Infrastructure and cloud environment Document

DOCUMENTATION

Fontys University Of Applied Sciences

Tsanko Nedelchev

Contents

[Introduction 2](#_Toc138011433)

[Kubernetes Cluster Setup 3](#_Toc138011434)

[Google Artifact Registry 5](#_Toc138011435)

[Image Management Workflow 6](#_Toc138011436)

[The Kubernetes Manifest Files 7](#_Toc138011437)

[Monitoring and Alerting 8](#_Toc138011438)

# 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

A screenshot of a computer

Description automatically generated with medium confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

A black background with white text

Description automatically generated with low confidence

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

A screenshot of a computer

Description automatically generated

# Google Artifact Registry

A screenshot of a computer

Description automatically generated with medium confidence

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

A picture containing text, screenshot, diagram, font

Description automatically generated

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.

A picture containing text, screenshot, font, black

Description automatically generated

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

A black screen with white text

Description automatically generated with low confidence

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

|  |  |
| --- | --- |
| A screen shot of a computer program  Description automatically generated with low confidence | 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.

|  |  |
| --- | --- |
| A screenshot of a computer  Description automatically generated with medium confidence | 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 screenshot of a computer

Description automatically generated with low confidence

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.

A screenshot of a computer

Description automatically generated with medium confidence

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.

A picture containing text, electronics, screenshot, software

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

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.

A picture containing text, screenshot, font

Description automatically generated

# 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.