Access Control and Security in Kubernetes and GKE

Access Control and Security in Kubernetes and GKE

01 Authentication and authorization

02 Kubernetes role-based access control

03 Workload Identity

04 Kubernetes Control Plane security

05 Pod security

Access Control and Security in Kubernetes and GKE

01 Authentication and authorization

02 Kubernetes role-based access control

03 Workload Identity

04 Kubernetes Control Plane security

05 Pod security

Kubernetes user access falls into two categories

Typically defined through a Cloud Identity domain.

Individual user accounts

Workload Identity bridges the gap between Kubernetes service accounts and IAM service accounts.

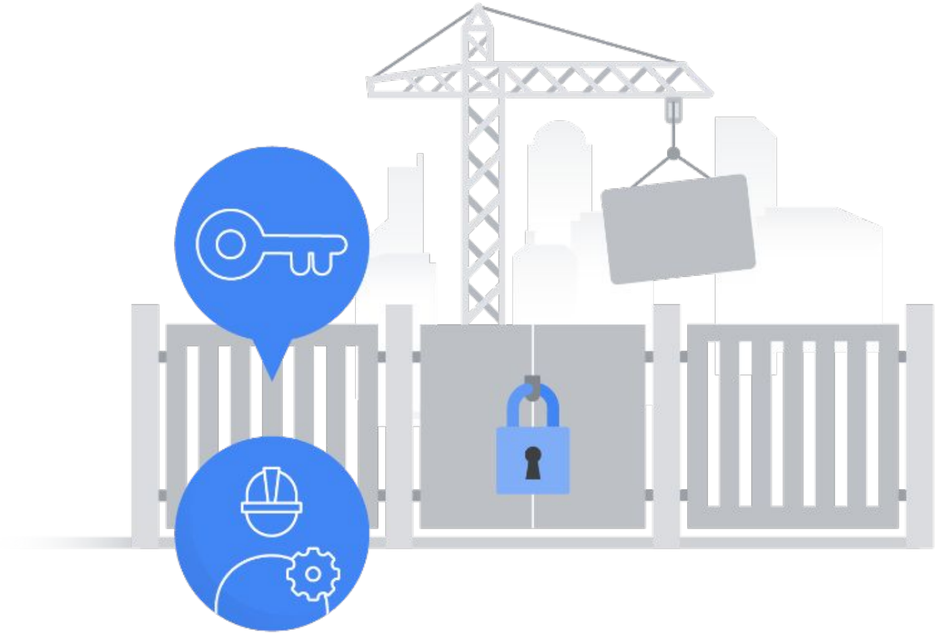
It provides access to other Google Cloud services.

Kubernetes service accounts

Workload Identity securely authenticates pods

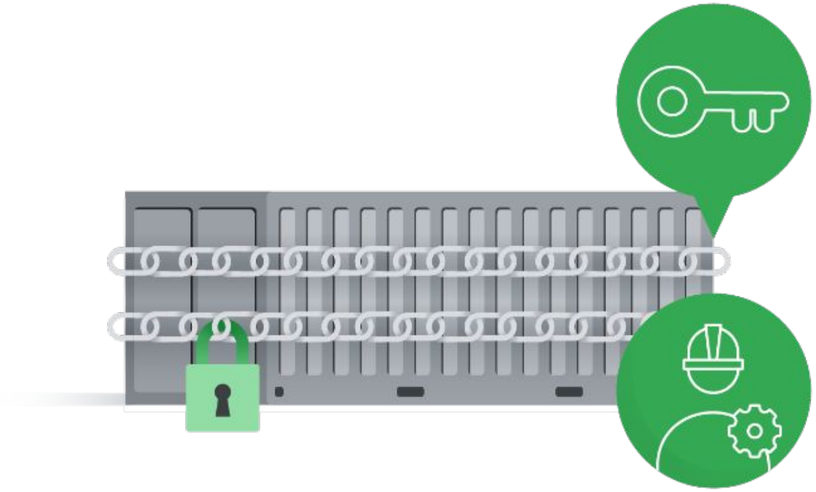
- Securely authenticate pods by keeping service account credentials outside the pods themselves.
- Pods automatically receive their service account credentials from the GKE metadata server.
- The GKE metadata server is a subset of the Compute Engine metadata server endpoints.
- Workload Identity simplifies pod authentication by eliminating the need to manage credentials within the pods.

Account authorization with IAM and RBAC



IAM

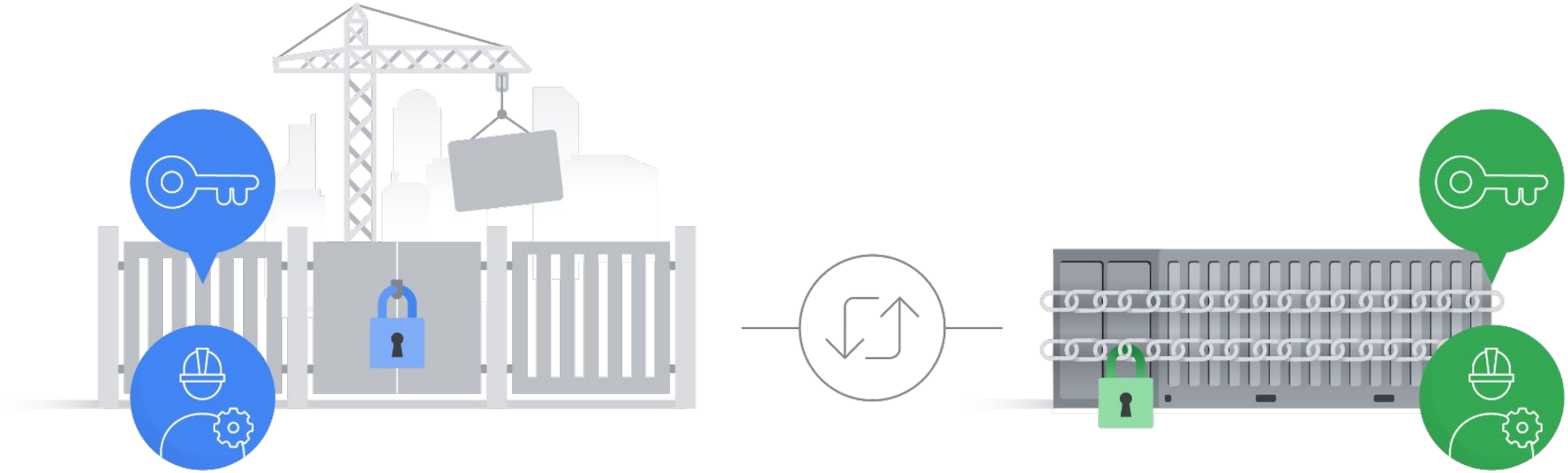
Define precise permissions
at the project or cluster level.



RBAC

Define granular permissions
at the cluster and namespace levels.

Comprehensive access control



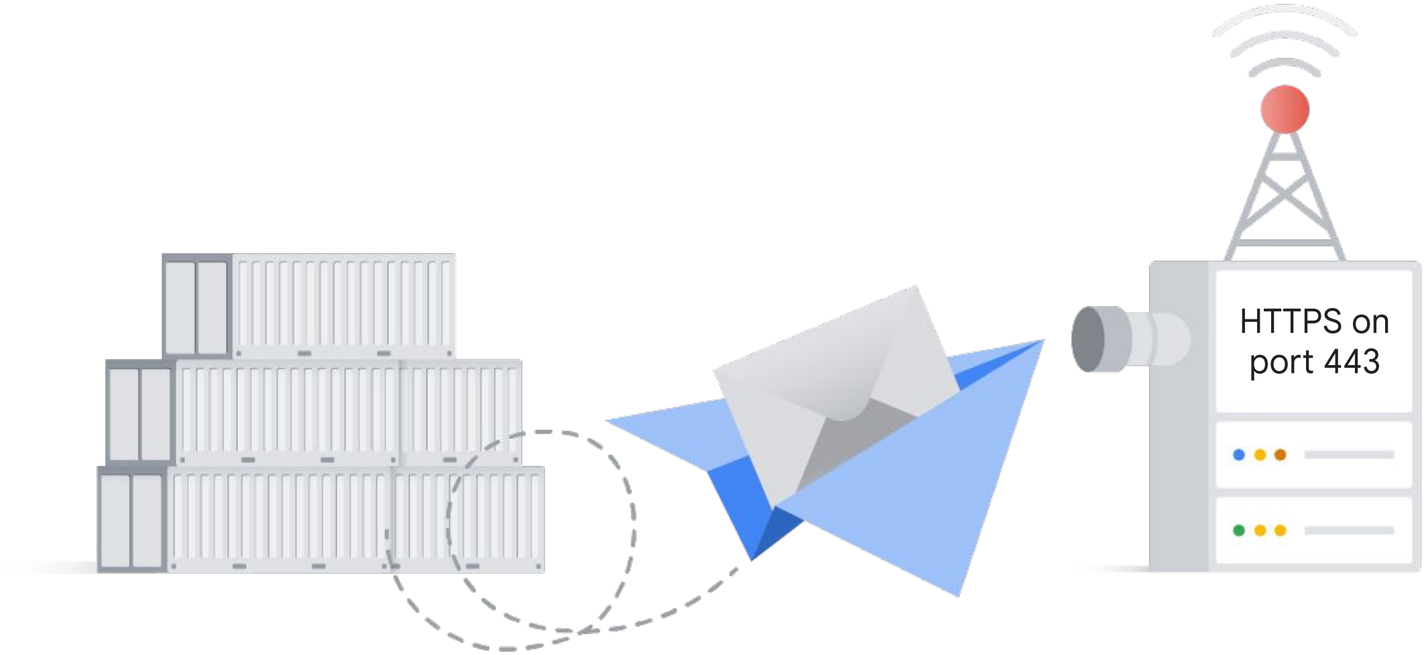
IAM

Who can access and modify
the **overall configurations.**

RBAC

Who can interact with
the **individual objects in a cluster.**

Authenticating requests outside a cluster



GKE manages end-user authentication for you.
It is best practice to authenticate using OpenID connect tokens.

Access Control and Security in Kubernetes and GKE

01 Authentication and authorization

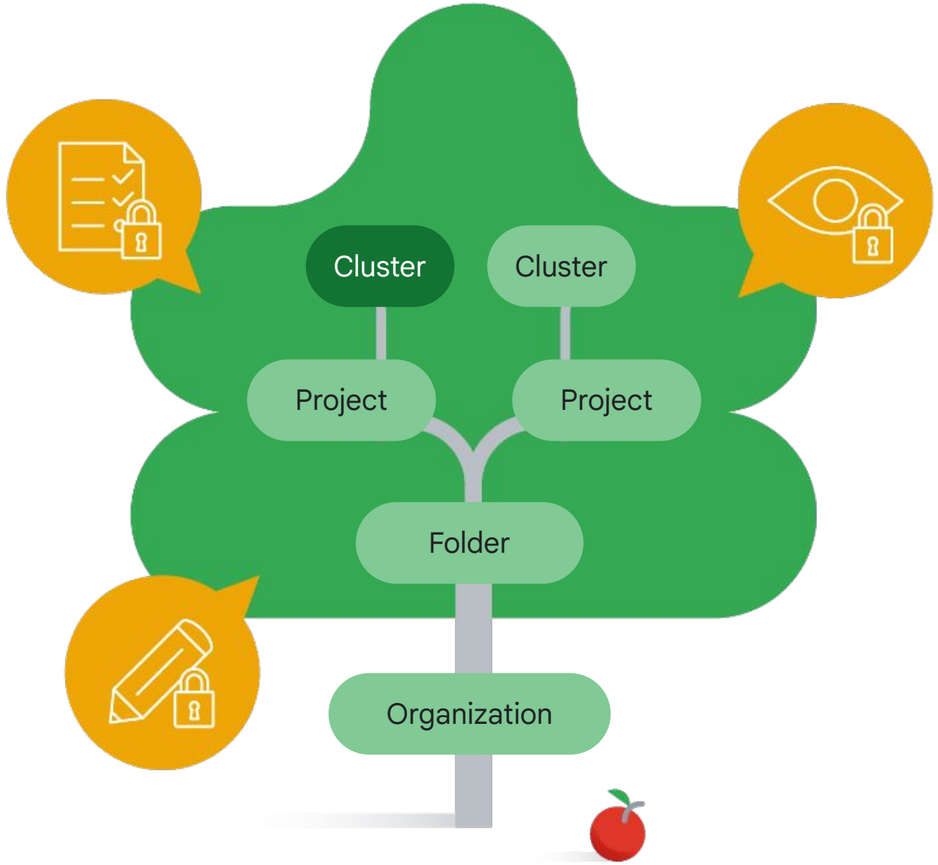
02 Kubernetes role-based access control

03 Workload Identity

04 Kubernetes Control Plane security

05 Pod security

Enhanced security with Kubernetes RBAC + IAM



IAM manages access at the GKE and cluster levels.

RBAC provides control over individual Kubernetes resources inside the cluster.

What Kubernetes RBAC controls

Subjects

Users, groups, or service accounts authorized to interact with the Kubernetes API server.

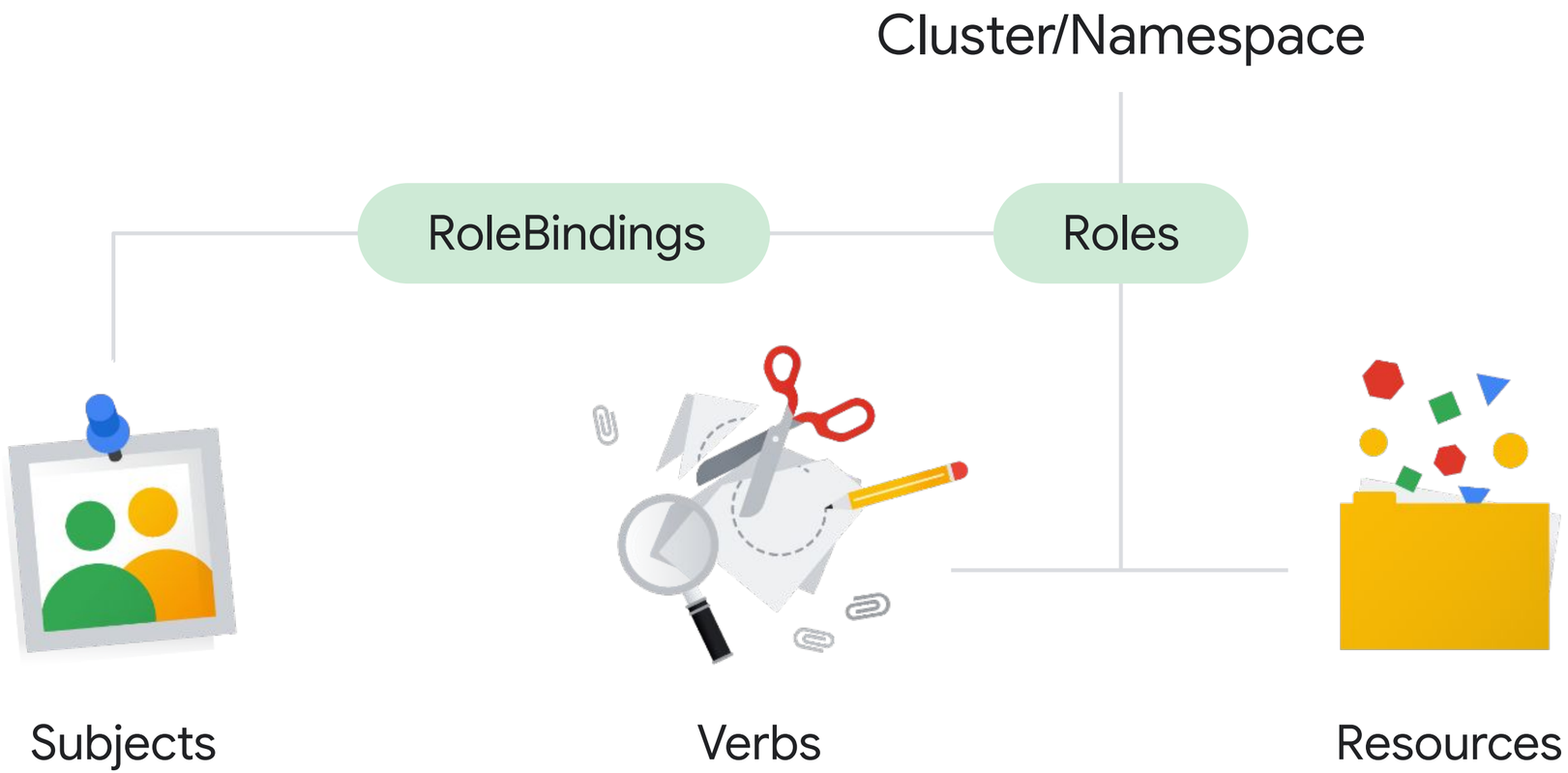
Verbs

Define the specific actions allowed for each resource, such as creating, modifying, or viewing.

Resources

Encompass the Kubernetes objects the subjects can access.

What Kubernetes RBAC Controls



Defining namespace RBAC roles in manifests

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: pod-reader
  namespace: default  # Specify the namespace where the Role applies
rules:
- apiGroups: [""]  # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
```

RBAC ClusterRoles are defined at the cluster level

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: cluster-admin  # Example name for a ClusterRole
rules:
- apiGroups: ["", "extensions", "apps"]  # Multiple API groups
  resources: ["*"]  # Apply to all resources within those groups
  verbs: ["*"]  # Allow all actions on those resources
```

What RBAC rule sets control

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

- Used to limit the scope to:
  - Specific instances of a resource.
  - Verbs specified as **patch** and **update.**

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

- Used to specify access to Pod logs.
- Assigned as a unit, all or none.

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

- Used to specify get and post actions for non-resource endpoints.
- Unique to ClusterRoles.

Attaching RoleBindings and ClusterRoleBindings

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

kubectl api-resources command

kubectl api-resources

```
--namespaced=true – namespaced resources
--namespaced=false – cluster-wide resources
```

- Cluster-level resources: ClusterRoles
- Namespaced resources: Roles
- Multiple namespaces: ClusterRoles

Access Control and Security in Kubernetes and GKE

01 Authentication and authorization

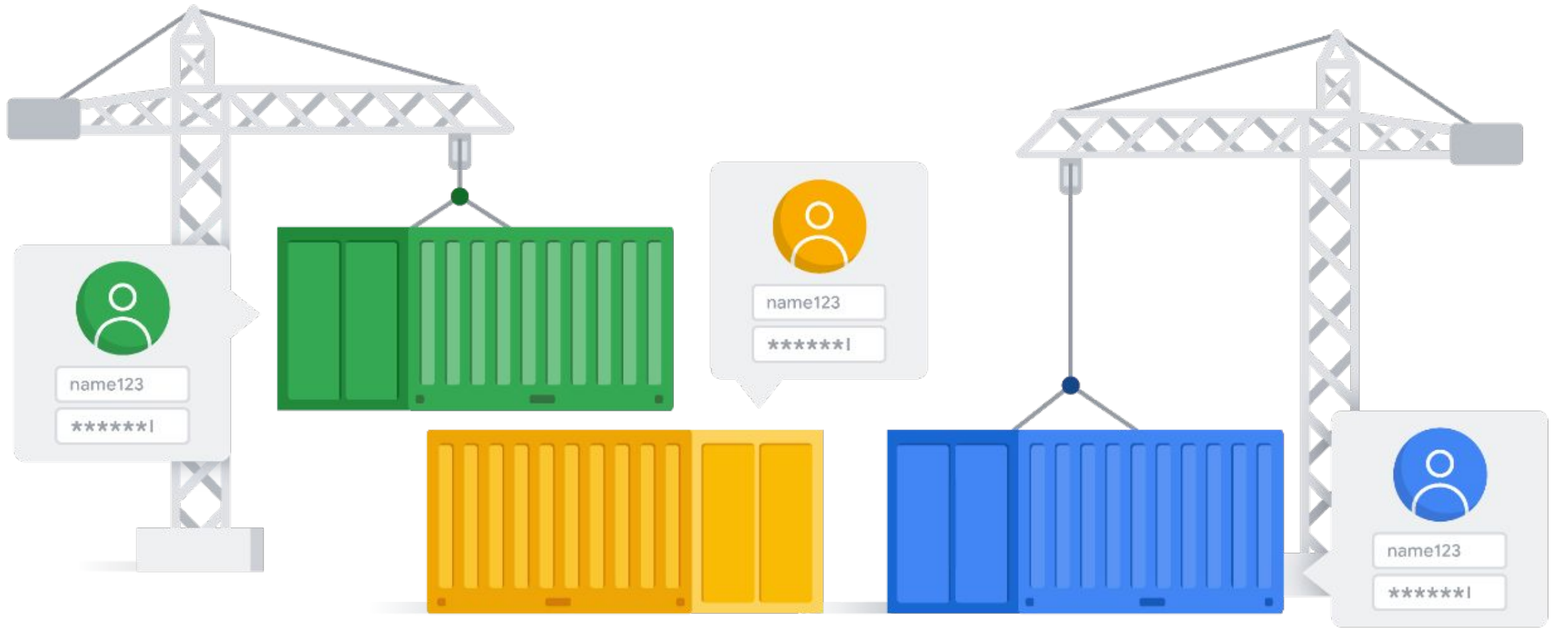
02 Kubernetes role-based access control

03 Workload Identity

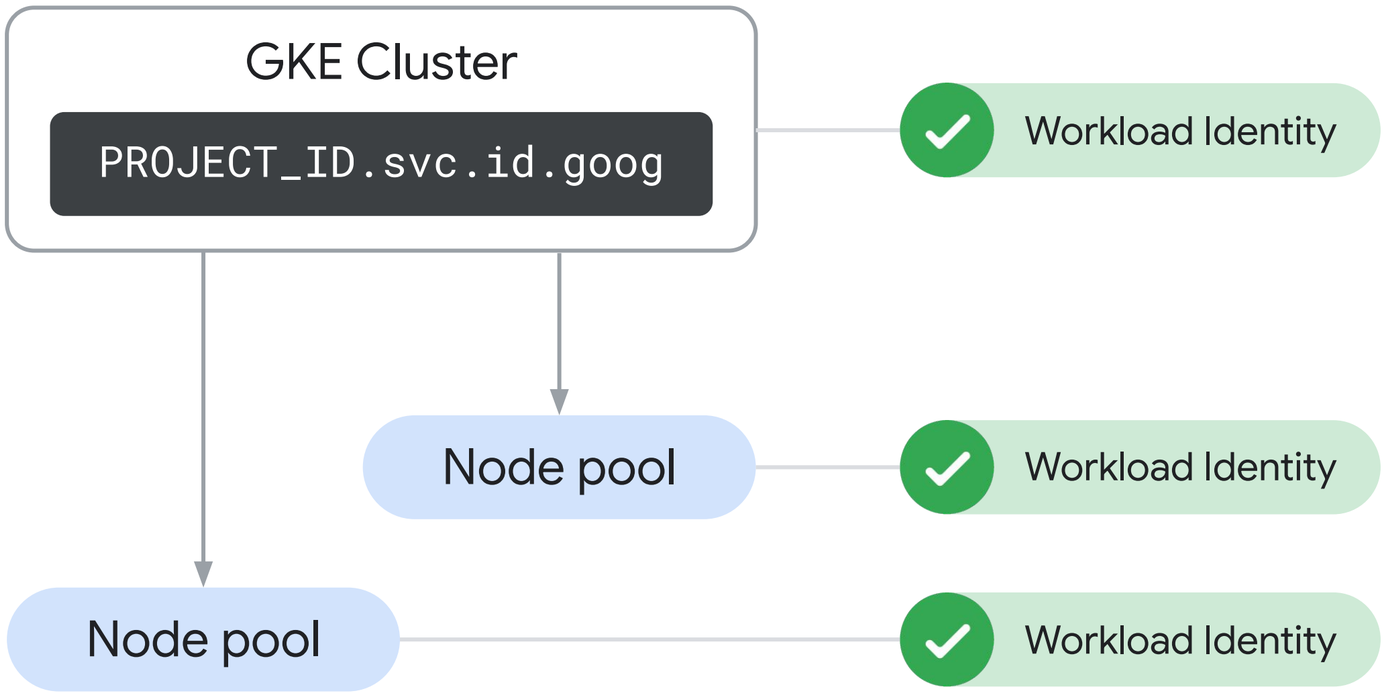
04 Kubernetes Control Plane security

05 Pod security

GKE Workload Identity simplifies how containerized applications access Google Cloud services



How do you use the Workload Identify feature?



Commands to enable Workload Identity on clusters

gcloud container clusters create

Used to enable Workload Identity on a new standard cluster

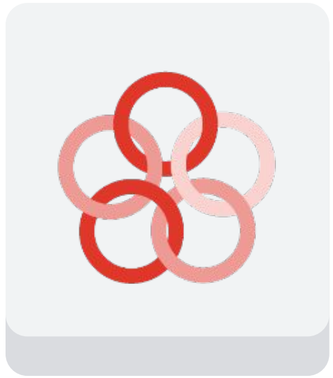
```
gcloud container clusters create MyTestCluster
--region=us-central1-a
--workload-pool=myProjectID.svc.id.goog
```

gcloud container clusters update

Used to enable Workload Identity on an existing standard cluster

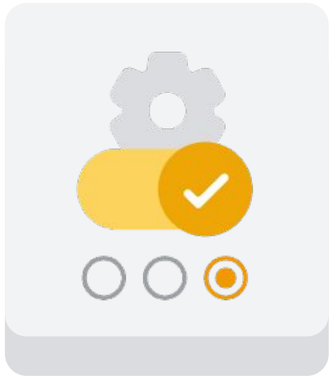
```
gcloud container clusters update MyTestCluster
--region=us-central1-a
--workload-pool=myProjectID.svc.id.goog
```

Create a new node pool before migrating applications



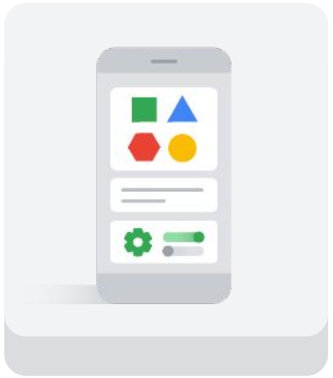
Step 1

Create a new node pool



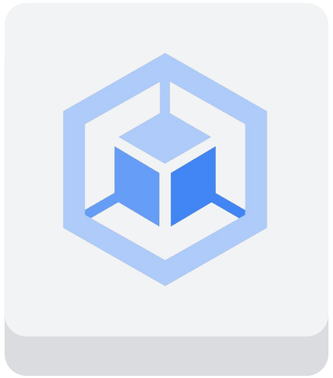
Step 2

Enable Workload Identity



Step 3

Configure application authentication



Step 4

Assign Kubernetes service account

Configuring an application to use Workload Identity

01

Create a new namespace for the service account.

```
kubectl create namespace NAMESPACE
```

02

Create a new Kubernetes ServiceAccount for your application.

```
kubectl create serviceaccount KSA_NAME --namespace NAMESPACE
```

03

Ensure the IAM service account has the required roles.

```
gcloud projects add-iam-policy-binding myProjectID \
--role="ROLE_NAME"
--member=principal://iam.googleapis.com/projects/PROJECT_NUMBER/locations/global/
workloadIdentityPools/PROJECT_ID.svc.id.goog/subject/ns/NAMESPACE/sa/KSA_NAME \
--condition=None
```

Configuring an application to use Workload Identity

04

Grant role to ServiceAccount.

```
gcloud storage buckets add-iam-policy-binding
gs://BUCKET --role=roles/storage.objectViewer \
--member=principal://iam.googleapis.com/project
s/PROJECT_NUMBER/locations/global/workloadIdent
ityPools/PROJECT_ID.svc.id.goog/subject/ns/NAME
SPACE/sa/KSA_NAME \
--condition=None
```

05

Add Workload Identity in pod manifest.

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pod
  namespace: NAMESPACE
spec:
  serviceAccountName: KSA_NAME
  containers:
  - name: test-pod
  [...]
```

06

Apply this to your cluster with a kubectl apply command.

```
kubectl apply -f WorkloadID.yaml
```

Access Control and Security in Kubernetes and GKE

01 Authentication and authorization

02 Kubernetes role-based access control

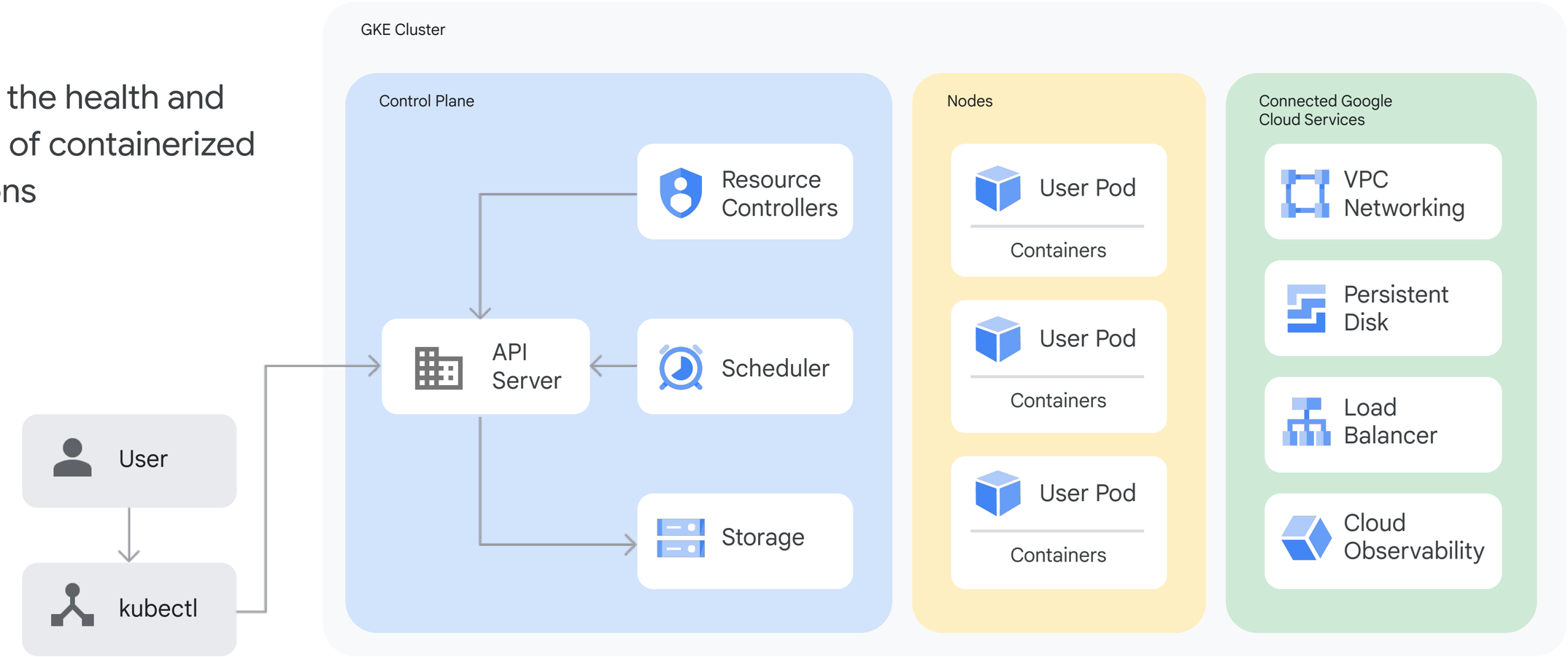
03 Workload Identity

04 Kubernetes Control Plane security

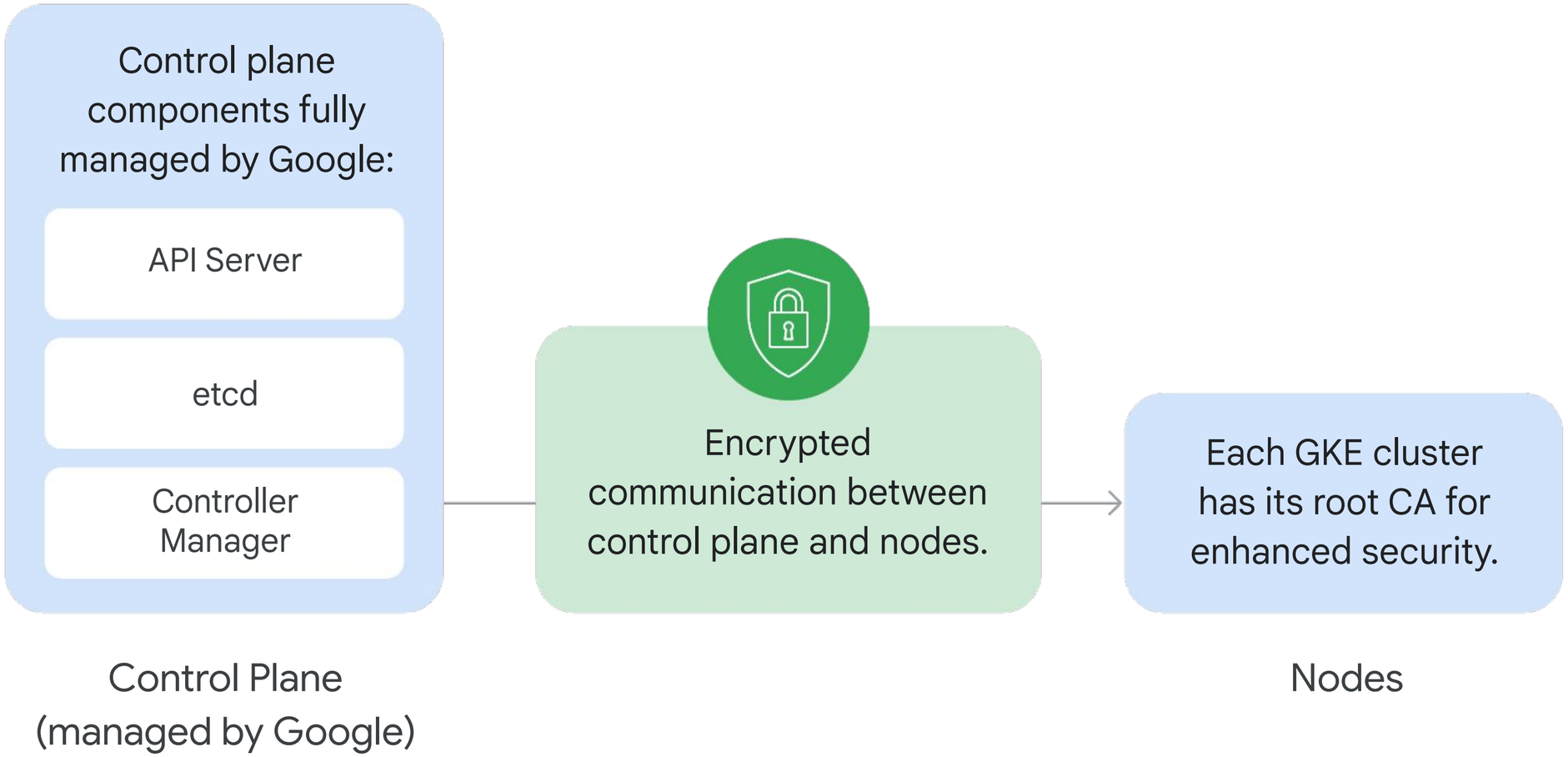
05 Pod security

The Kubernetes control plane is the brain of a cluster

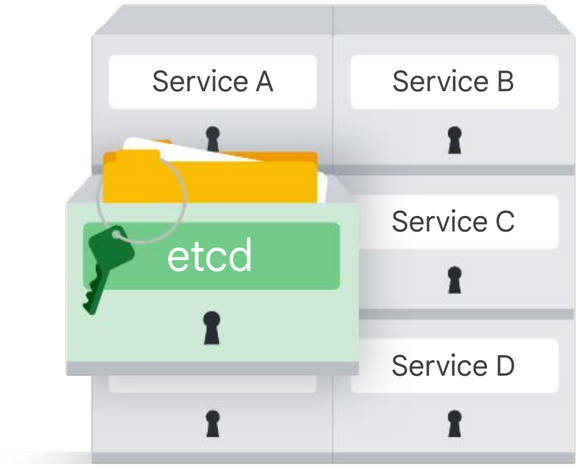
Maintains the health and operation of containerized applications

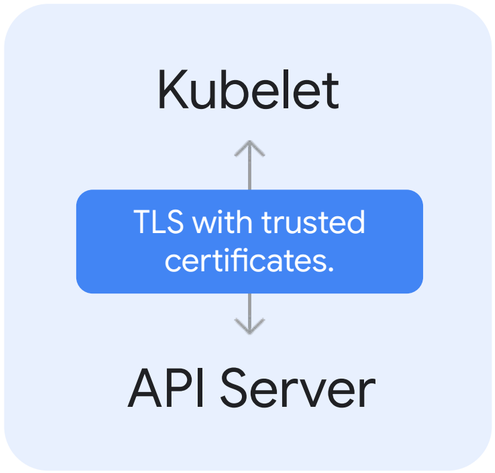


Google manages all control plane components

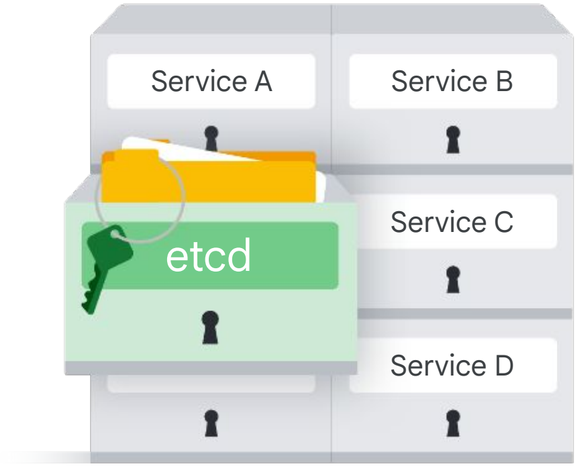


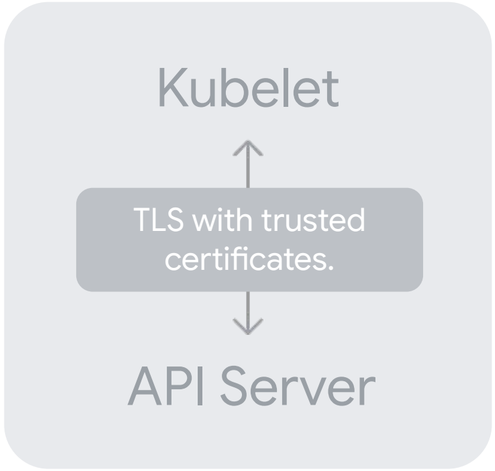
Ways to secure the GKE control plane





GKE isolates etcd databases from other cluster service

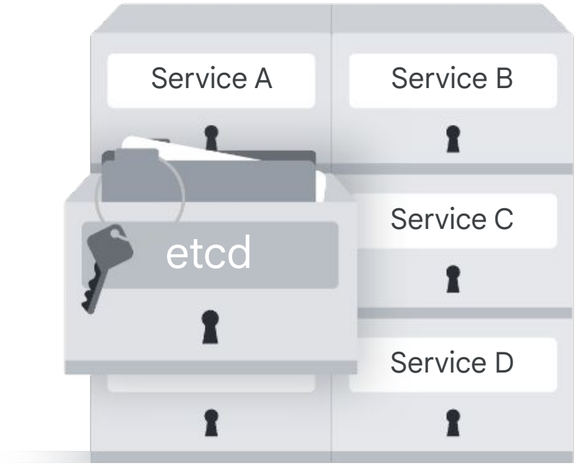


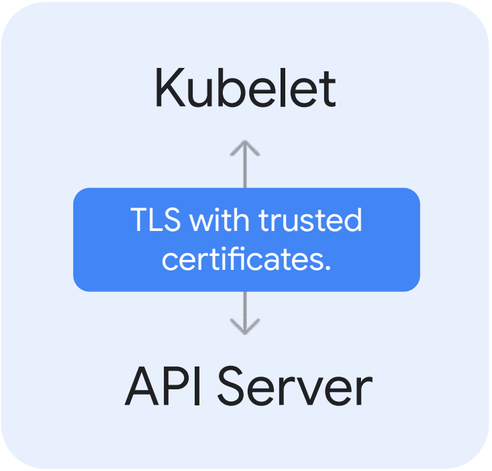


Uses a dedicated certificate authority for each cluster.

Even if a service is compromised, the trust and security of the etcd databases remain independent.

Secure communication: Kubelets and the API server

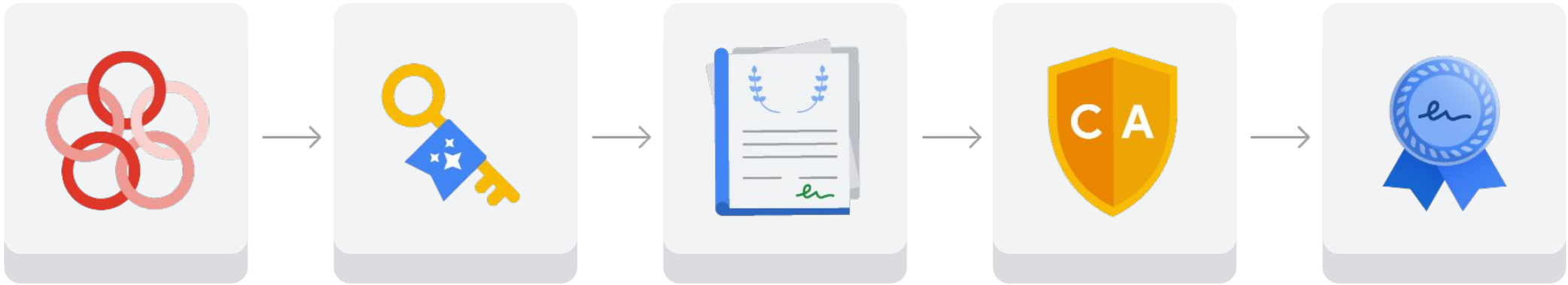




Maintain secure communication with the API server through trusted certificates issued by the cluster's root CA.

Safeguards the confidentiality and integrity of interactions between Kubelets and the API server.

How kubelets obtain certificates



New node provisioned

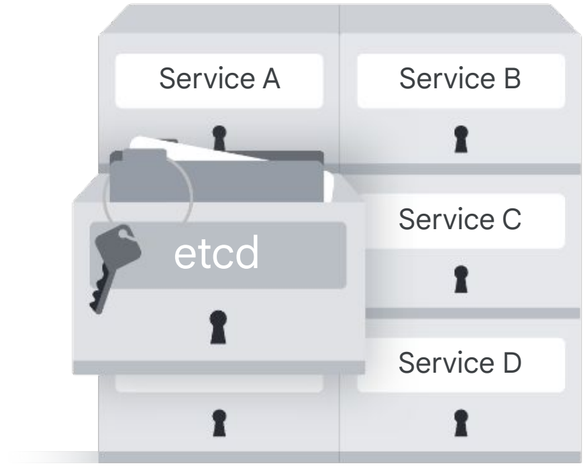
Shared secret

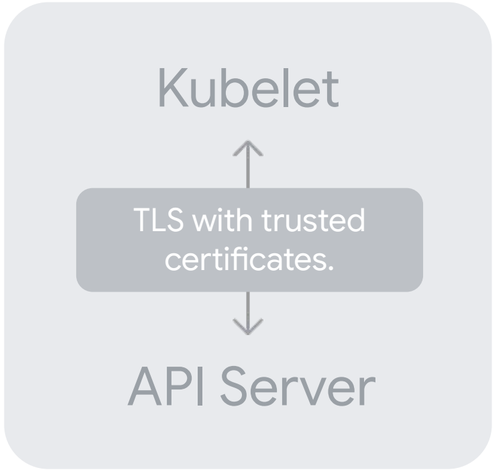
Certificate signing request (CSR)

Cluster CA

Signed certificate issued

Credential rotation means regularly rotating credentials for GKE control plane component

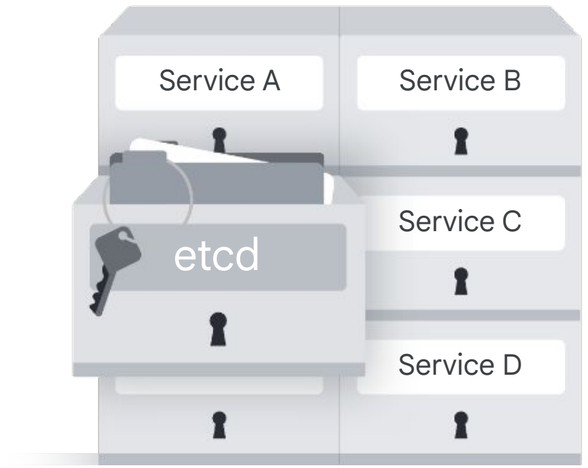


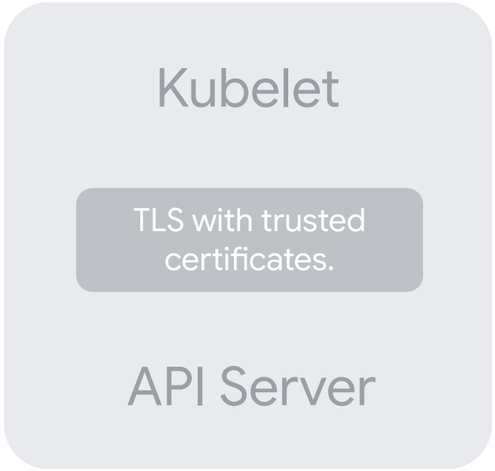


Frequency considerations:

- Balance security needs with operational overhead.
- Align rotation frequency with your organization's security policies.
- Consider sensitivity of data and assets managed by the cluster.

The credential rotation process in GKE





1. Create a new IP address for the cluster control plane.
2. Issue new certificate credentials to the control plane with the new IP address.
3. Update the nodes to use the new IP address and credentials.

4. Update all API clients outside the cluster to use the new credentials.
5. Remove the old IP address and old credentials.

Commands to help complete a credential rotation

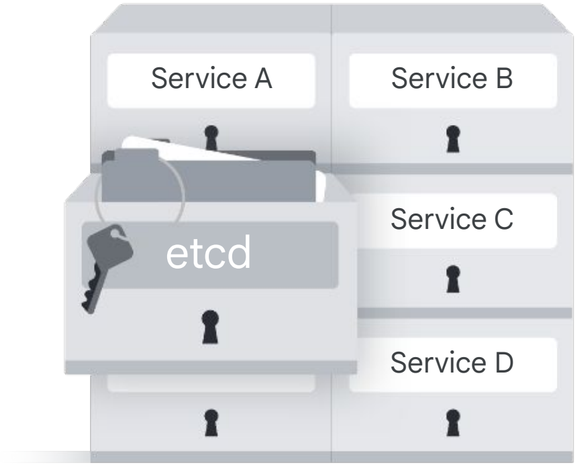
`start-credential-rotation`
gcloud command

```
gcloud container clusters update [NAME] \
    --region=[REGION_NAME] \
    --start-credential-rotation
```

`complete-credential-rotation`
gcloud command

```
gcloud container clusters update [NAME] \
    --region=[REGION_NAME] \
    --complete-credential-rotation
```

Credential rotations may disrupt the cluster API server



Kubelet

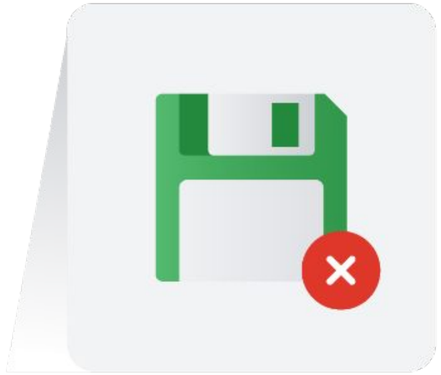
TLS with trusted certificates.

API Server

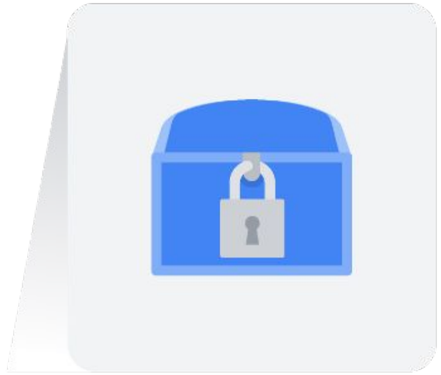
- Can cause a temporary pause in new Pod creation.
- Existing Pods continue to run without disruption.
- Schedule Pod deployments before or after credential rotation.

Secure GKE with a layered approach

Fine-tuning permissions



Disabling outdated access



Hiding sensitive data

Access Control and Security in Kubernetes and GKE

01 Authentication and authorization

02 Kubernetes role-based access control

03 Workload Identity

04 Kubernetes Control Plane security

05 Pod security

Secure Pods for a robust GKE environment

Pod Security Standards (PSS)

Predefined security configurations for Pods.

Pod Security Standards (PSS)

Privileged

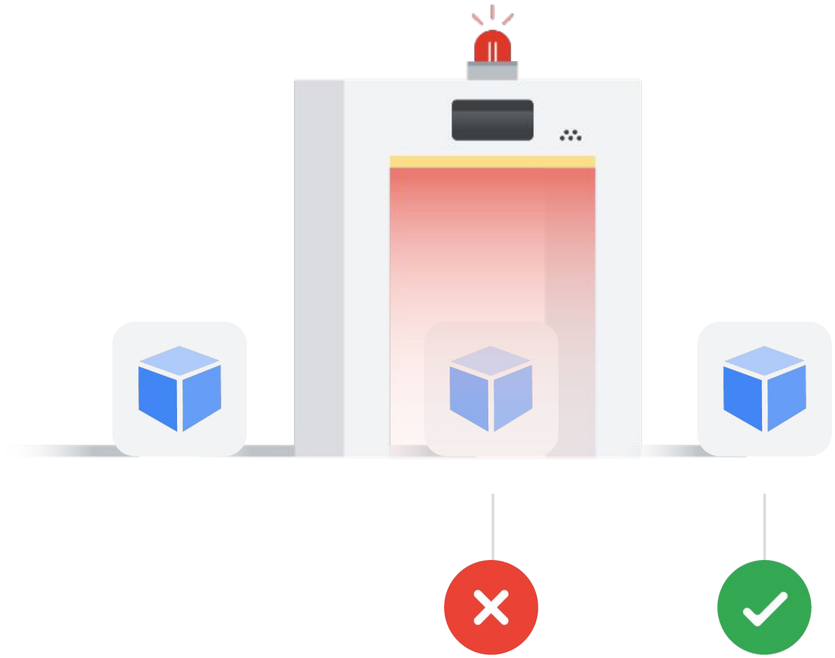
Baseline

Restricted

Range from permissive to highly restrictive.

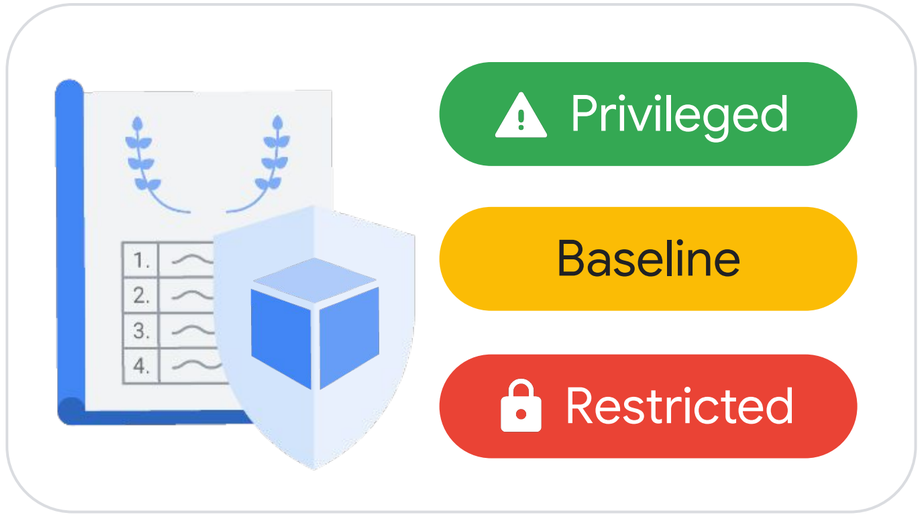
Pre-defined PSS levels

Pod Security Admission (PSA)



- Kubernetes admission controller designed for GKE.
- Enforces Pod Security Standards (PSS) in GKE clusters.
- Intercepts Pod creation requests and evaluates against namespace policies.
- Denies creation of Pods that violate security policies.

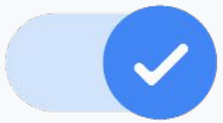
The advantages of combining PSS and PSA



PSS + PSA

- PSS makes security easier by offering ready-to-use policies.
- PSA guarantees all Pods follow PSS policies.
- Stricter Pod security measures reduce attack surfaces.
- PSS policies can help organizations meet compliance requirements.

Steps to implement PSS and PSA

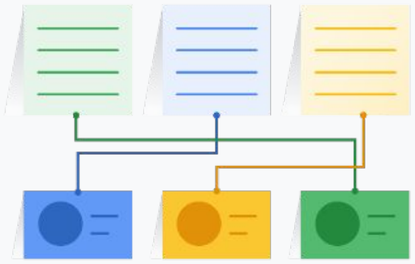


01

Enable the PodSecurityPolicy feature in your GKE cluster.

02

Define the appropriate PSS policy for each namespace.



03

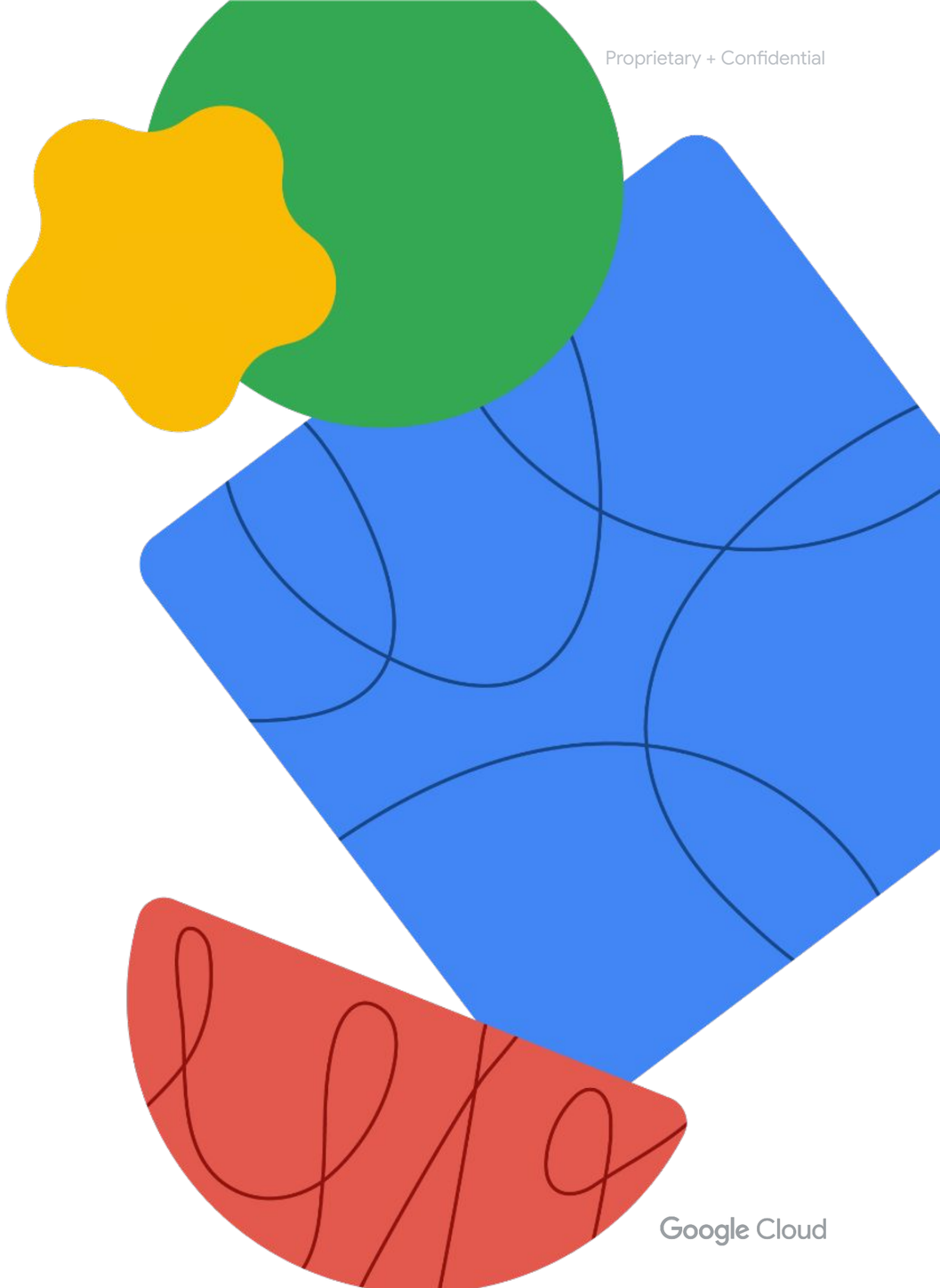
Assign the chosen PSS policies to each namespace.

04

Regularly monitor Pod security compliance.

Quiz questions

Let’s pause for a quick check in.



Quiz | Question 1

Question

In a Kubernetes pod, what type of account do containerized processes use to communicate with the kube-apiserver running on the Kubernetes cluster control plane?

A. A Google Cloud Service Account.

B. A Kubernetes Service account.

C. An IAM user account.

D. A Kubernetes normal user account.

Quiz | Question 1

Answer

In a Kubernetes pod, what type of account do containerized processes use to communicate with the kube-apiserver running on the Kubernetes cluster control plane?

A. A Google Cloud Service Account.

B. A Kubernetes Service account. 

C. An IAM user account.

D. A Kubernetes normal user account.

Quiz | Question 2

Question

You need to implement account controls to grant junior admin users the ability to view details about production clusters and applications, and to fully manage and test lab resources inside your GKE cluster environments. Which account control mechanism is best suited for this granular control?

A. Radius

B. IAM

C. OAuth2

D. Kubernetes RBAC

Quiz | Question 2

Answer

You need to implement account controls to grant junior admin users the ability to view details about production clusters and applications, and to fully manage and test lab resources inside your GKE cluster environments. Which account control mechanism is best suited for this granular control?

A. Radius

B. IAM

C. OAuth2

D. Kubernetes RBAC 

Quiz | Question 3

Question

Workload Identity is a feature that bridges the gap between what two Google Cloud services?

A. Kubernetes service accounts and IAM service accounts

B. Google Kubernetes Engine and IAM

C. Google Compute Engine and IAM

D. Kubernetes and IAM

Quiz | Question 3

Answer

Workload Identity is a feature that bridges the gap between what two Google Cloud services?

A. Kubernetes service accounts and IAM service accounts

B. Google Kubernetes Engine and IAM

C. Google Compute Engine and IAM

D. Kubernetes and IAM

Quiz | Question 4

Question

How can you use Pod securityContexts to control security? Choose all responses that are correct (2 correct responses).

A. Limit access to some Linux capabilities, like granting certain privileges to a process, but not to all root user privileges.

B. Configure audit logging to redirect all Pod logs to an external webhook backend for persistent event auditing.

C. Limit access to Google Cloud services and resources to prevent Pod users from accessing Cloud Storage objects.

D. Enable AppArmor, which uses security profiles to restrict individual program actions

Quiz | Question 4

Answer

How can you use Pod securityContexts to control security? Choose all responses that are correct (2 correct responses).

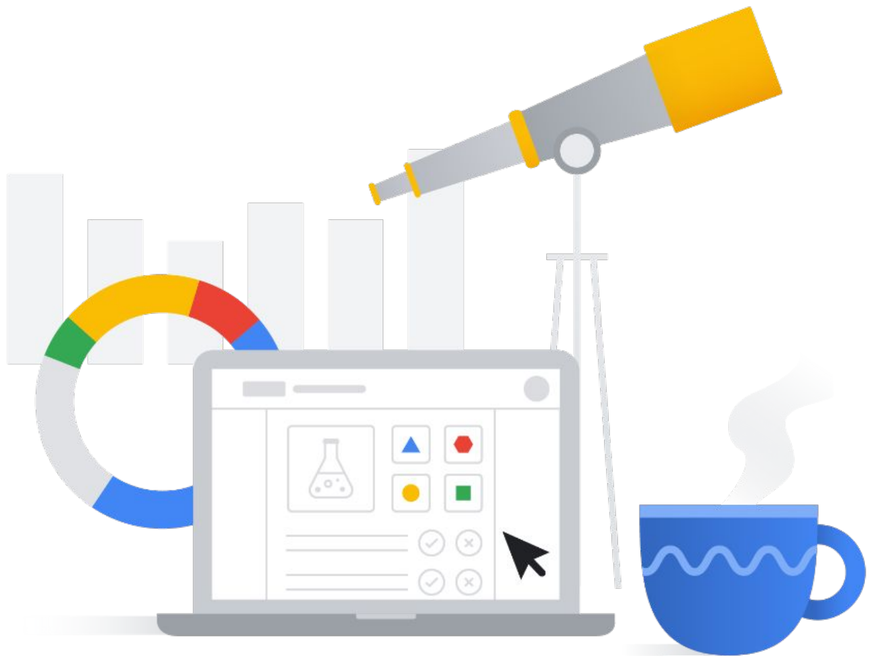
A. Limit access to some Linux capabilities, like granting certain privileges to a process, but not to all root user privileges. 

B. Configure audit logging to redirect all Pod logs to an external webhook backend for persistent event auditing.

C. Limit access to Google Cloud services and resources to prevent Pod users from accessing Cloud Storage objects.

D. Enable AppArmor, which uses security profiles to restrict individual program actions. 

Lab: Securing Google Kubernetes Engine with IAM and Pod Security Admission



01 Use IAM to control GKE access.

02 Create and use Pod security admission to control Pod creation.

03 Perform IP address and credential rotation.