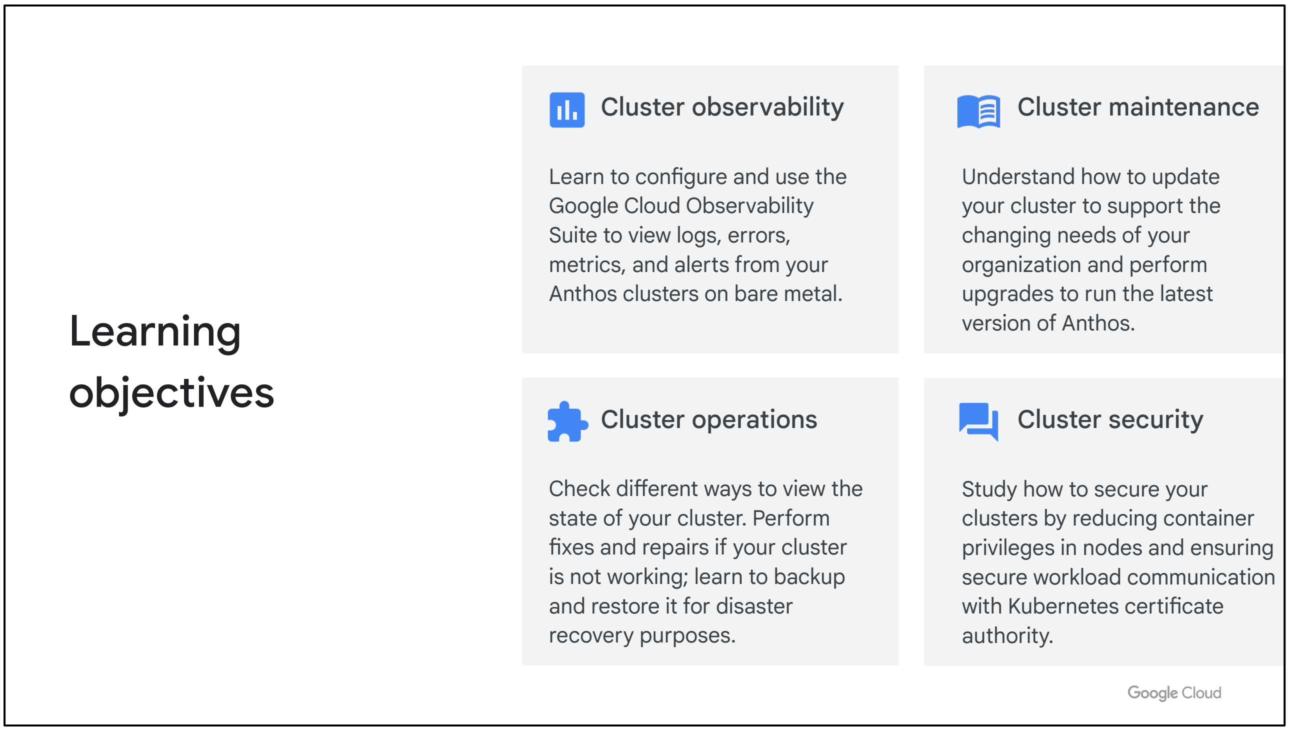
Google Cloud

Operating and Managing Clusters

Welcome to Operating and Managing Clusters.



Learning objectives

Cluster observability

Learn to configure and use the Google Cloud Observability Suite to view logs, errors, metrics, and alerts from your Anthos clusters on bare metal.

Cluster maintenance

Understand how to update your cluster to support the changing needs of your organization and perform upgrades to run the latest version of Anthos.

Cluster operations

Check different ways to view the state of your cluster. Perform fixes and repairs if your cluster is not working; learn to backup and restore it for disaster recovery purposes.

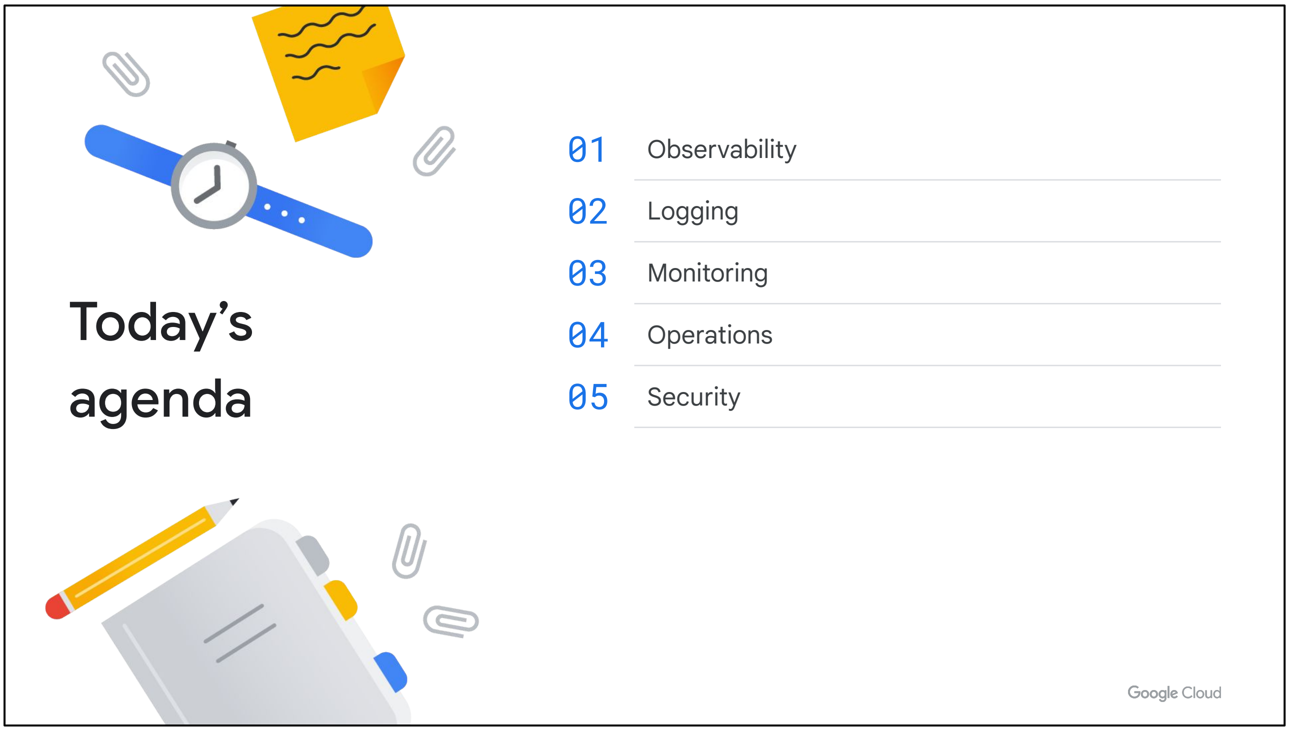
Cluster security

Study how to secure your clusters by reducing container privileges in nodes and ensuring secure workload communication with Kubernetes certificate authority.

Google Cloud

In this module, you learn how to:

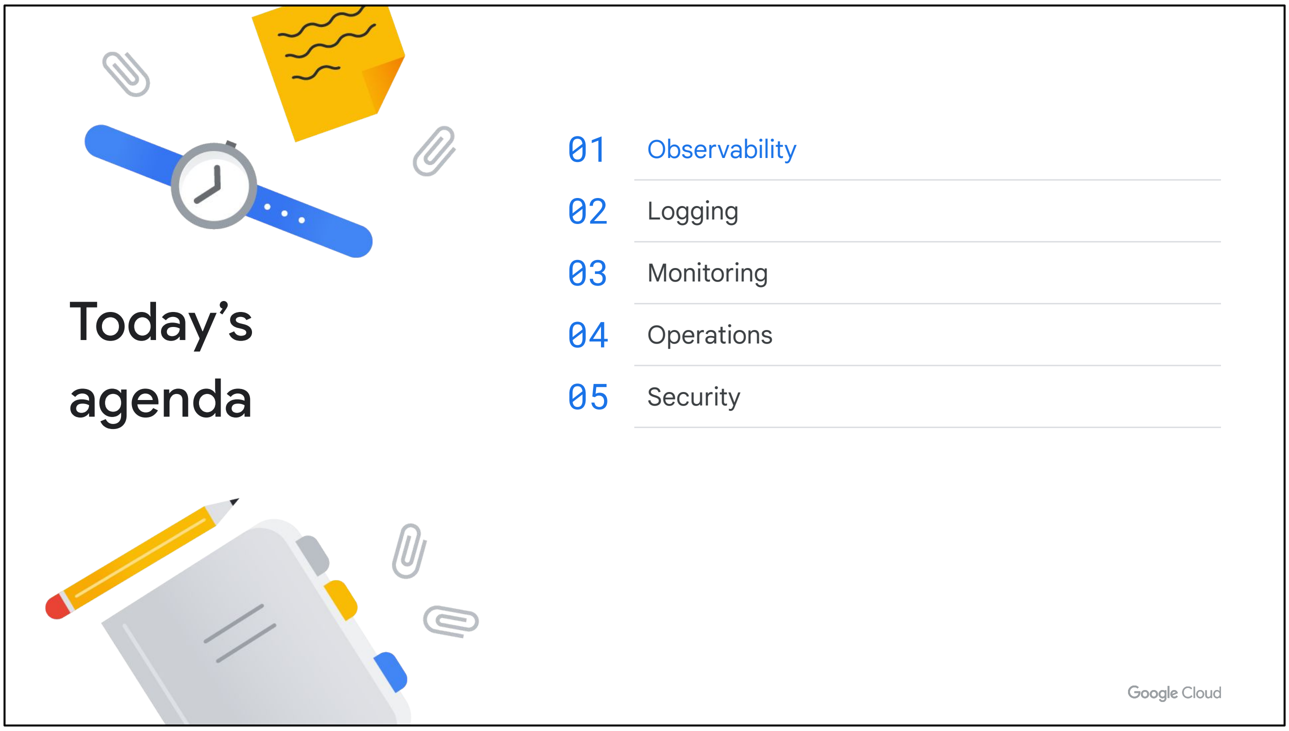
- Leverage Cloud Logging and Monitoring to track and respond to Anthos on bare metal cluster logs and metrics.
- Update your Anthos on bare metal clusters to enable new features, and upgrade clusters with new Anthos releases.
- Back up, restore, and troubleshoot Anthos on bare metal clusters.
- Secure your Anthos on bare metal clusters.



Today's agenda

Google Cloud

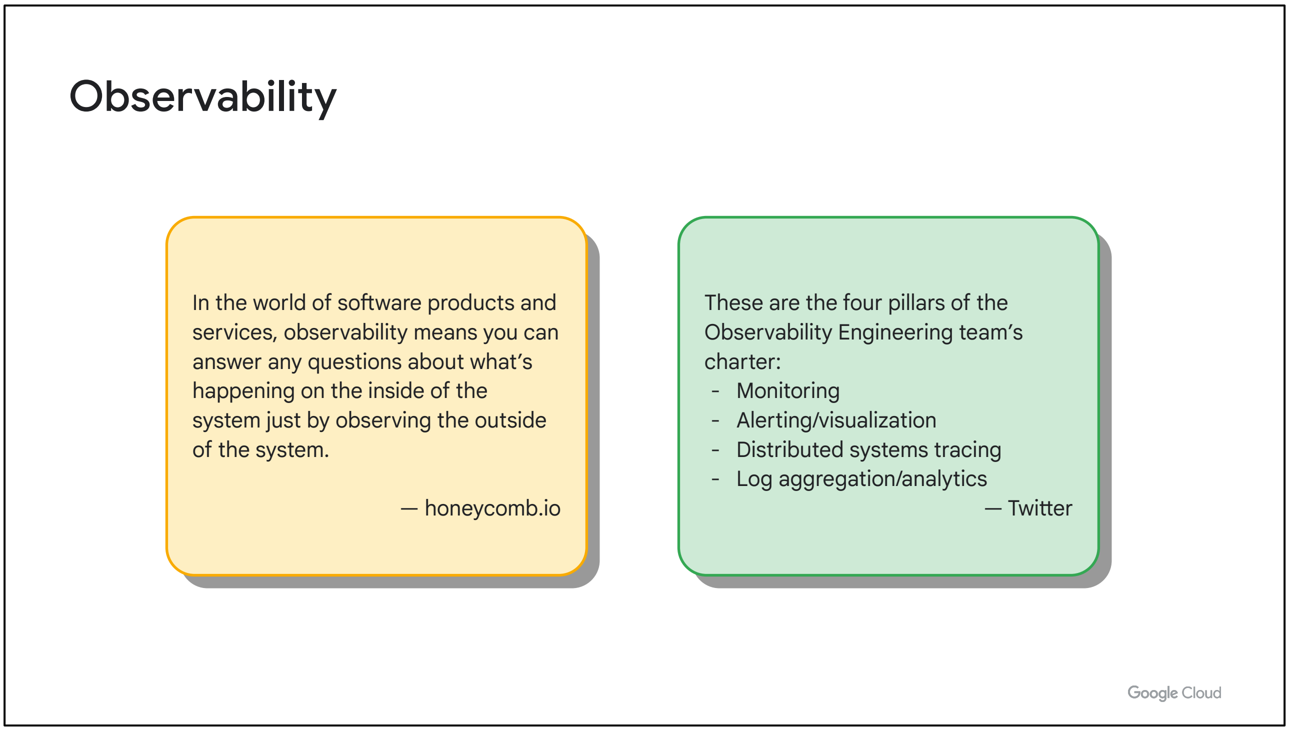
Here is our agenda.



Today's agenda

Google Cloud

We start with configuring and using observability features in Anthos.



Observability

In the world of software products and services, observability means you can answer any questions about what's happening on the inside of the system just by observing the outside of the system.

— honeycomb.io

These are the four pillars of the Observability Engineering team's charter:

- Monitoring
- Alerting/visualization
- Distributed systems tracing
- Log aggregation/analytics

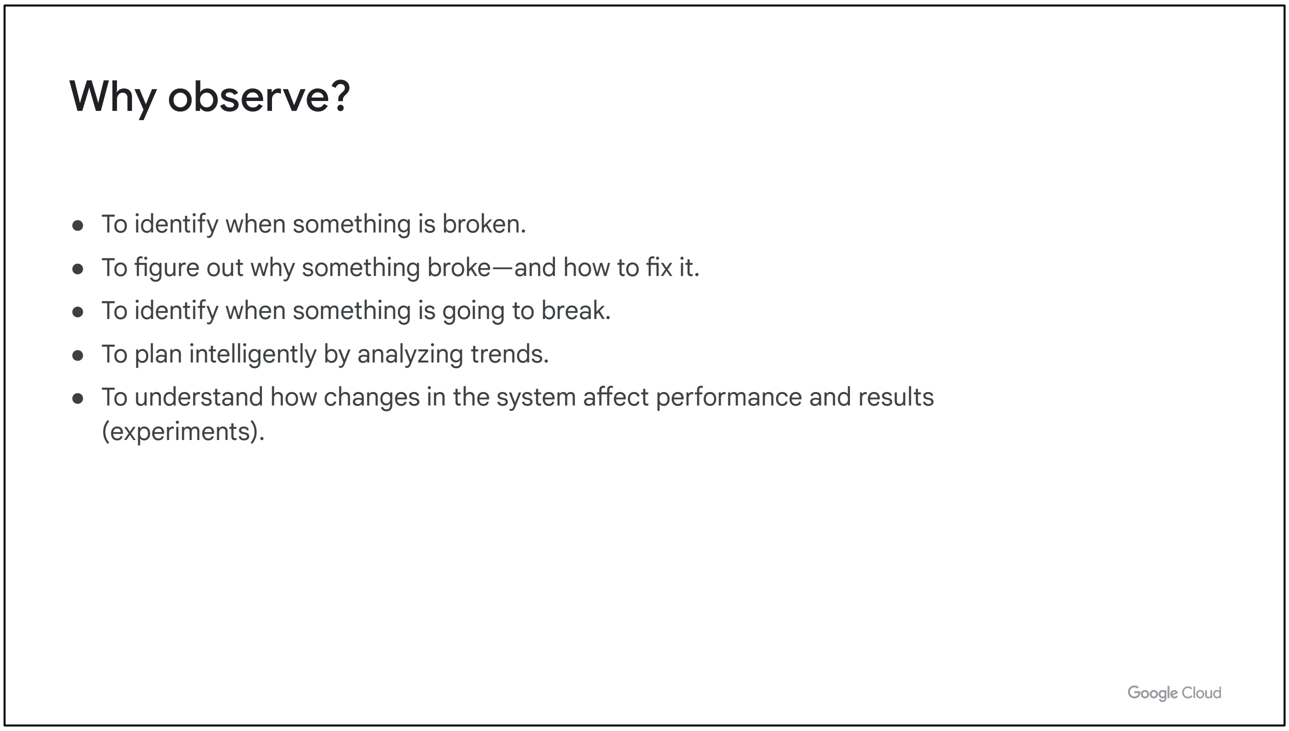
— Twitter

Google Cloud

What do we mean by observability?

The goal of observability is understanding how your services are performing, and why.

In most cases the term observability is used to mean a collection of products and processes that address: metrics, logs, traces, dashboards, alerts, etc.

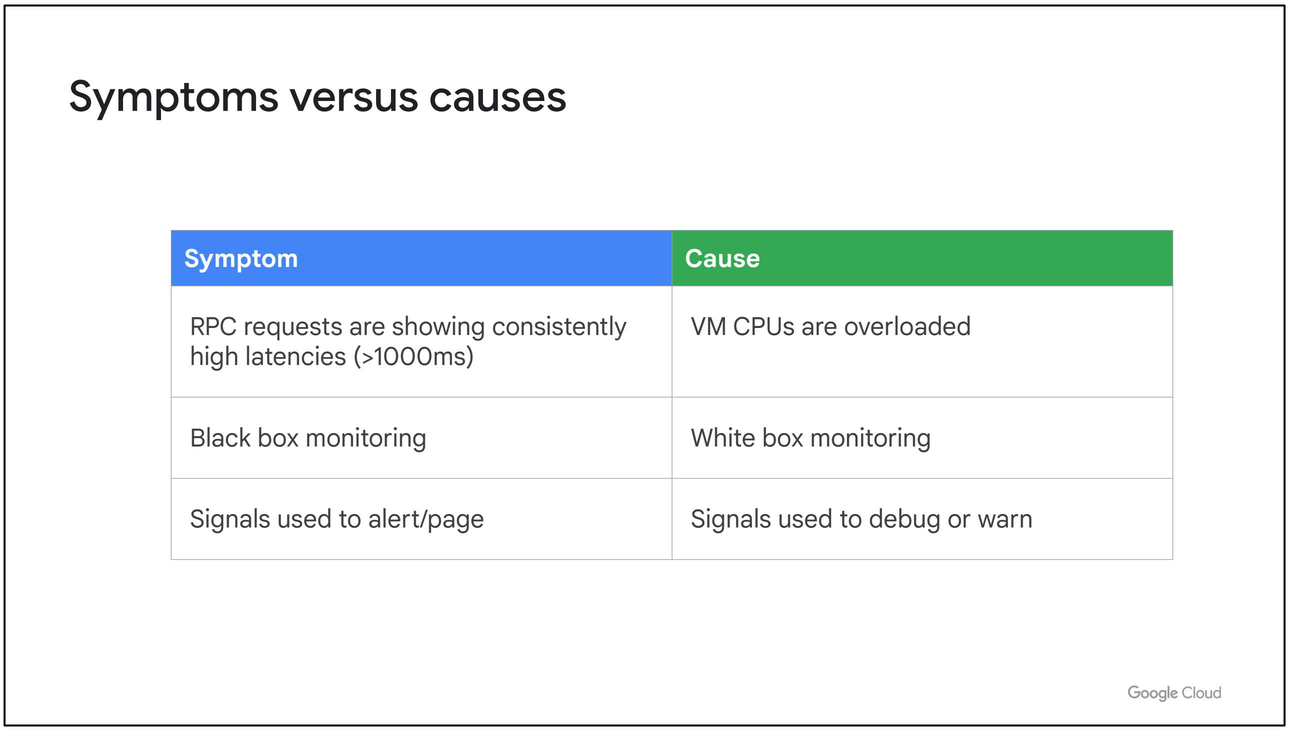


Why observe?

- To identify when something is broken.
- To figure out why something broke—and how to fix it.
- To identify when something is going to break.
- To plan intelligently by analyzing trends.
- To understand how changes in the system affect performance and results (experiments).

Google Cloud

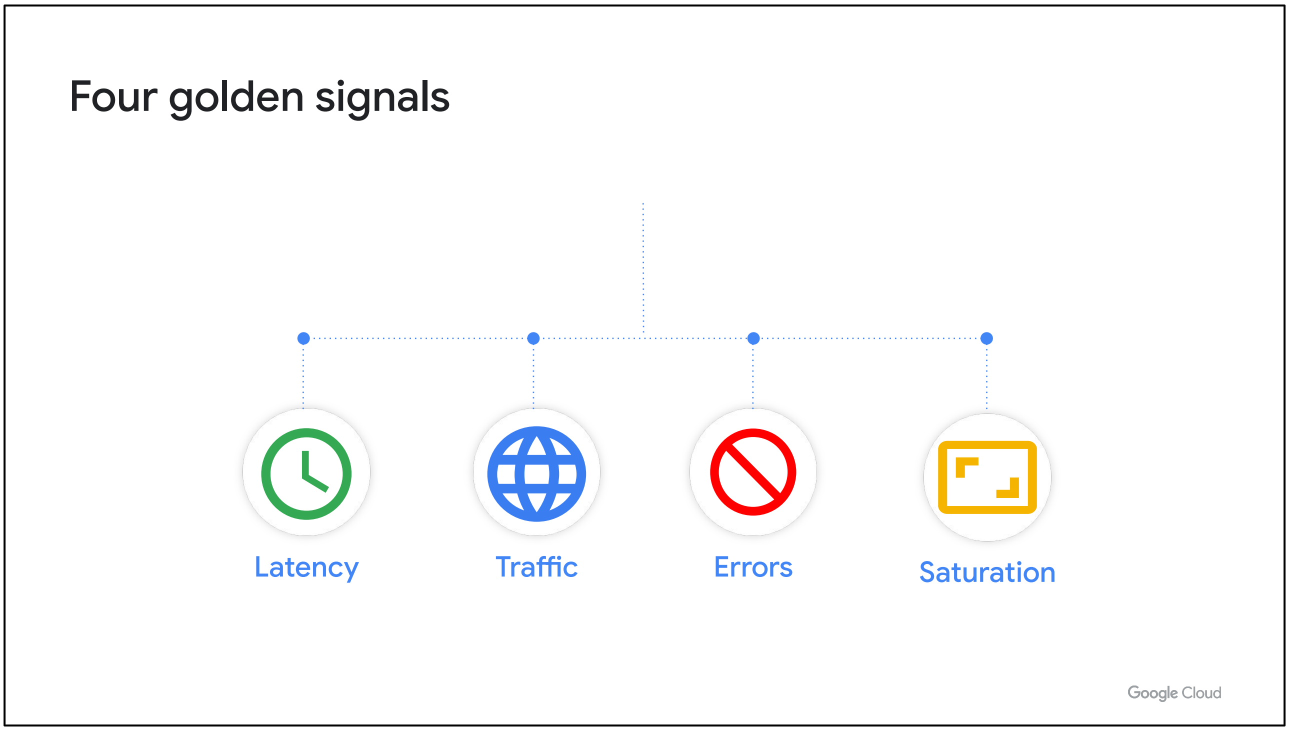
Observability solutions allow teams to recognize when services are broken and what has caused the failure. They can also help teams predict yet-to-happen failures, do capacity planning, and understand how product changes might affect performance.


Symptoms versus causes

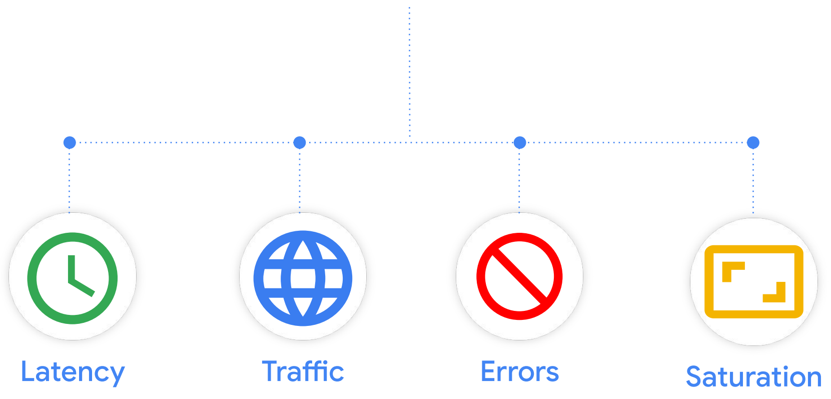
Symptom	Cause
RPC requests are showing consistently high latencies (>1000ms)	VM CPUs are overloaded
Black box monitoring	White box monitoring
Signals used to alert/page	Signals used to debug or warn

Google Cloud

Observability typically encompasses external signals about your service's performance (symptoms) and internal signals about your service's operation (things like resource utilization) that can be helpful in understanding why the performance is what it is. A symptom might be something like slow page RPCs—this doesn't require any internal tooling on the application, but it's important and might actually trigger alerts to on-call engineers. A cause for slow RPC might be that the server CPUs are overloaded—this kind of signal is typically monitored and used to warn or debug.



Four golden signals



Google Cloud

Google's Site Reliability Engineering, or SRE, team has identified what it considers to be the four golden signals.

The first is latency. It's basically the time it takes for your application to service a request.

The second is traffic. It is a measure of how much demand is being placed on your system. It can be, for instance, the number of HTTP requests per second, or the number of concurrent sessions in your database.

Errors represent the rate of requests that fail, either explicitly (like 500s errors), or implicitly (200 responses but with incorrect content).

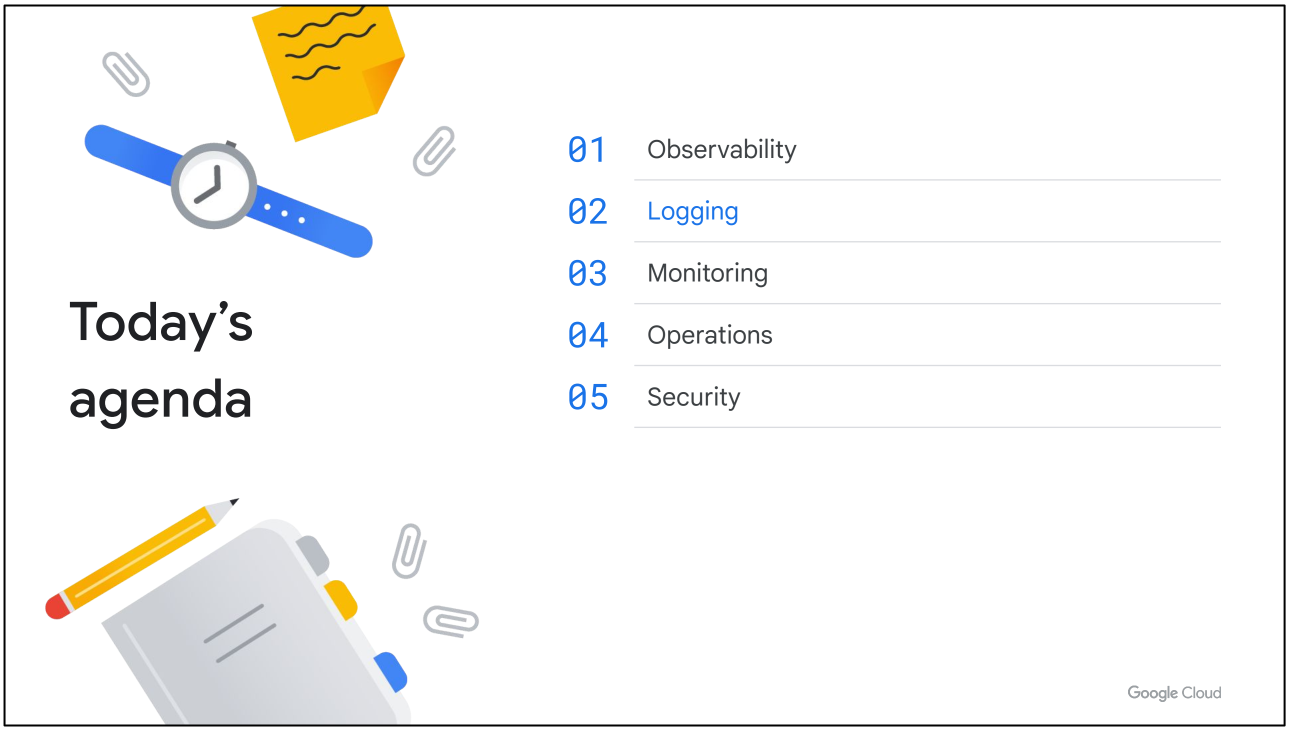
Saturation represents how "full" your service is. To be able to use that, you should have a utilization target for your service. Be careful because some systems degrade in performance well before they achieve 100% utilization.

Logging and monitoring

- Unified interface for multiple environments.
- Analyze and monitor your operational telemetry.
- Set up appropriate performance and availability indicators.
- Use built-in observability to troubleshoot and improve your applications.

Google Cloud

Google Cloud's operations suite provides multiple observability tools including: Cloud Logging, Cloud Monitoring, Cloud Trace, and others. These tools integrate with your Anthos on bare metal clusters, providing control panels that allow you to manage resources across multiple clusters and multiple operating environments. You can use them to track performance, alert on failures, troubleshoot applications, etc.



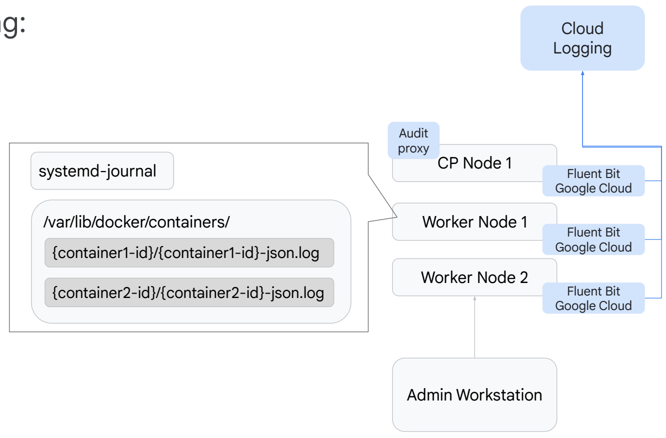
Today's agenda

Google Cloud

Let's dig a bit deeper into logging.

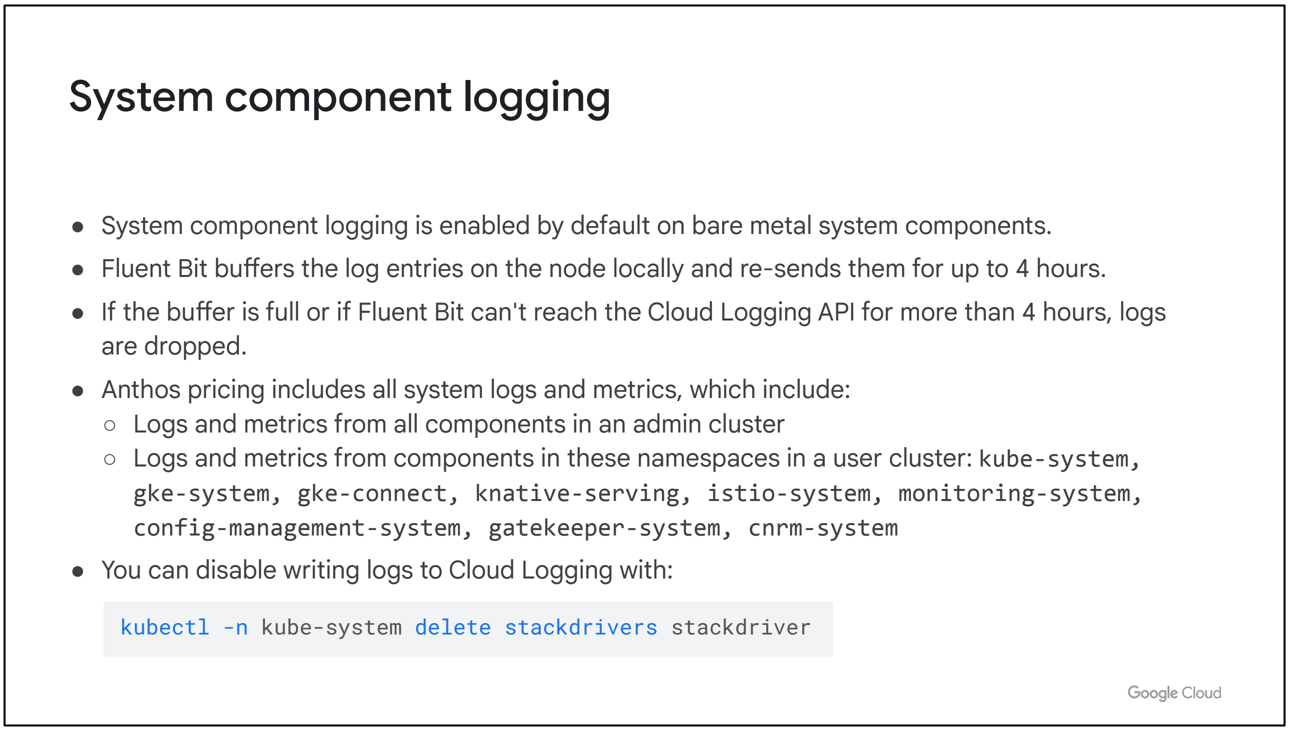
Logging in Anthos clusters on bare metal

- There are different types of logging:
  - Cluster logs
  - Application logs
  - Audit logs



Google Cloud

On each Anthos on bare metal cluster, a log forwarder and an audit proxy are installed; The log forwarder is implemented as a Fluent Bit daemonset. It scrapes logs and forwards them to the Cloud Logging service. It can collect and forward several types of logs. The audit-proxy is deployed as a daemonset that runs on the control plane nodes. It buffers and writes audit log entries to Cloud Audit Logs. Audit log entries include all Kubernetes API requests made to Anthos clusters on bare metal.



System component logging

- System component logging is enabled by default on bare metal system components.
- Fluent Bit buffers the log entries on the node locally and re-sends them for up to 4 hours.
- If the buffer is full or if Fluent Bit can't reach the Cloud Logging API for more than 4 hours, logs are dropped.
- Anthos pricing includes all system logs and metrics, which include:
  - Logs and metrics from all components in an admin cluster
  - Logs and metrics from components in these namespaces in a user cluster: `kube-system`, `gke-system`, `gke-connect`, `knative-serving`, `istio-system`, `monitoring-system`, `config-management-system`, `gatekeeper-system`, `cnrm-system`
- You can disable writing logs to Cloud Logging with:

```
kubectl -n kube-system delete stackdrivers stackdriver
```

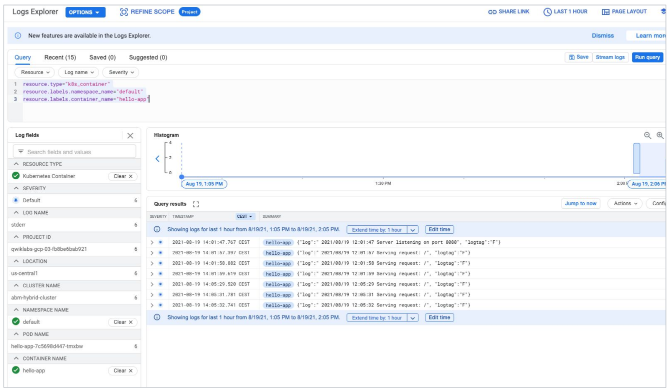
Google Cloud

By default, only system logs are collected. The log forwarder buffers logs locally, then forwards them to Cloud Logging. The logging agent will keep log events in the buffer for up to 4 hours, and if it can't deliver them in that time, they are dropped. The logs collected are for workloads in the kube-system, gke-system, gke-connect, istio-system, and config-management-system namespaces, and storing these logs for 30 days in Cloud Logging is free. If you want to disable writing logs to Cloud Logging because you've chosen to use another Security Information and Event Management, or SIEM, solution, you can do that by deleting the stackdriver object.

Application logging

- Access your application data remotely without direct line of sight to your cluster and after your pods have died by using Cloud Logging.
- Enable in the cluster config file:

```
clusterOperations:
  enableApplication: true
```



```
resource.type="k8s_container"
resource.labels.namespace_name="default"
resource.labels.container_name="hello-app"
```

RESOURCE TYPE	
Kubernetes Container	Clear
SEVERITY	
Default	6
LOG NAME	
stderr	6
PROJECT ID	
qwiklabs-gcp-03-fb8be6bab921	6
LOCATION	
us-central1	6
CLUSTER NAME	
abm-hybrid-cluster	6
NAMESPACE NAME	
default	Clear
POD NAME	
hello-app-7c5698d447-tmxbw	6
CONTAINER NAME	
hello-app	Clear

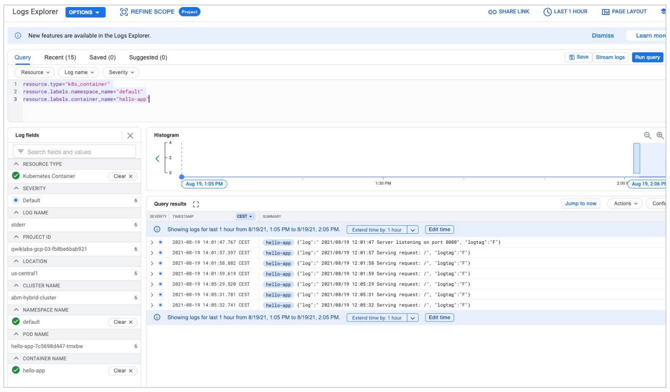
```
SEVERITY  TIMESTAMP              CEST  SUMMARY
Showing logs for last 1 hour from 8/19/21, 1:05 PM to 8/19/21, 2:05 PM.  Extend time by: 1 hour  Edit time
2021-08-19 14:01:47.767 CEST  hello-app {"log":" 2021/08/19 12:01:47 Server listening on port 8080", "logtag":"F"}
2021-08-19 14:01:57.397 CEST  hello-app {"log":" 2021/08/19 12:01:57 Serving request: /", "logtag":"F"}
2021-08-19 14:01:58.882 CEST  hello-app {"log":" 2021/08/19 12:01:58 Serving request: /", "logtag":"F"}
2021-08-19 14:01:59.619 CEST  hello-app {"log":" 2021/08/19 12:01:59 Serving request: /", "logtag":"F"}
2021-08-19 14:05:29.520 CEST  hello-app {"log":" 2021/08/19 12:05:29 Serving request: /", "logtag":"F"}
2021-08-19 14:05:31.781 CEST  hello-app {"log":" 2021/08/19 12:05:31 Serving request: /", "logtag":"F"}
2021-08-19 14:05:32.741 CEST  hello-app {"log":" 2021/08/19 12:05:32 Serving request: /", "logtag":"F"}
Showing logs for last 1 hour from 8/19/21, 1:05 PM to 8/19/21, 2:05 PM.  Extend time by: 1 hour  Edit time
```

Google Cloud

You can enable application logging on user clusters, which will scrape the logs from all the containers running on your cluster, and make them visible in Cloud Logging. Google recommends this for apps that run only on bare metal clusters, or are spread across GKE and bare metal clusters. For applications with components running on Anthos clusters on bare metal and traditional on-premises infrastructure, you might consider other solutions for an end-to-end view of those applications. This isn't enabled by default, but is easy to configure in your cluster config file.

Application logging in Cloud Logging

- Ingestion and storage of application logs in Cloud Logging is not included in Anthos pricing.
- Log ingestion costs $0.50/GiB and includes <30 days of storage.
- Default log retention is 30 days.
- Logs can be retained for up to 3650 days for $0.01/GiB/month.
- Cloud Logging can also route logs to other services such as:
  - Cloud Storage
  - BigQuery
  - Pub/Sub



New features are available in the Logs Explorer.

```
resource.type="k8s_container"
resource.labels.namespace_name="default"
resource.labels.container_name="hello-app"
```

RESOURCE TYPE	
Kubernetes Container	
SEVERITY	
Default	6
LOG NAME	
stderr	6
PROJECT ID	
qwiklabs-gcp-03-fb8be6bab921	6
LOCATION	
us-central1	6
CLUSTER NAME	
abm-hybrid-cluster	6
NAMESPACE NAME	
default	
POD NAME	
hello-app-7c5698d447-tmxbw	6
CONTAINER NAME	
hello-app	

Showing logs for last 1 hour from 8/19/21, 1:05 PM to 8/19/21, 2:05 PM.

```
SEVERITY  TIMESTAMP                    CEST  SUMMARY
2021-08-19 14:01:47.767 CEST  hello-app  {"log":" 2021/08/19 12:01:47 Server listening on port 8080", "logtag":"F"}
2021-08-19 14:01:57.397 CEST  hello-app  {"log":" 2021/08/19 12:01:57 Serving request: /", "logtag":"F"}
2021-08-19 14:01:58.882 CEST  hello-app  {"log":" 2021/08/19 12:01:58 Serving request: /", "logtag":"F"}
2021-08-19 14:01:59.619 CEST  hello-app  {"log":" 2021/08/19 12:01:59 Serving request: /", "logtag":"F"}
2021-08-19 14:05:29.520 CEST  hello-app  {"log":" 2021/08/19 12:05:29 Serving request: /", "logtag":"F"}
2021-08-19 14:05:31.781 CEST  hello-app  {"log":" 2021/08/19 12:05:31 Serving request: /", "logtag":"F"}
2021-08-19 14:05:32.741 CEST  hello-app  {"log":" 2021/08/19 12:05:32 Serving request: /", "logtag":"F"}
```

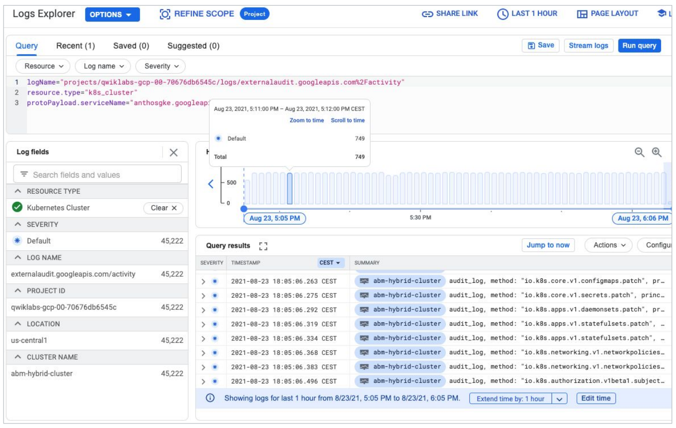
Showing logs for last 1 hour from 8/19/21, 1:05 PM to 8/19/21, 2:05 PM.

Google Cloud

Writing logs to and storing them in Cloud Logging is billable and not included in your Anthos subscription. Google charges fifty cents per GB to write the log files into Cloud Logging and store them for 30 days. While default retention is 30 days, you can configure the service to retain logs for up to 10 years, with storage charged at one cent per GB per month. Logs ingested into Cloud Logging can also be routed to other services in Google Cloud, or even outside of Google Cloud.

Audit logging with Cloud Logging

- Centralize audit logs for all Anthos clusters.
- Log entries written to Cloud Audit Logs are immutable.
- Cloud Audit Logs entries are retained for 400 days.
- Cloud Audit Logs is included in the price of Anthos.



```
logName="projects/qwiklabs-gcp-00-70676db6545c/logs/externalaudit.googleapis.com%2Factivity"
resource.type="k8s_cluster"
protoPayload.serviceName="anthosgke.googleap
```

Aug 23, 2021, 5:11:00 PM – Aug 23, 2021, 5:12:00 PM CEST

Default	749
Total	749

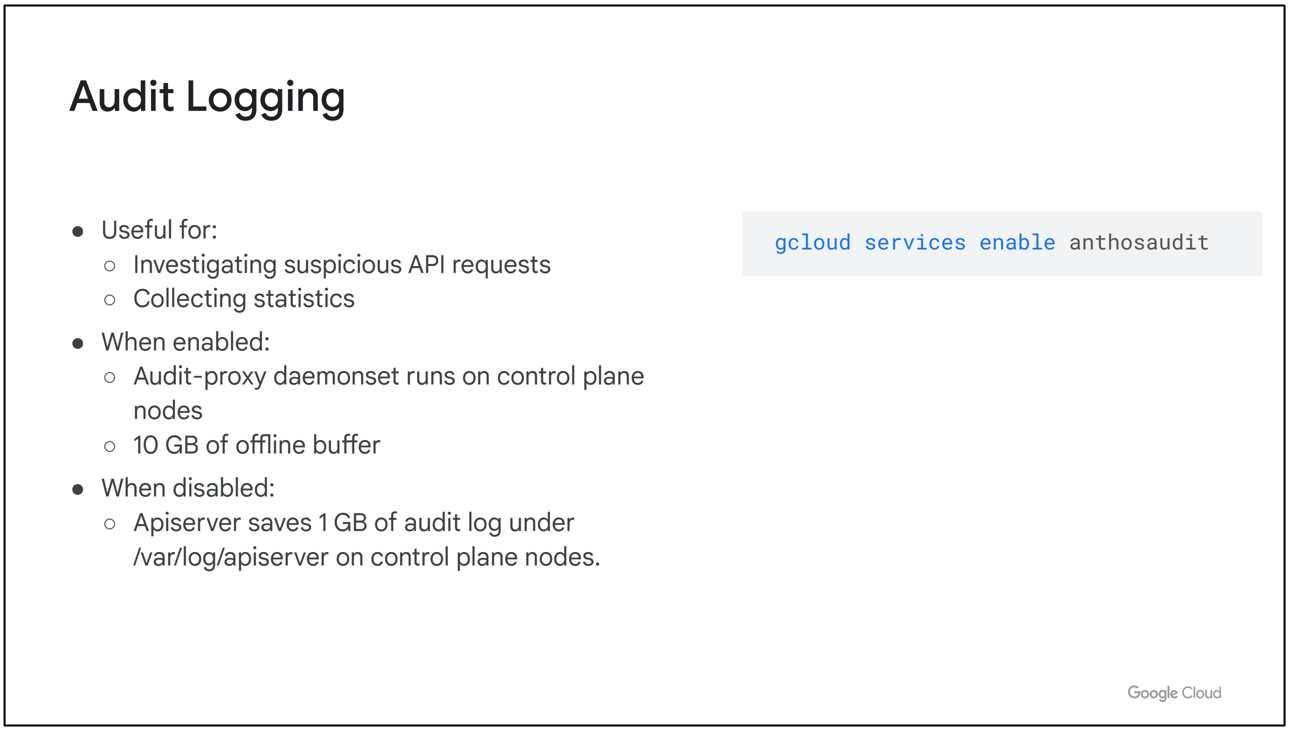
RESOURCE TYPE	
Kubernetes Cluster	Clear
SEVERITY	
Default	45,222
LOG NAME	
externalaudit.googleapis.com/activity	45,222
PROJECT ID	
qwiklabs-gcp-00-70676db6545c	45,222
LOCATION	
us-central1	45,222
CLUSTER NAME	
abm-hybrid-cluster	45,222

```
SEVERITY  TIMESTAMP                     CEST  SUMMARY
          2021-08-23 18:05:06.263 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.core.v1.configmaps.patch", pr…
          2021-08-23 18:05:06.275 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.core.v1.secrets.patch", princ…
          2021-08-23 18:05:06.292 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.apps.v1.daemonsets.patch", pr…
          2021-08-23 18:05:06.319 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.apps.v1.statefulsets.patch", …
          2021-08-23 18:05:06.334 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.apps.v1.statefulsets.patch", …
          2021-08-23 18:05:06.368 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.networking.v1.networkpolicies…
          2021-08-23 18:05:06.383 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.networking.v1.networkpolicies…
          2021-08-23 18:05:06.496 CEST  abm-hybrid-cluster  audit_log, method: "io.k8s.authorization.v1beta1.subject…
```

Showing logs for last 1 hour from 8/23/21, 5:05 PM to 8/23/21, 6:05 PM.

Google Cloud

To log all the API calls sent to the API server on a bare metal cluster, you can enable audit logging in your Anthos deployment. This will write log entries for API calls to Google Cloud Audit Logs. These logs are retained for 400 days and are immutable. There is no fee for using this feature, and it gives you an easy way to see a consolidated snapshot of activity across all your Anthos clusters.

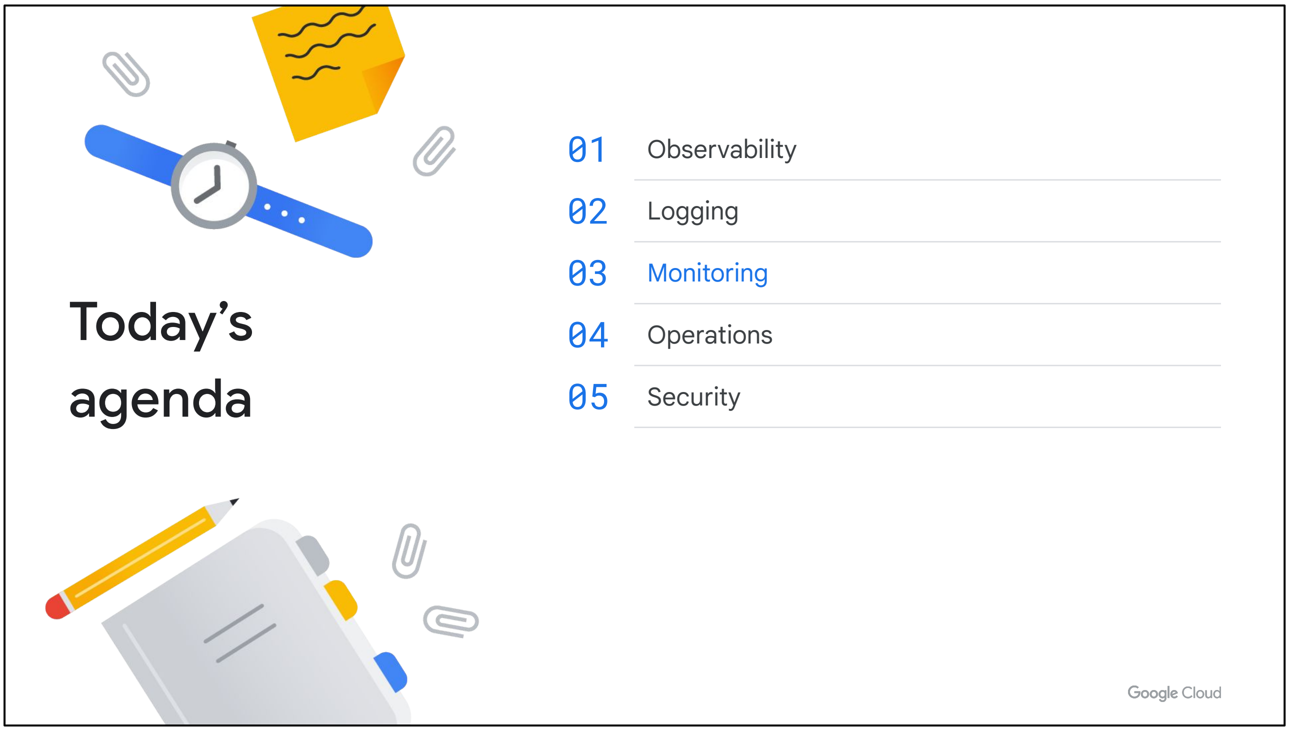


Audit Logging

- Useful for:
  - Investigating suspicious API requests
  - Collecting statistics
- When enabled:
  - Audit-proxy daemonset runs on control plane nodes
  - 10 GB of offline buffer
- When disabled:
  - Apiserver saves 1 GB of audit log under /var/log/apiserver on control plane nodes.

```
gcloud services enable anthosaudit
```

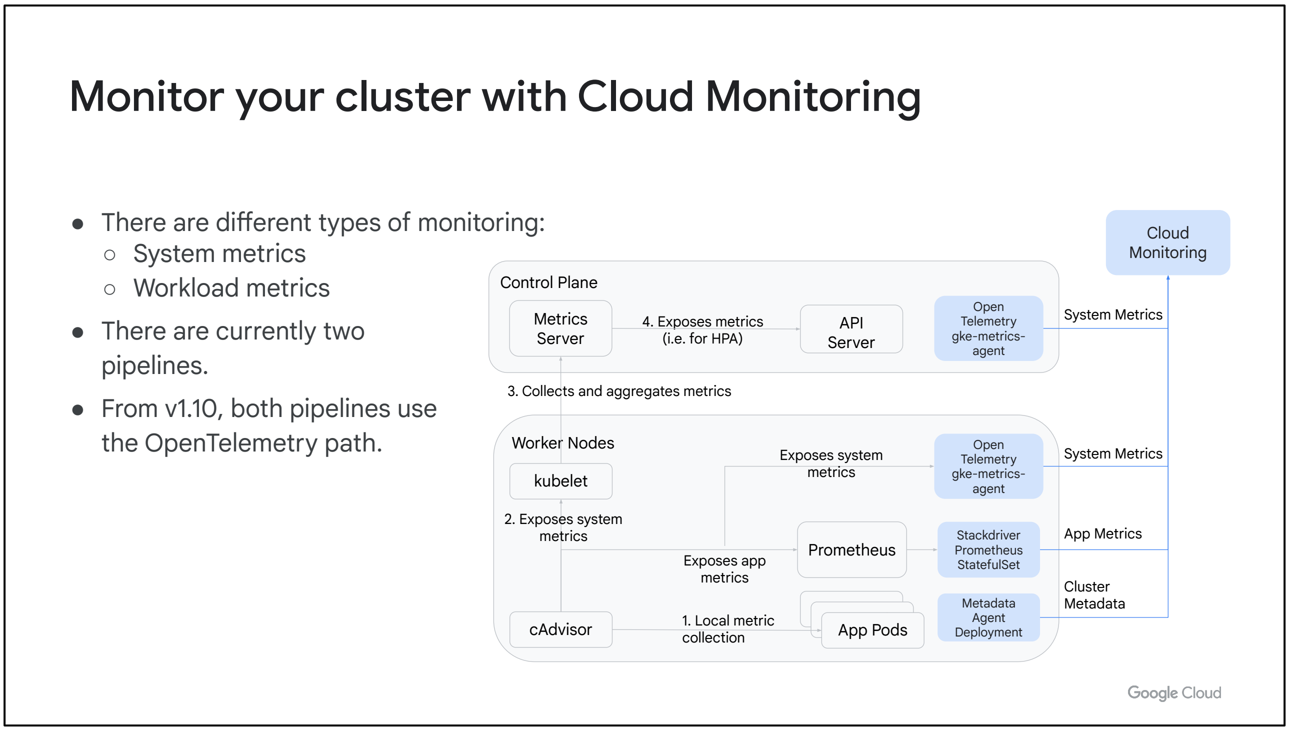
Anthos audit logs are useful for investigating suspicious activity and profiling use of your clusters. Audit logging is disabled by default through version 1.8; it's enabled by default in version 1.9. You can modify the config file when creating clusters with version 1.8 or earlier to enable the feature. Audit Logging buffers events locally until they can be delivered to the Logging service, with a 10 GB buffer. If the buffer fills, the oldest events are dropped. If audit logging is not enabled, then there is a 1 GB audit log written on the control plane nodes.



Today's agenda

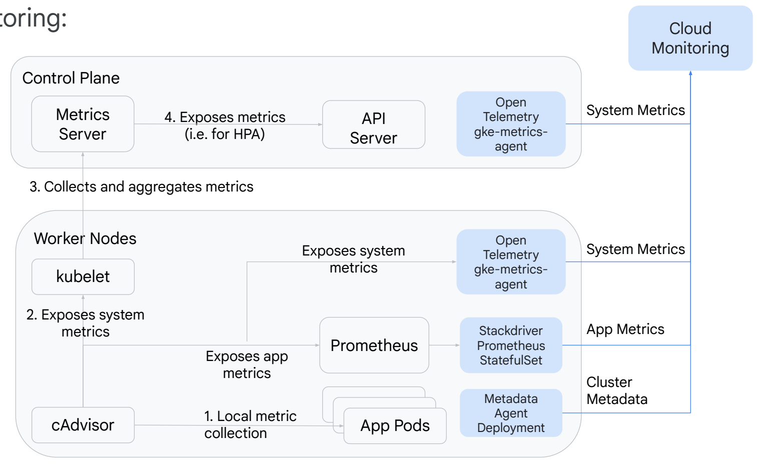
Google Cloud

Anthos bare metal clusters can also forward metrics to the Cloud Monitoring service; let's explore how that works.



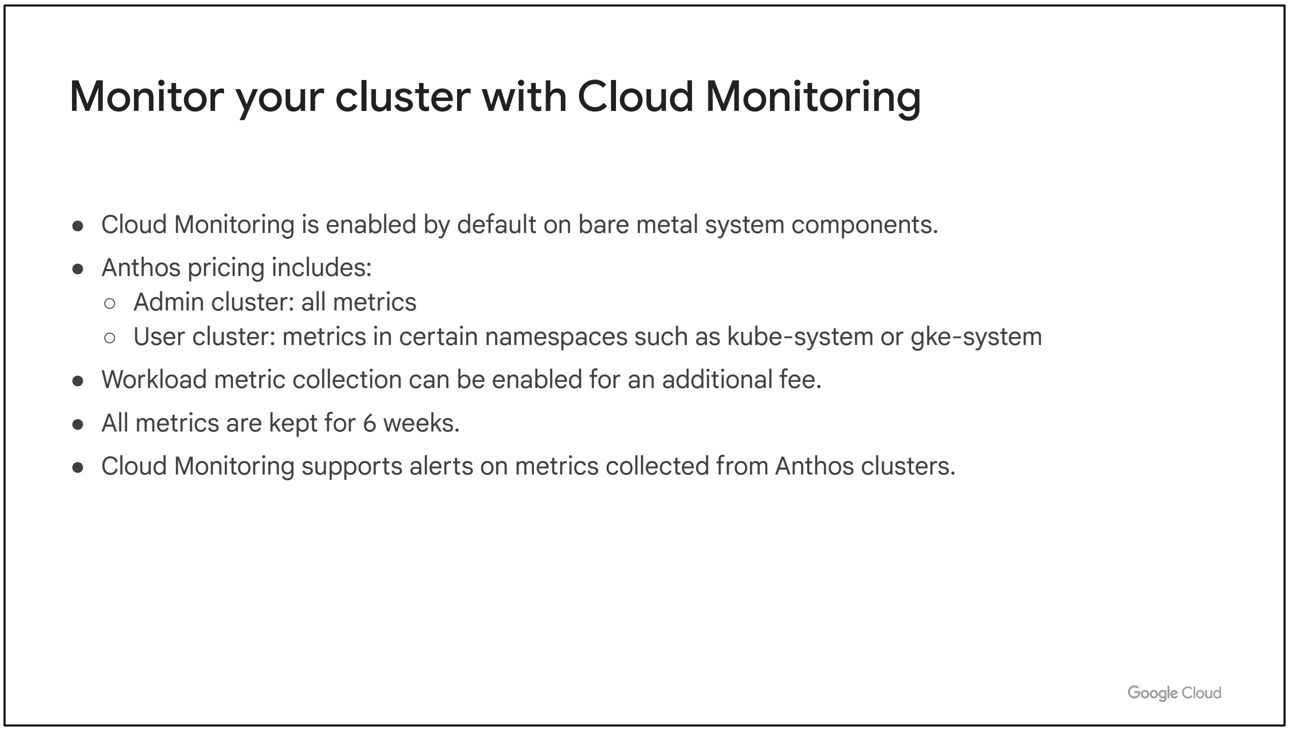
Monitor your cluster with Cloud Monitoring

- There are different types of monitoring:
  - System metrics
  - Workload metrics
- There are currently two pipelines.
- From v1.10, both pipelines use the OpenTelemetry path.


Cloud Monitoring
Control Plane
Metrics Server
4. Exposes metrics (i.e. for HPA)
API Server
Open Telemetry gke-metrics-agent
System Metrics
3. Collects and aggregates metrics
Worker Nodes
kubelet
Exposes system metrics
Open Telemetry gke-metrics-agent
System Metrics
2. Exposes system metrics
Exposes app metrics
Prometheus
Stackdriver Prometheus StatefulSet
App Metrics
Cluster Metadata
Metadata Agent Deployment
cAdvisor
1. Local metric collection
App Pods

Google Cloud

At the moment, Anthos on bare metal v1.9 has two pipelines to collect system and workload metrics. System metrics are collected by the Open Telemetry daemonset, which collects metrics directly from the cAdvisor and the Metrics Server. Kubernetes collects these metrics to manage the lifecycle of the workloads. For example, the HorizontalPodAutoscaler decides to autoscale based on the CPU utilization metric collected from the pods. Workload metrics are still collected using the legacy pipeline, which leverages Prometheus and a Prometheus-to-Stackdriver StatefulSet to collect, buffer, and forward metrics to Cloud Monitoring. App metrics are configured by the developers and depend on the application. In v1.10, it is planned that the two pipelines will be unified to use the Open Telemetry path, which has a 90% lower resource footprint, it's more reliable, has 6.5 GB of offline buffer and does not have any dependency on Persistent Volumes. In addition to metrics, there is a metadata agent component that is implemented as a Kubernetes deployment and is responsible for sending metadata objects such as namespaces, labels, or annotations to Google Cloud.

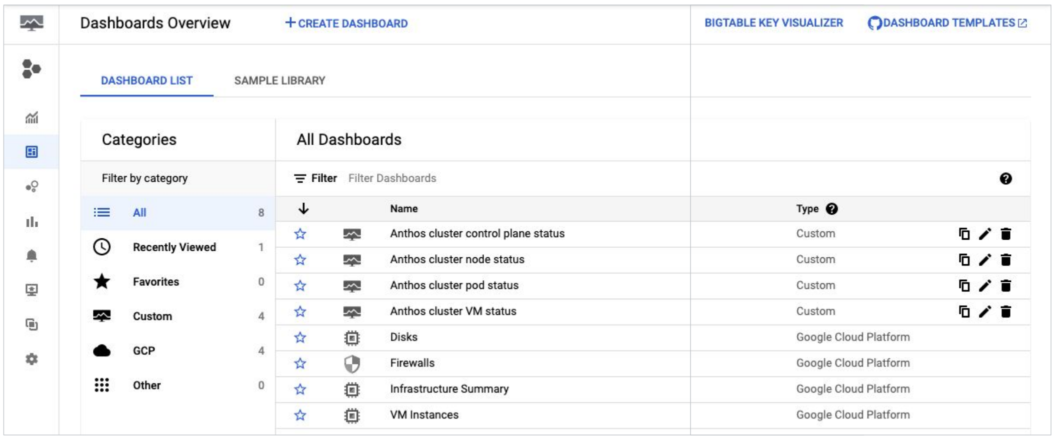


Monitor your cluster with Cloud Monitoring

- Cloud Monitoring is enabled by default on bare metal system components.
- Anthos pricing includes:
  - Admin cluster: all metrics
  - User cluster: metrics in certain namespaces such as kube-system or gke-system
- Workload metric collection can be enabled for an additional fee.
- All metrics are kept for 6 weeks.
- Cloud Monitoring supports alerts on metrics collected from Anthos clusters.

By default, Anthos bare metal system metrics are collected. System metrics include metrics from pods running in the admin cluster and the pods running in the user cluster in the kube-system, gke-system, gke-connect, istio-system, and config-management-system namespaces. All system metrics are included in the Anthos pricing. Additionally, you can enable the collection of workload metrics for an additional fee. All metrics are kept in Cloud Monitoring for 6 weeks. Once metrics are collected in Cloud Monitoring, you can enable Alerts to notify you when a threshold is reached.

Anthos creates Cloud Monitoring dashboards for you...



Categories

Filter by category

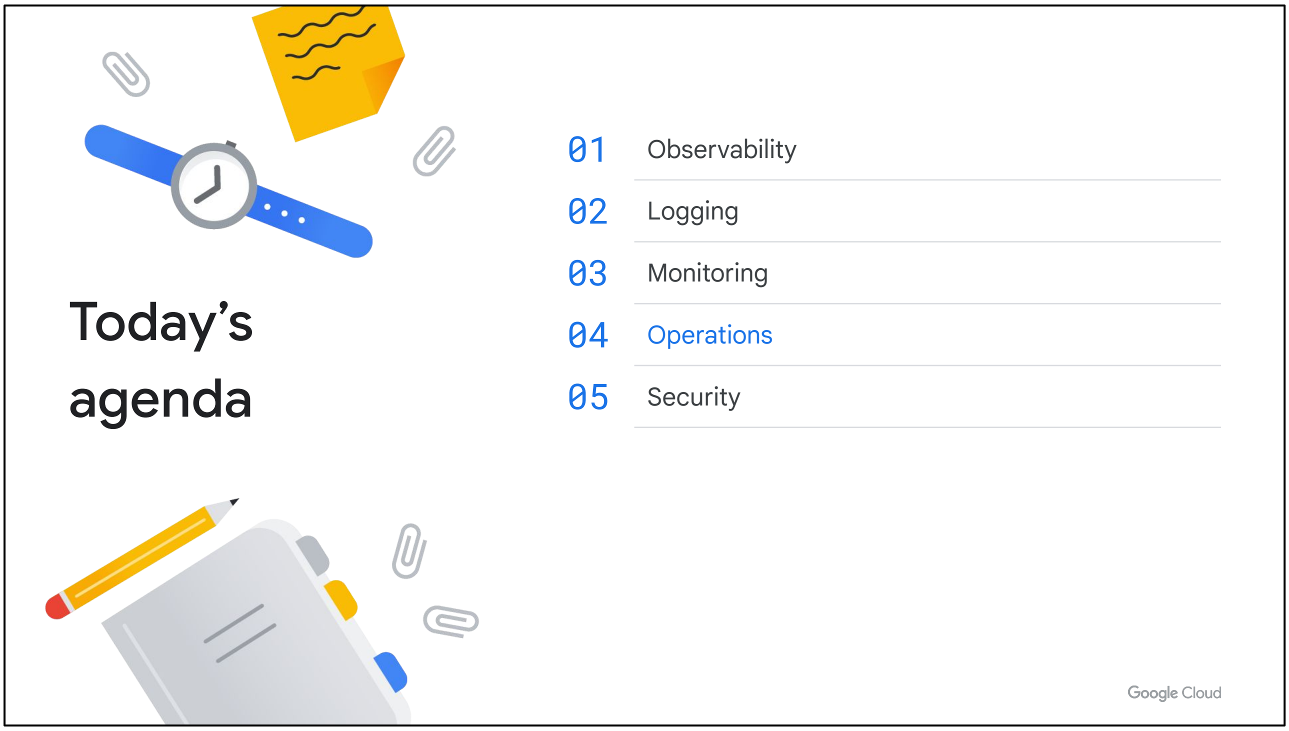
All	8
Recently Viewed	1
Favorites	0
Custom	4
GCP	4
Other	0

All Dashboards

Name	Type
Anthos cluster control plane status	Custom
Anthos cluster node status	Custom
Anthos cluster pod status	Custom
Anthos cluster VM status	Custom
Disks	Google Cloud Platform
Firewalls	Google Cloud Platform
Infrastructure Summary	Google Cloud Platform
VM Instances	Google Cloud Platform

Google Cloud

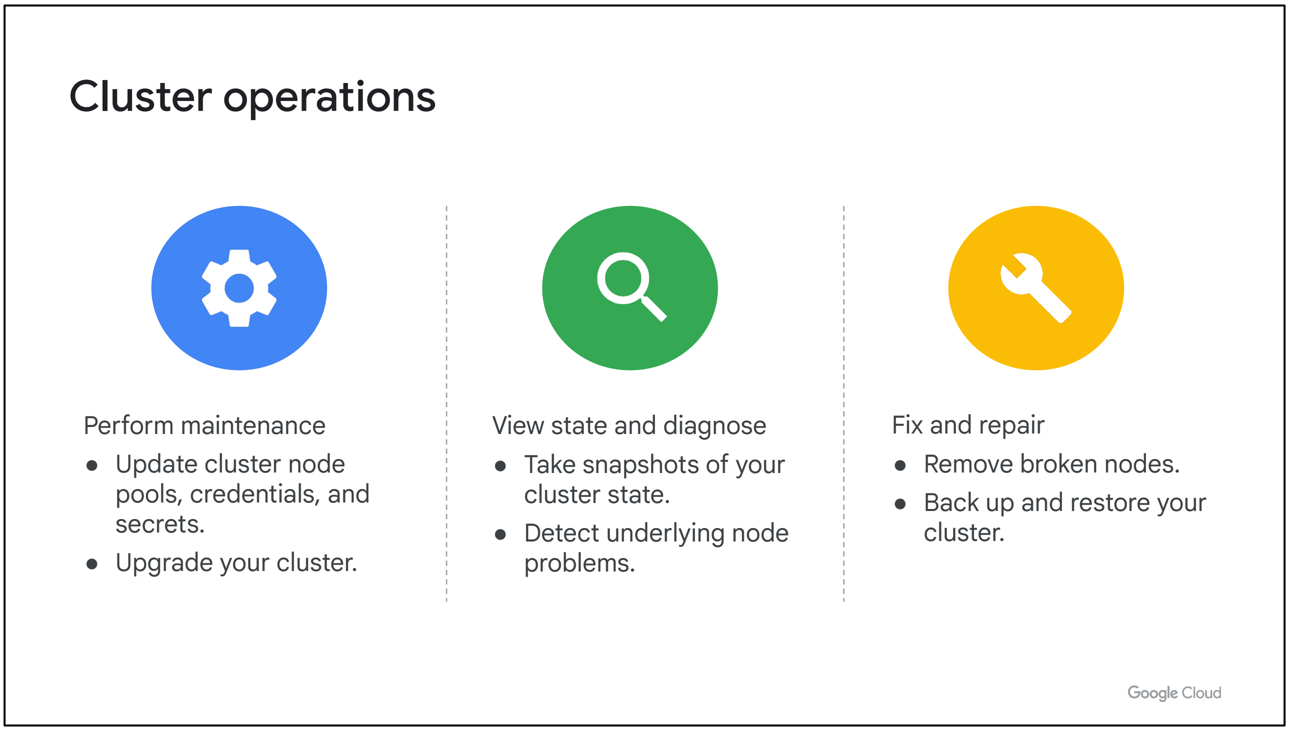
Once you enable metric collection, Anthos creates Cloud Monitoring dashboards for you, so that you can dive deep into the state of your Anthos cluster control plane nodes, worker nodes, pods, and cluster VMs.



Today's agenda

Google Cloud

Deploying new clusters is great, but once they're up and running, you have to actually manage them. Let's look into common operations activities for Anthos on bare metal clusters.



Cluster operations

Perform maintenance

- Update cluster node pools, credentials, and secrets.
- Upgrade your cluster.

View state and diagnose

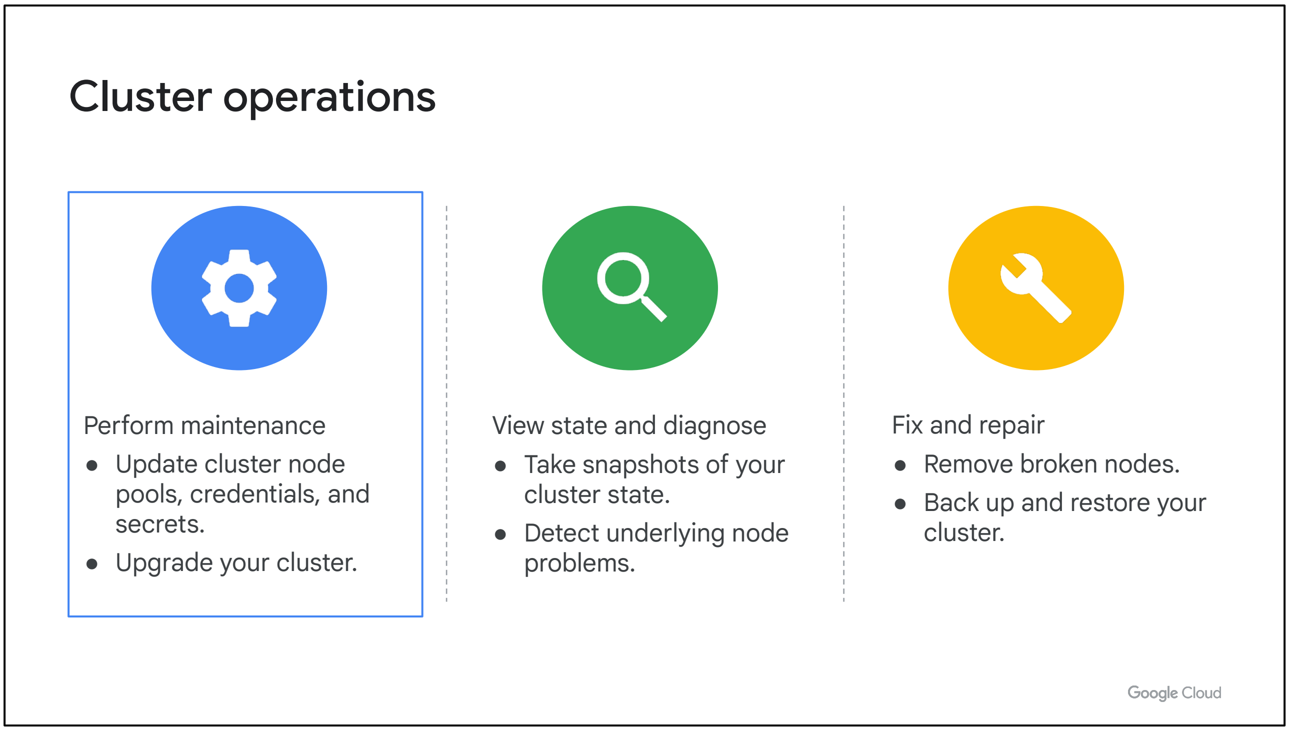
- Take snapshots of your cluster state.
- Detect underlying node problems.

Fix and repair

- Remove broken nodes.
- Back up and restore your cluster.

Google Cloud

Ongoing operations entails maintenance, debugging, repairs, etc.



Cluster operations

Perform maintenance

- Update cluster node pools, credentials, and secrets.
- Upgrade your cluster.

View state and diagnose

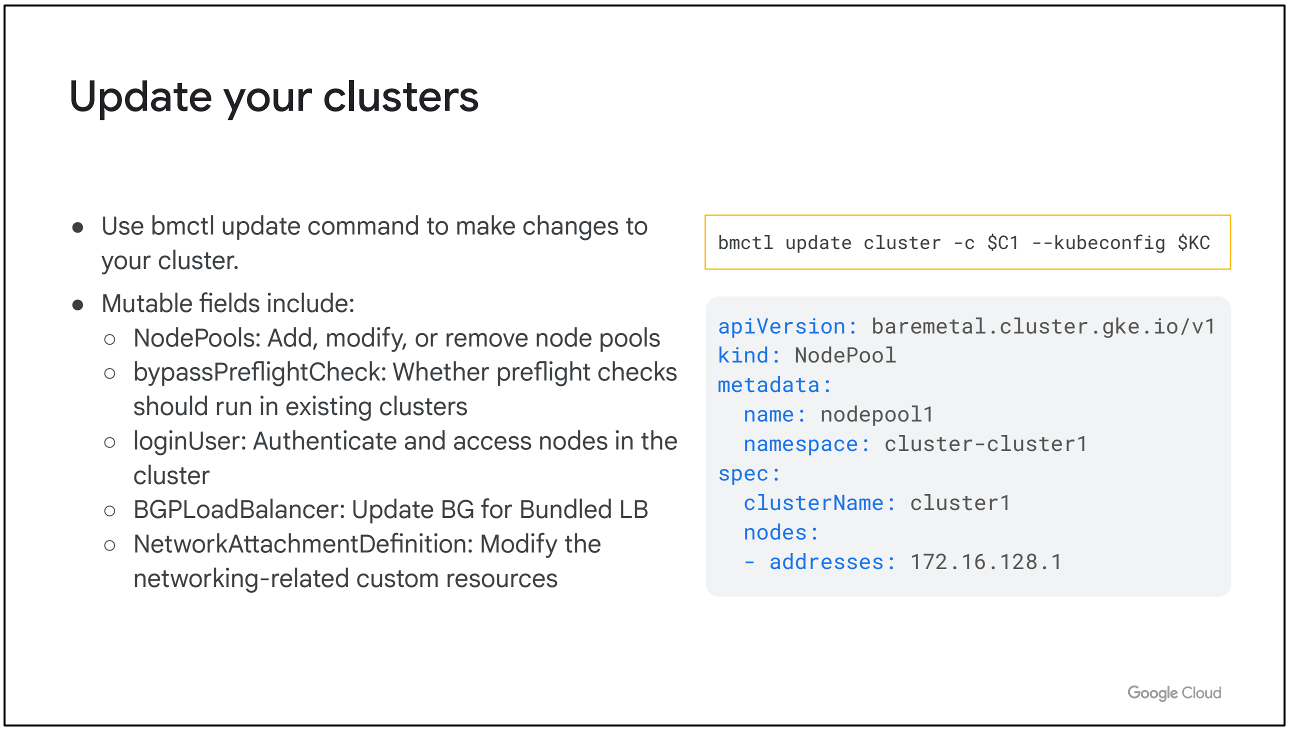
- Take snapshots of your cluster state.
- Detect underlying node problems.

Fix and repair

- Remove broken nodes.
- Back up and restore your cluster.

Google Cloud

Clusters sometimes need to be updated. For example, when you add more nodes to your cluster or need to update the hardware or software of existing nodes. It's also likely that you will upgrade your clusters to newer versions of Anthos over their lifespan.



Update your clusters

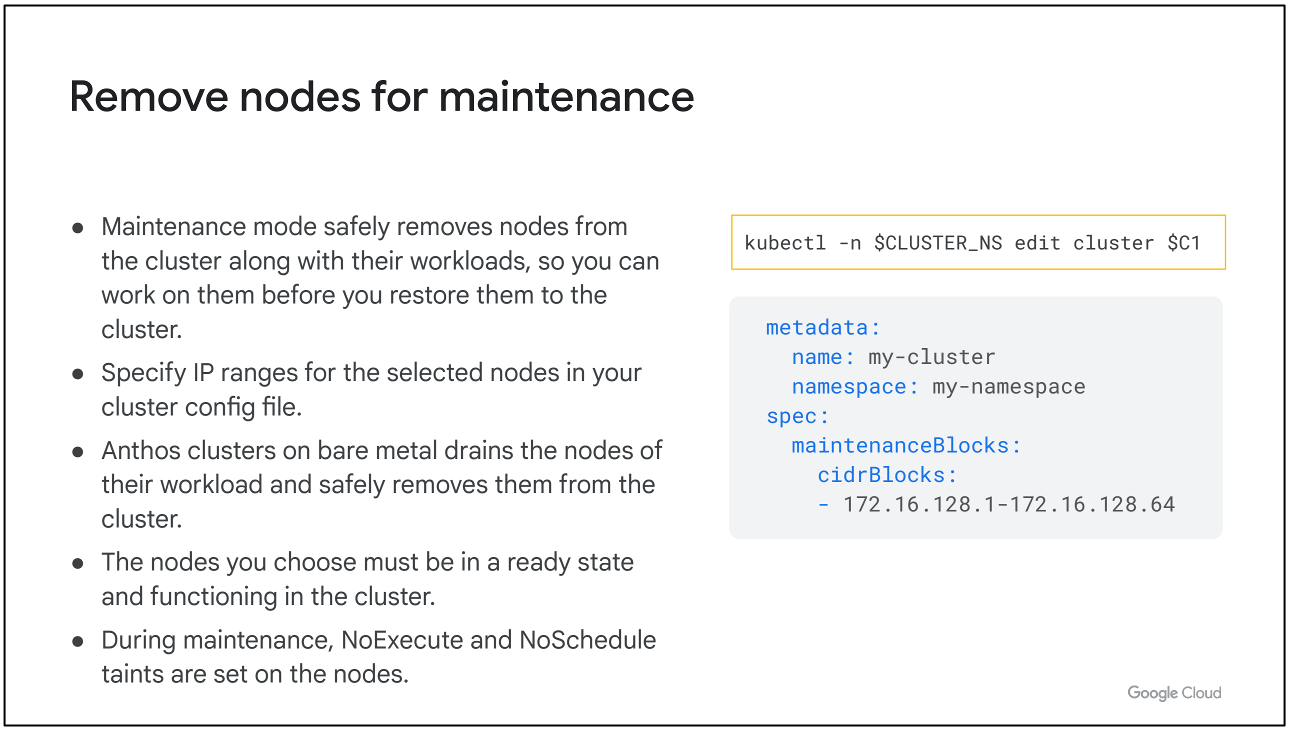
- Use bmctl update command to make changes to your cluster.
- Mutable fields include:
  - NodePools: Add, modify, or remove node pools
  - bypassPreflightCheck: Whether preflight checks should run in existing clusters
  - loginUser: Authenticate and access nodes in the cluster
  - BGPLoadBalancer: Update BG for Bundled LB
  - NetworkAttachmentDefinition: Modify the networking-related custom resources

```
bmctl update cluster -c $C1 --kubeconfig $KC
```

```
apiVersion: baremetal.cluster.gke.io/v1
kind: NodePool
metadata:
  name: nodepool1
  namespace: cluster-cluster1
spec:
  clusterName: cluster1
  nodes:
  - addresses: 172.16.128.1
```

Google Cloud

Let’s take a look at the first use case. Imagine you need to add additional nodes to your cluster because your workload demands have increased. Once those machines have been installed and configured, you need to modify the NodePool configuration section in the cluster config file and include the addresses of the new machines. For the effects to take place, you must run bmctl update. You could also remove nodes. In this case, the nodes are first drained of pods but will not be removed from the cluster if pods can't be rescheduled on other nodes. Additionally, there are other mutable fields in the cluster configuration that you can change. For instance, the bypassPreflightCheck specifies whether preflight checks should run in existing clusters when changes take place, loginUser specifies the authentication and access details to nodes in the cluster, BGPLoadBalancer updates the connectivity for Bundled LB, or the NetworkAttachmentDefinition modifies the networking-related custom resources.



Remove nodes for maintenance

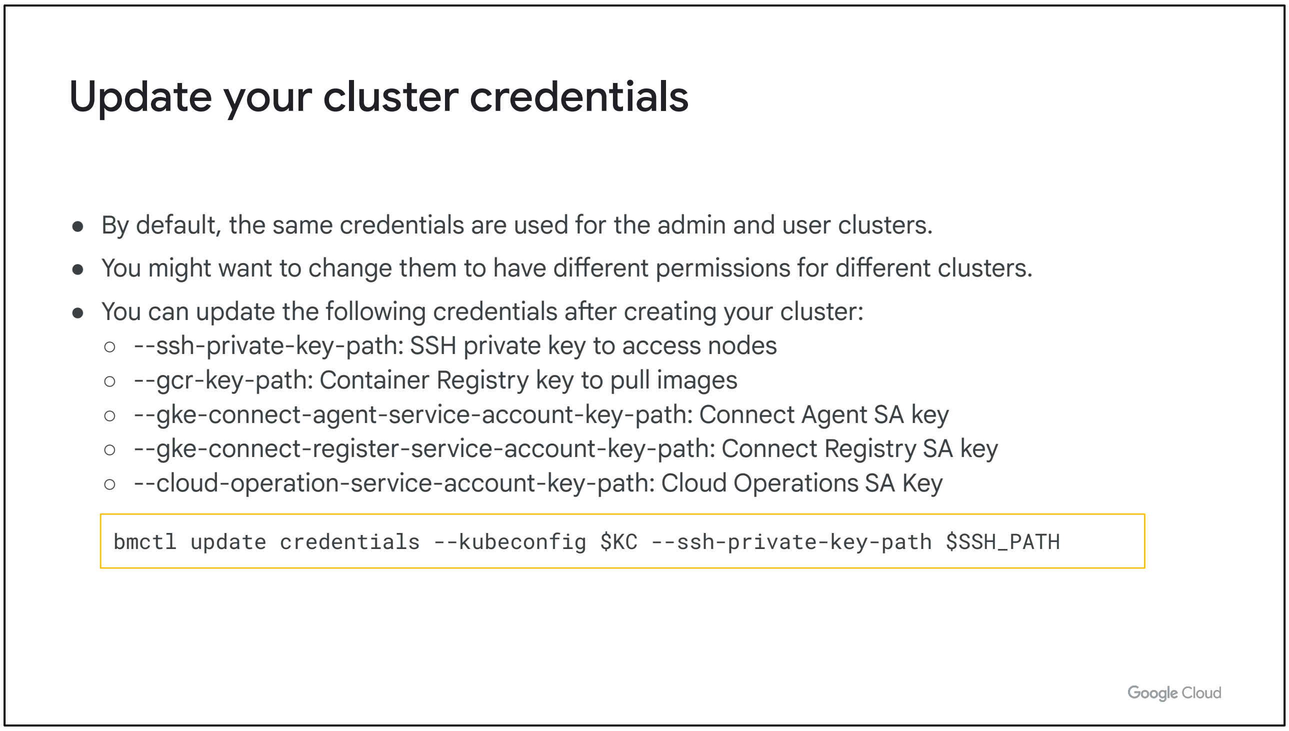
- Maintenance mode safely removes nodes from the cluster along with their workloads, so you can work on them before you restore them to the cluster.
- Specify IP ranges for the selected nodes in your cluster config file.
- Anthos clusters on bare metal drains the nodes of their workload and safely removes them from the cluster.
- The nodes you choose must be in a ready state and functioning in the cluster.
- During maintenance, NoExecute and NoSchedule taints are set on the nodes.

```
kubectl -n $CLUSTER_NS edit cluster $C1
```

```
metadata:
  name: my-cluster
  namespace: my-namespace
spec:
  maintenanceBlocks:
    cidrBlocks:
    - 172.16.128.1-172.16.128.64
```

Google Cloud

Sometimes, you need to shortly remove nodes for maintenance. Maybe you need to add a hardware component to the server such as an additional memory card or you might need to update some software. Maintenance mode safely removes nodes from the cluster along with their workloads, so you can work on them before you restore them to the cluster. To do so, you must specify IP ranges for the nodes to go into maintenance in your cluster config file and use kubectl on the admin cluster to apply the changes. Anthos clusters on bare metal drains the nodes of their workload, and safely removes them from the cluster. The nodes you choose must be in a ready state, and functioning in the cluster. During maintenance, NoExecute and NoSchedule taints are set on the nodes.



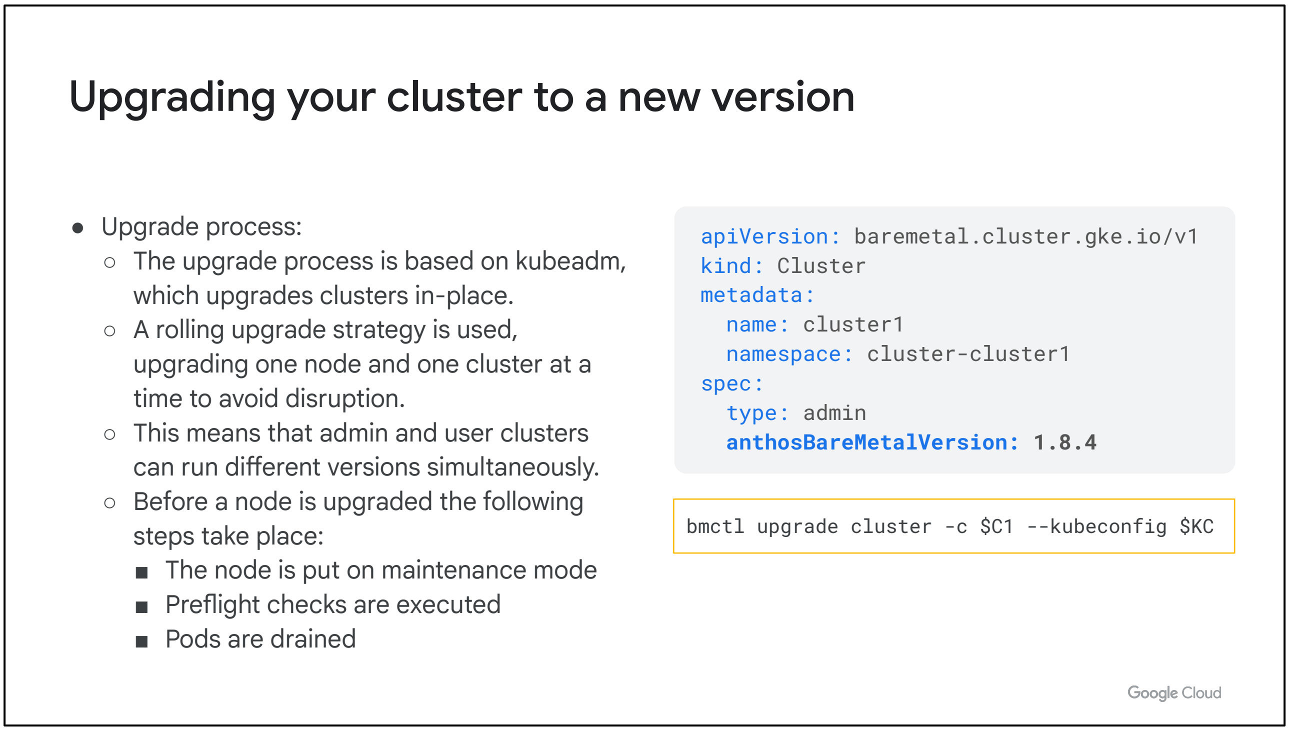
Update your cluster credentials

- By default, the same credentials are used for the admin and user clusters.
- You might want to change them to have different permissions for different clusters.
- You can update the following credentials after creating your cluster:
  - --ssh-private-key-path: SSH private key to access nodes
  - --gcr-key-path: Container Registry key to pull images
  - --gke-connect-agent-service-account-key-path: Connect Agent SA key
  - --gke-connect-register-service-account-key-path: Connect Registry SA key
  - --cloud-operation-service-account-key-path: Cloud Operations SA Key

```
bmctl update credentials --kubeconfig $KC --ssh-private-key-path $SSH_PATH
```

Google Cloud

When Anthos clusters are first created, the same credentials are used for both the admin and user clusters. You might want to change them to have different permissions for different clusters, since it's likely that different development teams use different clusters and admin clusters should only be accessible by platform administrators. To update your cluster credentials, update one or multiple keys from the list in the slide and use the "bmctl update credentials" command to execute the update.



Upgrading your cluster to a new version

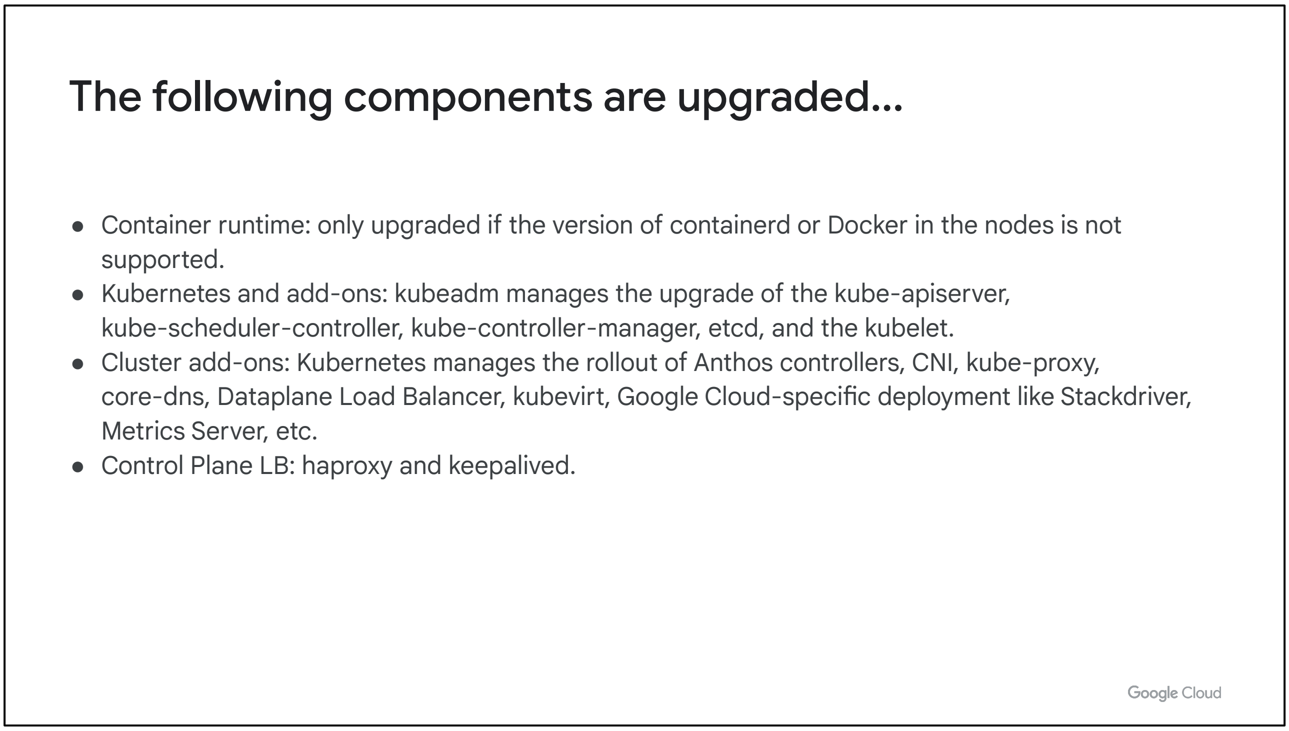
- Upgrade process:
  - The upgrade process is based on kubeadm, which upgrades clusters in-place.
  - A rolling upgrade strategy is used, upgrading one node and one cluster at a time to avoid disruption.
  - This means that admin and user clusters can run different versions simultaneously.
  - Before a node is upgraded the following steps take place:
    - The node is put on maintenance mode
    - Preflight checks are executed
    - Pods are drained

```
apiVersion: baremetal.cluster.gke.io/v1
kind: Cluster
metadata:
  name: cluster1
  namespace: cluster-cluster1
spec:
  type: admin
  anthosBareMetalVersion: 1.8.4
```

```
bmctl upgrade cluster -c $C1 --kubeconfig $KC
```

Google Cloud

Another common maintenance activity involves upgrading your clusters to a new revision. Anthos provides a kubeadm-based bmctl command to simplify the process, providing an in-place, rolling update of the nodes in your cluster, which happens one node at a time to avoid disruption. The process works as follows. You must change the Anthos bare metal version in the cluster resource and run the bmct upgrade cluster command. Then, a node is put on maintenance mode before upgrading. Preflight checks are executed on the node and pods are drained. This process takes place node by node, one cluster at a time. Admin and user clusters do not need to be updated simultaneously and can run different versions.

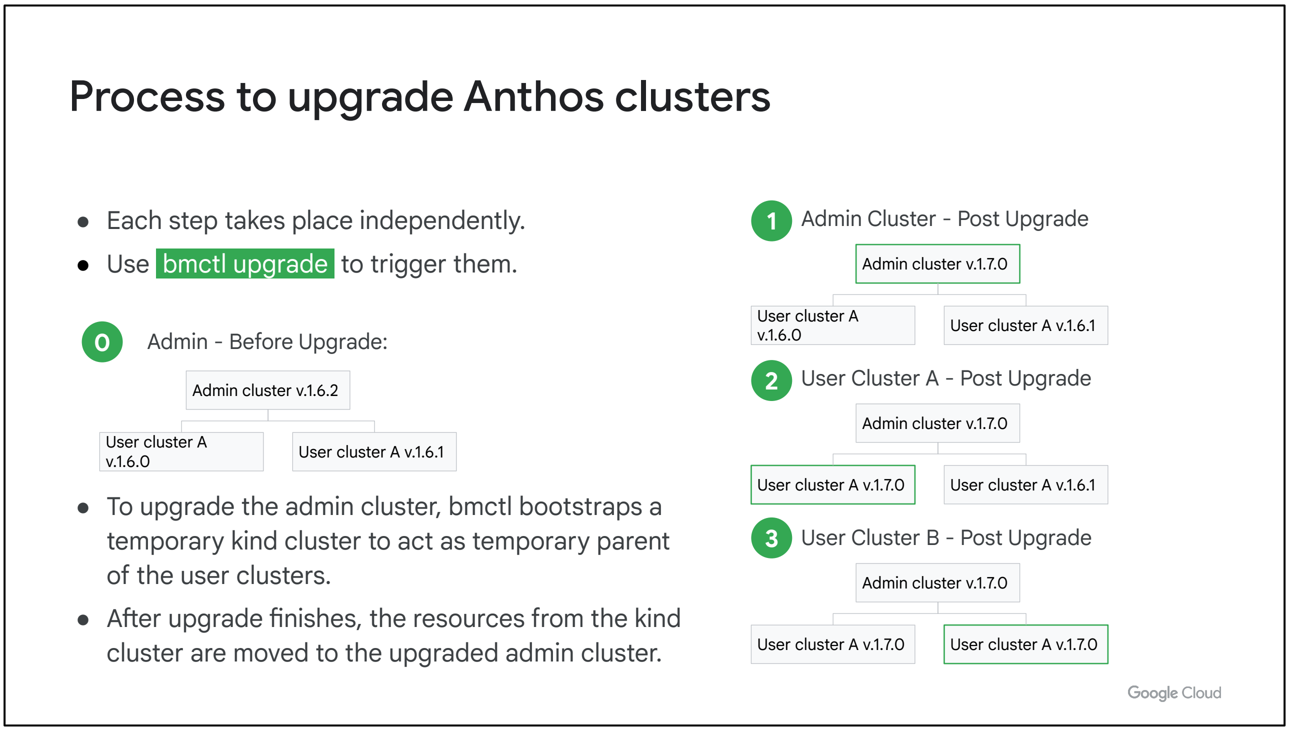


The following components are upgraded...

- Container runtime: only upgraded if the version of containerd or Docker in the nodes is not supported.
- Kubernetes and add-ons: kubeadm manages the upgrade of the kube-apiserver, kube-scheduler-controller, kube-controller-manager, etcd, and the kubelet.
- Cluster add-ons: Kubernetes manages the rollout of Anthos controllers, CNI, kube-proxy, core-dns, Dataplane Load Balancer, kubevirt, Google Cloud-specific deployment like Stackdriver, Metrics Server, etc.
- Control Plane LB: haproxy and keepalived.

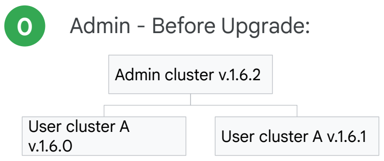
Google Cloud

The following components are upgraded. The container runtime, in case the current version is not currently supported. Kubernetes and it's add-ons: here's where bmctl leverages kubeadm to upgrade kube-apiserver, kube-scheduler-controller, kube-controller-manager, etcd, and the kubelet. Cluster add-ons, so Kubernetes manages the rollout of Anthos controllers, CNI, kube-proxy, core-dns, Dataplane Load Balancer, kubevirt, Google Cloud-specific deployment like Stackdriver, Metrics Server, etc. And finally, the Control Plane Load Balancer, where haproxy and keepalived are upgraded.

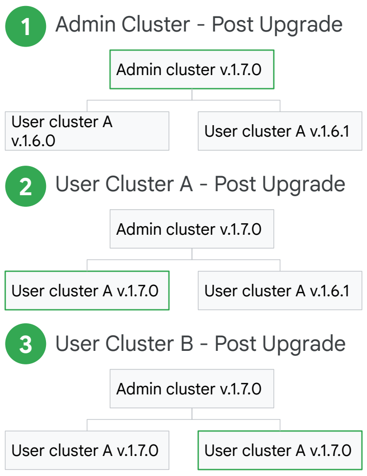


Process to upgrade Anthos clusters

- Each step takes place independently.
- Use bmctl upgrade to trigger them.



- To upgrade the admin cluster, bmctl bootstraps a temporary kind cluster to act as temporary parent of the user clusters.
- After upgrade finishes, the resources from the kind cluster are moved to the upgraded admin cluster.



Google Cloud

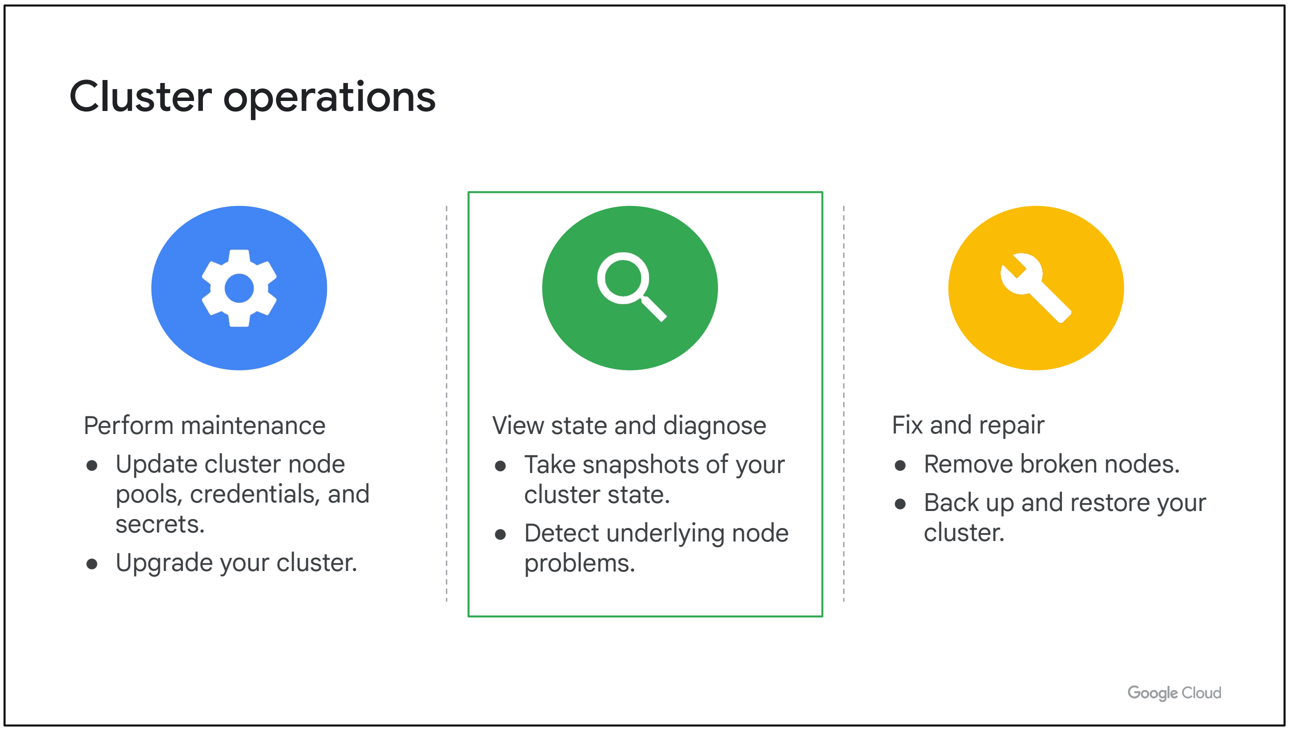
These are the steps to upgrade Anthos clusters. Each step takes place independently. Remember that clusters can be running different versions simultaneously. Step 0 displays a diagram of three clusters all running different versions of Anthos.

The fist step entails upgrading the admin cluster from version 1.6.2 to version 1.7. To do so, a temporary Kind cluster is bootstrapped to act as temporary parent of the user clusters. The upgrade of the cluster and components takes place as described on the previous slide, and then the resources from the Kind cluster are moved to the upgraded admin cluster and the Kind cluster is destroyed.

You can see the result in Step 1. Now, if we want to update User Cluster A, we run bmctl upgrade again, and node by node the cluster is upgraded to version 1.7.

Now, in Step 2, we see that the user clusters are running different versions without problems.

Finally, we can choose to run the same process one more time to have the three clusters on the same version.



Cluster operations

Perform maintenance

- Update cluster node pools, credentials, and secrets.
- Upgrade your cluster.

View state and diagnose

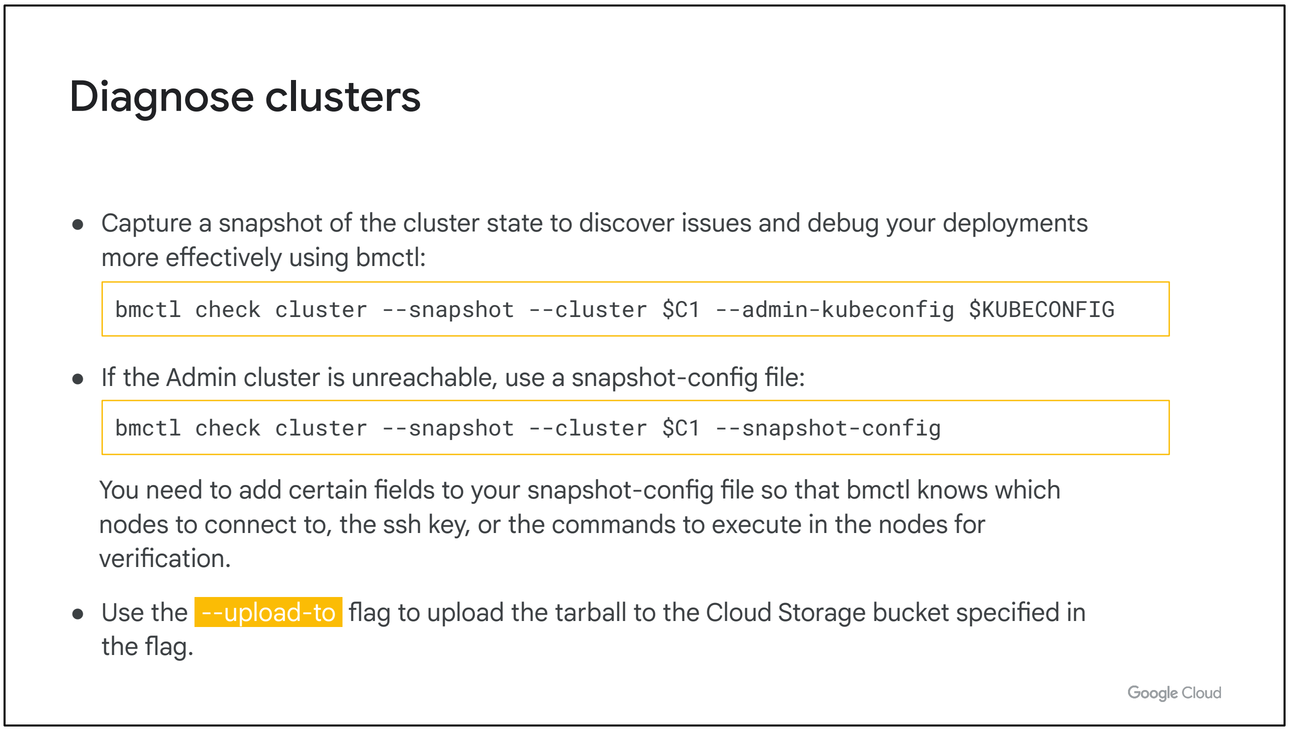
- Take snapshots of your cluster state.
- Detect underlying node problems.

Fix and repair

- Remove broken nodes.
- Back up and restore your cluster.

Google Cloud

Let's now investigate how we can view the state and diagnose clusters. Usually, we will receive an alert from Cloud Monitoring pointing out that a resource might be having problems, and if the logs stored in Cloud Logging are not enough, you might use the following tools to narrow down the issue.



Diagnose clusters

- Capture a snapshot of the cluster state to discover issues and debug your deployments more effectively using bmctl:

```
bmctl check cluster --snapshot --cluster $C1 --admin-kubeconfig $KUBECONFIG
```

- If the Admin cluster is unreachable, use a snapshot-config file:

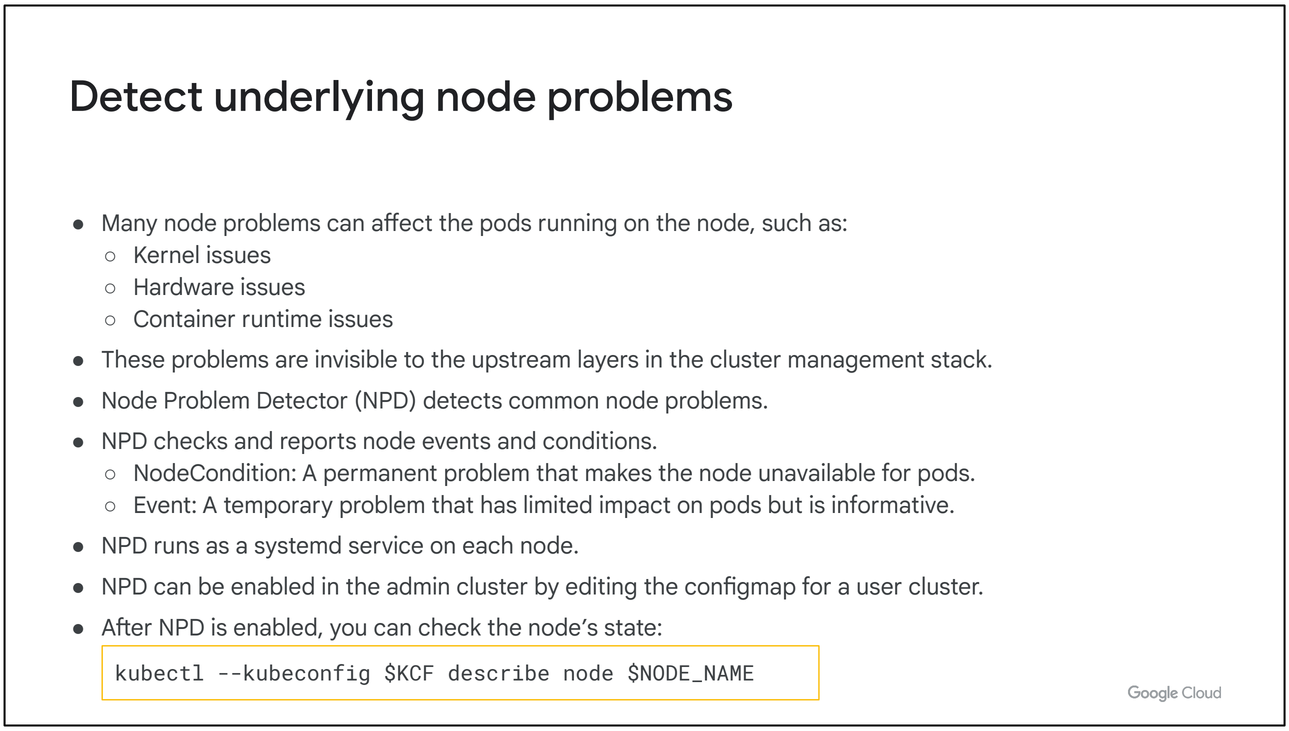
```
bmctl check cluster --snapshot --cluster $C1 --snapshot-config
```

  You need to add certain fields to your snapshot-config file so that bmctl knows which nodes to connect to, the ssh key, or the commands to execute in the nodes for verification.

- Use the `--upload-to` flag to upload the tarball to the Cloud Storage bucket specified in the flag.

Google Cloud

A very useful command is the bmctl check cluster, which basically runs the same preflight checks on the nodes and network that are executed when the cluster is first created. Preflight checks verify that the Anthos software is installed and running correctly and that there is connectivity between the machines and Google Cloud. You can choose to export the snapshot file to the admin workstation or upload it to a Cloud Storage bucket for easier consumption.



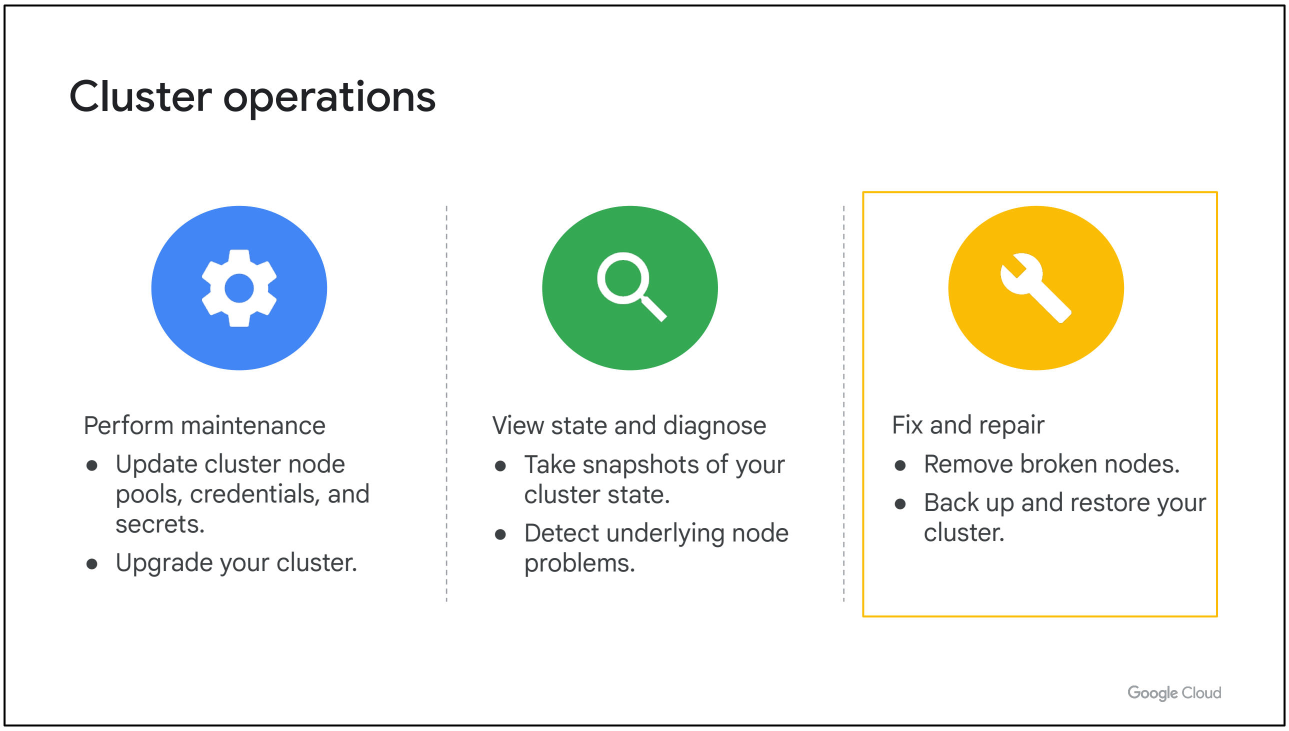
Detect underlying node problems

- Many node problems can affect the pods running on the node, such as:
  - Kernel issues
  - Hardware issues
  - Container runtime issues
- These problems are invisible to the upstream layers in the cluster management stack.
- Node Problem Detector (NPD) detects common node problems.
- NPD checks and reports node events and conditions.
  - NodeCondition: A permanent problem that makes the node unavailable for pods.
  - Event: A temporary problem that has limited impact on pods but is informative.
- NPD runs as a systemd service on each node.
- NPD can be enabled in the admin cluster by editing the configmap for a user cluster.
- After NPD is enabled, you can check the node’s state:

```
kubectl --kubeconfig $KCF describe node $NODE_NAME
```

Google Cloud

Sometimes, the problem might be in the node itself, but Kubernetes has no way to identify that something is not working correctly. Node problems include kernel, hardware and container runtime issues. To make those problems visible to the upstream layers in the cluster management stack, you can enable Node Problem Detector, or NPD, which runs as a systemd service on each node and notifies with Kubernetes Events and NodeConditions. An Event is typically a temporary problem that has limited impact on pods but is informative, while a NodeCondition is a permanent problem that makes the node unavailable for pods. NPD can be enabled in the admin cluster by editing the configmap for a user cluster. Once NDP is enabled, problems will be made available directly as part of your node object and you view them by running the “kubectl describe node” command.



Cluster operations

Perform maintenance

- Update cluster node pools, credentials, and secrets.
- Upgrade your cluster.

View state and diagnose

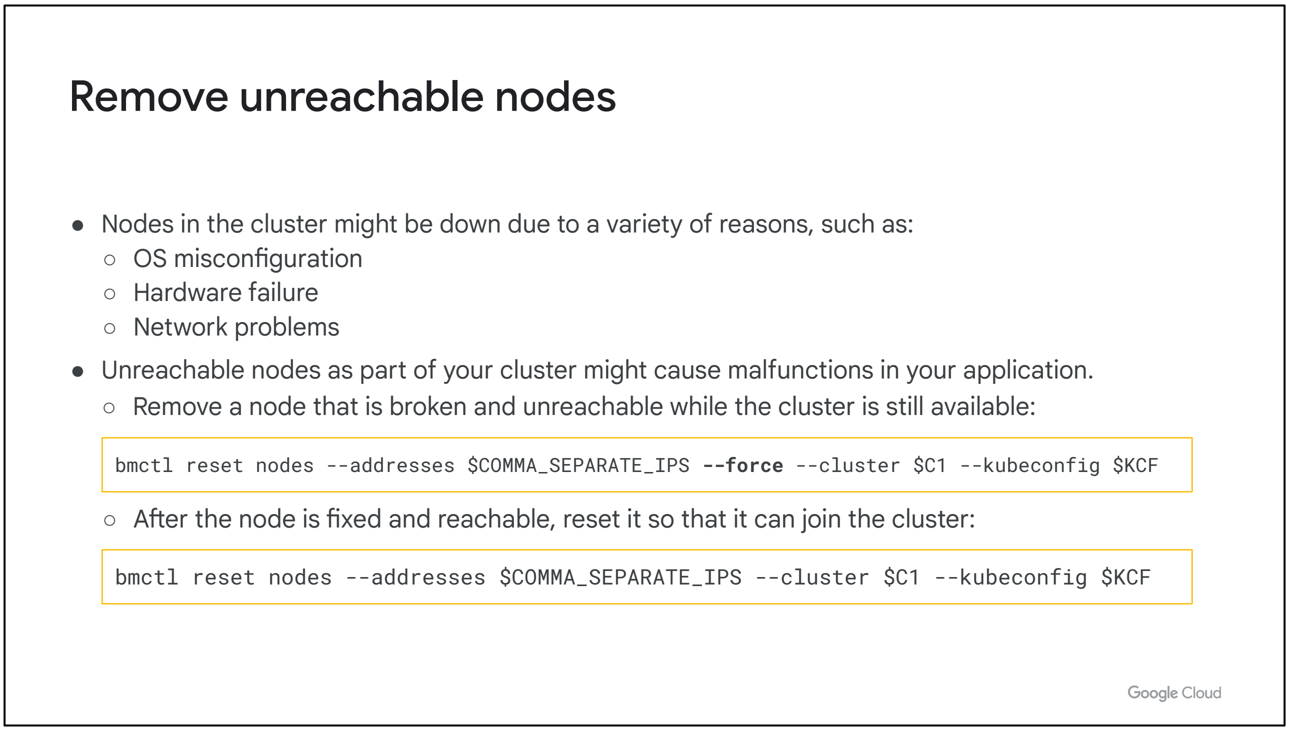
- Take snapshots of your cluster state.
- Detect underlying node problems.

Fix and repair

- Remove broken nodes.
- Back up and restore your cluster.

Google Cloud

Once problems are identified, it’s time to fix them. We can either fix one node at a time, or recreate part of or the whole cluster with a backup.



Remove unreachable nodes

- Nodes in the cluster might be down due to a variety of reasons, such as:
  - OS misconfiguration
  - Hardware failure
  - Network problems
- Unreachable nodes as part of your cluster might cause malfunctions in your application.
  - Remove a node that is broken and unreachable while the cluster is still available:

```
bmctl reset nodes --addresses $COMMA_SEPARATE_IPS --force --cluster $C1 --kubeconfig $KCF
```

  - After the node is fixed and reachable, reset it so that it can join the cluster:

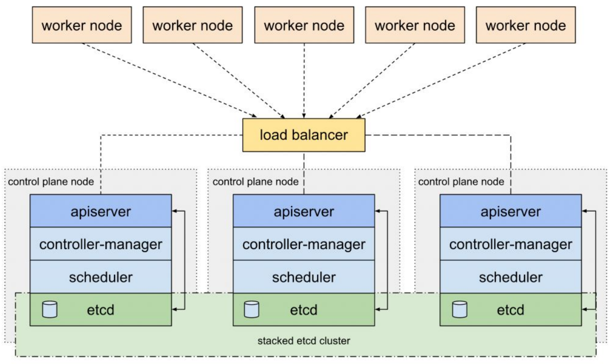
```
bmctl reset nodes --addresses $COMMA_SEPARATE_IPS --cluster $C1 --kubeconfig $KCF
```

Google Cloud

If the nodes are broken or in an unreachable state, scheduling them up for maintenance and gracefully draining the pods will not be possible. Nodes might reach that state due to a variety of reasons such as OS misconfigurations, hardware failure or network problems. If your nodes reach such state, you can force-remove nodes by executing the “bmctl reset nodes --force” command. By removing broken nodes, stuck unreachable application pods that were deployed on broken nodes should reschedule to healthy nodes in the cluster. Once the nodes are fixed and reachable, you can run the “bmctl reset nodes” command to join them back to the cluster.

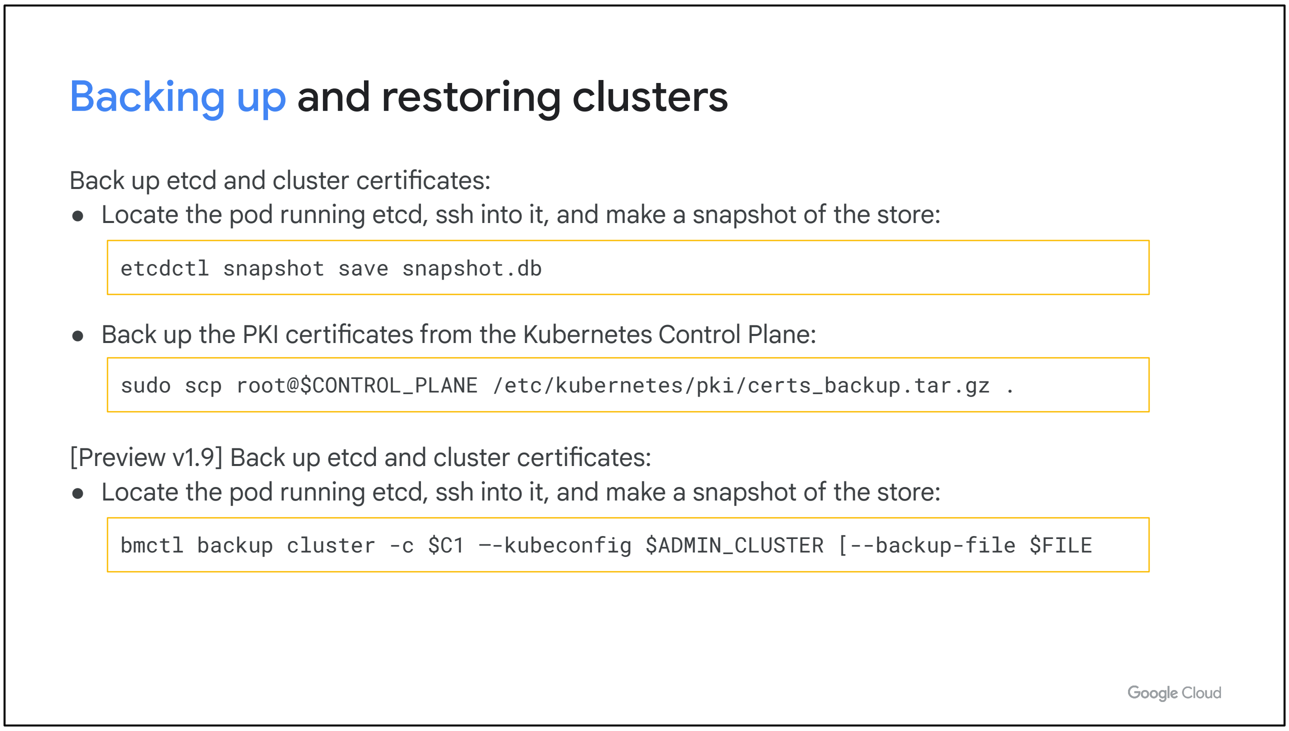
Backing up and restoring clusters

- All Kubernetes objects are stored on etcd.
- Periodically backing up the etcd cluster data is important to recover Kubernetes clusters under disaster scenarios, such as losing all control plane nodes.
- Create a snapshot of etcd, including all the Kubernetes states and critical information.
- In order to keep the sensitive Kubernetes data safe, encrypt the snapshot files.



Google Cloud

We talked so far about worker nodes breaking, but what happens if there is a failure on the control plane nodes? All Kubernetes objects in a cluster are stored on the control plane nodes, specifically in the etcd database. Periodically backing up etcd is important to recover Kubernetes clusters under disaster scenarios, such as losing all control plane nodes. To back up the database, you need to create a snapshot of etcd and ideally encrypt the snapshot files to keep them safe.



Backing up and restoring clusters

Back up etcd and cluster certificates:

- Locate the pod running etcd, ssh into it, and make a snapshot of the store:

```
etcdctl snapshot save snapshot.db
```

- Back up the PKI certificates from the Kubernetes Control Plane:

```
sudo scp root@$CONTROL_PLANE /etc/kubernetes/pki/certs_backup.tar.gz .
```

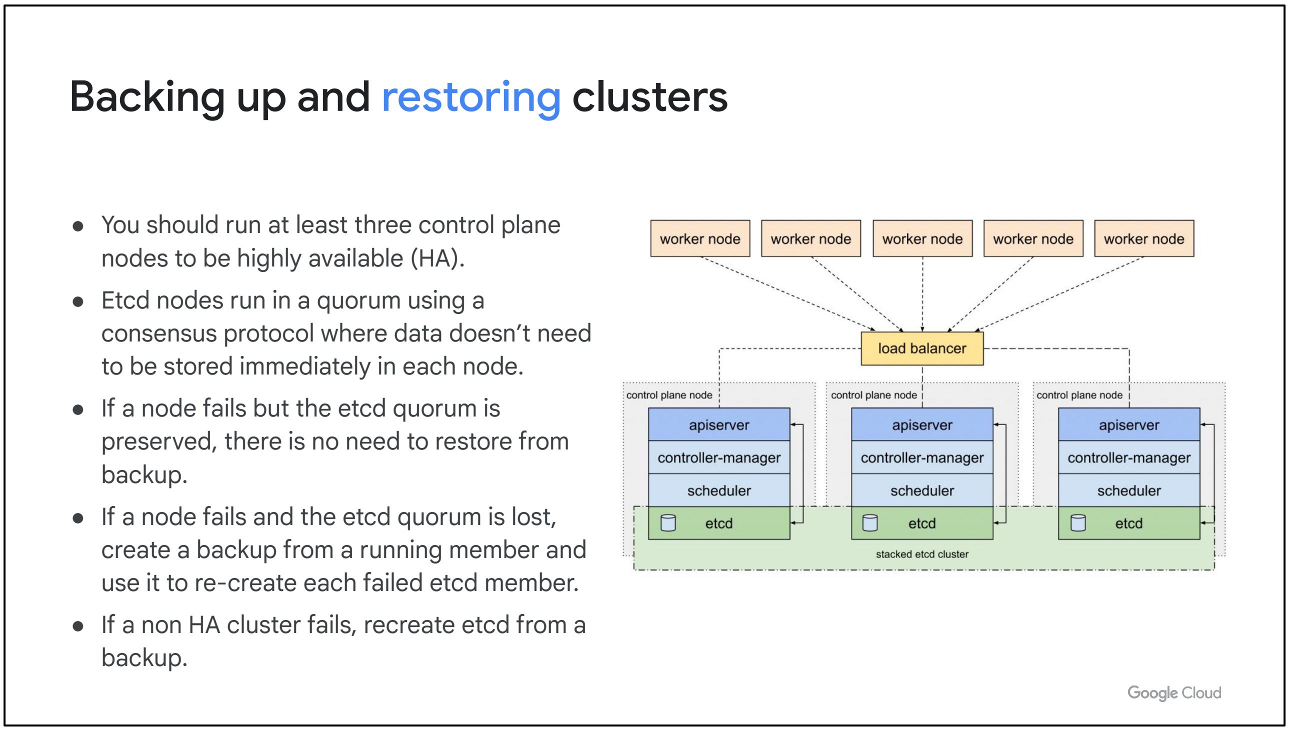
[Preview v1.9] Back up etcd and cluster certificates:

- Locate the pod running etcd, ssh into it, and make a snapshot of the store:

```
bmctl backup cluster -c $C1 —-kubeconfig $ADMIN_CLUSTER [--backup-file $FILE
```

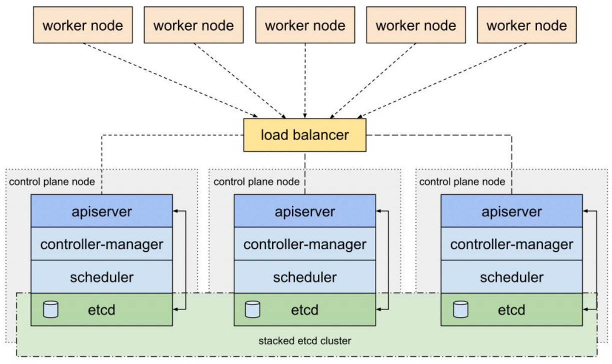
Google Cloud

In open-source Kubernetes, you would locate and SSH into the pod running etcd and create a snapshot of the store using the etcdctl command. Afterwards, you would export it to a location outside of the cluster. Anthos simplifies that process with the “bmctl backup cluster” command, which creates a snapshot of etcd on your admin workstation.



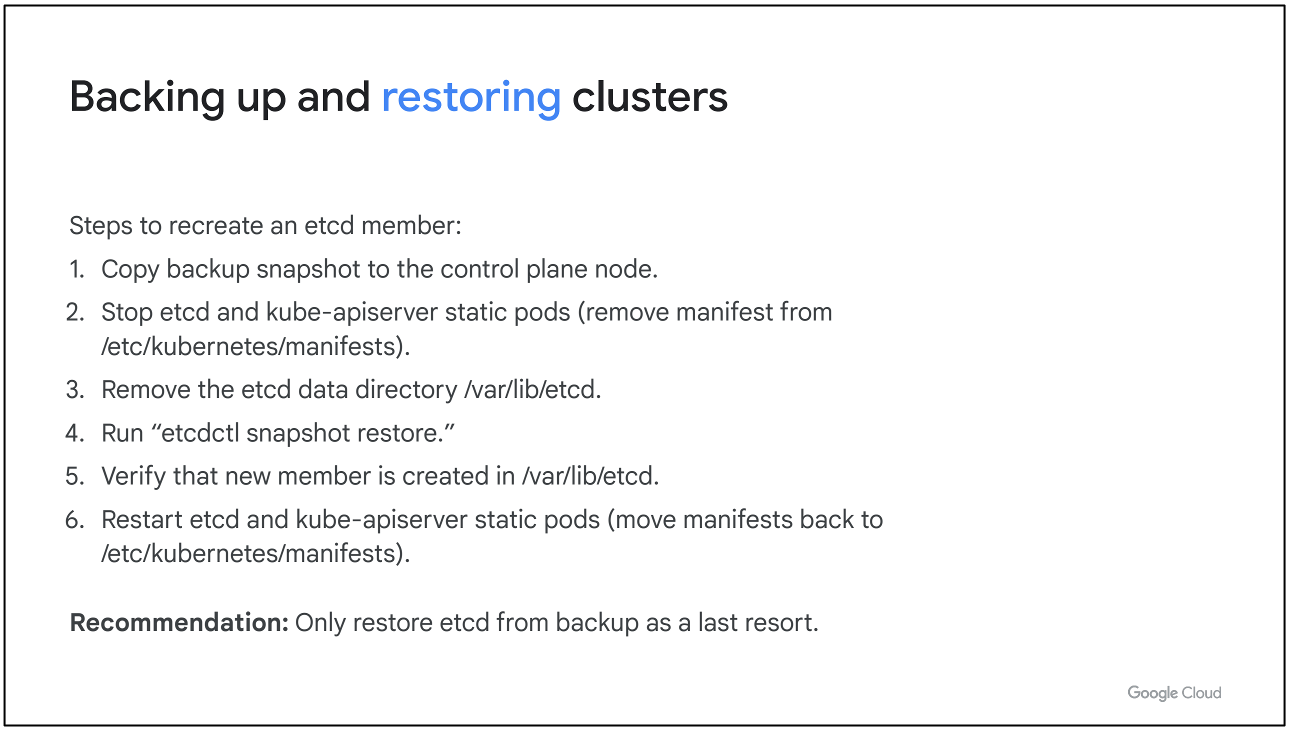
Backing up and restoring clusters

- You should run at least three control plane nodes to be highly available (HA).
- Etcd nodes run in a quorum using a consensus protocol where data doesn't need to be stored immediately in each node.
- If a node fails but the etcd quorum is preserved, there is no need to restore from backup.
- If a node fails and the etcd quorum is lost, create a backup from a running member and use it to re-create each failed etcd member.
- If a non HA cluster fails, recreate etcd from a backup.



Google Cloud

Backing up etcd should happen periodically, but when should you restore the control plane from a backup? Well, ideally, you have a highly available control plane running at least three nodes. Etcd nodes run in a quorum using a consensus protocol, where data doesn't need to be stored immediately in each node. If a node fails, but the etcd quorum is preserved, there is no need to restore from backup. If a node fails, and the etcd quorum is lost, create a backup from a running member and use it to recreate each failed etcd member. If you are not running a highly available cluster and there is a control plane failure, you will have to recreate etcd from a backup.



Backing up and restoring clusters

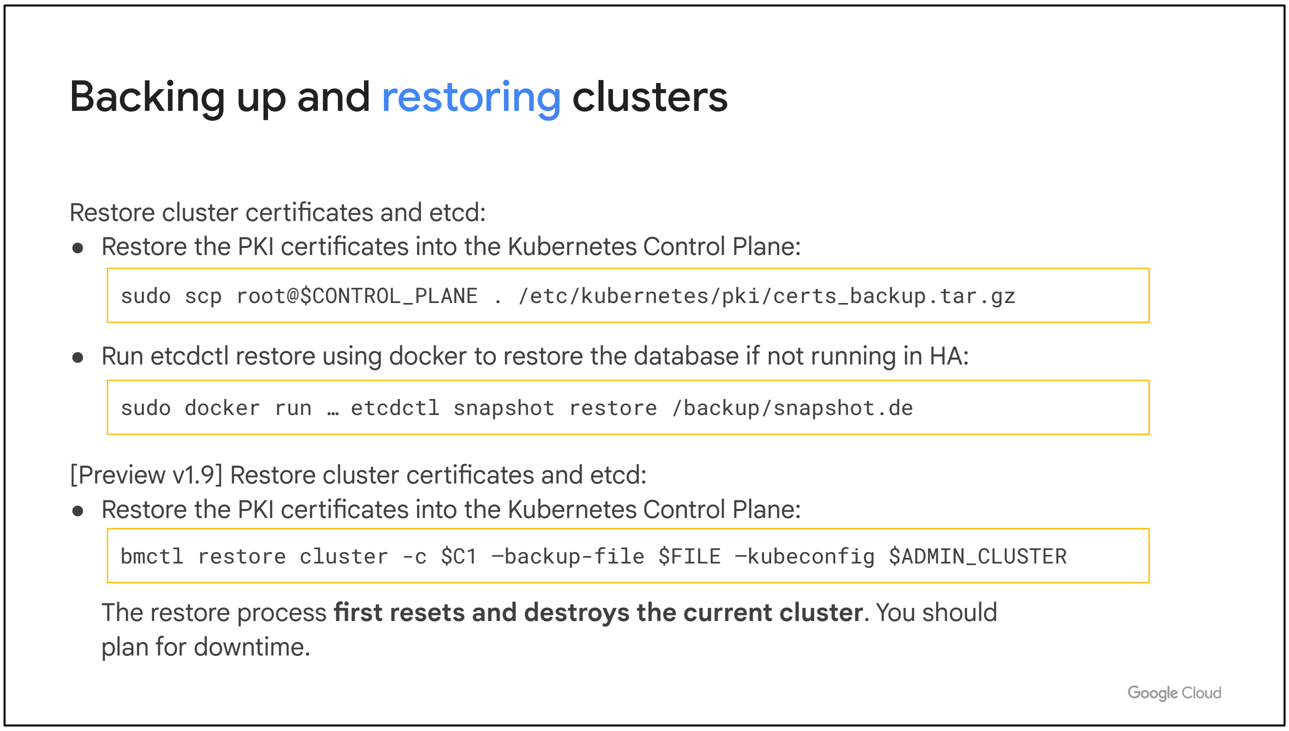
Steps to recreate an etcd member:

1. Copy backup snapshot to the control plane node.
2. Stop etcd and kube-apiserver static pods (remove manifest from /etc/kubernetes/manifests).
3. Remove the etcd data directory /var/lib/etcd.
4. Run "etcdctl snapshot restore."
5. Verify that new member is created in /var/lib/etcd.
6. Restart etcd and kube-apiserver static pods (move manifests back to /etc/kubernetes/manifests).

Recommendation: Only restore etcd from backup as a last resort.

Google Cloud

These are the steps to recreate an etcd member: Copy the backup snapshot to the control plane node. Stop etcd and kube-apiserver static pods (remove manifest from /etc/kubernetes/manifests). Remove the etcd data directory /var/lib/etcd. Run "etcdctl snapshot restore". Verify that the new member is created in /var/lib/etcd. Restart etcd and kube-apiserver static pods (move manifests back to /etc/kubernetes/manifests). Recommendation: Only restore etcd from backup as a last resort.



Backing up and restoring clusters

Restore cluster certificates and etcd:

- Restore the PKI certificates into the Kubernetes Control Plane:

```
sudo scp root@$CONTROL_PLANE . /etc/kubernetes/pki/certs_backup.tar.gz
```

- Run etcdctl restore using docker to restore the database if not running in HA:

```
sudo docker run … etcdctl snapshot restore /backup/snapshot.de
```

[Preview v1.9] Restore cluster certificates and etcd:

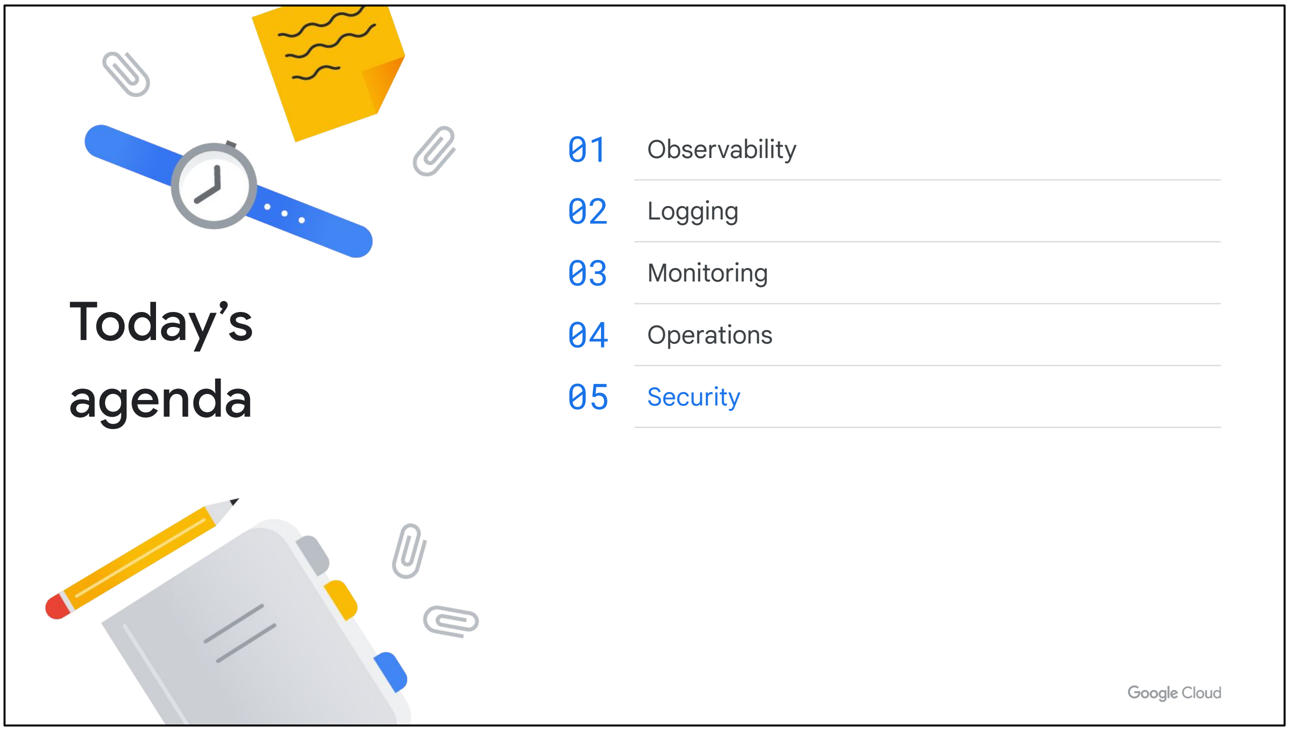
- Restore the PKI certificates into the Kubernetes Control Plane:

```
bmctl restore cluster -c $C1 –backup-file $FILE –kubeconfig $ADMIN_CLUSTER
```

The restore process **first resets and destroys the current cluster**. You should plan for downtime.

Google Cloud

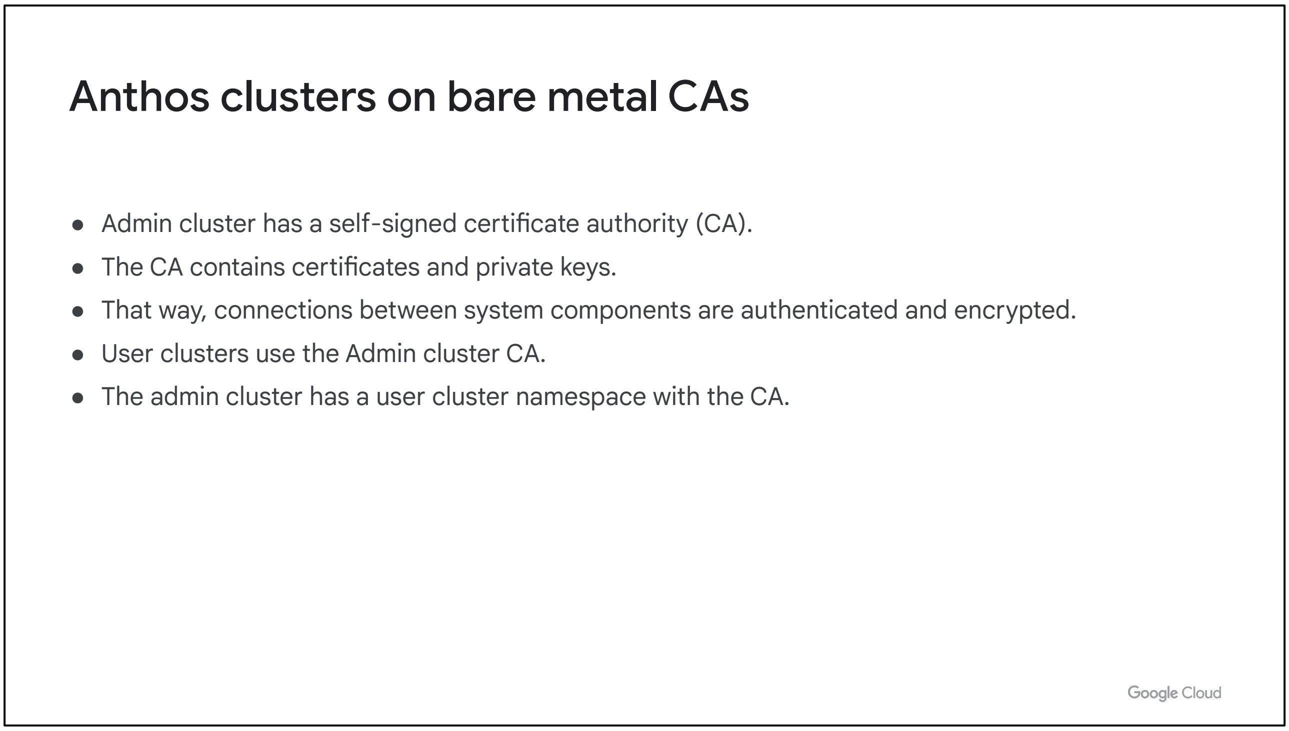
In the use case where we need to recreate a cluster from a backup, the Kubernetes open-source method includes creating the cluster, SSHing in the pod where etcd is running, and restoring the backup with the etcdctl tool. To simplify this process, Anthos provides the “bmctl restore cluster” command, which automatically resets and destroys the cluster and recreates it from the backup. Regardless of the option you choose, you should plan for downtime.



Today's agenda

Google Cloud

Let's take a look at security, the last section in the Operating and Managing Anthos Clusters on Bare Metal module.

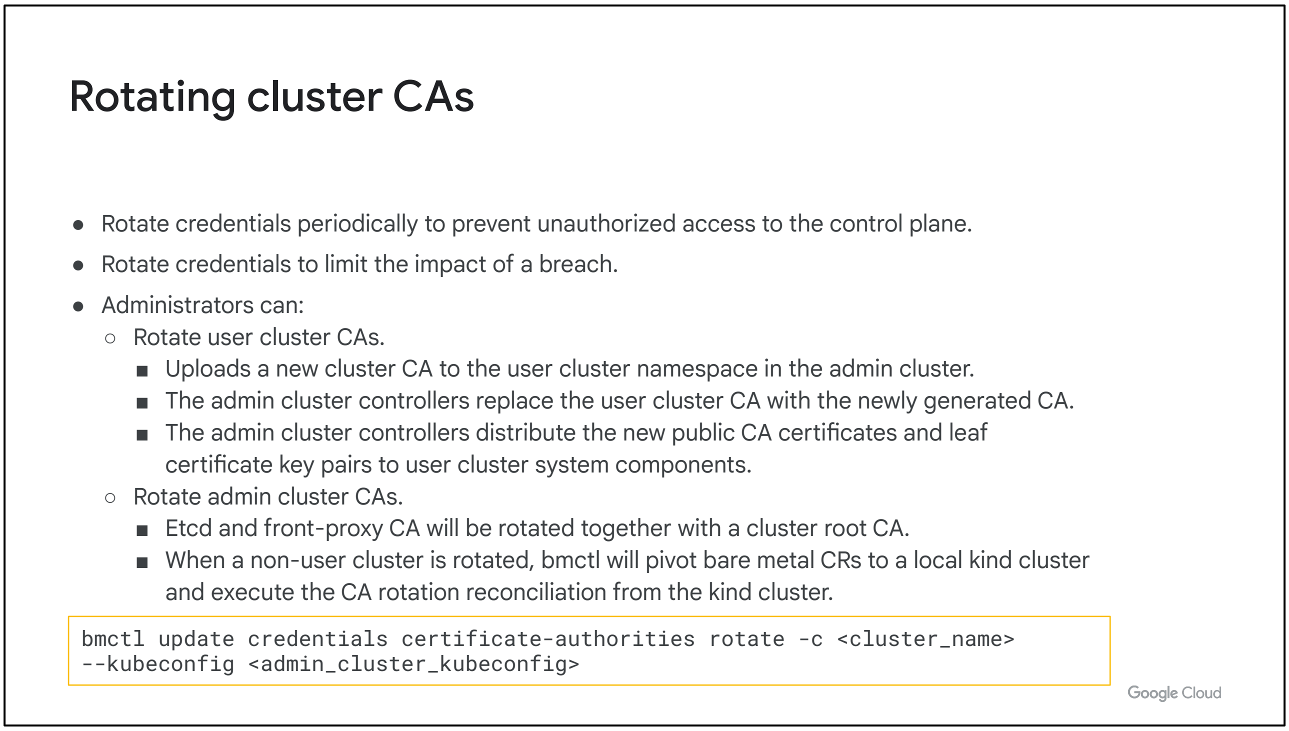


Anthos clusters on bare metal CAs

- Admin cluster has a self-signed certificate authority (CA).
- The CA contains certificates and private keys.
- That way, connections between system components are authenticated and encrypted.
- User clusters use the Admin cluster CA.
- The admin cluster has a user cluster namespace with the CA.

Google Cloud

All control plane communication both between cluster components as well as between admin and user clusters is authenticated and encrypted. That's possible thanks to a Certificate Authority, which establishes trust by storing, verifying, and distributing certificates and private keys among all parties. The admin cluster contains a self-signed Certificate Authority and the user clusters use the admin's cluster CA. Specifically, the admin cluster has a namespace per user cluster where the cluster CA is located. That way all communication is secured.



Rotating cluster CAs

- Rotate credentials periodically to prevent unauthorized access to the control plane.
- Rotate credentials to limit the impact of a breach.
- Administrators can:
  - Rotate user cluster CAs.
    - Uploads a new cluster CA to the user cluster namespace in the admin cluster.
    - The admin cluster controllers replace the user cluster CA with the newly generated CA.
    - The admin cluster controllers distribute the new public CA certificates and leaf certificate key pairs to user cluster system components.
  - Rotate admin cluster CAs.
    - Etcd and front-proxy CA will be rotated together with a cluster root CA.
    - When a non-user cluster is rotated, bmctl will pivot bare metal CRs to a local kind cluster and execute the CA rotation reconciliation from the kind cluster.

```
bmctl update credentials certificate-authorities rotate -c <cluster_name>
--kubeconfig <admin_cluster_kubeconfig>
```

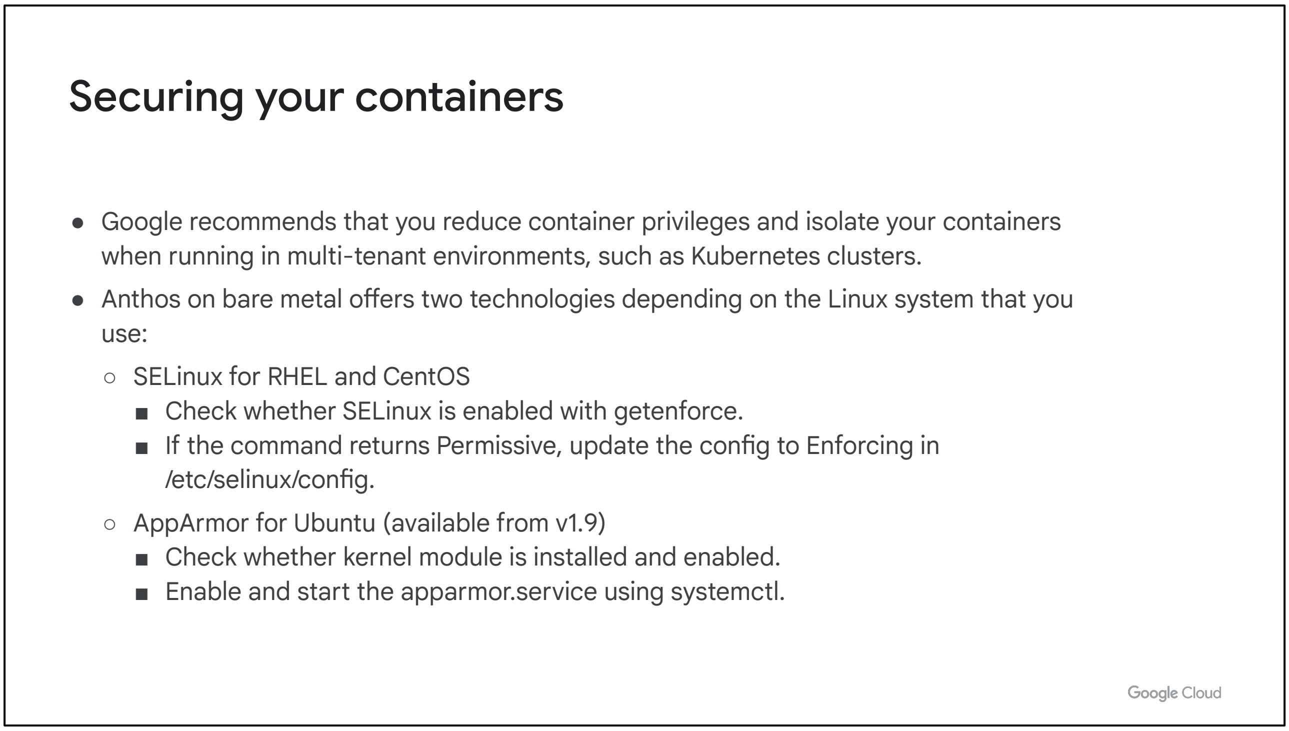
Google Cloud

It’s recommended to rotate the CA credentials periodically to prevent unauthorized access to the control plane. Additionally, in a scenario where a bad actor gains access to your cluster, you must update cluster credentials to limit the impact of the breach.

Administrators can rotate the user cluster CAs, by updating the CA in the user cluster namespace hosted in the admin cluster. After that’s updated, the admin cluster controllers replace the user cluster CA with the newly generated CA and distribute the new public CA certificates and leaf certificate key pairs to user cluster system components.

Administrators can also rotate the admin cluster CAs. In this case, the Etcd and front-proxy CA will be rotated together with cluster root CA. When rotating a non-user cluster, bmctl will pivot bare metal CRs to local kind cluster and execute the CA rotation reconciliation from kind cluster.

Both admin and user cluster CA updates can be performed using the “bmctl update credentials certificate-authorities rotate” command .



Securing your containers

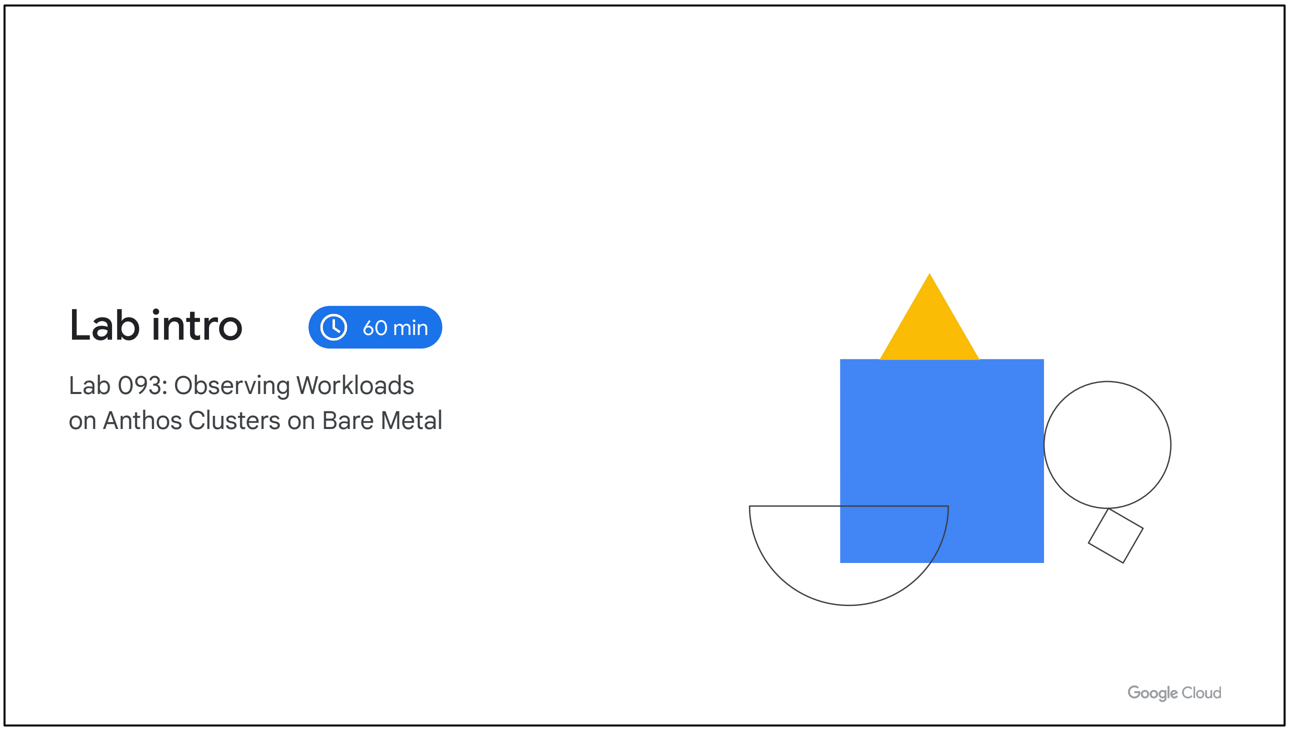
- Google recommends that you reduce container privileges and isolate your containers when running in multi-tenant environments, such as Kubernetes clusters.
- Anthos on bare metal offers two technologies depending on the Linux system that you use:
  - SELinux for RHEL and CentOS
    - Check whether SELinux is enabled with getenforce.
    - If the command returns Permissive, update the config to Enforcing in /etc/selinux/config.
  - AppArmor for Ubuntu (available from v1.9)
    - Check whether kernel module is installed and enabled.
    - Enable and start the apparmor.service using systemctl.

Google Cloud

One common attack vector is to escalate container privileges and escape the user-space into the kernel. To avoid that possibility, it is recommended to reduce container privileges and isolate your containers especially when running in multi-tenant environments.

Anthos on bare metal offers you two technologies depending on the Linux system that you use: SELinux is available for RHEL and CentOS. You can check whether it's running with the getenforce command. If the command returns Permissive, update the config to Enforcing in /etc/selinux/config.

AppArmor is available for Ubuntu when running Anthos v1.9 and upwards. You can check whether it's running by verifying that the kernel module is installed and enabled. If it's not enabled, you can enable and start the apparmor.service using systemctl.



Lab intro

60 min

Lab 093: Observing Workloads on Anthos Clusters on Bare Metal

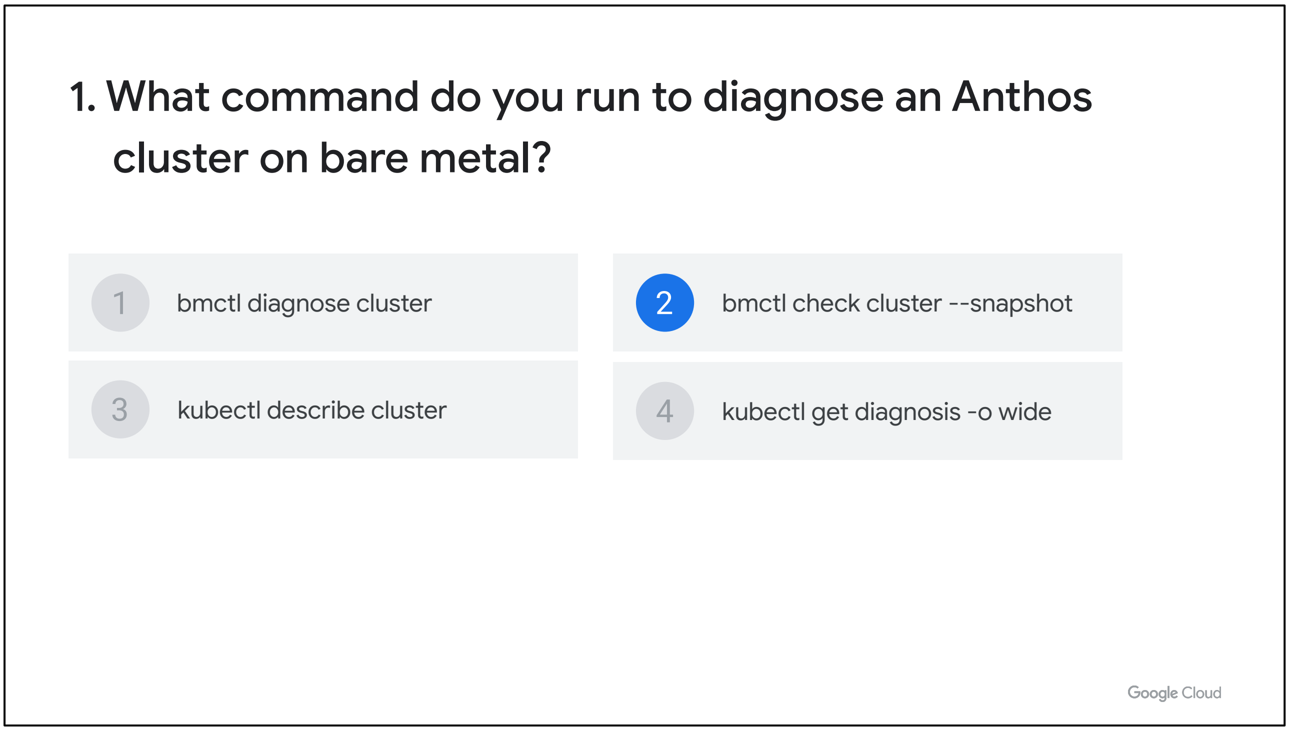
Google Cloud

Questions and answers

Google Cloud

1. What command do you run to diagnose an Anthos cluster on bare metal?

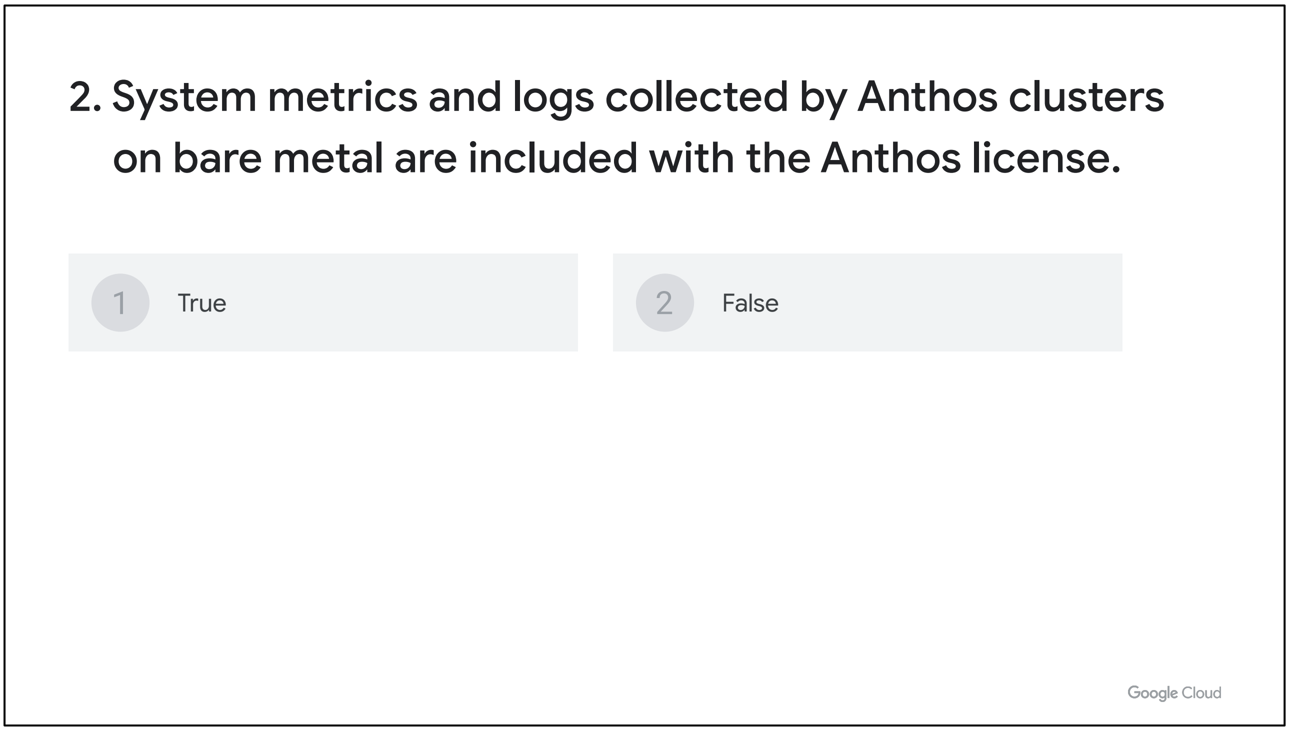
1. bmctl diagnose cluster
2. bmctl check cluster --snapshot
3. kubectl describe cluster
4. kubectl get diagnosis -o wide



1. What command do you run to diagnose an Anthos cluster on bare metal?

1 bmctl diagnose cluster	[✓ 2] bmctl check cluster --snapshot
3 kubectl describe cluster	4 kubectl get diagnosis -o wide

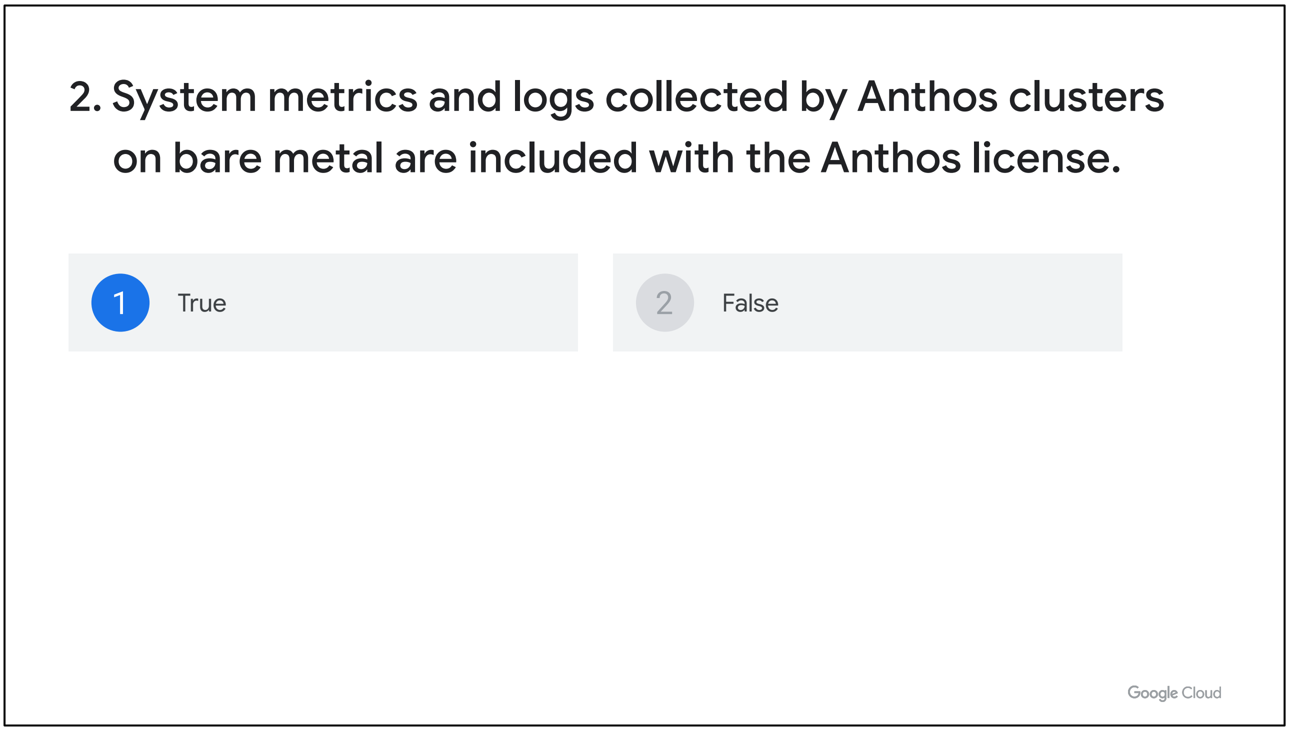
Google Cloud



2. System metrics and logs collected by Anthos clusters on bare metal are included with the Anthos license.

1. True
2. False

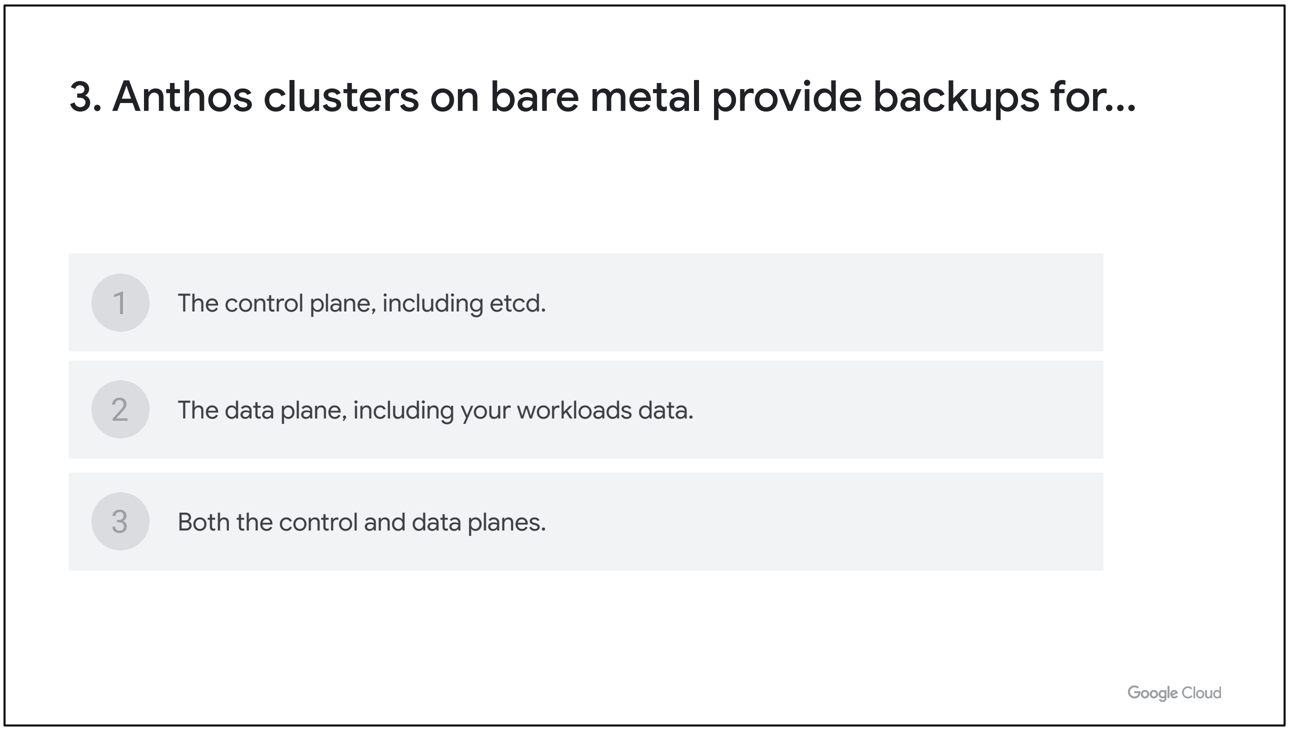
Google Cloud



2. System metrics and logs collected by Anthos clusters on bare metal are included with the Anthos license.

[✓ 1 True] 2 False

Google Cloud



3. Anthos clusters on bare metal provide backups for...

1 The control plane, including etcd.

2 The data plane, including your workloads data.

3 Both the control and data planes.

Google Cloud

3. Anthos clusters on bare metal provide backups for...

[✓ 1] The control plane, including etcd.

2 The data plane, including your workloads data.

3 Both the control and data planes.

Google Cloud