

CIS Google Kubernetes Engine (GKE) Benchmark

v1.4.0 - 04-19-2023

Terms of Use

Please see the below link for our current terms of use:

<https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/>

Table of Contents

Terms of Use	1
Table of Contents	2
Overview	5
Intended Audience	5
Consensus Guidance	6
Typographical Conventions	7
Recommendation Definitions	8
Title	8
Assessment Status	8
Automated	8
Manual	8
Profile	8
Description	8
Rationale Statement	8
Impact Statement	9
Audit Procedure	9
Remediation Procedure	9
Default Value	9
References	9
CIS Critical Security Controls® (CIS Controls®)	9
Additional Information	9
Profile Definitions	10
Acknowledgements	11
Recommendations	12
1 Control Plane Components	12
2 Control Plane Configuration	12
2.1 Authentication and Authorization	13
2.1.1 Client certificate authentication should not be used for users (Manual)	14
2.2 Logging	16
2.2.1 Ensure that a minimal audit policy is created (Manual)	17
2.2.2 Ensure that the audit policy covers key security concerns (Manual)	19
3 Worker Nodes	20
3.1 Worker Node Configuration Files	21
3.1.1 Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Manual)	22
3.1.2 Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)	24
3.1.3 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Manual) ...	26

3.1.4 Ensure that the kubelet configuration file ownership is set to root:root (Manual)	28
3.2 Kubelet	30
3.2.1 Ensure that the --anonymous-auth argument is set to false (Automated)	31
3.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	34
3.2.3 Ensure that the --client-ca-file argument is set as appropriate (Automated)	36
3.2.4 Ensure that the --read-only-port argument is set to 0 (Manual)	38
3.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)	40
3.2.6 Ensure that the --make-iptables-util-chains argument is set to true (Automated)	42
3.2.7 Ensure that the --hostname-override argument is not set (Manual)	44
3.2.8 Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture (Automated)	46
3.2.9 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	49
3.2.10 Ensure that the --rotate-certificates argument is not set to false (Automated)	51
3.2.11 Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)	54
4 Policies	56
4.1 RBAC and Service Accounts	57
4.1.1 Ensure that the cluster-admin role is only used where required (Manual)	58
4.1.2 Minimize access to secrets (Manual)	60
4.1.3 Minimize wildcard use in Roles and ClusterRoles (Manual)	62
4.1.4 Minimize access to create pods (Manual)	64
4.1.5 Ensure that default service accounts are not actively used. (Manual)	66
4.1.6 Ensure that Service Account Tokens are only mounted where necessary (Manual)	68
4.2 Pod Security Standards	70
4.2.1 Minimize the admission of privileged containers (Manual)	71
4.2.2 Minimize the admission of containers wishing to share the host process ID namespace (Manual)	73
4.2.3 Minimize the admission of containers wishing to share the host IPC namespace (Manual)	75
4.2.4 Minimize the admission of containers wishing to share the host network namespace (Manual)	77
4.2.5 Minimize the admission of containers with allowPrivilegeEscalation (Manual)	79
4.2.6 Minimize the admission of root containers (Manual)	81
4.2.7 Minimize the admission of containers with added capabilities (Manual)	83
4.2.8 Minimize the admission of containers with capabilities assigned (Manual)	85
4.3 Network Policies and CNI	87
4.3.1 Ensure that the CNI in use supports Network Policies (Manual)	88
4.3.2 Ensure that all Namespaces have Network Policies defined (Manual)	90
4.4 Secrets Management	92
4.4.1 Prefer using secrets as files over secrets as environment variables (Manual)	93
4.4.2 Consider external secret storage (Manual)	95
4.5 Extensible Admission Control	97
4.5.1 Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)	98
4.6 General Policies	100
4.6.1 Create administrative boundaries between resources using namespaces (Manual)	101
4.6.2 Ensure that the seccomp profile is set to docker/default in your pod definitions (Manual)	103
4.6.3 Apply Security Context to Your Pods and Containers (Manual)	105
4.6.4 The default namespace should not be used (Manual)	107
5 Managed services	108
5.1 Image Registry and Image Scanning	109
5.1.1 Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider (Automated)	110
5.1.2 Minimize user access to GCR (Manual)	112
5.1.3 Minimize cluster access to read-only for GCR (Manual)	116

5.1.4 Minimize Container Registries to only those approved (Manual)	119
5.2 Identity and Access Management (IAM)	122
5.2.1 Ensure GKE clusters are not running using the Compute Engine default service account (Automated)	123
5.2.2 Prefer using dedicated GCP Service Accounts and Workload Identity (Manual)	127
5.3 Cloud Key Management Service (Cloud KMS)	130
5.3.1 Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS (Automated)	131
5.4 Node Metadata	135
5.4.1 Ensure legacy Compute Engine instance metadata APIs are Disabled (Automated)	136
5.4.2 Ensure the GKE Metadata Server is Enabled (Automated)	139
5.5 Node Configuration and Maintenance	142
5.5.1 Ensure Container-Optimized OS (cos_containerd) is used for GKE node images (Automated)	143
5.5.2 Ensure Node Auto-Repair is enabled for GKE nodes (Automated)	146
5.5.3 Ensure Node Auto-Upgrade is enabled for GKE nodes (Automated)	149
5.5.4 When creating New Clusters - Automate GKE version management using Release Channels (Manual)	152
5.5.5 Ensure Shielded GKE Nodes are Enabled (Manual)	155
5.5.6 Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled (Automated)	158
5.5.7 Ensure Secure Boot for Shielded GKE Nodes is Enabled (Automated)	160
5.6 Cluster Networking	163
5.6.1 Enable VPC Flow Logs and Intranode Visibility (Automated)	164
5.6.2 Ensure use of VPC-native clusters (Automated)	166
5.6.3 Ensure Master Authorized Networks is Enabled (Automated)	169
5.6.4 Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)	172
5.6.5 Ensure clusters are created with Private Nodes (Automated)	175
5.6.6 Consider firewalling GKE worker nodes (Manual)	177
5.6.7 Ensure Network Policy is Enabled and set as appropriate (Manual)	181
5.6.8 Ensure use of Google-managed SSL Certificates (Manual)	184
5.7 Logging	186
5.7.1 Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled (Automated)	187
5.7.2 Enable Linux auditd logging (Manual)	191
5.8 Authentication and Authorization	194
5.8.1 Ensure Basic Authentication using static passwords is Disabled (Automated)	195
5.8.2 Ensure authentication using Client Certificates is Disabled (Automated)	198
5.8.3 Manage Kubernetes RBAC users with Google Groups for GKE (Manual)	201
5.8.4 Ensure Legacy Authorization (ABAC) is Disabled (Automated)	203
5.9 Storage	206
5.9.1 Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD) (Manual)	207
5.10 Other Cluster Configurations	211
5.10.1 Ensure Kubernetes Web UI is Disabled (Automated)	212
5.10.2 Ensure that Alpha clusters are not used for production workloads (Automated)	215
5.10.3 Ensure Pod Security Policy is Enabled and set as appropriate (Manual)	217
5.10.4 Consider GKE Sandbox for running untrusted workloads (Manual)	220
5.10.5 Ensure use of Binary Authorization (Automated)	223
5.10.6 Enable Cloud Security Command Center (Cloud SCC) (Manual)	226
Appendix: Summary Table	228
Appendix: Change History	<i>Error! Bookmark not defined.</i>

Overview

All CIS Benchmarks focus on technical configuration settings used to maintain and/or increase the security of the addressed technology, and they should be used in **conjunction** with other essential cyber hygiene tasks like:

- Monitoring the base operating system for vulnerabilities and quickly updating with the latest security patches
- Monitoring applications and libraries for vulnerabilities and quickly updating with the latest security patches

In the end, the CIS Benchmarks are designed as a key **component** of a comprehensive cybersecurity program.

This document provides prescriptive guidance for running Google Kubernetes Engine Service (GKE) following recommended security controls. This benchmark only includes controls which can be modified by an end user of Google GKE and is designed to supersede all previous version of the Google Kubernetes Engine Service (GKE) Benchmark. It addresses and has been tested against Kubernetes version/s 1.25, 1.24 and 1.23. As a rule, this benchmark will address the Kubernetes versions available for cluster creation at the beginning of the consensus release period and will address Kubernetes versions that become available after that date in the next release. For information on GKE's performance against the Kubernetes CIS benchmarks, for items which cannot be audited or modified, see the GKE documentation at <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>. For the latest GKE hardening guide, see [g.co/gke/hardening](https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks). To obtain the latest version of this guide, please visit [www.cisecurity.org](https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks). If you have questions, comments, or have identified ways to improve this guide, please write us at support@cisecurity.org.

Intended Audience

This document is intended for cluster administrators, security specialists, auditors, and any personnel who plan to develop, deploy, assess, or secure solutions that incorporate Google Kubernetes Engine (GKE).

Consensus Guidance

This CIS Benchmark was created using a consensus review process comprised of a global community of subject matter experts. The process combines real world experience with data-based information to create technology specific guidance to assist users to secure their environments. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS Benchmark undergoes two phases of consensus review. The first phase occurs during initial Benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the Benchmark. This discussion occurs until consensus has been reached on Benchmark recommendations. The second phase begins after the Benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the Benchmark. If you are interested in participating in the consensus process, please visit <https://workbench.cisecurity.org/>.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
<code>Stylized Monospace font</code>	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
<code>Monospace font</code>	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<i><italic font in brackets></i>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
<i>Italic font</i>	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Recommendation Definitions

The following defines the various components included in a CIS recommendation as applicable. If any of the components are not applicable it will be noted or the component will not be included in the recommendation.

Title

Concise description for the recommendation's intended configuration.

Assessment Status

An assessment status is included for every recommendation. The assessment status indicates whether the given recommendation can be automated or requires manual steps to implement. Both statuses are equally important and are determined and supported as defined below:

Automated

Represents recommendations for which assessment of a technical control can be fully automated and validated to a pass/fail state. Recommendations will include the necessary information to implement automation.

Manual

Represents recommendations for which assessment of a technical control cannot be fully automated and requires all or some manual steps to validate that the configured state is set as expected. The expected state can vary depending on the environment.

Profile

A collection of recommendations for securing a technology or a supporting platform. Most benchmarks include at least a Level 1 and Level 2 Profile. Level 2 extends Level 1 recommendations and is not a standalone profile. The Profile Definitions section in the benchmark provides the definitions as they pertain to the recommendations included for the technology.

Description

Detailed information pertaining to the setting with which the recommendation is concerned. In some cases, the description will include the recommended value.

Rationale Statement

Detailed reasoning for the recommendation to provide the user a clear and concise understanding on the importance of the recommendation.

Impact Statement

Any security, functionality, or operational consequences that can result from following the recommendation.

Audit Procedure

Systematic instructions for determining if the target system complies with the recommendation

Remediation Procedure

Systematic instructions for applying recommendations to the target system to bring it into compliance according to the recommendation.

Default Value

Default value for the given setting in this recommendation, if known. If not known, either not configured or not defined will be applied.

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

The mapping between a recommendation and the CIS Controls is organized by CIS Controls version, Safeguard, and Implementation Group (IG). The Benchmark in its entirety addresses the CIS Controls safeguards of (v7) "5.1 - Establish Secure Configurations" and (v8) "4.1 - Establish and Maintain a Secure Configuration Process" so individual recommendations will not be mapped to these safeguards.

Additional Information

Supplementary information that does not correspond to any other field but may be useful to the user.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

Level 1

- **Level 2**

Level 2

Acknowledgements

This Benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Authors

Andrew Martin
Rowan Baker
Kevin Ward

Editors

Randall Mowen and Mark Larinde

Contributors

Rory Mccune
Jordan Liggitt
Liz Rice
Maya Kaczorowski
Andrew Kiggins
Greg Castle

Recommendations

1 Control Plane Components

Under the [GCP Shared Responsibility Model](#), Google manages the GKE control plane components for you. The control plane includes the Kubernetes API server, etcd, and a number of controllers. Google is responsible for securing the control plane, though you might be able to configure certain options based on your requirements. Section 3 of this Benchmark addresses these configurations.

You as the end user are responsible for securing your nodes, containers, and Pods and that is what this Benchmark specifically addresses.

[This document describes how cluster control plane components are secured in Google Kubernetes](#)

2 Control Plane Configuration

This section contains recommendations for cluster-wide areas, such as authentication and logging. These recommendations apply to all deployments.

2.1 Authentication and Authorization

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

Profile Applicability:

- Level 1

Description:

Kubernetes provides the option to use client certificates for user authentication. However as there is no way to revoke these certificates when a user leaves an organization or loses their credential, they are not suitable for this purpose.

It is not possible to fully disable client certificate use within a cluster as it is used for component to component authentication.

Rationale:

With any authentication mechanism the ability to revoke credentials if they are compromised or no longer required, is a key control. Kubernetes client certificate authentication does not allow for this due to a lack of support for certificate revocation.

See also Recommendation 5.8.2 for GKE specifically.

Impact:

External mechanisms for authentication generally require additional software to be deployed.

Audit:

Review user access to the cluster and ensure that users are not making use of Kubernetes client certificate authentication.

You can verify the availability of client certificates in your GKE cluster. See Recommendation 5.8.2.

Remediation:

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 5.8.2.

Default Value:

See the GKE documentation for the default value.







References:

1. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

Additional Information:

The lack of certificate revocation was flagged up as a high risk issue in the recent Kubernetes security audit. Without this feature, client certificate authentication is not suitable for end users.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>6.2 Establish an Access Revoking Process</u> Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

2.2 Logging

2.2.1 Ensure that a minimal audit policy is created (Manual)

Profile Applicability:

- Level 1

Description:

Kubernetes can audit the details of requests made to the API server. The `--audit-policy-file` flag must be set for this logging to be enabled.

Rationale:

Logging is an important detective control for all systems, to detect potential unauthorised access.

Impact:

Audit logs will be created on the master nodes, which will consume disk space. Care should be taken to avoid generating too large volumes of log information as this could impact the availability of the cluster nodes.

Audit:

This control cannot be audited in GKE.

Remediation:

This control cannot be modified in GKE.




Default Value:




See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/>
2. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	6.2 <u>Activate audit logging</u> Ensure that local logging has been enabled on all systems and networking devices.			

2.2.2 Ensure that the audit policy covers key security concerns (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that the audit policy created for the cluster covers key security concerns.

Rationale:

Security audit logs should cover access and modification of key resources in the cluster, to enable them to form an effective part of a security environment.

Impact:

Increasing audit logging will consume resources on the nodes or other log destination.

Audit:

This control cannot be audited in GKE.

Remediation:

This control cannot be modified in GKE.




Default Value:

See the GKE documentation for the default value.

References:

1. <https://github.com/k8scop/k8s-security-dashboard/blob/master/configs/kubernetes/adv-audit.yaml>
2. <https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/#audit-policy>
3. <https://github.com/kubernetes/kubernetes/blob/master/cluster/gce/gci/configure-helper.sh#L735>
4. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.5 Collect Detailed Audit Logs Configure detailed audit logging for enterprise assets containing sensitive data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.			
v7	14.9 Enforce Detail Logging for Access or Changes to Sensitive Data Enforce detailed audit logging for access to sensitive data or changes to sensitive data (utilizing tools such as File Integrity Monitoring or Security Information and Event Monitoring).			

3 Worker Nodes

This section consists of security recommendations for the components that run on GKE worker nodes.

3.1 Worker Node Configuration Files

This section covers recommendations for configuration files on the worker nodes.

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

Profile Applicability:

- Level 1

Description:

If `kube-proxy` is running, and if it is configured by a kubeconfig file, ensure that the proxy kubeconfig file has permissions of 644 or more restrictive.

Rationale:

The `kube-proxy` kubeconfig file controls various parameters of the `kube-proxy` service in the worker node. You should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

It is possible to run `kube-proxy` with the kubeconfig parameters configured as a Kubernetes ConfigMap instead of a file. In this case, there is no proxy kubeconfig file.

Impact:

None.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node.
2. Run the following command on each node to find the appropriate kubeconfig file:

```
ps -ef | grep kube-proxy
```

The output of the above command should return something similar to `--kubeconfig /var/lib/kube-proxy/kubeconfig` which is the location of the kubeconfig file.

3. Run this command to obtain the kubeconfig file permissions:

```
stat -c %a /var/lib/kube-proxy/kubeconfig
```

The output of the above command gives you the kubeconfig file's permissions. Verify that if a file is specified and it exists, the permissions are 644 or more restrictive.

Remediation:

Run the below command (based on the file location on your system) on the each worker node. For example,

```
chmod 644 <proxy kubeconfig file>
```






Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kube-proxy/>
2. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

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

Profile Applicability:

- Level 1

Description:

If `kube-proxy` is running, ensure that the file ownership of its kubeconfig file is set to `root:root`.

Rationale:

The kubeconfig file for `kube-proxy` controls various parameters for the `kube-proxy` service in the worker node. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Impact:

None

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate kubeconfig file:

```
ps -ef | grep kube-proxy
```

The output of the above command should return something similar to `--kubeconfig /var/lib/kube-proxy/kubeconfig` which is the location of the kubeconfig file.

3. Run this command to obtain the kubeconfig file ownership:

```
stat -c %U:%G /var/lib/kube-proxy/kubeconfig
```

The output of the above command gives you the kubeconfig file's ownership. Verify that the ownership is set to `root:root`.

Remediation:

Run the below command (based on the file location on your system) on each worker node. For example,

```
chown root:root <proxy kubeconfig file>
```






Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kube-proxy/>
2. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

3.1.3 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file has permissions of 644 or more restrictive.

Rationale:

The kubelet reads various parameters, including security settings, from a config file specified by the `--config` argument. If this file is specified you should restrict its file permissions to maintain the integrity of the file. The file should be writable by only the administrators on the system.

Impact:

None.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Run the following command:

```
stat -c %a /home/kubernetes/kubelet-config.yaml
```

The output of the above command is the Kubelet config file's permissions. Verify that the permissions are 644 or more restrictive.

Remediation:

Run the following command (using the config file location identified in the Audit step)

```
chmod 644 /var/lib/kubelet/config.yaml
```






Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/>
2. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

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

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet refers to a configuration file with the `--config` argument, that file is owned by root:root.

Rationale:

The kubelet reads various parameters, including security settings, from a config file specified by the `--config` argument. If this file is specified you should restrict its file permissions to maintain the integrity of the file. The file should be owned by root:root.

Impact:

None.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Run the following command:

```
stat -c %U:%G /home/kubernetes/kubelet-config.yaml
```

The output of the above command is the Kubelet config file's ownership. Verify that the ownership is set to `root:root`

Remediation:

Run the following command (using the config file location identified in the Audit step)

```
chown root:root /etc/kubernetes/kubelet.conf
```






Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/tasks/administer-cluster/kubelet-config-file/>
2. <https://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

3.2 Kubelet

This section contains recommendations for kubelet configuration.

Kubelet settings may be configured using arguments on the running kubelet executable, or they may be taken from a Kubelet config file. If both are specified, the executable argument takes precedence.

To find the Kubelet config file, run the following command:

```
ps -ef | grep kubelet | grep config
```

If the `--config` argument is present, this gives the location of the Kubelet config file. This config file could be in JSON or YAML format depending on your distribution.

3.2.1 Ensure that the `--anonymous-auth` argument is set to false (Automated)

Profile Applicability:

- Level 1

Description:

Disable anonymous requests to the Kubelet server.

Rationale:

When enabled, requests that are not rejected by other configured authentication methods are treated as anonymous requests. These requests are then served by the Kubelet server. You should rely on authentication to authorize access and disallow anonymous requests.

Impact:

Anonymous requests will be rejected.

Audit:

If using a Kubelet configuration file, check that there is an entry for `authentication:`
`anonymous: enabled` set to `false`.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

4. Verify that the `--anonymous-auth` argument is set to `false`.

This executable argument may be omitted, provided there is a corresponding entry set to `false` in the Kubelet config file.

Remediation:

If using a Kubelet config file, edit the file to set `authentication: anonymous: enabled` to `false`.

If using executable 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.

```
--anonymous-auth=false
```

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

```
systemctl daemon-reload  
systemctl restart kubelet.service
```







Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authentication>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.3 Disable Dormant Accounts Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.			
v7	14.6 Protect Information through Access Control Lists Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.			

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

Profile Applicability:

- Level 1

Description:

Do not allow all requests. Enable explicit authorization.

Rationale:

Kubelets, by default, allow all authenticated requests (even anonymous ones) without needing explicit authorization checks from the apiserver. You should restrict this behavior and only allow explicitly authorized requests.

Impact:

Unauthorized requests will be denied.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

If the `--authorization-mode` argument is present check that it is not set to `AlwaysAllow`. If it is not present check that there is a Kubelet config file specified by `--config`, and that file sets `authorization: mode` to something other than `AlwaysAllow`. It is also possible to review the running configuration of a Kubelet via the `/configz` endpoint on the Kubelet API port (typically `10250/TCP`). Accessing these with appropriate credentials will provide details of the Kubelet's configuration.

Remediation:

If using a Kubelet config file, edit the file to set `authorization: mode` to `Webhook`.

If using executable 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_AUTHZ_ARGS` variable.

```
--authorization-mode=Webhook
```

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

```
systemctl daemon-reload
systemctl restart kubelet.service
```







Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <https://kubernetes.io/docs/admin/kubelet-authentication-authorization/#kubelet-authentication>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	4.2 Change Default Passwords Before deploying any new asset, change all default passwords to have values consistent with administrative level accounts.			

3.2.3 Ensure that the `--client-ca-file` argument is set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Enable Kubelet authentication using certificates.

Rationale:

The connections from the apiserver to the kubelet are used for fetching logs for pods, attaching (through kubectl) to running pods, and using the kubelet's port-forwarding functionality. These connections terminate at the kubelet's HTTPS endpoint. By default, the apiserver does not verify the kubelet's serving certificate, which makes the connection subject to man-in-the-middle attacks, and unsafe to run over untrusted and/or public networks. Enabling Kubelet certificate authentication ensures that the apiserver could authenticate the Kubelet before submitting any requests.

Impact:

You require TLS to be configured on apiserver as well as kubelets.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that the `--client-ca-file` argument exists and is set to the location of the client certificate authority file.

If the `--client-ca-file` argument is not present, check that there is a Kubelet config file specified by `--config`, and that the file sets `authentication: x509: clientCAFile` to the location of the client certificate authority file.

Remediation:

If using a Kubelet config file, edit the file to set `authentication: x509: clientCAFile` to the location of the client CA file.

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_AUTHZ_ARGS` variable.

```
--client-ca-file=<path/to/client-ca-file>
```

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

```
systemctl daemon-reload
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-authentication-authorization/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 <u>Encrypt Sensitive Data in Transit</u> Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).			
v7	14.4 <u>Encrypt All Sensitive Information in Transit</u> Encrypt all sensitive information in transit.			

3.2.4 Ensure that the `--read-only-port` argument is set to 0 (Manual)

Profile Applicability:

- Level 1

Description:

Disable the read-only port.

Rationale:

The Kubelet process provides a read-only API in addition to the main Kubelet API. Unauthenticated access is provided to this read-only API which could possibly retrieve potentially sensitive information about the cluster.

Impact:

Removal of the read-only port will require that any service which made use of it will need to be re-configured to use the main Kubelet API.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that the `--read-only-port` argument exists and is set to 0.

If the `--read-only-port` argument is not present, check that there is a Kubelet config file specified by `--config`. Check that if there is a `readOnlyPort` entry in the file, it is set to 0.

Remediation:

If using a Kubelet config file, edit the file to set `readOnlyPort` to 0.

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.

```
--read-only-port=0
```

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

```
systemctl daemon-reload
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 <u>Use of Secure Network Management and Communication Protocols</u> Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).			
v7	9.2 <u>Ensure Only Approved Ports, Protocols and Services Are Running</u> Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.			

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

Profile Applicability:

- Level 1

Description:

Do not disable timeouts on streaming connections.

Rationale:

Setting idle timeouts ensures that you are protected against Denial-of-Service attacks, inactive connections and running out of ephemeral ports.

Note: By default, `--streaming-connection-idle-timeout` is set to 4 hours which might be too high for your environment. Setting this as appropriate would additionally ensure that such streaming connections are timed out after serving legitimate use cases.

Impact:

Long-lived connections could be interrupted.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that the `--streaming-connection-idle-timeout` argument is not set to 0.
If the argument is not present, and there is a Kubelet config file specified by `--config`, check that it does not set `streamingConnectionIdleTimeout` to 0.

Remediation:

If using a Kubelet config file, edit the file to set `streamingConnectionIdleTimeout` to a value other than 0.

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.

```
--streaming-connection-idle-timeout=5m
```

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

```
systemctl daemon-reload  
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <https://github.com/kubernetes/kubernetes/pull/18552>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 <u>Use of Secure Network Management and Communication Protocols</u> Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).			
v7	9.2 <u>Ensure Only Approved Ports, Protocols and Services Are Running</u> Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.			

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

Profile Applicability:

- Level 1

Description:

Allow Kubelet to manage iptables.

Rationale:

Kubelets can automatically manage the required changes to iptables based on how you choose your networking options for the pods. It is recommended to let kubelets manage the changes to iptables. This ensures that the iptables configuration remains in sync with pods networking configuration. Manually configuring iptables with dynamic pod network configuration changes might hamper the communication between pods/containers and to the outside world. You might have iptables rules too restrictive or too open.

Impact:

Kubelet would manage the iptables on the system and keep it in sync. If you are using any other iptables management solution, then there might be some conflicts.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that if the `--make-iptables-util-chains` argument exists then it is set to `true`.
If the `--make-iptables-util-chains` argument does not exist, and there is a Kubelet config file specified by `--config`, verify that the file does not set `makeIPTablesUtilChains` to `false`.

Remediation:

If using a Kubelet config file, edit the file to set `makeIPTablesUtilChains: true`.

If using command line arguments, edit the kubelet service file

`/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove the `--make-iptables-util-chains` argument from the `KUBELET_SYSTEM_PODS_ARGS` variable.

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

```
systemctl daemon-reload  
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.3 Securely Manage Network Infrastructure Securely manage network infrastructure. Example implementations include version-controlled-infrastructure-as-code, and the use of secure network protocols, such as SSH and HTTPS.			
v7	11.1 Maintain Standard Security Configurations for Network Devices Maintain standard, documented security configuration standards for all authorized network devices.			

3.2.7 Ensure that the `--hostname-override` argument is not set (Manual)

Profile Applicability:

- Level 1

Description:

Do not override node hostnames.

Rationale:

Overriding hostnames could potentially break TLS setup between the kubelet and the apiserver. Additionally, with overridden hostnames, it becomes increasingly difficult to associate logs with a particular node and process them for security analytics. Hence, you should setup your kubelet nodes with resolvable FQDNs and avoid overriding the hostnames with IPs. `--hostname-override` also may have some undefined/unsupported behaviors.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that `--hostname-override` argument does not exist.

Note This setting is not configurable via the Kubelet config file.

Remediation:

Edit the kubelet service file `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove the `--hostname-override` argument from the `KUBELET_SYSTEM_PODS_ARGS` variable.

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

```
systemctl daemon-reload
systemctl restart kubelet.service
```



Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <https://github.com/kubernetes/kubernetes/issues/22063>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.7 Use Standard Hardening Configuration Templates for Application Infrastructure Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.			
v7	5 Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers			

3.2.8 Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture (Automated)

Profile Applicability:

- Level 1

Description:

Security relevant information should be captured. The `--event-qps` flag on the Kubelet can be used to limit the rate at which events are gathered. Setting this too low could result in relevant events not being logged, however the unlimited setting of `0` could result in a denial of service on the kubelet. The recommended value is 5 or higher.

Rationale:

It is important to capture all events and not restrict event creation. Events are an important source of security information and analytics that ensure that your environment is consistently monitored using the event data.

Impact:

Setting this parameter to `0` could result in a denial of service condition due to excessive events being created. The cluster's event processing and storage systems should be scaled to handle expected event loads.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Review the value set for the `--event-qps` argument and determine whether this has been set to an appropriate level for the cluster. The value of `0` can be used to ensure that all events are captured but `5` or higher is recommended.

If the `--eventrecordqps` argument does not exist, check that there is a Kubelet config file specified by `--config` and review the value in this location.

Remediation:

If using a Kubelet config file, edit the file to set `--event-qps` 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
```




Default Value:








By default, `--event-qps` argument is set to `5`.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/apis/kubeletconfig/v1beta1/types.go>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.			

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.5 Collect Detailed Audit Logs Configure detailed audit logging for enterprise assets containing sensitive data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.			
v7	6.2 <u>Activate audit logging</u> Ensure that local logging has been enabled on all systems and networking devices.			
v7	6.3 <u>Enable Detailed Logging</u> Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.			

3.2.9 Ensure that the `--tls-cert-file` and `--tls-private-key-file` arguments are set as appropriate (Automated)

Profile Applicability:

- Level 1

Description:

Setup TLS connection on the Kubelets.

Rationale:

Kubelet communication contains sensitive parameters that should remain encrypted in transit. Configure the Kubelets to serve only HTTPS traffic.

Impact:

TLS and client certificate authentication must be configured for your Kubernetes cluster deployment.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that the `--tls-cert-file` and `--tls-private-key-file` arguments exist and they are set as appropriate.

If these arguments are not present, check that there is a Kubelet config specified by `--config` and that it contains appropriate settings for `tlsCertFile` and `tlsPrivateKeyFile`.

Remediation:

If using a Kubelet config file, edit the file to set `tlsCertFile` to the location of the certificate file to use to identify this Kubelet, and `tlsPrivateKeyFile` to the location of the corresponding private key file.

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 parameters in `KUBELET_CERTIFICATE_ARGS` variable.

```
--tls-cert-file=<path/to/tls-certificate-file> --tls-private-key-file=<path/to/tls-key-file>
```

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

```
systemctl daemon-reload
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://kubernetes.io/docs/admin/kubelet/>
2. <http://rootsquash.com/2016/05/10/securing-the-kubernetes-api/>
3. <https://github.com/kelseyhightower/docker-kubernetes-tls-guide>
4. <https://jvns.ca/blog/2017/08/05/how-kubernetes-certificates-work/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).			
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.			

3.2.10 Ensure that the `--rotate-certificates` argument is not set to false (Automated)

Profile Applicability:

- Level 1

Description:

Enable kubelet client certificate rotation.

Rationale:

The `--rotate-certificates` setting causes the kubelet to rotate its client certificates by creating new CSRs as its existing credentials expire. This automated periodic rotation ensures that there is no downtime due to expired certificates and thus addressing availability in the CIA (Confidentiality, Integrity, and Availability) security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to implement rotation yourself.

Note: This feature also requires the `RotateKubeletClientCertificate` feature gate to be enabled.

Impact:

None

Audit:

Using Google Cloud Console Using Command Line

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that the `--rotate-certificates` argument is not present, or is set to `true`.
If the `--rotate-certificates` argument is not present, verify that if there is a Kubelet config file specified by `--config`, that file does not contain `rotateCertificates: false`.

Remediation:

If using a Kubelet config file, edit the file to add the line `rotateCertificates: true` or remove it altogether to use the default value.

If using command line arguments, edit the kubelet service file

`/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` on each worker node and remove `--rotate-certificates=false` argument from the `KUBELET_CERTIFICATE_ARGS` variable.

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

```
systemctl daemon-reload  
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://github.com/kubernetes/kubernetes/pull/41912>
2. <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet-tls-bootstrapping/#kubelet-configuration>
3. <https://kubernetes.io/docs/reference/command-line-tools-reference/feature-gates/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 <u>Encrypt Sensitive Data in Transit</u> Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).			
v7	14.4 <u>Encrypt All Sensitive Information in Transit</u> Encrypt all sensitive information in transit.			

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

Profile Applicability:

- Level 1

Description:

Enable kubelet server certificate rotation.

Rationale:

`RotateKubeletServerCertificate` causes the kubelet to both request a serving certificate after bootstrapping its client credentials and rotate the certificate as its existing credentials expire. This automated periodic rotation ensures that there are no downtimes due to expired certificates and thus addressing availability in the CIA security triad.

Note: This recommendation only applies if you let kubelets get their certificates from the API server. In case your kubelet certificates come from an outside authority/tool (e.g. Vault) then you need to take care of rotation yourself.

Impact:

None

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster to open the Details page, then click on the desired Node pool to open the Node pool Details page
3. Note the name of the desired node
4. Go to VM Instances by visiting <https://console.cloud.google.com/compute/instances>
5. Find the desired node and click on 'SSH' to open an SSH connection to the node.

Using Command Line

1. SSH to the relevant node
2. Run the following command on each node to find the appropriate Kubelet config file:

```
ps -ef | grep kubelet
```

The output of the above command should return something similar to `--config /home/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

3. Open the Kubelet config file:

```
sudo vim /home/kubernetes/kubelet-config.yaml
```

Verify that `RotateKubeletServerCertificate` argument exists and is set to `true`.

Remediation:

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_CERTIFICATE_ARGS` variable.

```
--feature-gates=RotateKubeletServerCertificate=true
```

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

```
systemctl daemon-reload  
systemctl restart kubelet.service
```





Default Value:

See the GKE documentation for the default value.

References:

1. <https://github.com/kubernetes/kubernetes/pull/45059>
2. <https://kubernetes.io/docs/admin/kubelet-tls-bootstrapping/#kubelet-configuration>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 Encrypt Sensitive Data in Transit Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).			
v7	14.4 Encrypt All Sensitive Information in Transit Encrypt all sensitive information in transit.			

4 Policies

This section contains recommendations for various Kubernetes policies which are important to the security of the environment.

4.1 RBAC and Service Accounts

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

Profile Applicability:

- Level 1

Description:

The RBAC role `cluster-admin` provides wide-ranging powers over the environment and should be used only where and when needed.

Rationale:

Kubernetes provides a set of default roles where RBAC is used. Some of these roles such as `cluster-admin` provide wide-ranging privileges which should only be applied where absolutely necessary. Roles such as `cluster-admin` allow super-user access to perform any action on any resource. When used in a `ClusterRoleBinding`, it gives full control over every resource in the cluster and in all namespaces. When used in a `RoleBinding`, it gives full control over every resource in the rolebinding's namespace, including the namespace itself.

Impact:

Care should be taken before removing any `clusterrolebindings` from the environment to ensure they were not required for operation of the cluster. Specifically, modifications should not be made to `clusterrolebindings` with the `system:` prefix as they are required for the operation of system components.

Audit:

Obtain a list of the principals who have access to the `cluster-admin` role by reviewing the `clusterrolebinding` output for each role binding that has access to the `cluster-admin` role.

```
kubectl get clusterrolebindings -o=custom-  
columns=NAME:.metadata.name,ROLE:.roleRef.name,SUBJECT:.subjects[*].name
```

Review each principal listed and ensure that `cluster-admin` privilege is required for it.

Remediation:

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]
```







Default Value:

By default a single `clusterrolebinding` called `cluster-admin` is provided with the `system:masters` group as its principal.

References:

1. <https://kubernetes.io/docs/admin/authorization/rbac/#user-facing-roles>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 <u>Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	4.3 <u>Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.1.2 Minimize access to secrets (Manual)

Profile Applicability:

- Level 1

Description:

The Kubernetes API stores secrets, which may be service account tokens for the Kubernetes API or credentials used by workloads in the cluster. Access to these secrets should be restricted to the smallest possible group of users to reduce the risk of privilege escalation.

Rationale:

Inappropriate access to secrets stored within the Kubernetes cluster can allow for an attacker to gain additional access to the Kubernetes cluster or external resources whose credentials are stored as secrets.

Impact:

Care should be taken not to remove access to secrets to system components which require this for their operation

Audit:

Review the users who have `get`, `list` or `watch` access to `secrets` objects in the Kubernetes API.

Remediation:






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

Default Value:

By default in a kubeadm cluster the following list of principals have `get` privileges on `secret` objects

CLUSTERROLEBINDING	SUBJECT
TYPE SA-NAMESPACE	
cluster-admin	system:masters
Group	
system:controller:clusterrole-aggregation-controller	clusterrole-
aggregation-controller ServiceAccount kube-system	
system:controller:expand-controller	expand-controller
ServiceAccount kube-system	
system:controller:generic-garbage-collector	generic-garbage-
collector ServiceAccount kube-system	
system:controller:namespace-controller	namespace-controller
ServiceAccount kube-system	
system:controller:persistent-volume-binder	persistent-volume-
binder ServiceAccount kube-system	
system:kube-controller-manager	system:kube-controller-
manager User	

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.1 <u>Establish and Maintain a Secure Configuration Process</u> Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.			
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

4.1.3 Minimize wildcard use in Roles and ClusterRoles (Manual)

Profile Applicability:

- Level 1

Description:

Kubernetes Roles and ClusterRoles provide access to resources based on sets of objects and actions that can be taken on those objects. It is possible to set either of these to be the wildcard "*" which matches all items.

Use of wildcards is not optimal from a security perspective as it may allow for inadvertent access to be granted when new resources are added to the Kubernetes API either as CRDs or in later versions of the product.

Rationale:

The principle of least privilege recommends that users are provided only the access required for their role and nothing more. The use of wildcard rights grants is likely to provide excessive rights to the Kubernetes API.

Audit:

Retrieve the roles defined across each namespaces in the cluster and review for wildcards

```
kubectl get roles --all-namespaces -o yaml
```






Retrieve the cluster roles defined in the cluster and review for wildcards

```
kubectl get clusterroles -o yaml
```

Remediation:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.2 Use Unique Passwords Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.			
v7	4.4 Use Unique Passwords Where multi-factor authentication is not supported (such as local administrator, root, or service accounts), accounts will use passwords that are unique to that system.			

4.1.4 Minimize access to create pods (Manual)

Profile Applicability:

- Level 1

Description:

The ability to create pods in a namespace can provide a number of opportunities for privilege escalation, such as assigning privileged service accounts to these pods or mounting hostPaths with access to sensitive data (unless Pod Security Policies are implemented to restrict this access)

As such, access to create new pods should be restricted to the smallest possible group of users.

Rationale:

The ability to create pods in a cluster opens up possibilities for privilege escalation and should be restricted, where possible.

Impact:

Care should be taken not to remove access to pods to system components which require this for their operation

Audit:

Review the users who have create access to pod objects in the Kubernetes API.

Remediation:

Where possible, remove `create` access to `pod` objects in the cluster.

Default Value:

By default in a kubeadm cluster the following list of principals have `create` privileges on `pod` objects

CLUSTERROLEBINDING	SUBJECT
TYPE SA-NAMESPACE	
cluster-admin	system:masters
Group	
system:controller:clusterrole-aggregation-controller	clusterrole-
aggregation-controller ServiceAccount kube-system	
system:controller:daemon-set-controller	daemon-set-controller
ServiceAccount kube-system	
system:controller:job-controller	job-controller
ServiceAccount kube-system	
system:controller:persistent-volume-binder	persistent-volume-
binder ServiceAccount kube-system	
system:controller:replicaset-controller	replicaset-controller
ServiceAccount kube-system	
system:controller:replication-controller	replication-controller
ServiceAccount kube-system	
system:controller:statefulset-controller	statefulset-controller
ServiceAccount kube-system	

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.7 Allowlist Authorized Scripts Use technical controls, such as digital signatures and version control, to ensure that only authorized scripts, such as specific .ps1, .py, etc., files, are allowed to execute. Block unauthorized scripts from executing. Reassess bi-annually, or more frequently.			●
v7	4.7 Limit Access to Script Tools Limit access to scripting tools (such as Microsoft PowerShell and Python) to only administrative or development users with the need to access those capabilities.		●	●

4.1.5 Ensure that default service accounts are not actively used. (Manual)

Profile Applicability:

- Level 1

Description:

The `default` service account should not be used to ensure that rights granted to applications can be more easily audited and reviewed.

Rationale:

Kubernetes provides a `default` service account which is used by cluster workloads where no specific service account is assigned to the pod.

Where access to the Kubernetes API from a pod is required, a specific service account should be created for that pod, and rights granted to that service account.

The default service account should be configured such that it does not provide a service account token and does not have any explicit rights assignments.

Impact:

All workloads which require access to the Kubernetes API will require an explicit service account to be created.

Audit:

For each namespace in the cluster, review the rights assigned to the default service account and ensure that it has no roles or cluster roles bound to it apart from the defaults.

Additionally ensure that the `automountServiceAccountToken: false` setting is in place for each default service account.

Remediation:

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
```












Default Value:

By default the `default` service account allows for its service account token to be mounted in pods in its namespace.

References:

1. <https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.3 <u>Disable Dormant Accounts</u> Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.			
v7	4.3 <u>Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			
v7	16.9 <u>Disable Dormant Accounts</u> Automatically disable dormant accounts after a set period of inactivity.			

4.1.6 Ensure that Service Account Tokens are only mounted where necessary (Manual)

Profile Applicability:

- Level 1

Description:

Service accounts tokens should not be mounted in pods except where the workload running in the pod explicitly needs to communicate with the API server

Rationale:

Mounting service account tokens inside pods can provide an avenue for privilege escalation attacks where an attacker is able to compromise a single pod in the cluster.

Avoiding mounting these tokens removes this attack avenue.

Impact:

Pods mounted without service account tokens will not be able to communicate with the API server, except where the resource is available to unauthenticated principals.

Audit:

Review pod and service account objects in the cluster and ensure that the option below is set, unless the resource explicitly requires this access.

```
automountServiceAccountToken: false
```

Remediation:

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




Default Value:

By default, all pods get a service account token mounted in them.

References:

1. <https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software</u> Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.			
v7	<u>14.7 Enforce Access Control to Data through Automated Tools</u> Use an automated tool, such as host-based Data Loss Prevention, to enforce access controls to data even when data is copied off a system.			

4.2 Pod Security Standards

Pod Security Standards (PSS) are recommendations for securing deployed workloads to reduce the risks of container breakout. There are a number of ways if implementing PSS, including the built-in Pod Security Admission controller, or external policy control systems which integrate with Kubernetes via validating and mutating webhooks.

The previous feature described in this document, pod security policy (preview), was deprecated with version 1.21, and removed as of version 1.25. After pod security policy (preview) is deprecated, you must disable the feature on any existing clusters using the deprecated feature to perform future cluster upgrades and stay within Azure support.

4.2.1 Minimize the admission of privileged containers (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `securityContext.privileged` flag set to `true`.

Rationale:

Privileged containers have access to all Linux Kernel capabilities and devices. A container running with full privileges can do almost everything that the host can do. This flag exists to allow special use-cases, like manipulating the network stack and accessing devices.

There should be at least one admission control policy defined which does not permit privileged containers.

If you need to run privileged containers, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.containers[].securityContext.privileged: true`, `spec.initContainers[].securityContext.privileged: true` and `spec.ephemeralContainers[].securityContext.privileged: true` will not be permitted.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of privileged containers.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of privileged containers.

To enable PSA for a namespace in your cluster, set the `pod-security.kubernetes.io/enforce` label with the policy value you want to enforce.

```
kubectl label --overwrite ns NAMESPACE pod-  
security.kubernetes.io/enforce=restricted
```

The above command enforces the restricted policy for the NAMESPACE namespace. You can also enable Pod Security Admission for all your namespaces. For example:

```
kubectl label --overwrite ns --all pod-security.kubernetes.io/warn=baseline
```







Default Value:

By default, there are no restrictions on the creation of privileged containers.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>
2. <https://cloud.google.com/kubernetes-engine/docs/how-to/podsecurityadmission>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.4 <u>Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	4.3 <u>Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.2 Minimize the admission of containers wishing to share the host process ID namespace (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `hostPID` flag set to true.

Rationale:

A container running in the host's PID namespace can inspect processes running outside the container. If the container also has access to ptrace capabilities this can be used to escalate privileges outside of the container.

There should be at least one admission control policy defined which does not permit containers to share the host PID namespace.

If you need to run containers which require `hostPID`, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.hostPID: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostPID` containers

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostPID` containers.







Default Value:

By default, there are no restrictions on the creation of `hostPID` containers.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.3 Minimize the admission of containers wishing to share the host IPC namespace (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `hostIPC` flag set to true.

Rationale:

A container running in the host's IPC namespace can use IPC to interact with processes outside the container.

There should be at least one admission control policy defined which does not permit containers to share the host IPC namespace.

If you need to run containers which require `hostIPC`, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.hostIPC: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostIPC` containers

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostIPC` containers.







Default Value:

By default, there are no restrictions on the creation of `hostIPC` containers.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.4 Minimize the admission of containers wishing to share the host network namespace (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `hostNetwork` flag set to true.

Rationale:

A container running in the host's network namespace could access the local loopback device, and could access network traffic to and from other pods.

There should be at least one admission control policy defined which does not permit containers to share the host network namespace.

If you need to run containers which require access to the host's network namespaces, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.hostNetwork: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of `hostNetwork` containers

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of `hostNetwork` containers.







Default Value:

By default, there are no restrictions on the creation of `hostNetwork` containers.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.5 Minimize the admission of containers with `allowPrivilegeEscalation` (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers to be run with the `allowPrivilegeEscalation` flag set to `true`. Allowing this right can lead to a process running a container getting more rights than it started with.

It's important to note that these rights are still constrained by the overall container sandbox, and this setting does not relate to the use of privileged containers.

Rationale:

A container running with the `allowPrivilegeEscalation` flag set to `true` may have processes that can gain more privileges than their parent.

There should be at least one admission control policy defined which does not permit containers to allow privilege escalation. The option exists (and is defaulted to `true`) to permit `setuid` binaries to run.

If you have need to run containers which use `setuid` binaries or require privilege escalation, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods defined with `spec.allowPrivilegeEscalation: true` will not be permitted unless they are run under a specific policy.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy disallows the admission of containers which allow privilege escalation.

Remediation:

Add policies to each namespace in the cluster which has user workloads to restrict the admission of containers with `.spec.allowPrivilegeEscalation` set to `true`.







Default Value:

By default, there are no restrictions on contained process ability to escalate privileges, within the context of the container.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.6 Minimize the admission of root containers (Manual)

Profile Applicability:

- Level 2

Description:

Do not generally permit containers to be run as the root user.

Rationale:

Containers may run as any Linux user. Containers which run as the root user, whilst constrained by Container Runtime security features still have a escalated likelihood of container breakout.

Ideally, all containers should run as a defined non-UID 0 user.

There should be at least one admission control policy defined which does not permit root containers.

If you need to run root containers, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods with containers which run as the root user will not be permitted.

Audit:

List the policies in use for each namespace in the cluster, ensure that each policy restricts the use of root containers by setting `MustRunAsNonRoot` or `MustRunAs` with the range of UIDs not including 0.

Remediation:

Create a policy for each namespace in the cluster, ensuring that either `MustRunAsNonRoot` or `MustRunAs` with the range of UIDs not including 0, is set.







Default Value:

By default, there are no restrictions on the use of root containers and if a User is not specified in the image, the container will run as root.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.7 Minimize the admission of containers with added capabilities (Manual)

Profile Applicability:

- Level 1

Description:

Do not generally permit containers with capabilities assigned beyond the default set.

Rationale:

Containers run with a default set of capabilities as assigned by the Container Runtime. Capabilities outside this set can be added to containers which could expose them to risks of container breakout attacks.

There should be at least one policy defined which prevents containers with capabilities beyond the default set from launching.

If you need to run containers with additional capabilities, this should be defined in a separate policy and you should carefully check to ensure that only limited service accounts and users are given permission to use that policy.

Impact:

Pods with containers which require capabilities out with the default set will not be permitted.

Audit:

List the policies in use for each namespace in the cluster, ensure that policies are present which prevent `allowedCapabilities` to be set to anything other than an empty array.

Remediation:

Ensure that `allowedCapabilities` is not present in policies for the cluster unless it is set to an empty array.







Default Value:

By default, there are no restrictions on adding capabilities to containers.

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>
2. <https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.2.8 Minimize the admission of containers with capabilities assigned (Manual)

Profile Applicability:

- Level 2

Description:

Do not generally permit containers with capabilities

Rationale:

Containers run with a default set of capabilities as assigned by the Container Runtime. Capabilities are parts of the rights generally granted on a Linux system to the root user.

In many cases applications running in containers do not require any capabilities to operate, so from the perspective of the principal of least privilege use of capabilities should be minimized.

Impact:

Pods with containers require capabilities to operate will not be permitted.

Audit:

List the policies in use for each namespace in the cluster, ensure that at least one policy requires that capabilities are dropped by all containers.

Remediation:

Review the use of capabilities in applications running on your cluster. Where a namespace contains applications which do not require any Linux capabilities to operate consider adding a policy which forbids the admission of containers which do not drop all capabilities.







Default Value:

By default, there are no restrictions on the creation of containers with additional capabilities

References:

1. <https://kubernetes.io/docs/concepts/security/pod-security-admission/>
2. <https://www.nccgroup.trust/uk/our-research/abusing-privileged-and-unprivileged-linux-containers/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>5.4 Restrict Administrator Privileges to Dedicated Administrator Accounts</u> Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user's primary, non-privileged account.			
v7	<u>4.3 Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

4.3 Network Policies and CNI

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

Profile Applicability:

- Level 1

Description:

There are a variety of CNI plugins available for Kubernetes. If the CNI in use does not support Network Policies it may not be possible to effectively restrict traffic in the cluster.

Rationale:

Kubernetes network policies are enforced by the CNI plugin in use. As such it is important to ensure that the CNI plugin supports both Ingress and Egress network policies.

See also Recommendation 6.6.7 for GKE specifically.

Impact:

None.

Audit:

Ensure CNI plugin supports network policies.

Remediation:

To use a CNI plugin with Network Policy, enable Network Policy in GKE, and the CNI plugin will be updated. See Recommendation 6.6.7.

Default Value:

This will depend on the CNI plugin in use.





References:

1. <https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/>

Additional Information:

One example here is Flannel (<https://github.com/flannel-io/flannel>) which does not support Network policy unless Calico is also in use.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>16.5 Use Up-to-Date and Trusted Third-Party Software Components</u> Use up-to-date and trusted third-party software components. When possible, choose established and proven frameworks and libraries that provide adequate security. Acquire these components from trusted sources or evaluate the software for vulnerabilities before use.			
v7	<u>18.4 Only Use Up-to-date And Trusted Third-Party Components</u> Only use up-to-date and trusted third-party components for the software developed by the organization.			

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

Profile Applicability:

- Level 2

Description:

Use network policies to isolate traffic in your cluster network.

Rationale:

Running different applications on the same Kubernetes cluster creates a risk of one compromised application attacking a neighboring application. Network segmentation is important to ensure that containers can communicate only with those they are supposed to. A network policy is a specification of how selections of pods are allowed to communicate with each other and other network endpoints.

Network Policies are namespace scoped. When a network policy is introduced to a given namespace, all traffic not allowed by the policy is denied. However, if there are no network policies in a namespace all traffic will be allowed into and out of the pods in that namespace.

Impact:

Once network policies are in use within a given namespace, traffic not explicitly allowed by a network policy will be denied. As such it is important to ensure that, when introducing network policies, legitimate traffic is not blocked.

Audit:

Run the below command and review the `NetworkPolicy` objects created in the cluster.

```
kubectl get networkpolicy --all-namespaces
```

ensure that each namespace defined in the cluster has at least one `Network Policy`.

Remediation:

Follow the documentation and create `NetworkPolicy` objects as you need them.

Default Value:







By default, network policies are not created.

References:

1. <https://kubernetes.io/docs/concepts/services-networking/networkpolicies/>
2. <https://octetz.com/posts/k8s-network-policy-apis>

3. <https://kubernetes.io/docs/tasks/configure-pod-container/declare-network-policy/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.4 <u>Perform Traffic Filtering Between Network Segments</u> Perform traffic filtering between network segments, where appropriate.			
v7	14.1 <u>Segment the Network Based on Sensitivity</u> Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).			
v7	14.2 <u>Enable Firewall Filtering Between VLANs</u> Enable firewall filtering between VLANs to ensure that only authorized systems are able to communicate with other systems necessary to fulfill their specific responsibilities.			

4.4 Secrets Management

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

Profile Applicability:

- Level 2

Description:

Kubernetes supports mounting secrets as data volumes or as environment variables. Minimize the use of environment variable secrets.

Rationale:

It is reasonably common for application code to log out its environment (particularly in the event of an error). This will include any secret values passed in as environment variables, so secrets can easily be exposed to any user or entity who has access to the logs.

Impact:

Application code which expects to read secrets in the form of environment variables would need modification

Audit:

Run the following command to find references to objects which use environment variables defined from secrets.

```
kubectl get all -o jsonpath='{range .items[?(@..secretKeyRef)]} {.kind}{.metadata.name} {"\n"}{end}' -A
```

Remediation:

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

Default Value:

By default, secrets are not defined

References:

1. <https://kubernetes.io/docs/concepts/configuration/secret/#using-secrets>

Additional Information:

Mounting secrets as volumes has the additional benefit that secret values can be updated without restarting the pod

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.10 Perform Application Layer Filtering Perform application layer filtering. Example implementations include a filtering proxy, application layer firewall, or gateway.			●
v7	14.4 <u>Encrypt All Sensitive Information in Transit</u> Encrypt all sensitive information in transit.		●	●
v7	14.8 <u>Encrypt Sensitive Information at Rest</u> Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.4.2 Consider external secret storage (Manual)

Profile Applicability:

- Level 2

Description:

Consider the use of an external secrets storage and management system, instead of using Kubernetes Secrets directly, if you have more complex secret management needs. Ensure the solution requires authentication to access secrets, has auditing of access to and use of secrets, and encrypts secrets. Some solutions also make it easier to rotate secrets.

Rationale:

Kubernetes supports secrets as first-class objects, but care needs to be taken to ensure that access to secrets is carefully limited. Using an external secrets provider can ease the management of access to secrets, especially where secrets are used across both Kubernetes and non-Kubernetes environments.

Impact:

None

Audit:

Review your secrets management implementation.



Remediation:

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

Default Value:

By default, no external secret management is configured.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 <u>Encrypt Sensitive Data at Rest</u> Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	14.8 Encrypt Sensitive Information at Rest Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			●

4.5 Extensible Admission Control

4.5.1 Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)

Profile Applicability:

- Level 2

Description:

Configure Image Provenance for your deployment.

Rationale:

Kubernetes supports plugging in provenance rules to accept or reject the images in your deployments. You could configure such rules to ensure that only approved images are deployed in the cluster.

See also Recommendation 6.10.5 for GKE specifically.

Impact:

You need to regularly maintain your provenance configuration based on container image updates.

Audit:

Review the pod definitions in your cluster and verify that image provenance is configured as appropriate.

See also Recommendation 6.10.5 for GKE specifically.

Remediation:

Follow the Kubernetes documentation and setup image provenance.

See also Recommendation 6.10.5 for GKE specifically.




Default Value:

By default, image provenance is not set.

References:

1. <https://kubernetes.io/docs/admin/admission-controllers/#imagepolicywebhook>
2. <https://github.com/kubernetes/community/blob/master/contributors/design-proposals/image-provenance.md>
3. <https://hub.docker.com/r/dnurmi/anchore-toolbox/>
4. <https://github.com/kubernetes/kubernetes/issues/22888>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.			
v7	18 <u>Application Software Security</u> Application Software Security			

4.6 General Policies

These policies relate to general cluster management topics, like namespace best practices and policies applied to pod objects in the cluster.

4.6.1 Create administrative boundaries between resources using namespaces (Manual)

Profile Applicability:

- Level 1

Description:

Use namespaces to isolate your Kubernetes objects.

Rationale:

Limiting the scope of user permissions can reduce the impact of mistakes or malicious activities. A Kubernetes namespace allows you to partition created resources into logically named groups. Resources created in one namespace can be hidden from other namespaces. By default, each resource created by a user in Kubernetes cluster runs in a default namespace, called `default`. You can create additional namespaces and attach resources and users to them. You can use Kubernetes Authorization plugins to create policies that segregate access to namespace resources between different users.

Impact:

You need to switch between namespaces for administration.

Audit:

Run the below command and review the namespaces created in the cluster.

```
kubectl get namespaces
```

Ensure that these namespaces are the ones you need and are adequately administered as per your requirements.

Remediation:

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

Default Value:

By default, Kubernetes starts with two initial namespaces:

1. `default` - The default namespace for objects with no other namespace
2. `kube-system` - The namespace for objects created by the Kubernetes system

References:

1. <https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>

2. <http://blog.kubernetes.io/2016/08/security-best-practices-kubernetes-deployment.html>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.8 <u>Establish and Maintain Dedicated Computing Resources for All Administrative Work</u> Establish and maintain dedicated computing resources, either physically or logically separated, for all administrative tasks or tasks requiring administrative access. The computing resources should be segmented from the enterprise's primary network and not be allowed internet access.			●
v7	12.1 <u>Maintain an Inventory of Network Boundaries</u> Maintain an up-to-date inventory of all of the organization's network boundaries.	●	●	●
v7	14 <u>Controlled Access Based on the Need to Know</u> Controlled Access Based on the Need to Know			

4.6.2 Ensure that the seccomp profile is set to docker/default in your pod definitions (Manual)

Profile Applicability:

- Level 2

Description:

Enable `docker/default` seccomp profile in your pod definitions.

Rationale:

Seccomp (secure computing mode) is used to restrict the set of system calls applications can make, allowing cluster administrators greater control over the security of workloads running in the cluster. Kubernetes disables seccomp profiles by default for historical reasons. You should enable it to ensure that the workloads have restricted actions available within the container.

Impact:

If the `docker/default` seccomp profile is too restrictive for you, you would have to create/manage your own seccomp profiles. Also, you need to enable all alpha features for this to work. There is no individual switch to turn on this feature.

Audit:

Review the pod definitions in your cluster. It should create a line as below:

```
annotations:
  seccomp.security.alpha.kubernetes.io/pod: docker/default
```

Remediation:

Seccomp is an alpha feature currently. By default, all alpha features are disabled. So, you would need to enable alpha features in the apiserver by passing "`--feature-gates=AllAlpha=true`" argument.

Edit the `/etc/kubernetes/apiserver` file on the master node and set the `KUBE_API_ARGS` parameter to "`--feature-gates=AllAlpha=true`"

```
KUBE_API_ARGS="--feature-gates=AllAlpha=true"
```

Based on your system, restart the `kube-apiserver` service. For example:

```
systemctl restart kube-apiserver.service
```

Use `annotations` to enable the `docker/default` seccomp profile in your pod definitions. An example is as below:


```

apiVersion: v1
kind: Pod
metadata:
  name: trustworthy-pod
  annotations:
    seccomp.security.alpha.kubernetes.io/pod: docker/default
spec:
  containers:
  - name: trustworthy-container
    image: sotrustworthy:latest

```








Default Value:

By default, seccomp profile is set to `unconfined` which means that no seccomp profiles are enabled.

References:

1. <https://github.com/kubernetes/kubernetes/issues/39845>
2. <https://github.com/kubernetes/kubernetes/pull/21790>
3. <https://github.com/kubernetes/community/blob/master/contributors/design-proposals/seccomp.md#examples>
4. <https://docs.docker.com/engine/security/seccomp/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.7 Use Standard Hardening Configuration Templates for Application Infrastructure Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.			
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.			
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

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

Profile Applicability:

- Level 2

Description:

Apply Security Context to Your Pods and Containers

Rationale:

A security context defines the operating system security settings (uid, gid, capabilities, SELinux role, etc..) applied to a container. When designing your containers and pods, make sure that you configure the security context for your pods, containers, and volumes. A security context is a property defined in the deployment yaml. It controls the security parameters that will be assigned to the pod/container/volume. There are two levels of security context: pod level security context, and container level security context.

Impact:

If you incorrectly apply security contexts, you may have trouble running the pods.

Audit:

Review the pod definitions in your cluster and verify that you have security contexts defined as appropriate.

Remediation:

Follow the Kubernetes documentation and apply security contexts to your pods. For a suggested list of security contexts, you may refer to the CIS Security Benchmark for Docker Containers.




Default Value:

By default, no security contexts are automatically applied to pods.

References:

1. <https://kubernetes.io/docs/concepts/policy/security-context/>
2. <https://learn.cisecurity.org/benchmarks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.			
v7	5 Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers Secure Configuration for Hardware and Software on Mobile Devices, Laptops, Workstations and Servers			

4.6.4 The default namespace should not be used (Manual)

Profile Applicability:

- Level 2

Description:

Kubernetes provides a default namespace, where objects are placed if no namespace is specified for them. Placing objects in this namespace makes application of RBAC and other controls more difficult.

Rationale:

Resources in a Kubernetes cluster should be segregated by namespace, to allow for security controls to be applied at that level and to make it easier to manage resources.

Impact:

None

Audit:

Run this command to list objects in default namespace

```
kubectl get all
```

The only entries there should be system managed resources such as the `kubernetes` service




Remediation:




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

Default Value:

Unless a namespace is specific on object creation, the `default` namespace will be used

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.6 Securely Manage Enterprise Assets and Software Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.			

5 Managed services

This section consists of security recommendations for the direct configuration of Kubernetes managed service components, namely, Google Kubernetes Engine (GKE). These recommendations are directly applicable for features which exist only as part of a managed service.

5.1 Image Registry and Image Scanning

This section contains recommendations relating to container image registries and securing images in those registries, such as Google Container Registry (GCR).

5.1.1 Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider (Automated)

Profile Applicability:

- Level 1

Description:

Scan images stored in Google Container Registry (GCR) for vulnerabilities.

Rationale:

Vulnerabilities in software packages can be exploited by hackers or malicious users to obtain unauthorized access to local cloud resources. GCR Container Analysis and other third party products allow images stored in GCR to be scanned for known vulnerabilities.

Impact:

None.

Audit:

Using Google Cloud Console

1. Go to GCR by visiting <https://console.cloud.google.com/gcr>
2. Select Settings and check if Vulnerability scanning is Enabled.

Using Command Line

```
gcloud services list --enabled --filter containerregistry
```

Ensure that the Container Registry API is listed within the output.

Remediation:

Using Google Cloud Console

1. Go to GCR by visiting <https://console.cloud.google.com/gcr>
2. Select Settings and Click Enable Vulnerability Scanning.

Using Command Line

```
gcloud services enable containerscanning.googleapis.com
```







Default Value:

By default, GCR Container Analysis is disabled.

References:

1. <https://cloud.google.com/container-registry/docs/container-analysis>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets</u> Perform automated vulnerability scans of externally-exposed enterprise assets using a SCAP-compliant vulnerability scanning tool. Perform scans on a monthly, or more frequent, basis.			
v7	<u>3 Continuous Vulnerability Management</u> Continuous Vulnerability Management			
v7	<u>3.1 Run Automated Vulnerability Scanning Tools</u> Utilize an up-to-date SCAP-compliant vulnerability scanning tool to automatically scan all systems on the network on a weekly or more frequent basis to identify all potential vulnerabilities on the organization's systems.			
v7	<u>3.2 Perform Authenticated Vulnerability Scanning</u> Perform authenticated vulnerability scanning with agents running locally on each system or with remote scanners that are configured with elevated rights on the system being tested.			

5.1.2 Minimize user access to GCR (Manual)

Profile Applicability:

- Level 1

Description:

Restrict user access to GCR, limiting interaction with build images to only authorized personnel and service accounts.

Rationale:

Weak access control to GCR may allow malicious users to replace built images with vulnerable or backdoored containers.

Impact:

Care should be taken not to remove access to GCR for accounts that require this for their operation. Any account granted the Storage Object Viewer role at the project level can view all objects stored in GCS for the project.

Audit:

\$PROJECT_ID is a parameter in the audit process below.

Please set the PROJECT_ID equal to the project you would like to audit on your GKE Cluster.

Using Google Cloud Console:
GCR bucket permissions

1. Go to Storage Browser by visiting <https://console.cloud.google.com/storage/browser>
2. From the list of storage buckets, select `artifacts.[PROJECT_ID].appspot.com` for the GCR bucket
3. Under the Permissions tab, review the roles for each member and ensure only authorized users have the Storage Admin, Storage Object Admin, Storage Object Creator, Storage Legacy Bucket Owner, Storage Legacy Bucket Writer and Storage Legacy Object Owner roles.

Users may have permissions to use Service Accounts and thus Users could inherit privileges on the GCR Bucket. To check the accounts that could do this:

1. Go to IAM by visiting <https://console.cloud.google.com/iam-admin/iam>
2. Apply the filter `Role: Service Account User`.

Note that other privileged project level roles will have the ability to write and modify objects and the GCR bucket. Consult the GCP CIS benchmark and IAM documentation for further reference.

Using Command Line:

To check GCR bucket specific permissions

```
gsutil iam get gs://artifacts.$PROJECT_ID.appspot.com
```

The output of the command will return roles associated with the GCR bucket and which members have those roles.

Additionally, run the following to identify users and service accounts that hold privileged roles at the project level, and thus inherit these privileges within the GCR bucket:

```
gcloud projects get-iam-policy $PROJECT_ID \
--flatten="bindings[].members" \
--format='table(bindings.members,bindings.role)' \
--filter="bindings.role:roles/storage.admin OR
bindings.role:roles/storage.objectAdmin OR \
bindings.role:roles/storage.objectCreator OR
bindings.role:roles/storage.legacyBucketOwner OR \
bindings.role:roles/storage.legacyBucketWriter OR
bindings.role:roles/storage.legacyObjectOwner"
```

The output from the command lists the service accounts that have create/modify permissions.

Users may have permissions to use Service Accounts and thus Users could inherit privileges on the GCR Bucket. To check the accounts that could do this:

```
gcloud projects get-iam-policy [PROJECT_ID] \
--flatten="bindings[].members" \
--format='table(bindings.members)' \
--filter="bindings.role:roles/iam.serviceAccountUser"
```

Note that other privileged project level roles will have the ability to write and modify objects and the GCR bucket. Consult the GCP CIS benchmark and IAM documentation for further reference.

Remediation:

Using Google Cloud Console:

To modify roles granted at the GCR bucket level

1. Go to Storage Browser by visiting <https://console.cloud.google.com/storage/browser>
2. From the list of storage buckets, select `artifacts.[PROJECT_ID].appspot.com` for the GCR bucket
3. Under the Permissions tab, modify permissions of the identified member via the drop down role menu and change the Role to Storage Object Viewer for read-only access.

For a User or Service account with Project level permissions inherited by the GCR bucket, or the Service Account User Role:

1. Go to IAM by visiting <https://console.cloud.google.com/iam-admin/iam>
2. Find the User or Service account to be modified and click on the corresponding pencil icon
3. Remove the create/modify role (Storage Admin / Storage Object Admin / Storage Object Creator / Service Account User) on the user or service account
4. If required add the Storage Object Viewer role - note with caution that this permits the account to view all objects stored in GCS for the project.

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]
```







Default Value:

By default, GCR is disabled and access controls are set during initialisation.

References:

1. <https://cloud.google.com/container-registry/docs/access-control>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	14.6 <u>Protect Information through Access Control Lists</u> Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.			

5.1.3 Minimize cluster access to read-only for GCR (Manual)

Profile Applicability:

- Level 1

Description:

Configure the Cluster Service Account with Storage Object Viewer Role to only allow read-only access to GCR.

Rationale:

The Cluster Service Account does not require administrative access to GCR, only requiring pull access to containers to deploy onto GKE. Restricting permissions follows the principles of least privilege and prevents credentials from being abused beyond the required role.

Impact:

A separate dedicated service account may be required for use by build servers and other robot users pushing or managing container images.

Any account granted the Storage Object Viewer role at the project level can view all objects stored in GCS for the project.

Audit:

Using Google Cloud Console

1. Go to Storage Browser by visiting <https://console.cloud.google.com/storage/browser>
2. From the list of storage buckets, select `artifacts.$PROJECT_ID.appspot.com` for the GCR bucket
3. Under the Permissions tab, review the role for GKE Service account and ensure that only the Storage Object Viewer role is set.

Using Command Line

GCR bucket permissions

```
gsutil iam get gs://artifacts.$PROJECT_ID.appspot.com
```

The output of the command will return roles associated with the GCR bucket. If listed, ensure the GKE Service account is set to "role": "roles/storage.objectViewer". If the GKE Service Account has project level permissions that are inherited within the bucket, ensure that these are not privileged:

```
gcloud projects get-iam-policy [PROJECT_ID] \
--flatten="bindings[].members" \
--format='table(bindings.members,bindings.role)' \
--filter="bindings.role:roles/storage.admin OR
bindings.role:roles/storage.objectAdmin OR \
bindings.role:roles/storage.objectCreator OR
bindings.role:roles/storage.legacyBucketOwner OR \
bindings.role:roles/storage.legacyBucketWriter OR
bindings.role:roles/storage.legacyObjectOwner"
```

Your GKE Service Account should not be output when this command is run.

Remediation:

Using Google Cloud Console

For an account explicitly granted access to the bucket:

1. Go to Storage Browser by visiting <https://console.cloud.google.com/storage/browser>
2. From the list of storage buckets, select `artifacts.[PROJECT_ID].appspot.com` for the GCR bucket
3. Under the Permissions tab, modify permissions of the identified GKE Service Account via the drop-down role menu and change to the Role to Storage Object Viewer for read-only access.

For an account that inherits access to the bucket through Project level permissions:

1. Go to IAM console by visiting <https://console.cloud.google.com/iam-admin>
2. From the list of accounts, identify the required service account and select the corresponding pencil icon
3. Remove the Storage Admin / Storage Object Admin / Storage Object Creator roles.
4. Add the Storage Object Viewer role- note with caution that this permits the account to view all objects stored in GCS for the project.
5. Click `SAVE`

Using Command Line

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]
```






Default Value:

The default permissions for the cluster Service account is dependent on the initial configuration and IAM policy.

References:

1. <https://cloud.google.com/container-registry/docs/access-control>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	3.2 <u>Perform Authenticated Vulnerability Scanning</u> Perform authenticated vulnerability scanning with agents running locally on each system or with remote scanners that are configured with elevated rights on the system being tested.			

5.1.4 Minimize Container Registries to only those approved (Manual)

Profile Applicability:

- Level 2

Description:

Use Binary Authorization to allowlist (whitelist) only approved container registries.

Rationale:

Allowing unrestricted access to external container registries provides the opportunity for malicious or unapproved containers to be deployed into the cluster. Allowlisting only approved container registries reduces this risk.

See also Recommendation 6.10.5.

Impact:

All container images to be deployed to the cluster must be hosted within an approved container image registry. If public registries are not on the allowlist, a process for bringing commonly used container images into an approved private registry and keeping them up to date will be required.

Audit:*Using Google Cloud Console:*

Check that Binary Authorization is enabled for the GKE cluster:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the cluster and on the Details pane, ensure that Binary Authorization is set to 'Enabled'.

Then assess the contents of the policy:

1. Go to Binary Authorization by visiting <https://console.cloud.google.com/security/binary-authorization>
2. Ensure the project default rule is not set to 'Allow all images' under Policy deployment rules.
3. Review the list of 'Images exempt from policy' for unauthorized container registries.

Using Command Line:

Check that Binary Authorization is enabled for the GKE cluster:

```
gcloud container clusters describe $CLUSTER_NAME --zone $COMPUTE_ZONE --format json | jq .binaryAuthorization
```

This will return the following if Binary Authorization is enabled:

```
{
  "enabled": true
}
```

Then assess the contents of the policy:

```
gcloud container binauthz policy export > current-policy.yaml
```

Ensure that the current policy is not configured to allow all images (`evaluationMode: ALWAYS_ALLOW`).

Review the list of `admissionWhitelistPatterns` for unauthorized container registries.

```
cat current-policy.yaml
admissionWhitelistPatterns:
...
defaultAdmissionRule:
  evaluationMode: ALWAYS_ALLOW
```

Remediation:

Using Google Cloud Console:

1. Go to Binary Authorization by visiting <https://console.cloud.google.com/security/binary-authorization>
2. Enable Binary Authorization API (if disabled)
3. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
4. Select Kubernetes cluster for which Binary Authorization is disabled
5. Click EDIT
6. Set Binary Authorization to 'Enabled'
7. Click SAVE
8. Return to the Binary Authorization by visiting <https://console.cloud.google.com/security/binary-authorization>
9. Set an appropriate policy for your cluster and enter the approved container registries under 'Image paths'.

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]
```







Default Value:

By default, Binary Authorization is disabled along with container registry allowlisting.

References:

1. <https://cloud.google.com/binary-authorization/docs/policy-yaml-reference>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.5 Allowlist Authorized Software Use technical controls, such as application allowlisting, to ensure that only authorized software can execute or be accessed. Reassess bi-annually, or more frequently.			
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			
v7	5.3 Securely Store Master Images Store the master images and templates on securely configured servers, validated with integrity monitoring tools, to ensure that only authorized changes to the images are possible.			

5.2 Identity and Access Management (IAM)

This section contains recommendations relating to using Cloud IAM with GKE.

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

Profile Applicability:

- Level 1

Description:

Create and use minimally privileged Service accounts to run GKE cluster nodes instead of using the Compute Engine default Service account. Unnecessary permissions could be abused in the case of a node compromise.

Rationale:

A GCP service account (as distinct from a Kubernetes ServiceAccount) is an identity that an instance or an application can use to run GCP API requests on your behalf. This identity is used to identify virtual machine instances to other Google Cloud Platform services. By default, Kubernetes Engine nodes use the Compute Engine default service account. This account has broad access by default, as defined by access scopes, making it useful to a wide variety of applications on the VM, but it has more permissions than are required to run your Kubernetes Engine cluster.

You should create and use a minimally privileged service account to run your Kubernetes Engine cluster instead of using the Compute Engine default service account, and create separate service accounts for each Kubernetes Workload (See Recommendation 6.2.2).

Kubernetes Engine requires, at a minimum, the node service account to have the monitoring.viewer, monitoring.metricWriter, and logging.logWriter roles. Additional roles may need to be added for the nodes to pull images from GCR.

Impact:

Instances are automatically granted the <https://www.googleapis.com/auth/cloud-platform> scope to allow full access to all Google Cloud APIs. This is so that the IAM permissions of the instance are completely determined by the IAM roles of the Service account. Thus if Kubernetes workloads were using cluster access scopes to perform actions using Google APIs, they may no longer be able to, if not permitted by the permissions of the Service account. To remediate, follow Recommendation 6.2.2.

The Service account roles listed here are the minimum required to run the cluster. Additional roles may be required to pull from a private instance of Google Container Registry (GCR).

Audit:

The audit script for this recommendation utilizes 3 variables:

\$NODE_POOL

\$CLUSTER_NAME

\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the cluster under test and click on each Node pool to bring up the Node pool details page. Ensure that for each Node pool the Service account is not set to default under the Security heading.

To check the permissions allocated to the service account are the minimum required for cluster operation:

1. Go to IAM by visiting <https://console.cloud.google.com/iam-admin/iam>
2. From the list of Service accounts, ensure each cluster Service account has only the following roles:
 - Logs Writer
 - Monitoring Metric Writer
 - Monitoring Viewer

Using Command line:

To check which Service account is set for an existing cluster, run the following command:

```
gcloud container node-pools describe $NODE_POOL \
  --cluster $CLUSTER_NAME --zone $COMPUTE_ZONE \
  --format json | jq '.config.serviceAccount'
```

The output of the above command will return default if default Service account is used for Project access.

To check that the permissions allocated to the service account are the minimum required for cluster operation:

```
gcloud projects get-iam-policy [PROJECT_ID] \
  --flatten="bindings[].members" \
  --format='table(bindings.role)' \
  --filter="bindings.members:[SERVICE_ACCOUNT]"
```

Review the output to ensure that the service account only has the roles required to run the cluster:

- roles/logging.logWriter

- `roles/monitoring.metricWriter`
- `roles/monitoring.viewer`

Remediation:

Using Google Cloud Console:

To create a minimally privileged service account.

1. Go to Service Accounts by visiting <https://console.cloud.google.com/iam-admin/serviceaccounts>
2. Click on CREATE SERVICE ACCOUNT
3. Enter Service Account Details
4. Click CREATE
5. Within Service Account permissions add the following roles:
 - Logs Writer
 - Monitoring Metric Writer
 - Monitoring Viewer
6. Click CONTINUE
7. Grant users access to this service account and create keys as required
8. Click DONE.

To create a Node pool to use the Service account:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the cluster name within which the Node pool will be launched
3. Click on ADD NODE POOL
4. Within the Node Pool options select the minimally privileged service account from the Service Account drop down under the 'Security' heading
5. Click SAVE to launch the Node pool.

You will need to migrate your workloads to the new Node pool, and delete Node pools that use the default service account to complete the remediation.

Using Command Line:

To create a minimally privileged service account:

```
gcloud iam service-accounts create [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]
```

You will need to migrate your workloads to the new Node pool, and delete Node pools that use the default service account to complete the remediation.




Default Value:

By default, nodes use the Compute Engine default service account when you create a new cluster.

References:

1. https://cloud.google.com/compute/docs/access/service-accounts#compute_engine_default_service_account

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	4.3 <u>Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

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

Profile Applicability:

- Level 1

Description:

Kubernetes workloads should not use cluster node service accounts to authenticate to Google Cloud APIs. Each Kubernetes Workload that needs to authenticate to other Google services using Cloud IAM should be provisioned a dedicated Service account. Enabling Workload Identity manages the distribution and rotation of Service account keys for the workloads to use.

Rationale:

Manual approaches for authenticating Kubernetes workloads running on GKE against Google Cloud APIs are: storing service account keys as a Kubernetes secret (which introduces manual key rotation and potential for key compromise); or use of the underlying nodes' IAM Service account, which violates the principle of least privilege on a multitenanted node, when one pod needs to have access to a service, but every other pod on the node that uses the Service account does not.

Once a relationship between a Kubernetes Service account and a GCP Service account has been configured, any workload running as the Kubernetes Service account automatically authenticates as the mapped GCP Service account when accessing Google Cloud APIs on a cluster with Workload Identity enabled.

Impact:

During the Workload Identity beta, a GCP project can have a maximum of 20 clusters with Workload Identity enabled.

Workload Identity replaces the need to use Metadata Concealment and as such, the two approaches are incompatible. The sensitive metadata protected by Metadata Concealment is also protected by Workload Identity.

When Workload Identity is enabled, you can no longer use the Compute Engine default Service account. Correspondingly, Workload Identity can't be used with Pods running in the host network. You may also need to modify workloads in order for them to use Workload Identity as described within <https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity>

GKE infrastructure pods such as Stackdriver will continue to use the Node's Service account.

Audit:

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on each cluster to bring up the Details pane, make sure for each cluster Workload Identity is set to 'Enabled' under the 'Cluster' section and ensure that the Workload Identity Namespace is set to the namespace of the GCP project containing the cluster, e.g:
`$PROJECT_ID.svc.id.goog`
3. Additionally, click on each Node pool within each cluster to observe the Node pool Details pane, and ensure that the GKE Metadata Server is 'Enabled'.

Using Command Line:

```
gcloud beta container clusters describe $CLUSTER_NAME --zone $CLUSTER_ZONE
```

If Workload Identity is enabled, the following fields should be present, and the `$PROJECT_ID` should be set to the namespace of the GCP project containing the cluster:

```
workloadIdentityConfig:  
  identityNamespace:[PROJECT_ID].svc.id.goog
```

For each Node pool, ensure the following is set.

```
workloadMetadataConfig:  
  nodeMetadata: GKE_METADATA_SERVER
```

You will also need to manually audit each Kubernetes workload requiring Google Cloud API access to ensure that Workload Identity is being used and not some other method.

Remediation:

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, select your cluster for which Workload Identity is disabled.
3. Click on EDIT
4. Set Workload Identity to 'Enabled' and set the Workload Identity Namespace to the namespace of the Cloud project containing the cluster, e.g:
`[PROJECT_ID].svc.id.goog`
5. Click SAVE and wait for the cluster to update
6. Once the cluster has updated, select each Node pool within the cluster Details page
7. For each Node pool, select EDIT within the Node pool Details page
8. Within the Edit node pool pane, check the 'Enable GKE Metadata Server' checkbox and click SAVE.

Using Command Line:

```
gcloud beta container clusters update [CLUSTER_NAME] --zone [CLUSTER_ZONE] \
--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`.

Then, modify existing Node pools to enable `GKE_METADATA_SERVER`:

```
gcloud beta container node-pools update [NODEPOOL_NAME] \
--cluster=[CLUSTER_NAME] --zone [CLUSTER_ZONE] \
--workload-metadata-from-node=GKE_METADATA_SERVER
```

You may also need to modify workloads 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 your hosted workloads as Node pools are updated, it may be more appropriate to create new Node Pools.




Default Value:

By default, Workload Identity is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	4.3 <u>Ensure the Use of Dedicated Administrative Accounts</u> Ensure that all users with administrative account access use a dedicated or secondary account for elevated activities. This account should only be used for administrative activities and not internet browsing, email, or similar activities.			

5.3 Cloud Key Management Service (Cloud KMS)

This section contains recommendations relating to using Cloud KMS with GKE.

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

Profile Applicability:

- Level 1

Description:

Encrypt Kubernetes secrets, stored in etcd, at the application-layer using a customer-managed key in Cloud KMS.

Rationale:

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.

Application-layer Secrets Encryption provides an additional layer of security for sensitive data, such as user defined Secrets and Secrets required for the operation of the cluster, such as service account keys, which are all stored in etcd.

Using this functionality, you can use a key, that you manage in Cloud KMS, to encrypt data at the application layer. This protects against attackers in the event that they manage to gain access to etcd.

Impact:

To use the Cloud KMS CryptoKey to protect etcd in the cluster, the 'Kubernetes Engine Service Agent' Service account must hold the 'Cloud KMS CryptoKey Encrypter/Decrypter' role.

Audit:

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on each cluster to bring up the Details pane, and ensure Application-layer Secrets Encryption is set to 'Enabled'.

Using Command Line:

```
gcloud container clusters describe $CLUSTER_NAME --format json | jq  
'databaseEncryption'
```

If configured correctly, the output from the command returns a response containing the following detail:

```
keyName=projects/[PROJECT_ID]/locations/[LOCATION]/keyRings/[RING_NAME]/cryptoKeys/[KEY_NAME]  
state=ENCRYPTED
```

Remediation:

To enable Application-layer Secrets Encryption, several configuration items are required. These include:

- A key ring
- A key
- A GKE service account with Cloud KMS CryptoKey Encrypter/Decrypter role

Once these are created, Application-layer Secrets Encryption can be enabled on an existing or new cluster.

Using Google Cloud Console:

To create a key

1. Go to Cloud KMS by visiting <https://console.cloud.google.com/security/kms>
2. Select CREATE KEY RING
3. Enter a Key ring name and the region where the keys will be stored
4. Click CREATE
5. Enter a Key name and appropriate rotation period within the Create key pane
6. Click CREATE

To enable on a new cluster

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER
3. Expand the template by clicking 'Availability, networking, security, and additional features' and check the 'Enable Application-layer Secrets Encryption' checkbox.
4. Select the desired Key as the customer-managed key and if prompted grant permissions to the GKE Service account
5. Click CREATE.

To enable on an existing cluster

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click to edit cluster you want to modify.
3. Enable Application-layer Secrets Encryption and choose the desired Key
4. Click SAVE.

Using Command Line:

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]/cryptoKey  
s/[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]/cryptoKey  
s/[KEY_NAME] \  
  --project [CLUSTER_PROJECT_ID]
```




Default Value:

By default, Application-layer Secrets Encryption is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/encrypting-secrets>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 <u>Encrypt Sensitive Data at Rest</u> Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.			
v7	14.8 <u>Encrypt Sensitive Information at Rest</u> Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			

5.4 Node Metadata

This section contains recommendations relating to node metadata in GKE.

5.4.1 Ensure legacy Compute Engine instance metadata APIs are Disabled (Automated)

Profile Applicability:

- Level 1

Description:

Disable the legacy GCE instance metadata APIs for GKE nodes. Under some circumstances, these can be used from within a pod to extract the node's credentials.

Rationale:

The legacy GCE metadata endpoint allows simple HTTP requests to be made returning sensitive information. To prevent the enumeration of metadata endpoints and data exfiltration, the legacy metadata endpoint must be disabled.

Without requiring a custom HTTP header when accessing the legacy GCE metadata endpoint, a flaw in an application that allows an attacker to trick the code into retrieving the contents of an attacker-specified web URL could provide a simple method for enumeration and potential credential exfiltration. By requiring a custom HTTP header, the attacker needs to exploit an application flaw that allows them to control the URL and also add custom headers in order to carry out this attack successfully.

Impact:

Any workloads using the legacy GCE metadata endpoint will no longer be able to retrieve metadata from the endpoint. Use Workload Identity instead.

Audit:

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, make sure that the Details page for each Node pool displays under the 'GCE instance metadata' heading: `disable-legacy-endpoints: true`.

Using Command Line:

To check if the legacy metadata API is disabled for an individual Node pool, run the following command:

```
gcloud container node-pools describe $NODE_POOL \
  --cluster $CLUSTER_NAME --zone $COMPUTE_ZONE \
  --format json | jq '.config.metadata'
```

Alternatively to audit all of the clusters Node pools simultaneously, run the following command:

```
gcloud container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE \
  --format json | jq .nodePools[].config.metadata
```

For each of the Node pools with the correct setting the output of the above command returns:

```
"disable-legacy-endpoints": "true"
```

Remediation:

The legacy GCE metadata endpoint must be disabled upon the cluster or node-pool creation. For GKE versions 1.12 and newer, the legacy GCE metadata endpoint is disabled by default.

Using Google Cloud Console:

To update an existing cluster, create a new Node pool with the legacy GCE metadata endpoint disabled:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the name of cluster to be upgraded and click ADD NODE POOL.
3. Ensure that GCE instance metadata is set to the key:value pair of `disable-legacy-endpoints: true`
4. Click SAVE

You will need to migrate workloads from any existing non-conforming Node pools, to the new Node pool, then delete non-conforming Node pools to complete the remediation.

Using Command Line:

To update an existing cluster, create a new Node pool with the legacy GCE metadata endpoint disabled:

```
gcloud container node-pools create [POOL_NAME] \
  --metadata disable-legacy-endpoints=true \
  --cluster [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE]
```

You will need to migrate workloads from any existing non-conforming Node pools, to the new Node pool, then delete non-conforming Node pools to complete the remediation.





Default Value:

Note: In GKE cluster versions 1.12 and newer, the `--metadata=disable-legacy-endpoints=true` setting is automatically enabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/protecting-cluster-metadata#disable-legacy-apis>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>16.7 Use Standard Hardening Configuration Templates for Application Infrastructure</u> Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.			
v7	<u>5.2 Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

5.4.2 Ensure the GKE Metadata Server is Enabled (Automated)

Profile Applicability:

- Level 2

Description:

Running the GKE Metadata Server prevents workloads from accessing sensitive instance metadata and facilitates Workload Identity

Rationale:

Every node stores its metadata on a metadata server. Some of this metadata, such as kubelet credentials and the VM instance identity token, is sensitive and should not be exposed to a Kubernetes workload. Enabling the GKE Metadata server prevents pods (that are not running on the host network) from accessing this metadata and facilitates Workload Identity.

When unspecified, the default setting allows running pods to have full access to the node's underlying metadata server.

Impact:

The GKE Metadata Server must be run when using Workload Identity. Because Workload Identity replaces the need to use Metadata Concealment, the two approaches are incompatible.

When the GKE Metadata Server and Workload Identity are enabled, unless the Pod is running on the host network, Pods cannot use the the Compute Engine default service account.

You may also need to modify workloads in order for them to use Workload Identity as described within <https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity>.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the name of the cluster of interest and for each Node pool within the cluster, open the Details pane, and ensure that the GKE Metadata Server is set to 'Enabled'.

Using Command Line

To check whether the GKE Metadata Server is enabled for each Node pool within a cluster, run the following command:

```
gcloud beta container clusters describe $CLUSTER_NAME \
  --zone $CLUSTER_ZONE \
  --format json | jq .nodePools[].config.workloadMetadataConfig
```

This should return the following for each Node pool:

```
{
  "nodeMetadata": GKE_METADATA_SERVER
}
```

Null ({ }) is returned if the GKE Metadata Server is not enabled.

Remediation:

The GKE Metadata Server requires Workload Identity to be enabled on a cluster. Modify the cluster to enable Workload Identity and enable the GKE Metadata Server.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, select the cluster for which Workload Identity is disabled.
3. Click on EDIT
4. Set Workload Identity to 'Enabled' and set the Workload Identity Namespace to the namespace of the Cloud project containing the cluster, e.g:
[PROJECT_ID].svc.id.goog
5. Click SAVE and wait for the cluster to update
6. Once the cluster has updated, select each Node pool within the cluster Details page
7. For each Node pool, select EDIT within the Node pool details page
8. Within the Edit node pool pane, check the 'Enable GKE Metadata Server checkbox'
9. Click SAVE.

Using Command Line

```
gcloud beta 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 beta container node-pools update [NODEPOOL_NAME] \
  --cluster=[CLUSTER_NAME] \
  --workload-metadata-from-node=GKE_METADATA_SERVER
```

You may also need to modify workloads in order for them to use Workload Identity as described within <https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity>.





Default Value:

By default, running pods to have full access to the node's underlying metadata server.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/protecting-cluster-metadata#concealment>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>16.7 Use Standard Hardening Configuration Templates for Application Infrastructure</u> Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.			
v7	<u>5.2 Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.			

5.5 Node Configuration and Maintenance

This section contains recommendations relating to node configurations in GKE.

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

Profile Applicability:

- Level 2

Description:

Use Container-Optimized OS (cos_containerd) as a managed, optimized and hardened base OS that limits the host's attack surface.

Rationale:

COS is an operating system image for Compute Engine VMs optimized for running containers. With COS, you can bring up your containers on Google Cloud Platform quickly, efficiently, and securely.

Using COS as the node image provides the following benefits:

- Run containers out of the box: COS instances come pre-installed with the container runtime and `cloud-init`. With a COS instance, you can bring up your container at the same time you create your VM, with no on-host setup required.
- Smaller attack surface: COS has a smaller footprint, reducing your instance's potential attack surface.
- Locked-down by default: COS instances include a locked-down firewall and other security settings by default.

Impact:

If modifying an existing cluster's Node pool to run COS, the upgrade operation used is long-running and will block other operations on the cluster (including delete) until it has run to completion.

COS nodes also provide an option with `containerd` as the main container runtime directly integrated with Kubernetes instead of `docker`. Thus, on these nodes, Docker cannot view or access containers or images managed by Kubernetes. Your applications should not interact with Docker directly. For general troubleshooting or debugging, use `crictl` instead.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, select the cluster under test.
3. Under the 'Node pools' section, make sure that for each of the Node pools, 'Container-Optimized OS (cos_containerd)' is listed in the 'Image type' column.

Using Command line

To check Node image type for an existing cluster's Node pool:

```
gcloud container node-pools describe $NODE_POOL \
--cluster $CLUSTER_NAME --zone $COMPUTE_ZONE \
--format json | jq '.config.imageType'
```

The output of the above command returns `cos`, if COS is used for Node images.

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the Kubernetes cluster which does not use COS
3. Under the Node pools heading, select the Node Pool that requires alteration
4. Click EDIT
5. Under the Image Type heading click CHANGE
6. From the pop-up menu select Container-Optimized OS (cos_containerd) and click CHANGE
7. Repeat for all non-compliant Node pools.

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 [POOL_NAME]
```



Default Value:

Container-Optimized OS (cos_containerd) is the default option for a cluster node image.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/concepts/node-images>
2. <https://cloud.google.com/container-optimized-os/docs/>
3. <https://cloud.google.com/container-optimized-os/docs/concepts/security>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.5 Allowlist Authorized Software Use technical controls, such as application allowlisting, to ensure that only authorized software can execute or be accessed. Reassess bi-annually, or more frequently.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.2 Maintain Secure Images Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

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

Profile Applicability:

- Level 1

Description:

Nodes in a degraded state are an unknown quantity and so may pose a security risk.

Rationale:

Kubernetes Engine's node auto-repair feature helps you keep the nodes in your cluster in a healthy, running state. When enabled, Kubernetes Engine makes periodic checks on the health state of each node in your cluster. If a node fails consecutive health checks over an extended time period, Kubernetes Engine initiates a repair process for that node.

Impact:

If multiple nodes require repair, Kubernetes Engine might repair them in parallel. Kubernetes Engine limits number of repairs depending on the size of the cluster (bigger clusters have a higher limit) and the number of broken nodes in the cluster (limit decreases if many nodes are broken).

Node auto-repair is not available on Alpha Clusters.

Audit:

This Audit process utilize 3 parameterized variables.

Please set your environment to include the following Parameters.

\$POOL_NAME

\$CLUSTER_NAME

\$COMPUTE_ZONE

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, select the desired cluster. For each Node pool, view the Node pool Details pane and ensure that under the 'Management' heading, 'Auto-repair' is set to 'Enabled'.

Using Command Line:

To check the existence of node auto-repair for an existing cluster's Node pool, run:

```
gcloud container node-pools describe $POOL_NAME --cluster $CLUSTER_NAME --zone $COMPUTE_ZONE --format json | jq '.management'
```

Ensure the output of the above command has JSON key attribute `autoRepair` set to `true`:

```
{
  "autoRepair": true
}
```

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which node auto-repair is disabled
3. Click on the name of the Node pool that requires node auto-repair to be enabled
4. Within the Node pool details pane click EDIT
5. Under the 'Management' heading, ensure the 'Enable Auto-repair' box is checked.
6. Click SAVE.

Using Command Line

To enable node auto-repair for an existing cluster with Node pool, run the following command:

```
gcloud container node-pools update $POOL_NAME --cluster $CLUSTER_NAME --zone $COMPUTE_ZONE --enable-autorepair
```



Default Value:



Node auto-repair is enabled by default.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/concepts/node-auto-repair>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets</u> Perform automated vulnerability scans of externally-exposed enterprise assets using a SCAP-compliant vulnerability scanning tool. Perform scans on a monthly, or more frequent, basis.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	3.1 Run Automated Vulnerability Scanning Tools Utilize an up-to-date SCAP-compliant vulnerability scanning tool to automatically scan all systems on the network on a weekly or more frequent basis to identify all potential vulnerabilities on the organization's systems.			

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

Profile Applicability:

- Level 1

Description:

Node auto-upgrade keeps nodes at the current Kubernetes and OS security patch level to mitigate known vulnerabilities.

Rationale:

Node auto-upgrade helps you keep the nodes in your cluster or Node pool up to date with the latest stable patch version of Kubernetes as well as the underlying node operating system. Node auto-upgrade uses the same update mechanism as manual node upgrades.

Node pools with node auto-upgrade enabled are automatically scheduled for upgrades when a new stable Kubernetes version becomes available. When the upgrade is performed, the Node pool is upgraded to match the current cluster master version. From a security perspective, this has the benefit of applying security updates automatically to the Kubernetes Engine when security fixes are released.

Impact:

Enabling node auto-upgrade does not cause your nodes to upgrade immediately. Automatic upgrades occur at regular intervals at the discretion of the Kubernetes Engine team.

To prevent upgrades occurring during a peak period for your cluster, you should define a maintenance window. A maintenance window is a four-hour timeframe that you choose in which automatic upgrades should occur. Upgrades can occur on any day of the week, and at any time within the timeframe. To prevent upgrades from occurring during certain dates, you should define a maintenance exclusion. A maintenance exclusion can span multiple days.

Audit:

This Audit process utilize 3 parameterized variables.
please set your environment to include the following Parmaters.

\$POOL_NAME

\$CLUSTER_NAME

\$COMPUTE_ZONE

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting
<https://console.cloud.google.com/kubernetes/list>

2. From the list of clusters, select the desired cluster. For each Node pool, view the Node pool Details pane and ensure that under the 'Management' heading, 'Auto-upgrade' is set to 'Enabled'.

Using Command Line

To check the existence of node auto-upgrade for an existing cluster's Node pool, run:

```
gcloud container node-pools describe [NODE_POOL_NAME] \
  --cluster [CLUSTER_NAME] --zone [COMPUTE_ZONE] \
  --format json | jq '.management'
```

Ensure the output of the above command has JSON key attribute `autoUpgrade` set to `true`:

```
{
  "autoUpgrade": true
}
```

If node auto-upgrade is disabled, the output of the above command output will not contain the `autoUpgrade` entry.

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which node auto-upgrade is disabled
3. Click on the name of the Node pool that requires node auto-repair to be enabled
4. Within the Node pool details pane click EDIT
5. Under the 'Management' heading, ensure the 'Enable auto-repair' box is checked.
Click SAVE.

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] \
  --cluster [CLUSTER_NAME] --zone [COMPUTE_ZONE] \
  --enable-autoupgrade
```

Default Value:

Node auto-upgrade is enabled by default.

Even if a cluster has been created with node auto-repair enabled, this only applies to the default Node pool. Subsequent node pools do not have node auto-upgrade enabled by default.

References:













1. <https://cloud.google.com/kubernetes-engine/docs/concepts/node-auto-upgrades>

2. <https://cloud.google.com/kubernetes-engine/docs/how-to/maintenance-windows-and-exclusions>

Additional Information:

Node auto-upgrades is not available for Alpha Clusters.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	<u>7.3 Perform Automated Operating System Patch Management</u> Perform operating system updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.			
v7	<u>2.2 Ensure Software is Supported by Vendor</u> Ensure that only software applications or operating systems currently supported by the software's vendor are added to the organization's authorized software inventory. Unsupported software should be tagged as unsupported in the inventory system.			
v7	<u>3.4 Deploy Automated Operating System Patch Management Tools</u> Deploy automated software update tools in order to ensure that the operating systems are running the most recent security updates provided by the software vendor.			
v7	<u>3.5 Deploy Automated Software Patch Management Tools</u> Deploy automated software update tools in order to ensure that third-party software on all systems is running the most recent security updates provided by the software vendor.			

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

Profile Applicability:

- Level 1

Description:

Subscribe to the Regular or Stable Release Channel to automate version upgrades to the GKE cluster and to reduce version management complexity to the number of features and level of stability required.

Rationale:

Release Channels signal a graduating level of stability and production-readiness. These are based on observed performance of GKE clusters running that version and represent experience and confidence in the cluster version.

The Regular release channel upgrades every few weeks and is for production users who need features not yet offered in the Stable channel. These versions have passed internal validation, but don't have enough historical data to guarantee their stability. Known issues generally have known workarounds.

The Stable release channel upgrades every few months and is for production users who need stability above all else, and for whom frequent upgrades are too risky. These versions have passed internal validation and have been shown to be stable and reliable in production, based on the observed performance of those clusters.

Critical security patches are delivered to all release channels.

Impact:

Once release channels are enabled on a cluster, they cannot be disabled. To stop using release channels, you must recreate the cluster without the `--release-channel` flag.

Node auto-upgrade is enabled (and cannot be disabled), so your cluster is updated automatically from releases available in the chosen release channel.

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, select the desired cluster
3. Within the Details pane, if using a release channel, the release channel should be set to the 'Regular' or 'Stable' channel.

Using Command Line

Run the following command:

```
gcloud beta container clusters describe [CLUSTER_NAME] \
--zone [COMPUTE_ZONE] \
--format json | jq .releaseChannel.channel
```

The output of the above command will return `regular` or `stable` if these release channels are being used to manage automatic upgrades for your cluster.

Remediation:

Currently, cluster Release Channels are only configurable at cluster provisioning time.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting:
<https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER
3. Under the 'Master Version' heading, click the 'Use Release Channels' button
4. Select the 'Regular' or 'Stable' channels from the 'Release Channel' drop down menu
5. Configure the rest of the cluster settings as desired
6. Click CREATE.

Using Command Line

Create a new cluster by running the following command:

```
gcloud beta container clusters create [CLUSTER_NAME] \
--zone [COMPUTE_ZONE] \
--release-channel [RELEASE_CHANNEL]
```

where `[RELEASE_CHANNEL]` is `stable` or `regular` according to your needs.




Default Value:







Currently, release channels are not enabled by default.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.4 Perform Automated Application Patch Management Perform application updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	3.4 <u>Deploy Automated Operating System Patch Management Tools</u> Deploy automated software update tools in order to ensure that the operating systems are running the most recent security updates provided by the software vendor.			
v7	3.5 <u>Deploy Automated Software Patch Management Tools</u> Deploy automated software update tools in order to ensure that third-party software on all systems is running the most recent security updates provided by the software vendor.			

5.5.5 Ensure Shielded GKE Nodes are Enabled (Manual)

Profile Applicability:

- Level 1

Description:

Shielded GKE Nodes provides verifiable integrity via secure boot, virtual trusted platform module (vTPM)-enabled measured boot, and integrity monitoring.

Rationale:

Shielded GKE nodes protects clusters against boot- or kernel-level malware or rootkits which persist beyond infected OS.

Shielded GKE nodes run firmware which is signed and verified using Google's Certificate Authority, ensuring that the nodes' firmware is unmodified and establishing the root of trust for Secure Boot. GKE node identity is strongly protected via virtual Trusted Platform Module (vTPM) and verified remotely by the master node before the node joins the cluster. Lastly, GKE node integrity (i.e., boot sequence and kernel) is measured and can be monitored and verified remotely.

Impact:

After you enable Shielded GKE Nodes in a cluster, any nodes created in a Node pool without Shielded GKE Nodes enabled, or created outside of any Node pool, aren't able to join the cluster.

Shielded GKE Nodes can only be used with Container-Optimized OS (COS), COS with containerd, and Ubuntu node images.

Audit:

This Audit process utilize 1 parameterized variable.
Please set your environment to include the following Parameters.
\$CLUSTER_NAME

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the cluster under test from the list of clusters, and ensure that Shielded GKE Nodes are 'Enabled' under the Details pane.

Using Command Line:

Run the following command:

```
gcloud beta container clusters describe $CLUSTER_NAME \
  --format json | jq '.shieldedNodes'
```

This will return the following if Shielded GKE Nodes are enabled:

```
{
  "enabled": true
}
```

Remediation:

Using Google Cloud Console:

To update an existing cluster to use Shielded GKE nodes:

1. Navigate to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the cluster which you wish to enable Shielded GKE Nodes and Click EDIT
3. Locate the 'Shielded GKE Nodes' drop-down menu and select 'Enabled'
4. Click SAVE.

Using Command Line:

To migrate an existing cluster, you will need to specify the `--enable-shielded-nodes` flag on a cluster update command:

```
gcloud beta container clusters update $CLUSTER_NAME \
  --zone $CLUSTER_ZONE \
  --enable-shielded-nodes
```

Default Value:



Currently, Shielded GKE Nodes are not enabled by default.

If Shielded GKE Nodes are enabled, Integrity Monitoring (through Stackdriver) is enabled by default and Secure Boot is disabled by default.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/shielded-gke-nodes>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.3 Securely Store Master Images Store the master images and templates on securely configured servers, validated with integrity monitoring tools, to ensure that only authorized changes to the images are possible.			

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

Profile Applicability:

- Level 1

Description:

Enable Integrity Monitoring for Shielded GKE Nodes to be notified of inconsistencies during the node boot sequence.

Rationale:

Integrity Monitoring provides active alerting for Shielded GKE nodes which allows administrators to respond to integrity failures and prevent compromised nodes from being deployed into the cluster.

Impact:

None.

Audit:

This Audit process utilize 3 parameterized variables.

Please set your environment to include the following Parameters.

\$POOL_NAME

\$CLUSTER_NAME

\$COMPUTE_ZONE

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the name of the cluster under test.
3. Open the Details pane for each Node pool within the cluster, and ensure that 'Integrity monitoring' is set to 'Enabled' under the Security heading.

Using Command Line

To check if Integrity Monitoring is enabled for the Node pools in your cluster, run the following command for each Node pool:

```
gcloud beta container node-pools describe $POOL_NAME \
  --cluster $CLUSTER_NAME --zone $COMPUTE_ZONE \
  --format json | jq '.config.shieldedInstanceConfig'
```

This will return

```
{
  "enableIntegrityMonitoring": true
}
```

if Integrity Monitoring is enabled.

Remediation:

Once a Node pool is provisioned, it cannot be updated to enable Integrity Monitoring. You must create new Node pools within the cluster with Integrity Monitoring enabled Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the cluster requiring the update and click ADD NODE POOL
3. Ensure that the 'Integrity monitoring' checkbox is checked under the 'Shielded options' Heading.
4. Click SAVE.

You will also need to migrate workloads from existing non-conforming Node pools to the newly created Node pool, then the non-conforming pools.

Using Command Line

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

```
gcloud beta container node-pools create [NODEPOOL_NAME] \
  --cluster [CLUSTER_NAME] --zone [COMPUTE_ZONE] \
  --shielded-integrity-monitoring
```

You will also need to migrate workloads from existing non-conforming Node pools to the newly created Node pool, then delete the non-conforming pools.



Default Value:

Integrity Monitoring is disabled by default on GKE clusters. Integrity Monitoring is enabled by default for Shielded GKE Nodes; however, if Secure Boot is enabled at creation time, Integrity Monitoring is disabled.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/shielded-gke-nodes#system_integrity_monitoring

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.3 Securely Store Master Images Store the master images and templates on securely configured servers, validated with integrity monitoring tools, to ensure that only authorized changes to the images are possible.			

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

Profile Applicability:

- Level 2

Description:

Enable Secure Boot for Shielded GKE Nodes to verify the digital signature of node boot components.

Rationale:

An attacker may seek to alter boot components to persist malware or root kits during system initialisation. Secure Boot helps ensure that the system only runs authentic software by verifying the digital signature of all boot components, and halting the boot process if signature verification fails.

Impact:

Secure Boot will not permit the use of third-party unsigned kernel modules.

Audit:

This Audit process utilize 3 parameterized variables.

Please set your environment to include the following Parameters.

\$POOL_NAME

\$CLUSTER_NAME

\$COMPUTE_ZONE

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the name of the cluster under test.
3. Open the Details pane for each Node pool within the cluster, and ensure that 'Secure boot' is set to 'Enabled' under the Security heading.

Using Command Line

To check if Secure Boot is enabled for the Node pools in your cluster, run the following command for each Node pool:

```
gcloud beta container node-pools describe $POOL_NAME \
  --cluster $CLUSTER_NAME --zone $COMPUTE_ZONE \
  --format json | jq .config.shieldedInstanceConfig
```

This will return

```
{
  "enableSecureBoot": true
}
```

if Secure Boot is enabled.

Remediation:

Once a Node pool is provisioned, it cannot be updated to enable Secure Boot. You must create new Node pools within the cluster with Secure Boot enabled.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the cluster requiring the update and click ADD NODE POOL
3. Ensure that the 'Secure boot' checkbox is checked under the 'Shielded options' Heading
4. Click SAVE.

You will also need to migrate workloads from existing non-conforming Node pools to the newly created Node pool, then delete the non-conforming pools.

Using Command Line

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

```
gcloud beta container node-pools create [NODEPOOL_NAME] \
  --cluster [CLUSTER_NAME] --zone [COMPUTE_ZONE] \
  --shielded-secure-boot
```

You will also need to migrate workloads from existing non-conforming Node pools to the newly created Node pool, then delete the non-conforming pools.



Default Value:

By default, Secure Boot is disabled in GKE clusters. By default, Secure Boot is disabled when Shielded GKE Nodes is enabled.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/shielded-gke-nodes#secure_boot

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.3 Securely Store Master Images Store the master images and templates on securely configured servers, validated with integrity monitoring tools, to ensure that only authorized changes to the images are possible.			

5.6 Cluster Networking

This section contains recommendations relating to network security configurations in GKE.

5.6.1 Enable VPC Flow Logs and Intranode Visibility (Automated)

Profile Applicability:

- Level 2

Description:

Enable VPC Flow Logs and Intranode Visibility to see pod-level traffic, even for traffic within a worker node.

Rationale:

Enabling Intranode Visibility makes your intranode pod to pod traffic visible to the networking fabric. With this feature, you can use VPC Flow Logs or other VPC features for intranode traffic.

Impact:

This is a beta feature. Enabling it on existing cluster causes the cluster master and the cluster nodes to restart, which might cause disruption.

Audit:

The audit script for this recommendation utilizes 2 variables:

\$CLUSTER_NAME

\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the desired cluster, and under the 'Cluster' section, make sure that 'Intranode visibility' is set to 'Enabled'.

Using Command Line

Run this command:

```
gcloud beta container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE \
  --format json | jq '.networkConfig.enableIntraNodeVisibility'
```

The result should return `true` if Intranode Visibility is Enabled.

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>

2. Select Kubernetes clusters for which intranode visibility is disabled
3. Click on EDIT
4. Set 'Intranode visibility' to 'Enabled'
5. Click SAVE.

Using Command Line

To enable intranode visibility on an existing cluster, run the following command:

```
gcloud beta container clusters update [CLUSTER_NAME] \
  --enable-intra-node-visibility
```





Default Value:

By default, Intranode Visibility is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/intranode-visibility>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.5 Collect Detailed Audit Logs Configure detailed audit logging for enterprise assets containing sensitive data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.			
v7	6.3 Enable Detailed Logging Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.			

5.6.2 Ensure use of VPC-native clusters (Automated)

Profile Applicability:

- Level 1

Description:

Create Alias IPs for the node network CIDR range in order to subsequently configure IP-based policies and firewalling for pods. A cluster that uses Alias IPs is called a 'VPC-native' cluster.

Rationale:

Using Alias IPs has several benefits:

- Pod IPs are reserved within the network ahead of time, which prevents conflict with other compute resources.
- The networking layer can perform anti-spoofing checks to ensure that egress traffic is not sent with arbitrary source IPs.
- Firewall controls for Pods can be applied separately from their nodes.
- Alias IPs allow Pods to directly access hosted services without using a NAT gateway.

Impact:

You cannot currently migrate an existing cluster that uses routes for Pod routing to a cluster that uses Alias IPs.

Cluster IPs for internal services remain only available from within the cluster. If you want to access a Kubernetes Service from within the VPC, but from outside of the cluster, use an internal load balancer.

Audit:

The audit script for this recommendation utilizes 3 variables:

`$CLUSTER_NAME`

`$COMPUTE_ZONE`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the desired cluster to open the Details page. Under the 'Cluster' section, make sure 'VPC native (using alias IP)' is set to 'Enabled'.

Using Command Line:

To check Alias IP is enabled for an existing cluster, run the following command:

```
gcloud container clusters describe [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --format json | jq '.ipAllocationPolicy.useIpAliases'
```

The output of the above command should return `true`, if VPC-native (using alias IP) is enabled. If VPC-native (using alias IP) is disabled, the above command will return null (`{ }`).

Remediation:

Use of Alias IPs cannot be enabled on an existing cluster. To create a new cluster using Alias IPs, follow the instructions below.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER
3. Configure your cluster as desired. Then, click 'Availability, networking, security, and additional features'
4. In the 'VPC-native' section, leave 'Enable VPC-native (using alias IP)' selected
5. Click CREATE.

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
```



Default Value:

By default, VPC-native (using alias IP) is enabled when you create a new cluster in the Google Cloud Console, however this is disabled when creating a new cluster using the gcloud CLI, unless the `--enable-ip-alias` argument is specified.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/alias-ips>
2. <https://cloud.google.com/vpc/docs/alias-ip>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.4 <u>Perform Traffic Filtering Between Network Segments</u> Perform traffic filtering between network segments, where appropriate.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	<p>11 <u>Secure Configuration for Network Devices, such as Firewalls, Routers and Switches</u></p> <p>Secure Configuration for Network Devices, such as Firewalls, Routers and Switches</p>			
v7	<p>14.1 <u>Segment the Network Based on Sensitivity</u></p> <p>Segment the network based on the label or classification level of the information stored on the servers, locate all sensitive information on separated Virtual Local Area Networks (VLANs).</p>		●	●

5.6.3 *Ensure Master Authorized Networks is Enabled (Automated)*

Profile Applicability:

- Level 1

Description:

Enable Master Authorized Networks to restrict access to the cluster's control plane (master endpoint) to only an allowlist (whitelist) of authorized IPs.

Rationale:

Authorized networks are a way of specifying a restricted range of IP addresses that are permitted to access your cluster's control plane. Kubernetes Engine uses both Transport Layer Security (TLS) and authentication to provide secure access to your cluster's control plane from the public internet. This provides you the flexibility to administer your cluster from anywhere; however, you might want to further restrict access to a set of IP addresses that you control. You can set this restriction by specifying an authorized network.

Master Authorized Networks blocks untrusted IP addresses. Google Cloud Platform IPs (such as traffic from Compute Engine VMs) can reach your master through HTTPS provided that they have the necessary Kubernetes credentials.

Restricting access to an authorized network can provide additional security benefits for your container cluster, including:

- Better protection from outsider attacks: Authorized networks provide an additional layer of security by limiting external, non-GCP access to a specific set of addresses you designate, such as those that originate from your premises. This helps protect access to your cluster in the case of a vulnerability in the cluster's authentication or authorization mechanism.
- Better protection from insider attacks: Authorized networks help protect your cluster from accidental leaks of master certificates from your company's premises. Leaked certificates used from outside GCP and outside the authorized IP ranges (for example, from addresses outside your company) are still denied access.

Impact:

When implementing Master Authorized Networks, be careful to ensure all desired networks are on the allowlist (whitelist) to prevent inadvertently blocking external access to your cluster's control plane.

Audit:

The audit script for this recommendation utilizes 3 variables:

\$CLUSTER_NAME

\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting:
<https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the cluster to open the Details page and make sure 'Master authorized networks' is set to 'Enabled'.

Using Command Line

To check Master Authorized Networks status for an existing cluster, run the following command;

```
gcloud container clusters describe $CLUSTER_NAME
--zone $COMPUTE_ZONE
--format json | jq '.masterAuthorizedNetworksConfig'
```

The output should return

```
{
  "enabled": true
}
```

if Master Authorized Networks is enabled. If Master Authorized Networks is disabled, the above command will return null ({ }).

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting
<https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which Master Authorized Networks is disabled
3. Click on EDIT
4. Set 'Master authorized networks' to 'Enabled' and add authorize networks
5. Click SAVE.

Using Command Line

To enable Master 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`).







Default Value:

By default, Master Authorized Networks is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/authorized-networks>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 <u>Configure Data Access Control Lists</u> Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.			
v7	14.6 <u>Protect Information through Access Control Lists</u> Protect all information stored on systems with file system, network share, claims, application, or database specific access control lists. These controls will enforce the principle that only authorized individuals should have access to the information based on their need to access the information as a part of their responsibilities.			

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

Profile Applicability:

- Level 2

Description:

Disable access to the Kubernetes API from outside the node network if it is not required.

Rationale:

In a private cluster, the master node has two endpoints, a private and public endpoint. The private endpoint is the internal IP address of the master, behind an internal load balancer in the master's VPC network. Nodes communicate with the master using the private endpoint. The public endpoint enables the Kubernetes API to be accessed from outside the master's VPC network.

Although Kubernetes API requires an authorized token to perform sensitive actions, a vulnerability could potentially expose the Kubernetes publically with unrestricted access. Additionally, an attacker may be able to identify the current cluster and Kubernetes API version and determine whether it is vulnerable to an attack. Unless required, disabling public endpoint will help prevent such threats, and require the attacker to be on the master's VPC network to perform any attack on the Kubernetes API.

Impact:

To enable a Private Endpoint, the cluster has to also be configured with private nodes, a private master IP range and IP aliasing enabled.

If the Private Endpoint flag `--enable-private-endpoint` is passed to the gcloud CLI, or the external IP address undefined in the Google Cloud Console during cluster creation, then all access from a public IP address is prohibited.

Audit:

The audit script for this recommendation utilizes 3 variables:

`$CLUSTER_NAME`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the desired cluster, and within the Details pane, make sure the 'Endpoint' does not have a public IP address.

Using Command Line

Run this command:

```
gcloud container clusters describe $CLUSTER_NAME \
--format json | jq '.privateClusterConfig.enablePrivateEndpoint'
```

The output of the above command returns `true` if a Private Endpoint is enabled with Public Access disabled.

For an additional check, the endpoint parameter can be queried with the following command:

```
gcloud container clusters describe $CLUSTER_NAME \
--format json | jq '.endpoint'
```

The output of the above command returns a private IP address if Private Endpoint is enabled with Public Access disabled.

Remediation:

Once a cluster is created without enabling Private Endpoint only, it cannot be remediated. Rather, the cluster must be recreated.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER
3. Configure the cluster as desired then click 'Availability, networking, security, and additional features'
4. Under 'Network Security' ensure the 'Private cluster' checkbox is checked
5. Clear the 'Access master using its external IP address' checkbox.
6. Configure other settings as required
7. Click CREATE.

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]`.




Default Value:

By default, the Private Endpoint is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/private-clusters>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.			
v7	12 <u>Boundary Defense</u> Boundary Defense			

5.6.5 Ensure clusters are created with Private Nodes (Automated)

Profile Applicability:

- Level 1

Description:

Disable public IP addresses for cluster nodes, so that they only have private IP addresses. Private Nodes are nodes with no public IP addresses.

Rationale:

Disabling public IP addresses on cluster nodes restricts access to only internal networks, forcing attackers to obtain local network access before attempting to compromise the underlying Kubernetes hosts.

Impact:

To enable Private Nodes, the cluster has to also be configured with a private master IP range and IP Aliasing enabled.

Private Nodes do not have outbound access to the public internet. If you want to provide outbound Internet access for your private nodes, you can use Cloud NAT or you can manage your own NAT gateway.

To access Google Cloud APIs and services from private nodes, Private Google Access needs to be set on Kubernetes Engine Cluster Subnets.

Audit:

The audit script for this recommendation utilizes 1 variable:

`$CLUSTER_NAME`

Please set this parameter on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the desired cluster, and within the Details pane, make sure 'Private Clusters' is set to 'Enabled'.

Using Command Line

Run this command:

```
gcloud container clusters describe $CLUSTER_NAME \
  --format json | jq '.privateClusterConfig.enablePrivateNodes'
```

The output of the above command returns `true` if Private Nodes is enabled.

Remediation:

Once a cluster is created without enabling Private Nodes, it cannot be remediated. Rather the cluster must be recreated.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER
3. Configure the cluster as desired then click 'Availability, networking, security, and additional features'
4. Under 'Network Security' ensure the 'Private cluster' checkbox is checked
5. Configure other settings as required
6. Click CREATE.

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]`.




Default Value:

By default, Private Nodes are disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/private-clusters>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 Implement and Manage a Firewall on Servers Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.			
v7	12 Boundary Defense Boundary Defense			

5.6.6 Consider firewalling GKE worker nodes (Manual)

Profile Applicability:

- Level 2

Description:

Reduce the network attack surface of GKE nodes by using Firewalls to restrict ingress and egress traffic.

Rationale:

Utilizing stringent ingress and egress firewall rules minimizes the ports and services exposed to an network-based attacker, whilst also restricting egress routes within or out of the cluster in the event that a compromised component attempts to form an outbound connection.

Impact:

All instances targeted by a firewall rule, either using a tag or a service account will be affected. Ensure there are no adverse effects on other instances using the target tag or service account before implementing the firewall rule.

Audit:

Using Google Cloud Console

1. Go to Compute Engine by visiting <https://console.cloud.google.com/compute/instances>
2. For each instance within your cluster, use the 'more actions' menu (3 vertical dots) and select to 'View network details'
3. If there are multiple network interfaces attached to the instance, select the network interface to view in the 'Network interface' details section and see all the rules that apply to the network interface, within the 'Firewall rules' tab. Make sure the firewall rules are appropriate for your environment.

Using Command Line

For the instance being evaluated, obtain its Service account and tags:

```
gcloud compute instances describe [INSTANCE_NAME] \
  --zone [COMPUTE_ZONE] \
  --format json | jq '{tags: .tags.items[],
serviceaccount:.serviceAccounts[].email, network:
.networkInterfaces[].network}'
```

this will return:

```
{
  "tags": "[TAG]",
  "serviceaccount": "[SERVICE_ACCOUNT]"
  "network":
    "https://www.googleapis.com/compute/v1/projects/[PROJECT_ID]/global/networks/[NETWORK]"
}
```

Then, observe the firewall rules applied to the instance by using the following command, replacing [TAG] and [SERVICE_ACCOUNT] as appropriate:

```
gcloud compute firewall-rules list \
  --format="table(
    name,
    network,
    direction,
    priority,
    sourceRanges.list():label=SRC_RANGES,
    destinationRanges.list():label=DEST_RANGES,
    allowed[].map().firewall_rule().list():label=ALLOW,
    denied[].map().firewall_rule().list():label=DENY,
    sourceTags.list():label=SRC_TAGS,
    sourceServiceAccounts.list():label=SRC_SVC_ACCT,
    targetTags.list():label=TARGET_TAGS,
    targetServiceAccounts.list():label=TARGET_SVC_ACCT,
    disabled
  )" \
  --filter="targetTags.list():[TAG] OR
targetServiceAccounts.list():[SERVICE_ACCOUNT]"
```

Firewall rules may also be applied to a network without specifically targeting Tags or Service Accounts. These can be observed using the following, replacing [NETWORK] as appropriate:

```
gcloud compute firewall-rules list \
  --format="table(
    name,
    network,
    direction,
    priority,
    sourceRanges.list():label=SRC_RANGES,
    destinationRanges.list():label=DEST_RANGES,
    allowed[].map().firewall_rule().list():label=ALLOW,
    denied[].map().firewall_rule().list():label=DENY,
    sourceTags.list():label=SRC_TAGS,
    sourceServiceAccounts.list():label=SRC_SVC_ACCT,
    targetTags.list():label=TARGET_TAGS,
    targetServiceAccounts.list():label=TARGET_SVC_ACCT,
    disabled
  )" \
  --filter="network.list():[NETWORK] AND -targetTags.list():* AND -
targetServiceAccounts.list():*"
```

Remediation:

Using Google Cloud Console

1. Go to Firewall Rules by visiting <https://console.cloud.google.com/networking/firewalls/list>
2. Click CREATE FIREWALL RULE
3. Configure the firewall rule as required. Ensure the firewall targets your nodes correctly, either selecting the nodes using tags (under 'Targets', select 'Specified target tags', and set 'Target tags' to [TAG]), or using the Service account associated with node (under 'Targets', select 'Specified service account', set 'Service account scope' as appropriate, and 'Target service account' to [SERVICE_ACCOUNT])
4. Click CREATE.

Using Command Line

Use the following command to generate firewall rules, setting the variables as appropriate. You may want to use the target [TAG] and [SERVICE_ACCOUNT] previously identified.

```
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]
```

Default Value:

Every VPC network has two implied firewall rules. These rules exist, but are not shown in the Cloud Console:





- The implied allow egress rule: An egress rule whose action is `allow`, destination is `0.0.0.0/0`, and priority is the lowest possible (65535) lets any instance send traffic to any destination, except for traffic blocked by GCP. Outbound access may be restricted by a higher priority firewall rule. Internet access is allowed if no other firewall rules deny outbound traffic and if the instance has an external IP address or uses a NAT instance.
- The implied deny ingress rule: An ingress rule whose action is `deny`, source is `0.0.0.0/0`, and priority is the lowest possible (65535) protects all instances by blocking incoming traffic to them. Incoming access may be allowed by a higher priority rule. Note that the default network includes some additional rules that override this one, allowing certain types of incoming traffic.

The implied rules cannot be removed, but they have the lowest possible priorities.

References:

1. <https://cloud.google.com/vpc/docs/firewalls>
2. <https://cloud.google.com/vpc/docs/using-firewalls>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.4 <u>Implement and Manage a Firewall on Servers</u> Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.			
v7	9.5 <u>Implement Application Firewalls</u> Place application firewalls in front of any critical servers to verify and validate the traffic going to the server. Any unauthorized traffic should be blocked and logged.			

5.6.7 Ensure Network Policy is Enabled and set as appropriate (Manual)

Profile Applicability:

- Level 1

Description:

Use Network Policy to restrict pod to pod traffic within a cluster and segregate workloads.

Rationale:

By default, all pod to pod traffic within a cluster is allowed. Network Policy creates a pod-level firewall that can be used to restrict traffic between sources. Pod traffic is restricted by having a Network Policy that selects it (through the use of labels). Once there is any Network Policy in a namespace selecting a particular pod, that pod will reject any connections that are not allowed by any Network Policy. Other pods in the namespace that are not selected by any Network Policy will continue to accept all traffic.

Network Policies are managed via the Kubernetes Network Policy API and enforced by a network plugin, simply creating the resource without a compatible network plugin to implement it will have no effect. GKE supports Network Policy enforcement through the use of Calico.

Impact:

Network Policy requires the Network Policy add-on. This add-on is included automatically when a cluster with Network Policy is created, but for an existing cluster, needs to be added prior to enabling Network Policy.

Enabling/Disabling Network Policy causes a rolling update of all cluster nodes, similar to performing a cluster upgrade. This operation is long-running and will block other operations on the cluster (including delete) until it has run to completion.

If Network Policy is used, a cluster must have at least 2 nodes of type `n1-standard-1` or higher. The recommended minimum size cluster to run Network Policy enforcement is 3 `n1-standard-1` instances.

Enabling Network Policy enforcement consumes additional resources in nodes. Specifically, it increases the memory footprint of the `kube-system` process by approximately 128MB, and requires approximately 300 millicores of CPU.

Audit:

The audit script for this recommendation utilizes 3 variables:

\$CLUSTER_NAME

\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the desired cluster, and ensure that 'Network policy' is set to 'Enabled'.

Using Command Line

To check Network Policy is enabled for an existing cluster, run the following command,

```
gcloud container clusters describe $CLUSTER_NAME \
--zone $COMPUTE_ZONE \
--format json | jq '.networkPolicy'
```

The output of the above command should be:

```
{
  "enabled": true
}
```

if Network Policy is enabled. If Network policy is disabled, the above command output will return null ({ }).

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the cluster for which Network policy is disabled
3. Click EDIT
4. Set 'Network policy for master' to 'Enabled'
5. Click SAVE
6. Once the cluster has updated, repeat steps 1-3
7. Set 'Network Policy for nodes' to 'Enabled'
8. Click SAVE.

Using Command Line

To enable Network Policy for an existing cluster, firstly enable the Network Policy add-on:

```
gcloud container clusters update [CLUSTER_NAME] \
--zone [COMPUTE_ZONE] \
--update-addons NetworkPolicy=ENABLED
```

Then, enable Network Policy:

```
gcloud container clusters update [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --enable-network-policy
```








Default Value:

By default, Network Policy is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/network-policy>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	12.6 <u>Use of Secure Network Management and Communication Protocols</u> Use secure network management and communication protocols (e.g., 802.1X, Wi-Fi Protected Access 2 (WPA2) Enterprise or greater).			
v7	9.2 <u>Ensure Only Approved Ports, Protocols and Services Are Running</u> Ensure that only network ports, protocols, and services listening on a system with validated business needs, are running on each system.			
v7	9.4 <u>Apply Host-based Firewalls or Port Filtering</u> Apply host-based firewalls or port filtering tools on end systems, with a default-deny rule that drops all traffic except those services and ports that are explicitly allowed.			

5.6.8 Ensure use of Google-managed SSL Certificates (Manual)

Profile Applicability:

- Level 2

Description:

Encrypt traffic to HTTPS load balancers using Google-managed SSL certificates.

Rationale:

Encrypting traffic between users and your Kubernetes workload is fundamental to protecting data sent over the web.

Google-managed SSL Certificates are provisioned, renewed, and managed for your domain names. This is only available for HTTPS load balancers created using Ingress Resources, and not TCP/UDP load balancers created using Service of `type:LoadBalancer`.

Impact:

Google-managed SSL Certificates are less flexible than certificates you obtain and manage yourself. Managed certificates support a single, non-wildcard domain. Self-managed certificates can support wildcards and multiple subject alternative names (SANs).

Audit:

Using Command Line

Identify if there are any workloads exposed publicly using Services of

`type:LoadBalancer`:

```
kubectl get svc -A -o json | jq '.items[] | select(.spec.type=="LoadBalancer")'
```

You will need to consider using ingresses instead of these services in order to use Google managed SSL certificates.

For the ingresses within your cluster, run the following command:

```
kubectl get ingress -A -o json | jq .items[] | jq \
  '{name: .metadata.name, annotations: .metadata.annotations, namespace:
  .metadata.namespace, status: .status}'
```

The above command should return the name of the ingress, namespace, annotations and status. Check that the following annotation is present to ensure managed certificates are referenced.

```
"annotations": {  
  ...  
  "networking.gke.io/managed-certificates": "[EXAMPLE_CERTIFICATE]"  
},
```

For completeness, run the following command to ensure that the managed certificate resource exists:

```
kubectl get managedcertificates -A
```

The above command returns a list of managed certificates for which [EXAMPLE_CERTIFICATE] should exist within the same namespace as the ingress.

Remediation:

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>.





Default Value:

By default, Google-managed SSL Certificates are not created when an Ingress resource is defined.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/managed-certs>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.10 <u>Encrypt Sensitive Data in Transit</u> Encrypt sensitive data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).			
v7	14.4 <u>Encrypt All Sensitive Information in Transit</u> Encrypt all sensitive information in transit.			

5.7 Logging

This section contains recommendations relating to security-related logging in GKE.

5.7.1 Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled (Automated)

Profile Applicability:

- Level 1

Description:

Send logs and metrics to a remote aggregator to mitigate the risk of local tampering in the event of a breach.

Rationale:

Exporting logs and metrics to a dedicated, persistent datastore such as Stackdriver ensures availability of audit data following a cluster security event, and provides a central location for analysis of log and metric data collated from multiple sources.

Currently, there are two mutually exclusive variants of Stackdriver available for use with GKE clusters: Legacy Stackdriver Support and Stackdriver Kubernetes Engine Monitoring Support.

Although Stackdriver Kubernetes Engine Monitoring is the preferred option, starting with GKE versions 1.12.7 and 1.13, Legacy Stackdriver is the default option up through GKE version 1.13. The use of either of these services is sufficient to pass the benchmark recommendation.

However, note that as Legacy Stackdriver Support is not getting any improvements and lacks features present in Stackdriver Kubernetes Engine Monitoring, Legacy Stackdriver Support may be deprecated in favour of Stackdriver Kubernetes Engine Monitoring Support in future versions of this benchmark.

Impact:

Stackdriver Kubernetes Engine Monitoring and Legacy Stackdriver are incompatible because they have different data models. To move from Legacy Stackdriver to Stackdriver Kubernetes Engine Monitoring, you must manually change a number of your Stackdriver artifacts, including alerting policies, group filters, and log queries. See <https://cloud.google.com/monitoring/kubernetes-engine/migration>.

Audit:

The audit script for this recommendation utilizes 2 variables:

\$CLUSTER_NAME
\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console:

STACKDRIVER KUBERNETES ENGINE MONITORING SUPPORT (PREFERRED):

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the cluster, and make sure 'Stackdriver Kubernetes Engine Monitoring' is set to 'Enabled'.

LEGACY STACKDRIVER SUPPORT:

Both Logging and Monitoring support must be enabled.

For Logging:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the cluster, and make sure 'Legacy Stackdriver Logging' is set to 'Enabled'.

For Monitoring:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the cluster, and make sure 'Legacy Stackdriver Monitoring' is set to 'Enabled'.

Using Command Line

STACKDRIVER KUBERNETES ENGINE MONITORING SUPPORT (PREFERRED):

Run the following commands:

```
gcloud container clusters describe [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --format json | jq '.loggingService'
gcloud container clusters describe [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --format json | jq '.monitoringService'
```

The output of the above commands should return `logging.googleapis.com/kubernetes` and `monitoring.googleapis.com/kubernetes` respectively if Stackdriver Kubernetes Engine Monitoring is Enabled.

LEGACY STACKDRIVER SUPPORT:

Both Logging and Monitoring support must be enabled.

For Logging, run the following command:

```
gcloud container clusters describe [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --format json | jq '.loggingService'
```

The output should return `logging.googleapis.com` if Legacy Stackdriver Logging is Enabled.

For Monitoring, run the following command:

```
gcloud container clusters describe [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --format json | jq '.monitoringService'
```

The output should return `monitoring.googleapis.com` if Legacy Stackdriver Monitoring is Enabled.

Remediation:

Using Google Cloud Console

STACKDRIVER KUBERNETES ENGINE MONITORING SUPPORT (PREFERRED):

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which Stackdriver Kubernetes Engine Monitoring is disabled
3. Click on EDIT
4. Set 'Stackdriver Kubernetes Engine Monitoring' to 'Enabled'
5. Click SAVE.

LEGACY STACKDRIVER SUPPORT:

Both Logging and Monitoring support must be enabled.

For Logging:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which logging is disabled
3. Click on EDIT
4. Set 'Legacy Stackdriver Logging' to 'Enabled'
5. Click SAVE.

For Monitoring:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which monitoring is disabled
3. Click on EDIT
4. Set 'Legacy Stackdriver Monitoring' to 'Enabled'
5. Click SAVE.

Using Command Line

STACKDRIVER KUBERNETES ENGINE MONITORING SUPPORT (PREFERRED):

To enable Stackdriver Kubernetes Engine Monitoring for an existing cluster, run the following command:

```
gcloud container clusters update [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --enable-stackdriver-kubernetes
```

LEGACY STACKDRIVER SUPPORT

Both Logging and Monitoring support must be enabled.

To enable Legacy Stackdriver Logging for an existing cluster, run the following command:

```
gcloud container clusters update [CLUSTER_NAME] --zone [COMPUTE_ZONE] --
logging-service logging.googleapis.com
```

To enable Legacy Stackdriver Monitoring for an existing cluster, run the following command:

```
gcloud container clusters update [CLUSTER_NAME] --zone [COMPUTE_ZONE] --
monitoring-service monitoring.googleapis.com
```







Default Value:

Stackdriver Kubernetes Engine monitoring is enabled by default starting in GKE version 1.14; Legacy Stackdriver Logging and Monitoring support is enabled by default for earlier versions.

References:

1. <https://cloud.google.com/monitoring/kubernetes-engine/>
2. <https://cloud.google.com/monitoring/kubernetes-engine/legacy-stackdriver/monitoring>
3. <https://cloud.google.com/monitoring/kubernetes-engine/legacy-stackdriver/logging>
4. <https://cloud.google.com/monitoring/kubernetes-engine/migration>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 <u>Collect Audit Logs</u> Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.			
v7	6.2 <u>Activate audit logging</u> Ensure that local logging has been enabled on all systems and networking devices.			

5.7.2 Enable Linux auditd logging (Manual)

Profile Applicability:

- Level 2

Description:

Run the auditd logging daemon to obtain verbose operating system logs from GKE nodes running Container-Optimized OS (COS).

Rationale:

Auditd logs provide valuable information about the state of the cluster and workloads, such as error messages, login attempts, and binary executions. This information can be used to debug issues or to investigate security incidents.

Impact:

Increased logging activity on a node increases resource usage on that node, which may affect the performance of your workload and may incur additional resource costs. Audit logs sent to Stackdriver consume log quota from the project. You may need to increase your log quota and storage to accommodate the additional logs.

Note that the provided logging daemonset only works on nodes running Container-Optimized OS (COS).

Audit:

Using Google Cloud Console

1. Navigate to the Kubernetes Engine workloads by visiting <https://console.cloud.google.com/kubernetes/workload>
2. Observe the workloads and ensure that all filters are removed
3. If the unmodified example auditd logging daemonset <https://raw.githubusercontent.com/GoogleCloudPlatform/k8s-node-tools/master/os-audit/cos-auditd-logging.yaml> is being used, ensure that the `cos-auditd-logging` daemonset is being run in the `cos-auditd` namespace with the number of running pods reporting as expected.

Using Command Line

If using the unmodified example auditd logging daemonset, run

```
kubectl get daemonsets -n cos-audit
```

and observe that the `cos-auditd-logging` daemonset is running as expected.

If the name or namespace of the daemonset has been modified and is unknown, you can search for the container being used by the daemonset:


```
kubectl get daemonsets -A -o json | jq '.items[] | select
(.spec.template.spec.containers[].image | contains ("gcr.io/stackdriver-
agents/stackdriver-logging-agent"))' | jq '{name: .metadata.name, annotations:
.metadata.annotations."kubernetes.io/description", namespace:
.metadata.namespace, status: .status}'
```

The above command returns the name, namespace and status of the daemonsets that use the Stackdriver logging agent. The example auditd logging daemonset has a description within the annotation as output by the command above:

```
{
  "name": "cos-auditd-logging",
  "annotations": "DaemonSet that enables Linux auditd logging on COS nodes.",
  "namespace": "cos-auditd",
  "status": {...
}
```

Ensure that the status fields return that the daemonset is running as expected.

Remediation:

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 you defined a different Namespace in your manifests, replace `cos-auditd` with the name of the namespace you're using:

```
kubectl get pods --namespace=cos-auditd
```




Default Value:

By default, the auditd logging daemonset is not launched when a GKE cluster is created.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/linux-auditd-logging>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	8.2 Collect Audit Logs Collect audit logs. Ensure that logging, per the enterprise's audit log management process, has been enabled across enterprise assets.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	6.3 Enable Detailed Logging Enable system logging to include detailed information such as an event source, date, user, timestamp, source addresses, destination addresses, and other useful elements.		●	●

5.8 Authentication and Authorization

This section contains recommendations relating to authentication and authorization in GKE.

5.8.1 Ensure Basic Authentication using static passwords is Disabled (Automated)

Profile Applicability:

- Level 1

Description:

Disable Basic Authentication (basic auth) for API server authentication as it uses static passwords which need to be rotated.

Rationale:

Basic Authentication allows a user to authenticate to a Kubernetes cluster with a username and static password which is stored in plaintext (without any encryption). Disabling Basic Authentication will prevent attacks like brute force and credential stuffing. It is recommended to disable Basic Authentication and instead use another authentication method such as OpenID Connect.

GKE manages authentication via gcloud using the OpenID Connect token method, setting up the Kubernetes configuration, getting an access token, and keeping it up to date. This means Basic Authentication using static passwords and Client Certificate authentication, which both require additional management overhead of key management and rotation, are not necessary and should be disabled.

When Basic Authentication is disabled, you will still be able to authenticate to the cluster with other authentication methods, such as OpenID Connect tokens. See also Recommendation 6.8.2 to disable authentication using Client Certificates.

Impact:

Users will no longer be able to authenticate with a static password. You will have to configure and use alternate authentication mechanisms, such as OpenID Connect tokens.

Audit:

The audit script for this recommendation utilizes 3 variables:

`$NODE_POOL`

`$CLUSTER_NAME`

`$COMPUTE_ZONE`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, select the desired cluster

3. Within the Details pane, click 'Show cluster certificate' next to 'Endpoint' heading
4. Within the 'Cluster Credentials' pop-up the message, "Basic authentication for this cluster is disabled " should be displayed, and the field 'Basic Auth credentials (Username and Password)' missing.

Using Command Line

To check Basic Authentication status for an existing cluster's nodes, run the following command:

```
gcloud container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE \
  --format json | jq '.masterAuth.password and .masterAuth.username'
```

The output of the above command should return `false`, if Basic Authentication is disabled. If Basic Authentication is enabled, the above command will return `true`.

Remediation:

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the Kubernetes cluster for which Basic Authentication is currently enabled
3. Click on EDIT
4. Set 'Basic authentication' to 'Disabled'
5. Click SAVE.

Using Command Line:

To update an existing cluster and disable Basic Authentication by removing the static password:

```
gcloud container clusters update [CLUSTER_NAME] \
  --no-enable-basic-auth
```





Default Value:

Clusters created from GKE version 1.12 have Basic Authentication and Client Certificate issuance disabled by default.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#restrict_authn_methods

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.1 <u>Establish and Maintain an Inventory of Accounts</u> Establish and maintain an inventory of all accounts managed in the enterprise. The inventory must include both user and administrator accounts. The inventory, at a minimum, should contain the person's name, username, start/stop dates, and department. Validate that all active accounts are authorized, on a recurring schedule at a minimum quarterly, or more frequently.			
v8	6.8 <u>Define and Maintain Role-Based Access Control</u> Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			
v7	16 <u>Account Monitoring and Control</u> Account Monitoring and Control			

5.8.2 Ensure authentication using Client Certificates is Disabled (Automated)

Profile Applicability:

- Level 1

Description:

Disable Client Certificates, which require certificate rotation, for authentication. Instead, use another authentication method like OpenID Connect.

Rationale:

With Client Certificate authentication, a client presents a certificate that the API server verifies with the specified Certificate Authority. In GKE, Client Certificates are signed by the cluster root Certificate Authority. When retrieved, the Client Certificate is only base64 encoded and not encrypted.

GKE manages authentication via gcloud for you using the OpenID Connect token method, setting up the Kubernetes configuration, getting an access token, and keeping it up to date. This means Basic Authentication using static passwords and Client Certificate authentication, which both require additional management overhead of key management and rotation, are not necessary and should be disabled.

When Client Certificate authentication is disabled, you will still be able to authenticate to the cluster with other authentication methods, such as OpenID Connect tokens. See also Recommendation 6.8.1 to disable authentication using static passwords, known as Basic Authentication.

Impact:

Users will no longer be able to authenticate with the pre-provisioned x509 certificate. You will have to configure and use alternate authentication mechanisms, such as OpenID Connect tokens.

Audit:

The audit script for this recommendation utilizes 3 variables:

`$CLUSTER_NAME`

`$COMPUTE_ZONE`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on the desired cluster. On the Details pane, make sure 'Client certificate' is set to 'Disabled'.

Using Command line

To check that the client certificate has not been issued, run the following command:

```
gcloud container clusters describe $CLUSTER_NAME \
--zone $COMPUTE_ZONE \
--format json | jq '.masterAuth.clientKey'
```

The output of the above command returns null ({ }) if the client certificate has not been issued for the cluster (Client Certificate authentication is disabled).

Remediation:

Currently, there is no way to remove a client certificate from an existing cluster. Thus a new cluster must be created.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER
3. Configure as required and the click on 'Availability, networking, security, and additional features' section
4. Ensure that the 'Issue a client certificate' checkbox is not ticked
5. Click CREATE.

Using Command Line

Create a new cluster without a Client Certificate:

```
gcloud container clusters create [CLUSTER_NAME] \
--no-issue-client-certificate
```


Default Value:

Clusters created from GKE version 1.12 have Basic Authentication and Client Certificate issuance disabled by default.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#restrict_authn_methods

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 <u>Define and Maintain Role-Based Access Control</u> Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	16 <u>Account Monitoring and Control</u> Account Monitoring and Control			

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

Profile Applicability:

- Level 2

Description:

Cluster Administrators should leverage G Suite Groups and Cloud IAM to assign Kubernetes user roles to a collection of users, instead of to individual emails using only Cloud IAM.

Rationale:

On- and off-boarding users is often difficult to automate and prone to error. Using a single source of truth for user permissions via G Suite Groups reduces the number of locations that an individual must be off-boarded from, and prevents users gaining unique permissions sets that increase the cost of audit.

Impact:

When migrating to using security groups, an audit of `RoleBindings` and `ClusterRoleBindings` is required to ensure all users of the cluster are managed using the new groups and not individually.

When managing `RoleBindings` and `ClusterRoleBindings`, be wary of inadvertently removing bindings required by service accounts.

Audit:

Using G Suite Admin Console and Google Cloud Console

1. Navigate to manage G Suite Groups in the Google Admin console at <https://admin.google.com/dashboard>
2. Ensure there is a group named `gke-security-groups@[yourdomain.com]`. The group must be named exactly `gke-security-groups`
3. Ensure only further groups (not individual users) are included in the `gke-security-groups` group as members
4. Go to the Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
5. From the list of clusters, click on the desired cluster. In the Details pane, make sure 'Google Groups for RBAC' is set to 'Enabled'.

Remediation:

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 beta container clusters create my-cluster \
  --security-group="gke-security-groups@[yourdomain.com]"
```

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

Default Value:

Google Groups for GKE is disabled by default.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control#google-groups-for-gke>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 <u>Define and Maintain Role-Based Access Control</u> Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	16.2 <u>Configure Centralized Point of Authentication</u> Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.		●	●

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

Profile Applicability:

- Level 1

Description:

Legacy Authorization, also known as Attribute-Based Access Control (ABAC) has been superseded by Role-Based Access Control (RBAC) and is not under active development. RBAC is the recommended way to manage permissions in Kubernetes.

Rationale:

In Kubernetes, RBAC is used to grant permissions to resources at the cluster and namespace level. RBAC allows you to define roles with rules containing a set of permissions, whilst the legacy authorizer (ABAC) in Kubernetes Engine grants broad, statically defined permissions. As RBAC provides significant security advantages over ABAC, it is recommended option for access control. Where possible, legacy authorization must be disabled for GKE clusters.

Impact:

Once the cluster has the legacy authorizer disabled, you must grant your user the ability to create authorization roles using RBAC to ensure that your role-based access control permissions take effect.

Audit:

The audit script for this recommendation utilizes 3 variables:

\$CLUSTER_NAME

\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on each cluster to open the Details pane, and make sure 'Legacy Authorization' is set to 'Disabled'.

Using Command Line

To check Legacy Authorization status for an existing cluster, run the following command:

```
gcloud container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE] \
  --format json | jq '.legacyAbac'
```

The output should return null ({}) if Legacy Authorization is Disabled. If Legacy Authorization is Enabled, the above command will return `true` value.

Remediation:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which Legacy Authorization is enabled
3. Click EDIT
4. Set 'Legacy Authorization' to 'Disabled'
5. Click SAVE.

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
```

Default Value:

Kubernetes Engine clusters running GKE version 1.8 and later disable the legacy authorization system by default, and thus role-based access control permissions take effect with no special action required.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control>
2. [https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#leave abac disabled default for 110](https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#leave_abac_disabled_default_for_110)

Additional Information:

On clusters running GKE 1.6 or 1.7, Kubernetes Service accounts have full permissions on the Kubernetes API by default. To ensure that your role-based access control permissions take effect for a Kubernetes service account, you must create or update your cluster with the option `--no-enable-legacy-authorization`. This requirement is removed for clusters running GKE version 1.8 or higher.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 <u>Define and Maintain Role-Based Access Control</u> Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.			●
v7	4 <u>Controlled Use of Administrative Privileges</u> Controlled Use of Administrative Privileges			
v7	16 <u>Account Monitoring and Control</u> Account Monitoring and Control			

5.9 Storage

This section contains recommendations relating to security-related configurations for storage in GKE.

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

Profile Applicability:

- Level 1

Description:

Use Customer-Managed Encryption Keys (CMEK) to encrypt node boot and dynamically-provisioned attached Google Compute Engine Persistent Disks (PDs) using keys managed within Cloud Key Management Service (Cloud KMS).

Rationale:

GCE persistent disks are encrypted at rest by default using envelope encryption with keys managed by Google. For additional protection, users can manage the Key Encryption Keys using Cloud KMS.

Impact:

While GKE CMEK is in beta, encryption of dynamically-provisioned attached disks requires the use of the self-provisioned Compute Engine Persistent Disk CSI Driver v0.5.1 or higher.

If you are configuring CMEK with a regional cluster, the cluster must run GKE 1.14 or higher.

Audit:

Using Google Cloud Console:

FOR NODE BOOT DISKS:

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on each cluster, and click on any Node pools
3. On the Node pool Details page, under the 'Security' heading, check that 'Boot disk encryption type' is set to 'Customer managed' with the desired key.

FOR ATTACHED DISKS:

1. Go to Compute Engine Disks by visiting <https://console.cloud.google.com/compute/disks>
2. Select each disk used by the cluster, and ensure the 'Encryption Type' is listed as 'Customer Managed'.

Using Command Line:

FOR NODE BOOT DISKS:

Run this command:

```
gcloud beta container node-pools describe [NODE_POOL_NAME] \
--cluster [CLUSTER_NAME]
```

Verify that the output of the above command includes a `diskType` of either `pd-standard` or `pd-ssd`, and the `bootDiskKmsKey` is specified as the desired key.

FOR ATTACHED DISKS:

Identify the Persistent Volumes Used by the cluster:

```
kubectl get pv -o json | jq '.items[].metadata.name'
```

For each volume used, check that it is encrypted using a customer managed key by running the following command:

```
gcloud compute disks describe [PV_NAME] \
--zone [COMPUTE-ZONE] \
--format json | jq '.diskEncryptionKey.kmsKeyName'
```

This returns null (`{ }`) if a customer-managed encryption key is not used to encrypt the disk.

Remediation:

This cannot be remediated by updating an existing cluster. You must either recreate the desired node pool or create a new cluster.

Using Google Cloud Console:

FOR NODE BOOT DISKS:

To create a new node pool,

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select Kubernetes clusters for which node boot disk CMEK is disabled
3. Click ADD NODE POOL
4. Ensure Boot disk type is 'Standard persistent disk' or 'SSD persistent disk'
5. Select 'Enable customer-managed encryption for Boot Disk' and select the Cloud KMS encryption key you desire
6. Click SAVE.

To create a new cluster,

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click CREATE CLUSTER

3. Under the 'default-pool' heading, click 'More options'
4. In the Node pool edit window, select 'Standard persistent disk' or 'SSD Persistent Disk' as the Boot disk type
5. Select 'Enable customer-managed encryption for Boot Disk' check box and choose the Cloud KMS encryption key you desire
6. Configure the rest of the cluster settings as desired
7. Click CREATE.

Click Save.

FOR ATTACHED DISKS:

This is not possible using Google Cloud Console.

Using Command Line:

FOR NODE BOOT DISKS:

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 beta 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 beta 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]
```

FOR ATTACHED DISKS:

Follow the instructions detailed at <https://cloud.google.com/kubernetes-engine/docs/how-to/using-cmek>.




Default Value:

Persistent disks are encrypted at rest by default, but are not encrypted using Customer-Managed Encryption Keys by default. By default, the Compute Engine Persistent Disk CSI Driver is not provisioned within the cluster.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/using-cmek>
2. <https://cloud.google.com/compute/docs/disks/customer-managed-encryption>
3. <https://cloud.google.com/security/encryption-at-rest/default-encryption/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 <u>Encrypt Sensitive Data at Rest</u> Encrypt sensitive data at rest on servers, applications, and databases containing sensitive data. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.			
v7	14.8 <u>Encrypt Sensitive Information at Rest</u> Encrypt all sensitive information at rest using a tool that requires a secondary authentication mechanism not integrated into the operating system, in order to access the information.			

5.10 Other Cluster Configurations

This section contains recommendations relating to any remaining security-related cluster configurations in GKE.

5.10.1 Ensure Kubernetes Web UI is Disabled (Automated)

Profile Applicability:

- Level 1

Description:

The Kubernetes Web UI (Dashboard) has been a historical source of vulnerability and should only be deployed when necessary.

Rationale:

You should disable the Kubernetes Web UI (Dashboard) when running on Kubernetes Engine. The Kubernetes Web UI is backed by a highly privileged Kubernetes Service Account.

The Google Cloud Console provides all the required functionality of the Kubernetes Web UI and leverages Cloud IAM to restrict user access to sensitive cluster controls and settings.

Impact:

Users will be required to manage cluster resources using the Google Cloud Console or the command line. These require appropriate permissions. To use the command line, this requires the installation of the command line client, `kubectl`, on the user's device (this is already included in Cloud Shell) and knowledge of command line operations.

Audit:

The audit script for this recommendation utilizes 3 variables:

`$CLUSTER_NAME`

`$COMPUTE_ZONE`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. From the list of clusters, click on each cluster, click on 'Add-ons' to expand, and make sure 'Kubernetes dashboard' is set to 'Disabled'.

Using Command Line

Run the following command:

```
gcloud container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE \
  --format json | jq '.addonsConfig.kubernetesDashboard'
```

Ensure the output of the above command has JSON key attribute disabled set to `true`:

```
{  
  "disabled": true  
}
```

Remediation:

Using Google Cloud Console

1. Go to Kubernetes GCP Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Select the Kubernetes cluster for which the web UI is enabled
3. Click EDIT
4. Click on the 'Add-ons' heading to expand, and set 'Kubernetes dashboard' to 'Disabled'
5. Click SAVE.

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
```



Default Value:






The Kubernetes web UI (Dashboard) does not have admin access by default in GKE 1.7 and higher. The Kubernetes web UI is disabled by default in GKE 1.10 and higher. In GKE 1.15 and higher, the Kubernetes web UI add-on `KubernetesDashboard` is no longer supported as a managed add-on.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster#disable_kubernetes_dashboard

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.8 <u>Uninstall or Disable Unnecessary Services on Enterprise Assets and Software</u> Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.			

Controls Version	Control	IG 1	IG 2	IG 3
v7	2.2 <u>Ensure Software is Supported by Vendor</u> Ensure that only software applications or operating systems currently supported by the software's vendor are added to the organization's authorized software inventory. Unsupported software should be tagged as unsupported in the inventory system.			
v7	18.4 <u>Only Use Up-to-date And Trusted Third-Party Components</u> Only use up-to-date and trusted third-party components for the software developed by the organization.			

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

Profile Applicability:

- Level 1

Description:

Alpha clusters are not covered by an SLA and are not production-ready.

Rationale:

Alpha clusters are designed for early adopters to experiment with workloads that take advantage of new features before those features are production-ready. They have all Kubernetes API features enabled, but are not covered by the GKE SLA, do not receive security updates, have node auto-upgrade and node auto-repair disabled, and cannot be upgraded. They are also automatically deleted after 30 days.

Impact:

Users and workloads will not be able to take advantage of features included within Alpha clusters.

Audit:

The audit script for this recommendation utilizes 3 variables:

\$CLUSTER_NAME

\$COMPUTE_ZONE

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. If a cluster appears under the 'Kubernetes alpha clusters' heading, it is an Alpha cluster.

Using Command Line

Run the command:

```
gcloud container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE-ZONE \
  --format json | jq '.enableKubernetesAlpha'
```

The output of the above command will return `true` if it is an Alpha cluster.

Remediation:

Alpha features cannot be disabled. To remediate, a new cluster must be created.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/>
2. Click CREATE CLUSTER
3. Unless Node Auto-Upgrade and Node Auto-Repair are disabled, under 'Availability, networking, security, and additional features', the option 'Enable Kubernetes alpha features in this cluster' will not be available. Ensure this feature is not checked
4. Click CREATE.

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.





Default Value:

By default, Kubernetes Alpha features are disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/concepts/alpha-clusters>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.8 <u>Separate Production and Non-Production Systems</u> Maintain separate environments for production and non-production systems.			
v7	18.9 <u>Separate Production and Non-Production Systems</u> Maintain separate environments for production and nonproduction systems. Developers should not have unmonitored access to production environments.			

5.10.3 Ensure Pod Security Policy is Enabled and set as appropriate (Manual)

Profile Applicability:

- Level 1

Description:

Pod Security Policy should be used to prevent privileged containers where possible and enforce namespace and workload configurations.

Rationale:

A Pod Security Policy is a cluster-level resource that controls security sensitive aspects of the pod specification. A `PodSecurityPolicy` object defines a set of conditions that a pod must run with in order to be accepted into the system, as well as defaults for the related fields. When a request to create or update a Pod does not meet the conditions in the Pod Security Policy, that request is rejected and an error is returned. The Pod Security Policy admission controller validates requests against available Pod Security Policies.

PodSecurityPolicies specify a list of restrictions, requirements, and defaults for Pods created under the policy. See further details on recommended policies in Recommendation section 5.2.

Impact:

If you enable the Pod Security Policy controller without first defining and authorizing any actual policies, no users, controllers, or service accounts can create or update Pods. If you are working with an existing cluster, you should define and authorize policies before enabling the controller.

Audit:

The audit script for this recommendation utilizes 3 variables:

`$CLUSTER_NAME`

`$COMPUTE_ZONE`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console:

There is no means of auditing the Pod Security Policy Admission controller from the console.

Using Command Line

To check Pod Security Policy Admission Controller is enabled for an existing cluster, run the following command,

```
gcloud beta container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE \
  --format json | jq '.podSecurityPolicyConfig'
```

Ensure the output of the above command has JSON key attribute enabled set to `true`:

```
{
  "enabled": true
}
```

If Pod Security Policy is disabled, the above command output will return null (`{ }`).

Remediation:

Using Google Cloud Console

There is no means of enabling the Pod Security Policy Admission controller on an existing or new cluster from the console.

Using Command Line

To enable Pod Security Policy for an existing cluster, run the following command:

```
gcloud beta container clusters update [CLUSTER_NAME] \
  --zone [COMPUTE_ZONE] \
  --enable-pod-security-policy
```

Default Value:

By default, Pod Security Policy is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/pod-security-policies>
2. <https://kubernetes.io/docs/concepts/policy/pod-security-policy>

Additional Information:

Pod Security Policy is Beta in Kubernetes and in GKE. This feature is not covered by any SLA or deprecation policy and might be subject to backward-incompatible changes.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.1 Centralize Security Event Alerting Centralize security event alerting across enterprise assets for log correlation and analysis. Best practice implementation requires the use of a SIEM, which includes vendor-defined event correlation alerts. A log analytics platform configured with security-relevant correlation alerts also satisfies this Safeguard.		●	●
v7	5.1 Establish Secure Configurations Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.4 Deploy System Configuration Management Tools Deploy system configuration management tools that will automatically enforce and redeploy configuration settings to systems at regularly scheduled intervals.		●	●

5.10.4 Consider GKE Sandbox for running untrusted workloads (Manual)

Profile Applicability:

- Level 2

Description:

Use GKE Sandbox to restrict untrusted workloads as an additional layer of protection when running in a multi-tenant environment.

Rationale:

GKE Sandbox provides an extra layer of security to prevent untrusted code from affecting the host kernel on your cluster nodes.

When you enable GKE Sandbox on a Node pool, a sandbox is created for each Pod running on a node in that Node pool. In addition, nodes running sandboxed Pods are prevented from accessing other GCP services or cluster metadata. Each sandbox uses its own userspace kernel.

Multi-tenant clusters and clusters whose containers run untrusted workloads are more exposed to security vulnerabilities than other clusters. Examples include SaaS providers, web-hosting providers, or other organizations that allow their users to upload and run code. A flaw in the container runtime or in the host kernel could allow a process running within a container to 'escape' the container and affect the node's kernel, potentially bringing down the node.

The potential also exists for a malicious tenant to gain access to and exfiltrate another tenant's data in memory or on disk, by exploiting such a defect.

Impact:

Using GKE Sandbox requires the node image to be set to Container-Optimized OS with containerd (`cos_containerd`).

It is not currently possible to use GKE Sandbox along with the following Kubernetes features:

- Accelerators such as GPUs or TPUs
- Istio
- Monitoring statistics at the level of the Pod or container
- Hostpath storage
- Per-container PID namespace
- CPU and memory limits are only applied for Guaranteed Pods and Burstable Pods, and only when CPU and memory limits are specified for all containers running in the Pod

- Pods using PodSecurityPolicies that specify host namespaces, such as hostNetwork, hostPID, or hostIPC
- Pods using PodSecurityPolicy settings such as privileged mode
- VolumeDevices
- Portforward
- Linux kernel security modules such as Seccomp, Apparmor, or Selinux Sysctl, NoNewPrivileges, bidirectional MountPropagation, FSGroup, or ProcMount

Audit:

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on each cluster, and click on any Node pools that are not provisioned by default
3. On the Node pool Details page, under the 'Security' heading on the Node pool details page, check that 'Sandbox with gVisor' is set to 'Enabled'.

The default node pool cannot use GKE Sandbox.

Using Command Line

Run this command:

```
gcloud container node-pools describe [NODE_POOL_NAME] \
  --zone [COMPUTE_ZONE] \
  --cluster [CLUSTER_NAME] \
  --format json | jq '.config.sandboxConfig'
```

The output of the above command will return the following if the Node pool is running a sandbox:

```
{
  "sandboxType": "gvisor"
}
```

If there is no sandbox, the above command output will be null ({}).

The default node pool cannot use GKE Sandbox.

Remediation:

Once a node pool is created, GKE Sandbox cannot be enabled, rather a new node pool is required. The default node pool (the first node pool in your cluster, created when the cluster is created) cannot use GKE Sandbox.

Using Google Cloud Console

1. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/>
2. Click ADD NODE POOL
3. Configure the Node pool with following settings:
 - For the node version, select v1.12.6-gke.8 or higher

- For the node image, select 'Container-Optimized OS with Containerd (cos_containerd) (beta)'
 - Under 'Security', select 'Enable sandbox with gVisor'
4. Configure other Node pool settings as required
 5. Click SAVE.

Using Command Line

To enable GKE Sandbox on an existing cluster, a new Node pool must be created.

```
gcloud container node-pools create [NODE_POOL_NAME] \
  --zone=[COMPUTE_ZONE] \
  --cluster=[CLUSTER_NAME] \
  --image-type=cos_containerd \
  --sandbox type=gvisor
```

Default Value:

By default, GKE Sandbox is disabled.

References:

1. <https://cloud.google.com/kubernetes-engine/docs/how-to/sandbox-pods>





Additional Information:

The default node pool (the first node pool in your cluster, created when the cluster is created) cannot use GKE Sandbox.

When using GKE Sandbox, your cluster must have at least two node pools. You must always have at least one node pool where GKE Sandbox is disabled. This node pool must contain at least one node, even if all your workloads are sandboxed.

It is optional but recommended that you enable Stackdriver Logging and Stackdriver Monitoring, by adding the flag `--enable-stackdriver-kubernetes`. gVisor messages are logged.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	16.8 Separate Production and Non-Production Systems Maintain separate environments for production and non-production systems.			
v7	18.9 Separate Production and Non-Production Systems Maintain separate environments for production and nonproduction systems. Developers should not have unmonitored access to production environments.			

5.10.5 Ensure use of Binary Authorization (Automated)

Profile Applicability:

- Level 2

Description:

Binary Authorization helps to protect supply-chain security by only allowing images with verifiable cryptographically signed metadata into the cluster.

Rationale:

Binary Authorization provides software supply-chain security for images that you deploy to GKE from Google Container Registry (GCR) or another container image registry.

Binary Authorization requires images to be signed by trusted authorities during the development process. These signatures are then validated at deployment time. By enforcing validation, you can gain tighter control over your container environment by ensuring only verified images are integrated into the build-and-release process.

Impact:

Care must be taken when defining policy in order to prevent inadvertent denial of container image deployments. Depending on policy, attestations for existing container images running within the cluster may need to be created before those images are redeployed or pulled as part of the pod churn.

To prevent key system images from being denied deployment, consider the use of global policy evaluation mode, which uses a global policy provided by Google and exempts a list of Google-provided system images from further policy evaluation.

Audit:

The audit script for this recommendation utilizes 2 variables:

`$CLUSTER_NAME`

`$COMPUTE_ZONE`

Please set these parameters on the system where you will be executing your gcloud audit script or command.

Using Google Cloud Console

To check that Binary Authorization is enabled for the GKE cluster:

1. Go to the Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
2. Click on the desired cluster. On the Details pane, ensure that 'Binary Authorization' is set to 'Enabled'.
Then, assess the contents of the policy:
3. Go to Binary Authorization by visiting <https://console.cloud.google.com/security/binary-authorization>

4. Ensure a policy is defined and the project default rule is not configured to 'Allow all images'.

Using Command Line

To check that Binary Authorization is enabled for the GKE cluster:

```
gcloud container clusters describe $CLUSTER_NAME \
  --zone $COMPUTE_ZONE \
  --format json | jq .binaryAuthorization
```

The above command output will be the following if Binary Authorization is enabled:

```
{
  "enabled": true
}
```

Then, assess the contents of the policy:

```
gcloud container binauthz policy export > current-policy.yaml
```

Ensure that the current policy is not configured to allow all images (evaluationMode: ALWAYS_ALLOW):

```
cat current-policy.yaml
...
defaultAdmissionRule:
  evaluationMode: ALWAYS_ALLOW
```

Remediation:

Using Google Cloud Console

1. Go to Binary Authorization by visiting <https://console.cloud.google.com/security/binary-authorization>
2. Enable the Binary Authorization API (if disabled)
3. Go to Kubernetes Engine by visiting <https://console.cloud.google.com/kubernetes/list>
4. Select the Kubernetes cluster for which Binary Authorization is disabled
5. Click EDIT
6. Set 'Binary Authorization' to 'Enabled'
7. Click SAVE
8. Return to Binary Authorization at <https://console.cloud.google.com/security/binary-authorization>
9. Set an appropriate policy for your cluster.

Using Command Line

Update the cluster to enable Binary Authorization:

```
gcloud container cluster update [CLUSTER_NAME] \
  --zone [COMPUTE-ZONE] \
  --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]
```

Default Value:

By default, Binary Authorization is disabled.

References:

1. <https://cloud.google.com/binary-authorization/>

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v7	5.2 <u>Maintain Secure Images</u> Maintain secure images or templates for all systems in the enterprise based on the organization's approved configuration standards. Any new system deployment or existing system that becomes compromised should be imaged using one of those images or templates.		●	●

5.10.6 Enable Cloud Security Command Center (Cloud SCC) (Manual)

Profile Applicability:

- Level 1

Description:

Enable Cloud Security Command Center (Cloud SCC) to provide a centralized view of security for your GKE clusters.

Rationale:

Cloud Security Command Center (Cloud SCC) is the canonical security and data risk database for GCP. Cloud SCC enables you to understand your security and data attack surface by providing asset inventory, discovery, search, and management.

Impact:

None.

Audit:

Using Google Cloud Console

1. Navigate to the Cloud SCC dashboard at <https://console.cloud.google.com/security/command-center/dashboard>
2. Observe the 'Assets Summary' is populated with resources.

Using Command Line

To determine if Cloud SCC is enabled and indexing assets, observe the output of the following command:

```
gcloud alpha scc assets list $PROJECT_ID
```

If the output of the above command returns GKE assets, Cloud SCC is enabled and indexing GKE resources.

Remediation:

Follow the instructions at <https://cloud.google.com/security-command-center/docs/quickstart-scc-setup>.

Default Value:

By default, Cloud SCC is disabled.

References:




1. <https://cloud.google.com/security-command-center/>

2. <https://cloud.google.com/security-command-center/docs/quickstart-scc-setup>

Additional Information:

Cloud SCC is only available at the organization level. Your GCP projects must belong to a GCP organization.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	15.6 Monitor Service Providers Monitor service providers consistent with the enterprise's service provider management policy. Monitoring may include periodic reassessment of service provider compliance, monitoring service provider release notes, and dark web monitoring.			
v7	5.5 Implement Automated Configuration Monitoring Systems Utilize a Security Content Automation Protocol (SCAP) compliant configuration monitoring system to verify all security configuration elements, catalog approved exceptions, and alert when unauthorized changes occur.			

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	Control Plane Components		
2	Control Plane Configuration		
2.1	Authentication and Authorization		
2.1.1	Client certificate authentication should not be used for users (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
2.2	Logging		
2.2.1	Ensure that a minimal audit policy is created (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
2.2.2	Ensure that the audit policy covers key security concerns (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3	Worker Nodes		
3.1	Worker Node Configuration Files		
3.1.1	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3.2	Kubelet		
3.2.1	Ensure that the --anonymous-auth argument is set to false (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)	<input type="checkbox"/>	<input type="checkbox"/>

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
3.2.3	Ensure that the --client-ca-file argument is set as appropriate (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.4	Ensure that the --read-only-port argument is set to 0 (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.7	Ensure that the --hostname-override argument is not set (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.10	Ensure that the --rotate-certificates argument is not set to false (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
3.2.11	Ensure that the RotateKubeletServerCertificate argument is set to true (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
4	Policies		
4.1	RBAC and Service Accounts		
4.1.1	Ensure that the cluster-admin role is only used where required (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.1.2	Minimize access to secrets (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.1.3	Minimize wildcard use in Roles and ClusterRoles (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.1.4	Minimize access to create pods (Manual)	<input type="checkbox"/>	<input type="checkbox"/>

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.1.5	Ensure that default service accounts are not actively used. (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.1.6	Ensure that Service Account Tokens are only mounted where necessary (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2	Pod Security Standards		
4.2.1	Minimize the admission of privileged containers (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.3	Network Policies and CNI		
4.3.1	Ensure that the CNI in use supports Network Policies (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.3.2	Ensure that all Namespaces have Network Policies defined (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.4	Secrets Management		
4.4.1	Prefer using secrets as files over secrets as environment variables (Manual)	<input type="checkbox"/>	<input type="checkbox"/>

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.4.2	Consider external secret storage (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.5	Extensible Admission Control		
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.6	General Policies		
4.6.1	Create administrative boundaries between resources using namespaces (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.6.2	Ensure that the seccomp profile is set to docker/default in your pod definitions (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.6.3	Apply Security Context to Your Pods and Containers (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
4.6.4	The default namespace should not be used (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5	Managed services		
5.1	Image Registry and Image Scanning		
5.1.1	Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.1.3	Minimize cluster access to read-only for GCR (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.1.4	Minimize Container Registries to only those approved (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.2	Identity and Access Management (IAM)		
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity (Manual)	<input type="checkbox"/>	<input type="checkbox"/>

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.3	Cloud Key Management Service (Cloud KMS)		
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.4	Node Metadata		
5.4.1	Ensure legacy Compute Engine instance metadata APIs are Disabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.4.2	Ensure the GKE Metadata Server is Enabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.5	Node Configuration and Maintenance		
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE node images (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.5.2	Ensure Node Auto-Repair is enabled for GKE nodes (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.5.5	Ensure Shielded GKE Nodes are Enabled (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.6	Cluster Networking		
5.6.1	Enable VPC Flow Logs and Intranode Visibility (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.6.2	Ensure use of VPC-native clusters (Automated)	<input type="checkbox"/>	<input type="checkbox"/>

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.6.3	Ensure Master Authorized Networks is Enabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.6.5	Ensure clusters are created with Private Nodes (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.6.6	Consider firewalling GKE worker nodes (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.6.7	Ensure Network Policy is Enabled and set as appropriate (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.6.8	Ensure use of Google-managed SSL Certificates (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.7	Logging		
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.7.2	Enable Linux auditd logging (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.8	Authentication and Authorization		
5.8.1	Ensure Basic Authentication using static passwords is Disabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.8.2	Ensure authentication using Client Certificates is Disabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.8.3	Manage Kubernetes RBAC users with Google Groups for GKE (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.8.4	Ensure Legacy Authorization (ABAC) is Disabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.9	Storage		
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD) (Manual)	<input type="checkbox"/>	<input type="checkbox"/>

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.10	Other Cluster Configurations		
5.10.1	Ensure Kubernetes Web UI is Disabled (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.2	Ensure that Alpha clusters are not used for production workloads (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.3	Ensure Pod Security Policy is Enabled and set as appropriate (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.4	Consider GKE Sandbox for running untrusted workloads (Manual)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.5	Ensure use of Binary Authorization (Automated)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.6	Enable Cloud Security Command Center (Cloud SCC) (Manual)	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	<input type="checkbox"/>	<input type="checkbox"/>
2.2.1	Ensure that a minimal audit policy is created	<input type="checkbox"/>	<input type="checkbox"/>
3.2.1	Ensure that the --anonymous-auth argument is set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture	<input type="checkbox"/>	<input type="checkbox"/>
4.1.1	Ensure that the cluster-admin role is only used where required	<input type="checkbox"/>	<input type="checkbox"/>
4.1.5	Ensure that default service accounts are not actively used.	<input type="checkbox"/>	<input type="checkbox"/>
4.2.1	Minimize the admission of privileged containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned	<input type="checkbox"/>	<input type="checkbox"/>
4.6.1	Create administrative boundaries between resources using namespaces	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
4.6.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	<input type="checkbox"/>	<input type="checkbox"/>
4.6.4	The default namespace should not be used	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	<input type="checkbox"/>	<input type="checkbox"/>
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	<input type="checkbox"/>	<input type="checkbox"/>
5.6.3	Ensure Master Authorized Networks is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.7	Ensure Network Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.10.1	Ensure Kubernetes Web UI is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.10.3	Ensure Pod Security Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	<input type="checkbox"/>	<input type="checkbox"/>
2.2.1	Ensure that a minimal audit policy is created	<input type="checkbox"/>	<input type="checkbox"/>
3.1.1	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.2.1	Ensure that the --anonymous-auth argument is set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	<input type="checkbox"/>	<input type="checkbox"/>
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.4	Ensure that the --read-only-port argument is set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture	<input type="checkbox"/>	<input type="checkbox"/>
3.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.10	Ensure that the --rotate-certificates argument is not set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.11	Ensure that the RotateKubeletServerCertificate argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
4.1.1	Ensure that the cluster-admin role is only used where required	<input type="checkbox"/>	<input type="checkbox"/>
4.1.2	Minimize access to secrets	<input type="checkbox"/>	<input type="checkbox"/>
4.1.3	Minimize wildcard use in Roles and ClusterRoles	<input type="checkbox"/>	<input type="checkbox"/>
4.1.4	Minimize access to create pods	<input type="checkbox"/>	<input type="checkbox"/>
4.1.5	Ensure that default service accounts are not actively used.	<input type="checkbox"/>	<input type="checkbox"/>
4.2.1	Minimize the admission of privileged containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned	<input type="checkbox"/>	<input type="checkbox"/>
4.3.1	Ensure that the CNI in use supports Network Policies	<input type="checkbox"/>	<input type="checkbox"/>
4.3.2	Ensure that all Namespaces have Network Policies defined	<input type="checkbox"/>	<input type="checkbox"/>
4.4.1	Prefer using secrets as files over secrets as environment variables	<input type="checkbox"/>	<input type="checkbox"/>
4.6.1	Create administrative boundaries between resources using namespaces	<input type="checkbox"/>	<input type="checkbox"/>
4.6.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	<input type="checkbox"/>	<input type="checkbox"/>
4.6.4	The default namespace should not be used	<input type="checkbox"/>	<input type="checkbox"/>
5.1.1	Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.3	Minimize cluster access to read-only for GCR	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
5.1.4	Minimize Container Registries to only those approved	<input type="checkbox"/>	<input type="checkbox"/>
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	<input type="checkbox"/>	<input type="checkbox"/>
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	<input type="checkbox"/>	<input type="checkbox"/>
5.4.1	Ensure legacy Compute Engine instance metadata APIs are Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.4.2	Ensure the GKE Metadata Server is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE node images	<input type="checkbox"/>	<input type="checkbox"/>
5.5.2	Ensure Node Auto-Repair is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	<input type="checkbox"/>	<input type="checkbox"/>
5.5.5	Ensure Shielded GKE Nodes are Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.1	Enable VPC Flow Logs and Intranode Visibility	<input type="checkbox"/>	<input type="checkbox"/>
5.6.2	Ensure use of VPC-native clusters	<input type="checkbox"/>	<input type="checkbox"/>
5.6.3	Ensure Master Authorized Networks is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.7	Ensure Network Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.6.8	Ensure use of Google-managed SSL Certificates	<input type="checkbox"/>	<input type="checkbox"/>
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.7.2	Enable Linux auditd logging	<input type="checkbox"/>	<input type="checkbox"/>
5.8.3	Manage Kubernetes RBAC users with Google Groups for GKE	<input type="checkbox"/>	<input type="checkbox"/>
5.10.1	Ensure Kubernetes Web UI is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.10.2	Ensure that Alpha clusters are not used for production workloads	<input type="checkbox"/>	<input type="checkbox"/>
5.10.3	Ensure Pod Security Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.10.4	Consider GKE Sandbox for running untrusted workloads	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
5.10.5	Ensure use of Binary Authorization	<input type="checkbox"/>	<input type="checkbox"/>
5.10.6	Enable Cloud Security Command Center (Cloud SCC)	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	<input type="checkbox"/>	<input type="checkbox"/>
2.2.1	Ensure that a minimal audit policy is created	<input type="checkbox"/>	<input type="checkbox"/>
2.2.2	Ensure that the audit policy covers key security concerns	<input type="checkbox"/>	<input type="checkbox"/>
3.1.1	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.2.1	Ensure that the --anonymous-auth argument is set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	<input type="checkbox"/>	<input type="checkbox"/>
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.4	Ensure that the --read-only-port argument is set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture	<input type="checkbox"/>	<input type="checkbox"/>
3.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.10	Ensure that the --rotate-certificates argument is not set to false	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
3.2.11	Ensure that the RotateKubeletServerCertificate argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
4.1.1	Ensure that the cluster-admin role is only used where required	<input type="checkbox"/>	<input type="checkbox"/>
4.1.2	Minimize access to secrets	<input type="checkbox"/>	<input type="checkbox"/>
4.1.3	Minimize wildcard use in Roles and ClusterRoles	<input type="checkbox"/>	<input type="checkbox"/>
4.1.4	Minimize access to create pods	<input type="checkbox"/>	<input type="checkbox"/>
4.1.5	Ensure that default service accounts are not actively used.	<input type="checkbox"/>	<input type="checkbox"/>
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	<input type="checkbox"/>	<input type="checkbox"/>
4.2.1	Minimize the admission of privileged containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned	<input type="checkbox"/>	<input type="checkbox"/>
4.3.1	Ensure that the CNI in use supports Network Policies	<input type="checkbox"/>	<input type="checkbox"/>
4.3.2	Ensure that all Namespaces have Network Policies defined	<input type="checkbox"/>	<input type="checkbox"/>
4.4.1	Prefer using secrets as files over secrets as environment variables	<input type="checkbox"/>	<input type="checkbox"/>
4.4.2	Consider external secret storage	<input type="checkbox"/>	<input type="checkbox"/>
4.6.1	Create administrative boundaries between resources using namespaces	<input type="checkbox"/>	<input type="checkbox"/>
4.6.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
4.6.4	The default namespace should not be used	<input type="checkbox"/>	<input type="checkbox"/>
5.1.1	Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.3	Minimize cluster access to read-only for GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.4	Minimize Container Registries to only those approved	<input type="checkbox"/>	<input type="checkbox"/>
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	<input type="checkbox"/>	<input type="checkbox"/>
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	<input type="checkbox"/>	<input type="checkbox"/>
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS	<input type="checkbox"/>	<input type="checkbox"/>
5.4.1	Ensure legacy Compute Engine instance metadata APIs are Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.4.2	Ensure the GKE Metadata Server is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE node images	<input type="checkbox"/>	<input type="checkbox"/>
5.5.2	Ensure Node Auto-Repair is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	<input type="checkbox"/>	<input type="checkbox"/>
5.5.5	Ensure Shielded GKE Nodes are Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.1	Enable VPC Flow Logs and Intranode Visibility	<input type="checkbox"/>	<input type="checkbox"/>
5.6.2	Ensure use of VPC-native clusters	<input type="checkbox"/>	<input type="checkbox"/>
5.6.3	Ensure Master Authorized Networks is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.6	Consider firewalling GKE worker nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.6.7	Ensure Network Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.6.8	Ensure use of Google-managed SSL Certificates	<input type="checkbox"/>	<input type="checkbox"/>
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.7.2	Enable Linux auditd logging	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
5.8.3	Manage Kubernetes RBAC users with Google Groups for GKE	<input type="checkbox"/>	<input type="checkbox"/>
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.1	Ensure Kubernetes Web UI is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.10.2	Ensure that Alpha clusters are not used for production workloads	<input type="checkbox"/>	<input type="checkbox"/>
5.10.3	Ensure Pod Security Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.10.4	Consider GKE Sandbox for running untrusted workloads	<input type="checkbox"/>	<input type="checkbox"/>
5.10.5	Ensure use of Binary Authorization	<input type="checkbox"/>	<input type="checkbox"/>
5.10.6	Enable Cloud Security Command Center (Cloud SCC)	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v7 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
	No unmapped recommendations to CIS Controls v7.0	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	<input type="checkbox"/>	<input type="checkbox"/>
2.2.1	Ensure that a minimal audit policy is created	<input type="checkbox"/>	<input type="checkbox"/>
3.1.1	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.2.1	Ensure that the --anonymous-auth argument is set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture	<input type="checkbox"/>	<input type="checkbox"/>
4.1.1	Ensure that the cluster-admin role is only used where required	<input type="checkbox"/>	<input type="checkbox"/>
4.1.2	Minimize access to secrets	<input type="checkbox"/>	<input type="checkbox"/>
4.1.3	Minimize wildcard use in Roles and ClusterRoles	<input type="checkbox"/>	<input type="checkbox"/>
4.1.5	Ensure that default service accounts are not actively used.	<input type="checkbox"/>	<input type="checkbox"/>
4.2.1	Minimize the admission of privileged containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned	<input type="checkbox"/>	<input type="checkbox"/>
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	<input type="checkbox"/>	<input type="checkbox"/>
4.6.3	Apply Security Context to Your Pods and Containers	<input type="checkbox"/>	<input type="checkbox"/>
4.6.4	The default namespace should not be used	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.3	Minimize cluster access to read-only for GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	<input type="checkbox"/>	<input type="checkbox"/>
5.6.3	Ensure Master Authorized Networks is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.5	Ensure clusters are created with Private Nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.6.6	Consider firewalling GKE worker nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.7.2	Enable Linux auditd logging	<input type="checkbox"/>	<input type="checkbox"/>
5.8.1	Ensure Basic Authentication using static passwords is Disabled	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	<input type="checkbox"/>	<input type="checkbox"/>
2.2.1	Ensure that a minimal audit policy is created	<input type="checkbox"/>	<input type="checkbox"/>
2.2.2	Ensure that the audit policy covers key security concerns	<input type="checkbox"/>	<input type="checkbox"/>
3.1.1	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.2.1	Ensure that the --anonymous-auth argument is set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	<input type="checkbox"/>	<input type="checkbox"/>
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.4	Ensure that the --read-only-port argument is set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
3.2.7	Ensure that the --hostname-override argument is not set	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture	<input type="checkbox"/>	<input type="checkbox"/>
3.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.10	Ensure that the --rotate-certificates argument is not set to false	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
3.2.11	Ensure that the RotateKubeletServerCertificate argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
4.1.1	Ensure that the cluster-admin role is only used where required	<input type="checkbox"/>	<input type="checkbox"/>
4.1.2	Minimize access to secrets	<input type="checkbox"/>	<input type="checkbox"/>
4.1.3	Minimize wildcard use in Roles and ClusterRoles	<input type="checkbox"/>	<input type="checkbox"/>
4.1.5	Ensure that default service accounts are not actively used.	<input type="checkbox"/>	<input type="checkbox"/>
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	<input type="checkbox"/>	<input type="checkbox"/>
4.2.1	Minimize the admission of privileged containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned	<input type="checkbox"/>	<input type="checkbox"/>
4.3.1	Ensure that the CNI in use supports Network Policies	<input type="checkbox"/>	<input type="checkbox"/>
4.3.2	Ensure that all Namespaces have Network Policies defined	<input type="checkbox"/>	<input type="checkbox"/>
4.4.2	Consider external secret storage	<input type="checkbox"/>	<input type="checkbox"/>
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	<input type="checkbox"/>	<input type="checkbox"/>
4.6.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	<input type="checkbox"/>	<input type="checkbox"/>
4.6.3	Apply Security Context to Your Pods and Containers	<input type="checkbox"/>	<input type="checkbox"/>
4.6.4	The default namespace should not be used	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
5.1.1	Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.3	Minimize cluster access to read-only for GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.4	Minimize Container Registries to only those approved	<input type="checkbox"/>	<input type="checkbox"/>
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS	<input type="checkbox"/>	<input type="checkbox"/>
5.4.1	Ensure legacy Compute Engine instance metadata APIs are Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.4.2	Ensure the GKE Metadata Server is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE node images	<input type="checkbox"/>	<input type="checkbox"/>
5.5.2	Ensure Node Auto-Repair is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	<input type="checkbox"/>	<input type="checkbox"/>
5.6.1	Enable VPC Flow Logs and Intranode Visibility	<input type="checkbox"/>	<input type="checkbox"/>
5.6.2	Ensure use of VPC-native clusters	<input type="checkbox"/>	<input type="checkbox"/>
5.6.3	Ensure Master Authorized Networks is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.5	Ensure clusters are created with Private Nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.6.6	Consider firewalling GKE worker nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.6.7	Ensure Network Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.6.8	Ensure use of Google-managed SSL Certificates	<input type="checkbox"/>	<input type="checkbox"/>
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.7.2	Enable Linux auditd logging	<input type="checkbox"/>	<input type="checkbox"/>
5.8.1	Ensure Basic Authentication using static passwords is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.1	Ensure Kubernetes Web UI is Disabled	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
5.10.2	Ensure that Alpha clusters are not used for production workloads	<input type="checkbox"/>	<input type="checkbox"/>
5.10.3	Ensure Pod Security Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.10.4	Consider GKE Sandbox for running untrusted workloads	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v8 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
2.1.1	Client certificate authentication should not be used for users	<input type="checkbox"/>	<input type="checkbox"/>
2.2.1	Ensure that a minimal audit policy is created	<input type="checkbox"/>	<input type="checkbox"/>
2.2.2	Ensure that the audit policy covers key security concerns	<input type="checkbox"/>	<input type="checkbox"/>
3.1.1	Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.2	Ensure that the proxy kubeconfig file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 or more restrictive	<input type="checkbox"/>	<input type="checkbox"/>
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	<input type="checkbox"/>	<input type="checkbox"/>
3.2.1	Ensure that the --anonymous-auth argument is set to false	<input type="checkbox"/>	<input type="checkbox"/>
3.2.2	Ensure that the --authorization-mode argument is not set to AlwaysAllow	<input type="checkbox"/>	<input type="checkbox"/>
3.2.3	Ensure that the --client-ca-file argument is set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.4	Ensure that the --read-only-port argument is set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.5	Ensure that the --streaming-connection-idle-timeout argument is not set to 0	<input type="checkbox"/>	<input type="checkbox"/>
3.2.6	Ensure that the --make-iptables-util-chains argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
3.2.7	Ensure that the --hostname-override argument is not set	<input type="checkbox"/>	<input type="checkbox"/>
3.2.8	Ensure that the --eventrecordqps argument is set to 5 or higher or a level which ensures appropriate event capture	<input type="checkbox"/>	<input type="checkbox"/>
3.2.9	Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
3.2.10	Ensure that the --rotate-certificates argument is not set to false	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
3.2.11	Ensure that the RotateKubeletServerCertificate argument is set to true	<input type="checkbox"/>	<input type="checkbox"/>
4.1.1	Ensure that the cluster-admin role is only used where required	<input type="checkbox"/>	<input type="checkbox"/>
4.1.2	Minimize access to secrets	<input type="checkbox"/>	<input type="checkbox"/>
4.1.3	Minimize wildcard use in Roles and ClusterRoles	<input type="checkbox"/>	<input type="checkbox"/>
4.1.4	Minimize access to create pods	<input type="checkbox"/>	<input type="checkbox"/>
4.1.5	Ensure that default service accounts are not actively used.	<input type="checkbox"/>	<input type="checkbox"/>
4.1.6	Ensure that Service Account Tokens are only mounted where necessary	<input type="checkbox"/>	<input type="checkbox"/>
4.2.1	Minimize the admission of privileged containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.2	Minimize the admission of containers wishing to share the host process ID namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.3	Minimize the admission of containers wishing to share the host IPC namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.4	Minimize the admission of containers wishing to share the host network namespace	<input type="checkbox"/>	<input type="checkbox"/>
4.2.5	Minimize the admission of containers with allowPrivilegeEscalation	<input type="checkbox"/>	<input type="checkbox"/>
4.2.6	Minimize the admission of root containers	<input type="checkbox"/>	<input type="checkbox"/>
4.2.7	Minimize the admission of containers with added capabilities	<input type="checkbox"/>	<input type="checkbox"/>
4.2.8	Minimize the admission of containers with capabilities assigned	<input type="checkbox"/>	<input type="checkbox"/>
4.3.1	Ensure that the CNI in use supports Network Policies	<input type="checkbox"/>	<input type="checkbox"/>
4.3.2	Ensure that all Namespaces have Network Policies defined	<input type="checkbox"/>	<input type="checkbox"/>
4.4.1	Prefer using secrets as files over secrets as environment variables	<input type="checkbox"/>	<input type="checkbox"/>
4.4.2	Consider external secret storage	<input type="checkbox"/>	<input type="checkbox"/>
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	<input type="checkbox"/>	<input type="checkbox"/>
4.6.1	Create administrative boundaries between resources using namespaces	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
4.6.2	Ensure that the seccomp profile is set to docker/default in your pod definitions	<input type="checkbox"/>	<input type="checkbox"/>
4.6.3	Apply Security Context to Your Pods and Containers	<input type="checkbox"/>	<input type="checkbox"/>
4.6.4	The default namespace should not be used	<input type="checkbox"/>	<input type="checkbox"/>
5.1.1	Ensure Image Vulnerability Scanning using GCR Container Analysis or a third party provider	<input type="checkbox"/>	<input type="checkbox"/>
5.1.2	Minimize user access to GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.3	Minimize cluster access to read-only for GCR	<input type="checkbox"/>	<input type="checkbox"/>
5.1.4	Minimize Container Registries to only those approved	<input type="checkbox"/>	<input type="checkbox"/>
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS	<input type="checkbox"/>	<input type="checkbox"/>
5.4.1	Ensure legacy Compute Engine instance metadata APIs are Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.4.2	Ensure the GKE Metadata Server is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE node images	<input type="checkbox"/>	<input type="checkbox"/>
5.5.2	Ensure Node Auto-Repair is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.3	Ensure Node Auto-Upgrade is enabled for GKE nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	<input type="checkbox"/>	<input type="checkbox"/>
5.6.1	Enable VPC Flow Logs and Intranode Visibility	<input type="checkbox"/>	<input type="checkbox"/>
5.6.2	Ensure use of VPC-native clusters	<input type="checkbox"/>	<input type="checkbox"/>
5.6.3	Ensure Master Authorized Networks is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.6.5	Ensure clusters are created with Private Nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.6.6	Consider firewalling GKE worker nodes	<input type="checkbox"/>	<input type="checkbox"/>
5.6.7	Ensure Network Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.6.8	Ensure use of Google-managed SSL Certificates	<input type="checkbox"/>	<input type="checkbox"/>
5.7.1	Ensure Stackdriver Kubernetes Logging and Monitoring is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.7.2	Enable Linux auditd logging	<input type="checkbox"/>	<input type="checkbox"/>
5.8.1	Ensure Basic Authentication using static passwords is Disabled	<input type="checkbox"/>	<input type="checkbox"/>

Recommendation		Set Correctly	
		Yes	No
5.8.2	Ensure authentication using Client Certificates is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.8.3	Manage Kubernetes RBAC users with Google Groups for GKE	<input type="checkbox"/>	<input type="checkbox"/>
5.8.4	Ensure Legacy Authorization (ABAC) is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD)	<input type="checkbox"/>	<input type="checkbox"/>
5.10.1	Ensure Kubernetes Web UI is Disabled	<input type="checkbox"/>	<input type="checkbox"/>
5.10.2	Ensure that Alpha clusters are not used for production workloads	<input type="checkbox"/>	<input type="checkbox"/>
5.10.3	Ensure Pod Security Policy is Enabled and set as appropriate	<input type="checkbox"/>	<input type="checkbox"/>
5.10.4	Consider GKE Sandbox for running untrusted workloads	<input type="checkbox"/>	<input type="checkbox"/>
5.10.6	Enable Cloud Security Command Center (Cloud SCC)	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: CIS Controls v8 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	<input type="checkbox"/>	<input type="checkbox"/>
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	<input type="checkbox"/>	<input type="checkbox"/>
5.5.5	Ensure Shielded GKE Nodes are Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	<input type="checkbox"/>	<input type="checkbox"/>
5.10.5	Ensure use of Binary Authorization	<input type="checkbox"/>	<input type="checkbox"/>

Appendix: Change History

Date	Version	Changes for this version
Apr 9, 2023	1.3.0	The current state of check GKE 3.2.9 is at best undefined: "set to 0 or a level which ensures appropriate event capture". (Ticket 17664)
Apr 9, 2023	1.3.0	Ticket #16491 Edited recommendation 5.2.10
Apr 3, 2023	1.3.0	Ticket # 16624 Updated Recommendation 5.3.2 moved flags in audit process to end of the command line
Apr 3, 2023	1.3.0	Ticket # 16625Updated Recommendation 5.7.1 default value statement.
Apr 3, 2023	1.3.0	Ticket #18522 Set Pod Security Policy Recommendations to Manual in preparation for PSP removal in v1.25 and beyond.
Mar 22, 2023	1.3.0	Ticket #17029 Recommendation 4.2.1 Minimize the admission of privileged containers is deprecated.

Date	Version	Changes for this version
Mar 13, 2023	1.3.0	Ticket #16686 Updated recommendation 4.2.4 to resolve conflict with 4.2.8
Mar 13, 2023	1.3.0	Ticket # 15917 edited –event-qps option with eventRecordQPS