Kubernetes CIS Benchmark Report

gke_sigma-tractor-458905-e0_us-central1-c_cluster-1

Date: 21-05-2025 15:13:58

Summary total

Status	Count
PASS	12
FAIL	1
WARN	55
INFO	0

Summary controlplane

Status	Count
PASS	0
FAIL	0
WARN	1
INFO	0

2 Control Plane Configuration

2.1 Authentication and Authorization

Test ID: 2.1.1 Status: WARN

2.1.1 Client certificate authentication should not be used for users (Manual)

== Remediations controlplane ==
2.1.1 Alternative mechanisms provided by Kubernetes such as the use of OIDC should be implemented in place of client certificates.

You can remediate the availability of client certificates in your GKE cluster. See Recommendation 5.8.1.

Summary node

Status	Count

PASS	12
FAIL	1
WARN	0
INFO	0

3 Worker Node Security Configuration

3.1 Worker Node Configuration Files

Test ID: 3.1.1 Status: PASS

3.1.1 Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Manual)

Test ID: 3.1.2 Status: PASS

3.1.2 Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)

Test ID: 3.1.3 Status: PASS

3.1.3 Ensure that the kubelet configuration file has permissions set to 600 (Manual)

Test ID: 3.1.4 Status: PASS

3.1.4 Ensure that the kubelet configuration file ownership is set to root:root (Manual)

3.2 Kubelet

Test ID: 3.2.1 Status: PASS

3.2.1 Ensure that the Anonymous Auth is Not Enabled (Automated)

Test ID: 3.2.2 Status: PASS

3.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)

Test ID: 3.2.3 Status: PASS

3.2.3 Ensure that a Client CA File is Configured (Automated)

Test ID: 3.2.4 Status: PASS

3.2.4 Ensure that the --read-only-port argument is disabled (Automated)

Test ID: 3.2.5 Status: PASS

3.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)

Test ID: 3.2.6 Status: PASS

3.2.6 Ensure that the --make-iptables-util-chains argument is set to true (Automated)

Test ID: 3.2.7 Status: FAIL

3.2.7 Ensure that the --eventRecordQPS argument is set to 0 or a level which ensures appropriate event capture (Automated)

Test ID: 3.2.8 Status: PASS

3.2.8 Ensure that the --rotate-certificates argument is not present or is set to true (Automated)

Test ID: 3.2.9 Status: PASS

3.2.9 Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)

== Remediations node ==
3.2.7 If using a Kubelet config file, edit the file to set eventRecordQPS: to an appropriate level.

If using command line arguments, edit the kubelet service file /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
on each worker node and set the below parameter in KUBELET_SYSTEM_PODS_ARGS variable.

Based on your system, restart the kubelet service. For example:

systemctl daemon-reload
systemctl restart kubelet.service

Summary policies

Status	Count
PASS	0
FAIL	0
WARN	20
INFO	0

4 Kubernetes Policies

4.1 RBAC and Service Accounts

Test ID: 4.1.1 Status: WARN

4.1.1 Ensure that the cluster-admin role is only used where required (Automated)

Test ID: 4.1.2 Status: WARN

4.1.2 Minimize access to secrets (Automated)

Test ID: 4.1.3 Status: WARN

4.1.3 Minimize wildcard use in Roles and ClusterRoles (Automated)

Test ID: 4.1.4 Status: WARN

4.1.4 Ensure that default service accounts are not actively used (Automated)

Test ID: 4.1.5 Status: WARN

4.1.5 Ensure that Service Account Tokens are only mounted where necessary (Automated)

Test ID: 4.1.6 Status: WARN

4.1.6 Avoid use of system:masters group (Automated)

Test ID: 4.1.7 Status: WARN

4.1.7 Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)

Test ID: 4.1.8 Status: WARN

4.1.8 Avoid bindings to system:anonymous (Automated)

Test ID: 4.1.9 Status: WARN

4.1.9 Avoid non-default bindings to system:unauthenticated (Automated)

Test ID: 4.1.10 Status: WARN

4.1.10 Avoid non-default bindings to system:authenticated (Automated)

4.2 Pod Security Standards

Test ID: 4.2.1 Status: WARN

4.2.1 Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces. (Manual)

4.3 Network Policies and CNI

Test ID: 4.3.1 Status: WARN

4.3.1 Ensure that the CNI in use supports Network Policies (Manual)

Test ID: 4.3.2 Status: WARN

4.3.2 Ensure that all Namespaces have Network Policies defined (Automated)

4.4 Secrets Management

Test ID: 4.4.1 Status: WARN

4.4.1 Prefer using secrets as files over secrets as environment variables (Automated)

Test ID: 4.4.2 Status: WARN

4.4.2 Consider external secret storage (Manual)

4.5 Extensible Admission Control

Test ID: 4.5.1 Status: WARN

4.5.1 Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)

4.6 General Policies

Test ID: 4.6.1 Status: WARN

4.6.1 Create administrative boundaries between resources using namespaces (Manual)

Test ID: 4.6.2 Status: WARN

4.6.2 Ensure that the seccomp profile is set to RuntimeDefault in your pod definitions (Automated)

Test ID: 4.6.3 Status: WARN

4.6.3 Apply Security Context to Your Pods and Containers (Manual)

Test ID: 4.6.4 Status: WARN

4.6.4 The default namespace should not be used (Automated)

== Remediations policies ==
4.1.1 Identify all clusterrolebindings to the cluster-admin role. Check if they are used and if they need this role or if they could use a role with fewer privileges.

Where possible, first bind users to a lower privileged role and then remove the clusterrolebinding to the cluster-admin role :

kubectl delete clusterrolebinding [name]

4.1.2 Where possible, remove get, list and watch access to secret objects in the cluster.

4.1.3 Where possible replace any use of wildcards in clusterroles and roles with specific objects or actions.

4.1.4 Create explicit service accounts wherever a Kubernetes workload requires specific access to the Kubernetes API server.

Modify the configuration of each default service account to include this value

automountServiceAccountToken: false

4.1.5 Modify the definition of pods and service accounts which do not need to mount service account tokens to disable it.

4.1.6 Remove the system:masters group from all users in the cluster.

4.1.7 Where possible, remove the impersonate, bind and escalate rights from subjects.

4.1.8 Identify all clusterrolebindings and rolebindings to the user system:anonymous. Check if they are used and review the permissions associated with the binding using the commands in the Audit section above or refer to GKE documentation (https://cloud.google.com/kubernetes-engine/docs/best-practices/rbac#detect-prevent-default).

Strongly consider replacing unsafe bindings with an authenticated, user-defined group. Where possible, bind to non-default, user-defined groups with least-privilege roles.

If there are any unsafe bindings to the user system:anonymous, proceed to delete them after consideration for cluster operations with only necessary, safer bindings.

kubectl delete clusterrolebinding [CLUSTER_ROLE_BINDING_NAME]
kubectl delete rolebinding [ROLE_BINDING_NAME] --namespace [ROLE_BINDING_NAMESPACE]

4.1.9 Identify all non-default clusterrolebindings and rolebindings to the group system:unauthenticated. Check if they are used and review the permissions associated with the binding using the commands in the Audit section above or refer to GKE documentation (https://cloud.google.com/kubernetes-engine/docs/best-practices/rbac#detect-prevent-default).

Strongly consider replacing non-default, unsafe bindings with an authenticated, user-defined group. Where possible, bind to non-default, user-defined groups with least-privilege roles.

If there are any non-default, unsafe bindings to the group system:unauthenticated, proceed to delete them after consideration for cluster operations with only necessary, safer bindings.

kubectl delete clusterrolebinding [CLUSTER_ROLE_BINDING_NAME]
kubectl delete rolebinding [ROLE_BINDING_NAME] --namespace [ROLE_BINDING_NAMESPACE]

4.1.10 Identify all non-default clusterrolebindings and rolebindings to the group system:authenticated. Check if they are used and review the permissions associated with the binding using the commands in the Audit section above or refer to GKE documentation.

Strongly consider replacing non-default, unsafe bindings with an authenticated, user-defined group. Where possible, bind to non-default, user-defined groups with least-privilege roles.

If there are any non-default, unsafe bindings to the group system:authenticated, proceed to delete them after consideration for cluster operations with only necessary, safer bindings.

kubectl delete clusterrolebinding [CLUSTER_ROLE_BINDING_NAME]
kubectl delete rolebinding [ROLE_BINDING_NAME] --namespace [ROLE_BINDING_NAMESPACE]

4.2.1 Ensure that Pod Security Admission is in place for every namespace which contains user workloads.
Run the following command to enforce the Baseline profile in a namespace:

kubectl label namespace pod-security.kubernetes.io/enforce=baseline

4.3.1 To use a CNI plugin with Network Policy, enable Network Policy in GKE, and the CNI plugin

will be updated. See Recommendation 5.6.7.

4.3.2 Follow the documentation and create NetworkPolicy objects as needed.
See: https://cloud.google.com/kubernetes-engine/docs/how-to/network-policy#creating_a_network_policy for more information.

4.4.1 if possible, rewrite application code to read secrets from mounted secret files, rather than from environment variables.

4.4.2 Refer to the secrets management options offered by your cloud provider or a third-party secrets management solution.

4.5.1 Follow the Kubernetes documentation and setup image provenance.
Also see recommendation 5.10.4.

4.6.1 Follow the documentation and create namespaces for objects in your deployment as you need them.

4.6.2 Use security context to enable the RuntimeDefault seccomp profile in your pod definitions. An example is as below:

{
"namespace": "kube-system",
"name": "metrics-server-v0.7.0-dbcc8ddf6-gz7d4",
"seccompProfile": "RuntimeDefault"
}

4.6.3 Follow the Kubernetes documentation and apply security contexts to your pods. For a suggested list of security contexts, you may refer to the CIS Google Container-Optimized OS Benchmark.

4.6.4 Ensure that namespaces are created to allow for appropriate segregation of Kubernetes resources and that all new resources are created in a specific namespace.

Summary managedservices

Status	Count
PASS	0
FAIL	0
WARN	34
INFO	0

5 Managed Services

5.1 Image Registry and Image Scanning

Test ID: 5.1.1 Status: WARN

5.1.1 Ensure Image Vulnerability Scanning is enabled (Automated)

Test ID: 5.1.2 Status: WARN

5.1.2 Minimize user access to Container Image repositories (Manual)

Test ID: 5.1.3 Status: WARN

5.1.3 Minimize cluster access to read-only for Container Image repositories (Manual)

Test ID: 5.1.4 Status: WARN

5.1.4 Ensure only trusted container images are used (Manual)

5.2 Identity and Access Management (IAM)

Test ID: 5.2.1 Status: WARN

5.2.1 Ensure GKE clusters are not running using the Compute Engine default service account (Automated))

Test ID: 5.2.2 Status: WARN

5.2.2 Prefer using dedicated GCP Service Accounts and Workload Identity (Manual)

5.3 Cloud Key Management Service (Cloud KMS)

Test ID: 5.3.1 Status: WARN

5.3.1 Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS (Automated)

5.4 Node Metadata

Test ID: 5.4.1 Status: WARN

5.4.1 Ensure the GKE Metadata Server is Enabled (Automated)

5.5 Node Configuration and Maintenance

Test ID: 5.5.1 Status: WARN

5.5.1 Ensure Container-Optimized OS (cos_containerd) is used for GKE node images (Automated)

Test ID: 5.5.2 Status: WARN

5.5.2 Ensure Node Auto-Repair is enabled for GKE nodes (Automated)

Test ID: 5.5.3 Status: WARN

5.5.3 Ensure Node Auto-Upgrade is enabled for GKE nodes (Automated)

Test ID: 5.5.4 Status: WARN

5.5.4 When creating New Clusters - Automate GKE version management using Release Channels (Automated)

Test ID: 5.5.5 Status: WARN

5.5.5 Ensure Shielded GKE Nodes are Enabled (Automated)

Test ID: 5.5.6 Status: WARN

5.5.6 Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled (Automated)

Test ID: 5.5.7 Status: WARN

5.5.7 Ensure Secure Boot for Shielded GKE Nodes is Enabled (Automated)

5.6 Cluster Networking

Test ID: 5.6.1 Status: WARN

5.6.1 Enable VPC Flow Logs and Intranode Visibility (Automated)

Test ID: 5.6.2 Status: WARN

5.6.2 Ensure use of VPC-native clusters (Automated)

Test ID: 5.6.3 Status: WARN

5.6.3 Ensure Control Plane Authorized Networks is Enabled (Automated)

Test ID: 5.6.4 Status: WARN

5.6.4 Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Manual)

Test ID: 5.6.5 Status: WARN

5.6.5 Ensure clusters are created with Private Nodes (Manual)

Test ID: 5.6.6 Status: WARN

5.6.6 Consider firewalling GKE worker nodes (Manual)

Test ID: 5.6.7 Status: WARN

5.6.7 Ensure use of Google-managed SSL Certificates (Automated)

5.7 Logging

Test ID: 5.7.1 Status: WARN

5.7.1 Ensure Logging and Cloud Monitoring is Enabled (Automated)

Test ID: 5.7.2 Status: WARN

5.7.2 Enable Linux auditd logging (Manual)

5.8 Authentication and Authorization

Test ID: 5.8.1 Status: WARN

5.8.1 Ensure authentication using Client Certificates is Disabled (Automated)

Test ID: 5.8.2 Status: WARN

5.8.2 Manage Kubernetes RBAC users with Google Groups for GKE (Manual)

Test ID: 5.8.3 Status: WARN

5.8.3 Ensure Legacy Authorization (ABAC) is Disabled (Automated)

5.9 Storage

Test ID: 5.9.1 Status: WARN

5.9.1 Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD) (Manual)

Test ID: 5.9.2 Status: WARN

5.9.2 Enable Customer-Managed Encryption Keys (CMEK) for Boot Disks (Automated)

5.10 Other Cluster Configurations

Test ID: 5.10.1 Status: WARN

5.10.1 Ensure Kubernetes Web UI is Disabled (Automated)

Test ID: 5.10.2 Status: WARN

5.10.2 Ensure that Alpha clusters are not used for production workloads (Automated)

Test ID: 5.10.3 Status: WARN

5.10.3 Consider GKE Sandbox for running untrusted workloads (Manual)

Test ID: 5.10.4 Status: WARN

5.10.4 Ensure use of Binary Authorization (Automated)

Test ID: 5.10.5 Status: WARN

5.10.5 Enable Security Posture (Manual)

== Remediations managedservices ==
5.1.1 For Images Hosted in GCR:
Using Command Line:

gcloud services enable containeranalysis.googleapis.com

For Images Hosted in AR:
Using Command Line:

gcloud services enable containerscanning.googleapis.com

5.1.2 For Images Hosted in AR:
Using Command Line:

gcloud artifacts repositories set-iam-policy <repository-name> <path-to-policy-file> \
--location <repository-location>

To learn how to configure policy files see: https://cloud.google.com/artifact-registry/docs/access-control#grant

For Images Hosted in GCR:
Using Command Line:
To change roles at the GCR bucket level:
Firstly, run the following if read permissions are required:

gsutil iam ch <type>:<email_address>:objectViewer gs://artifacts.<project_id>.appspot.com

Then remove the excessively privileged role (Storage Admin / Storage Object Admin / Storage Object Creator) using:

gsutil iam ch -d <type>:<email_address>:<role> gs://artifacts.<project_id>.appspot.com

where:
<type> can be one of the following:
user, if the <email_address> is a Google account.
serviceAccount, if <email_address> specifies a Service account.
<email_address> can be one of the following:
a Google account (for example, someone@example.com).
a Cloud IAM service account.

To modify roles defined at the project level and subsequently inherited within the GCR bucket, or the Service Account User role, extract the IAM policy file, modify it accordingly and apply it using:

gcloud projects set-iam-policy <project_id> <policy_file>

5.1.3 For Images Hosted in AR:
Using Command Line:
Add artifactregistry.reader role

gcloud artifacts repositories add-iam-policy-binding <repository> \
--location=<repository-location> \
--member='serviceAccount:<email-address>' \
--role='roles/artifactregistry.reader'

Remove any roles other than artifactregistry.reader

gcloud artifacts repositories remove-iam-policy-binding <repository> \
--location <repository-location> \
--member='serviceAccount:<email-address>' \
--role='<role-name>'

For Images Hosted in GCR:
For an account explicitly granted to the bucket:
Firstly add read access to the Kubernetes Service Account:

gsutil iam ch <type>:<email_address>:objectViewer gs://artifacts.<project_id>.appspot.com

where:
<type> can be one of the following:
user, if the <email_address> is a Google account.
serviceAccount, if <email_address> specifies a Service account.
<email_address> can be one of the following:
a Google account (for example, someone@example.com).
a Cloud IAM service account.

Then remove the excessively privileged role (Storage Admin / Storage Object Admin / Storage Object Creator) using:

gsutil iam ch -d <type>:<email_address>:<role> gs://artifacts.<project_id>.appspot.com

For an account that inherits access to the GCR Bucket through Project level permissions, modify the Projects IAM policy file accordingly, then upload it using:

gcloud projects set-iam-policy <project_id> <policy_file>

5.1.4 Using Command Line:
Update the cluster to enable Binary Authorization:

gcloud container cluster update <cluster_name> --enable-binauthz

Create a Binary Authorization Policy using the Binary Authorization Policy Reference: https://cloud.google.com/binary-authorization/docs/policy-yaml-reference for guidance.

Import the policy file into Binary Authorization:

gcloud container binauthz policy import <yaml_policy>

5.2.1 Using Command Line:
To create a minimally privileged service account:

gcloud iam service-accounts create <node_sa_name> \
--display-name "GKE Node Service Account"
export NODE_SA_EMAIL=gcloud iam service-accounts list \
--format='value(email)' --filter='displayName:GKE Node Service Account'

Grant the following roles to the service account:

export PROJECT_ID=gcloud config get-value project
gcloud projects add-iam-policy-binding <project_id> --member \
serviceAccount:<node_sa_email> --role roles/monitoring.metricWriter
gcloud projects add-iam-policy-binding <project_id> --member \
serviceAccount:<node_sa_email> --role roles/monitoring.viewer
gcloud projects add-iam-policy-binding <project_id> --member \
serviceAccount:<node_sa_email> --role roles/logging.logWriter

To create a new Node pool using the Service account, run the following command:

gcloud container node-pools create <node_pool> \
--service-account=<sa_name>@<project_id>.iam.gserviceaccount.com \
--cluster=<cluster_name> --zone <compute_zone>

Note: The workloads will need to be migrated to the new Node pool, and the old node pools that use the default service account should be deleted to complete the remediation.

5.2.2 Using Command Line:

gcloud container clusters update <cluster_name> --zone <cluster_zone> \
--workload-pool <project_id>.svc.id.goog

Note that existing Node pools are unaffected. New Node pools default to --workload-metadata-from-node=GKE_METADATA_SERVER.

Then, modify existing Node pools to enable GKE_METADATA_SERVER:

gcloud container node-pools update <node_pool_name> --cluster <cluster_name> \
--zone <cluster_zone> --workload-metadata=GKE_METADATA

Workloads may need to be modified in order for them to use Workload Identity as described within: https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity. Also consider the effects on the availability of hosted workloads as Node pools are updated. It may be more appropriate to create new Node Pools.

5.3.1 To create a key:
Create a key ring:

gcloud kms keyrings create <ring_name> --location <location> --project \
<key_project_id>

Create a key:

gcloud kms keys create <key_name> --location <location> --keyring <ring_name> \
--purpose encryption --project <key_project_id>

Grant the Kubernetes Engine Service Agent service account the Cloud KMS CryptoKey Encrypter/Decrypter role:

gcloud kms keys add-iam-policy-binding <key_name> --location <location> \
--keyring <ring_name> --member serviceAccount:<service_account_name> \
--role roles/cloudkms.cryptoKeyEncrypterDecrypter --project <key_project_id>

To create a new cluster with Application-layer Secrets Encryption:

gcloud container clusters create <cluster_name> --cluster-version=latest \
--zone <zone> \
--database-encryption-key
projects/<key_project_id>/locations/<location>/keyRings/<ring_name>/cryptoKeys/<key_name> \
--project <cluster_project_id>

To enable on an existing cluster:

gcloud container clusters update <cluster_name> --zone <zone> \
--database-encryption-key
projects/<key_project_id>/locations/<location>/keyRings/<ring_name>/cryptoKeys/<key_name> \
--project <cluster_project_id>

5.4.1 Using Command Line:

gcloud container clusters update <cluster_name> --identity-namespace=<project_id>.svc.id.goog

Note that existing Node pools are unaffected. New Node pools default to --workload-metadata-from-node=GKE_METADATA_SERVER.

To modify an existing Node pool to enable GKE Metadata Server:

gcloud container node-pools update <node_pool_name> --cluster=<cluster_name> \
--workload-metadata-from-node=GKE_METADATA_SERVER

Workloads may need modification in order for them to use Workload Identity as described within: https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity.

5.5.1 Using Command Line:
To set the node image to cos for an existing cluster's Node pool:

gcloud container clusters upgrade <cluster_name> --image-type cos_containerd \
--zone <compute_zone> --node-pool <node_pool_name>

5.5.2 Using Command Line:
To enable node auto-repair for an existing cluster's Node pool:

gcloud container node-pools update <node_pool_name> --cluster <cluster_name> \
--zone <compute_zone> --enable-autorepair

5.5.3 Using Command Line:
To enable node auto-upgrade for an existing cluster's Node pool, run the following command:

gcloud container node-pools update <node_pool_name> --cluster <cluster_name> \
--zone <cluster_zone> --enable-autoupgrade

5.5.4 Using Command Line:
Create a new cluster by running the following command:

gcloud container clusters create <cluster_name> --zone <cluster_zone> \
--release-channel <release_channel>

where <release_channel> is stable or regular, according to requirements.

5.5.5 Using Command Line:
To migrate an existing cluster, the flag --enable-shielded-nodes needs to be specified in the cluster update command:

gcloud container clusters update <cluster_name> --zone <cluster_zone> \

--enable-shielded-nodes

5.5.6 Using Command Line:
To create a Node pool within the cluster with Integrity Monitoring enabled, run the following command:

gcloud container node-pools create <node_pool_name> --cluster <cluster_name> \
--zone <compute_zone> --shielded-integrity-monitoring

Workloads from existing non-conforming Node pools will need to be migrated to the newly created Node pool, then delete non-conforming Node pools to complete the remediation

5.5.7 Using Command Line:
To create a Node pool within the cluster with Secure Boot enabled, run the following command:

gcloud container node-pools create <node_pool_name> --cluster <cluster_name> \
--zone <compute_zone> --shielded-secure-boot

Workloads will need to be migrated from existing non-conforming Node pools to the newly created Node pool, then delete the non-conforming pools.

5.6.1 Using Command Line:
1. Find the subnetwork name associated with the cluster.

gcloud container clusters describe <cluster_name> \
--region <cluster_region> - -format json | jq '.subnetwork'

2. Update the subnetwork to enable VPC Flow Logs.
gcloud compute networks subnets update <subnet_name> --enable-flow-logs

5.6.2 Using Command Line:
To enable Alias IP on a new cluster, run the following command:

gcloud container clusters create <cluster_name> --zone <compute_zone> \
--enable-ip-alias

If using Autopilot configuration mode:

gcloud container clusters create-auto <cluster_name> \
--zone <compute_zone>

5.6.3 Using Command Line:
To enable Control Plane Authorized Networks for an existing cluster, run the following command:

gcloud container clusters update <cluster_name> --zone <compute_zone> \
--enable-master-authorized-networks

Along with this, you can list authorized networks using the --master-authorized-networks flag which contains a list of up to 20 external networks that are allowed to connect to your cluster's control plane through HTTPS. You provide these networks as a comma-separated list of addresses in CIDR notation (such as 90.90.100.0/24).

5.6.4 Using Command Line:
Create a cluster with a Private Endpoint enabled and Public Access disabled by including the --enable-private-endpoint flag within the cluster create command:

gcloud container clusters create <cluster_name> --enable-private-endpoint

Setting this flag also requires the setting of --enable-private-nodes, --enable-ip-alias and --master-ipv4-cidr=<master_cidr_range>.

5.6.5 Using Command Line:
To create a cluster with Private Nodes enabled, include the --enable-private-nodes flag within the cluster create command:

gcloud container clusters create <cluster_name> --enable-private-nodes

Setting this flag also requires the setting of --enable-ip-alias and --master-ipv4-cidr=<master_cidr_range>.

5.6.6 Using Command Line:
Use the following command to generate firewall rules, setting the variables as appropriate:

gcloud compute firewall-rules create <firewall_rule_name> \
--network <network> --priority <priority> --direction <direction> \
--action <action> --target-tags <tag> \
--target-service-accounts <service_account> \
--source-ranges <source_cidr_range> --source-tags <source_tags> \
--source-service-accounts <source_service_account> \
--destination-ranges <destination_cidr_range> --rules <rules>

5.6.7 If services of type:LoadBalancer are discovered, consider replacing the Service with an Ingress.

To configure the Ingress and use Google-managed SSL certificates, follow the instructions as listed at: https://cloud.google.com/kubernetes-engine/docs/how-to/managed-certs.

5.7.1 To enable Logging for an existing cluster, run the following command:
gcloud container clusters update <cluster_name> --zone <compute_zone> \
--logging=<components_to_be_logged>

See https://cloud.google.com/sdk/gcloud/reference/container/clusters/update#--logging for a list of available components for logging.

To enable Cloud Monitoring for an existing cluster, run the following command:
gcloud container clusters update <cluster_name> --zone <compute_zone> \
--monitoring=<components_to_be_logged>

See https://cloud.google.com/sdk/gcloud/reference/container/clusters/update#--monitoring for a list of available components for Cloud Monitoring.

5.7.2 Using Command Line:
Download the example manifests:

curl
https://raw.githubusercontent.com/GoogleCloudPlatform/k8s-node-tools/master/os-audit/cos-auditd-logging.yaml > cos-auditd-logging.yaml

Edit the example manifests if needed. Then, deploy them:
kubectl apply -f cos-auditd-logging.yaml

Verify that the logging Pods have started. If a different Namespace was defined in the manifests, replace cos-auditd with the name of the namespace being used:
kubectl get pods --namespace=cos-auditd

5.8.1 Using Command Line:
Create a new cluster without a Client Certificate:
gcloud container clusters create [CLUSTER_NAME] \
--no-issue-client-certificate

5.8.2 Using Command Line:
Follow the G Suite Groups instructions at: https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control#google-groups-for-gke.

Then, create a cluster with:
gcloud container clusters create <cluster_name> --security-group <security_group_name>

Finally create Roles, ClusterRoles, RoleBindings, and ClusterRoleBindings that reference the G Suite Groups.

5.8.3 Using Command Line:
To disable Legacy Authorization for an existing cluster, run the following command:
gcloud container clusters update <cluster_name> --zone <compute_zone> \
--no-enable-legacy-authorization

5.9.1 Using Command Line:
Follow the instructions detailed at: https://cloud.google.com/kubernetes-engine/docs/how-to/using-cmek.

5.9.2 Using Command Line:
Create a new node pool using customer-managed encryption keys for the node boot disk, of <disk_type> either pd-standard or pd-ssd:
gcloud container node-pools create <cluster_name> --disk-type <disk_type> \
--boot-disk-kms-key
projects/<key_project_id>/locations/<location>/keyRings/<ring_name>/cryptoKeys/<key_name>

Create a cluster using customer-managed encryption keys for the node boot disk, of <disk_type> either pd-standard or pd-ssd:
gcloud container clusters create <cluster_name> --disk-type <disk_type> \
--boot-disk-kms-key
projects/<key_project_id>/locations/<location>/keyRings/<ring_name>/cryptoKeys/<key_name>

5.10.1 Using Command Line:
To disable the Kubernetes Dashboard on an existing cluster, run the following command:
gcloud container clusters update <cluster_name> --zone <zone> \
--update-addons=KubernetesDashboard=DISABLED

5.10.2 Using Command Line:

Upon creating a new cluster
gcloud container clusters create [CLUSTER_NAME] \
--zone [COMPUTE_ZONE]

Do not use the --enable-kubernetes-alpha argument.

5.10.3 Using Command Line:
To enable GKE Sandbox on an existing cluster, a new Node pool must be created,
which can be done using:
gcloud container node-pools create <node_pool_name> --zone <compute-zone> \
--cluster <cluster_name> --image-type=cos_containerd --sandbox="type=gvisor"

5.10.4 Using Command Line:
Update the cluster to enable Binary Authorization:
gcloud container cluster update <cluster_name> --zone <compute_zone> \
--binauthz-evaluation-mode=<evaluation_mode>

Example:
gcloud container clusters update $CLUSTER_NAME --zone $COMPUTE_ZONE \
--binauthz-evaluation-mode=PROJECT_SINGLETON_POLICY_ENFORCE

See:
https://cloud.google.com/sdk/gcloud/reference/container/clusters/update#--binauthz-evaluation-mode
for more details around the evaluation modes available.

Create a Binary Authorization Policy using the Binary Authorization Policy Reference:
https://cloud.google.com/binary-authorization/docs/policy-yaml-reference for guidance.

Import the policy file into Binary Authorization:
gcloud container binauthz policy import <yaml_policy>

5.10.5 Enable security posture via the UI, gCloud or API.
https://cloud.google.com/kubernetes-engine/docs/how-to/protect-workload-configuration