

Monitoring GKE with Prometheus and Cloud Monitoring

1- Initial setup steps

Use Cloud Shell to create a Kubernetes cluster which will deploy the demo app. Start by configuring your environment.

Create a PROJECT_ID variable, replacing [PROJECT_ID] with the Project ID from lab page: `export PROJECT_ID=[PROJECT_ID]`

Enable the required APIs: `gcloud services enable compute.googleapis.com container.googleapis.com spanner.googleapis.com cloudprofiler.googleapis.com`

Clone the Petclinic demo app to your project:

```
git clone https://github.com/saturnism/spring-petclinic-gcp
```

Now create the container cluster, the container for the app, and enable Google Cloud services for your project:

```
gcloud container clusters create petclinic \
  --region=us-central1 \
  --num-nodes=2 \
  --machine-type=n1-standard-2 \
  --enable-autorepair \
  --enable-stackdriver-kubernetes
```

*This will take a few minutes to build.

2- Set up Cloud Monitoring

Set up the Cloud Monitoring agents namespace and RBAC:

```
kubectl apply -f
https://storage.googleapis.com/stackdriver-kubernetes/freshness-fixes/rbac-setup.yaml
--as=admin --as-group=system:masters
```

Set up the project id:

```
kubectl create configmap \
  --namespace=stackdriver-agents \
  google-cloud-config \
  --from-literal=project_id=$PROJECT_ID \
  --from-literal=credentials_path=""
```

Set up cluster name and location:

```
kubectl create configmap \
  --namespace=stackdriver-agents \
  cluster-config \
  --from-literal=cluster_name=petclinic \
  --from-literal=cluster_location=us-central1
```

Set up the agents:

```
kubectl apply -f
https://storage.googleapis.com/stackdriver-kubernetes/freshness-fixes/agents.yaml
```

Verify that the pods are running:

```
kubectl get pods --namespace=stackdriver-agents
```

3- Install the Cloud Monitoring Prometheus Collector

Now you'll install Cloud Monitoring Prometheus Collector to collect Prometheus metrics into Kubernetes Engine Monitoring:

```
kubectl apply -f https://storage.googleapis.com/spls/gsp243/rbac-setup.yaml
--as=admin --as-group=system:masters
```

```
curl -s https://storage.googleapis.com/spls/gsp243/prometheus-service.yaml | \
  sed -e "s/(\s*_kubernetes_cluster_name:*\).*\1 'petclinic'/g" | \
  sed -e "s/(\s*_kubernetes_location:*\).*\1 'us-central1'/g" | \
  sed -e "s/(\s*_stackdriver_project_id:*\).*\1 '${PROJECT_ID}'/g" | \
  kubectl apply -f -
```

Install Istio (basics):

```
cd ~/
export ISTIO_VERSION=0.7.1
```

```
curl -L https://git.io/getLatestIstio | sh -
cd istio-$ISTIO_VERSION
kubectl apply -f install/kubernetes/istio.yaml --as=admin --as-group=system:masters
```

Create and configure a Cloud Spanner database:

```
cd ~/
gcloud spanner instances create petclinic --config=regional-us-central1 --nodes=1
--description="PetClinic Spanner Instance"
gcloud spanner databases create petclinic --instance=petclinic
gcloud spanner databases ddl update petclinic --instance=petclinic
--ddl="$(cat spring-petclinic-gcp/db/spanner.ddl)"
```

This may take a couple of minutes.

4- Generate a service account and grant appropriate permissions

Create a new service account for the microservices:

```
gcloud iam service-accounts create petclinic --display-name "Petclinic Service
Account"
```

Run the following to grant IAM roles to the service account:

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member serviceAccount:petclinic@$PROJECT_ID.iam.gserviceaccount.com \
--role roles/cloudprofiler.agent
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member serviceAccount:petclinic@$PROJECT_ID.iam.gserviceaccount.com \
--role roles/clouddebugger.agent
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member serviceAccount:petclinic@$PROJECT_ID.iam.gserviceaccount.com \
--role roles/cloudtrace.agent
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member serviceAccount:petclinic@$PROJECT_ID.iam.gserviceaccount.com \
--role roles/spanner.databaseUser
```

Create a new JSON Service Account Key (keep this secure!):

```
gcloud iam service-accounts keys create ~/petclinic-service-account.json \
--iam-account petclinic@$PROJECT_ID.iam.gserviceaccount.com
```

Store Service Account as a Kubernetes secret:

```
gcloud iam service-accounts keys create ~/petclinic-service-account.json \
--iam-account petclinic@$PROJECT_ID.iam.gserviceaccount.com
```

```
kubectl create secret generic petclinic-credentials
--from-file=$HOME/petclinic-service-account.json
```

5- Deploy the Petclinic demo app

Go into the Petclinic demo directory:

```
cd ~/spring-petclinic-gcp
```

Deploy the demo app on Kubernetes:

```
kubectl apply -f kubernetes/
```

6- Verify the setup

To see the set of microservices that make up your webstore application, in the Cloud Console, select **Navigation menu > Kubernetes Engine > Workloads**. You should see something like the following:

Workloads						
REFRESH DEPLOY DELETE						
Workloads are deployable units of computing that can be created and managed in a cluster.						
In system object: False Filter workloads						
<input type="checkbox"/>	Name ↑	Status	Type	Pods	Namespace	Cluster
<input type="checkbox"/>	api-gateway	OK	Deployment	2/2	default	petclinic
<input type="checkbox"/>	customers-service	OK	Deployment	2/2	default	petclinic
<input type="checkbox"/>	heapster	OK	Deployment	1/1	stackdriver-agents	petclinic
<input type="checkbox"/>	istio-ca	OK	Deployment	1/1	istio-system	petclinic
<input type="checkbox"/>	istio-ingress	OK	Deployment	1/1	istio-system	petclinic
<input type="checkbox"/>	istio-mixer	OK	Deployment	1/1	istio-system	petclinic
<input type="checkbox"/>	istio-pilot	OK	Deployment	1/1	istio-system	petclinic
<input type="checkbox"/>	load-generator	OK	Deployment	0/0	default	petclinic
<input type="checkbox"/>	prometheus	OK	Deployment	1/1	default	petclinic
<input type="checkbox"/>	stackdriver-logging-agent	OK	Daemon Set	6/6	stackdriver-agents	petclinic
<input type="checkbox"/>	stackdriver-metadata-agent	OK	Daemon Set	6/6	stackdriver-agents	petclinic
<input type="checkbox"/>	stackdriver-metadata-agent-cluster-level	OK	Deployment	1/1	stackdriver-agents	petclinic
<input type="checkbox"/>	veto-service	OK	Deployment	2/2	default	petclinic
<input type="checkbox"/>	visits-service	OK	Deployment	2/2	default	petclinic

Note: It may take a few minutes for all the services to be available. You can refresh this page until all workloads show **OK** status.

Do not continue until your workloads page shows that your services are available.

7- Test your demo app!

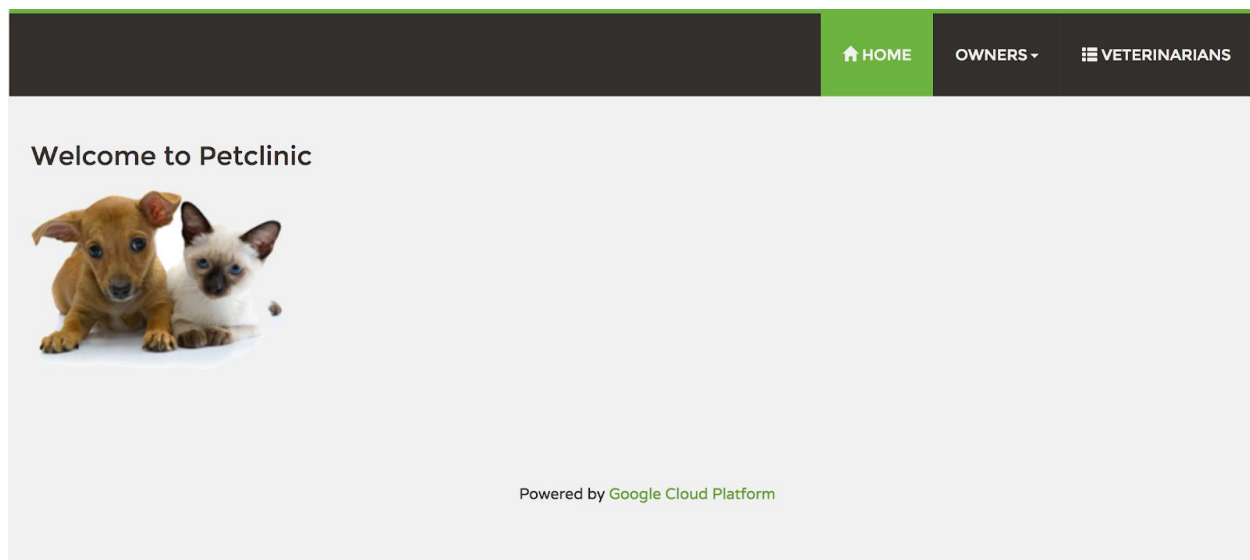
This lab puts you in charge of a demo Kubernetes app called PetClinic, which you just deployed on Google Kubernetes Engine. Before monitoring the application, explore it. To view the application, you need to find the endpoint. Run the following to find the Ingress IP address: `kubectl get ingress`

You should see something like:

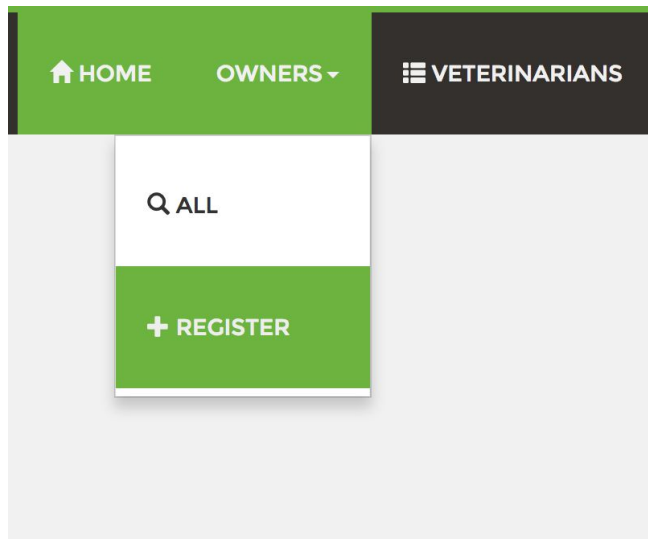
NAME	HOSTS	ADDRESS	PORTS	AGE
petclinic-ingress	*	35.238.194.243	80	18m

If the IP address entry is empty, give it another minute and retry the command.

Enter the IP address in a new browser window. You should see something like the following:



Now you'll add new pet owners and pets just as an employee of the PetClinic would do. Click on the **OWNERS** menu, then select the **+ REGISTER** option to add a new pet owner.



After adding a new owner click on the pet owner's name to see the details of that owner. Then click on the **Add New Pet** button to add a pet to that owner.

	HOME	OWNERS	VETERINARIANS
Owner Information			
Name	JD Velasquez		
Address	123 Google Way		
City	Mountain View		
Telephone	1234567890		
Edit Owner		Add New Pet	
Pets and Visits			

Add one or more pets, or add a visit to the clinic, to continue to use the app.

Owner Information			
Name	JD Velasquez		
Address	123 Google Way		
City	Mountain View		
Telephone	1234567890		
Edit Owner		Add New Pet	
Pets and Visits			
Delete Pet Edit Pet		Add Visit	
Name	Beta's friend	Visit Date	Description
Birth Date	1982 Feb 01	2018 May 22	Beta's friend is ill. Coming to see Dr. Restart
Type	DOG		

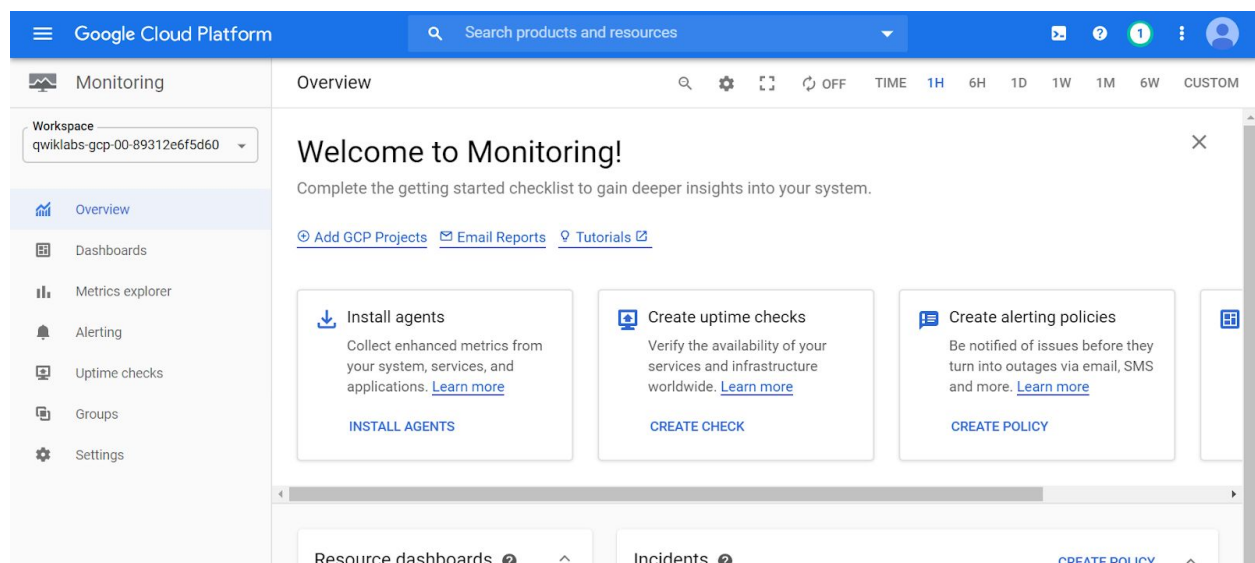
Now you're ready to monitor this app!

8- Create a Monitoring workspace

Now set up a Monitoring workspace that's tied to your Google Cloud Project. The following steps create a new account that has a free trial of Monitoring.

1. In the Cloud Console, click **Navigation menu > Monitoring**.
2. Wait for your workspace to be provisioned.

When the Monitoring dashboard opens, your workspace is ready.



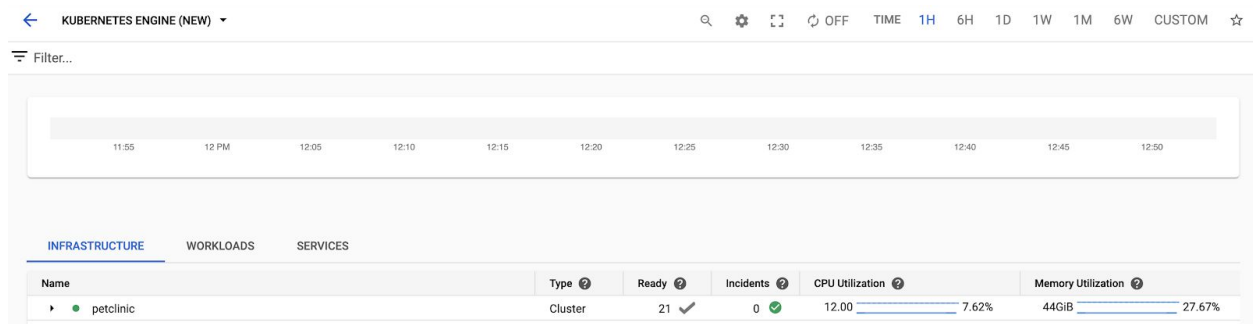
9- Kubernetes Monitoring

Kubernetes Engine Monitoring gives you comprehensive observability of your Kubernetes applications no matter where they run. It offers you a multi-cluster monitoring solution that lets you inspect the various Kubernetes components in your app from multiple perspectives. You can monitor metrics associated with different Kubernetes resources, like a cluster, a node, a pod, or a container.

In the left menu, click **Dashboard**, and then click **Kubernetes Engine (new)** in the Dashboard list.








Note: It may take a few minutes for the Kubernetes option to appear. Refresh your Cloud Monitoring page until you see Kubernetes.

You should see your petclinic cluster:



This page shows the monitored Kubernetes components (e.g., clusters, deployments, services, pods) in different views, but organized hierarchically.

While on the **Infrastructure** tab, click on the arrow next to the petclinic name to expand the cluster:

INFRASTRUCTURE		WORKLOADS		SERVICES		
Name		Type ?	Ready ?	Incidents ?	CPU Utilization ?	Memory Utilization ?
▼ petclinic		Cluster	21 ✓	0 ✓	12.00 <div><div></div></div> 7.62%	44GiB <div><div></div></div> 27.67%
▶  gke-petclinic-default-pool-0dd00f38-d4w9		Node	4 ✓	0 ✓	2.00 <div><div></div></div> 6.06%	7.3GiB <div><div></div></div> 21.08%
▶  gke-petclinic-default-pool-0dd00f38-gszx		Node	3 ✓	0 ✓	2.00 <div><div></div></div> 6.61%	7.3GiB <div><div></div></div> 25.57%
▶  gke-petclinic-default-pool-9f896fbe-3z8w		Node	2 ✓	0 ✓	2.00 <div><div></div></div> 6.64%	7.3GiB <div><div></div></div> 44.07%
▶  gke-petclinic-default-pool-9f896fbe-g5t0		Node	5 ✓	0 ✓	2.00 <div><div></div></div> 6.89%	7.3GiB <div><div></div></div> 27.20%
▶  gke-petclinic-default-pool-abc1e438-sfr7		Node	4 ✓	0 ✓	2.00 <div><div></div></div> 5.05%	7.3GiB <div><div></div></div> 17.04%
▶  gke-petclinic-default-pool-abc1e438-vdsm		Node	3 ✓	0 ✓	2.00 <div><div></div></div> 11.58%	7.3GiB <div><div></div></div> 31.18%
▶  Unscheduled		Node				

In this view, you can see the health of each node in the cluster, and their CPU/Memory utilization. You can also expand it to see what's running on this node, and its resource consumption.

Expand one of the nodes, and see the actual Pods. Expand further to see the containers within the pod.

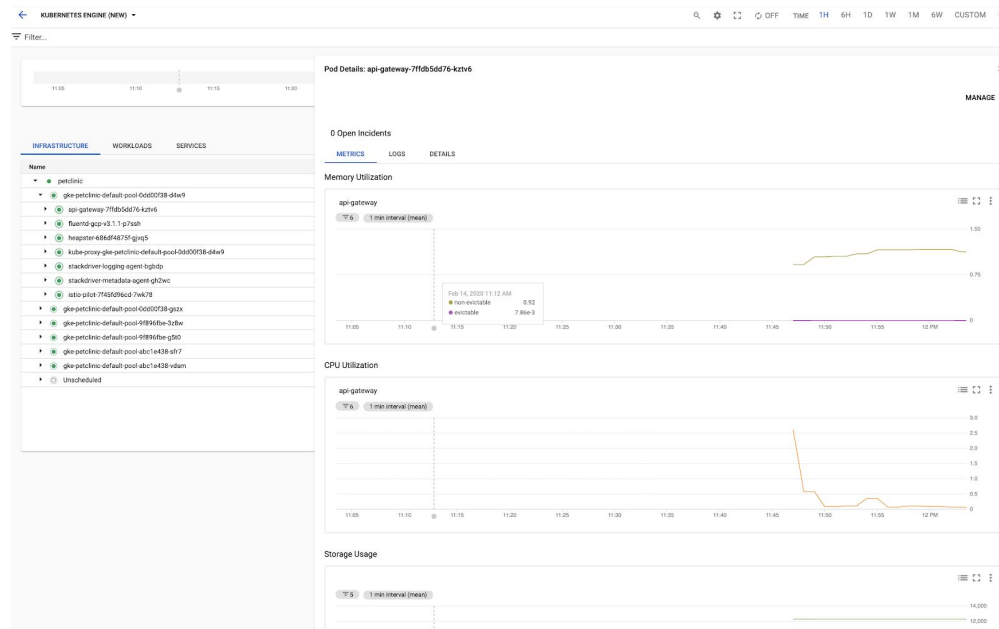
INFRASTRUCTURE WORKLOADS SERVICES						
NAME	RESOURCE TYPE	READY	INCIDENTS	CPU UTILIZATION		MEMORY UTILIZATION
petclinic	Cluster	14 ✓	0 ✓	12.00	5.25%	43.8GB 40.95%
gke-petclinic-default-poc	Node	7 ✓	0 ✓	2.00	4.32%	7.30GB 25.21%
api-gateway-9785d7c4	Pod	✓	0 ✓	0.20	7.24%	512MB 57.23%
fluentd-gcp-v3.1.0-kqnj	Pod	✓	0 ✓	0.10	7.56%	300MB 26.86%
kube-proxy-gke-petclin	Pod	✓	0 ✓	0.10	3.15%	0
metadata-agent-mncqj	Pod	✓	0 ✓	0.04	3.95%	0 4.35%
istio-ca-75fb7dc8d5-4f	Pod	✓	0 ✓			0
gke-petclinic-default-poc	Node	7 ✓	0 ✓	2.00	3.50%	7.30GB 36.10%

The **Workloads** tab has the Kubernetes workloads organized into namespaces, and deployments (or other types of workloads). You can also expand based on this view. This is useful for application owners to see the current metrics of their deployments.

The **Services** tab has the Kubernetes Services, and also the Pods that are selected/load-balanced by the service.

On any of these tabs you can click the resources and see their monitored metrics. For instance, click on a Pod, and you'll see key metrics associated with that Pod, including Restart Counts, CPU Utilization, Memory Utilization, Network Usage, and even Logs associated with the containers inside of this pod. Similarly, you see the details of all Containers inside of each Pod.

In a single place, you have metrics, logs, events, and metadata for all your Kubernetes components, which gives you rich observability of your Kubernetes application.



10- Creating an Alerting Policy


You can increase observability of your Kubernetes application by creating an alerting policy using the new Kubernetes metrics.

In the left menu, click **Alerting**, and then click **Create Policy**.

You add an alerting policy by defining conditions. For this lab, you want to be notified when a container is consuming 80% of the CPU limit for over 1 minute. If the alert condition is triggered, a notification will be sent, which can also link to relevant

documentation, like a run book or standard operating procedures, to respond to the issue.

Click **Add Condition**.

 Create new alerting policy

Conditions

Conditions describe when apps and services are considered unhealthy. When conditions are met, they trigger alerting policy violations.

ADD CONDITION

Policy triggers

Triggers when
ANY condition is met

Notifications (optional)

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

ADD NOTIFICATION CHANNEL

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.

Documentation

☐ Preview Markdown

SAVE

CANCEL

In the metrics drop down you can see several container metrics, such as CPU usage, Memory usage, Restart count (how many times the container was restarted), and also utilization against a resource limit or a resource request. For example, you can monitor containers that are consuming too much memory and may get automatically evicted by Kubernetes. Start typing **CPU limit utilization** and select it from the dropdown:

METRIC

UPTIME CHECK

PROCESS HEALTH

Target ?

Find resource type and metric ?

cpu limit uti

Metrics

CPU limit utilizationk8s_containerkubernetes.io/container/cpu/limit_utilization

CPU utilizationgke_containercontainer.googleapis.com/container/cpu/utilization

In the Configuration section, set the following parameters:

Condition triggers if: **All time series violate**

Condition: **is above**

Threshold: **0.8**

For: **1 minute**

Configuration

Condition triggers if

All time series violate ▼

Condition	Threshold	For
is above ▼	0.8 %	1 minute ▼

ADD

CANCEL

Click **Add**.

Name the alerting policy to give it more meaning.

Then click **Save**.

Your alerting policy should now be ready. You can find it in the **Policies** section of the **Alerting** page.

11- Inspect your Kubernetes environment with Metrics Explorer

In the left menu, select **Metrics Explorer**.

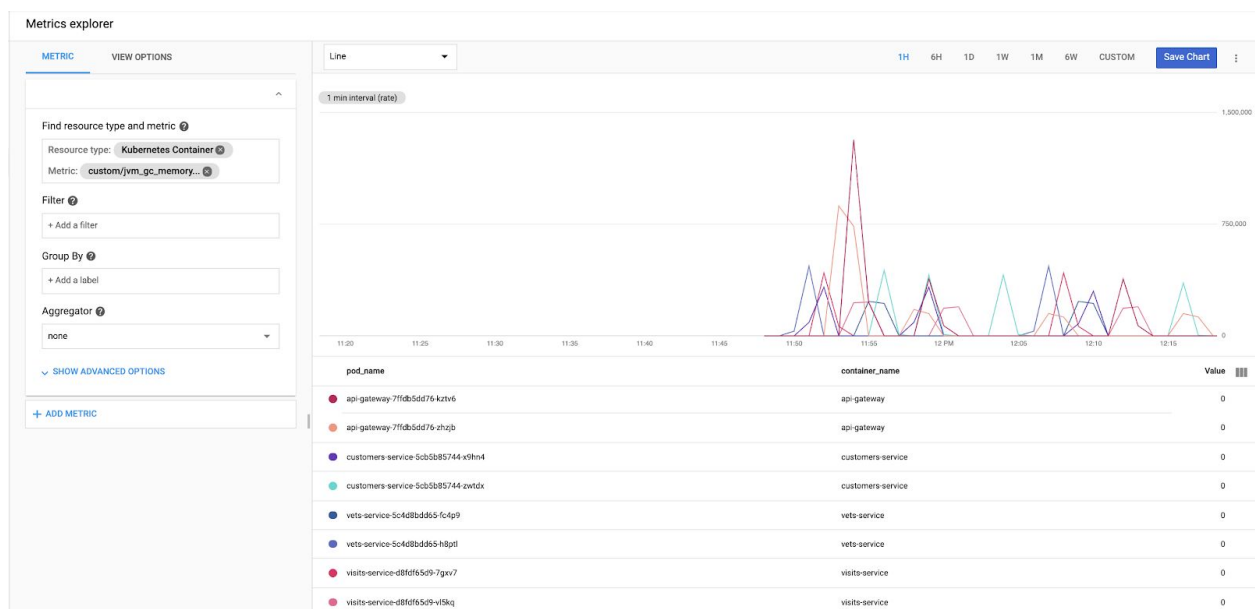
All of the supported metrics are available in the Metrics Explorer. Since you turned on Cloud Monitoring Prometheus monitoring support, the Prometheus metrics exposed by your microservices can be monitored as well.

For example, for Go applications, you can see various Prometheus metrics like `go_threads`. For a Java application, you can find others like `jvm_memory_max_bytes`.

Type `jvm_gc_mem` into the Resource type and select the `jvm_gc_memory_allocated_bytes_total` metric.

Type **Kubernetes** into the Resource type and then select **Kubernetes Container**.

You should now be able to inspect this Prometheus metric for all containers in your cluster.



You can Filter and Group By different attributes, such as: namespace, controller name (Deployment name), controller type (Deployment, StatefulSet, etc), and even Kubernetes Labels.

You can also group by these attributes, so you can get an aggregated metric by labels, by deployments, or by service names. Feel free to explore!

You have successfully:

- **Increased observability** for your Kubernetes environment.
- **Inspected** the various Kubernetes components in your application.
- **Proactively monitored** your Kubernetes application.
- **Added an alerting policy** triggered by specific conditions on Kubernetes metrics.
- **Monitored** Prometheus metrics