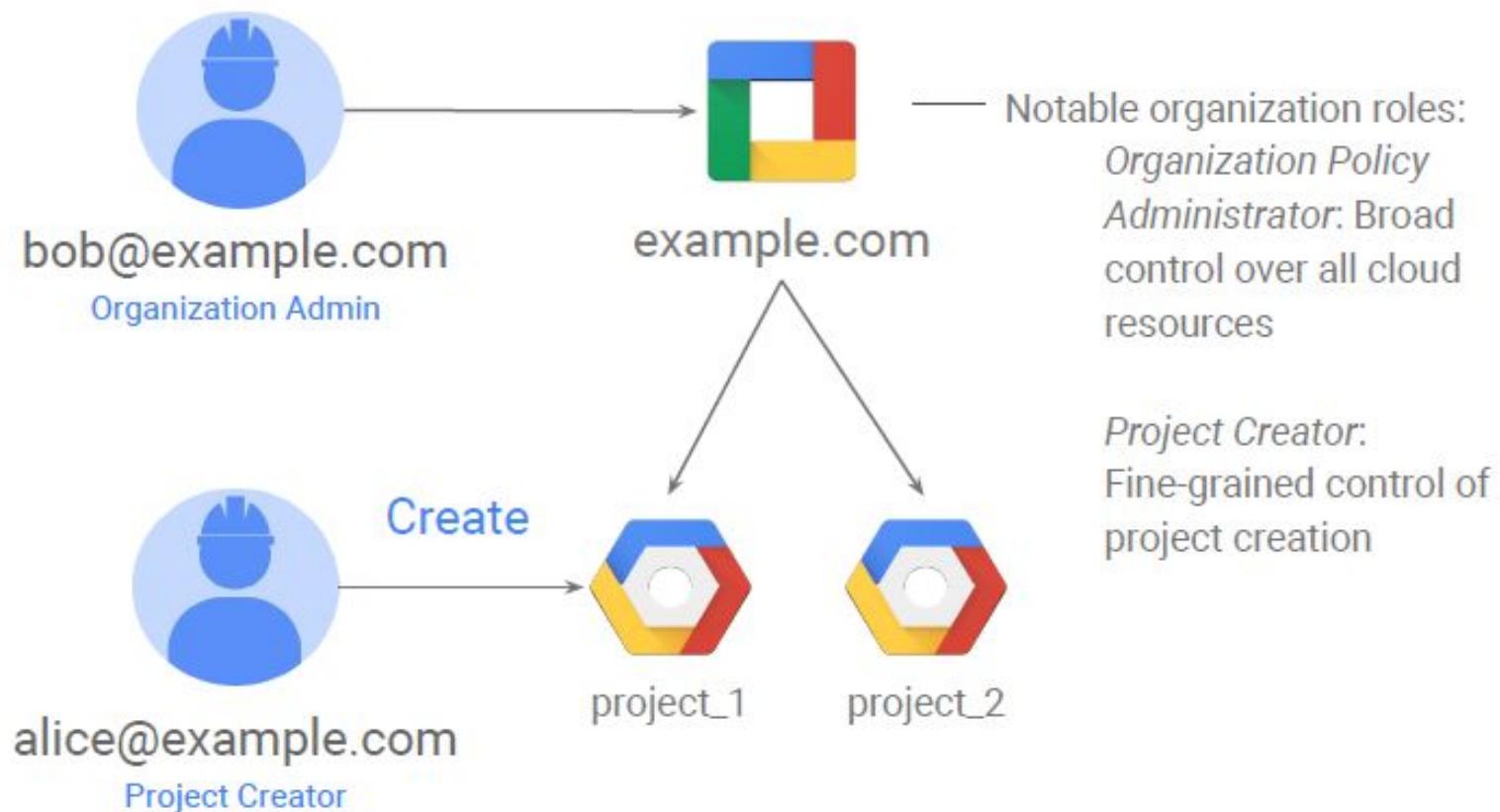
- Folders group projects under an organization.
- Folders can contain projects, other folders, or both.
- Use folders to assign policies.



The organization node organizes projects

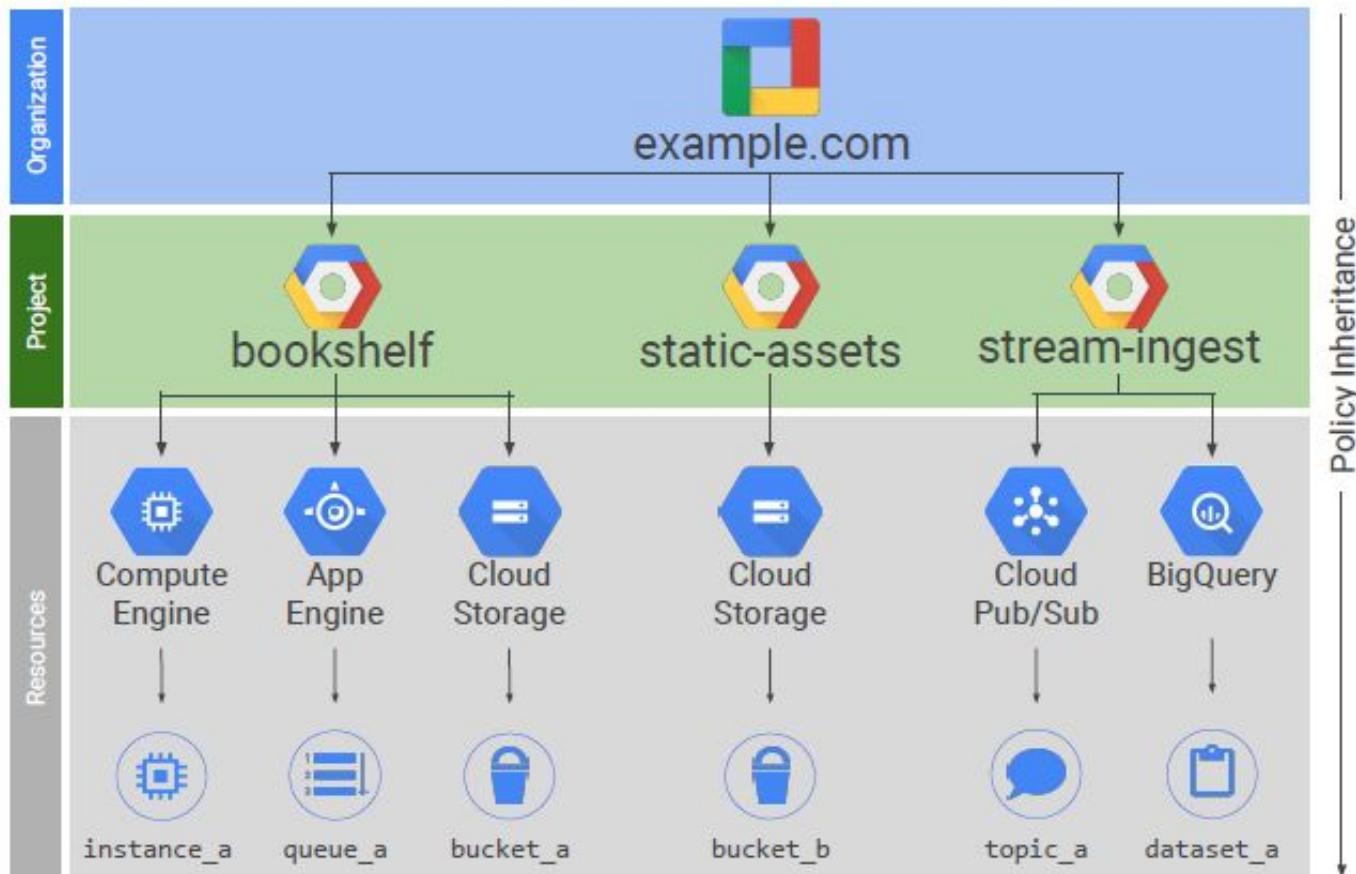


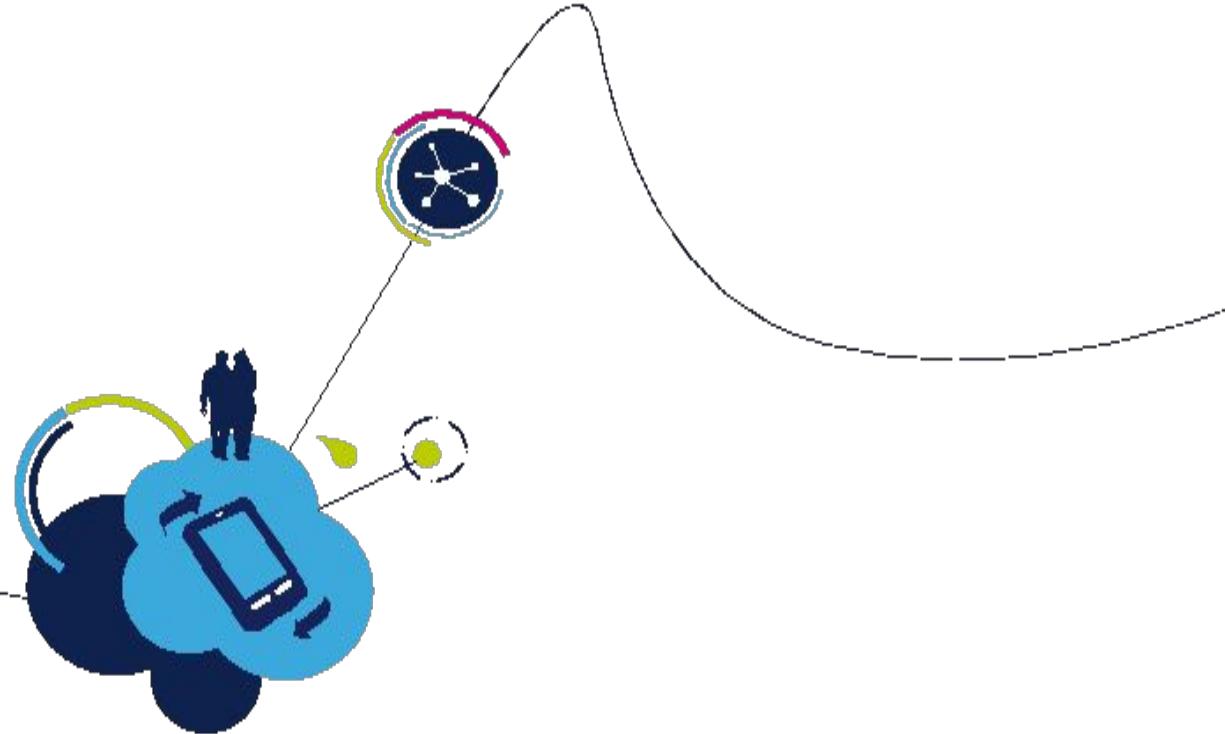
The organization node organizes projects



An example IAM resource hierarchy

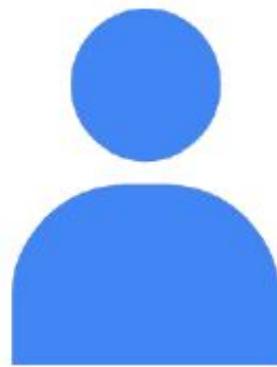
- A policy is set on a resource.
 - Each policy contains a set of roles and role members.
- Resources inherit policies from parent.
 - Resource policies are a union of parent and resource.
- A less restrictive parent policy overrides a more restrictive resource policy.





Identity and Access Management (IAM)

Google Cloud Identity and Access Management defines...



Who

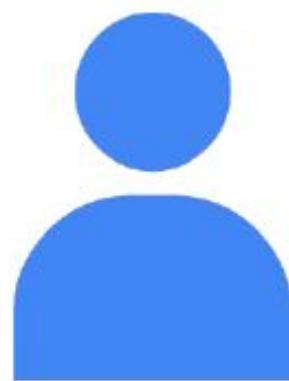


can do what



on which resource

Who: IAM policies can apply to any of four types of principals



Who



Google account or Cloud Identity user
test@gmail.com test@example.com



Service account
test@project_id.iam.gserviceaccount.com



Google group
test@googlegroups.com

G Suite

Cloud Identity or G Suite domain
example.com

Can do what: IAM roles are collections of related permissions



can do what



InstanceAdmin
Role

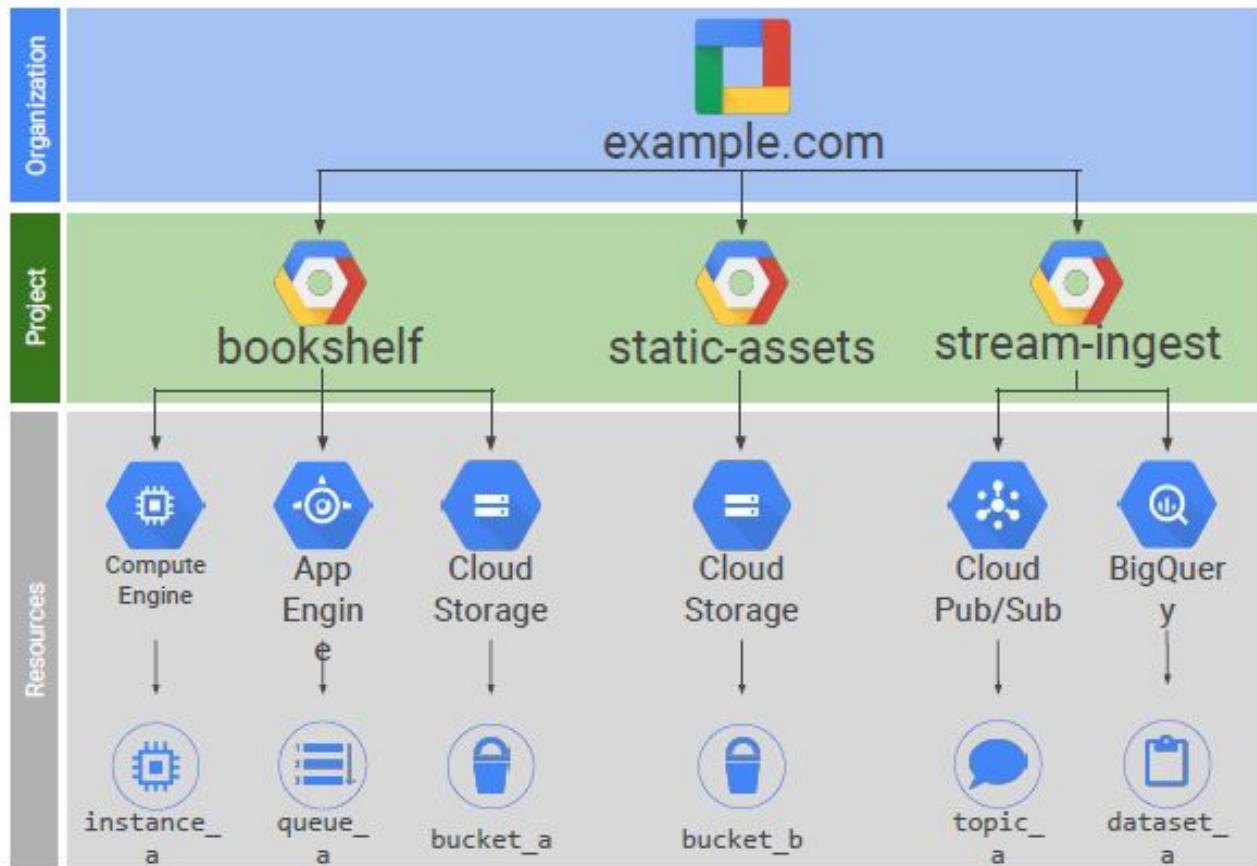


Service	Resource	Verb
compute	instances	list
compute	instances	delete
compute	instances	start
...		

On which resource: Users get roles on specific items in the hierarchy



on which resource

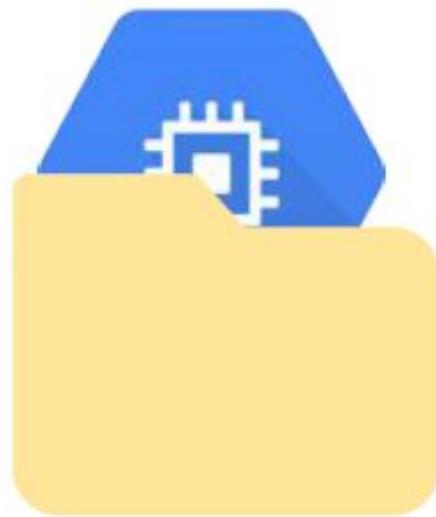


There are three types of IAM roles

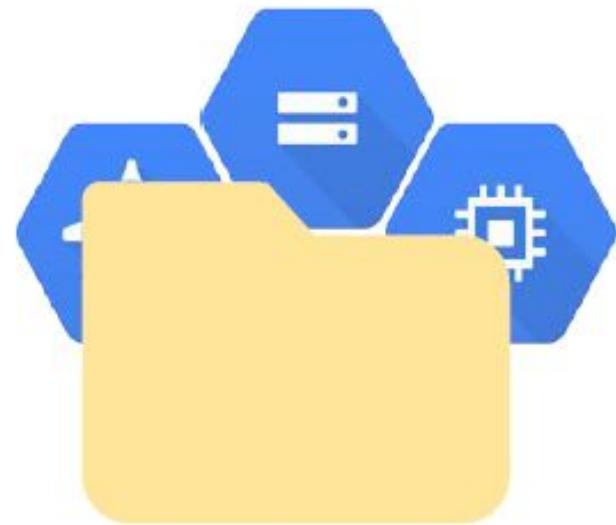
Primitive



Predefined



Custom



IAM **primitive** roles apply across all GCP services in a project



can do what

on all resources

IAM primitive roles offer fixed, coarse-grained levels of access



Owner

- Invite members
- Remove members
- Delete projects
- And...



Editor

- Deploy applications
- Modify code
- Configure services
- And...



Viewer

- Read-only access



Billing administrator

- Manage billing
- Add and remove administrators

A project can have multiple owners, editors, viewers, and billing administrators.

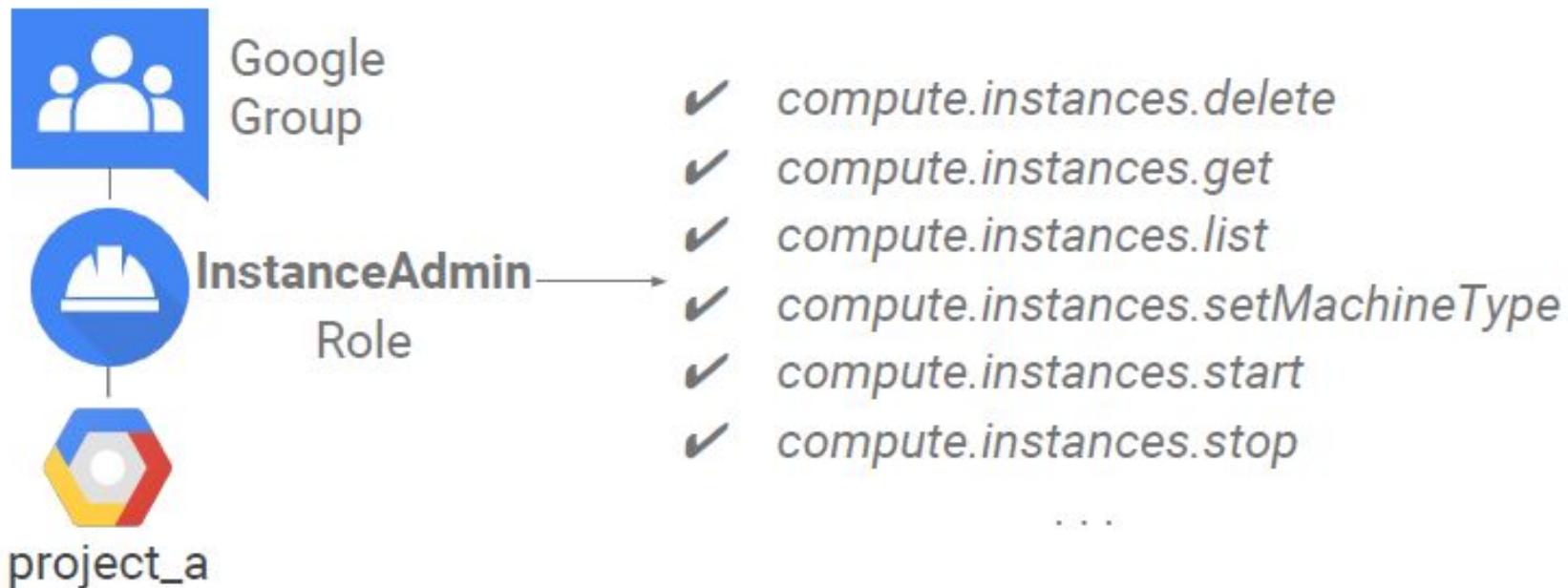
IAM **predefined** roles apply to a particular GCP service in a project



can do what

on Compute Engine resources
in this project, or folder, or org

IAM predefined roles offer more fine-grained permissions on particular services



IAM custom roles let you define a precise set of permissions



Service Accounts control server-to-server interactions

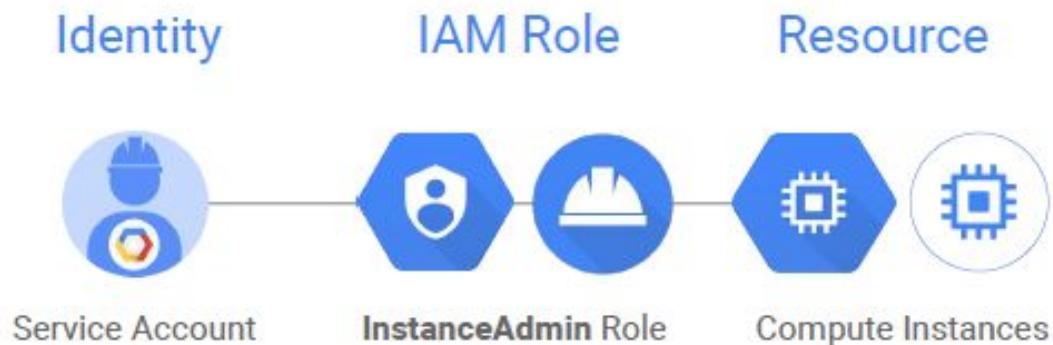
- Provide an identity for carrying out **server-to-server** interactions in a project
- Used to **authenticate** from one service to another
- Used to **control privileges** used by resources
 - So that applications can perform actions on behalf of authenticated end users
- Identified with an **email** address:

PROJECT_NUMBER-compute@developer.gserviceaccount.com

PROJECT_ID@appspot.gserviceaccount.com

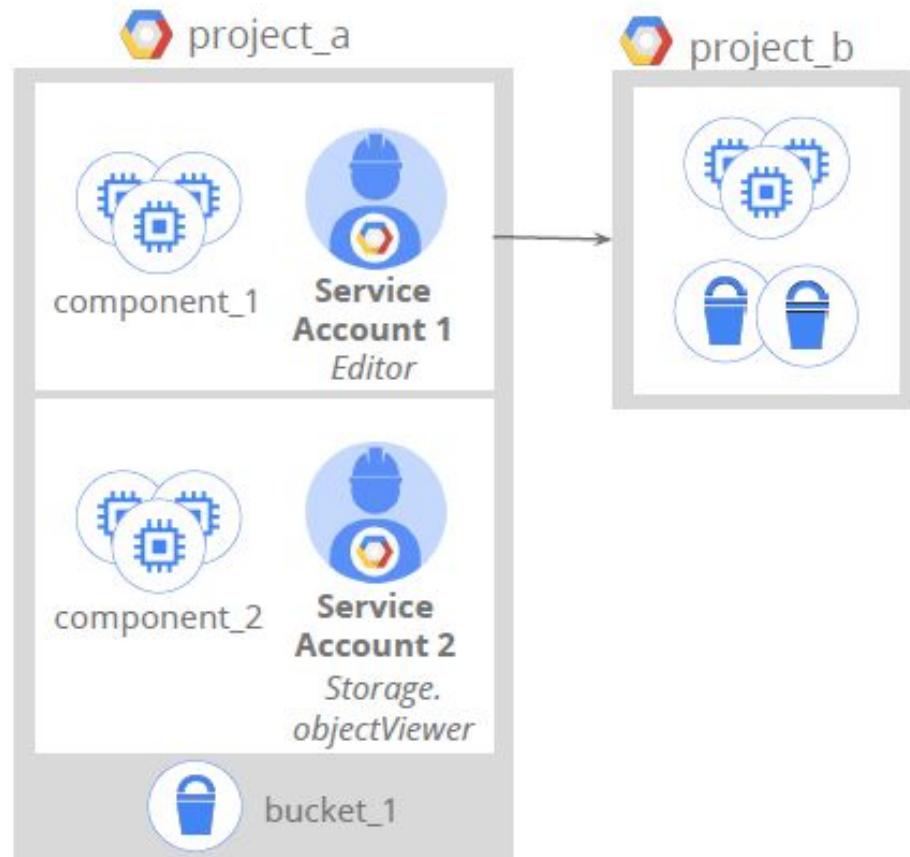
Service Accounts and IAM

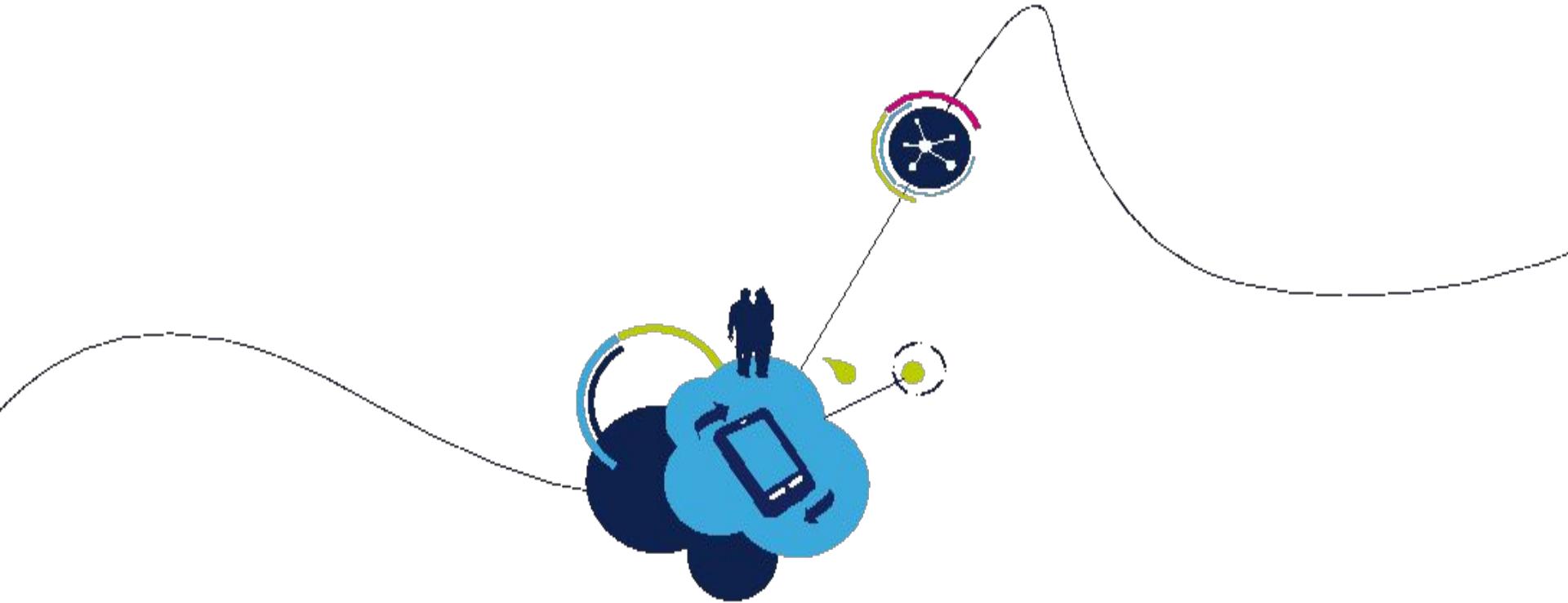
- Service accounts authenticate using keys.
 - Google manages keys for Compute Engine and App Engine.
- You can assign a predefined or custom IAM role to the service account.



Example: Service Accounts and IAM

- VMs running component_1 are granted **Editor** access to project_b using *Service Account 1*.
- VMs running component_2 are granted **objectViewer** access to bucket_1 using *Service Account 2*.
- Service account permissions can be changed without recreating VMs.





Interacting with Google Cloud Platform

There are four ways to interact with GCP

**Cloud Platform
Console**

Web user
interface



**Cloud Shell and
Cloud SDK**

Command-line
interface



**Cloud Console
Mobile App**

For iOS and
Android



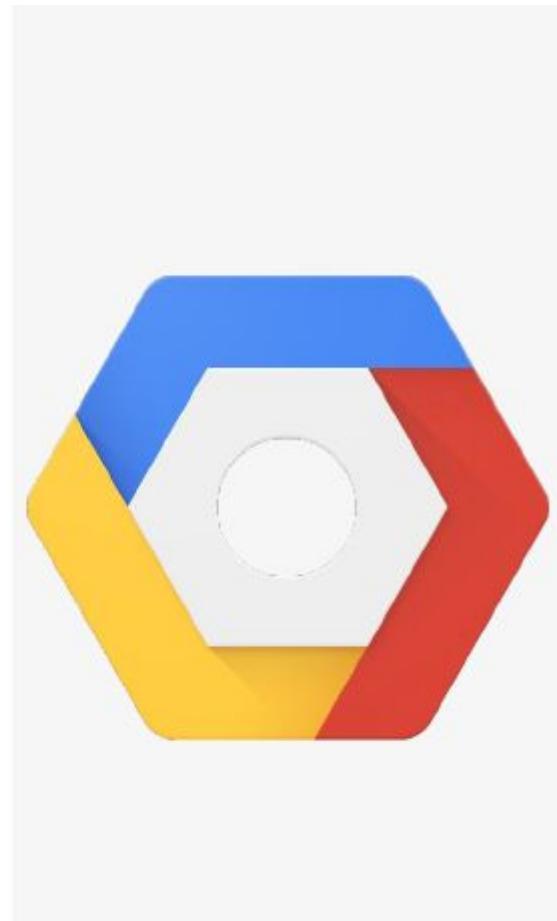
REST-based API

For custom
applications



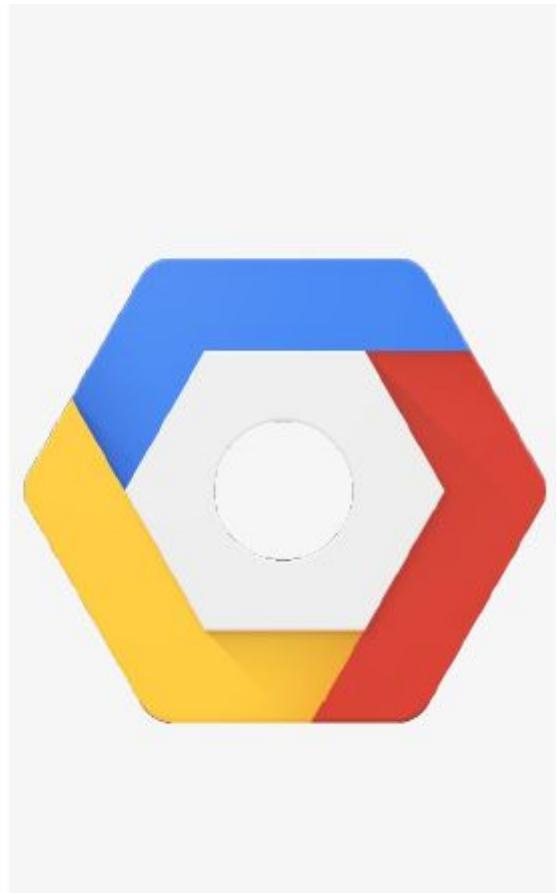
Google Cloud Platform Console

- Centralized console for all project data
- Developer tools
 - Cloud Source Repositories
 - Cloud Shell
 - Test Lab (mobile app testing)
- Access to product APIs
- Manage and create projects



Google Cloud SDK

- SDK includes CLI tools for Cloud Platform products and services
 - gcloud, gsutil (Cloud Storage), bq (BigQuery)
- Available as Docker image
- Available via Cloud Shell
 - Containerized version of Cloud SDK running on Compute Engine instance



RESTful APIs

- Programmatic access to products and services
 - Typically use JSON as an interchange format
 - Use OAuth 2.0 for authentication and authorization
- Enabled through the Google Cloud Platform Console
- To help you control spend, most include daily quotas and rates (limits)
 - Quotas and rates can be raised by request

Use APIs Explorer to help you write your code

- The [APIs Explorer](#) is an interactive tool that lets you easily try Google APIs using a browser.
- With the APIs Explorer, you can:
 - Browse quickly through available APIs and versions.
 - See methods available for each API and what parameters they support along with inline documentation.
 - Execute requests for any method and see responses in real time.
 - Easily make authenticated and authorized API calls.

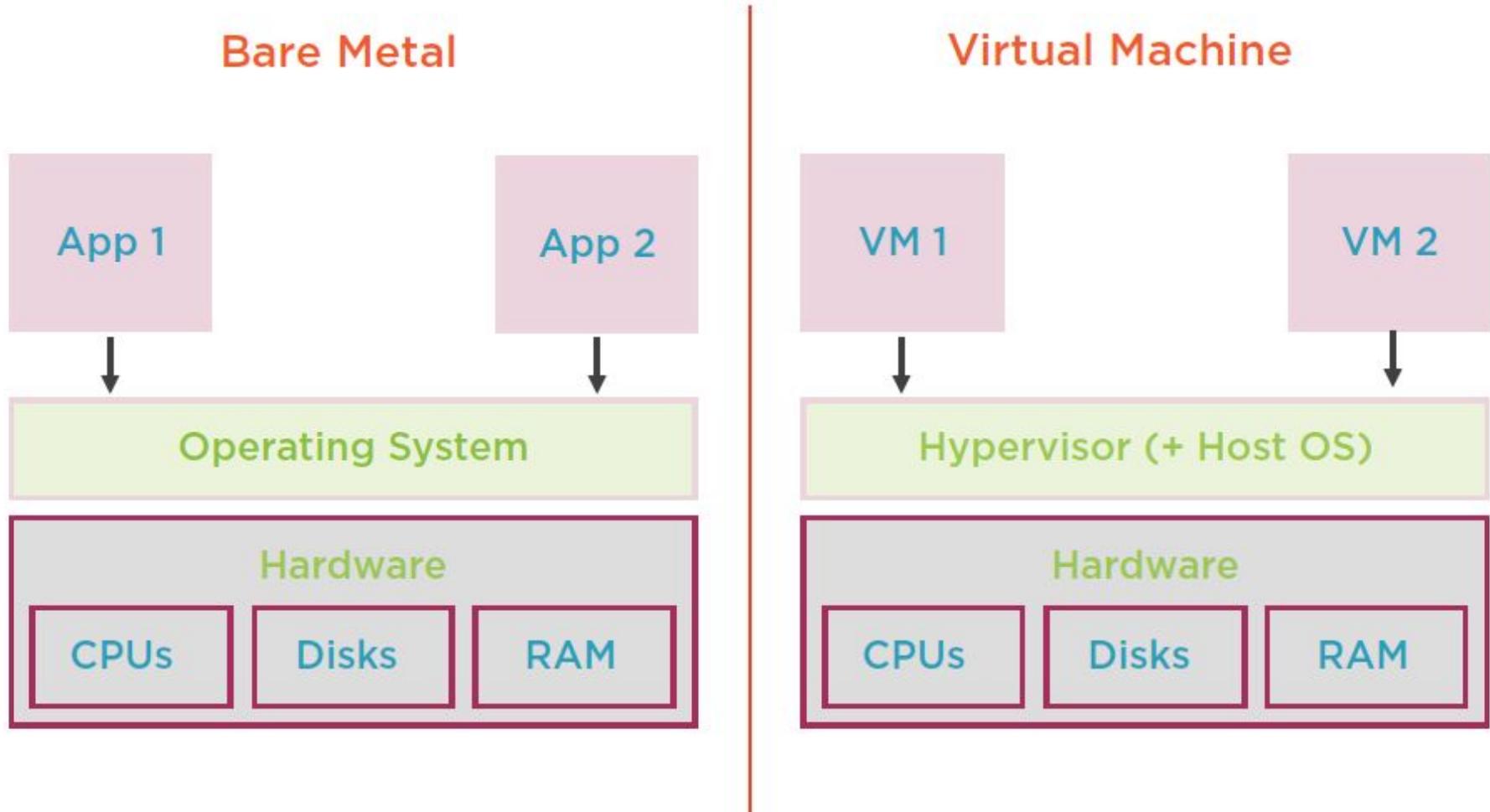
Use client libraries to control GCP resources from within your code

- [Cloud Client Libraries](#)
 - Community-owned, hand-crafted client libraries
- [Google API Client Libraries](#)
 - Open source, generated
 - Support various languages
 - Java, Python, JavaScript, PHP, .NET, Go, Node.js, Ruby, Objective-C, Dart

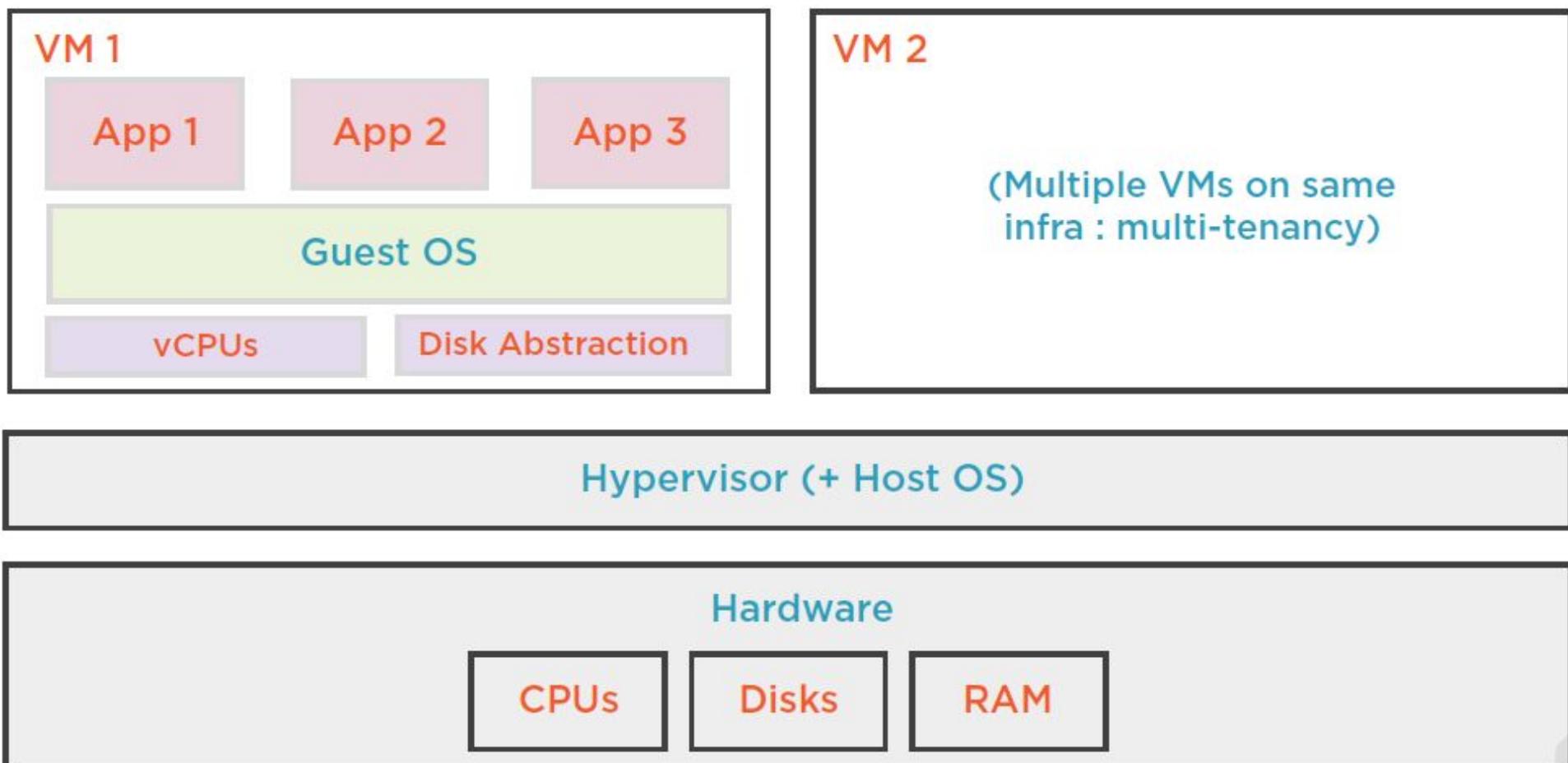
Google Kubernetes Engine (GKE)

Introducing Containers

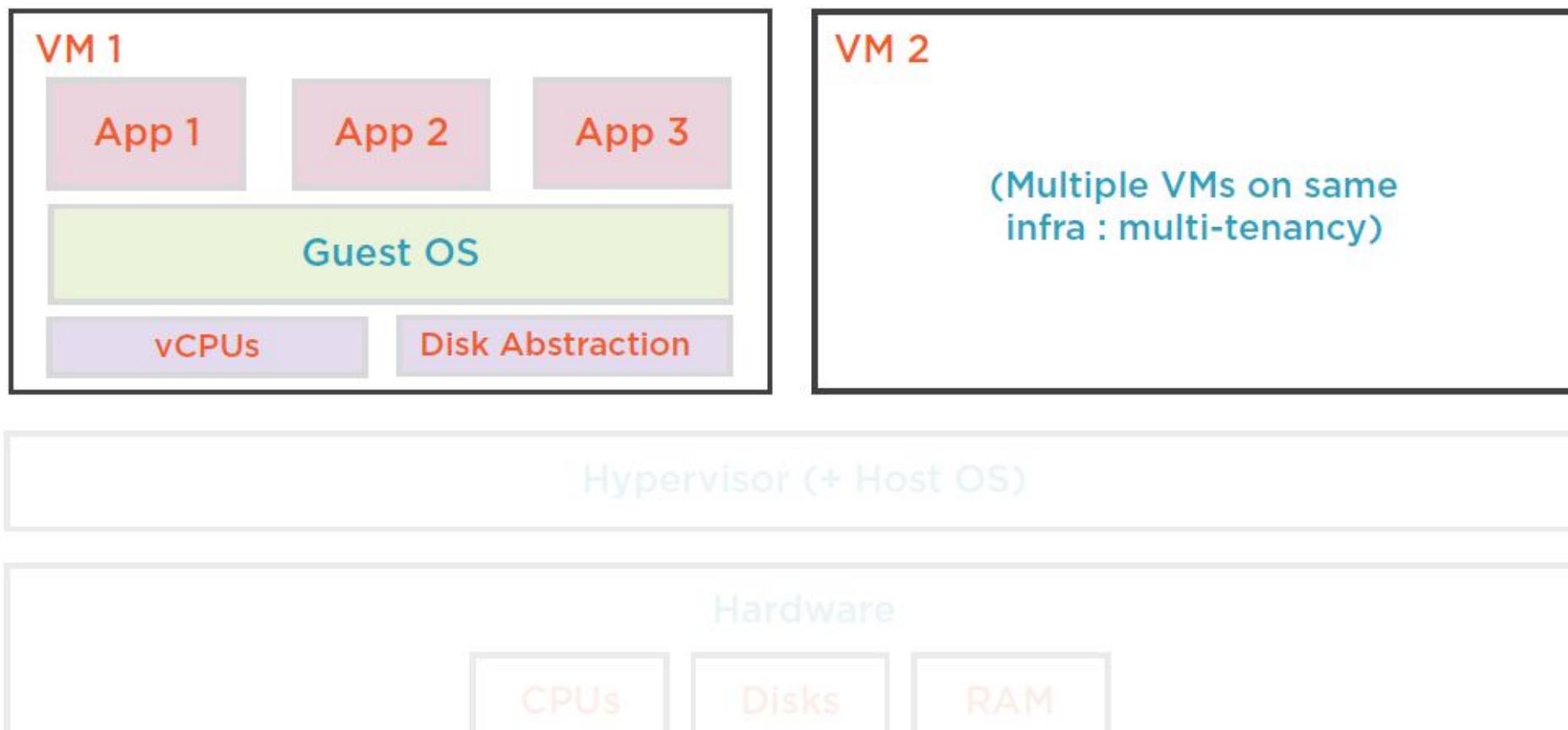
Bare Metal and Virtual Machines



Apps on Virtual Machines

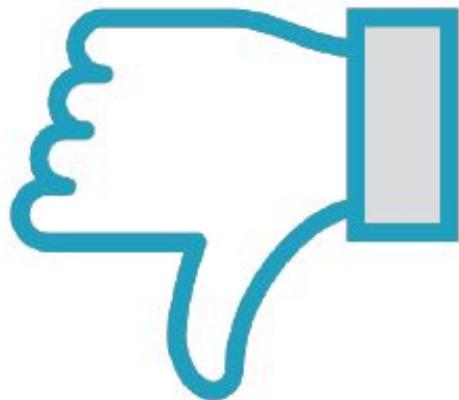


Apps on Virtual Machines



VMs are far more lightweight than bare metal environments and are often used to run applications

Drawbacks of VMs



Contain guest OS

- Introduces platform dependency
- Bloats image size to GB (apps far smaller)

Heavyweight

- Slow to boot up

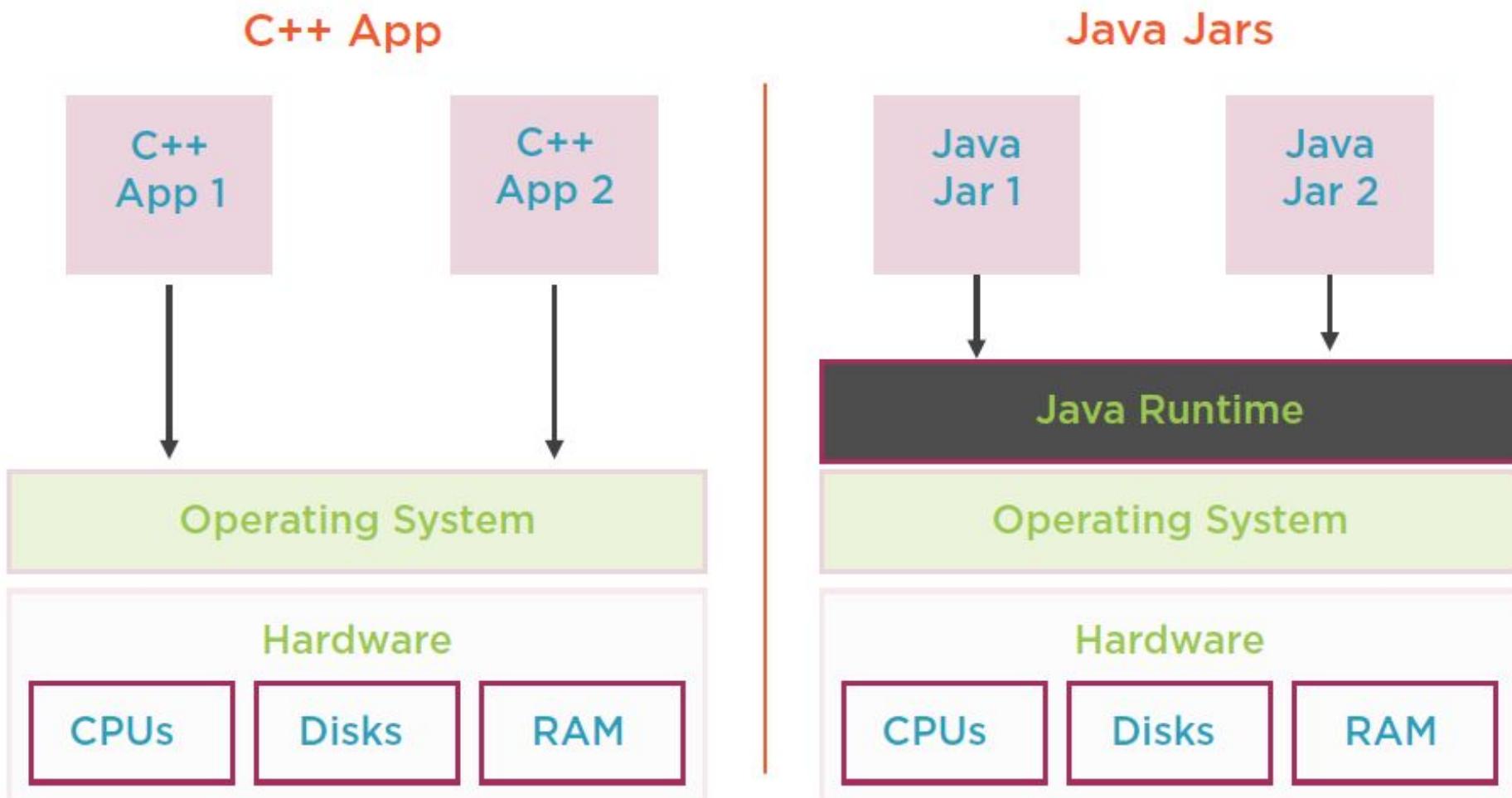
Not trivial to migrate

- VM migration tools needed

Container

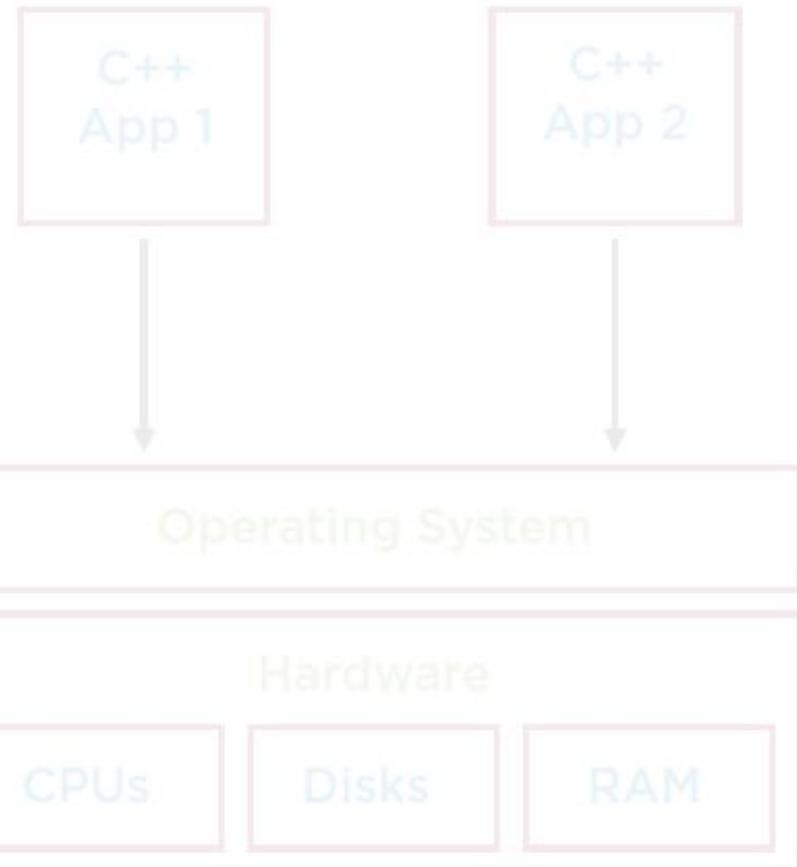
A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings

Containers Are ‘Like’ Jars

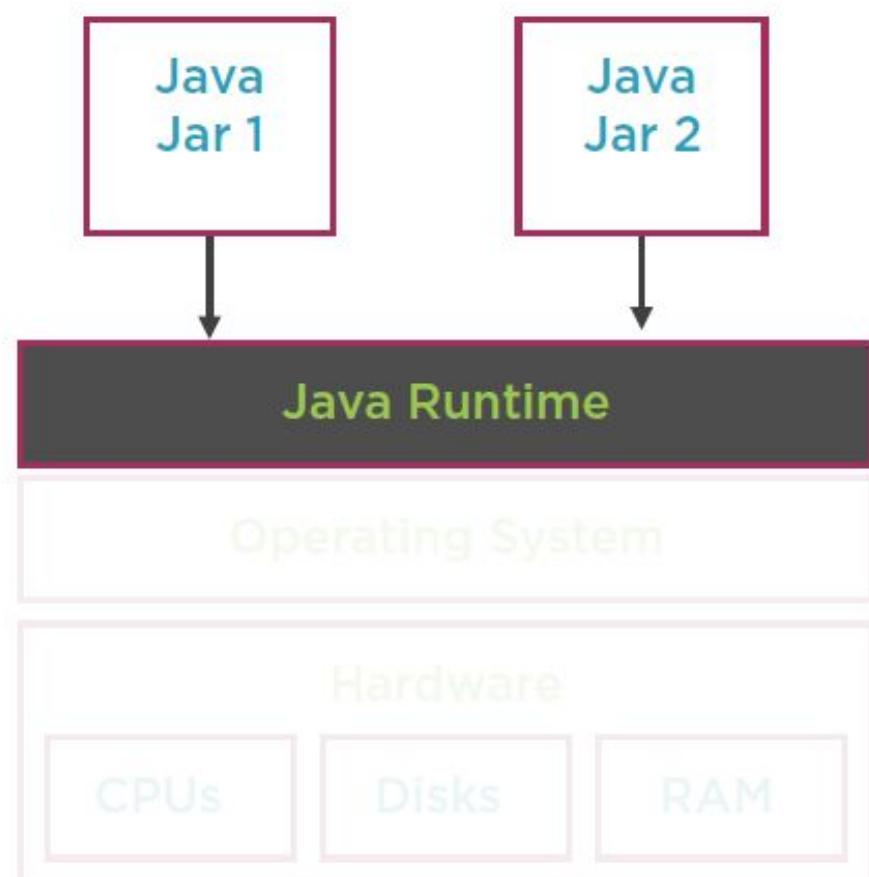


Containers Are ‘Like’ Jars

C++ App

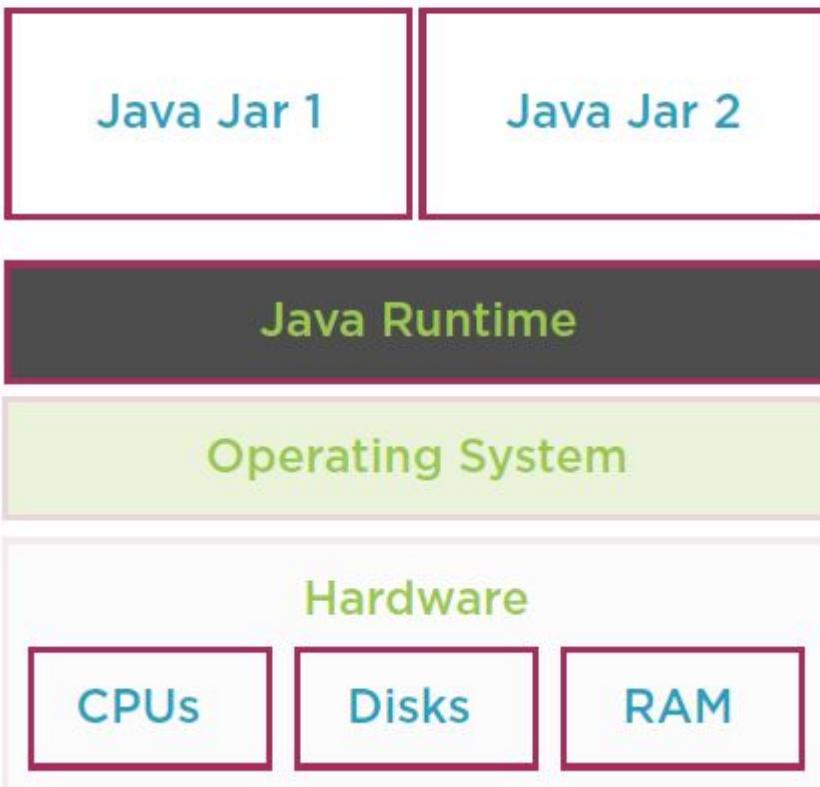


Java Jars

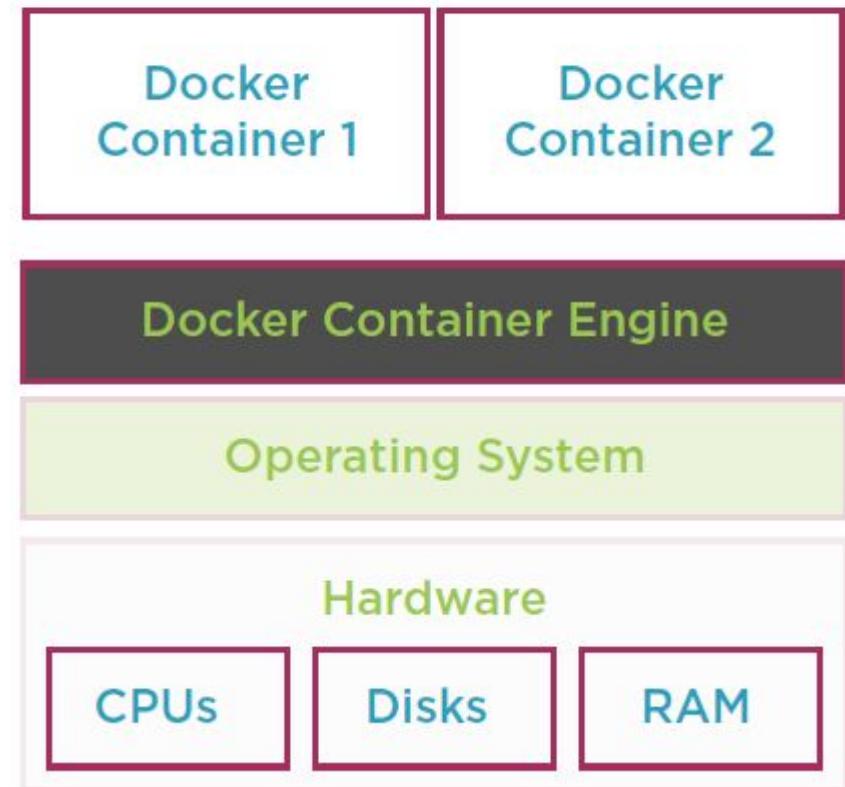


Containers Are ‘Like’ Jars

Java Jars



Docker Containers



Java Jars and Containers

Java Jar Files

- Files containing apps
- Platform independent
- Run on layer of abstraction

Java Runtime

Containers

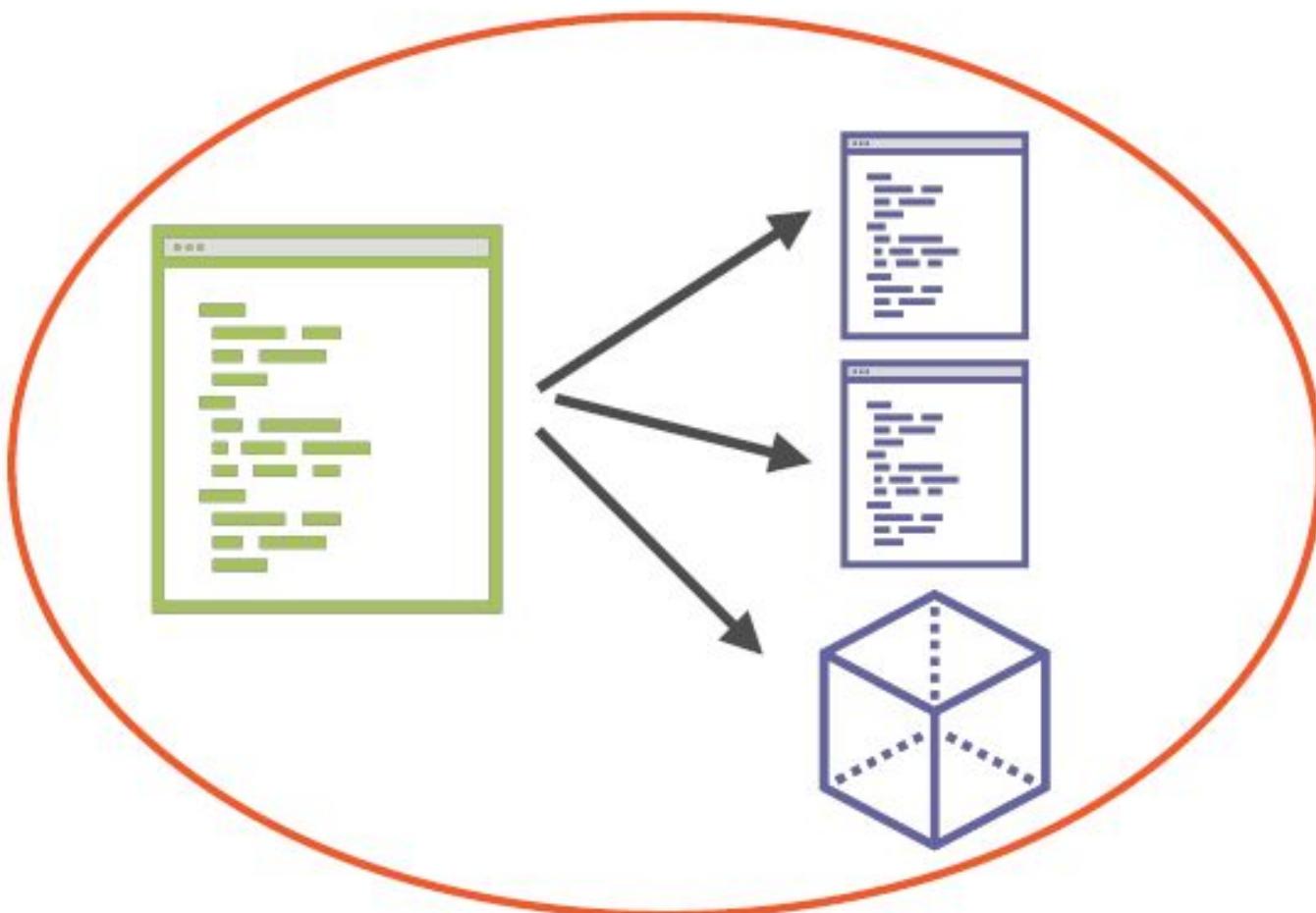
- Files containing apps
- Platform independent
- Run on layer of abstraction

Docker Runtime

Container

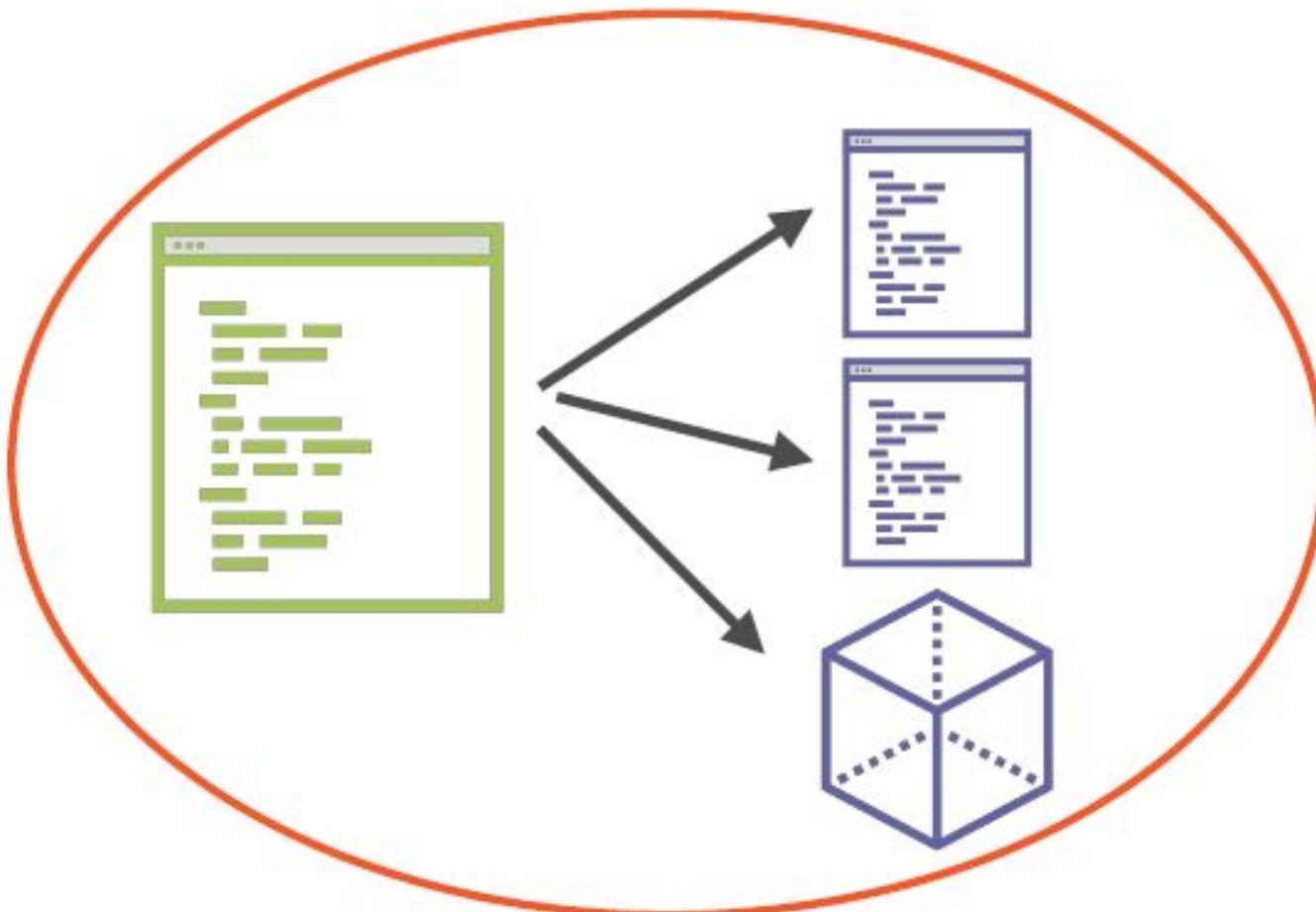
A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings

Docker Containers



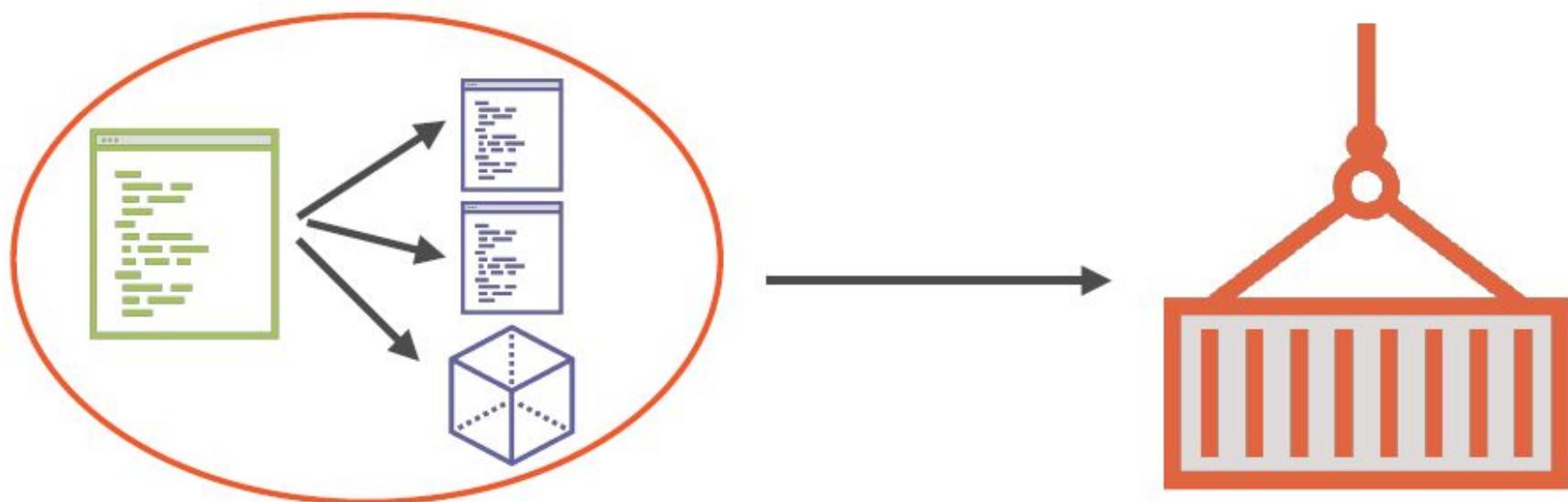
A container packages code and its dependencies into an image

Docker Containers



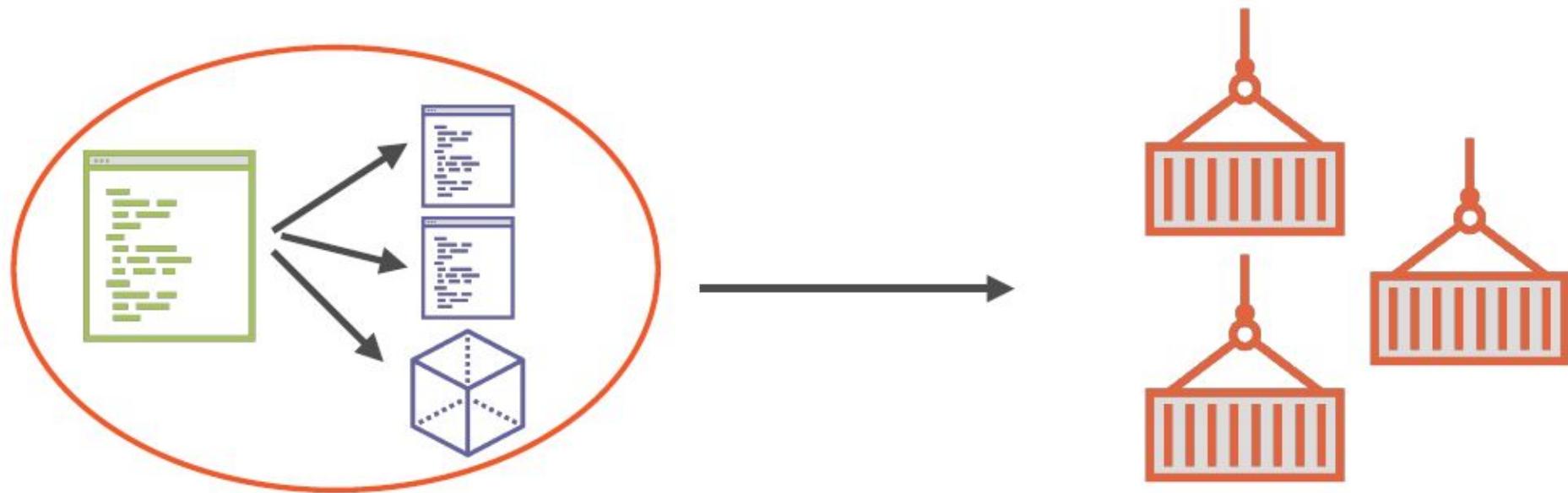
A fully self-contained environment

Docker Containers



Code is abstracted away from the machine

Docker Containers



Create as many containers as you want
from the same image

Introducing Kubernetes



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



Standalone Container Limitations

No autohealing

- Crashed containers won't restart automatically
- Need higher level orchestration

No scaling or autoscaling

- Overloaded containers don't spawn more automatically
- Need higher level orchestration

No load balancing

- Containers can't share load automatically
- Need higher level orchestration

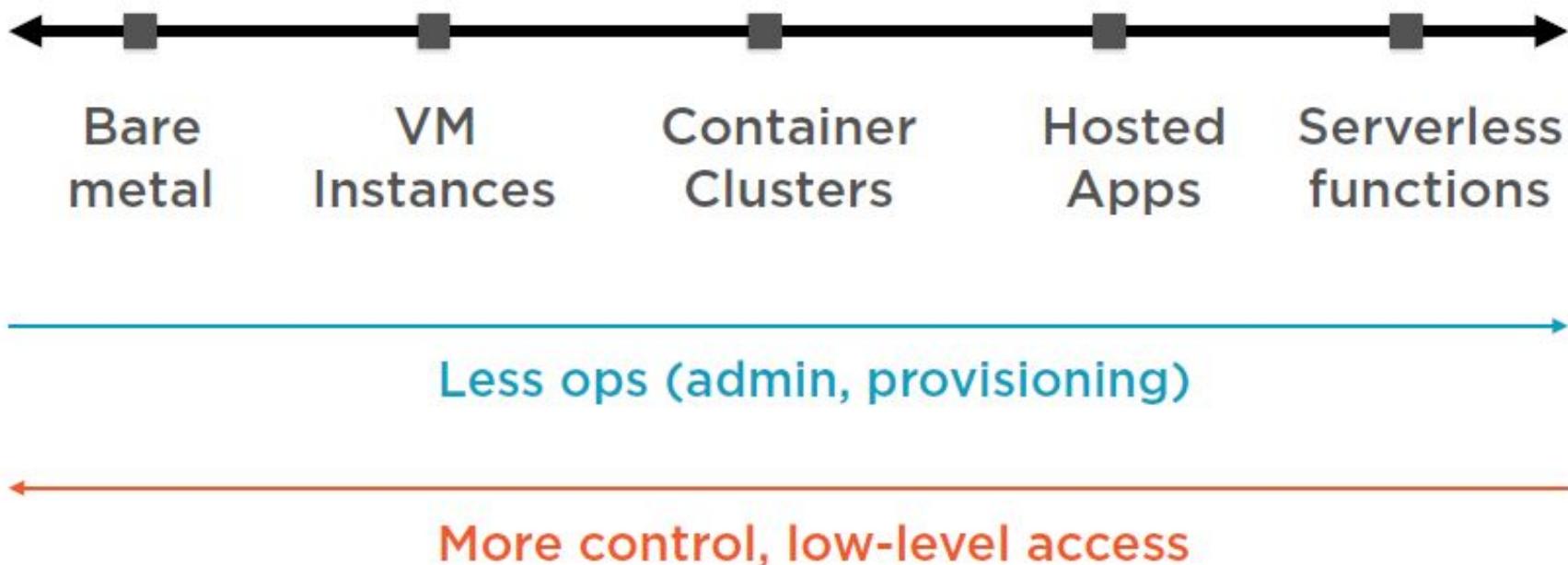
No isolation

- Crashing containers can take each other down
- Need sandbox to separate them

Kubernetes

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

Compute Choices



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



Hybrid, Multi-cloud

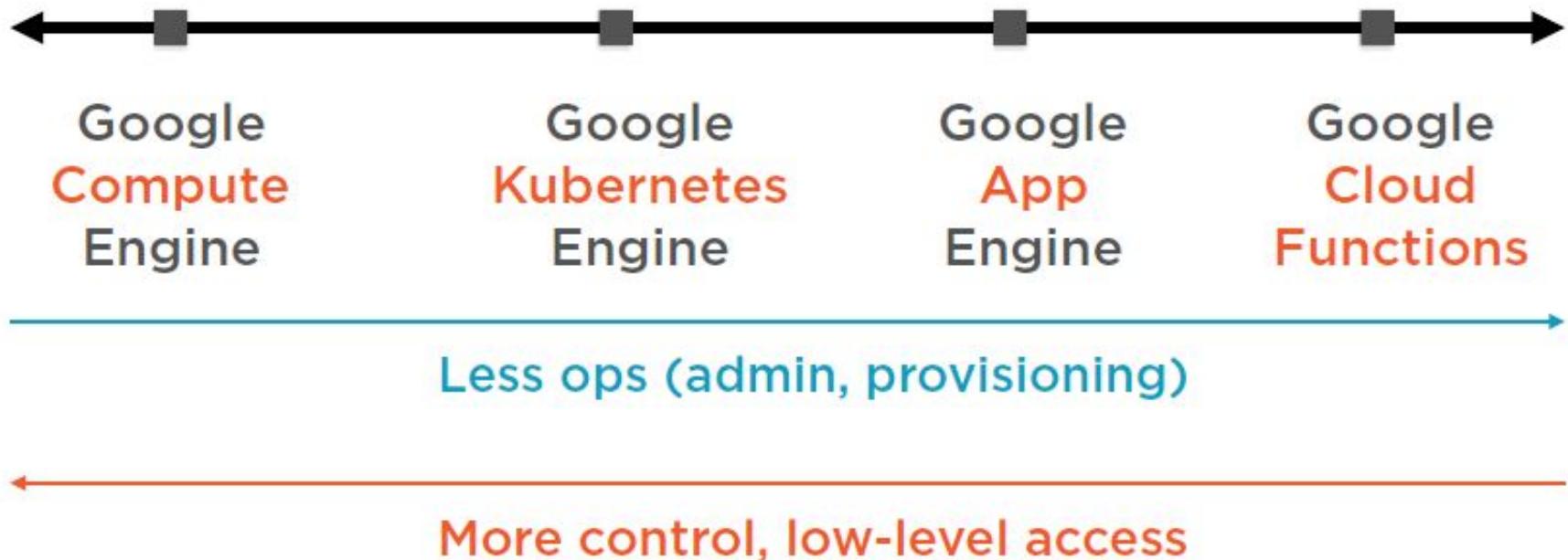
Hybrid: Runs on-premise and on cloud

- Provides smooth migration path

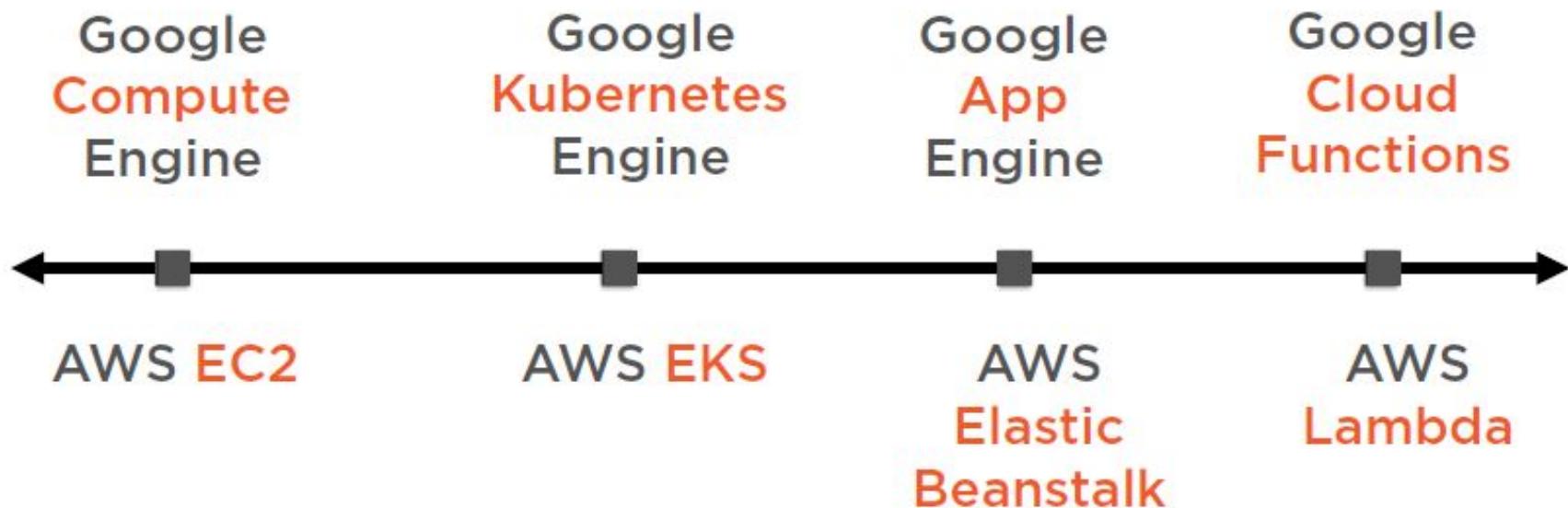
Multi-cloud: Supported by all big cloud platforms

- Important for strategic independence
- Amazon-Whole Foods merger

GCP Compute Choices



GCP Compute Choices



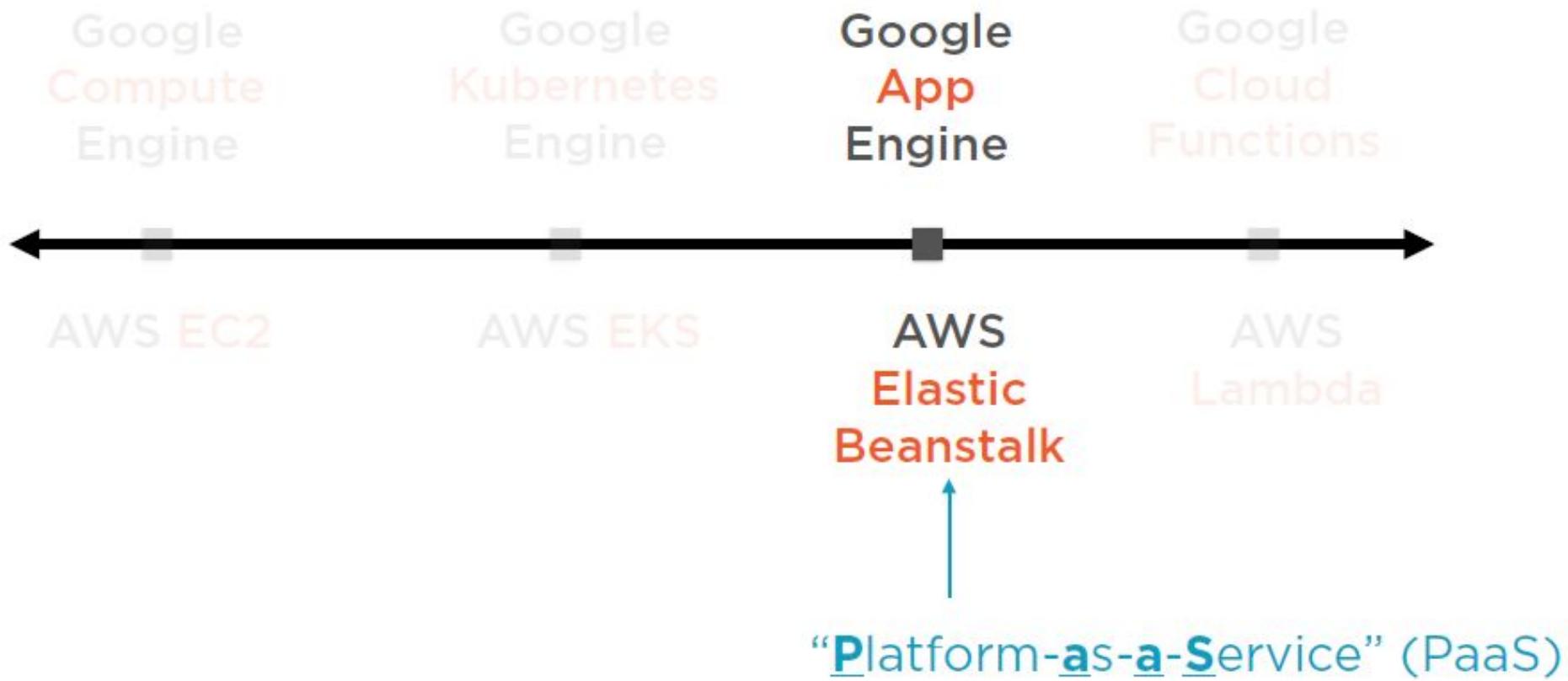
Every major cloud platform supports the same range of compute choices

GCP Compute Choices



“Infrastructure-as-a-Service” (IaaS)

GCP Compute Choices



IaaS vs. PaaS

Infrastructure-as-a-Service

Heavy operational burden

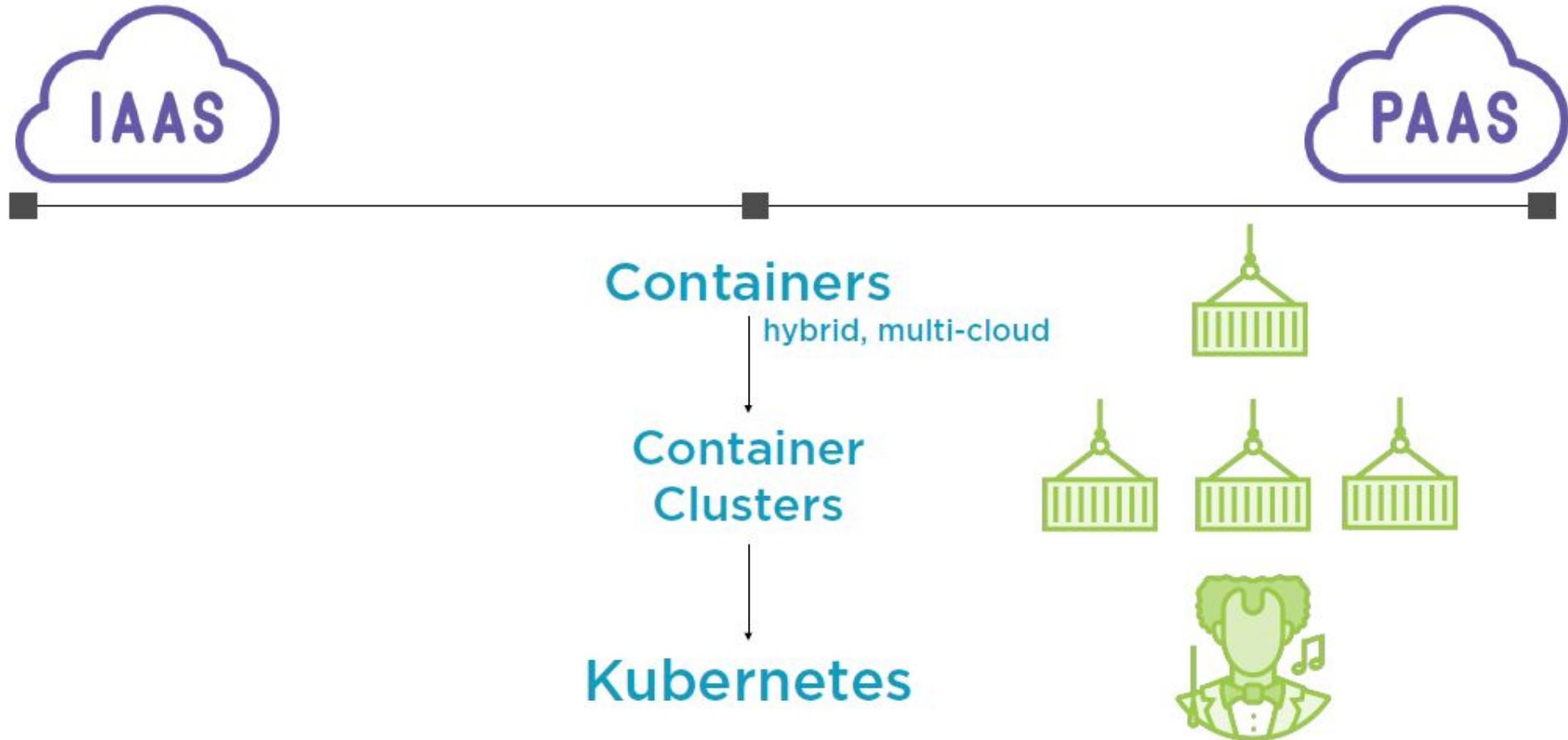
Migration is hard

Platform-as-a-Service

Provider lock-in

Migration is very hard

Compute Choices





Kubernetes as Orchestrator

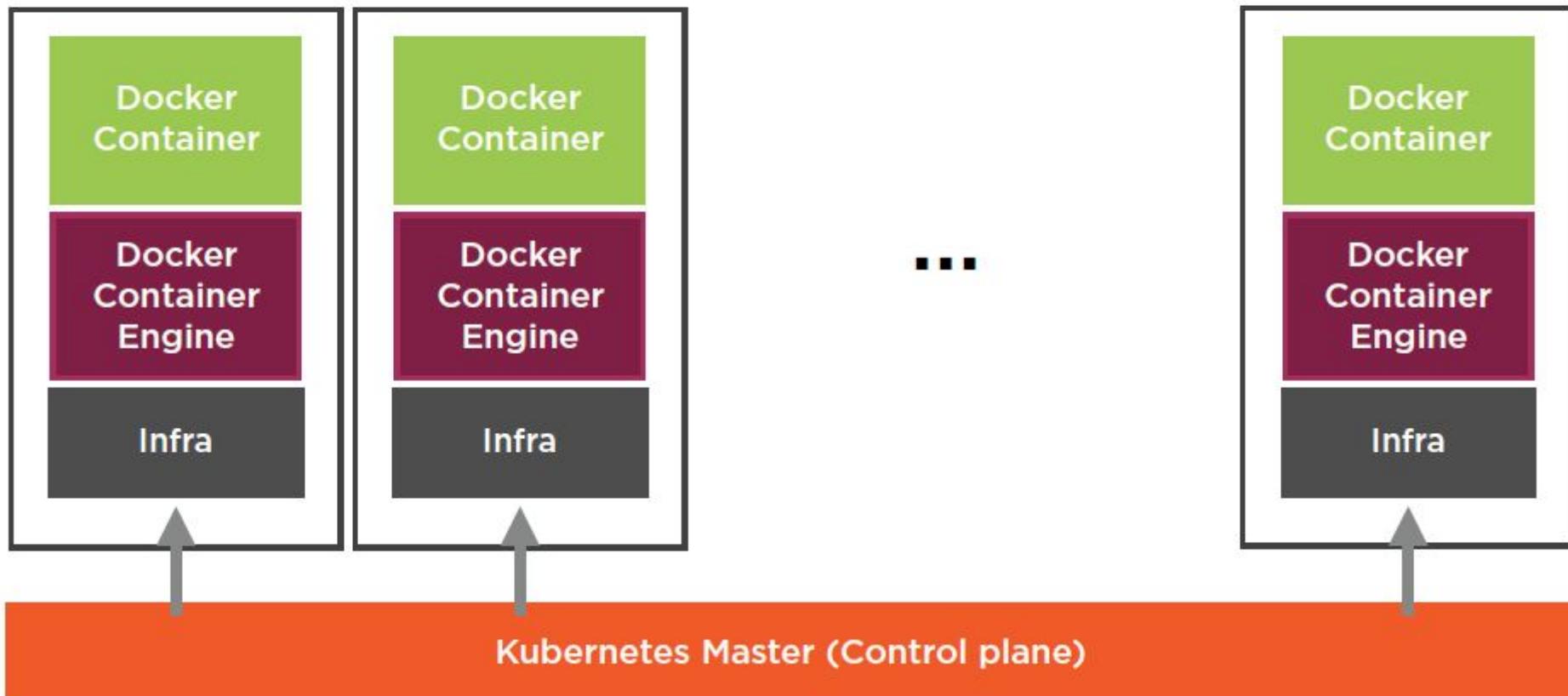
- Fault-tolerance**
- Autohealing**
- Isolation**
- Scaling**
- Autoscaling**
- Load balancing**

Kubernetes: Cluster Orchestration

Node 1

Node 2

Node N



Kubernetes: Cluster Orchestration

Node 1

Node 2

Node N

Docker Container

Docker Container Engine

Infra

Docker Container

Docker Container Engine

Infra

Docker Container

Docker Container Engine

Infra

...

Kubernetes Master (Control plane)

Kubernetes: Containers Run Within Pods

Node 1



Node 2



...

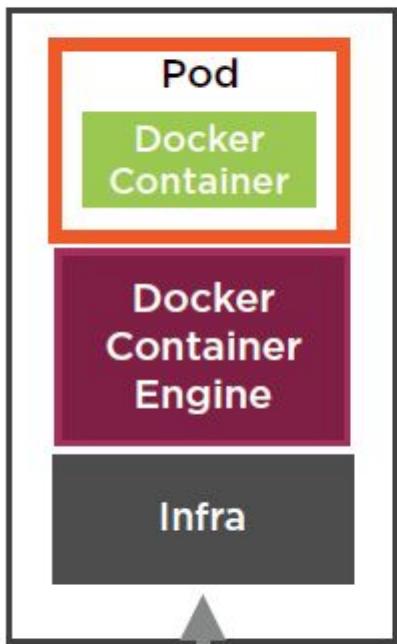
Node N



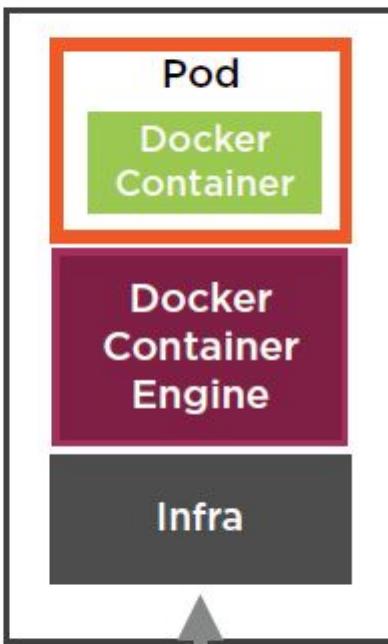
Kubernetes Master (Control plane)

Kubernetes: Cluster Orchestration

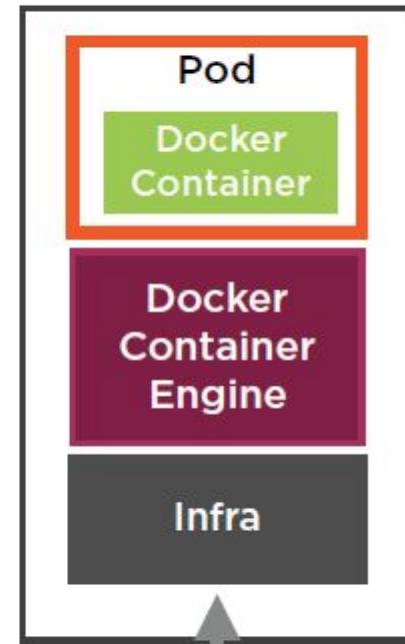
Node 1



Node 2



Node N



...

Kubernetes Master (Control plane)

Pods as Atomic Units

Container deployment

All containers in pod are deployed,
or none are

Node association

Entire pod is hosted on the same
node

Pod is atomic unit of deployment in Kubernetes

The ReplicaSet Object

**Multiple identical pods which
are replicas of each other**

ReplicaSet

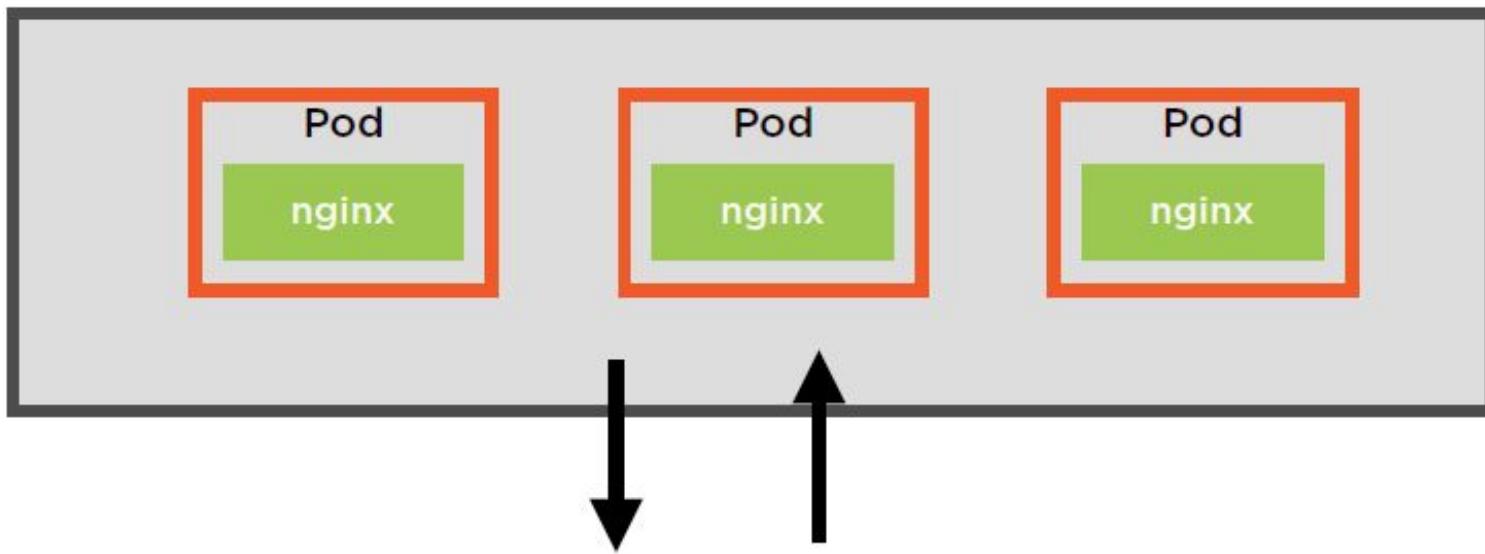


Self healing and autoscaling for our pods

The Deployment Object

**Adds on deployment and
rollback functionality**

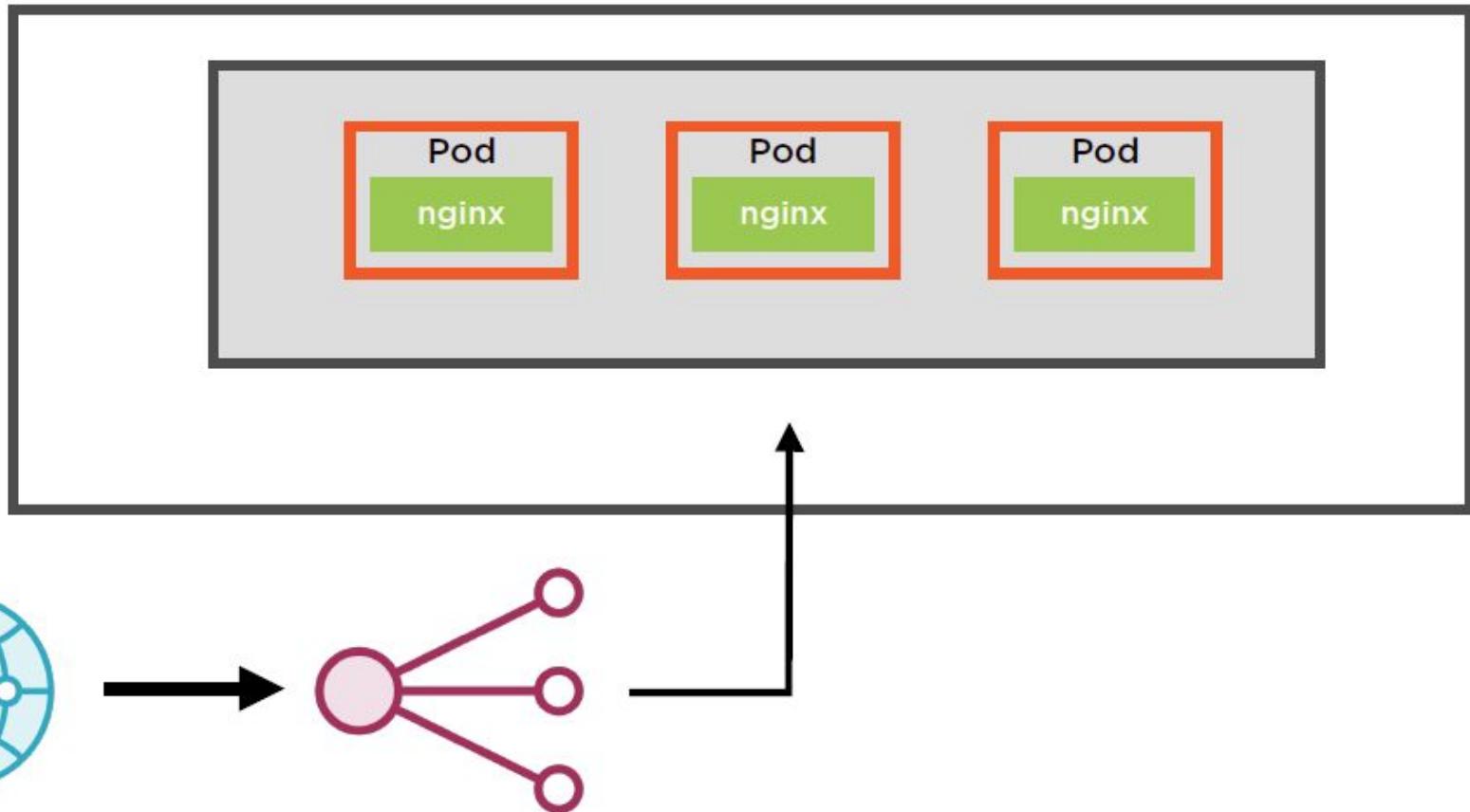
Deployment



Support for versions, and production-level operations such as rollbacks

Services provide stable IP
addresses for external
connections and load balancing

Service



Introducing Google Kubernetes Engine

A GKE Cluster

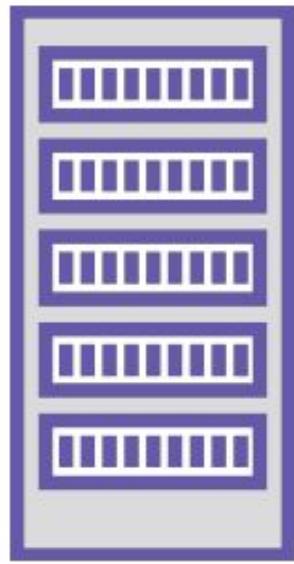
**Made up of nodes, arranged in
node pools, running container
optimized node images**



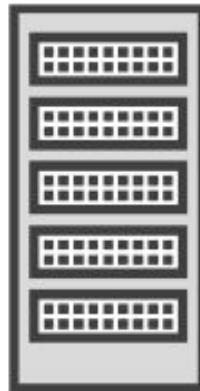
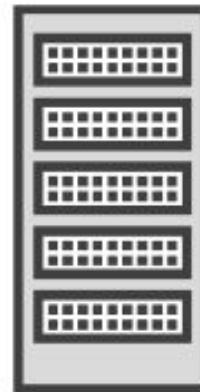
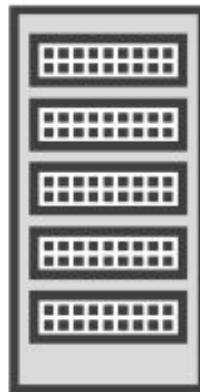
Google Kubernetes Engine (GKE)

Google Kubernetes Engine
Service for working with Kubernetes clusters on GCP
Runs Kubernetes on GCE VM instances

Kubernetes Clusters



Master node



Worker nodes



Master

One or more nodes designated master nodes

Managed by GKE

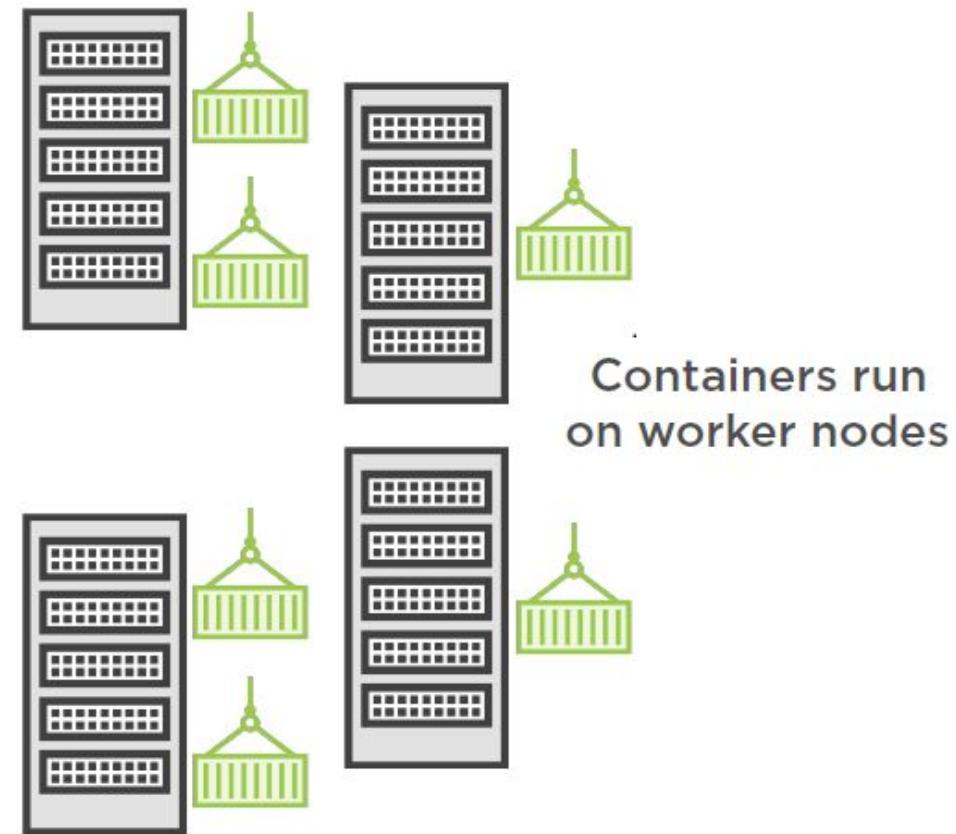
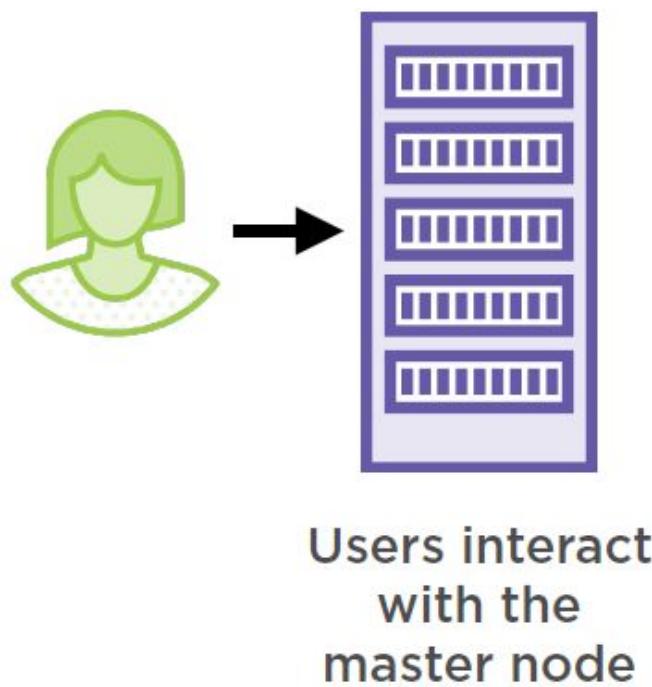
Not visible directly to user

Multi-master for high-availability

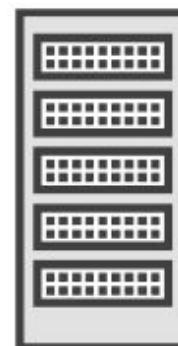
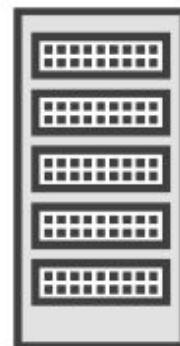
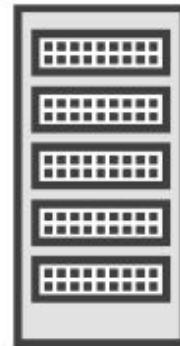
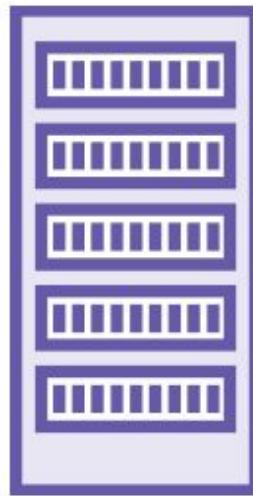
Kubernetes Control Plane directed from here

All user interactions with Kubernetes clusters are via the **kube-apiserver** running on the master node

Kubernetes Clusters

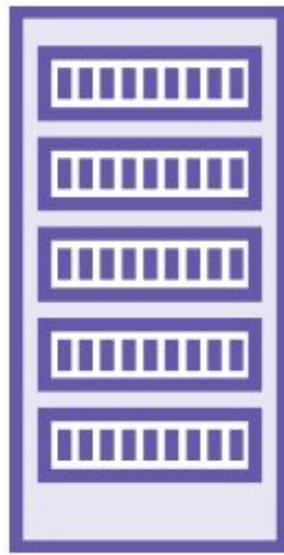


Nodes

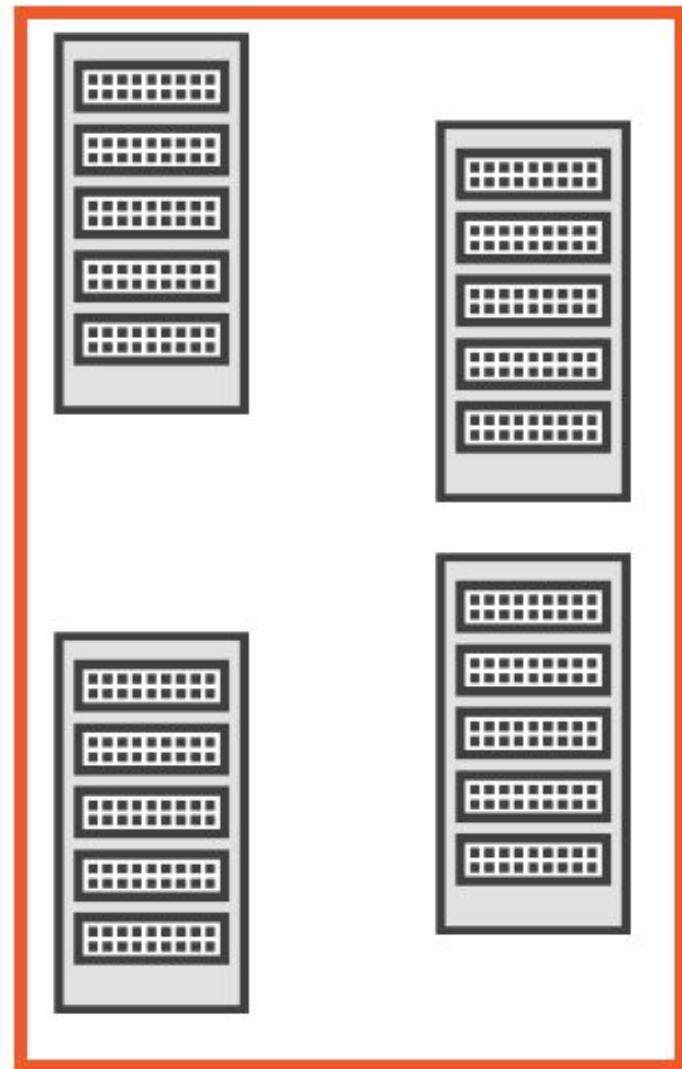


Nodes are on-premise or cloud VMs
on which containers are run

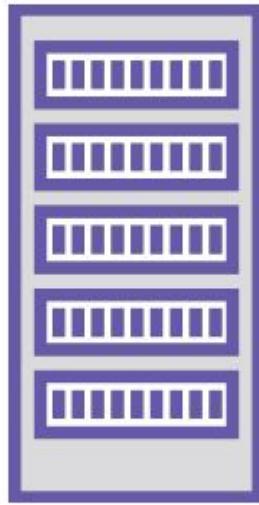
Node Pools



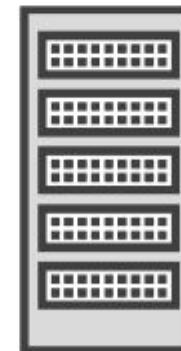
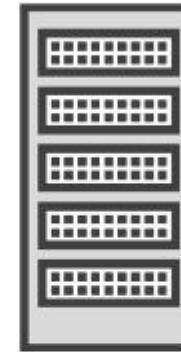
A subset of node instances which have the same configuration are called node pools



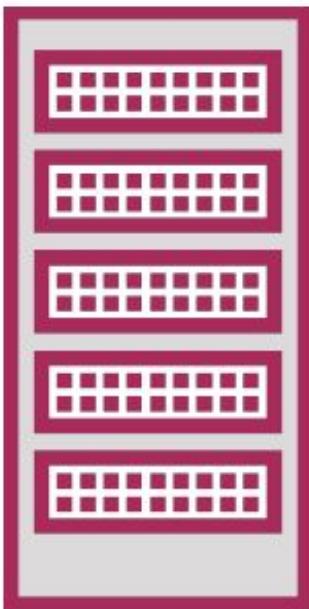
Node Images



Special operating system images
are available on the Google Cloud
to run on Kubernetes nodes



Node Images



Container-optimized OS

- Enhances node security
- Supported by teams at Google

Container-optimized OS with `containerd`

- `containerd` as the main container runtime

Ubuntu

- Optimized for GKE
- Additional support for XFS, CephFS, Sysdig or Debian packages

Lab: Creating a GKE Cluster

Forget Containers, Meet Pods!

Pods as Atomic Units

Container deployment

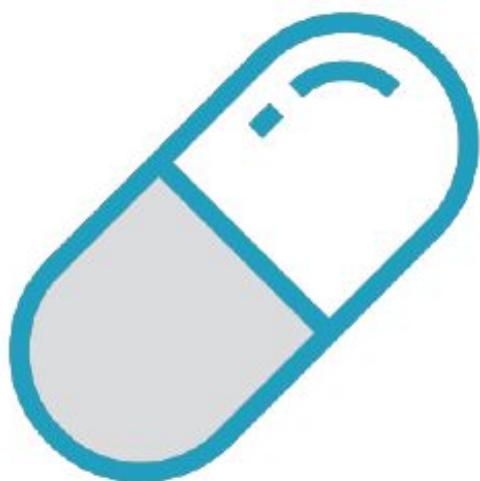
All containers in pod are deployed,
or none are

Node association

Entire pod is hosted on the same
node

Pod is atomic unit of deployment in Kubernetes

Pods on Kubernetes Nodes



Encapsulates one or more containers

Pods run on nodes

Nodes are controlled by master

In GKE

- Nodes are GCE VM instances
- Master is managed by GKE service

Pods on Kubernetes Nodes



Can not run a container without enclosing pod

Pods provide isolation between containers

Pod acts as sandbox for enclosed containers

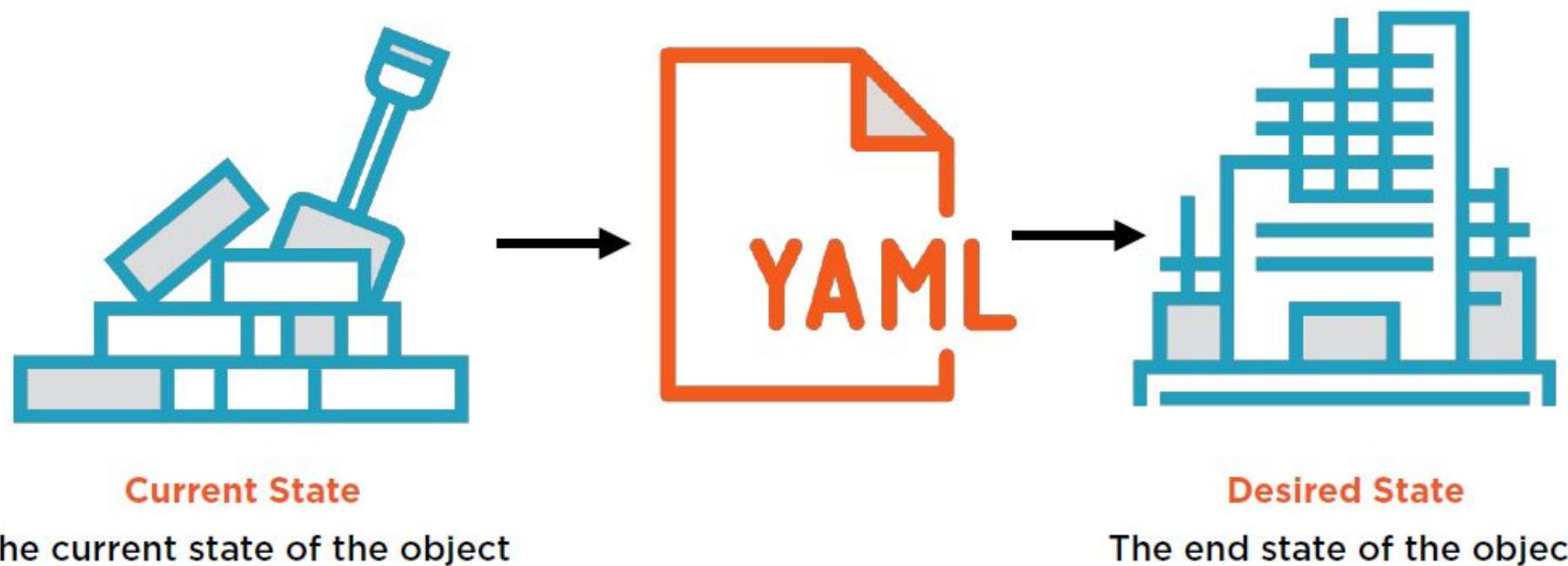
Multi-container pods are possible

- tightly-coupled
- not usually recommended

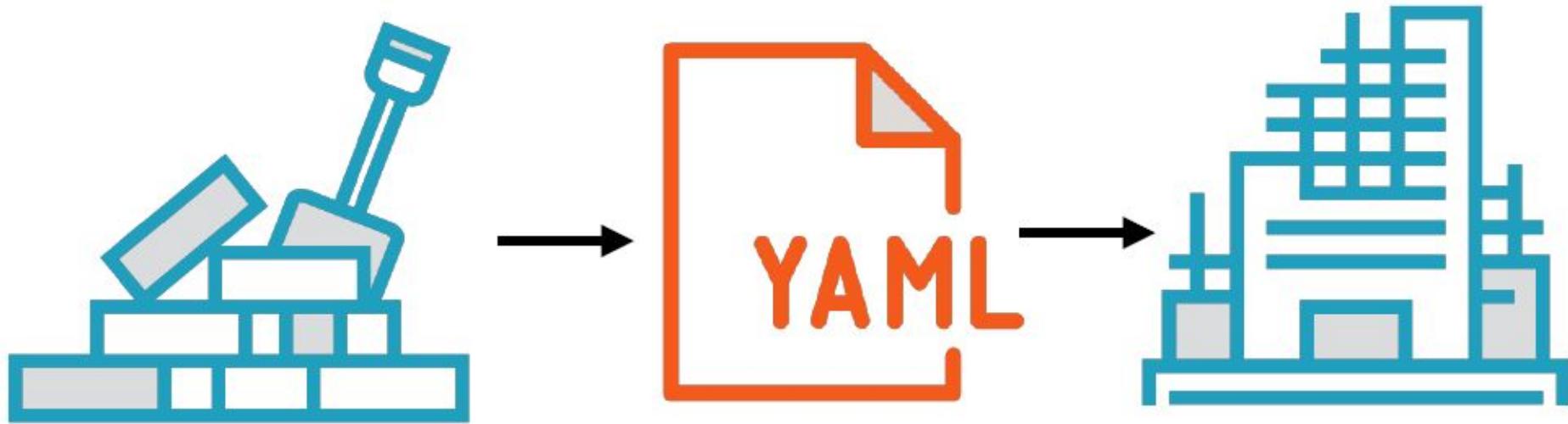
The **Pod object** is the lowest level of abstraction around a container

Every object is associated with a specification file which represents the **desired end state** of the object

YAML Specification Files



YAML Specification Files



Controllers in the Kubernetes cluster run reconciliation loops to get the actual state to match the desired state

Pod Specification File

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - args:
    - /server
    image: k8s.gcr.io/liveness
    livenessProbe:
      httpGet:
        # when "host" is not defined, "PodIP" will be used
        # host: my-host
        # when "scheme" is not defined, "HTTP" scheme will be used. Only "HTTP" and "HTTPS" are allowed
        # scheme: HTTPS
        path: /healthz
        port: 8080
      httpHeaders:
      - name: X-Custom-Header
        value: Awesome
    initialDelaySeconds: 15
    timeoutSeconds: 1
  name: liveness
```

Pod Specification File

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - args:
    - /server
    image: k8s.gcr.io/liveness
  livenessProbe:
    httpGet:
      # when "host" is not defined, "PodIP" will be used
      # host: my-host
      # when "scheme" is not defined, "HTTP" scheme will be used. Only "HTTP" and "HTTPS" are allowed
      # scheme: HTTPS
      path: /healthz
      port: 8080
      httpHeaders:
      - name: X-Custom-Header
        value: Awesome
    initialDelaySeconds: 15
    timeoutSeconds: 1
  name: liveness
```

Which container image(s)?

Available on which port?

Using the Google Kubernetes Engine almost completely eliminates the need to explicitly configure YAML files

Simply use the web console or the gcloud command line utility



Pod Spec

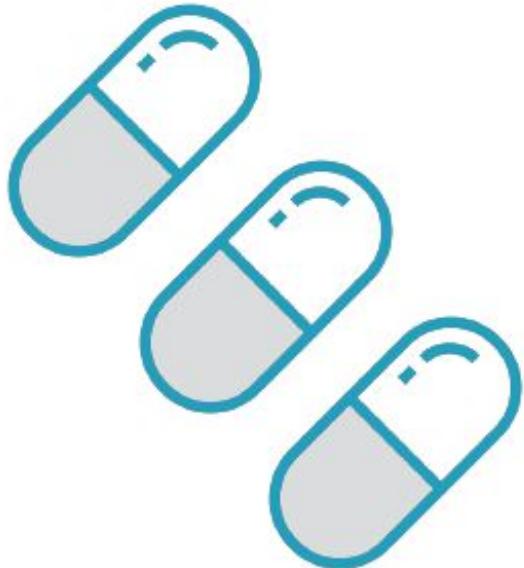
Pods are automatically assigned unique IP addresses

For each container

- Container image with source repository
- Port on which container will be available

Pods can specify shared storage volumes

Pod can contain multiple containers



Multi-container Pods

Avoid unless necessary

Tight coupling, no isolation

No independent scaling possible

Prefer service-oriented architecture to monolithic design



Pod Status

Pending:

- Request accepted, but not yet fully created

Running:

- Pod bound to node, all containers started

Succeeded:

- All containers terminated successfully (will not be restarted)

Failed:

- All containers have terminated, and at least one failed

Unknown:

- Pod status could not be queried

The Anatomy of a Cluster



Limitation of Standalone Pods

No autohealing

- Crashed pods won't restart automatically
- Need higher level orchestration

No scaling or autoscaling

- Overloaded pods don't spawn more automatically
- Need higher level orchestration

No load balancing

- Pod IP addresses are ephemeral
- Pods can't share load automatically
- Need higher level orchestration

Higher-level Abstractions



ReplicaSet, ReplicationController

- Scaling and healing

Deployment

- Versioning and rollback

Service

- Static (non-ephemeral) IP addresses
- Stable networking

Persistent volumes

- Non-ephemeral storage



Kubernetes as Orchestrator

Fault tolerance

- Recover from pod or node failures

Autohealing

- Crashed containers restart

Scaling

- ReplicaSets for multiple copies of the pod



Kubernetes as Orchestrator

Autoscaling

- Scale up and scale down based on load
- Horizontal pod autoscalers

Isolation

- Containers run inside pods

Load balancing

- Distribute traffic to containers



Kubernetes as Orchestrator

Master

Worker nodes

Job scheduling

Resource allocation

Comparing actual and desired state



Kubernetes as Orchestrator

Docker container engine : Java Runtime
Docker containers : Jar files
Kubernetes : Hadoop



Kubernetes as Orchestrator

User interacts with Kubernetes

- kubectl and other command line tools
- YAML config files

Kubernetes relays and orchestrates pods

ReplicaSets and Deployments

Higher-level Abstractions



ReplicaSet, ReplicationController

- Scaling and healing

Deployment

- Versioning and rollback

Service

- Static (non-ephemeral) IP addresses
- Stable networking

Persistent volumes

- Non-ephemeral storage

ReplicaSet



Multiple copies of the pod are managed
together using a ReplicaSet

Self-healing and autoscaling for our pods

Multiple instances of the pod are created and deployed to clusters

Replicas are represented by a ReplicaSet object



ReplicaSet

If pod crashes, ReplicaSet will start a new one

Key to scaling and healing

All pods are replicas of each other



ReplicaSet

Loosely coupled with pods

**A ReplicaSet object will govern all pods
that match its label selector**

Users must ensure no conflicts, orphans

The ReplicaSet Object

**Multiple identical pods which
are replicas of each other**

Higher-level Abstractions



ReplicaSet, ReplicationController

- Scaling and healing

Deployment

- Versioning and rollback

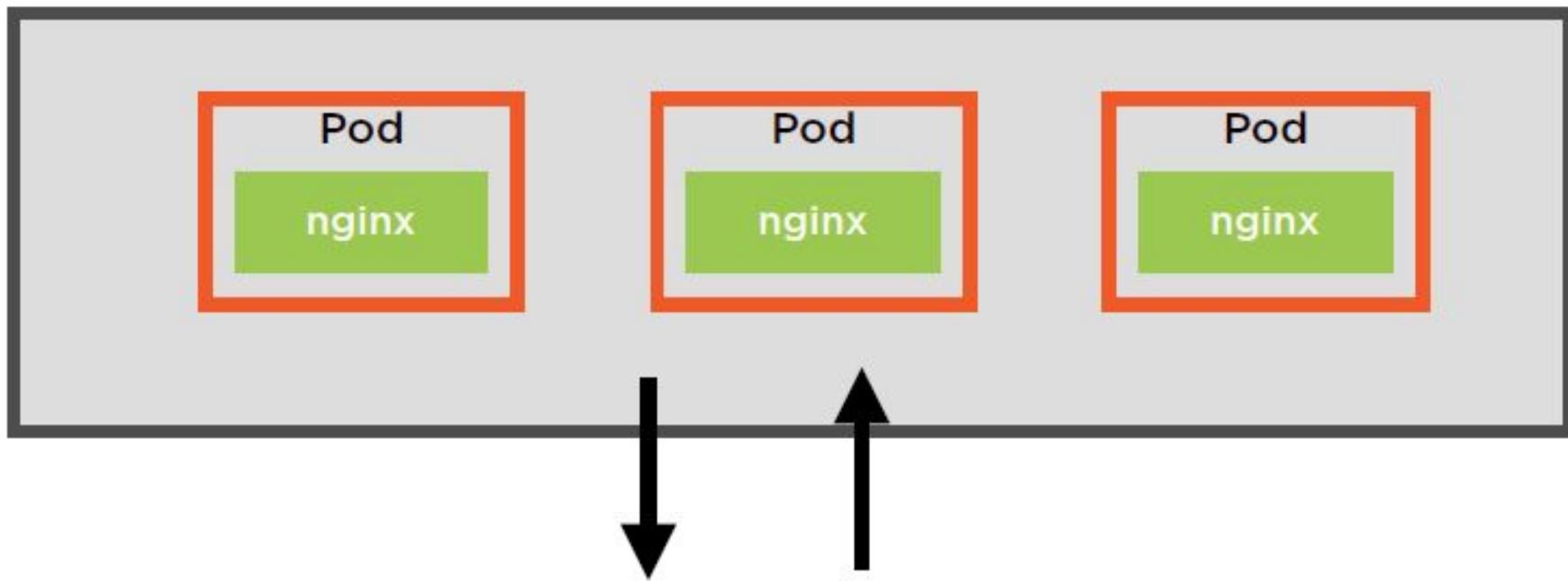
Service

- Static (non-ephemeral) IP addresses
- Stable networking

Persistent volumes

- Non-ephemeral storage

Deployment



Works with ReplicaSets to add versioning,
rollback and other advanced deployment
functionality

The Deployment Object

**Adds on deployment and
rollback functionality**



Deployment Objects

Easy to push out new version of container

Triggers creation of new ReplicaSet and new containers

Pods in old ReplicaSet gradually reduced to zero



Deployment Rollbacks

Every change to a Deployment object triggers creation of a new revision

Trivial to rollback to previous revision

Offers versioning support



Use Cases of Deployments

Manual scaling: Edit number of replicas

Autoscaling: Use HPA with deployment as target

Can pause/resume deployments midway

Can monitor deployment status

Accessing Applications

Higher-level Abstractions



ReplicaSet, ReplicationController

- Scaling and healing

Deployment

- Versioning and rollback

Service

- Static (non-ephemeral) IP addresses
- Stable networking

Persistent volumes

- Non-ephemeral storage



Ephemeral IP Addresses

- Containers expose ports in pod spec
- Pod IP addresses are ephemeral
- Poses a problem for clients
- How are they to know where to send request?

Services provide stable IP
addresses for external
connections and load balancing



Service Objects

Provides stable (non-ephemeral) IP address

Connects to set of back-end pods

Set of pods changes dynamically

- Logically selected via label selector

Front-end IP remains unchanged

Basic load balancing too



Endpoint Object

Each service has associated endpoint object

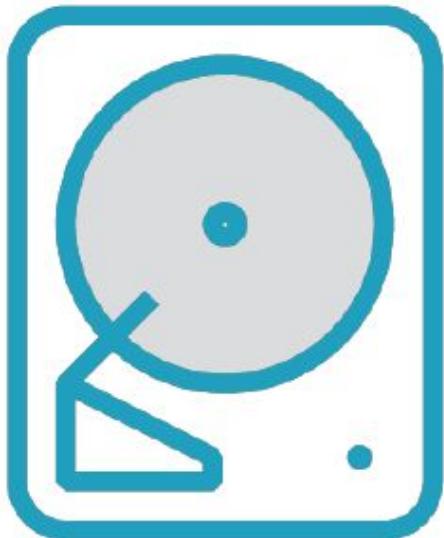
Dynamic list of pods selected by service

Populated by Kubernetes

**Dynamically updates as pods are created/
deleted**

Volume Abstractions

Storage with Containers



On disk files within a container

- Only accessible to the container itself
- Ephemeral: is lost when the container stops or crashes

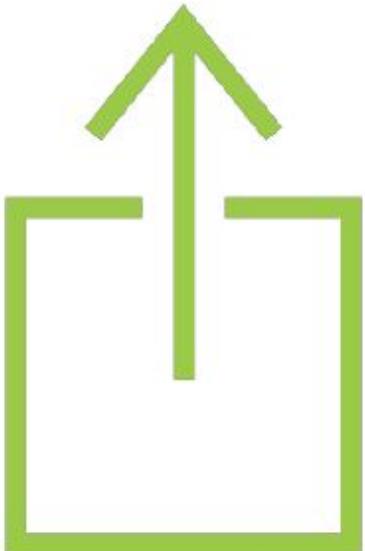
Volume abstractions

- A directory accessible to all containers in a pod
- Specified in the PodSpec
- Have the same lifetime as the enclosing pod

For durable storage use
persistent volumes

The volume is preserved even
when the pod is removed and
can be handed off to another pod

Higher-level Abstractions



ReplicaSet, ReplicationController

- Scaling and healing

Deployment

- Versioning and rollback

Service

- Static (non-ephemeral) IP addresses
- Stable networking

Persistent volumes

- Non-ephemeral storage

Volumes

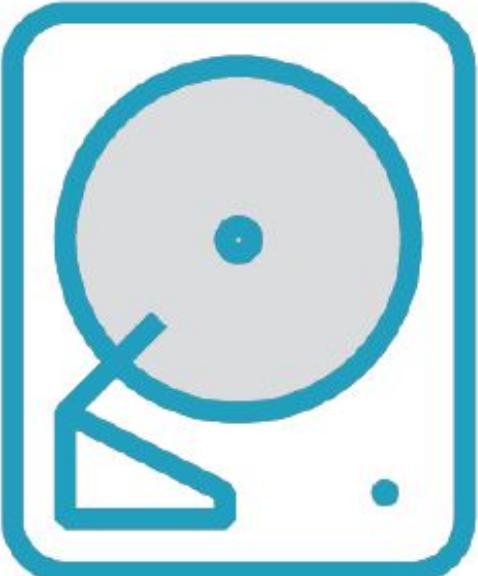
Permanence

Storage that outlives an individual container or individual pod

Shared state

Safe mechanism for containers in pod to share state

Persistent volumes are volumes whose life is not tied to an individual pod



Using Volumes

- Define volume in pod spec
- Have each container mount volume
- Each container mounts independently
- At different path

Volumes and Persistent Volumes

Volumes

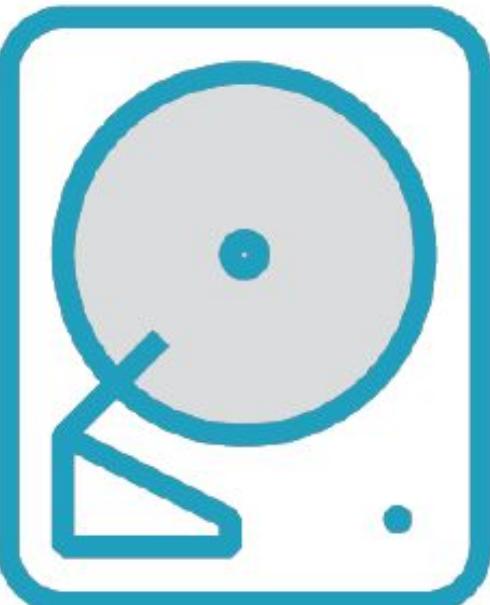
Storage abstraction with life longer than individual container inside pod

Life tied to life of pod

Persistent Volumes

Storage abstraction with life longer than individual pod

Life not tied to life of pod



Cloud-specific Persistent Volumes

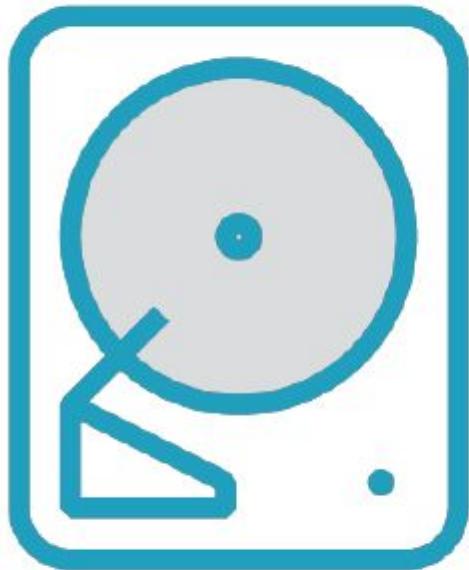
awsElasticBlockStore

azureDisk

azureFile

gcePersistentDisk

Persistent Volumes



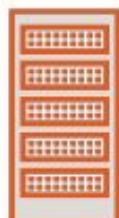
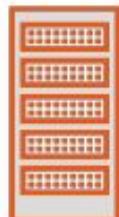
Two types of provisioning

- Static: Administrator pre-creates volume
- Dynamic: Containers gain access by filing PersistentVolumeClaim

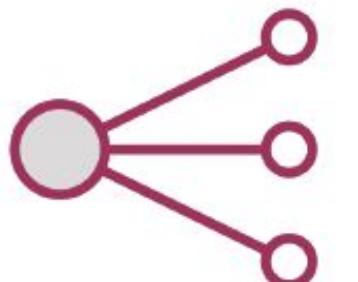
Load Balancing

Load Balancing

Backend
Service



What IP address
to access?

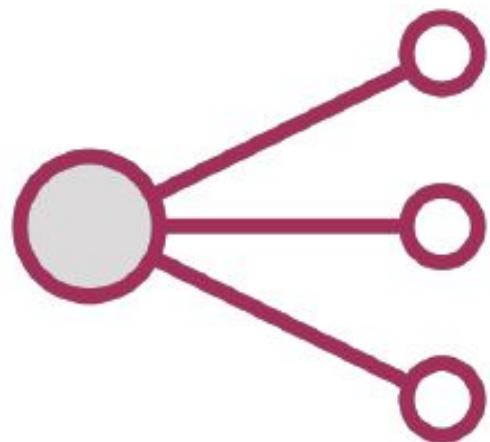


Which specific node
services request?

User Request

Load Balancer

Load Balancer to the rescue



Load Balancers

Stable front-end IP

Forwarding rules to funnel traffic

Connect to backend service

Distribute load intelligently

Health checks to avoid unhealthy instances

Exposing a service on the GKE allows the option of configuring a load balancer to distribute traffic

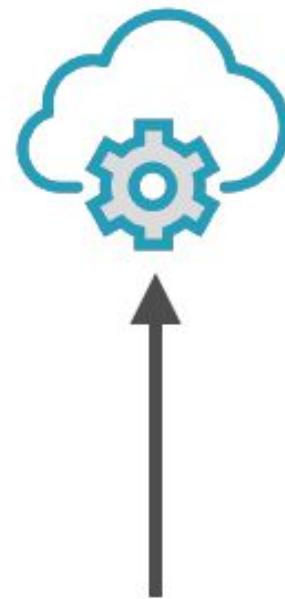
Ingress

Ingress Object

Kubernetes object defining a collection of rules that allow inbound connections to reach cluster services. On GKE, a single ingress object can control access to multiple services

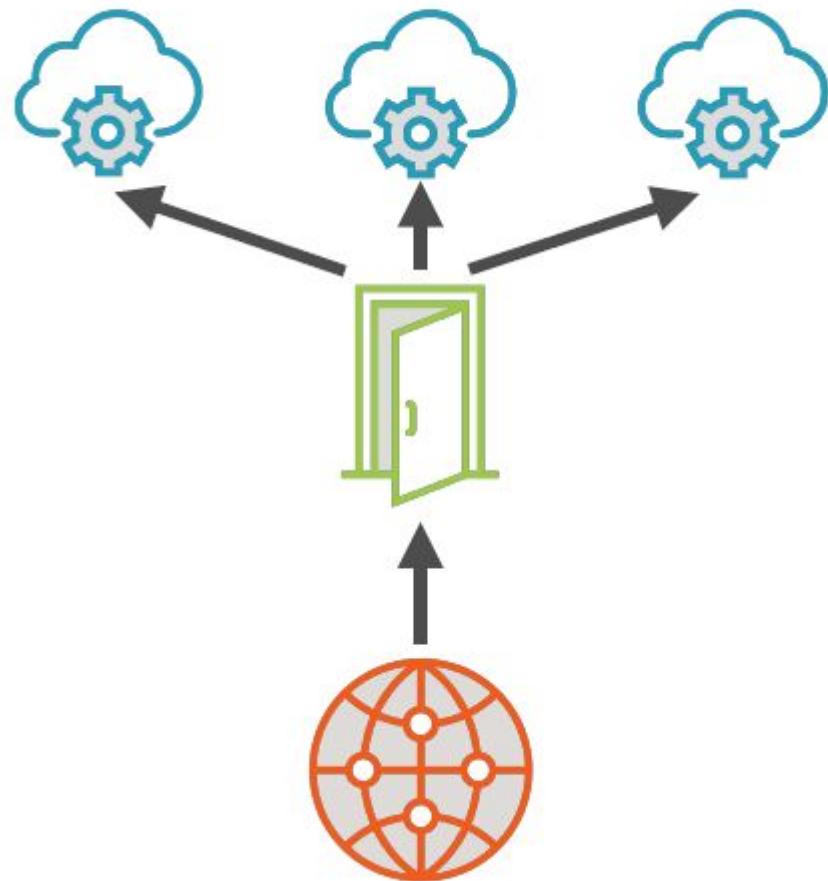
A single service can expose an IP address for access

Ingress

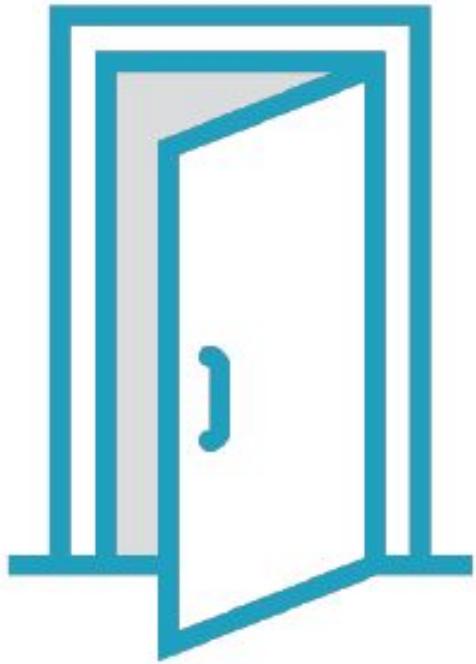


With multiple services it makes sense to have rules defined using an ingress object

Ingress



Ingress Objects



Can be configured to support

- Externally-reachable URLs
- Load balancing
- SSL termination
- Name-based virtual hosting



Ingress Objects

GKE fully supports both ingress and load balancer objects

On other platforms need to do some additional work

(Need to configure ingress object as pod)



GKE Ingress Objects

GKE clusters have `HttpLoadBalancing` add-on enabled

Causes additional controller to run on master

This controller supports HTTP(S) load balancing for ingress objects

StatefulSets and DaemonSets

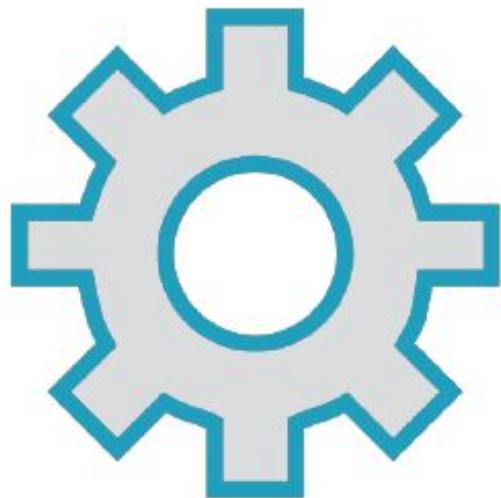


StatefulSets

A set of pods, similar to ReplicaSet

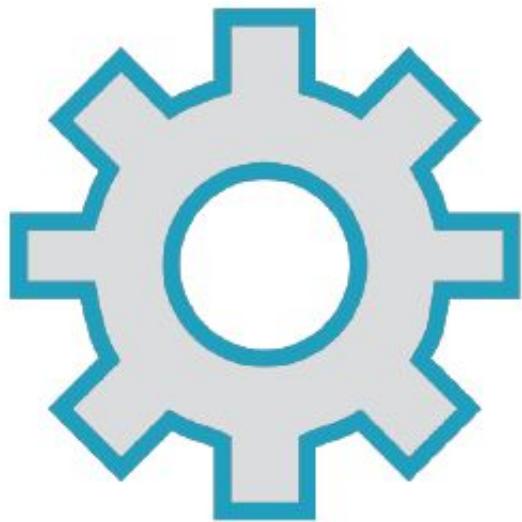
Important difference from ReplicaSet

- Pods created unique
- Identified by name
- Not interchangeable
- Always associated with persistent volume



DaemonSet

Manages groups of replicated pods
Attempts to keep one pod per node
Across all nodes or a subset
As nodes added, pods created too
As nodes removed, pods are garbage collected



DaemonSet

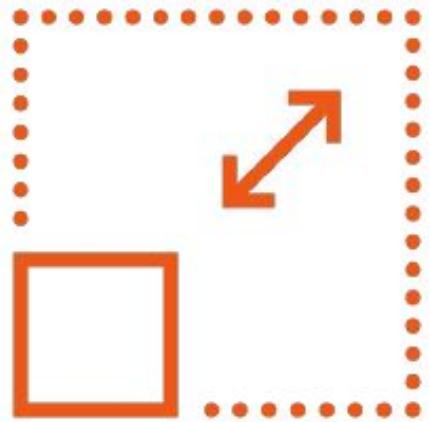
Cluster storage daemons

Log collection daemons

Node monitoring daemons

Horizontal Pod Autoscaler

Autoscaling with Kubernetes



Autoscaling

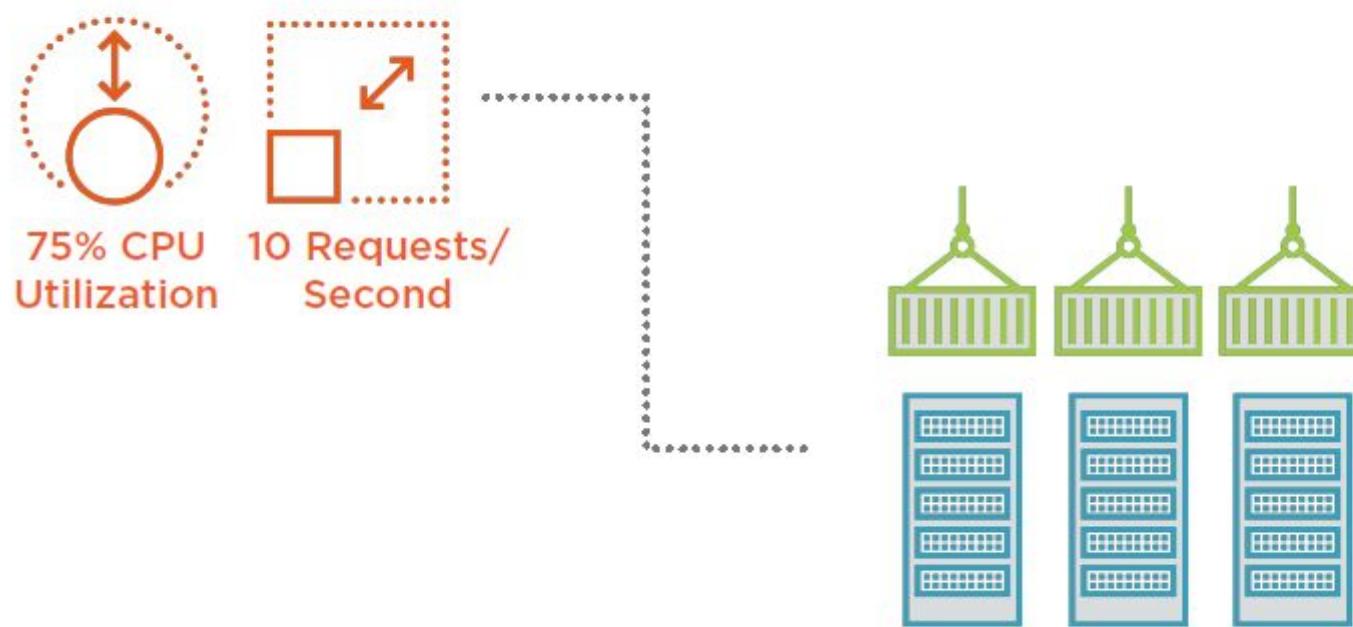
Compute capacity automatically changes with changing need



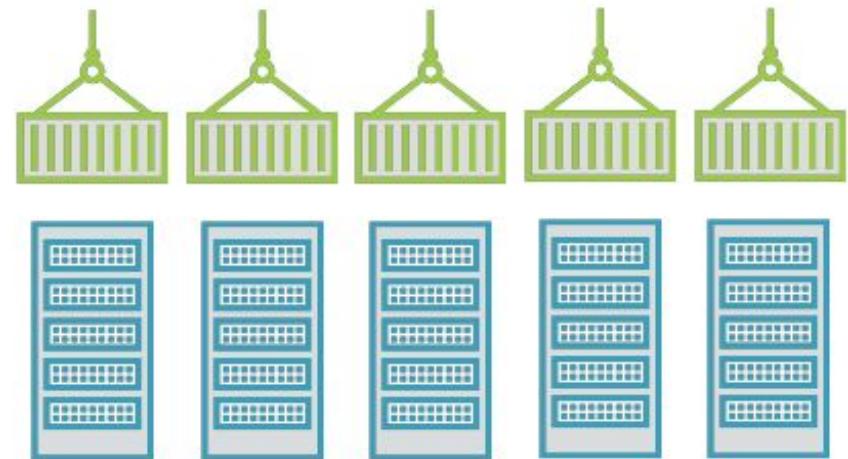
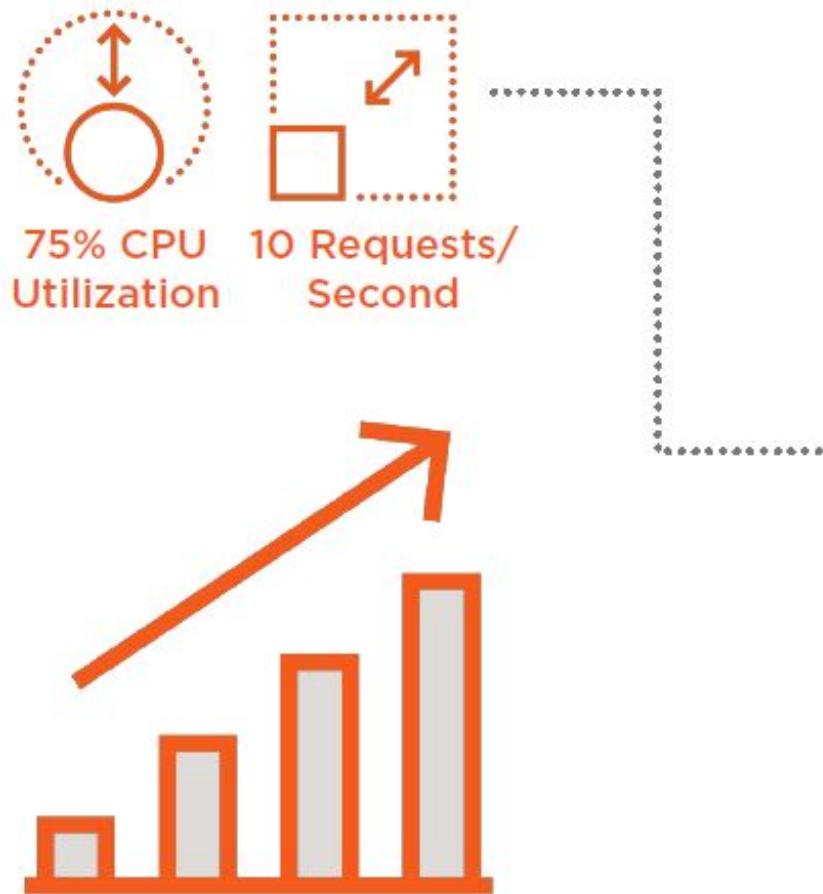
Autohealing

Platform ensures health of compute resources

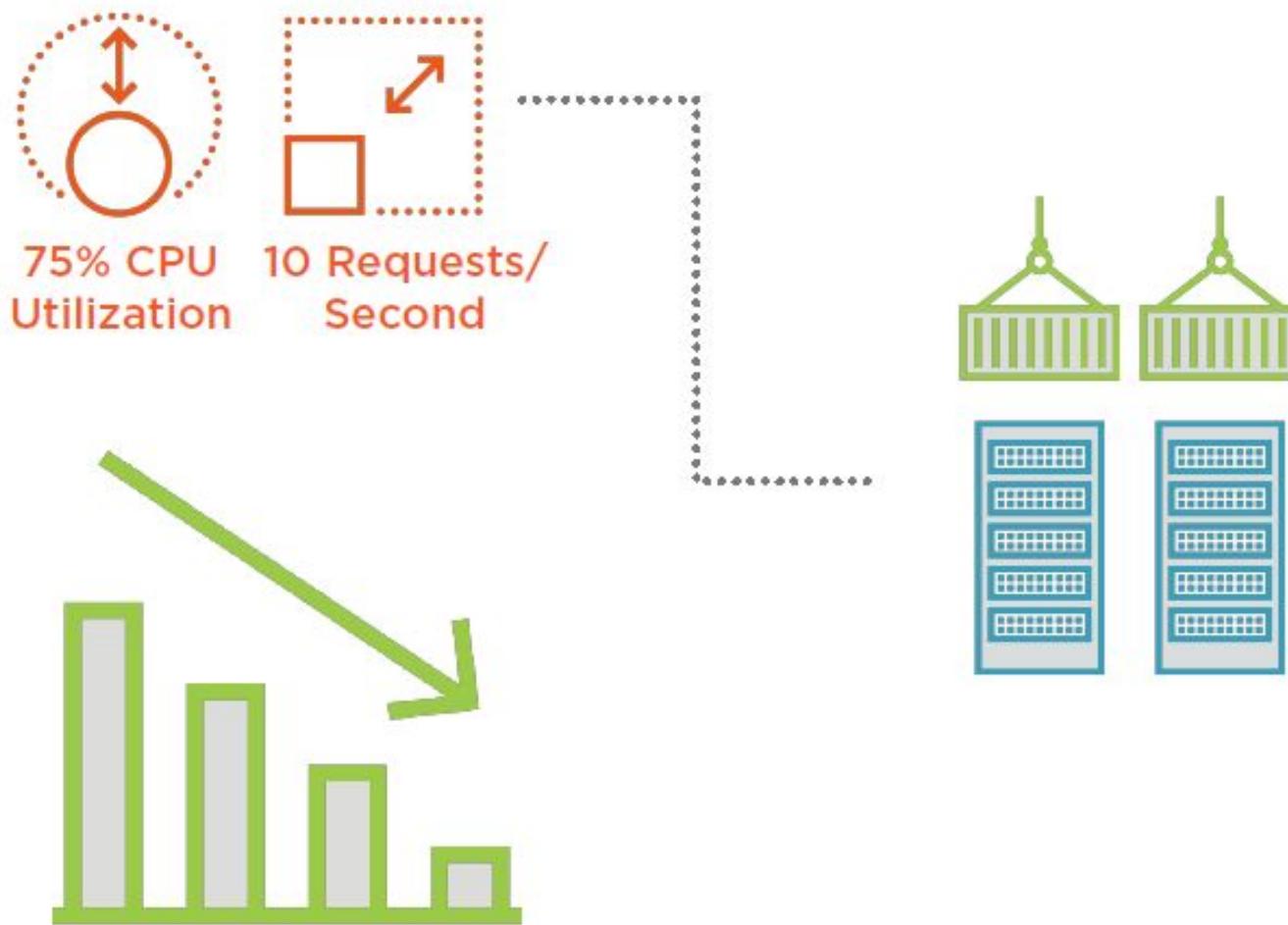
Autoscaling with Kubernetes



Autoscaling with Kubernetes



Autoscaling with Kubernetes





Horizontal Pod Autoscaler

Even higher level abstraction
Specify any scalable object as target
Along with autoscaling policy



HPA Targets

Scalable objects

- ReplicaSet
- Deployment

Can't target non-scalable objects

- DaemonSets



Preventing Thrashing

Thrashing occurs when HPA can't find stable size

Use cool-down periods to avoid this

Set intervals between successive operations of same kind

Label Selectors on Kubernetes

Labels

Key-value pairs attached to objects such as pods. Used to specify meaningful identifying attributes of objects. Not meant to be unique.

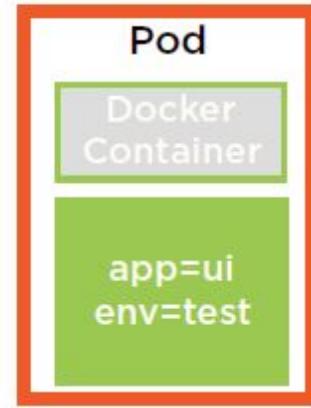
Label Selectors

Allow clients, users, or higher-level abstractions to identify groups of objects. Core grouping primitive in a Kubernetes cluster.

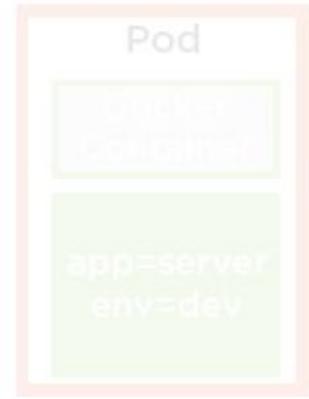
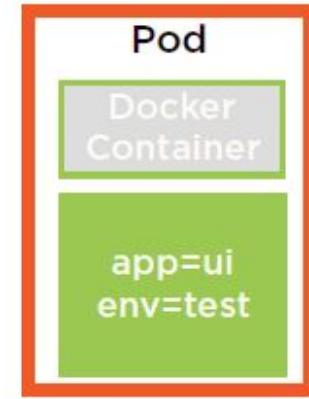
Label Selectors

**Allow higher-level abstractions
to be loosely coupled with
objects they manage**

Label Selectors

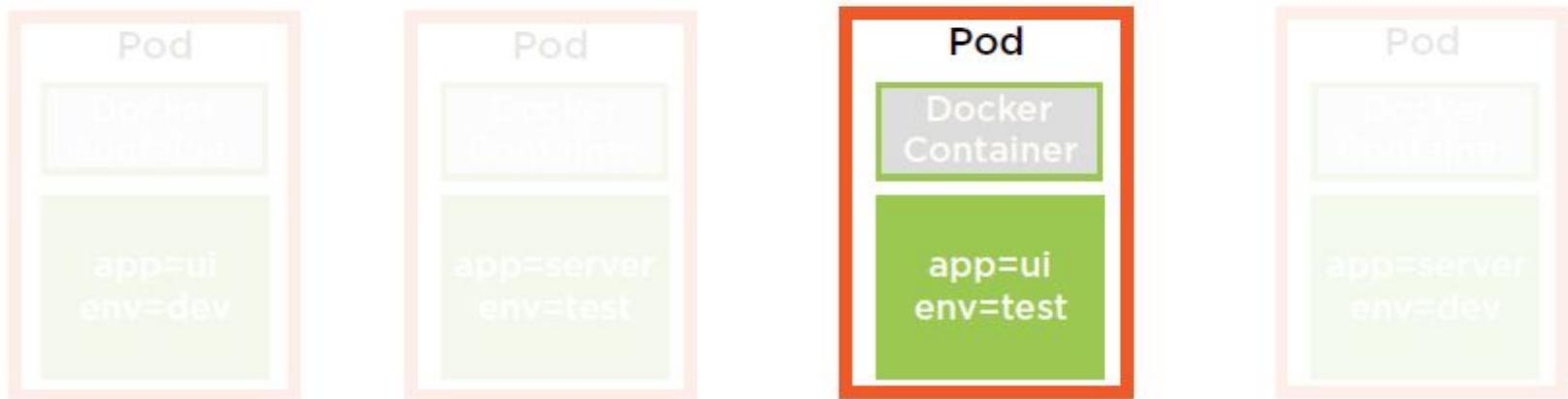


Label Selectors



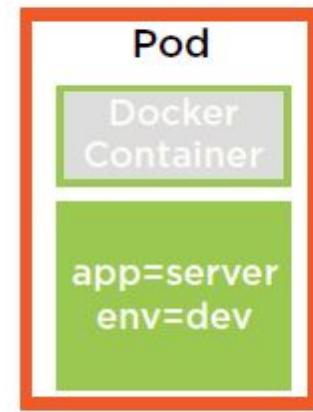
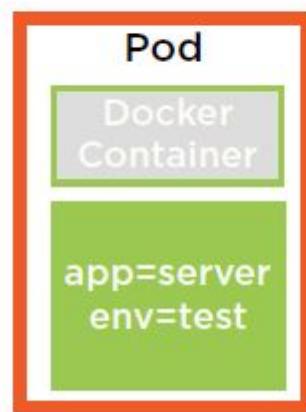
app=ui

Label Selectors



app=ui
env=test

Label Selectors



app=server

Jobs and CronJobs



Job

A controller object which represents a finite task

Manages the task until it completes execution

Does not maintain a desired cluster state

Best for long-running, batch operations

Two Types of Jobs

Non-parallel job

Creates one pod and uses that pod to run the job through to completion

Parallel job

Creates multiple pods and is complete when a certain count of pods terminate successfully

The Job controller is responsible for re-creating pods if its pods fail or are terminated

Two Types of Jobs

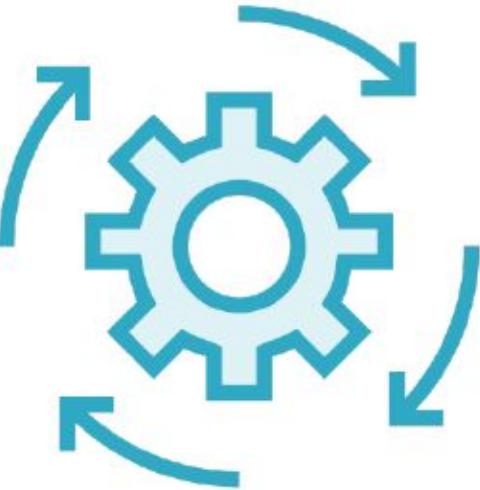
Non-parallel job

Creates one pod and uses that pod to run the job through to completion

Parallel job

Creates multiple pods and is complete when a certain count of pods terminate successfully

Can also specify deadlines for Jobs so that they do not re-try creating pods forever



CronJob

A controller object to perform finite time-related tasks

Tasks may be run once or repeatedly at a time interval

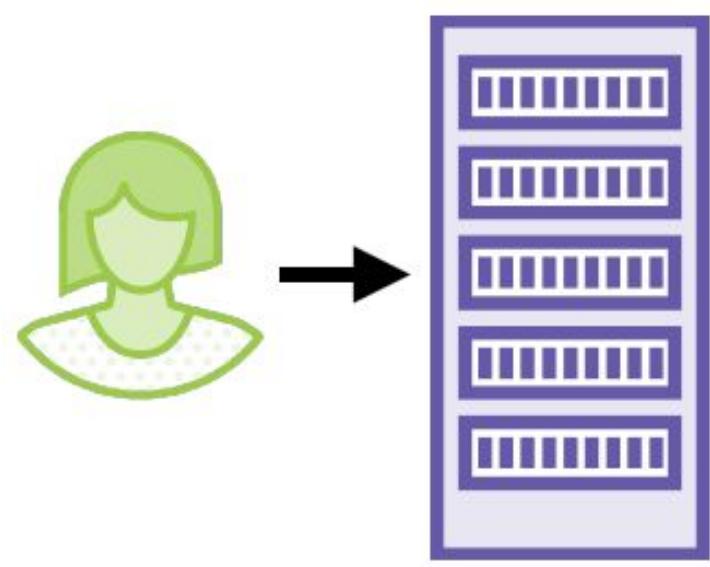
Uses Job objects under the hood to manage tasks

Best for automated, repeated tasks:

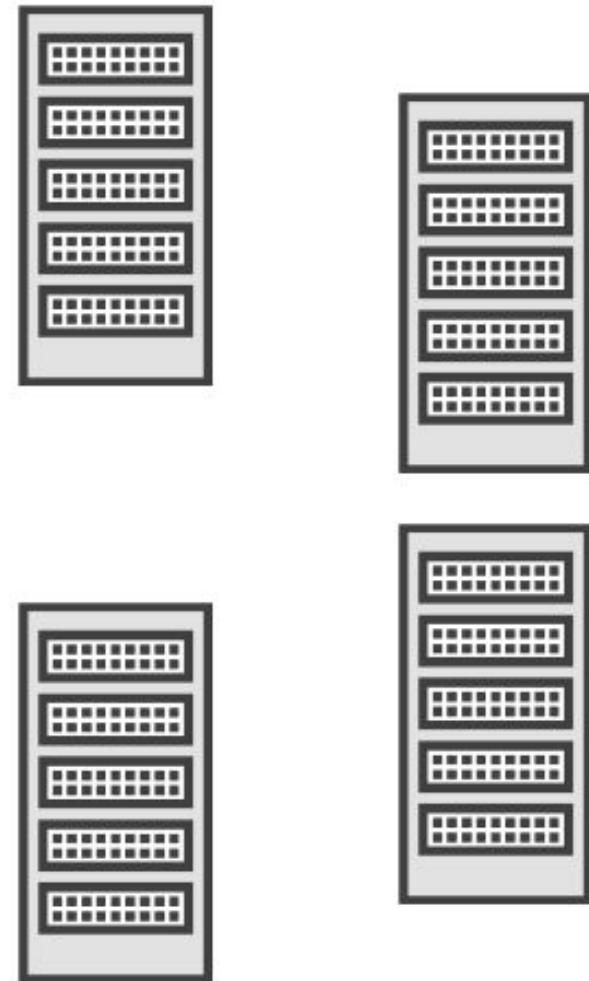
- Backups, generating reports, sending email

Controlling Scheduling with Node Taints

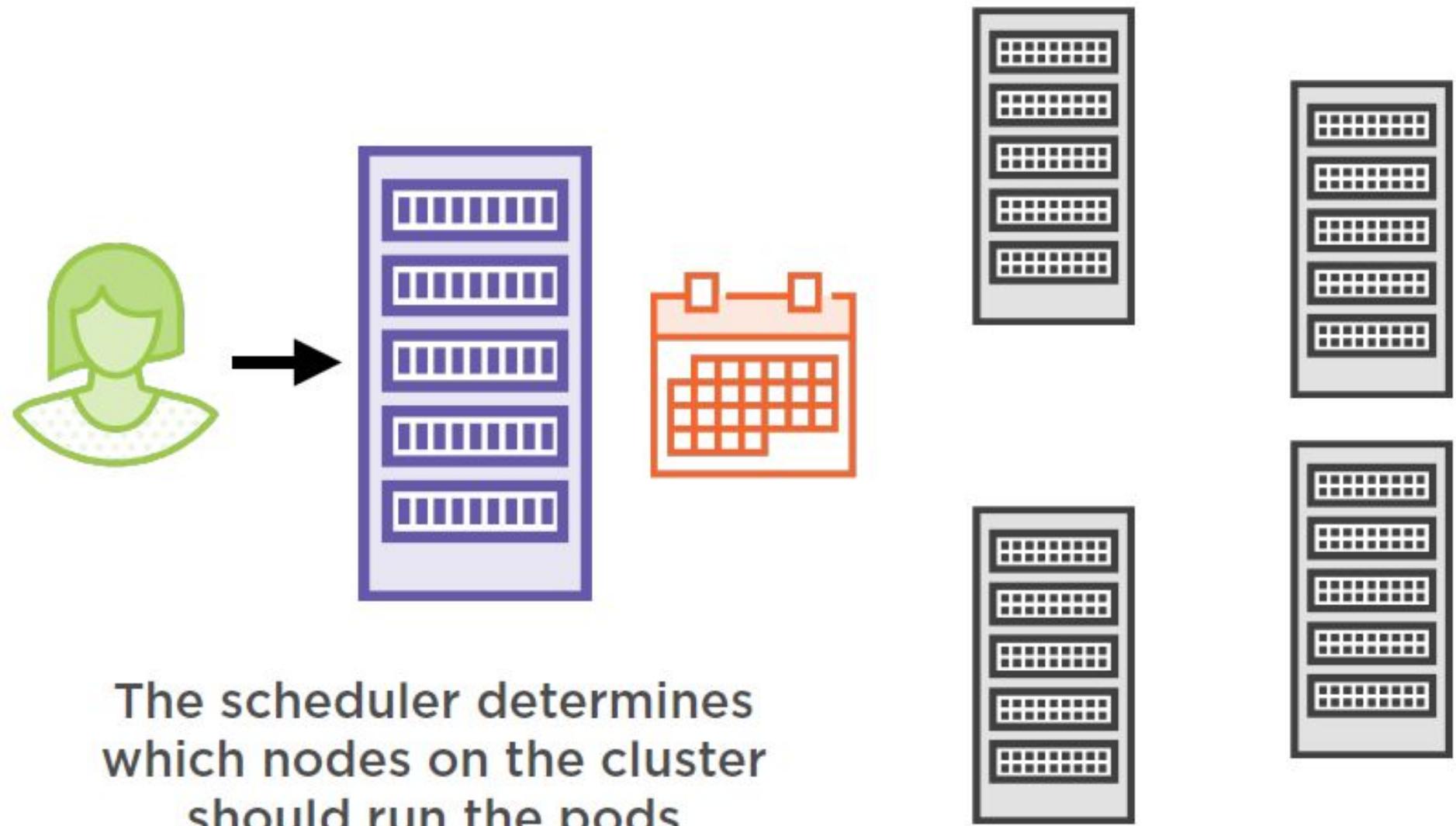
Workload Submitted to a Cluster



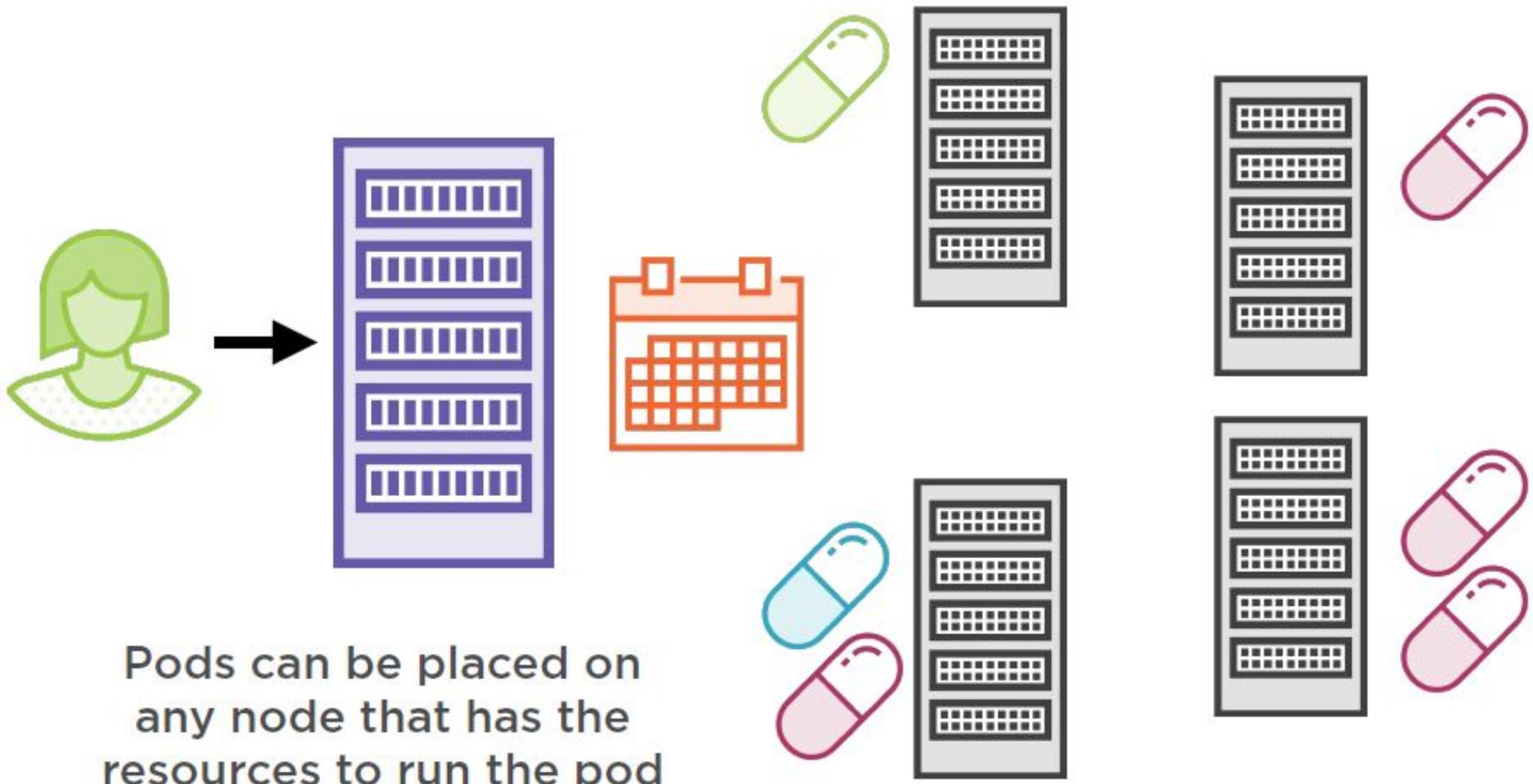
User creates a deployment and submits it to a cluster



Workload Submitted to a Cluster



Workload Submitted to a Cluster



Control Where Workloads Run

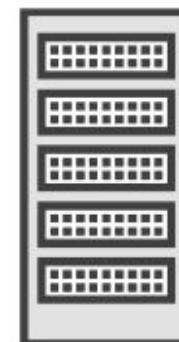
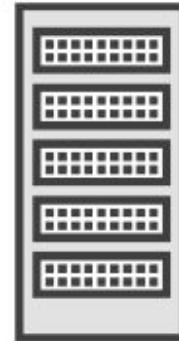
Identify certain nodes
in the cluster for
types of workloads



Identification using
node taints

Node Taints

Not all pods can be scheduled on tainted nodes



Pods which are configured with tolerations can be used on tainted nodes



Node Taints

A property of nodes that allow them to repel a set of pods

Control where pods run

Key-value pairs associated with an effect

Node Tolerations

**Applied to pods and allow the
pods to be scheduled on tainted
nodes**



Node Taints

NoSchedule: Pods are not scheduled on this node unless they tolerate this taint

PreferNoSchedule: Avoid scheduling pods unless they tolerate this taint

NoExecute: Evict running pods and do not schedule new ones

Toleration in a PodSpec

```
tolerations:  
- key: env  
  operator: Equal  
  value: test  
  effect: NoSchedule
```

Allows this pod to be scheduled on a node which has the taint:

`env=test:NoSchedule`



Applying Node Taints on the GKE

Taint all nodes on the cluster

Taint only nodes which belong to a node pool

Taint individual nodes in the cluster



Taints and Tolerations Use Cases

Allow nodes to be dedicated to a certain set of users

Keep regular applications away from nodes which have specialized hardware

Separate real-time and batch operations

Stateless and Stateful Applications

Two Types of Applications



Stateless applications
do not store state in
persistent storage



Stateful applications
use persistent storage
to save data

Stateless Applications



Stateless Applications

Data and application state stay with the client

Not stored to clusters or persistent storage

Can scale by simply adding more replicas

Stateless Applications



Stateless Applications

Frontend code typically stateless

Get more pods up and running to handle increasing traffic

Deployed using the Deployment controller

Pods are uniform and non-unique

Stateless Applications



Stateless Applications

Deployments simply specify the desired state of application

Number of pods, container version, pod labels

Change YAML specification to update state

Stateful Applications



Stateful Applications

Save data to persistent storage for use by other services

Scaling more involved might need to consider read/write latencies

Stateful Applications



StatefulSets

A set of pods, similar to ReplicaSet

Important difference from ReplicaSet

- Pods created unique
- Identified by name
- Not interchangeable
- Always associated with persistent volume



Secrets

Secrets

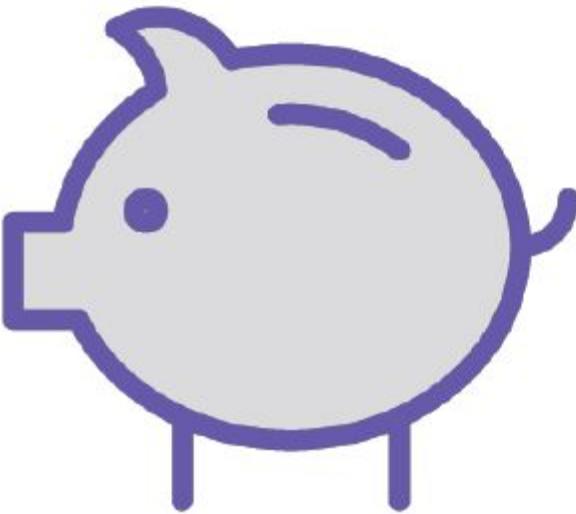
Objects holding sensitive information

Store passwords, tokens, keys in a cluster

Safer than putting it in a PodSpec or an image

Can be created by users or the system

To use a secret, a pod needs to reference the secret



Compute and Storage

GCE instances are used for nodes in the cluster

Billed for instances based on GCE usage

Per-second basis with 1 minute minimum cost

Costs vary based on machine type, region, type of disk etc.

Access Control and Security in Kubernetes and Google Kubernetes Engine (GKE)

In Kubernetes, there are two main types of users

Normal users



Service accounts

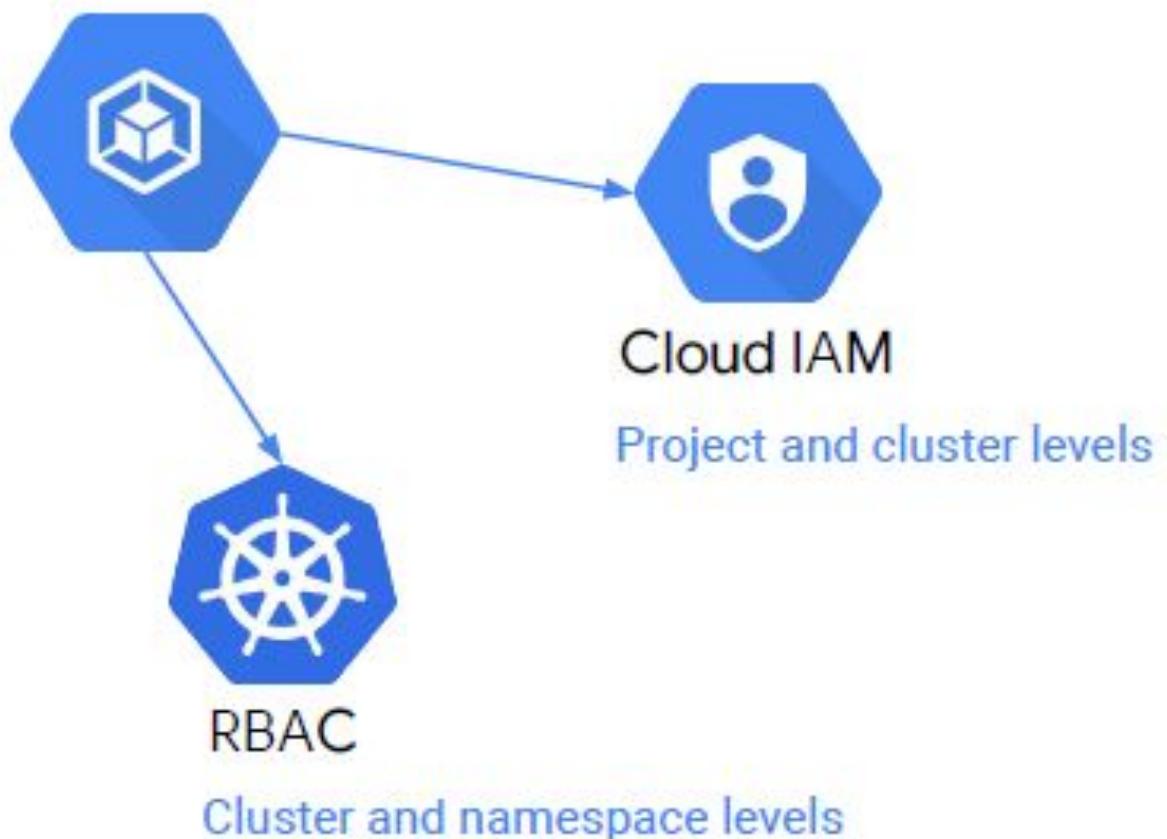


Managed by
Cloud IAM

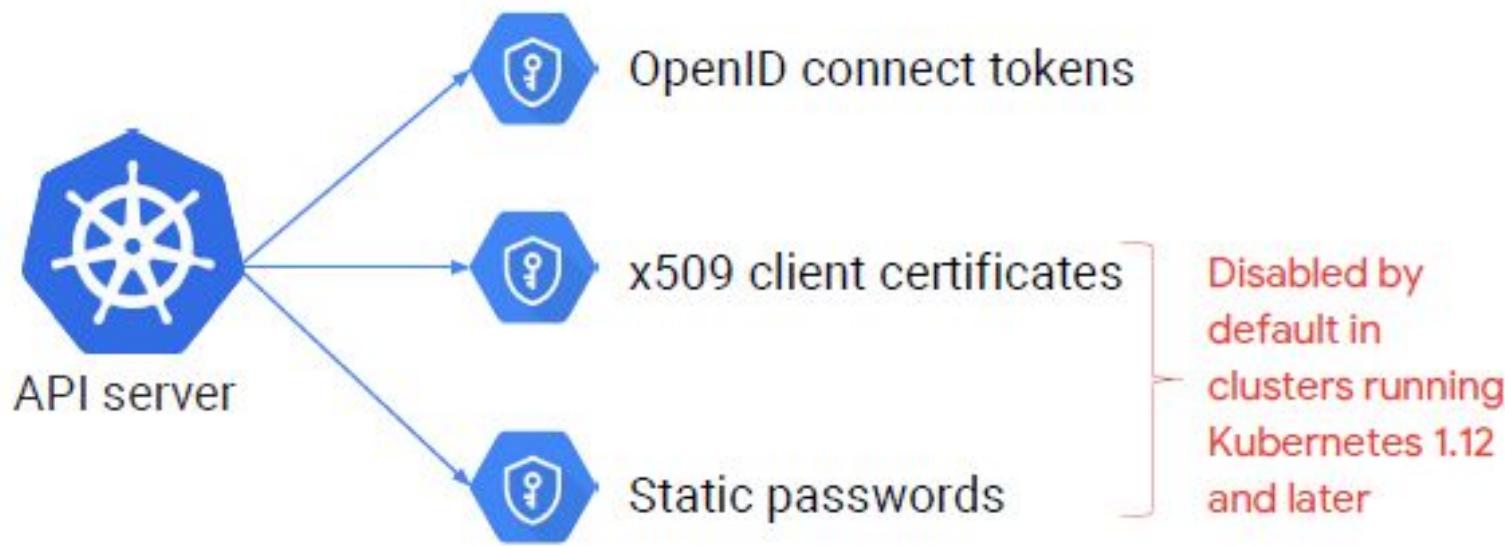
Managed by
Kubernetes



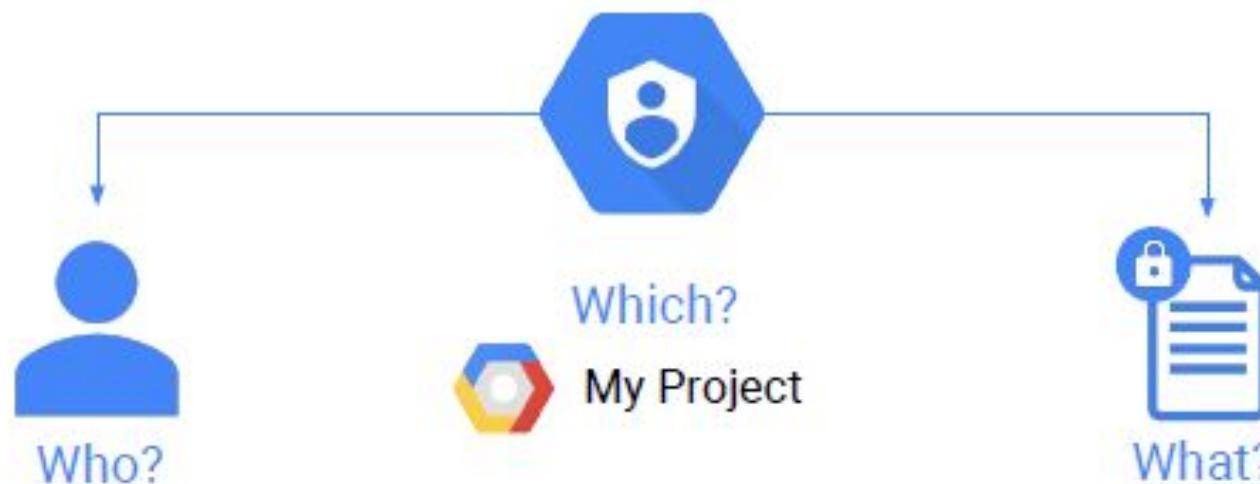
There are two main ways to authorize in GKE



API server authenticates in different ways



Three elements are defined in Cloud IAM access control



Identity/Member

example@gmail.com



Which?
My Project

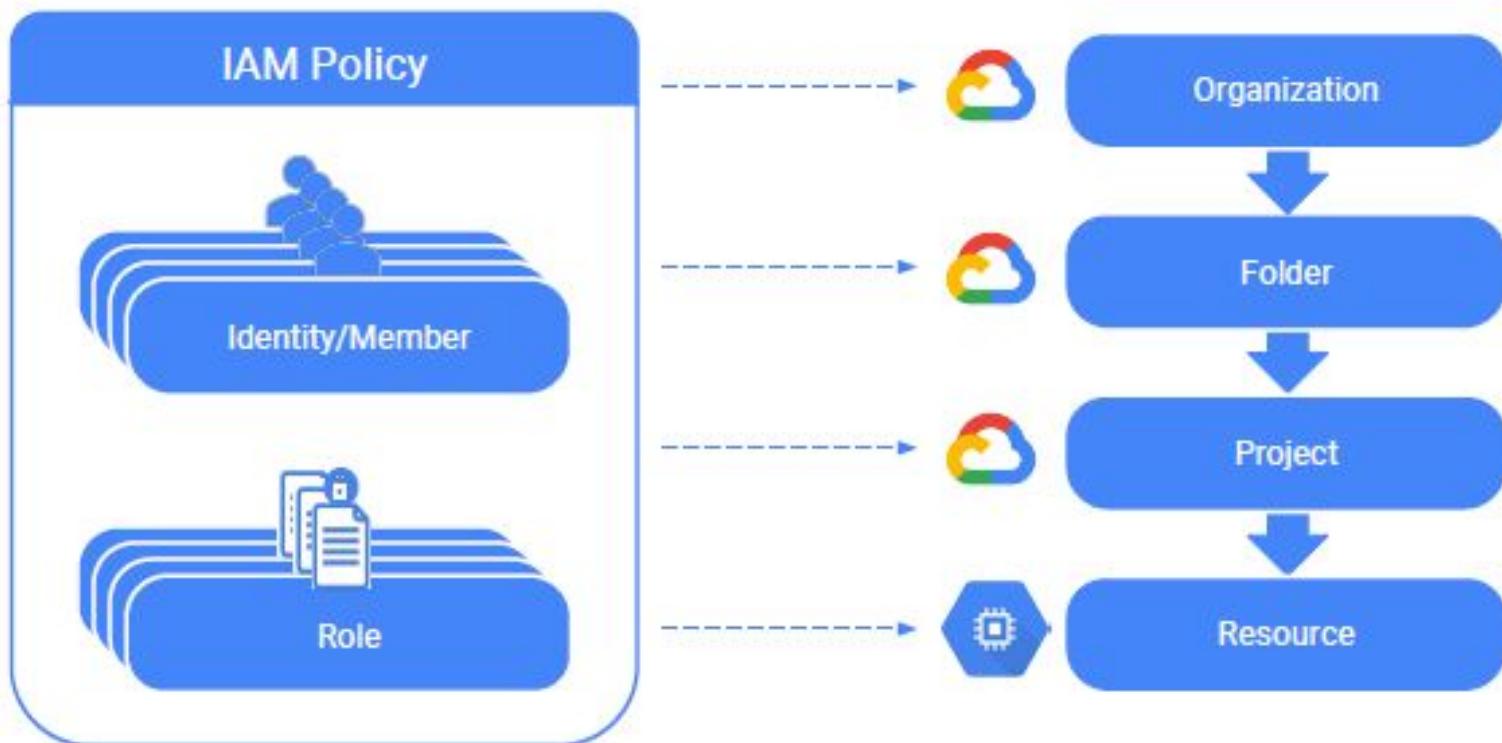


What?

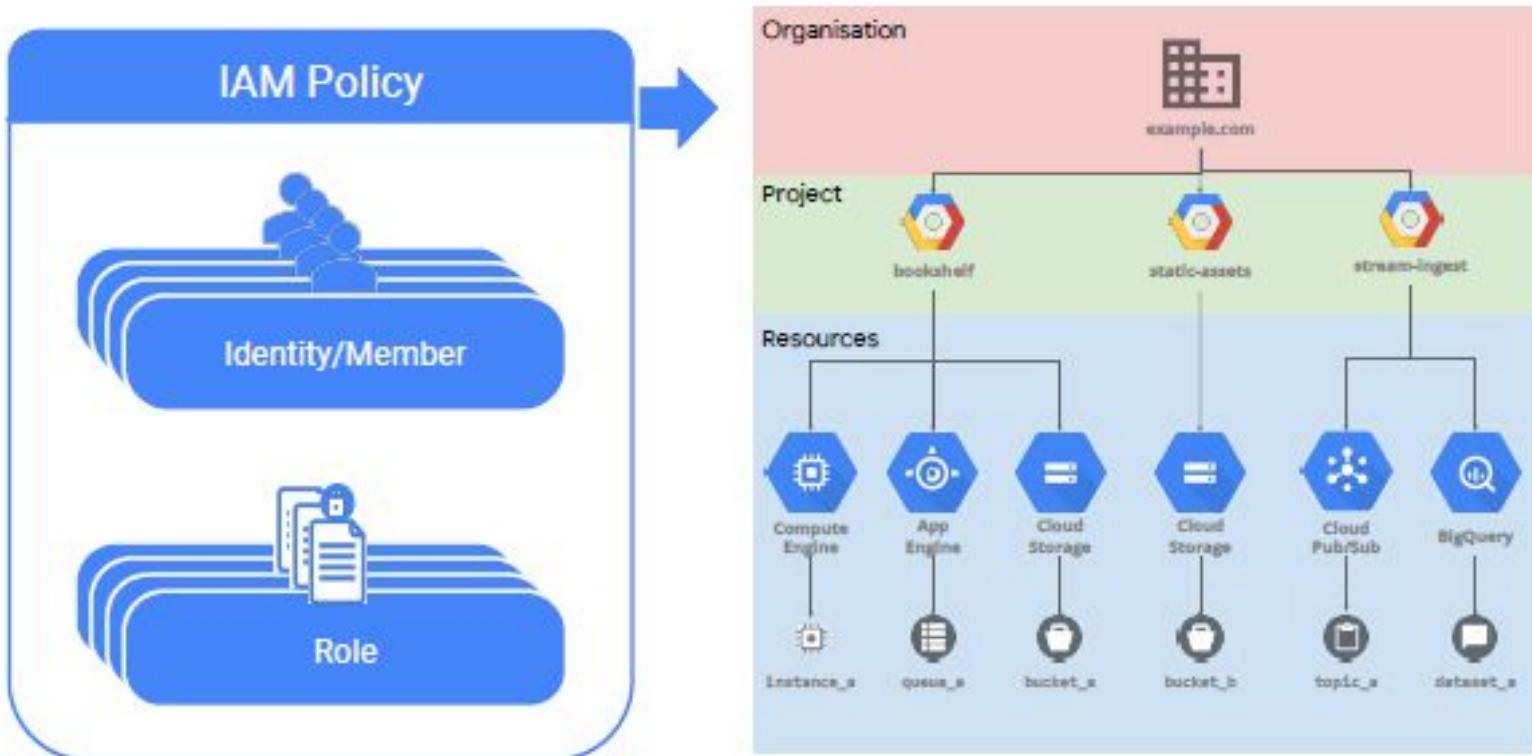
roles/compute.viewer:
compute.instances.list
compute.instances.get
...



How a Cloud IAM policy grants roles to users



How a Cloud IAM policy grants roles to users



Primitive roles grant global, project-level access



Viewer role

- Read-only permissions



Editor role

- Write permissions
- All permissions of a viewer role



Owner role

- Manage roles and permissions
- Set up project billing
- All permissions of an editor role



GKE predefined Cloud IAM roles provide granular access to Kubernetes resources

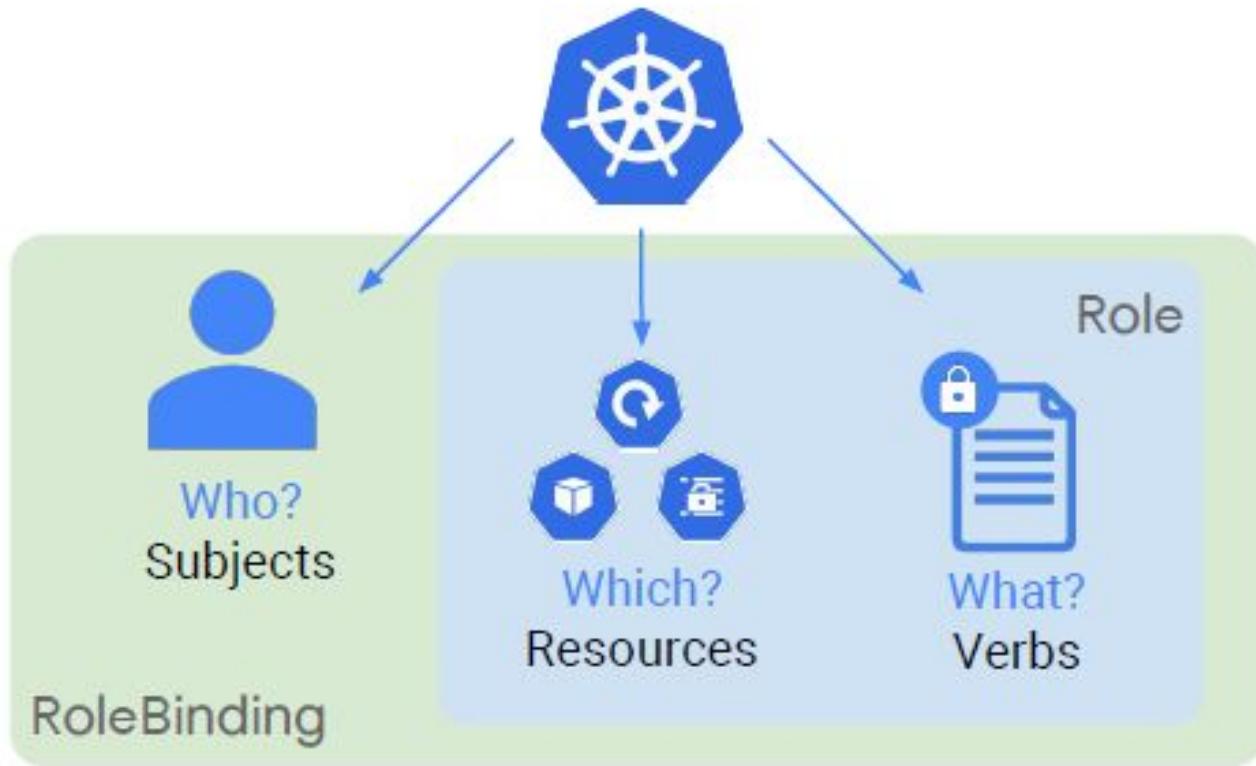
GKE Viewer	GKE Developer	GKE Admin	GKE Cluster Admin
Read-only permissions to cluster and Kubernetes resources	Full access to Kubernetes resources within clusters	Full access to clusters and their Kubernetes resources	Create/delete/update/view clusters No access to Kubernetes resources



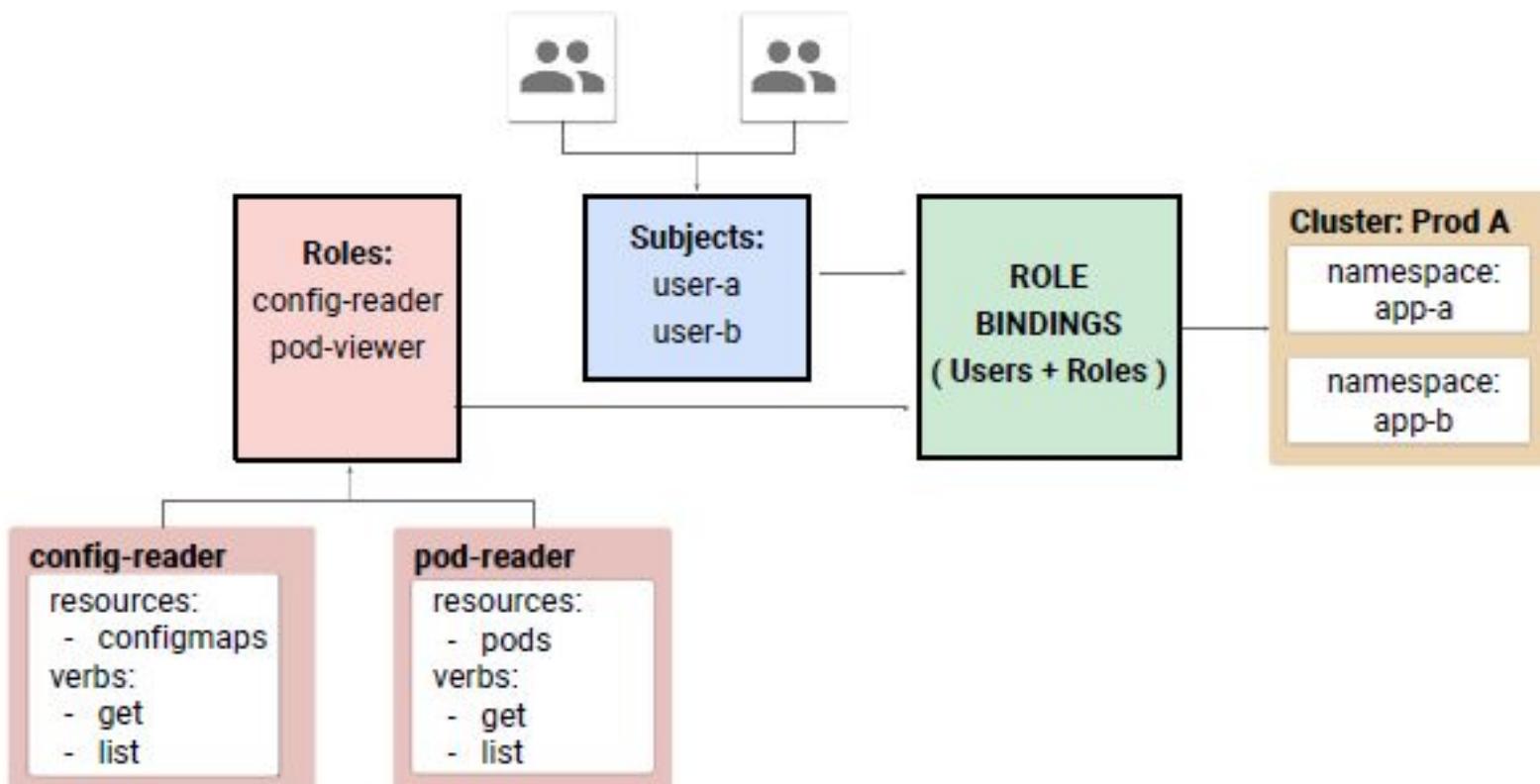
Use custom roles where predefined roles are insufficient



Kubernetes RBAC concepts



Kubernetes RBAC components



A role contains rules that represent a set of permissions

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: demo-role
rules:
  - apiGroups: []
    resource: ["pods"]
    verbs: ["get", "list", "watch"]
```

Group	Version	Kind
core	v1	Pod

Kubernetes API



A ClusterRole grants permissions at the cluster level

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: demo-clusterrole
rules:
  - apiGroups: []
    resource: ["storageclasses"]
    verbs: ["get", "list", "watch"]
```



Examples of how to refer to different resource types

```
rules:  
- apiGroups: [""]  
  resources: ["pods"]  
  verbs: ["get", "list", "watch"]
```

```
rules:  
- apiGroups: [""]  
  resources: ["pods"]  
  resourceNames: ["demo-pod"]  
  verbs: ["patch", "update"]
```

```
rules:  
- apiGroups: [""]  
  resources: ["pods", "services"]  
  verbs: ["get", "list", "watch"]
```

```
rules:  
- nonResourceURLs: ["metrics/",  
  "/metrics/*"]  
  verbs: ["get", "post"]
```



Attaching roles to RoleBindings

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  namespace: default
  name: demo-rolebinding
subjects:
- kind: User
  name: "joe@example.com"
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: demo-role
  apiGroup: rbac.authorization.k8s.io
```



ClusterRoleBinding has a cluster-wide scope

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: demo-clusterrolebinding
subjects:
- kind: User
  name: "admin@example.com"
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: demo-clusterrole
  apiGroup: rbac.authorization.k8s.io
```



Examples of how to refer to different subject types

subjects

```
- kind: User  
  name: "joe@demo.com"  
  apiGroup: rbac.authorization...
```

subjects

```
- kind: Group  
  name: system.serviceaccounts  
  apiGroup: rbac.authorization...
```

subjects

```
- kind: Group  
  name: "Developers"  
  apiGroup: rbac.authorization...
```

subjects

```
- kind: Group  
  name: system.serviceaccounts:qa  
  apiGroup: rbac.authorization...
```

subjects

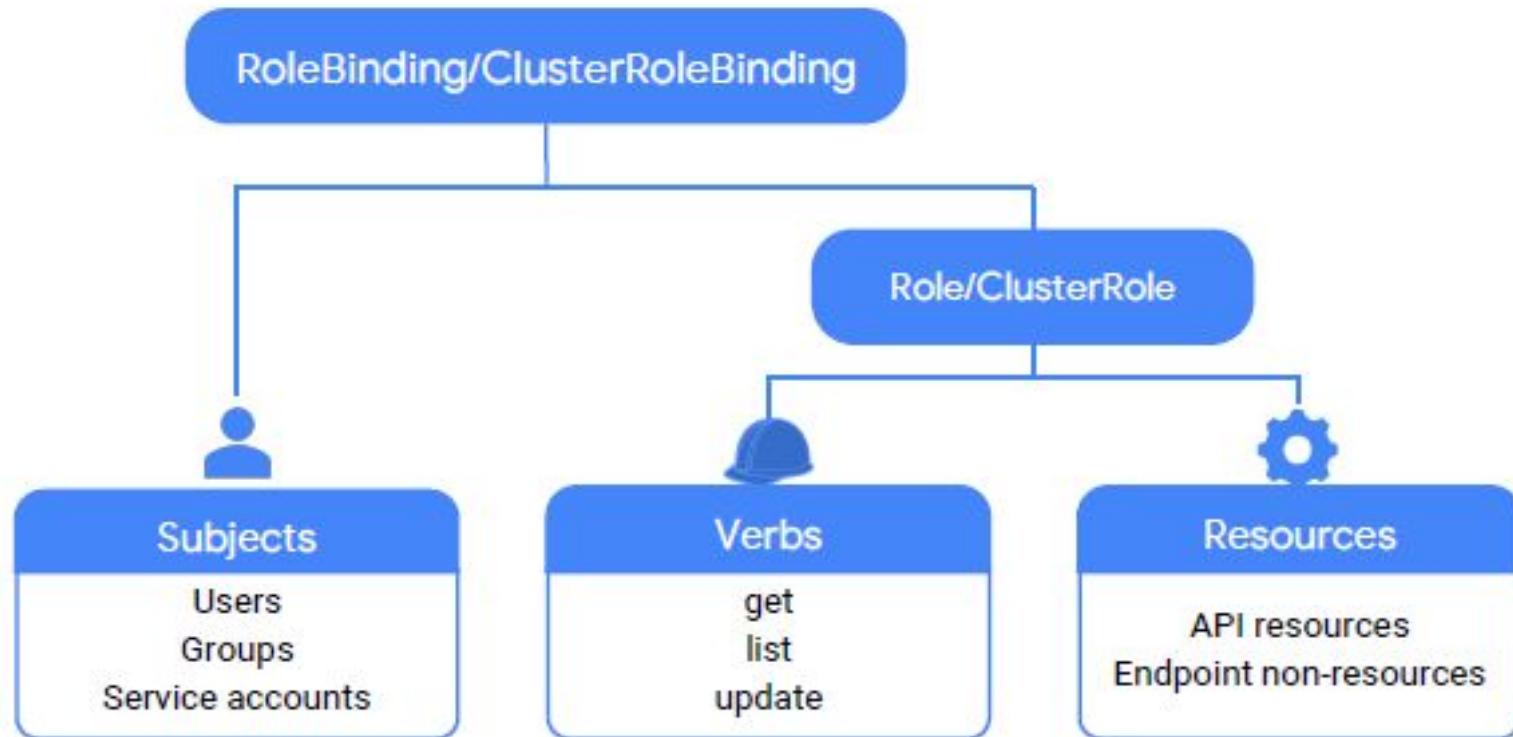
```
- kind: ServiceAccount  
  name: default  
  namespace: kube-system
```

subjects

```
- kind: Group  
  name: system.authenticated  
  apiGroup: rbac.authorization...
```



Kubernetes RBAC summary



Not all resources are NameSpaced

```
$ kubectl api-resources
```

<i>NAME</i>	<i>... NAMESPACED</i>
<i>bindings</i>	<i>true</i>
<i>componentstatuses</i>	<i>false</i>
<i>configmaps</i>	<i>true</i>
<i>endpoints</i>	<i>true</i>
<i>...</i>	
<i>namespaces</i>	<i>false</i>
<i>nodes</i>	<i>false</i>
<i>...</i>	



ABAC authorization isn't recommended

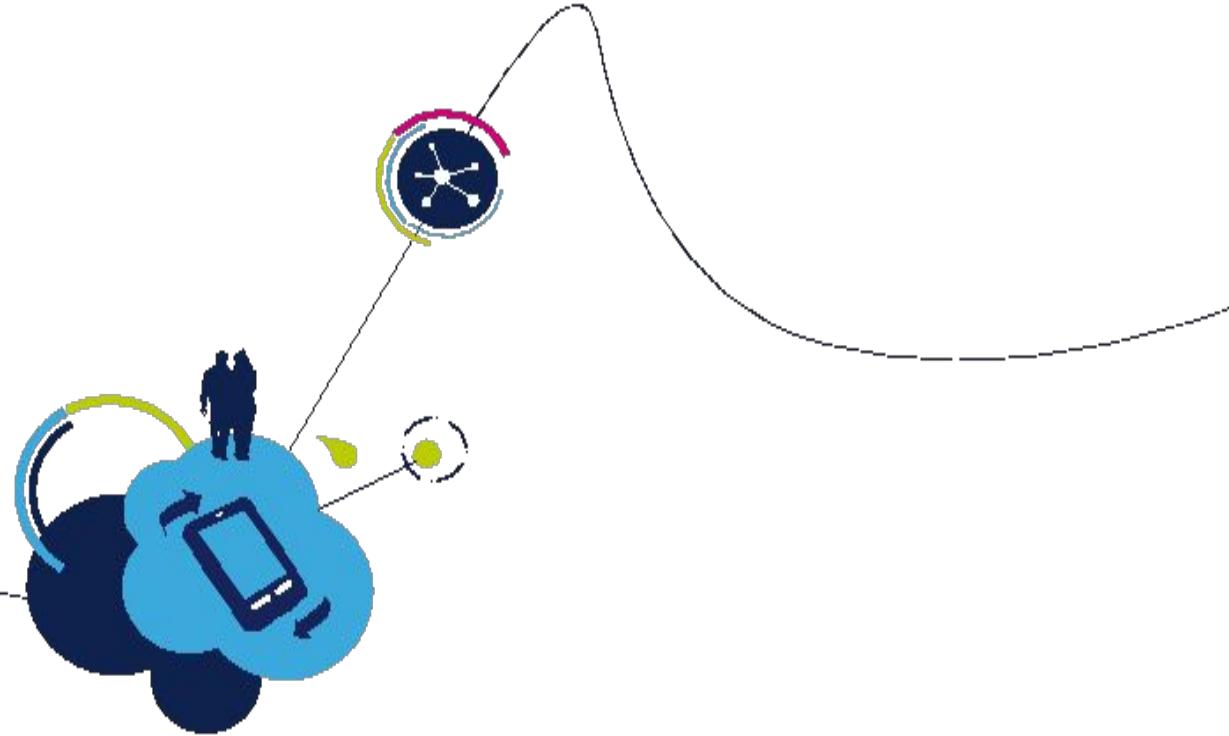


- Disable attribute-based access control (ABAC)
Disabled by default in GKE 1.8+



- Use RBAC instead





Questions?