CIS Google Kubernetes Engine (GKE) Benchmark

v1.9.0 - 11-25-2025

Terms of Use

Table of Contents

Overview

All CIS Benchmarks™ (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 and applications for vulnerabilities and quickly updating with the latest security patches.
- End-point protection (Antivirus software, Endpoint Detection and Response (EDR), etc.).
- Logging and monitoring user and system activity.

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

Important Usage Information

All Benchmarks are available free for non-commercial use from the CIS Website. They can be used to manually assess and remediate systems and applications. In lieu of manual assessment and remediation, there are several tools available to assist with assessment:

- CIS Configuration Assessment Tool (CIS-CAT® Pro Assessor)
- CIS Benchmarks™ Certified 3rd Party Tooling

These tools make the hardening process much more scalable for large numbers of systems and applications.

NOTE: Some tooling focuses only on the Benchmark Recommendations that can be fully automated (skipping ones marked **Manual**). It is important that ***ALL*** Recommendations (**Automated** and **Manual**) be addressed since all are important for properly securing systems and are typically in scope for audits.

Key Stakeholders

Cybersecurity is a collaborative effort, and cross functional cooperation is imperative within an organization to discuss, test, and deploy Benchmarks in an effective and efficient way. The Benchmarks are developed to be best practice configuration guidelines applicable to a wide range of use cases. In some organizations, exceptions to specific Recommendations will be needed, and this team should work to prioritize the problematic Recommendations based on several factors like risk, time, cost, and labor. These exceptions should be properly categorized and documented for auditing purposes.

Apply the Correct Version of a Benchmark

Benchmarks are developed and tested for a specific set of products and versions and applying an incorrect Benchmark to a system can cause the resulting pass/fail score to be incorrect. This is due to the assessment of settings that do not apply to the target systems. To assure the correct Benchmark is being assessed:

- **Deploy the Benchmark applicable to the way settings are managed in the environment:** An example of this is the Microsoft Windows family of Benchmarks, which have separate Benchmarks for Group Policy, Intune, and Stand-alone systems based upon how system management is deployed. Applying the wrong Benchmark in this case will give invalid results.

- **Use the most recent version of a Benchmark**: This is true for all Benchmarks, but especially true for cloud technologies. Cloud technologies change frequently and using an older version of a Benchmark may have invalid methods for auditing and remediation.

Exceptions

The guidance items in the Benchmarks are called recommendations and not requirements, and exceptions to some of them are expected and acceptable. The Benchmarks strive to be a secure baseline, or starting point, for a specific technology, with known issues identified during Benchmark development are documented in the Impact section of each Recommendation. In addition, organizational, system specific requirements, or local site policy may require changes as well, or an exception to a Recommendation or group of Recommendations (e.g. A Benchmark could Recommend that a Web server not be installed on the system, but if a system's primary purpose is to function as a Webserver, there should be a documented exception to this Recommendation for that specific server).

In the end, exceptions to some Benchmark Recommendations are common and acceptable, and should be handled as follows:

- The reasons for the exception should be reviewed cross-functionally and be well documented for audit purposes.
- A plan should be developed for mitigating, or eliminating, the exception in the future, if applicable.
- If the organization decides to accept the risk of this exception (not work toward mitigation or elimination), this should be documented for audit purposes.

It is the responsibility of the organization to determine their overall security policy, and which settings are applicable to their unique needs based on the overall risk profile for the organization.

Remediation

CIS has developed Build Kits for many technologies to assist in the automation of hardening systems. Build Kits are designed to correspond to Benchmark's "Remediation" section, which provides the manual remediation steps necessary to make that Recommendation compliant to the Benchmark.

When remediating systems (changing configuration settings on deployed systems as per the Benchmark's Recommendations), please approach this with caution and test thoroughly.

The following is a reasonable remediation approach to follow:

- CIS Build Kits, or internally developed remediation methods should never be applied to production systems without proper testing.
- Proper testing consists of the following:
  - Understand the configuration (including installed applications) of the targeted systems. Various parts of the organization may need different configurations (e.g., software developers vs standard office workers).
  - Read the Impact section of the given Recommendation to help determine if there might be an issue with the targeted systems.
  - Test the configuration changes with representative lab system(s). If issues arise during testing, they can be resolved prior to deploying to any production systems.
  - When testing is complete, initially deploy to a small sub-set of production systems and monitor closely for issues. If there are issues, they can be resolved prior to deploying more broadly.
  - When the initial deployment above is completes successfully, iteratively deploy to additional systems and monitor closely for issues. Repeat this process until the full deployment is complete.

Summary

Using the Benchmarks Certified tools, working as a team with key stakeholders, being selective with exceptions, and being careful with remediation deployment, it is possible to harden large numbers of deployed systems in a cost effective, efficient, and safe manner.

NOTE: As previously stated, the PDF versions of the CIS Benchmarks™ are available for free, non-commercial use on the CIS Website. All other formats of the CIS Benchmarks™ (MS Word, Excel, and Build Kits) are available for CIS SecureSuite® members.

CIS-CAT® Pro is also available to CIS SecureSuite® members.

Target Technology Details

This document provides prescriptive guidance for running Google Kubernetes Engine (GKE) v1.31, 1.32, 1.33 & 1.34 following recommended security controls. This benchmark only includes controls which can be modified by an end user of GKE. 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.

To obtain the latest version of this guide, please visit www.cisecurity.org. 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
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, UI/Menu selections or examples. Text should be interpreted exactly as presented.
`<Monospace font in brackets>`	Text set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to reference other relevant settings, CIS Benchmarks and/or Benchmark Communities. Also, used to denote the title of a book, article, or other publication.
Bold font	Additional information or caveats things like **Notes**, **Warnings**, or **Cautions** (usually just the word itself and the rest of the text normal).

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**

  Items in this profile intend to:

  - be practical and prudent
  - provide a clear security benefit
  - not inhibit the utility of the technology beyond acceptable means

- **Level 2**

  Extends Level 1

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:

This benchmark was developed by Rowan Baker, Andrew Martin, and Kevin Ward, with input from Randall Mowen, Greg Castle, Andrew Kiggins, Iulia Ion, Jordan Liggitt, Maya Kaczorowski, Mark Wolters and members of the Google Compliance team.

With Special Thanks to the Google team of: Poonam Lamba, Michele Chubirka, Shannon Kularathana, Vinayak Goyal, Andrew Peabody and Padma Padmalatha.

Author/s
Andrew Martin
Rowan Baker
Kevin Ward

Editor/s
Randall Mowen
Poonam Lamba
Michele Chubirka
Shannon Kularathana
Vinayak Goyal

Contributor/s
Rory Mccune
Jordan Liggitt
Cliff Barbier
Maya Kaczorowski
Mark Wolters
Iulia Ion
Andrew Kiggins
Greg Castle
Mark Larinde
Andrew Thompson
Gareth Boyes

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

Overview 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.

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 kubeconfig file permissions are set to 644 or more restrictive (Automated)

Profile Applicability:

- Level 1

Description:

If `kubelet` 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 `kubelet` kubeconfig file controls various parameters of the `kubelet` 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 `kubelet` with the kubeconfig parameters configured as a Kubernetes ConfigMap instead of a file. In this case, there is no proxy kubeconfig file.

Impact:

Ensuring that the kubeconfig file permissions are set to 644 or more restrictive significantly strengthens the security posture of the Kubernetes environment by preventing unauthorized modifications. This restricts write access to the kubeconfig file, ensuring only administrators can alter crucial kubelet configurations, thereby reducing the risk of malicious alterations that could compromise the cluster's integrity.

However, this configuration may slightly impact usability, as it limits the ability for non-administrative users to make quick adjustments to the kubelet settings. Administrators will need to balance security needs with operational flexibility, potentially requiring adjustments to workflows for managing kubelet configurations.

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

Method 1

SSH to the worker nodes

To check to see if the kubelet service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate kubeconfig file:

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

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

Run this command to obtain the kubeconfig file permissions:

```
stat -c %a /var/lib/kubelet/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.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file permissions on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the permission level of the file:

```
ls -l /host/var/lib/kubelet/kubeconfig
```

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 <kubeconfig file>
```

Default Value:

See the GKE documentation for the default value.

References:

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

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists 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.	●	●	●
v8	13.4 Perform Traffic Filtering Between Network Segments Perform traffic filtering between network segments, where appropriate.		●	●
v8	13.6 Collect Network Traffic Flow Logs Collect network traffic flow logs and/or network traffic to review and alert upon from network devices.		●	●
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.		●	●

3.1.2 Ensure that the kubelet kubeconfig file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1

Description:

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

Rationale:

The kubeconfig file for `kubelet` controls various parameters for the `kubelet` 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:

Overly permissive file access increases the security risk to the platform.

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

Method 1

SSH to the worker nodes

To check to see if the kubelet service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

Run the following command on each node to find the appropriate kubeconfig file:

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

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

Run this command to obtain the kubeconfig file ownership:

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

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

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file ownership on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the ownership of the file:

```
ls -l /host/var/lib/kubelet/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://cloud.google.com/kubernetes-engine/docs/concepts/cis-benchmarks

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists 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 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.		●	●

3.1.3 Ensure that the kubelet configuration file has permissions set to 644 (Automated)

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet configuration file exists, it has permissions of 644.

Rationale:

The kubelet reads various parameters, including security settings, from a config file specified by the `--config` argument. If this file exists, 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:

Overly permissive file access increases the security risk to the platform.

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

Method 1

First, SSH to the relevant worker node:

To check to see if the kubelet service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

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 /etc/kubernetes/kubelet-config.yaml` which is the location of the Kubelet config file.

Run the following command:

```
stat -c %a /etc/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.

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file permissions on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the permission level of the file:

```
ls -l /host/etc/kubernetes/kubelet-config.yaml
```

Verify that if a file is specified and it exists, the permissions are `644` or more restrictive.

Remediation:

Run the following command (using the kubelet config file location):

```
chmod 644 <kubelet_config_file>
```

Default Value:

The default permissions for the kubelet configuration file are 600.

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 Configure Data Access Control Lists 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 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.		●	●

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

Profile Applicability:

- Level 1

Description:

Ensure that if the kubelet configuration file exists, it 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:

Overly permissive file access increases the security risk to the platform.

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

Method 1

First, SSH to the relevant worker node:

To check to see if the kubelet service is running:

```
sudo systemctl status kubelet
```

The output should return `Active: active (running) since..`

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 /etc/kubernetes/kubelet-config.yaml` which is the location of the kubelet config file.

Run the following command:

```
stat -c %U:%G /etc/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`

Method 2

Create and Run a Privileged Pod.

You will need to run a pod that is privileged enough to access the host's file system. This can be achieved by deploying a pod that uses the hostPath volume to mount the node's file system into the pod.

Here's an example of a simple pod definition that mounts the root of the host to /host within the pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: file-check
spec:
  volumes:
  - name: host-root
    hostPath:
      path: /
      type: Directory
  containers:
  - name: nsenter
    image: busybox
    command: ["sleep", "3600"]
    volumeMounts:
    - name: host-root
      mountPath: /host
    securityContext:
      privileged: true
```

Save this to a file (e.g., file-check-pod.yaml) and create the pod:

```
kubectl apply -f file-check-pod.yaml
```

Once the pod is running, you can exec into it to check file ownership on the node:

```
kubectl exec -it file-check -- sh
```

Now you are in a shell inside the pod, but you can access the node's file system through the /host directory and check the ownership of the file:

```
ls -l /etc/kubernetes/kubelet-config.yaml
```

The output of the above command gives you the 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 <kubelet_config_file>
```

Default Value:

The default file ownership is root:root.

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 Configure Data Access Control Lists 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 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 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.

Here’s a concise and effective CLI statement to list all principals (users, groups, or service accounts) bound to the cluster-admin role:

```
kubectl get clusterrolebinding -o jsonpath='{range
.items[?(@.roleRef.name=="cluster-admin")]}{.metadata.name}{"\n"}{range
.subjects[*]}{.kind}{"\t"}{.name}{"\n"}{end}{"\n"}{end}'
```

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/concepts/cluster-administration/
2. https://kubernetes.io/docs/reference/access-authn-authz/rbac/

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	4.3 Ensure the Use of Dedicated Administrative Accounts 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.

Below is a command to print which objects have get, list or watch granted for each matching role including roles that grant access via wildcards like resources: ["*","secrets/*"] or verbs: ["*"]:

```
kubectl get clusterrole,role -A -o json | jq -r '
  def wanted: ["get","list","watch"];
  .items[] as $r
  | [ $r.rules[]?
      | select(
          ((.apiGroups? // [""]) | any(.=="" or .=="*"))
          and ((.resources? // []) | any(.=="secrets" or .=="secrets/*" or
.=="*"))
          and ((.verbs? // []) | any(.=="*" or .=="get" or .=="list" or
.=="watch"))
        )
      | if ((.verbs? // []) | any(.=="*"))
        then wanted[] else (.verbs[]? | select(IN("get","list","watch"))) end
    ] as $verbs
  | select($verbs | length > 0)
  | "\($r.kind): \($r.metadata.name) (namespace: \($r.metadata.namespace //
"cluster-wide")) | verbs: \($verbs | unique | join(","))"
'
```

Sample output:

```
ClusterRole: admin (namespace: cluster-wide) | verbs: get,list,watch
ClusterRole: cluster-admin (namespace: cluster-wide) | verbs: get,list,watch
ClusterRole: edit (namespace: cluster-wide) | verbs: get,list,watch
ClusterRole: system:aggregate-to-edit (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:cloud-controller-manager (namespace: cluster-wide) |
verbs: get,list,watch
ClusterRole: system:controller:generic-garbage-collector (namespace: cluster-
wide) | verbs: get,list,watch
ClusterRole: system:controller:namespace-controller (namespace: cluster-wide)
| verbs: get,list,watch
ClusterRole: system:controller:resourcequota-controller (namespace: cluster-
wide) | verbs: list,watch
ClusterRole: system:gcp-controller-manager (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:gke-common-webhooks (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:glbc-status (namespace: cluster-wide) | verbs: get
ClusterRole: system:kube-controller-manager (namespace: cluster-wide) |
verbs: get,list,watch
ClusterRole: system:kubestore-collector (namespace: cluster-wide) | verbs:
get,list,watch
ClusterRole: system:node (namespace: cluster-wide) | verbs: get,list,watch
Role: operator (namespace: gmp-public) | verbs: get,list,watch
Role: operator (namespace: gmp-system) | verbs: get,list,watch
Role: system:controller:bootstrap-signer (namespace: kube-system) | verbs:
get,list,watch
Role: system:controller:token-cleaner (namespace: kube-system) | verbs:
get,list,watch
```

Remediation:

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

Default Value:

```
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 Establish and Maintain a Secure Configuration Process 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 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.		●	●

4.1.3 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

Here's a null-safe, column-formatted command that shows only Roles and ClusterRoles that use a wildcard (*) anywhere in verbs, resources, or apiGroups—and tells you which field(s) use the wildcard:

```
kubectl get clusterrole,role -A -o json | jq -r '
  def has_star(a): (a // []) | any(. == "*");

  .items[]
  | . as $r
  | ( any($r.rules[]?; has_star(.verbs)) )      as $wv
  | ( any($r.rules[]?; has_star(.resources)) )  as $wr
  | ( any($r.rules[]?; has_star(.apiGroups)) )  as $wg
  | select($wv or $wr or $wg)
  | [
      $r.kind,
      $r.metadata.name,
      ($r.metadata.namespace // "cluster-wide"),
      ([ if $wv then "verbs" else empty end,
         if $wr then "resources" else empty end,
         if $wg then "apiGroups" else empty end ] | join(","))
    ]
  | @tsv
' | awk -F'\t' 'BEGIN {
  OFS="\t";
  printf "%-15s %-40s %-20s %-20s\n", "KIND", "NAME", "NAMESPACE",
"WILDCARD_IN"
  print  "--------------- ---------------------------------------- ----------
---------- --------------------"
} {
  printf "%-15s %-40s %-20s %-20s\n", $1, $2, $3, $4
}'
```

Sample Output from command:

```
KIND            NAME                                     NAMESPACE
WILDCARD_IN
--------------- ---------------------------------------- --------------------
--------------------
ClusterRole     cluster-admin                            cluster-wide
verbs,resources,apiGroups
ClusterRole     external-metrics-reader                  cluster-wide
resources
ClusterRole     kubelet-api-admin                        cluster-wide
verbs
ClusterRole     system:cloud-controller-manager          cluster-wide
resources,apiGroups
ClusterRole     system:controller:disruption-controller  cluster-wide
apiGroups
ClusterRole     system:controller:generic-garbage-collector cluster-wide
resources,apiGroups
ClusterRole     system:controller:horizontal-pod-autoscaler cluster-wide
resources,apiGroups
ClusterRole     system:controller:namespace-controller   cluster-wide
resources,apiGroups
ClusterRole     system:controller:resourcequota-controller cluster-wide
resources,apiGroups
ClusterRole     system:gcp-controller-manager            cluster-wide
verbs
ClusterRole     system:gke-common-webhooks               cluster-wide
verbs,resources,apiGroups
ClusterRole     system:gke-hpa-actor                     cluster-wide
resources,apiGroups
ClusterRole     system:glbc-status                       cluster-wide
verbs
ClusterRole     system:kube-controller-manager           cluster-wide
resources,apiGroups
ClusterRole     system:kubelet-api-admin                 cluster-wide
verbs
ClusterRole     system:kubestore-collector               cluster-wide
verbs,resources,apiGroups
ClusterRole     system:managed-certificate-controller    cluster-wide
verbs
ClusterRole     system:metrics-server-nanny              cluster-wide
verbs
Role            gke-spiffe-leaderelection                kube-system
verbs
```

Remediation:

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

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/rbac/

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

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.

```
kubectl get serviceaccounts/default -o yaml
```

Sample Output:

```
apiVersion: v1
automountServiceAccountToken: false
kind: ServiceAccount
metadata:
  creationTimestamp: "2025-10-18T16:11:09Z"
  name: default
  namespace: default
...
```

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 Disable Dormant Accounts Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts 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 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	16.9 Disable Dormant Accounts Automatically disable dormant accounts after a set period of inactivity.	●	●	●

4.1.5 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
echo "=== ServiceAccounts with automountServiceAccountToken=false ==="
printf "%-25s %-40s\n" "NAMESPACE" "SERVICEACCOUNT"
echo "------------------------- ----------------------------------------"
kubectl get sa -A -o jsonpath='{range
.items[?(@.automountServiceAccountToken==false)]}{.metadata.namespace}{"\t"}{
.metadata.name}{"\n"}{end}' \
| sort | awk -F'\t' '{printf "%-25s %-40s\n", $1, $2}'

echo ""
echo "=== Pods with automountServiceAccountToken=false ==="
printf "%-25s %-40s\n" "NAMESPACE" "POD"
echo "------------------------- ----------------------------------------"
kubectl get pods -A -o jsonpath='{range
.items[?(@.spec.automountServiceAccountToken==false)]}{.metadata.namespace}{"
\t"}{.metadata.name}{"\n"}{end}' \
| sort | awk -F'\t' '{printf "%-25s %-40s\n", $1, $2}'
```

Note:

- kubectl get ... -A scans all namespaces.
- jsonpath filters for `automountServiceAccountToken == false`.
- awk formats the output into aligned columns.

- If a section is empty, no objects explicitly set `automountServiceAccountToken: false` — meaning they still auto-mount tokens by default, and may need remediation.

Sample Output:

```
=== ServiceAccounts with automountServiceAccountToken=false ===
NAMESPACE                 SERVICEACCOUNT
------------------------- ----------------------------------------
default                   default
kube-system               metrics-reader

=== Pods with automountServiceAccountToken=false ===
NAMESPACE                 POD
------------------------- ----------------------------------------
dev                       api-service
prod                      nginx-test
```

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	4.8 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	14.7 Enforce Access Control to Data through Automated Tools 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.			●

4.1.6 Avoid use of system:masters group (Automated)

Profile Applicability:

- Level 1

Description:

The special group `system:masters` should not be used to grant permissions to any user or service account, except where strictly necessary (e.g. bootstrapping access prior to RBAC being fully available)

Rationale:

The `system:masters` group has unrestricted access to the Kubernetes API hard-coded into the API server source code. An authenticated user who is a member of this group cannot have their access reduced, even if all bindings and cluster role bindings which mention it, are removed.

When combined with client certificate authentication, use of this group can allow for irrevocable cluster-admin level credentials to exist for a cluster.

GKE includes the `CertificateSubjectRestriction` admission controller which rejects requests for the `system:masters` group.

`CertificateSubjectRestriction` "This admission controller observes creation of `CertificateSigningRequest` resources that have a spec.signerName of kubernetes.io/kube-apiserver-client. It rejects any request that specifies a 'group' (or 'organization attribute') of system:masters." https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#certificatesubjectrestriction

Impact:

Once the RBAC system is operational in a cluster `system:masters` should not be specifically required, as ordinary bindings from principals to the `cluster-admin` cluster role can be made where unrestricted access is required.

Audit:

Review a list of all credentials which have access to the cluster and ensure that the group `system:masters` is not used.

```
kubectl get clusterrolebinding -o json | jq -r '
  .items[] as $b
  | select([ $b.subjects[]? | select(.kind=="Group" and
.name=="system:masters") ] | length > 0)
  | ($b.subjects[]? | select(.kind=="Group" and .name=="system:masters"))
  | "\(.kind)\t\(.name)\t(bound in: \($b.metadata.name))"
'
```

Sample Output:

```
Group   system:masters  (bound in: cluster-admin)
```

Remediation:

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

Default Value:

By default some clusters will create a "break glass" client certificate which is a member of this group. Access to this client certificate should be carefully controlled and it should not be used for general cluster operations.

References:

1. https://github.com/kubernetes/kubernetes/blob/master/pkg/registry/rbac/escalation_check.go#L38

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 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

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

Profile Applicability:

- Level 1

Description:

Cluster roles and roles with the impersonate, bind or escalate permissions should not be granted unless strictly required. Each of these permissions allow a particular subject to escalate their privileges beyond those explicitly granted by cluster administrators

Rationale:

The impersonate privilege allows a subject to impersonate other users gaining their rights to the cluster. The bind privilege allows the subject to add a binding to a cluster role or role which escalates their effective permissions in the cluster. The escalate privilege allows a subject to modify cluster roles to which they are bound, increasing their rights to that level.

Each of these permissions has the potential to allow for privilege escalation to cluster-admin level.

Impact:

There are some cases where these permissions are required for cluster service operation, and care should be taken before removing these permissions from system service accounts.

Audit:

Review the users who have access to cluster roles or roles which provide the impersonate, bind or escalate privileges.

```
kubectl get clusterrole,role -A -o json | jq -r '
  def risky_verbs: ["bind","impersonate","escalate"];

  .items[]
  | . as $r
  | [ $r.rules[]?
      | select(
          (.verbs // [] | any(. as $v | $v == "bind" or $v == "impersonate"
or $v == "escalate"))
        )
      | {
          resources: ((.resources // ["*"]) | join(",")),
          verbs: ((.verbs // []) |
map(select(IN("bind","impersonate","escalate"))) | join(","))
        }
    ] as $matches
  | select($matches | length > 0)
  | [
      $r.kind,
      $r.metadata.name,
      ($r.metadata.namespace // "cluster-wide"),
      ($matches | map(.resources) | unique | join(",")),
      ($matches | map(.verbs) | unique | join(","))
    ] | @tsv
' | awk -F'\t' 'BEGIN{
  printf "%-15s %-40s %-20s %-40s %-
30s\n","KIND","NAME","NAMESPACE","RESOURCES","VERBS"
  print  "--------------- ---------------------------------------- ----------
---------- ---------------------------------------- -------------------------
-----"
}{
  printf "%-15s %-40s %-20s %-40s %-30s\n",$1,$2,$3,$4,$5
}'
```

Sample Output:

```
KIND            NAME                                     NAMESPACE
RESOURCES                                VERBS
--------------- ---------------------------------------- --------------------
---------------------------------------- ------------------------------
ClusterRole     admin                                    cluster-wide
serviceaccounts                          impersonate
ClusterRole     edit                                     cluster-wide
serviceaccounts                          impersonate
ClusterRole     cluster-admin                            cluster-wide
roles,clusterroles,users                 bind,impersonate,escalate
ClusterRole     custom-admin                             cluster-wide
rolebindings,clusterrolebindings         bind
Role            tenant-admin                             dev
rolebindings                             bind
```

Remediation:

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

Default Value:

In a default kubeadm cluster, the system:masters group and clusterrole-aggregation-controller service account have access to the escalate privilege. The system:masters group also has access to bind and impersonate.

References:

1. https://www.impidio.com/blog/kubernetes-rbac-security-pitfalls
2. https://raesene.github.io/blog/2020/12/12/Escalating_Away/
3. https://raesene.github.io/blog/2021/01/16/Getting-Into-A-Bind-with-Kubernetes/

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 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			

4.1.8 Avoid bindings to system:anonymous (Automated)

Profile Applicability:

- Level 2

Description:

Avoid ClusterRoleBindings or RoleBindings with the user `system:anonymous`.

Rationale:

Kubernetes assigns user `system:anonymous` to API server requests that have no authentication information provided. Binding a role to user `system:anonymous` gives any unauthenticated user the permissions granted by that role and is strongly discouraged.

Impact:

Unauthenticated users will have privileges and permissions associated with roles associated with the configured bindings.

Care should be taken before removing any `clusterrolebindings` or `rolebindings` from the environment to ensure they were not required for operation of the cluster. Use a more specific and authenticated user for cluster operations.

Audit:

Both CusterRoleBindings and RoleBindings should be audited. Use the following command to confirm there are no `ClusterRoleBindings` to `system:anonymous`:

```
$ kubectl get clusterrolebindings -o json   | jq -r '["Name"], ["-----"],
(.items[] | select((.subjects | length) > 0) | select(any(.subjects[]; .name
== "system:anonymous")) | [.metadata.namespace, .metadata.name]) | @tsv'
```

There should be no `ClusterRoleBindings` listed. If any bindings exist, review their permissions with the following command and reassess their privilege.

```
$ kubectl get clusterrolebinding [CLUSTER_ROLE_BINDING_NAME] -o json \
    | jq ' .roleRef.name +" " + .roleRef.kind' \
    | sed -e 's/"//g' \
    | xargs -l bash -c 'kubectl get $1 $0 -o yaml'
```

Confirm that there are no `RoleBindings` including the `system:anonymous` user:

```
$ kubectl get rolebindings -A -o json | jq -r '["Namespace", "Name"], ["-----
----", "-----"], (.items[] | select((.subjects | length) > 0) |
select(any(.subjects[]; .name == "system:anonymous")) | [.metadata.namespace,
.metadata.name]) | @tsv'
```

There should be no `RoleBindings` listed.

If any bindings exist, review their permissions with the following command and reassess their privilege.

```
$ kubectl get rolebinding [ROLE_BINDING_NAME] --namespace
[ROLE_BINDING_NAMESPACE] -o json \
    | jq ' .roleRef.name +" " + .roleRef.kind' \
    | sed -e 's/"//g' \
    | xargs -l bash -c 'kubectl get $1 $0 -o yaml --namespace
[ROLE_BINDING_NAMESPACE]'
```

Remediation:

Identify all `clusterrolebindings` and `rolebindings` to the user system:anonymous. Check if they are used and review the permissions associated with the binding using the commands in the Audit section above or refer to GKE documentation.

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

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

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

Default Value:

No `clusterrolebindings` nor `rolebindings` with user `system:anonymous`.

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/rbac/#discovery-roles

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.5 Establish and Maintain an Inventory of Service Accounts Establish and maintain an inventory of service accounts. The inventory, at a minimum, must contain department owner, review date, and purpose. Perform service account reviews to validate that all active accounts are authorized, on a recurring schedule at a minimum quarterly, or more frequently.		●	●
v7	16.8 Disable Any Unassociated Accounts Disable any account that cannot be associated with a business process or business owner.	●	●	●

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

Profile Applicability:

- Level 1

Description:

Avoid non-default `ClusterRoleBindings` and `RoleBindings` with the group `system:unauthenticated`, except the `ClusterRoleBinding` `system:public-info-viewer`.

Rationale:

Kubernetes assigns the group `system:unauthenticated` to API server requests that have no authentication information provided. Binding a role to this group gives any unauthenticated user the permissions granted by that role and is strongly discouraged.

Impact:

Unauthenticated users will have privileges and permissions associated with roles associated with the configured bindings.

Care should be taken before removing any non-default `clusterrolebindings` or `rolebindings` from the environment to ensure they were not required for operation of the cluster. Leverage a more specific and authenticated user for cluster operations.

Audit:

Both `CusterRoleBindings` and `RoleBindings` should be audited. Use the following command to confirm there are no non-default `ClusterRoleBindings` to group `system:unauthenticated`:

```
$ kubectl get clusterrolebindings -o json   | jq -r '["Name"], ["-----"],
(.items[] | select((.subjects | length) > 0) | select(any(.subjects[]; .name
== "system:unauthenticated")) | [.metadata.namespace, .metadata.name]) |
@tsv'
```

Only the following default `ClusterRoleBinding` should be displayed:

```
Name
-----
        system:public-info-viewer
```

If any non-default bindings exist, review their permissions with the following command and reassess their privilege.

```
$ kubectl get clusterrolebinding [CLUSTER_ROLE_BINDING_NAME] -o json \
    | jq ' .roleRef.name +" " + .roleRef.kind' \
    | sed -e 's/"//g' \
    | xargs -l bash -c 'kubectl get $1 $0 -o yaml'
```

Confirm that there are no RoleBindings including the `system:unauthenticated` group:

```
$ kubectl get rolebindings -A -o json | jq -r '["Namespace", "Name"], ["-----
----", "-----"], (.items[] | select((.subjects | length) > 0) |
select(any(.subjects[]; .name == "system:unauthenticated")) |
[.metadata.namespace, .metadata.name]) | @tsv'
```

There should be no `RoleBindings` listed.

If any bindings exist, review their permissions with the following command and reassess their privilege.

```
$ kubectl get rolebinding [ROLE_BINDING_NAME] --namespace
[ROLE_BINDING_NAMESPACE] -o json \
    | jq ' .roleRef.name +" " + .roleRef.kind' \
    | sed -e 's/"//g' \
    | xargs -l bash -c 'kubectl get $1 $0 -o yaml --namespace
[ROLE_BINDING_NAMESPACE]'
```

Remediation:

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

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

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

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

Default Value:

`ClusterRoleBindings` with group s`ystem:unauthenticated`:

- `system:public-info-viewer`

No `RoleBindings` with the group `system:unauthenticated`.

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/rbac/#discovery-roles

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.5 Establish and Maintain an Inventory of Service Accounts Establish and maintain an inventory of service accounts. The inventory, at a minimum, must contain department owner, review date, and purpose. Perform service account reviews to validate that all active accounts are authorized, on a recurring schedule at a minimum quarterly, or more frequently.		●	●
v7	16.8 Disable Any Unassociated Accounts Disable any account that cannot be associated with a business process or business owner.	●	●	●

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

Profile Applicability:

- Level 1

Description:

Avoid non-default `ClusterRoleBindings` and `RoleBindings` with the group `system:authenticated`, except the `ClusterRoleBindings` `system:basic-user`, `system:discovery`, and `system:public-info-viewer`.

Google's approach to authentication is to make authenticating to Google Cloud and GKE as simple and secure as possible without adding complex configuration steps. The group `system:authenticated` includes all users with a Google account, which includes all Gmail accounts. Consider your authorization controls with this extended group scope when granting permissions. Thus, group `system:authenticated` is not recommended for non-default use.

Rationale:

GKE assigns the group `system:authenticated` to API server requests made by any user who is signed in with a Google Account, including all Gmail accounts. In practice, this isn't meaningfully different from `system:unauthenticated` because anyone can create a Google Account.

Binding a role to the group `system:authenticated` gives any user with a Google Account, including all Gmail accounts, the permissions granted by that role and is strongly discouraged.

Impact:

Authenticated users in group `system:authenticated` should be treated similarly to users in `system:unauthenticated`, having privileges and permissions associated with roles associated with the configured bindings.

Care should be taken before removing any non-default `clusterrolebindings` or `rolebindings` from the environment to ensure they were not required for operation of the cluster. Leverage a more specific and authenticated user for cluster operations.

Audit:

Use the following command to confirm there are no non-default `ClusterRoleBindings` to `system:authenticated`:

```
$ kubectl get clusterrolebindings -o json | jq -r '["Name"], ["-----"],
(.items[] | select((.subjects | length) > 0) | select(any(.subjects[]; .name
== "system:unauthenticated")) | [.metadata.namespace, .metadata.name]) |
@tsv'
```

Only one or more of the following default `ClusterRoleBindings` should be displayed:

```
Name
-----
        system:basic-user
        system:discovery
        system:public-info-viewer
```

If any non-default bindings exist, review their permissions with the following command and reassess their privilege.

```
$ kubectl get clusterrolebinding [CLUSTER_ROLE_BINDING_NAME] -o json \
    | jq ' .roleRef.name +" " + .roleRef.kind' \
    | sed -e 's/"//g' \
    | xargs -l bash -c 'kubectl get $1 $0 -o yaml'
```

Confirm that there are no `RoleBindings` including the `system:authenticated` group:

```
$ kubectl get rolebindings -A -o json | jq -r '["Namespace", "Name"], ["-----
----", "-----"], (.items[] | select((.subjects | length) > 0) |
select(any(.subjects[]; .name == "system:unauthenticated")) |
[.metadata.namespace, .metadata.name]) | @tsv'
```

There should be no `RoleBindings` listed.

If any bindings exist, review their permissions with the following command and reassess their privilege.

```
$ kubectl get rolebinding [ROLE_BINDING_NAME] --namespace
[ROLE_BINDING_NAMESPACE] -o json \
    | jq ' .roleRef.name +" " + .roleRef.kind' \
    | sed -e 's/"//g' \
    | xargs -l bash -c 'kubectl get $1 $0 -o yaml --namespace
[ROLE_BINDING_NAMESPACE]'
```

Remediation:

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

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

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

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

Default Value:

`ClusterRoleBindings` with group `system:authenticated`:

- `system:basic-user`
- `system:discovery`

No `RoleBindings` with the group `system:authenticated`.

References:

1. https://kubernetes.io/docs/reference/access-authn-authz/rbac/#discovery-roles

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	5.5 Establish and Maintain an Inventory of Service Accounts Establish and maintain an inventory of service accounts. The inventory, at a minimum, must contain department owner, review date, and purpose. Perform service account reviews to validate that all active accounts are authorized, on a recurring schedule at a minimum quarterly, or more frequently.		●	●
v7	16.8 Disable Any Unassociated Accounts Disable any account that cannot be associated with a business process or business owner.	●	●	●

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.

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

Profile Applicability:

- Level 1

Description:

The Pod Security Standard Baseline profile defines a baseline for container security. You can enforce this by using the built-in Pod Security Admission controller.

Rationale:

Without an active mechanism to enforce the Pod Security Standard Baseline profile, it is not possible to limit the use of containers with access to underlying cluster nodes, via mechanisms like privileged containers, or the use of hostPath volume mounts.

Impact:

Enforcing a baseline profile will limit the use of containers.

Audit:

Run the following command to list the namespaces that don't have the the baseline policy enforced.

```
diff \
<(kubectl get namespace -l pod-security.kubernetes.io/enforce=baseline -o
jsonpath='{range .items[*]}{.metadata.name}{"\n"}') \
<(kubectl get namespace -o jsonpath='{range
.items[*]}{.metadata.name}{"\n"}')
```

Remediation:

Ensure that Pod Security Admission is in place for every namespace which contains user workloads.

Run the following command to enforce the Baseline profile in a namespace:

```
kubectl label namespace <namespace-name> pod-
security.kubernetes.io/enforce=baseline
```

Default Value:

By default, Pod Security Admission is enabled but no policies are in place.

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.		●	●

4.3 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 5.6.7.

Impact:

None

Audit:

Review the documentation of CNI plugin in use by the cluster, and confirm that it supports Ingress and Egress 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 5.6.7.

Default Value:

This will depend on the CNI plugin in use.

References:

1. https://kubernetes.io/docs/concepts/services-networking/network-policies/
2. https://kubernetes.io/docs/concepts/extend-kubernetes/compute-storage-net/network-plugins/
3. https://cloud.google.com/kubernetes-engine/docs/concepts/network-overview

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	16.5 Use Up-to-Date and Trusted Third-Party Software Components 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	18.4 Only Use Up-to-date And Trusted Third-Party Components 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 (Automated)

Profile Applicability:

- Level 2

Description:

Use network policies to isolate traffic in the 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 commands and review the `NetworkPolicy` objects created in the cluster.

Command line statement to show namespaces with 1 or more Network Policies set:

```
kubectl get ns -o jsonpath='{range .items[*]}{.metadata.name}{"\n"}{end}' |
while read ns; do
  count=$(kubectl get networkpolicy -n "$ns" --no-headers 2>/dev/null | wc -l
| tr -d ' ')
  if [ "$count" -gt 0 ]; then
    echo -e "${ns}\t${count}"
  fi
done
```

Command line statement to show namespaces with 0 Network Policies set:

```
for ns in $(kubectl get ns -o jsonpath='{.items[*].metadata.name}'); do
  count=$(kubectl get networkpolicy -n "$ns" --no-headers 2>/dev/null | wc -l
| tr -d ' ')
  if [ "$count" -eq 0 ]; then
    echo -e "${ns}\t${count}"
  fi
done
```

Remediation:

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

Default Value:

By default, network policies are not created.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/network-policy#creating_a_network_policy
2. https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
3. https://cloud.google.com/kubernetes-engine/docs/concepts/network-overview

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.4 Perform Traffic Filtering Between Network Segments Perform traffic filtering between network segments, where appropriate.		●	●
v7	14.1 Segment the Network Based on Sensitivity 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 Enable Firewall Filtering Between VLANs 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 (Automated)

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.

```
echo -e "NAMESPACE\tPOD\tCONTAINER"
kubectl get pods -A -o jsonpath='{range
.items[?(@.spec.containers[*].env[*].valueFrom.secretKeyRef)]}{.metadata.name
space}{"\t"}{.metadata.name}{"\t"}{.spec.containers[*].name}{"\n"}{end}'
```

Sample Output:

```
NAMESPACE      POD                          CONTAINER
default         my-app-588f67fd8d-vqc6k          my-app-container
prod             api-6d8b4f79cc-def34     api-container
```

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	3 Data Protection Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.			
v7	13 Data Protection Data Protection			

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 more complex secret management is required. 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 secrests 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 the cloud service provider or a third-party secrets management solution.

Default Value:

By default, no external secret management is configured.

References:

1. https://kubernetes.io/docs/concepts/configuration/secret/
2. https://cloud.google.com/secret-manager/docs/overview

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3 Data Protection Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.			
v7	13 Data Protection Data Protection			

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 the deployment.

Rationale:

Kubernetes supports plugging in provenance rules to accept or reject the images in deployments. Rules can be configured to ensure that only approved images are deployed in the cluster.

Also see recommendation 5.1.4.

Impact:

Regular maintenance for the provenance configuration should be carried out, based on container image updates.

Audit:

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

Also see recommendation 5.1.4.

Remediation:

Follow the Kubernetes documentation and setup image provenance.

Also see recommendation 5.1.4.

Default Value:

By default, image provenance is not set.

References:

1. https://kubernetes.io/docs/concepts/containers/images/
2. https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/

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 Application Software Security 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 four 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
3. `kube-node-lease` - Namespace used for node heartbeats
4. `kube-public` - Namespace used for public information in a cluster

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13 Network Monitoring and Defense Operate processes and tooling to establish and maintain comprehensive network monitoring and defense against security threats across the enterprise's network infrastructure and user base.			
v7	12 Boundary Defense Boundary Defense			

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

Profile Applicability:

- Level 2

Description:

Enable `RuntimeDefault` seccomp profile in the 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. It should be enabled to ensure that the workloads have restricted actions available within the container.

Impact:

If the `RuntimeDefault` seccomp profile is too restrictive for you, you would have to create/manage your own `Localhost` seccomp profiles.

Audit:

Review the pod definitions output for all namespaces in the cluster with the command below.

```
kubectl get pods --all-namespaces -o json | jq -r '.items[] |
select(.metadata.annotations."seccomp.security.alpha.kubernetes.io/pod" ==
"runtime/default" or .spec.securityContext.seccompProfile.type ==
"RuntimeDefault") | {namespace: .metadata.namespace, name: .metadata.name,
seccompProfile: .spec.securityContext.seccompProfile.type}'
```

Remediation:

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

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

Default Value:

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

References:

1. https://kubernetes.io/docs/tutorials/security/seccomp/
2. https://cloud.google.com/kubernetes-engine/docs/concepts/seccomp-in-gke

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	5.2 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 Pods and Containers (Manual)

Profile Applicability:

- Level 2

Description:

Apply Security Context to 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 containers and pods, make sure that the security context is configured for 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, there may be issues running the pods.

Audit:

Review the pod definitions in the cluster and verify that the security contexts have been 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 Google Container-Optimized OS Benchmark.

Default Value:

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

References:

1. https://kubernetes.io/docs/concepts/workloads/pods/
2. https://kubernetes.io/docs/concepts/containers/
3. https://kubernetes.io/docs/tasks/configure-pod-container/security-context/
4. https://learn.cisecurity.org/benchmarks

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4 <u>Secure Configuration of Enterprise Assets and Software</u> Establish and maintain the secure configuration of enterprise assets (end-user devices, including portable and mobile; network devices; non-computing/IoT devices; and servers) and software (operating systems and applications).			
v7	5.1 <u>Establish Secure Configurations</u> Maintain documented, standard security configuration standards for all authorized operating systems and software.	●	●	●

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:

To list all Kubernetes objects in the default namespace of your GKE cluster, use:

```
kubectl get all -n default

or

kubectl get all -n default -o wide
```

If you want a complete inventory of all resource types in the default namespace:

```
kubectl get
all,configmaps,secrets,pvc,ingress,serviceaccounts,networkpolicies -n default
```

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	12.2 Establish and Maintain a Secure Network Architecture Establish and maintain a secure network architecture. A secure network architecture must address segmentation, least privilege, and availability, at a minimum.		●	●
v7	2.10 Physically or Logically Segregate High Risk Applications Physically or logically segregated systems should be used to isolate and run software that is required for business operations but incur higher risk for the organization.			●

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 is enabled (Automated)

Profile Applicability:

- Level 2

Description:

Note: GCR is now deprecated, being superseded by Artifact Registry starting 15th May 2024. Runtime Vulnerability scanning is available via GKE Security Posture

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

Rationale:

Vulnerabilities in software packages can be exploited by malicious users to obtain unauthorized access to local cloud resources. GCR Container Analysis API or Artifact Registry Container Scanning API allow images stored in GCR or AR respectively to be scanned for known vulnerabilities.

Impact:

None.

Audit:

For Images Hosted in GCR:

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
```

Ensure that the `Container Registry API` and `Container Analysis API` are listed in the output.

For Images Hosted in AR:

Using Google Cloud Console:

1. Go to AR by visiting https://console.cloud.google.com/artifacts
2. Select Settings and check if `Vulnerability scanning` is Enabled.

Using Command Line:

```
gcloud services list --enabled
```

Ensure that `Container Scanning API` and `Artifact Registry API` are listed in the output.

Remediation:

For Images Hosted in GCR:

Using Google Cloud Console

1. Go to GCR by visiting: https://console.cloud.google.com/gcr
2. Select Settings and, under the Vulnerability Scanning heading, click the TURN ON button.

Using Command Line

```
gcloud services enable containeranalysis.googleapis.com
```

For Images Hosted in AR:

Using Google Cloud Console

1. Go to GCR by visiting: https://console.cloud.google.com/artifacts
2. Select Settings and, under the Vulnerability Scanning heading, click the ENABLE button.

Using Command Line

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

Default Value:

By default, GCR Container Analysis and AR Container Scanning are disabled.

References:

1. https://cloud.google.com/artifact-registry/docs/analysis
2. https://cloud.google.com/artifact-analysis/docs/os-overview
3. https://console.cloud.google.com/marketplace/product/google/containerregistry.googleapis.com
4. https://cloud.google.com/kubernetes-engine/docs/concepts/about-configuration-scanning
5. https://containersecurity.googleapis.com

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets 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	3 Continuous Vulnerability Management Continuous Vulnerability Management			
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.		●	●
v7	3.2 Perform Authenticated Vulnerability Scanning 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 Container Image repositories (Manual)

Profile Applicability:

- Level 2

Description:

Note: GCR is now deprecated, see the references for more details.

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

Rationale:

Weak access control to GCR or AR may allow malicious users to replace built images with vulnerable or back-doored containers.

Impact:

Care should be taken not to remove access to GCR or AR 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:

For Images Hosted in AR:

1. Go to Artifacts Browser by visiting https://console.cloud.google.com/artifacts
2. From the list of artifacts select each repository with format `Docker`
3. Under the Permissions tab, review the roles for each member and ensure only authorized users have the Artifact Registry Administrator, Artifact Registry Reader, Artifact Registry Repository Administrator and Artifact Registry Writer roles.

Users may have permissions to use Service Accounts and thus Users could inherit privileges on the AR repositories. 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 AR repositories. Consult the GCP CIS benchmark and IAM documentation for further reference.

Using Command Line:

```
gcloud artifacts repositories get-iam-policy <repository-name> --location
<repository-location>
```

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

For Images Hosted in GCR:

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:

For Images Hosted in AR:

Using Google Cloud Console:

1. Go to Artifacts Browser by visiting https://console.cloud.google.com/artifacts
2. From the list of artifacts select each repository with format `Docker`
3. Under the Permissions tab, modify the roles for each member and ensure only authorized users have the Artifact Registry Administrator, Artifact Registry Reader, Artifact Registry Repository Administrator and Artifact Registry Writer roles.

Using Command Line:

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

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

For Images Hosted in GCR:

Using 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/
2. https://cloud.google.com/kubernetes-engine/docs/how-to/service-accounts
3. https://cloud.google.com/kubernetes-engine/docs/how-to/iam
4. https://cloud.google.com/artifact-registry/docs/access-control#grant

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists 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 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.	●	●	●

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

Profile Applicability:

- Level 2

Description:

Note: GCR is now deprecated, see the references for more details.

Configure the Cluster Service Account with Artifact Registry Viewer Role to only allow read-only access to AR repositories. 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 or AR, 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:

For Images Hosted in AR:

Using Google Cloud Console

1. Go to Artifacts Browser by visiting https://console.cloud.google.com/artifacts
2. From the list of repositories, for each repository with Format Docker
3. Under the Permissions tab, review the role for GKE Service account and ensure that only the Artifact Registry Viewer role is set.

Using Command Line:

```
gcloud artifacts repositories get-iam-policy <repository-name> --location
<repository-location>
```

The output of the command will return roles associated with the AR repository. If listed, ensure the GKE Service account is set to `"role": "roles/artifactregistry.reader"`.

For Images Hosted in GCR:

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:

For Images Hosted in AR:

Using Google Cloud Console:

1. Go to Artifacts Browser by visiting https://console.cloud.google.com/artifacts
2. From the list of repositories, for each repository with Format Docker
3. Under the Permissions tab, modify the permissions for GKE Service account and ensure that only the Artifact Registry Viewer role is set.

Using Command Line: Add artifactregistry.reader role

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

Remove any roles other than `artifactregistry.reader`

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

For Images Hosted in GCR:

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/
2. https://cloud.google.com/kubernetes-engine/docs/how-to/service-accounts
3. https://cloud.google.com/kubernetes-engine/docs/how-to/iam

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists 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 Perform Authenticated Vulnerability Scanning 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 Ensure only trusted container images are used (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. Ensuring only trusted container images are used reduces this risk.

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: 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. Select the cluster for which Binary Authorization is disabled.
3. Under the details pane, within the Security section, ensure that 'Binary Authorization' is set to 'Enabled'. Then, assess the contents of the policy:
4. Go to Binary Authorization by visiting: https://console.cloud.google.com/security/binary-authorization
5. Ensure a policy is defined and the project default rule is not configured to 'Allow all images'.

Using Command Line:

Check that Binary Authorization is enabled for the GKE cluster, first define 3 variables for Cluster Name, Location and Project and then run the command below:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq .binaryAuthorization
```

This will return one of 2 possible responses if Binary Authorization is enabled:

```
{
  "evaluationMode": "PROJECT_SINGLETON_POLICY_ENFORCE"
}

OR

{
  "evaluationMode": "PROJECT_SINGLETON_POLICY_AUDIT_AND_WARN"
}
```

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. Create an appropriate policy for use with the cluster. See https://cloud.google.com/binary-authorization/docs/policy-yaml-reference for guidance.
4. Go to Kubernetes Engine by visiting: https://console.cloud.google.com/kubernetes/list.
5. Select the cluster for which Binary Authorization is disabled.
6. Under the details pane, within the Security section, click on the pencil icon named `Edit Binary Authorization`.
7. Check the box next to `Enable Binary Authorization`.
8. Choose `Enforce` policy and provide a directory for the policy to be used.
9. Click `SAVE CHANGES`.

Using Command Line:

Update the cluster to enable Binary Authorization:

```
gcloud container cluster update <cluster_name> --location <location> --
project <project_id> --binauthz-evaluation-mode=<evaluation_mode>

Example:
gcloud container clusters update $CLUSTER_NAME --location $LOCATION --project
$PROJECT_ID --binauthz-evaluation-mode=PROJECT_SINGLETON_POLICY_ENFORCE
```

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

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

Import the policy file into Binary Authorization:

```
gcloud container binauthz policy import <yaml_policy>
```

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
2. https://cloud.google.com/binary-authorization/docs/setting-up

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 be used to run GCP API requests. 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.

A minimally privileged service account should be created and used to run the Kubernetes Engine cluster instead of using the Compute Engine default service account, and create separate service accounts for each Kubernetes Workload (See recommendation 5.2.2).

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 5.2.2.

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:

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 an existing cluster, first define 4 variables for Node Pool, Cluster Name, Location and Project, and then run the following command:

```
gcloud container node-pools describe $NODE_POOL --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --format json | jq
'.config.serviceAccount'
```

The output of the above command will return the specific Service account being 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 AND CONTINUE`.
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 details, select the `Security` subheading, and under `Identity defaults, select the minimally privileged service account from the Service Account drop-down.
5. Click `CREATE to launch the Node pool.

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

Using Command Line:

To create a minimally privileged service account:

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

Grant the following roles to the service account:

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

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

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

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

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
v8	4.7 Manage Default Accounts on Enterprise Assets and Software Manage default accounts on enterprise assets and software, such as root, administrator, and other pre-configured vendor accounts. Example implementations can include: disabling default accounts or making them unusable.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts 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 2

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:

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, the Compute Engine default Service account can not be used. Correspondingly, Workload Identity can't be used with Pods running in the host network. Workloads may also need to be modified in order for them to use Workload Identity, as described within: https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity

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, for example: `<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'.

To check an existing cluster, first define 3 variables for Cluster Name, Location and Project, and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq .workloadIdentityConfig
```

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:
  workloadPool:<project_id>.svc.id.goog
```

For each Node pool, ensure the following is set.

```
workloadMetadataConfig:
    nodeMetadata: GKE_METADATA_SERVER
```

Each Kubernetes workload requiring Google Cloud API access will need to be manually audited 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 the cluster for which Workload Identity is disabled.
3. Within the `Details` pane, under the `Security` section, click on the pencil icon named `Edit workload identity`.
4. Enable Workload Identity and set the workload pool to the namespace of the Cloud project containing the cluster, for example: `<project_id>.svc.id.goog`.
5. Click `SAVE CHANGES` 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 container clusters update <cluster_name> --location <location> --
workload-pool <project_id>.svc.id.goog
```

Note that existing Node pools are unaffected. New Node pools default to `--workload-metadata-from-node=GKE_METADATA_SERVER`. Then, modify existing Node pools to enable `GKE_METADATA_SERVER`:

```
gcloud container node-pools update <node_pool_name> --cluster <cluster_name>
--location <location> --workload-metadata=GKE_METADATA
```

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

Default Value:

By default, Workload Identity is disabled.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity
2. https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-architecture
3. https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	4.7 Manage Default Accounts on Enterprise Assets and Software Manage default accounts on enterprise assets and software, such as root, administrator, and other pre-configured vendor accounts. Example implementations can include: disabling default accounts or making them unusable.	●	●	●
v7	4.3 Ensure the Use of Dedicated Administrative Accounts 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 2

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'.

To check an existing cluster, first define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.databaseEncryption'
```

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

```
{
  "currentState": "CURRENT_STATE_ENCRYPTED",
  "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`, and choose the required cluster mode.
3. Within the `Security` heading, under `CLUSTER`, check `Encrypt secrets at the application layer` checkbox.
4. Select the kms 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. Select the cluster to be updated.
3. Under the Details pane, within the Security heading, click on the pencil named Application-layer secrets encryption.
4. Enable `Encrypt secrets at the application layer` and choose a kms key.

5. Click `SAVE CHANGES`.

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 Encrypt Sensitive Data at Rest 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.			●

5.4 Node Metadata

This section contains recommendations relating to node metadata in GKE.

5.4.1 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.

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

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 define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq
'.nodePools[].config.workloadMetadataConfig'
```

This should return the following for each Node pool:

```
{
    "mode": "GKE_METADATA"
}
```

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. Under the `DETAILS` pane, navigate down to the `Security` subsection.
4. Click on the pencil icon named `Edit Workload Identity`, click on `Enable Workload Identity` in the pop-up window, and select a workload pool from the drop-down box. By default, it will be the namespace of the Cloud project containing the cluster, for example: `<project_id>.svc.id.goog`.
5. Click `SAVE CHANGES` 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 container clusters update <cluster_name> --identity-
namespace=<project_id>.svc.id.goog
```

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

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

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

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

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
2. https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity
3. https://cloud.google.com/kubernetes-engine/docs/concepts/workload-identity

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	5.2 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 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 1

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, the containers can be brought up 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, the container can be brought up at the same time as the VM is created, with no on-host setup required.
- Smaller attack surface: COS has a smaller footprint, reducing the 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. 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, first define 4 variables for Node Pool, Cluster Name, Location and Project and then run the following command:

:

```
gcloud container node-pools describe $NODE_POOL --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --format json | jq
'.config.imageType'
```

The output of the above command should return the following output

```
"config": {
  ..
  "imageType": "COS_CONTAINERD",
  ..
}
```

if COS_CONTAINERD 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-optimised OS with containerd (cos_containerd) (default)` 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
--location <location> --node-pool <node_pool_name>
```

Default Value:

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

References:

1. https://cloud.google.com/kubernetes-engine/docs/concepts/using-containerd

2. https://cloud.google.com/kubernetes-engine/docs/concepts/node-images

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.		●	●

5.5.2 Ensure Node Auto-Repair is Enabled for GKE Nodes (Automated)

Profile Applicability:

- Level 2

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 the cluster in a healthy, running state. When enabled, Kubernetes Engine makes periodic checks on the health state of each node in the 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:

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, first define 4 variables for Node Pool, Cluster Name, Location and Project and then run the following command:

```
gcloud container node-pools describe $POOL_NAME --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --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 the Kubernetes cluster containing the node pool for which auto-repair is disabled.
3. Select the Node pool by clicking on the name of the pool.
4. Navigate to the Node pool details pane and click `EDIT`.
5. Under the `Management` heading, check the `Enable auto-repair` box.
6. Click `SAVE`.
7. Repeat steps 2-6 for every cluster and node pool with auto-upgrade disabled.

Using Command Line

To enable node auto-repair for an existing cluster's Node pool:

```
gcloud container node-pools update <node_pool_name> --cluster <cluster_name>
--location <location> --enable-autorepair
```

Default Value:

Node auto-repair is enabled by default.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/node-auto-repair

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets 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	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 2

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 the 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 the 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 the cluster, a maintenance window should be defined. A maintenance window is a four-hour timeframe that can be chosen, during 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, a maintenance exclusion should be defined. A maintenance exclusion can span multiple days.

Note: As part of the GKE node Auto-Upgrade process, the node is recreated. When a node is created, it automatically receives the latest version of the node image. For COS images, this is the primary OS upgrade mechanism.

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. 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, first define 4 variables for Node Pool, Cluster Name, Location and Project and then run the following command:

```
gcloud container node-pools describe $POOL_NAME --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --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 the Kubernetes cluster containing the node pool for which auto-upgrade disabled.
3. Select the Node pool by clicking on the name of the pool.
4. Navigate to the Node pool details pane and click `EDIT`.
5. Under the Management heading, check the `Enable auto-repair` box.
6. Click `SAVE`.
7. Repeat steps 2-6 for every cluster and node pool with auto-upgrade disabled.

Using Command Line

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

```
gcloud container node-pools update <node_pool_name> --cluster <cluster_name>
--location <location> --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
3. https://cloud.google.com/kubernetes-engine/docs/concepts/node-images
4. https://cloud.google.com/kubernetes-engine/docs/concepts/node-images

Additional Information:

Node auto-upgrades is not available for Alpha Clusters.

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.3 Perform Automated Operating System Patch Management Perform operating system updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.	●	●	●
v7	2.2 Ensure Software is Supported by Vendor 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	3.4 Deploy Automated Operating System Patch Management Tools 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 Deploy Automated Software Patch Management Tools 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 (Automated)

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, the cluster must be recreated without the `--release-channel` flag.

Node auto-upgrade is enabled (and cannot be disabled), so the 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:

To check for Release Channel within a cluster, first define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq .releaseChannel.channel
```

Ensure the output of the above command has JSON key attribute channel set to REGULAR or STABLE:

```
"releaseChannel": {
    "channel": "REGULAR"
  },
```

The output of the above command will return `regular` or `stable` if these release channels are being used to manage automatic upgrades for the 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`, and choose `CONFIGURE` for the required cluster mode.
3. Under the Control plane version heading, click the `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 required.
6. Click `CREATE`.

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

```
gcloud container clusters create <cluster_name> --location <location> --
release-channel <release_channel>
```

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

Default Value:

Currently, release channels are not enabled by default.

References:

1. https://cloud.google.com/kubernetes-engine/docs/concepts/release-channels
2. https://cloud.google.com/kubernetes-engine/docs/how-to/node-auto-upgrades
3. https://cloud.google.com/kubernetes-engine/docs/how-to/maintenance-windows-and-exclusions

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.	●	●	●
v7	3.4 Deploy Automated Operating System Patch Management Tools 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 Deploy Automated Software Patch Management Tools 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 (Automated)

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 Shielded GKE Nodes is enabled 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:

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:

To check for Shielded GKE Nodes within a cluster, first define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.shieldedNodes'
```

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

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

Remediation:

Note: From version 1.18, clusters will have Shielded GKE nodes enabled by default.

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 for which `Shielded GKE Nodes` is to be enabled.
3. With in the `Details` pane, under the `Security` heading, click on the pencil icon named `Edit Shields GKE nodes`.
4. Check the box named `Enable Shield GKE nodes`.
5. Click `SAVE CHANGES`.

Using Command Line:

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

```
gcloud container clusters update <cluster_name> --location <location> --
enable-shielded-nodes
```

Default Value:

Clusters will have Shielded GKE nodes enabled by default, as of version v1.18

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
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.		●	●

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.		●	●
v7	18.11 Use Standard Hardening Configuration Templates for Databases For applications that rely on a database, use standard hardening configuration templates. All systems that are part of critical business processes should also be tested.		●	●

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:

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 the cluster, first define 4 variables for Node Pool, Cluster Name, Location and Project and then run the following command for each Node pool:

```
gcloud container node-pools describe $POOL_NAME --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --format json | jq
.config.shieldedInstanceConfig
```

This will return the following, if Integrity Monitoring is enabled:

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

Remediation:

Once a Node pool is provisioned, it cannot be updated to enable Integrity Monitoring. New Node pools must be created 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`.

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

Using Command Line

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

```
gcloud container node-pools create <node_pool_name> --cluster <cluster_name>
--location <location> --shielded-integrity-monitoring
```

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

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
2. https://cloud.google.com/compute/shielded-vm/docs/integrity-monitoring

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.5 Perform Automated Vulnerability Scans of Internal Enterprise Assets Perform automated vulnerability scans of internal enterprise assets on a quarterly, or more frequent, basis. Conduct both authenticated and unauthenticated scans, using a SCAP-compliant vulnerability scanning tool.		●	●
v8	7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets 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	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:

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 the cluster, first define 4 variables for Node Pool, Cluster Name, Location and Project and then run the following command for each Node pool:

```
gcloud container node-pools describe $POOL_NAME --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --format json | jq
.config.shieldedInstanceConfig
```

This will return the value below, if Secure Boot is enabled:

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

Remediation:

Once a Node pool is provisioned, it cannot be updated to enable Secure Boot. New Node pools must be created 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`.

Workloads will need to be migrated 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 container node-pools create <node_pool_name> --cluster <cluster_name>
--location <location> --shielded-secure-boot
```

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

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
2. https://cloud.google.com/kubernetes-engine/docs/how-to/hardening-your-cluster

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	7.5 Perform Automated Vulnerability Scans of Internal Enterprise Assets Perform automated vulnerability scans of internal enterprise assets on a quarterly, or more frequent, basis. Conduct both authenticated and unauthenticated scans, using a SCAP-compliant vulnerability scanning tool.		●	●
v8	7.6 Perform Automated Vulnerability Scans of Externally-Exposed Enterprise Assets 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	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 intranode pod to pod traffic visible to the networking fabric. With this feature, VPC Flow Logs or other VPC features can be used for intranode traffic.

Impact:

Enabling it on existing cluster causes the cluster master and the cluster nodes to restart, which might cause disruption.

Audit:

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:

To check for Intranode Visibility in the cluster, first define 3 variables Cluster Name, Location and Project and then run the following command for each Node pool:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq
'.networkConfig.enableIntraNodeVisibility'
```

The result should return:

```
{
    "enableIntraNodeVisibility": true
}
```

if Intranode Visibility is Enabled.

Remediation:

Enable Intranode Visibility: 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. Within the `Details` pane, under the `Network` section, click on the pencil icon named `Edit intranode visibility`.
4. Check the box next to `Enable Intranode visibility`.
5. Click `SAVE CHANGES`.

Using Command Line:

To enable intranode visibility on an existing cluster, run the following command:

```
gcloud container clusters update <cluster_name> --enable-intra-node-
visibility
```

Enable VPC Flow Logs: Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting: https://console.cloud.google.com/kubernetes/list.
2. Select Kubernetes clusters for which VPC Flow Logs are disabled.
3. Select `Nodes` tab.
4. Select Node Pool without VPC Flow Logs enabled.
5. Select an Instance Group within the node pool.
6. Select an `Instance Group Member`.
7. Select the `Subnetwork` under Network Interfaces.
8. Click on `EDIT`.
9. Set Flow logs to `On`.
10. Click `SAVE`.

Using Command Line:

1. Find the subnetwork name associated with the cluster.

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

2. Update the subnetwork to enable VPC Flow Logs.

```
gcloud compute networks subnets update <subnet_name> --enable-flow-logs
```

Default Value:

By default, Intranode Visibility is disabled.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/intranode-visibility
2. https://cloud.google.com/vpc/docs/using-flow-logs

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.		●	●
v8	13.4 Perform Traffic Filtering Between Network Segments Perform traffic filtering between network segments, where appropriate.		●	●
v8	13.6 Collect Network Traffic Flow Logs Collect network traffic flow logs and/or network traffic to review and alert upon from network devices.		●	●
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:

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 'Networking' section, make sure 'VPC-native traffic routing' is set to 'Enabled'.

Using Command Line:

To check Alias IP is enabled for an existing cluster, first define 3 variables Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.ipAllocationPolicy.useIpAliases'
```

The output of the above command should return

```
{
    "useIpAliases": true
}
```

if VPC-native (using alias IP) is enabled. If VPC-native (using alias IP) is disabled, the above command will return null (`{ }`).

Remediation:

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:

If using Standard configuration mode:

1. Go to Kubernetes Engine by visiting: https://console.cloud.google.com/kubernetes/list
2. Click `CREATE CLUSTER`, and select Standard configuration mode.
3. Configure your cluster as desired , then, click `Networking` under `CLUSTER` in the navigation pane.
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> --location <location> --
enable-ip-alias
```

Default Value:

By default, VPC-native (using alias IP) is enabled when you create a new cluster. The exception is clusters created before GKE version 1.21.0-gke.1500 (September 2021) using the gcloud CLI without --enable-ip-alias."

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/alias-ips
2. https://cloud.google.com/kubernetes-engine/docs/concepts/alias-ips

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	13.4 Perform Traffic Filtering Between Network Segments Perform traffic filtering between network segments, where appropriate.		●	●
v7	11 Secure Configuration for Network Devices, such as Firewalls, Routers and Switches Secure Configuration for Network Devices, such as Firewalls, Routers and Switches			
v7	14.1 Segment the Network Based on Sensitivity 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).		●	●

5.6.3 Ensure Control Plane Authorized Networks is Enabled (Automated)

Profile Applicability:

- Level 2

Description:

Enable Control Plane Authorized Networks to restrict access to the cluster's control plane to only an allowlist 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.

Control Plane 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 Control Plane Authorized Networks, be careful to ensure all desired networks are on the allowlist to prevent inadvertently blocking external access to your cluster's control plane.

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 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, first define 3 variables Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.masterAuthorizedNetworksConfig'
```

The output should return

```
{
  "gcpPublicCidrsAccessEnabled": true
}
```

if Control Plane 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 Control Plane Authorized Networks is disabled
3. Within the Details pane, under the Networking heading, click on the pencil icon named Edit control plane authorised networks.
4. Check the box next to Enable control plane authorised networks.
5. Click SAVE CHANGES.

Using Command Line:

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

```
gcloud container clusters update <cluster_name> --location <location> --
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, Control Plane Authorized Networks is disabled.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/authorized-networks
2. https://cloud.google.com/kubernetes-engine/docs/how-to/latest/network-isolation
3. https://cloud.google.com/kubernetes-engine/docs/how-to/latest/network-isolation

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.3 Configure Data Access Control Lists 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 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.	●	●	●

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:

Using Google Cloud Console:

1. Go to Kubernetes Engine by visiting https://console.cloud.google.com/kubernetes/list
2. Select the required cluster, and within the Details pane, make sure the 'Endpoint' does not have a public IP address.

Using Command Line:

To check Private Endpoint status for an existing cluster, first define 3 variables Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq
'.privateClusterConfig.enablePrivateEndpoint'
```

The output of the above command returns

```
{
      "enablePrivateEndpoint": 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, and choose CONFIGURE for the Standard mode cluster.
3. Configure the cluster as required then click Networking under CLUSTER in the navigation pane.
4. Under IPv4 network access, click the Private cluster radio button.
5. Uncheck the Access control plane using its external IP address checkbox.
6. In the Control plane IP range textbox, provide an IP range for the control plane.
7. Configure the other settings as required, and 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/latest/network-isolation

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			

5.6.5 Ensure clusters are created with Private Nodes (Automated)

Profile Applicability:

- Level 1

Description:

Private Nodes are nodes with no public IP addresses. Disable public IP addresses for cluster nodes, so that they only have private 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:

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:

To check Private Nodes status for an existing cluster, first define 3 variables Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq
'.privateClusterConfig.enablePrivateNodes'
```

The output of the above command returns

```
{
      "enablePrivateNodes": 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 required then click Networking under CLUSTER in the navigation pane.
4. Under IPv4 network access, click the Private cluster radio button.
5. Configure the other settings as required, and 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/latest/network-isolation

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, first define 3 variables Instance Name, Location and Project and then run the following command:

```
gcloud compute instances describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --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 the 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:

```
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/kubernetes-engine/docs/concepts/cluster-architecture
2. https://cloud.google.com/vpc/docs/using-firewalls
3. https://cloud.google.com/kubernetes-engine/docs/how-to/tags-firewall-policies

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	9.5 Implement Application Firewalls 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 use of Google-managed SSL Certificates (Automated)

Profile Applicability:

- Level 2

Description:

Encrypt traffic to HTTPS load balancers using Google-managed SSL certificates.

Rationale:

Encrypting traffic between users and the Kubernetes workload is fundamental to protecting data sent over the web.

Google-managed SSL Certificates are provisioned, renewed, and managed for 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 that are self obtained and managed. 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")'
```

Consider using ingresses instead of these services in order to use Google managed SSL certificates.

For the ingresses within the 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
2. https://cloud.google.com/kubernetes-engine/docs/concepts/ingress

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.		●	●

5.7 Logging

This section contains recommendations relating to security-related logging in GKE.

5.7.1 Ensure Logging and Cloud 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 Cloud Operations for GKE 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.

Audit:

Using Google Cloud Console:

LOGGING AND CLOUD 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 of interest.
3. Under the details pane, within the Features section, ensure that `Logging` is `Enabled`.
4. Also ensure that `Cloud Monitoring` is `Enabled`

LEGACY STACKDRIVER SUPPORT:

This option cannot be check in the GCP console.

Using Command Line:

LOGGING AND CLOUD MONITORING SUPPORT (PREFERRED):

First define 3 variables for Cluster Name, Location and Project and then run the following commands:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.loggingService'
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.monitoringService'
```

The output of the above commands should return `logging.googleapis.com/kubernetes` and `monitoring.googleapis.com/kubernetes` respectively if Logging and Cloud Monitoring is Enabled.

LEGACY STACKDRIVER SUPPORT:

Note: This functionality was decommissioned on 31st March 2021, kept here for posterity (see: https://cloud.google.com/stackdriver/docs/deprecations/legacy for more information)

Both Logging and Monitoring support must be enabled.

For Logging, run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --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 --location $LOCATION --
project $PROJECT_ID --format json | jq '.monitoringService'
```

The output should return `monitoring.googleapis.com` if Legacy Stackdriver Monitoring is Enabled.

Remediation:

Using Google Cloud Console: To enable Logging:

1. Go to Kubernetes Engine by visiting: https://console.cloud.google.com/kubernetes/list.
2. Select the cluster for which Logging is disabled.
3. Under the details pane, within the Features section, click on the pencil icon named `Edit logging`.
4. Check the box next to `Enable Logging`.
5. In the drop-down Components box, select the components to be logged.
6. Click `SAVE CHANGES`, and wait for the cluster to update.

To enable Cloud Monitoring:

1. Go to Kubernetes Engine by visiting: https://console.cloud.google.com/kubernetes/list.
2. Select the cluster for which Logging is disabled.

3. Under the details pane, within the Features section, click on the pencil icon named `Edit Cloud Monitoring`.
4. Check the box next to `Enable Cloud Monitoring`.
5. In the drop-down Components box, select the components to be logged.
6. Click `SAVE CHANGES`, and wait for the cluster to update.

Using Command Line: To enable Logging for an existing cluster, run the following command:

```
gcloud container clusters update <cluster_name> --location <location> --
logging=<components_to_be_logged>
```

See https://cloud.google.com/kubernetes-engine/docs/how-to/configure-metrics#available-metrics for a list of available metrics for logging.

To enable Cloud Monitoring for an existing cluster, run the following command:

```
gcloud container clusters update <cluster_name> --location <location> --
monitoring=<components_to_be_logged>
```

See https://cloud.google.com/kubernetes-engine/docs/concepts/about-logs#available-logs for a list of available logging values.

Default Value:

Logging and Cloud Monitoring is enabled by default starting in GKE version 1.14; Legacy Logging and Monitoring support is enabled by default for earlier versions.

References:

1. https://cloud.google.com/stackdriver/docs/solutions/gke/observing
2. https://cloud.google.com/stackdriver/docs/solutions/gke/managing-logs
3. https://cloud.google.com/stackdriver/docs/solutions/gke/installing
4. https://cloud.google.com/kubernetes-engine/docs/how-to/configure-metrics#available-metrics
5. https://cloud.google.com/kubernetes-engine/docs/concepts/about-logs#available-logs

Additional Information:

Default logging levels are the best place to start. The References section provides additional details about options, metrics, and configuration details

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.	●	●	●
v7	6.2 Activate audit logging 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 the workload and may incur additional resource costs. Audit logs sent to Stackdriver consume log quota from the project. The log quota may require increasing 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, 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 a different Namespace was defined in the manifests, replace `cos-auditd` with the name of the namespace being used:

```
kubectl get pods --namespace=cos-auditd
```

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
2. https://cloud.google.com/container-optimized-os/docs

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.	●	●	●
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 authentication using Client Certificates is Disabled (Automated)

Profile Applicability:

- Level 1

Description:

Do not make use of client certificate authentication in GKE, as the credentials cannot be revoked. 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 not be used where possible.

When Client Certificate creation is disabled GKE will not provide a client certificate on cluster creation, however users will still be able to use the Certificate Signing Request (CSR) API to create new client certificates, if they have access to it.

Impact:

Users will no longer be able to authenticate with the pre-provisioned x509 certificate. You should configure and use alternate authentication mechanisms, such as OpenID Connect tokens.

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 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, first define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --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).

Note. Depreciated as of v1.19. For Basic Authentication, Legacy authorization can be edited for standard clusters but cannot be edited in Autopilot clusters.

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
```

In addition it's important to restrict access to the CSR API in Kubernetes to prevent users from using it to issue new client certificate credentials.

Default Value:

Google Kubernetes Engine (GKE), both Basic Authentication and Client Certificate issuance are disabled by default for new clusters. This change was implemented starting with GKE version 1.12 to enhance security by reducing the attack surface associated with these authentication methods.

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.			●
v7	16 <u>Account Monitoring and Control</u> Account Monitoring and Control			

5.8.2 Manage Kubernetes RBAC users for GKE with groups in Google Workspace (Manual)

Profile Applicability:

- Level 2

Description:

Cluster Administrators should leverage groups in Google Workspace 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 groups in Google Workspace 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 groups in Google Workspace Admin Console and Google Cloud Console

1. Navigate to manage groups in Google Workspace 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`.

Using Command Line:

To check RBAC users for GKE with groups in Google Workspace for an existing cluster, first define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '{Enabled:
.authenticatorGroupsConfig.enabled, "Security Group":
.authenticatorGroupsConfig.securityGroup}'
```

- Check that the Enabled is 'True'.
- Check that Security Group has the format 'gke-security-groups@DOMAIN'. The local-part MUST be 'gke-security-groups'. DOMAIN should be the domain associated with Google Workspace.
- Verify that the Security Group shown exists in Google Workspace. (out of the scope of this document)

Remediation:

Follow the groups in Google Workspace instructions at: Configure Google Groups for RBAC.

Command line statement to create a new cluster

```
gcloud container clusters create CLUSTER_NAME \
    --location=CONTROL_PLANE_LOCATION \
    --security-group="gke-security-groups@DOMAIN"
```

Command line statement to update an existing cluster

```
gcloud container clusters update CLUSTER_NAME \
    --location=CONTROL_PLANE_LOCATION \
    --security-group="gke-security-groups@DOMAIN"
```

In both create and update statements, replace the following:

- CLUSTER_NAME: the name of the new cluster.
- CONTROL_PLANE_LOCATION: the Compute Engine location of the control plane of your cluster. Provide a region for regional clusters, or a zone for zonal clusters.
- DOMAIN: the domain name of the gke-security-groups group you created.

NOTE: To run at the console, click on the following link and follow the numbered steps in the console tab: Enable Google Groups for RBAC on clusters

Finally create `Roles`, `ClusterRoles`, `RoleBindings`, and `ClusterRoleBindings` that reference the groups in Google Workspace.

Default Value:

Google Groups for GKE is disabled by default.

References:

1. https://cloud.google.com/kubernetes-engine/docs/how-to/google-groups-rbac
2. https://cloud.google.com/kubernetes-engine/docs/how-to/role-based-access-control

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	6.8 Define and Maintain Role-Based Access Control 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 Configure Centralized Point of Authentication Configure access for all accounts through as few centralized points of authentication as possible, including network, security, and cloud systems.		●	●

5.8.3 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 the definition of 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, the user must be granted the ability to create authorization roles using RBAC to ensure that the role-based access control permissions take effect.

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 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, first define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.legacyAbac'
```

The output should return null

```
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> --location <location> --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

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 the role-based access control permissions take effect for a Kubernetes service account, the cluster must be created or updated 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 Define and Maintain Role-Based Access Control 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	4 Controlled Use of Administrative Privileges Controlled Use of Administrative Privileges			
v7	16 Account Monitoring and Control 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 2

Description:

Use Customer-Managed Encryption Keys (CMEK) to encrypt 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:

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 CMEK is being configured with a regional cluster, the cluster must run GKE 1.14 or higher.

Audit:

Using Google Cloud Console:

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:

Identify the Persistent Volumes Used by the cluster:

```
kubectl get pv -o json | jq '.items[].metadata.name'
```

For each volume used create 2 variables for Persistent Volume Name, Location and Project and then check that it is encrypted using a customer managed key by running the following command:

```
gcloud compute disks describe $PV_NAME --location $LOCATION --project
$PROJECT_ID --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. The node pool must either be recreated or a new cluster created.

Using Google Cloud Console:

This is not possible using Google Cloud Console.

Using Command Line:

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/
4. https://cloud.google.com/kubernetes-engine/docs/concepts/persistent-volumes
5. https://cloud.google.com/sdk/gcloud/reference/container/node-pools/create

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest 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 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.			●

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

Profile Applicability:

- Level 2

Description:

Use Customer-Managed Encryption Keys (CMEK) to encrypt node boot disks 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:

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 CMEK is being configured with a regional cluster, the cluster must run GKE 1.14 or higher.

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
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.

Using Command Line:

To check for Customer-Managed Encryption Keys (CMEK) first define 4 variables for Node Pool, Cluster Name, Location and Project and then run this command:

```
gcloud container node-pools describe $NODE_POOL --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --format="table(name,
config.diskType)"
```

Sample Output:

```
NAME    DISK_TYPE
pool1   pd-balanced
```

Verify that the output of the above command includes a `diskType` of either `pd-standard`, `pd-balanced` or `pd-ssd`, and the `bootDiskKmsKey` is specified as the desired key.

Remediation:

This cannot be remediated by updating an existing cluster. The node pool must either be recreated or a new cluster created.

Using Google Cloud Console:

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. In the Nodes section, under machine configuration, 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 to be used.
6. Click `CREATE`.

To create a new cluster:

1. Go to Kubernetes Engine by visiting: https://console.cloud.google.com/kubernetes/list
2. Click `CREATE` and click `CONFIGURE for the required cluster mode.
3. Under `NODE POOLS, expand the default-pool list and click` Nodes.
4. In the Configure node settings pane, 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 to be used.
6. Configure the rest of the cluster settings as required.
7. Click `CREATE`.

Using Command Line:

Create a new node pool using customer-managed encryption keys for the node boot disk, of `<disk_type>` either `pd-standard` or `pd-ssd`:

```
gcloud container node-pools create <cluster_name> --disk-type <disk_type> --
boot-disk-kms-key
projects/<key_project_id>/locations/<location>/keyRings/<ring_name>/cryptoKey
s/<key_name>
```

Create a cluster using customer-managed encryption keys for the node boot disk, of `<disk_type>` either `pd-standard` or `pd-ssd`:

```
gcloud container clusters create <cluster_name> --disk-type <disk_type> --
boot-disk-kms-key
projects/<key_project_id>/locations/<location>/keyRings/<ring_name>/cryptoKey
s/<key_name>
```

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/
4. https://cloud.google.com/kubernetes-engine/docs/concepts/persistent-volumes
5. https://cloud.google.com/sdk/gcloud/reference/container/node-pools/create

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	3.11 Encrypt Sensitive Data at Rest 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 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.			●

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:

Note: 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.

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:

Using Google Cloud Console:

Currently not possible, due to the add-on having been removed. Must use the command line.

Using Command Line:

First define 3 variables for Cluster Name, Location and Project and then run the following command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID  --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:

Currently not possible, due to the add-on having been removed. Must use the command line.

Using Command Line:

To disable the Kubernetes Dashboard on an existing cluster, run the following command:

```
gcloud container clusters update <cluster_name> --location <location> --
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 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.		●	●
v7	2.2 Ensure Software is Supported by Vendor 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 Only Use Up-to-date And Trusted Third-Party Components 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:

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

First define 2 variables for Cluster Name, Location and Project and then run the command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --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, and choose "SWITCH TO STANDARD CLUSTER" in the upper right corner of the screen.
3. Under Features in the the CLUSTER section, "Enable Kubernetes alpha features in this cluster" will not be available by default and to use Kubernetes alpha features in this cluster, first disable release channels.

   Note: It will only be available if the cluster is created with a Static version for the Control plane version, along with both Automatically upgrade nodes to the next available version and Enable auto-repair being checked under the Node pool details for each node.

4. Configure the other settings as required and click CREATE.

Using Command Line:

Upon creating a new cluster

```
gcloud container clusters create <cluster name> --location <location>
```

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 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.		●	●

5.10.3 Consider GKE Sandbox for running untrusted workloads (Automated)

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:

First define 3 variable for Node Pool, Cluster Name, Location and Project and then run this command:

```
gcloud container node-pools describe $NODE_POOL --cluster $CLUSTER_NAME --
location $LOCATION --project $PROJECT_ID --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. Select a cluster and 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) (default)`.
   - 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, which can be done using:

```
  gcloud container node-pools create <node_pool_name> --location <location> -
-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/concepts/sandbox-pods
2. https://cloud.google.com/kubernetes-engine/docs/concepts/node-pools
3. 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.		●	●

Controls Version	Control	IG 1	IG 2	IG 3
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.		●	●

5.10.4 Enable Security Posture (Automated)

Profile Applicability:

- Level 2

Description:

Rationale:

The security posture dashboard provides insights about your workload security posture at the runtime phase of the software delivery life-cycle.

Impact:

GKE security posture configuration auditing checks your workloads against a set of defined best practices. Each configuration check has its own impact or risk. Learn more about the checks: About Kubernetes security posture scanning and Automatically audit workloads for configuration issues

Example: The host namespace check identifies pods that share host namespaces. Pods that share host namespaces allow Pod processes to communicate with host processes and gather host information, which could lead to a container escape

Audit:

Define 3 variables for Cluster Name, Location and Project, then run this command:

```
gcloud container clusters describe $CLUSTER_NAME --location $LOCATION --
project $PROJECT_ID --format json | jq '.securityPostureConfig'
```

Sample output from command line statement:

```
{
  "mode": "BASIC",
  "vulnerabilityMode": "VULNERABILITY_DISABLED"
}
```

The output of Mode should be BASIC or ENTERPRISE.

Remediation:

To enable Security Posture for a new cluster, be sure to add `--security-posture=standard` or `--security-posture=enterprise` when creating.

To update Security Posture for an existing cluster, run the following command:

```
gcloud container clusters update CLUSTER_NAME \
    --location=CONTROL_PLANE_LOCATION \
    --security-posture=standard
```

Replace the following:

- CLUSTER_NAME: the name of your cluster.

- CONTROL_PLANE_LOCATION: the location of the control plane of your cluster. Provide a region for regional Standard and Autopilot clusters, or a zone for zonal Standard clusters.

For console users follow the numbered steps in the console tab using the link provided: Enable configuration auditing on an existing cluster

Default Value:

GKE security posture has multiple features. Not all are on by default. Configuration auditing is enabled by default for new standard and autopilot clusters.

securityPostureConfig: mode: BASIC

References:

1. https://cloud.google.com/kubernetes-engine/docs/concepts/about-security-posture-dashboard
2. https://cloud.google.com/kubernetes-engine/docs/concepts/about-configuration-scanning
3. https://cloud.google.com/kubernetes-engine/docs/how-to/protect-workload-configuration
4. https://cloud.google.com/kubernetes-engine/docs/reference/rest/v1/projects.locations.clusters#Cluster.SecurityPostureConfig

CIS Controls:

Controls Version	Control	IG 1	IG 2	IG 3
v8	2.4 Utilize Automated Software Inventory Tools Utilize software inventory tools, when possible, throughout the enterprise to automate the discovery and documentation of installed software.		●	●
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**		
3	**Worker Nodes**		
3.1	**Worker Node Configuration Files**		
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive (Automated)	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root (Automated)	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644 (Automated)	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root (Automated)	☐	☐
4	**Policies**		
4.1	**RBAC and Service Accounts**		
4.1.1	Ensure that the cluster-admin role is only used where required (Manual)	☐	☐
4.1.2	Minimize access to secrets (Manual)	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles (Manual)	☐	☐
4.1.4	Ensure that default service accounts are not actively used (Automated)	☐	☐
4.1.5	Ensure that Service Account Tokens are only mounted where necessary (Manual)	☐	☐
4.1.6	Avoid use of system:masters group (Automated)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.1.7	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster (Manual)	☐	☐
4.1.8	Avoid bindings to system:anonymous (Automated)	☐	☐
4.1.9	Avoid non-default bindings to system:unauthenticated (Automated)	☐	☐
4.1.10	Avoid non-default bindings to system:authenticated (Automated)	☐	☐
4.2	**Pod Security Standards**		
4.2.1	Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces. (Manual)	☐	☐
4.3	**Network Policies and CNI**		
4.3.1	Ensure that the CNI in use supports Network Policies (Manual)	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined (Automated)	☐	☐
4.4	**Secrets Management**		
4.4.1	Prefer using secrets as files over secrets as environment variables (Automated)	☐	☐
4.4.2	Consider external secret storage (Manual)	☐	☐
4.5	**Extensible Admission Control**		
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller (Manual)	☐	☐
4.6	**General Policies**		
4.6.1	Create administrative boundaries between resources using namespaces (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
4.6.2	Ensure that the seccomp profile is set to RuntimeDefault in the pod definitions (Automated)	☐	☐
4.6.3	Apply Security Context to Pods and Containers (Manual)	☐	☐
4.6.4	The default namespace should not be used (Manual)	☐	☐
5	**Managed services**		
5.1	**Image Registry and Image Scanning**		
5.1.1	Ensure Image Vulnerability Scanning is enabled (Automated)	☐	☐
5.1.2	Minimize user access to Container Image repositories (Manual)	☐	☐
5.1.3	Minimize cluster access to read-only for Container Image repositories (Manual)	☐	☐
5.1.4	Ensure only trusted container images are used (Manual)	☐	☐
5.2	**Identity and Access Management (IAM)**		
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account (Automated)	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity (Manual)	☐	☐
5.3	**Cloud Key Management Service (Cloud KMS)**		
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS (Automated)	☐	☐
5.4	**Node Metadata**		
5.4.1	Ensure the GKE Metadata Server is Enabled (Automated)	☐	☐
5.5	**Node Configuration and Maintenance**		

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE Node images (Automated)	☐	☐
5.5.2	Ensure Node Auto-Repair is Enabled for GKE Nodes (Automated)	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes (Automated)	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels (Automated)	☐	☐
5.5.5	Ensure Shielded GKE Nodes are Enabled (Automated)	☐	☐
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled (Automated)	☐	☐
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled (Automated)	☐	☐
5.6	**Cluster Networking**		
5.6.1	Enable VPC Flow Logs and Intranode Visibility (Automated)	☐	☐
5.6.2	Ensure use of VPC-native clusters (Automated)	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled (Automated)	☐	☐
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled (Automated)	☐	☐
5.6.5	Ensure clusters are created with Private Nodes (Automated)	☐	☐
5.6.6	Consider firewalling GKE worker nodes (Manual)	☐	☐
5.6.7	Ensure use of Google-managed SSL Certificates (Automated)	☐	☐
5.7	**Logging**		

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.7.1	Ensure Logging and Cloud Monitoring is Enabled (Automated)	☐	☐
5.7.2	Enable Linux auditd logging (Manual)	☐	☐
5.8	**Authentication and Authorization**		
5.8.1	Ensure authentication using Client Certificates is Disabled (Automated)	☐	☐
5.8.2	Manage Kubernetes RBAC users for GKE with groups in Google Workspace (Manual)	☐	☐
5.8.3	Ensure Legacy Authorization (ABAC) is Disabled (Automated)	☐	☐
5.9	**Storage**		
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD) (Manual)	☐	☐
5.9.2	Enable Customer-Managed Encryption Keys (CMEK) for Boot Disks (Automated)	☐	☐
5.10	**Other Cluster Configurations**		
5.10.1	Ensure Kubernetes Web UI is Disabled (Automated)	☐	☐
5.10.2	Ensure that Alpha clusters are not used for production workloads (Automated)	☐	☐
5.10.3	Consider GKE Sandbox for running untrusted workloads (Automated)	☐	☐
5.10.4	Enable Security Posture (Automated)	☐	☐

Appendix: CIS Controls v7 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.4	Ensure that default service accounts are not actively used	☐	☐
4.1.8	Avoid bindings to system:anonymous	☐	☐
4.1.9	Avoid non-default bindings to system:unauthenticated	☐	☐
4.1.10	Avoid non-default bindings to system:authenticated	☐	☐
4.2.1	Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces.	☐	☐
4.6.3	Apply Security Context to Pods and Containers	☐	☐
5.1.2	Minimize user access to Container Image repositories	☐	☐
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled	☐	☐
5.7.1	Ensure Logging and Cloud Monitoring is Enabled	☐	☐
5.10.1	Ensure Kubernetes Web UI is Disabled	☐	☐

Appendix: CIS Controls v7 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Ensure that default service accounts are not actively used	☐	☐
4.1.8	Avoid bindings to system:anonymous	☐	☐
4.1.9	Avoid non-default bindings to system:unauthenticated	☐	☐
4.1.10	Avoid non-default bindings to system:authenticated	☐	☐
4.2.1	Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces.	☐	☐
4.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.6.2	Ensure that the seccomp profile is set to RuntimeDefault in the pod definitions	☐	☐
4.6.3	Apply Security Context to Pods and Containers	☐	☐
5.1.1	Ensure Image Vulnerability Scanning is enabled	☐	☐
5.1.2	Minimize user access to Container Image repositories	☐	☐
5.1.3	Minimize cluster access to read-only for Container Image repositories	☐	☐
5.1.4	Ensure only trusted container images are used	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	☐	☐
5.4.1	Ensure the GKE Metadata Server is Enabled	☐	☐
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE Node images	☐	☐
5.5.2	Ensure Node Auto-Repair is Enabled for GKE Nodes	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	☐	☐
5.5.5	Ensure Shielded GKE Nodes are Enabled	☐	☐
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	☐	☐
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	☐	☐
5.6.1	Enable VPC Flow Logs and Intranode Visibility	☐	☐
5.6.2	Ensure use of VPC-native clusters	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled	☐	☐
5.6.7	Ensure use of Google-managed SSL Certificates	☐	☐
5.7.1	Ensure Logging and Cloud Monitoring is Enabled	☐	☐
5.7.2	Enable Linux auditd logging	☐	☐
5.8.2	Manage Kubernetes RBAC users for GKE with groups in Google Workspace	☐	☐
5.10.1	Ensure Kubernetes Web UI is Disabled	☐	☐
5.10.2	Ensure that Alpha clusters are not used for production workloads	☐	☐
5.10.3	Consider GKE Sandbox for running untrusted workloads	☐	☐
5.10.4	Enable Security Posture	☐	☐

Appendix: CIS Controls v7 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Ensure that default service accounts are not actively used	☐	☐
4.1.5	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.1.8	Avoid bindings to system:anonymous	☐	☐
4.1.9	Avoid non-default bindings to system:unauthenticated	☐	☐
4.1.10	Avoid non-default bindings to system:authenticated	☐	☐
4.2.1	Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces.	☐	☐
4.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.6.2	Ensure that the seccomp profile is set to RuntimeDefault in the pod definitions	☐	☐
4.6.3	Apply Security Context to Pods and Containers	☐	☐
4.6.4	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning is enabled	☐	☐
5.1.2	Minimize user access to Container Image repositories	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.1.3	Minimize cluster access to read-only for Container Image repositories	☐	☐
5.1.4	Ensure only trusted container images are used	☐	☐
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS	☐	☐
5.4.1	Ensure the GKE Metadata Server is Enabled	☐	☐
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE Node images	☐	☐
5.5.2	Ensure Node Auto-Repair is Enabled for GKE Nodes	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	☐	☐
5.5.5	Ensure Shielded GKE Nodes are Enabled	☐	☐
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	☐	☐
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	☐	☐
5.6.1	Enable VPC Flow Logs and Intranode Visibility	☐	☐
5.6.2	Ensure use of VPC-native clusters	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled	☐	☐
5.6.6	Consider firewalling GKE worker nodes	☐	☐
5.6.7	Ensure use of Google-managed SSL Certificates	☐	☐
5.7.1	Ensure Logging and Cloud Monitoring is Enabled	☐	☐
5.7.2	Enable Linux auditd logging	☐	☐
5.8.2	Manage Kubernetes RBAC users for GKE with groups in Google Workspace	☐	☐
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD)	☐	☐
5.9.2	Enable Customer-Managed Encryption Keys (CMEK) for Boot Disks	☐	☐
5.10.1	Ensure Kubernetes Web UI is Disabled	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.10.2	Ensure that Alpha clusters are not used for production workloads	☐	☐
5.10.3	Consider GKE Sandbox for running untrusted workloads	☐	☐
5.10.4	Enable Security Posture	☐	☐

Appendix: CIS Controls v7 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
	No unmapped recommendations to CIS Controls v7	☐	☐

Appendix: CIS Controls v8 IG 1 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Ensure that default service accounts are not actively used	☐	☐
4.1.6	Avoid use of system:masters group	☐	☐
4.1.7	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐
5.1.2	Minimize user access to Container Image repositories	☐	☐
5.1.3	Minimize cluster access to read-only for Container Image repositories	☐	☐
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.6.5	Ensure clusters are created with Private Nodes	☐	☐
5.6.6	Consider firewalling GKE worker nodes	☐	☐
5.7.1	Ensure Logging and Cloud Monitoring is Enabled	☐	☐
5.7.2	Enable Linux auditd logging	☐	☐

Appendix: CIS Controls v8 IG 2 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Ensure that default service accounts are not actively used	☐	☐
4.1.5	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.1.6	Avoid use of system:masters group	☐	☐
4.1.7	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
4.1.8	Avoid bindings to system:anonymous	☐	☐
4.1.9	Avoid non-default bindings to system:unauthenticated	☐	☐
4.1.10	Avoid non-default bindings to system:authenticated	☐	☐
4.2.1	Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces.	☐	☐
4.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.6.2	Ensure that the seccomp profile is set to RuntimeDefault in the pod definitions	☐	☐
4.6.4	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning is enabled	☐	☐
5.1.2	Minimize user access to Container Image repositories	☐	☐
5.1.3	Minimize cluster access to read-only for Container Image repositories	☐	☐
5.1.4	Ensure only trusted container images are used	☐	☐
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS	☐	☐
5.4.1	Ensure the GKE Metadata Server is Enabled	☐	☐
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE Node images	☐	☐
5.5.2	Ensure Node Auto-Repair is Enabled for GKE Nodes	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	☐	☐
5.5.5	Ensure Shielded GKE Nodes are Enabled	☐	☐
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	☐	☐
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	☐	☐
5.6.1	Enable VPC Flow Logs and Intranode Visibility	☐	☐
5.6.2	Ensure use of VPC-native clusters	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled	☐	☐
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.6.5	Ensure clusters are created with Private Nodes	☐	☐
5.6.6	Consider firewalling GKE worker nodes	☐	☐
5.6.7	Ensure use of Google-managed SSL Certificates	☐	☐
5.7.1	Ensure Logging and Cloud Monitoring is Enabled	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.7.2	Enable Linux auditd logging	☐	☐
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD)	☐	☐
5.9.2	Enable Customer-Managed Encryption Keys (CMEK) for Boot Disks	☐	☐
5.10.1	Ensure Kubernetes Web UI is Disabled	☐	☐
5.10.2	Ensure that Alpha clusters are not used for production workloads	☐	☐
5.10.3	Consider GKE Sandbox for running untrusted workloads	☐	☐
5.10.4	Enable Security Posture	☐	☐

Appendix: CIS Controls v8 IG 3 Mapped Recommendations

Recommendation		Set Correctly	
		Yes	No
3.1.1	Ensure that the kubeconfig file permissions are set to 644 or more restrictive	☐	☐
3.1.2	Ensure that the kubelet kubeconfig file ownership is set to root:root	☐	☐
3.1.3	Ensure that the kubelet configuration file has permissions set to 644	☐	☐
3.1.4	Ensure that the kubelet configuration file ownership is set to root:root	☐	☐
4.1.1	Ensure that the cluster-admin role is only used where required	☐	☐
4.1.2	Minimize access to secrets	☐	☐
4.1.3	Minimize wildcard use in Roles and ClusterRoles	☐	☐
4.1.4	Ensure that default service accounts are not actively used	☐	☐
4.1.5	Ensure that Service Account Tokens are only mounted where necessary	☐	☐
4.1.6	Avoid use of system:masters group	☐	☐
4.1.7	Limit use of the Bind, Impersonate and Escalate permissions in the Kubernetes cluster	☐	☐
4.1.8	Avoid bindings to system:anonymous	☐	☐
4.1.9	Avoid non-default bindings to system:unauthenticated	☐	☐
4.1.10	Avoid non-default bindings to system:authenticated	☐	☐
4.2.1	Ensure that the cluster enforces Pod Security Standard Baseline profile or stricter for all namespaces.	☐	☐
4.3.1	Ensure that the CNI in use supports Network Policies	☐	☐
4.3.2	Ensure that all Namespaces have Network Policies defined	☐	☐
4.5.1	Configure Image Provenance using ImagePolicyWebhook admission controller	☐	☐

Recommendation		Set Correctly	
		Yes	No
4.6.2	Ensure that the seccomp profile is set to RuntimeDefault in the pod definitions	☐	☐
4.6.4	The default namespace should not be used	☐	☐
5.1.1	Ensure Image Vulnerability Scanning is enabled	☐	☐
5.1.2	Minimize user access to Container Image repositories	☐	☐
5.1.3	Minimize cluster access to read-only for Container Image repositories	☐	☐
5.1.4	Ensure only trusted container images are used	☐	☐
5.2.1	Ensure GKE clusters are not running using the Compute Engine default service account	☐	☐
5.2.2	Prefer using dedicated GCP Service Accounts and Workload Identity	☐	☐
5.3.1	Ensure Kubernetes Secrets are encrypted using keys managed in Cloud KMS	☐	☐
5.4.1	Ensure the GKE Metadata Server is Enabled	☐	☐
5.5.1	Ensure Container-Optimized OS (cos_containerd) is used for GKE Node images	☐	☐
5.5.2	Ensure Node Auto-Repair is Enabled for GKE Nodes	☐	☐
5.5.3	Ensure Node Auto-Upgrade is Enabled for GKE Nodes	☐	☐
5.5.4	When creating New Clusters - Automate GKE version management using Release Channels	☐	☐
5.5.5	Ensure Shielded GKE Nodes are Enabled	☐	☐
5.5.6	Ensure Integrity Monitoring for Shielded GKE Nodes is Enabled	☐	☐
5.5.7	Ensure Secure Boot for Shielded GKE Nodes is Enabled	☐	☐
5.6.1	Enable VPC Flow Logs and Intranode Visibility	☐	☐
5.6.2	Ensure use of VPC-native clusters	☐	☐
5.6.3	Ensure Control Plane Authorized Networks is Enabled	☐	☐
5.6.4	Ensure clusters are created with Private Endpoint Enabled and Public Access Disabled	☐	☐
5.6.5	Ensure clusters are created with Private Nodes	☐	☐
5.6.6	Consider firewalling GKE worker nodes	☐	☐
5.6.7	Ensure use of Google-managed SSL Certificates	☐	☐
5.7.1	Ensure Logging and Cloud Monitoring is Enabled	☐	☐

Recommendation		Set Correctly	
		Yes	No
5.7.2	Enable Linux auditd logging	☐	☐
5.8.1	Ensure authentication using Client Certificates is Disabled	☐	☐
5.8.2	Manage Kubernetes RBAC users for GKE with groups in Google Workspace	☐	☐
5.8.3	Ensure Legacy Authorization (ABAC) is Disabled	☐	☐
5.9.1	Enable Customer-Managed Encryption Keys (CMEK) for GKE Persistent Disks (PD)	☐	☐
5.9.2	Enable Customer-Managed Encryption Keys (CMEK) for Boot Disks	☐	☐
5.10.1	Ensure Kubernetes Web UI is Disabled	☐	☐
5.10.2	Ensure that Alpha clusters are not used for production workloads	☐	☐
5.10.3	Consider GKE Sandbox for running untrusted workloads	☐	☐
5.10.4	Enable Security Posture	☐	☐

Appendix: CIS Controls v8 Unmapped Recommendations

Recommendation		Set Correctly	
		Yes	No
	No unmapped recommendations to CIS Controls v8	☐	☐

Appendix: Change History

Date	Version	Changes for this version
November 1, 2025	1.9.0	Added and tested support for the latest Kubernetes Cluster version 1.34
October 15th, 2025	1.9.0	Added and tested support for the Kubernetes Cluster version 1.33
October 12, 2025	1.9.0	Updated recommendation 5.10.2 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.10.1 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.9.2 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.9.1 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.8.3 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.8.1 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.5.6 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.5.5 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.5.4 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.5.3 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.5.2 audit statement

Date	Version	Changes for this version
October 12, 2025	1.9.0	Updated recommendation 5.5.1 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.4.1 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.3.1 audit statement
October 12, 2025	1.9.0	Updated recommendation 5.2.2 audit statement
October 10th, 2025	1.9.0	Updated recommendation 5.2.1 audit statement
October 7th, 2025	1.9.0	Updated recommendation 4.6.4 audit statement
October 7th, 2025	1.9.0	Updated recommendation 4.3.2 audit statement
October 7th, 2025	1.9.0	Updated recommendation 4.1.4 audit statement
October 5, 2025	1.9.0	Updated recommendation 4.1.5 audit statement
October 5, 2025	1.9.0	Updated recommendation 4.1.3 audit statement
October 5, 2025	1.9.0	Updated recommendation 4.1.2 audit statement
October 5, 2025	1.9.0	Updated recommendation 4.1.1 audit statement
October 1, 2025	1.9.0	Updated recommendation 3.1.4 audit statement
September 25, 2025	1.9.0	Updated recommendation 5.10.4 (Ticket #25954)
September 25, 2025	1.9.0	Updated recommendation 5.8.2 (Ticket #25915)

Date	Version	Changes for this version
September 25, 2025	1.9.0	Updated recommendation 5.6.7 (Ticket #25510)
September 25, 2025	1.9.0	Updated recommendation 3.1.3 (Ticket #25739)
June 1, 2025	1.8.0	Added and tested support for the latest Kubernetes Cluster version 1.32
May 30, 2025	1.8.0	Updated recommendation 5.6.4 (Ticket 24845)
May 30, 2025	1.8.0	Updated recommendation 5.6.2 (Ticket 24736)
May 30, 2025	1.8.0	Updated recommendation 5.6.6 (Ticket 24953)
May 30, 2025	1.8.0	Updated recommendation 5.6.3 (Ticket 24832)
May 30, 2025	1.8.0	Updated recommendation 5.6.5 (Ticket 24846)
May 30, 2025	1.8.0	Updated recommendation 5.6.1 (Ticket 24728)
May 12, 2024	1.8.0	Updated recommendation 4.6.1 (Ticket 23808)
May 11, 2024	1.8.0	Updated recommendation 5.6.2 (Ticket 24745)
May 5, 2025	1.8.0	Updated recommendation 5.5.3 (Ticket 24256)
May 5, 2025	1.8.0	Updated recommendation 5.3.1 (Ticket 24183)

Date	Version	Changes for this version
May 5, 2025	1.8.0	Updated recommendation 5.4.1 (Ticket 24232)
May 5, 2025	1.8.0	Updated recommendation 5.6.5 (Ticket 24847)
May 5, 2025	1.8.0	Updated recommendation 5.5.4 (Ticket 24259)
April 15, 2025	1.8.0	Edited recommendations in section 5.5 (Ticket 23807)
April 15, 2025	1.8.0	Edited recommendation 3.1.1 (Ticket 23939)
April 15, 2025	1.8.0	Edited recommendation 3.1.2 (Ticket 23940)
November 12, 2024	1.7.0	Edited recommendation 5.8.1
November 12, 2024	1.7.0	Added and tested support for the latest Kubernetes Cluster version/s
November 1, 2024	1.7.0	Removed recommendation 2.1.1
August, 27, 2024	1.6.1	Bug Fix Update
May 30, 2024	1.6.0	Removed recommendation 5.10.5 (Ticket 21588)

Date	Version	Changes for this version
June 1, 2024	1.6.0	Modified recommendations to automate AAC 2.1.2 3.1.1 3.1.2 3.1.3 3.1.4 4.1.6 4.3.2 4.4.1 4.6.2 4.6.4 5.1.4 5.2.2 5.6.7 5.10.3
May 30, 2024	1.6.0	Added recommendation 4.1.10 Avoid non-default bindings to system:authenticated
May 30, 2024	1.6.0	Added recommendation 4.1.9 Avoid non-default bindings to system:unauthenticated
May 30, 2024	1.6.0	Added recommendation 4.1.8 Avoid bindings to system:anonymous
May 10, 2040	1.6.0	Rather than specifying just Pod Security Admission, this control should require the cluster have an appropriate mechanism to ensure compliance with Pod Security Standards Baseline profile. (Ticket 21157)
May 10, 2024	1.6.0	Merged recommendation 5.6.7 with 4.3 (Ticket 21769)
May 9, 2024	1.6.0	Edited level 1 profile definition (Ticket 21599)

Date	Version	Changes for this version
April 22, 2024	1.6.0	Removed legacy Compute Engine Instance metadata control (Ticket 21587)
Apr 21, 2023	1.5.0	Reference 5.6.7 not 6.6.7 (Ticket 18561)
Apr 21, 2023	1.5.0	Update references (Ticket 18562)
Apr 21, 2023	1.5.0	Update reference from 5.2 to 4.2 (Ticket 18563)
Apr 21, 2023	1.5.0	Edit recommendation Title (Ticket 18564)
Apr 21, 2023	1.5.0	Review output of DOXC - PDF and generation of extraneous subheads when expanding the TOC. (Ticket 18566)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 3.1.1 (Ticket 19758)
Sep 28, 2023	1.5.0	Proposed changes for GKE control 3.2.5 (--streaming-connection-idle-timeout) (Ticket 19782)
Sep 28, 2023	1.5.0	Proposed changes for GKE control 3.2.6 (--make-iptables-util-chains) (Ticket 19803)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 3.2.4 (Ticket 19764)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 3.2.2 (Ticket 19763)
Sep 28, 2023	1.5.0	UPDATE: Proposed change for control 3.2.1 (Ticket 19762)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 3.1.4 (Ticket 19761)

Date	Version	Changes for this version
Sep 28, 2023	1.5.0	UPDATE: Proposed change for control 3.1.3 (Ticket 19760)
Sep 28, 2023	1.5.0	UPDATE: Proposed change for control 3.1.2 (Ticket 19759)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 3.2.7 (Ticket 19765)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 3.2.8 (Ticket 19882)
Sep 28, 2023	1.5.0	Proposed changes for 'Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate' (Ticket 19917)
Sep 28, 2023	1.5.0	Proposed change to GKE Kubelet "--rotate-certificates" control (Ticket 19826)
Sep 28, 2023	1.5.0	UPDATE - Proposed changes for control 3.2.11 (Ticket 19852)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.1.1 (Ticket 19766)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.1.2 (Ticket 19767)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.1.3 (Ticket 19768)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.1.4 (Ticket 19769)
Sep 28, 2023	1.5.0	UPDATE: Proposed update to control 4.1.5 (Ticket 19770)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.1.6 (Ticket 19771)

Date	Version	Changes for this version
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.2.1 (Ticket 19772)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.2.2 (Ticket 19773)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.2.3 (Ticket 19774)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.2.4 (Ticket 19775)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.2.5 (Ticket 19918)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.2.6 (Ticket 19778)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.2.7 (Ticket 19919)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.2.8 (Ticket 19779)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.3.1 (Ticket 19776)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.3.2 (Ticket 19780)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.4.1 (Ticket 19781)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.4.2 (Ticket 19784)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.5.1 (Ticket 19785)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.6.1 (Ticket 19777)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.6.2 (Ticket 19786)

Date	Version	Changes for this version
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 4.6.3 (Ticket 19787)
Sep 28, 2023	1.5.0	UPDATE: Proposed change to control 4.6.4 (Ticket 19788)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.1.1 (Ticket 19789)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.1.2 (Ticket 19790)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.1.3 (Ticket 19792)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.1.4 (Ticket 19810)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.2.1 (Ticket 19793)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.2.2 (Ticket 19794)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.3.1 (Ticket 19797)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.4.2 (Ticket 19811)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.5.1 (Ticket 19812)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.5.2 (Ticket 19798)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.5.3 (Ticket 19799)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.5.4 (Ticket 19800)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.5.5 (Ticket 19801)

Date	Version	Changes for this version
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.5.5 (Ticket 19802)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.5.6 (Ticket 19804)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.5.7 (Ticket 19813)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.6.1 (Ticket 19814)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.6.3 (Ticket 19920)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.6.4 (Ticket 19922)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.6.5 (Ticket 19921)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.6.6 (Ticket 19815)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.6.7 (Ticket 19805)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.6.8 (Ticket 19816)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.7.2 (Ticket 19817)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.8.3 (Ticket 19818)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.9.1 (Ticket 19806)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.10.1 (Ticket 19807)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.10.3 (potential removal?) (Ticket 19808)

Date	Version	Changes for this version
Sep 28, 2023	1.5.0	UPDATE: Proposed changes for control 5.10.4 (Ticket 19821)
Sep 28, 2023	1.5.0	UPDATE: Proposed changes to control 5.10.6 (potential removal/reworking?) (Ticket 19809)
Oct 4, 2023	1.5.0	UPDATE: Additional proposed changes for control 5.6.3 (Ticket 19931)
Oct 5, 2023	1.5.0	UPDATE: Proposed changes to control 5.7.1 (Ticket 19945)
Oct 5, 2023	1.5.0	UPDATE: Proposed changes to control 5.8.4 (Ticket 19957)
Oct 5, 2023	1.5.0	UPDATE: Proposed changes to control 5.10.5 (Ticket 19946)
Oct 5, 2023	1.5.0	Recommend that control 'Ensure Basic Authentication using static passwords is Disabled' be removed (Ticket 19954)
Oct 5, 2023	1.5.0	DELETE: Consider deleting/removing control 5.8.2. given the deprecation of basic auth (Ticket 19958)
Oct 6, 2023	1.5.0	Proposed change for Ensure Basic Authentication using static passwords is Disabled (Ticket 19961)
Oct 11, 2023	1.5.0	Update: Control 5.1.2 - change drop down to drop-down (Ticket 19969)
Oct 11, 2023	1.5.0	UPDATE - Control 5.2.1, change drop down to drop-down (Ticket 19970)

Date	Version	Changes for this version
Oct 11, 2023	1.5.0	UPDATE: Control 5.2.2, change e.g to for example (Ticket 19971)
Oct 11, 2023	1.5.0	UPDATE: Couple of changes to control 5.4.2 (Ticket 19975)
Oct 11, 2023	1.5.0	UPDATE: Control 5.5.4, change drop down to drop-down (Ticket 19972)
Oct 11, 2023	1.5.0	UPDATE: Proposed changes for control 5.6.2 (Ticket 19973)
Oct 11, 2023	1.5.0	UPDATE: Control 5.7.1, change drop down to drop-down (Ticket 19974)
Oct 11, 2023	1.5.0	UPDATE: Proposed changes for control 5.8.1 (Ticket 19968)

Date	Version	Changes for this version
Apr 9, 2023	1.4.0	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	1.4.0	Ticket #16491 Edited recommendation 5.2.10
Apr 3, 2023	1.4.0	Ticket # 16624 Updated Recommendation 5.3.2 moved flags in audit process to end of the command line
Apr 3, 2023	1.4.0	Ticket # 16625Updated Recommendation 5.7.1 default value statement.
Apr 3, 2023	1.4.0	Ticket #18522 Set Pod Security Policy Recommendations to Manual in preparation for PSP removal in v1.25 and beyond.
Mar 22, 2023	1.4.0	Ticket #17029 Recommendation 4.2.1 Minimize the admission of privileged containers is deprricated.

Date	Version	Changes for this version
Mar 13, 2023	1.4.0	Ticket #16686 Updated recommendation 4.2.4 to resolve conflict with 4.2.8
Mar 13, 2023	1.4.0	Ticket # 15917 edited –event-qps option with eventRecordQPS