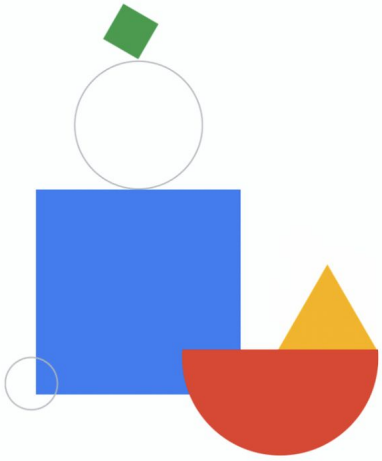
Developing Applications with Google Cloud: Foundations

Module 8: Monitoring and performance tuning



Welcome to Developing Applications with Google Cloud: Foundations, module 8: Monitoring and performance tuning.

Agenda

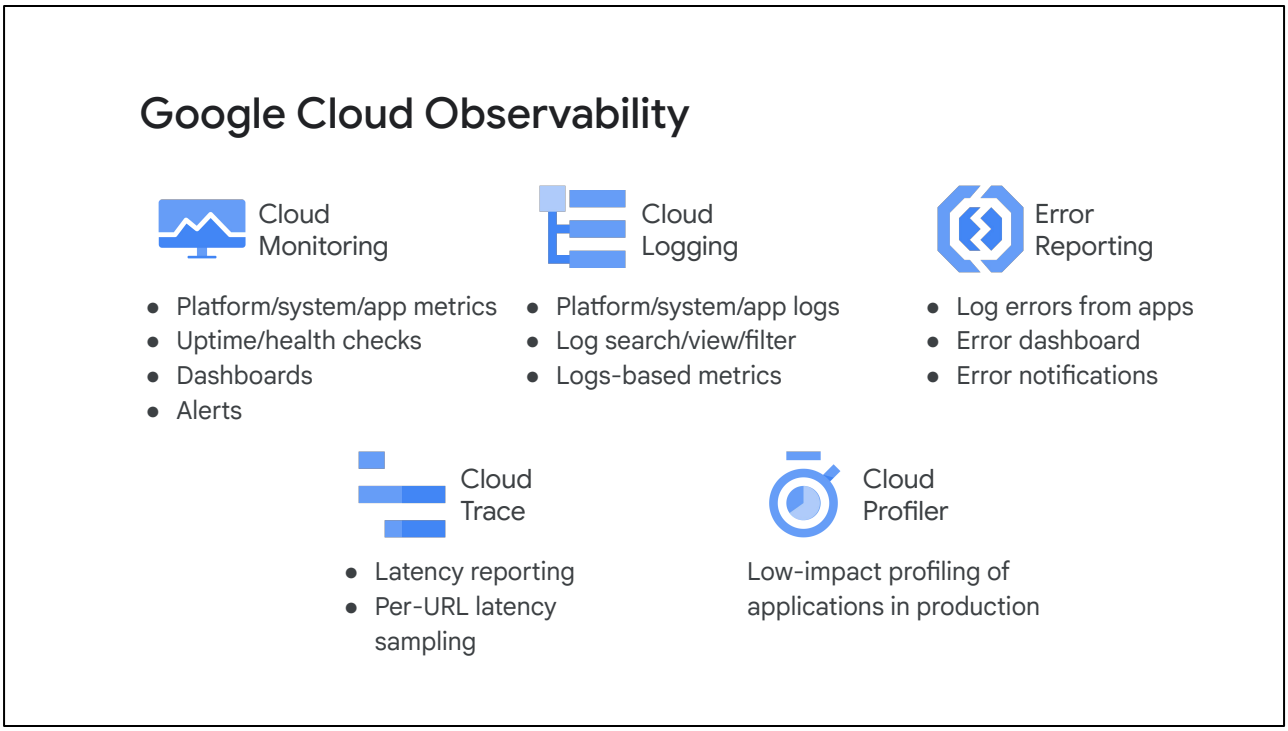
01	Google Cloud Observability
02	Logging and metrics
03	Error Reporting
04	Managing performance

Google Cloud Observability combines metrics, logs, and metadata, whether you're running on Google Cloud, Amazon Web Services, on-premises infrastructure, or a hybrid cloud. You can quickly understand service behaviors and issues, from a single comprehensive view of your environment, and act if needed.

Agenda

01	Google Cloud Observability
02	Logging and metrics
03	Error Reporting
04	Managing performance

We start by looking at the components of Google Cloud Observability.



The integrated managed services in Google Cloud Observability help you manage your running services and applications.

Cloud Monitoring provides visibility into the performance, availability, and overall health of cloud applications. You can collect metrics, events, and metadata from Google Cloud services and applications, and create alerting policies to provide timely awareness of problems in your applications.

Cloud Logging is a fully managed service that ingests application and platform log data, as well as custom log data from GKE environments, VMs, and other services inside and outside of Google Cloud. Cloud Logging lets you search and filter your logs, and provides you the ability to troubleshoot and analyze log data for your applications.

Error Reporting counts, analyzes, and aggregates errors produced in your running cloud services. You add the error reporting library to your application, and then report an error when it occurs. The Error Reporting dashboard displays a summary of each error found and the number of occurrences of each error, and notifications can be sent when a new or resolved error occurs.

Cloud Trace is a distributed tracing system for Google Cloud that collects latency data from applications and displays it in near real-time in the Google Cloud console. Cloud Trace helps you understand how long it takes your application to handle incoming requests and analyzes latency across services in the application.

Cloud Profiler uses statistical techniques, and low-impact instrumentation that runs across all production application instances to provide a complete picture of an application's performance without slowing it down. Cloud Profiler helps you identify and eliminate potential performance issues.

Why you should monitor your applications

Cloud Monitoring

- Build dashboards to answer basic questions about your application.
- Show trends in application usage patterns.
- Alert on what needs urgent attention.
- Conduct retrospective analysis.

Cloud Monitoring helps increase reliability by giving users the ability to monitor Google Cloud and multi-cloud environments to identify trends and prevent issues. With Cloud Monitoring, you can reduce monitoring overhead and improve your signal-to-noise ratio, letting you detect and fix problems faster.

Why should application developers care about monitoring their applications? Monitoring is the foundation of application reliability.

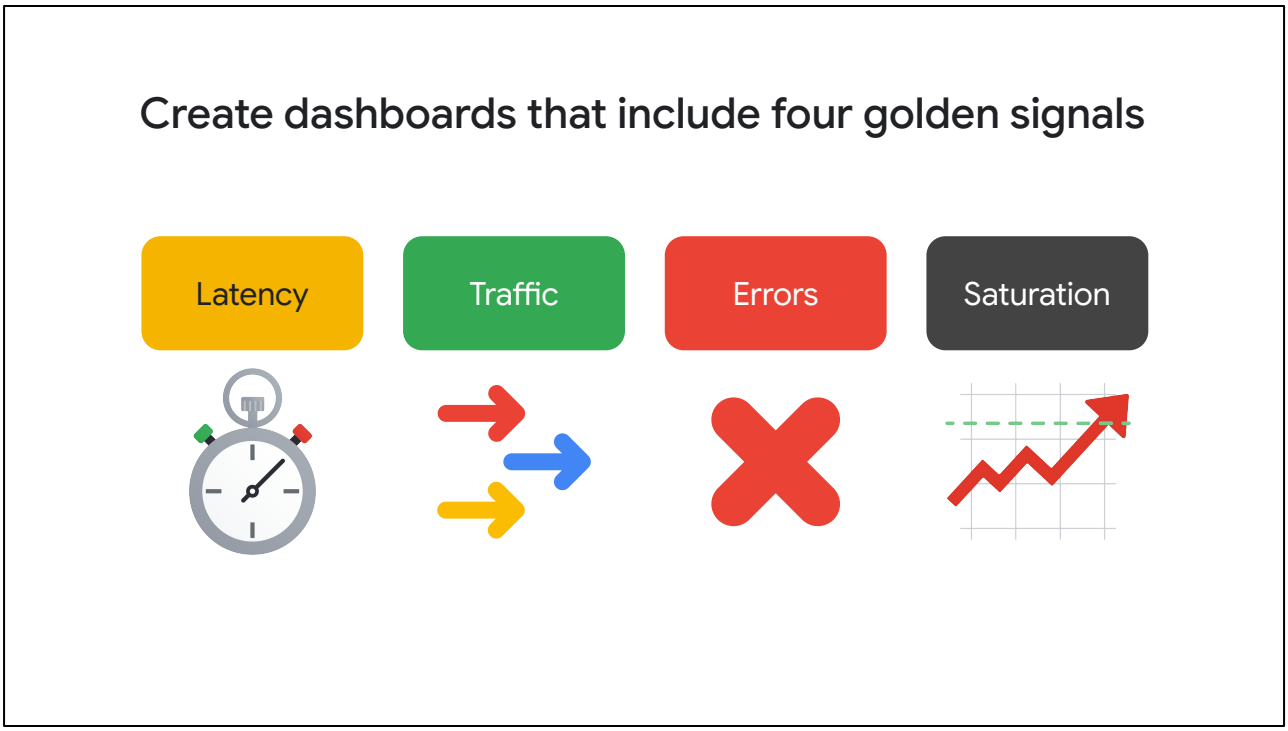
With Cloud Monitoring, you can build custom dashboards and use out-of-the-box dashboards to answer basic questions about your application. You can monitor your apps and infrastructure, whether they are running in Google Cloud, on-premises, or in other clouds. These dashboards let you monitor the health of your infrastructure and applications and easily spot anomalies.

Monitoring over time also lets you see trends in application usage patterns. How large is my database, and how fast is it growing? How quickly is my daily active user count growing? Which features of my application are used the most?

Monitoring also can tell you what needs urgent attention. Is the application broken, or will it soon break? Alerting on metric trends or limits lets you be notified of issues and fix them before they cause catastrophic failures.

Monitoring can also be useful for conducting retrospective analysis. The latency of our application increased dramatically overnight -- what else happened around the same

time? Is there a condition that I can alert on to let me fix the issue before it affects users?



At a minimum, there are some key metrics that you should capture for your applications.

You should create application dashboards that include the **four golden signals: Latency, traffic, errors, and saturation**.

[The Four Golden Signals: https://sre.google/sre-book/monitoring-distributed-systems/#xref_monitoring_golden-signals]

- **Latency** is the amount of time that it takes to serve a request. Make sure to distinguish between the latency of **successful** and **unsuccessful** requests. For example, an HTTP error that occurs due to a loss of connection to a database or another backend service, might be solved really quickly. However, because an HTTP 500 error indicates a failed request, including 500 errors in your overall latency might result in misleading metrics.
- **Traffic** is a measure of how much demand is placed on your system. It's measured as a system-specific metric. For example, web server traffic is measured as the number of HTTP or HTTPS requests per second. Traffic to a NoSQL database is measured as the number of read or write operations per second.
- **Errors** indicate the number of **failed** requests. Criteria for failure might be anything like an explicit error, such as an HTTP 500 error, or a successful HTTP 200 response but with incorrect content. It might also be a policy error.

- For example, your application promises a response time of one second, but some requests take over a second.
- **Saturation** indicates how full your application is, or what resources are being stretched and reaching target capacity. Systems can degrade in performance before they achieve 100% utilization, so make sure to set utilization targets carefully.

Agenda

01	Google Cloud Observability
02	Logging and metrics
03	Error Reporting
04	Managing performance

Cloud Logging is one of the tools that can be most valuable for app developers. Logs and metrics will help you understand how your application is working.

Why you should log

Cloud Logging

- Troubleshoot your application.
- Analyze application performance.
- Alert when things go wrong.
- Monitor trends over time.

A robust system of logging is crucial for developer productivity and to help you understand the state of your application.

Cloud Logging is a real-time log-management system with storage, search, analysis, and monitoring support. Cloud Logging automatically collects logs from Google Cloud resources. You can also collect logs from your applications. Logging is an important tool for developers.

Logs can help you troubleshoot your application. Using the Logs Explorer you can view individual log entries and find related log entries. Logging details about the flow of calls within your application helps you understand how your application is working, or why it's not working as intended.

Logs can also be used to analyze application performance. For example, you can use the Log Analytics interface to perform aggregate operations on your logs. With this interface, you use SQL to query your log data and help you understand performance.

You can configure Cloud Logging to notify you when certain kinds of events occur in your logs. You can create a log-based alert to send a notification when a particular pattern appears in a log entry.

Alternatively, you might want to monitor trends or the occurrence of events over time. For these situations, you can create a log-based metric. Log-based metrics are

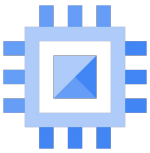
suitable whenever you want to count the occurrences of a message and be notified when the number of occurrences crosses a threshold. Metrics can also be used to observe trends in your data, and receive a notification if the values change in an unacceptable way.

[Cloud Logging overview: https://cloud.google.com/logging/docs/overview]

Install Ops Agent to capture logs in VMs

Ops Agent

- High throughput logging
  - Standard system logs
  - Customized file-based logs
  - Flexible processing



Compute Engine

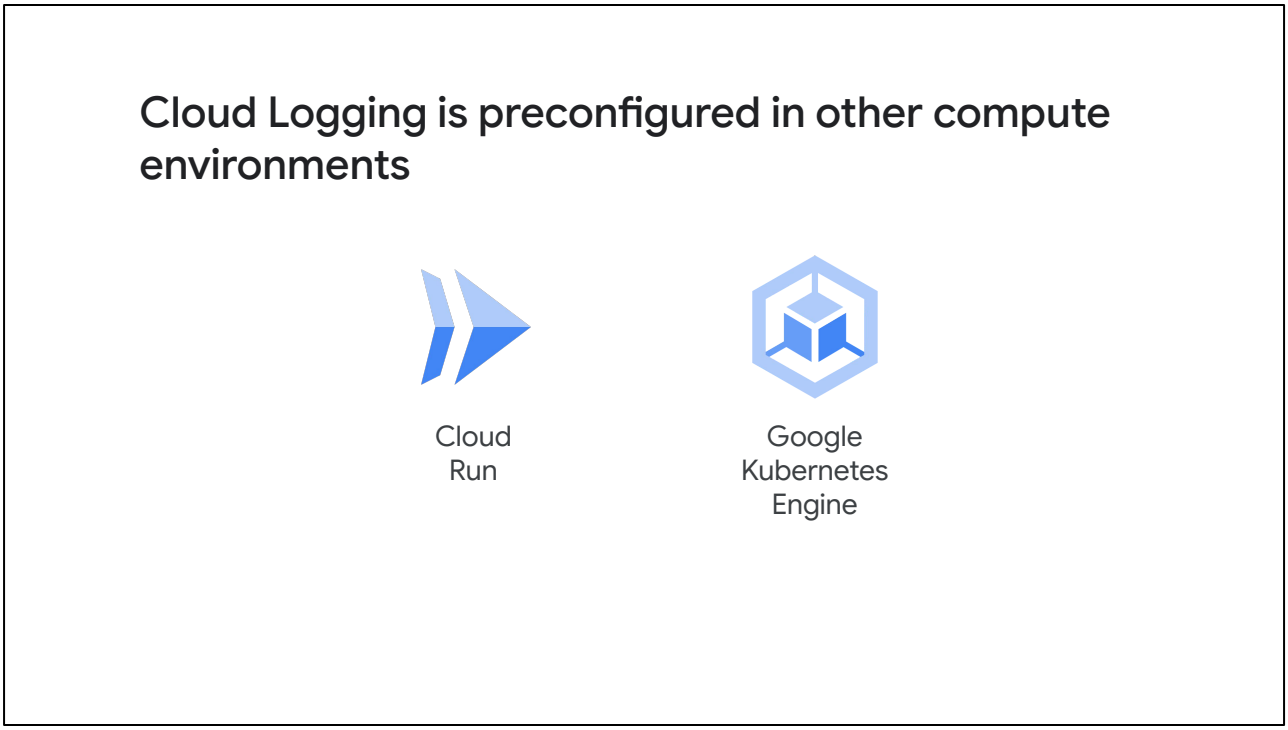
- Monitoring metrics
  - System metrics (CPU, disk, memory, network, processes, ...)
  - Third-party metrics support

You can install the Ops Agent on Compute Engine instances to stream logs and metrics from third-party applications into Cloud Logging. The Ops Agent uses Fluent Bit for high throughput logging, and the OpenTelemetry Collector for metrics.

Ops Agent automatically collects logs from standard system log locations like /var/log/syslog. File-based logs can also be configured, with customizable paths and refresh interval. Ops Agent also supports flexible processing of logged data. You can configure Ops Agent to parse text logs into structured logs, modify log entries by removing, renaming, or setting fields, or exclude logs based on labels and regular expressions.

Ops Agent also supports collection of standard system metrics without any configuration. Collected system metrics include CPU, disk, memory, network, and processes. Curated third-party application metrics can also be collected. Apache Tomcat, Apache Web Server, and nginx are examples of third-party applications that are supported by Ops Agent.

[Ops Agent: https://cloud.google.com/stackdriver/docs/solutions/agents/ops-agent
Fluent Bit: https://fluentbit.io
OpenTelemetry Collector: https://opentelemetry.io/docs/collector/]



Cloud Logging is pre-configured in other compute environments.

Cloud Run services and functions have built-in support for logging. Logs written by applications are automatically sent to Cloud Logging.

You can also enable logging on Google Kubernetes Engine by enabling the Observability for GKE integration for your cluster. Kubernetes logs are not stored permanently within GKE. For example, GKE container logs are removed when their host Pod is removed. System logs are periodically removed to free up space for new logs, and cluster events are removed after one hour. Sending logs to Cloud Logging ensures that log entries are kept as long as you deem necessary.

Writing text and structured logs


```
// logging text strings to stdout and stderr (textPayload)
console.log('This is a log entry.');

// writing structured logs (jsonPayload)
const globalLogFields = {};
globalLogFields['logging.googleapis.com/labels'] =
  {"appid":"inventory","contact":"commerce@example.org"}

const entry = Object.assign(
  {
    severity: 'NOTICE',
    message: 'Inventory system rebooted.',
    component: 'inventory-tablet-service'
  },
  globalLogFields
);
console.log(JSON.stringify(entry));
```

For Cloud Run services and functions, you can simply write to stdout and stderr, and the logs will be automatically delivered to Cloud Logging. Text strings are put into the textPayload field of the log entry. Other fields like the time the log was received, the resource that produced the log entry, and the log name will also be logged with a text message.

It's recommended to use structured logs instead of text logs. Text logs do not have a log level, so it can be hard to find the really important content inside the text logs. Fields within structured log data can be searched using queries, which makes log analysis much easier. For a structured log entry, you log a single line of serialized JSON, which is placed in the jsonPayload field of the log entry.

When you use structured JSON data, some special fields are stripped from the jsonPayload and written to the corresponding field in the generated log entry. For example, you can set the log level by specifying a severity field. The message property will be used as the main display text of the log when it’s provided.

In this example, the first line would create a text log entry. The rest of the code shows an example of how to create a structured log entry. The structured entry uses special fields to create labels, the log severity, and a component field within the log entry.

Prometheus

- Is an open source systems monitoring and alerting toolkit
- Is a standard solution for Kubernetes workloads
- Collects and stores metrics as time series data
- Provides a powerful query language
- Can be difficult to manage infrastructure

Another tool you might want to consider for logging and alerting is Prometheus.

Prometheus is an open source systems monitoring and alerting toolkit. It can monitor services running on VMs and Kubernetes.

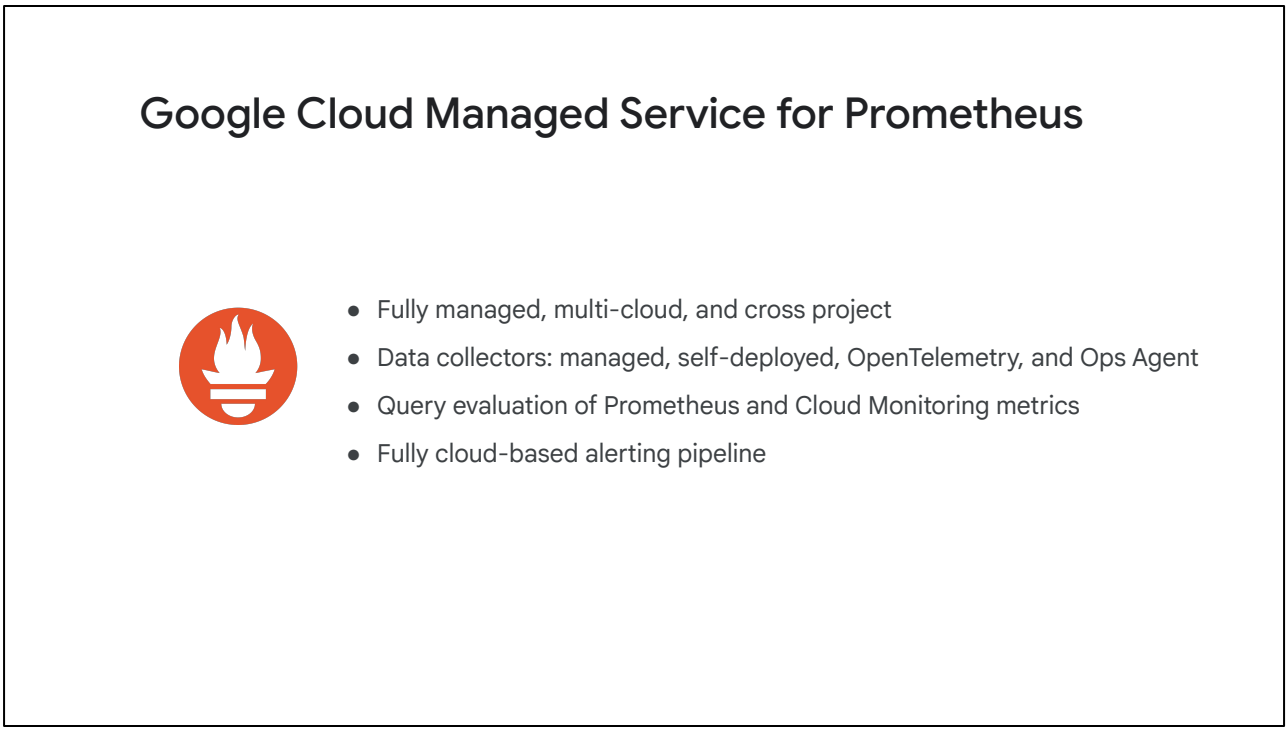
It's a very popular solution for monitoring Kubernetes workloads and clusters, and alerting when the workloads and clusters are not healthy.

Prometheus collects and stores metrics as time series data. This time series data can help you see trends in your application metrics.

Prometheus also provides a powerful query language, PromQL, which can be used to create dashboards and alerts from your metrics.

Prometheus provides many benefits, but it can be difficult to manage and scale Prometheus infrastructure. One solution to this problem is to use Google Cloud Managed Service for Prometheus.

[Prometheus: https://prometheus.io/]



Google Cloud Managed Service for Prometheus is a fully managed Prometheus solution that removes the burden of manually managing and operating Prometheus at scale. The solution is also multi-cloud and cross project. All of your applications, whether running in Google Cloud, other clouds, or on-premises, can be managed in a single pane of glass using Cloud Monitoring.

There are many data collectors available. Managed collectors are recommended for all Kubernetes environments, including GKE. For managed collectors, operation of Prometheus is fully handled by the Kubernetes operator. Self-deployed collectors are drop-in replacements for the normal Prometheus binary. OpenTelemetry Collectors and Ops Agent can also both collect metrics that can be used with Prometheus.

Managed Service for Prometheus supports any query UI that can call the PromQL query API, including Cloud Monitoring and Grafana. Over 1500 free metrics available in Cloud Monitoring can also be queried, even without sending data to Managed Service for Prometheus.

Managed Service for Prometheus also provides a fully cloud-based alerting pipeline for your Cloud Monitoring and Prometheus metrics.

[Google Cloud Managed Service for Prometheus:

https://cloud.google.com/stackdriver/docs/managed-prometheus
Google Cloud metrics: https://cloud.google.com/monitoring/api/metrics_gcp]

Agenda

01	Google Cloud Observability
02	Logging and metrics
03	Error Reporting
04	Managing performance

Error Reporting notifies you of errors in your application and helps you investigate the cause.

Why you should use Error Reporting

Error Reporting

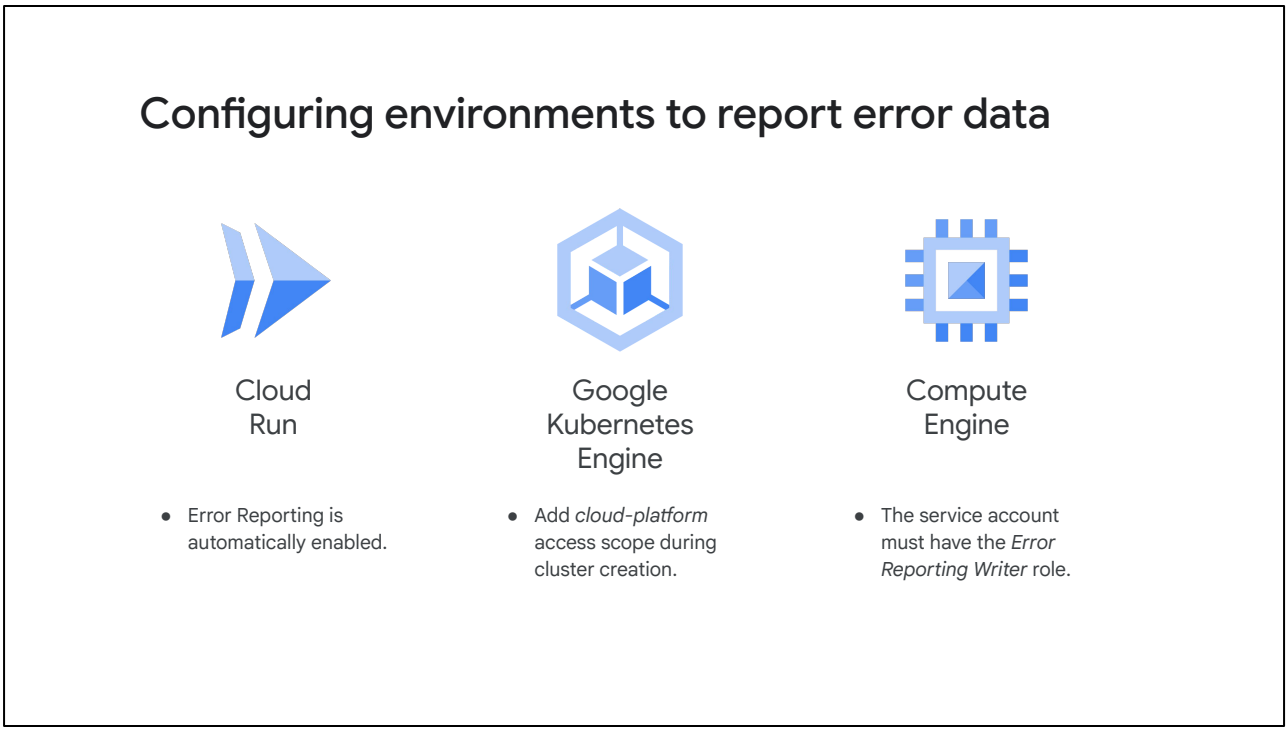
- Real-time exception monitoring and alerting
- Intelligent error grouping
- Stack trace exploration

Error Reporting counts, analyzes, and aggregates the crashes in your running cloud services. A centralized error management interface provides sorting and filtering capabilities, and shows error details such as timing, occurrences, first and last-seen dates, and number of affected users. You can opt in to receive email and mobile alerts when new errors are found.

Application error events are processed and displayed in the interface within seconds. Errors are either reported by the Error Reporting API or are inferred to be errors when Error Reporting inspects log entries for common text patterns such as stack traces. Error events are intelligently grouped into error groups and deduplicated by analyzing stack traces, helping you understand the different errors you're encountering regardless of the number of occurrences. You can see your application's top or new errors in a clear dashboard.

Stack traces are parsed and displayed in a way that helps you focus on what matters. The stack trace for an error event is displayed along with a list of occurrences within the error group, so it's easy to investigate similar error events at the same time.

[Error Reporting overview:
https://cloud.google.com/error-reporting/docs/grouping-errors
Troubleshoot errors with Duet AI assistance:
https://cloud.google.com/error-reporting/docs/troubleshoot-errors-duet-ai]



You might need to enable Error Reporting, depending on where your service is running.

Error Reporting is automatically enabled for Cloud Run services and functions.

You can enable Error Reporting on Google Kubernetes Engine by adding the *cloud-platform* access scope when you create the cluster.

Apps running on Compute Engine will report error events by granting the VM's service account the *Error Reporting Writer* role.

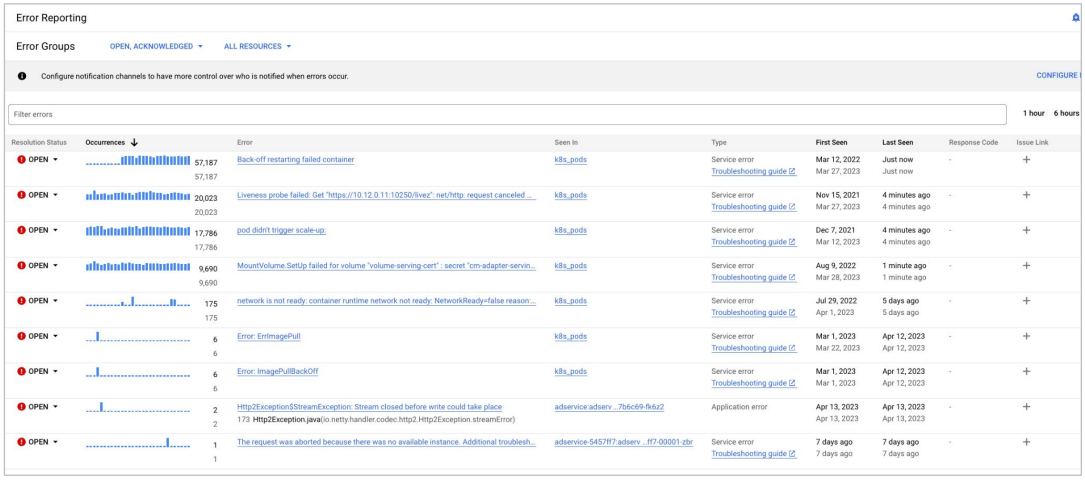
Reporting errors in a Node.js application

```
// import the library
const {ErrorReporting} = require('@google-cloud/error-reporting');

// create a client
const errors = new ErrorReporting();

// create a new error event
const errorEvent = errors.event();

// add error details
errorEvent.setMessage('My error message');
errorEvent.setUser('user@example.com');

// report the error event
errors.report(errorEvent, () => {
  console.log('Done reporting error!');
});
```

Error Reporting can be used in many popular languages, including Go, Java, Node.js, PHP, Python, Ruby, or .NET. You can use client libraries, the REST API, or send errors with Cloud Logging.

Here's an example of using the client library to report errors in a Node.js app. After including the error-reporting library, you just need to create a client and a new error event, add the details to the error event, and then report the error. The error event is reported asynchronously so that your code can continue processing without waiting for delivery of the error event.

Managing error groups



To see your errors, open the Error Reporting page in the Google Cloud console.

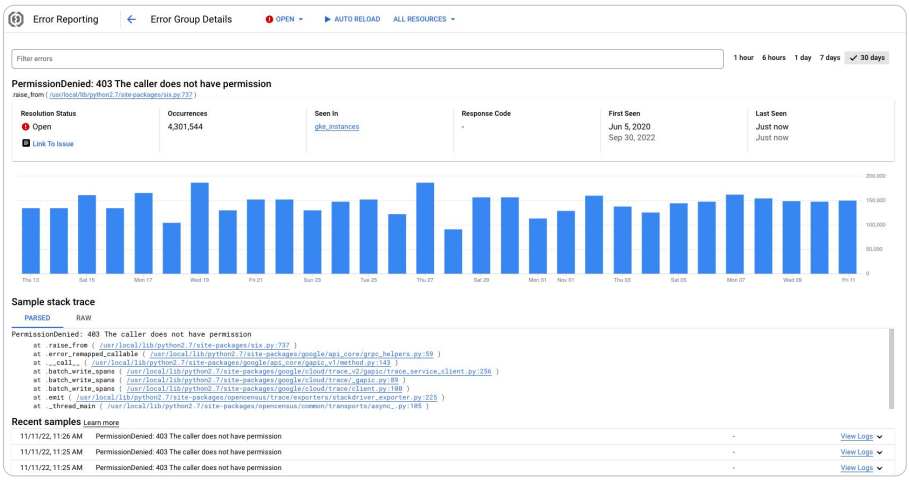
By default, Error Reporting shows you a list of recently occurring open and acknowledged errors in order of frequency.

Errors are grouped and de-duplicated by analyzing their stack traces.

Error Reporting recognizes the common frameworks used for your language and groups errors accordingly.

You can sort errors based on number of occurrences or by first and last seen. You can also link an issue tracker link to an error group.

Exploring an error group



Selecting an error entry opens an error details page.

On this page, you can examine information about the error group, including the number of occurrences over time, specific error events, and stack traces.

To view the log entry associated with a sample error, click View Logs from any entry in the Recent samples panel, and you will be taken to Logs Explorer.

Agenda

01	Google Cloud Observability
02	Logging
03	Error Reporting
04	Managing Performance

Google Cloud provides other tools that can help you manage the performance and stability of your application.

Application Performance Management (APM) tools

Cloud Trace

Cloud Profiler

Cloud Trace is a distributed tracing system that collects latency data from your applications and displays it in the Google Cloud console. You can track how requests propagate through your application and receive detailed near-real time performance insights.

Cloud Profiler is a statistical, low-overhead profiler that continuously gathers CPU usage and memory-allocation information from your applications. It attributes that information to the source code that generated it, which helps you identify the parts of your application that are consuming the most resources.

First, we look at Cloud Trace.

Why you should use Cloud Trace

Cloud Trace

- Is a distributed tracing system that collects and analyzes latency in your apps.
- Reports latency data on a per URL basis.
- Identifies changes in performance.

Cloud Trace is a distributed tracing system that collects and analyzes latency in your applications. It helps you understand how long it takes your application to handle incoming requests, and how long it takes to complete operations like RPC calls when handling the requests.

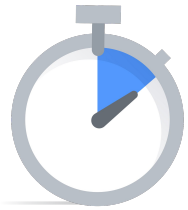
Latency data is reported on a per URL basis, letting you focus on the operations that are showing the most latency.

Cloud Trace can help you identify changes in performance. Cloud Trace automatically creates a daily report that compares the previous day's performance with the performance from the same day of the previous week for the top endpoints. You can also create a custom analysis report and select which traces are included in the report.

[Cloud Trace overview: https://cloud.google.com/trace/docs/overview]

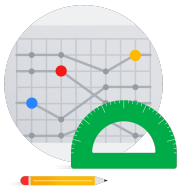
Sending trace data to Cloud Trace

Automatic tracing



Latency data is automatically collected for incoming and outgoing HTTP requests from Cloud Run services and functions

Instrumenting the application



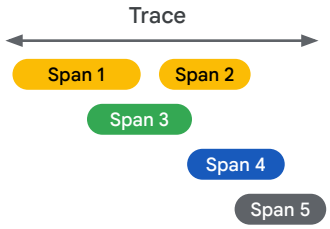
Use OpenTelemetry (recommended), the Cloud Trace API, or Google's Cloud Trace client libraries

There are two ways to send trace data to Cloud Trace.

The first is automatic tracing. Some configurations support automatic tracing. Latency data for incoming and outgoing HTTP requests from Cloud Run services and functions is automatically collected. However, latency data within the services is not collected.

You can also instrument the application for tracing, which provides more detailed information than is collected for automatic tracing. You may choose to instrument applications that are running on Cloud Run. When it's available for your language, OpenTelemetry and the associated Cloud Trace exporter are the recommended solution. You can also use write custom methods to send tracing data by using the Cloud Trace API, or use the Cloud Trace client libraries.

Traces and spans



- A trace describes the time that it takes to complete a single operation.
- A span describes the time that it takes to complete a sub operation.
- A trace consists of one or more spans.

A tracing client collects timing data and sends it to Cloud Trace. You then use the Google Cloud console to view and analyze the data.

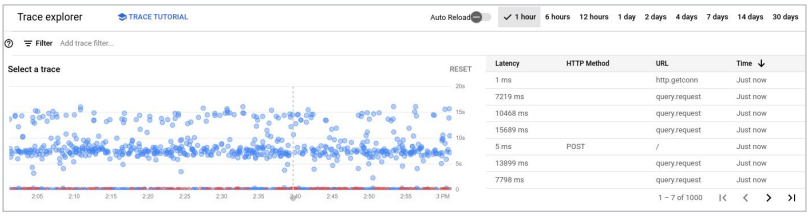
A trace describes the time that it takes to complete a single operation.

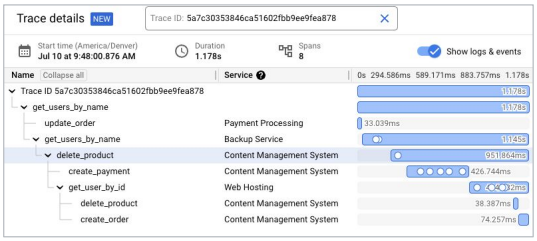
A span describes the time that it takes to complete a sub operation within the trace.

A trace consists of one or more spans.

Viewing trace details

You can view and filter traces...





...and then explore a specific trace, including the spans contained in that trace.

Visualization of traces and spans can help you track down performance issues in your applications.

The **Trace explorer** page lets you find and examine individual traces in detail. The scatter plot displays a dot for each request in the selected time interval. The (x,y) coordinates for a request correspond to the time and latency of the request. You can filter the shown requests by request attributes like methods or status codes.

To explore a trace, you click a dot in the scatter plot. The **Trace details** pane displays details about this trace and the spans contained within the trace. The dots on a span indicate events that were annotated during completion of the span sub operation. The visual indication of latency can help you determine which parts of your application are causing the performance issues.

Why you should use Cloud Profiler

Cloud Profiler

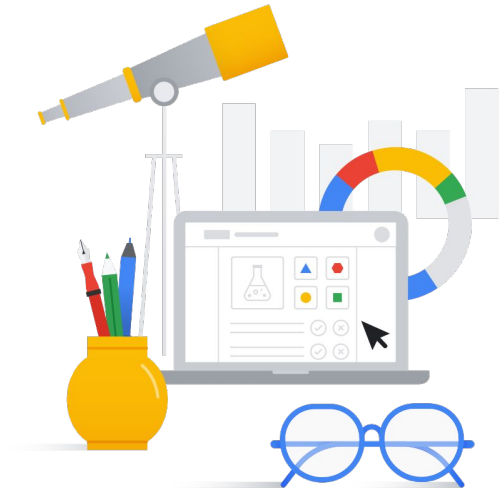
- Low-overhead profiler that continuously gathers CPU and memory usage information about your application.
- Attributes that information to the source code that generated it.

Understanding the performance of production systems is notoriously difficult. Attempting to measure performance in test environments often fails to replicate the characteristics of the production environment.

Cloud Profiler is a statistical, low-overhead profiler that continuously gathers CPU usage and memory-allocation information from your production applications.

Cloud Profiler attributes that information to the source code that generated it, which helps you identify the parts of your application that are consuming the most resources.

[Cloud Profiler overview: https://cloud.google.com/profiler/docs/about-profiler]



Hands-on lab:

Deploying and Maintaining Your Application

In this lab, "Deploying and Maintaining Your Application," you containerize your bookshelf application and deploy it to Cloud Run. You also explore tools in Google Cloud Observability that can help you maintain the health of your application.

In this module, you learned ...

- **Cloud Monitoring** can identify trends and help you fix issues in your applications.
- It is recommended you monitor the four golden signals for your applications:
  - latency
  - traffic
  - errors
  - saturation

The services in Google Cloud Observability help you manage your running services and applications.

Cloud Monitoring can identify trends and help you fix issues in your applications.

It's recommended that you monitor the four golden signals for your applications: latency, traffic, errors, and saturation.

In this module, you learned ...

- Logs in **Cloud Logging** can help you to understand the current state of your application.
- You can use **Error Reporting** to analyze and aggregate the crashes in your running cloud services.
- **Cloud Trace** and **Cloud Profiler** are used to analyze and improve the performance of your applications in production environments.

Logs in **Cloud Logging** can help you to understand the current state of your application.

You can use **Error Reporting** to analyze and aggregate the crashes in your running cloud services.

Cloud Trace and **Cloud Profiler** are used to analyze and improve the performance of your applications in production environments.