# Configuring Google Cloud Services for Observability

## Agenda

Working with Agents

- Monitoring
- Logging
- Images and Agent Policies

Non-VM Resources

Exposing Custom Metrics

## Agenda

Working with Agents

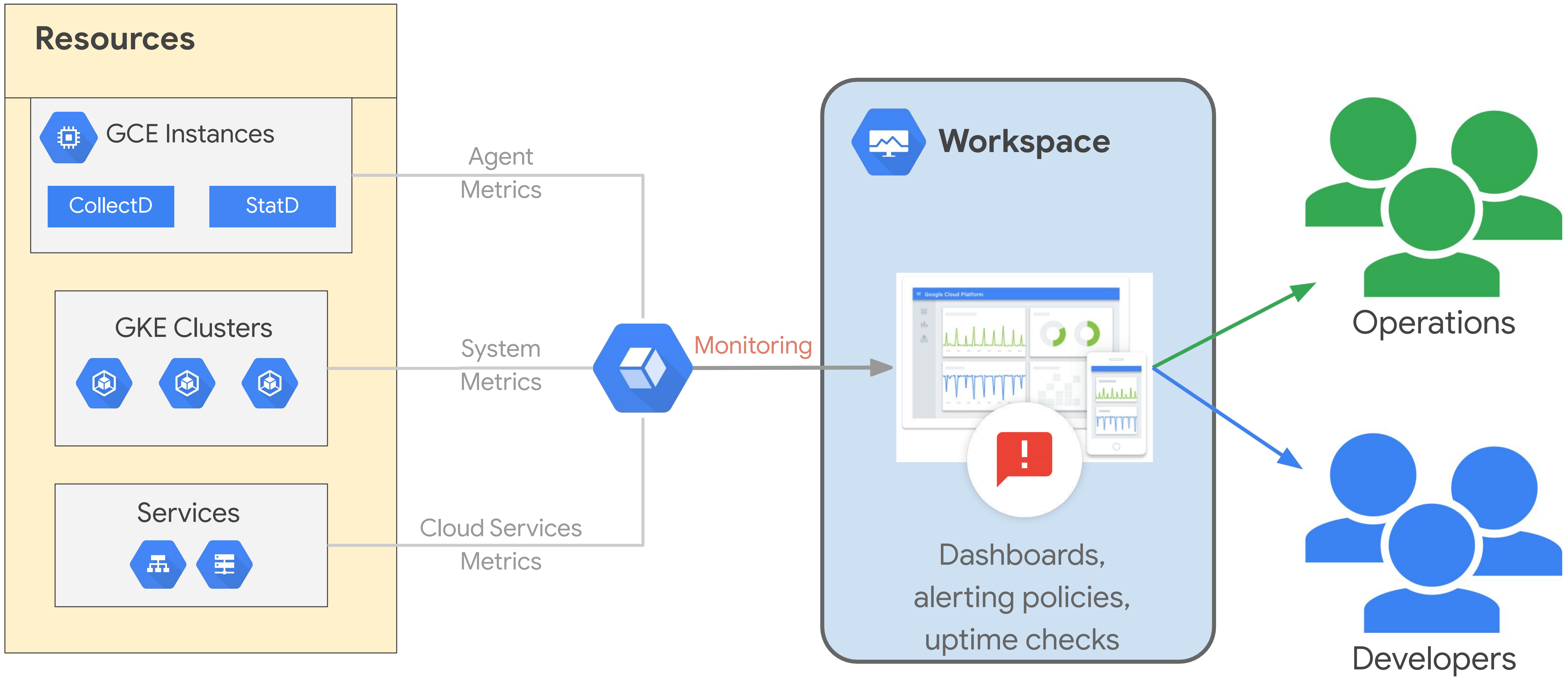
- Monitoring
- Logging
- Images and Agent Policies

Non-VM Resources

Exposing Custom Metrics

Google Cloud

# Monitoring Workspace



## OS Monitoring agent

**Gathers system and application metrics from VM instances and sends them to Monitoring**

- Based on the open-source collectd
- Gathers additional system resources and application metrics
- Optional, but recommended
- Supports third-party applications, such as:
    - Apache/Nginx/MySQL
- Additional support offered through BindPlane from Blue Medora
- Supports major operating systems:
    - CentOS, Debian, Red Hat Enterprise Linux
    - Ubuntu LTS, SUSE Linux Enterprise Server
    - Windows server

## Services with “other” Monitoring support

Don’t try to manually install or configure the agent

- App Engine standard has monitoring built-in
- App Engine flex has agent pre-installed and configured
- GKE nodes has monitoring configurable and enabled by default
- Anthos GKE On-Prem agent collects system but not application metrics
- Cloud Run provides integrated monitoring support
- Cloud Function supports integrated monitoring

## Installing the Monitoring agent

SUSE

```
curl -sSO
https://dl.google.com/cloudagents/add-monitoring-agent-repo.sh  sudo
bash add-monitoring-agent-repo.sh
sudo zypper install
stackdriver-agent  sudo service
stackdriver-agent start
```

Debian 10

```
curl -sSO
https://dl.google.com/cloudagents/add-monitoring-agent-repo.sh  sudo
bash add-monitoring-agent-repo.sh
sudo apt-get update
sudo apt-get install
stackdriver-agent  sudo service
stackdriver-agent start
```

## Installing the Monitoring agent

Centos 8

```
curl -sSO
https://dl.google.com/cloudagents/add-monitoring-agent-repo.sh  sudo
bash add-monitoring-agent-repo.sh
sudo yum install -y
stackdriver-agent  sudo service
stackdriver-agent start
```

Other

- All other Linux distros, see here
- Windows, see here

Google Cloud

## Verifying Monitoring agent authorization

Execute on the VM

```
curl --silent --connect-timeout 1 -f -H "Metadata-Flavor: Google" \
https://169.254.169.254/computeMetadata/v1/instance/service-accounts/default/scopes
```

Check for one or more of the following

```
https://www.googleapis.com/auth/monitoring.admin
https://www.googleapis.com/auth/monitoring.write
https://www.googleapis.com/auth/cloud-platform
```

## Agenda

## OS Logging agent

**Streams logs from common third-party applications and system software to Google Cloud Logging**

- Supports third-party applications, such as:
  - Apache/Tomcat/Nginx
  - Chef/Jenkins/Puppet
  - Cassandra/Mongodb/MySQL
- Based on fluentd log data collector—can add own fluentd configuration files
- Supports major operating systems:
  - CentOS
  - Debian
  - Red Hat Enterprise Linux
  - Ubuntu LTS
  - SUSE
  - Windows Server

## Services with “other” Logging support

Don’t try to manually install or configure the agent

- App Engine flex and standard have built-in support for logging
- GKE nodes can enable GKE logging
- Anthos GKE On-Prem agent collects system but not app logs
- Cloud Run has built-in logging support
- Cloud Functions have built-in logging support

## Installing the Logging agent

Linux

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

Windows (PowerShell terminal)

```
cd $env:UserProfile;
(New-Object Net.WebClient).DownloadFile(
    "https://dl.google.com/cloudagents/windows/StackdriverLogging-v1-10.exe",
    ".\StackdriverLogging-v1-10.exe")
.\StackdriverLogging-v1-10.exe
```

## Installing the Ops agent

### Linux

```
curl -sSO
https://dl.google.com/cloudagents/add-google-cloud-ops-agent-repo.sh sudo
bash add-google-cloud-ops-agent-repo.sh --also-install
```

### Windows (PowerShell terminal)

```
(New-Object
Net.WebClient).DownloadFile("https://dl.google.com/cloudagents/add-google-cloud-ops-agent-repo.ps1",
"${env:UserProfile}\add-google-cloud-ops-agent-repo.ps1") Invoke-Expression
"${env:UserProfile}\add-google-cloud-ops-agent-repo.ps1 -AlsoInstall"
```

The Ops Agent is the primary agent for collecting telemetry 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 OpenTelemetry Collector for metrics.

## Verifying Logging agent authorization

Execute on the VM

```
curl --silent --connect-timeout 1 -f -H "Metadata-Flavor: Google" \
https://169.254.169.254/computeMetadata/v1/instance/service-accounts/default/scopes
```

Check for one or more of the following

```
https://www.googleapis.com/auth/logging.write
https://www.googleapis.com/auth/logging.admin
https://www.googleapis.com/auth/cloud-platform
```

Google Cloud

## Agenda

Working with Agents

- Monitoring
- Logging

Images and Agent Policies

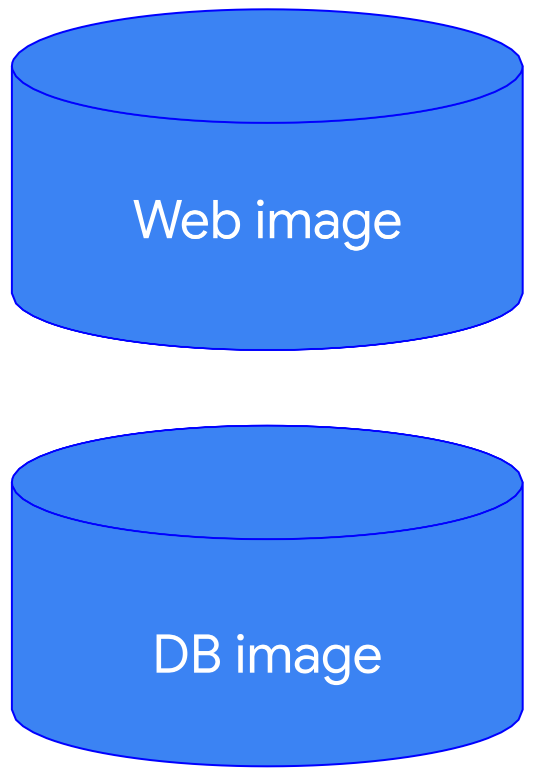
Non-VM Resources

Exposing Custom Metrics

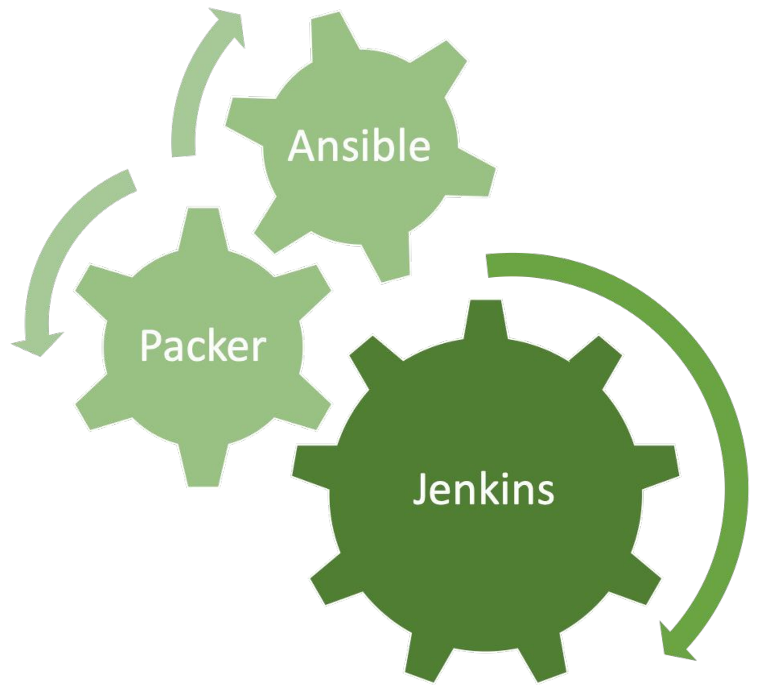
# Organizational maturity

What's an image?

Core image library

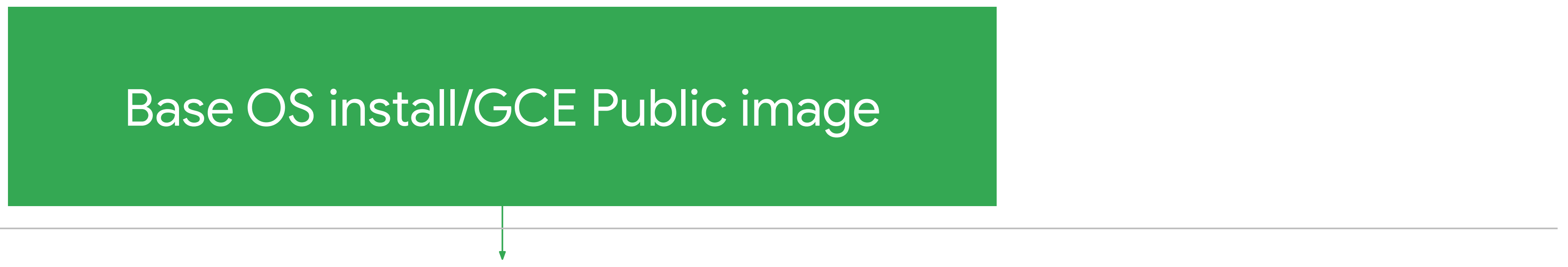


Robust image factory

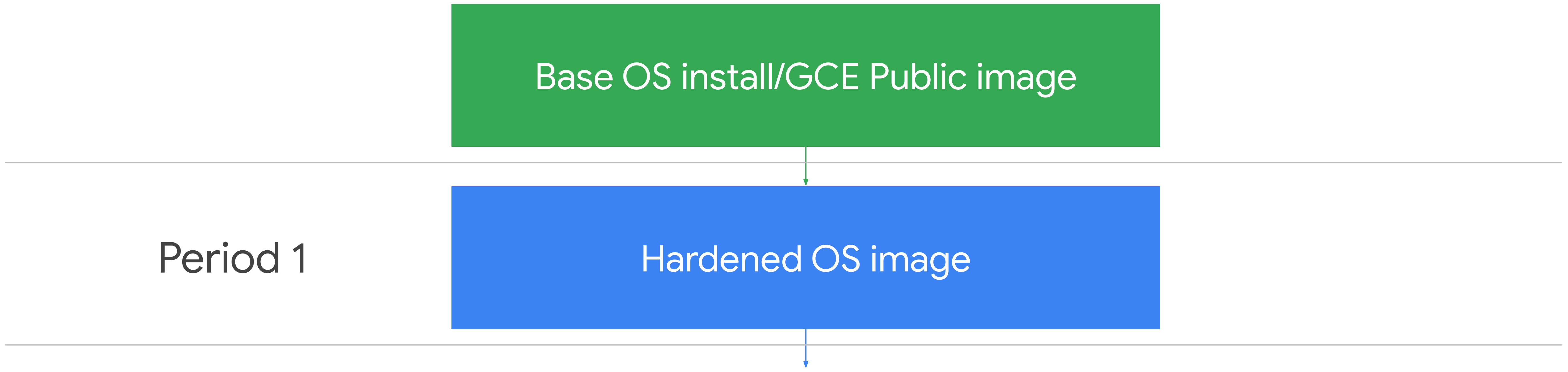


Google Cloud

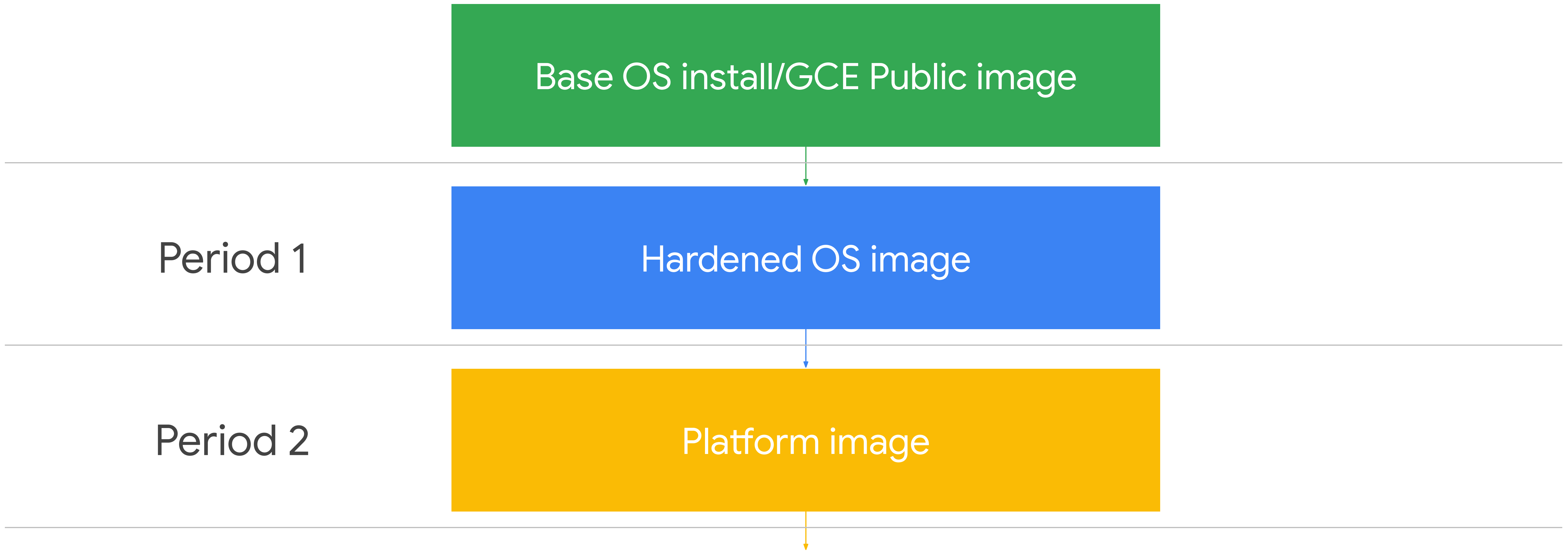
# Basic image management scheme



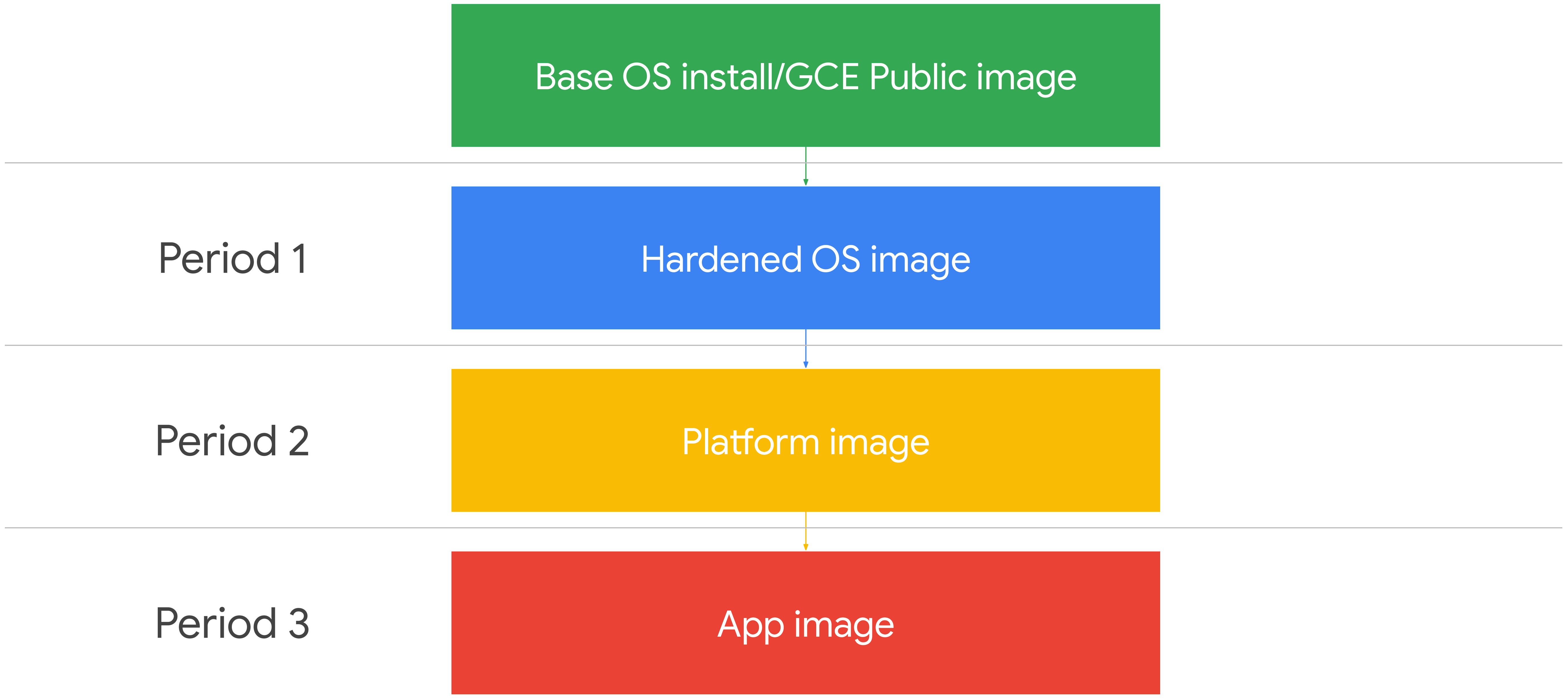
## Basic image management scheme



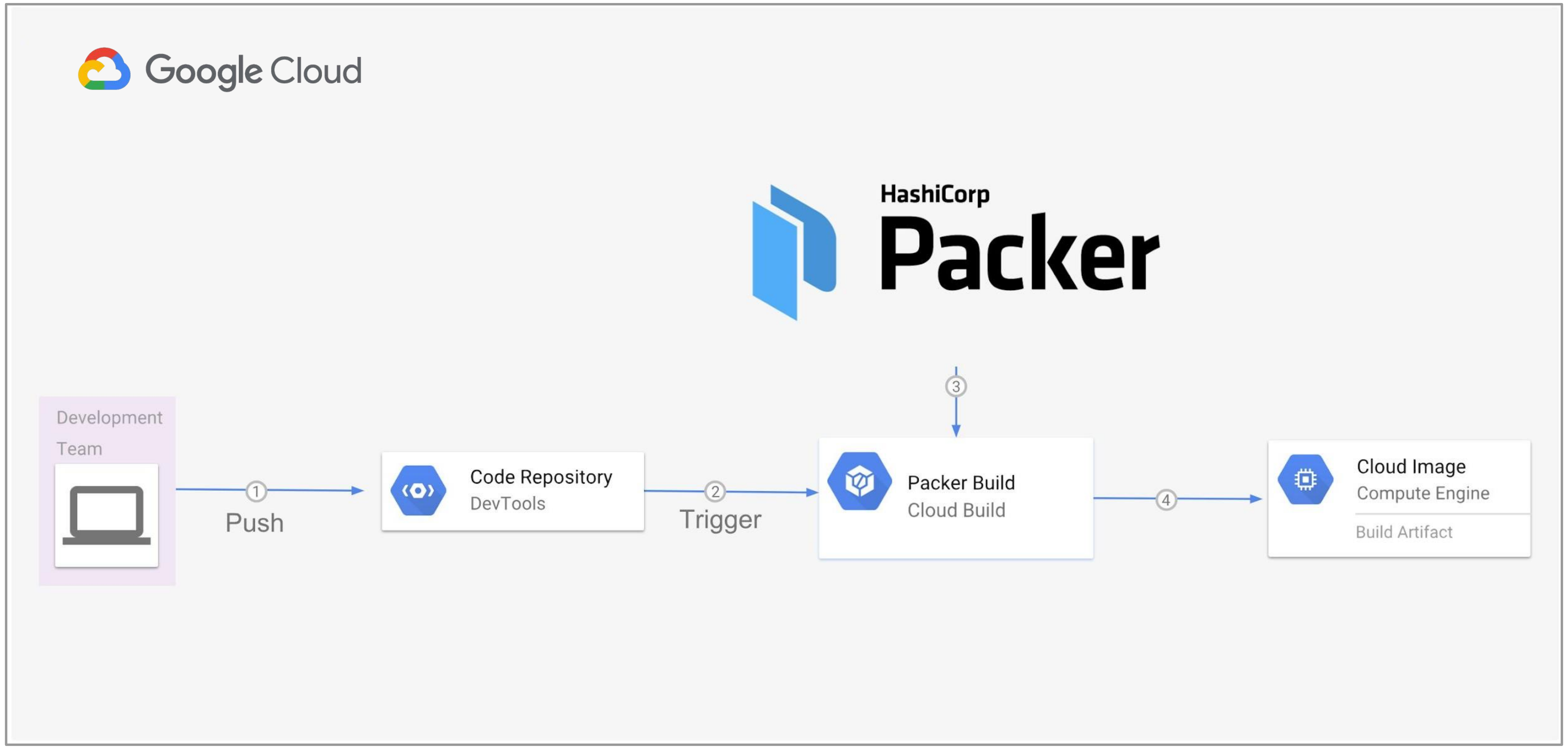
## Basic image management scheme



## Basic image management scheme



# Packer can automate image builds



# Leverage Agent Policies to aid with agent automation

- Automate installation and maintenance of Monitoring and Logging agents
- May apply to a fleet of VMs matching user-specified criteria
  - Current support for Linux VMs which support the agents

Here’s an example policy that targets all CentOS 7 VMs with the labels *env=test* and *app=myproduct*

```
gcloud alpha compute instances ops-agents policies create
ops-agents-policy-safe-rollout \
--agent-rules=
"type=logging,version=current-major,package-state=installed,enable-autoupgrade=
true; \  type=metrics,version=current-major,
package-state=installed,enable-autoupgrade=true" \
--os-types=short-name=centos,version=7 --group-labels=env=test,app=myproduct
```

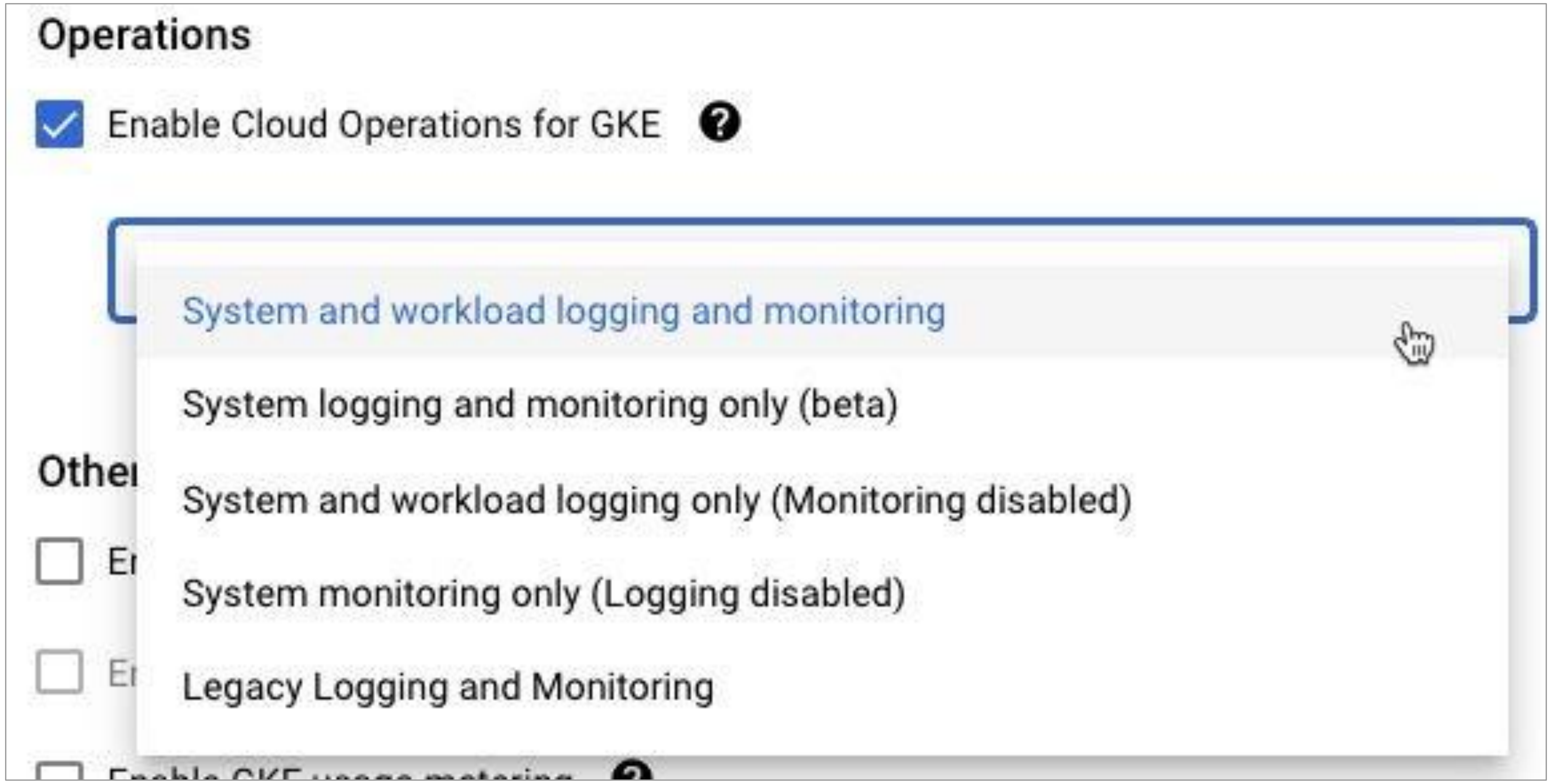
Google Cloud

## Agenda

# App Engine

- Standard and Flex support Monitoring
  - Check documentation for metric details
- Standard and Flex support Logging
  - Write to stdout or stderr from code
  - May also use Logging APIs (like Winston on Node.js)
- Logs viewable under GAE Application resource

## Google Kubernetes Engine


Operations

[✓] Enable Cloud Operations for GKE

System and workload logging and monitoring

System logging and monitoring only (beta)

System and workload logging only (Monitoring disabled)

System monitoring only (Logging disabled)

Legacy Logging and Monitoring

Othe

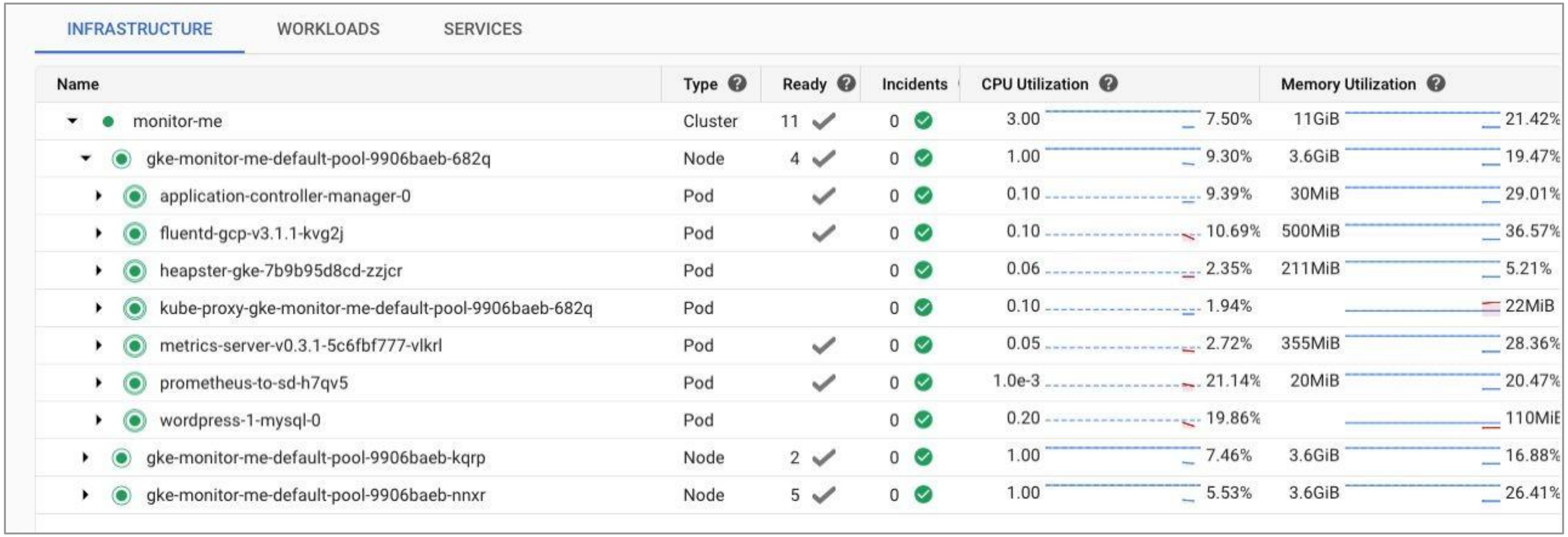
Er

Er

Enable GKE usage metering

Google Cloud

# GKE monitoring with the default dashboard

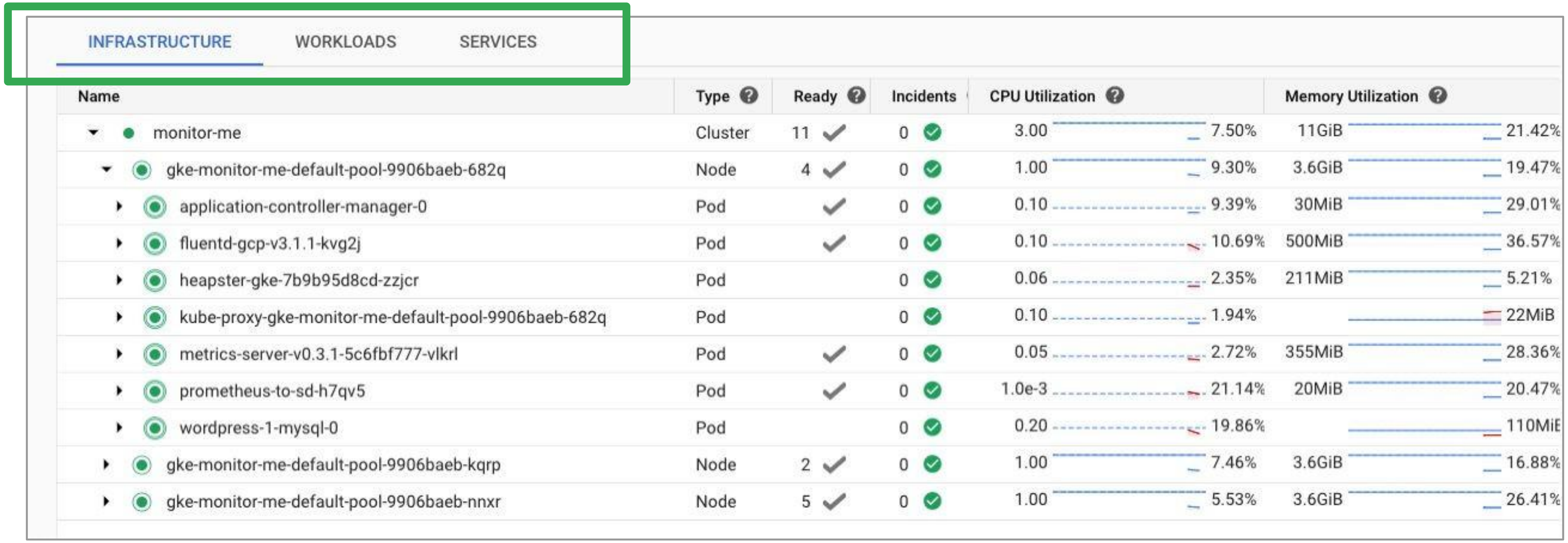


INFRASTRUCTURE WORKLOADS SERVICES

| Name | Type | Ready | Incidents | CPU Utilization | | Memory Utilization | |
|---|---|---|---|---|---|---|---|
| monitor-me | Cluster | 11 ✓ | 0 ✓ | 3.00 | 7.50% | 11GiB | 21.42% |
| gke-monitor-me-default-pool-9906baeb-682q | Node | 4 ✓ | 0 ✓ | 1.00 | 9.30% | 3.6GiB | 19.47% |
| application-controller-manager-0 | Pod | ✓ | 0 ✓ | 0.10 | 9.39% | 30MiB | 29.01% |
| fluentd-gcp-v3.1.1-kvg2j | Pod | ✓ | 0 ✓ | 0.10 | 10.69% | 500MiB | 36.57% |
| heapster-gke-7b9b95d8cd-zzjcr | Pod | | 0 ✓ | 0.06 | 2.35% | 211MiB | 5.21% |
| kube-proxy-gke-monitor-me-default-pool-9906baeb-682q | Pod | | 0 ✓ | 0.10 | 1.94% | | 22MiB |
| metrics-server-v0.3.1-5c6fbf777-vlkrl | Pod | ✓ | 0 ✓ | 0.05 | 2.72% | 355MiB | 28.36% |
| prometheus-to-sd-h7qv5 | Pod | ✓ | 0 ✓ | 1.0e-3 | 21.14% | 20MiB | 20.47% |
| wordpress-1-mysql-0 | Pod | | 0 ✓ | 0.20 | 19.86% | | 110Mi |
| gke-monitor-me-default-pool-9906baeb-kqrp | Node | 2 ✓ | 0 ✓ | 1.00 | 7.46% | 3.6GiB | 16.88% |
| gke-monitor-me-default-pool-9906baeb-nnxr | Node | 5 ✓ | 0 ✓ | 1.00 | 5.53% | 3.6GiB | 26.41% |

Google Cloud

# View the cluster from three perspectives

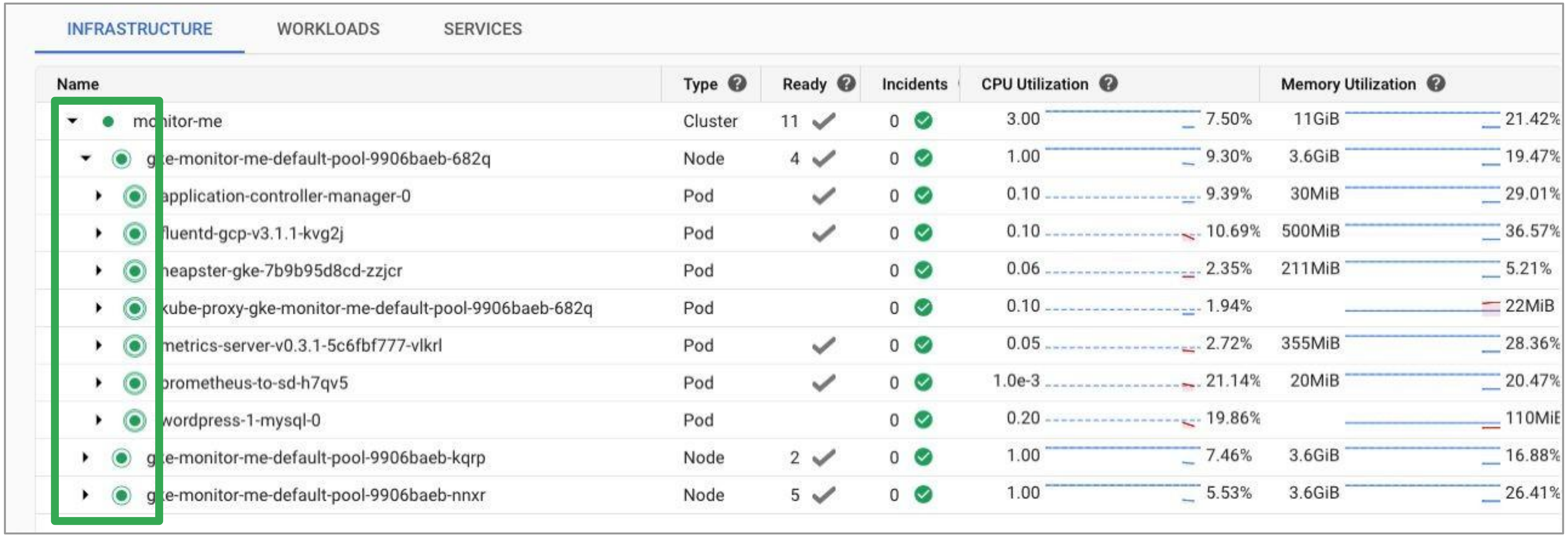


INFRASTRUCTURE WORKLOADS SERVICES

| Name | Type | Ready | Incidents | CPU Utilization | | Memory Utilization | |
|---|---|---|---|---|---|---|---|
| monitor-me | Cluster | 11 ✓ | 0 ✓ | 3.00 | 7.50% | 11GiB | 21.42% |
| gke-monitor-me-default-pool-9906baeb-682q | Node | 4 ✓ | 0 ✓ | 1.00 | 9.30% | 3.6GiB | 19.47% |
| application-controller-manager-0 | Pod | ✓ | 0 ✓ | 0.10 | 9.39% | 30MiB | 29.01% |
| fluentd-gcp-v3.1.1-kvg2j | Pod | ✓ | 0 ✓ | 0.10 | 10.69% | 500MiB | 36.57% |
| heapster-gke-7b9b95d8cd-zzjcr | Pod | | 0 ✓ | 0.06 | 2.35% | 211MiB | 5.21% |
| kube-proxy-gke-monitor-me-default-pool-9906baeb-682q | Pod | | 0 ✓ | 0.10 | 1.94% | | 22MiB |
| metrics-server-v0.3.1-5c6fbf777-vlkrl | Pod | ✓ | 0 ✓ | 0.05 | 2.72% | 355MiB | 28.36% |
| prometheus-to-sd-h7qv5 | Pod | ✓ | 0 ✓ | 1.0e-3 | 21.14% | 20MiB | 20.47% |
| wordpress-1-mysql-0 | Pod | | 0 ✓ | 0.20 | 19.86% | | 110MiB |
| gke-monitor-me-default-pool-9906baeb-kqrp | Node | 2 ✓ | 0 ✓ | 1.00 | 7.46% | 3.6GiB | 16.88% |
| gke-monitor-me-default-pool-9906baeb-nnxr | Node | 5 ✓ | 0 ✓ | 1.00 | 5.53% | 3.6GiB | 26.41% |

Google Cloud

# Resources preceded by a status indicator

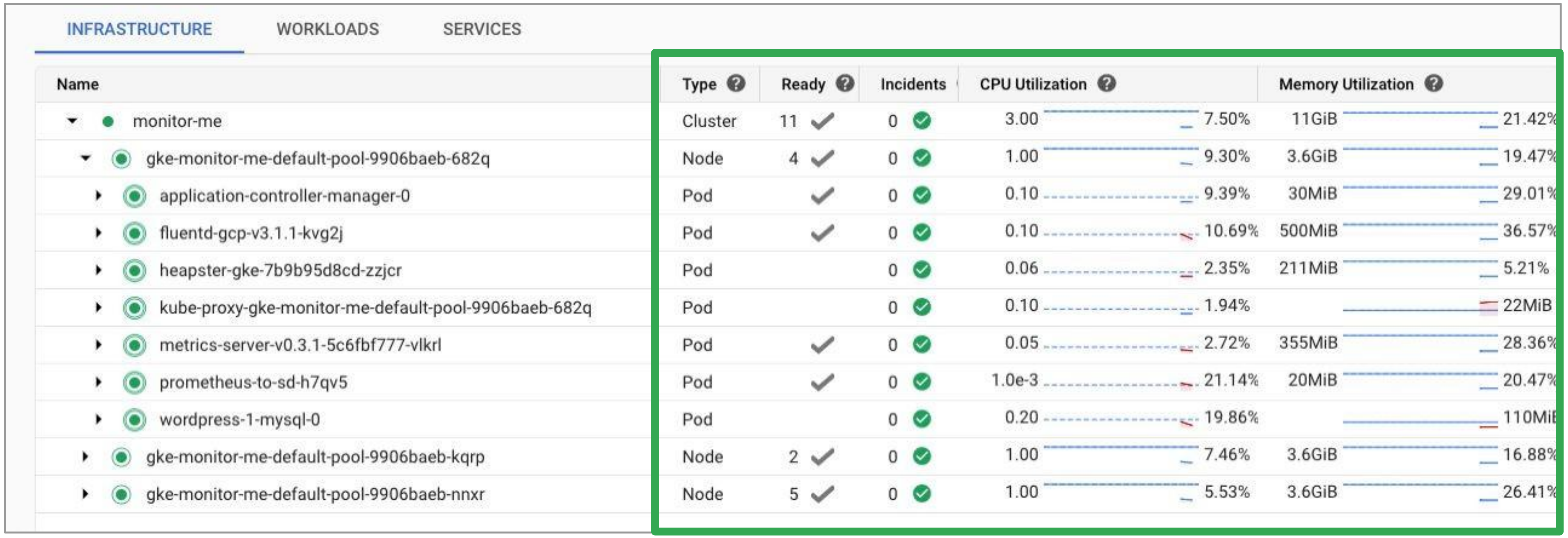


INFRASTRUCTURE | WORKLOADS | SERVICES

| Name | Type | Ready | Incidents | CPU Utilization | | Memory Utilization | |
|---|---|---|---|---|---|---|---|
| ● mc itor-me | Cluster | 11 ✓ | 0 ✓ | 3.00 | 7.50% | 11GiB | 21.42% |
| ● g e-monitor-me-default-pool-9906baeb-682q | Node | 4 ✓ | 0 ✓ | 1.00 | 9.30% | 3.6GiB | 19.47% |
| ● application-controller-manager-0 | Pod | ✓ | 0 ✓ | 0.10 | 9.39% | 30MiB | 29.01% |
| ● luentd-gcp-v3.1.1-kvg2j | Pod | ✓ | 0 ✓ | 0.10 | 10.69% | 500MiB | 36.57% |
| ● eapster-gke-7b9b95d8cd-zzjcr | Pod | | 0 ✓ | 0.06 | 2.35% | 211MiB | 5.21% |
| ● kube-proxy-gke-monitor-me-default-pool-9906baeb-682q | Pod | | 0 ✓ | 0.10 | 1.94% | | 22MiB |
| ● metrics-server-v0.3.1-5c6fbf777-vlkrl | Pod | ✓ | 0 ✓ | 0.05 | 2.72% | 355MiB | 28.36% |
| ● prometheus-to-sd-h7qv5 | Pod | ✓ | 0 ✓ | 1.0e-3 | 21.14% | 20MiB | 20.47% |
| ● vordpress-1-mysql-0 | Pod | | 0 ✓ | 0.20 | 19.86% | | 110Mi |
| ● g e-monitor-me-default-pool-9906baeb-kqrp | Node | 2 ✓ | 0 ✓ | 1.00 | 7.46% | 3.6GiB | 16.88% |
| ● g e-monitor-me-default-pool-9906baeb-nnxr | Node | 5 ✓ | 0 ✓ | 1.00 | 5.53% | 3.6GiB | 26.41% |

Google Cloud

# Status information provided for each GKE object

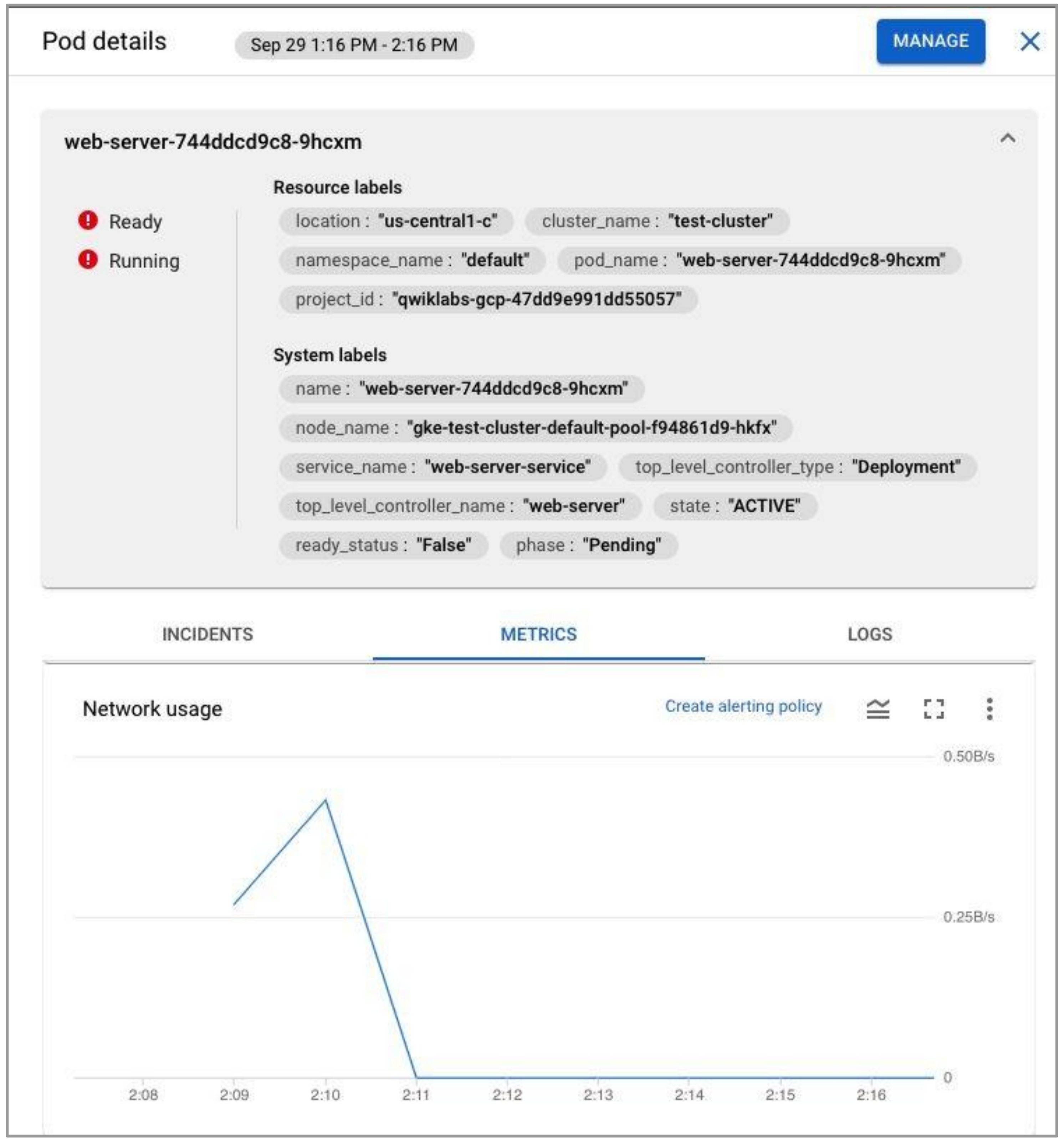


INFRASTRUCTURE WORKLOADS SERVICES

| Name | Type | Ready | Incidents | CPU Utilization | | Memory Utilization | |
|---|---|---|---|---|---|---|---|
| monitor-me | Cluster | 11 ✓ | 0 ✓ | 3.00 | 7.50% | 11GiB | 21.42% |
| gke-monitor-me-default-pool-9906baeb-682q | Node | 4 ✓ | 0 ✓ | 1.00 | 9.30% | 3.6GiB | 19.47% |
| application-controller-manager-0 | Pod | ✓ | 0 ✓ | 0.10 | 9.39% | 30MiB | 29.01% |
| fluentd-gcp-v3.1.1-kvg2j | Pod | ✓ | 0 ✓ | 0.10 | 10.69% | 500MiB | 36.57% |
| heapster-gke-7b9b95d8cd-zzjcr | Pod | | 0 ✓ | 0.06 | 2.35% | 211MiB | 5.21% |
| kube-proxy-gke-monitor-me-default-pool-9906baeb-682q | Pod | | 0 ✓ | 0.10 | 1.94% | | 22MiB |
| metrics-server-v0.3.1-5c6fbf777-vlkrl | Pod | ✓ | 0 ✓ | 0.05 | 2.72% | 355MiB | 28.36% |
| prometheus-to-sd-h7qv5 | Pod | ✓ | 0 ✓ | 1.0e-3 | 21.14% | 20MiB | 20.47% |
| wordpress-1-mysql-0 | Pod | | 0 ✓ | 0.20 | 19.86% | | 110MiB |
| gke-monitor-me-default-pool-9906baeb-kqrp | Node | 2 ✓ | 0 ✓ | 1.00 | 7.46% | 3.6GiB | 16.88% |
| gke-monitor-me-default-pool-9906baeb-nnxr | Node | 5 ✓ | 0 ✓ | 1.00 | 5.53% | 3.6GiB | 26.41% |

Google Cloud

## Select a pod to view details including metrics



Pod details Sep 29 1:16 PM - 2:16 PM MANAGE

web-server-744ddcd9c8-9hcxm

❗ Ready
❗ Running

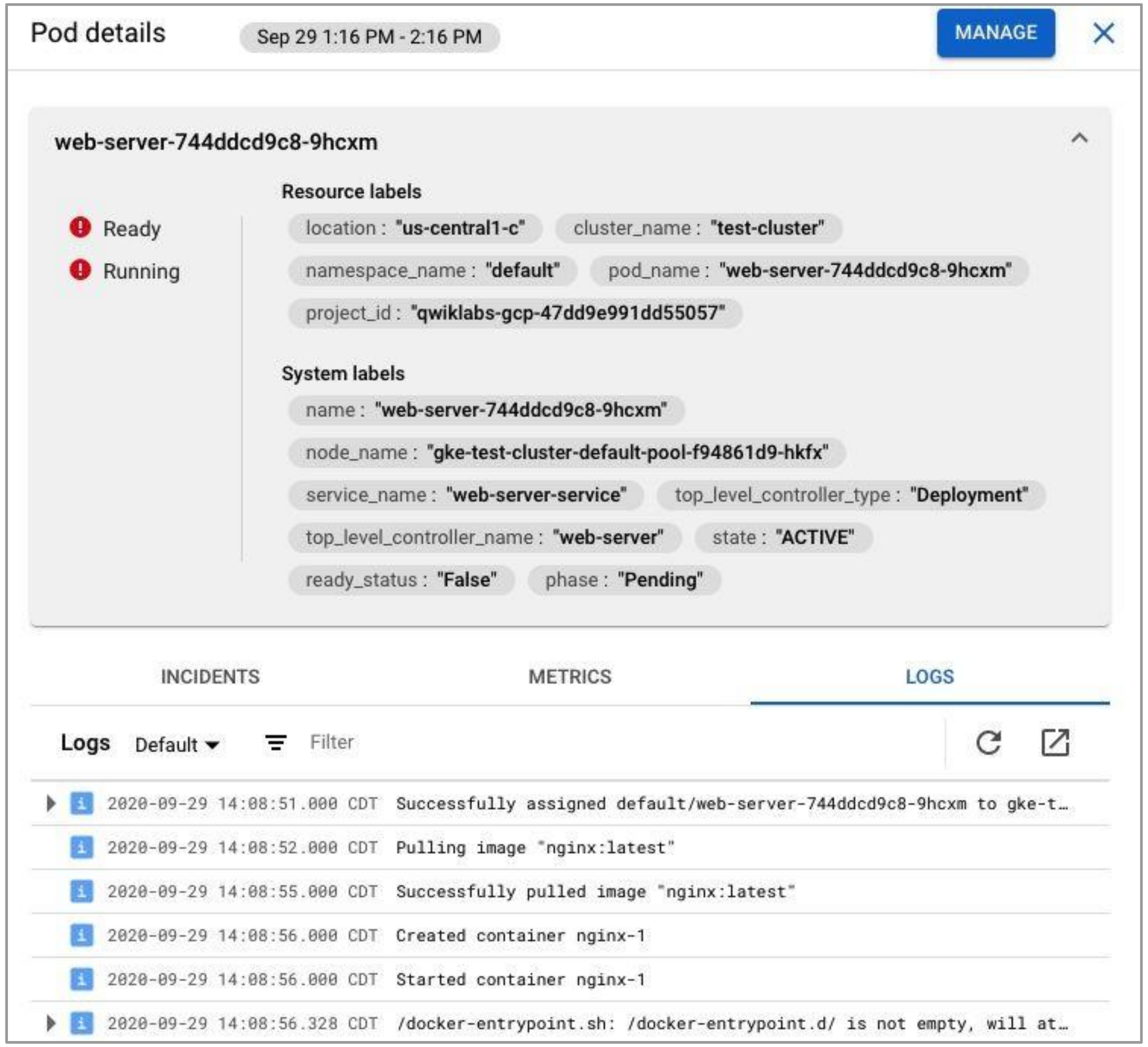
**Resource labels**
location : "us-central1-c"
cluster_name : "test-cluster"
namespace_name : "default"
pod_name : "web-server-744ddcd9c8-9hcxm"
project_id : "qwiklabs-gcp-47dd9e991dd55057"

**System labels**
name : "web-server-744ddcd9c8-9hcxm"
node_name : "gke-test-cluster-default-pool-f94861d9-hkfx"
service_name : "web-server-service"
top_level_controller_type : "Deployment"
top_level_controller_name : "web-server"
state : "ACTIVE"
ready_status : "False"
phase : "Pending"

INCIDENTS METRICS LOGS

Network usage Create alerting policy

# Logs tab displays the latest entries



Pod details   Sep 29 1:16 PM - 2:16 PM   MANAGE

web-server-744ddcd9c8-9hcxm

Ready
Running

Resource labels

location : "us-central1-c"
cluster_name : "test-cluster"
namespace_name : "default"
pod_name : "web-server-744ddcd9c8-9hcxm"
project_id : "qwiklabs-gcp-47dd9e991dd55057"

System labels

name : "web-server-744ddcd9c8-9hcxm"
node_name : "gke-test-cluster-default-pool-f94861d9-hkfx"
service_name : "web-server-service"
top_level_controller_type : "Deployment"
top_level_controller_name : "web-server"
state : "ACTIVE"
ready_status : "False"
phase : "Pending"

INCIDENTS   METRICS   LOGS

Logs   Default   Filter

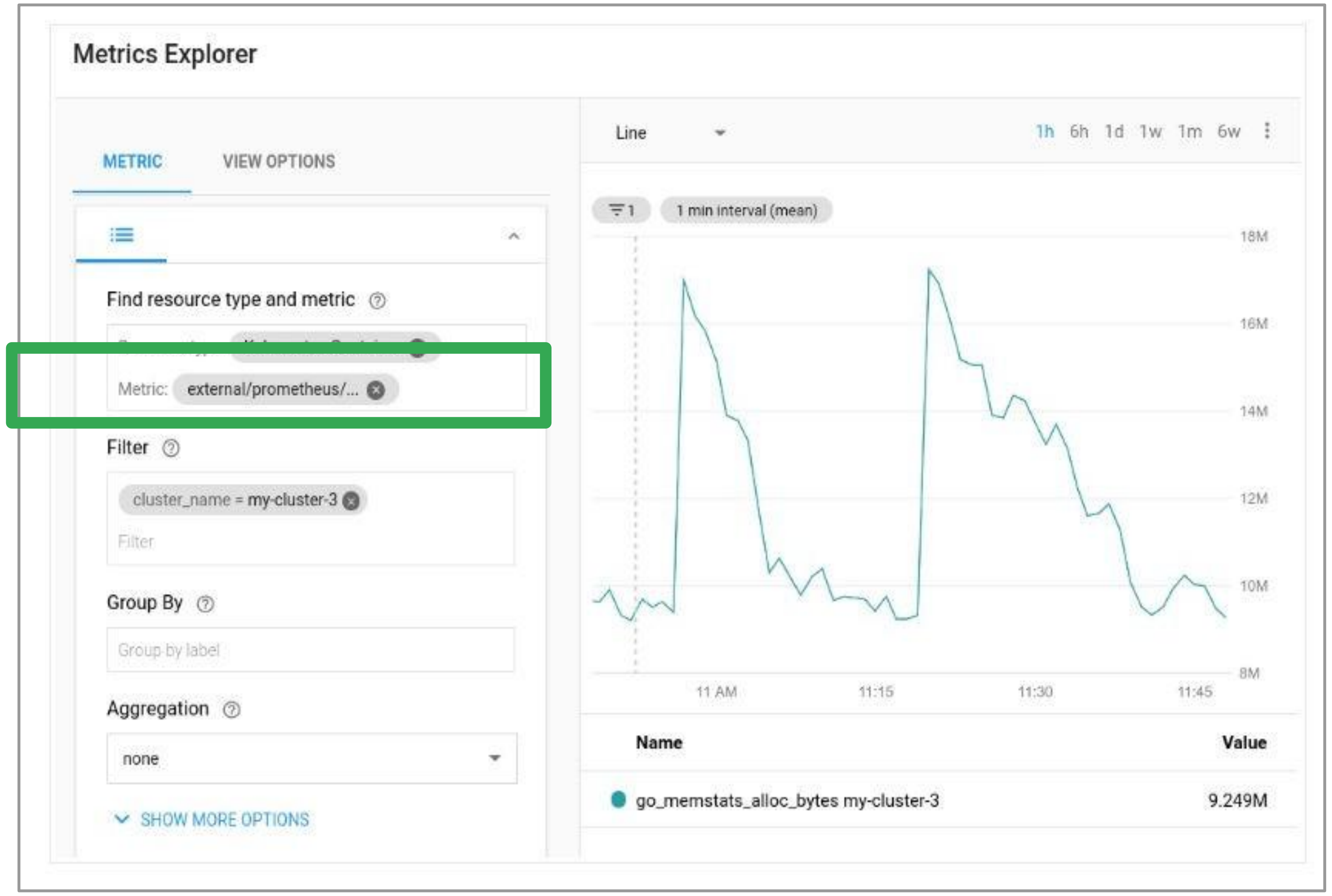
```
2020-09-29 14:08:51.000 CDT Successfully assigned default/web-server-744ddcd9c8-9hcxm to gke-t…
2020-09-29 14:08:52.000 CDT Pulling image "nginx:latest"
2020-09-29 14:08:55.000 CDT Successfully pulled image "nginx:latest"
2020-09-29 14:08:56.000 CDT Created container nginx-1
2020-09-29 14:08:56.000 CDT Started container nginx-1
2020-09-29 14:08:56.328 CDT /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will at…
```

## What is Prometheus?

- Prometheus is an optional monitoring tool for Kubernetes
  - Supported with GKE Monitoring
- Service metrics using Prometheus exposition format can be exported and  made visible as external metrics

## Configure Prometheus for GKE

- Install Prometheus and the Collector
- Metrics can be viewed as external metrics
  - *external/prometheus/**



Metrics Explorer

Metric: external/prometheus/...

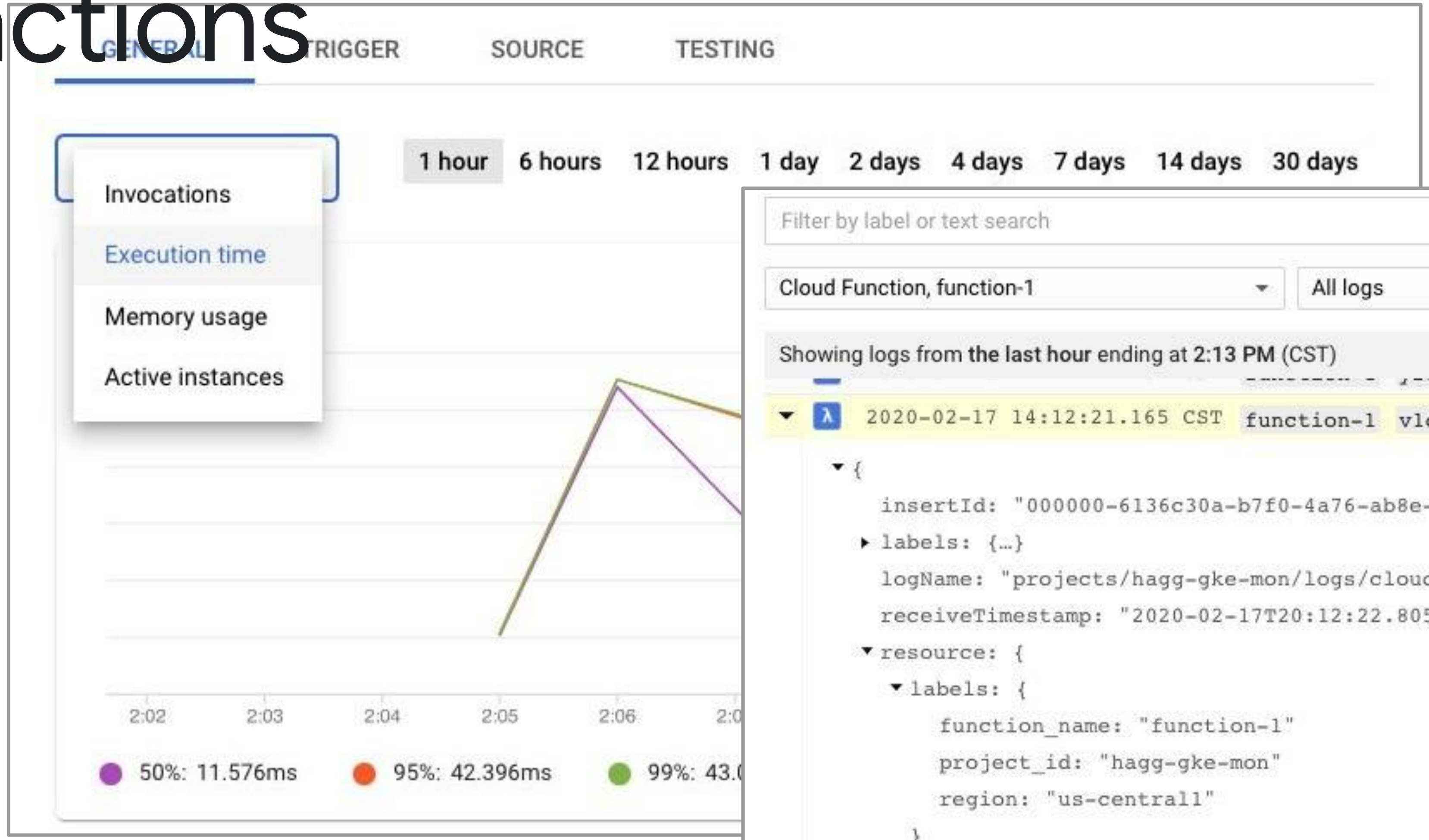
Filter: cluster_name = my-cluster-3

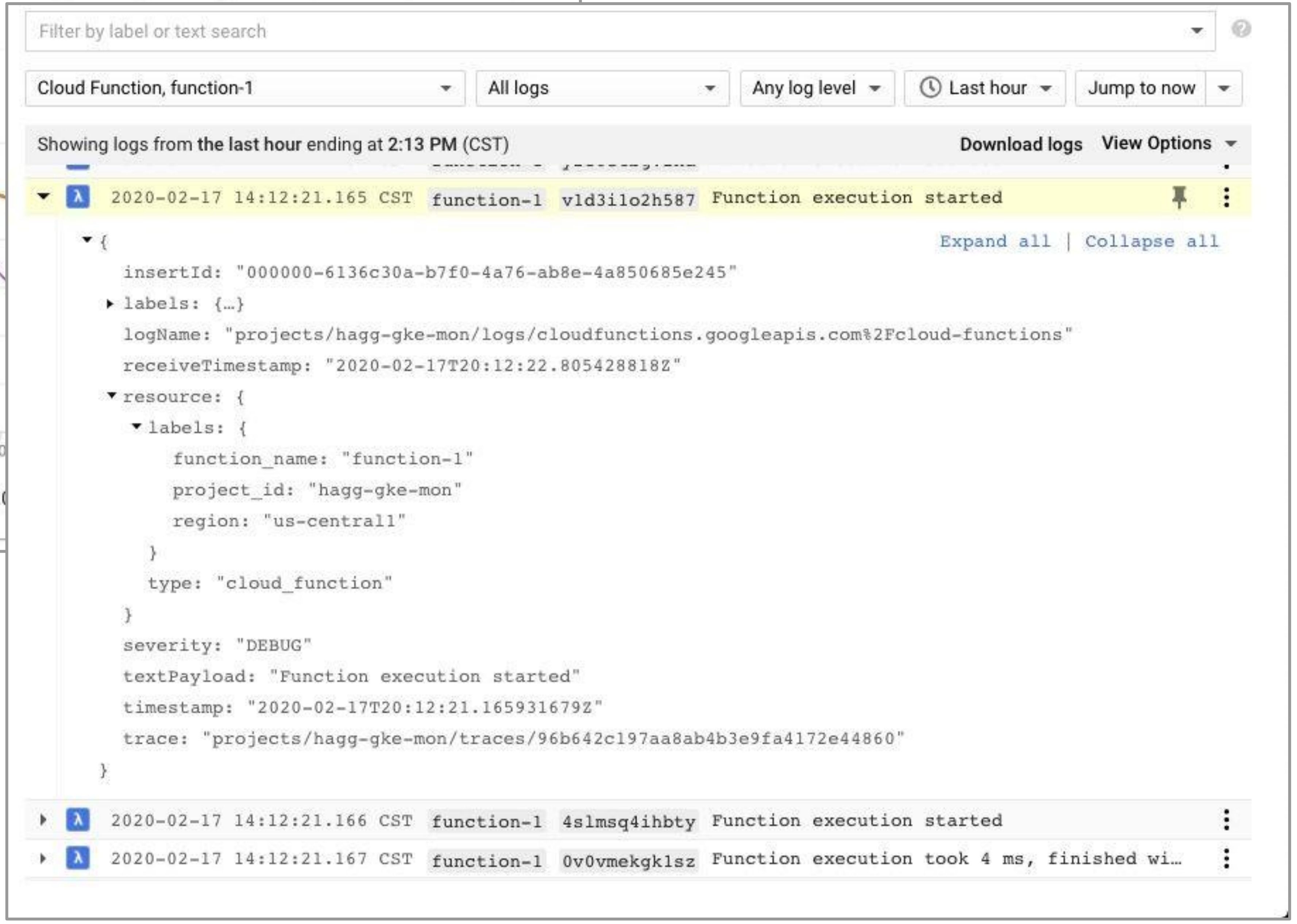
Group By: Group by label

Aggregation: none

| Name | Value |
|---|---|
| go_memstats_alloc_bytes my-cluster-3 | 9.249M |

# Cloud Functions



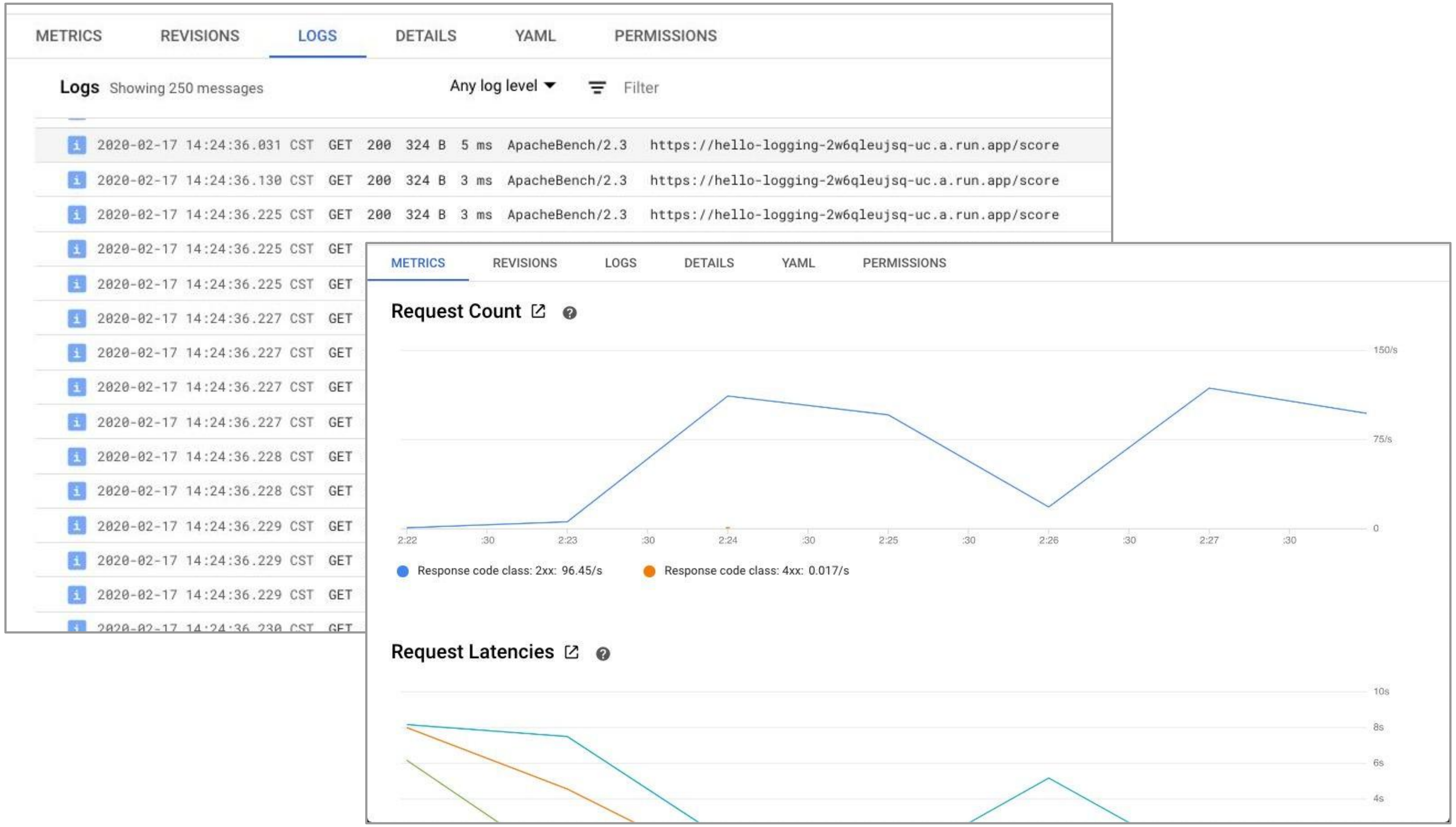

Filter by label or text search
Cloud Function, function-1
All logs
Any log level
Last hour
Jump to now
Download logs
View Options
Expand all | Collapse all

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

```
2020-02-17 14:12:21.165 CST  function-1  vld3ilo2h587  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



```
2020-02-17 14:24:36.031 CST  GET  200  324 B  5 ms  ApacheBench/2.3   https://hello-logging-2w6qleujsq-uc.a.run.app/score
2020-02-17 14:24:36.130 CST  GET  200  324 B  3 ms  ApacheBench/2.3   https://hello-logging-2w6qleujsq-uc.a.run.app/score
2020-02-17 14:24:36.225 CST  GET  200  324 B  3 ms  ApacheBench/2.3   https://hello-logging-2w6qleujsq-uc.a.run.app/score
2020-02-17 14:24:36.225 CST  GET
2020-02-17 14:24:36.225 CST  GET
2020-02-17 14:24:36.227 CST  GET
2020-02-17 14:24:36.227 CST  GET
2020-02-17 14:24:36.227 CST  GET
2020-02-17 14:24:36.227 CST  GET
2020-02-17 14:24:36.228 CST  GET
2020-02-17 14:24:36.228 CST  GET
2020-02-17 14:24:36.229 CST  GET
2020-02-17 14:24:36.229 CST  GET
2020-02-17 14:24:36.229 CST  GET
2020-02-17 14:24:36.230 CST  GET
```

## Agenda

Working with Agents

- Monitoring
- Logging
- Images and Agent Policies

Non-VM Resources

Exposing Custom Metrics

Google Cloud

## Exposing custom metrics

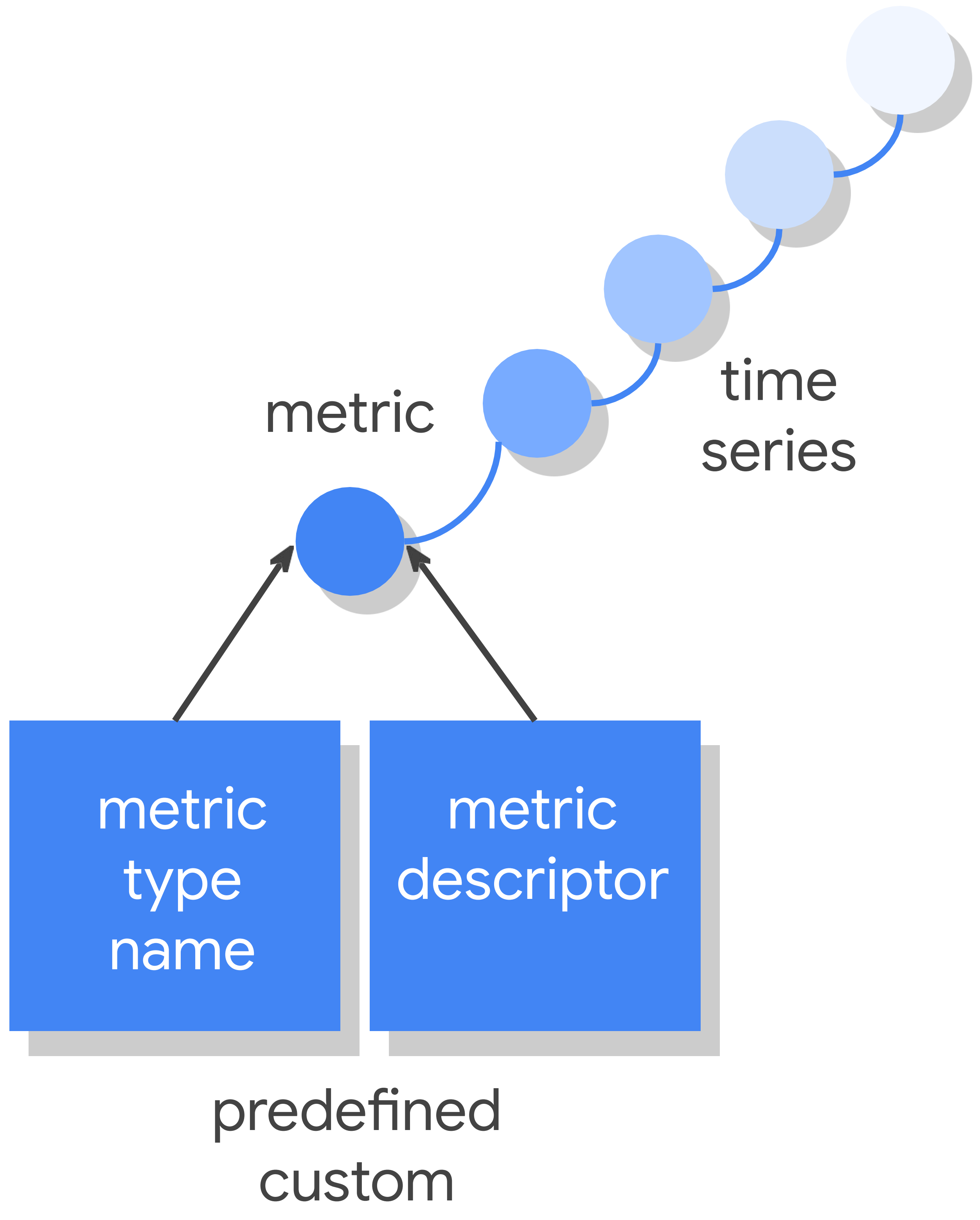
Two fundamental approaches:

- Use the Cloud Monitoring API
- Use OpenCensus

## Custom metrics

Custom metric descriptor example in Python:

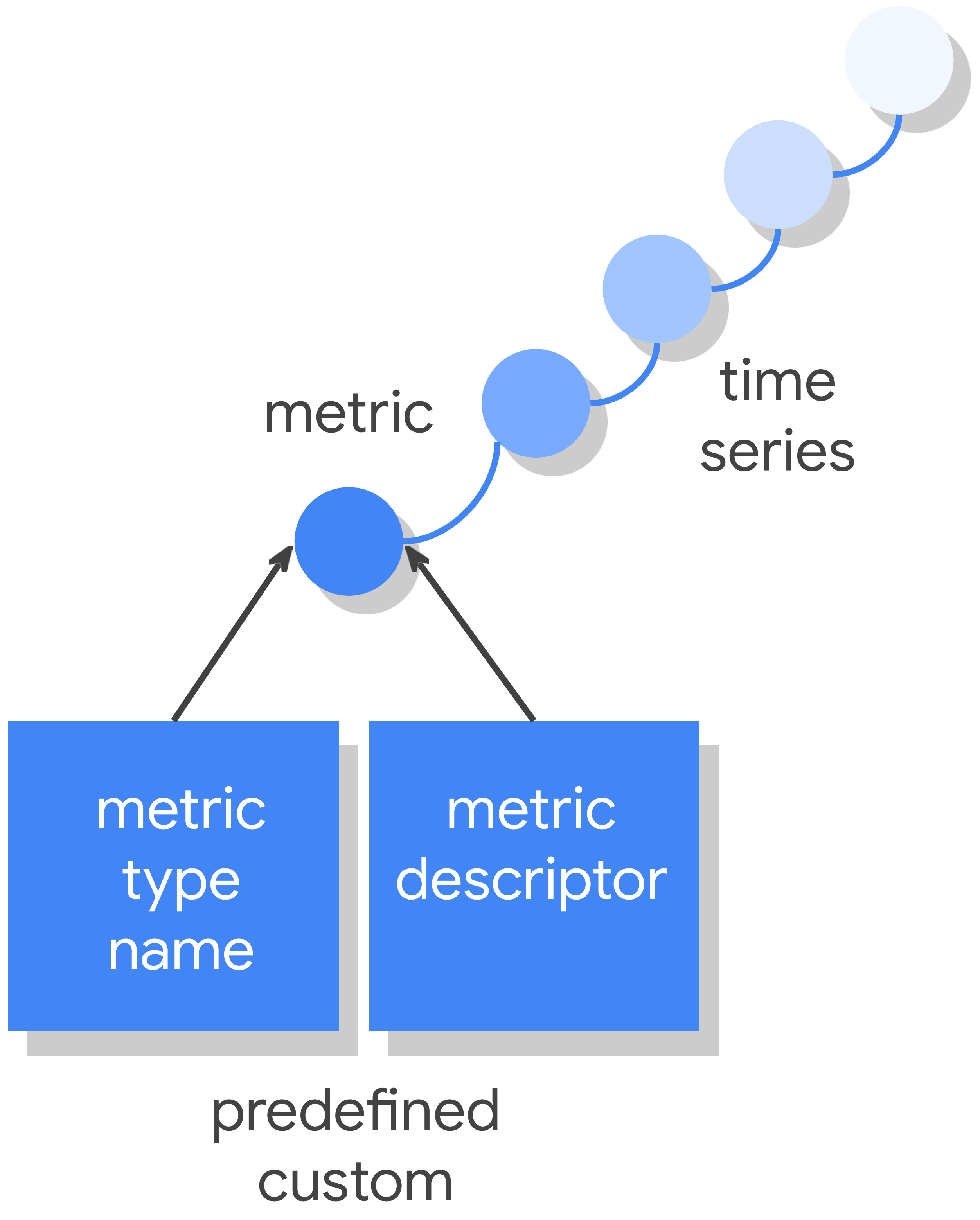
```
client = monitoring_v3.MetricServiceClient()
project_name =
client.project_path(project_id)
descriptor = monitoring_v3.types.MetricDescriptor()

descriptor.type =

('custom.googleapis.com/my_metric')

descriptor.metric_kind = (

  monitoring_v3.enums.MetricDescriptor.MetricKind.GAUGE)

descriptor.value_type = (
  monitoring_v3.enums.MetricDescriptor.ValueType.DOUBL
  E)

descriptor.description = 'Custom metric example.'


client.create_metric_descriptor(project_name, descriptor)
```



## Custom metrics

Custom metric descriptor example in Python:

```
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.DOUBL
  E)

descriptor.description = 'Custom metric example.'
client.create_metric_descriptor(project_name, descriptor)
```

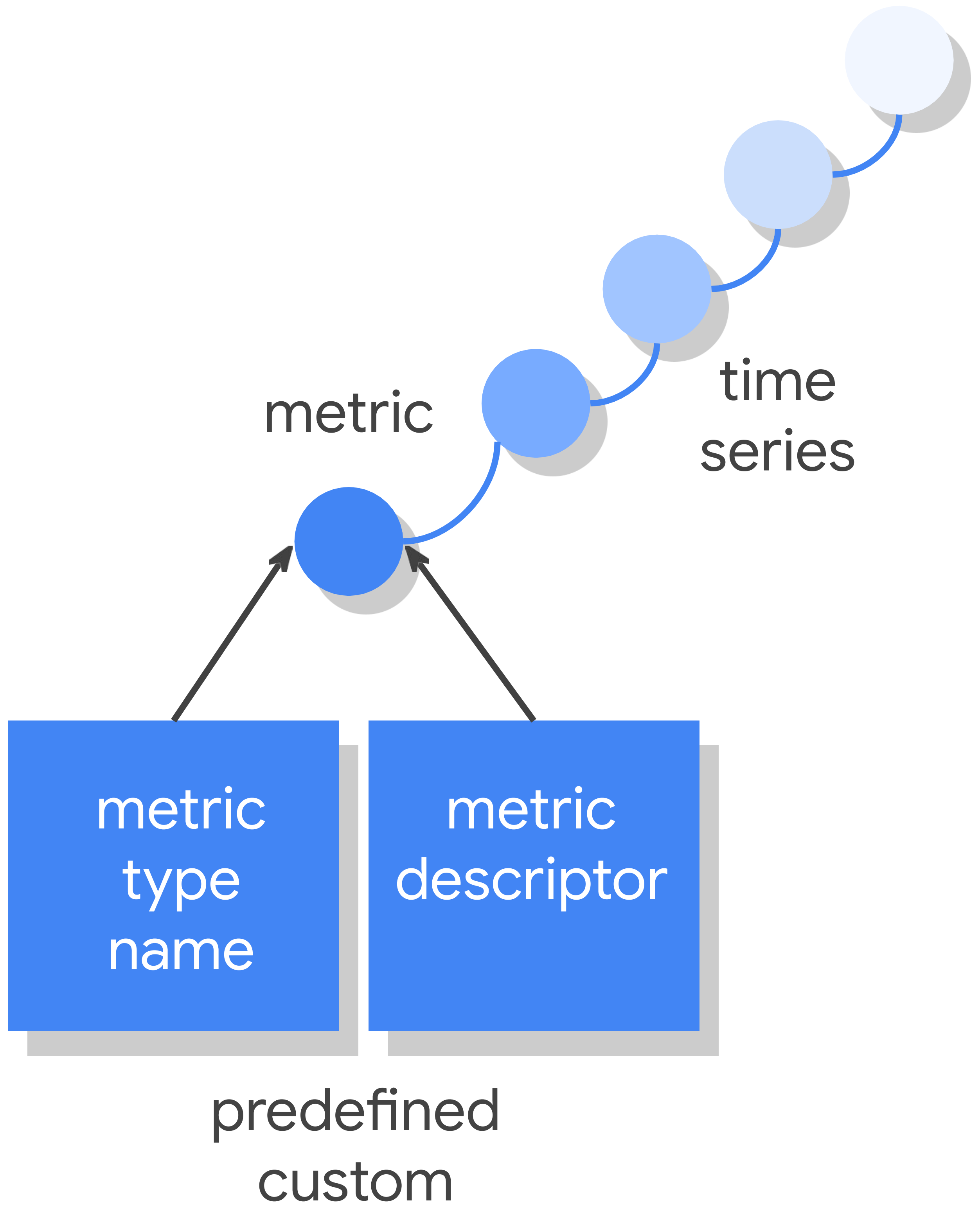


Google Cloud

# Custom metrics

Custom metric descriptor example in Python:

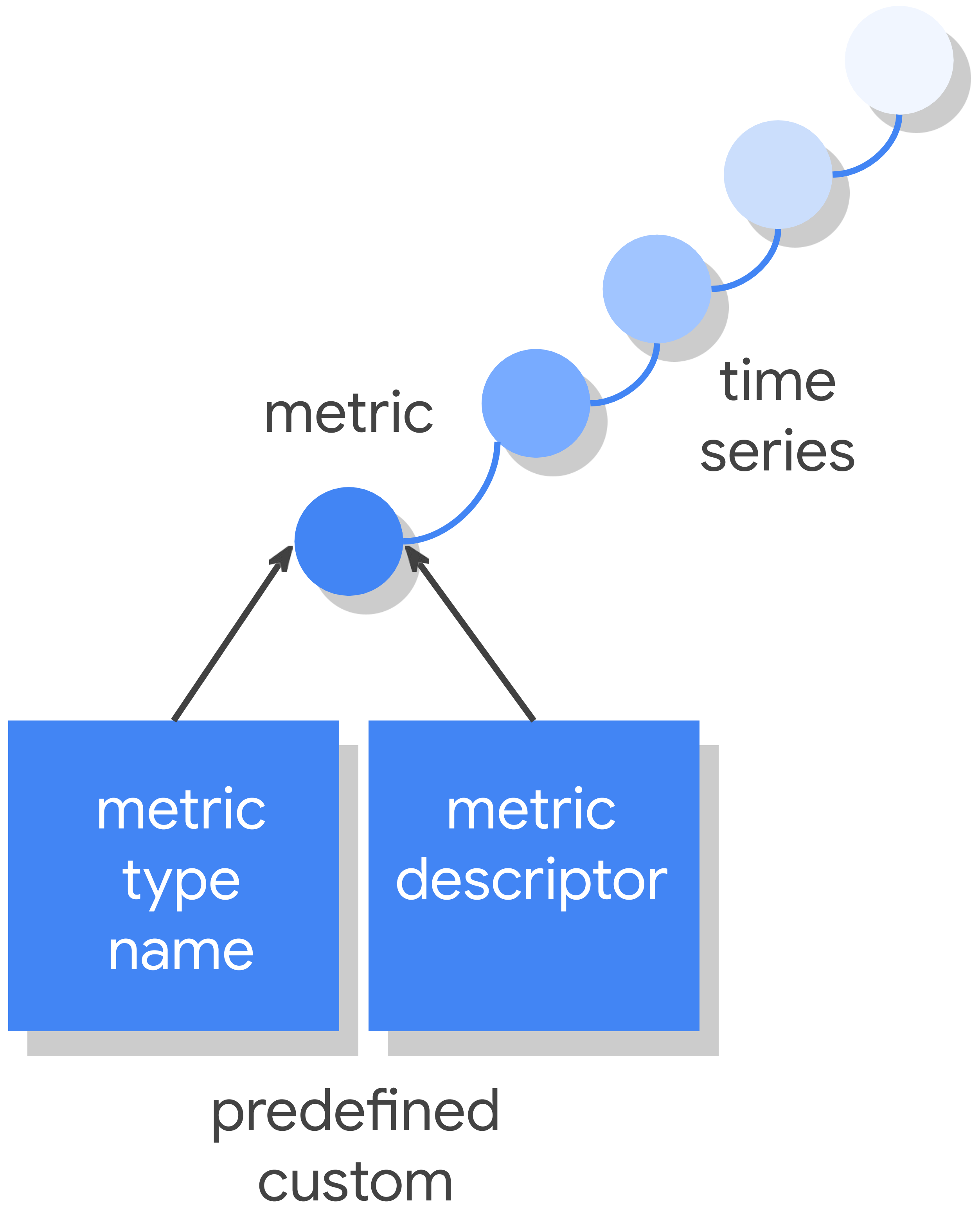
```
client = monitoring_v3.MetricServiceClient()
project_name =
client.project_path(project_id)

descriptor = monitoring_v3.types.MetricDescriptor()
descriptor.type = ('custom.googleapis.com/my_metric')

descriptor.metric_kind = (
  monitoring_v3.enums.MetricDescriptor.MetricKind.GAUG
  E)

descriptor.value_type = (
  monitoring_v3.enums.MetricDescriptor.ValueType.DOUBL
  E)
descriptor.description = 'Custom metric example.'


client.create_metric_descriptor(project_name, descriptor)
```



# Writing metrics

Using our custom metric, again in Python:

```
client =
monitoring_v3.MetricServiceClient()
project_name =
series = monitoring_v3.types.TimeSeries()
series.metric.type =
('custom.googleapis.com/my_metric')
series.resource.type = 'gce_instance'
series.resource.labels['instance_id']='126789012345
6789'  series.resource.labels['zone'] =
point = series.points.add()
point.value.double_value =
3.14  now = time.time()
point.interval.end_time.seconds =
int(now)
client.create_time_series(project_name, [series])
```



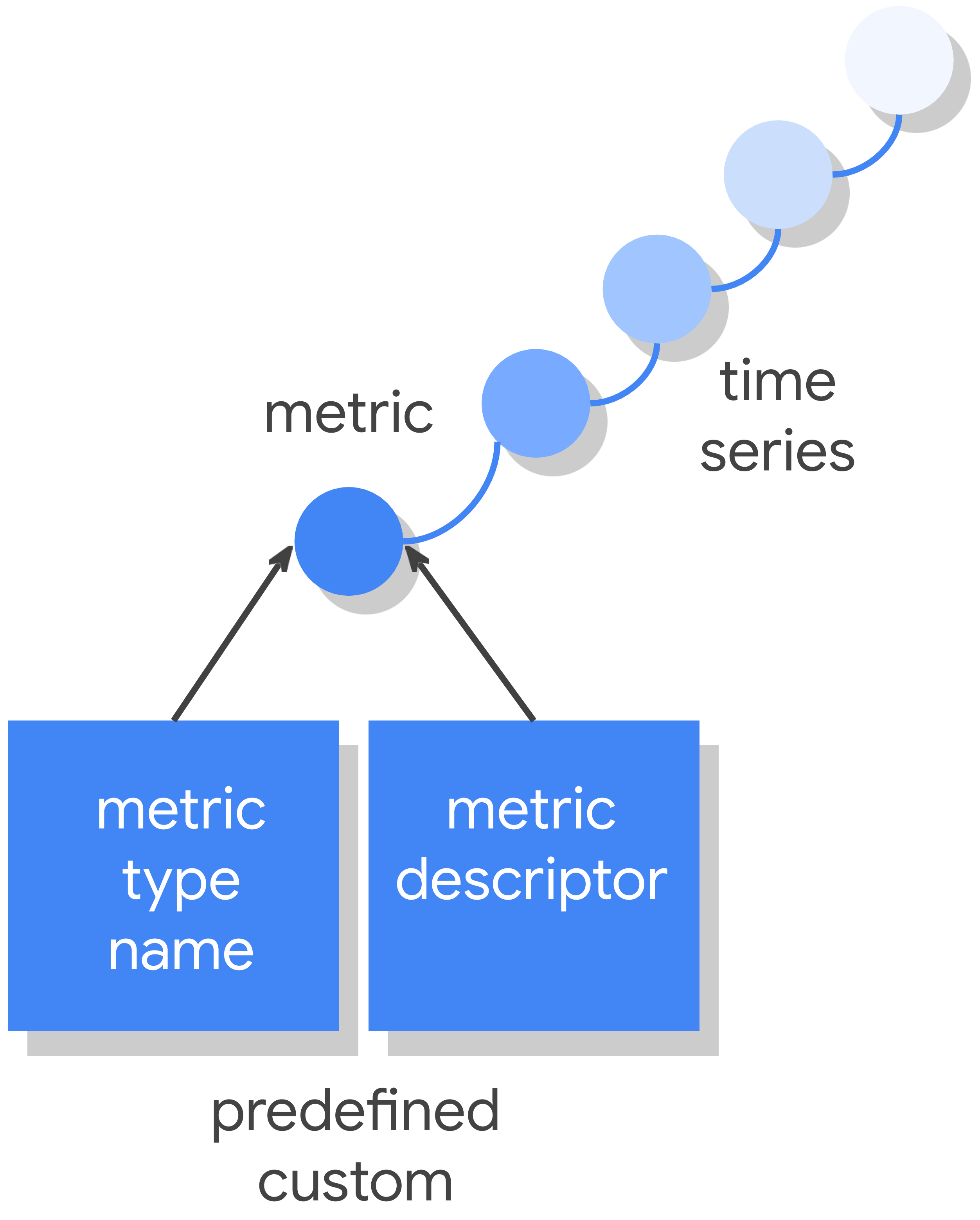
## Writing metrics

Using our custom metric, again in Python:

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

```
series = monitoring_v3.types.TimeSeries()
series.metric.type =
('custom.googleapis.com/my_metric')
series.resource.type = 'gce_instance'
series.resource.labels['instance_id']='126789012345
6789'  series.resource.labels['zone'] =
'us-central1-f'
```

```
point = series.points.add()
point.value.double_value =
3.14  now = time.time()
point.interval.end_time.seconds = int(now)

client.create_time_series(project_name, [series])
```

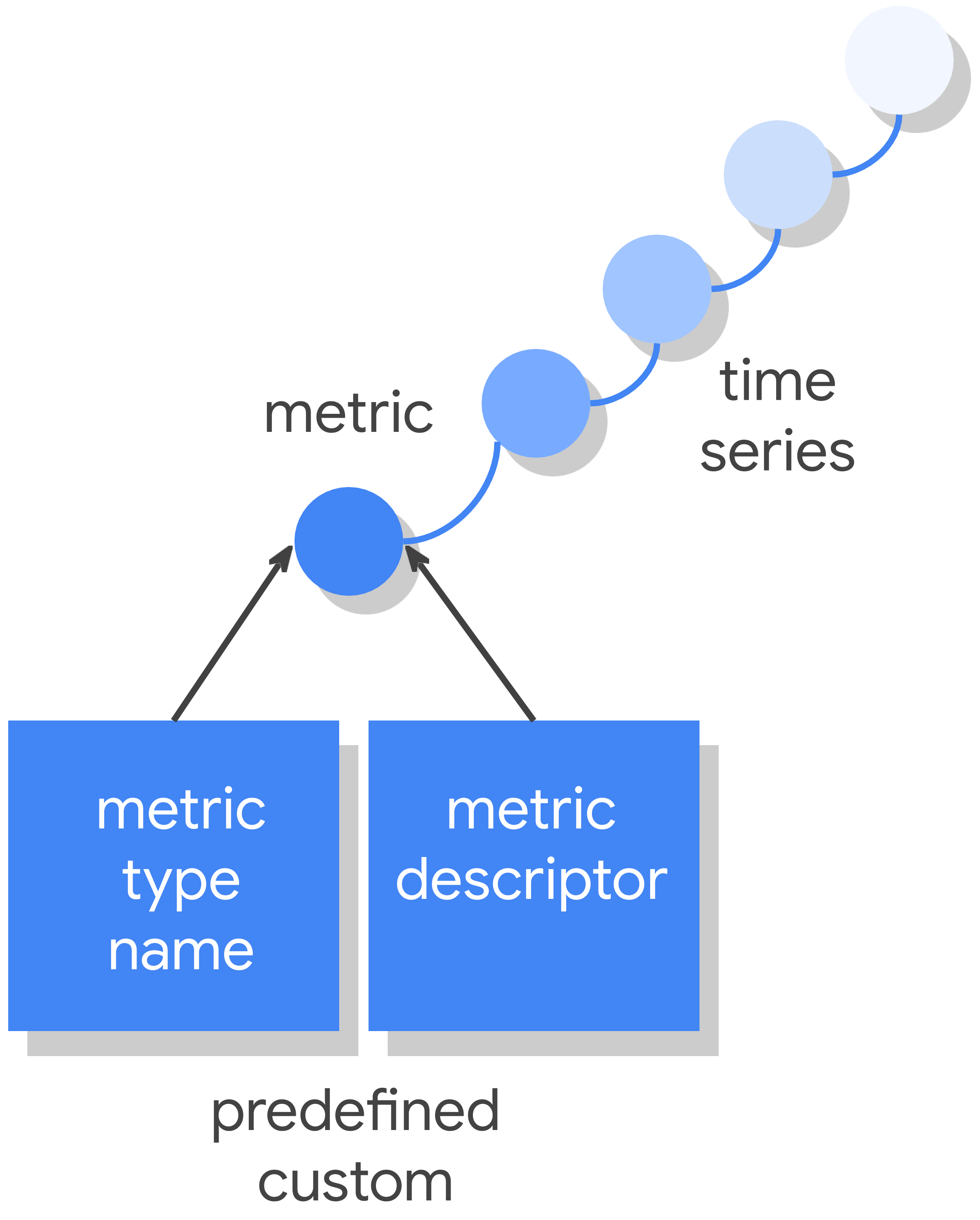


Google Cloud

# Writing metrics

Using our custom metric, again in Python:

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

series = monitoring_v3.types.TimeSeries()
series.metric.type =
('custom.googleapis.com/my_metric')
series.resource.type = 'gce_instance'
series.resource.labels['instance_id']='126789012345
6789'  series.resource.labels['zone'] =
'us-central1-f'
point = series.points.add()
point.value.double_value =
3.14  now = time.time()
point.interval.end_time.seconds = int(now)
client.create_time_series(project_name, [series])
```

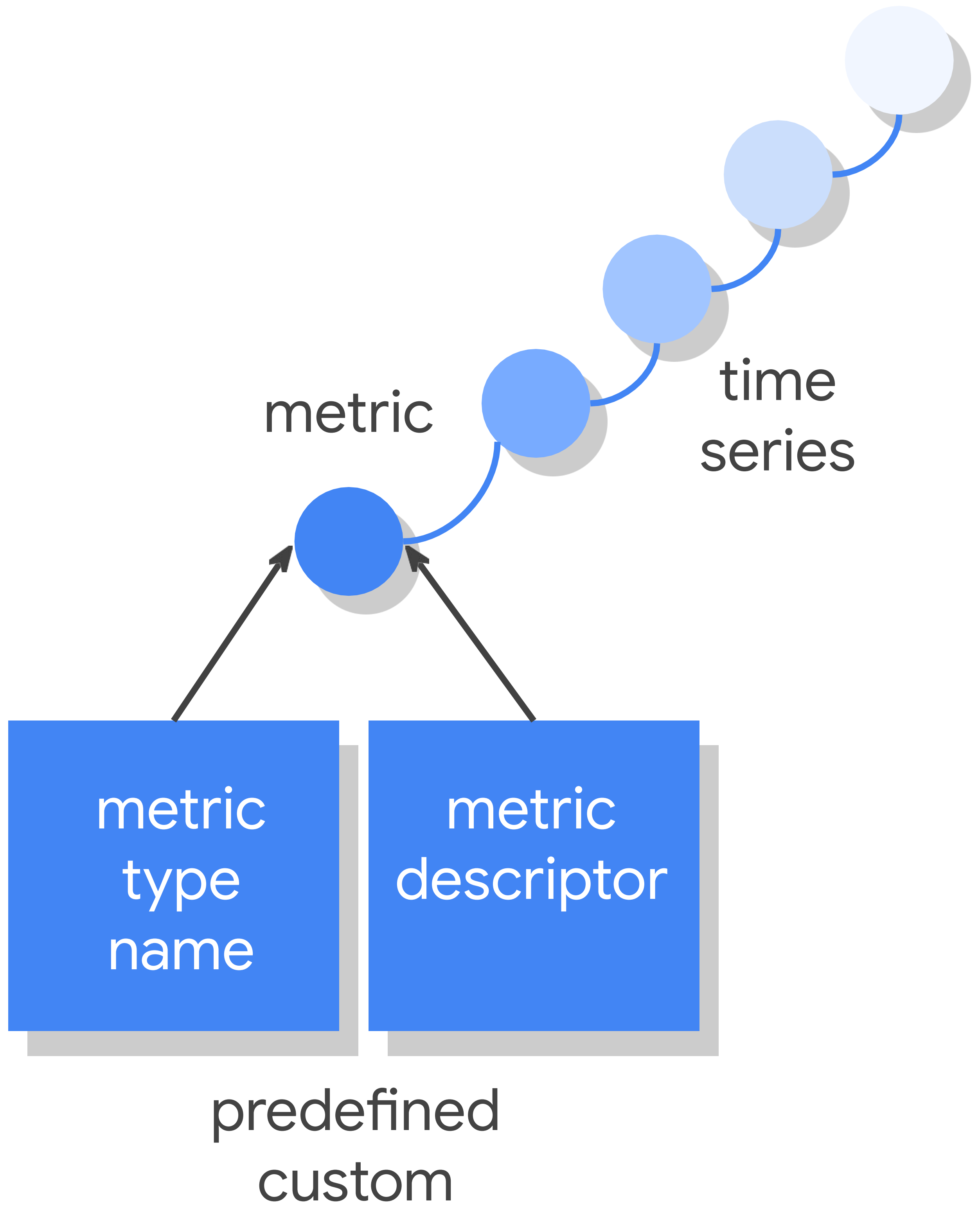


Google Cloud

# Writing metrics

Using our custom metric, again in Python:

```
client =
monitoring_v3.MetricServiceClient()
project_name =
series = monitoring_v3.types.TimeSeries()
series.metric.type =
('custom.googleapis.com/my_metric')
series.resource.type = 'gce_instance'
series.resource.labels['instance_id']='126789012345
6789'  series.resource.labels['zone'] =
point = series.points.add()
point.value.double_value =
3.14  now = time.time()
point.interval.end_time.seconds =
int(now)
client.create_time_series(project_name, [series])
```



Google Cloud

## What is OpenCensus?

- Open-source library to help capture, manipulate, and export  traces and metrics
  - Works with microservices and monoliths
- Supports many mainstream languages
  - Java, Python, Node.js, Go, C#, Erlang, and C++
- Low overhead and broadly supported
- OpenCensus is merging with OpenTracing to become OpenTelemetry
  - APIs planned to be backwards compatible

OpenCensus

Google Cloud

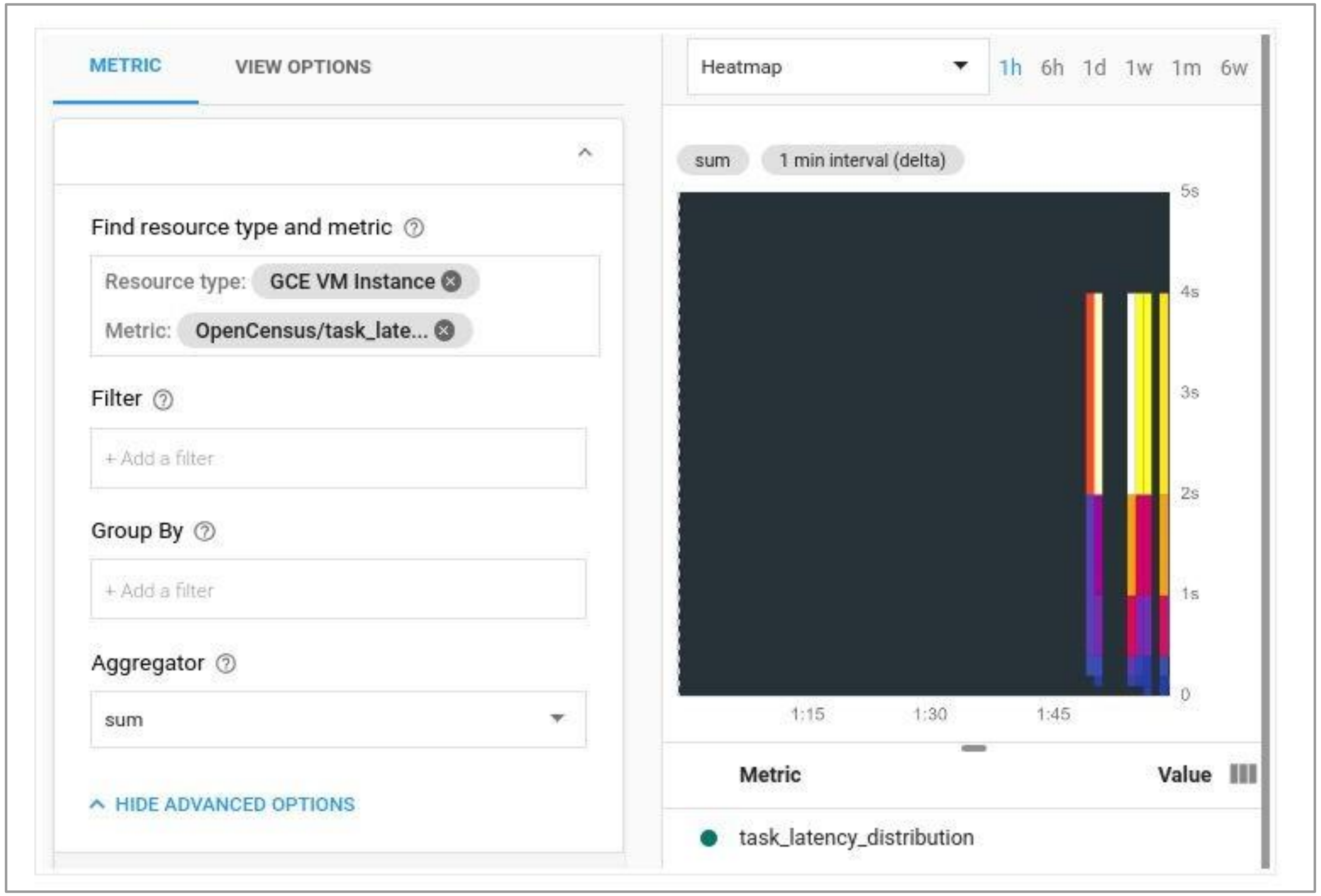
# Metrics expressed as measures and measurements

- A Measure represents a metric being recorded
  - **Name**: unique identifier
  - **Description**: purpose of the measure
  - **Unit**: string unit specifier, like “By”, “1”, or “ms”
    - Unit codes
  - Two measure value types: Int64 or a Float64
- A Measurement is a data point recorded as a Measure

## Views describe how measurements are collected

- A View represents the coupling of an Aggregation applied to a Measure and optionally Tags
- They contain:
    - Name: unique view name
    - Description
    - Measure: Measurement type
    - TagKeys: tagkeys used to group and filter metrics
    - Aggregation: How is the data gathered
        - Count, Distribution, Sum, or LastValue

# Exposing metrics from GCE using OpenCensus


METRIC VIEW OPTIONS

Find resource type and metric

Resource type: GCE VM Instance

Metric: OpenCensus/task_late...

Filter

+ Add a filter

Group By

+ Add a filter

Aggregator

sum

HIDE ADVANCED OPTIONS

Heatmap 1h 6h 1d 1w 1m 6w

sum 1 min interval (delta)

| Metric | Value |
|---|---|
| ● task_latency_distribution | |

Google Cloud

Google Cloud