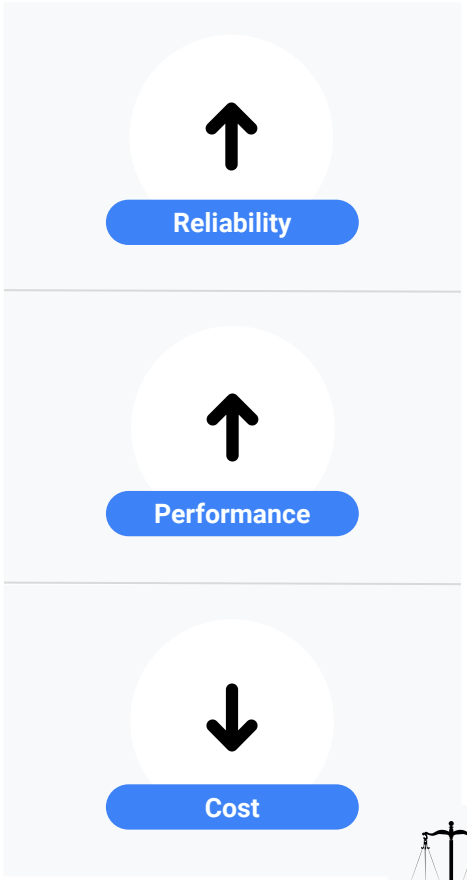
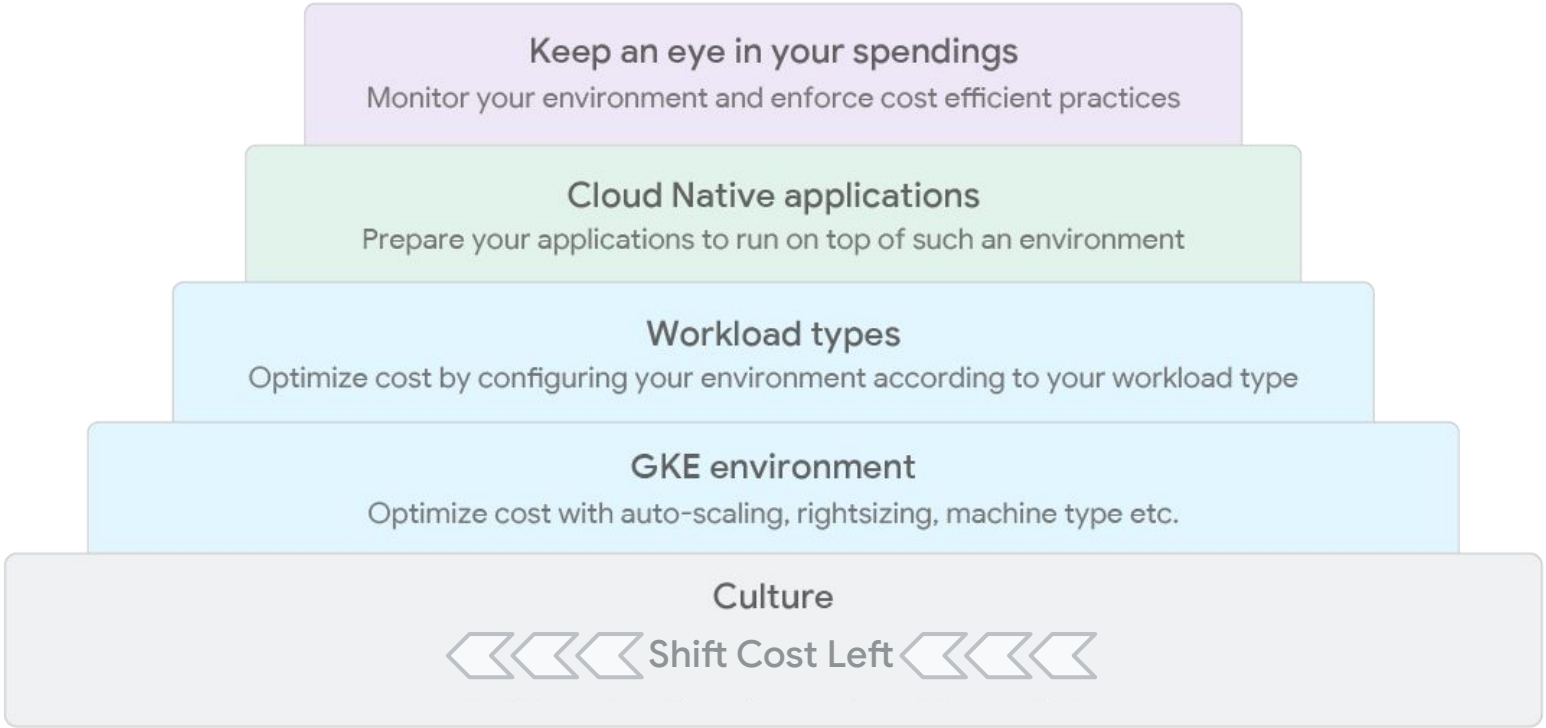
GKE Optimization

Continuously deliver **business value** **by** running **reliable**, **performant**, and **cost efficient** applications



Google Cloud



bit.ly/**gke**-cost-optimization

Google Cloud

Agenda

01

Understanding GKE options

To balance cost, reliability and scaling performance on GKE it is necessary to first understand what options do you have and how autoscale works

02

Reliable Kubernetes applications

Once you understand the basics of GKE autoscaling and other useful cost-optimized configurations, it is time to prepare your application to run on top of such an environment

03

Monitoring and enforcing

Platform/infrastructure teams need to understand which group/application is resource hungry and need to make sure everybody is following the company's policies

04

Culture

Culture is the heart of every innovation. To lowering cost it is not different.

Understanding GKE options

Google Cloud

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...

ERIN

It is open source platform for managing containers.

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

- Horizontal Pod Autoscaler (HPA)
- Vertical Pod Autoscaler (VPA)
- Different Pod Sizes
- More Pods

INFRASTRUCTURE

- Cluster Autoscaler (CA)
- Node Auto-Provisioning (NAP)
- Different Node Pools
- More Nodes

How does it handle traffic growth?

How do I secure apps using GKE ?

SECURE APPS

- Data Encryption
- Certified Images
- Private Clusters
- Idenity & Access - IAM - RBAC

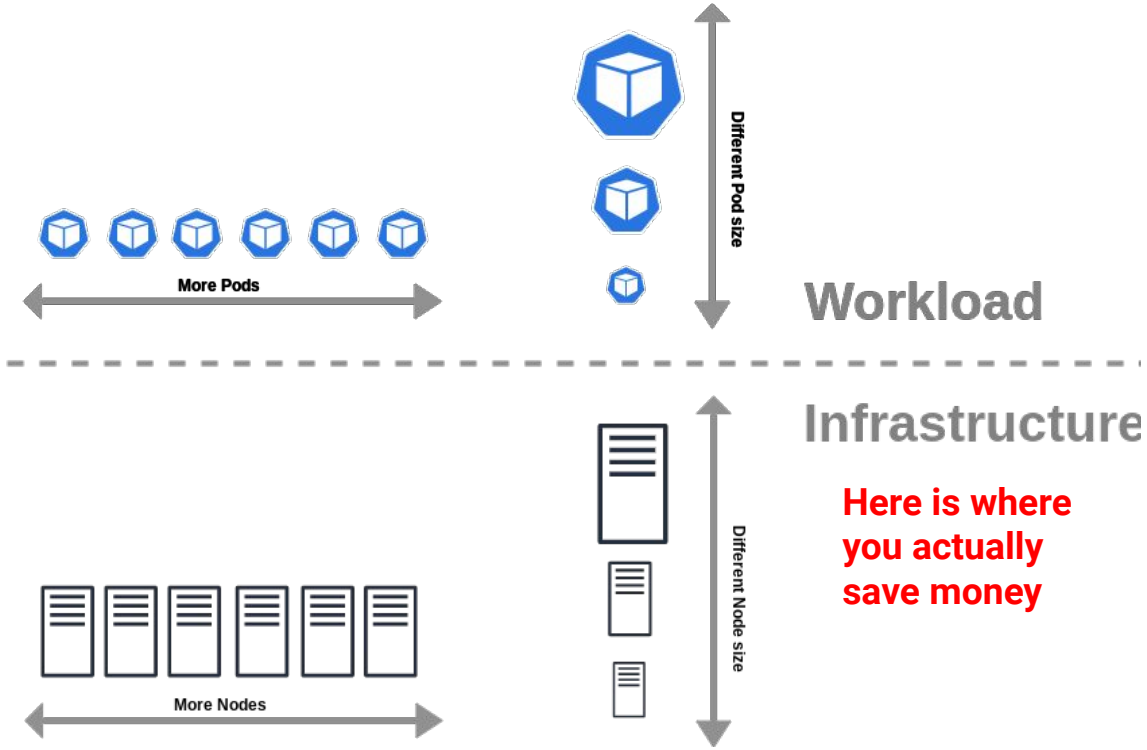
TRUSTED NETWORKING

- Global VPC
- Global Load Balancing
- Cloud Armor
- Network Policy

SOFTWARE SUPPLY CHAIN SECURITY

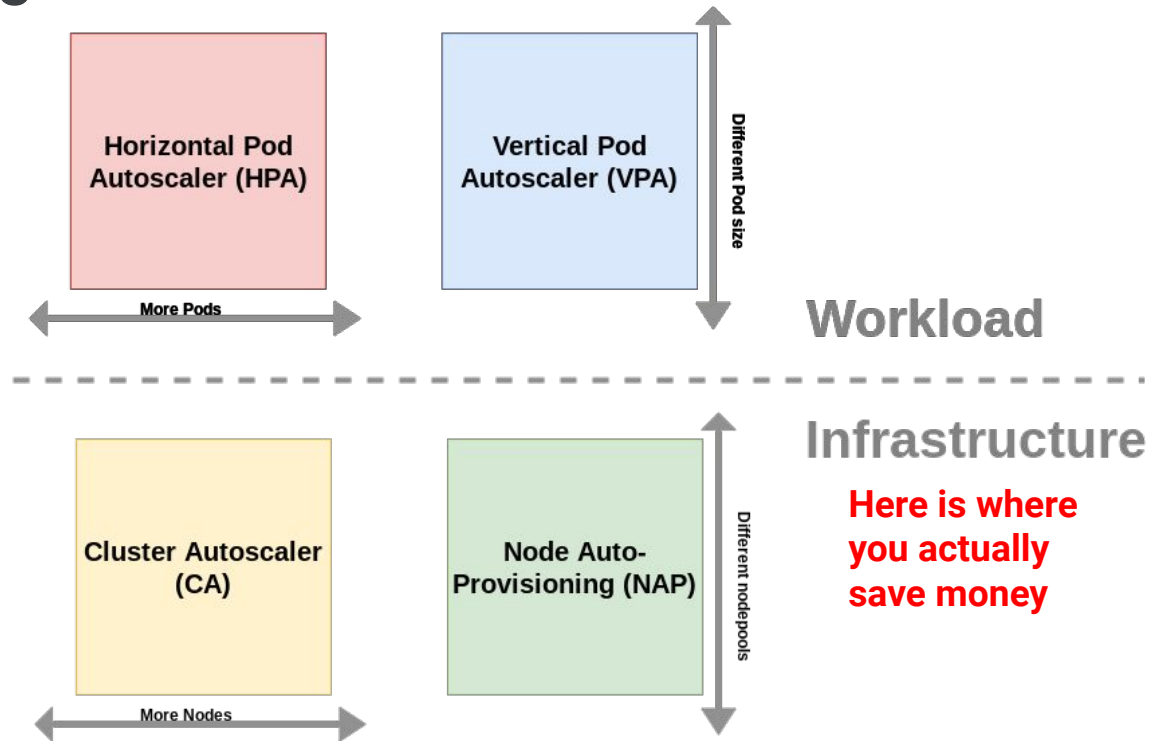
- Binary Authorization
- Vulnerability Scanning
- Managed Base Image

The 4 scalability dimensions



Google Cloud

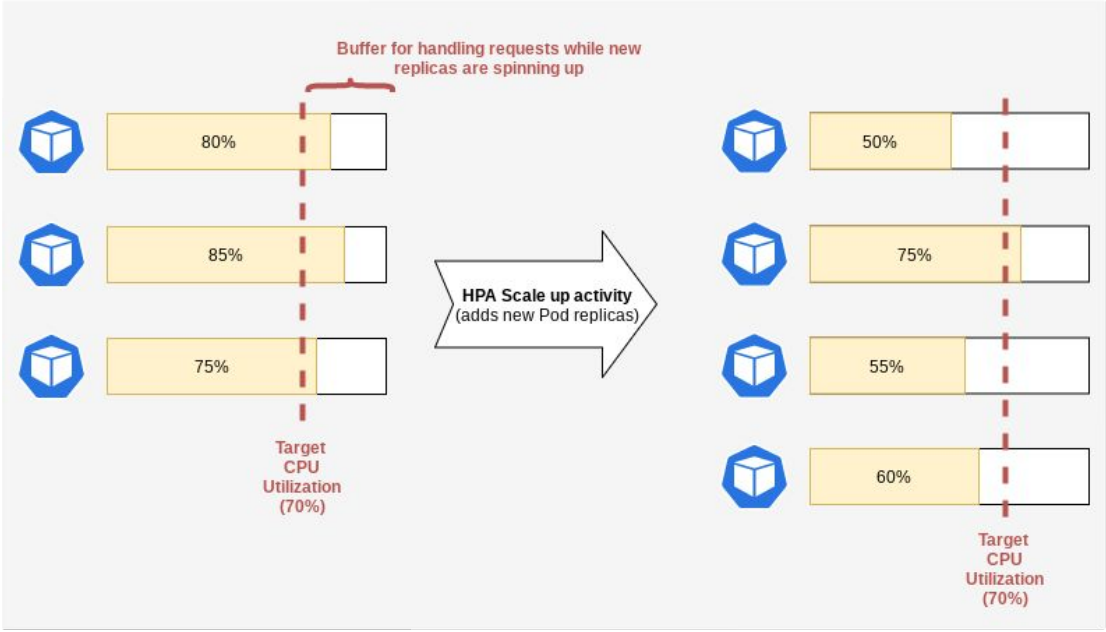
The only provider which supports the 4 scalability dimensions



Google Cloud

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.

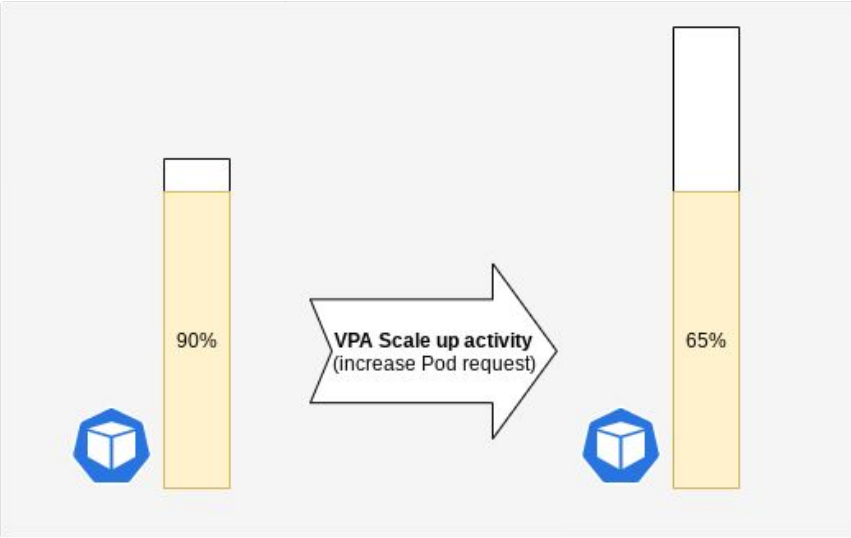


Google Cloud

For more information, see Configuring a Horizontal Pod Autoscaler.

Vertical Pod Autoscaler (VPA)

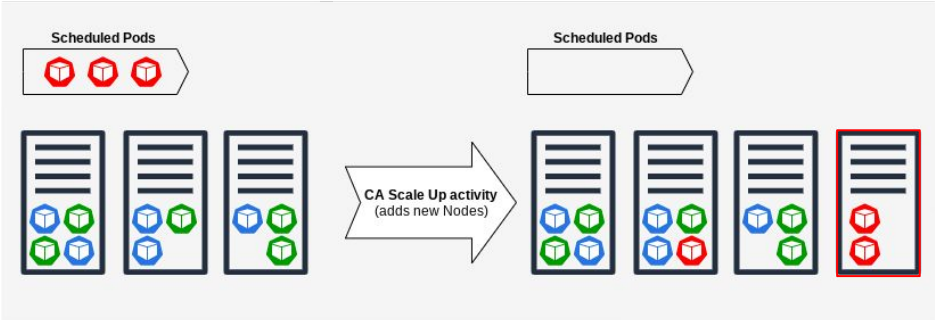
- Indicated for: stateless and stateful workloads not handled by HPA or when you don't know the proper Pod's resource requests
- Don't use VPA either Initial or Auto mode if you need to handle sudden spikes in traffic. Use HPA instead.
- Modes:
  - **Off**: recommendation
  - **Initial**: do not restart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

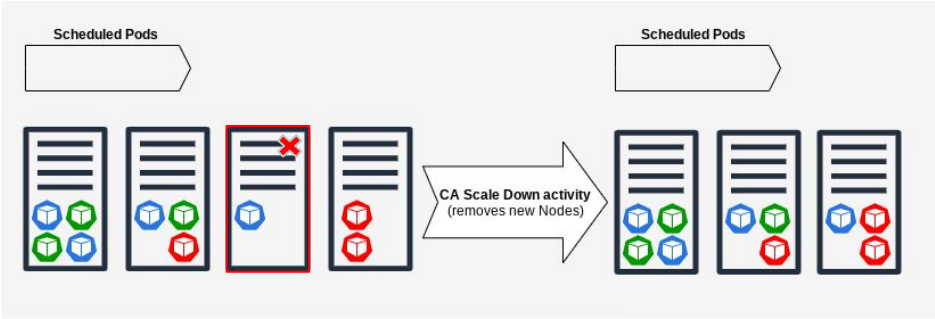


For more information, see Configuring Vertical Pod Autoscaling.

Google Cloud

Cluster Autoscaler (CA)



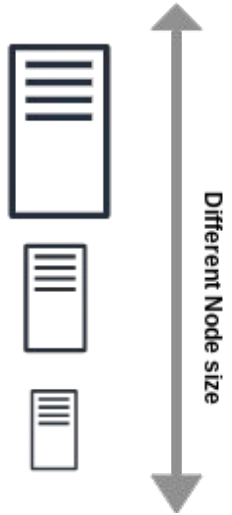


Google Cloud

For more information, see Autoscaling a cluster.

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

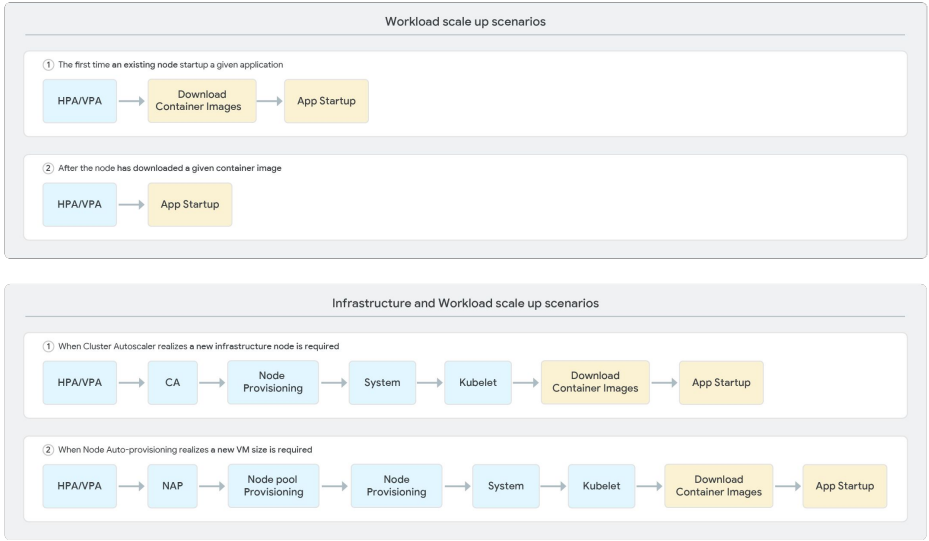


Google Cloud

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

Autoscaler and Over-provisioning

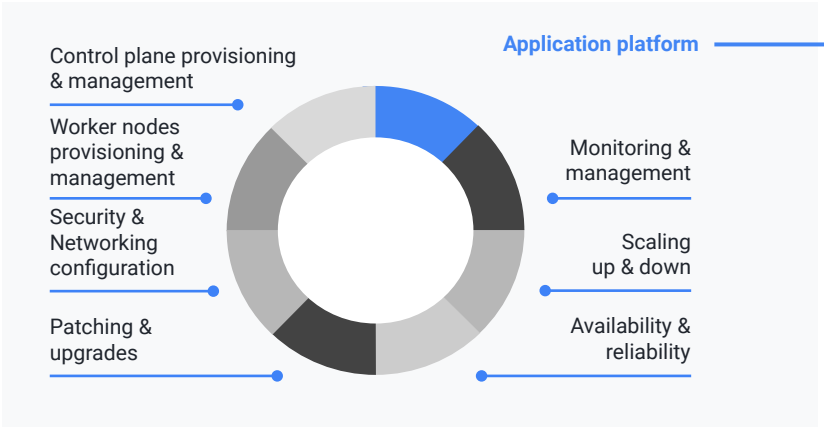
- There is no one configuration to fit all possible scenarios.
- Don't squeeze your cluster too much.
- You must over-provision but only for reserving the necessary buffer to handle the expected peak requests during scaling up activities.



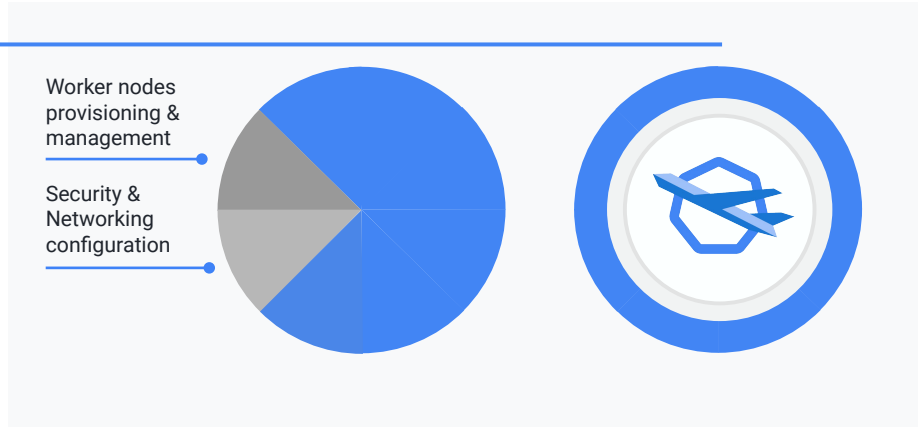
Google Cloud

GKE Autopilot

DIY Kubernetes Service



Google Kubernetes Engine (GKE)



GKE Autopilot is a new **mode of operation** in GKE
Mode of operation = level of control over a GKE cluster

Standard mode
Managed Kubernetes with configuration flexibility

Autopilot mode
Optimized Managed Kubernetes with hands-off experience

Google Cloud

For more information about the differences between Standard and Autopilot, see Autopilot overview and limitations

Demo

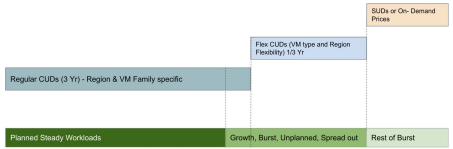
Compare Autopilot and Standard

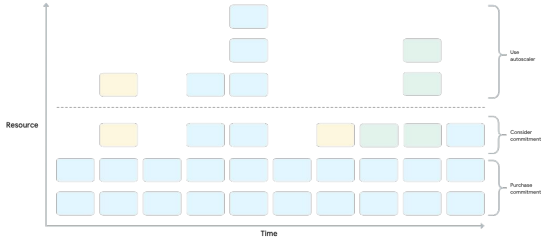
Sign for committed-use discounts

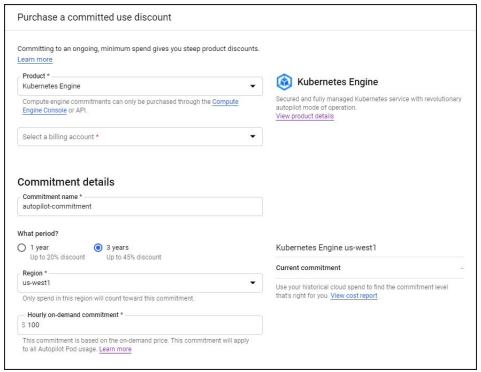
- GKE Standard

	Standard CUD	Flexible CUD (private preview)
Purchase unit	Resource based (e.g N1 vcpu, GB memory, local SSD, GPU)	Spend based (ex: $100)
Discount off On-demand rate	1 year discount 37% 3 year discount 55%	1 year discount 28% 3 year discount 46%
Machine family Eligibility	Applies to a specific machine family	Applies to ALL general purpose and compute optimized machine families
Regional Eligibility	Applies to a specific region	Applies to ALL regions

- GKE Auto Pilot:
  - Spend-based discount up to **45% for a 3 year commitment**
- Go incremental and mix Standard and Flexible CUDs







For more information, see Committed use discounts and Autopilot CUD

Order of Application of discounts in billing

1

Standard CUDs

Apply Standard CUDs to eligible usage (machine family and region specific)

2

Flex CUDs

On demand usage that wasn't discount by standard CUDs can be eligible for flex CUDs

3

SUDs (if applicable)

The remaining usage not covered by any CUDs can be eligible for flex SUDs

Spot and Preemptible VMs (PVMs)

- **Up to 60 - 91% discount**
- **Indicated for: fault-tolerant apps and jobs, development and staging environments**
- Avoid large machines to better find capacity and make sure your apps can graceful terminate in <25s
- Spot are enhanced PVMs
  - PVMs: GA → VM instances last a maximum of 24 hours
  - Spot VMs: Public Preview → VM instances have **no maximum uptime threshold.**
  - Preemption rate reduction is under development for both Spot and PVMs
- Price is considerably lower than standard Compute Engine VMs
  - Both Spot VMs and Preemptible VMs will be variable charging from 2022/Q1 (monthly reviewed depending on supply/demand)
- Others:
  - Pod Disruption Budget may not be respected
  - For GKE Standard: Set standard node pools fail back, in case of Spot/PVMs stockout.
  - For Node Auto Provisioning: set nodeselector to `cloud.google.com/gke-spot: true`, toleration to `cloud.google.com/gke-spot="true":NoSchedule,` and `terminationGracePeriodSeconds: 25`
  - For GKE Autopilot: nodeselector to `cloud.google.com/gke-spot: true` and `terminationGracePeriodSeconds: 25` are enough
- Prior GKE 1.20.5-gke.500+
  - Install Node Termination Handler (not an official Google project) provides an adapter for translating Compute Engine node termination events to graceful pod terminations in Kubernetes. Not needed on GKE 1.20.5-gke.500+: k8s Graceful Node Shutdown

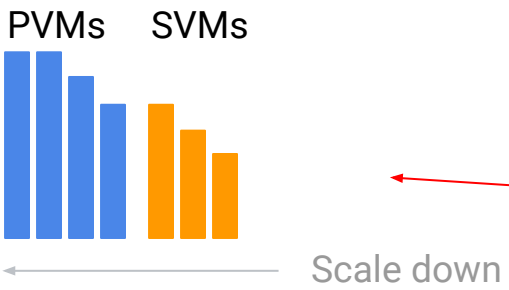
Google Cloud

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

CUD + Spot/Preemptible VMs strategy

Scale up (CUD VMs → Spot/PVMs → Standard VMs)

- Keep cluster warm at CUD
  - For example, Nodepools **with CA disable**
- Create Spot/PVM node pools
  - With CA enabled
  - This work best if your CUD is optimal and spikes above CUD are short (ie. less than 4 hours)
  - Avoid big Spot/PVMs to find capacity
  - Define a cap to avoid big disruption at bulk preemption
  - Install preemption handler on GKE < 1.20.5-gke.500+. Make sure you follow these steps
- Create Standard VM node pools - Affinity
  - With CA enabled
  - Define bigger max than PVM node pools to make sure your cluster will grow as needed in case of bulk preemption



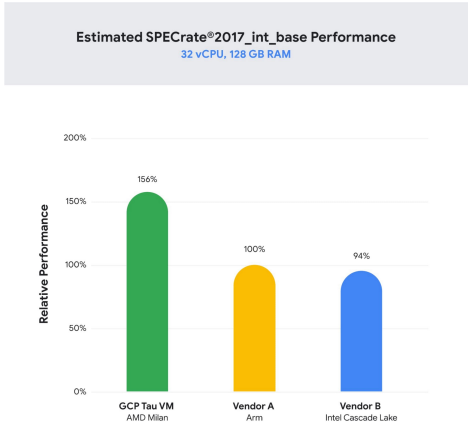
Scale down (Standard VMs → Spot/PVMs → CUD)

- HPA natively prefer recently created pods during scale down
- The best way to implement it is by using CA **optimized-utilization** profile
  - Optimize utilization prefers most utilized nodes. So downscale tend to evict Standard VMs first
  - Pay attention that not all workloads are prepared for optimized profile once it spin up/down nodes with much bigger frequency, so there may be some instability due to frequent scale ups.
    - Follow K8s best practices (small images, fast startup, graceful shutdown, readiness probe, etc)

Google Cloud

E2, N2D, and T2D (2021/Q3) Machine types

- **E2**s are the most cost-optimized VMs (**cheaper than any ARM offer from other vendors**)
  - Performance comparing to N1(99th percentile),
  - You don't choose compute platform (Intel, AMD).
- T2Ds delivers the best price/performance (**Equivalent price with better perf than Vendor X - ARM**)
  - Up to ~56% increase in performance, and 42% cost/performance
  - AMD Milan (3rd Gen): simultaneous Multithreading (SMT) disabled and at 1 vNuma node
  - T2Ds are meant for x86 compute intensive or highly parallel workloads
- **N2D**s cost-optimized AMD option
  - Best performance and lower price than N2s
- All these VM types can be launched as Preemptible VMs.



Google Cloud

For more information about E2 VMs and how it compares with other machine types at Google Cloud, see Performance-driven dynamic resource management in E2 VMs and Machine types.

Select the appropriate region

- Where do you run your GKE clusters?
- Did you know the compute price in US can be 30-40% cheaper than SAO?
- Note: consider data transfer between regions in your math.

Google Cloud

For more information on how to choose the right region, see Best practices for Compute Engine regions selection.

Review inter-region egress traffic in regional and multi-zonal clusters

- Either Regional or multi-zonal clusters
  - Need to pay egress traffic between the zones
- Suggestions
  - Deploy development clusters in single-zone
  - Enable GKE usage metering its network egress agent (not enable by default)
  - Consider inter-pod affinity/anti-affinity to co-locate dependent Pods from different services in the same nodes or in the same availability zone. Note this can compromise HA.
  - In GKE 1.23 you will be able to favor traffic to local zones using TopologyAwareHints. However, you may face some issues if the traffic is all concentrated in one unique zone. For example: HPA may not realize it needs to create new Pods because only Pods in one zone are running out of cpu (33%), all the others (66%) are running idle.

Review your logging and monitoring strategies

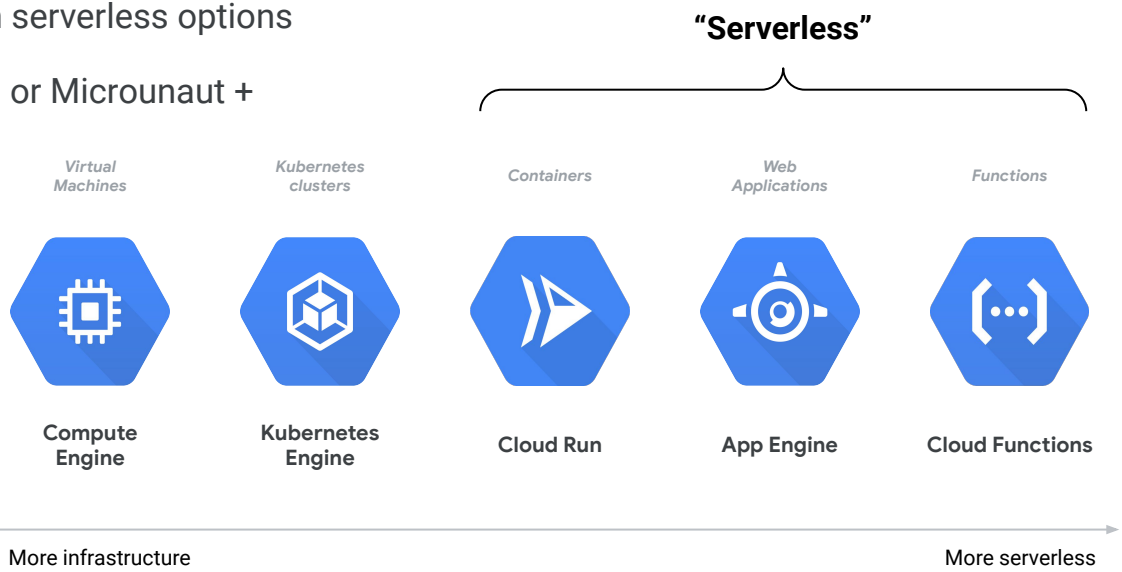
- Cloud Logging and Cloud Monitoring are pay-as-you-go systems
- The more you log and the longer you keep the logs, the more you pay
  - Exclude logs to avoid ingestion cost
    - However, the best place to do such a work is in the app
    - **Note**: If you create new buckets & routers, exclude routed logs from _Default, otherwise you will changed twice
  - Monitor log bytes per application
  - Create an alerting system for logs

Google Cloud

For more information, see Cost optimization for Cloud Logging, Cloud Monitoring, and Application Performance Management.

Long tail workloads

- Usually receive a few requests per day
- If your app accepts cold startup, you can consider Google's Serverless options
  - Cold startup benchmark on serverless options
  - For Java, consider Quarkus or Microunaut + GraalVM (see benchmark)

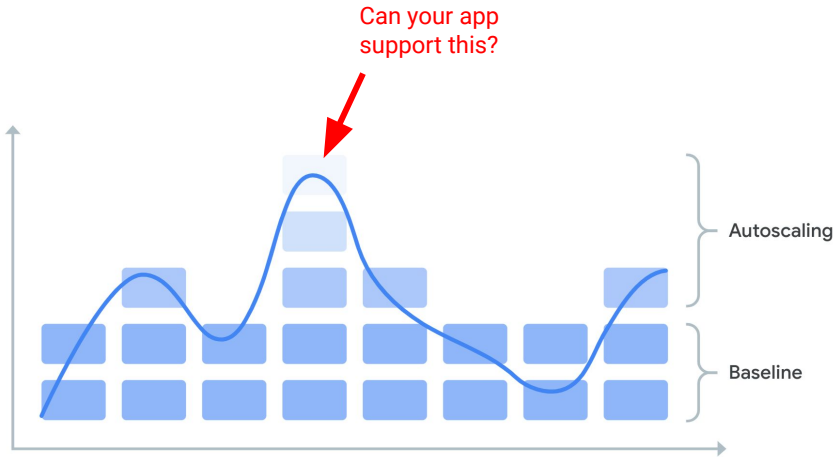


Reliable Kubernetes applications

Google Cloud

Understand your application capacity

- Per Pod capacity:
  - Isolate a single application Pod replica with autoscaling off, and then execute the tests simulating a real usage load
- Per app capacity
  - Configure your Cluster Autoscaler, container replicas, resources requests and limits, and either Horizontal Pod Autoscaler or Vertical Pod Autoscaler
  - Stress your application with more strength to simulate sudden bursts or spikes.
- To eliminate latency concerns, these tests must run from the same region/zone the application is running on Google Cloud
- Or use VPA in recommendation mode



Google Cloud

For one example on how use Kubernetes to perform such tests, see Distributed load testing using Google Kubernetes Engine tutorial.

Make sure your application can grow both vertically and horizontally

- Make sure your application can handle spikes by adding more CPU and memory, or by increasing the number of Pod replicas.
- This gives you the flexibility to experiment what fits your application better.
- Unfortunately, there are some applications that are single threaded or architecturally limited to a number of workers/sub-process

Set appropriate resources requests and limits

- Set your container resources to use the same amount of memory for both requests and limits, and a bigger or unbounded CPU limit.
- Vertical Pod Autoscaler in the recommendation mode can also be used to help you figure out CPU and memory usage for a given application
    - It is recommended to enable it in a production like environment to face real traffic

```
spec:
  containers:
    - name: wp
      image: wordpress
      resources:
        requests:
          memory: "128Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
```

Make sure your container is as lean as possible

- **Having the smallest image as possible.**
  - It is a best practice to have small images because everytime Cluster Autoscaler provisions a new node for your cluster, the node needs to download the images that will run in such a node. The smaller the image, the faster the node can download it.
  - Eg. with Go lang you can get images with ~10Mb. For Java consider jib (for lean images); GrallVM (for low latency native binaries); Quarkus, Micronaut, and Spring Boot (for Kubernetes Native Java stack).

- **Starting the application as fast as possible.**
  - Some applications can take minutes to start due to class loading, caching, etc. Whenever a Pod requires a long start up, your customer requests may fail while your application is booting up.

For more information about how to build containers, see Best practices for building containers.

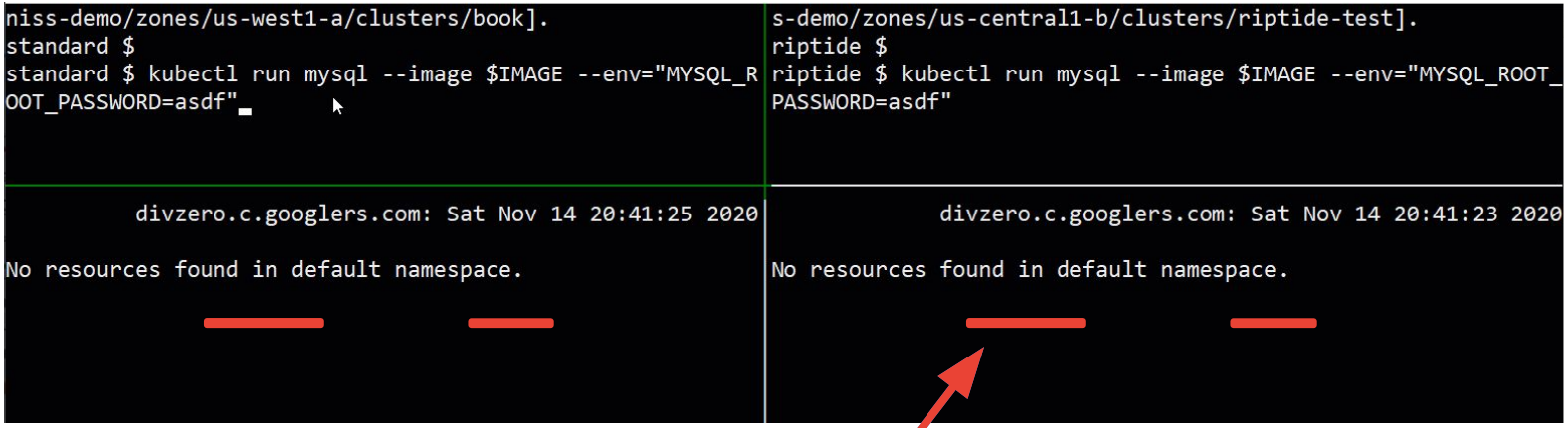
GKE image streaming for fast application startup and autoscaling

Regular image pulling

1. Container image is downloaded to the VM
2. Application starts

GKE image streaming

1. Application starts. Files are streamed from remote filesystem.



GKE Image Streaming runs mysql container (~140MB image) in 3s
(vs 16s in regular image pulling process)

Google Cloud

For more information read GKE image streaming for fast application startup and autoscaling

Monitoring and enforcing

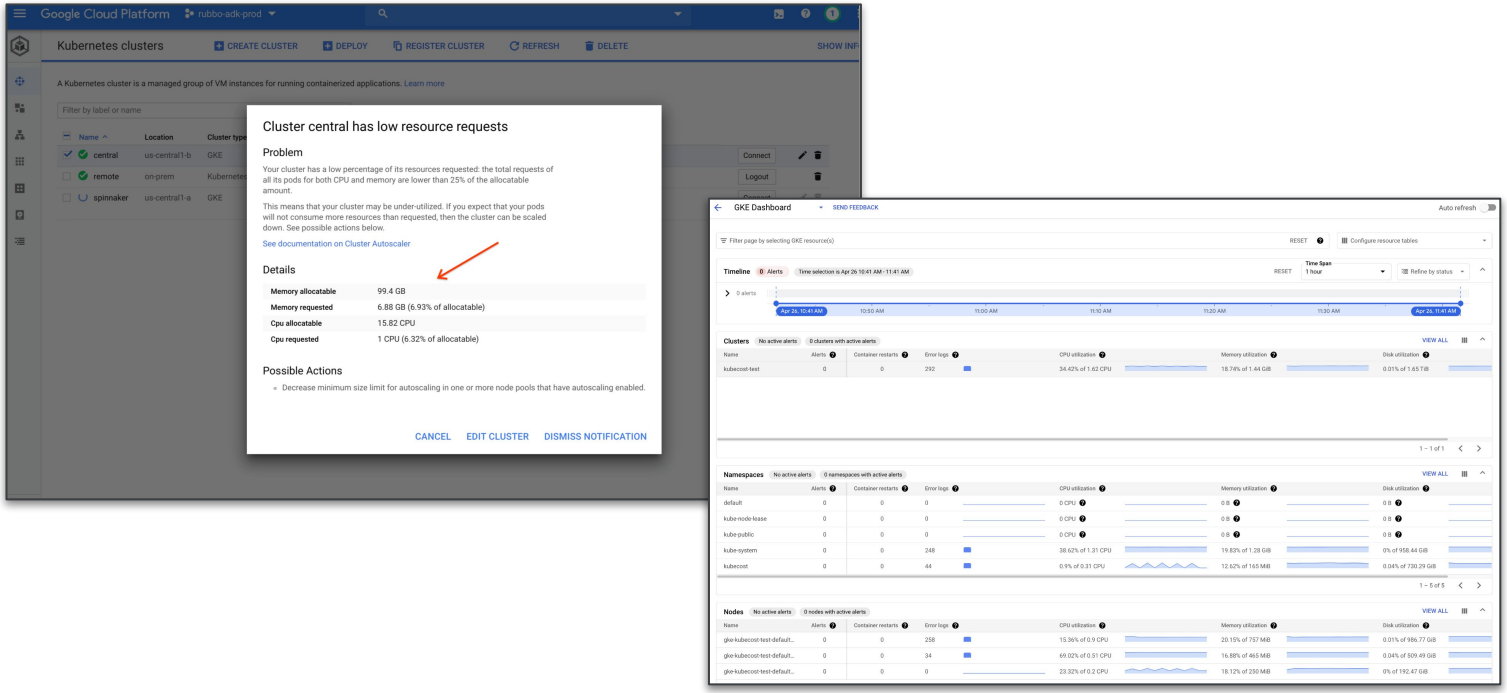
Google Cloud

The Golden Signals for GKE Cost Optimization

Continuously measure

To learn more how to set up your monitoring environment, see [Monitoring your GKE clusters for cost-optimization using Cloud Monitoring]

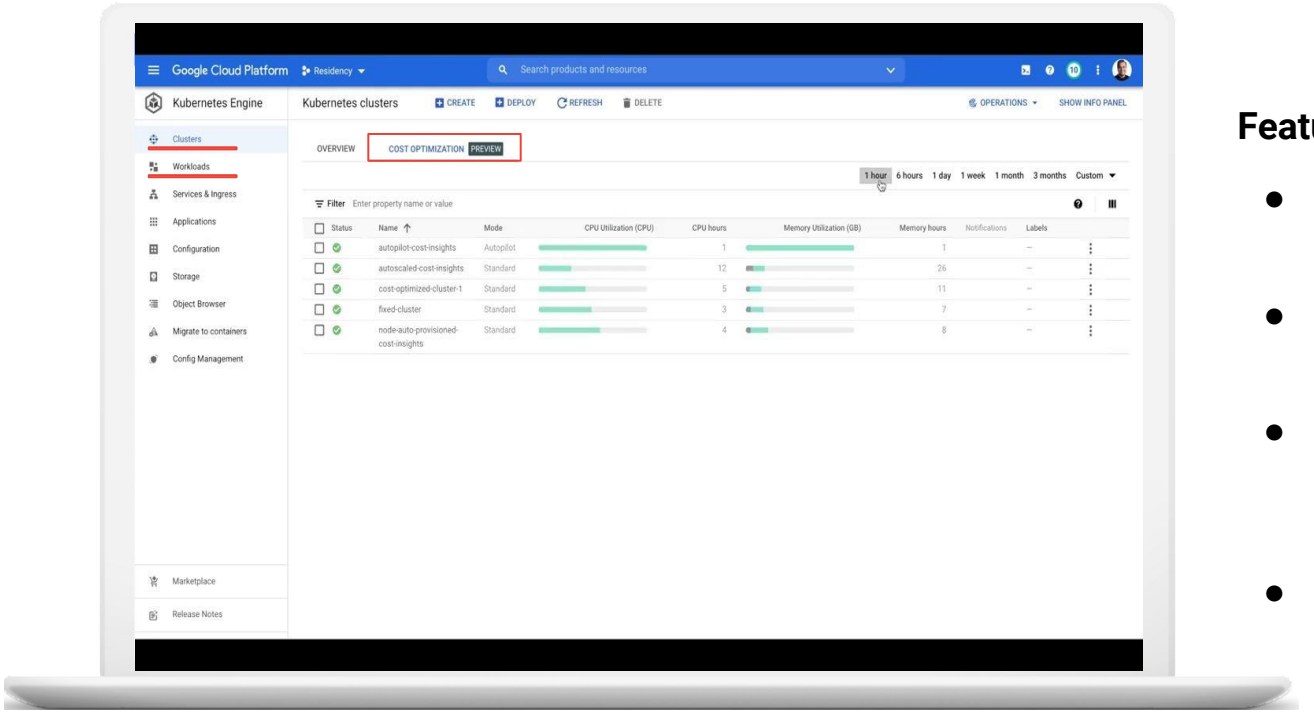
Observe your GKE clusters, set SLOs, and watch for recommendations



Google Cloud

For more details, see Observing your GKE clusters and Getting started with Recommendation Hub.

Cost optimization insights from day 1

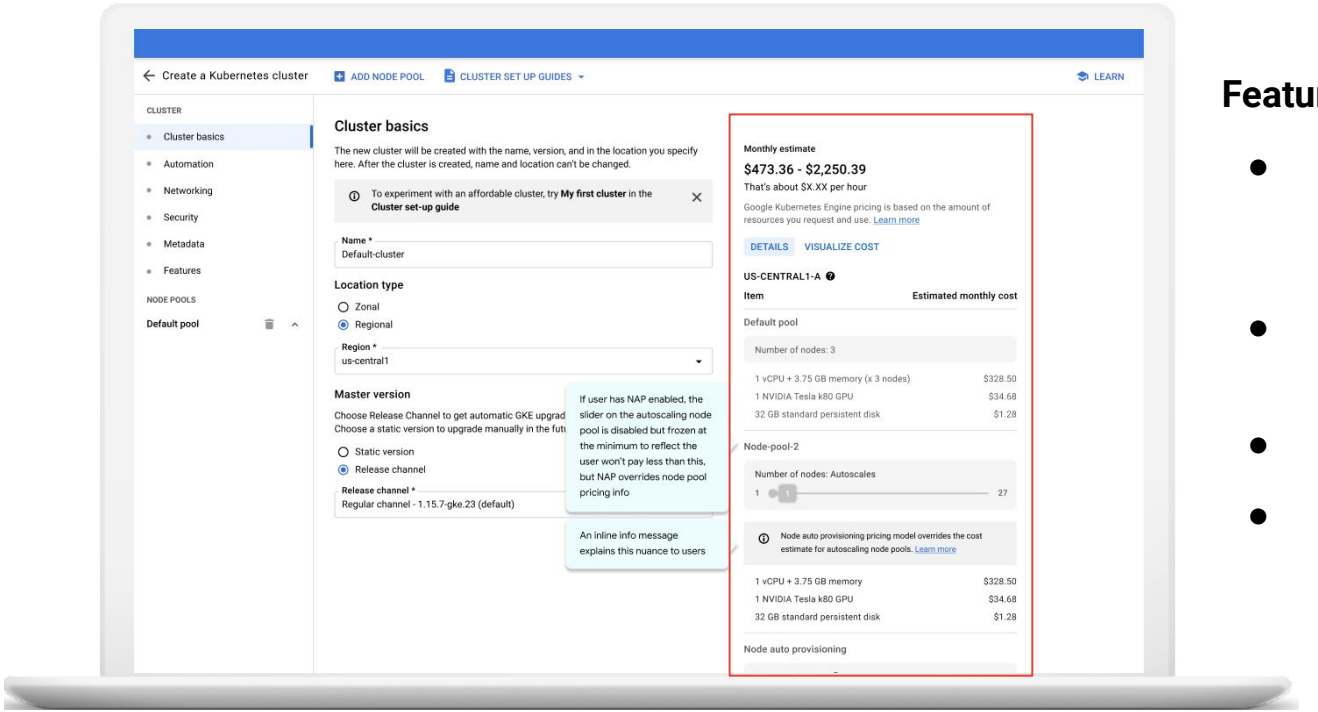


Features:

- Utilization, app right sizing, and bin packing info
- Cluster level and Workload level
- Timer picker to troubleshoot or to understand data over time
- Many more

Google Cloud

Cost Widget for GKE Std clusters (PREVIEW PLANNED FOR Q1)

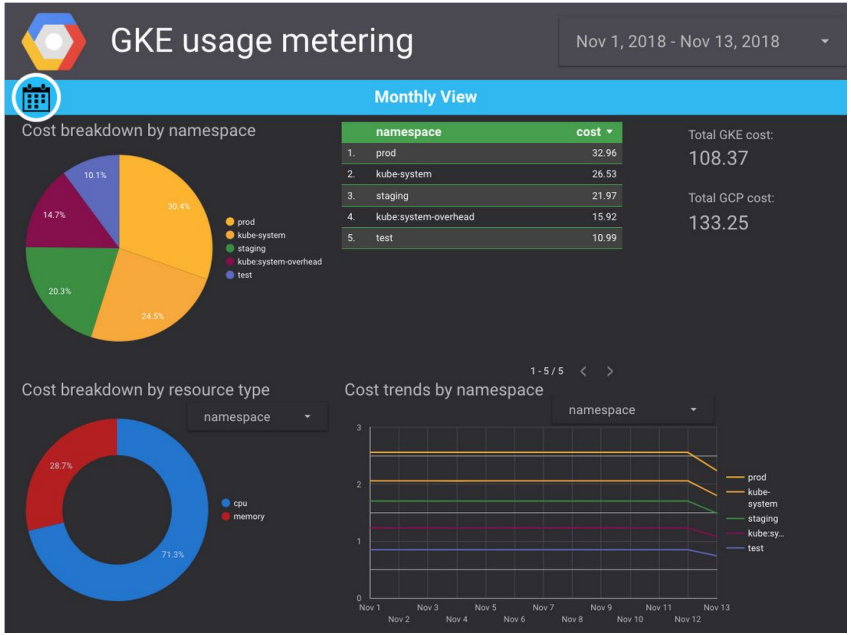


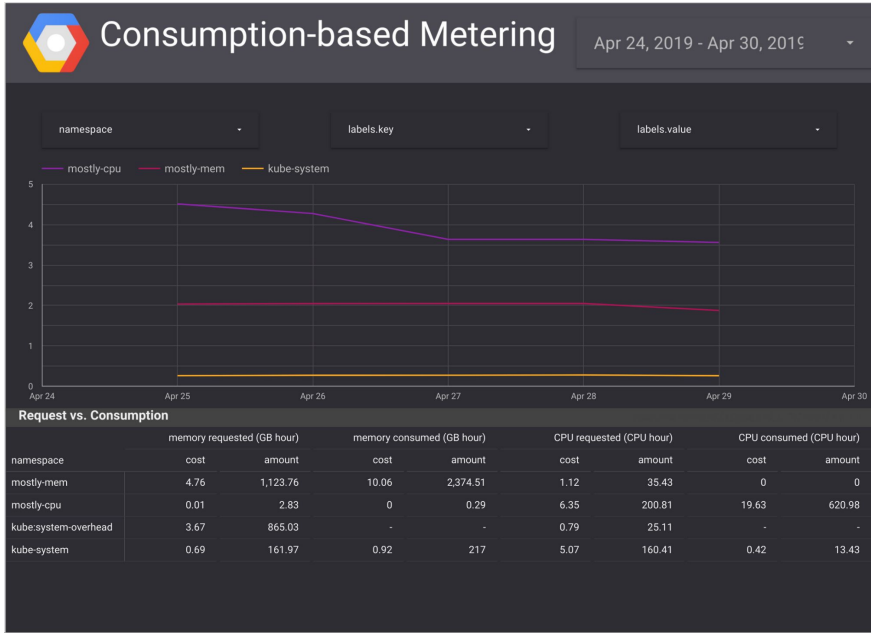
Features:

- Understand your GKE cost while creating / editing your clusters
- Understand peer Nodepool cost cost range
- To figure out autoscaling cost
- Many more

Google Cloud

Enable GKE usage metering





Google Cloud

For more about GKE usage metering and its prerequisites, see Understanding cluster resource usage.

Culture:
Shift-Left Cost

04

Google Cloud

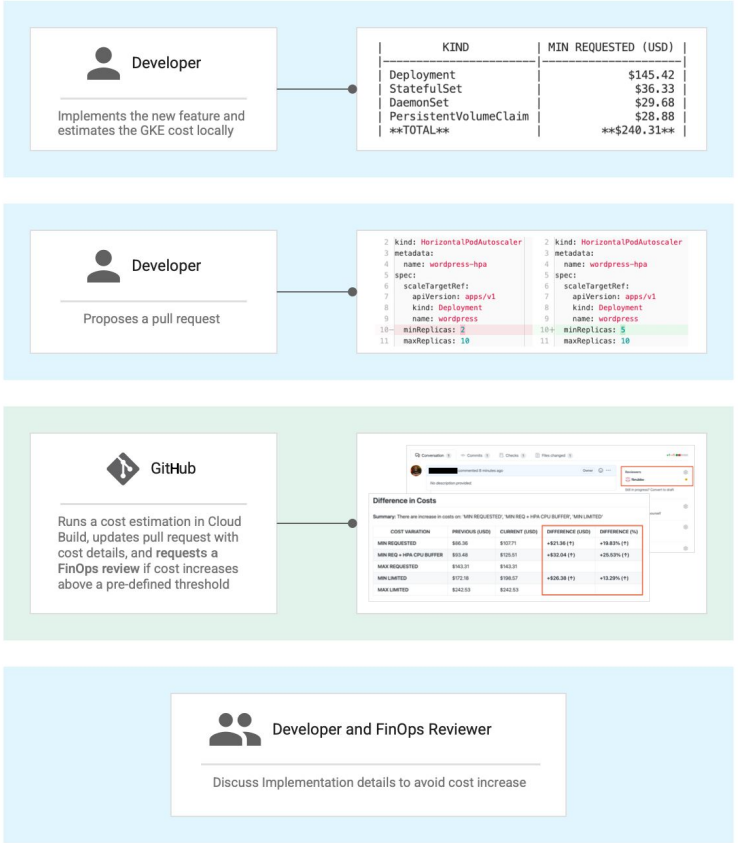
Shift-Left Cost

- Create learning incentives and programs
- Align your development team with the business goals
  - By creating metrics such as: **Cost per 1k clicks**, **Cost per user**, etc
  - Consider adding **cost per business goals** as part of the teams' end of year bonuses
- Make sure developers and operators have real-time visibility of their spend
- Define a chargeback policy for accountability
- Gamify it!!!!
- Provide spot bonuses to teams/developers
  - For those who manage to lower and keep the cost lower by x%
- Create guardrails since the beginning of the development cycle
  - Using K8s Resource Quotas, Anthos Policy Controller and Cost estimation
- Close the loop by delivering automated recommendations directly Developers
  - IDE recommendations, alerting systems, dashboards, etc

FinOps movement

Shift-left cost

Cost estimation at Merge/Pull Request



Google Cloud

To learn more how to implement this shift-left cost approach, see [Estimate your GKE costs early in the development cycle using Gitlab] and [Estimate your GKE costs early in the development cycle using Github]