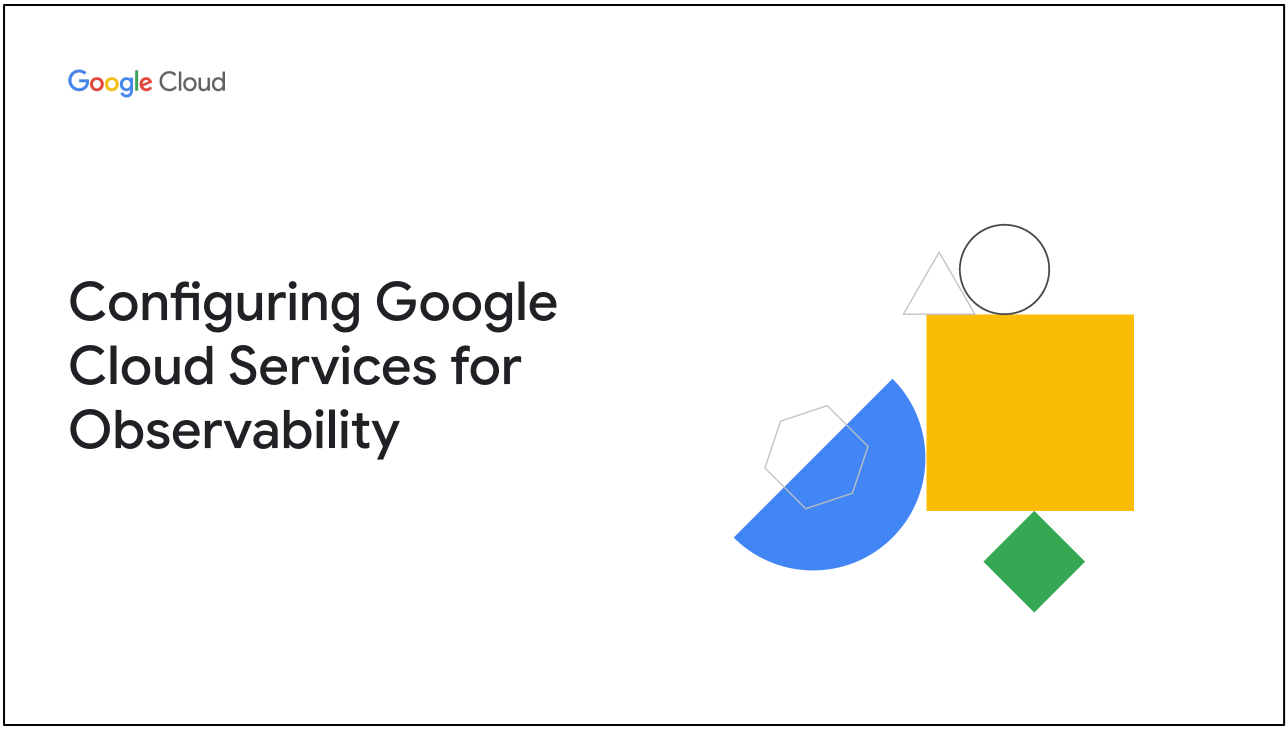
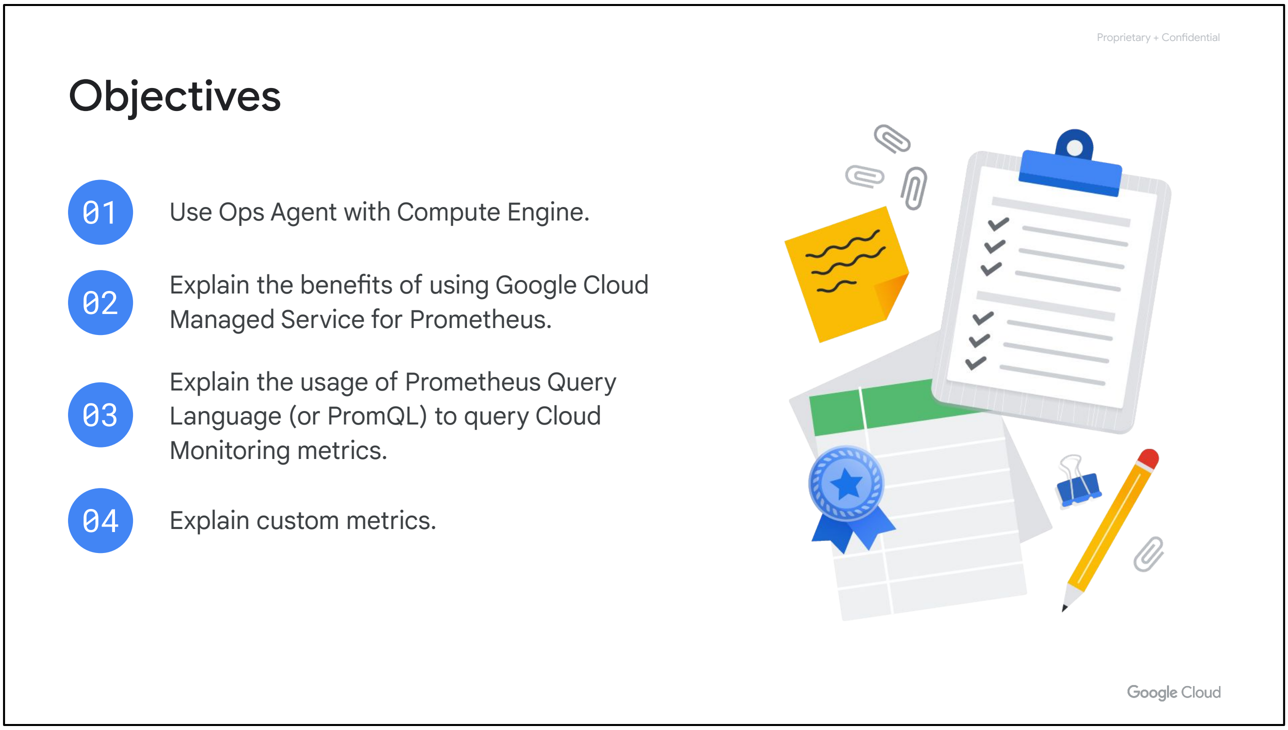


Google Cloud

Configuring Google Cloud Services for Observability

In the next part of our Google Cloud Observability discussion, let’s take some time to examine the art of configuring Google Cloud services for observability.



Objectives

01 Use Ops Agent with Compute Engine.

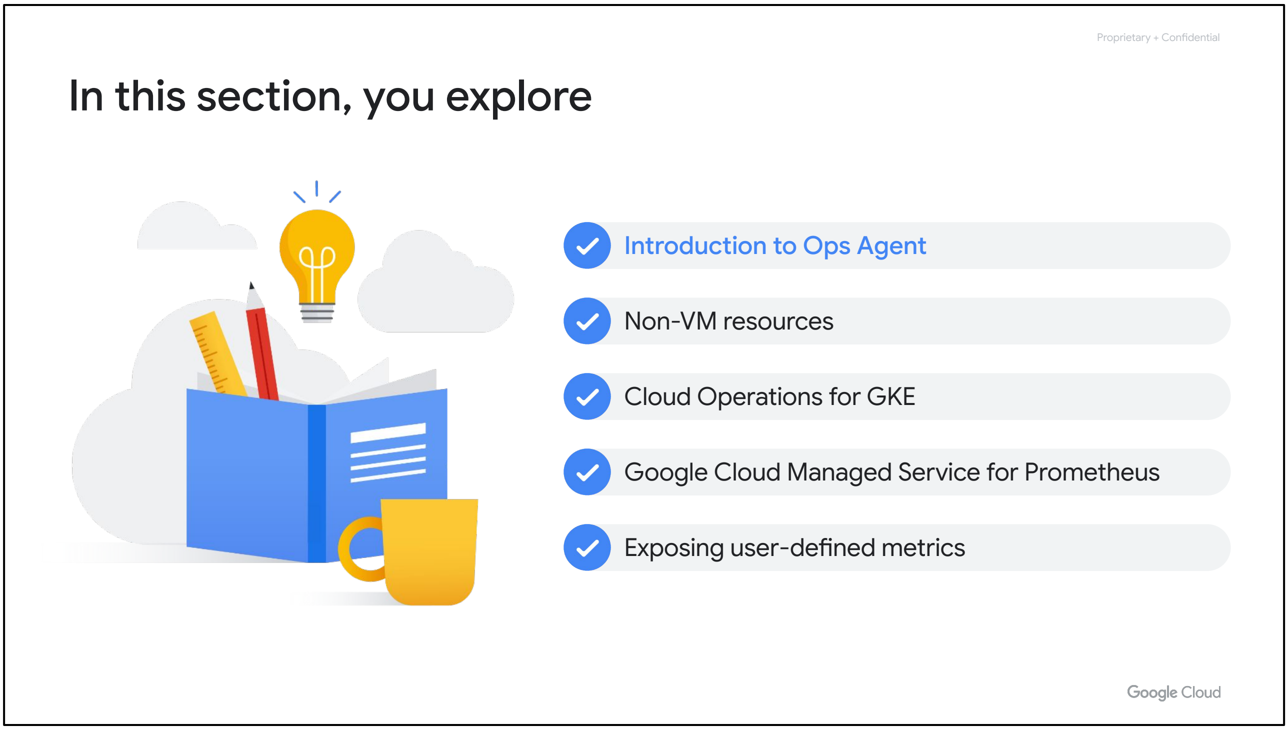
02 Explain the benefits of using Google Cloud Managed Service for Prometheus.

03 Explain the usage of Prometheus Query Language (or PromQL) to query Cloud Monitoring metrics.

04 Explain custom metrics.

In this module, we're going to spend a little time learning how to:

- Use Ops Agent with Compute Engine.
- Explain the benefits of using Google Cloud Managed Service for Prometheus.
- Explain the usage of Prometheus Query Language (or PromQL) to query Cloud Monitoring metrics.
- Explain custom metrics.



Proprietary + Confidential

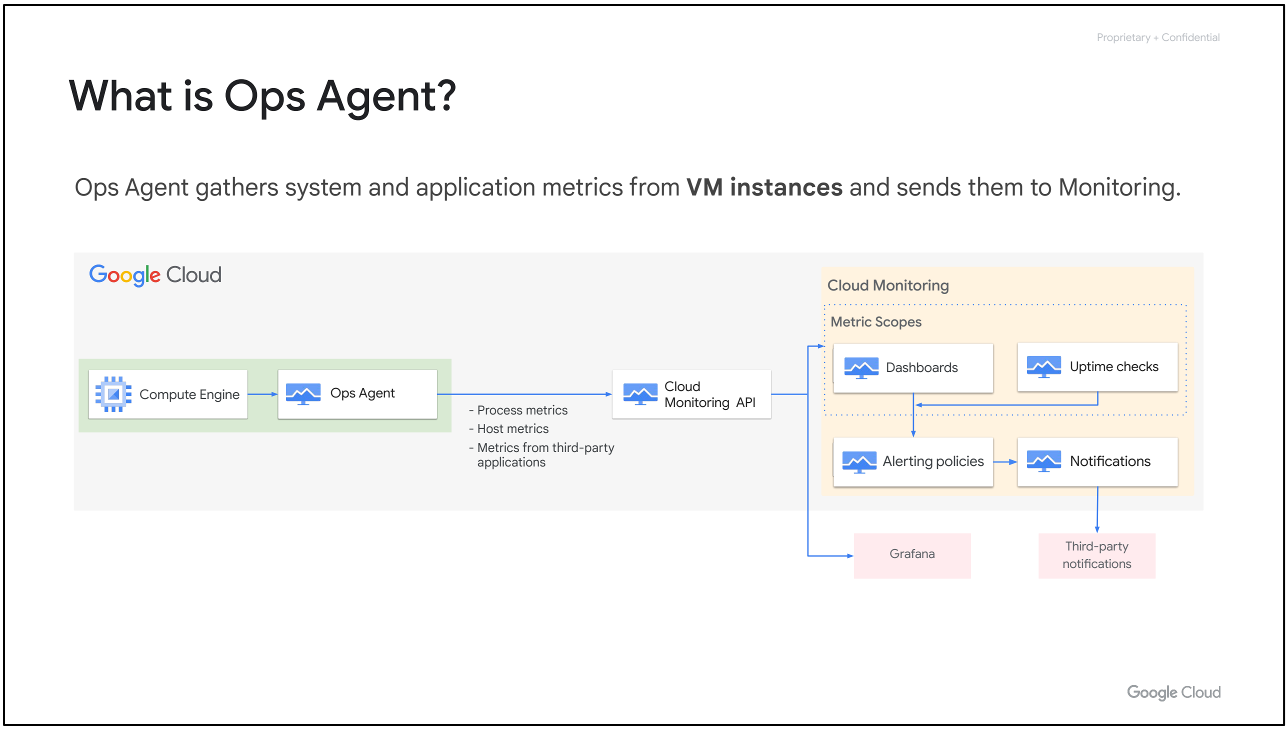
In this section, you explore

- ✓ Introduction to Ops Agent
- ✓ Non-VM resources
- ✓ Cloud Operations for GKE
- ✓ Google Cloud Managed Service for Prometheus
- ✓ Exposing user-defined metrics

Google Cloud

In this module, you learn how to:

- Use Ops Agent with Compute Engine.
- Enable and use Kubernetes Monitoring.
- Explain the benefits of using Google Cloud Managed Service for Prometheus.
- Explain the usage of PromQL to query Cloud Monitoring metrics.
- Explain the uses of Open Telemetry.
- Explain custom metrics.



What is Ops Agent?

Ops Agent gathers system and application metrics from **VM instances** and sends them to Monitoring.

Google Cloud

Let’s start with Ops Agent for Compute Engine.

As we discussed, monitoring data can originate at a number of different sources. With Google Compute Engine instances, because the VMs are running on Google hardware, Cloud Monitoring can access some instance metrics without the agent, including CPU utilization, some disk traffic metrics, network traffic, and uptime information, but that information can be augmented by installing agents into the VM operating system.

Many mission critical services use compute infrastructure directly and run on Google Compute Engine instances. How can we improve observability for those workloads?

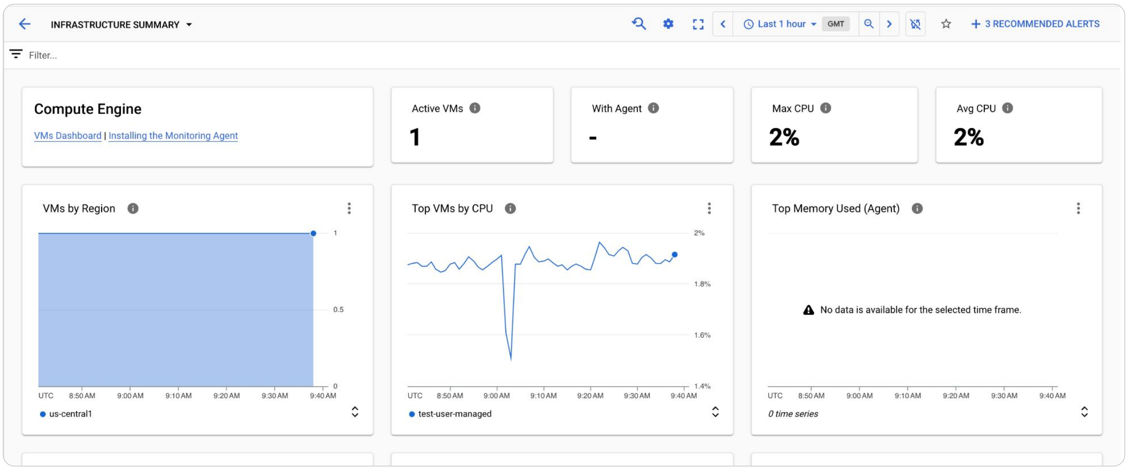
The Ops Agent is the primary agent for collecting telemetry data from your Compute Engine instances. Combining logging and metrics into a single agent, the Ops Agent uses Fluent Bit for logs, which supports high-throughput logging, and the Open Telemetry Collector for metrics.

These agents are required because for security reasons, the hypervisor cannot access some of the internal metrics inside a VM, for example, memory usage.

You can configure the Ops agent to monitor many third-party applications such as Apache, mySQL, Oracle database, SAP HANA, and NGINX. The Ops Agent collects metrics inside the VM, not at the hypervisor level. For a detailed list, refer to the documentation.

The Ops Agent supports most major operating systems such as CentOS, Ubuntu and Windows.

Why Ops Agent?



INFRASTRUCTURE SUMMARY

Filter...

Last 1 hour GMT

+ 3 RECOMMENDED ALERTS

Compute Engine

VMs Dashboard | Installing the Monitoring Agent

Active VMs: 1

With Agent: -

Max CPU: 2%

Avg CPU: 2%

No data is available for the selected time frame.

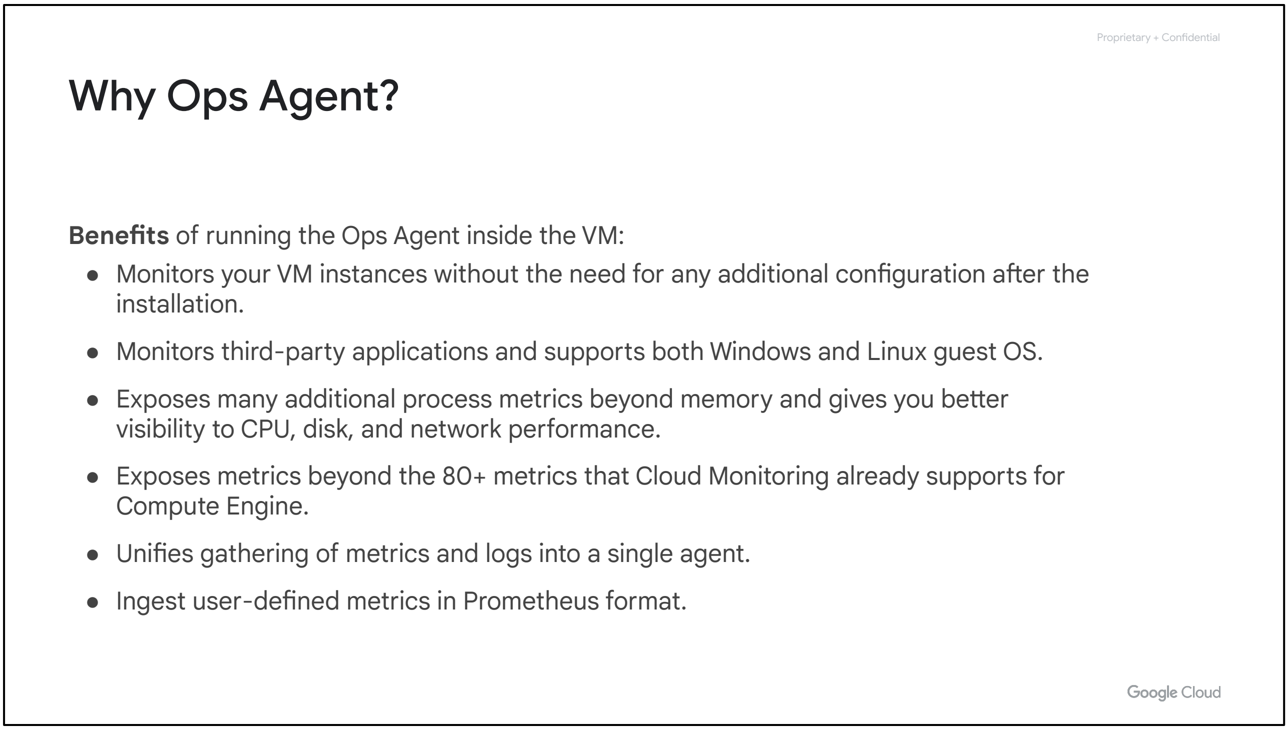
0 time series

VMs by Region
1
0.5
0
UTC 8:50 AM 9:00 AM 9:10 AM 9:20 AM 9:30 AM 9:40 AM
us-central1
Top VMs by CPU
2%
1.8%
1.6%
1.4%
UTC 8:50 AM 9:00 AM 9:10 AM 9:20 AM 9:30 AM 9:40 AM
test-user-managed
Top Memory Used (Agent)
UTC 8:50 AM 9:00 AM 9:10 AM 9:20 AM 9:30 AM 9:40 AM

We learned what Ops Agent is, let us next understand why we need Ops Agent. Here is an example of an infrastructure summary dashboard for a project with a few Compute Engine VMs in it.

Notice that we're not getting any data about how our instances are using memory. That's because the VM hypervisor only knows how much memory is allocated to each VM, not how much of the allocated memory the VM is actually using.

This is just one example of data that can only be gathered by a process running in the guest VM, such as an agent.



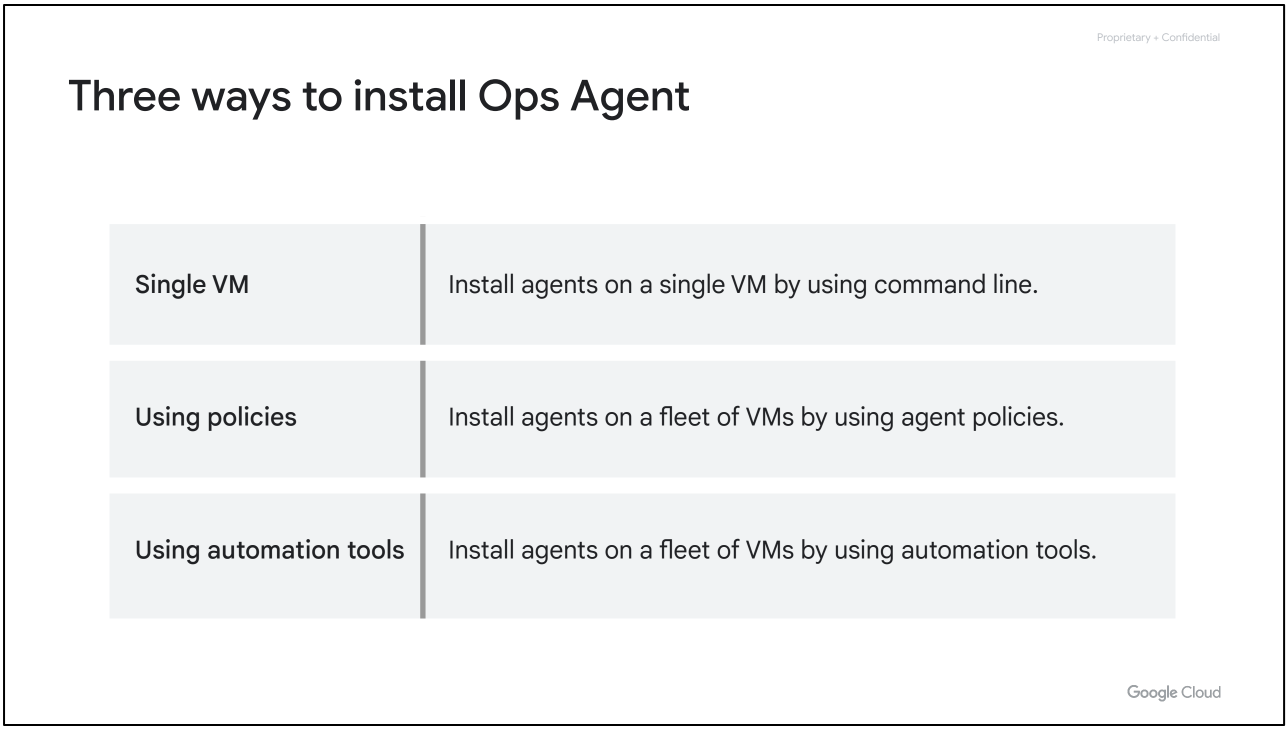
Why Ops Agent?

Benefits of running the Ops Agent inside the VM:

- Monitors your VM instances without the need for any additional configuration after the installation.
- Monitors third-party applications and supports both Windows and Linux guest OS.
- Exposes many additional process metrics beyond memory and gives you better visibility to CPU, disk, and network performance.
- Exposes metrics beyond the 80+ metrics that Cloud Monitoring already supports for Compute Engine.
- Unifies gathering of metrics and logs into a single agent.
- Ingest user-defined metrics in Prometheus format.

There are other benefits of running the Ops Agent inside the VM:

- It **monitors your VM instances** without the need for any additional configuration after the installation.
- Helps monitor 3rd party applications and also supports both windows and Linux guest OS.
- It **exposes many additional process metrics** beyond memory, and gives you better visibility to CPU, disk, and network performance.
- Exposes metrics beyond the 80+ metrics that Cloud Monitoring already supports for Compute Engine.
- The Ops Agent unifies gathering of metrics and logs into a single agent.
- And also ingests any user defined (Custom) metrics in Prometheus format.



Proprietary + Confidential

Three ways to install Ops Agent

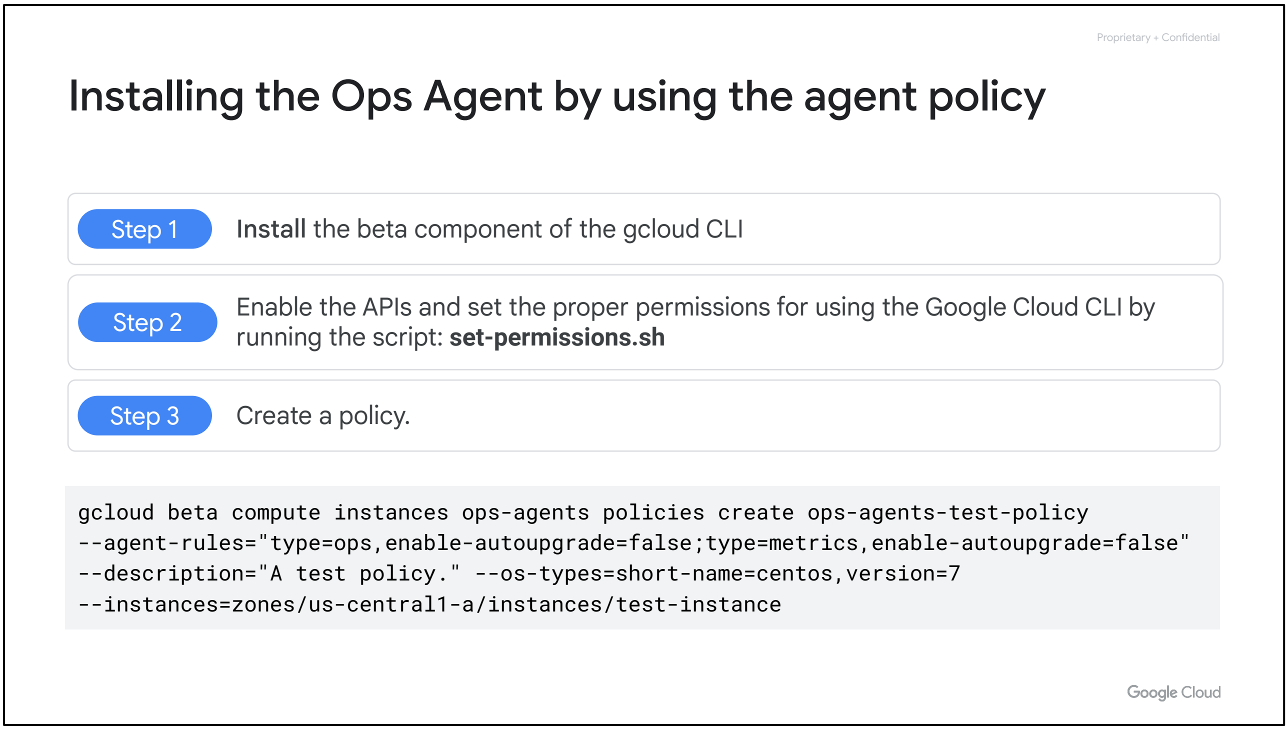
Single VM	Install agents on a single VM by using command line.
Using policies	Install agents on a fleet of VMs by using agent policies.
Using automation tools	Install agents on a fleet of VMs by using automation tools.

Google Cloud

You can install Ops Agent by using three different methods:

- Use the Google Cloud CLI or the Google Cloud console to install the agent on individual VMs.
- Use an Agent Policy that installs and manages agents on your fleet of VMs.
- Use automation tools, like Ansible, Chef, Puppet, and Terraform, to install and manage agents on your fleet of VMs.

We will cover the first two methods. For the automation process, refer to the documentation.



Proprietary + Confidential

Installing the Ops Agent by using the agent policy

Step 1 **Install** the beta component of the gcloud CLI

Step 2 Enable the APIs and set the proper permissions for using the Google Cloud CLI by running the script: **set-permissions.sh**

Step 3 Create a policy.

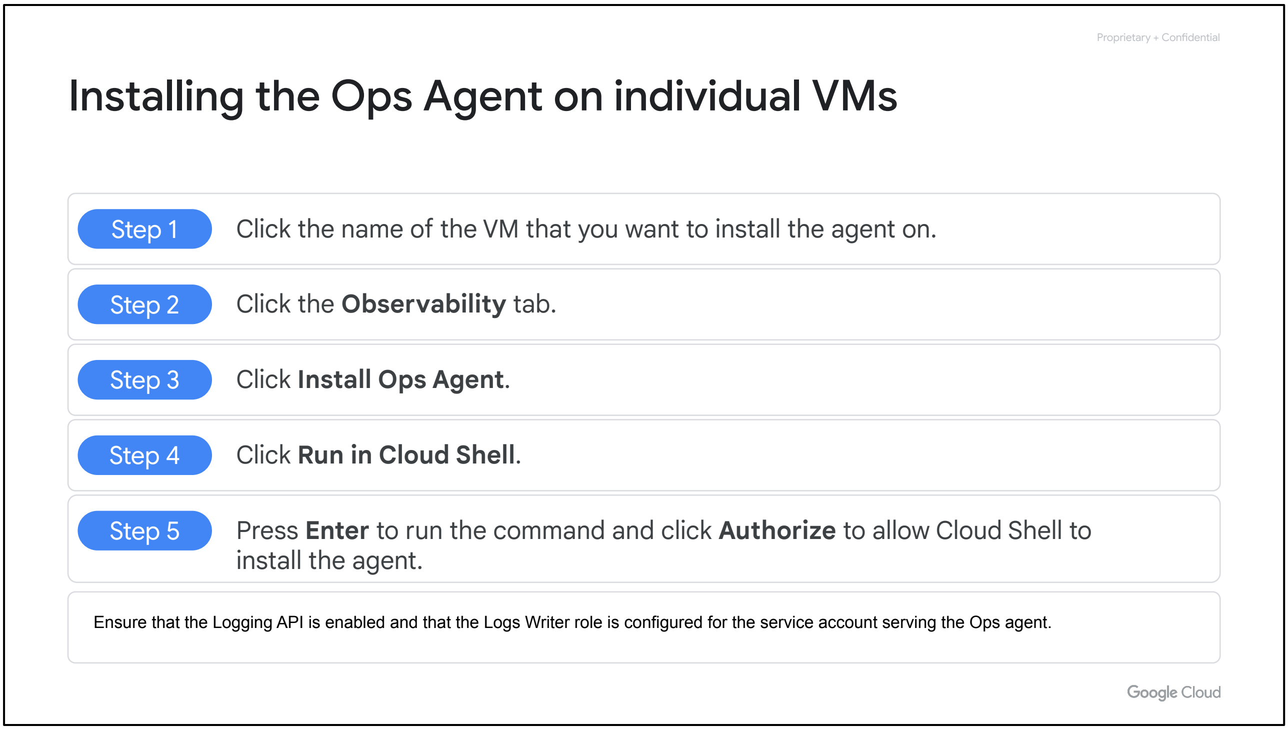
```
gcloud beta compute instances ops-agents policies create ops-agents-test-policy
--agent-rules="type=ops,enable-autoupgrade=false;type=metrics,enable-autoupgrade=false"
--description="A test policy." --os-types=short-name=centos,version=7
--instances=zones/us-central1-a/instances/test-instance
```

Google Cloud

Installing the Ops Agent is well documented on the Google site.:

1. Install the beta component
2. Enable the APIs
3. Create a policy

Here, you see examples to create a policy named ops-agents-test-policy. The policy targets a single CentOS 7 VM instance named zones/us-central1-a/instances/test-instance for testing or development. It also installs both Logging and Monitoring agents on that VM instance.
gcloud beta compute instances ops-agents policies create ops-agents-test-policy --agent-rules="type=logging,enable-autoupgrade=false;type=metrics,enable-autoupgrade=false" --description="A test policy." --os-types=short-name=centos,version=7 --instances=zones/us-central1-a/instances/test-instance



Proprietary + Confidential

Installing the Ops Agent on individual VMs

Step 1 Click the name of the VM that you want to install the agent on.

Step 2 Click the **Observability** tab.

Step 3 Click **Install Ops Agent**.

Step 4 Click **Run in Cloud Shell**.

Step 5 Press **Enter** to run the command and click **Authorize** to allow Cloud Shell to install the agent.

Ensure that the Logging API is enabled and that the Logs Writer role is configured for the service account serving the Ops agent.

Google Cloud

Go to the VM instances page in the Google Cloud console.

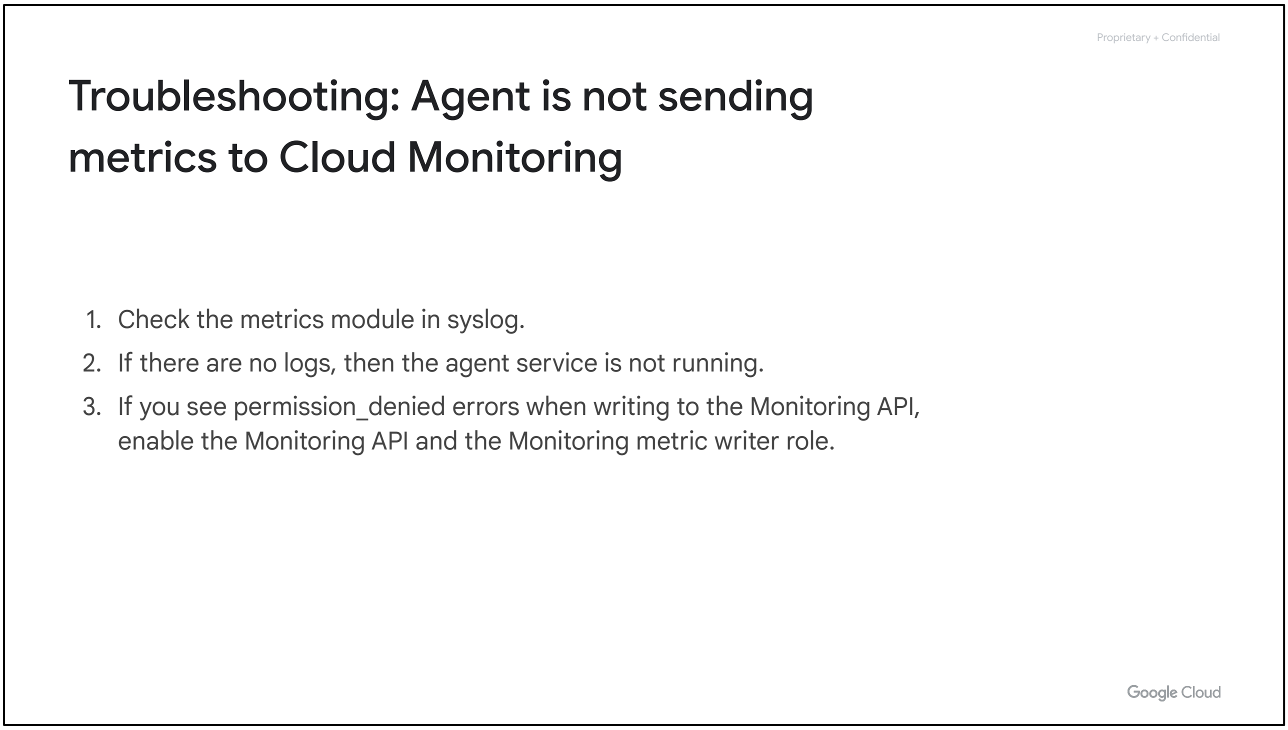
1. Click the name of the VM that you want to install the agent on. The Details page opens.
2. Click the Observability tab. The Observability page opens.
3. Click Install Ops Agent.
4. Click Run in Cloud Shell. Cloud Shell opens and pastes the installation command.
5. Press Enter on your keyboard to run the command.
6. Click Authorize to allow Cloud Shell to install the agent.

You can install a specific version of the agent and the steps also vary based on the operating system. For details refer to the documentation.

Instructor note: Please share your screen to show the process.

Ensure that the Logging API is enabled and that the Logs Writer role is configured for the service account serving the Ops agent.

For CLI steps, refer to the documentation.



Proprietary + Confidential

Troubleshooting: Agent is not sending metrics to Cloud Monitoring

1. Check the metrics module in syslog.
2. If there are no logs, then the agent service is not running.
3. If you see permission_denied errors when writing to the Monitoring API, enable the Monitoring API and the Monitoring metric writer role.

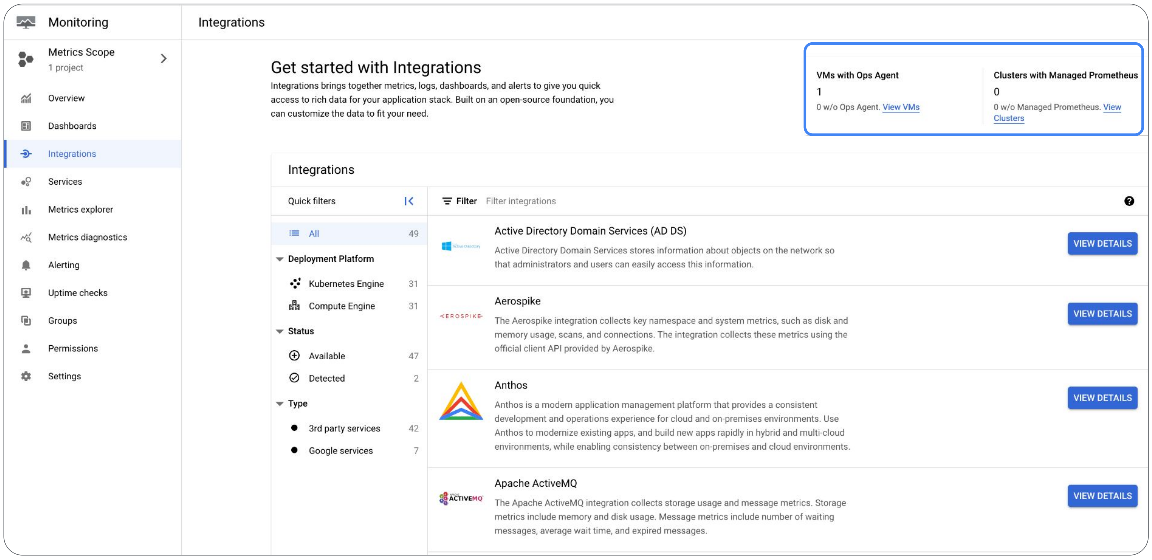
Google Cloud

Normally, you won't have to perform this step, but if the agent is not sending logs to Cloud Logging:

1. First, check the metrics module in syslog
2. If there are no logs, then agent service is not running.
3. If you see a 403 permission errors when writing to the Monitoring API, enable the Logging API and Logs Writer role

Make sure to check the Google documentation if you have questions.

Integrations page



Get started with Integrations

Integrations brings together metrics, logs, dashboards, and alerts to give you quick access to rich data for your application stack. Built on an open-source foundation, you can customize the data to fit your need.

VMs with Ops Agent
1
0 w/o Ops Agent. View VMs

Clusters with Managed Prometheus
0
0 w/o Managed Prometheus. View Clusters

Active Directory Domain Services (AD DS)

Active Directory Domain Services stores information about objects on the network so that administrators and users can easily access this information.

Aerospike

The Aerospike integration collects key namespace and system metrics, such as disk and memory usage, scans, and connections. The integration collects these metrics using the official client API provided by Aerospike.

Anthos

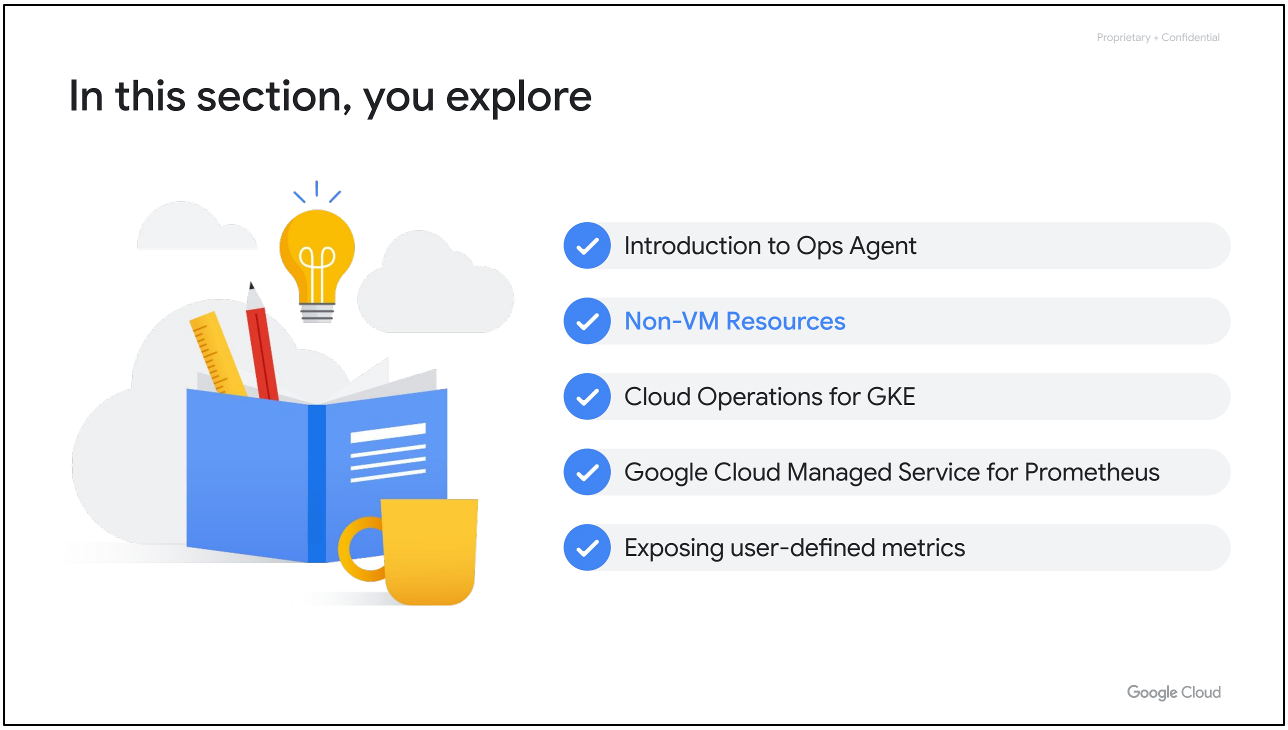
Anthos is a modern application management platform that provides a consistent development and operations experience for cloud and on-premises environments. Use Anthos to modernize existing apps, and build new apps rapidly in hybrid and multi-cloud environments, while enabling consistency between on-premises and cloud environments.

Apache ActiveMQ

The Apache ActiveMQ integration collects storage usage and message metrics. Storage metrics include memory and disk usage. Message metrics include number of waiting messages, average wait time, and expired messages.

Google Cloud

You can preview the dashboards and charts for telemetry data collected from the third-party applications such as Apache Web Server, MySQL, and Redis for deployments that run on Compute Engine and Google Kubernetes Engine. To view the logs, metrics and dashboards for data collected through Ops agent, navigate to **Monitoring**, and select the **Integrations** page.



Proprietary + Confidential

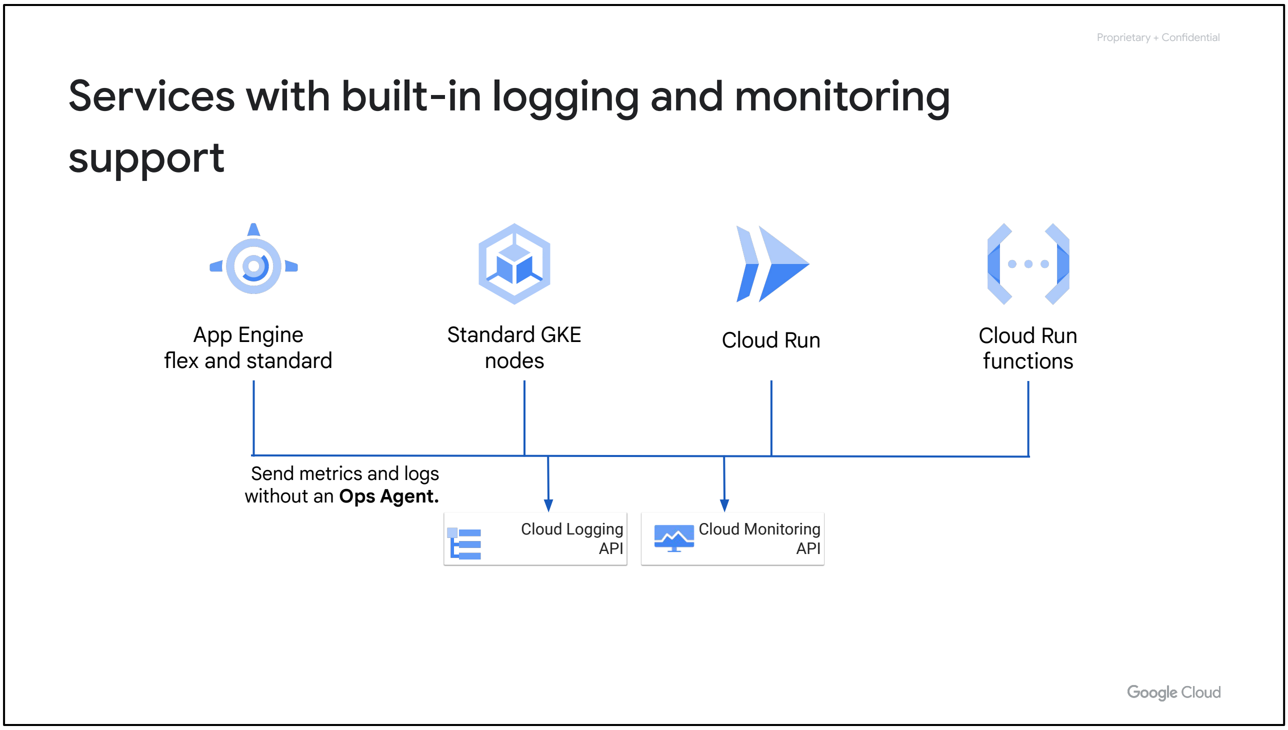
In this section, you explore

- ✓ Introduction to Ops Agent
- ✓ Non-VM Resources
- ✓ Cloud Operations for GKE
- ✓ Google Cloud Managed Service for Prometheus
- ✓ Exposing user-defined metrics

Google Cloud

In addition to Google Cloud virtual machines, there are a lot of Google Cloud resources that support some type of monitoring.

Let's look at a few of these.



Proprietary + Confidential

Services with built-in logging and monitoring support

App Engine flex and standard

Standard GKE nodes

Cloud Run

Cloud Run functions

Send metrics and logs without an **Ops Agent.**

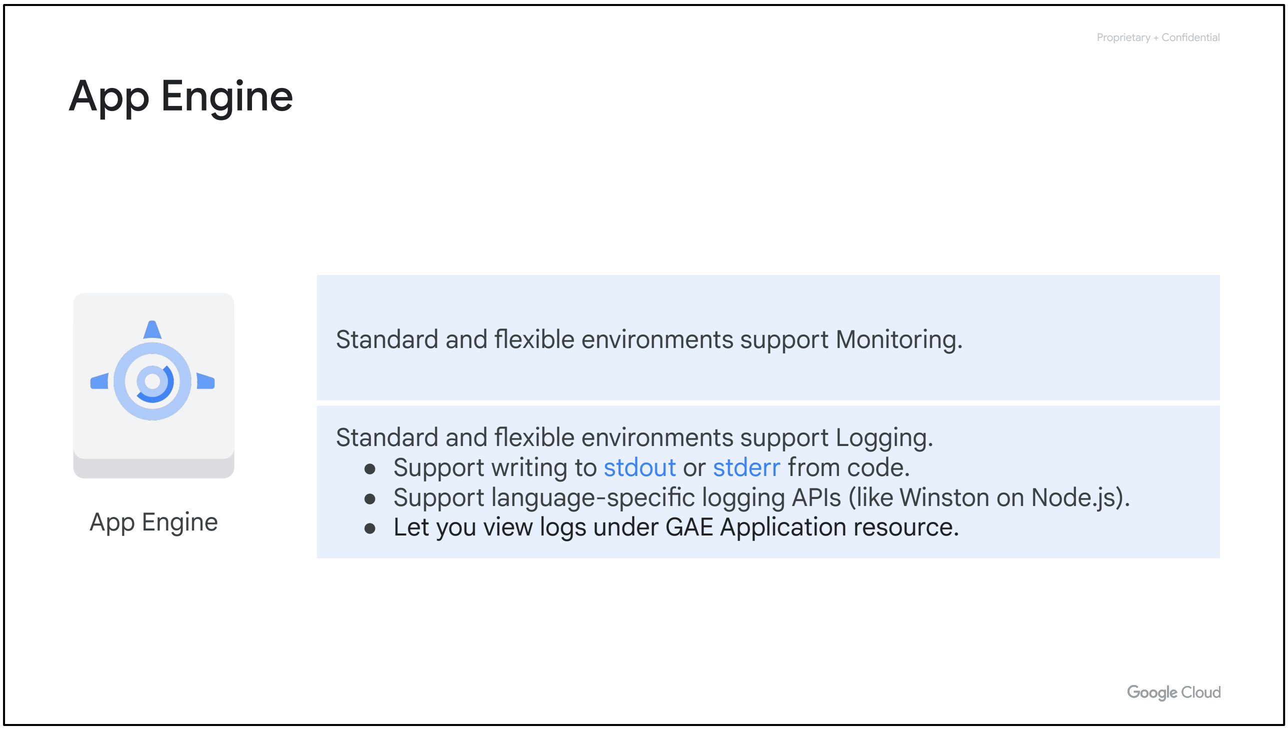
Cloud Logging API

Cloud Monitoring API

Google Cloud

When monitoring any of the following non-virtual machine systems in Google Cloud, the Ops Agent is not required, and should not be installed:

- App Engine standard environment has monitoring built-in.
- App Engine flexible environment is built on top of GKE and has the Monitoring agent pre-installed and configured.
- With Standard Google Kubernetes Engine nodes (VMs), Cloud Logging and Cloud Monitoring is an option which is enabled by default.
- Cloud Run provides integrated monitoring support.
- And Cloud Run functions supports integrated monitoring.



App Engine

App Engine

Standard and flexible environments support Monitoring.

Standard and flexible environments support Logging.

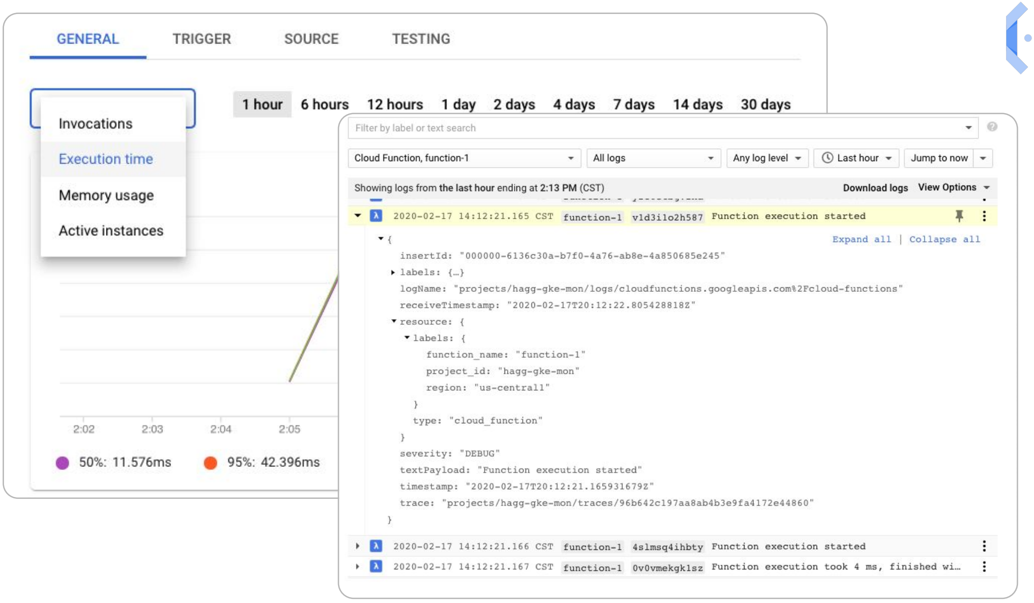
- Support writing to stdout or stderr from code.
- Support language-specific logging APIs (like Winston on Node.js).
- Let you view logs under GAE Application resource.

Google's App Engine standard and flexible environments both support monitoring. Make sure to check Google's documentation for the metric details.

They also both support logging by writing to standard out or error. For refined logging capabilities, review the language-specific logging APIs, such as Winston for Node.js.

Also, the logs are viewable under the GAE Application resource.

Monitoring and logging in Cloud Run functions



Showing logs from the last hour ending at 2:13 PM (CST)

```
2020-02-17 14:12:21.165 CST function-1 v1d3i1o2h587 Function execution started
{
  insertId: "000000-6136c30a-b7f0-4a76-ab8e-4a850685e245"
  labels: {…}
  logName: "projects/hagg-gke-mon/logs/cloudfunctions.googleapis.com%2Fcloud-functions"
  receiveTimestamp: "2020-02-17T20:12:22.805428818Z"
  resource: {
    labels: {
      function_name: "function-1"
      project_id: "hagg-gke-mon"
      region: "us-central1"
    }
    type: "cloud_function"
  }
  severity: "DEBUG"
  textPayload: "Function execution started"
  timestamp: "2020-02-17T20:12:21.165931679Z"
  trace: "projects/hagg-gke-mon/traces/96b642c197aa8ab4b3e9fa4172e44860"
}
2020-02-17 14:12:21.166 CST function-1 4slmsq4ihbty Function execution started
2020-02-17 14:12:21.167 CST function-1 0v0vmekgklsz Function execution took 4 ms, finished wi…
```

Google Cloud

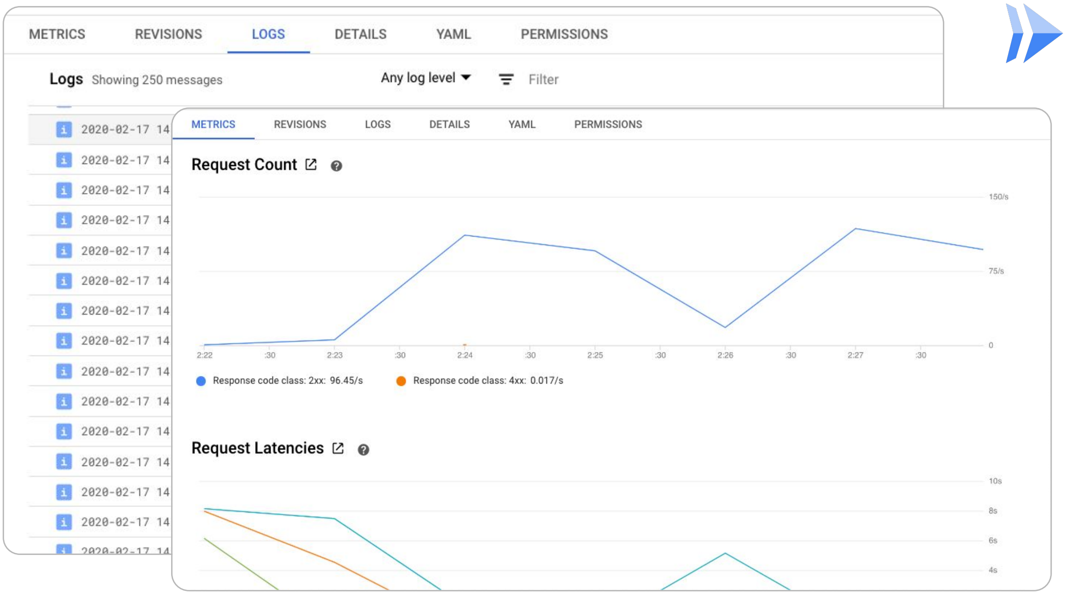
Cloud Run functions offers lightweight, purpose-built functions, typically invoked in response to an event. For example, you might upload a PDF file to a Cloud Storage bucket, the new file triggers an event that invokes a Cloud Run functions instance, which translates the PDF from English to Spanish.

Cloud Run functions monitoring is automatic and can provide you with access to invocations, execution times, memory usage, and active instances in the Google Cloud console. These metrics are also available in Cloud Monitoring, where you can set up custom alerting and dashboards for these metrics.

Cloud Run functions also support simple logging by default. Logs written to standard out or standard error will appear automatically in the Google Cloud console. The logging API can also be used to extend log support.

Monitoring and logging in Cloud Run

Proprietary + Confidential



```
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
i 2020-02-17 14
```

Google Cloud

Cloud Run is Google’s managed container service. It can run in a fully managed version, in which it acts as a sort of App Engine for containers, and it can also run on GKE, in which case it’s a managed version of the open-source KNative. Cloud Run is automatically integrated with Cloud Monitoring **with no setup or configuration required**. This means that metrics of your Cloud Run services are captured automatically when they are running.

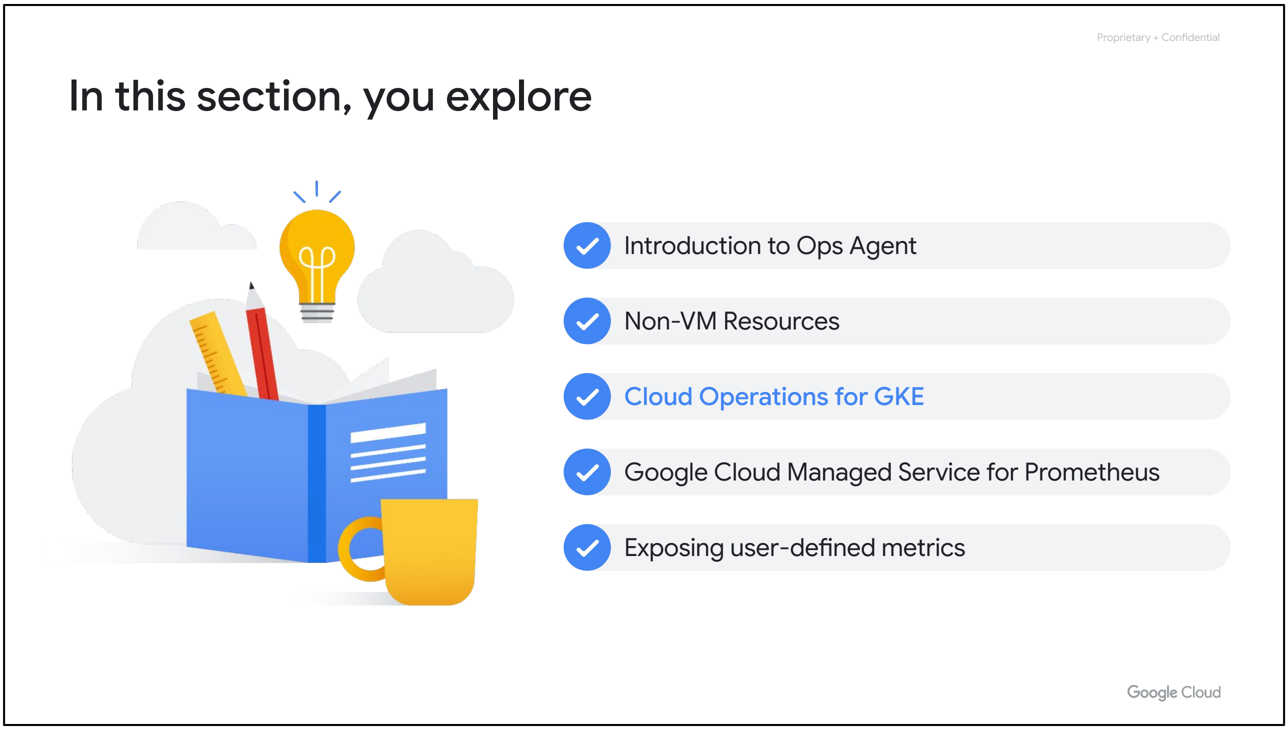
You can view metrics either in Cloud Monitoring or on the Cloud Run page in the console. Cloud Monitoring provides more charting and filtering options.

The resource type differs for fully managed Cloud Run and Cloud Run for Anthos:

- For fully managed Cloud Run, the monitoring resource name is "Cloud Run Revision" (*cloud_run_revision*).
- For Cloud Run for Anthos, the monitoring resource name is "Cloud Run on GKE Revision" (*knative_revision*).

Cloud Run has two types of logs which it automatically sends to Cloud Logging:

- Request logs: logs of requests sent to Cloud Run services.
- And Container logs: logs emitted from the container instances from your own code, written to standard out or standard error streams, or using the logging API.

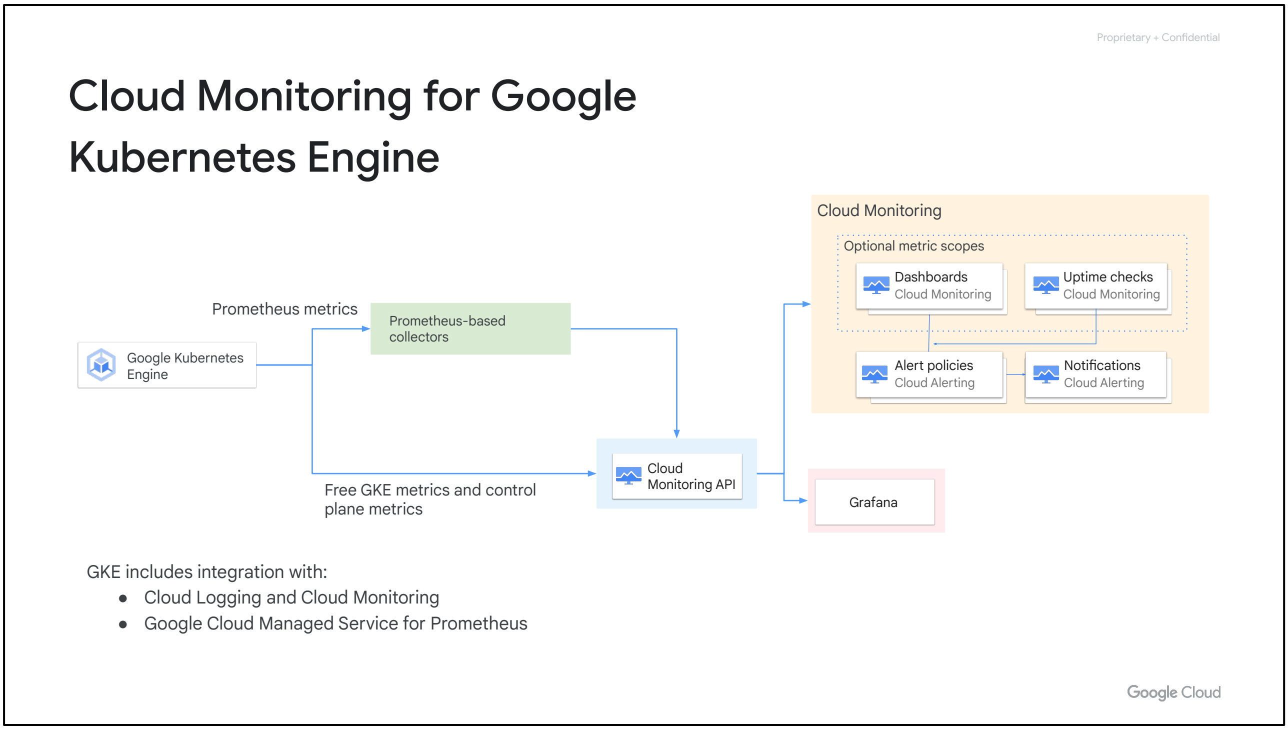


In this section, you explore

- ✓ Introduction to Ops Agent
- ✓ Non-VM Resources
- ✓ Cloud Operations for GKE
- ✓ Google Cloud Managed Service for Prometheus
- ✓ Exposing user-defined metrics

Google Cloud

Let's look at some of the options explicitly available for GKE.



Proprietary + Confidential

Cloud Monitoring for Google Kubernetes Engine

Cloud Monitoring

Optional metric scopes

Dashboards
Cloud Monitoring

Uptime checks
Cloud Monitoring

Alert policies
Cloud Alerting

Notifications
Cloud Alerting

Prometheus metrics

Prometheus-based collectors

Google Kubernetes Engine

Cloud Monitoring API

Free GKE metrics and control plane metrics

Grafana

GKE includes integration with:

- Cloud Logging and Cloud Monitoring
- Google Cloud Managed Service for Prometheus

Google Cloud

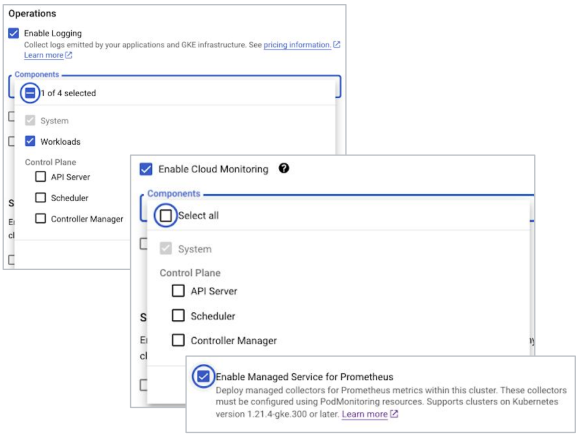
Google Kubernetes Engine (GKE) includes integration with Cloud Logging and Cloud Monitoring. Cloud Managed Service for Prometheus. When you create a GKE cluster that runs on Google Cloud, Cloud Logging and Cloud Monitoring are enabled by default and provide observability specifically tailored for Kubernetes.

You also enable Google Cloud Managed service for Prometheus to collect Prometheus metrics and monitor workloads running on GKE and non-GKE compute workloads. We will explore this in detail in the next section.

Configuring Cloud Operations for GKE clusters

For new cluster

- Cloud Monitoring and Cloud Logging is enabled by default.
- Under **Components**, select the components from which you want to collect logs and metrics.
- Optionally, you can enable Managed Service for Prometheus.



Operations

☑ Enable Logging
Collect logs emitted by your applications and GKE infrastructure. See pricing information.
Learn more

Components
⊟ 1 of 4 selected
☑ System
☑ Workloads
Control Plane
□ API Server
□ Scheduler
□ Controller Manager

☑ Enable Cloud Monitoring

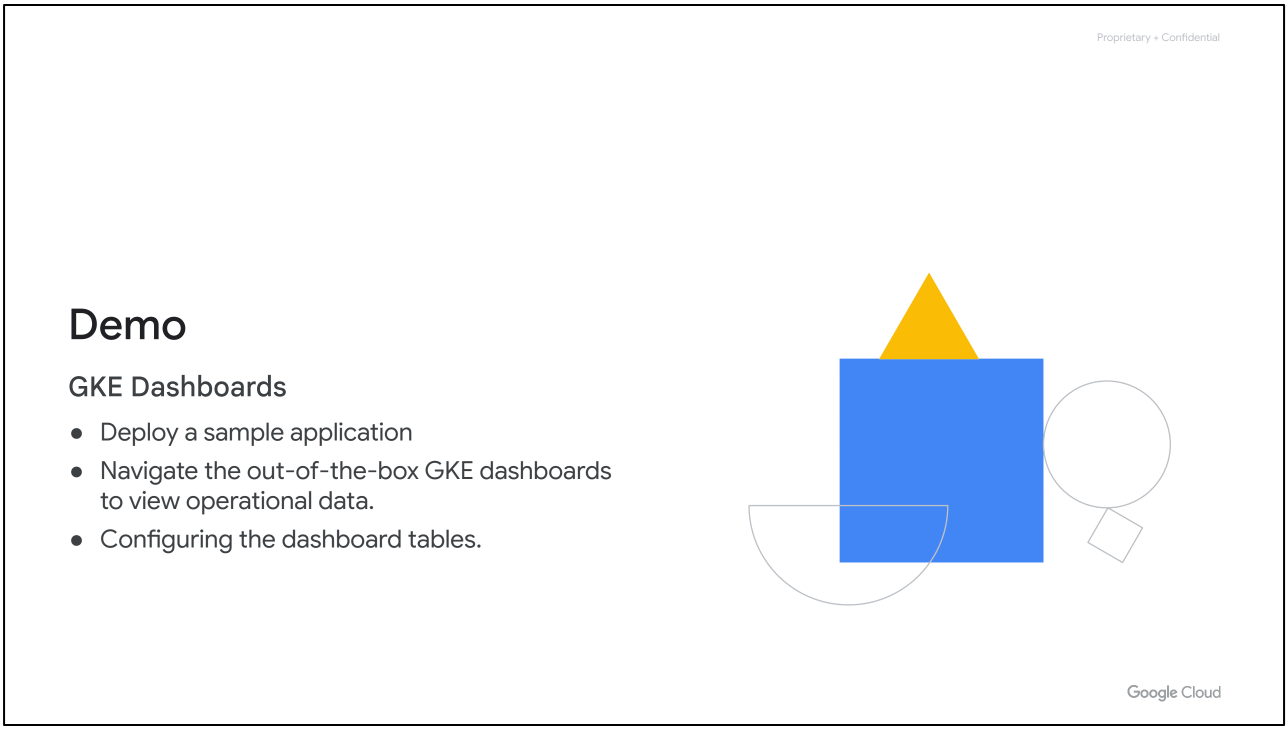
Components
□ Select all
☑ System
Control Plane
□ API Server
□ Scheduler
□ Controller Manager

☑ Enable Managed Service for Prometheus
Deploy managed collectors for Prometheus metrics within this cluster. These collectors must be configured using PodMonitoring resources. Supports clusters on Kubernetes version 1.21.4-gke.300 or later. Learn more

You can configure Cloud Logging and Cloud Monitoring for GKE clusters either during creation or after creation.

During cluster creation, navigate to Standard mode under Operations.

- For new cluster, Cloud Logging and Cloud Monitoring are enabled by default.
- You can change the components for which the metrics and logs are collected under the Components section.
- Enable managed collection by selecting Managed Service for Prometheus.



Demo

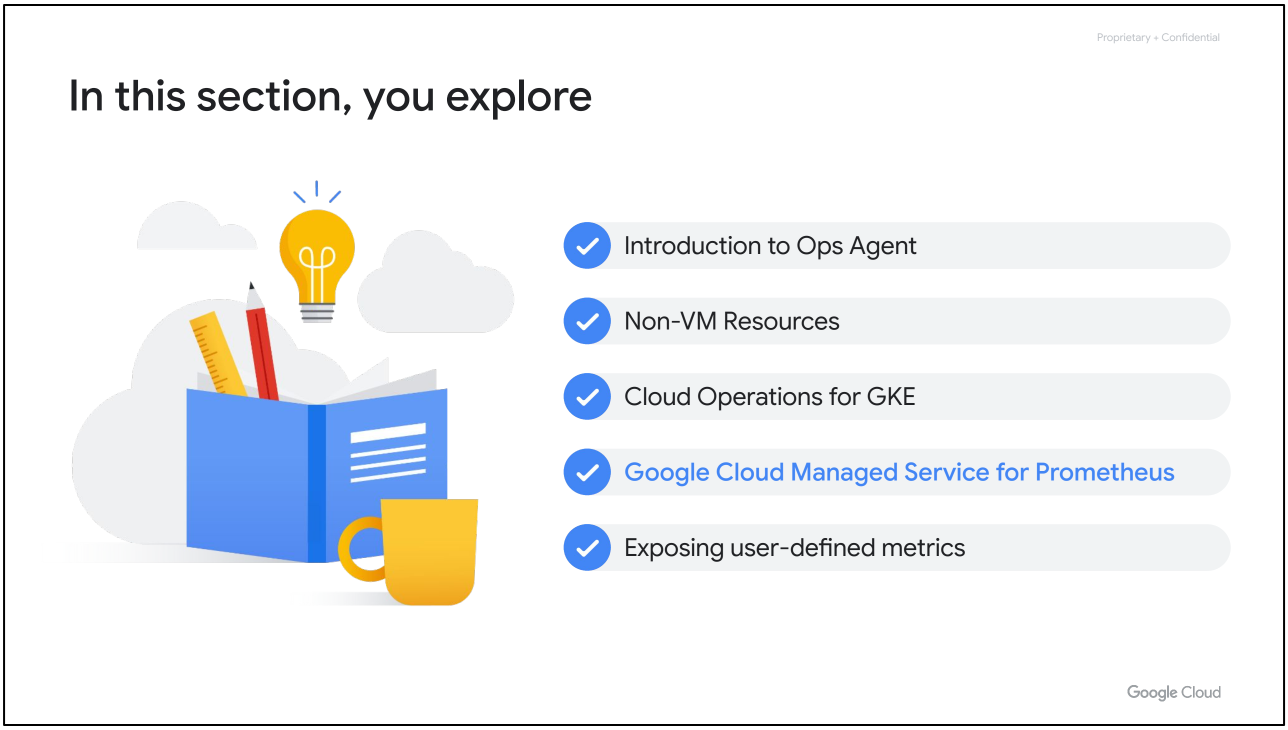
GKE Dashboards

- Deploy a sample application
- Navigate the out-of-the-box GKE dashboards to view operational data.
- Configuring the dashboard tables.

Instructor note: Please open the console to show the Demo using these instructions.
If you can't access the console, use this video.

In this demo:

- Navigate the out-of-the-box GKE dashboards to view operational data.
- Configuring the dashboard tables
- Previewing a few pre-built dashboards from the sample library

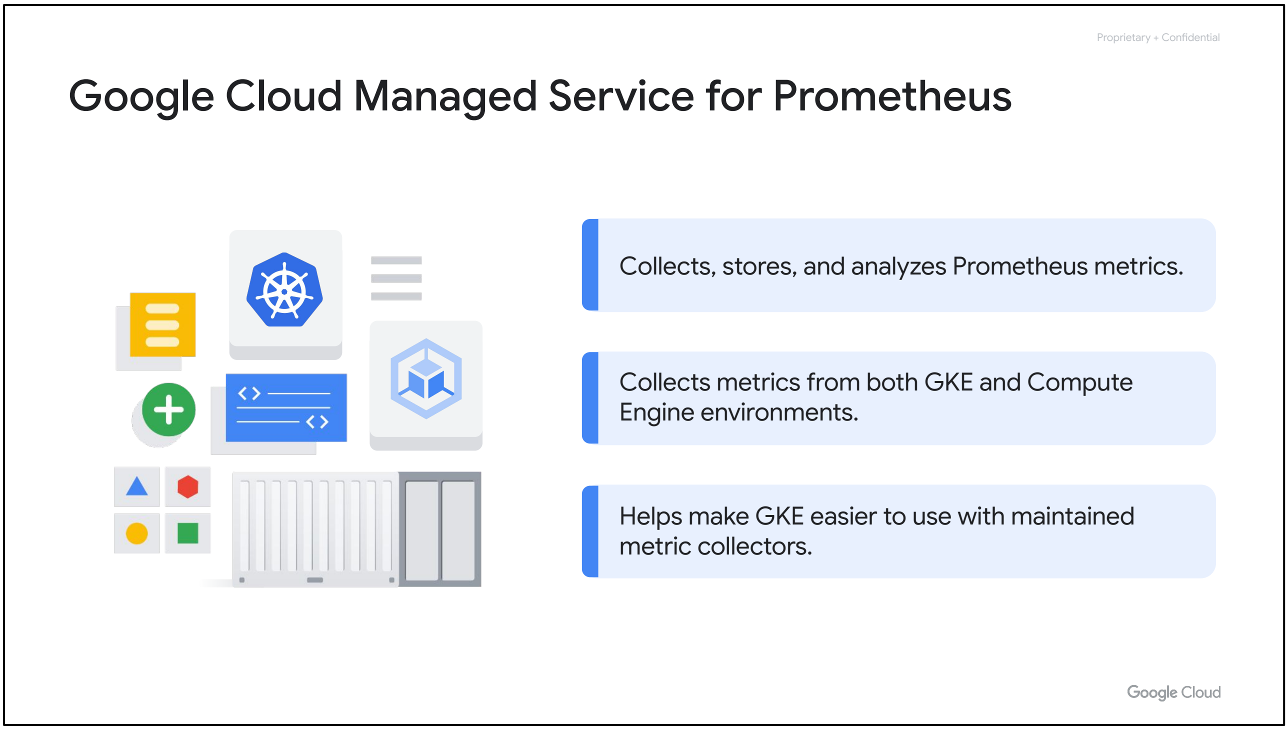


In this section, you explore

- ✓ Introduction to Ops Agent
- ✓ Non-VM Resources
- ✓ Cloud Operations for GKE
- ✓ Google Cloud Managed Service for Prometheus
- ✓ Exposing user-defined metrics

Google Cloud

Next, let’s look at what Google Cloud Managed Service for Prometheus is all about and how it helps collect metrics.



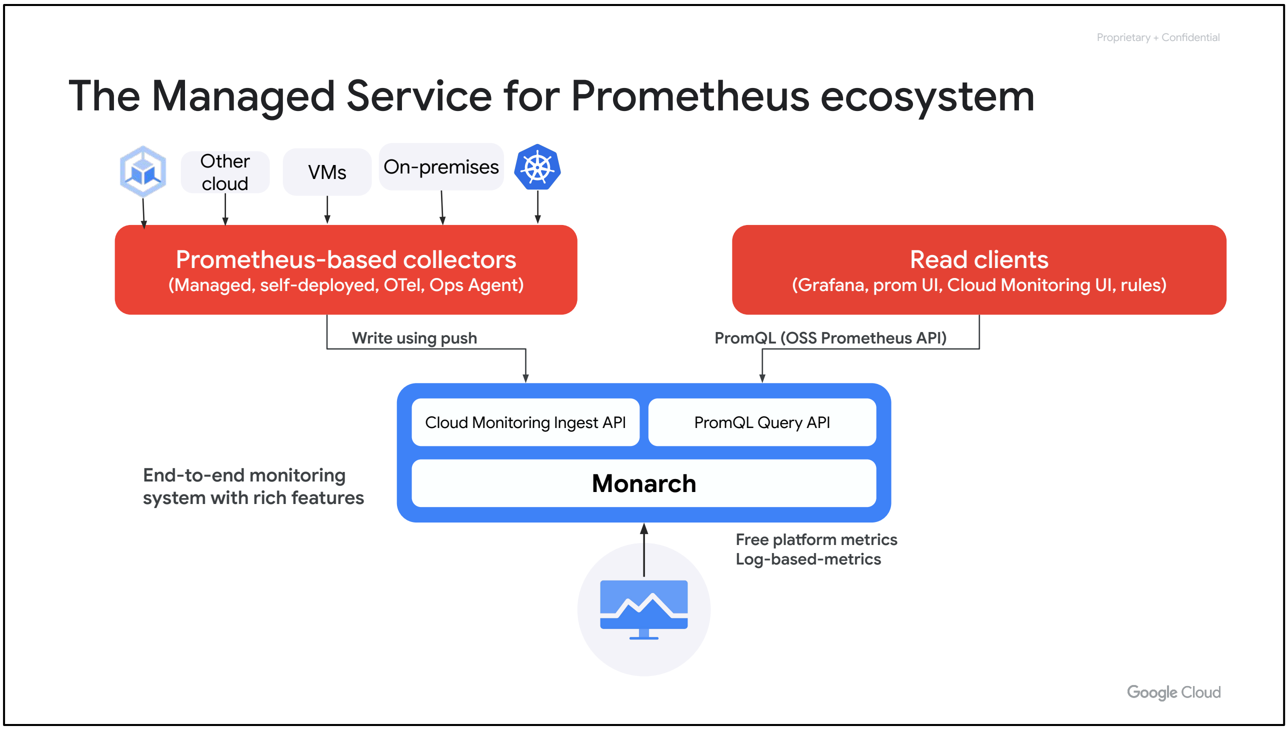
Google Cloud Managed Service for Prometheus

- Collects, stores, and analyzes Prometheus metrics.
- Collects metrics from both GKE and Compute Engine environments.
- Helps make GKE easier to use with maintained metric collectors.

Google Cloud Managed Service for Prometheus is a fully managed service that makes it easy to collect, store, and analyze Prometheus metrics.

Managed Service for Prometheus lets users collect metrics from both Kubernetes and VM environments at incredible scale without operational overhead by leveraging Monarch, Google's own globally available and scalable time series database.

For those getting started with Prometheus for the first time, Managed Service for Prometheus helps make Google Kubernetes Engine even easier to use with maintained metric collectors.



Proprietary + Confidential

The Managed Service for Prometheus ecosystem

Other cloud

VMs

On-premises

Prometheus-based collectors
(Managed, self-deployed, OTel, Ops Agent)

Read clients
(Grafana, prom UI, Cloud Monitoring UI, rules)

Write using push

PromQL (OSS Prometheus API)

Cloud Monitoring Ingest API

PromQL Query API

Monarch

End-to-end monitoring
system with rich features

Free platform metrics
Log-based-metrics

Google Cloud

It's built on top of Monarch, the same globally-scalable data store as Cloud Monitoring.

Monarch is an end-to-end monitoring system with high-level data modeling, data collection, querying, alerting and data management features. You can replace your existing Prometheus deployment to collect cluster and workload metrics and then query the data across multiple clusters by using PromQL.

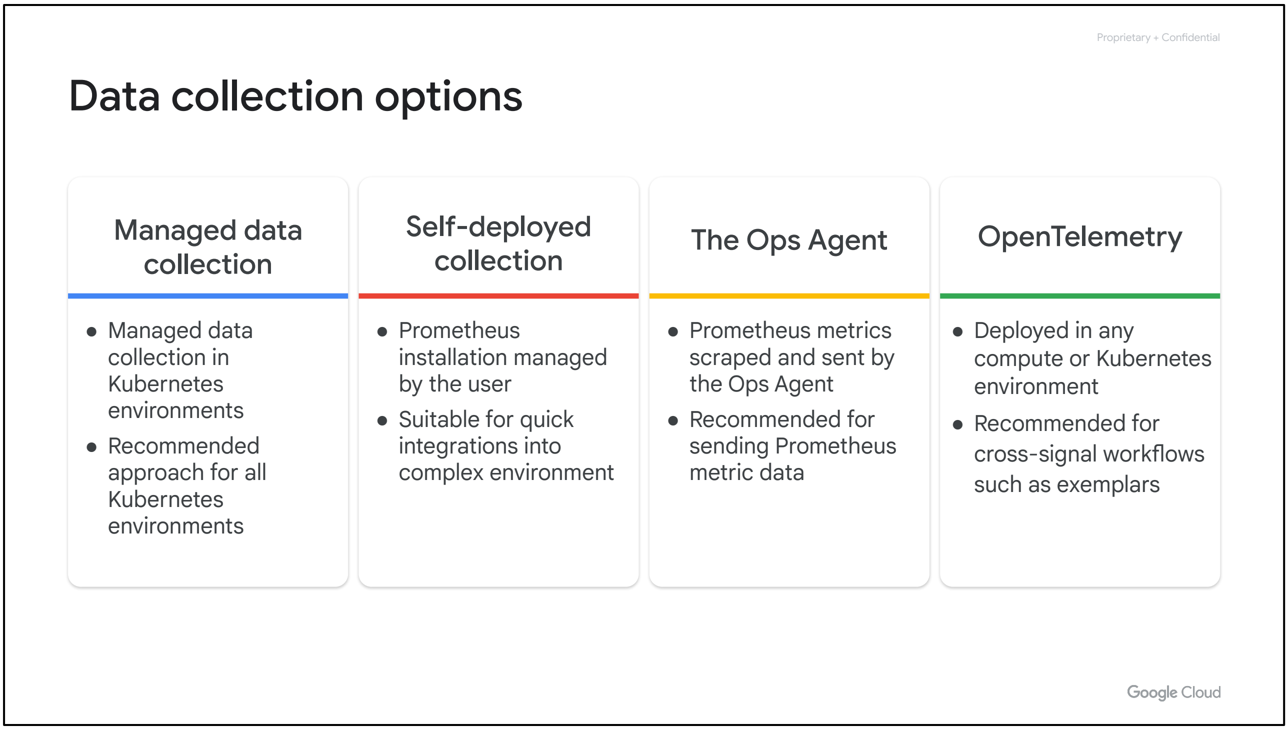
Managed Service for Prometheus splits responsibilities for data collection, query evaluation, rule and alert evaluation, and data storage into multiple components.

- **Query evaluation and data storage:** Monarch handles the query evaluation and data storage. Monarch can execute queries and unions results across all Google Cloud regions. It also supports two years of metric retention by default at no additional cost.
- **Data collection:** There are multiple choices available for data collection, which includes self-deployed, Ops Agent, OpenTelemetry, and managed collection. These collectors are responsible for scraping local exporters and forwarding that data to Monarch.
- **Rule evaluation:** Rule evaluation on the other hand is handled by locally run and configured rule evaluators. Refer to the documentation for latest information on the storage granularity and retention timeline. Another important Query Validator activity of Prometheus monitoring is making queries.

Any UI that can call the Prometheus query API is also supported in the managed service for Prometheus. That includes Grafana and Cloud Monitoring. Your existing dashboards in Grafana continue to work just as before and you can keep using any PromQL found in popular open source repositories and forums.

PromQL can be used to query:

- 1,500 free metrics in Cloud Monitoring
- Free Kubernetes metrics, custom metrics, and log-based metrics



Data collection options

Managed data collection

- Managed data collection in Kubernetes environments
- Recommended approach for all Kubernetes environments

Self-deployed collection

- Prometheus installation managed by the user
- Suitable for quick integrations into complex environment

The Ops Agent

- Prometheus metrics scraped and sent by the Ops Agent
- Recommended for sending Prometheus metric data

OpenTelemetry

- Deployed in any compute or Kubernetes environment
- Recommended for cross-signal workflows such as exemplars

Google Cloud

Data collection: For collecting the data Prometheus will monitor, you can use the service in one of four ways:

- Managed data collection
- Self-deployed data collection
- Using Ops Agent
- OpenTelemetry collection

Managed Service for Prometheus
Managed Service for Prometheus offers an operator for managed data collection in Kubernetes environments. We recommend that you use managed collection; because it eliminates the complexity of deploying, scaling, sharding, configuring, and maintaining Prometheus servers. Managed collection is supported for both GKE and non-GKE Kubernetes environments.

Self-deployed data collection
With self-deployed data collection, you manage your Prometheus installation as you always have. The only difference from upstream Prometheus is that you run the Managed Service for Prometheus drop-in replacement binary instead of the upstream Prometheus binary.

The Ops Agent
You can configure the Ops Agent on any Compute Engine instance to scrape and send Prometheus metrics to the global data store. Using an agent simplifies VM

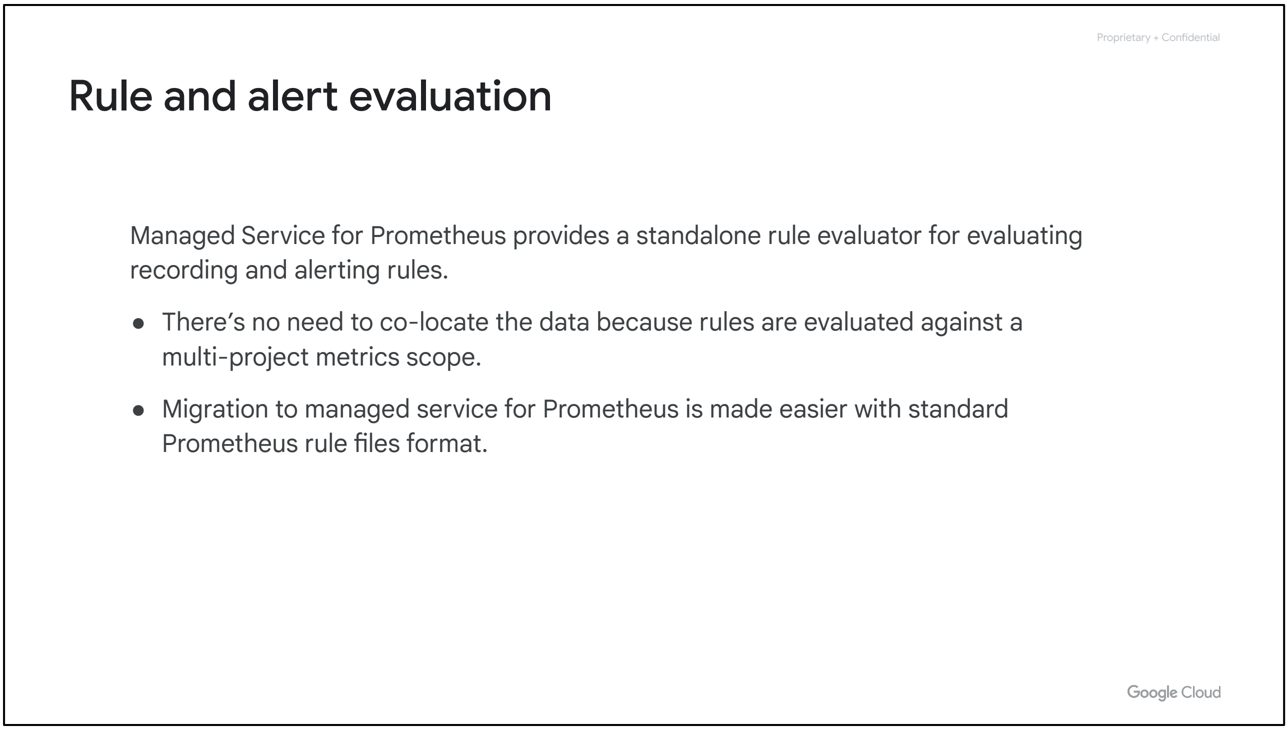
discovery and eliminates the need to install, deploy, or configure Prometheus in VM environments.

OpenTelemetry Collector

OpenTelemetry Collector uses a single collector to collect metrics from any environment and then sends them to any compatible backend. It is deployed either manually or by using Terraform in any compute or Kubernetes environment.

When you choose between collection options, consider the following aspects:

- Managed collection is a recommended approach for all Kubernetes environments and is especially suitable for more hands-off fully managed experience.
- Self-deployed collection is suitable for quick integration into more complex environment.
- Using the Ops Agent is the easiest way and is recommended to collect and send Prometheus metric data originating from Compute Engine environments, including both Linux and Windows distros.
- The OpenTelemetry Collector is best to support cross-singal workflows such as examplers.



Rule and alert evaluation

Managed Service for Prometheus provides a standalone rule evaluator for evaluating recording and alerting rules.

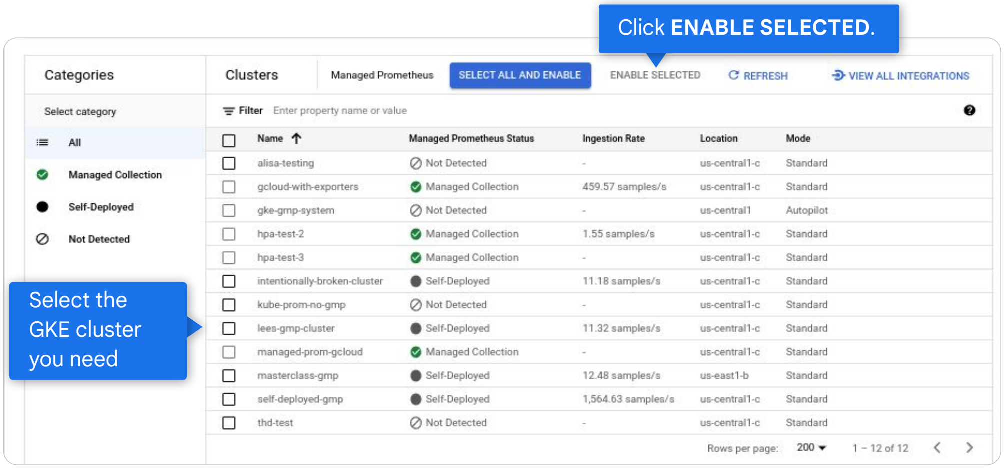
- There’s no need to co-locate the data because rules are evaluated against a multi-project metrics scope.
- Migration to managed service for Prometheus is made easier with standard Prometheus rule files format.

Rule and alert evaluation: Managed Service for Prometheus provides a standalone rule evaluator for evaluating recording and alerting rules against all Monarch data accessible in a metrics scope.

- No need to co-locate the data in a single Prometheus server or on a single Google Cloud project.
- The rule evaluator uses the standard Prometheus rule files format, which makes migration to Managed Service for Prometheus easier.

Proprietary + Confidential

Setting up managed collection for GKE clusters



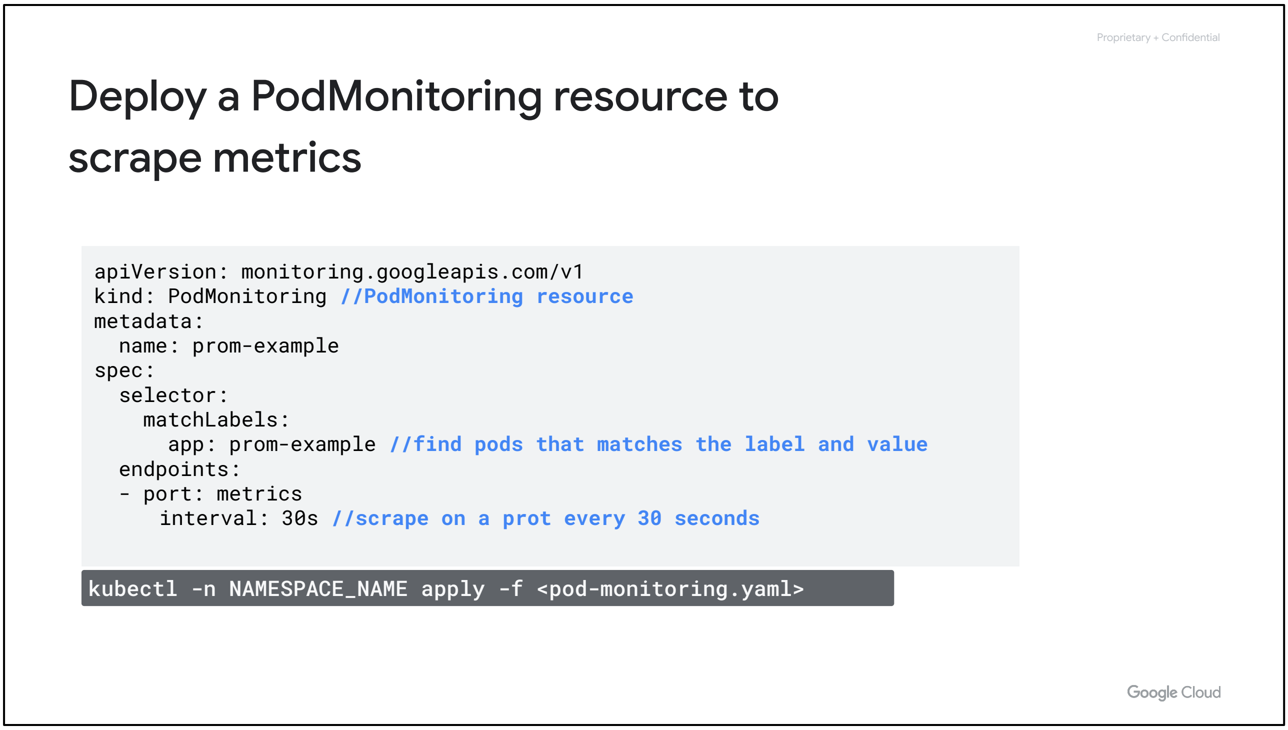
□	Name ↑	Managed Prometheus Status	Ingestion Rate	Location	Mode
□	alisa-testing	⊘ Not Detected	-	us-central1-c	Standard
□	gcloud-with-exporters	✓ Managed Collection	459.57 samples/s	us-central1-c	Standard
□	gke-gmp-system	⊘ Not Detected	-	us-central1	Autopilot
□	hpa-test-2	✓ Managed Collection	1.55 samples/s	us-central1-c	Standard
□	hpa-test-3	✓ Managed Collection	-	us-central1-c	Standard
□	intentionally-broken-cluster	● Self-Deployed	11.18 samples/s	us-central1-c	Standard
□	kube-prom-no-gmp	⊘ Not Detected	-	us-central1-c	Standard
□	lees-gmp-cluster	● Self-Deployed	11.32 samples/s	us-central1-c	Standard
□	managed-prom-gcloud	✓ Managed Collection	-	us-central1-c	Standard
□	masterclass-gmp	● Self-Deployed	12.48 samples/s	us-east1-b	Standard
□	self-deployed-gmp	● Self-Deployed	1,564.63 samples/s	us-central1-c	Standard
□	thd-test	⊘ Not Detected	-	us-central1-c	Standard

Click **ENABLE SELECTED.**

Select the GKE cluster you need

Google Cloud

You can enable a managed collection for your resource by selecting the GKE cluster you need and clicking ENABLE SELECTED.



Deploy a PodMonitoring resource to scrape metrics

```
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring //PodMonitoring resource
metadata:
  name: prom-example
spec:
  selector:
    matchLabels:
      app: prom-example //find pods that matches the label and value
  endpoints:
  - port: metrics
     interval: 30s //scrape on a prot every 30 seconds
```

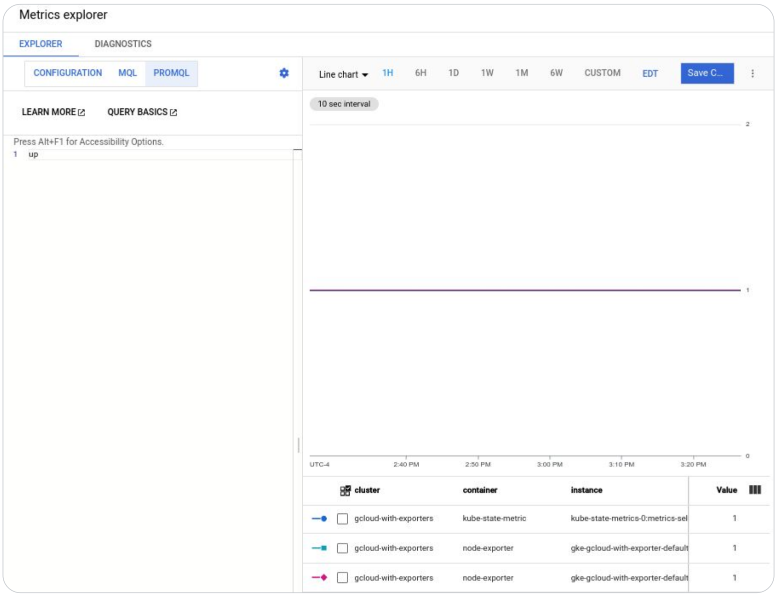
```
kubectl -n NAMESPACE_NAME apply -f <pod-monitoring.yaml>
```

After enabling managed collection, the in-cluster components are running, but no metrics are generated yet. You must deploy a PodMonitoring resource that scrapes a metrics endpoint to see any data in the Query UI.

The manifest shown on slide defines a PodMonitoring resource, **prom-example**, in the NAMESPACE_NAME namespace. The resource uses a Kubernetes label selector to find all the pods in the namespace that have the label app with the value **prom-example**. The matching pods are scraped on a port named metrics, every 30 seconds, on the /metrics HTTP path.

To apply this resource, run the command on screen.
To configure a horizontal collection that applies to a range of pods across all namespaces, use the ClusterPodMonitoring resource. The ClusterPodMonitoring resource provides the same interface as the PodMonitoring resource but does not limit discovered pods to a given namespace.

Using PromQL



```
up
```

	cluster	container	instance	Value
□	gcloud-with-exporters	kube-state-metric	kube-state-metrics-0:metrics-sel	1
□	gcloud-with-exporters	node-exporter	gke-gcloud-with-exporter-default	1
□	gcloud-with-exporters	node-exporter	gke-gcloud-with-exporter-default	1

Google Cloud

When working with metric data, including data from Managed Service for Prometheus, in Cloud Monitoring, you can use the following query tools provided by by Cloud Monitoring:

- PromQL
- Monitoring Query Language (MQL)
- Monitoring filters

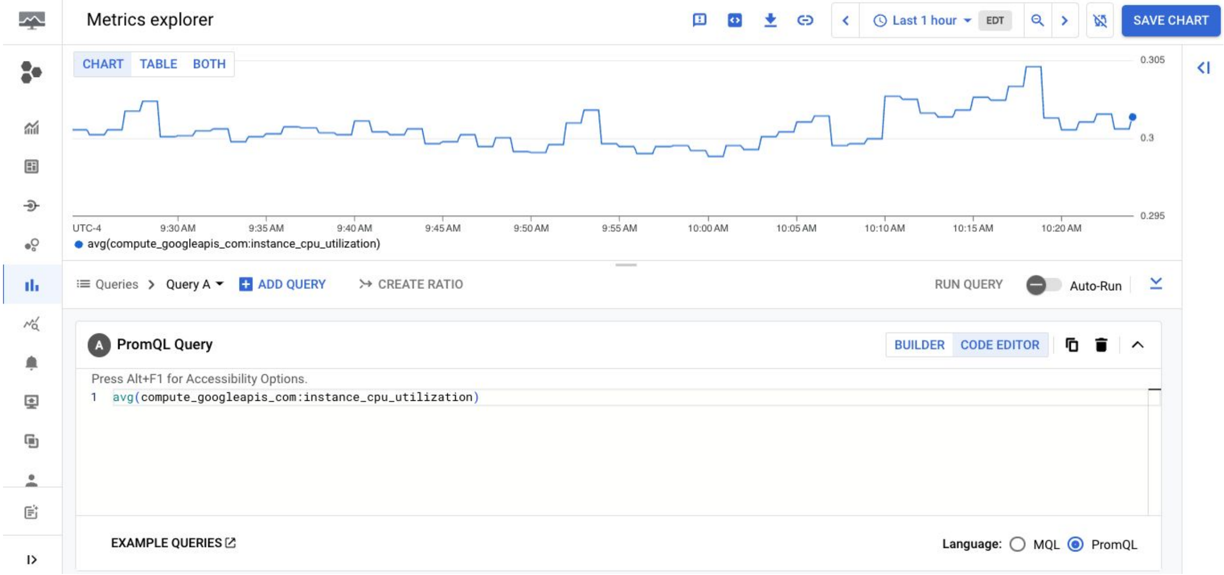
The simplest way to verify that your Prometheus data is being exported is to use the Cloud Monitoring Metrics Explorer page in the Google Cloud console.

In the Google Cloud console, follow these steps:

1. Go to Monitoring
2. In the Monitoring navigation pane, click Metrics Explorer.
3. Select the PromQL tab.
4. Enter the following query: up
5. Click Run Query.

For MQL and filter, refer to the documentation.

PromQL example



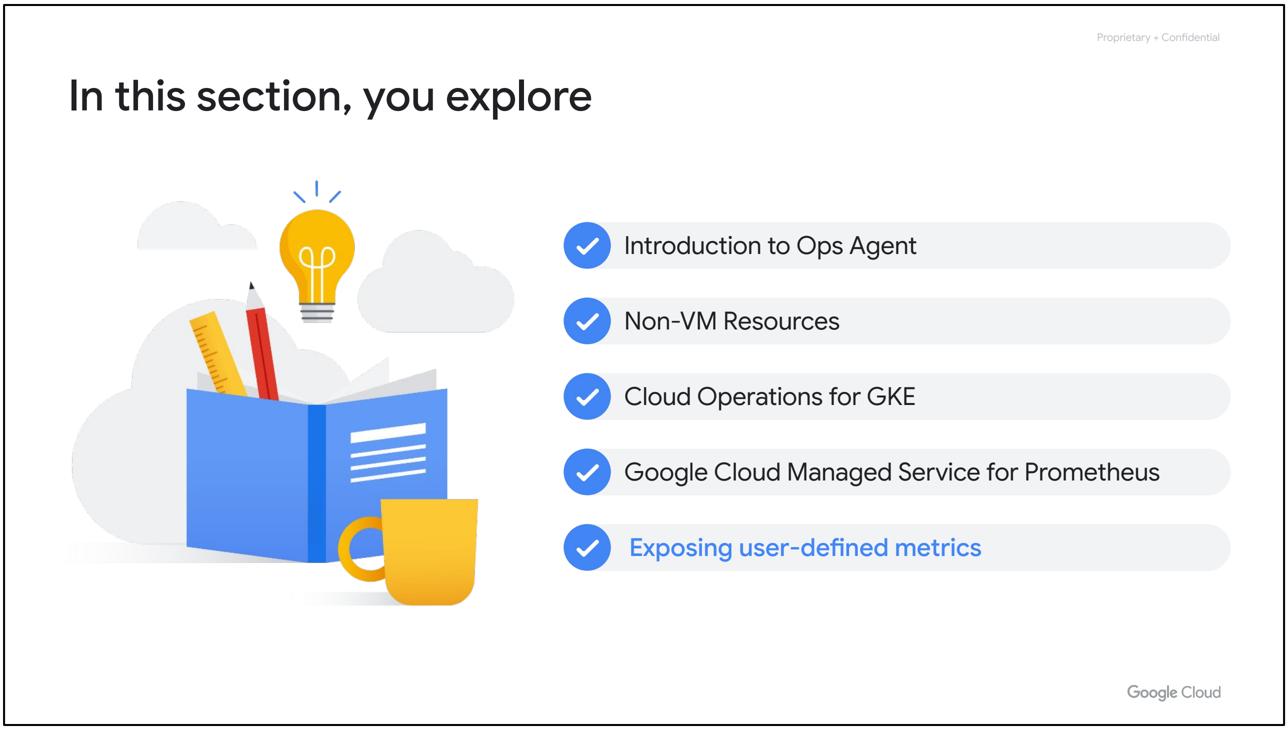
Press Alt+F1 for Accessibility Options.

```
avg(compute_googleapis_com:instance_cpu_utilization)
```

Language: ○ MQL ● PromQL

Here is another example of using PromQL. This PromQL query show the average CPU utilization of the compute instances in your Google Cloud environment. The chart shows the visual representation of the utilization within a span of 1 hour.

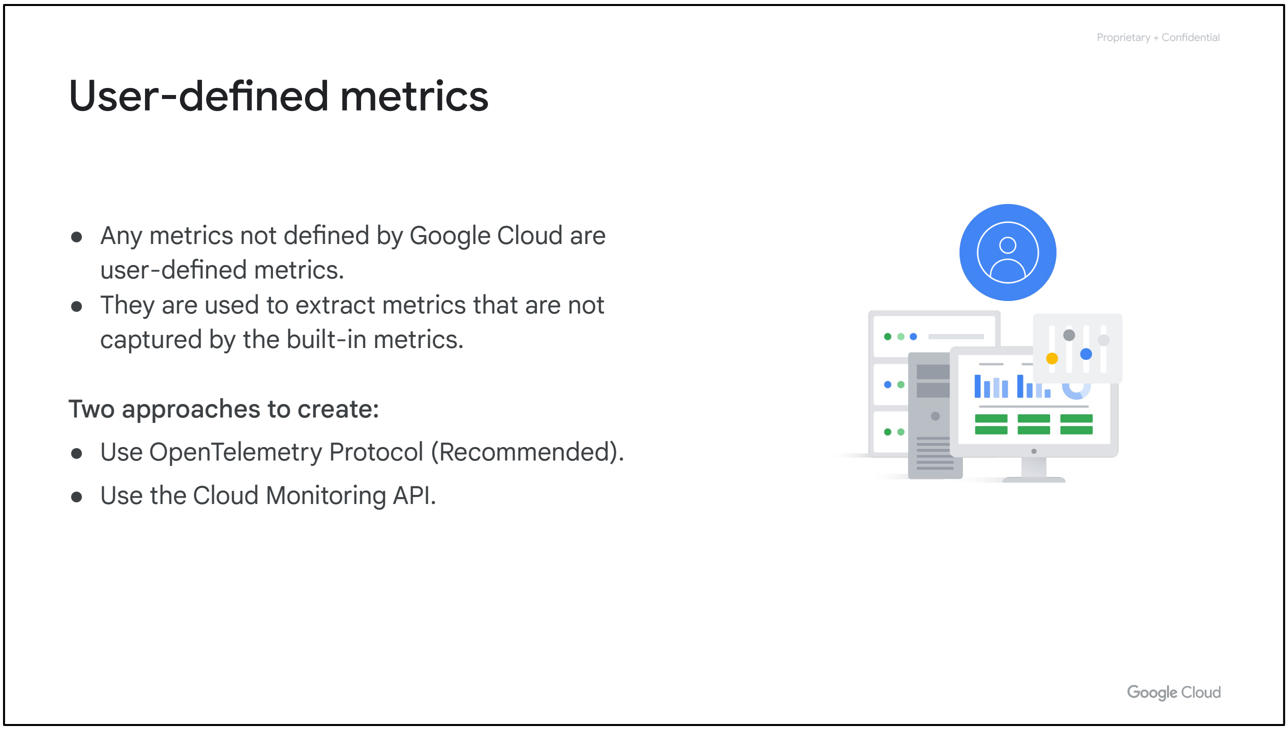
Let us next look at why one would choose Managed service over Prometheus.



In this section, you explore

- ✓ Introduction to Ops Agent
- ✓ Non-VM Resources
- ✓ Cloud Operations for GKE
- ✓ Google Cloud Managed Service for Prometheus
- ✓ Exposing user-defined metrics

Next, let’s look at what Google Cloud Managed Service for Prometheus is all about and how it helps collect metrics.



User-defined metrics

- Any metrics not defined by Google Cloud are user-defined metrics.
- They are used to extract metrics that are not captured by the built-in metrics.

Two approaches to create:

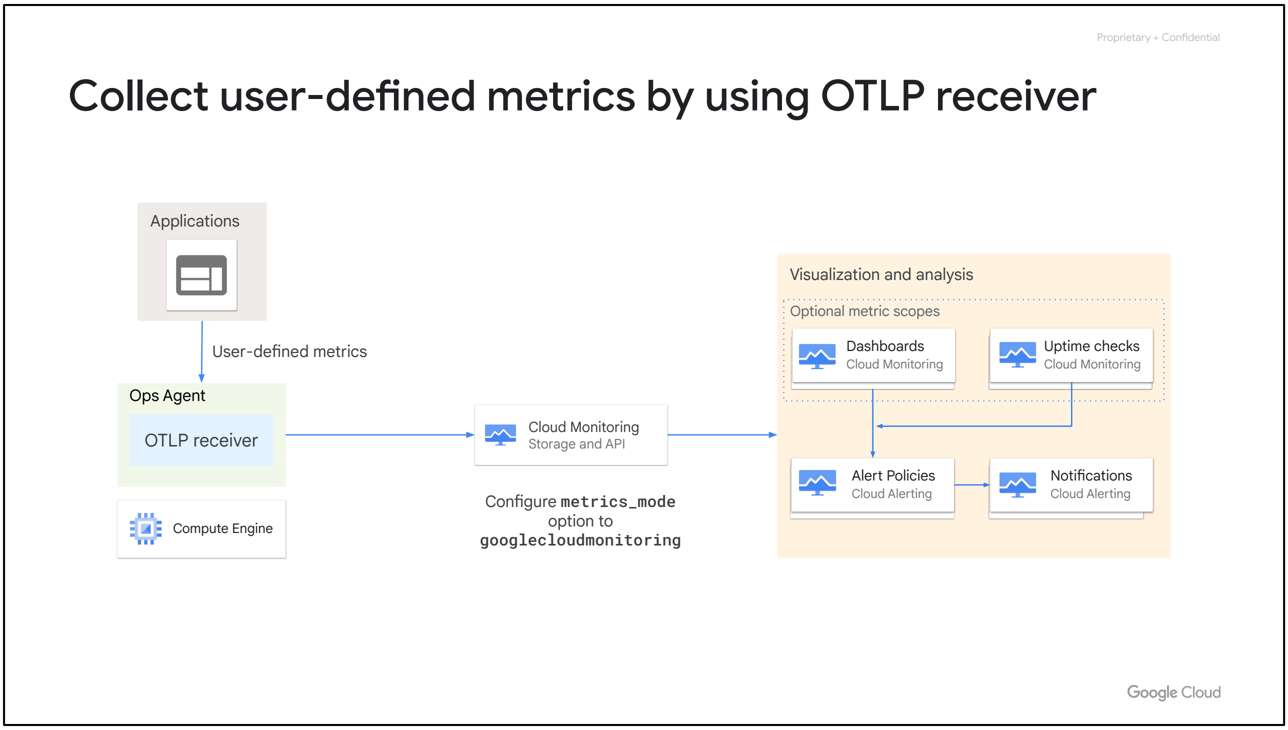
- Use OpenTelemetry Protocol (Recommended).
- Use the Cloud Monitoring API.

Application-specific metrics, also known as user or custom metrics, are metrics that you define and collect to capture information that the built-in Cloud Monitoring metrics cannot. You capture such metrics by using an API provided by a library to instrument your code, and then you send the metrics to Cloud Monitoring.

Custom metrics can be used in the same way as built-in metrics. That is, you can create charts and alerts for your custom metric data.

There are two fundamental approaches to creating custom metrics for Cloud Monitoring:

- You can use the classic Cloud Monitoring API.
- Or you can use the OpenTelemetry protocol and Ops Agent.



Collect user-defined metrics by using OTLP receiver

Applications

User-defined metrics

Ops Agent

OTLP receiver

Compute Engine

Cloud Monitoring
Storage and API

Configure `metrics_mode` option to `googlecloudmonitoring`

Visualization and analysis

Optional metric scopes

Dashboards
Cloud Monitoring

Uptime checks
Cloud Monitoring

Alert Policies
Cloud Alerting

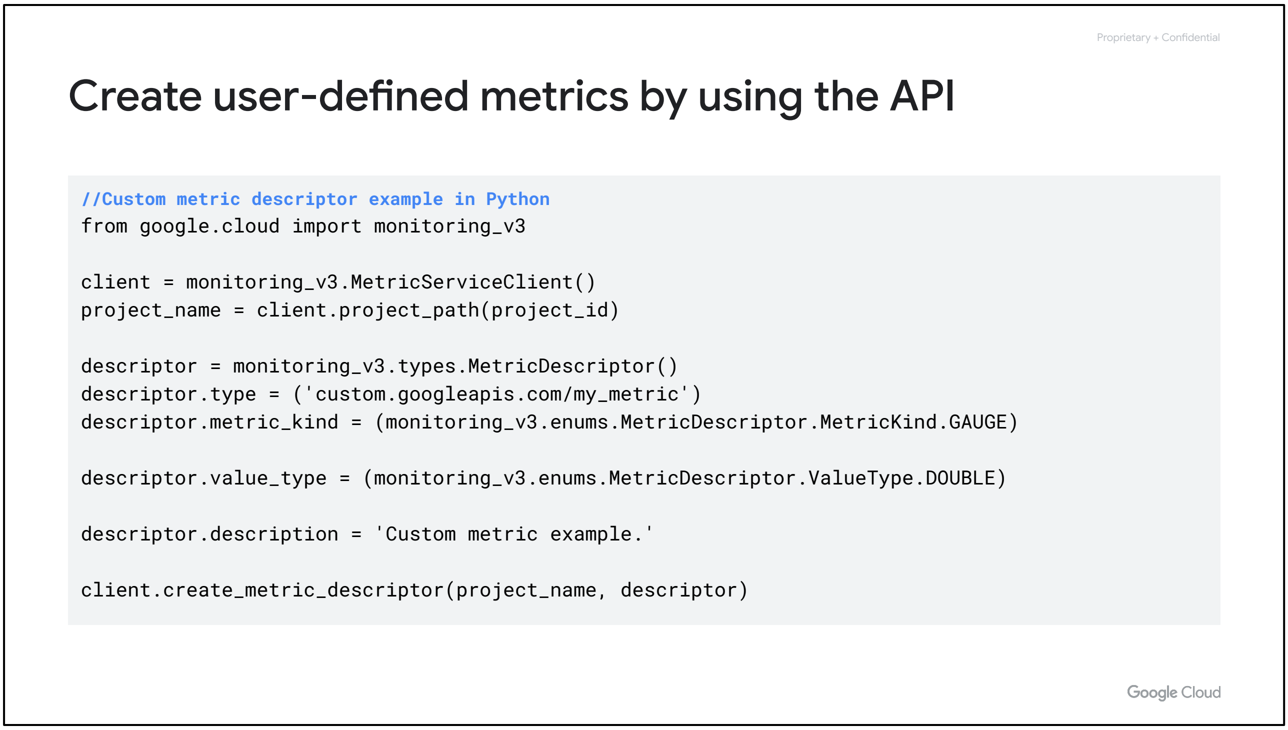
Notifications
Cloud Alerting

The OpenTelemetry Protocol (OTLP) receiver is a plugin installed on the Ops Agent that helps collect the **user-defined metrics form the application** and send those metrics to Cloud Monitoring for analysis and visualization. These metrics can then be used to create dashboards, uptime checks, and altering policies.

The ingestion and authorization is not required as it is handler at the agent level. To configure OTLP, you must install an Ops Agent and modify the user configuration file to include the OTLP file.

By default, the receiver uses the Prometheus API; the default value for the metrics_mode option is googlemanagedprometheus.

To receive the custom metrics from the OTLP receiver, set the OTLP receiver metrics_mode to googlecloudmonitoring.



Proprietary + Confidential

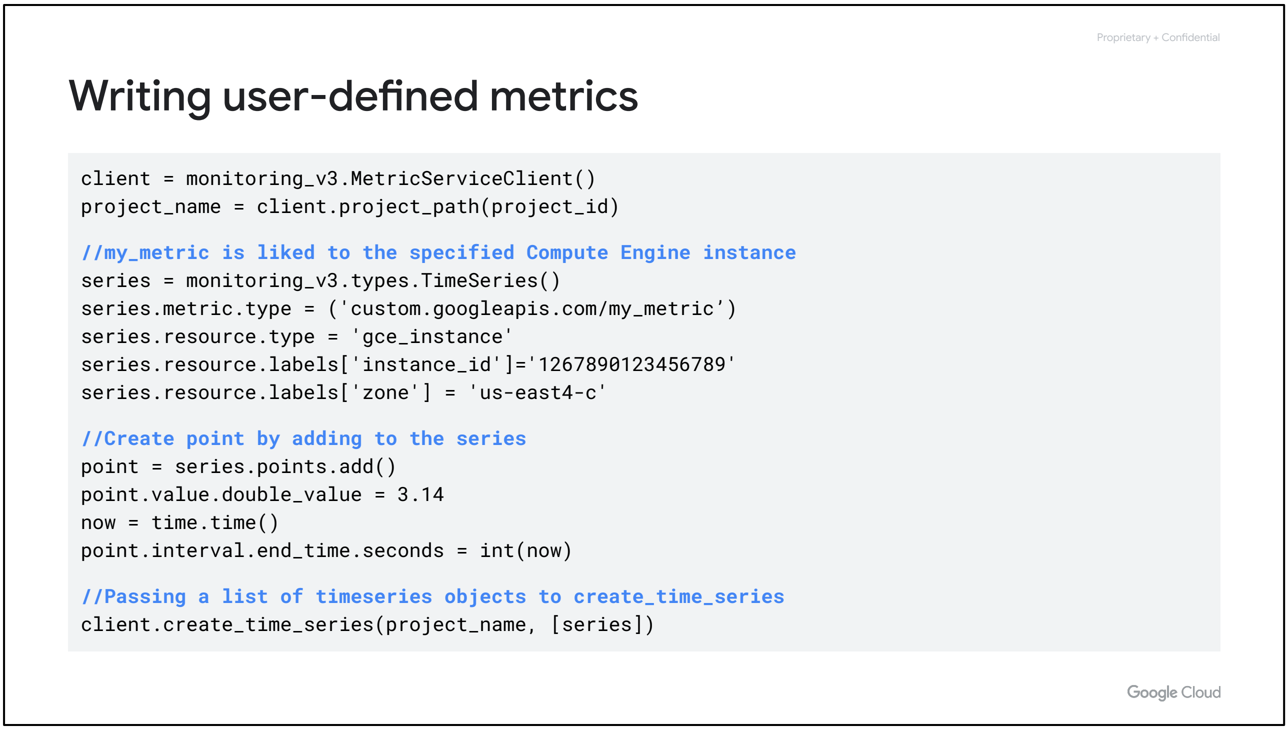
Create user-defined metrics by using the API

```
//Custom metric descriptor example in Python
from google.cloud import monitoring_v3

client = monitoring_v3.MetricServiceClient()
project_name = client.project_path(project_id)

descriptor = monitoring_v3.types.MetricDescriptor()
descriptor.type = ('custom.googleapis.com/my_metric')
descriptor.metric_kind = (monitoring_v3.enums.MetricDescriptor.MetricKind.GAUGE)

descriptor.value_type = (monitoring_v3.enums.MetricDescriptor.ValueType.DOUBLE)

descriptor.description = 'Custom metric example.'

client.create_metric_descriptor(project_name, descriptor)
```

Google Cloud

The steps used to create a custom metric using the API are well documented.

1. To begin, the data you collect for a custom metric must be associated with a descriptor for a custom metric type.
   - In this example, we create a gauge double metric named my_metric. It's a gauge metric of type double, with the description "Custom metric example."
2. After you collect the information you need for creating your custom metric type, call the create method, passing into a MetricDescriptor object.
   - You write data points by passing a list of TimeSeries objects to create_time_series.



Writing user-defined metrics

```
client = monitoring_v3.MetricServiceClient()
project_name = client.project_path(project_id)

//my_metric is liked to the specified Compute Engine instance
series = monitoring_v3.types.TimeSeries()
series.metric.type = ('custom.googleapis.com/my_metric’)
series.resource.type = 'gce_instance'
series.resource.labels['instance_id']='1267890123456789'
series.resource.labels['zone'] = 'us-east4-c'

//Create point by adding to the series
point = series.points.add()
point.value.double_value = 3.14
now = time.time()
point.interval.end_time.seconds = int(now)

//Passing a list of timeseries objects to create_time_series
client.create_time_series(project_name, [series])
```

You write data points by passing a list of TimeSeries objects to the function create_time_series. Each time series is identified by the metric and resource fields of the TimeSeries object.

These fields represent the metric type and the monitored resource from which the data was collected.

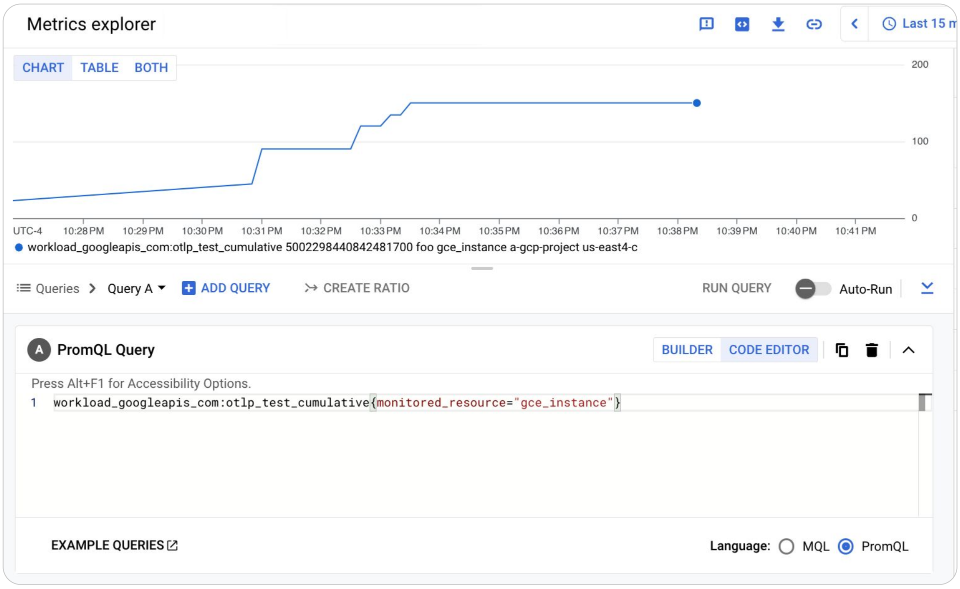
In this example, we use the my_metric described on the last slide to link our metric to the specified Compute Engine instance.

Next, we create the point by adding it to the series and adding the details.

Each TimeSeries object must contain a single Point object.

Finally, we report our metric.

Query custom metrics

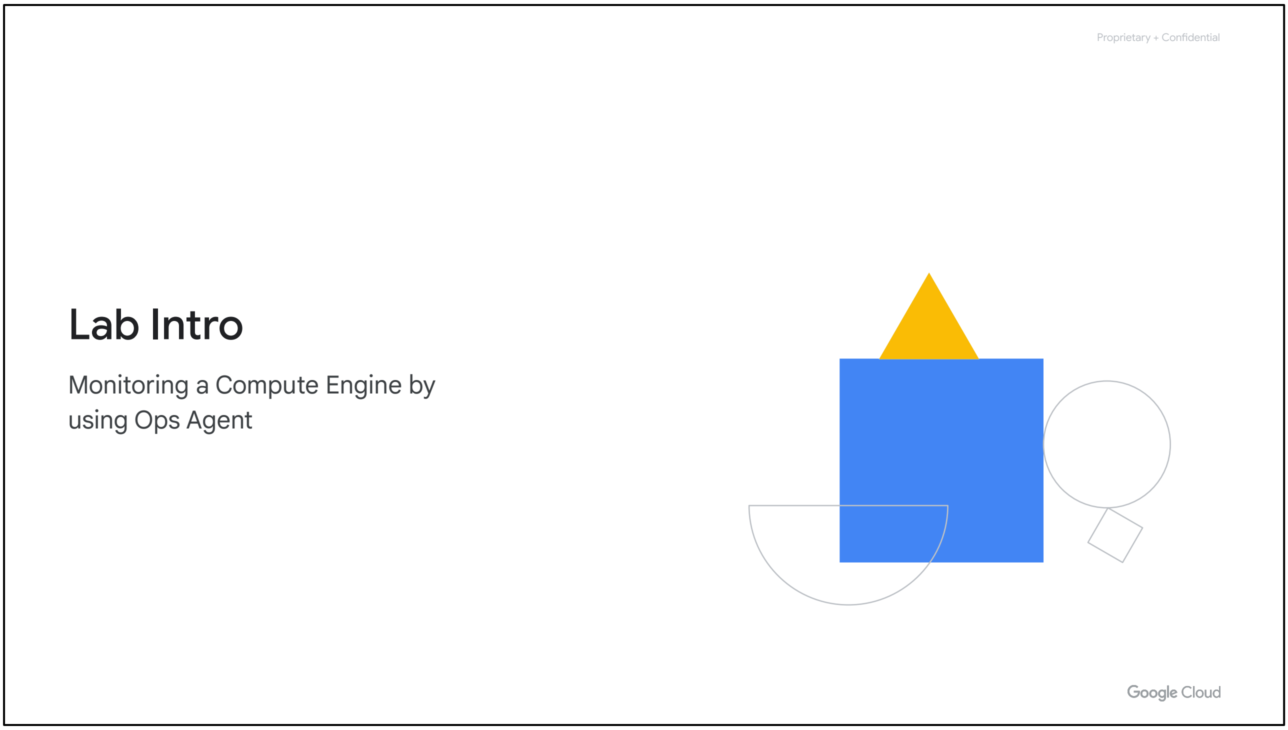


● workload_googleapis_com:otlp_test_cumulative 5002298440842481700 foo gce_instance a-gcp-project us-east4-c

Press Alt+F1 for Accessibility Options.

```
workload_googleapis_com:otlp_test_cumulative{monitored_resource="gce_instance"}
```

After you configure the metrics_mode to Promotetheus API or the Cloud Monitoring API in the OTLP receiver, you can then query the metrics by using Metrics Explorer, dashboards or even altering interface. Here we see an example from the Metrics explorer, where you query the user defined metrics ingested by the monitoring API.

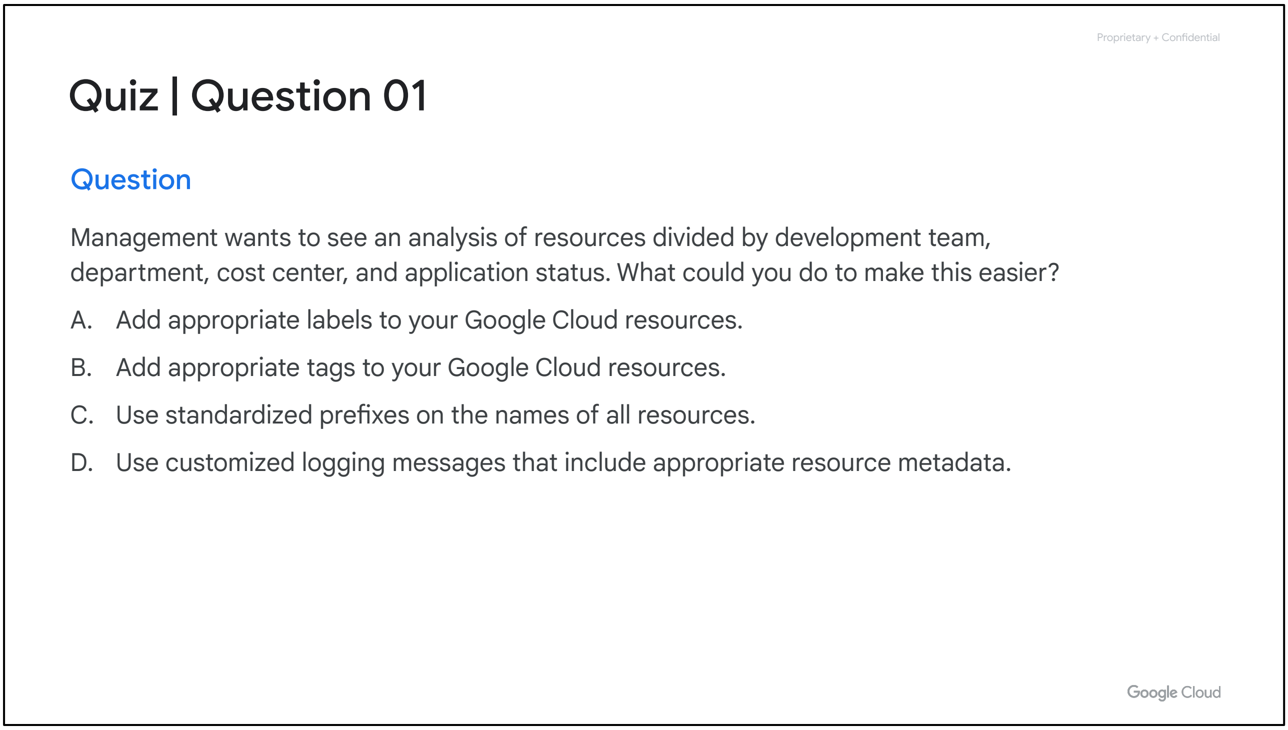


Lab Intro

Monitoring a Compute Engine by using Ops Agent

In this lab, you create a compute engine to install and configure an Ops Agent. You will generate traffic and view metrics on the predefined Apache dashboard and also create an alerting policy.

Knowledge Check



Quiz | Question 01

Question

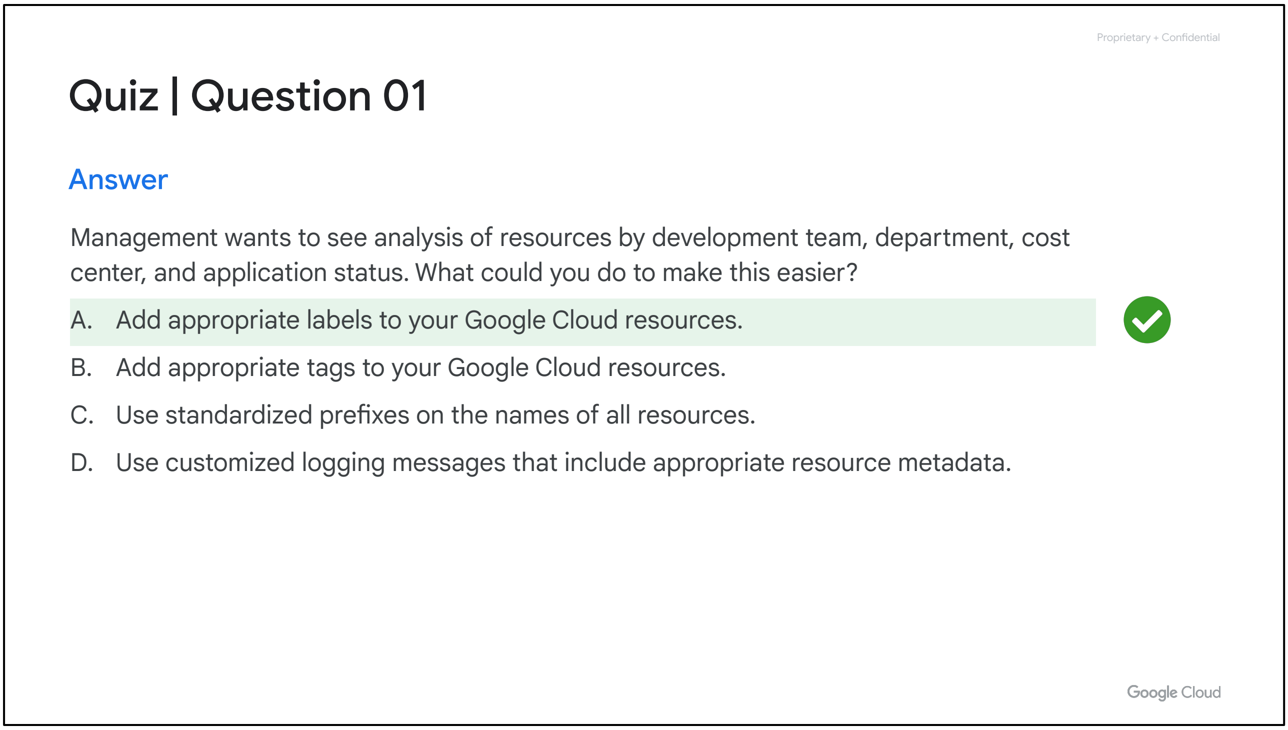
Management wants to see an analysis of resources divided by development team, department, cost center, and application status. What could you do to make this easier?

A. Add appropriate labels to your Google Cloud resources.

B. Add appropriate tags to your Google Cloud resources.

C. Use standardized prefixes on the names of all resources.

D. Use customized logging messages that include appropriate resource metadata.



Quiz | Question 01

Answer

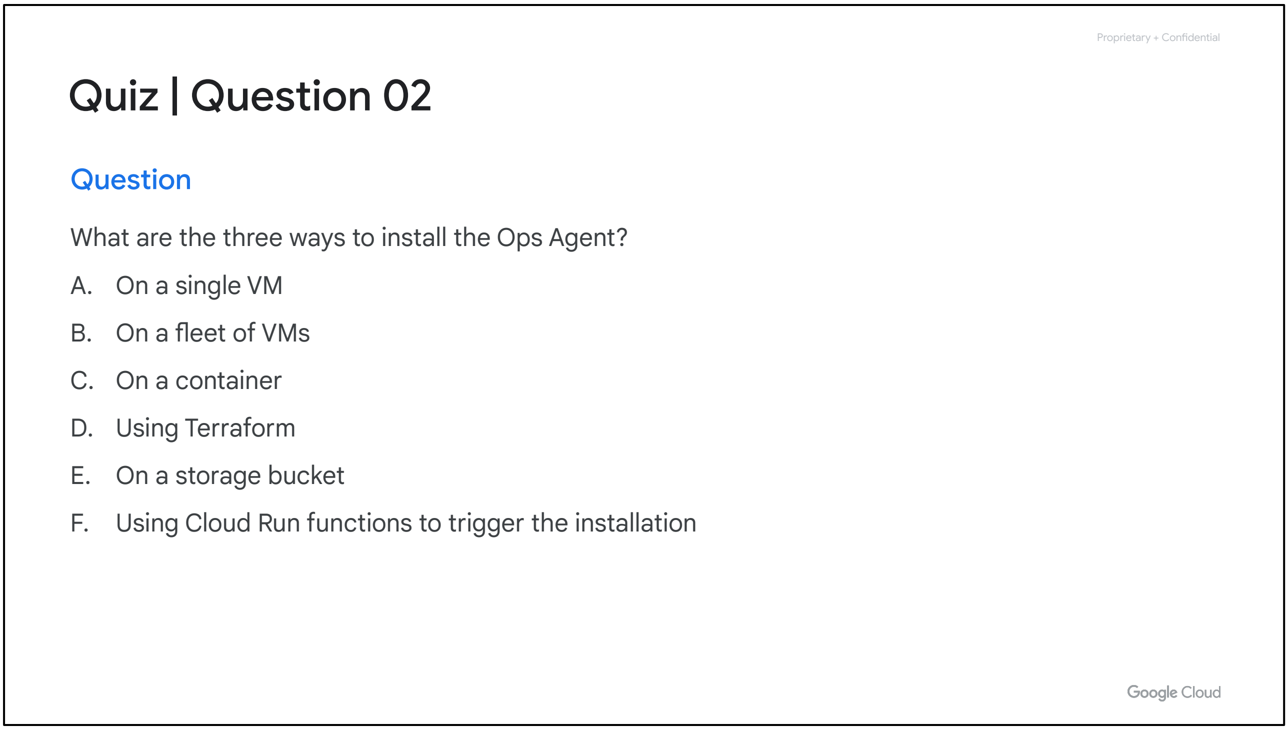
Management wants to see analysis of resources by development team, department, cost center, and application status. What could you do to make this easier?

A. Add appropriate labels to your Google Cloud resources. ✓

B. Add appropriate tags to your Google Cloud resources.

C. Use standardized prefixes on the names of all resources.

D. Use customized logging messages that include appropriate resource metadata.



Quiz | Question 02

Question

What are the three ways to install the Ops Agent?

A. On a single VM

B. On a fleet of VMs

C. On a container

D. Using Terraform

E. On a storage bucket

F. Using Cloud Run functions to trigger the installation

Quiz | Question 02

Answer

What are the three ways to install the Ops Agent?

A. On a single VM ✓

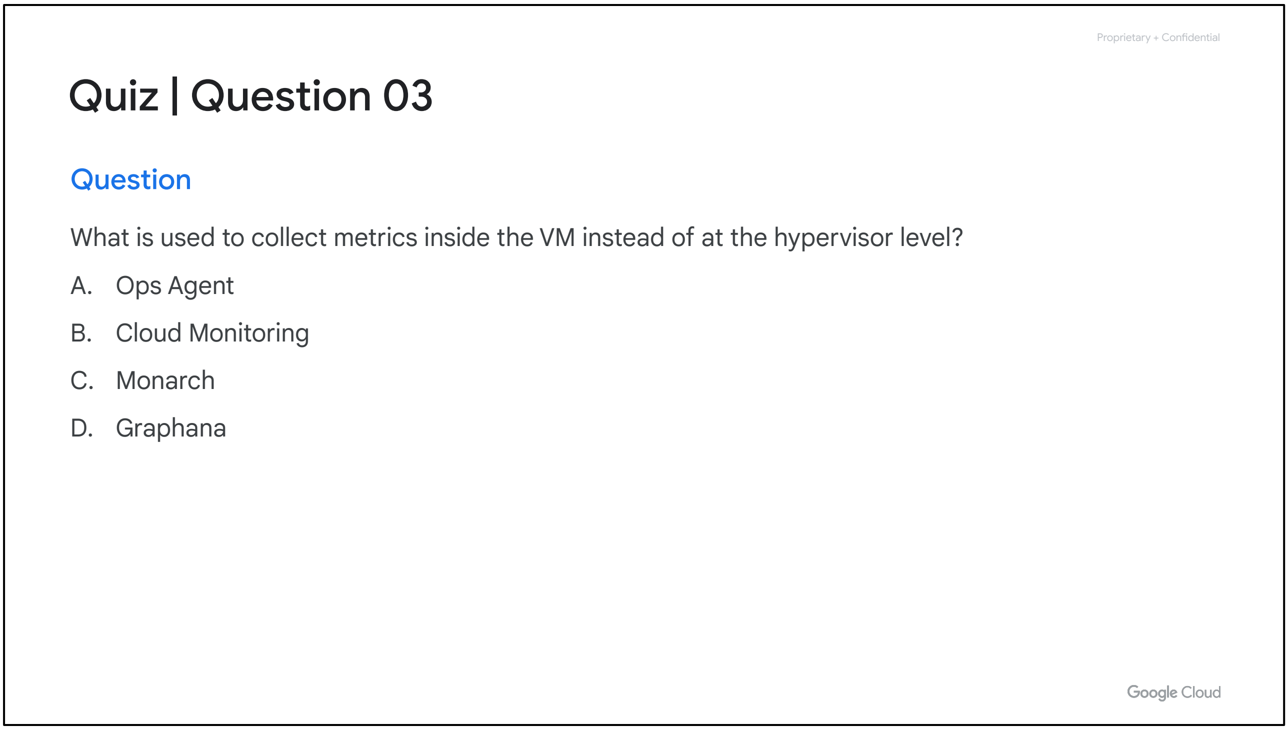
B. On a fleet of VMs ✓

C. On a container

D. Using Terraform ✓

E. On a storage bucket

F. Using Cloud Run functions to trigger the installation



Quiz | Question 03

Question

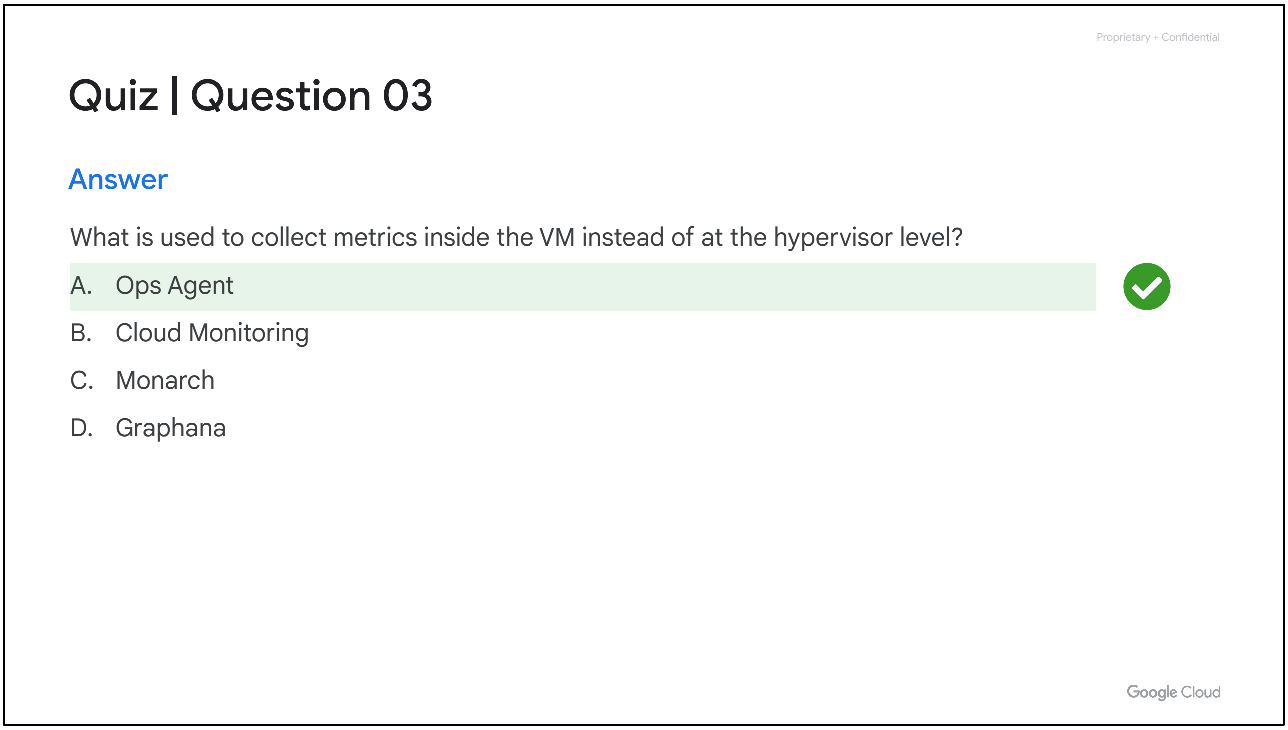
What is used to collect metrics inside the VM instead of at the hypervisor level?

A. Ops Agent

B. Cloud Monitoring

C. Monarch

D. Graphana



Quiz | Question 03

Answer

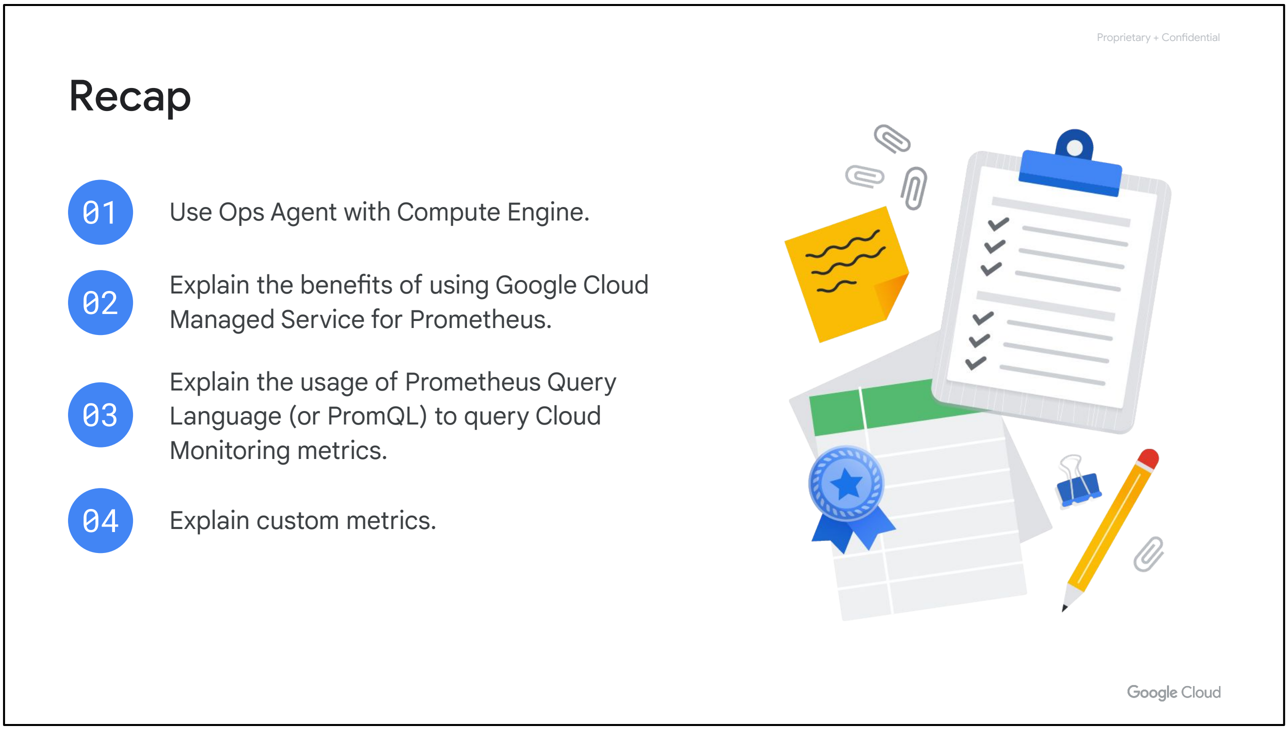
What is used to collect metrics inside the VM instead of at the hypervisor level?

A. Ops Agent ✓

B. Cloud Monitoring

C. Monarch

D. Graphana



Proprietary + Confidential

Recap

01 Use Ops Agent with Compute Engine.

02 Explain the benefits of using Google Cloud Managed Service for Prometheus.

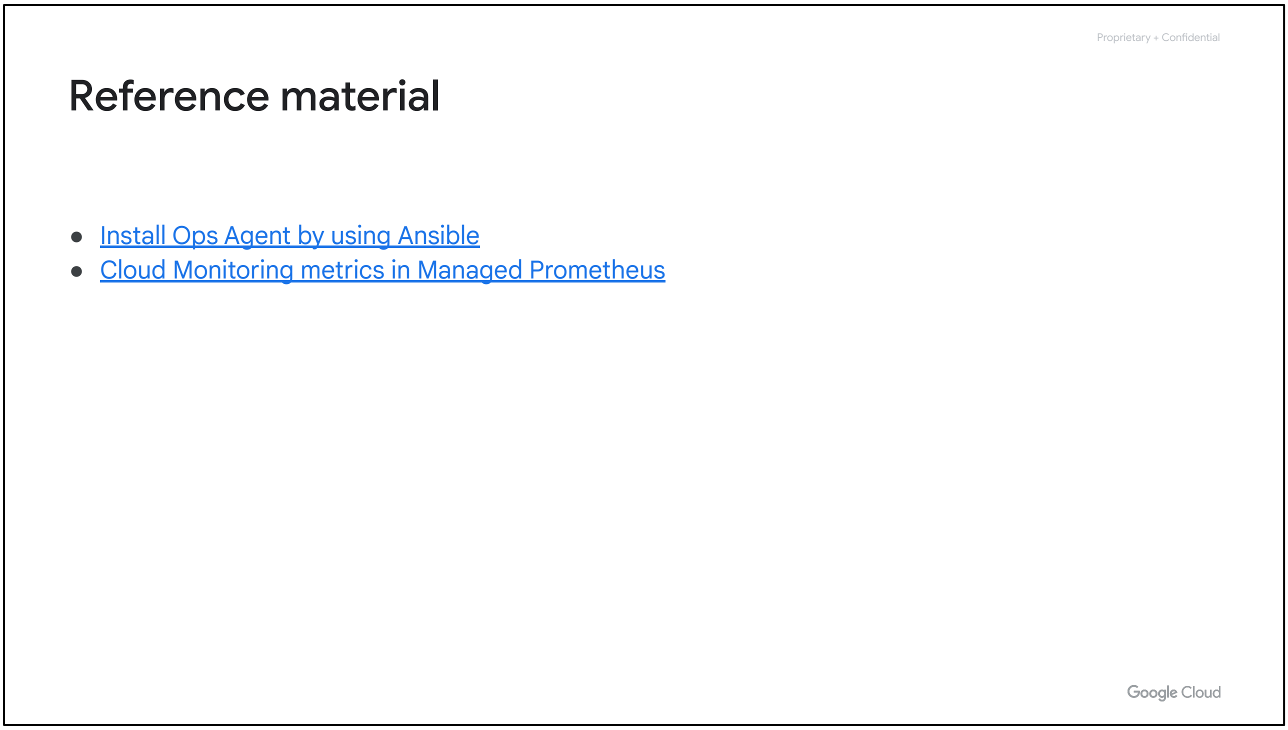
03 Explain the usage of Prometheus Query Language (or PromQL) to query Cloud Monitoring metrics.

04 Explain custom metrics.

Google Cloud

In this module, you learned how to:

- Use Ops Agent with Compute Engine.
- Explain the benefits of using Google Cloud Managed Service for Prometheus.
- Explain the usage of PromQL to query Cloud Monitoring metrics.
- Explain custom metrics.



Reference material

- Install Ops Agent by using Ansible
- Cloud Monitoring metrics in Managed Prometheus