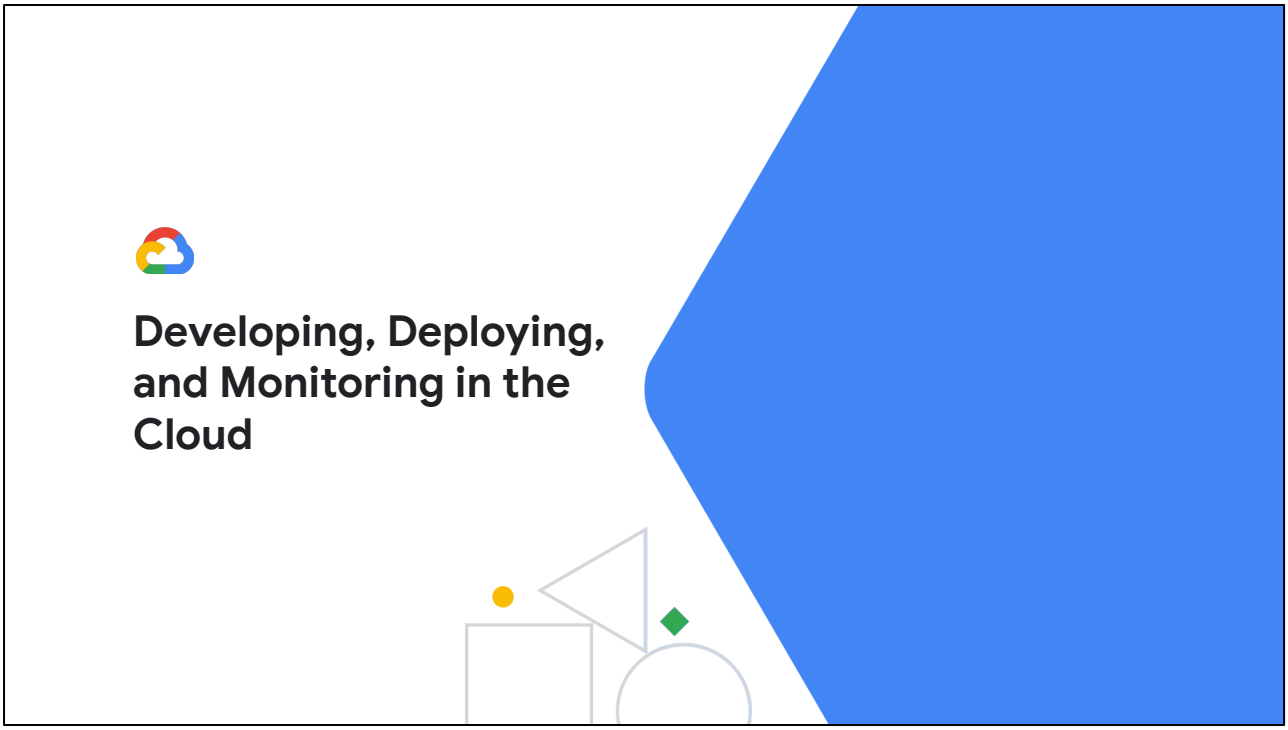


People create great applications in the Google Cloud. Popular tools for development, deployment, and monitoring work in Google Cloud. You also have options for tools that are tightly integrated with Google Cloud, and in this module I'll explain them.

Agenda

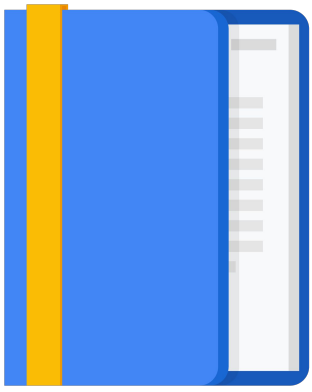
Development in the Cloud

Deployment: Infrastructure as Code

Monitoring: Proactive instrumentation

Lab

Resources



Google Cloud

Cloud Source Repositories

- Fully featured Git repositories hosted on Google Cloud.
- Supports collaborative development of cloud apps.
- Includes integration with Cloud Debugger.

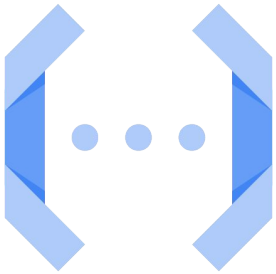
Google Cloud

Cloud Source Repositories provides Git version control to support collaborative development of any application or service, including those that run on App Engine and Compute Engine. If you are using Cloud Debugger, you can use Cloud Source Repositories and related tools to view debugging information alongside your code during application runtime. Cloud Source Repositories also provides a source viewer that you can use to browse and view repository files from within the Cloud Console.

With Cloud Source Repositories, you can have any number of private Git repositories, which allows you to organize the code associated with your cloud project in whatever way works best for you. Google Cloud diagnostics tools like Debugger and Error Reporting can use the code from your Git repositories to let you track down issues to specific errors in your deployed code without slowing down your users. If you've already got your code in GitHub or BitBucket repositories, you can bring that into your cloud project and use it just like any other repository, including browsing and diagnostics.

Cloud Functions

- Create single-purpose functions that respond to events without a server or runtime.
  - Event examples: New instance created, file added to Cloud Storage.
- Written in Javascript (Node.js), Python or Go; execute in managed Node.js environment on Google Cloud.



Google Cloud

Cloud Functions is a lightweight, event-based, asynchronous compute solution that allows you to create small, single-purpose functions that respond to cloud events without the need to manage a server or a runtime environment. You can use these functions to construct applications from bite-sized business logic. You can also use Cloud Functions to connect and extend cloud services.

You are billed, to the nearest 100 milliseconds, only while your code is running.

Cloud Functions are written in Javascript (Node.js), Python or Go and execute in a managed Node.js environment on Google Cloud. Events from Cloud Storage and Pub/Sub can trigger Cloud Functions asynchronously, or you can use HTTP invocation for synchronous execution.

Cloud Events are things that happen in your cloud environment. These might be things like changes to data in a database, files added to a storage system, or a new virtual machine instance being created.

Events occur whether or not you choose to respond to them. Creating a response to an event is done with a trigger. A trigger is a declaration that you are interested in a certain event or set of events. You create triggers to capture events and act on them.

Agenda

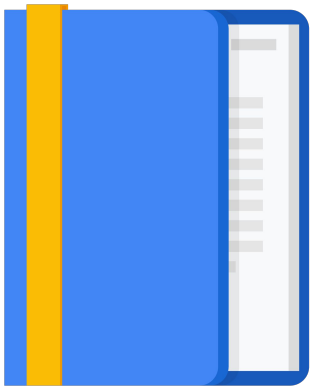
Development in the Cloud

Deployment: Infrastructure as Code

Monitoring: Proactive instrumentation

Lab

Resources



Google Cloud

Cloud Deployment Manager

- Infrastructure management service.
- Create a .yaml template describing your environment and use Deployment Manager to create resources.
- Provides repeatable deployments.

Google Cloud

Cloud Deployment Manager is an infrastructure management service that automates the creation and management of your Google Cloud resources for you.

Setting up your environment in Google Cloud can entail many steps: setting up compute, network, and storage resources and keeping track of their configurations. You can do it all by hand if you want to, taking an imperative approach. But it is more efficient to use a template. That means a specification of what the environment should look like, declarative rather than imperative.

Google Cloud provides Deployment Manager to let do just that. It's an infrastructure management service that automates the creation and management of your Google Cloud resources for you.

To use Deployment Manager, you create a template file, using either the YAML markup language or Python, that describes what you want the components of your environment to look like. Then you give the template to Deployment Manager, which figures out and does the actions needed to create the environment your template describes. If you need to change your environment, edit your template, and then tell Deployment Manager to update the environment to match the change.

Here's a tip: you can store and version-control your Deployment Manager templates in Cloud Source Repositories.

AWS CloudFormation is the Deployment Manager equivalent to manage infrastructure as code

- AWS CloudFormation: build reusable templates using JSON or YAML syntax.
- Deployment Manager: build a top-level deployment configuration in YAML and build reusable templates in Jinja or Python.
- Both: Define outputs for a template.

Google Cloud

You can build reusable AWS CloudFormation templates using JSON or YAML syntax.

With Deployment Manager, you can build a top-level deployment configuration in YAML and build reusable templates in Jinja or Python.

AWS CloudFormation and Cloud Deployment Manager both enable you to define outputs for a template. For example, you can expose the IP address of a database created in a template so that users can easily reference the IP address when creating their own templates.

Manage infrastructure with conditions to create or omit certain resources

Cloud Deployment Manager	AWS CloudFormation

Construct separate configurations with reusable templates:
 	
Webapp.yaml
properties: envtype
Webapptemplate.py
if context.env['envtype' == 'Test']:
# Use n1-standard-1 	
else:
# Use n1-highcpu-8 	
	Create resources based on results of a certain condition:
	
	Webapp.template
	Condition: Test -> Use t2.small
	Condition: Prod -> Use m4.xlarge

Google Cloud

Both Deployment Manager and AWS CloudFormation enable you to manage infrastructure with conditions to create or omit certain resources.

In Deployment Manager, you can set a property in the deployment configuration file. You can pass this property to a template that launches a different VM based on the value of the property.

In AWS CloudFormation, you achieve the same functionality by defining a template with conditions to create or omit certain resources. For example, consider an AWS CloudFormation template that creates resources for a web application environment. You can add conditions to use different VM types for the test stack and production stack.

Comparing deployment technologies (1/2)

	Cloud Deployment Manager	AWS CloudFormation
Instance of configuration	Called a deployment	Called a stack
Resources	Define Google Cloud resources to create in the deployment	Define AWS resources to create in the stack
Metadata	Define arbitrary name-value pairs and Deployment Manager–specific metadata keys	Define arbitrary name-value pairs and AWS CloudFormation–specific metadata keys

Google Cloud

Let's see how the two deployment technologies compare.

An instance of a configuration created using Deployment Manager is called a *deployment*. An instance of a configuration created using AWS CloudFormation is called a *stack*.

Both Deployment Manager and AWS CloudFormation allow you to define the resources to create in the instance, and

Both enable you to define arbitrary name-value pairs and technology-specific metadata key-value pairs.

Comparing deployment technologies (2/2)

	Cloud Deployment Manager	AWS CloudFormation
Run script when instance launches	Specify startup scripts with startup-script metadata key	Specify user-data scripts
Dependencies between resources	Specify relationships with dependsOn metadata key or use references to use attributes of one resource in another	Specify DependsOn attribute for resource
Parameters and environment variables	Define template properties and use predefined environment variables	Define parameters to a template and use pseudo-parameters

Google Cloud

In AWS CloudFormation templates, you can specify user-data scripts to run when launching Amazon EC2 instances. In a Deployment Manager configuration, you can define a metadata key called startup-script to use with inline commands or a script file to run when the vm launches.

In both AWS CloudFormation and Deployment Manager, you can specify that a resource depends on the creation of one or more other resources.

In AWS CloudFormation templates, you can use predefined pseudo-parameters such as AWS::Region or define your own parameters to make your template flexible. In Deployment Manager, template properties can be passed in as parameters to templates. Environment variables are equivalent to pseudo-parameters.

Agenda

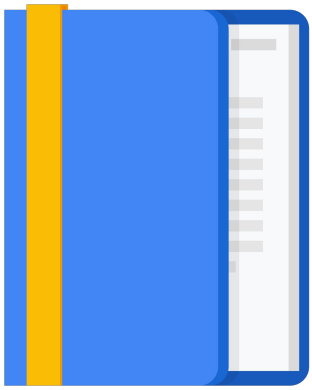
Development in the Cloud

Deployment: Infrastructure as Code

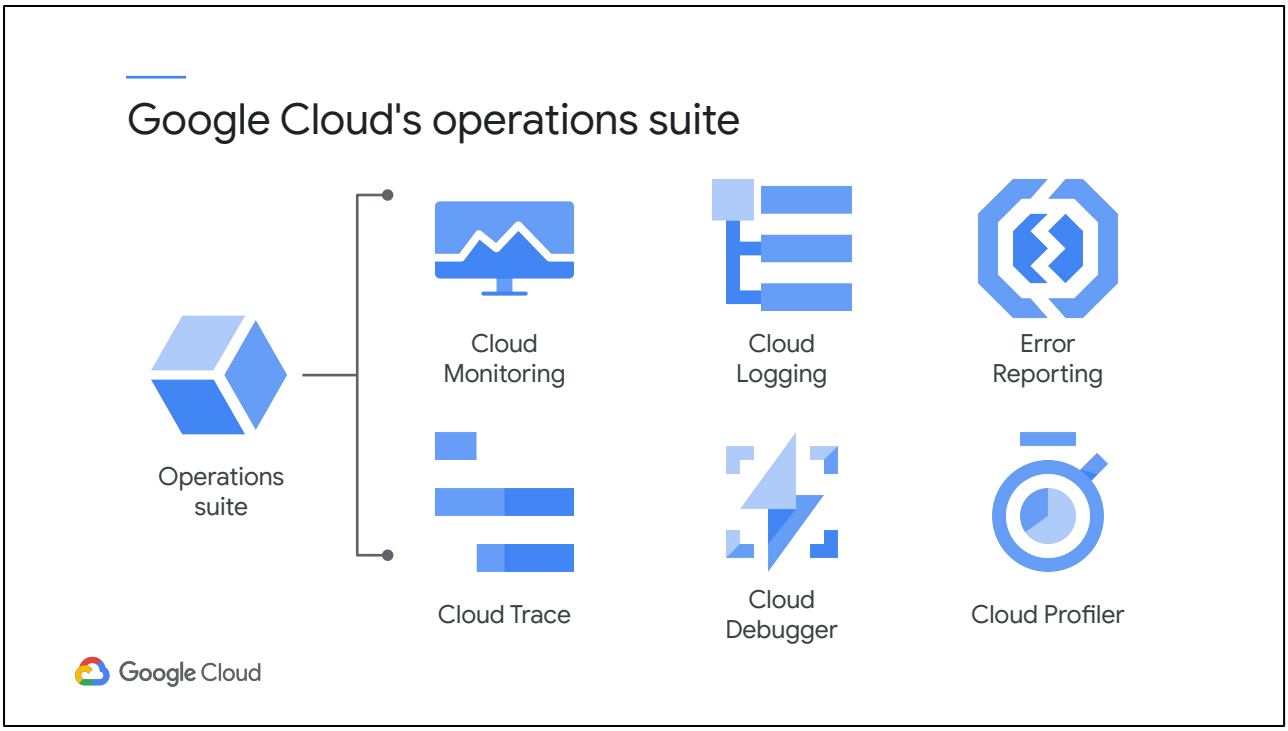
Monitoring: Proactive instrumentation

Lab

Resources

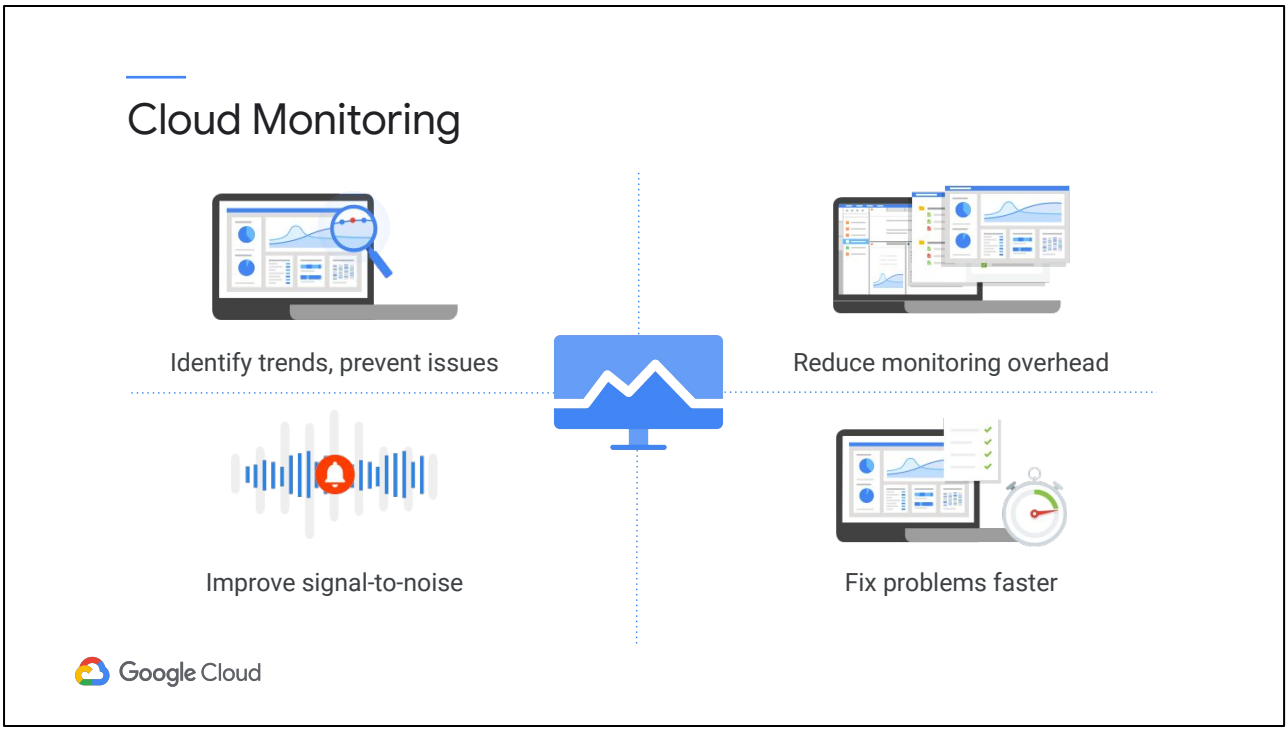


Google Cloud



Google Cloud's operations suite provides powerful monitoring, logging, and diagnostics for apps on Google Cloud. It equips you with insight into the health, performance, and availability of cloud-powered apps, enabling you to find and fix issues faster.

Google Cloud's operations suite gives you access to many different kinds of signals from your infrastructure platforms, virtual machines, containers, middleware, and application tier: logs, metrics, traces. It gives you insight into your application's health, performance, and availability, so if issues occur you can fix them faster.



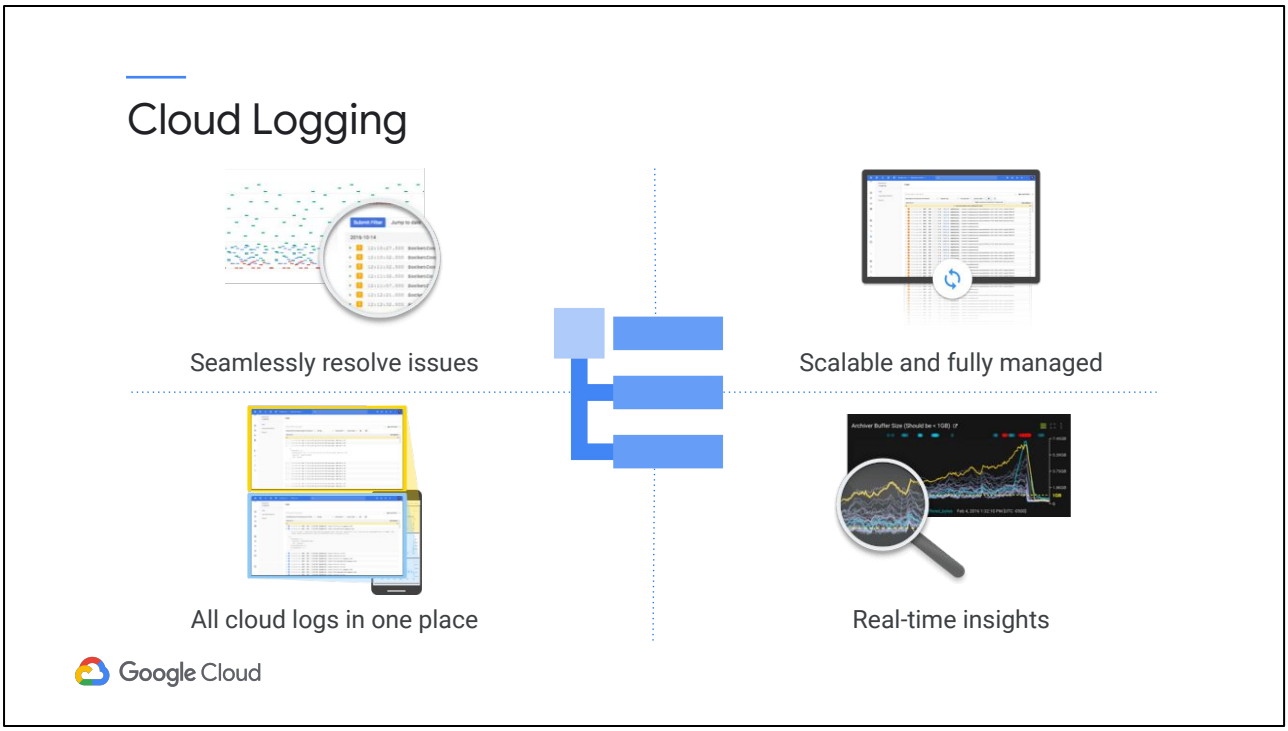
Let's start by looking at Cloud Monitoring, a full-stack monitoring service that discovers and monitors cloud resources automatically.

Flexible dashboards and rich visualization tools help you to identify emergent issues. Anomaly reporting, pattern detection, and exhaustion prediction provide insights into longer-term trends that may require attention.

Monitoring provides a single integrated service for metrics, dashboards, uptime monitoring, and alerting. This means you spend less time maintaining disparate systems.

Advanced alerting capabilities, including the rate of change, cluster aggregation, and multi-condition policies, help ensure you are notified when critical issues occur while reducing the likelihood of false positives.

Integrated uptime monitoring and health checks ensure quick notification of failures. It's possible to drill down from alerts to dashboards to logs and traces in order to identify the root cause of problems quickly.

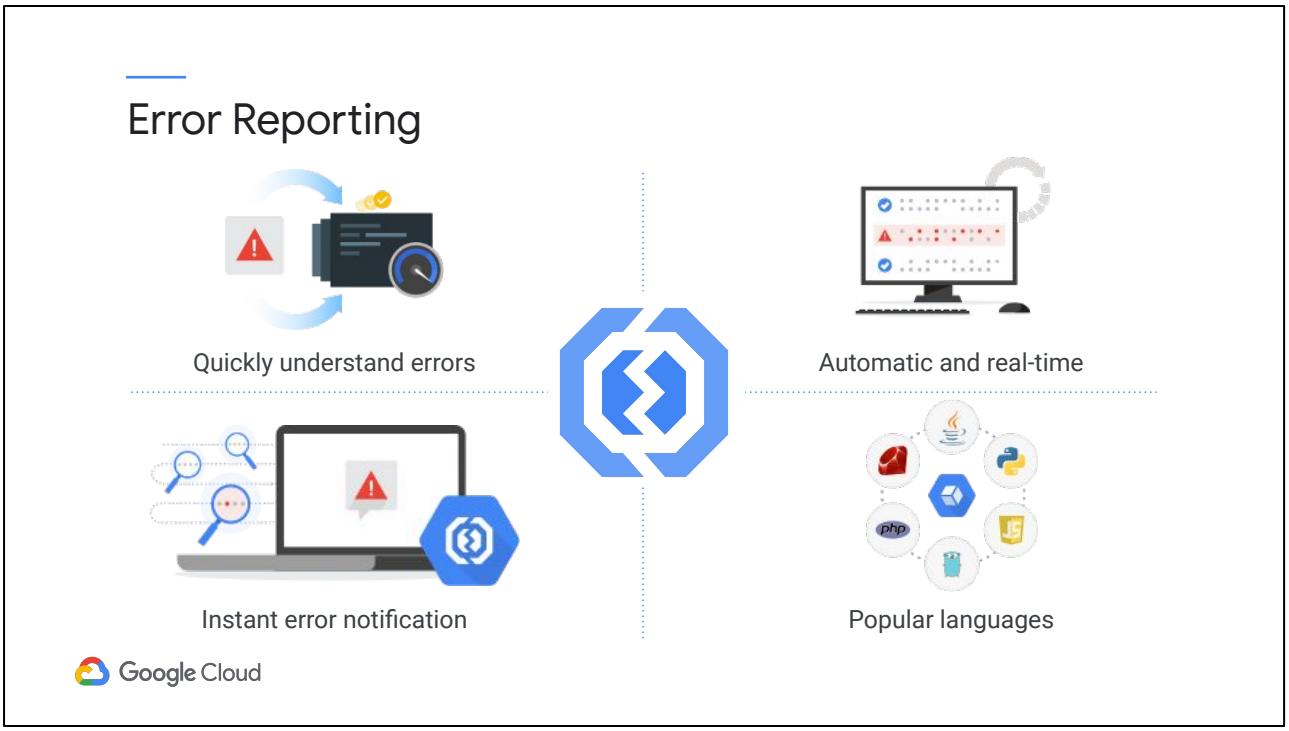


Cloud Logging is a fully integrated solution that works seamlessly with Cloud Monitoring, Trace, Error Reporting, and Debugger. The integration allows users to navigate between incidents, charts, traces, errors, and logs. This helps users quickly find the root causes of issues in their system and applications.

Logging is built to scale and works well at sub-second ingestion latency at terabytes per second. Logging is a fully managed solution that takes away the overhead of deploying or managing a cluster, thus allowing users to focus their energy on innovation and building a product.

Logging provides a central place for all your logs. You can also configure Logging to export logs to other systems automatically.

Logging allows you to analyze high-volume application and system logs in real time. Advanced log analysis can be achieved by combining the power of the operations suite with the data and analytics products of Google Cloud. For example, you can create powerful real-time metrics from the log data and analyze log data in real time in BigQuery.



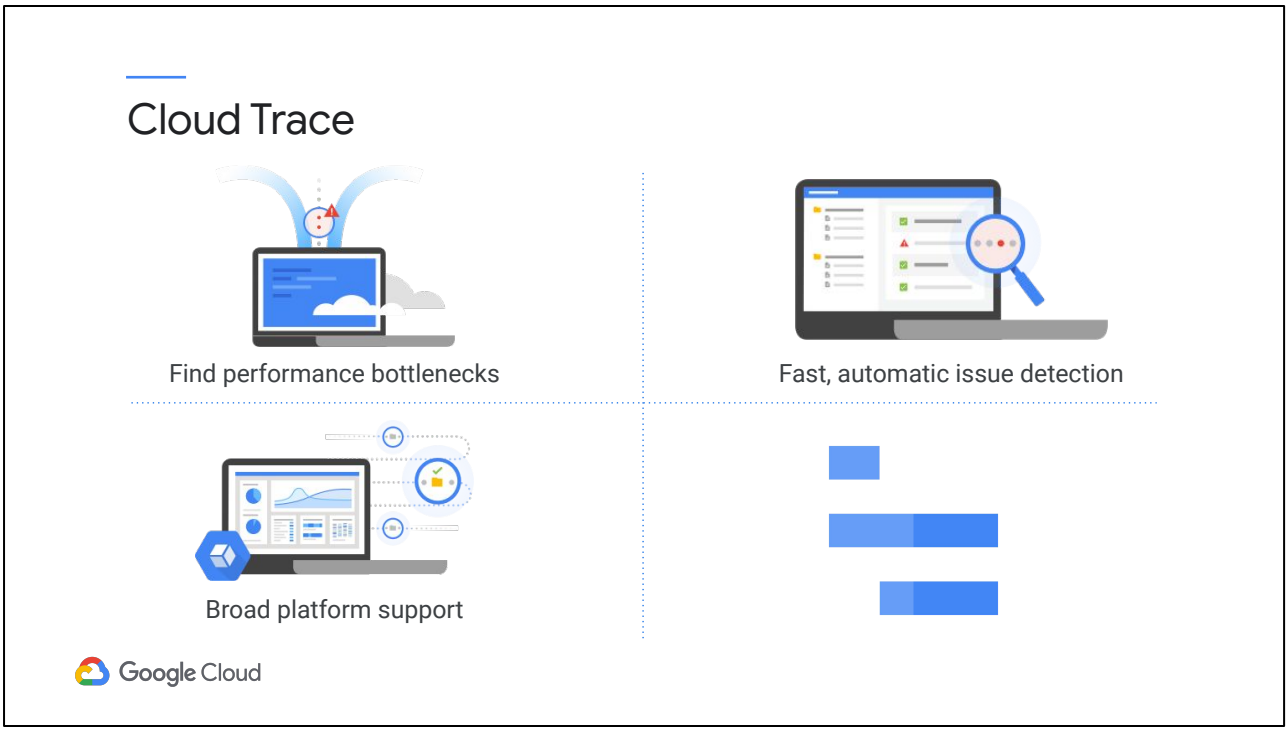
Error Reporting allows you to identify and understand application errors through real-time exception monitoring and alerting.

Error Reporting allows you to see your application's top errors in a dashboard.

Real production problems can be hidden in mountains of data. Error Reporting helps you see problems through the noise by constantly analyzing exceptions and intelligently aggregating them into meaningful groups tailored to your programming language and framework.

Error Reporting is constantly watching your service and instantly alerts you when a new application error cannot be grouped with existing ones. Directly jump from a notification to the details of the new error.

The exception stack trace parser is able to process Go, Java, .NET, Node.js, PHP, Python, and Ruby. You can also use Google's client libraries and REST APIs to send errors with Cloud Logging.

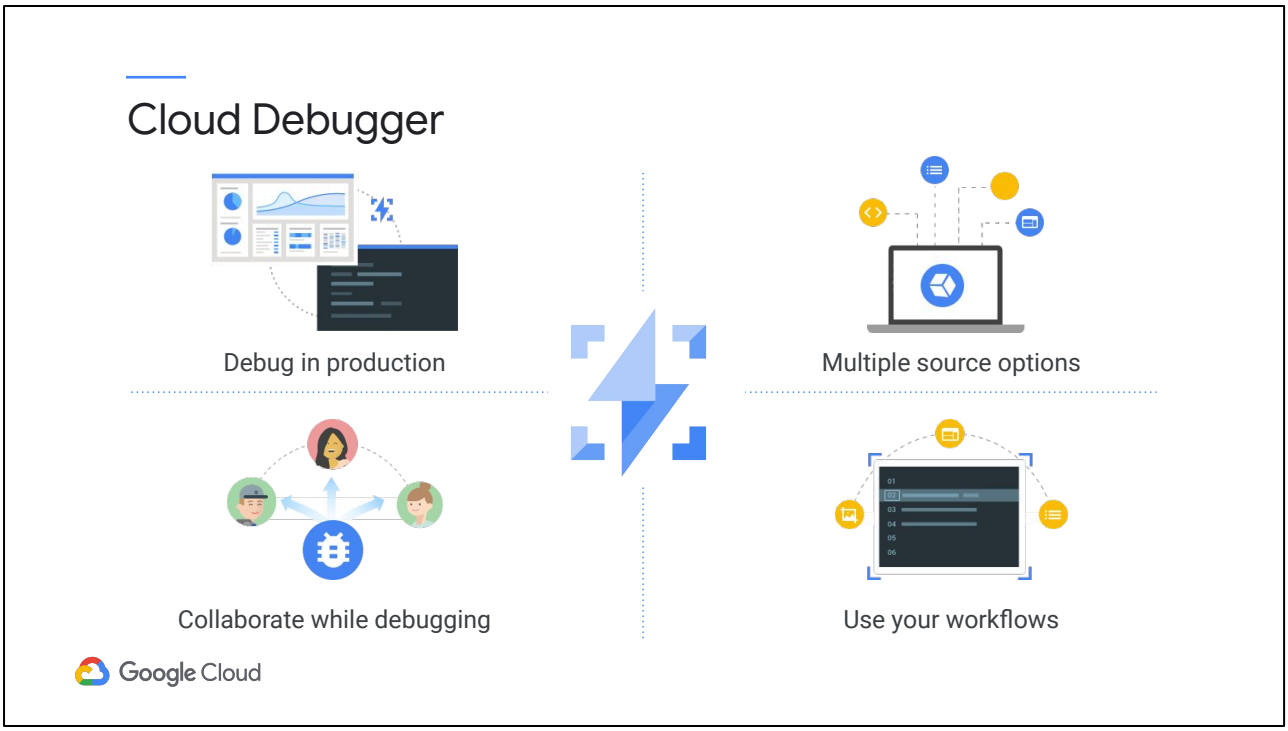


Cloud Trace is a distributed tracing system that collects latency data from applications and displays it in the Cloud Console.

Using Cloud Trace, you can inspect detailed latency information for a single request or view aggregate latency for your entire application. You can quickly find where bottlenecks are occurring and more quickly identify their root cause.

Trace continuously gathers and analyzes data from applications to automatically identify changes to an application's performance. These latency distributions, available through the Analysis Reports feature, can be compared over time or versions, and Trace will automatically generate an alert if it detects a significant shift in an application's latency profile.

The language-specific SDKs of Trace can analyze projects running on VMs. The Trace SDK is currently available for Java, Node.js, Ruby, and Go, and the Trace API can be used to submit and retrieve trace data from any source. A Zipkin collector is also available, which allows Zipkin tracers to submit data to Trace. Cloud Trace works out-of-the-box on many Google Cloud services like App Engine.



Cloud Debugger is a feature of Google Cloud that lets you inspect the state of a running application in real time, without stopping or slowing it down.

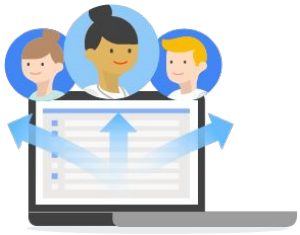
Debugger can be used with production applications. With a few mouse clicks, you can take a snapshot of your running application state or inject a new logging statement. A snapshot captures the call stack and variables at a specific code location the first time any instance executes that code. The injected log point behaves as if it were part of the deployed code writing the log messages to the same log stream.

Debugger is easier to use when source code is available. It knows how to display the correct version of the source code when a version control system is used, such as Cloud Source Repositories, GitHub, Bitbucket, or GitLab.

Users can easily collaborate with other team members by sharing their debug session. Sharing a debug session is as easy as sending the Console URL.

Debugger is integrated into existing developer workflows. Users can launch Debugger and take snapshots directly from Cloud Logging, Error Reporting, dashboards, integrated development environments, and the gcloud command-line interface.

Cloud Profiler



Low-impact production profiling

Broad platform support

Google Cloud

Poorly performing code increases the latency and cost of applications and web services every day. Cloud Profiler continuously analyzes the performance of CPU or memory-intensive functions executed across an application.

While it's possible to measure code performance in development environments, the results generally don't map well to what's happening in production. Many production profiling techniques either slow down code execution or can only inspect a small subset of a codebase. Profiler uses statistical techniques and extremely low-impact instrumentation that runs across all production application instances to provide a complete picture of an application's performance without slowing it down.

Profiler allows developers to analyze applications running anywhere, including Google Cloud, other cloud platforms, or on-premises, with support for Java, Go, Node.js, and Python.

Mapping Cloud Monitoring to Amazon CloudWatch (1/3)

	Cloud Monitoring	Amazon CloudWatch
Account structure	Workspace	AWS account
Types of metrics monitored	Google Cloud AWS Custom metrics	AWS On-premises metrics Custom metrics
Monitoring scope	Workspace can monitor 1–100 Google Cloud projects	Separate for each region

Google Cloud

AWS and Google Cloud both provide integrated monitoring services for their platforms. Amazon CloudWatch provides both logging and monitoring for AWS, while Cloud Logging and Cloud Monitoring provide logging and monitoring services, respectively, for Google Cloud. Let's see how these two technologies map against each other.

Amazon CloudWatch collects metrics for services used in an AWS account. In Cloud Monitoring, you can add Google Cloud projects to a Workspace, which then collects metrics for those Google Cloud projects. Metrics from all of the added Google Cloud projects are available in Cloud Monitoring under that Workspace.

Amazon CloudWatch automatically monitors AWS cloud services, and Cloud Monitoring automatically monitors Google Cloud services. Both services offer the ability to collect additional metrics through agents and to write custom metrics. You can configure both the Amazon CloudWatch agent and the Cloud Monitoring agent to ingest metrics from third-party sources.

Amazon CloudWatch designates metrics by AWS account and region. Metrics are separated by AWS account and then by region. Metrics are aggregated within the combination of AWS account and AWS region. Cloud Monitoring includes metrics from all Google Cloud projects that are added to a Workspace. Metrics are separated by the combination of the Workspace and Google Cloud project. You can aggregate metrics within the Cloud Monitoring account.

Mapping Cloud Monitoring to Amazon CloudWatch (2/3)

	Cloud Monitoring	Amazon CloudWatch
Metrics collection	Automatic for cloud services Monitoring agent Monitoring agent for Prometheus Custom metrics Logs-based metrics	Automatic for cloud services CloudWatch agents Custom metrics CloudWatch Logs metrics
Metric data types	Gauge Delta Cumulative	Double value Statistic sets
Retention	6 weeks	Specific to metric data type

Google Cloud

Both Cloud Monitoring and Amazon CloudWatch automatically collect metrics for their cloud services. The CloudWatch Agent provides the ability to collect more system-level metrics from EC2 instances and from on-premises servers. The Cloud Monitoring agent allows you to collect system and application metrics from virtual machine instances. Additionally, the Cloud Monitoring Prometheus agent collects metrics that are exposed through Prometheus, a logging service. Both technologies allow you to publish custom metrics.

The Amazon CloudWatch and Cloud Monitoring metric data types are similar in structure. The biggest difference in metric structures is that Cloud Monitoring metrics provide several metric and value types, and Amazon CloudWatch accepts statistic sets. The Amazon CloudWatch API accepts metrics that have a value type of Double. The API also accepts statistic sets that are aggregated metrics. Cloud Monitoring metrics accepts three types of metrics: Gauge, Delta, and Cumulative.

Cloud Monitoring will retain metric information for 6 weeks, while Amazon Cloudwatch will retain metric information based on the metric data type.

Mapping Cloud Monitoring to Amazon CloudWatch (3/3)

	Cloud Monitoring	Amazon CloudWatch
Alerts	Alerting policies	Alarms
Service availability checks	Uptime checks	Route 53 health check
Service access	Cloud Console Cloud SDK Cloud Monitoring API Google Cloud client libraries	Console AWS CLI CloudWatch API AWS SDKs

Google Cloud

Both Amazon CloudWatch and Cloud Monitoring provide the ability to take actions based on metric values. In Amazon CloudWatch, these thresholds are called alarms, and in Cloud Monitoring they are called alerting policies.

AWS and Cloud Monitoring both provide tools to verify the availability of a service by accessing it from locations around the world. AWS uses a Route 53 health check, while Cloud Monitoring uses uptime checks. Both checks surface metrics that can be used in alarms, alerts, and dashboards.

Both technologies support service access via the Cloud Console, their SDKs, APIs, and client libraries.

Agenda

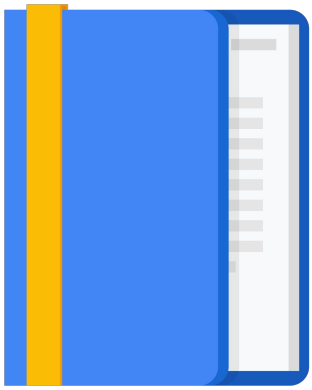
Development in the Cloud

Deployment: Infrastructure as Code

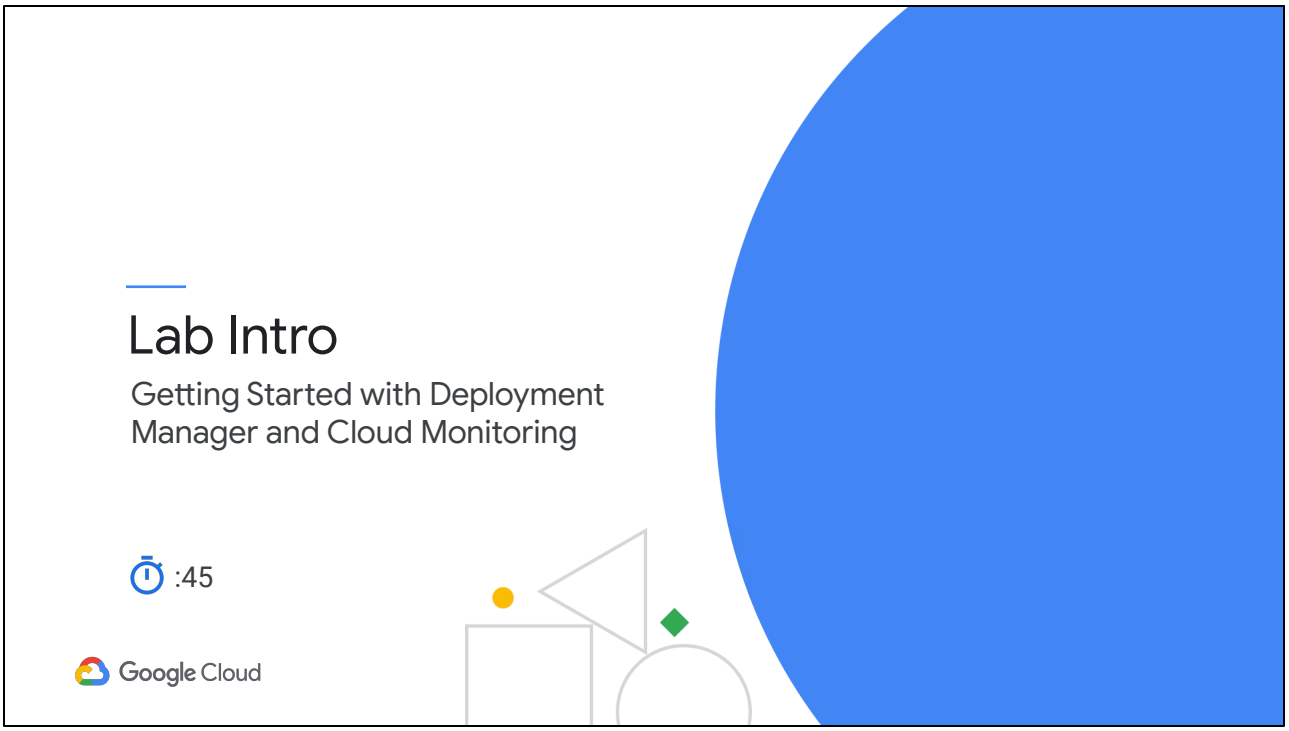
Monitoring: Proactive instrumentation

Lab

Resources



Google Cloud



The objectives of this lab are for you to:

- Create a Deployment Manager deployment,
- Update a Deployment Manager deployment, and
- View the load on a VM instance using Cloud Monitoring.

Agenda

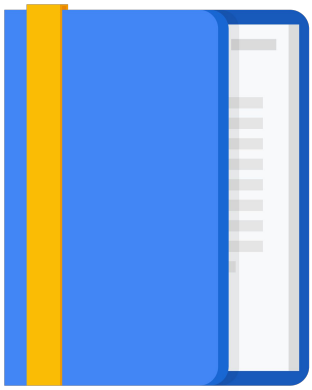
Development in the Cloud

Deployment: Infrastructure as Code

Monitoring: Proactive instrumentation

Lab

Resources



Google Cloud

Resources

Cloud Source Repositories https://cloud.google.com/source-repositories/docs/

Deployment Manager https://cloud.google.com/deployment-manager/docs/

Google Cloud operations suite https://cloud.google.com/stackdriver/docs/


Google Cloud