Multitenancy in Kubernetes:

Better walls make better tenants

Adrian Ludwin, Senior Software Engineer
Wednesday, June 17, 2020
aludwin@google.com

Google Cloud

Goal of this session

Understand Google's opinionated **best practices** for enterprise multitenancy on Kubernetes, including when and how to apply them.

Google Cloud

Topics

1. Why use multitenancy?
2. What are the principles of multitenancy?
3. How do I implement multitenancy?
4. Advanced topics
5. Conclusion

Why use multitenancy?

Google Cloud

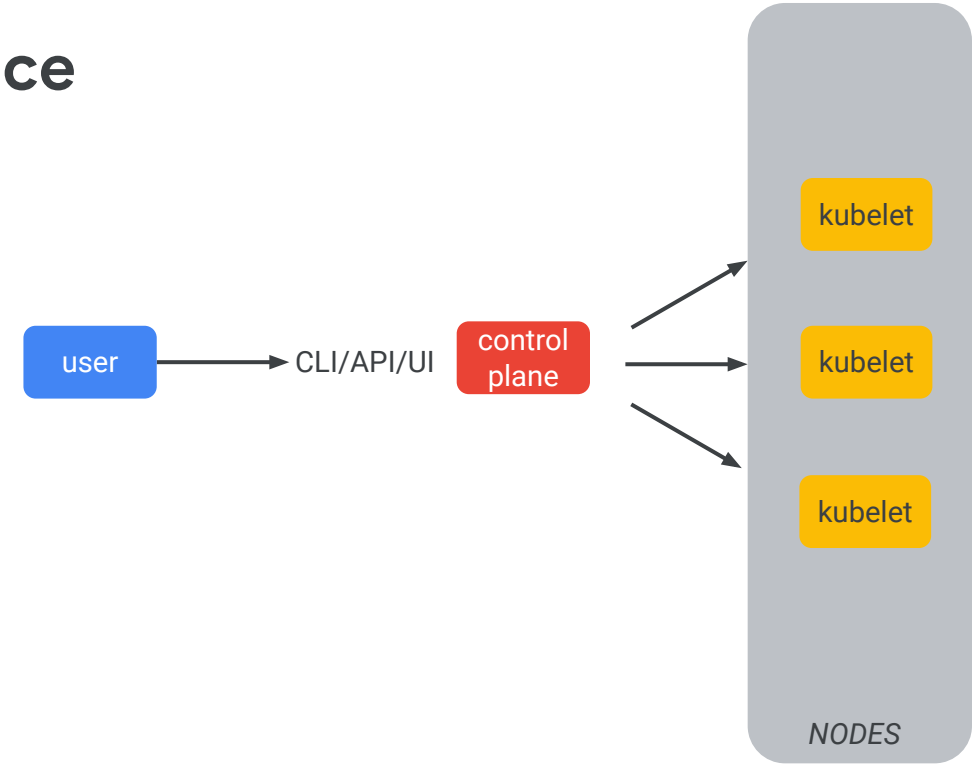
What companies care about

Cost

Velocity

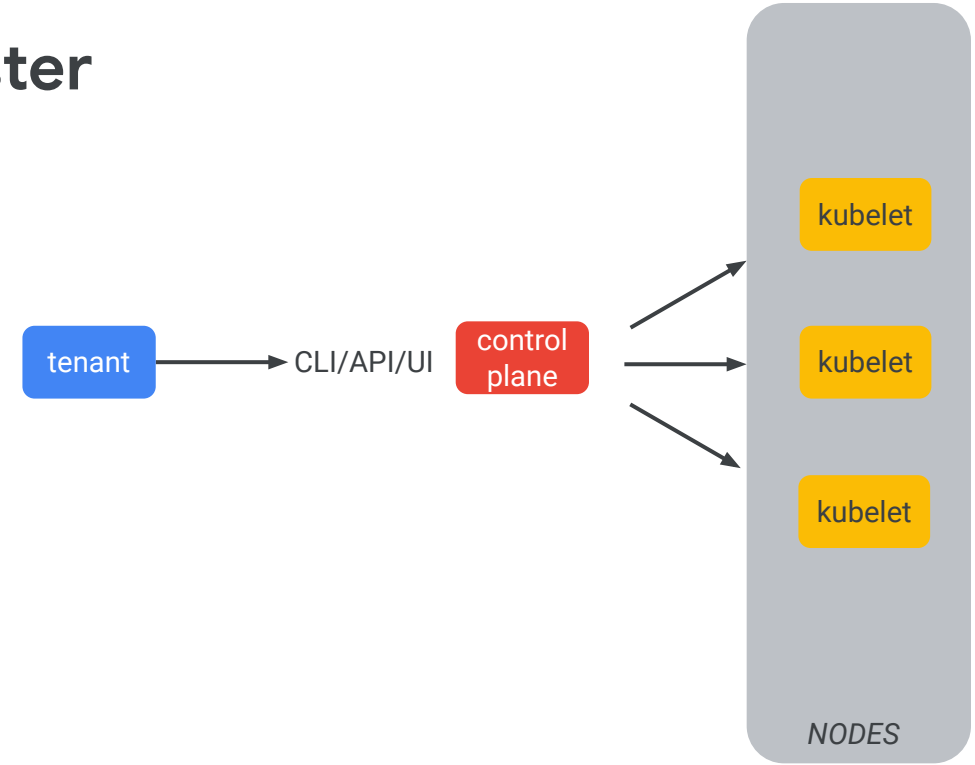
Google Cloud

Kubernetes at a glance



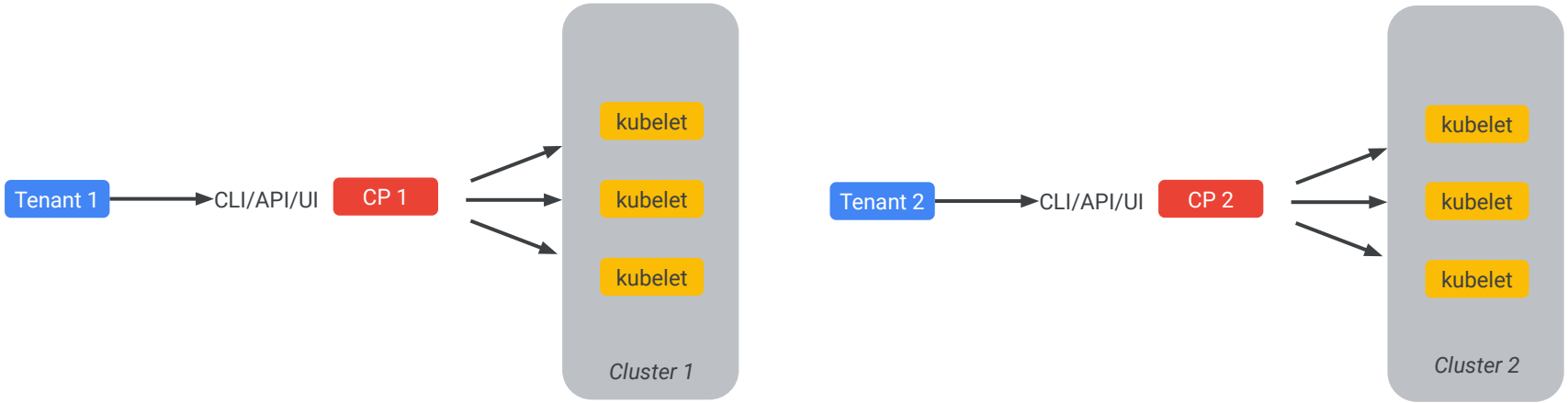
Google Cloud

One tenant, one cluster



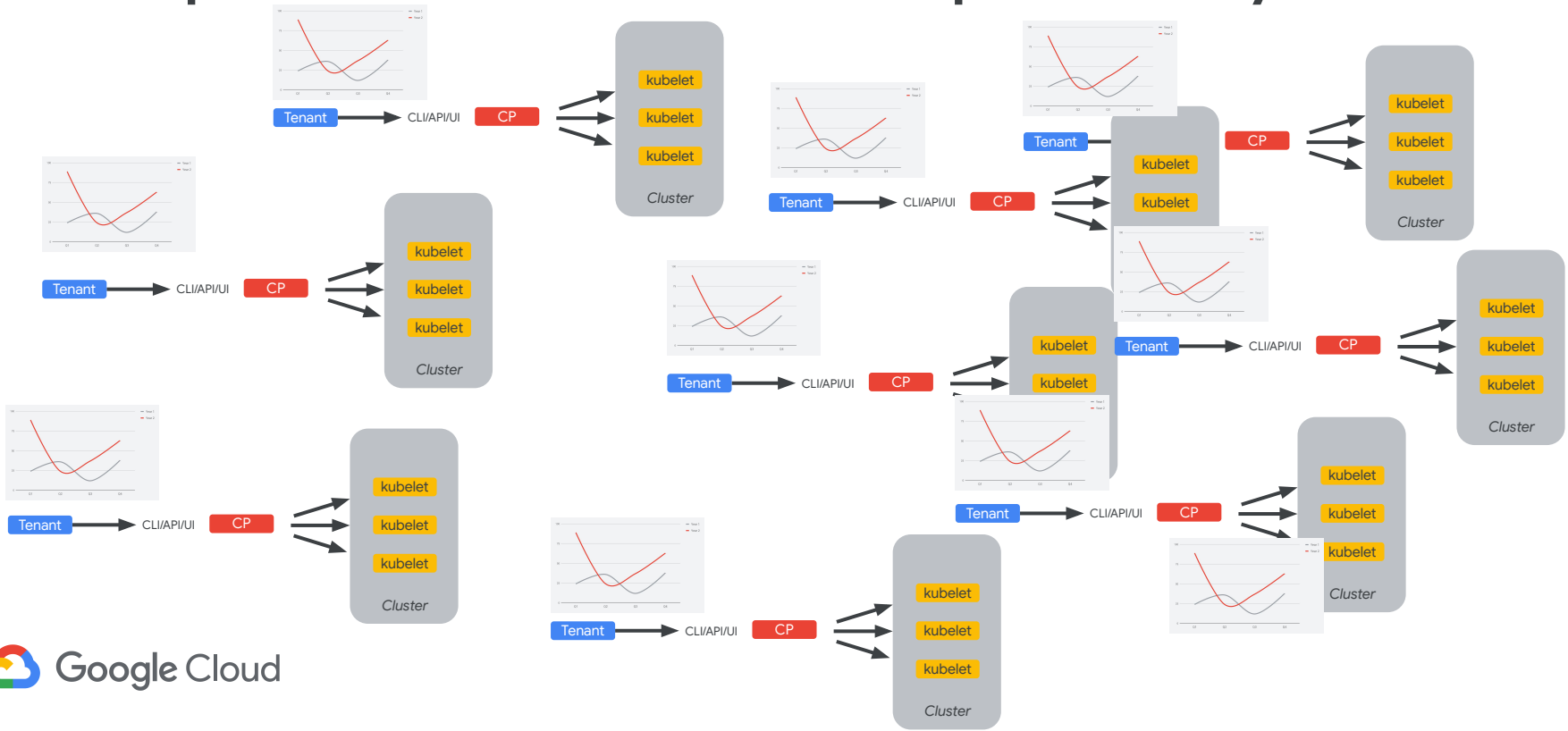
Google Cloud

Multiple tenants, multiple clusters



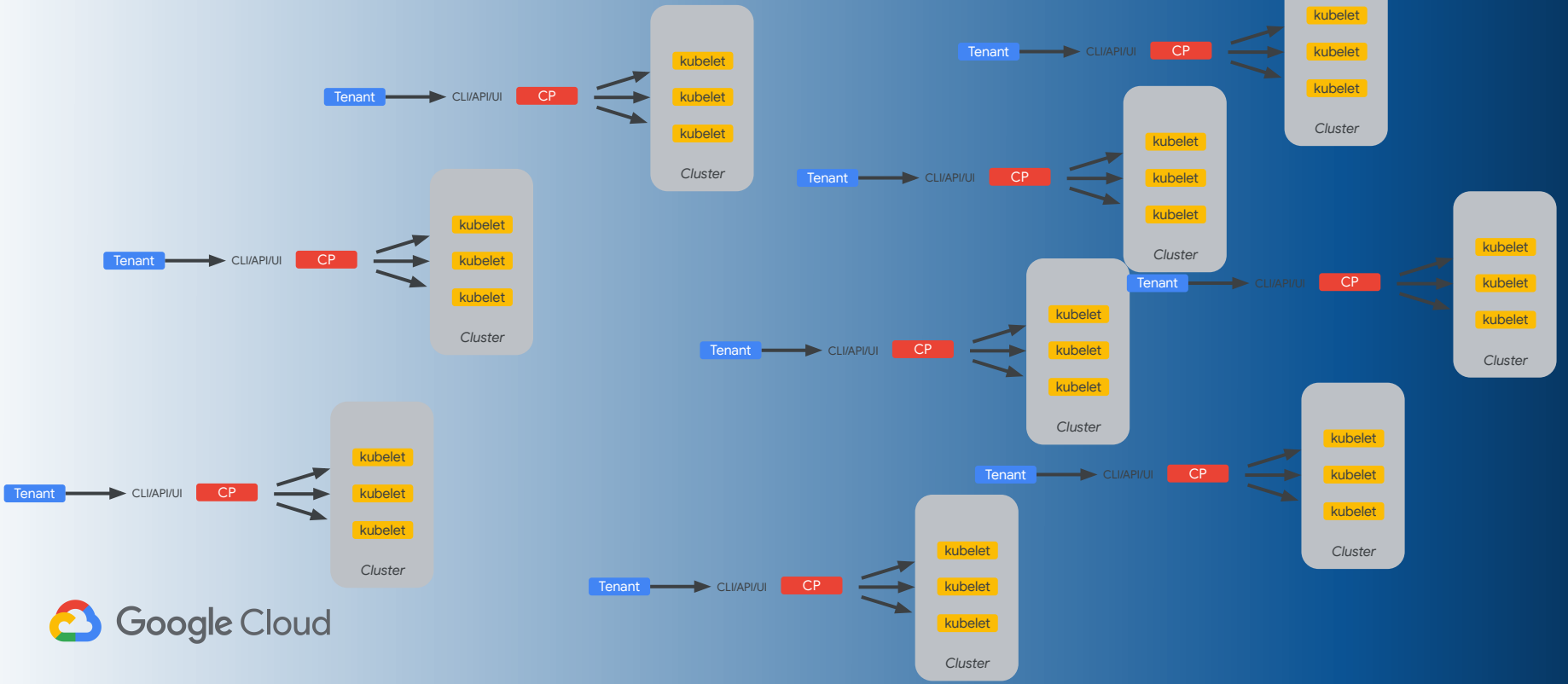
Google Cloud

Kubesprawl: how does this scale operationally?



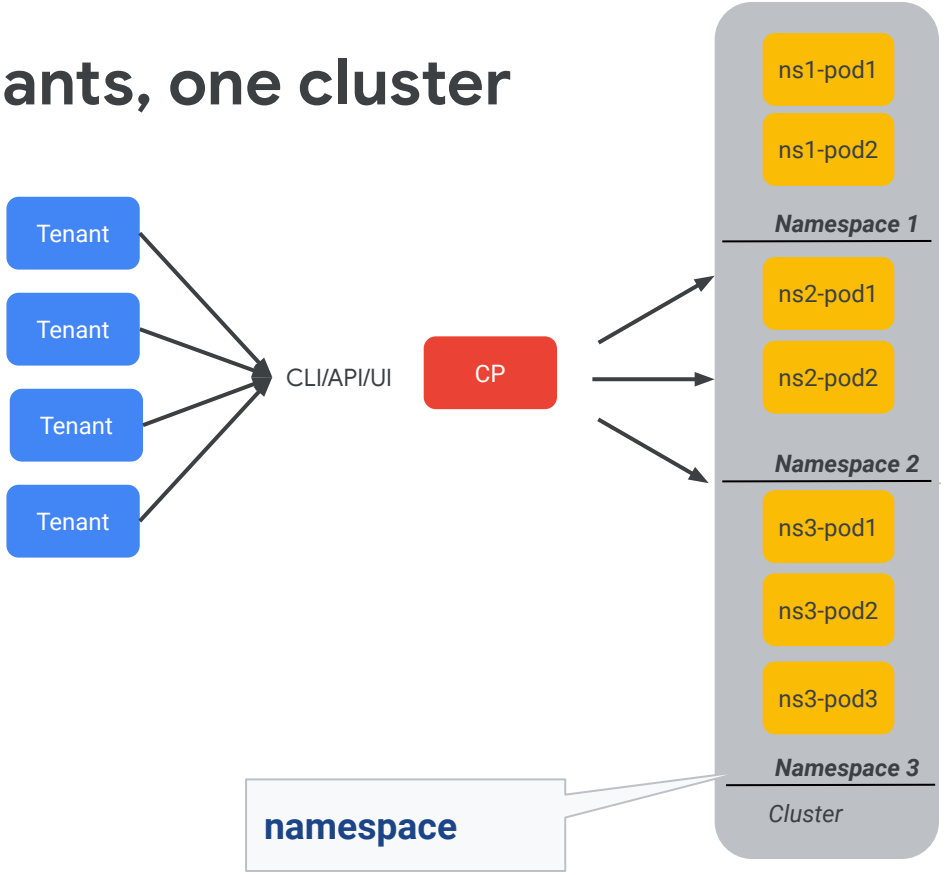
Google Cloud

Kubesprawl: how does this scale financially?



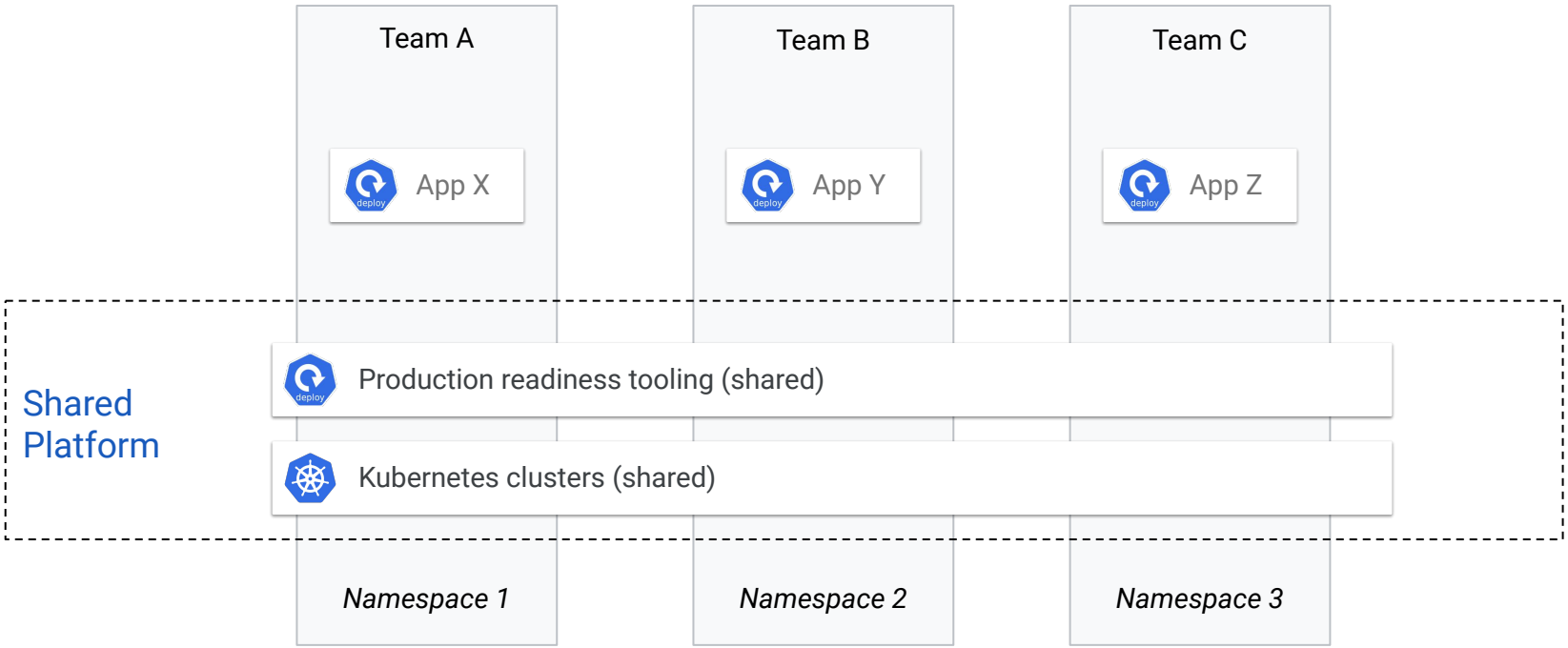
Google Cloud

Alternative: many tenants, one cluster



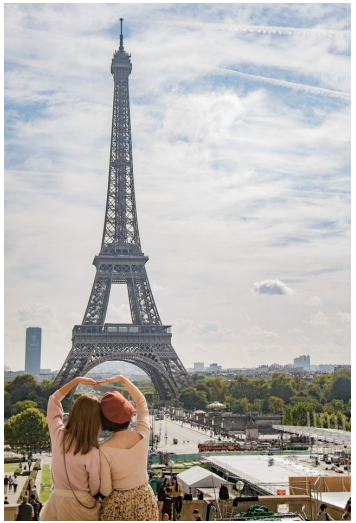
Google Cloud

Multitenancy reduces overhead and cost



When *should* you use multiple clusters?

Regionalization



Blast radius

Scalability

Cluster snowflakes

… mainly CRDs, webhooks, or cluster-scoped operators

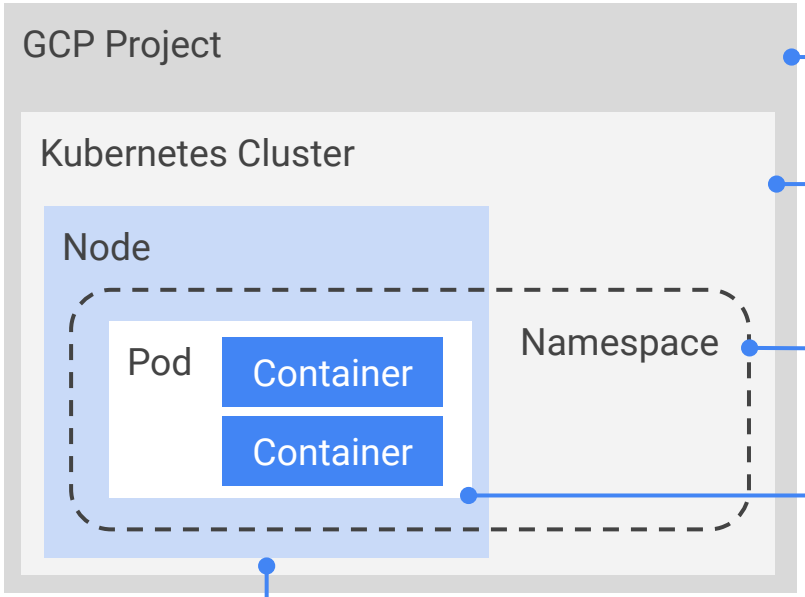
Google Cloud

See also: "The myth of the monocluster"

What are the principles of multitenancy?

Google Cloud

Overview of isolation in Kubernetes



Project: top level isolation

Cluster: strongest isolation within the same project

Namespace: allows control plane isolation within the same cluster (no workload isolation)

Pod: localhost “isolation”

Node: good resource (not security!) isolation

Google Cloud

What does “multitenancy in Kubernetes” mean?

Provide isolation and
fair resource sharing between
multiple users and their workloads
within a single cluster

Best Practices

1. Understand your needs. Not everyone needs everything all at once! Think about cost, overhead, and risk tolerance.
2. Understand the pros/cons of various approaches and technologies to solve your most critical problems.
3. Deploy your solutions, and keep them up-to-date, and iterate to achieve more benefits.

How do I implement multitenancy?

Google Cloud

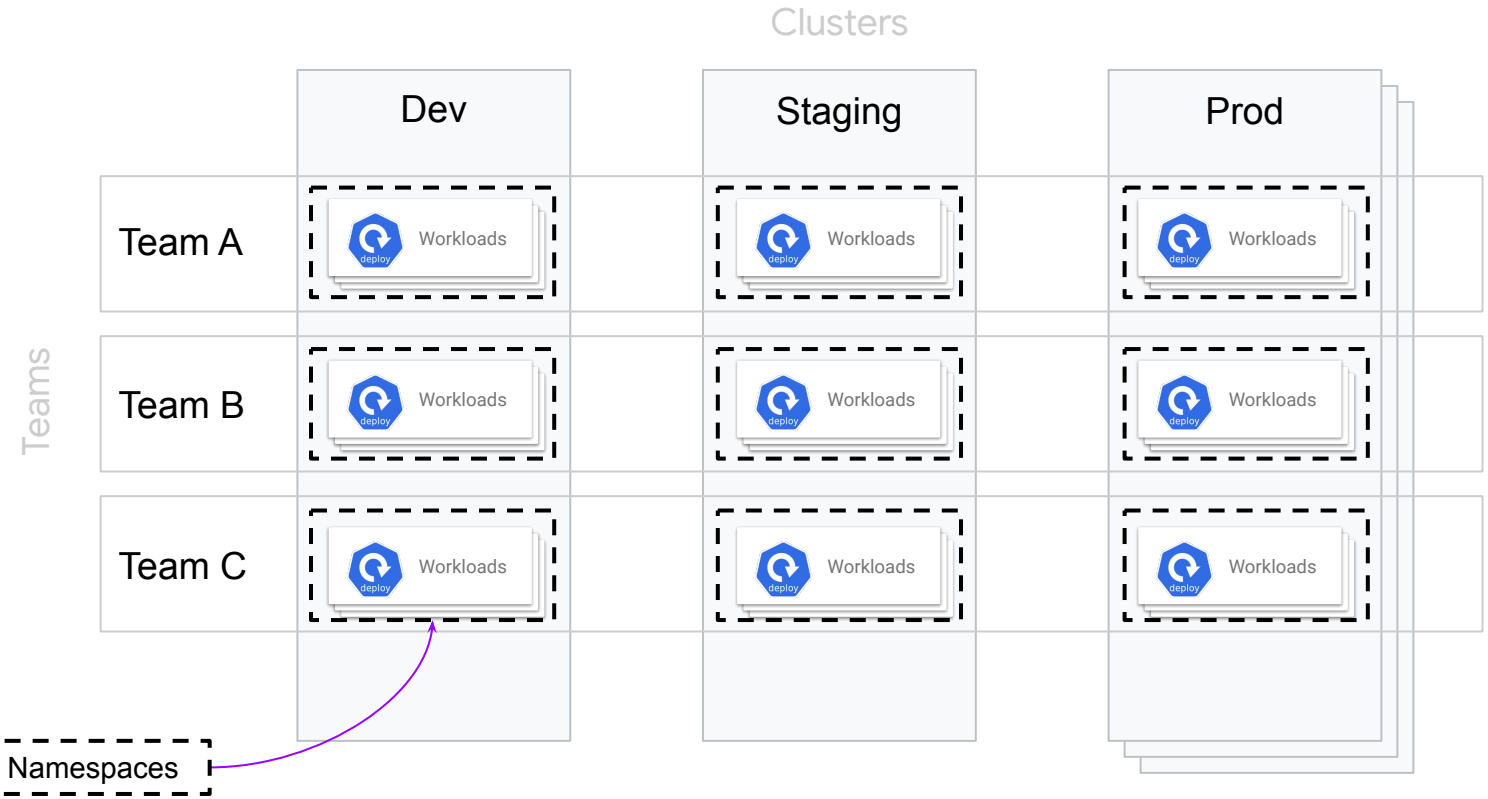
Namespaces

Namespaces are the primary unit of tenancy in Kubernetes.

By themselves, they don't do much except organize other objects - but almost all policies support namespaces by default.

Google Cloud

Multitenancy across clusters



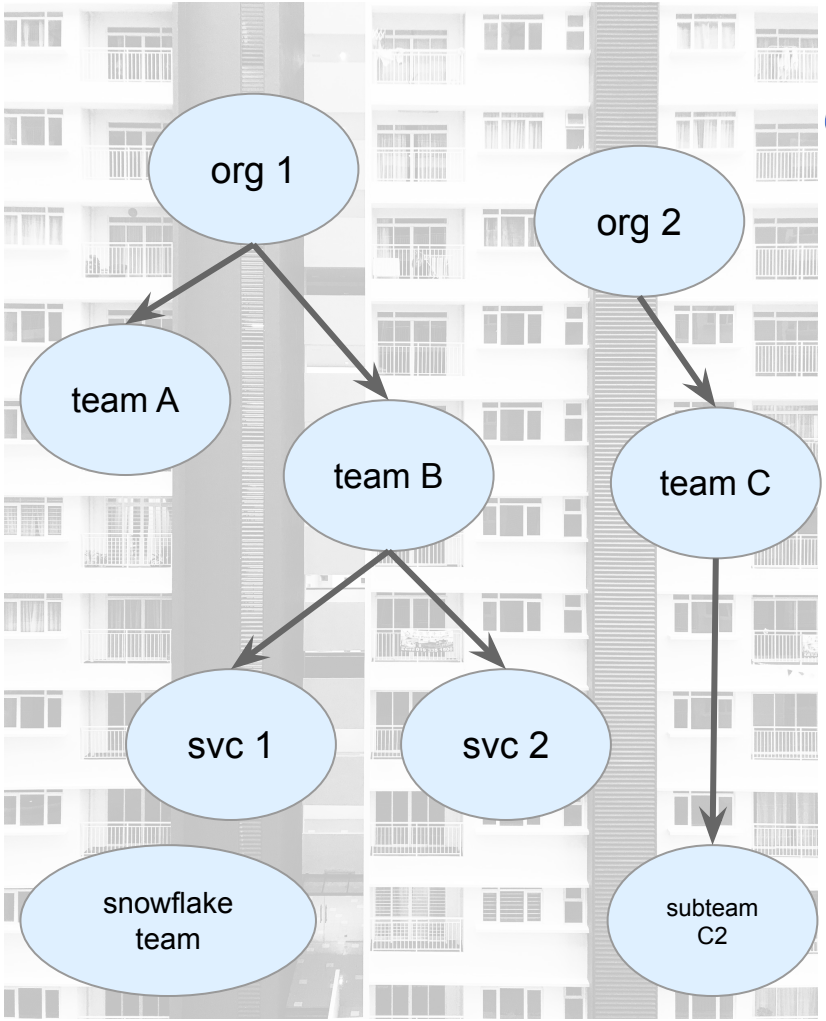
Properties of namespaces

- Require cluster-level permissions to create
- Fully independent policies
- Must be labeled manually to use in policy application (e.g. in Network Policies)
- Included in Kubernetes natively

Hierarchical namespaces

Traditional Kubernetes namespaces are flat, with no relation between them. Hierarchical namespaces express *ownership,* allow for admin delegation and cascading policies.

Hierarchical Namespaces are provided by the [Hierarchical Namespace Controller (HNC)](), a project of wg-multitenancy.



Google Cloud

Properties of hierarchical namespaces

- Can use namespace-level “subnamespace” permission to create
- Inherit policies from ancestors
- Can be selected individually or as subtrees
- Provided via the OSS Hierarchical Namespace Controller (HNC), or as a part of GKE’s Config Sync (available later in June 2020).

Applying multitenancy

1. Access Control
2. Resource Sharing
3. Runtime Isolation
4. Insights

Applying multitenancy

Access control

Tenancy is about *ownership*, and ownership is about control. Job #1 is ensuring that tenants can't control each others' resources.

Google Cloud

Authentication and authorization

Control Plane (apiserver)

Resources

Pluggable Auth
(GKE IAM)

RBAC

Cloud IAM Policies

{Cluster}Role
{Cluster}RoleBinding

Google Cloud

Role-Based Access Control (RBAC)

RBAC controls access to namespaces in Kubernetes. They're used for:

- Giving **humans** access to Kubernetes resources
  - On GKE, can use Google Groups to give groups of people identical access
- Giving **non-Kubernetes service accounts** access to the Kubernetes API
  - Example: GCP Service Accounts; also work with Google Groups
- Giving access to **pods** calling Kubernetes APIs (with Kubernetes Service Accounts)

Key RBAC concepts:

ClusterRole	A set of cluster-wide permissions. Some useful defaults are preset (e.g. "admin")
Role	Like a ClusterRole, but limited to a single namespace
ClusterRoleBinding	Give a role to one or more *subjects* (humans, SAs, etc) across the whole cluster.
RoleBinding	Give a role *within* a single namespace. You can also use ClusterRoles (e.g. "admin") to grant predefined permissions, but limited to that one namespace

GKE Workload Identity

Let Google **manage and rotate the credentials** that are used by your Kubernetes workloads to access GCP services.

- Workloads attaching a Kubernetes SA automatically authenticate as a separate GCP SA when accessing GCP API
- Allows the Oauth scopes and service accounts attached to node pools to follow least privilege

Replaces these workarounds:

- Using node (VM) identity for the pod
- Removes the need for exporting user managed service account keys and embedding them in kubernetes native secrets

Applying multitenancy

Google Cloud

Resource sharing

Many elements of the cluster are still shared, especially nodes and the apiserver itself. Make sure your tenants can share them fairly.

Google Cloud

Resource Sharing

- **Resource Quotas:** no one *namespace* can exceed resource usage. Also used to control dangerous objects (ingress, external services).
- **Limit Range:** no one *pod* can exceed resource usage
- **Pod Affinity/Anti-affinity:** keep pods scheduled together/apart
- **Pod Priority:** pick a winner when there isn't enough to go around

Applying multitenancy

Access Control

Resource Sharing

Runtime Isolation

Insights

1 2 3 4

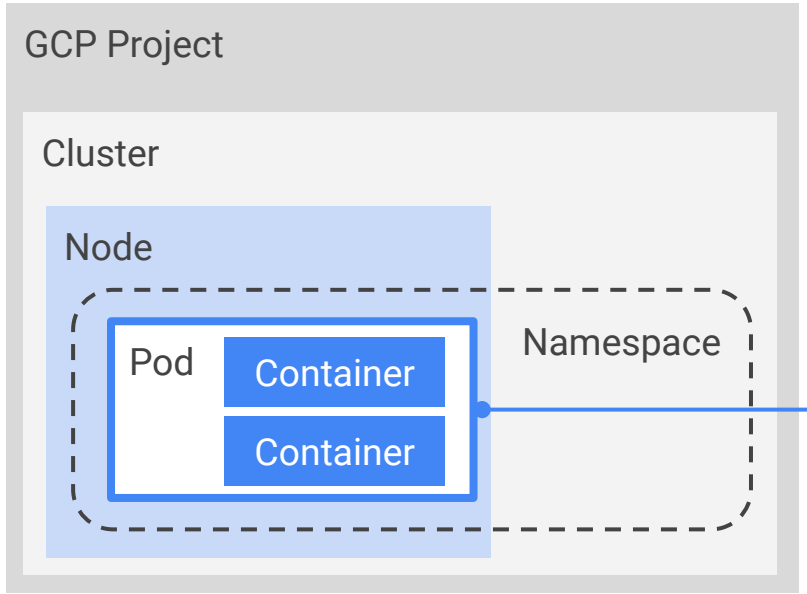
Google Cloud

Runtime isolation

Vulnerabilities and attacks are a reality, and containers aren’t a security boundary. Consider adding runtime isolation to stop anything getting out of your containers.

Google Cloud

Runtime boundaries in Kubernetes



Pod: a unit of scheduling and deploying workloads (no secure isolation by default)

Google Cloud

Runtime isolation: overview

- **Pod Security Context:** restrict a pod's workload (eg non-root).
- **Pod Security Policy:** enforce that pods *must* declare an appropriate context. **Hard to enable on a working cluster;** consider alternatives like OPA Gatekeeper or Anthos Policy Controller.
- **Network Policy:** forbid pods from talking to each other if they have no good reason to.
- **Runtime Class:** run pods in sandboxes like gVisor or Kata Containers.

Runtime isolation: GKE Sandbox

GKE's built-in Runtime Class protects your pods.

Adds a security boundary to containers in GKE based on **gVisor**.

Defense-in-depth security principles **without application changes**, new architecture models, or added complexity.

Google Cloud

Applying multitenancy

Access Control

Resource Sharing

Runtime Isolation

Insights

1 2 3 4

Google Cloud

Insights

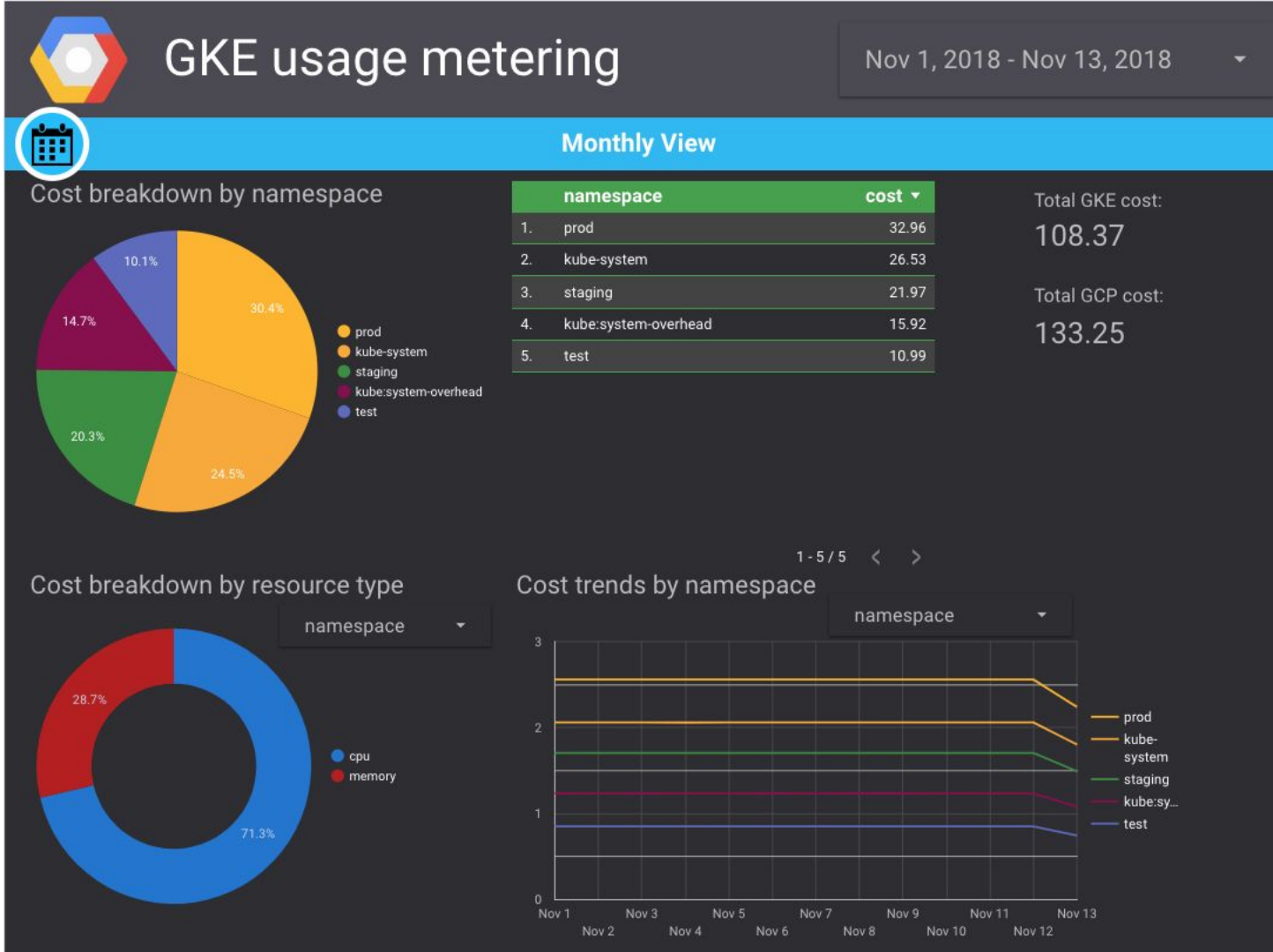
Tenants need to be able to observe themselves, and you need to be able to observe (and charge!) your tenants.

Google Cloud

GKE Usage Metering

- View workloads’ resource usage in BigQuery, broken down by namespace and labels
    - Memory, CPU, GPU, PD, network, etc.
- Join usage data with GCP Billing Export data to compute resource costs per tenant

GKE Usage Metering

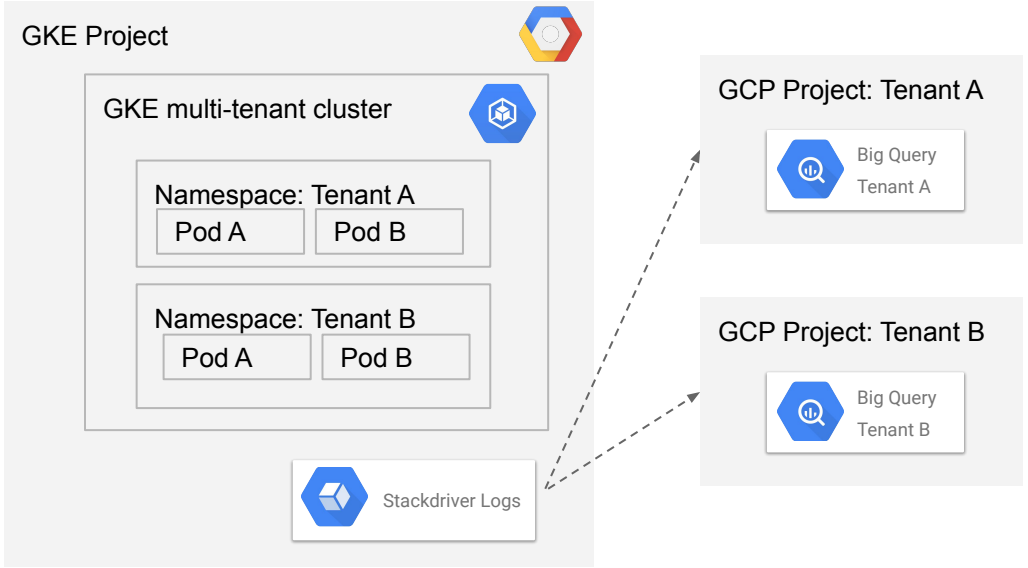


Google Cloud

Multitenant logging

Logs often refer to sensitive data, so consider controlling access to them carefully.

- Create a tenant project with a BigQuery dataset for each team.
- Use Log Routing to **filter logs by namespace** and send them to the correct BigQuery dataset.

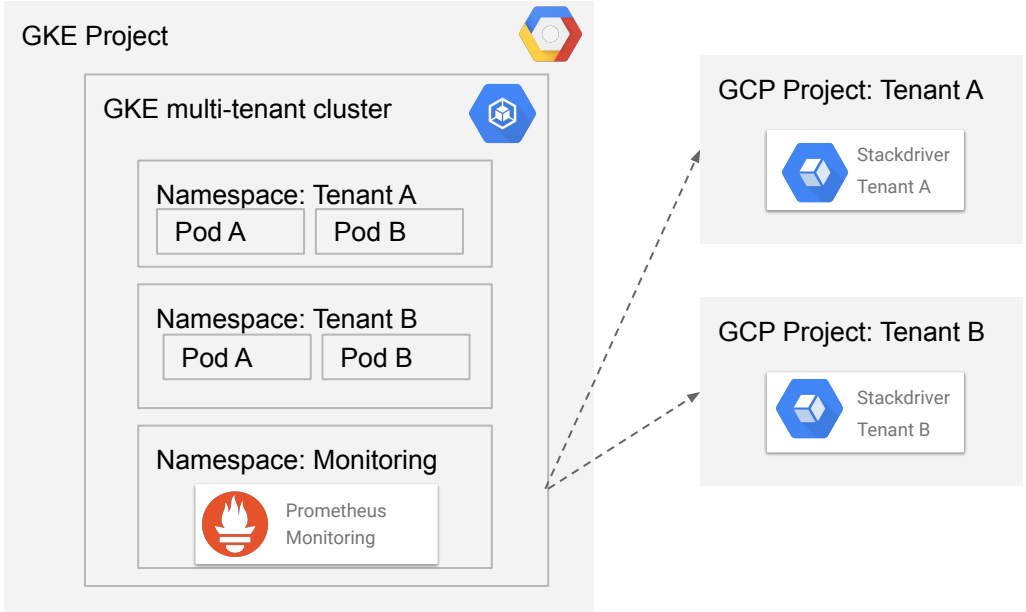


Google Cloud

Multitenant monitoring

Metrics are often less sensitive than logs, but you can still control access by tenant if you like.

- Create a tenant project per team.
- As part of the onboarding process, create a **Prometheus / Stackdriver adapter** with a **per namespace config**
- Send each tenant's metrics to the correct project.



Advanced topics

04

Google Cloud

Policy deployment

Be careful about storing your source-of-truth on your cluster.

- Check your policies (e.g. RBAC, Network Policy) into Git as YAML files.
- Have your cluster admin apply them based on Git, or use a CD tool like **GKE Config Sync** or **Anthos Config Management**.
- Test out changes on a canary cluster first, even in prod!
- Separate your policies from your workloads.
  - Possible exception: DaemonSets

Policy enforcement and auditing

Use OPA Gatekeeper (or Anthos Policy Controller) to define and apply custom policies.

- Define rules in Rego (Python-like rule language)
- Apply them across your clusters
- Audit violations
- Useful alternative to Pod Security Policies

Further exploration

Hard multitenancy is loosely defined as the condition where tenants are mutually hostile, not relatively co-operative.

- **Virtual Clusters:** an wg-multitenancy project to give each tenant its own control plane while sharing a data plane. Must be combined with sandboxing.
- **SaaS multitenancy:** many different instances of the same application for different **consumers**. Generally requires sandboxing and control plane automation.

Your needs will be very specific to your threat model!

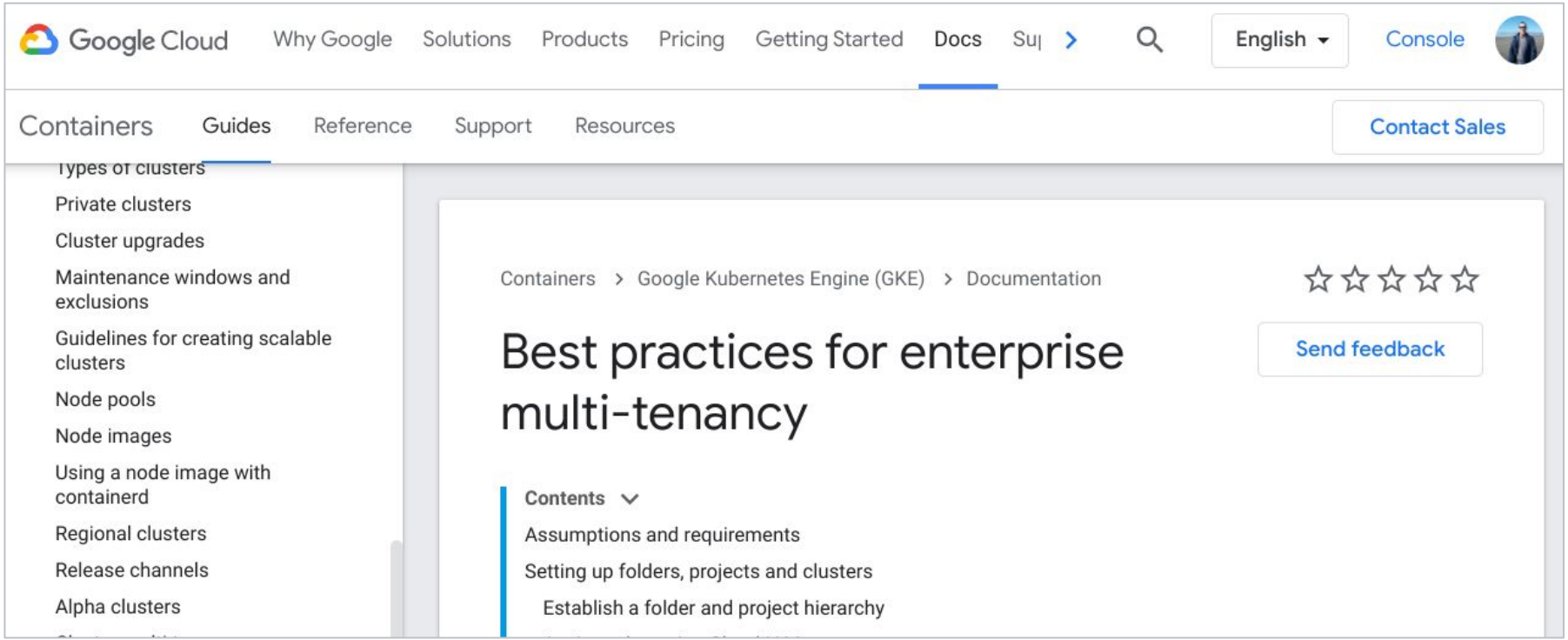
Conclusion

Google Cloud

Best Practices

1. Understand your needs. Not everyone needs everything all at once! Think about cost, overhead, and risk tolerance.
2. Understand the pros/cons of various approaches and technologies to solve your most critical problems.
3. Deploy your solutions, and keep them up-to-date, and iterate to achieve more benefits.

Learning more...



[Available from the GCP docs website]

Learning more...

Kubecon San Diego had some great presentations on building multitenancy systems on Kubernetes. My favourites include:

- Walls within walls: what if your attacker knows parkour?
- Kubernetes at Cruise: two years of multitenancy
- Plus two sessions from the multitenancy working group (wg-multitenancy)

Some other interesting links to follow include:

- The Multitenancy Working Group
- Mercari's experience with multitenant Istio

Multitenancy reduces overhead and cost