INFRASTRUCTURE AND CLOUD ENVIRONMENT DOCUMENT

DOCUMENTATION

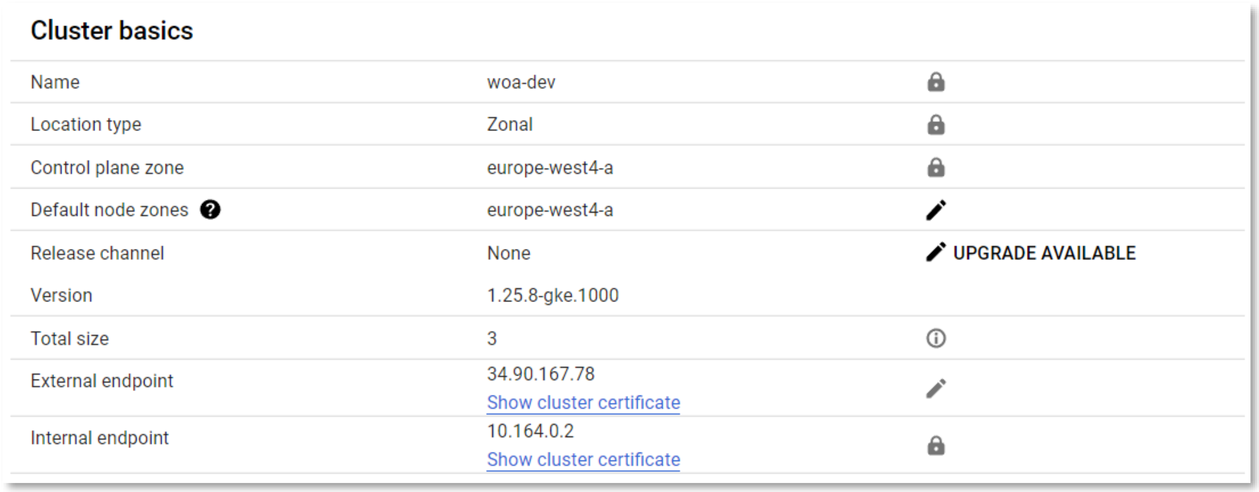
Fontys University Of Applied Sciences
Tsanko Nedelchev

Contents

Introduction

This document provides a comprehensive overview of the infrastructure setup for a Kubernetes cluster hosted on Google Kubernetes Engine (GKE). It aims to document the configuration and architecture details, as well as the integration with Google Artifact Registry for image management. The infrastructure is built around Google Kubernetes Engine, which offers a scalable and managed platform. It includes a Kubernetes cluster that hosts a set of microservices and a client application. Additionally, a RabbitMQ deployment is utilized for messaging purposes. The microservices and client application pods are designed for horizontal scaling, allowing up to 10 pods. The cluster features a managed system that incorporates monitoring and alerting capabilities. It triggers alerts when metrics exceed custom-defined thresholds, ensuring proactive response to potential issues. The cluster supports cloud development functionality, leveraging Scaffold during the development process. It also facilitates automated CI/CD deployments from GitHub using Docker images obtained from Google Artifact Registry. To ensure efficient image management, the cluster utilizes Google Artifact Registry. This managed service provides a secure and centralized repository for storing, versioning, and distributing container images. By leveraging Artifact Registry, the infrastructure streamlines the management of container images and enhances the deployment process. Overall, this infrastructure setup on Google Kubernetes Engine, combined with integration with Google Artifact Registry, offers a robust and scalable platform for hosting microservices and deploying applications with ease.

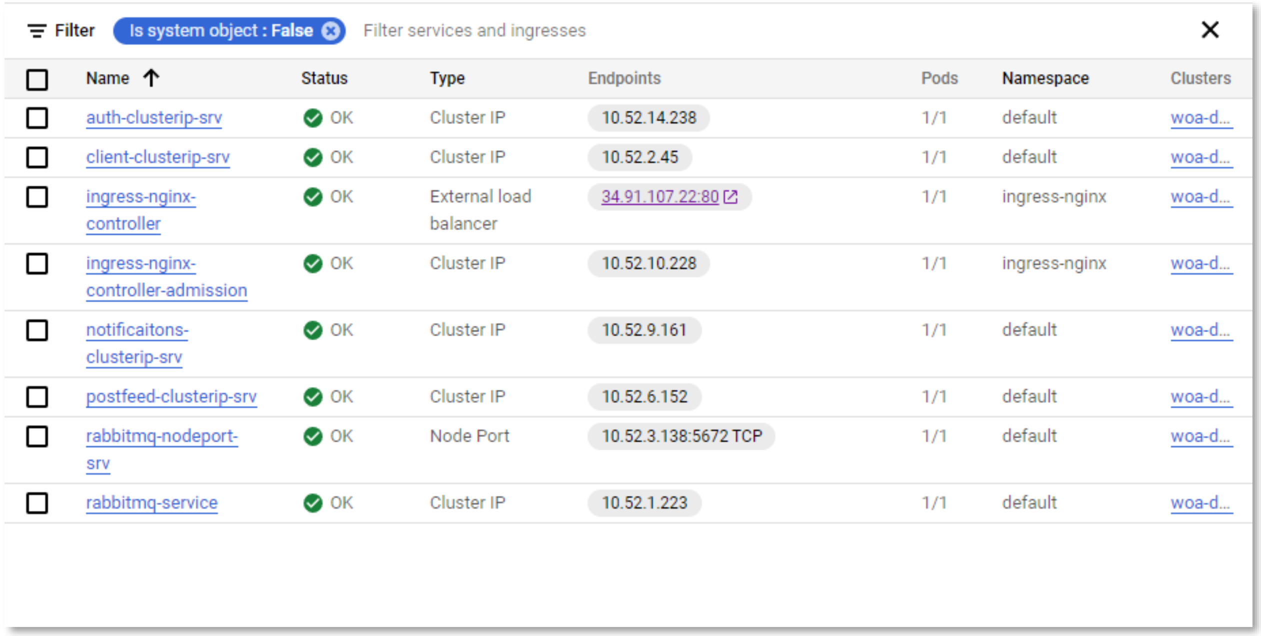
Kubernetes Cluster Setup



Cluster basics

Name	woa-dev	🔒
Location type	Zonal	🔒
Control plane zone	europe-west4-a	🔒
Default node zones	europe-west4-a	✎
Release channel	None	✎ UPGRADE AVAILABLE
Version	1.25.8-gke.1000	
Total size	3	ⓘ
External endpoint	34.90.167.78 Show cluster certificate	✎
Internal endpoint	10.164.0.2 Show cluster certificate	🔒

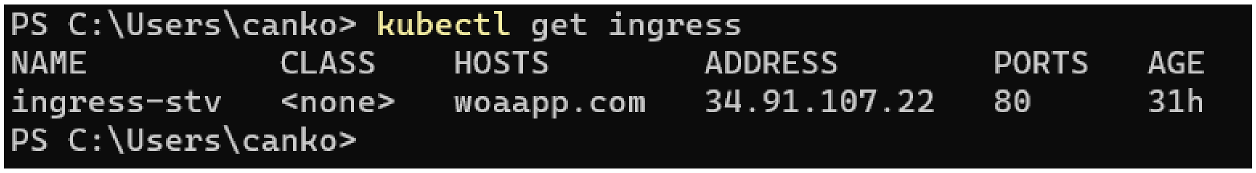
The Google Kubernetes Cluster is setup in zone europe-west4-a which is the Netherlands zone in order to ensure the best connectivity and higher availability. It runs on the default GKE version 1.25.8-gke.1000. It contains 3 nodes.



Filter Is system object : False Filter services and ingresses

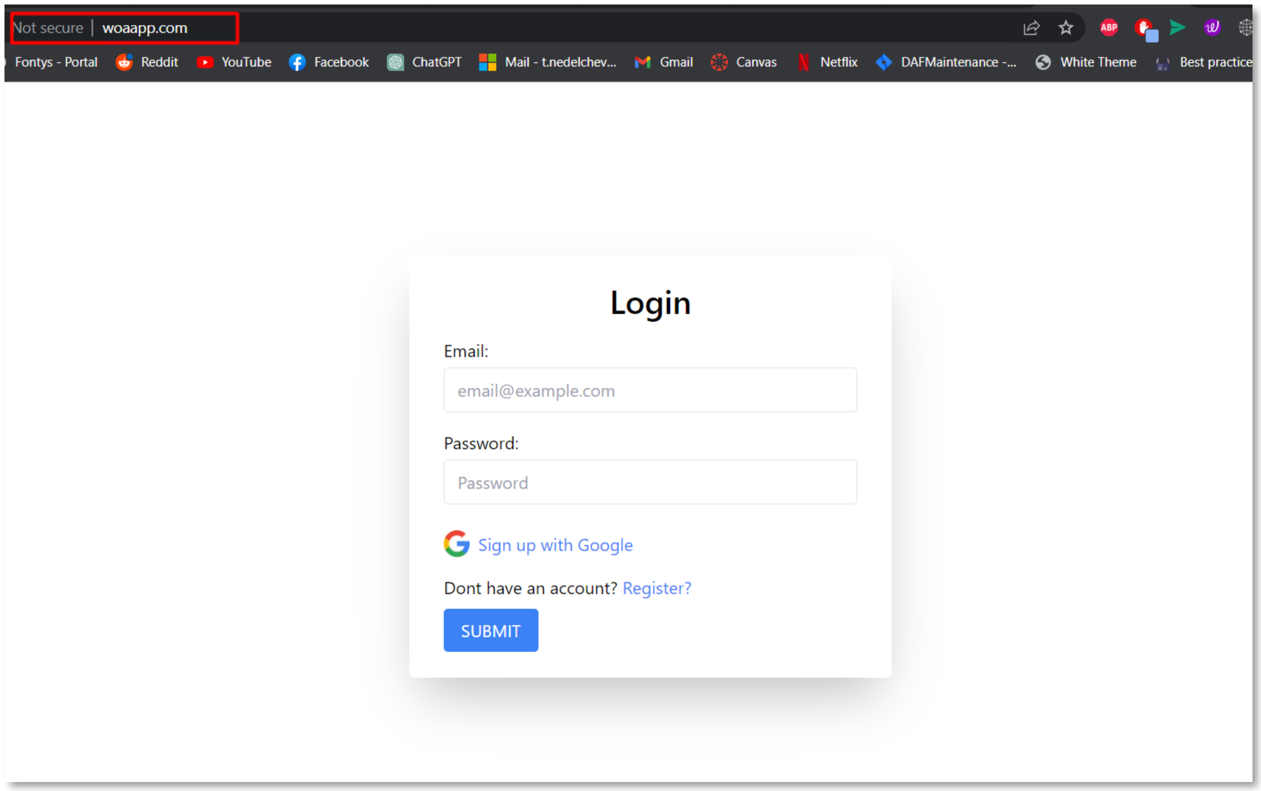
Name ↑	Status	Type	Endpoints	Pods	Namespace	Clusters
auth-clusterip-srv	OK	Cluster IP	10.52.14.238	1/1	default	woa-d...
client-clusterip-srv	OK	Cluster IP	10.52.2.45	1/1	default	woa-d...
ingress-nginx-controller	OK	External load balancer	34.91.107.22:80	1/1	ingress-nginx	woa-d...
ingress-nginx-controller-admission	OK	Cluster IP	10.52.10.228	1/1	ingress-nginx	woa-d...
notificaitons-clusterip-srv	OK	Cluster IP	10.52.9.161	1/1	default	woa-d...
postfeed-clusterip-srv	OK	Cluster IP	10.52.6.152	1/1	default	woa-d...
rabbitmq-nodeport-srv	OK	Node Port	10.52.3.138:5672 TCP	1/1	default	woa-d...
rabbitmq-service	OK	Cluster IP	10.52.1.223	1/1	default	woa-d...

The cluster hosts several microservices deployments inside pods along with their services, a RabbitMQ deployment for messaging between the microservices and an ingress-nginx load balancer that exposes an external IP address of 34.91.107.22.

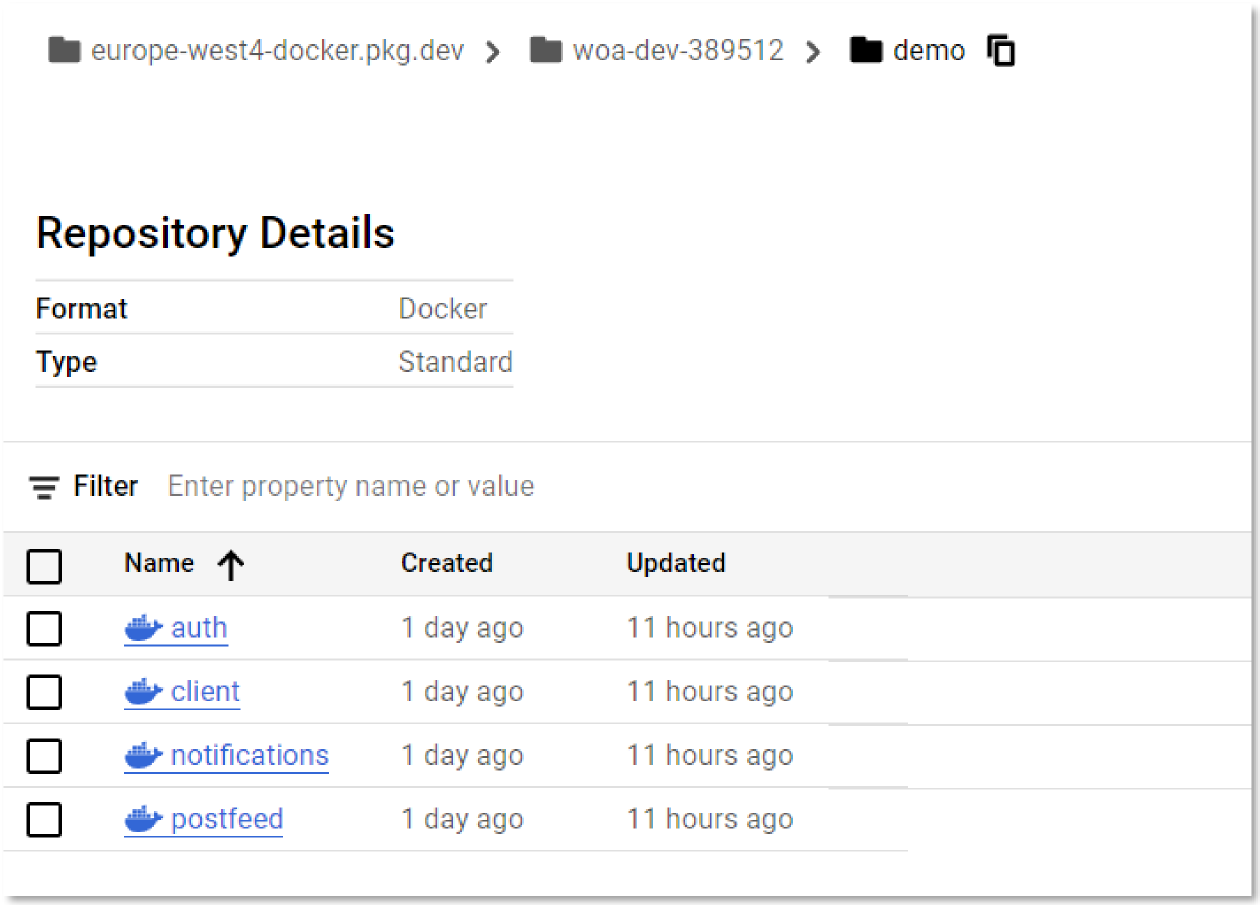


```
PS C:\Users\canko> kubectl get ingress
NAME          CLASS    HOSTS        ADDRESS        PORTS   AGE
ingress-stv   <none>   woaapp.com   34.91.107.22   80      31h
PS C:\Users\canko>
```

Furthermore, the ingress load balancer is configured with a custom domain name of woaapp.com which exposes the application.



Google Artifact Registry


europe-west4-docker.pkg.dev > woa-dev-389512 > demo

Repository Details

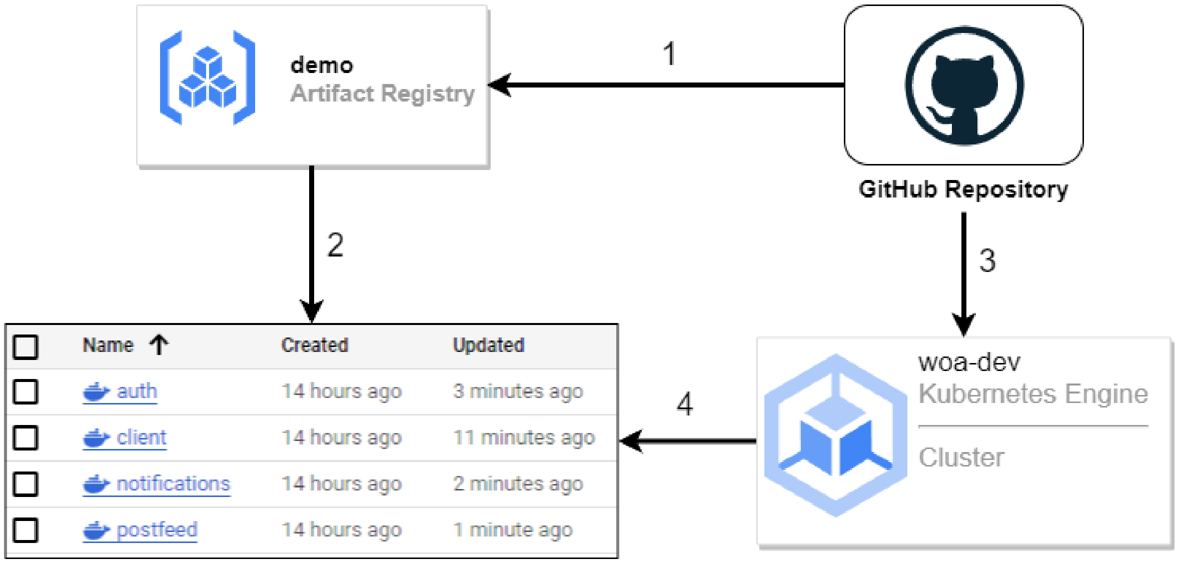
Format	Docker
Type	Standard

Filter Enter property name or value

Name	Created	Updated
auth	1 day ago	11 hours ago
client	1 day ago	11 hours ago
notifications	1 day ago	11 hours ago
postfeed	1 day ago	11 hours ago

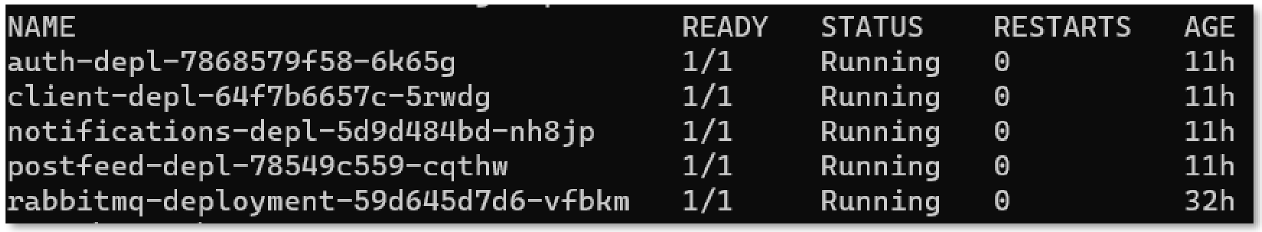
The google Artifact Registry hosts all the microservices' images including the client application. The images are pushed to the registry by GitHub using a google service account for authentication and from then on they can be pulled by the cluster configuration on deployment. Using the google artifact registry ensures that the images are always accessible to the cluster via a seamless integration and that they are safe from the outside world inside a private repository.

Image Management Workflow



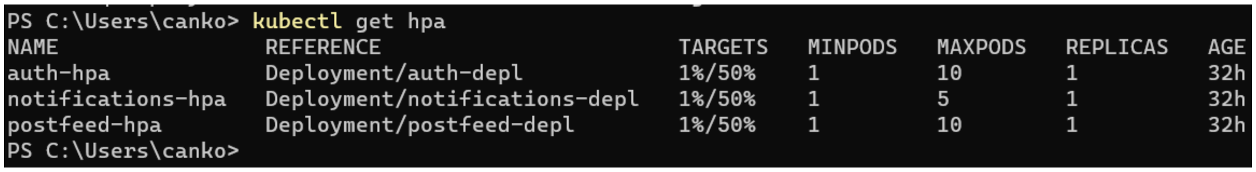
	Name	Created	Updated
☐	auth	14 hours ago	3 minutes ago
☐	client	14 hours ago	11 minutes ago
☐	notifications	14 hours ago	2 minutes ago
☐	postfeed	14 hours ago	1 minute ago

The image management workflow begins in GitHub as the new version of the application is pushed on the main branch which is the production branch. The GitHub action responsible for building and pushing the images is triggered and it builds the images using the repository secrets in order to securely set environment variables inside the Dockerfiles of all the microservices. Then the GitHub action pushes the built image to Google Artifact Registry where they are stored securely and ready to be pulled by the cluster configuration when it is redeployed. After the build action finishes it applies all Kubernetes manifests to the GKE cluster. The manifests are configured to pull their image from the Registry and start the pods.



```
NAME                               READY   STATUS    RESTARTS   AGE
auth-depl-7868579f58-6k65g         1/1     Running   0          11h
client-depl-64f7b6657c-5rwdg       1/1     Running   0          11h
notifications-depl-5d9d484bd-nh8jp 1/1     Running   0          11h
postfeed-depl-78549c559-cqthw      1/1     Running   0          11h
rabbitmq-deployment-59d645d7d6-vfbkm 1/1   Running   0          32h
```

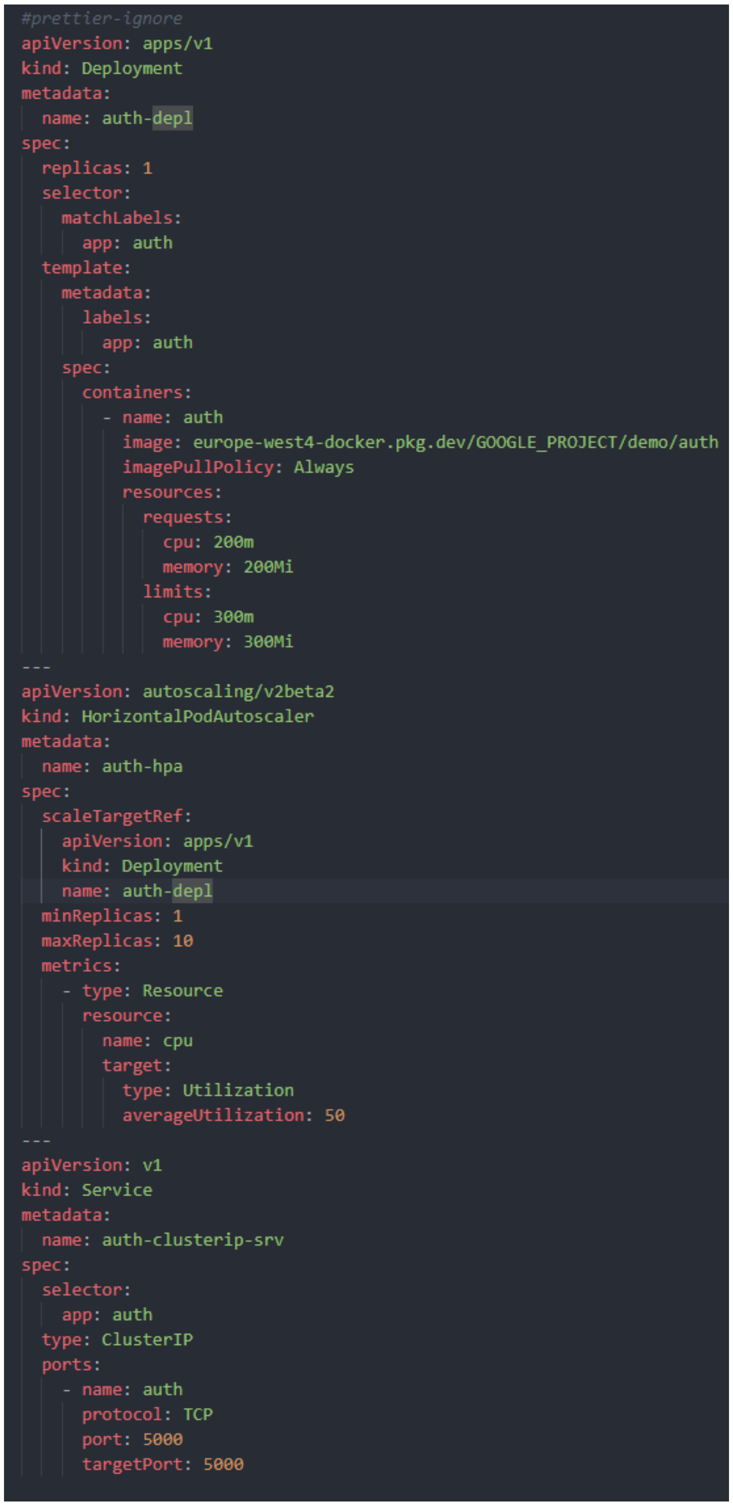
Once deployed the pods take about 10 minutes to stabilize and scale down.



```
PS C:\Users\canko> kubectl get hpa
NAME                REFERENCE                       TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
auth-hpa            Deployment/auth-depl            1%/50%    1         10        1          32h
notifications-hpa   Deployment/notifications-depl   1%/50%    1         5         1          32h
postfeed-hpa        Deployment/postfeed-depl        1%/50%    1         10        1          32h
PS C:\Users\canko>
```

Once they stabilize the pods run on 1 replica. The moment the usage goes above the target threshold the pods start scaling horizontally until the desired amount is reached. The pods scale up 10 seconds after the usage passes the threshold and scale down around 10 minutes after they are no longer needed.

The Kubernetes Manifest Files



```yaml
#prettier-ignore
apiVersion: apps/v1
kind: Deployment
metadata:
  name: auth-depl
spec:
  replicas: 1
  selector:
    matchLabels:
      app: auth
  template:
    metadata:
      labels:
        app: auth
    spec:
      containers:
        - name: auth
          image: europe-west4-docker.pkg.dev/GOOGLE_PROJECT/demo/auth
          imagePullPolicy: Always
          resources:
            requests:
              cpu: 200m
              memory: 200Mi
            limits:
              cpu: 300m
              memory: 300Mi
---
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: auth-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: auth-depl
  minReplicas: 1
  maxReplicas: 10
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 50
---
apiVersion: v1
kind: Service
metadata:
  name: auth-clusterip-srv
spec:
  selector:
    app: auth
  type: ClusterIP
  ports:
    - name: auth
      protocol: TCP
      port: 5000
      targetPort: 5000
```

The .yaml files contain the deployment manifests along with the autoscaling setup and the clusterIP service exposing them to the rest of the pods inside the cluster.

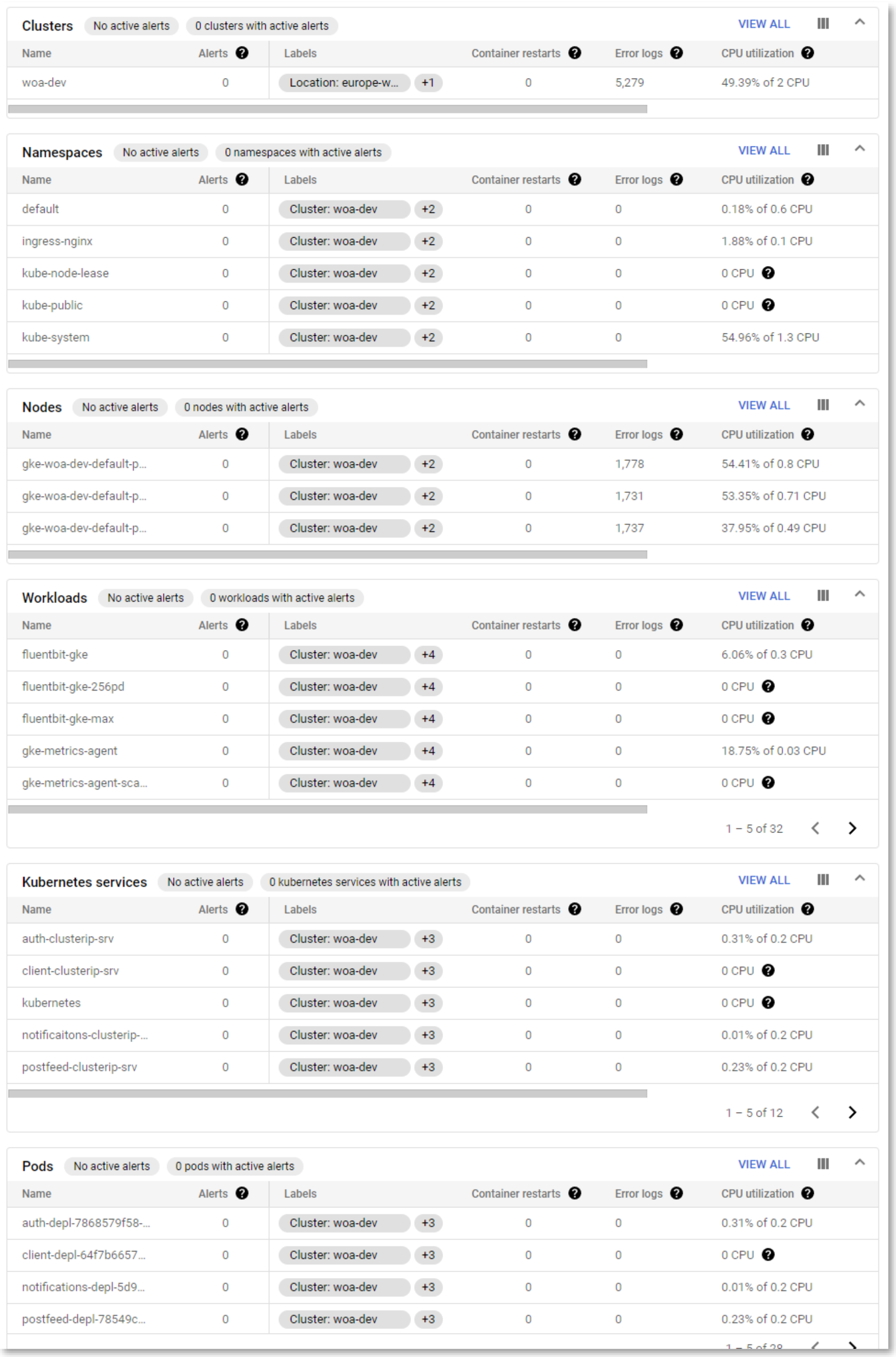
The deployment manifest specifies the name of the image that needs to be pulled from the Artifact Registry and contains a variable that gets filled by the CI/CD pipeline through the repository secrets in order to specify the name of the project that the Artifact Registry is also a part of.
Furthermore it contains the resources that the container wants to reserve from the cluster.

The autoscaling setup defines the number of replicas the pods can scale to and the condition that it must follow in order to determine when it should scale.

The service exposes the pod's port and IP to the rest of the pods inside the cluster.

Monitoring and Alerting

In terms of monitoring GKE offers a lot. Right out of the gate it offers a few pre-made dashboards for monitoring purposes that show all kinds of metrics connected to the Kubernetes cluster.



Clusters No active alerts · 0 clusters with active alerts · VIEW ALL

Name	Alerts	Labels		Container restarts	Error logs	CPU utilization
woa-dev	0	Location: europe-w...	+1	0	5,279	49.39% of 2 CPU

Namespaces No active alerts · 0 namespaces with active alerts · VIEW ALL

Name	Alerts	Labels		Container restarts	Error logs	CPU utilization
default	0	Cluster: woa-dev	+2	0	0	0.18% of 0.6 CPU
ingress-nginx	0	Cluster: woa-dev	+2	0	0	1.88% of 0.1 CPU
kube-node-lease	0	Cluster: woa-dev	+2	0	0	0 CPU
kube-public	0	Cluster: woa-dev	+2	0	0	0 CPU
kube-system	0	Cluster: woa-dev	+2	0	0	54.96% of 1.3 CPU

Nodes No active alerts · 0 nodes with active alerts · VIEW ALL

Name	Alerts	Labels		Container restarts	Error logs	CPU utilization
gke-woa-dev-default-p...	0	Cluster: woa-dev	+2	0	1,778	54.41% of 0.8 CPU
gke-woa-dev-default-p...	0	Cluster: woa-dev	+2	0	1,731	53.35% of 0.71 CPU
gke-woa-dev-default-p...	0	Cluster: woa-dev	+2	0	1,737	37.95% of 0.49 CPU

Workloads No active alerts · 0 workloads with active alerts · VIEW ALL

Name	Alerts	Labels		Container restarts	Error logs	CPU utilization
fluentbit-gke	0	Cluster: woa-dev	+4	0	0	6.06% of 0.3 CPU
fluentbit-gke-256pd	0	Cluster: woa-dev	+4	0	0	0 CPU
fluentbit-gke-max	0	Cluster: woa-dev	+4	0	0	0 CPU
gke-metrics-agent	0	Cluster: woa-dev	+4	0	0	18.75% of 0.03 CPU
gke-metrics-agent-sca...	0	Cluster: woa-dev	+4	0	0	0 CPU

1 – 5 of 32

Kubernetes services No active alerts · 0 kubernetes services with active alerts · VIEW ALL

Name	Alerts	Labels		Container restarts	Error logs	CPU utilization
auth-clusterip-srv	0	Cluster: woa-dev	+3	0	0	0.31% of 0.2 CPU
client-clusterip-srv	0	Cluster: woa-dev	+3	0	0	0 CPU
kubernetes	0	Cluster: woa-dev	+3	0	0	0 CPU
notificaitons-clusterip-...	0	Cluster: woa-dev	+3	0	0	0.01% of 0.2 CPU
postfeed-clusterip-srv	0	Cluster: woa-dev	+3	0	0	0.23% of 0.2 CPU

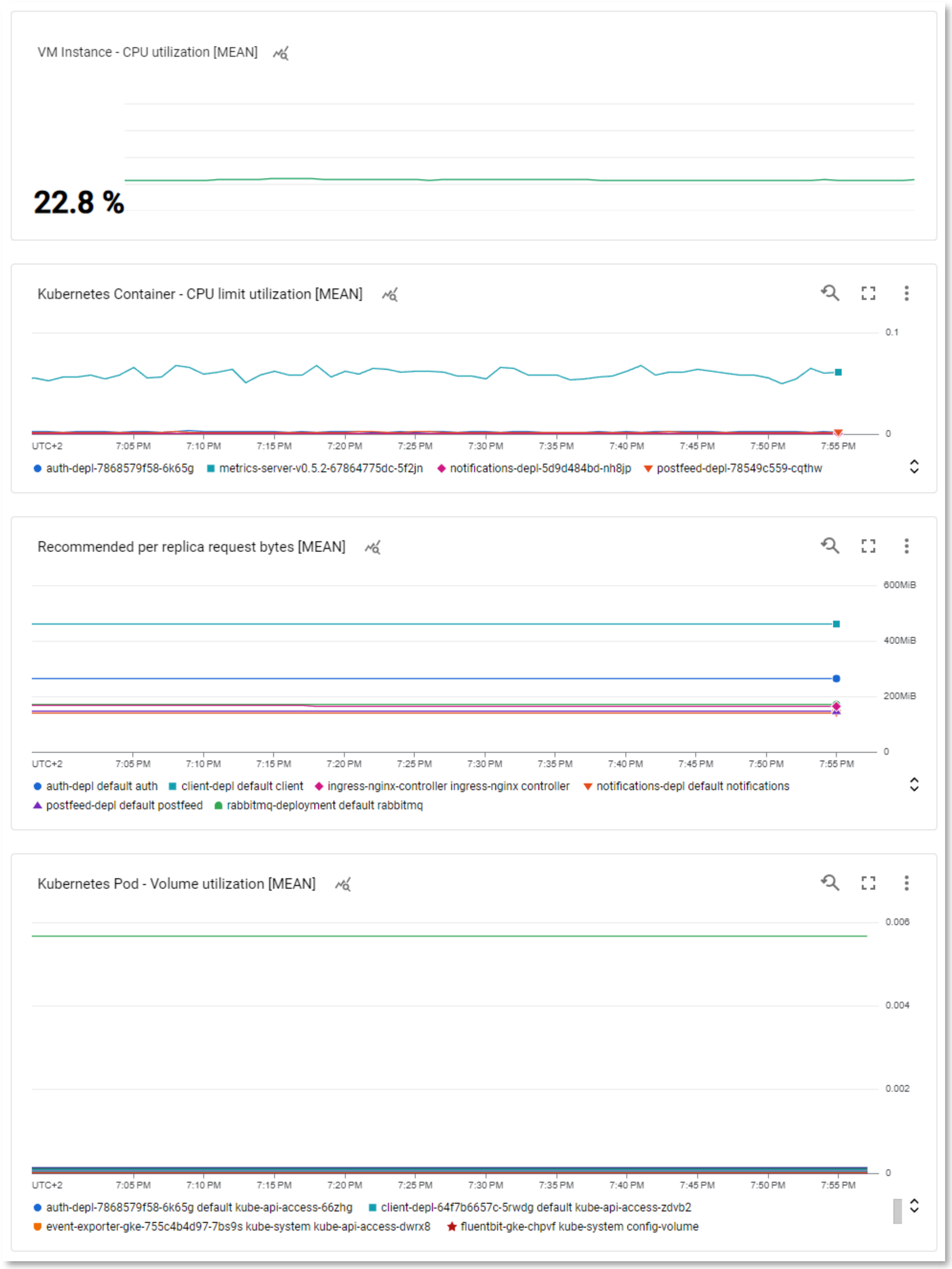
1 – 5 of 12

Pods No active alerts · 0 pods with active alerts · VIEW ALL

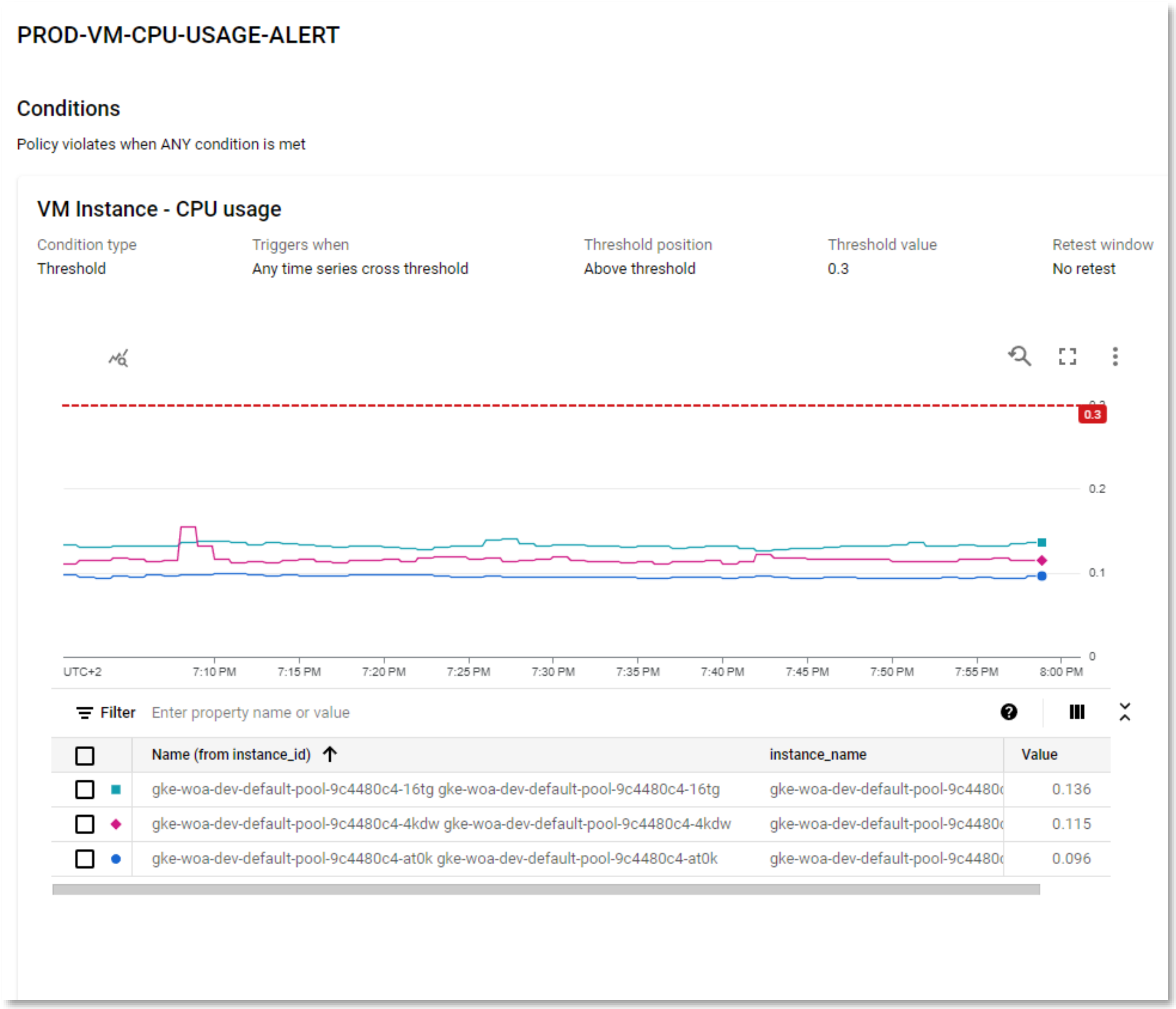
Name	Alerts	Labels		Container restarts	Error logs	CPU utilization
auth-depl-7868579f58-...	0	Cluster: woa-dev	+3	0	0	0.31% of 0.2 CPU
client-depl-64f7b6657...	0	Cluster: woa-dev	+3	0	0	0 CPU
notifications-depl-5d9...	0	Cluster: woa-dev	+3	0	0	0.01% of 0.2 CPU
postfeed-depl-78549c...	0	Cluster: woa-dev	+3	0	0	0.23% of 0.2 CPU

A default GKE monitoring Dashboard displaying metrics about the cluster, the namespaces, the nodes, the workloads, the services and the pods.

It however offers the possibility to create a custom dashboard as well as set alerting on whichever metric is desired



A custom dashboard displaying metrics about the VM Instance on which the Kubernetes cluster is being hosted on as well as some other metrics about the cluster itself and the pods inside.



PROD-VM-CPU-USAGE-ALERT

Conditions

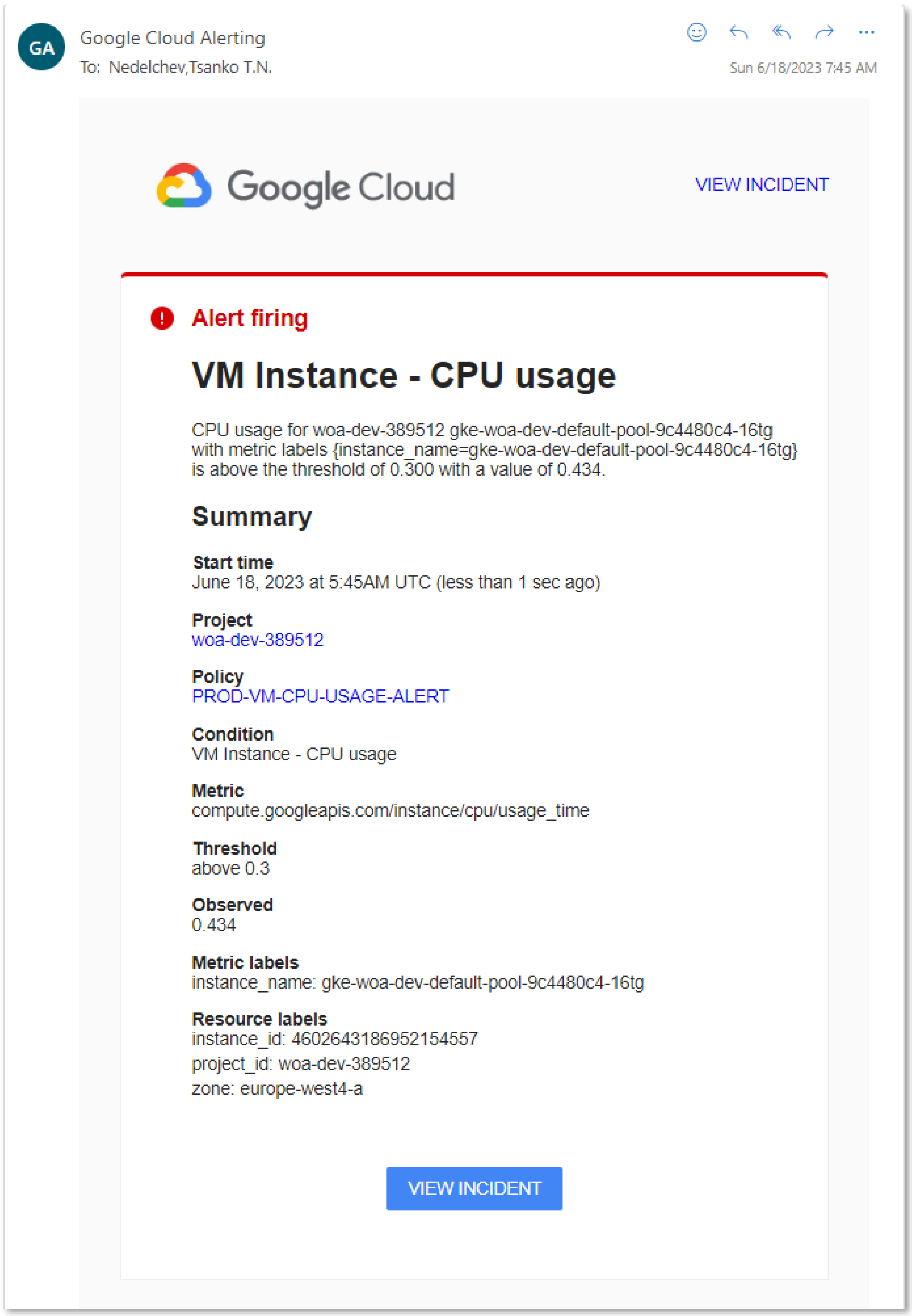
Policy violates when ANY condition is met

VM Instance - CPU usage

Condition type	Triggers when	Threshold position	Threshold value	Retest window
Threshold	Any time series cross threshold	Above threshold	0.3	No retest

		Name (from instance_id)	instance_name	Value
☐	■	gke-woa-dev-default-pool-9c4480c4-16tg gke-woa-dev-default-pool-9c4480c4-16tg	gke-woa-dev-default-pool-9c4480	0.136
☐	◆	gke-woa-dev-default-pool-9c4480c4-4kdw gke-woa-dev-default-pool-9c4480c4-4kdw	gke-woa-dev-default-pool-9c4480	0.115
☐	●	gke-woa-dev-default-pool-9c4480c4-at0k gke-woa-dev-default-pool-9c4480c4-at0k	gke-woa-dev-default-pool-9c4480	0.096

Alerting can easily be set by choosing the metric, defining the threshold that should be checked and the conditions of the alert. I have set my alerts to trigger if the CPU usage passes the threshold for over 2 minutes straight in order to minimize false positive alerting. I have also setup a channel that allows the alerts to send me an email notification straight to Outlook.


Google Cloud Alerting

To: Nedelchev,Tsanko T.N.

Sun 6/18/2023 7:45 AM

Alert firing

VM Instance - CPU usage

CPU usage for woa-dev-389512 gke-woa-dev-default-pool-9c4480c4-16tg with metric labels {instance_name=gke-woa-dev-default-pool-9c4480c4-16tg} is above the threshold of 0.300 with a value of 0.434.

Summary

Start time
June 18, 2023 at 5:45AM UTC (less than 1 sec ago)

Project
woa-dev-389512

Policy
PROD-VM-CPU-USAGE-ALERT

Condition
VM Instance - CPU usage

Metric
compute.googleapis.com/instance/cpu/usage_time

Threshold
above 0.3

Observed
0.434

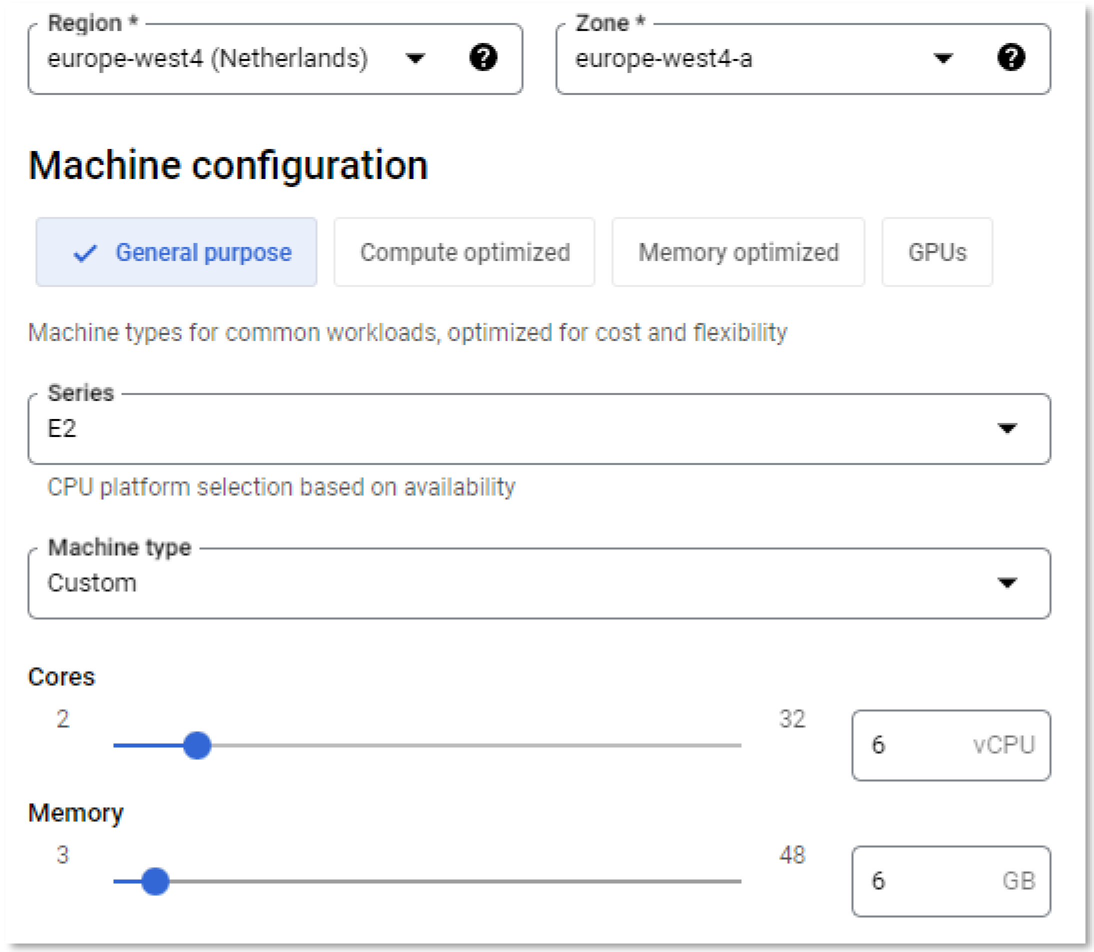
Metric labels
instance_name: gke-woa-dev-default-pool-9c4480c4-16tg

Resource labels
instance_id: 4602643186952154557
project_id: woa-dev-389512
zone: europe-west4-a

An alert that is triggered as a result of high cpu usage during testing of the environment with JMeter

Billing

For the cluster I'm running which is a general purpose E2 cluster with 6 virtual CPUs and 6gb of RAM



Region *: europe-west4 (Netherlands)

Zone *: europe-west4-a

Machine configuration

Machine types for common workloads, optimized for cost and flexibility

Series: E2

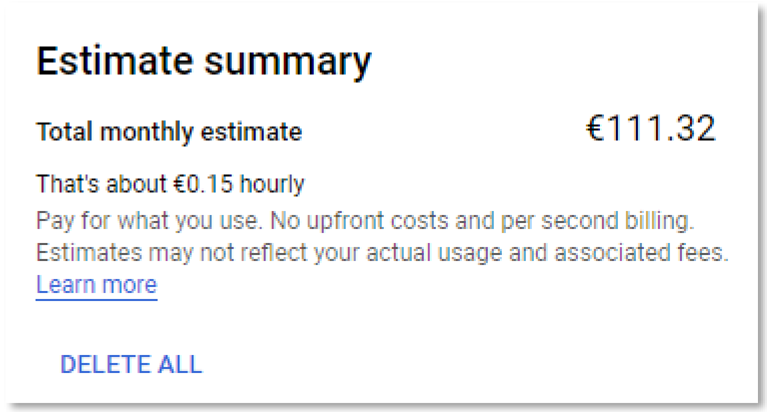
CPU platform selection based on availability

Machine type: Custom

Cores: 2 – 32: 6 vCPU

Memory: 3 – 48: 6 GB

I would have to pay an estimated cost of $111.32 per month or $0.15 per hour if I didn't have the free subscription.



Estimate summary

Total monthly estimate €111.32

That's about €0.15 hourly

Pay for what you use. No upfront costs and per second billing. Estimates may not reflect your actual usage and associated fees. Learn more

Conclusion

In conclusion, google cloud offers a lot of possibilities at a good price and supports all types of automated functionality to make the development process a breeze.
Currently I have a Kubernetes cluster running 3 microservices and a client application. Furthermore it has an ingress-nginx load balancer configuration that exposes the application to the outside world through a personal domain name. Upon an automated deployment the Kubernetes manifests pull the docker images through Google Artifact Registry where they are stored safe from the outside world. I have setup monitoring to the cluster having access to a few pre-generated dashboards and one custom one. Moreover, I have setup alerting to send me an email notification when the metrics reach a value above a certain threshold for a set period of time.