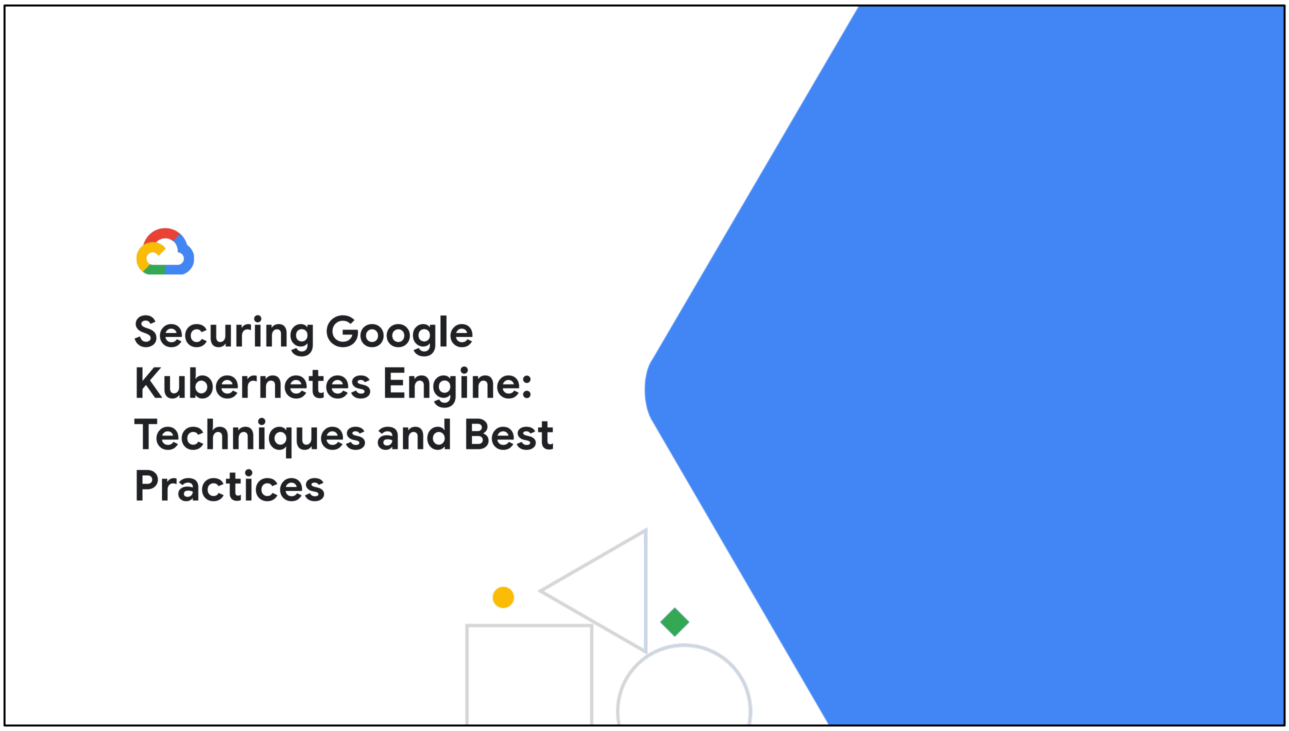
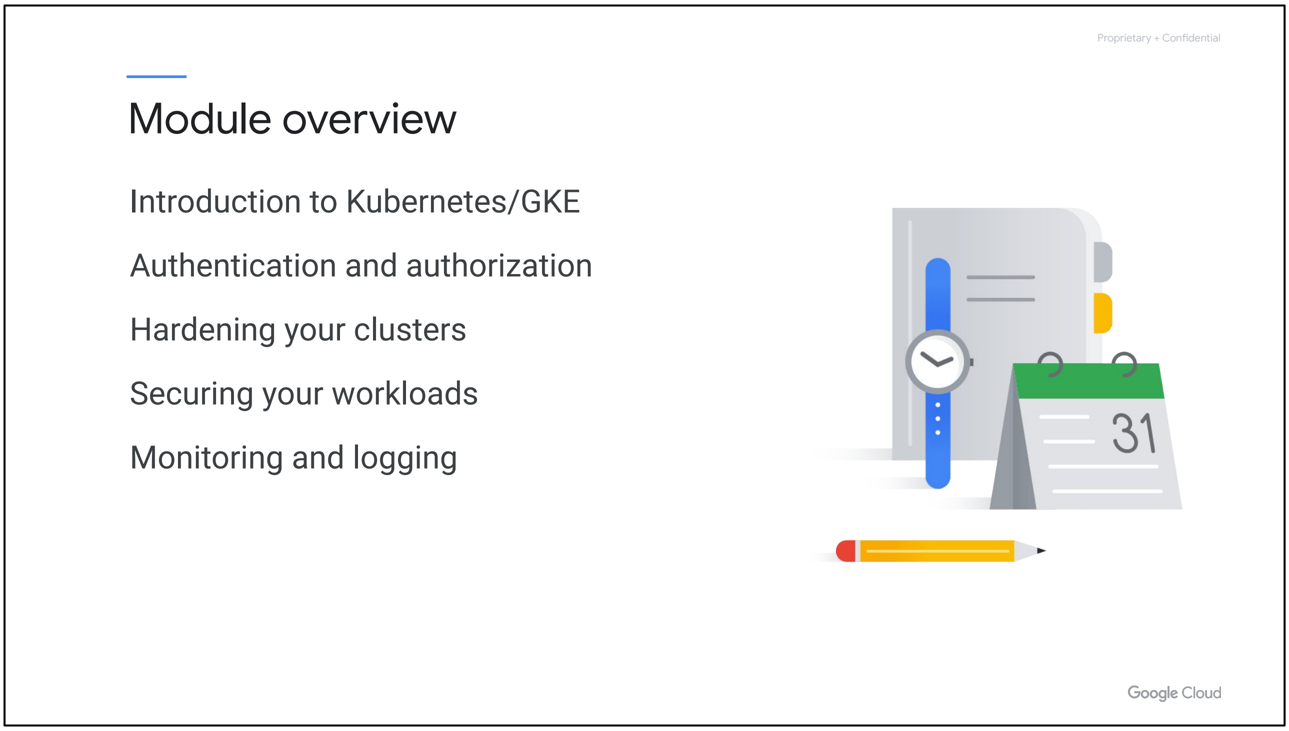


Securing Google Kubernetes Engine: Techniques and Best Practices

Welcome to the Securing Google Kubernetes Engine Techniques and Best Practices module.



Module overview

Introduction to Kubernetes/GKE

Authentication and authorization

Hardening your clusters

Securing your workloads

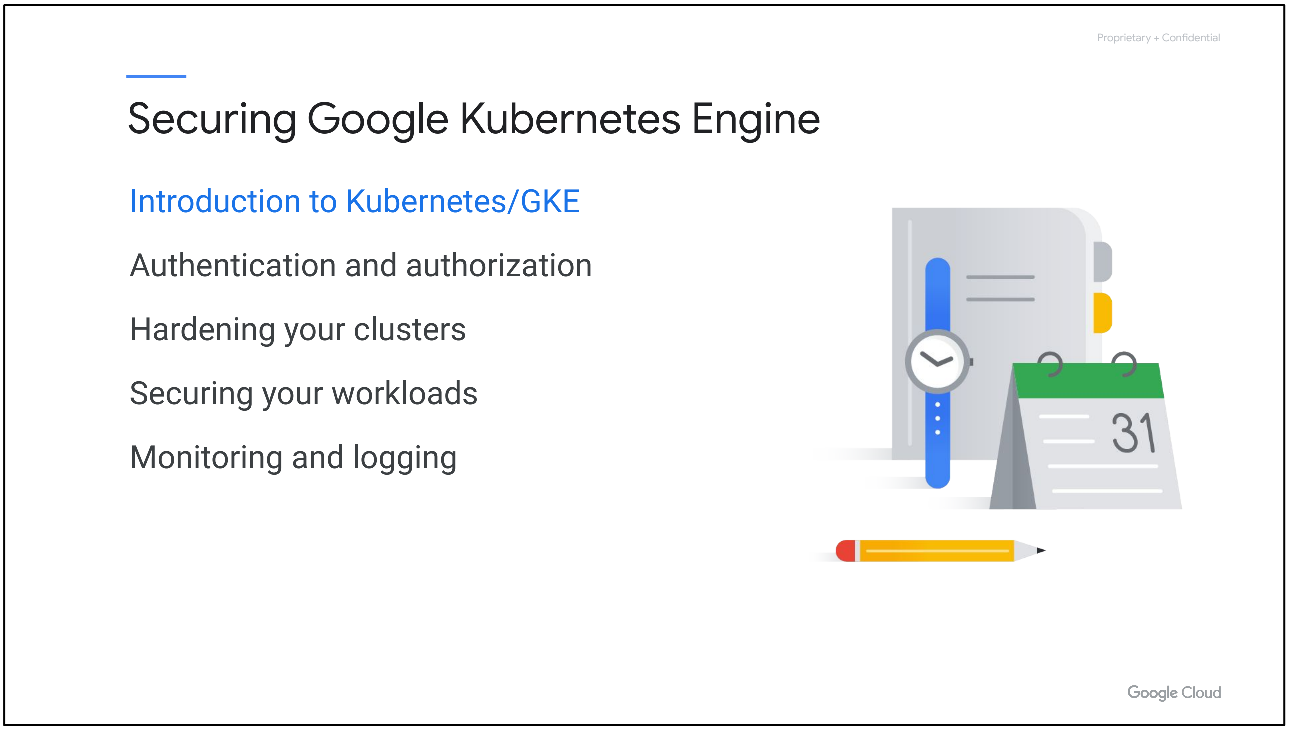
Monitoring and logging

Protecting workloads in Google Kubernetes Engine involves many layers of the stack, including the contents of your container image, the container runtime, the cluster network, and access to the cluster API server.

In this module, we will begin with an overview of Kubernetes - in particular, of Google Kubernetes Engine, or GKE.

Next, we will discuss how authentication and authorization work in Google Kubernetes Engine.

Then, we will talk about hardening your clusters, securing your workloads, and how to use logging and monitoring to make sure everything remains in good health.



Securing Google Kubernetes Engine

Introduction to Kubernetes/GKE

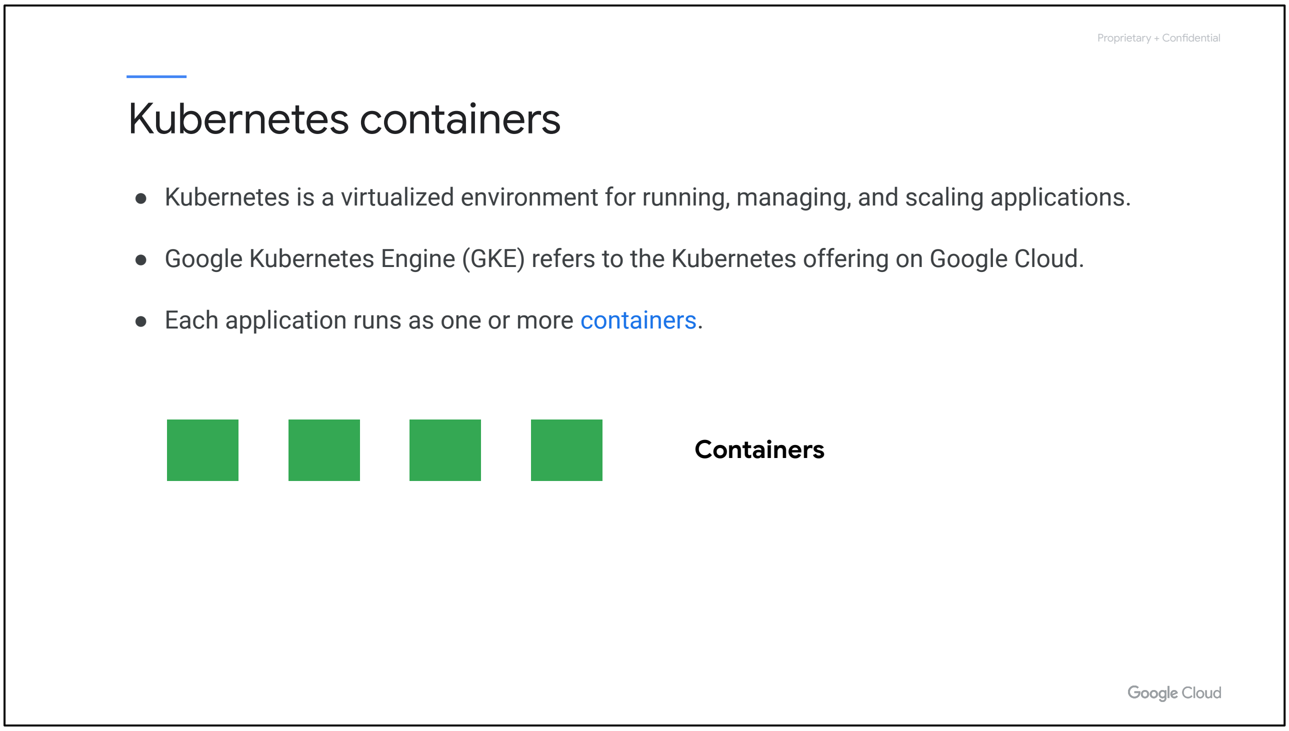
Authentication and authorization

Hardening your clusters

Securing your workloads

Monitoring and logging

In the next few slides, you will be given a brief overview of Kubernetes and its main architectural components. These slides serve only to provide a context for how to set up and manage security for Kubernetes.


Proprietary + Confidential

Kubernetes containers

- Kubernetes is a virtualized environment for running, managing, and scaling applications.
- Google Kubernetes Engine (GKE) refers to the Kubernetes offering on Google Cloud.
- Each application runs as one or more containers.

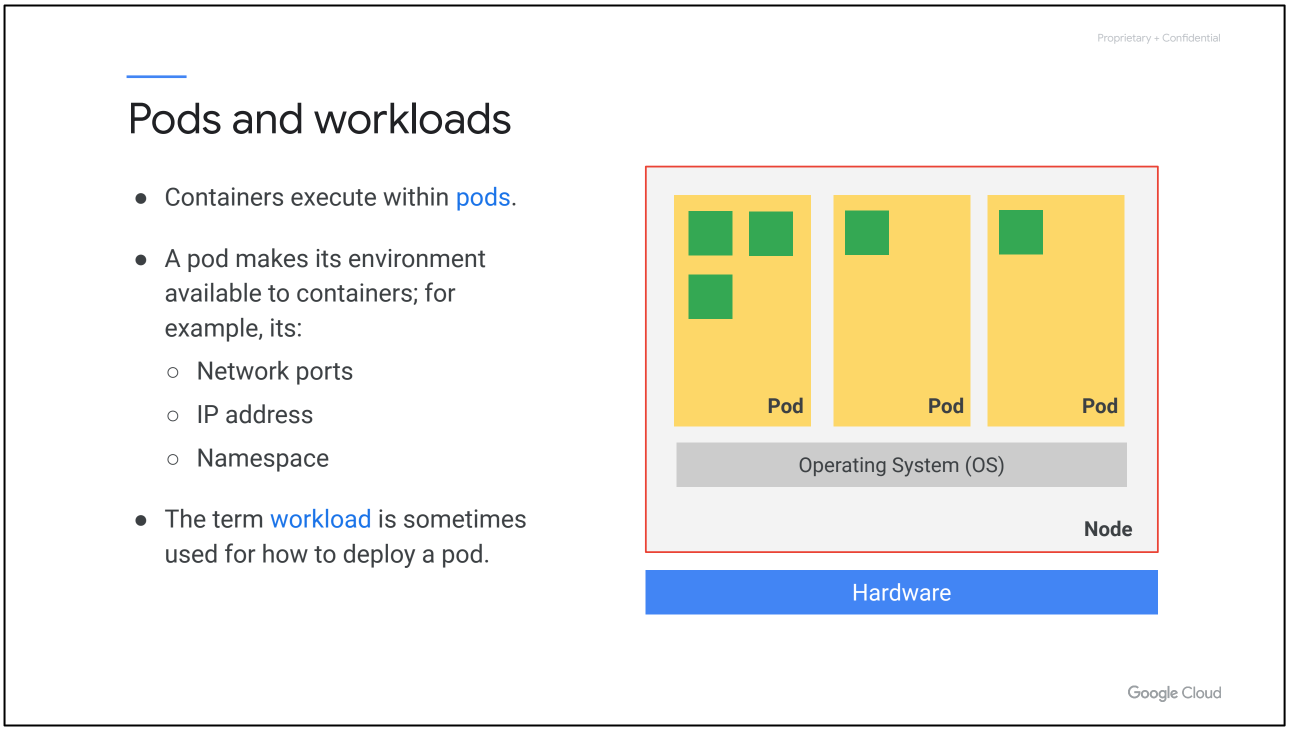
Containers

Google Cloud

Kubernetes is a virtualized environment in which you run, manage, and scale applications. Each application runs in a container. Google Kubernetes Engine (GKE) refers to the Kubernetes offering on Google Cloud. Kubernetes is also available from other vendors.

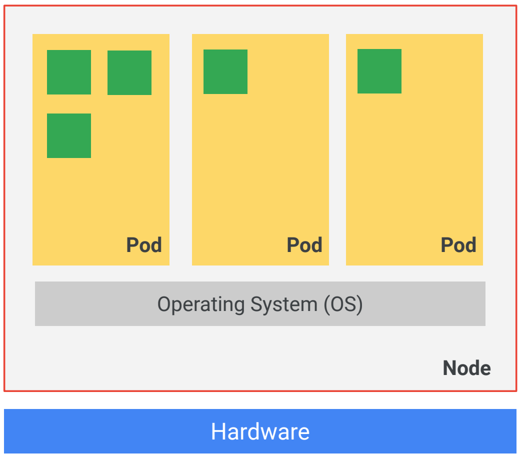
Each of your applications runs as one or more containers. A container is a lightweight, isolated user space for running application code. Containers are lightweight because they do not contain a full copy of the operating system. Instead, they contain a very scaled-down operating system with just the bare minimum needed to run a container.

Containers can be packed tightly onto the underlying system, which is very efficient. Starting and stopping a container means starting and stopping its scaled down operating system processes, not booting an entire virtual machine and initializing an operating system. Similar to a virtual machine, a container has its own filesystem, CPU, memory, process space, and more. Because they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.



Pods and workloads

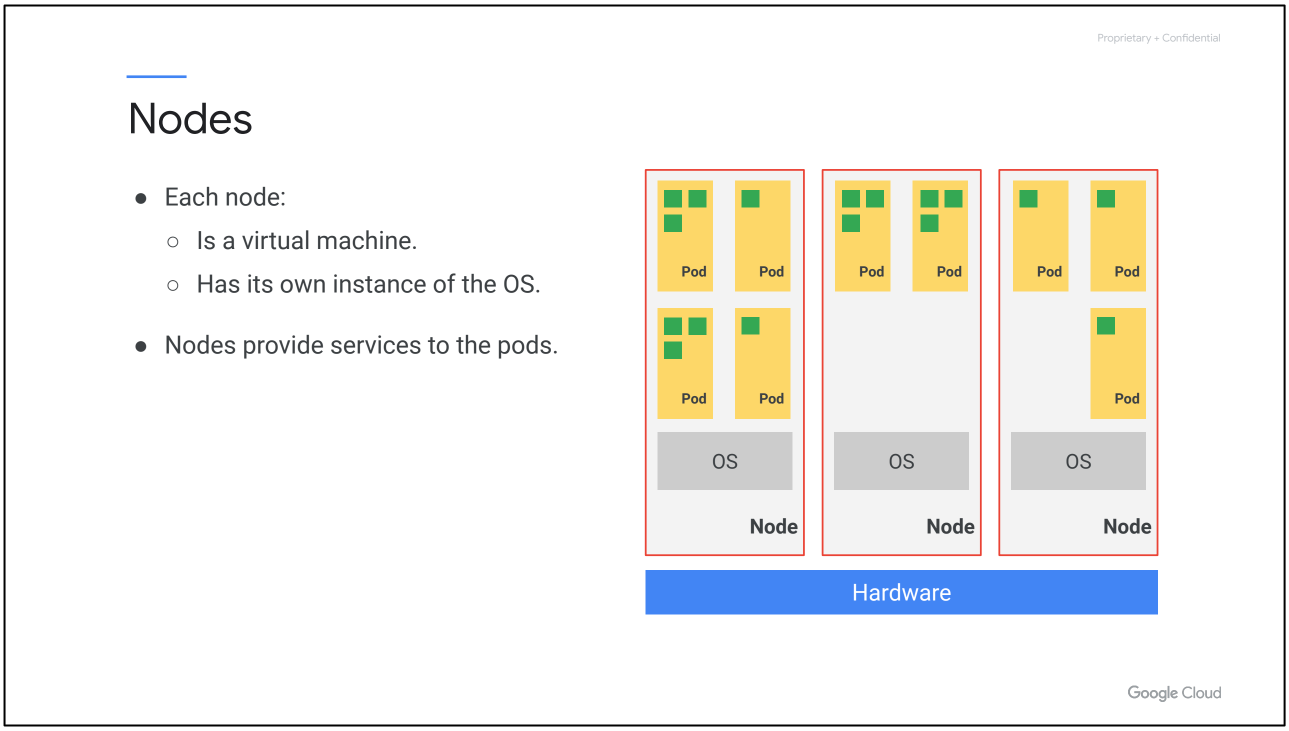
- Containers execute within pods.
- A pod makes its environment available to containers; for example, its:
    - Network ports
    - IP address
    - Namespace
- The term workload is sometimes used for how to deploy a pod.



Containers execute within a pod. The most common use case is for a pod to execute a single container. However, it is possible for a pod to run multiple containers. This is useful when containers are tightly coupled or share resources.

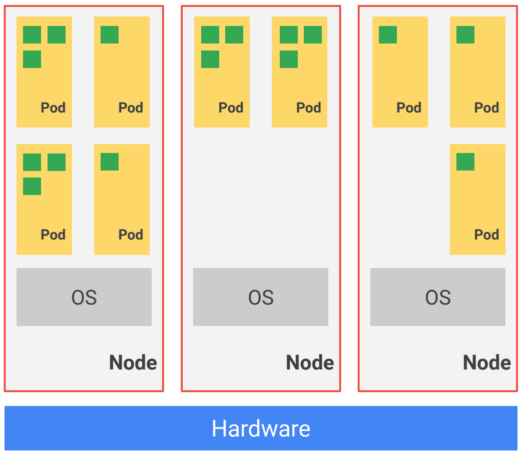
A pod makes its environment available to the containers running within it. For example, containers use the pod's network namespace, including its IP address and ports. Containers within a pod can communicate with each other using localhost (127.0.0.1). A pod can define storage volumes, which its containers can use.

Workload is a general term that refers to applications, microservices, daemons, or jobs: the sort of things that are implemented by pods.



Nodes

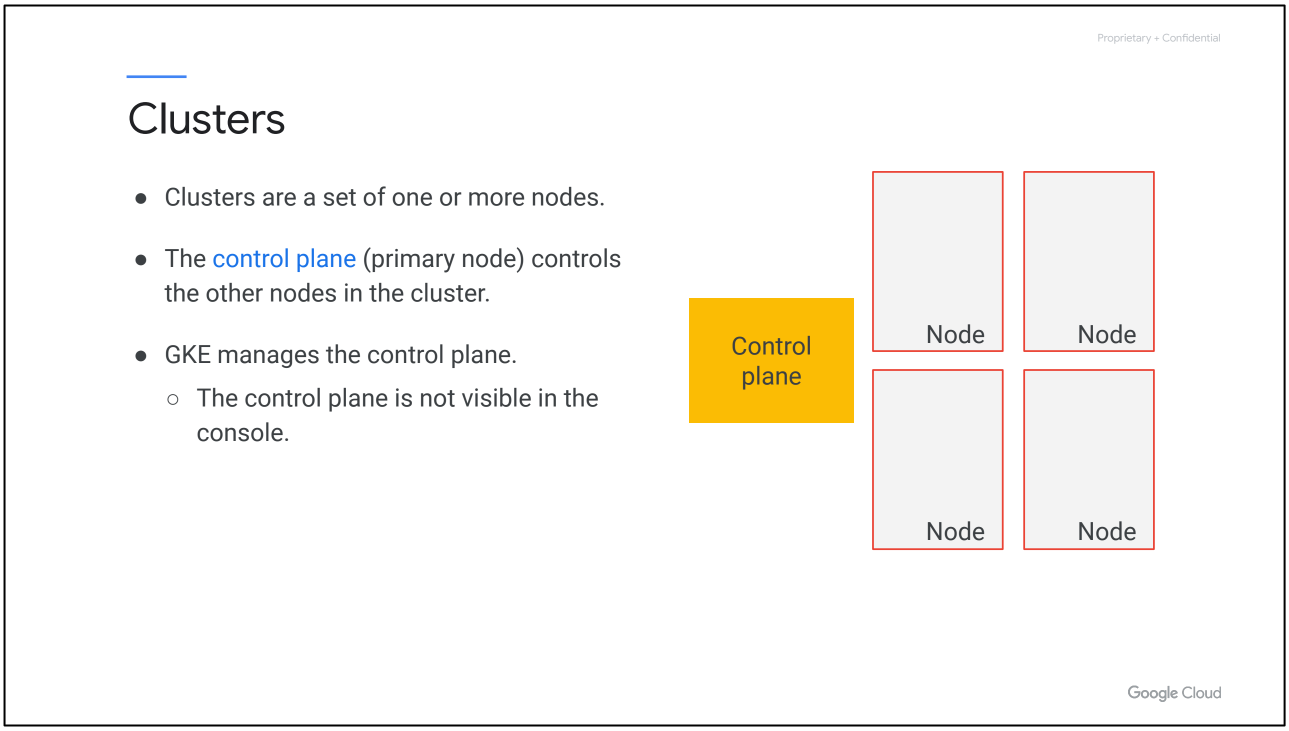
- Each node:
  - Is a virtual machine.
  - Has its own instance of the OS.
- Nodes provide services to the pods.



Nodes are the worker machines on which the pods exist. In GKE, each node is a separate, virtual machine.

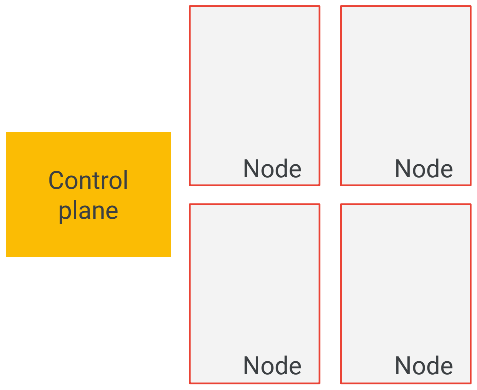
Each node—in other words, each virtual machine—has its own instance of the operating system.

Nodes provide services to the pods, such as hardware and the network infrastructure. The pods share these resources and then share them among the containers.



Clusters

- Clusters are a set of one or more nodes.
- The control plane (primary node) controls the other nodes in the cluster.
- GKE manages the control plane.
  - The control plane is not visible in the console.

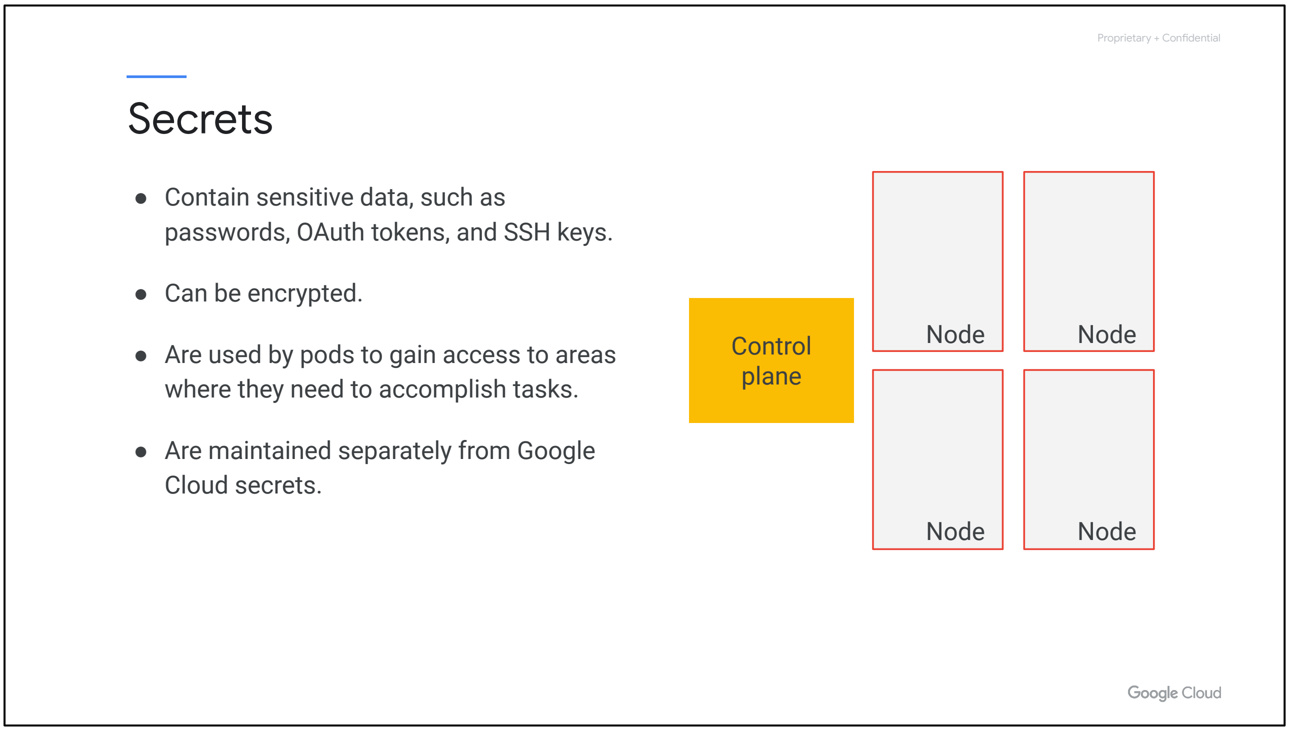


Clusters are a set of one or more nodes. The control plane, which is the primary node, controls the other nodes in the cluster. The cluster is visible in the Google Cloud console.

GKE manages the control plane internally and hides it from view; in the console, you will not see a “control plane” within the cluster.

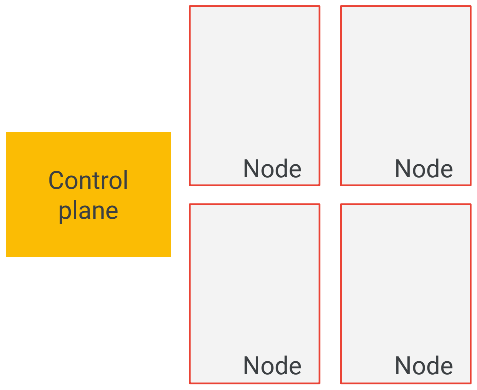
Kubernetes scales nodes in the cluster up or down as needed based on demand and on the Kubernetes configuration.

For more information, see the Architecting with Google Kubernetes Engine course or the Kubernetes documentation linked in the speaker notes.
- **Link:** kubernetes.io/docs/tutorials/kubernetes-basics/scale/scale-intro/



Secrets

- Contain sensitive data, such as passwords, OAuth tokens, and SSH keys.
- Can be encrypted.
- Are used by pods to gain access to areas where they need to accomplish tasks.
- Are maintained separately from Google Cloud secrets.

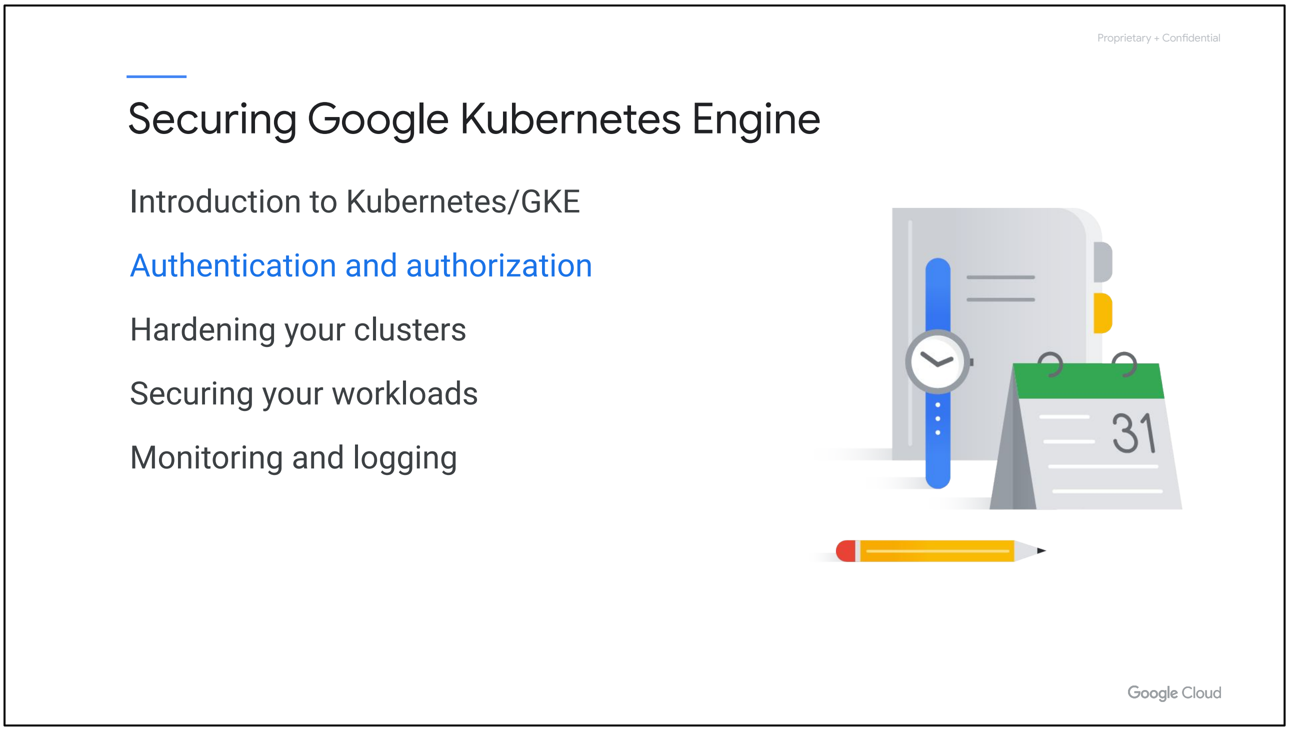


Kubernetes secrets contain sensitive data, such as passwords, OAuth tokens, and SSH keys.

Secrets can be encrypted. By default, GKE encrypts customer content stored at rest, including secrets. GKE handles and manages this default encryption for you without any additional action on your part. Pods use secrets to gain the access they need within your environment to accomplish tasks. GKE secrets are not the same as secrets managed within Secret Manager. You will learn more about this later.

GKE secrets are similar to Google Cloud secrets but they are not the same: they are separate entities. In GKE, a secret is created using the Kubernetes API and is available only within GKE. It is not accessible within other areas of Google Cloud. By default, secrets created within Google Cloud are not available within Kubernetes.

Later, you will learn that you can use Workload Identity to configure a Kubernetes service account to act as a Google service account. For more information about GKE secrets, see the GKE documentation.



Securing Google Kubernetes Engine

Introduction to Kubernetes/GKE

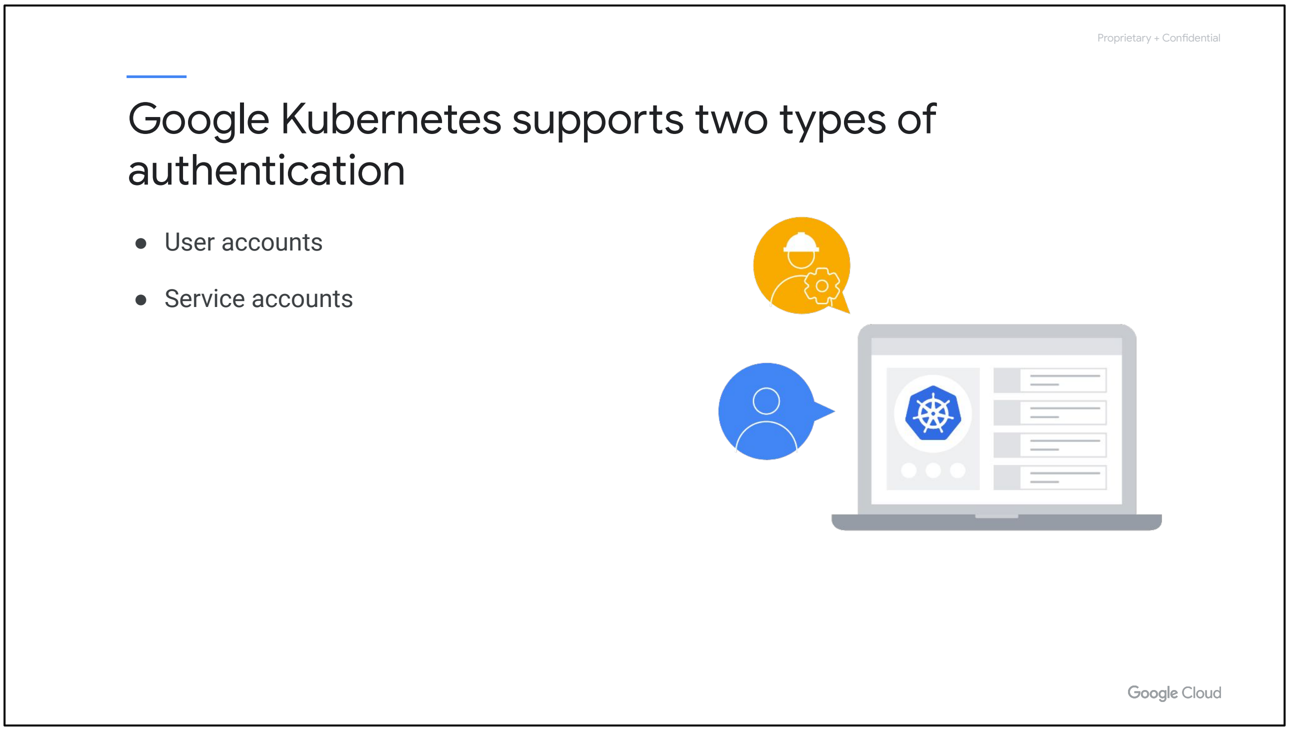
Authentication and authorization

Hardening your clusters

Securing your workloads

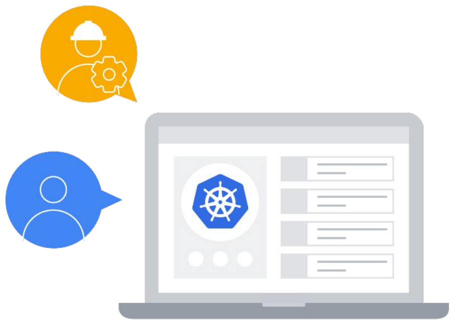
Monitoring and logging

Protecting workloads in Google Kubernetes Engine involves many layers of the stack, including the contents of your container image, the container runtime, the cluster network, and access to the cluster API server.



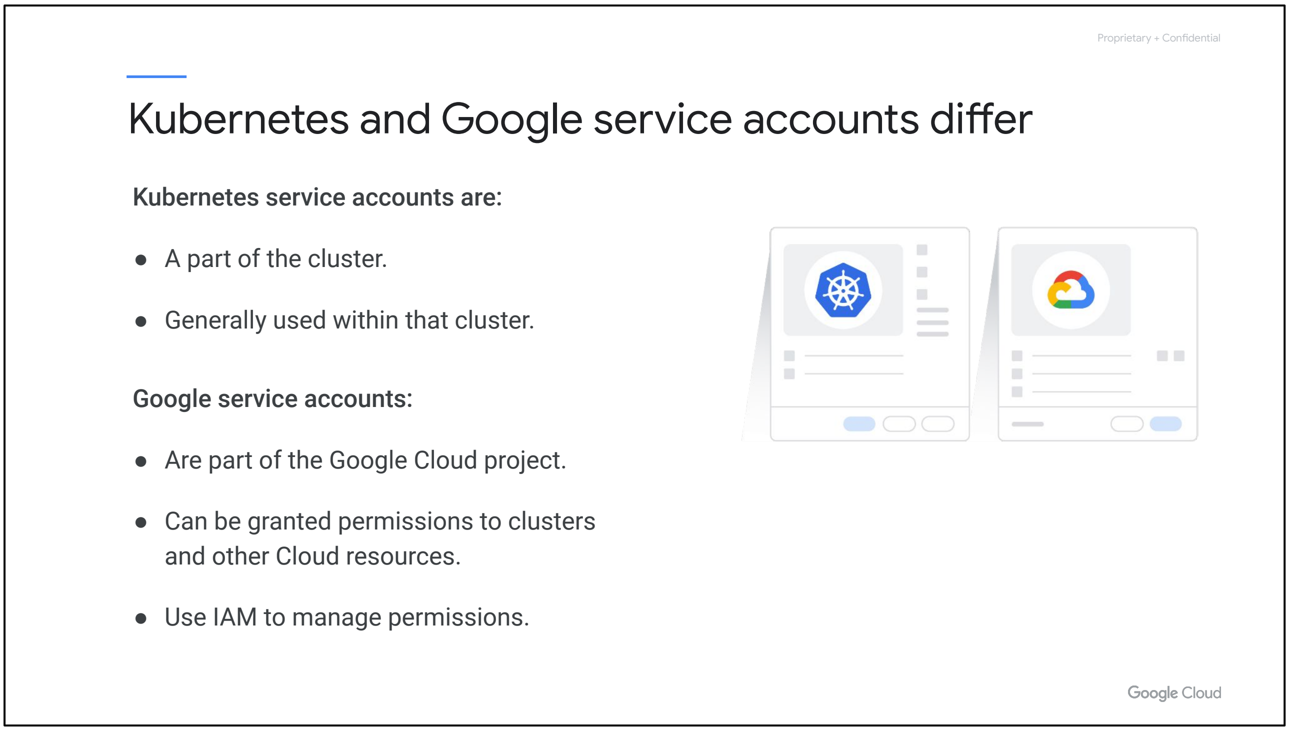
Google Kubernetes supports two types of authentication

- User accounts
- Service accounts



User accounts are accounts that are *known* to Kubernetes, but are not *managed* by Kubernetes; for example, you cannot create or delete them using **kubectl**. Kubernetes calls these types of accounts “normal users” and leaves their administration to an outside, independent service, like Google Cloud. Google Kubernetes User (“normal users”) accounts also come in two types: Google account, and Google Cloud service account. Once created, both types of accounts need to be authorized to create, read, update, or delete Kubernetes resources.

Service accounts are accounts that are created and managed by Kubernetes, but can only be used by Kubernetes-created entities, such as pods. These accounts are managed through the Kubernetes API, are bound to specific namespaces, and can be created automatically by the API server or manually via API calls. If a request is received that is not tied to either a known user or a known service account, it is treated as an anonymous request.



Kubernetes and Google service accounts differ

Kubernetes service accounts are:

- A part of the cluster.
- Generally used within that cluster.

Google service accounts:

- Are part of the Google Cloud project.
- Can be granted permissions to clusters and other Cloud resources.
- Use IAM to manage permissions.

Despite having similar names, Kubernetes service accounts and Google service accounts are different types of accounts.

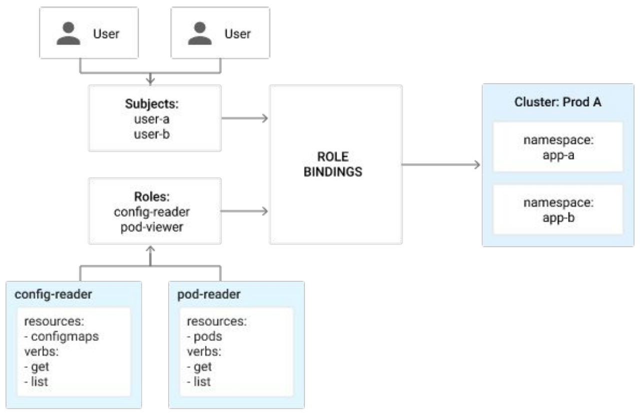
Kubernetes service accounts are part of the cluster in which they are defined and are typically used only within that cluster.

In contrast, Google Cloud service accounts are always a part of a Google Cloud project.

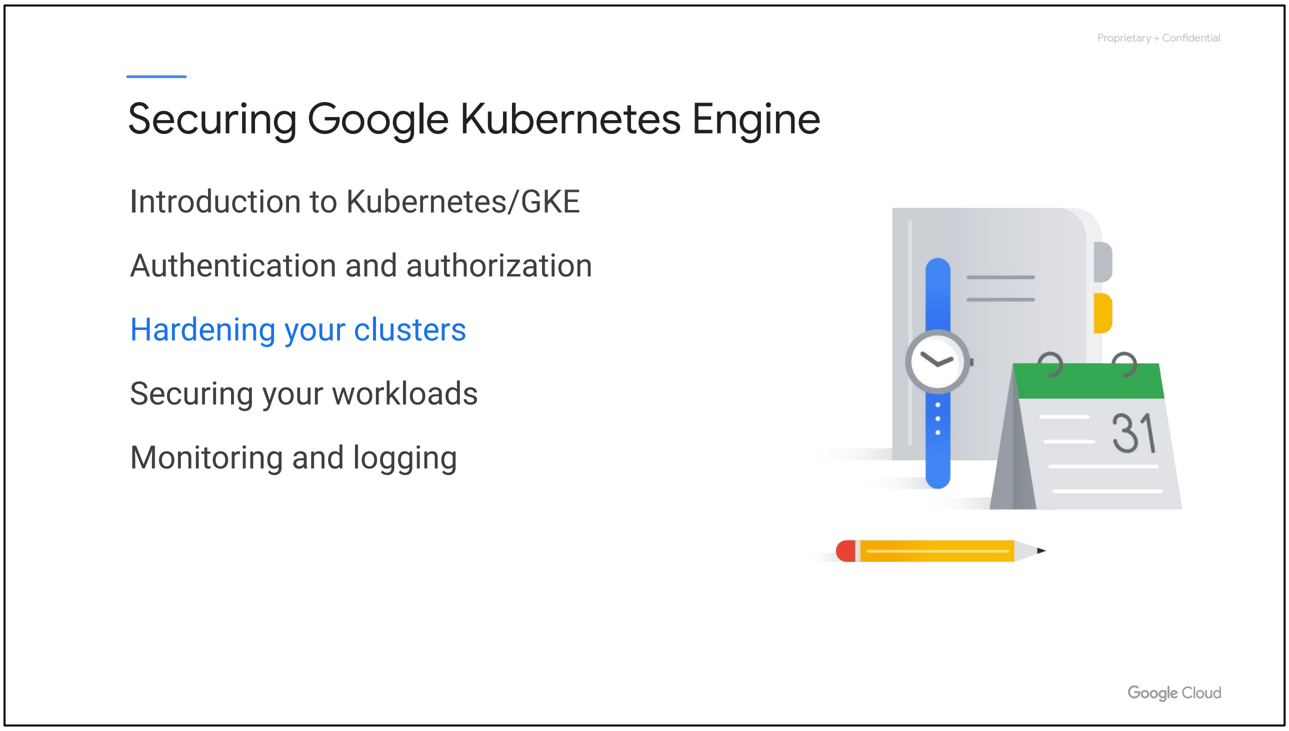
Google service accounts can easily be granted permissions by IAM both within clusters and Google Cloud projects, as well as to any other Google Cloud resource. This means that Google Cloud service accounts are more powerful than Kubernetes service accounts.

Therefore, in order to follow the security principle of least privilege, you should consider using Google Cloud service accounts only when their extra capabilities are needed, and use IAM to manage their permissions.

Role-based access control (RBAC) gives your resources finer access granularity



Role-Based Access Control allows you to create detailed policies that define which operations and resources you allow users and service accounts to access. Role Based Access Control controls access to Google Accounts, Google Cloud service accounts, and Kubernetes service accounts.



Securing Google Kubernetes Engine

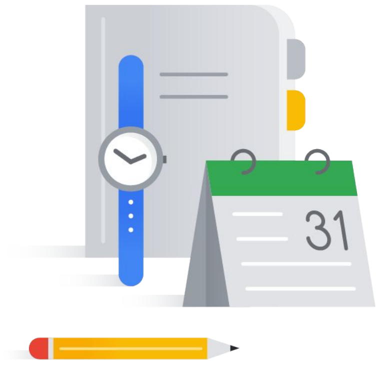
Introduction to Kubernetes/GKE

Authentication and authorization

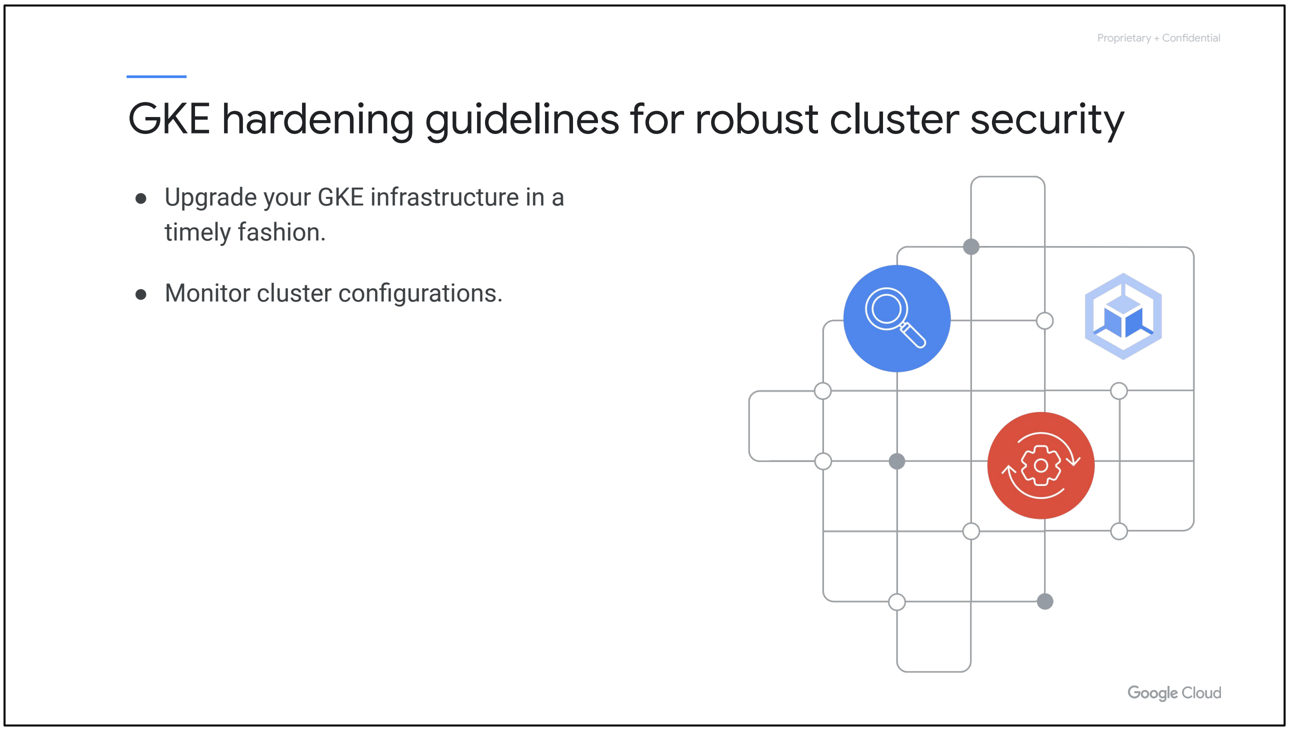
Hardening your clusters

Securing your workloads

Monitoring and logging



With the speed of development in Kubernetes, new features (including new security features) are released quite often. This module will guide you through recommended current guidelines for hardening your GKE clusters.



GKE hardening guidelines for robust cluster security

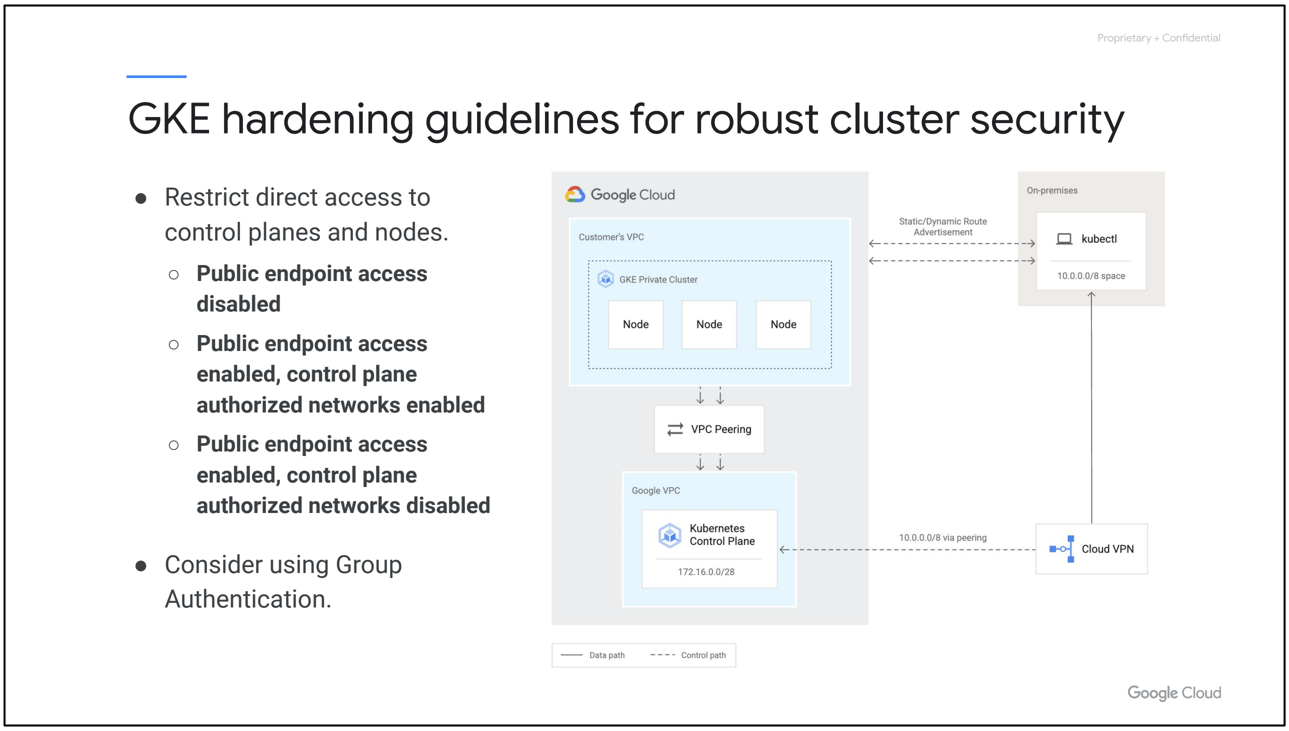
- Upgrade your GKE infrastructure in a timely fashion.
- Monitor cluster configurations.

First, keeping your version of Kubernetes up to date with recommended patches and updates is one of the most critical, yet the easiest, way to increase the security of your clusters. With Google Kubernetes Engine, the control planes are patched and upgraded for you automatically, and Node auto-upgrade also automatically upgrades nodes in your cluster. There are three types of automatic upgrades. From most frequent to least frequent updates, they are Rapid, Regular, and Stable.

Rapid provides the quickest access to new features, as soon as they are released. Regular provides updates on a more predictable cadence; it is the default update channel and is recommended for most users. Stable provides the least frequent updates, allowing more time for feature validation.

Rapid, regular, and stable updates are described in the Google Kubernetes documentation, on the release channels page. You can choose to disable Node auto-upgrade; however, if you do this, we recommend that you still upgrade monthly or on another set schedule of your choosing.

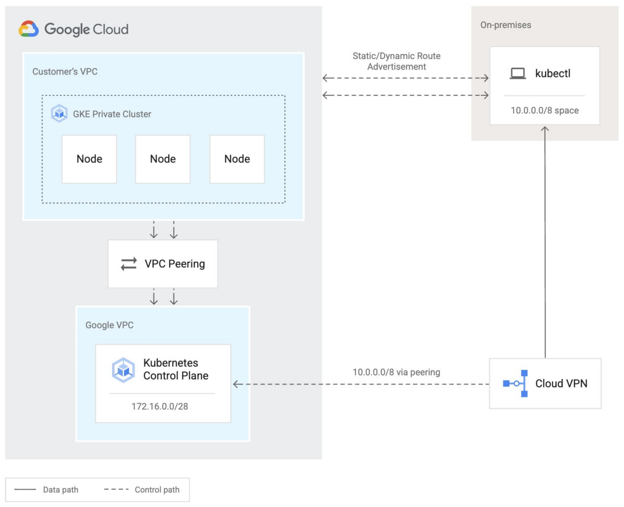
It is a good security practice to regularly audit your cluster configurations for deviations from your defined settings. Many of the recommendations covered in this section, as well as common misconfigurations, can be automatically monitored using the new Security Health Analytics. When Security Health Analytics has been enabled, it will run scans on your selected resources automatically, twice a day (12 hours apart) and alert you if it finds any anomalies.



Proprietary + Confidential

GKE hardening guidelines for robust cluster security

- Restrict direct access to control planes and nodes.
  - **Public endpoint access disabled**
  - **Public endpoint access enabled, control plane authorized networks enabled**
  - **Public endpoint access enabled, control plane authorized networks disabled**
- Consider using Group Authentication.



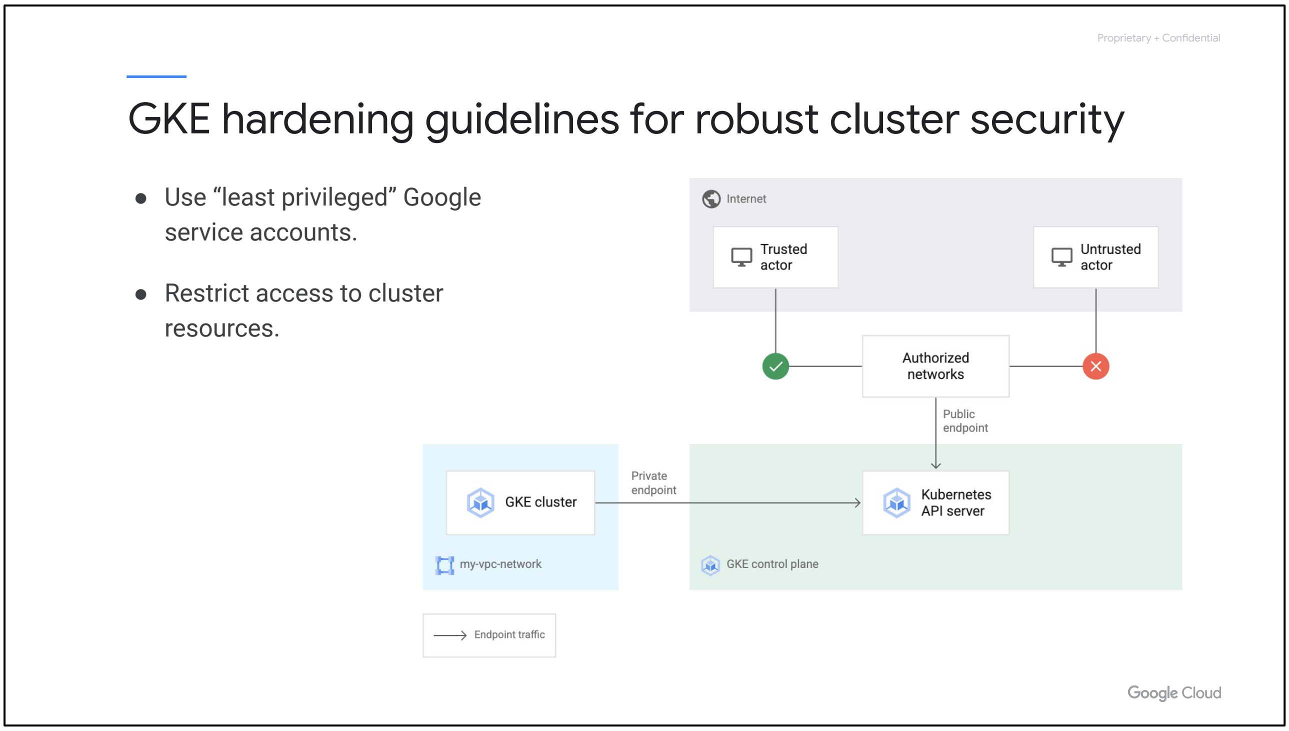
Google Cloud

Another best practice for hardening your clusters is to limit the exposure of your cluster control plane and nodes to the internet. By default, the GKE cluster control plane and nodes have internet-routable addresses that can be accessed from any IP address, but it is possible to change this **if you do so at the time of cluster creation**. It is possible to create a “private cluster” that provides network-level protection to your GKE cluster control plane. Your three options for this are:

- **Public endpoint access disabled:** prevents all internet access to both control planes and nodes and works with networks using Cloud Interconnect and Cloud VPN.
- **Public endpoint access enabled, control plane authorized networks enabled:** gives the control plane a public IP address, but installs a customer configurable firewall in front that allows public internet connections without the use of VPN.
- **Public endpoint access enabled, control plane authorized networks disabled:** this is the default setting and allows any public internet user to make connections to the control plane.

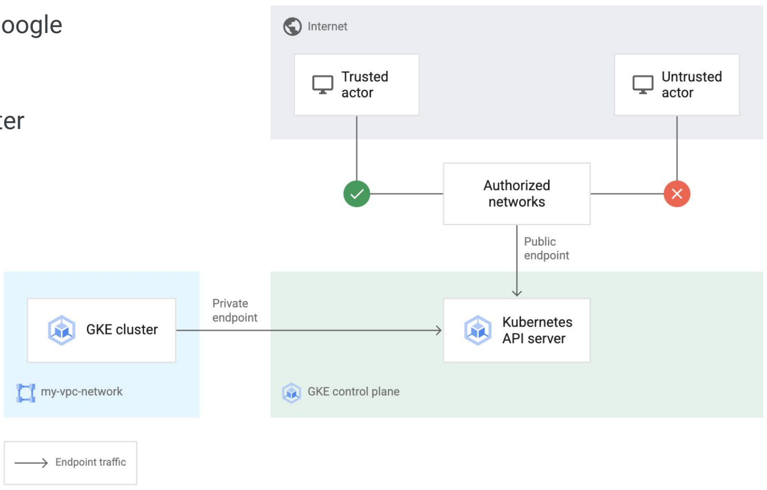
To disable direct internet access to nodes, specify the **gcloud** tool option **--enable-private-nodes** at cluster creation.

Using groups to control identities and access is a security best practice both in the Cloud and on-premises. Now, you can use Group Authentication with your GKE clusters, which removes the need to update your Role Based Access Control configuration whenever anyone is added or removed from the group. Keep in mind that Google Groups for GKE must be enabled while you are creating your clusters.



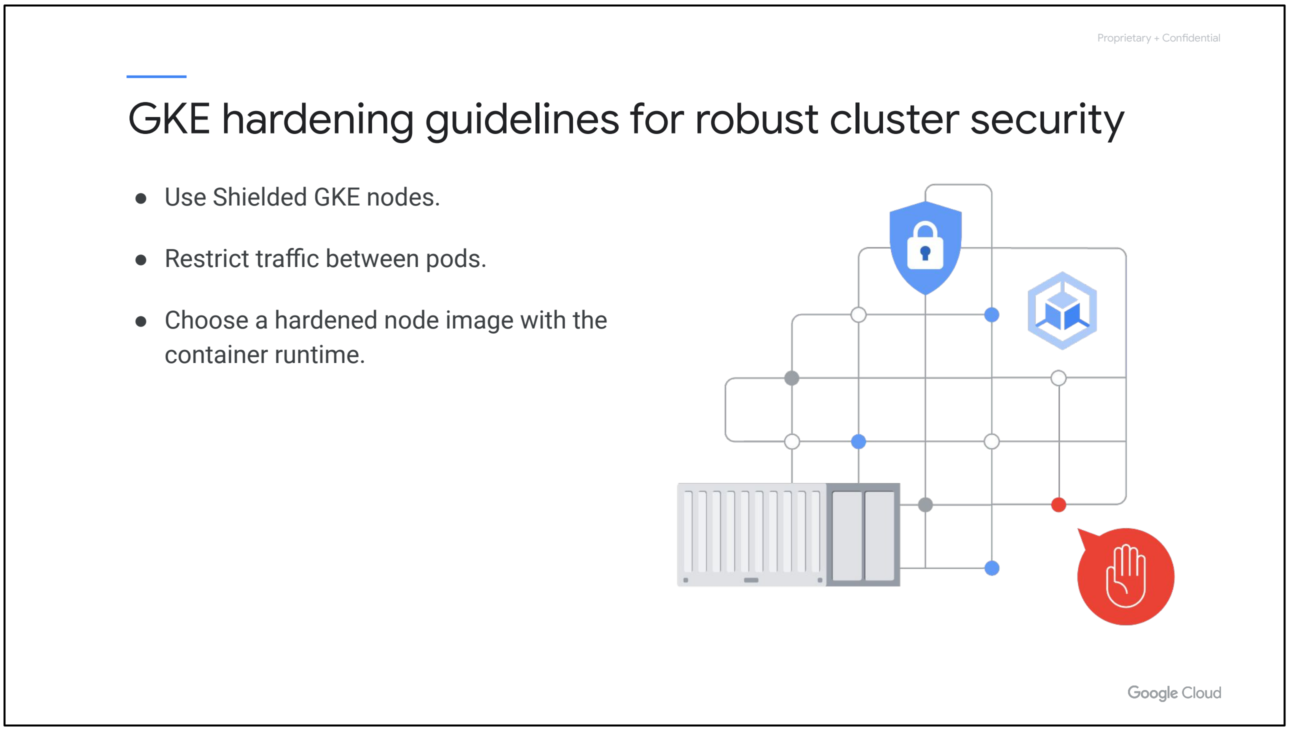
GKE hardening guidelines for robust cluster security

- Use “least privileged” Google service accounts.
- Restrict access to cluster resources.



Each node in GKE is given the Compute Engine default service account, which gives it broad access to resources, but more permissions than are likely to be required to run your GKE cluster. You should create and use a minimally privileged service account to run your GKE cluster instead of using this Compute Engine default service account. When creating this minimally privileged service account, note that GKE requires, at a minimum, the service account to have the monitoring.viewer, monitoring.metricWriter, and logging.logWriter roles.

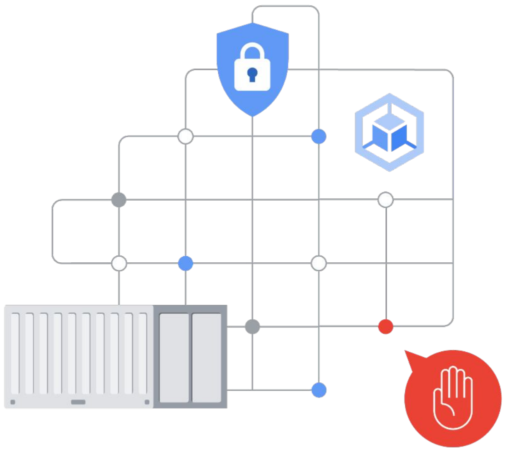
A foundational security concept is to give teams the least amount of privileges required to do their job, and you can do that in Kubernetes by creating separate namespaces or clusters for each team and environment. It is also a good plan to assign cost centers and appropriate labels to each namespace for accountability and chargeback.



Proprietary + Confidential

GKE hardening guidelines for robust cluster security

- Use Shielded GKE nodes.
- Restrict traffic between pods.
- Choose a hardened node image with the container runtime.

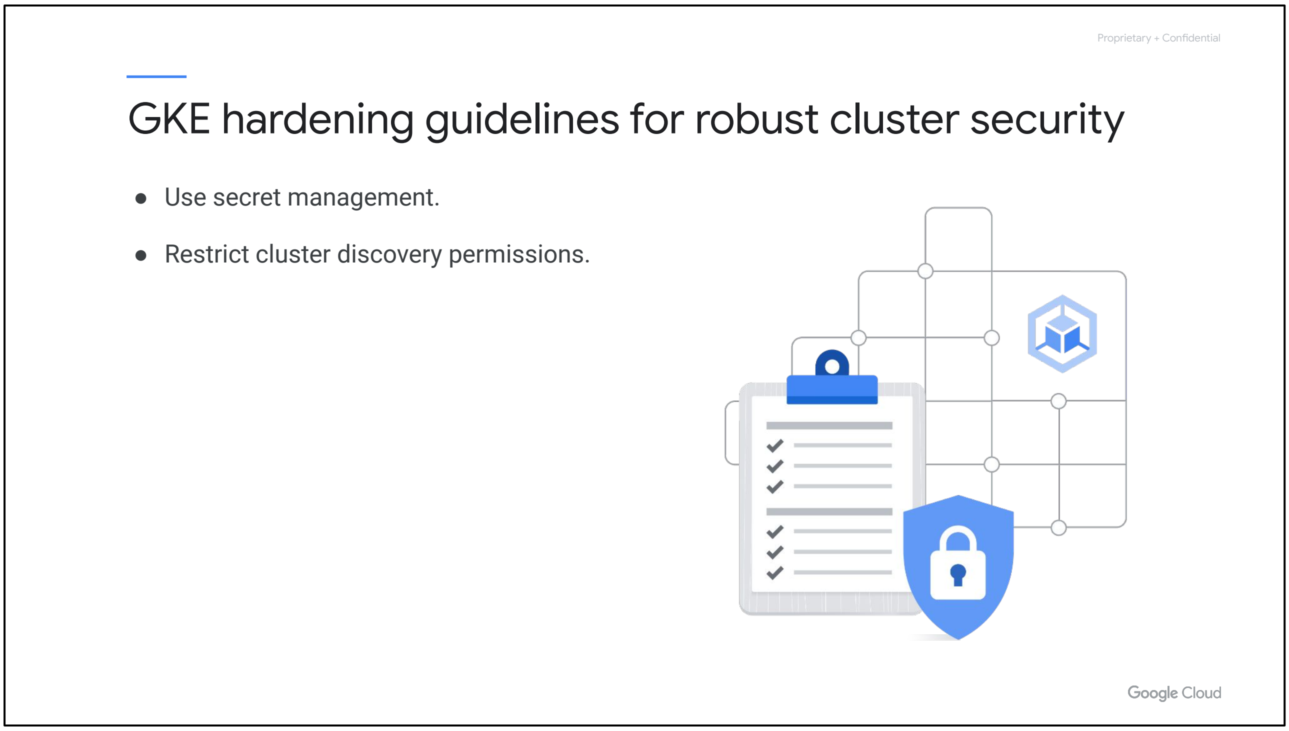


Google Cloud

Just as Shielded VMs provide verifiably secure Compute Engine instances, Shielded GKE nodes provide verifiable node identity and integrity. Upon cluster creation or update, specify the **gcloud** option **--enable-shielded-nodes**. Shielded GKE nodes should be enabled with secure boot; however, secure boot should not be used if you will need to use third-party unsigned kernel modules.

Pods in the same cluster, by default, are able to communicate with each other without restrictions. Allowing this level of communication is often unnecessary, and may even be inadvisable. Restricting network access to services makes it much more difficult for attackers to move laterally within your cluster, and also offers services some protection against accidental or deliberate denial of service. The two recommended ways to restrict traffic between pods are to use Istio (which also offers load balancing, service authorization, throttling, quota, and metrics) or to use Kubernetes Network Policies to provide basic access control functionality.

The Container-Optimized OS with containerd (cos_containerd) image is a variant of the Container-Optimized OS image with containerd as the main container runtime directly integrated with Kubernetes. containerd is the core runtime component of Docker and has been designed to deliver core container functionality for the Kubernetes Container Runtime Interface (CRI). It is significantly less complex than the full Docker daemon, and therefore has a smaller attack surface.



Proprietary + Confidential

GKE hardening guidelines for robust cluster security

- Use secret management.
- Restrict cluster discovery permissions.

Google Cloud

Secret management provides another level of security for authentication “secrets,” which are generally stored in the /etcd directory. To do this you will need to configure a third-party secrets manager, such as HashiCorp Vault, which can be integrated with GKE clusters and will need to be set up before creating your clusters.

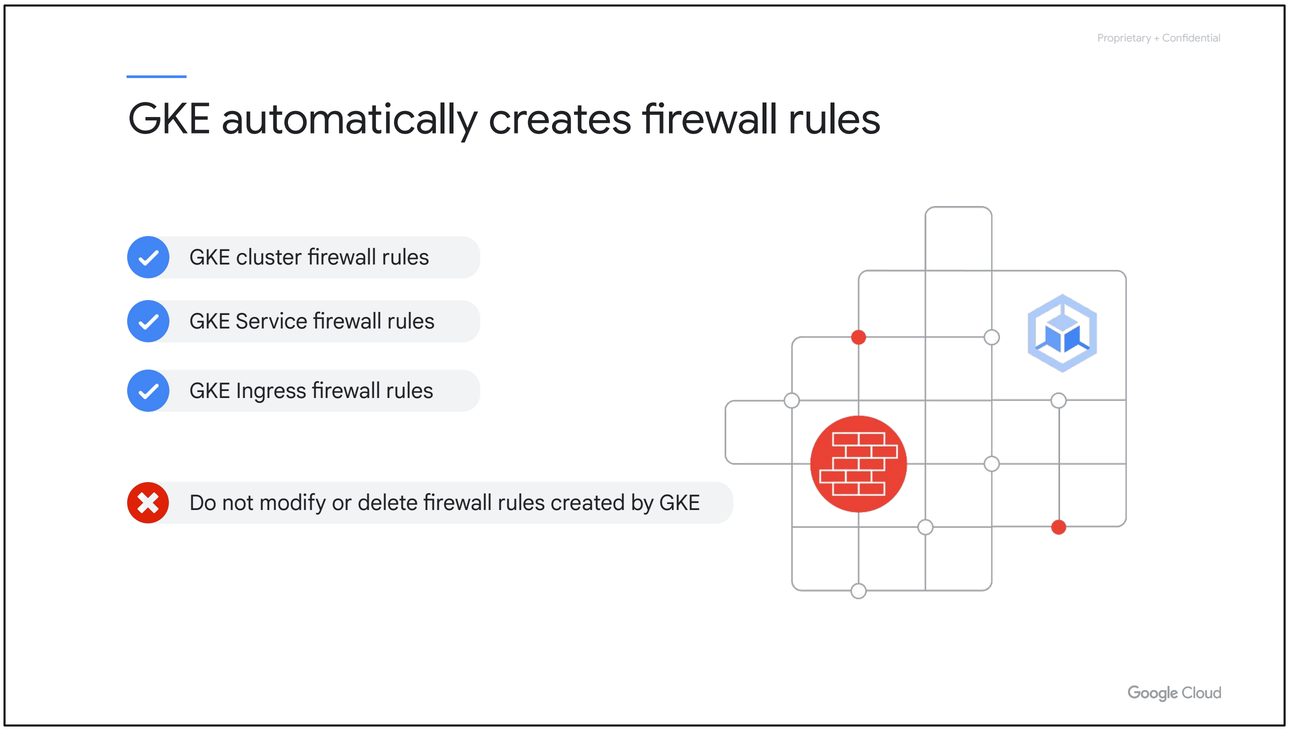
Another option is to use Kubernetes secrets natively in GKE, making sure to encrypt these at the application layer with a key that you manage. Some solutions will work both in GKE and in Anthos GKE deployed on-premises, and so may be more desirable if you are running workloads within a hybrid cloud environment.

By default, clusters come with a permissive set of discovery ClusterRoleBindings which can give broad access to information about a cluster's APIs. You should be aware that the system:authenticated Group can include any authenticated user (including any user with a Google account), and therefore does not represent a meaningful level of security for clusters on GKE. To harden your clusters against discovery exploits, you can:

- Configure authorized networks to restrict access to only a set of IP ranges.
- Set up a private cluster to restrict access to only certain VPCs.
- Curate the subjects of the default system:discovery and system:basic-user ClusterRoleBindings to allow access by only certain known Users and Groups.

For more tips on how to harden your cluster’s security, check out the link in the speaker notes of this module.

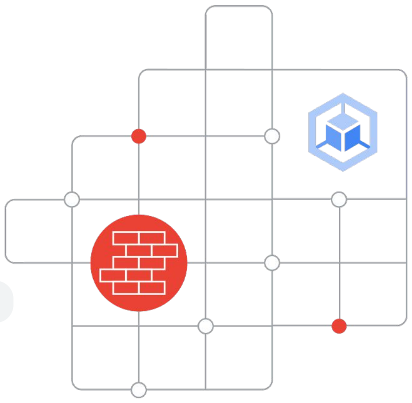
- **Link:** cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#containerd



GKE automatically creates firewall rules

- ✓ GKE cluster firewall rules
- ✓ GKE Service firewall rules
- ✓ GKE Ingress firewall rules

✗ Do not modify or delete firewall rules created by GKE

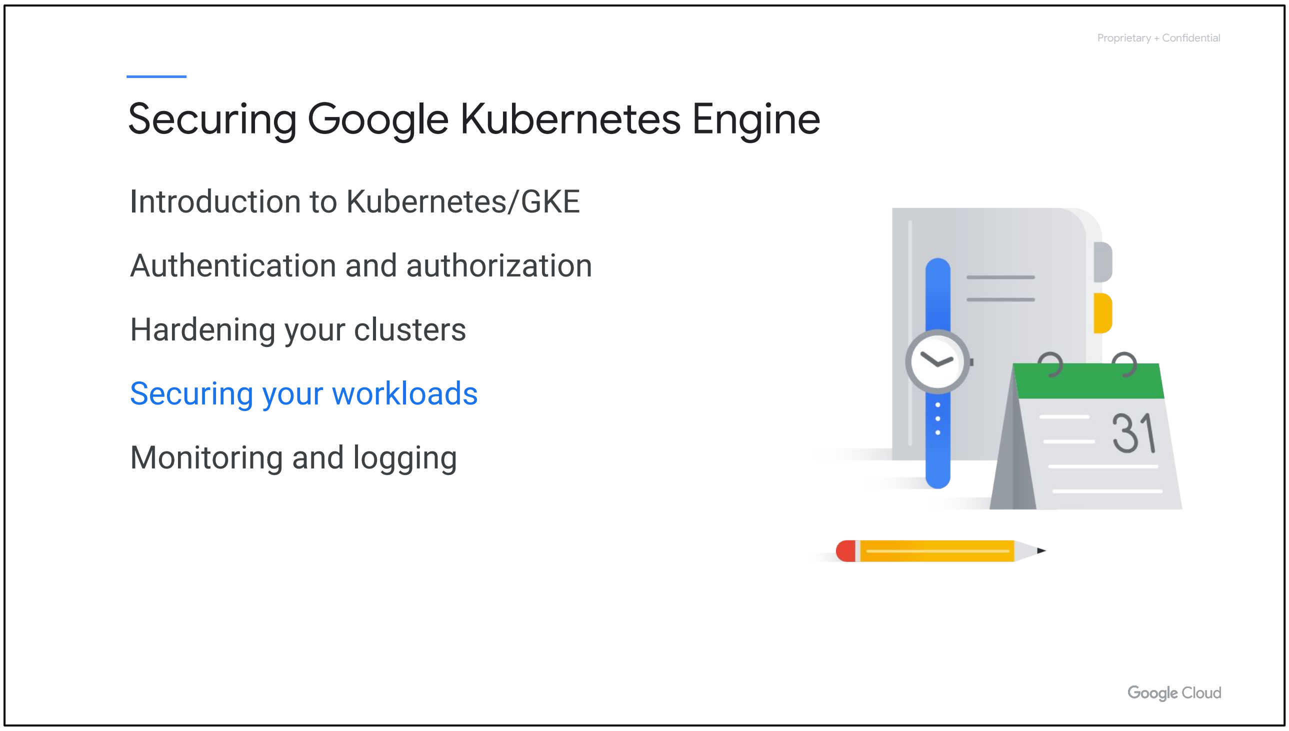


GKE creates ingress firewall rules automatically when creating GKE clusters, GKE Services, and GKE Ingresses.

The priority for all automatically created firewall rules is 1000, which is the default value for firewall rules. If you would like more control over firewall behavior, you can create firewall rules with a higher priority. Firewall rules with a higher priority are applied before automatically created firewall rules.

To avoid unexpected behavior in your clusters, do not modify or delete firewall rules created by GKE.

For more information on the firewall rules automatically created by GKE, refer to the documentation Automatically created firewall rules.



Securing Google Kubernetes Engine

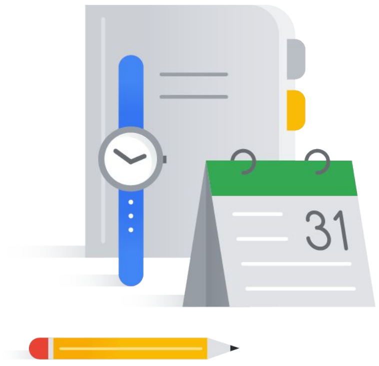
Introduction to Kubernetes/GKE

Authentication and authorization

Hardening your clusters

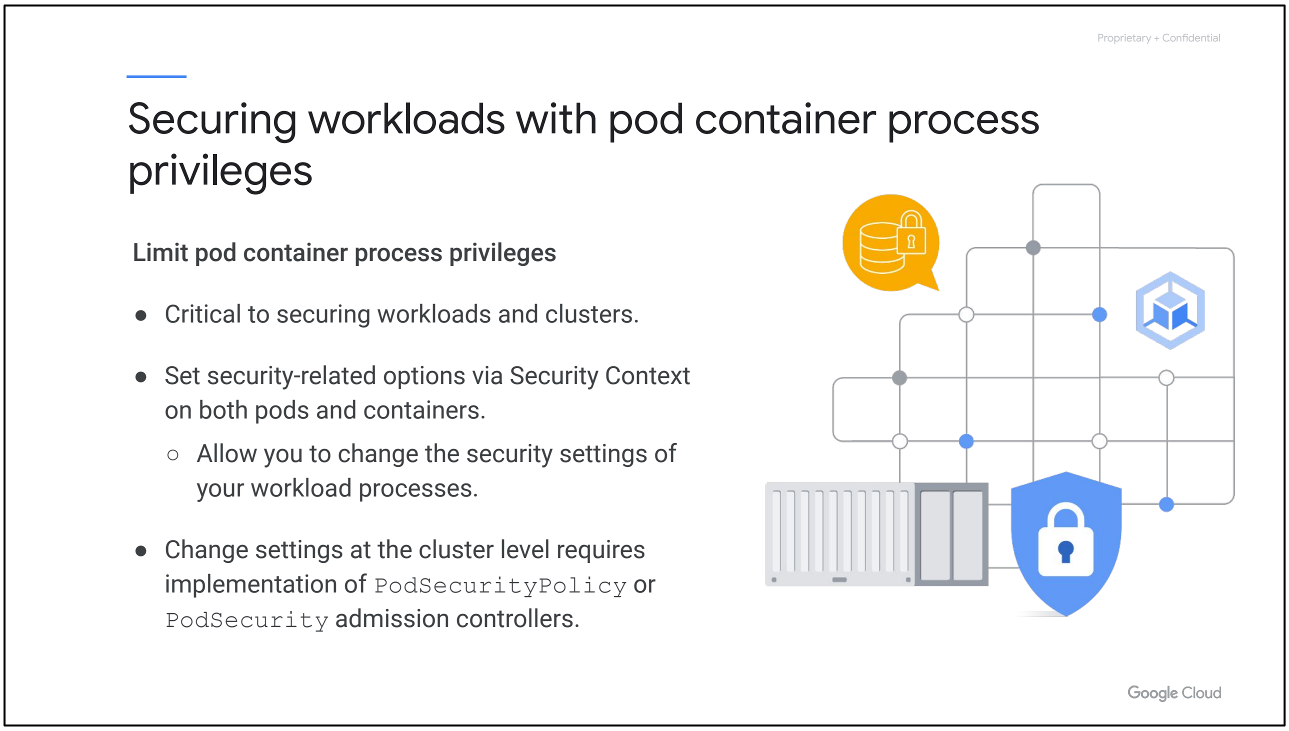
Securing your workloads

Monitoring and logging



Kubernetes allows users to quickly provision, scale, and update container-based workloads.

This lesson describes tactics that administrators and users can employ to limit the effect a running container can have on other Google Cloud resources.



Securing workloads with pod container process privileges

Limit pod container process privileges

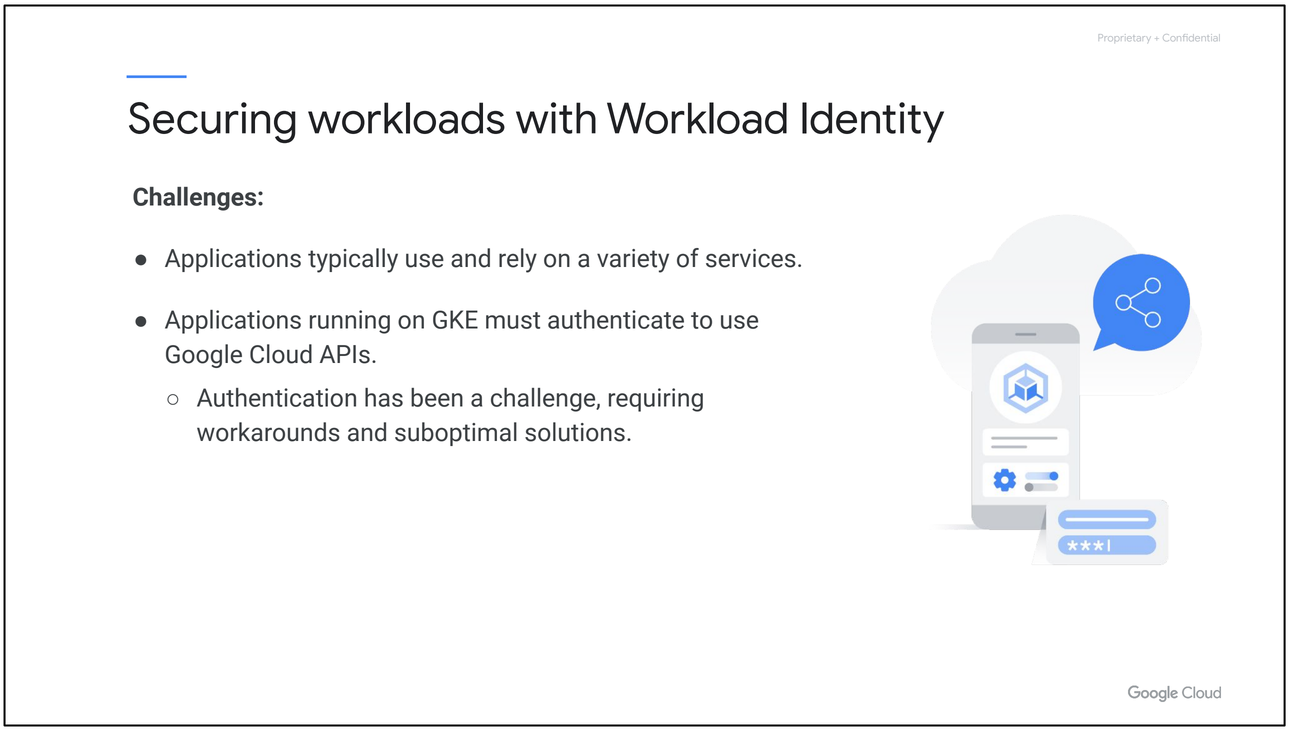
- Critical to securing workloads and clusters.
- Set security-related options via Security Context on both pods and containers.
  - Allow you to change the security settings of your workload processes.
- Change settings at the cluster level requires implementation of `PodSecurityPolicy` or `PodSecurity` admission controllers.

Limiting pod container permissions is both critical to securing your workloads and an important part of securing your clusters. Google Kubernetes Engine allows you to set security-related options via the Security Context on both pods and containers.

These settings allow you to change the security settings of your workload processes, for example, changing the “run as” user and group.

To change settings at the cluster level, you will need to implement `PodSecurityPolicy` or `PodSecurity` admission controllers based on the Kubernetes version, to ensure that all Pods in a cluster adhere to a minimum baseline policy that you have defined.

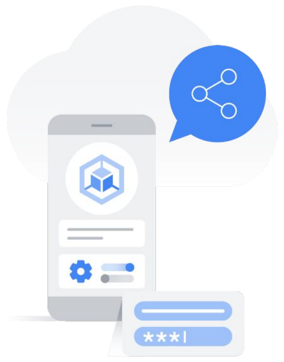
The Kubernetes project deprecated `PodSecurityPolicy` and removed the feature entirely in Kubernetes v1.25.



Securing workloads with Workload Identity

Challenges:

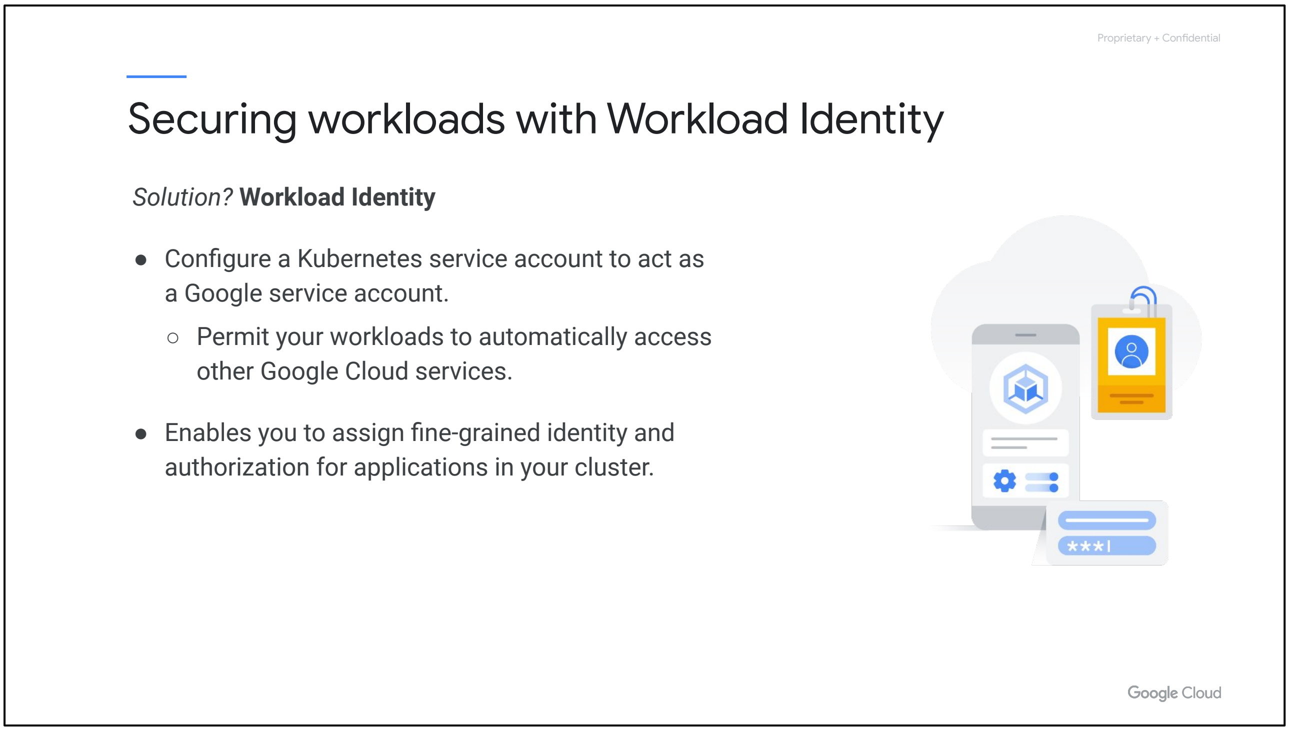
- Applications typically use and rely on a variety of services.
- Applications running on GKE must authenticate to use Google Cloud APIs.
  - Authentication has been a challenge, requiring workarounds and suboptimal solutions.



An application has needs. Maybe it needs to connect to a data warehouse, or connect to a machine learning training set. No matter what your application needs to do, there's a good chance it needs to connect to other services to get it done.

More specifically, applications running on GKE must authenticate to use Google Cloud APIs such as the Compute APIs, Storage and Database APIs, or Machine Learning APIs.

If that app runs on Kubernetes, this kind of authentication has traditionally been a challenge, requiring workarounds and suboptimal solutions.

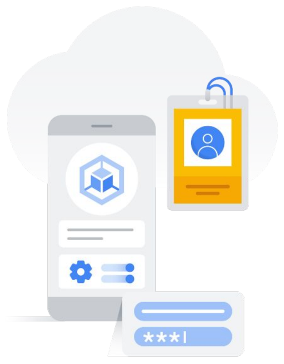


Proprietary + Confidential

Securing workloads with Workload Identity

Solution? **Workload Identity**

- Configure a Kubernetes service account to act as a Google service account.
    - Permit your workloads to automatically access other Google Cloud services.
- Enables you to assign fine-grained identity and authorization for applications in your cluster.



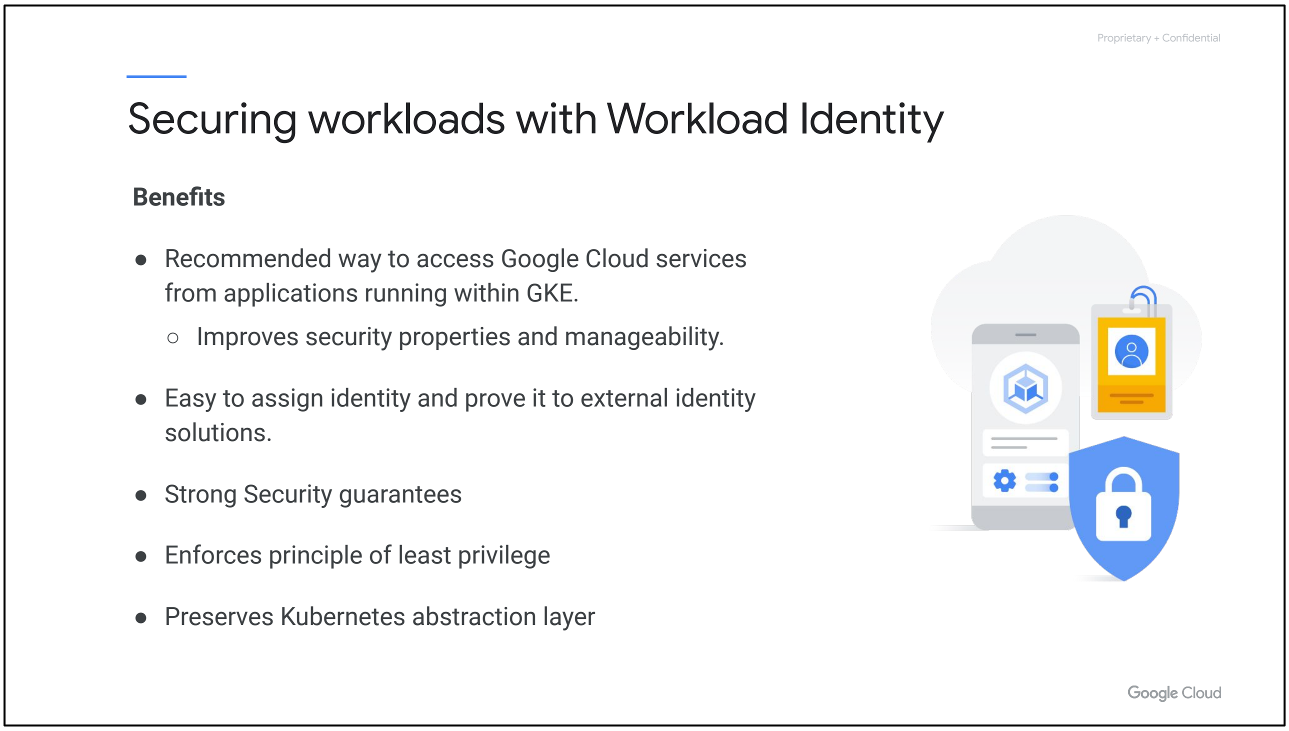
Google Cloud

The solution to this problem? Workload Identity.

With Workload Identity, you can configure a Kubernetes service account to act as a Google service account.

Workload Identity works by creating a relationship between Kubernetes service accounts and Cloud IAM service accounts, so you can use Kubernetes-native concepts to define which workloads run as which identities, and permit your workloads to automatically access other Google Cloud services—all without having to manage Kubernetes secrets or IAM service account keys!

Essentially, Workload Identity enables you to assign fine-grained identity and authorization for applications in your cluster.



Securing workloads with Workload Identity

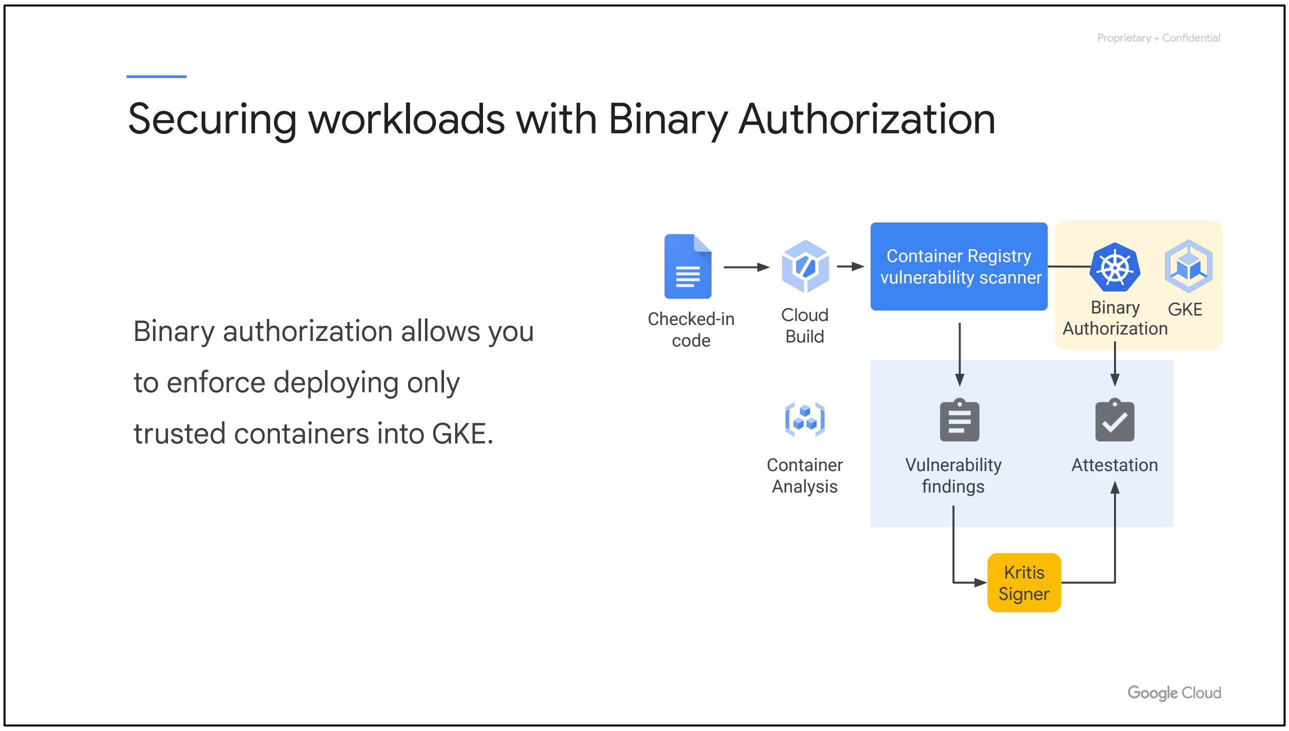
Benefits

- Recommended way to access Google Cloud services from applications running within GKE.
  - Improves security properties and manageability.
- Easy to assign identity and prove it to external identity solutions.
- Strong Security guarantees
- Enforces principle of least privilege
- Preserves Kubernetes abstraction layer

Workload Identity is the recommended way to access Google Cloud services from applications running within GKE due to its improved security properties and manageability.

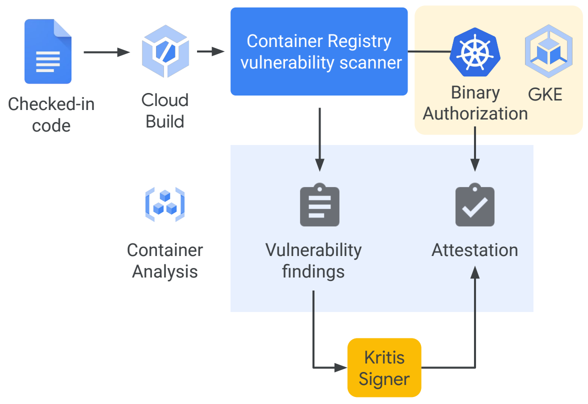
Workload Identity also:

- Allows you to easily assign identity and prove it to external identity solutions
- Provides strong security guarantee
- Enforces principle of least privilege
- And preserves Kubernetes abstraction layer



Securing workloads with Binary Authorization

Binary authorization allows you to enforce deploying only trusted containers into GKE.

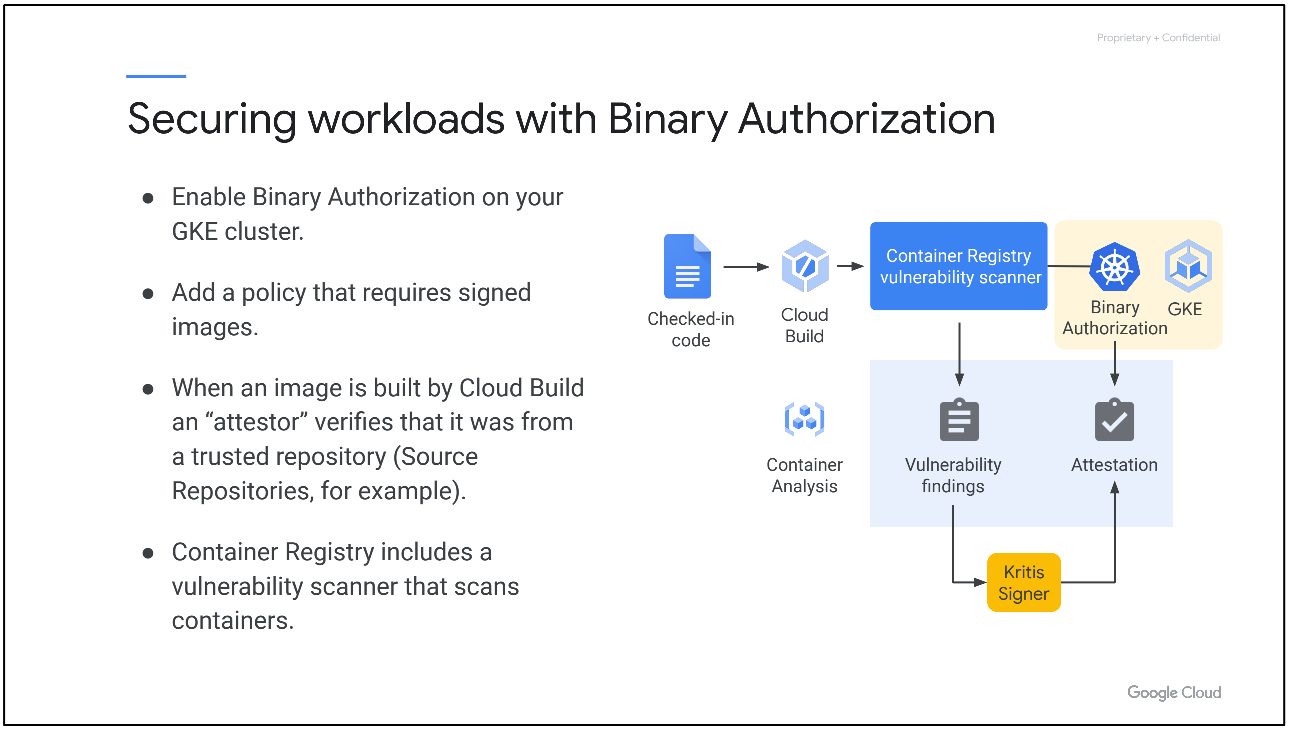


Binary authorization allows you to enforce deploying only trusted containers into GKE.

It provides software supply-chain security for applications that run in the Cloud. Binary Authorization works with images that you deploy to GKE from Container Registry or another container image registry.

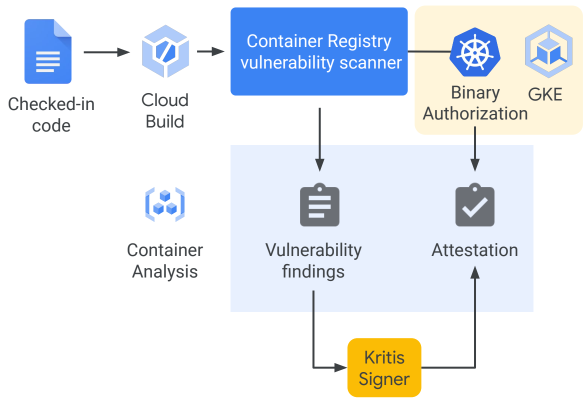
This allows you to ensure that the internal processes that safeguard the quality and integrity of your software have successfully completed before an application is deployed to your production environment.

So, how does Binary Authorization work? That comes in 4 easy steps.

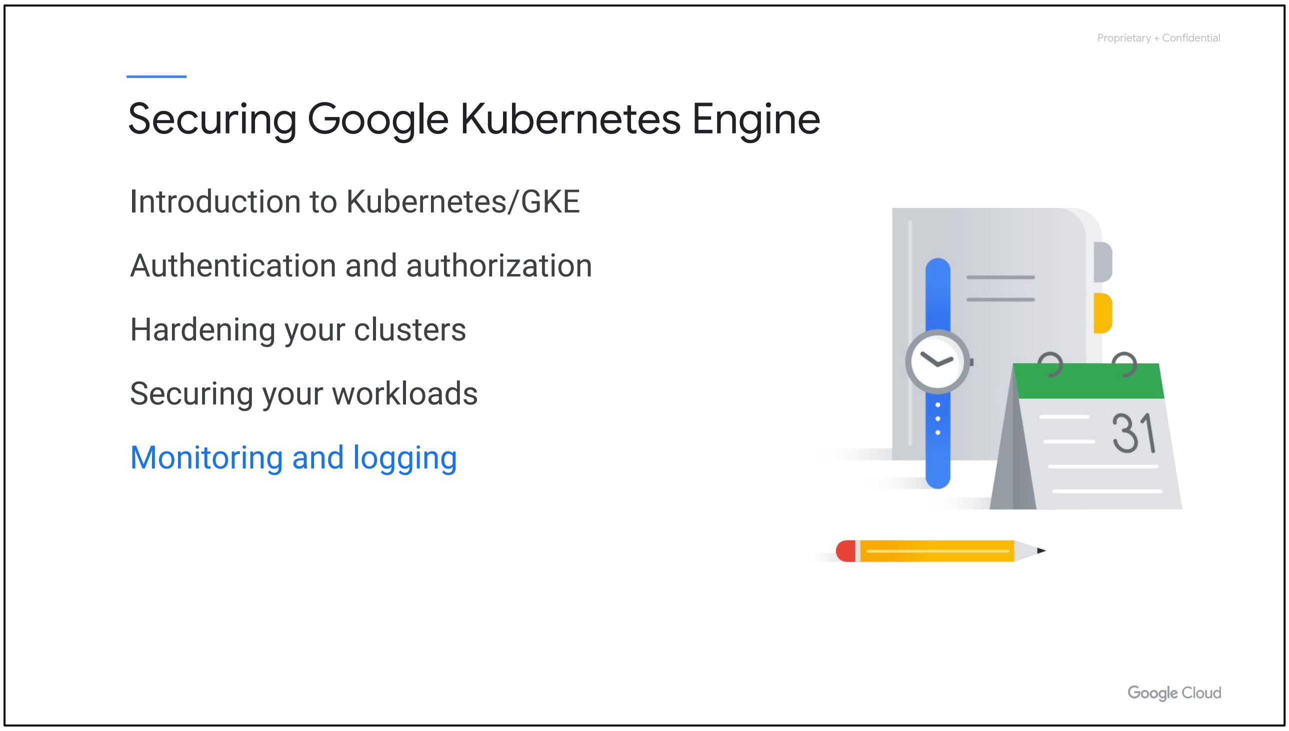


Securing workloads with Binary Authorization

- Enable Binary Authorization on your GKE cluster.
- Add a policy that requires signed images.
- When an image is built by Cloud Build an “attestor” verifies that it was from a trusted repository (Source Repositories, for example).
- Container Registry includes a vulnerability scanner that scans containers.



- First, you enable binary authorization on your GKE cluster.
- Next, you add a policy that requires signed images.
- Then, when an image is built by Cloud Build an “attestor” verifies that it was from a trusted repository (for example, Source Repositories, which are fully featured, private Git repositories hosted on Google Cloud).
- Finally, Container Registry includes a vulnerability scanner that scans containers.



Securing Google Kubernetes Engine

Introduction to Kubernetes/GKE

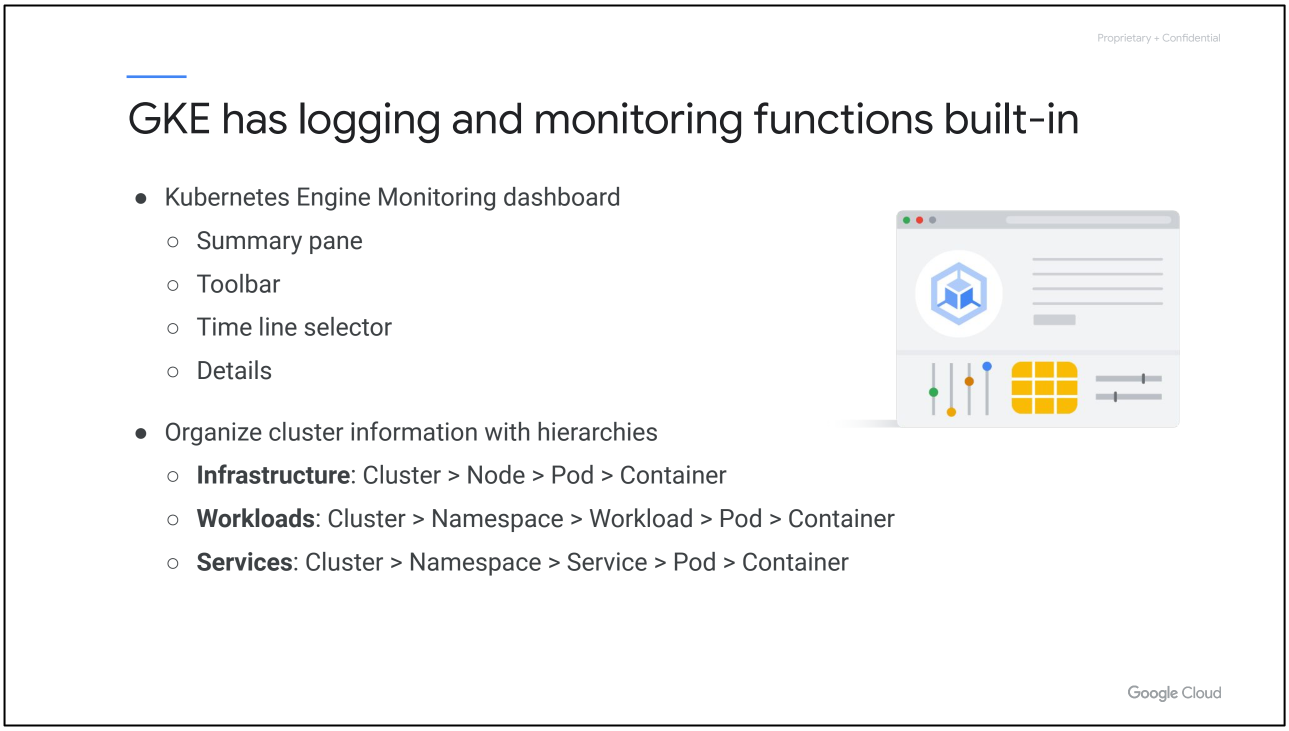
Authentication and authorization

Hardening your clusters

Securing your workloads

Monitoring and logging

GKE includes native integration with Cloud Monitoring and Cloud Logging. When you create a GKE cluster, Kubernetes Engine Monitoring is enabled by default and provides a monitoring dashboard specifically tailored for Kubernetes. You can control whether Kubernetes Engine Monitoring collects application logs, and you can even disable Cloud Monitoring and Cloud Logging completely.



Proprietary + Confidential

GKE has logging and monitoring functions built-in

- Kubernetes Engine Monitoring dashboard
  - Summary pane
  - Toolbar
  - Time line selector
  - Details
- Organize cluster information with hierarchies
  - **Infrastructure**: Cluster > Node > Pod > Container
  - **Workloads**: Cluster > Namespace > Workload > Pod > Container
  - **Services**: Cluster > Namespace > Service > Pod > Container

Google Cloud

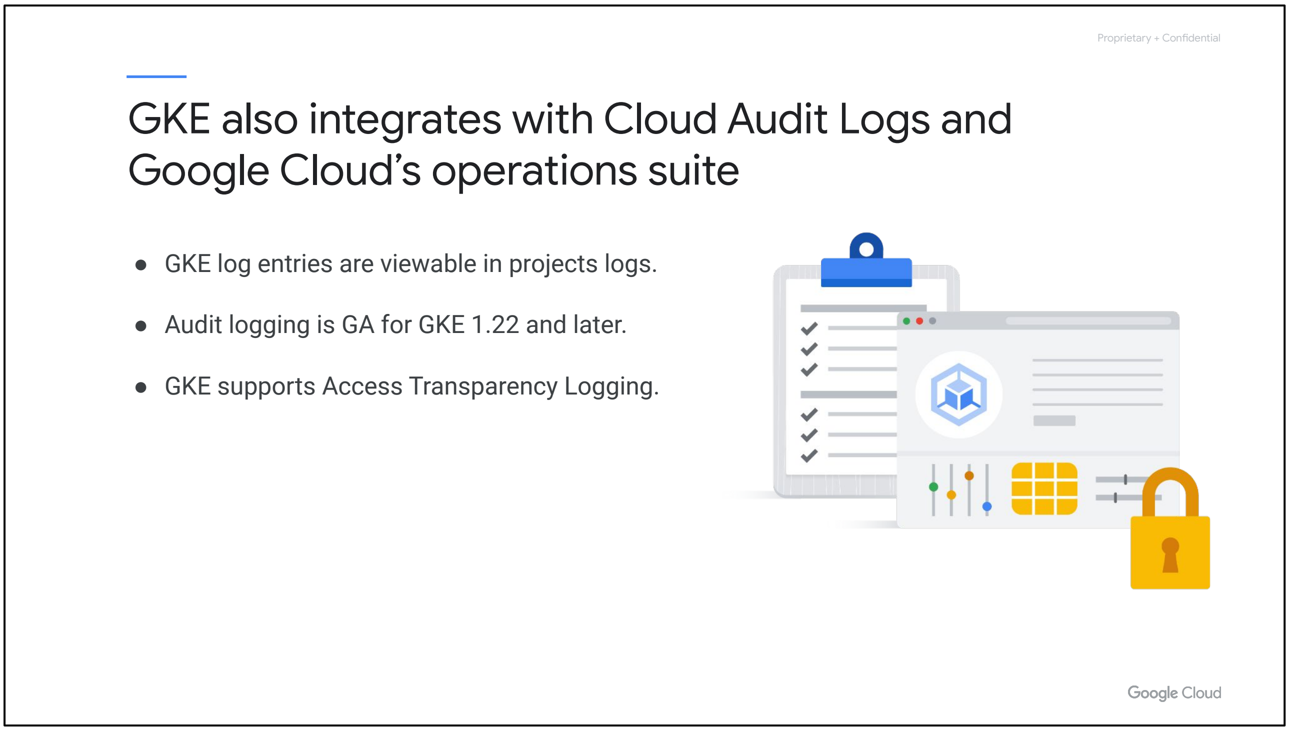
In the Kubernetes Engine Monitoring dashboard summary pane, you can view a cluster's key performance metrics, such as CPU utilization, memory utilization, and the number of open incidents. There are controls associated with this view that allow you to filter and summarize these metrics.

- The dashboard toolbar controls the time window for observations and provides dashboard settings and filters.
- The timeline event selector lets you select a specific time and display summaries of alerts.
- The details section lets you choose how your cluster information is presented to you.

The Kubernetes Engine Monitoring dashboard viewing tabs let you organize your cluster information using different hierarchies:

- **Infrastructure**: Aggregates resources by **Cluster**, then **Node**, then **Pod**, and then by **Container**.
- **Workloads**: Aggregates resources by **Cluster**, then **Namespace**, then **Workload**, then **Pod**, and lastly by **Container**.
- **Services**: Aggregates resources by **Cluster**, then **Namespace**, then **Service**, then **Pod**, and lastly by **Container**.

You can use the “resource type” filter to narrow the information reported in the dashboard to a specific kind of resource, which allows you to more closely inspect namespaces, nodes, workloads, services, pods, and containers. For pods and containers, you can also view metrics as a function of time and view log entries using the Logs Explorer.



GKE also integrates with Cloud Audit Logs and Google Cloud's operations suite

- GKE log entries are viewable in projects logs.
- Audit logging is GA for GKE 1.22 and later.
- GKE supports Access Transparency Logging.

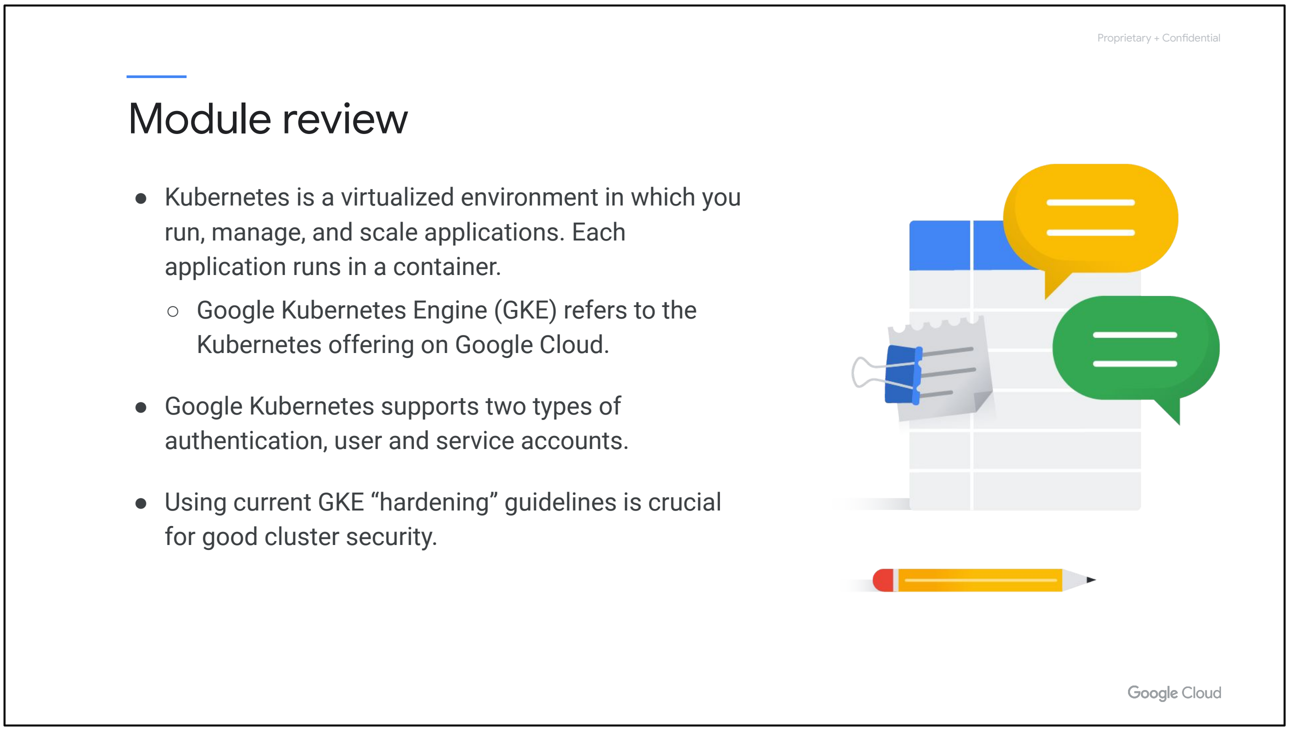
You can use the default Kubernetes Engine Monitoring or opt in to use Google Cloud's operations suite, formerly known as Stackdriver.

Both options are generally available as of GKE version 1.22 and later.

When deciding which monitoring and logging service to use, take the following into account:

- Kubernetes Engine Monitoring is the default option, starting with GKE version 1.15.
- Only Google Cloud's operations suite lets you disable Cloud Logging while still using Cloud Monitoring.

GKE also supports Access Transparency Logging, which records the actions taken by Google personnel in your Google Cloud resources (if ever taken).



Proprietary + Confidential

Module review

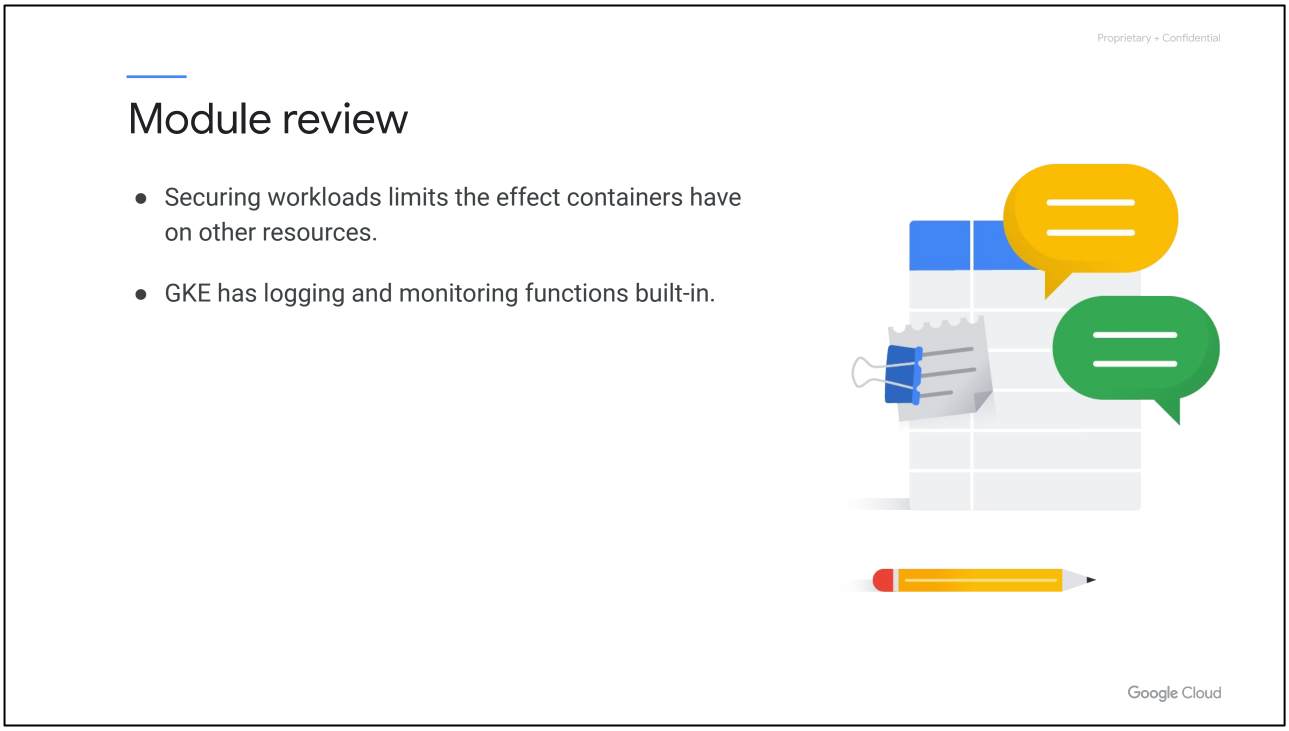
- Kubernetes is a virtualized environment in which you run, manage, and scale applications. Each application runs in a container.
  - Google Kubernetes Engine (GKE) refers to the Kubernetes offering on Google Cloud.
- Google Kubernetes supports two types of authentication, user and service accounts.
- Using current GKE “hardening” guidelines is crucial for good cluster security.

Google Cloud

In this module, we learned that Kubernetes is a virtualized environment in which you run, manage, and scale applications. Each application runs in a container. While Kubernetes is also available from other vendors, Google Kubernetes Engine, or GKE, refers to the Kubernetes offering on Google Cloud.

Google Kubernetes supports two types of authentication. User accounts are accounts that are known to Kubernetes, but are not managed by Kubernetes. Service accounts are accounts that are created and managed by Kubernetes, but can only be used by Kubernetes-created entities.

To ensure good cluster security, it is crucial that the current range of GKE “hardening” guidelines are followed.



Module review

- Securing workloads limits the effect containers have on other resources.
- GKE has logging and monitoring functions built-in.

Securing workloads by limiting pod container permissions, using Workload Identity, and enabling Binary Authorization limits the effect containers have on other resources.

GKE includes native integration with Cloud Monitoring and Cloud Logging. When you create a GKE cluster, Kubernetes Engine Monitoring is enabled by default and provides a monitoring dashboard specifically tailored for Kubernetes. You can control whether Kubernetes Engine Monitoring collects application logs, and you can even disable Cloud Monitoring and Cloud Logging completely.