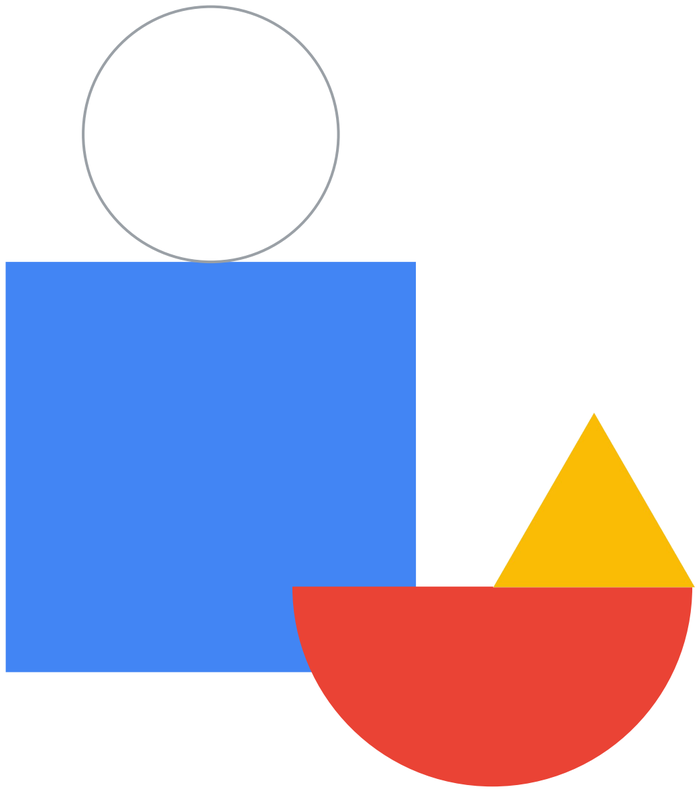
Google Cloud



Preparing for your Professional Cloud Architect Journey

Module 5: Managing Implementation and Ensuring Solution and Operations Reliability

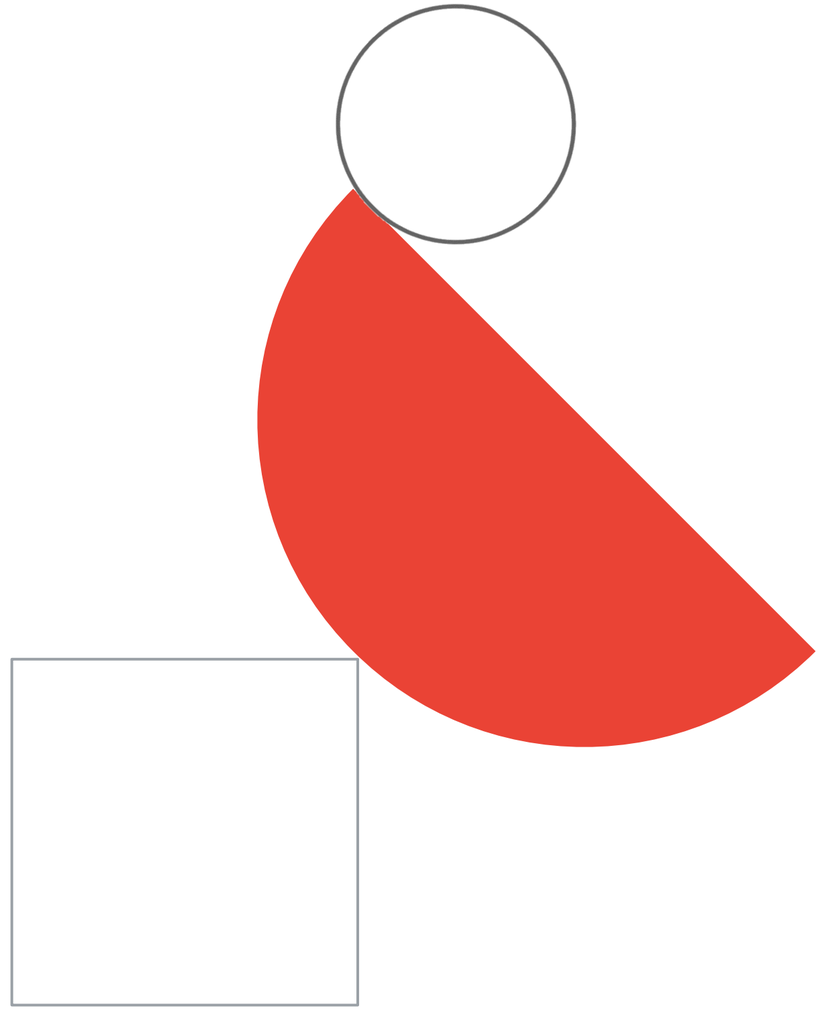
Module agenda

01 Implementation, operations, and reliability at Cymbal Direct

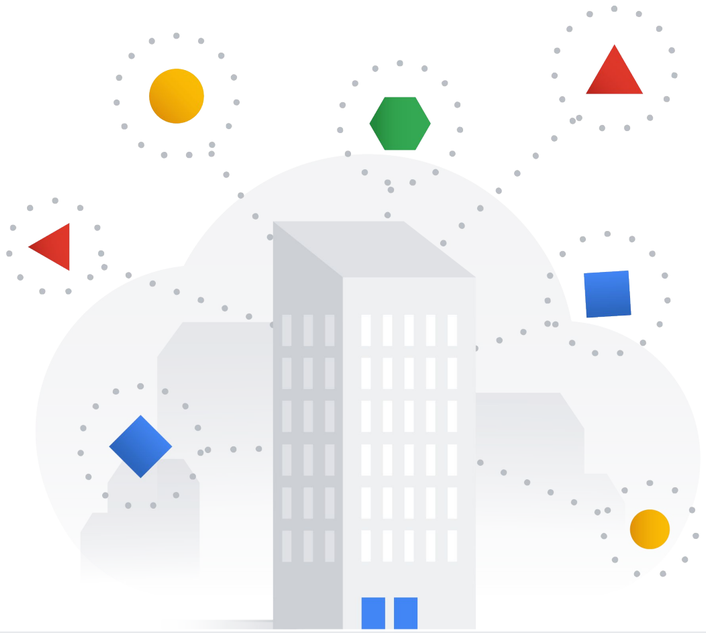
02 Diagnostic questions

03 Review and study planning

Implementation, operations, and reliability at Cymbal Direct



Your role in implementation, operations, and reliability

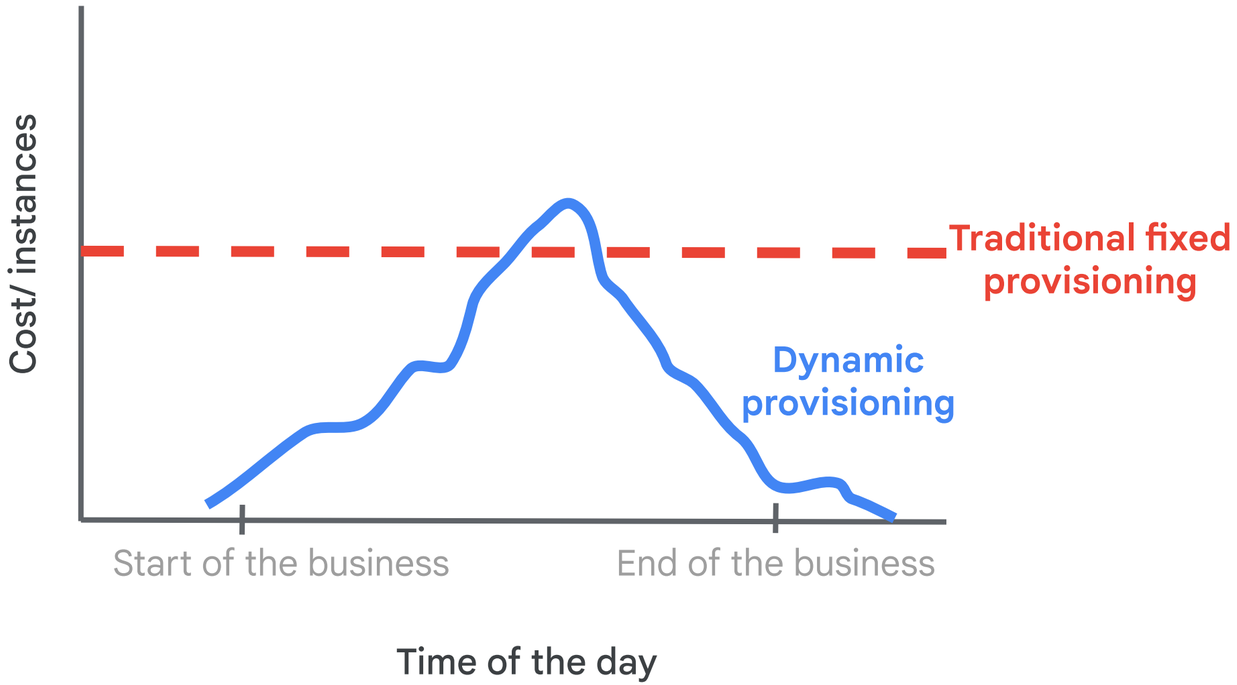


- Describe best practices for development and operations teams to ensure successful solution deployment.
- Explain methods to interact with Google Cloud programmatically.
- Explain methodologies for managing configuration and code updates and tools available for monitoring and analyzing KPIs.

Cymbal Direct

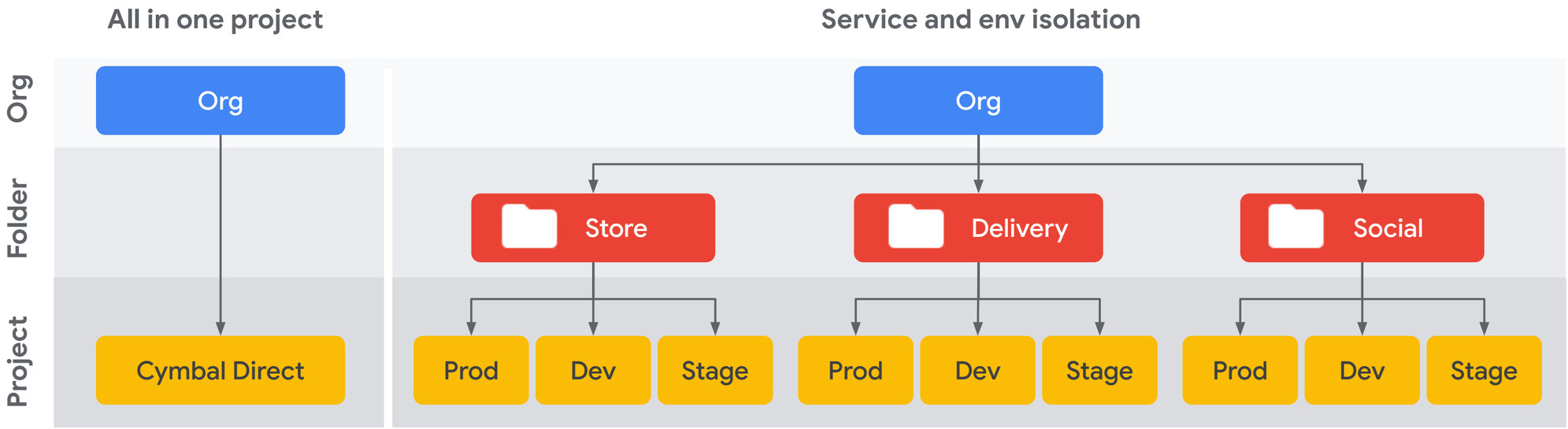
Best practices

Describe best practices for development and operations teams to ensure successful solution deployment.



Programmatic access

Explain methods to interact with Google Cloud programmatically



Cloud emulators: Spanner, Bigtable, Pub/Sub etc...

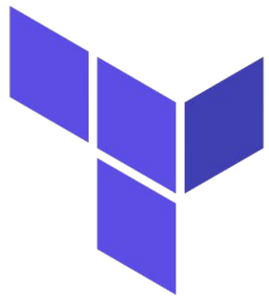
Configuration and code updates

Deployment Manager vs Terraform

- DM is in *maintenance mode* and customers are **discouraged from using it**
- The recommended way to define infrastructure is Terraform. Google actively maintains the Google Terraform module
- Customers using Deployment Manager will be able to migrate using dm-convert
- For customers heavily invested in Kubernetes, **Config Connector** is a Terraform alternative. It enables them to manage GCP resources with Kubernetes YAMLs. However it doesn't have feature parity with Terraform yet.

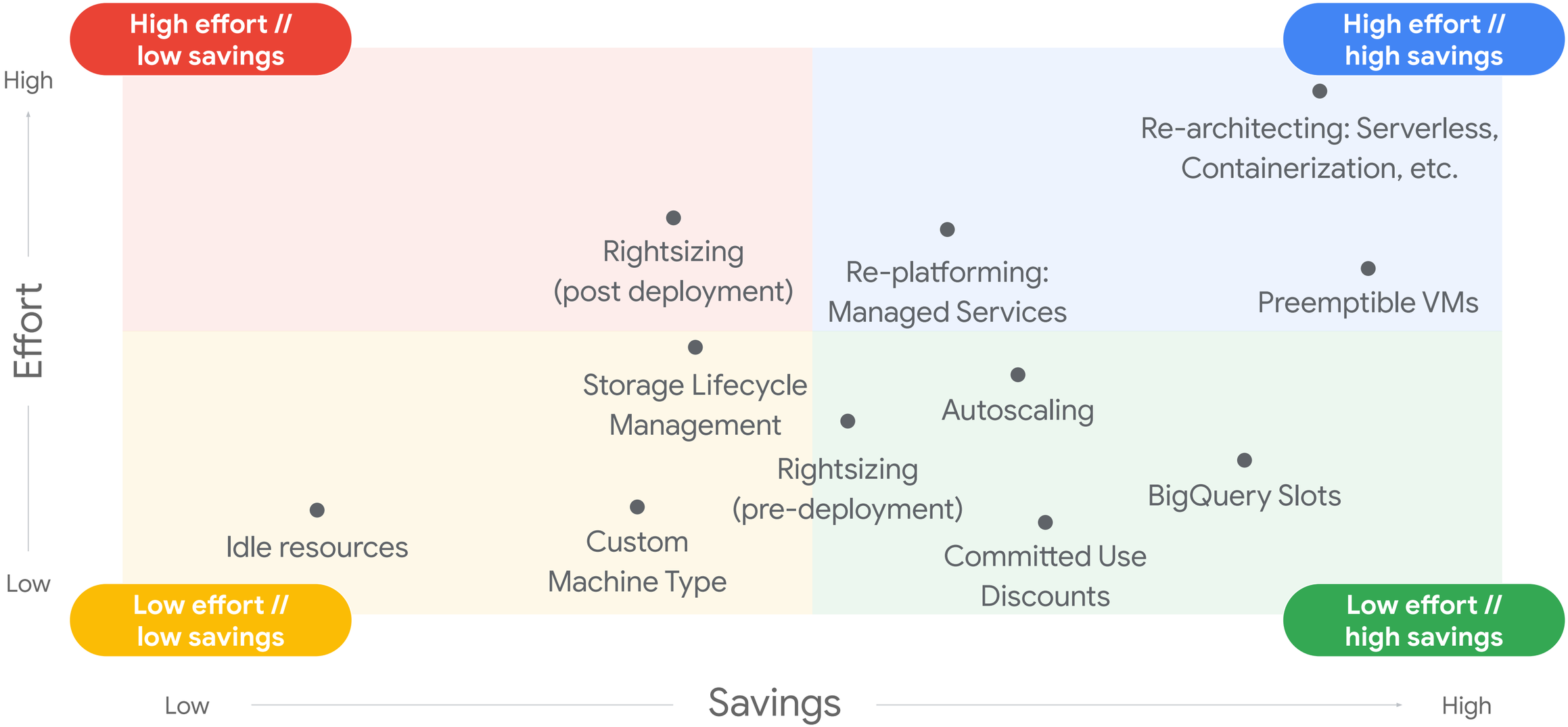
Cloud Deployment Manager

vs



Terraform

Cost Optimization Matrix



What is FinOps?

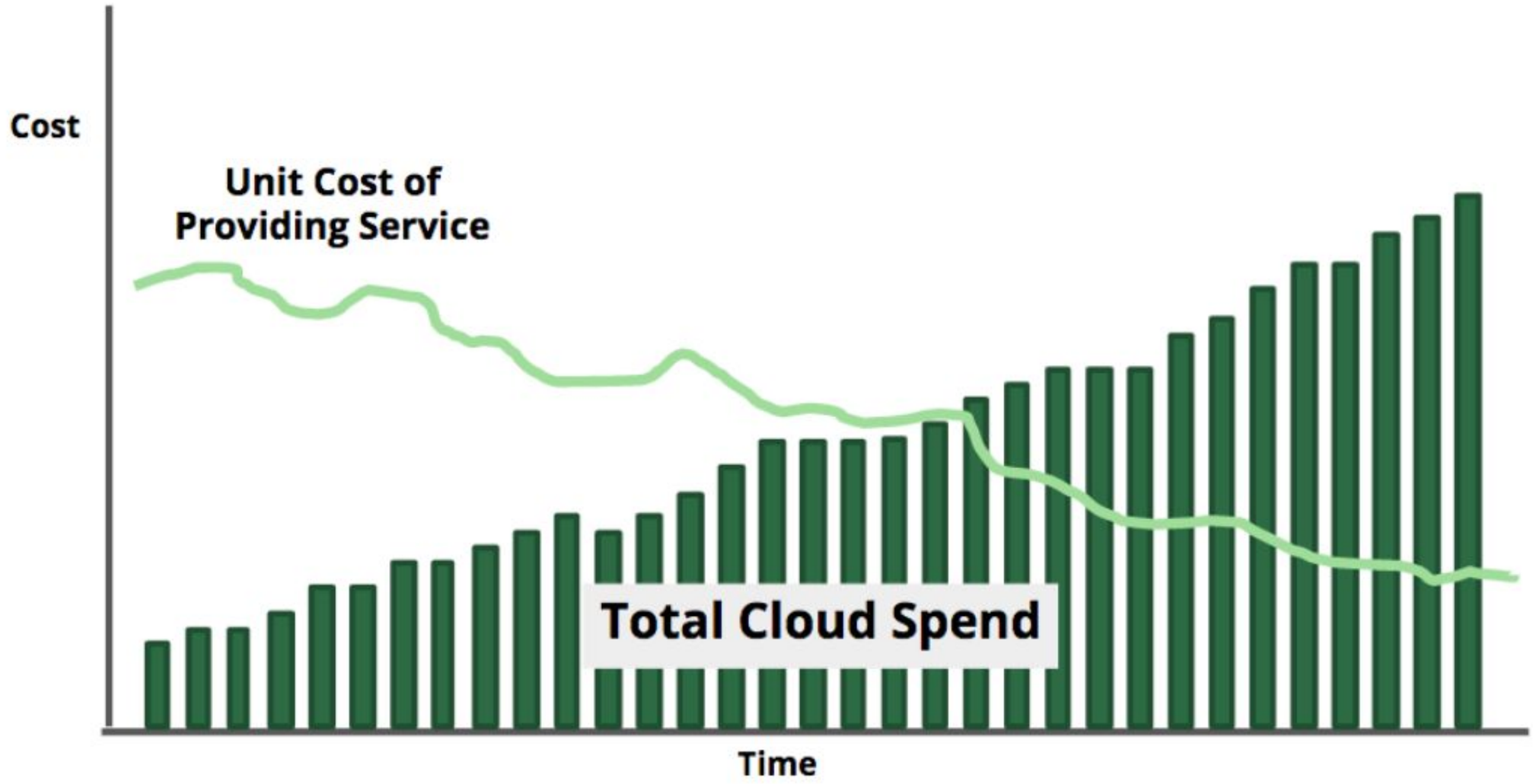
Cloud FinOps is an operational framework and cultural shift that brings technology, finance, and business together to **drive financial accountability and accelerate business value realization**

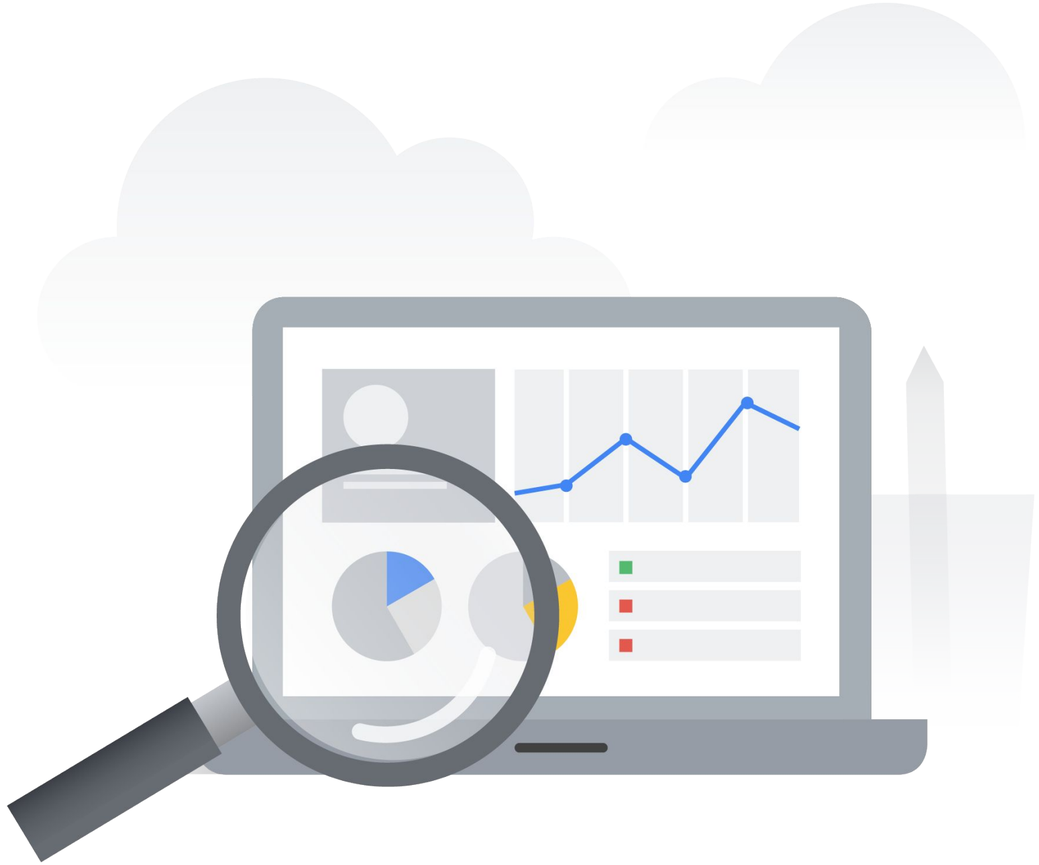
This is achieved through:

- A set of processes (what you do)
- A set of behaviours (how you do it)

...FinOps brings people together and empowers your teams to understand their individual impact to the bottom line...and act on it.

FinOps is NOT about spending less it's about spending wise!



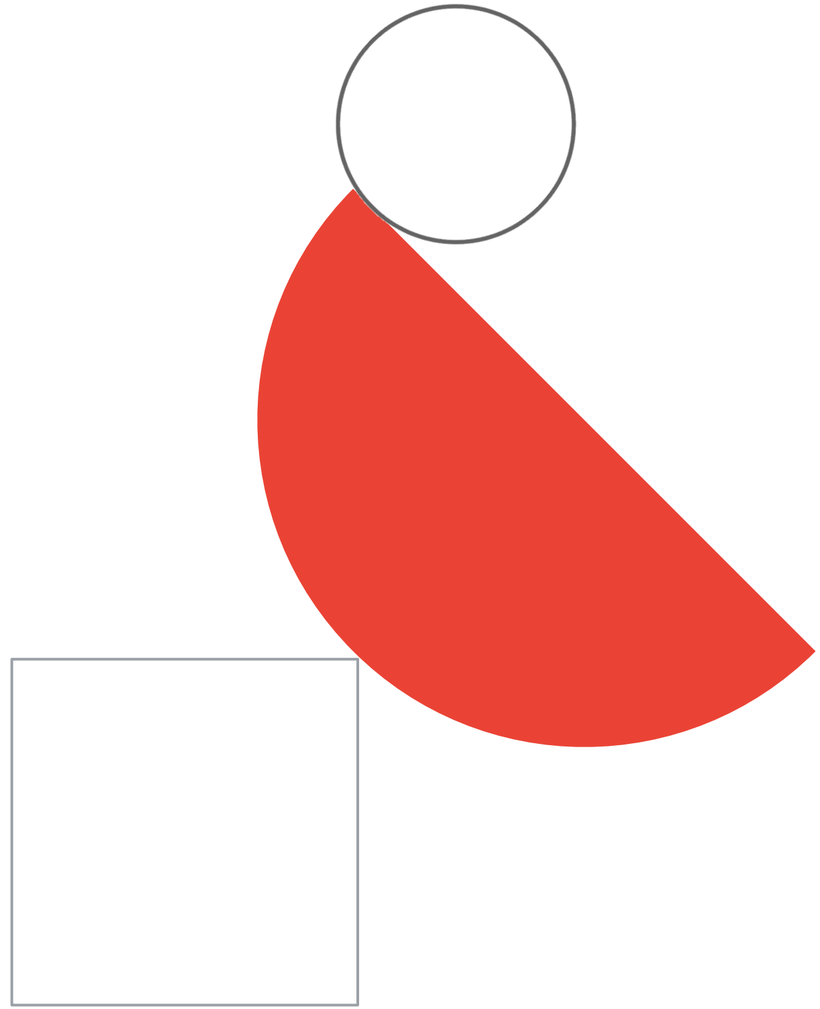


Monitoring and KPIs

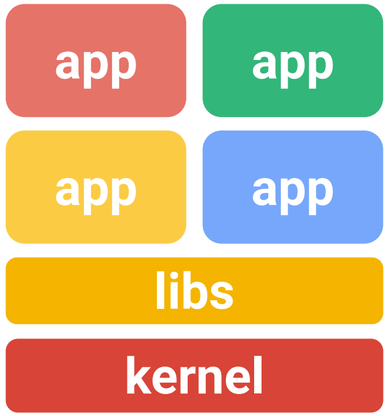
Tools available for monitoring and analyzing KPIs.

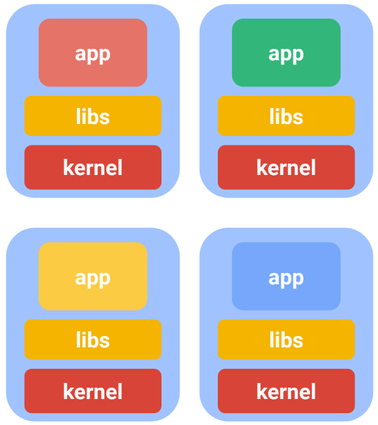
Tips and comments to the course content

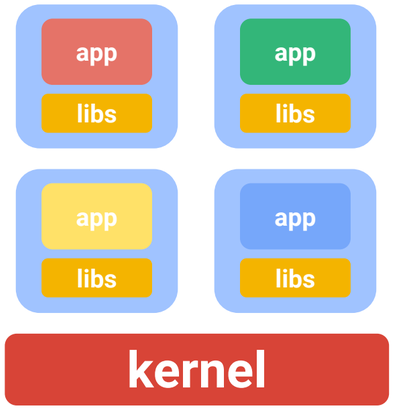
This week's focus: GKE



From physical servers to containers







Physical Machine

- No isolation
- Common libs
- Highly coupled apps and OS

VMs

- Isolation
- No common libs
- Expensive and inefficient
- Hard to manage

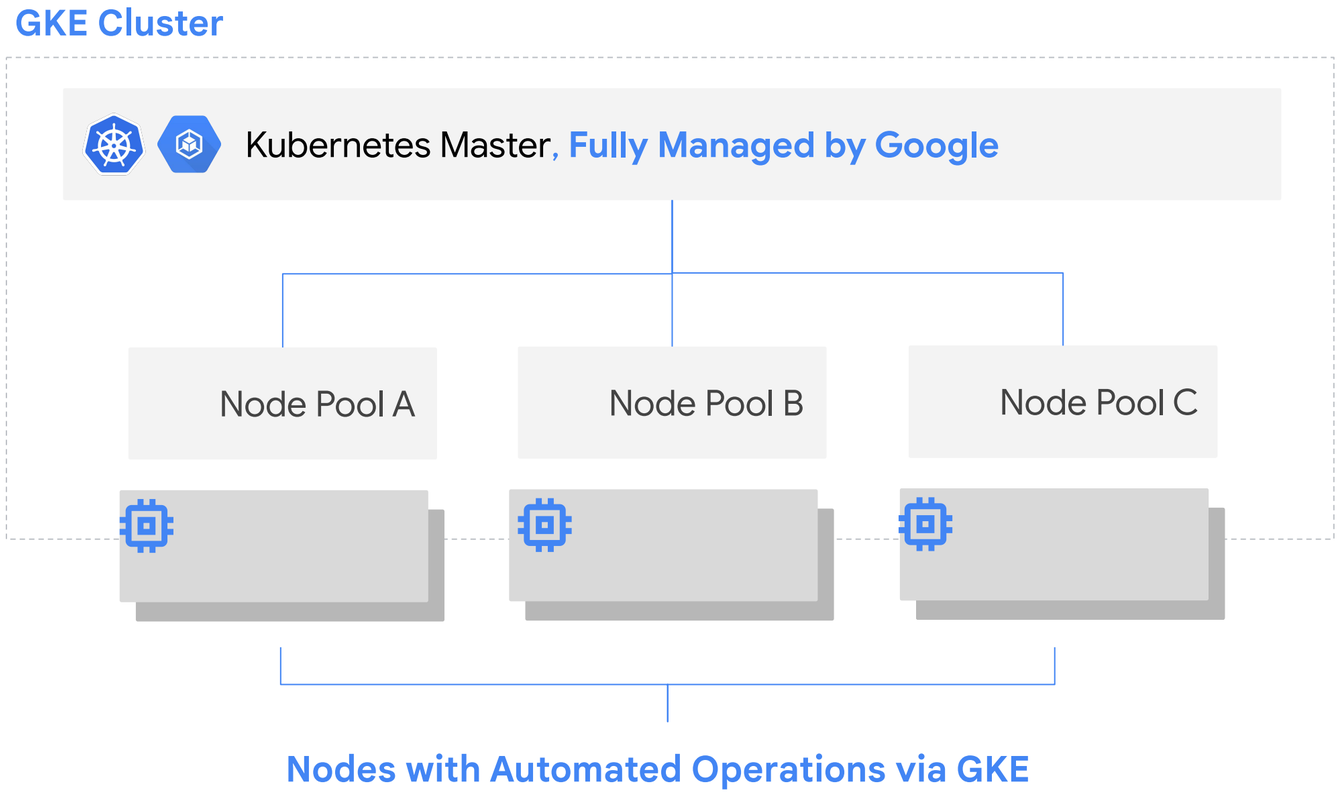
Containers

- Isolation
- No common libs
- Less overhead
- Less dependency on host OS

Google Kubernetes Engine

GKE is Google Cloud's Kubernetes Platform

deep integration with Google Cloud Platform features and services



Exam Tip: *I can't stress enough how important it is to understand Kubernetes concepts. Commit at least few hours for learning GKE - especially if you're not familiar with this technology. Slides below will give you a high level overview, but you should be much more familiar with this topic to feel comfortable during the exam.*

GKE: Autopilot Mode

GKE manages entire underlying infrastructure of the cluster including the nodes

High availability

Regional cluster; Regular Release Channel; Auto-Update; Auto-Repair; Surge Upgrade;

Network

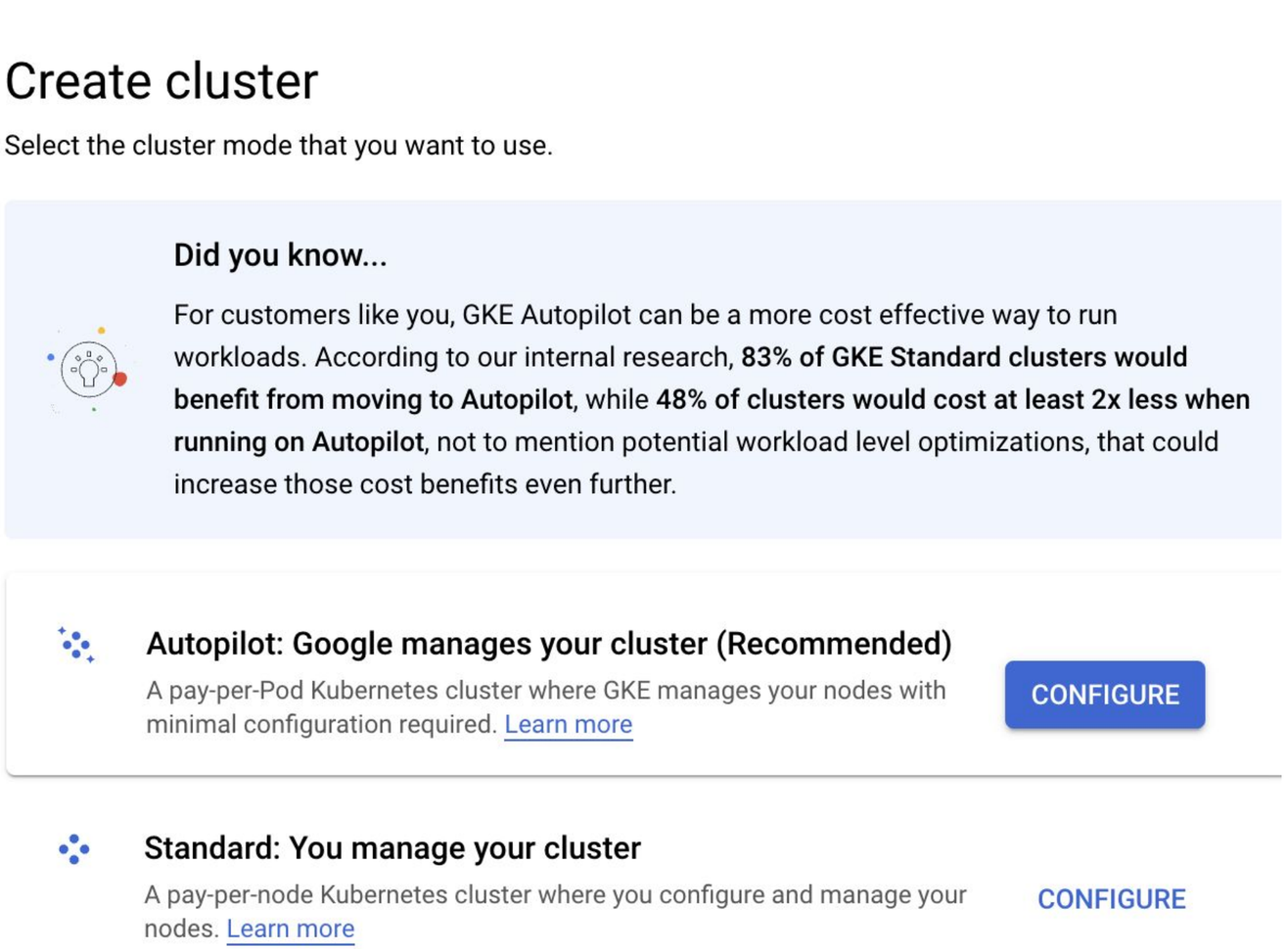
VPC Native (alias IP); IP-friendly (limit cluster size/ pods per node); full network flexibility

Highly Scalable

Node Auto Provision; Horizontal Pod Autoscaler; Vertical Pod Autoscaler

Secured by default

Workload Identity; Shielded Nodes; Secure-boot-disk; COS and Containerd, block known unsecure features.



PDB (Pod Disruption Budget)

A **PDB (Pod Disruption Budget)** limits the number of pods of a replicated application that can be taken down **simultaneously** from **voluntary** disruptions.

An Application Owner can create a **PodDisruptionBudget** object (PDB) for each application.

Examples

1. A quorum-based application would like to ensure that the number of replicas running is never brought below the number needed for a quorum.
2. A web front end might want to ensure that the number of replicas serving load never falls below a certain percentage of the total.

GKE fleet management **respects** PDB when maintenance events are initiated.

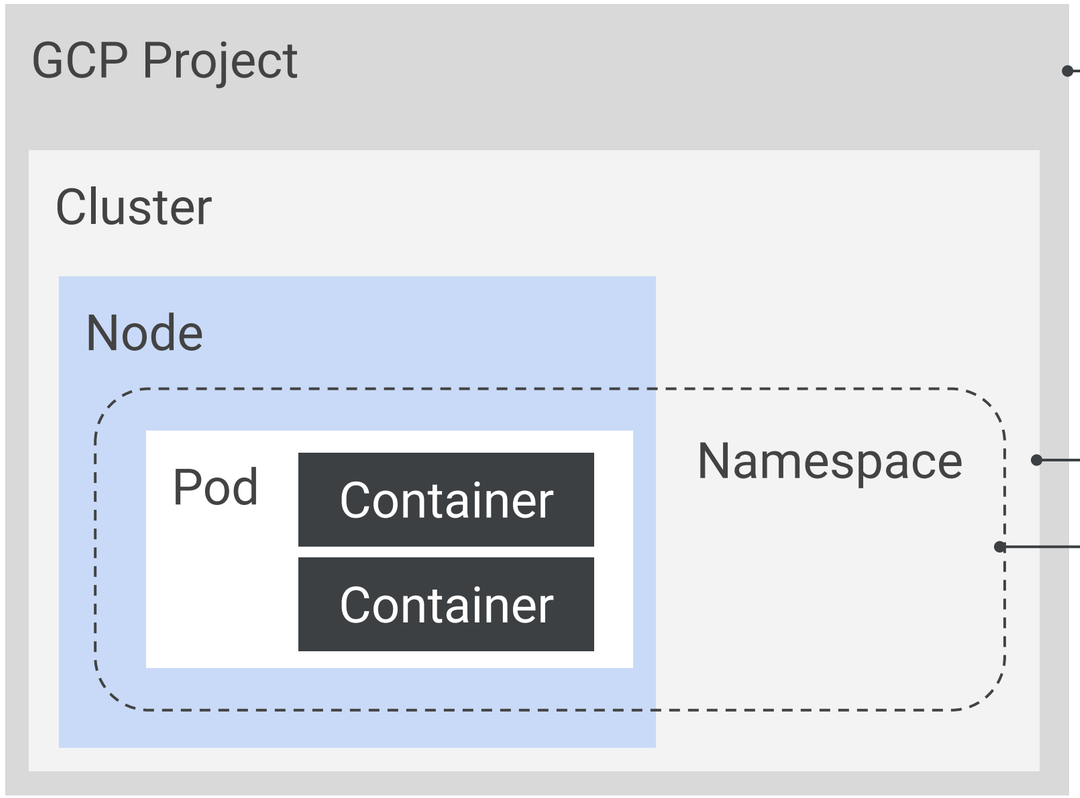
```
kind: PodDisruptionBudget
metadata:
 name: km-pdb
spec:
 minAvailable: 2
 selector:
   matchLabels:
     app: kobimysql
```

```
maxUnavailable: 1
```

Best practices for GKE upgrades

1. **Setup multiple environments:** at a minimum pre-production and production clusters
2. **Enroll Clusters in Release Channels**: Stable or Regular release channels for production clusters.
3. **Create continuous upgrade strategy**: Receive updates about new GKE versions through cluster upgrade notifications through Pub/Sub
4. **Schedule maintenance windows and exclusions**: to increase upgrade predictability
5. **Set tolerance for disruption:** To ensure that pods sufficient number of replicas, use Prod Disruption Budget

GKE: Using IAM and RBAC



Use IAM at the project level
Set roles for

- Cluster Admin: manage clusters
- Container Developer: API access within clusters

Use RBAC at the cluster and namespace level
Set permissions on individual clusters and namespaces

GKE: Defining resource quotas

- By default, pods **run with unbounded CPU and memory requests/limits**
- When several tenants share a cluster with a fixed number of nodes, there is a concern that one tenant could use more than its fair share of resources
- Resource quotas limit total memory, CPU, and storage that pods can use, and how many objects of each type (pods, load balancers, `ConfigMaps`, etc.) on a per-namespace basis

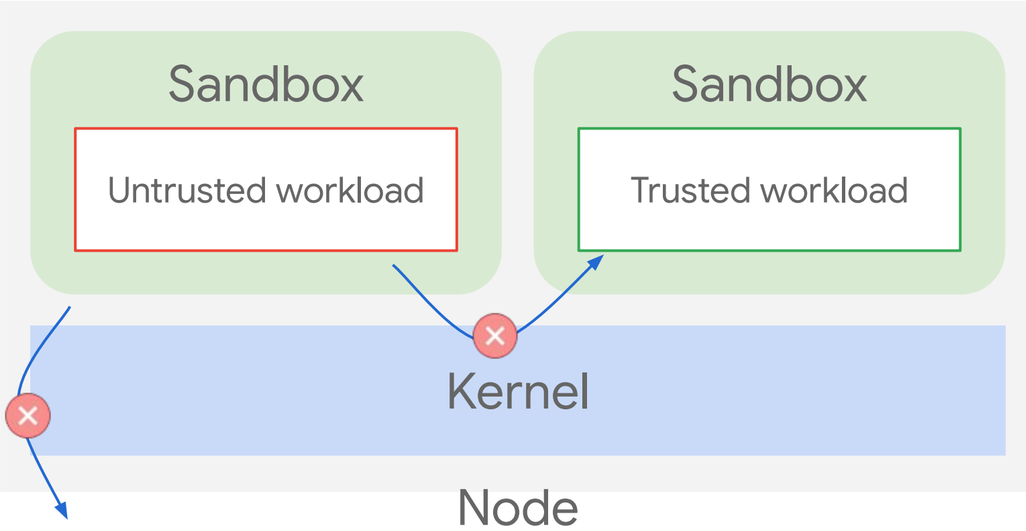
```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
spec:
  hard:
    requests.cpu: “1”
    requests.memory: 1Gi
    limits.cpu: “2”
    limits.memory: 2Gi
```

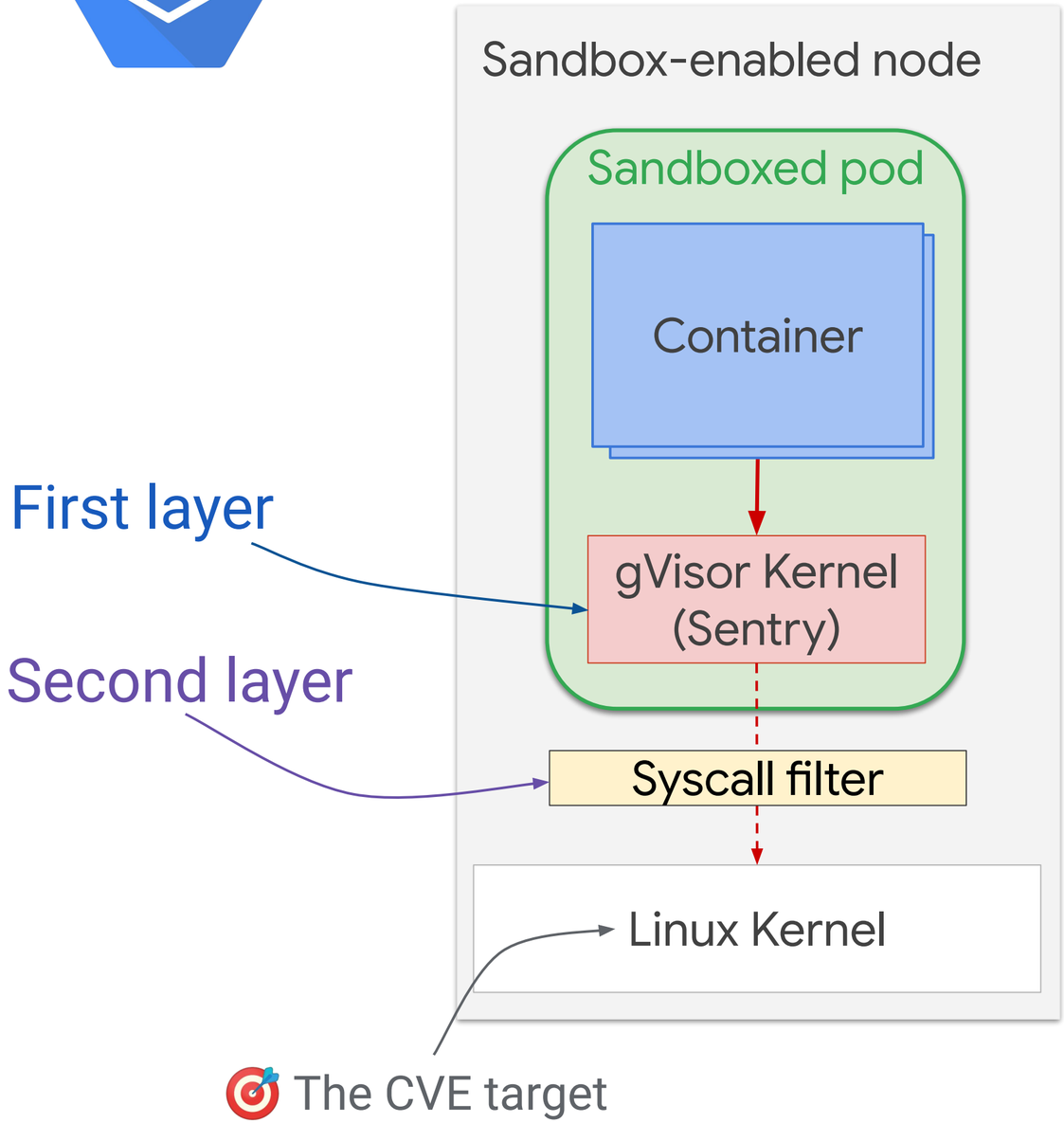
GKE Sandbox

Run **trusted and untrusted** workloads on the same node

Rather than achieving isolation via separate VMs, you can run workloads of different trust levels on the same node

Performance improvements from not having to allocate a new cluster to achieve isolation





Subnet sizes

Subnet size for nodes	Maximum nodes	Maximum Pod IP addresses needed	Recommended Pod address range
/29	4	1,024	/21
/28	12	3,072	/20
/27	28	7,168	/19
/26	60	15,360	/18
/25	124	31,744	/17
/24	252	64,512	/16
/23	508	130,048	/15
/22	1,020	261,120	/14
/21	2,044	523,264	/13
/20	4,092	1,047,552	/12
/19	8,188	2,096,128	/11 (maximum Pod address range)

*Assuming the default maximum of 110 pods per node

Subnet Size Example

Subnet size for nodes	Maximum nodes	Maximum Pod IP addresses needed	Recommended Pod address range
/29	4	1,024	/21

$2^{(32-29)} = 8$ (4 of these are reserved for GCP)

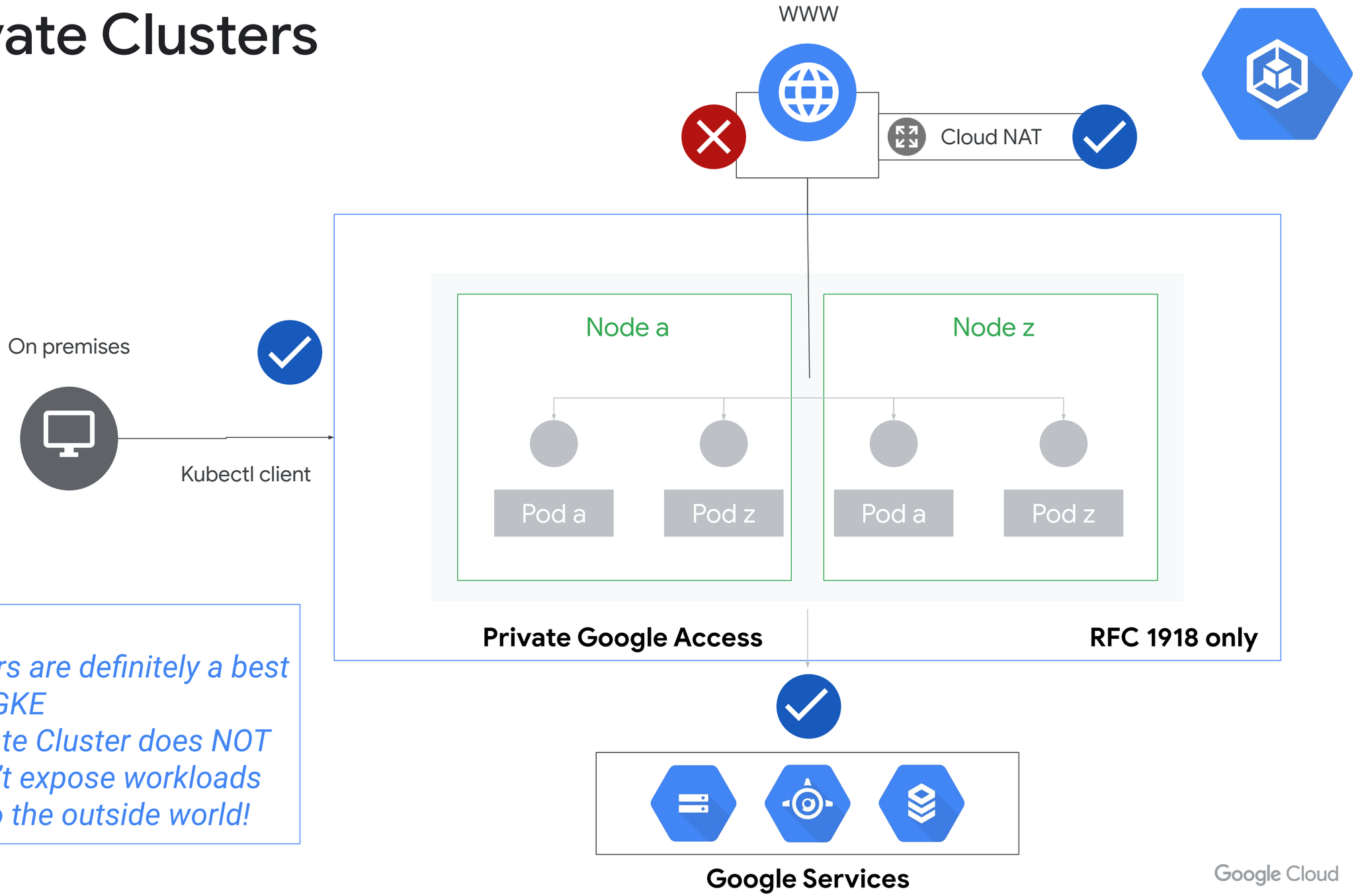
110 pods running in each node -> **4 * 110 = 440**

Twice number of IPs per pod **440 * 2 = 880**

2^{10} number of IPs for pods

*Assuming the default maximum of 110 pods per node

GKE: Private Clusters



Exam Tip:

- *Private Clusters are definitely a best practice with GKE*
- *Having a Private Cluster does NOT mean you can't expose workloads via Services to the outside world!*

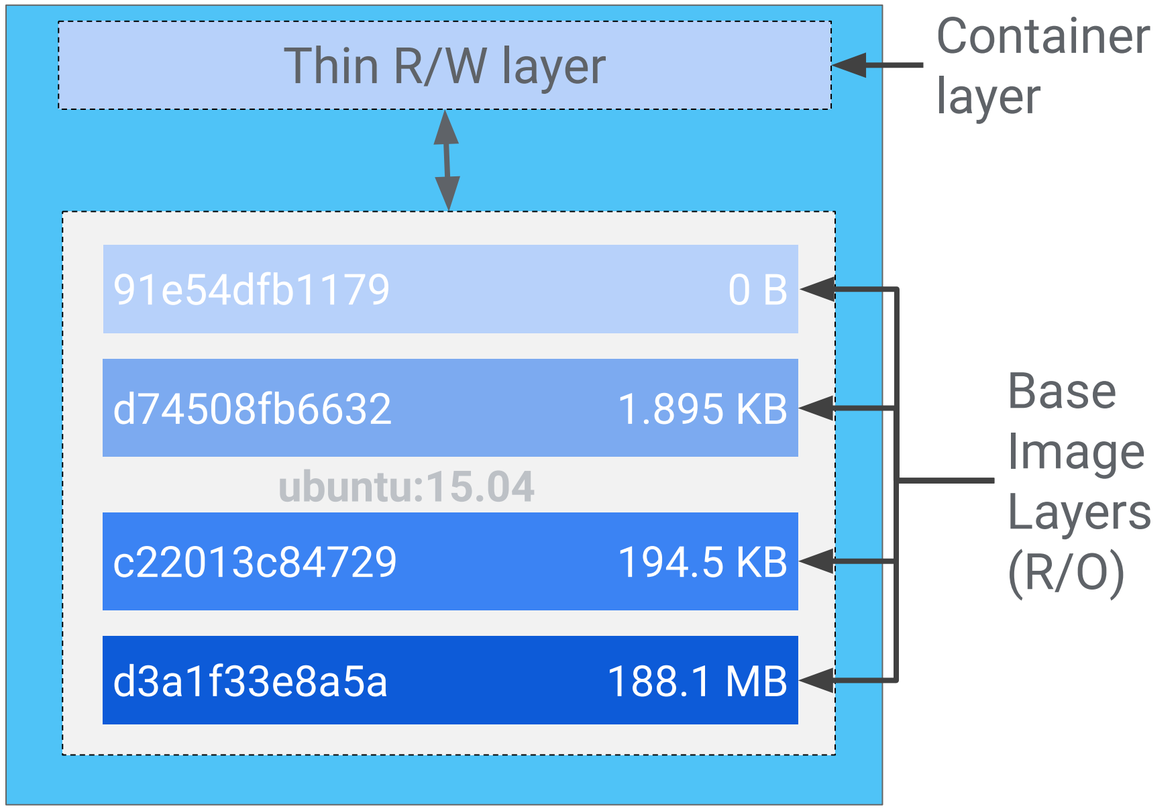
Container images: best practices.

Exam Tip: *Each instruction in a Dockerfile creates a layer.*

Dockerfile

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Container



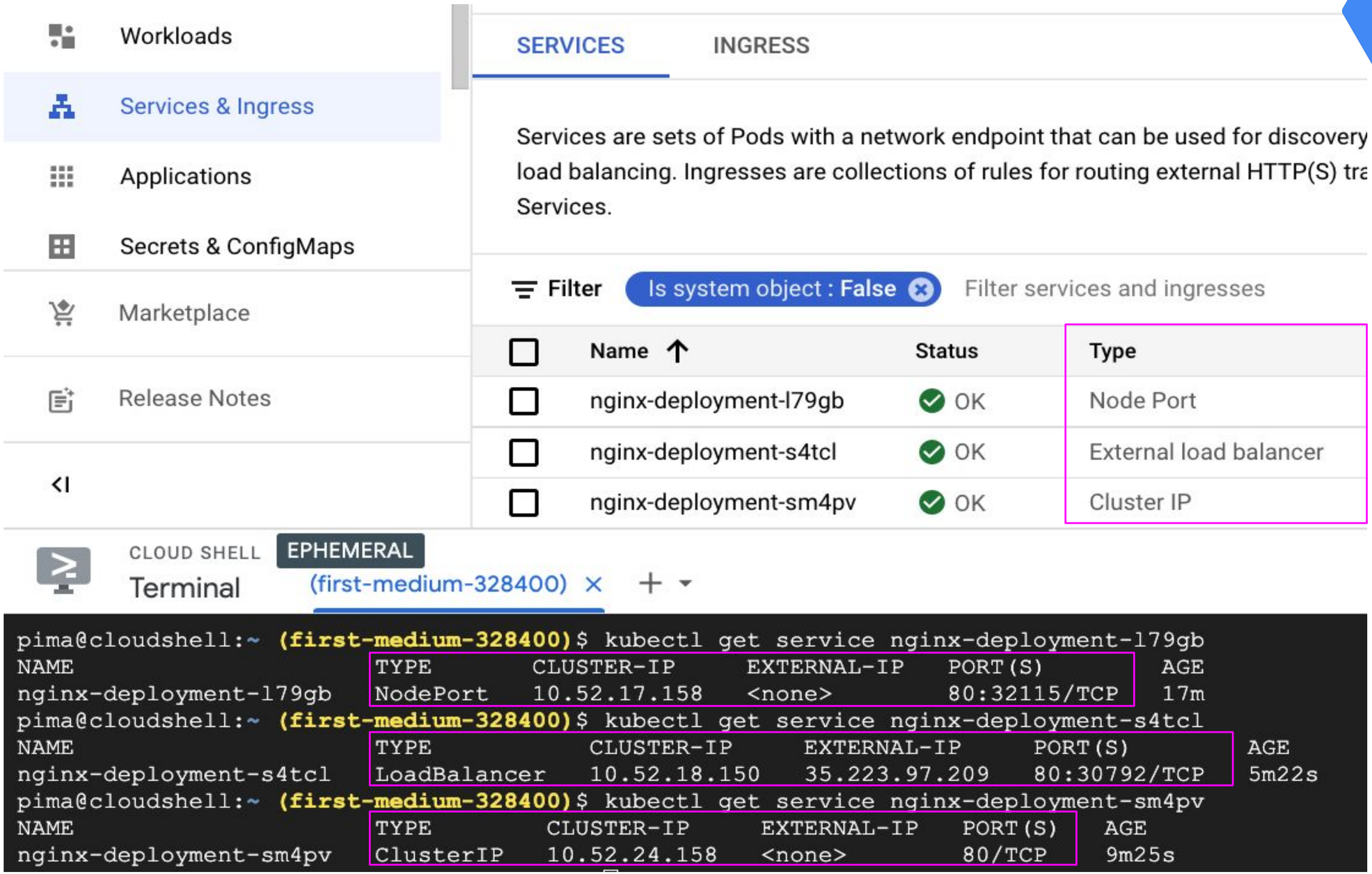
docker commands

```
$> docker build -t py-web-server
.
$> docker run -d py-web-server
$> docker images
$> docker ps
$> docker logs <container id>
```

Exam Tips: *Here are best practices for building container images:*

- *Use the smallest base image possible (when new versions are rolled out, only smallest image layers are changed). Eg. use "alpine" image rather than "centos" or "ubuntu" if possible.*
- *Use multi-stage builds (app can be built in a first "build" container and the result can be used in another container)*
- *Try to create images with common layers (if a layer already exists on a cluster, it does not have to be downloaded)*

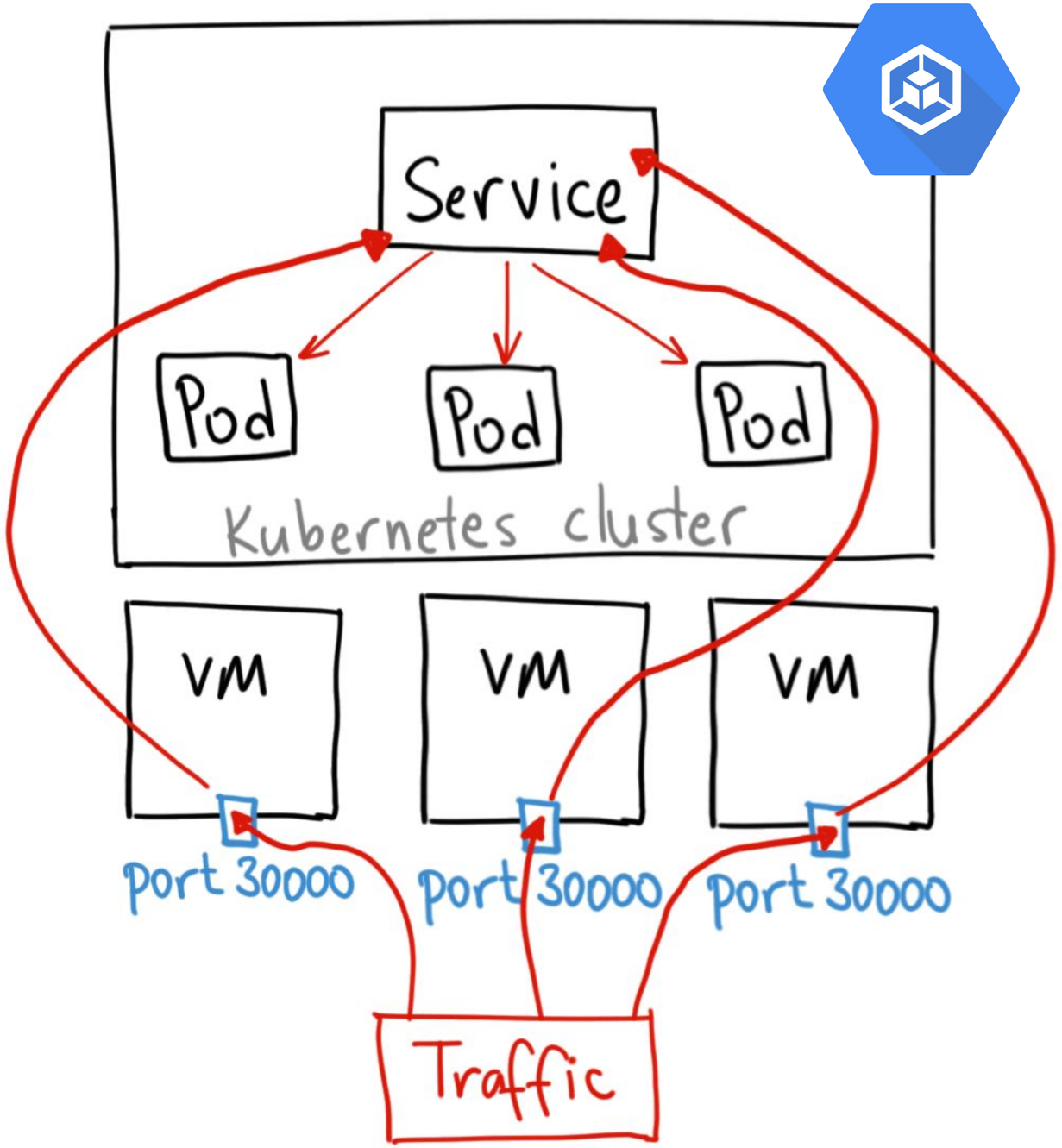
GKE: types of services



NodePort

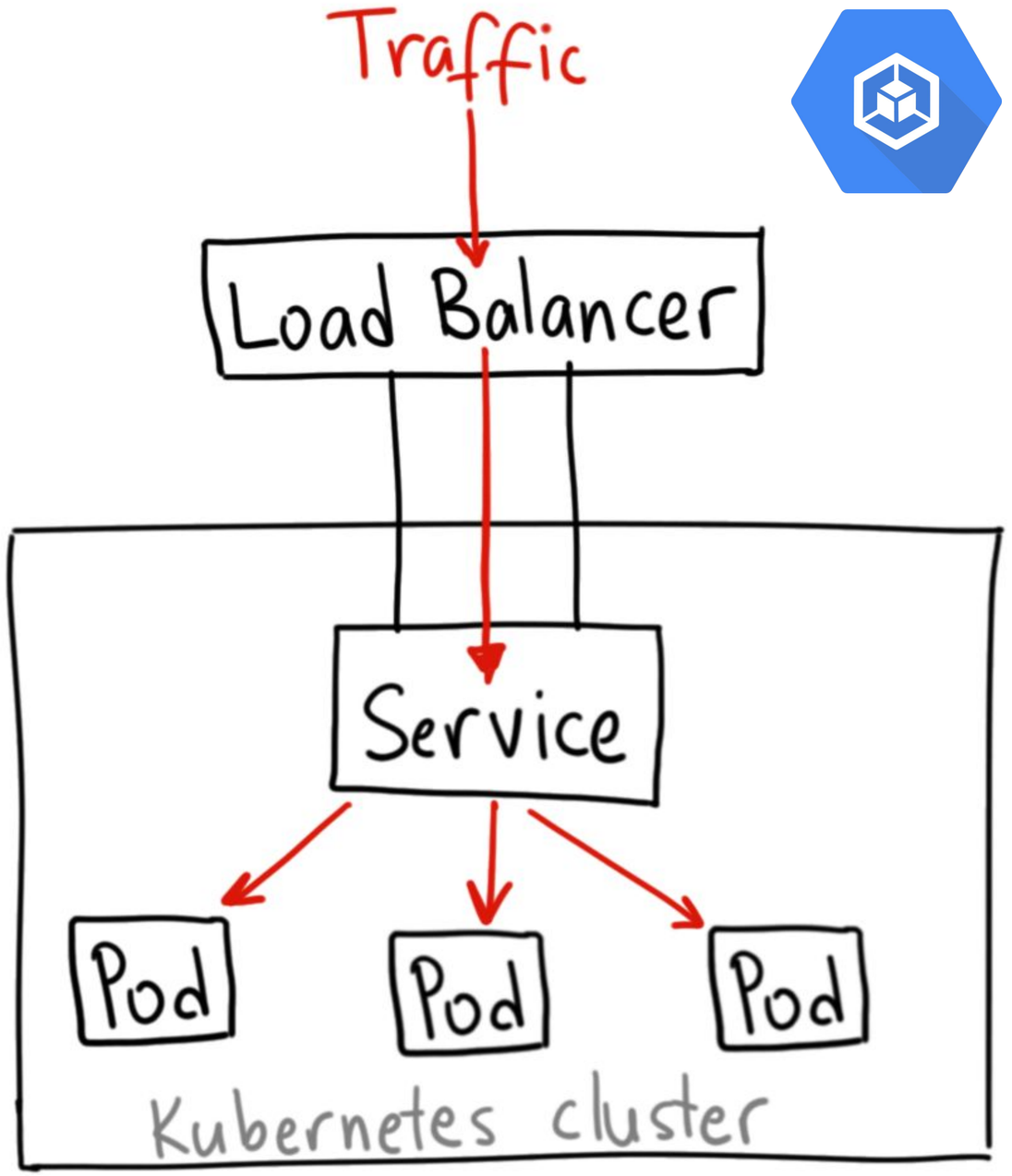
- Simple to understand model
- Portable: No external dependencies

- Limited to ports 30,000-32,767
- Clients need to adapt to Node IP changes
- Cannot use advanced features
- Suboptimal load balancing



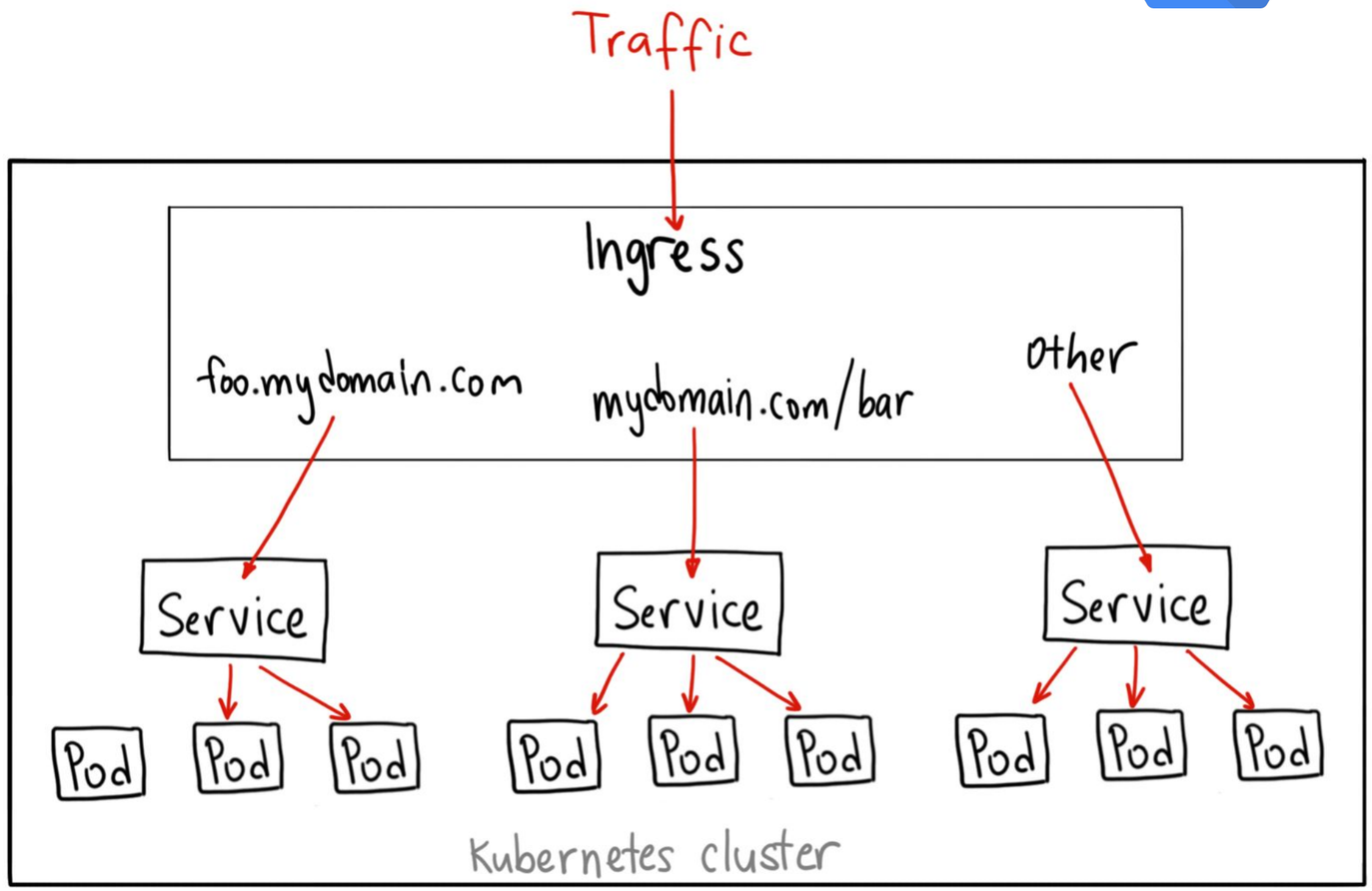
LoadBalancer

- A single VIP owned by a real LB
- Layer 4 (TCP/UDP) & Layer 3
- Exposed as an internal or external VIP



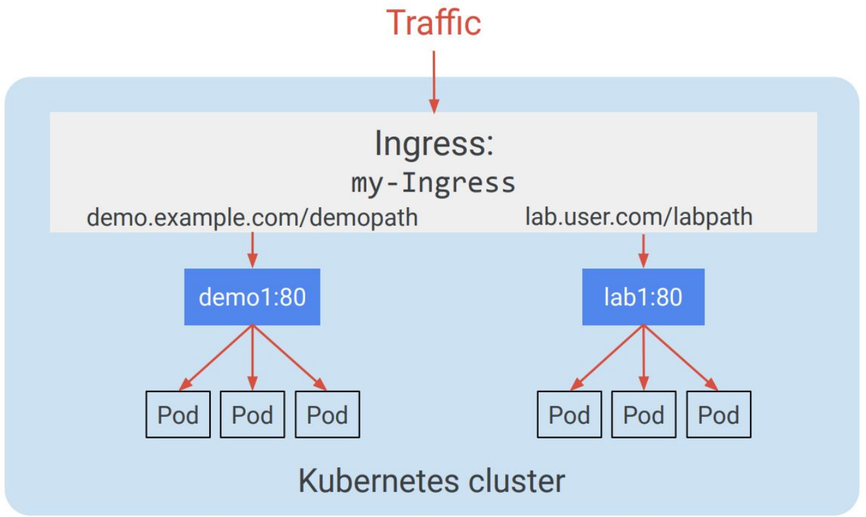
Ingress

- Portable across implementations (nginx-ingress, gce-ingress, aws-elb-ingress)
- A layer 7 representation
- HTTP, HTTPs, Path/Header/Host matching



Ingress: a service for Services

1

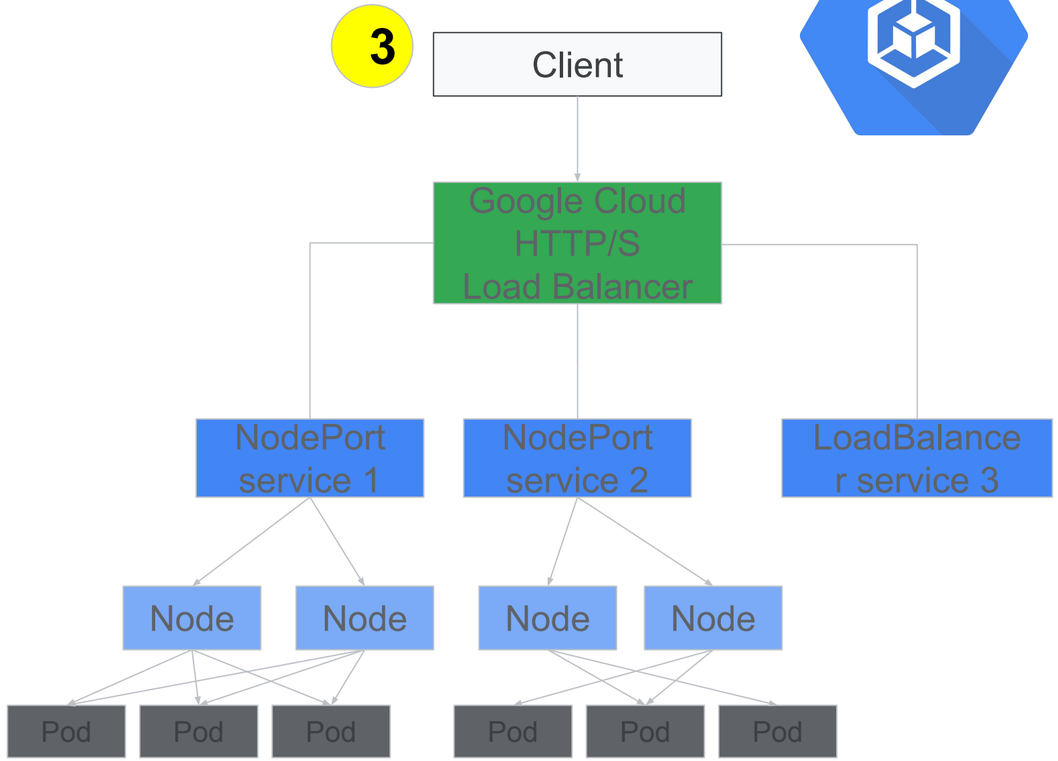


2

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-Ingress
spec:
  rules:
  - host: demo.example.com
    http:
      paths:
      - path: /demopath
        backend:
          serviceName: demo1
          servicePort: 80
```

```
- host: lab.user.com
  http:
    paths:
    - path: /labpath
      backend:
        serviceName: lab1
        servicePort: 80
```

3

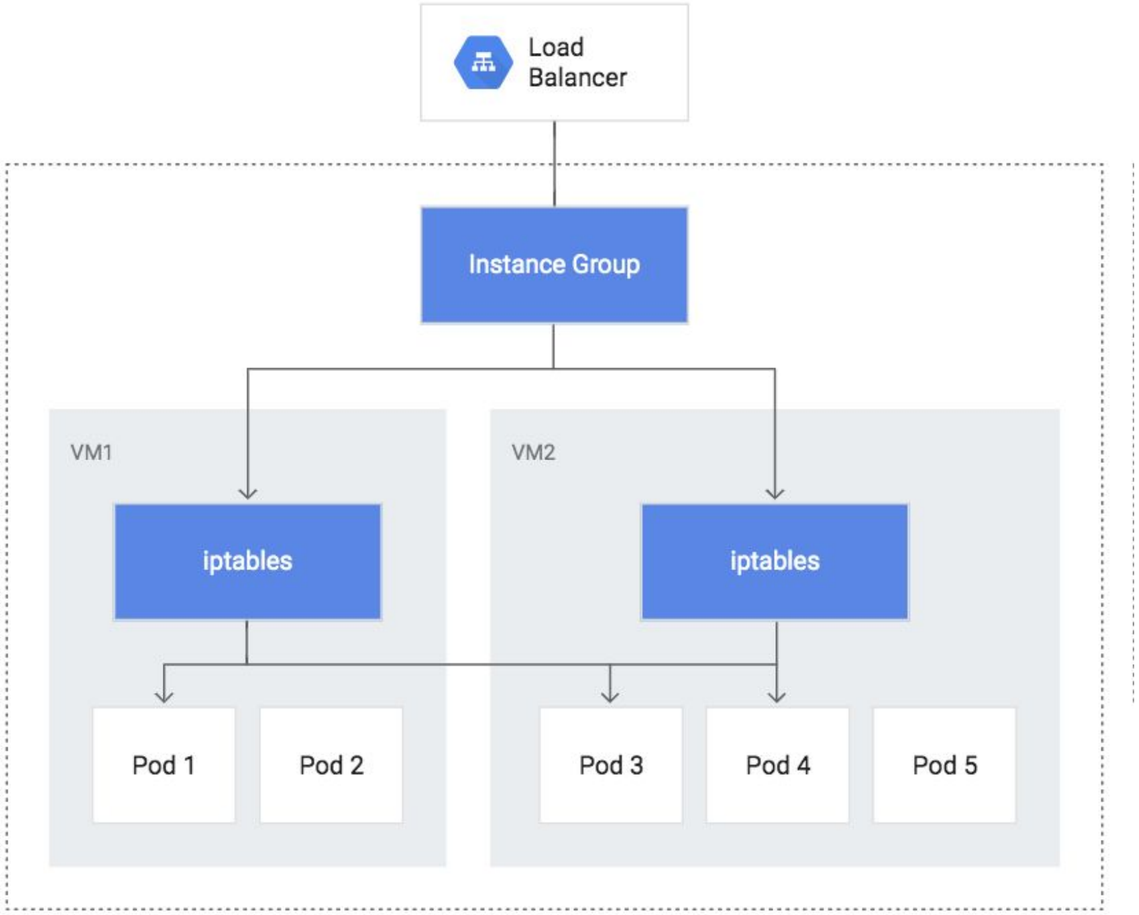


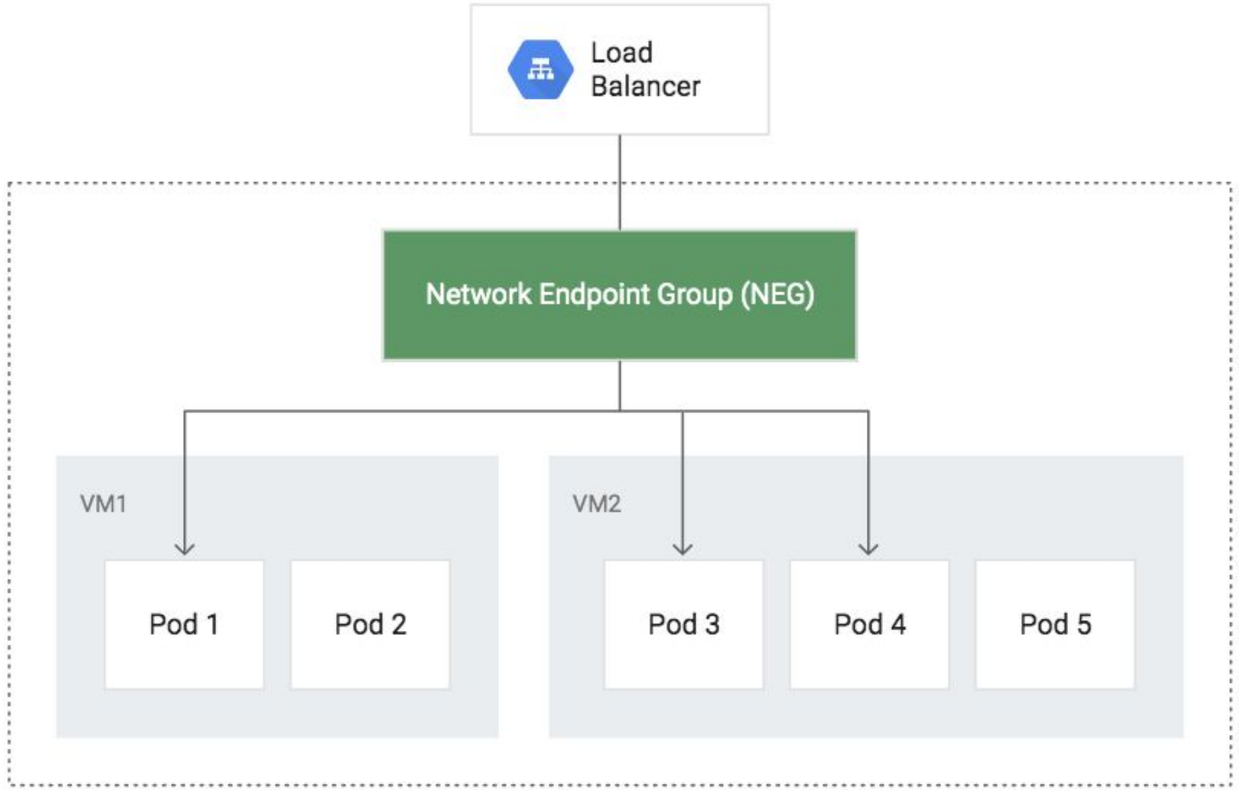
Additional Ingress features

- TLS termination
- Multiple SSL certificates
- HTTP/2 and gRPC
- Multi-cluster and multi-region support

Exam Tip: *Apart from single Services, there is so-called Ingress resource (it has **nothing** to do with ingress network traffic!) that handles all the incoming traffic and redirects each request to one of Services*

Ingress: standard (non-NEG) vs NEG





Exam Tip: *NEG is often preferred as a container-native load balancing type.*

Ingress natively supports many Google Cloud services

IAP

Cloud Armor

Cloud CDN

Exam Tip: *Since Ingress in GKE is actually based on global HTTP(S) Load Balancer, you can use additional services that complement this LB: IAP, Cloud Armor and Cloud CDN.*

GCP API access from k8s *without* Workload Identity

[Authenticate to Google Cloud using a service account | Kubernetes Engine]

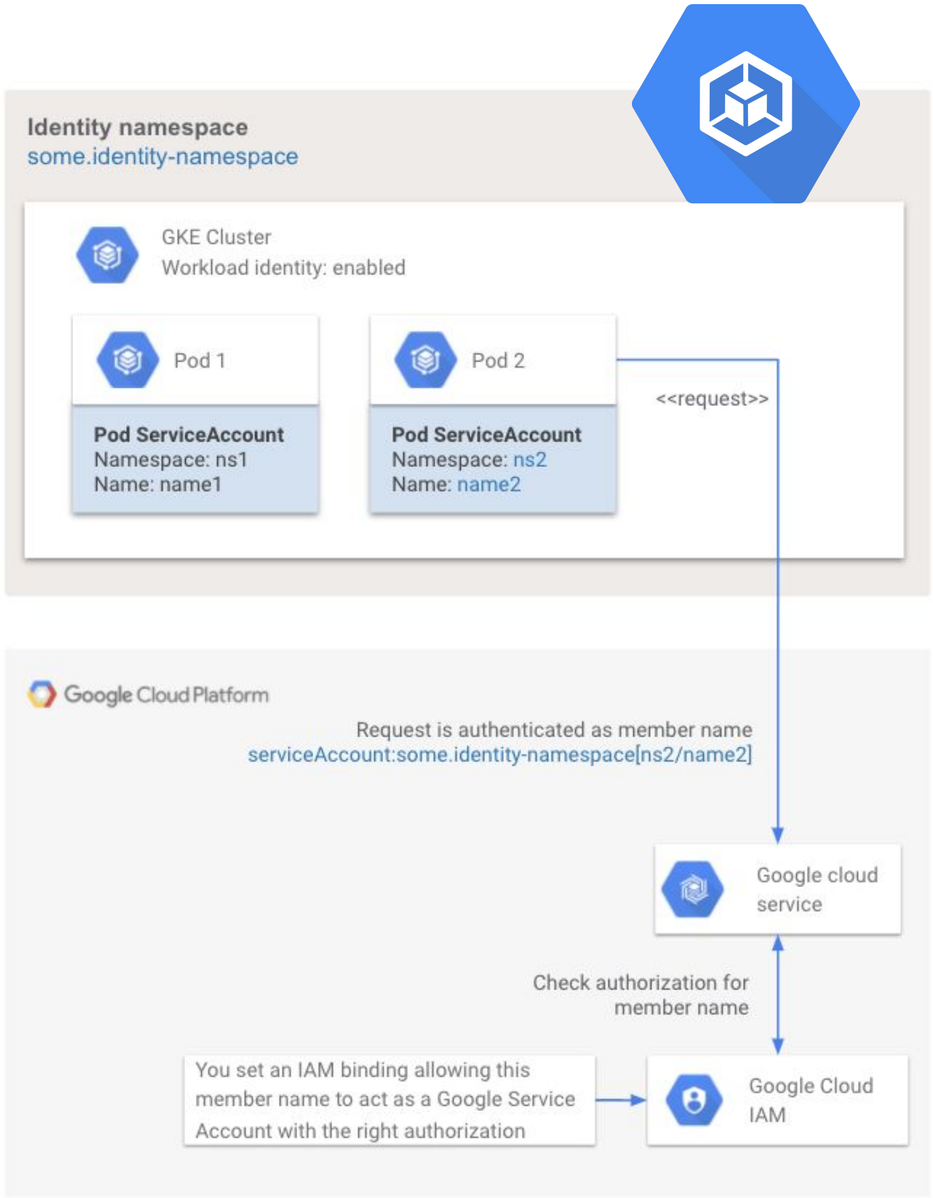
- Create a GCP Service Account (GSA)
- Create Keys for GSA
- Import GSA Keys as a k8s Secret
- For the k8s Workload:
  - Define a Volume with the Secret
  - Mount the Volume inside the container
  - Point $GOOGLE_APPLICATION_CREDENTIALS at the key file
- Workload can now authenticate to GCP APIs as the GSA

=> toilsome to setup & hard to secure

API access *with* Workload Identity

- Enable Workload Identity for the GKE cluster
- Run workload using a dedicated k8s service account (KSA)
- Grant KSA access to desired GCP resources using IAM roles
- Workload can now access GCP APIs by presenting (short-lived, auto-rotated) KSA tokens

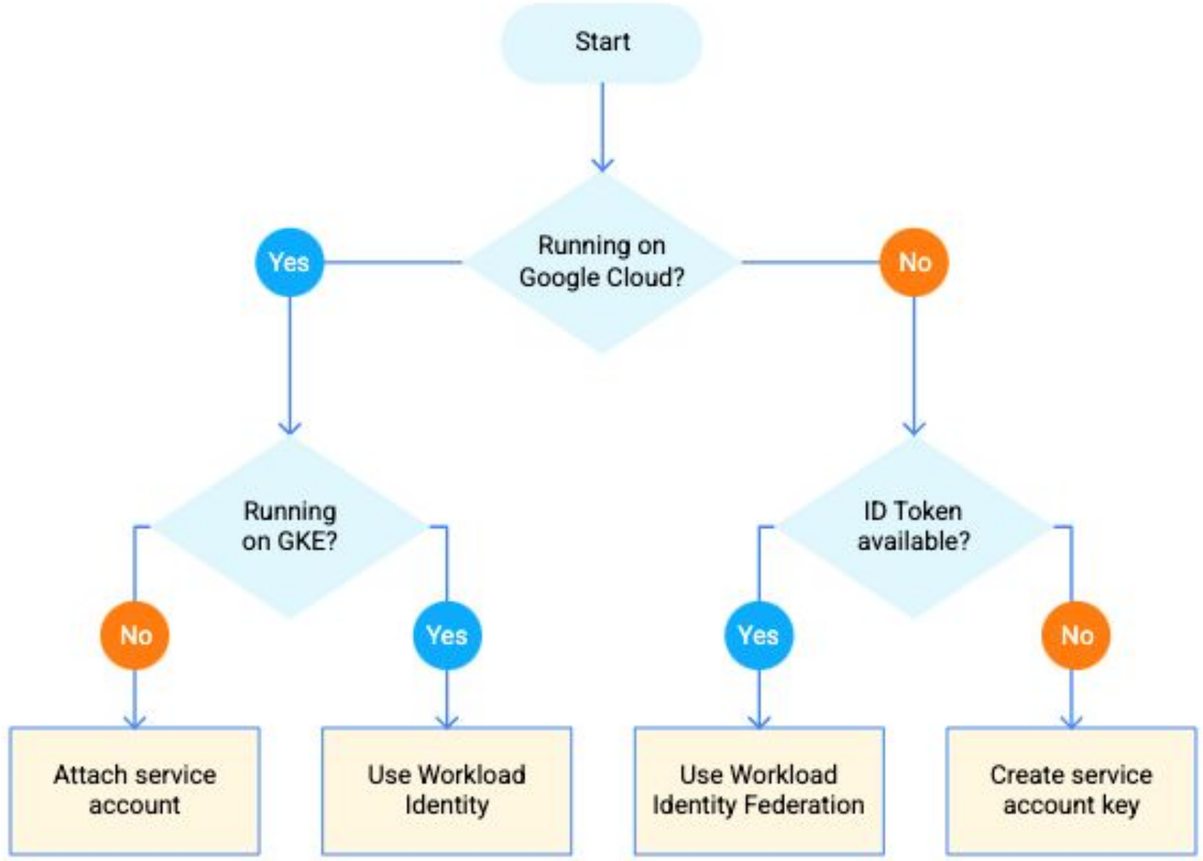
It just works!



Exam Tip: *Workload Identity is a best practice for a GKE which needs to access other GCP APIs.*

Workload Identity vs Workload Identity Federation

- Those are two different things! Both aim at limiting usage of Service Account keys, but:
  - Workload Identity = used when microservices deployed to your GKE cluster need to access other GCP resources / APIs.
  - Workload Identity Federation = when some services of yours deployed outside of GCP (in on-premises or other hyperscalers) need to access GCP resources / APIs.



How to bootstrap / change a GKE cluster

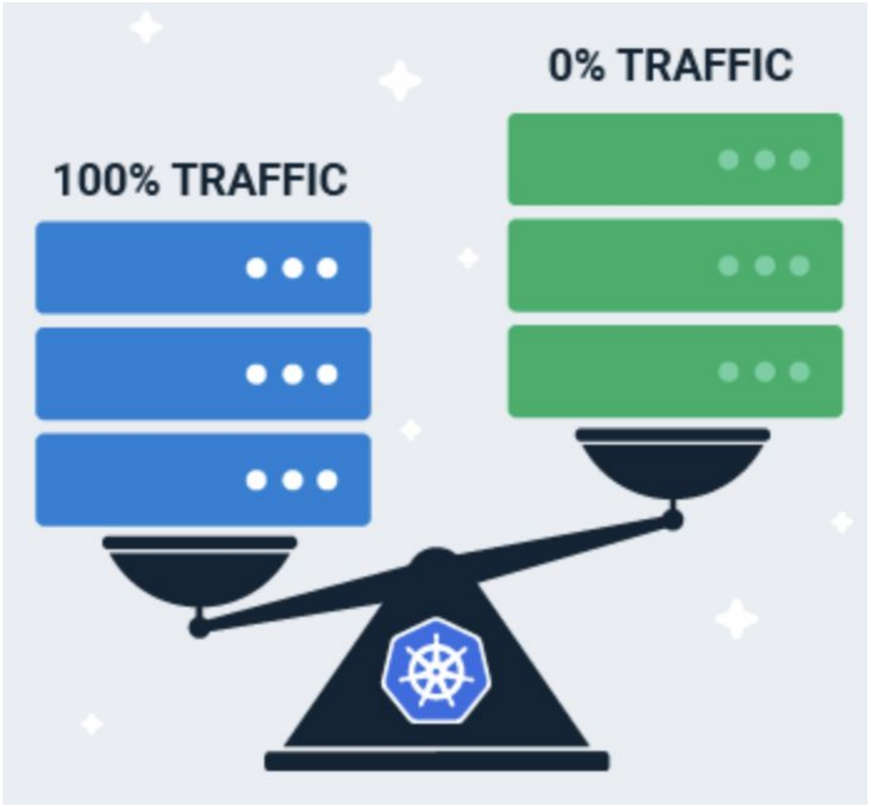
Exam Tips:

- *Make sure to differentiate:*
  - *When creating / modifying / deleting cluster (or its node pools), you should use gcloud command (since you're interacting with GCP to manage INFRASTRUCTURE for GKE). For example:*
    - *To create a cluster: 'gcloud container clusters create... '*
    - *To resize a cluster (change number of nodes): 'gcloud container clusters **resize** CLUSTER_NAME --node-pool POOL_NAME --num-nodes NUM_NODES'*
    - *To enable autoscaling on a node pool of existing cluster: 'gcloud container clusters **update** CLUSTER_NAME **--enable-autoscaling** --node-pool=POOL_NAME --min-nodes=MIN_NODES --max-nodes=MAX_NODES --region=COMPUTE_REGION'*
    - *To disable autoscaling on a node pool of existing cluster: 'gcloud container clusters update CLUSTER_NAME **--no-enable-autoscaling** --node-pool=POOL_NAME --region=COMPUTE_REGION'*
  - *When interacting with Kubernetes objects (eg. you'd like to deploy some Pods), you should use 'kubectl' command, eg:*
    - *To scale a deployment: 'kubectl autoscale deployment hello-app --cpu-percent=80 --min=1 --max=5'*

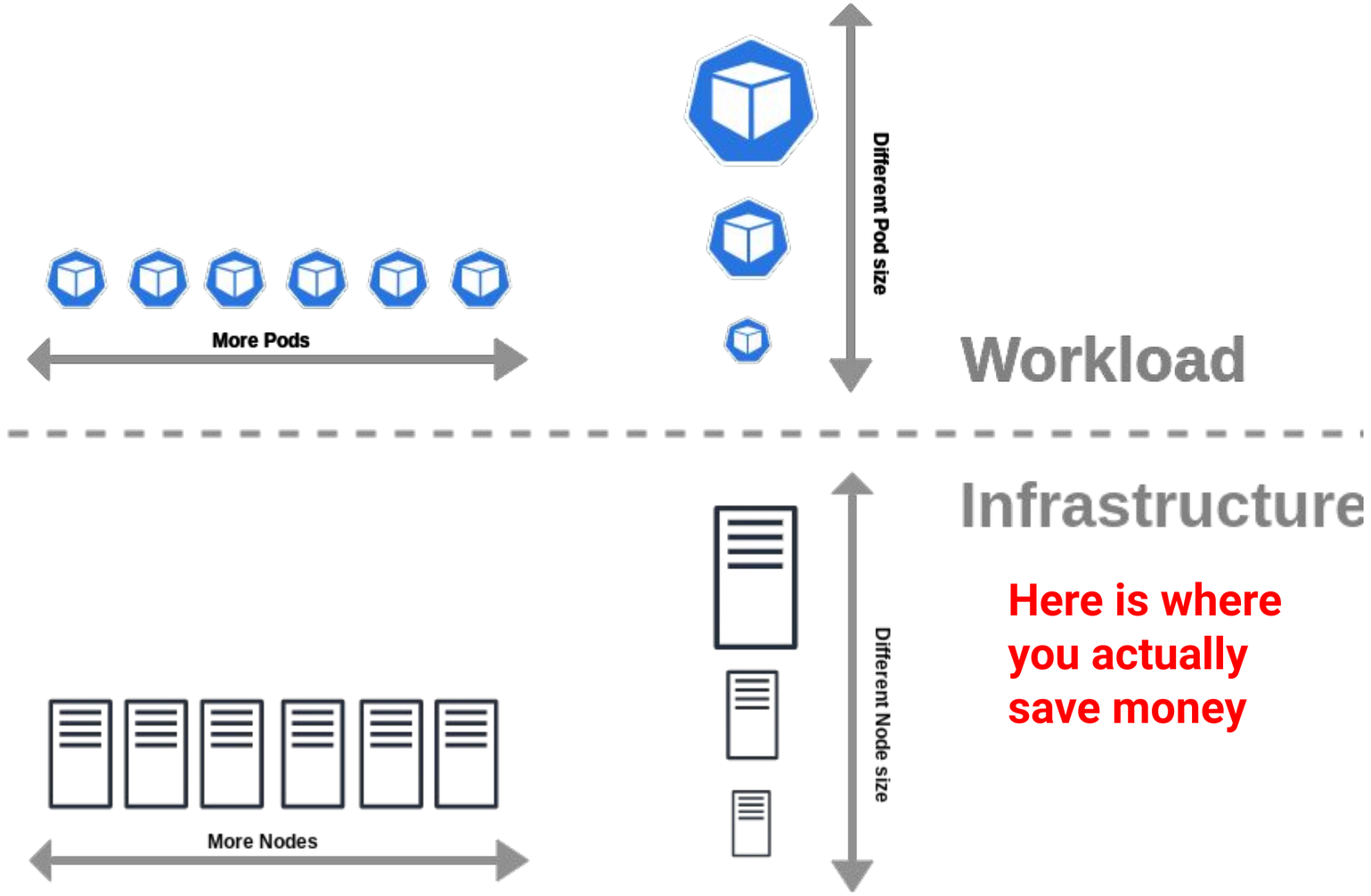
Advanced operations: A/B testing, rolling updates, canary testing

Exam Tips:

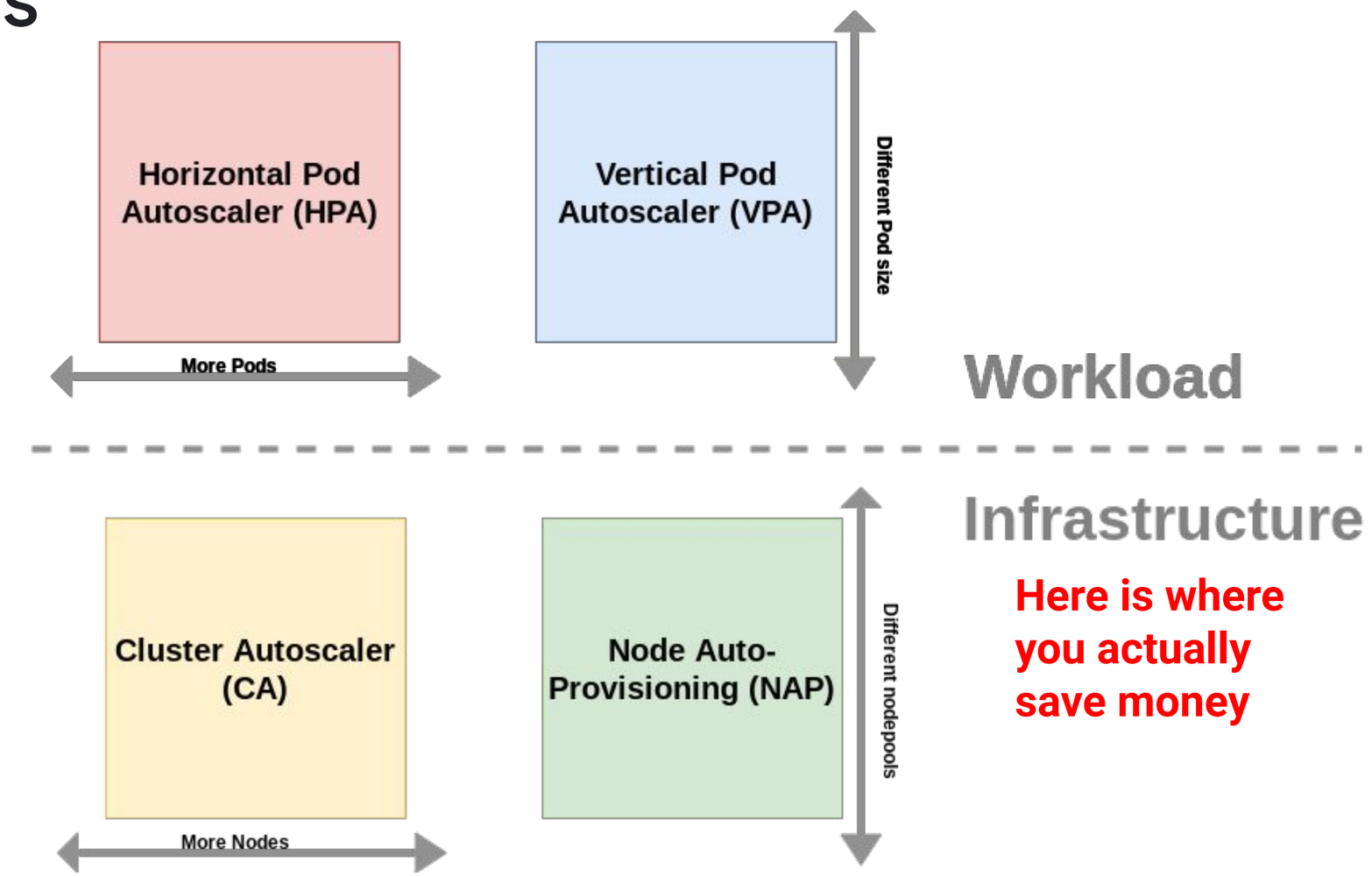
- *You should know what deployment options GKE offers and how each of them works on a high level. Here is a great resource to understand those concepts.*
- *Differentiate between deployment strategies and testing strategies.*
- *Be able to choose the right strategy under different circumstances, eg. minimal downtime, rollback duration etc.*
- *Deploying new version is important... but being able to quickly and reliably roll back to previous version is even more important!*
- *To start a rolling update of a new app in GKE:*
  - *kubectl set image deployment/hello-app hello-app=REGION-docker.pkg.dev/${PROJECT_ID}/hello-repo/hello-app:v2*



GKE: The 4 scalability dimensions



The only provider which supports the 4 scalability dimensions



Pod Placement

Requests & Limits

Requests specify how much resource (i.e. CPU and memory) a Container needs
Limits specify the amount of resources the container is allowed to use

Node Selector

Node selector is an approach to schedule Pods to a specific set of nodes (or GKE node pools) using matching labels

Affinity & Anti-Affinity

Affinity/anti-affinity is a scheduling feature to place Pods to Nodes using expressive rules against Pod and Node labels.

Taints & Tolerations

Taints are used to repel Pods from specific Nodes. **Tolerations** allow Pods to tolerate the taints

Pod Placement

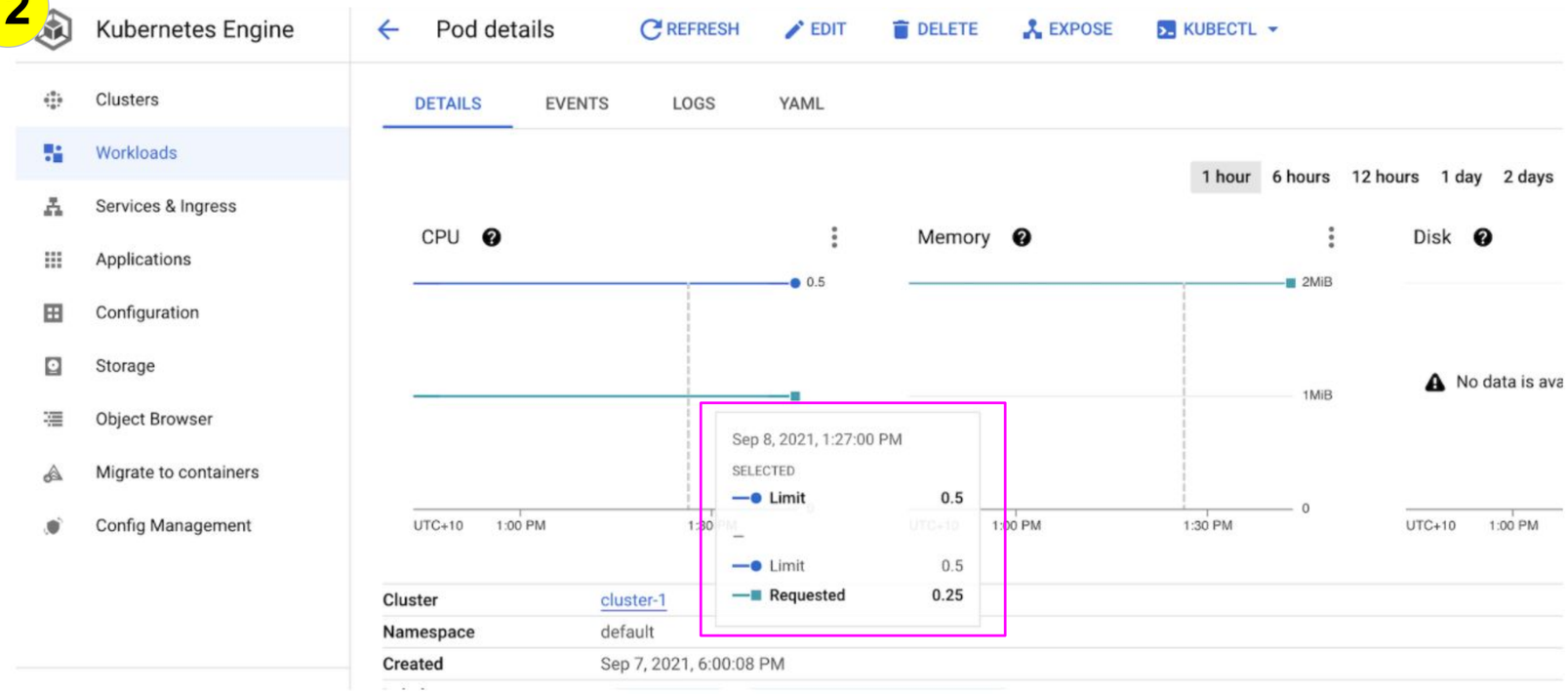
Requests & Limits

Requests specify how much resource (i.e. CPU and memory) a Container needs
Limits specify the amount of resources the container is allowed to use

1

```
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: app
    image: images.my-company.example/app:v4
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
  - name: log-aggregator
    image: images.my-company.example/log-aggregator:v6
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

2



Memory cgroup out of memory: Killed process - when Container hits a memory limit

Pod Placement

Node Selector

Node selector is an approach to schedule Pods to a specific set of nodes (or GKE node pools) using matching labels

1

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx:latest
    imagePullPolicy: IfNotPresent
  nodeSelector:
    cloud.google.com/gke-nodepool: app-pool
```

2

```
apiVersion: v1
kind: Pod
metadata:
  name: podthree
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx:latest
    imagePullPolicy: IfNotPresent
  nodeSelector:
    disktype: ssd
```

Pod Placement

Affinity & Anti-Affinity

Affinity/anti-affinity is a scheduling feature to place Pods to Nodes using expressive rules against Pod and Node labels.

```
pima@cloudshell:~/yaml-sample (first-medium-328400)$ cat redis.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
spec:
  selector:
    matchLabels:
      app: redis
  replicas: 3
  template:
    metadata:
      labels:
        app: redis
    spec:
      affinity:
        podAntiAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - labelSelector:
              matchExpressions:
              - key: app
                operator: In
                values:
                - redis
            topologyKey: "kubernetes.io/hostname"
      containers:
      - name: redis-server
        image: redis:latest
pima@cloudshell:~/yaml-sample (first-medium-328400)$
```

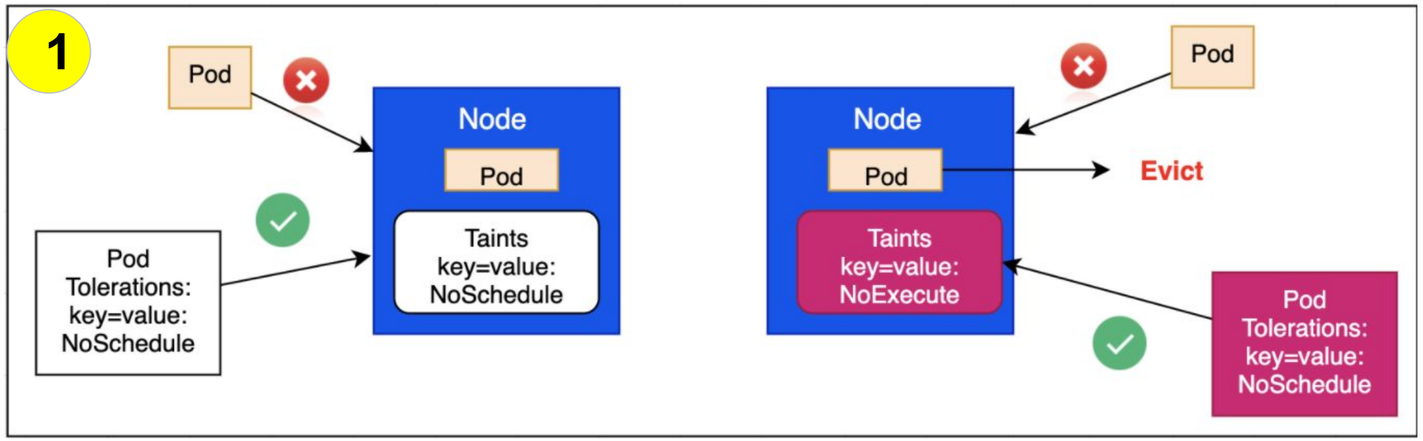
```
Error: Cannot schedule pods: node(s) didn't
match pod anti-affinity rules
```

Pod Placement

Taints & Tolerations

Taints are used to repel Pods from specific Nodes. **Tolerations** allow Pods to tolerate the taints

1



2

$ kubectl taint nodes NODE_NAME key=value:effect
$ kubectl taint nodes gke-123 service=web:NoExecute

4

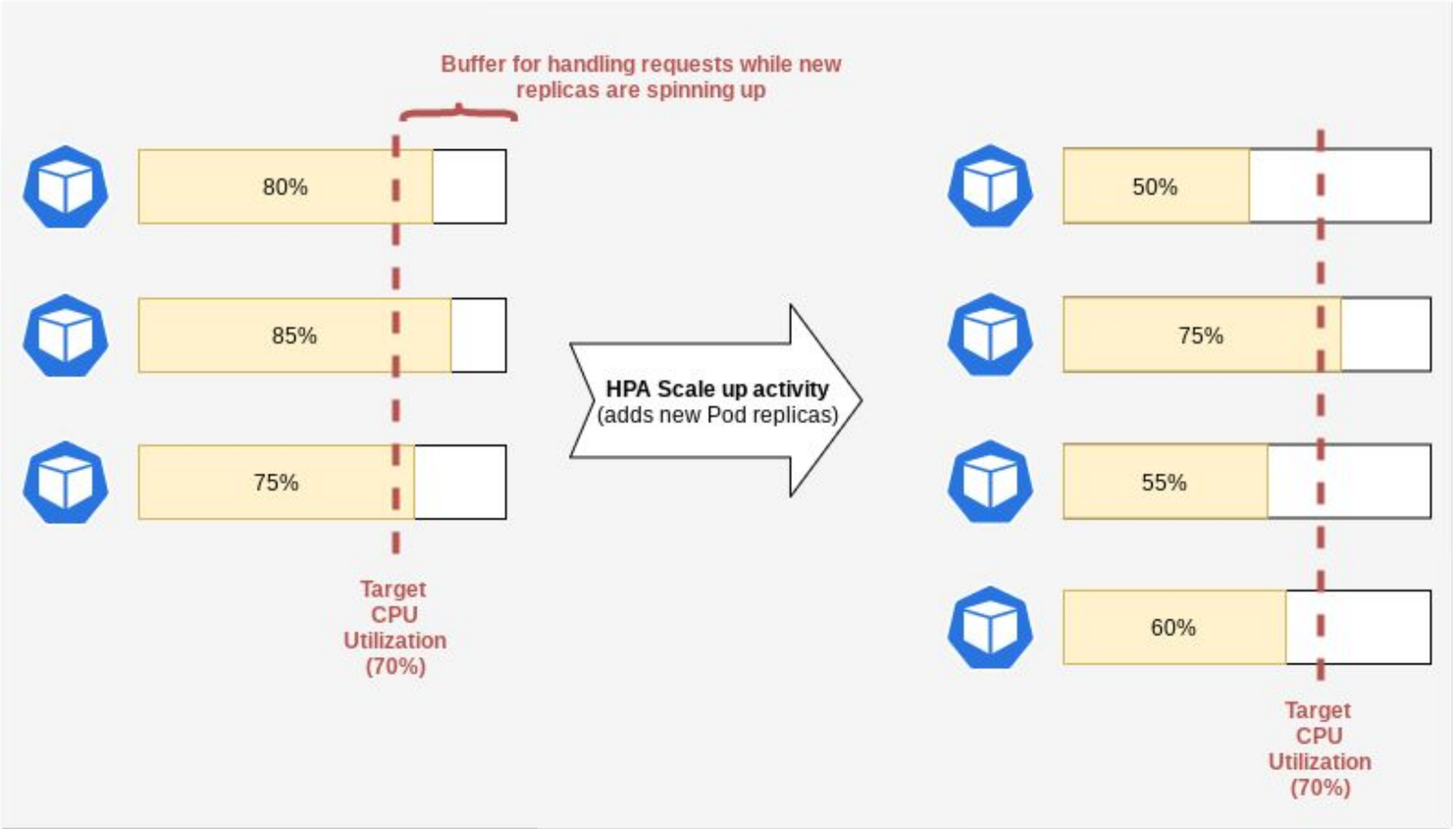
```
FailedScheduling: 0/3 nodes are available: 3 node(s) had
taint {service-web: true}, that the pod didn't tolerate
```

3

```
pima@cloudshell:~/yaml-sample (first-medium-328400)$ cat web.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: web
  replicas: 3
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
      - name: web-app
        image: nginx:latest
      tolerations:
      - key: "service-web"
        operator: "Equal"
        value: "true"
        effect: "NoSchedule"
```

Horizontal Pod Autoscaler (HPA)

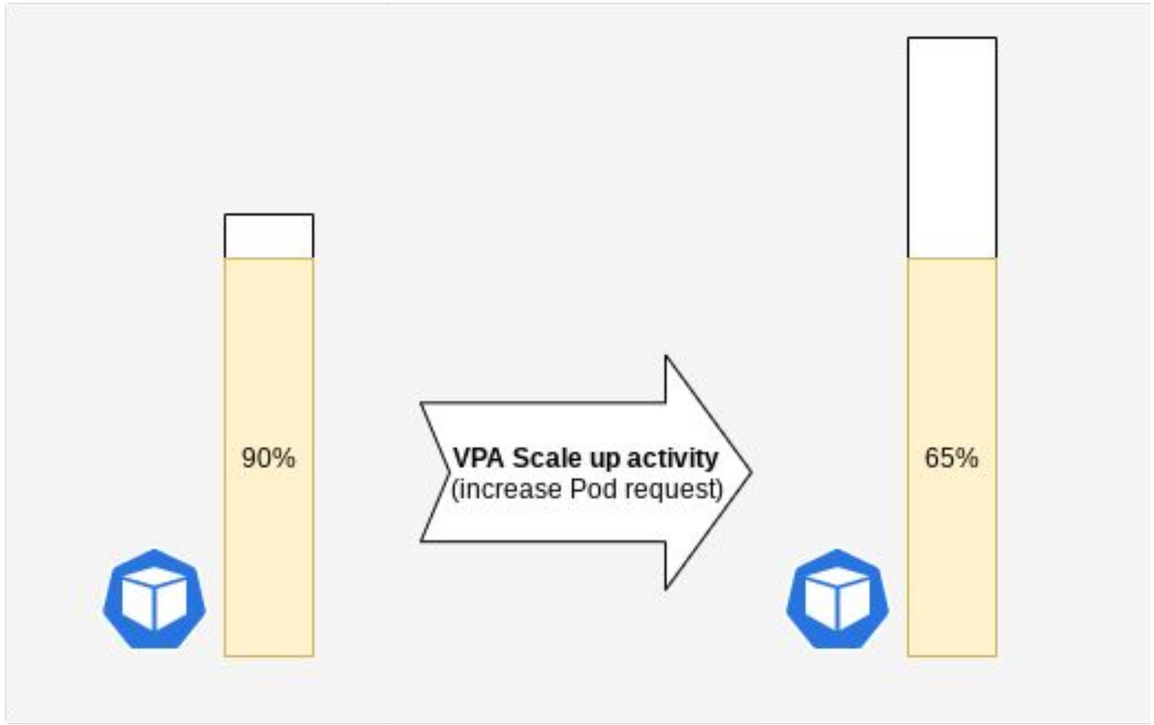
- Target Utilization: CPU or other custom metrics (eg. requests per second)
- Indicated for: stateless workers that can spin up reasonably fast
- Buffer size:
  - Small buffer prevents early scale ups, but it can overload your application during spikes.
  - Big buffer causes resource waste, increasing the cost of your bill.
  - Need to be enough for handling requests during two or three minutes in a spike.



For more information, see Configuring a Horizontal Pod Autoscaler.

Vertical Pod Autoscaler (VPA)

- Indicated for: stateless and stateful workloads not handled by HPA or when you don't know the proper Pod's resource requests
- Don't use VPA either Initial or Auto mode if you need to handle sudden spikes in traffic. Use HPA instead.
- Modes:
  - **Off**: recommendation
  - **Initial**: do not restart
  - **Auto**: restart
- Be careful when enabling the **Auto** Mode



For more information, see Configuring Vertical Pod Autoscaling.

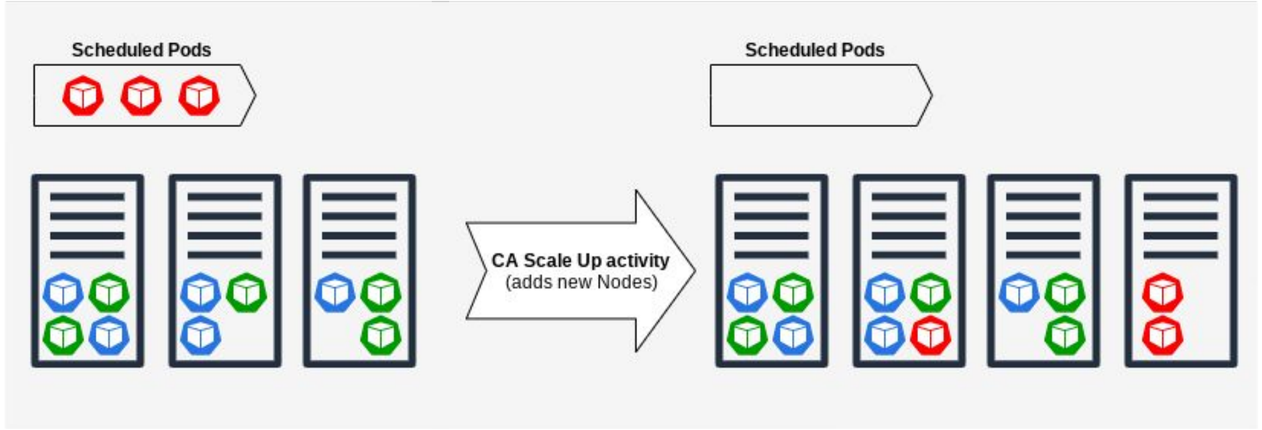
Mixing HPA and VPA

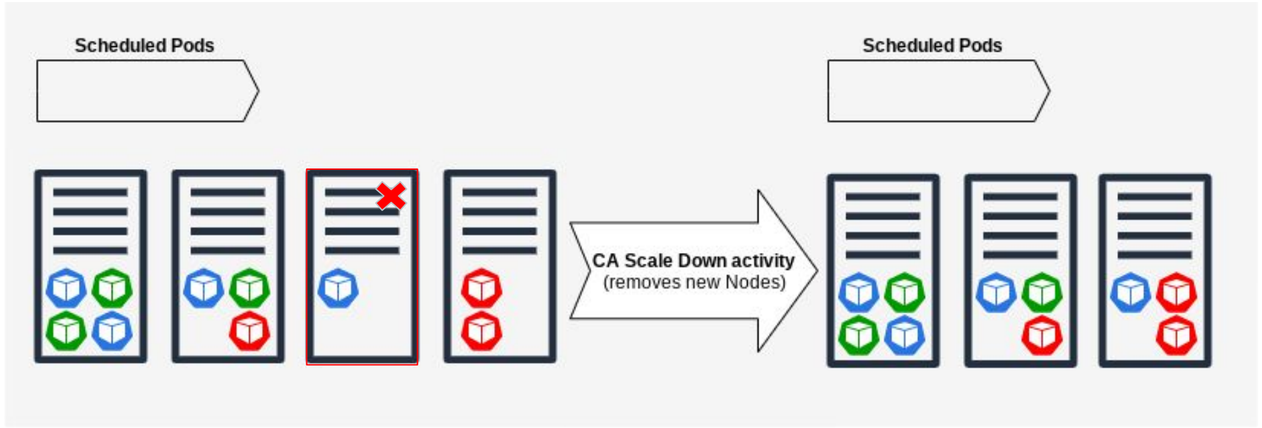
- **Never mix HPA and VPA on either CPU or Memory**

- Two options are safe to implement:
  - HPA using custom metrics
  - VPA in Recommendation Mode
- In either case, make sure your application is always running over HPA min-replicas

Cluster Autoscaler (CA)

- Indicated for: whenever you are using either HPA or VPA
- Optimized for the cost of infrastructure
- It is based on scheduling simulation and declared Pod requests
- Certain Pods cannot be restarted by any autoscaler. Blocking scale down. Kube-dns is the most common one. StatefulSets usually should not be restarted as well.
- If your workloads are resilient to nodes restarting inadvertently and to capacity losses, you can further improve cost savings by creating a cluster or node pool with preemptible VMs
- Learn how to analyse Cluster Autoscaler events in the logs.

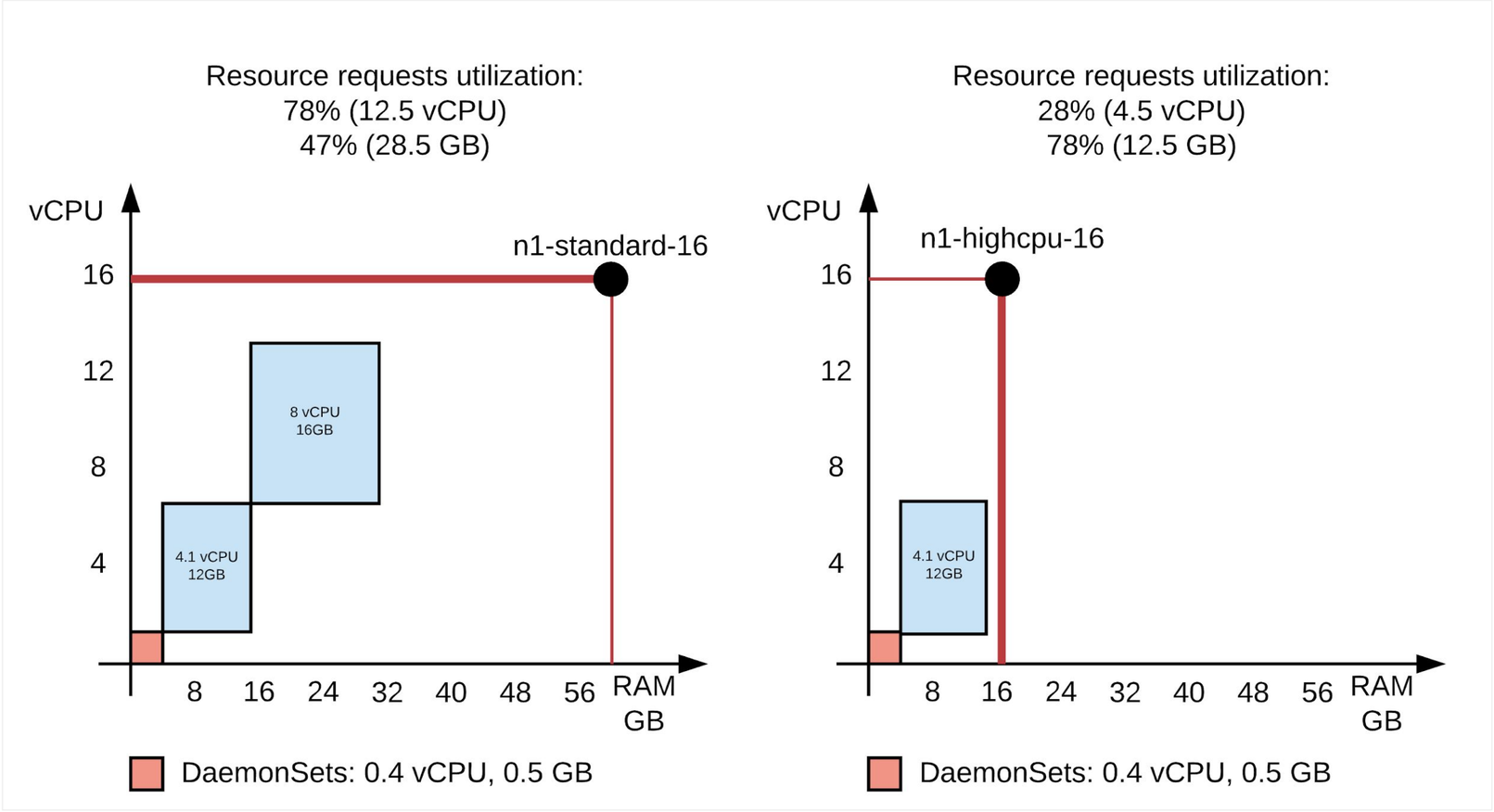




For more information, see Autoscaling a cluster.

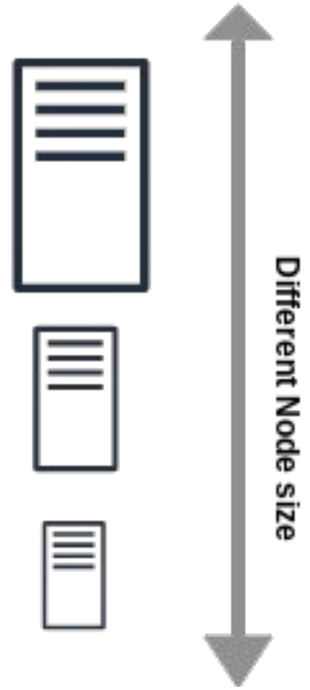
Binpacking

- Make sure your workload fit well inside the machine size
- You can create multiple node pools and use either nodeSelector or Node Affinity to select which node your pod must run.
- Another simpler option is to configure Node auto-provisioning



Node auto-provisioning (NAP)

- A mechanism of Cluster Autoscaler that new adds and deletes node pools automatically.
- Tends to reduce resources waste by dynamically creating node pools which best fits your workload
- If your workloads are resilient **to restarting inadvertently**, you can further improve cost savings configuring a preemptible VMs toleration in your Pod.
- It is a good practice to run Node auto-provisioning along with Vertical Pod Autoscaler.
- The autoscale latency can be slightly higher when new node pools need to be created

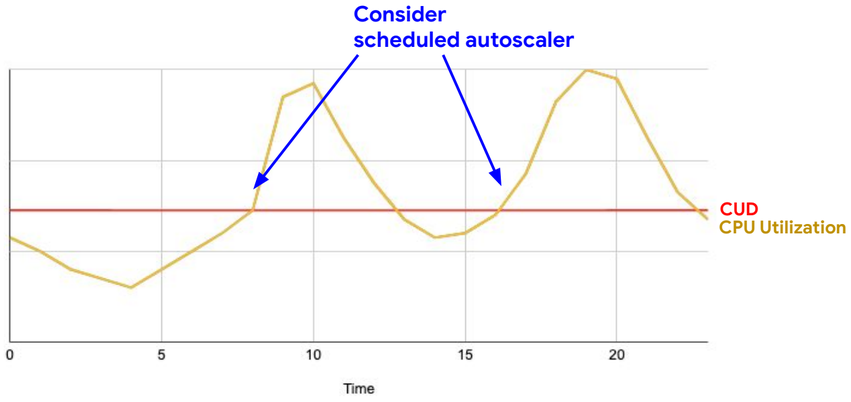


For more information, see Using Node auto-provisioning and Unsupported features.

Scheduled Autoscaler

Use when:

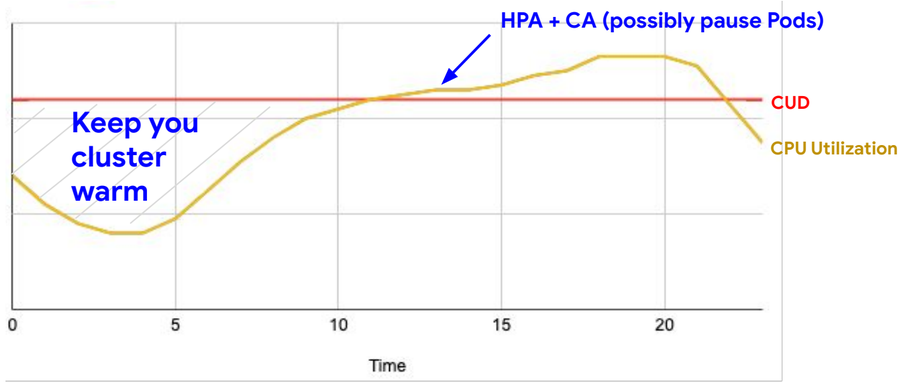
- You have **strong/quick spikes and/or deep valleys.**
- **Predictable traffic ebb and flow**.
  - Scaling down to the minimum during off-peak times.
  - Increase capacity previously to any known events
  - If your pods take too long to start



- You can combine this with Pause Pods if you are not sure which workload will spike

Avoid using when:

- Current spikes **can be handle by HPA and/or Pause Pod** without big over provisioning.
- If you purchased **committed use discount for your entire fleet.**



- For batch workloads.

For more information, see Reducing costs by scaling down GKE clusters during off-peak hours.

GKE: liveness & readiness probes

Exposing health status of your app

- **Liveness Probes**
  - Your app should expose a /health HTTP endpoint.
  - The app should send a "200 OK" response if it is considered healthy
  - Healthy means that the container doesn't need to be killed or restarted
  - Healthy means the app is ready and Its main dependencies are met

- **Readiness Probe**
  - Your app should either expose a /ready HTTP endpoint or use /health
  - The app should send a "200 OK" response if it is ready to receive traffic.
  - Ready means the app is healthy, init has completed, and a valid request does not return an error

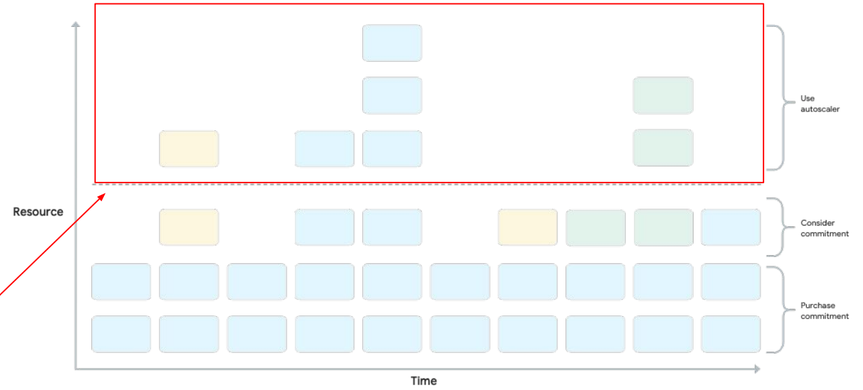
Exam Tip: *Make sure to differentiate between:*
- *Liveness and Readiness Probes (both Kubernetes-related topics)*
- *Health checks (for Load Balancers)*
- *Uptime checks*

Sign up for committed-use discounts

- It is highly recommended you purchase committed-use discounts
- 1 or 3 year contract
- The discount is up to
  - 57% for most resources like machine types or GPUs.
  - 70% for memory-optimized machine types

For more information, see Committed use discounts

Spot / Preemptible VMs



- **Warning**: use them on GKE clusters with caution.
- Indicated for: fault-tolerant jobs, and development environments
  - Can also be used for spikes through autoscale when CUD is relatively high (caution)
- VM instances last a maximum of 24 hours and provide **no** availability guarantees
- Are priced considerably lower than standard Compute Engine VMs
- Constraints:
  - Pod Disruption Budget may not be respected
  - Node preemption ignores the Pod grace period
  - It may take several minutes for GKE to detect that the node was preempted
- Node Termination Event Handler (not an official Google project) provides an adapter for translating Compute Engine node termination events to graceful pod terminations in Kubernetes
  - Will become deprecated with k8s Graceful Node Shutdown 1.20+
- It is indicated that you set a backup node pool without PVMs due to stockouts.

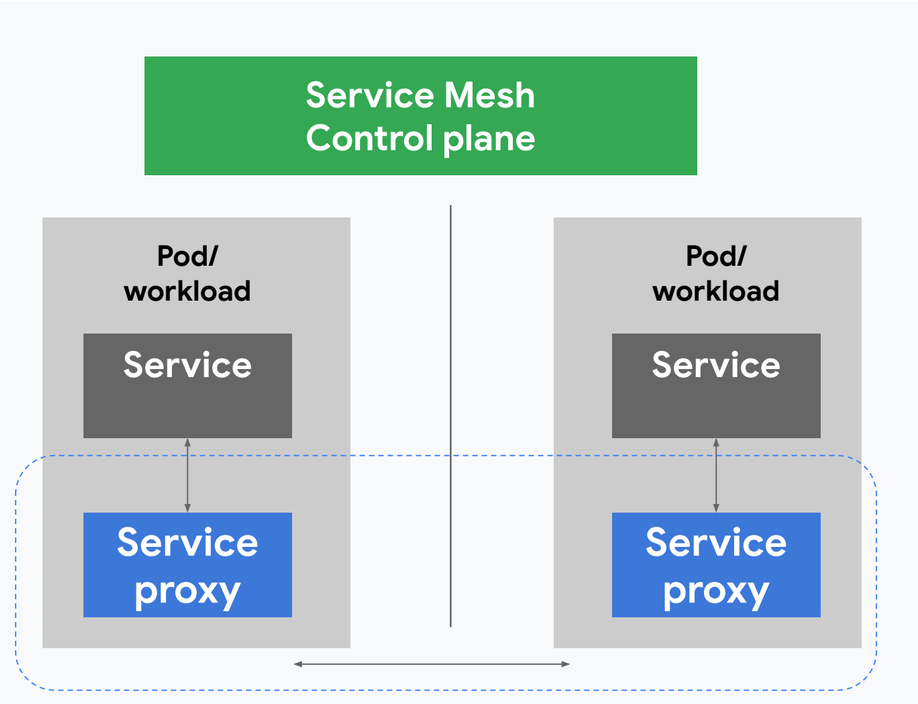
For more information, see Running preemptible VMs on GKE and follow Running web applications on GKE using cost-optimized PVMs tutorial for handling preemptions in a reliable way

Service Mesh (Istio / ASM)

Used for visibility, traffic control, security, policy enforcement etc

Outbound features:

- Service authentication
- Load balancing
- Timeouts, retries and circuit breakers
- Connection pool sizing
- Fine-grained routing
- Telemetry
- Request Tracing
- Fault Injection



Inbound features:

- Service authentication
- Authorization
- Rate limits
- Load shedding
- Telemetry
- Request Tracing
- Fault Injection

Exam Tip: *Service Mesh (Istio / Anthos Service Mesh) is often the right choice when advanced traffic management is required, eg. mutual TLS, Fault Injection, Traffic Splitting, Circuit Breaking, Connection Pooling etc.*

GOOGLE Kubernetes Engine

#GCPSketchnote

@PVERGADIA THECLOUDGIRL.DEV

9.07.2020

What is GKE?

Your app crashed in production!

SYS ADMIN SAM

Let's fix it once and for all with CONTAINER!

But it worked on my machine?!

DEVELOPER ERIN

CONTAINER FEATURES

- Portable
- Shareable
- Versions
- Isolation
- Introspection
- Fast Deployments
- Immutable
- Reusable

CONTAINER	VIRTUAL MACHINE	BARE-METAL SERVER
Application Code	Application Code	Application Code
Dependencies	Dependencies	Dependencies
Operating System	Operating System	Operating System
Hardware	Hardware	Hardware

But I have got lots of containers, how can I orchestrate all of them?

SAM

I have heard Kubernetes could be the answer...

It is open source platform for managing containers.

ERIN

SAM

Kubernetes is not that easy from Installation to Provisioning to Upgrades to Scaling & SLAs

How does GKE work?

SAM

User

kubectl

http://gRPC/GCP UI

Google Kubernetes Engine Container Cluster in GCP

REGIONAL CONTROL PLANE
- API Server
- Resource Controllers
- Storage
- Scheduler
- Replica in Zone B
- Replica in Zone C

NODES
- User Pod: Containers
- User Pod: Containers
- User Pod: Containers

CONNECTED GCP SERVICES
- VPC Networking
- Persistent Disk
- Load Balancer
- Cloud Monitoring

Color Legend:
- Google Kubernetes Engine provisions, maintains & operates
- Google Kubernetes Engine provisions User optionally maintains & operates

MANAGED KUBERNETES

KUBERNETES → GKE

I bet GKE is the answer with advanced cluster management!

- Easy Cluster Creation
- Load Balancing
- Auto Scaling
- Auto Upgrades
- Auto Repair
- Logging Monitoring

How do I use GKE ?

Cloud Code → Code Repository → Cloud Build → Container Registry → GKE ← Kubectl

WORKLOAD		INFRASTRUCTURE	
Horizontal Pod Autoscaler (HPA)	Vertical Pod Autoscaler (VPA)	Cluster Autoscaler (CA)	Node Auto-Provisioning (NAP)
More Pods	Different Pod Sizes	More Nodes	Different Node Pools

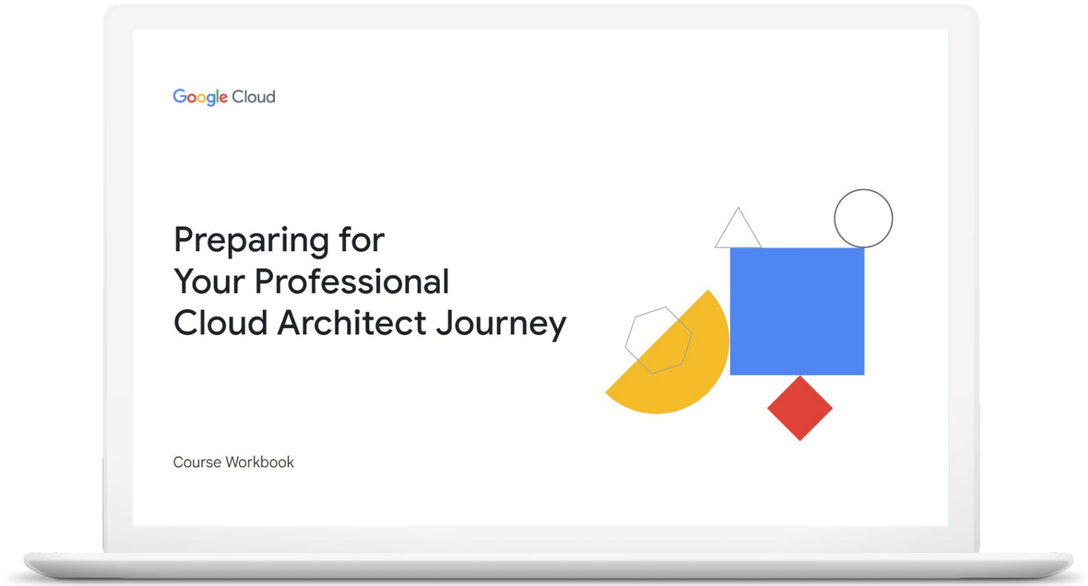
How does it handle traffic growth?

How do I secure apps using GKE ?

SECURE APPS	TRUSTED NETWORKING	SOFTWARE SUPPLY CHAIN SECURITY
Data Encryption	Global VPC	Binary Authorization
Certified Images	Global Load Balancing	Vulnerability Scanning
Private Clusters	Cloud Armor	Managed Base Image
Idenity & Access - IAM - RBAC	Network Policy	

Your study plan:

Managing implementation and ensuring solution and operations reliability



5.1 Advising development/operation team(s) to ensure a successful deployment of the solution

5.2 Interacting with Google Cloud programmatically

6.1-6.4 Monitoring/logging/profiling/alerting solution
Deployment and release management
Assisting with the support of deployed solutions
Evaluating quality control measures

5.1 Advising development/operation teams to ensure successful deployment of the solution

- Application development
- API best practices
- Testing frameworks (load/unit/integration)
- Data and system migration and management tooling

5.1 Diagnostic Question 01 Discussion

Cymbal Direct is working on a social media integration service in Google Cloud. Mahesh is a non-technical manager who wants to ensure that the **project doesn't exceed the budget** and **responds quickly to unexpected cost increases.** You need to set up access and billing for the project.

What should you do?

A. Assign the predefined **Billing Account Administrator role to Mahesh**. Create a project budget. Configure billing alerts to be sent to the **Billing Administrator.** Use resource **quotas to cap how many resources can be deployed.**

B. Assign the predefined **Billing Account Administrator role to Mahesh.** Create a project budget. Configure billing alerts to be sent to the **Project Owner.** Use resource **quotas to cap how much money can be spent.**

C. Use the predefined **Billing Account Administrator role for the Billing Administrator group,** and assign Mahesh to the group. Create a project budget. Configure billing alerts to be sent to the **Billing Account Administrator**. Use **resource quotas to cap how many resources can be deployed.**

D. Use the predefined **Billing Account Administrator role for the Billing Administrator group,** and assign Mahesh to the group. Create a project budget. Configure billing alerts to be sent to the **Billing Account Administrator.** Use **resource quotas to cap how much money can be spent.**

5.1 Diagnostic Question 01 Discussion

Cymbal Direct is working on a social media integration service in Google Cloud. Mahesh is a non-technical manager who wants to ensure that the **project doesn't exceed the budget** and **responds quickly to unexpected cost increases.** You need to set up access and billing for the project.

What should you do?

A. Assign the predefined **Billing Account Administrator role to Mahesh**. Create a project budget. Configure billing alerts to be sent to the **Billing Administrator.** Use resource **quotas to cap how many resources can be deployed.**

B. Assign the predefined **Billing Account Administrator role to Mahesh.** Create a project budget. Configure billing alerts to be sent to the **Project Owner.** Use resource **quotas to cap how much money can be spent.**

C. Use the predefined **Billing Account Administrator role for the Billing Administrator group,** and assign Mahesh to the group. Create a project budget. Configure billing alerts to be sent to the **Billing Account Administrator**. Use **resource quotas to cap how many resources can be deployed.**

D. Use the predefined **Billing Account Administrator role for the Billing Administrator group,** and assign Mahesh to the group. Create a project budget. Configure billing alerts to be sent to the **Billing Account Administrator.** Use **resource quotas to cap how much money can be spent.**

5.1 Diagnostic Question 02 Discussion

Your organization is planning a disaster recovery (DR) strategy. Your stakeholders require a **recovery time objective (RTO) of 0** and a **recovery point objective (RPO) of 0** for **zone outage.** They require an **RTO of 4 hours** and an **RPO of 1 hour** for a **regional outage.** Your application consists of a **web application and a backend MySQL database.** You need the most efficient solution to meet your recovery KPIs.

What should you do?

A. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to use both backends.** Use Cloud SQL with high availability (HA) enabled in us-east and a cross-region replica in us-west.

B. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to the us-east backend.** Use Cloud SQL with high availability (HA) enabled in us-east and a cross-region replica in us-west. **Manually promote the us-west Cloud SQL instance** and **change the load balancer backend to us-west.**

C. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to use both backends.** Use Cloud SQL with high availability (HA) enabled in us-east and back up the database every hour to a multi-region Cloud Storage bucket. **Restore the data to a Cloud SQL database in us-west** if there is a failure.

D. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to use both backends.** Use Cloud SQL with high availability (HA) enabled in us-east and **back up the database every hour** to a multi-region Cloud Storage bucket. **Restore the data to a Cloud SQL database in us-west** if there is a failure and **change the load balancer backend to us-west.**

5.1 Diagnostic Question 02 Discussion

Your organization is planning a disaster recovery (DR) strategy. Your stakeholders require a **recovery time objective (RTO) of 0** and a **recovery point objective (RPO) of 0** for **zone outage.** They require an **RTO of 4 hours** and an **RPO of 1 hour** for a **regional outage.** Your application consists of a **web application and a backend MySQL database.** You need the most efficient solution to meet your recovery KPIs.

What should you do?

A. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to use both backends.** Use Cloud SQL with high availability (HA) enabled in us-east and a cross-region replica in us-west.

B. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to the us-east backend.** Use Cloud SQL with high availability (HA) enabled in us-east and a cross-region replica in us-west. **Manually promote the us-west Cloud SQL instance** and **change the load balancer backend to us-west.**

C. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to use both backends.** Use Cloud SQL with high availability (HA) enabled in us-east and back up the database every hour to a multi-region Cloud Storage bucket. **Restore the data to a Cloud SQL database in us-west** if there is a failure.

D. Use a global HTTP(S) load balancer. Deploy the web application as Compute Engine managed instance groups (MIG) in two regions, us-west and us-east. **Configure the load balancer to use both backends.** Use Cloud SQL with high availability (HA) enabled in us-east and **back up the database every hour** to a multi-region Cloud Storage bucket. **Restore the data to a Cloud SQL database in us-west** if there is a failure and **change the load balancer backend to us-west.**

5.1 Advising development/operation team(s) to ensure successful deployment of the solution

Resources to start your journey

- Cloud Reference Architectures and Diagrams | Cloud Architecture Center
- What is DevOps? Research and Solutions | Google Cloud
- Develop and deliver apps with Cloud Code, Cloud Build, Google Cloud Deploy, and GKE | Cloud Architecture Center
- Google Cloud API design tips
- DevOps tech: Continuous testing | Google Cloud
- DevOps tech: Test data management | Google Cloud
- Testing Overview | Cloud Functions Documentation
- Database Migration Service | Google Cloud
- Cloud Migration Products & Services

5.2 Interacting with Google Cloud programmatically

- Google Cloud Shell
- Google Cloud SDK (gcloud, gsutil and bq)
- Cloud Emulators (e.g. Cloud Bigtable, Datastore, Spanner, Pub/Sub, Firestore)

5.2 Diagnostic Question 03 Discussion

Your environment has multiple projects used for development and testing. Each project has a budget, and each developer has a budget. A personal budget overrun can cause a project budget overrun. Several developers are creating resources for testing as part of their CI/CD pipeline but are not deleting these resources after their tests are complete. If the compute resource fails during testing, the test can be run again. You want to **reduce costs** and **notify the developer when a personal budget overrun causes a project budget overrun.**

What should you do?

A. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Create a group for the developers in each project**, and add them to the appropriate group. Create a notification channel for each group. Configure a billing alert to notify the group when their budget is exceeded. Modify the build scripts/pipeline to label all resources with the label “creator” set to the developer’s email address. Use spot (preemptible) instances wherever possible.

B. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Configure a billing alert to notify billing admins and users when their budget is exceeded.** Modify the build scripts/pipeline to label all resources with the label “creator” set to the developer’s email address. Use spot (preemptible) instances wherever possible.

C. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Create a Pub/Sub topic for developer-budget-notifications**. **Create a Cloud Function to notify the developer based on the labels**. Modify the build scripts/pipeline to label all resources with the label “creator” set to the developer’s email address. Use spot (preemptible) instances wherever possible.

D. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Create a Pub/Sub topic for developer-budget-notifications. Create a Cloud Function to notify the developer based on the labels.** Modify the build scripts/pipeline to label all resources with the label “creator” set to the developer’s email address. Use spot (preemptible) instances wherever possible. **Use Cloud Scheduler to delete resources older than 24 hours in each project.**

5.2 Diagnostic Question 03 Discussion

Your environment has multiple projects used for development and testing. Each project has a budget, and each developer has a budget. A personal budget overrun can cause a project budget overrun. Several developers are creating resources for testing as part of their CI/CD pipeline but are not deleting these resources after their tests are complete. If the compute resource fails during testing, the test can be run again. You want to **reduce costs** and **notify the developer when a personal budget overrun causes a project budget overrun.**

What should you do?

A. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Create a group for the developers in each project**, and add them to the appropriate group. Create a notification channel for each group. Configure a billing alert to notify the group when their budget is exceeded. Modify the build scripts/pipeline to label all resources with the label "creator" set to the developer's email address. Use spot (preemptible) instances wherever possible.

B. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Configure a billing alert to notify billing admins and users when their budget is exceeded.** Modify the build scripts/pipeline to label all resources with the label "creator" set to the developer's email address. Use spot (preemptible) instances wherever possible.

C. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Create a Pub/Sub topic for developer-budget-notifications**. **Create a Cloud Function to notify the developer based on the labels**. Modify the build scripts/pipeline to label all resources with the label "creator" set to the developer's email address. Use spot (preemptible) instances wherever possible.

D. Configure billing export to BigQuery. Create a Google Cloud budget for each project. **Create a Pub/Sub topic for developer-budget-notifications. Create a Cloud Function to notify the developer based on the labels.** Modify the build scripts/pipeline to label all resources with the label "creator" set to the developer's email address. Use spot (preemptible) instances wherever possible. **Use Cloud Scheduler to delete resources older than 24 hours in each project.**

5.2 Interacting with Google Cloud programmatically

Resources to start your journey

gcloud CLI overview | Google Cloud CLI Documentation

How Cloud Shell works

Google Cloud APIs

Testing apps locally with the emulator | Cloud Pub/Sub Documentation

Connect your app and start prototyping | Firebase Documentation

Use the emulator | Cloud Bigtable Documentation

Using the Cloud Spanner Emulator

6 Ensuring solution and operations reliability

6.1 Monitoring/logging/profiling/alerting solution

6.2 Deployment and release management

6.3 Assisting with the support of deployed solutions

6.4 Evaluating quality control measures

6.1 Diagnostic Question 04 Discussion

Your client has adopted a multi-cloud strategy that uses a virtual machine-based infrastructure. The client's website serves users across the globe. The client needs a **single dashboard view to monitor performance in their AWS and Google Cloud environments**. Your client previously experienced an extended outage and wants to establish a **monthly service level objective (SLO) of no outage longer than an hour.**

What should you do?

A. In Cloud Monitoring, create an uptime check for the URL your clients will access. Configure it to check from multiple regions. Use the Cloud Monitoring dashboard to view the uptime metrics over time and ensure that the SLO is met. Recommend an SLO of **97% uptime per month.**

B. In Cloud Monitoring, create an uptime check for the URL your clients will access. Configure it to check from multiple regions. Use the Cloud Monitoring dashboard to view the uptime metrics over time and ensure that the SLO is met. Recommend an SLO of **97% uptime per day.**

C. Authorize access to your Google Cloud project from AWS with a service account. Install the monitoring agent on AWS EC2 (virtual machines) and Compute Engine instances. Use Cloud Monitoring to create dashboards that use the **performance metrics from virtual machines** to ensure that the SLO is met.

D. Create a new project to use as an AWS connector project. Authorize access to the project from AWS with a service account. Install the monitoring agent on AWS EC2 (virtual machines) and Compute Engine instances. Use Cloud Monitoring to create dashboards that use the **performance metrics from virtual machines** to ensure that the SLO is met.

6.1 Diagnostic Question 04 Discussion

Your client has adopted a multi-cloud strategy that uses a virtual machine-based infrastructure. The client's website serves users across the globe. The client needs a **single dashboard view to monitor performance in their AWS and Google Cloud environments**. Your client previously experienced an extended outage and wants to establish a **monthly service level objective (SLO) of no outage longer than an hour.**

What should you do?

A. In Cloud Monitoring, create an uptime check for the URL your clients will access. Configure it to check from multiple regions. Use the Cloud Monitoring dashboard to view the uptime metrics over time and ensure that the SLO is met. Recommend an SLO of **97% uptime per month.**

B. In Cloud Monitoring, create an uptime check for the URL your clients will access. Configure it to check from multiple regions. Use the Cloud Monitoring dashboard to view the uptime metrics over time and ensure that the SLO is met. Recommend an SLO of **97% uptime per day.**

C. Authorize access to your Google Cloud project from AWS with a service account. Install the monitoring agent on AWS EC2 (virtual machines) and Compute Engine instances. Use Cloud Monitoring to create dashboards that use the **performance metrics from virtual machines** to ensure that the SLO is met.

D. Create a new project to use as an AWS connector project. Authorize access to the project from AWS with a service account. Install the monitoring agent on AWS EC2 (virtual machines) and Compute Engine instances. Use Cloud Monitoring to create dashboards that use the **performance metrics from virtual machines** to ensure that the SLO is met.

6.1 Diagnostic Question 05 Discussion

Cymbal Direct uses a proprietary service to manage on-call rotation and alerting. The on-call rotation service has an API for integration. Cymbal Direct wants to **monitor its environment for service availability** and **ensure that the correct person is notified**.

What should you do?

A. Ensure that VPC firewall rules allow access from the IP addresses used by Google Cloud's uptime-check servers. Create a Pub/Sub topic for alerting as a monitoring notification channel in Google Cloud's operations suite. Create an **uptime check for the appropriate resource's internal IP address,** with an alerting policy set to use the Pub/Sub topic. Create a Cloud Function that subscribes to the Pub/Sub topic to send the alert to the on-call API.

B. Ensure that VPC firewall rules allow access from the IP addresses used by Google Cloud's uptime-check servers. **Create a Pub/Sub topic** for alerting as a monitoring notification channel in Google Cloud's operations suite. Create an **uptime check for the appropriate resource's external IP address**, with an alerting policy set to use the Pub/Sub topic. Create a Cloud Function that subscribes to the Pub/Sub topic to send the alert to the on-call API.

C. Ensure that VPC **firewall rules allow access from the on-call API.** Create a Cloud Function to send the alert to the on-call API. Add Cloud Functions as a monitoring notification channel in Google Cloud's operations suite. Create an uptime check for the appropriate resource's external IP address, with an alerting policy set to use the Cloud Function.

D. Ensure that VPC firewall rules allow access from the IP addresses used by Google Cloud's uptime-check servers. Add the URL for the on-call rotation API as a monitoring notification channel in Google Cloud's operations suite. Create an **uptime check for the appropriate resource's internal IP address**, with an alerting policy set to use the API.

6.1 Diagnostic Question 05 Discussion

Cymbal Direct uses a proprietary service to manage on-call rotation and alerting. The on-call rotation service has an API for integration. Cymbal Direct wants to **monitor its environment for service availability** and **ensure that the correct person is notified**.

What should you do?

A. Ensure that VPC firewall rules allow access from the IP addresses used by Google Cloud's uptime-check servers. Create a Pub/Sub topic for alerting as a monitoring notification channel in Google Cloud's operations suite. Create an **uptime check for the appropriate resource's internal IP address,** with an alerting policy set to use the Pub/Sub topic. Create a Cloud Function that subscribes to the Pub/Sub topic to send the alert to the on-call API.

B. Ensure that VPC firewall rules allow access from the IP addresses used by Google Cloud's uptime-check servers. **Create a Pub/Sub topic** for alerting as a monitoring notification channel in Google Cloud's operations suite. Create an **uptime check for the appropriate resource's external IP address**, with an alerting policy set to use the Pub/Sub topic. Create a Cloud Function that subscribes to the Pub/Sub topic to send the alert to the on-call API.

C. Ensure that VPC **firewall rules allow access from the on-call API.** Create a Cloud Function to send the alert to the on-call API. Add Cloud Functions as a monitoring notification channel in Google Cloud's operations suite. Create an uptime check for the appropriate resource's external IP address, with an alerting policy set to use the Cloud Function.

D. Ensure that VPC firewall rules allow access from the IP addresses used by Google Cloud's uptime-check servers. Add the URL for the on-call rotation API as a monitoring notification channel in Google Cloud's operations suite. Create an **uptime check for the appropriate resource's internal IP address**, with an alerting policy set to use the API.

6.2 Diagnostic Question 06 Discussion

Cymbal Direct releases new versions of its drone delivery software every 1.5 to 2 months. Although most releases are successful, you have experienced three **problematic releases that made drone delivery unavailable** while software developers rolled back the release. You want to **increase the reliability of software releases** and prevent similar problems in the future.

What should you do?

A. Adopt a **"waterfall"** development process. Maintain the current release schedule. Ensure that documentation explains how all the features interact. Ensure that the entire application is tested in a staging environment before the release. Ensure that the process to roll back the release is documented. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility.

B. Adopt a **"waterfall"** development process. Maintain the current release schedule. Ensure that documentation explains how all the features interact. Automate testing of the application. Ensure that the process to roll back the release is well documented. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility.

C. Adopt an **"agile"** development process. **Maintain the current release schedule**. Automate build processes from a source repository. Automate testing after the build process. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility. Deploy the previous version if problems are detected and you need to roll back.

D. Adopt an **"agile"** development process. **Reduce the time between releases** as much as possible. Automate the build process from a source repository, which includes versioning and self-testing. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility. Use a canary deployment to detect issues that could cause rollback.

6.2 Diagnostic Question 06 Discussion

Cymbal Direct releases new versions of its drone delivery software every 1.5 to 2 months. Although most releases are successful, you have experienced three **problematic releases that made drone delivery unavailable** while software developers rolled back the release. You want to **increase the reliability of software releases** and prevent similar problems in the future.

What should you do?

A. Adopt a **"waterfall"** development process. Maintain the current release schedule. Ensure that documentation explains how all the features interact. Ensure that the entire application is tested in a staging environment before the release. Ensure that the process to roll back the release is documented. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility.

B. Adopt a **"waterfall"** development process. Maintain the current release schedule. Ensure that documentation explains how all the features interact. Automate testing of the application. Ensure that the process to roll back the release is well documented. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility.

C. Adopt an **"agile"** development process. **Maintain the current release schedule**. Automate build processes from a source repository. Automate testing after the build process. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility. Deploy the previous version if problems are detected and you need to roll back.

D. Adopt an **"agile"** development process. **Reduce the time between releases** as much as possible. Automate the build process from a source repository, which includes versioning and self-testing. Use Cloud Monitoring, Cloud Logging, and Cloud Alerting to ensure visibility. Use a canary deployment to detect issues that could cause rollback.

6.3 Diagnostic Question 07 Discussion

Cymbal Direct's warehouse and inventory system was written in Java. The system uses a **microservices architecture in GKE** and is instrumented with Zipkin. Seemingly at random, **a request will be 5-10 times slower** than others. The development team tried to reproduce the problem in testing, but failed to determine the cause of the issue.

What should you do?

A. **Create metrics in Cloud Monitoring for your microservices** to test whether they are intermittently unavailable or slow to respond to HTTPS requests. Use **Cloud Profiler to determine which functions/methods** in your application's code use the **most system resources.** Use **Cloud Trace to identify slow requests** and determine which microservices/calls take the most time to respond.

B. **Create metrics in Cloud Monitoring for your microservices** to test whether they are intermittently unavailable or slow to respond to HTTPS requests. Use **Cloud Trace to determine which functions/methods** in your application's code use the **most system resources. Use Cloud Profiler to identify slow requests a**nd determine which microservices/calls take the most time to respond.

C. **Use Error Reporting** to test whether your microservices are intermittently unavailable or slow to respond to HTTPS requests. Use Cloud Profiler to determine which functions/methods in your application's code use the most system resources. Use Cloud Trace to identify slow requests and determine which microservices/calls take the most time to respond.

D. **Use Error Reporting** to test whether your microservices are intermittently unavailable or slow to respond to HTTPS requests. Use Cloud Trace to determine which functions/methods in your application's code Use the most system resources. Use Cloud Profiler to identify slow requests and determine which microservices/calls take the most time to respond.

6.3 Diagnostic Question 07 Discussion

Cymbal Direct's warehouse and inventory system was written in Java. The system uses a **microservices architecture in GKE** and is instrumented with Zipkin. Seemingly at random, **a request will be 5-10 times slower** than others. The development team tried to reproduce the problem in testing, but failed to determine the cause of the issue.

What should you do?

A. **Create metrics in Cloud Monitoring for your microservices** to test whether they are intermittently unavailable or slow to respond to HTTPS requests. Use **Cloud Profiler to determine which functions/methods** in your application's code use the **most system resources.** Use **Cloud Trace to identify slow requests** and determine which microservices/calls take the most time to respond.

B. **Create metrics in Cloud Monitoring for your microservices** to test whether they are intermittently unavailable or slow to respond to HTTPS requests. Use **Cloud Trace to determine which functions/methods** in your application's code use the **most system resources. Use Cloud Profiler to identify slow requests a**nd determine which microservices/calls take the most time to respond.

C. **Use Error Reporting** to test whether your microservices are intermittently unavailable or slow to respond to HTTPS requests. Use Cloud Profiler to determine which functions/methods in your application's code use the most system resources. Use Cloud Trace to identify slow requests and determine which microservices/calls take the most time to respond.

D. **Use Error Reporting** to test whether your microservices are intermittently unavailable or slow to respond to HTTPS requests. Use Cloud Trace to determine which functions/methods in your application's code Use the most system resources. Use Cloud Profiler to identify slow requests and determine which microservices/calls take the most time to respond.

6.3 Diagnostic Question 08 Discussion

You are using Cloud Run to deploy a **Flask web application named app.py written in Python**. In your testing and staging environments, the application performed as expected. When the application was deployed to production, product search results displayed products that should have been filtered out based on the user's preferences. The developer believes **this performance issue would result from the 'user.productFilter' variable either not being set or not being evaluated correctly**. You want **visibility into what is happening, but also want to minimize user impact**, because this is not a critical bug.

What should you do?

A. Use ssh to connect to the **Compute Engine instance** where Cloud Run is running. Run the command **'python3 -m pdb app.py'** to debug the application.

B. Use ssh to connect to the **Compute Engine instance** where Cloud Run is running. Use the command **'pip install google-python-cloud-debugger' to install Cloud Debugger**. Use the 'gcloud debug' command to debug the application.

C. Modify the Dockerfile for the Cloud Run application. Change the RUN command to **'python3 -m pdb /app.py'.** Modify the script to import pdb. **Deploy to Cloud Run** as a canary build.

D. Modify the Dockerfile for the Cloud Run application. Add 'RUN '**pip install google-python-cloud-debugger**' to the Dockerfile. Modify the script to import googleclouddebugger. Use '**gcloud debug**' to debug the application.

6.3 Diagnostic Question 08 Discussion

You are using Cloud Run to deploy a **Flask web application named app.py written in Python**. In your testing and staging environments, the application performed as expected. When the application was deployed to production, product search results displayed products that should have been filtered out based on the user's preferences. The developer believes **this performance issue would result from the 'user.productFilter' variable either not being set or not being evaluated correctly**. You want **visibility into what is happening, but also want to minimize user impact**, because this is not a critical bug.

What should you do?

A. Use ssh to connect to the **Compute Engine instance** where Cloud Run is running. Run the command **'python3 -m pdb app.py'** to debug the application.

B. Use ssh to connect to the **Compute Engine instance** where Cloud Run is running. Use the command **'pip install google-python-cloud-debugger' to install Cloud Debugger**. Use the 'gcloud debug' command to debug the application.

C. Modify the Dockerfile for the Cloud Run application. Change the RUN command to **'python3 -m pdb /app.py'.** Modify the script to import pdb. **Deploy to Cloud Run** as a canary build.

D. Modify the Dockerfile for the Cloud Run application. Add 'RUN '**pip install google-python-cloud-debugger**' to the Dockerfile. Modify the script to import googleclouddebugger. Use '**gcloud debug**' to debug the application.

6.4 Diagnostic Question 09 Discussion

Cymbal Direct has a new social media integration service that pulls images of its products from social media sites and displays them in a gallery of customer images on your online store. You receive an alert from Cloud Monitoring at 3:34 AM on Saturday. The store is still online, but **the gallery does not appear.** The **CPU utilization is 30% higher than expected on the VMs** running the service, which causes the managed instance group (MIG) to scale to the maximum number of instances. You verify that the issue is real by checking the site and by checking the incidents timeline.

What should you do to resolve the issue?

A. Increase the maximum number of instances in the MIG and verify that this resolves the issue. Ensure that the ticket is annotated with your solution. Create a normal work ticket for the application developer with a link to the incident. **Mark the incident as closed.**

B. Check the incident documentation or labels to determine the on-call contact. **Appoint an incident commander, and open a chat channel, or conference call for emergency response**. Investigate and resolve the issue by increasing the maximum number of instances in the MIG, and verify that this resolves the issue. Mark the incident as closed.

C. Increase the maximum number of instances in the MIG and verify that this resolves the issue. Check the incident documentation or labels to determine the on-call contact. **Appoint an incident commander, and open a chat channel, or conference call for emergency response.** Investigate and resolve the root cause of the issue. Write a blameless post-mortem and identify steps to prevent the issue, to ensure a culture of continuous improvement.

D. Verify the high CPU is not user impacting**, increase the maximum** number of instances in the MIG and verify that this resolves the issue.

6.4 Diagnostic Question 09 Discussion

Cymbal Direct has a new social media integration service that pulls images of its products from social media sites and displays them in a gallery of customer images on your online store. You receive an alert from Cloud Monitoring at 3:34 AM on Saturday. The store is still online, but **the gallery does not appear.** The **CPU utilization is 30% higher than expected on the VMs** running the service, which causes the managed instance group (MIG) to scale to the maximum number of instances. You verify that the issue is real by checking the site and by checking the incidents timeline.

What should you do to resolve the issue?

A. Increase the maximum number of instances in the MIG and verify that this resolves the issue. Ensure that the ticket is annotated with your solution. Create a normal work ticket for the application developer with a link to the incident. **Mark the incident as closed.**

B. Check the incident documentation or labels to determine the on-call contact. **Appoint an incident commander, and open a chat channel, or conference call for emergency response**. Investigate and resolve the issue by increasing the maximum number of instances in the MIG, and verify that this resolves the issue. Mark the incident as closed.

C. Increase the maximum number of instances in the MIG and verify that this resolves the issue. Check the incident documentation or labels to determine the on-call contact. **Appoint an incident commander, and open a chat channel, or conference call for emergency response.** Investigate and resolve the root cause of the issue. Write a blameless post-mortem and identify steps to prevent the issue, to ensure a culture of continuous improvement.

D. Verify the high CPU is not user impacting**, increase the maximum** number of instances in the MIG and verify that this resolves the issue.

6.4 Diagnostic Question 10 Discussion

You need to adopt Site Reliability Engineering principles and increase visibility into your environment. You want to **minimize management overhead** and **reduce noise** generated by the information being collected. You also want to **streamline the process of reacting to analyzing and improving** your environment, and to ensure that **only trusted container images are deployed to production**.

What should you do?

A. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Only page the on-call contact about novel issues** or events that haven't been seen before. **Use GNU Privacy Guard (GPG)** to check container image signatures and ensure that only signed containers are deployed.

B. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Page the on-call contact** when issues that affect resources in the environment are detected. **Use GPG** to check container image signatures and ensure that only signed containers are deployed.

C. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Only page the on-call contact about novel issues** that violate a SLO or events that haven't been seen before. Use **Binary Authorization** to ensure that only signed container images are deployed.

D. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Page the on-call contact** when issues that affect resources in the environment are detected. Use **Binary Authorization** to ensure that only signed container images are deployed.

6.4 Diagnostic Question 10 Discussion

You need to adopt Site Reliability Engineering principles and increase visibility into your environment. You want to **minimize management overhead** and **reduce noise** generated by the information being collected. You also want to **streamline the process of reacting to analyzing and improving** your environment, and to ensure that **only trusted container images are deployed to production**.

What should you do?

A. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Only page the on-call contact about novel issues** or events that haven't been seen before. **Use GNU Privacy Guard (GPG)** to check container image signatures and ensure that only signed containers are deployed.

B. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Page the on-call contact** when issues that affect resources in the environment are detected. **Use GPG** to check container image signatures and ensure that only signed containers are deployed.

C. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Only page the on-call contact about novel issues** that violate a SLO or events that haven't been seen before. Use **Binary Authorization** to ensure that only signed container images are deployed.

D. Adopt Google Cloud's operations suite to gain visibility into the environment. Use Cloud Trace for distributed tracing, Cloud Logging for logging, and Cloud Monitoring for monitoring, alerting, and dashboards. **Page the on-call contact** when issues that affect resources in the environment are detected. Use **Binary Authorization** to ensure that only signed container images are deployed.

6.1 – 6.4 Ensuring solution and operations reliability

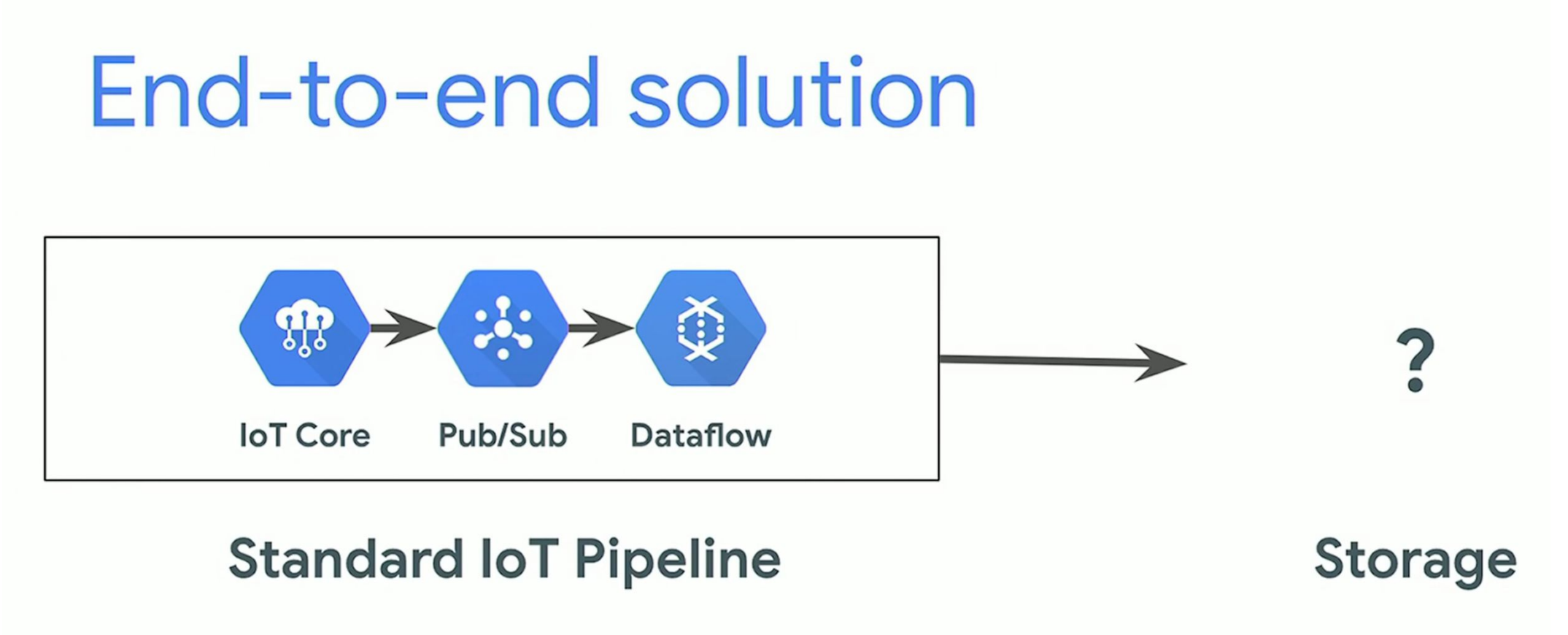
Resources to start your journey

- Google Cloud operations suite documentation
- Operations: Cloud Monitoring & Logging | Google Cloud
- Cloud operations grows with monitoring, logging, more | Google Cloud Blog
- Continuous Delivery | Google Cloud
- Concepts | Google Cloud Deploy
- Adopting SLOs | Cloud Architecture Center

Case study analysis

TerramEarth

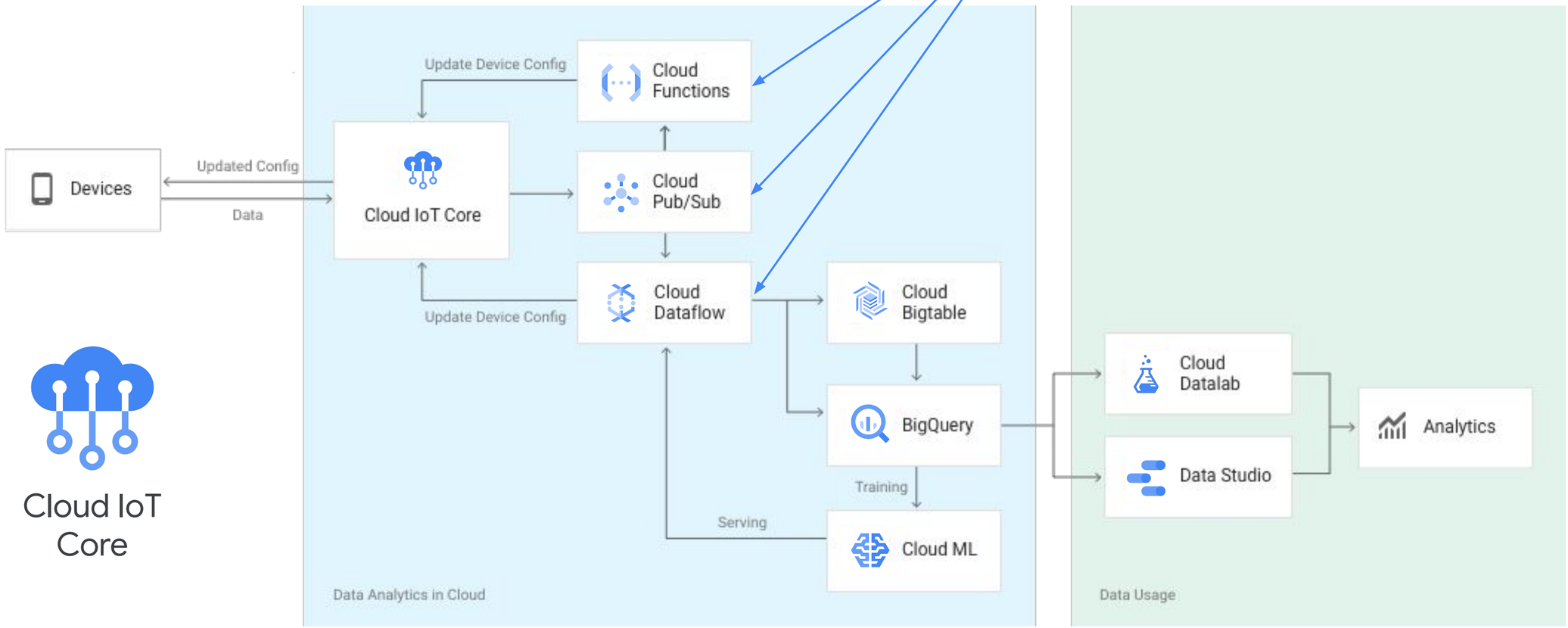
IoT pipeline



Internet of things (IoT)

TIP

Do you know how each of these services contribute to the IoT solution?



Cloud IoT Core

Proposed Technical Solutions

TerramEarth

- Similar to this architecture -> it's good to have a look at this solution.
- Telemetry streaming from vehicles: IoT Core service for managing devices and securing traffic.
- Critical IoT pipeline (near real-time checks of streaming telemetry data):
  - IoT Core -> Pub/Sub -> Dataflow (stream) -> BigQuery / BigTable (data storage) -> Application / BigQuery (query engine, since BigQuery can read from different sources, including BigTable)
- Analytics pipeline (daily batch uploads of telemetry from vehicles):
  - IoT Core -> Pub/Sub OR GCS -> Dataflow (batch) -> BigQuery (for analyzing this data) + optionally BigQuery ML (ML for predicting failures of vehicles based on data stored in BigQuery).
- Compute env: no concrete option. Possibilities that make most sense are: App Engine (portal for partners), GKE and Cloud Run.
- Interconnect: Possibly with 99.99% availability via multiple connections in different regions, using Cloud Router
- API creation, deployment and management: Apigee
- CI/CD:
  - Code repo: Cloud Source Repositories, Artifacts repo: Artifact Registry (which replaced old service: Container Registry)
  - Pipeline: GCP-native: Cloud Build + Cloud Deploy (new service, probably not yet covered in the PCA exam), 3rd party: Jenkins, Spinnaker
- Security-focused services: Binary Authorization, Web Security Scanner, Container Threat Detection, Data Loss Prevention, Organization Policy Service
- Secret management: Secret Manager
- Encryption key management: Key Management Service (KMS), possibly with HSM / EKM. What is envelope encryption?
- Network Monitoring:
  - VPC Flow logs -> Cloud Logging
  - Firewall rule logs -> Firewall Insights

Additional content 1

[READING]

- go through below sketchnotes:
  a. GCS
  b. Cloud SQL
  c. Cloud Spanner
  d. BigQuery
  e. BigTable
  f. Choosing a Database option
  g. Data Transfer Options
  h. [IMPORTANT] Cloud Operations Suite (Monitoring, Logging, Trace, Profiler, Debugger)
  i. Network Intelligence Center
  j. Dataproc
  k. Dataflow
  l. Data Loss Prevention
  m. Cloud compliance
  n. Data security
  o. Cloud security foundations

Additional content 2

- Read about private access options for GCP services.
- Go through this interactive Developer cheat sheet and make sure you are familiar with most of the services from different areas, mainly: Compute; Storage; Database; Networking; DevOps CI/CD; Identity and Security; Migration to Google Cloud.
- Play around with this online Architecture Diagramming Tool, which can:
  a. Help you sooo much in your future work as a cloud architect. No more 3rd party tools, no more searching for the appropriate GCP icons
  b. Visualize and memorize some of the common architectural diagrams. You can create different standard solutions like the one below with just one click of a mouse using the "Diagrams" area. Since many PCA questions require you to choose the right service setup, knowing those universal patterns will surely be helpful.
- Useful DLP blog post.
- Read about Cloud NAT service
- Secret Manager Best practices

Additional content 3

[VIDEOS]

- [HIGHLY RECOMMENDED] Example of how to define architecture for a serverless finance system: Designing a serverless finance system on Google Cloud
- Why you shouldn't aim at 100% uptime and what is an error budget: Why you shouldn't aim for 100% uptime
- SLIs, SLOs, SLAs in 8 mins: SLIs, SLOs, SLAs, oh my! (class SRE implements DevOps)
- DevOps vs SRE: What's the Difference Between DevOps and SRE? (class SRE implements DevOps)
- Cloud Operations Suite services: Cloud operations spotlight
- Private Service Connect: What is Private Service Connect?
- How to secure your cloud environment: How to secure your cloud environment
- Securing customer data: Securing customer data
- Network Connectivity Test: Get started with Connectivity Test in Network Intelligence Center
- Apigee: Intro to Apigee API management
- Apigee X: Introduction to Apigee X
- Securing hardware in GCP: Securing your hardware for your software
- Firewall Insights: Get Started with Firewall Insights in Network Intelligence Center
- Best Practices for Cloud Monitoring: Best Practices for Cloud Monitoring

Additional content 3

- Automating Cloud Monitoring dashboards: Automating Cloud Monitoring dashboards
- Top use-cases for serverless: Top use cases to start your serverless journey
- How to build APIs for serverless workloads with Google Cloud

[PODCASTS]

- Cloud Audit Logging
- The art of SLOs
- SRE vs DevOps
- Resiliency at Shopify

[DEEP DIVES]

- Web Security Scanner overview.
- Technical overview of Internet of Things.
- [video] Managing IoT Storage with Google's Cloud Platform (Google I/O'19).
- Chapters 2, 6, 8 and 17 (1st half) from Google SRE book.

Make sure to...

Enjoy the journey as much as the destination!

Review and study planning (advanced learning path)