



In this module, I'll give you an overview of the resource monitoring options in GCP.

The features covered in this module rely on Stackdriver, a service that provides monitoring, logging, and diagnostics for your applications.

Agenda

Stackdriver Overview

Monitoring

Lab

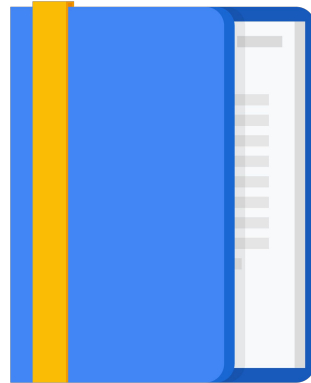
Logging

Error Reporting

Tracing

Debugging

Lab



In this module we are going to explore the Stackdriver Monitoring, Logging, Error Reporting, Tracing, and Debugging services. You will have the opportunity to apply these services in the two labs of this module.

Let me start by giving you a high-level overview of Stackdriver and its features.

Stackdriver overview



Stackdriver

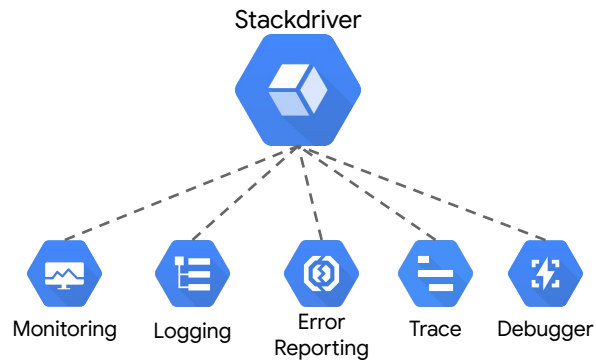
- Integrated monitoring, logging, diagnostics
- Manages across platforms
 - GCP and AWS
 - Dynamic discovery of GCP with smart defaults
 - Open-source agents and integrations
- Access to powerful data and analytics tools
- Collaboration with third-party software



Stackdriver dynamically discovers cloud resources and application services based on deep integration with Google Cloud Platform and Amazon Web Services. Because of its smart defaults, you can have core visibility into your cloud platform in minutes.

This provides you with access to powerful data and analytics tools plus collaboration with many different third-party software providers.

Multiple integrated products



As I mentioned earlier, Stackdriver has services for monitoring, logging, error reporting, fault tracing, and debugging. You only pay for what you use, and there are free usage allotments so that you can get started with no upfront fees or commitments. For more information about Stackdriver pricing, see the links section of this video: [\[https://cloud.google.com/stackdriver/pricing\]](https://cloud.google.com/stackdriver/pricing)

Now, in most other environments, these services are handled by completely different packages, or by a loosely integrated collection of software. When you see these functions working together in a single, comprehensive, and integrated service, you'll realize how important that is to creating reliable, stable, and maintainable applications.

Partner integrations

 bluemedora

 bmc

 matters

 sumologic

 tenable
network security

 Google Cloud

 OpsGenie

 splunk > enterprise

 netskope

 insightfinder

 pagerduty

Stackdriver also supports a rich and growing ecosystem of technology partners, as shown on this slide. This helps expand the IT ops, security, and compliance capabilities available to GCP customers. For more information about Stackdriver integrations, see the links section of the video

[\[https://cloud.google.com/stackdriver/partners\]](https://cloud.google.com/stackdriver/partners)

Agenda

Stackdriver Overview

Monitoring

Lab

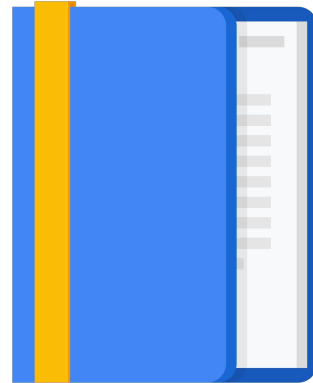
Logging

Error Reporting

Tracing

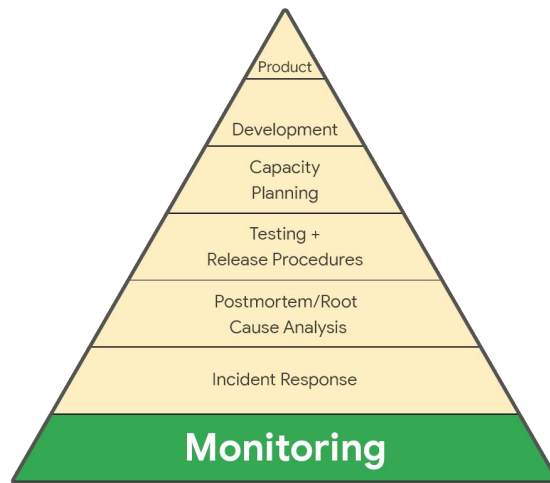
Debugging

Lab



Now that you understand Stackdriver from a high-level perspective, let's look at Stackdriver Monitoring.

Site reliability engineering



Monitoring is important to Google because it is at the base of site reliability engineering, or SRE.

SRE is a discipline that applies aspects of software engineering to operations whose goals are to create ultra-scalable and highly reliable software systems. This discipline has enabled Google to build, deploy, monitor, and maintain some of the largest software systems in the world.

If you want to learn more about SRE, I recommend exploring the free book written by members of Google's SRE team. It is in the links section of this video:

[\[https://landing.google.com/sre/book.html\]](https://landing.google.com/sre/book.html)

Monitoring



Monitoring

- Dynamic config and intelligent defaults
- Platform, system, and application metrics
 - Ingests data: Metrics, events, metadata
 - Generates insights through dashboards, charts, alerts
- Uptime/health checks
- Dashboards
- Alerts

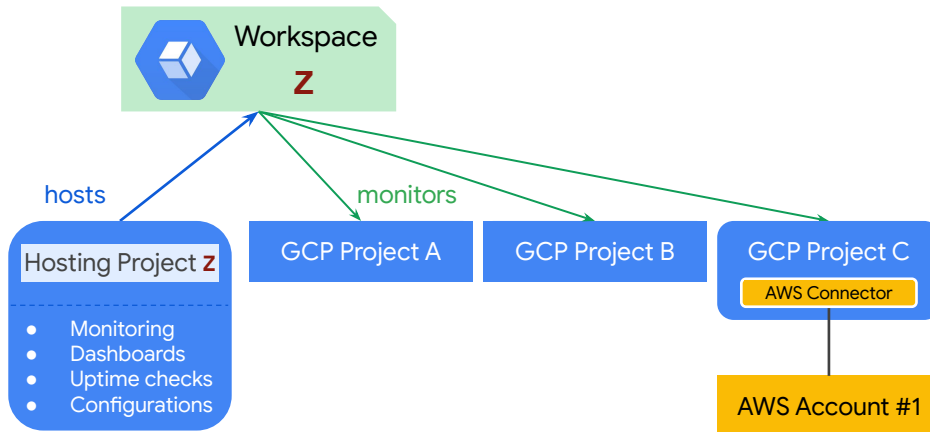


Stackdriver dynamically configures monitoring after resources are deployed and has intelligent defaults that allow you to easily create charts for basic monitoring activities.

This allows you to monitor your platform, system, and application metrics by ingesting data, such as metrics, events, and metadata. You can then generate insights from this data through dashboards, charts, and alerts.

For example, you can configure and measure uptime and health checks that send alerts via email.

Workspace is the root entity that holds monitoring and configuration information



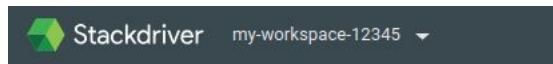
A Workspace is the root entity that holds monitoring and configuration information in Stackdriver Monitoring. Each Workspace can have between 1 and 100 monitored projects, including one or more GCP projects and any number of AWS accounts. You can have as many Workspaces as you want, but GCP projects and AWS accounts can't be monitored by more than one Workspace.

A Workspace contains the custom dashboards, alerting policies, uptime checks, notification channels, and group definitions that you use with your monitored projects. A Workspace can access metric data from its monitored projects, but the metric data and log entries remain in the individual projects.

The first monitored GCP project in a Workspace is called the hosting project, and it must be specified when you create the Workspace. The name of that project becomes the name of your Workspace. To access an AWS account, you must configure a project in GCP to hold the AWS Connector.

A Workspace is a “single pane of glass”

- Determine your monitoring needs up front
- Consider using separate Workspaces for data and control isolation

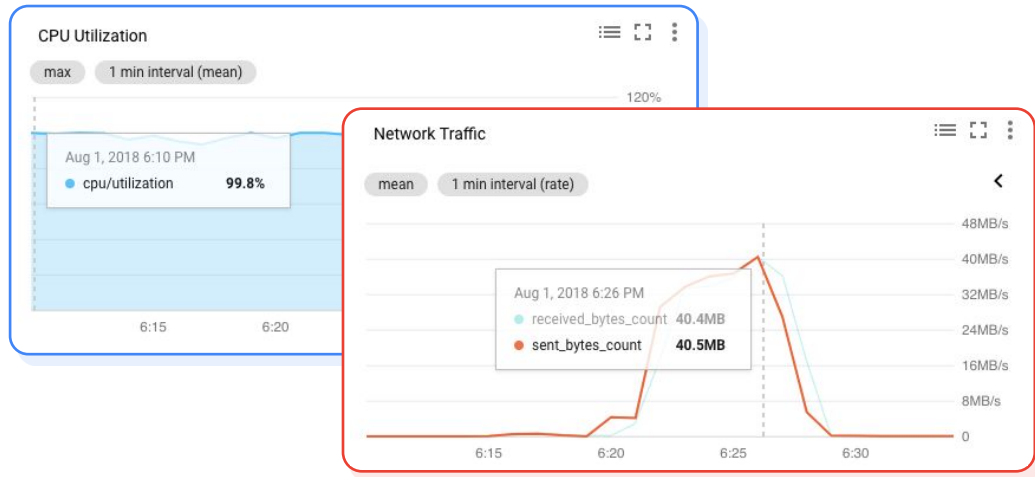


Because Workspaces can monitor all your GCP projects in a single place, a Workspace is a “single pane of glass” through which you can view resources from multiple GCP projects and AWS accounts. All Stackdriver users who have access to that Workspace have access to all data by default.

This means that a Stackdriver role assigned to one person on one project applies equally to all projects monitored by that Workspace.

In order to give people different roles per-project and to control visibility to data, consider placing the monitoring of those projects in separate Workspaces.

Dashboards visualize utilization and network traffic

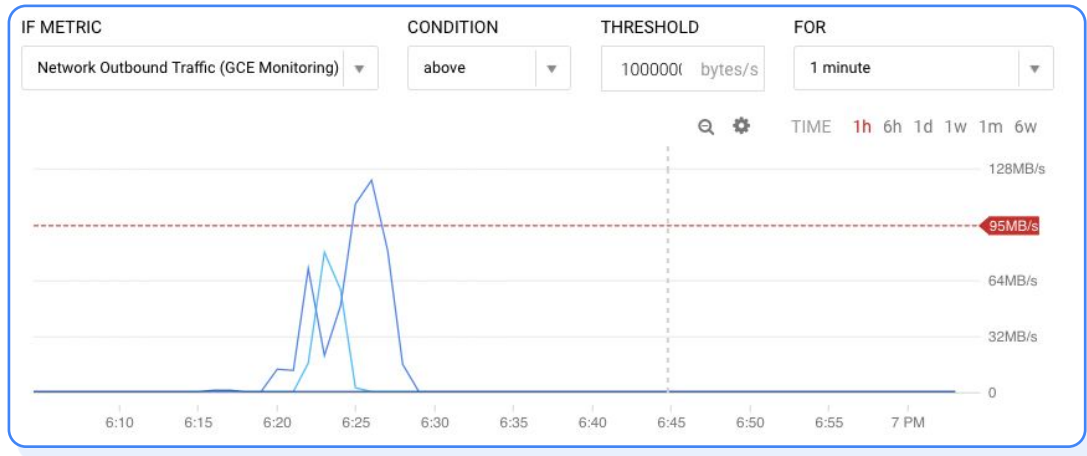


Stackdriver Monitoring allows you to create custom dashboards that contain charts of the metrics that you want to monitor. For example, you can create charts that display your instances' CPU utilization, the packets or bytes sent and received by those instances, and the packets or bytes dropped by the firewall of those instances.

In other words, charts provide visibility into the utilization and network traffic of your VM instances, as shown on this slide. These charts can be customized with filters to remove noise, groups to reduce the number of time series, and aggregates to group multiple time series together.

For a full list of supported metrics, see the documentation linked for this video: [\[https://cloud.google.com/monitoring/api/metrics_gcp\]](https://cloud.google.com/monitoring/api/metrics_gcp)

Alerting policies can notify you of certain conditions



Google Cloud

Now, although charts are extremely useful, they can only provide insight while someone is looking at them. But what if your server goes down in the middle of the night or over the weekend? Do you expect someone to always look at dashboards to determine whether your servers are available or have enough capacity or bandwidth?

If not, you want to create alerting policies that notify you when specific conditions are met.

For example, as shown on this slide, you can create an alerting policy when the network egress of your VM instance goes above a certain threshold for a specific timeframe. When this condition is met, you or someone else can be automatically notified through email, SMS, or other channels in order to troubleshoot this issue.

You can also create an alerting policy that monitors your Stackdriver usage and alerts you when you approach the threshold for billing. For more information about this, see the links section of this video:

[\[https://cloud.google.com/stackdriver/pricing#alert-usage\]](https://cloud.google.com/stackdriver/pricing#alert-usage)

Creating an alert

Create new alerting policy

1 Conditions

Basic Conditions

HTTP check on instance summer01
Violates when: Uptime Check Health on Instance (GCE) summer01 fails

[Edit](#) [Delete](#)

+ Add Another Condition

2 Notifications (optional)

When alerting policy violations occur, you will be notified via these channels. [Learn more](#)

Email demo@example.com

+ Add Another Notification

3 Documentation (optional)

When email notifications are sent, they'll include any text entered here. This can convey useful information about the problem and ways to approach fixing it.

[Edit](#) [Preview](#) [Markdown Formatting Help](#)

 Main Server health check failed
+ Server named summer01 failed a [Stackdriver uptime check](#)
+ IP Address of the server is: 104.197.58.79

4 Name this policy

A policy's name is used in identifying which policies were triggered, as well as managing configurations of different policies.

Uptime Check Policy

[Save Policy](#) [Cancel](#)



Here is an example of what creating an alerting policy looks like. On the left, you can see an HTTP check condition on the summer01 instance. This will send an email that is customized with the content of the documentation section on the right.

Let's discuss some best practices when creating alerts:

- I recommend alerting on symptoms, and not necessarily causes. For example, you want to monitor failing queries of a database and then identify whether the database is down.
- Next, make sure that you are using multiple notification channels, like email and SMS. This helps avoid a single point of failure in your alerting strategy.
- I also recommend customizing your alerts to the audience's need by describing what actions need to be taken or what resources need to be examined.
- Finally, avoid noise, because this will cause alerts to be dismissed over time. Specifically, adjust monitoring alerts so that they are actionable and don't just set up alerts on everything possible.

Uptime checks test the availability of your public services

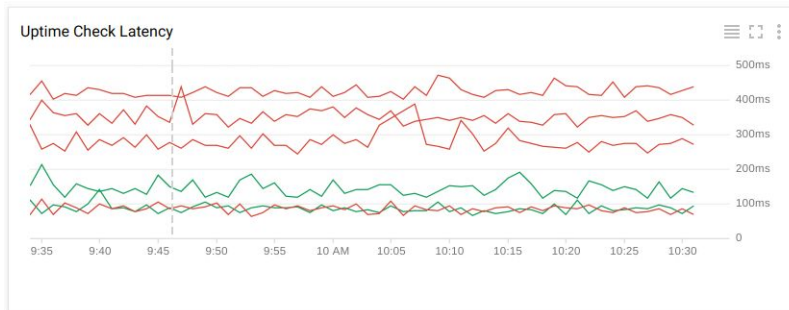
CHECKS	VIRGINIA	OREGON	IOWA	BELGIUM	SINGAPORE	SAO PAULO	POLICIES
Instance 1	✓	✓	✓	✓	✓	✓	
Instance 2	✓	✓	✓	✓	✓	✓	
Instance 3	✓	✓	✓	✓	✓	✓	




Uptime checks can be configured to test the availability of your public services from locations around the world, as you can see on this slide. The type of uptime check can be set to HTTP, HTTPS, or TCP. The resource to be checked can be an App Engine application, a Compute Engine instance, a URL of a host, or an AWS instance or load balancer.


For each uptime check, you can create an alerting policy and view the latency of each global location.

Uptime check example



Uptime 


100.000%

Outages 


0 minutes

Location Results

All locations passed



Check config



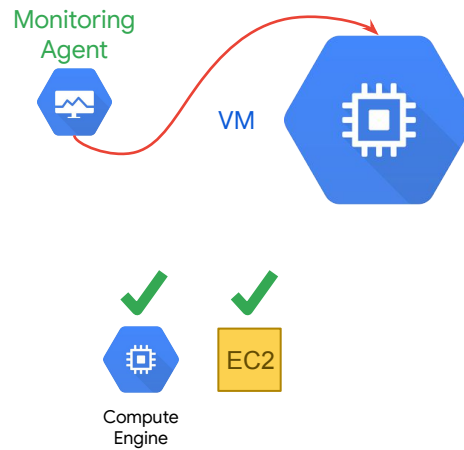
Check Type	HTTP
Resource	summer01
Path	/
Check Every	1 minute
Port	80
Locations	Global
Timeout	10 seconds



Here is an example of an HTTP uptime check. The resource is checked every minute with a 10-second timeout. Uptime checks that do not get a response within this timeout period are considered failures.

So far there is a 100% uptime with no outages.

Monitoring agent



Stackdriver Monitoring can access some metrics without the Monitoring agent, including CPU utilization, some disk traffic metrics, network traffic, and uptime information.

However, to access additional system resources and application services, you should install the Monitoring agent.

The Monitoring agent is supported for Compute Engine and EC2 instances.

Installing Monitoring agent

Install Monitoring agent

```
curl -O https://repo.stackdriver.com/stack-install.sh  
sudo bash stack-install.sh --write-gcm
```



The Monitoring agent can be installed with these two simple commands, which you could include in your startup script.

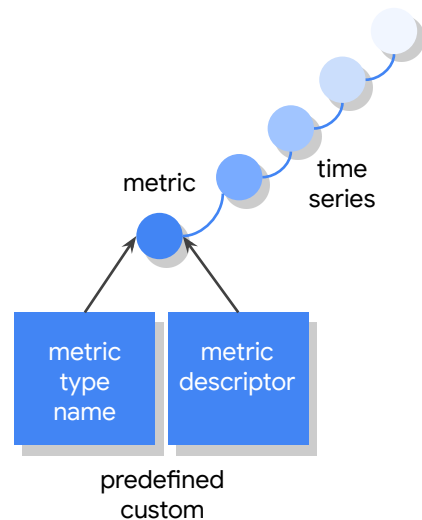
This assumes that you have a VM instance running Linux that is being monitored by a Workspace, and that your instance has the proper credentials for the agent.

Custom metrics

Custom metric example in Python:

```
client = monitoring.Client()
descriptor = client.metric_descriptor(
    'custom.googleapis.com/my_metric',

    metric_kind=monitoring.MetricKind.GAUGE,
    value_type=monitoring.ValueType.DOUBLE,
    description='This is a simple example
of a custom metric.')
descriptor.create()
```



If the standard metrics provided by Stackdriver monitoring do not fit your needs, you can create custom metrics.

For example, imagine a game server that has a capacity of 50 users. What metric indicator might you use to trigger scaling events? From an infrastructure perspective, you might consider using CPU load or perhaps network traffic load as values that are somewhat correlated with the number of users. But with a Custom Metric, you could actually pass the current number of users directly from your application into Stackdriver.

To get started with creating custom metrics, see the links section of this video:

[\[https://cloud.google.com/monitoring/custom-metrics/creating-metrics#monitoring-create-metric-python\]](https://cloud.google.com/monitoring/custom-metrics/creating-metrics#monitoring-create-metric-python)

Lab

Resource Monitoring



Let's take some of the monitoring concepts that we just discussed and apply them in a lab.

In this lab, you learn how to use Stackdriver Monitoring to gain insight into applications that run on GCP. Specifically, you will enable Stackdriver monitoring, add charts to dashboards and create alerts, resource groups, and uptime checks.

Lab review

Resource Monitoring



In this lab, you got an overview of Stackdriver Monitoring. You learned how to monitor your project, create alerts with multiple conditions, add charts to dashboards, create resource groups, and create uptime checks for your services.

Monitoring is critical to your application's health, and Stackdriver provides a rich set of features for monitoring your infrastructure, visualizing the monitoring data, and triggering alerts and events for you.

You can stay for a lab walkthrough, but remember that GCP's user interface can change, so your environment might look slightly different.

Agenda

Stackdriver Overview

Monitoring

Lab

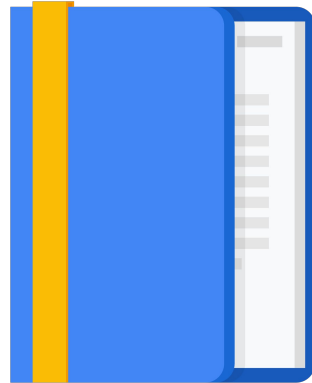
Logging

Error Reporting

Tracing

Debugging

Lab



Monitoring is the basis of Stackdriver, but the service also provides logging, error reporting, tracing, and debugging. Let's learn about logging.

Logging



Logging

- Platform, systems, and application logs
 - API to write to logs
 - 30-day retention
- Log search/view/filter
- Log-based metrics
- Monitoring alerts can be set on log events
- Data can be exported to Cloud Storage, BigQuery, and Cloud Pub/Sub



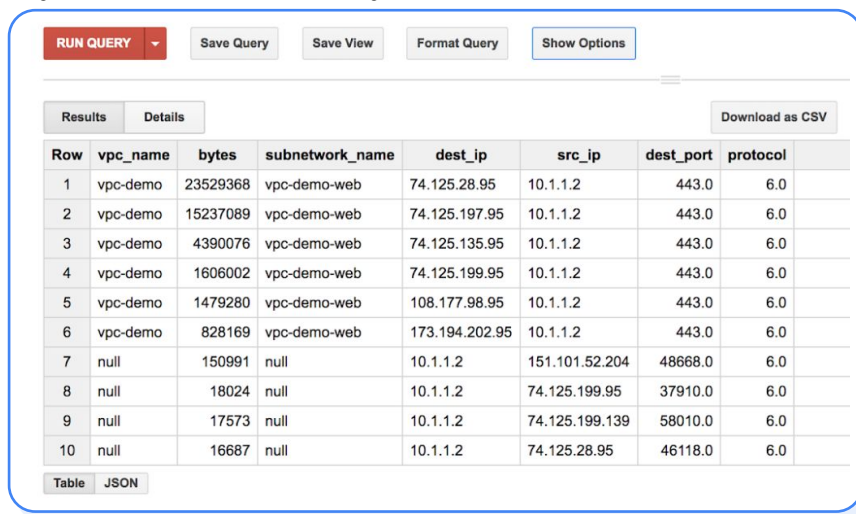
Stackdriver Logging allows you to store, search, analyze, monitor, and alert on log data and events from GCP and AWS. It is a fully managed service that performs at scale and can ingest application and system log data from thousands of VMs.

Logging includes storage for logs, a user interface called the Logs Viewer, and an API to manage logs programmatically. The service lets you read and write log entries, search and filter your logs, and create log-based metrics.

Logs are only retained for 30 days, but you can export your logs to Cloud Storage buckets, BigQuery datasets, and Cloud Pub/Sub topics.

Exporting logs to Cloud Storage makes sense for storing logs for more than 30 days, but why should you export to BigQuery or Cloud Pub/Sub?

Analyze logs in BigQuery and visualize in Data Studio



The screenshot shows the BigQuery interface with a query result table. The table has columns: Row, vpc_name, bytes, subnetwork_name, dest_ip, src_ip, dest_port, and protocol. The first 6 rows show data for 'vpc-demo' and the last 4 rows show 'null' values for 'vpc_name' and 'subnetwork_name'.

Row	vpc_name	bytes	subnetwork_name	dest_ip	src_ip	dest_port	protocol
1	vpc-demo	23529368	vpc-demo-web	74.125.28.95	10.1.1.2	443.0	6.0
2	vpc-demo	15237089	vpc-demo-web	74.125.197.95	10.1.1.2	443.0	6.0
3	vpc-demo	4390076	vpc-demo-web	74.125.135.95	10.1.1.2	443.0	6.0
4	vpc-demo	1606002	vpc-demo-web	74.125.199.95	10.1.1.2	443.0	6.0
5	vpc-demo	1479280	vpc-demo-web	108.177.98.95	10.1.1.2	443.0	6.0
6	vpc-demo	828169	vpc-demo-web	173.194.202.95	10.1.1.2	443.0	6.0
7	null	150991	null	10.1.1.2	151.101.52.204	48668.0	6.0
8	null	18024	null	10.1.1.2	74.125.199.95	37910.0	6.0
9	null	17573	null	10.1.1.2	74.125.199.139	58010.0	6.0
10	null	16687	null	10.1.1.2	74.125.28.95	46118.0	6.0



Exporting logs to BigQuery allows you to analyze logs and even visualize them in Data Studio.

BigQuery runs extremely fast SQL queries on gigabytes to petabytes of data. This allows you to analyze logs, such as your network traffic, so that you can better understand traffic growth to forecast capacity, network usage to optimize network traffic expenses, or network forensics to analyze incidents.

For example, in this screenshot I queried my logs to identify the top IP addresses that have exchanged traffic with my web server. Depending on where these IP addresses are and who they belong to, I could relocate part of my infrastructure to save on networking costs or deny some of these IP addresses if I don't want them to access my web server.

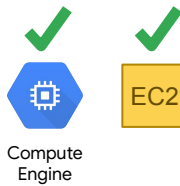
If you want to visualize your logs, I recommend connecting your BigQuery tables to Data Studio. Data Studio transforms your raw data into the metrics and dimensions that you can use to create easy-to-understand reports and dashboards.

I mentioned that you can also export logs to Cloud Pub/Sub. This enables you to stream logs to applications or endpoints.

Installing Logging agent

Install Logging agent

```
curl -sSO https://dl.google.com/cloudagents/install-logging-agent.sh  
sudo bash install-logging-agent.sh
```



Similar to Stackdriver's Monitoring agent, it's a best practice to install the Logging agent on all your VM instances. The Logging agent can be installed with these two simple commands, which you could include in your startup script.

This agent is supported for Compute Engine and EC2 instances.

Agenda

Stackdriver Overview

Monitoring

Lab

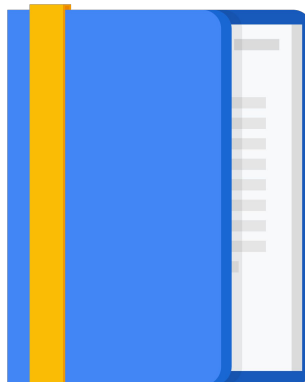
Logging

Error Reporting

Tracing

Debugging

Lab



Let's learn about another feature of Stackdriver: Error Reporting.

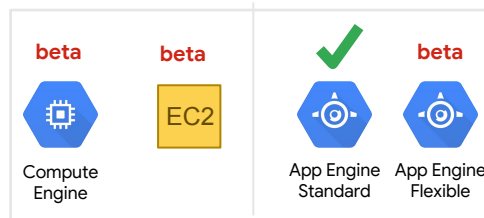
Error Reporting

Aggregate and display errors for running cloud services

- Error notifications
- Error dashboard
- Go, Java, .NET, Node.js, PHP, Python, and Ruby



Error Reporting



Stackdriver Error Reporting counts, analyzes, and aggregates the errors in your running cloud services. A centralized error management interface displays the results with sorting and filtering capabilities, and you can even set up real-time notifications when new errors are detected.

As of this recording, Stackdriver Error Reporting is Generally Available for the App Engine standard environment and is a Beta feature for App Engine flexible environment, Compute Engine, and AWS EC2.

In terms of programming languages, the exception stack trace parser is able to process Go, Java, .NET, Node.js, PHP, Python, and Ruby.

By the way, I'm mentioning App Engine because you will explore Error Reporting in an app deployed to App Engine in the upcoming lab.

Agenda

Monitoring

Lab

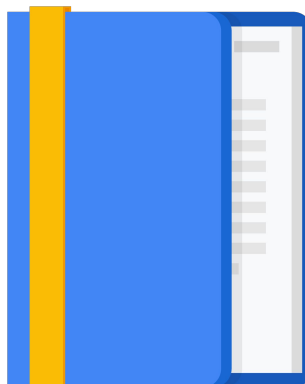
Logging

Error Reporting

Tracing

Debugging

Lab



Tracing is another Stackdriver feature integrated into GCP.

Tracing



Trace

Tracing system

- Displays data in near real-time
- Latency reporting
- Per-URL latency sampling

Collects latency data

- App Engine
- Google HTTP(S) load balancers
- Applications instrumented with the Stackdriver Trace SDKs



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

Stackdriver Trace automatically analyzes all of your application's traces to generate in-depth latency reports that surface performance degradations and can capture traces from App Engine, HTTP(S) load balancers, and applications instrumented with the Stackdriver Trace API.

Managing the amount of time it takes for your application to handle incoming requests and perform operations is an important part of managing overall application performance. Stackdriver Trace is actually based on the tools used at Google to keep our services running at extreme scale.

Agenda

Stackdriver Overview

Monitoring

Lab

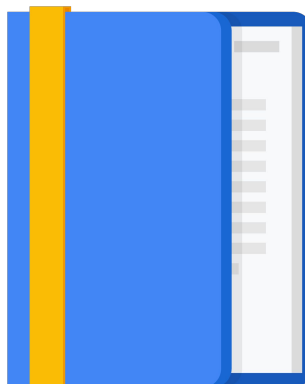
Logging

Error Reporting

Tracing

Debugging

Lab



Finally, let's cover the last Stackdriver feature of this module, which is the debugger.

Debugging



Debugger

- Inspect an application without stopping it or slowing it down significantly.
- Debug snapshots:
 - Capture call stack and local variables of a running application.
- Debug logpoints:
 - Inject logging into a service without stopping it.
- Java, Python, Go, Node.js and Ruby



Stackdriver Debugger is a feature of GCP that lets you inspect the state of a running application, in real time, without stopping or slowing it. Specifically, the debugger adds less than 10ms to the request latency when the application state is captured. In most cases, this is not noticeable by users.

These features allow you to understand the behavior of your code in production and analyze its state to locate those hard-to-find bugs. With just a few mouse clicks, you can take a snapshot of your running application's state or inject a new logging statement.

Stackdriver Debugger supports multiple languages, including Java, Python, Go, Node.js and Ruby.

Lab

Error Reporting and Debugging



Let's apply what we just learned about Stackdriver logging, error reporting, tracing, and debugging in a lab.

In this lab, you'll deploy a small "Hello, World" application to App Engine and instrument it with Stackdriver. Then you'll plant a bug in the application, which will expose you to Stackdriver's error reporting and debugging features.

Lab review

Error Reporting and Debugging



In this lab, you deployed an application to App Engine. Then you introduced a bug in the code, which broke the application. You used Stackdriver Error Reporting to identify and analyze the issue and found the root cause using Stackdriver Debugger. Finally, you modified the code to fix the problem and saw the results in Stackdriver.

Having all of these tools integrated into GCP allows you to focus on your code and any troubleshooting that goes with it.

You can stay for a lab walkthrough, but remember that GCP's user interface can change, so your environment might look slightly different.

Review

Resource Monitoring



In this module, I gave you an overview of Stackdriver and its monitoring, logging, error reporting, fault tracing, and debugging features. Having all of these integrated into GCP allows you to operate and maintain your applications, which is known as site reliability engineering or SRE.

If you're interested in learning more about SRE, you can explore the book or some of our SRE courses.

Review

Essential Cloud Infrastructure: Core Services



Thank you for taking the “Essential Cloud Infrastructure: Core Services” course. I hope you have a better understanding of how to administer IAM, choose between the different data storage services in GCP, examine billing of GCP resources and monitor those resources. Hopefully the demos and labs made you feel more comfortable using the different GCP services that we covered.

Elastic Cloud Infrastructure: Scaling and Automation

1. Interconnecting Networks
2. Load Balancing and Autoscaling
3. Infrastructure Automation
4. Managed Services



Next, I recommend enrolling in the “Elastic Cloud Infrastructure: Scaling and Automation” course of the “Architecting with Google Compute Engine” series.

1. In that course, we start by going over the different options to interconnect networks to enable you to connect your infrastructure to GCP.
2. Next, we’ll go over GCP’s load balancing and autoscaling services, which you will get to explore directly.
3. Then, we’ll cover infrastructure automation services like Deployment Manager and Terraform, so that you can automate the deployment of GCP infrastructure services.
4. Lastly, we’ll talk about other managed services that you might want to leverage in GCP.

Enjoy that course!