

Cloud Architecture / Infrastructure

Table of Contents

[Diagram 3](#_Toc188202874)

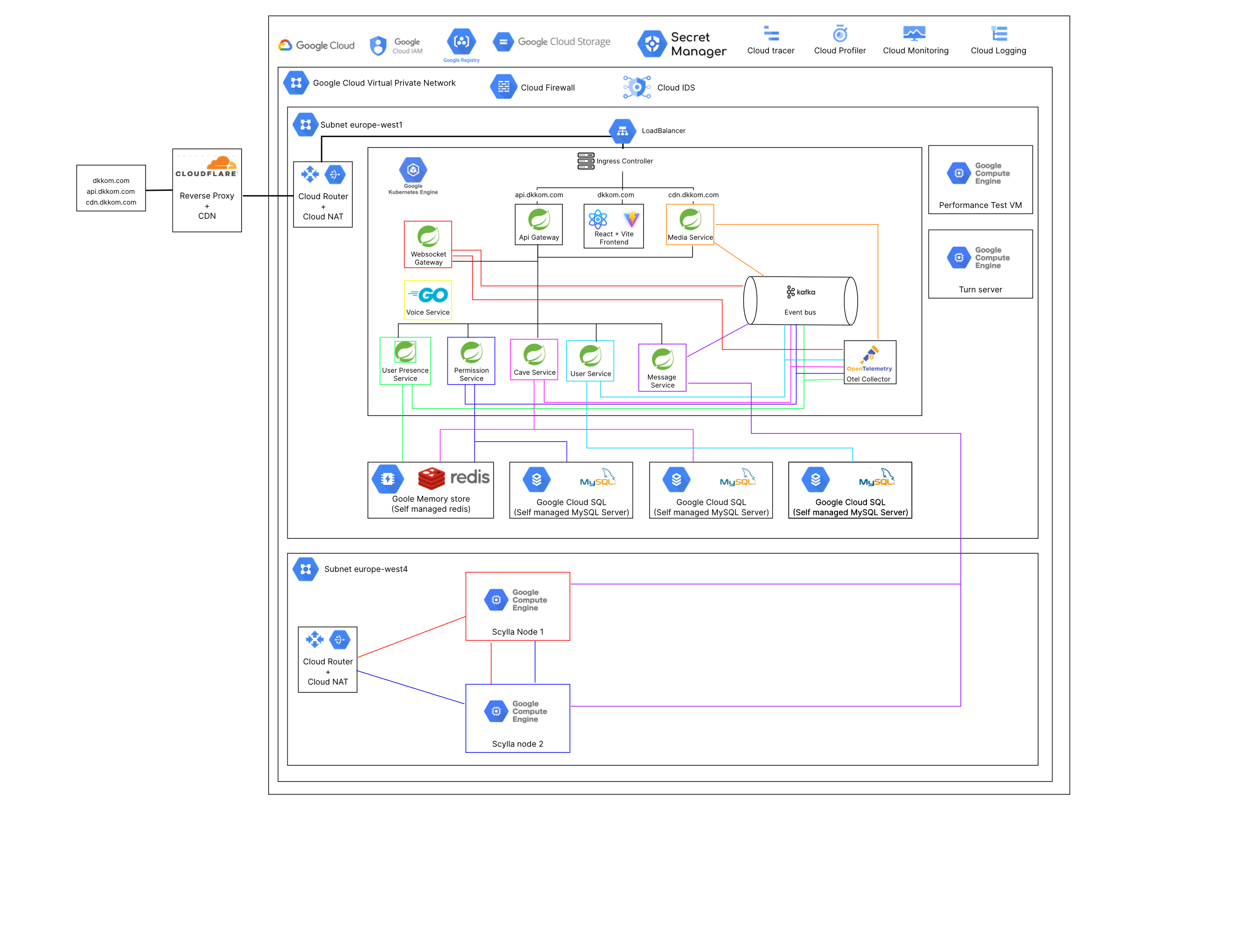
[Breakdown 3](#_Toc188202875)

[Cloudflare 3](#_Toc188202876)

[Google cloud 4](#_Toc188202877)

[Conclusion 18](#_Toc188202878)

# Diagram



## Breakdown

For this cloud architecture there are two main components, google cloud as the actual main cloud provider where the services are being hosted and Cloudflare that is acting as not just a secure reverse proxy but also as CDN for static content.

### Cloudflare

Cloudflare plays a crucial in this setup since all requests are routed through their servers. Cloudflare out of the box comes with a lot of perks, but for me I mainly look for two ones: CDN properties and security.

Cloudflare’s CDN plays a very helpful in my application. One of the functionalities I have is the capability to send pictures for this is very helpful to have this image available as fast as possible worldwide. Whenever my backend service serves an image, Cloudflare caches it and distributes it across their network making it so that whenever this image is requested again instead of pulling from my backend It pulls from their CDN.

The most beneficial aspect of Cloudflare by far is their security. Since every request and connection is routed through their servers they analyze and block all kinds of unauthorized or malicious traffic, making it a great option for first contact. They have amazing free services like: WAF, Bot protection, DDoS protection.

A screenshot of a computer

Description automatically generated

Cloudflare’s DNS configuration for my domains, also where I configure the proxy behavior of the requests. Currently I only interact with Cloudflare’s DNS changing UI, but the security features are also in use, but they are being used under the hood so I cannot demonstrate that in a visual way.

### Google cloud

Google cloud is main cloud provider where I host all components of my application. I choose google cloud because apart from being an industry standard also gives 270 euros worth of credits which is by far the most out of any cloud provider at this level, allowing me to almost fully explore its resources.

From google cloud I make use of the following resources:

* Google Router
* Google NAT
* Google VPC (Private networks)
* Google Cloud storage
* Secret Manager
* Memory Storage
* Google Cloud SQL
* GKE (Google Kubernetes cluster)
* Google Cloud IAM
* Load Balancer
* Artifact Registry
* Cloud Tracer
* Cloud Profiler
* Cloud Logs
* Cloud monitoring
* Cloud IDS (Intrusion Detection system)
* Cloud Firewall
* Cloud compute engine

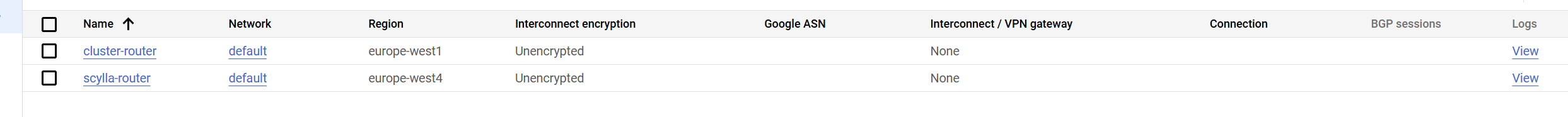
#### Cloud Router and Google NAT

These two services are crucial for my applications accessibility. Almost all my components are in a private network, so they cannot access the internet. I have two use cases where I use them. I have combination of Router and NAT for my cluster on my europe-west1 subnet and another combination on my subnet europe-west4 for my Scylla cluster. These use cases require access to the internet to download some dependencies

A screenshot of a computer

Description automatically generated

*NAT overview*



*Router overview*

#### Cloud VPC

This is also a very important piece of my architecture. Almost all my components belong to my main network. Inside this network I have multiple subnetworks, and I make use of the europe-wes1 and west4.

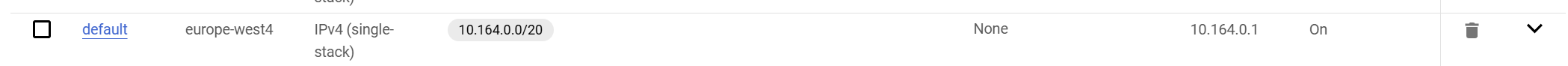
A close-up of a computer screen

Description automatically generated

*VPC Main network*



*Europe-west1 subnet*



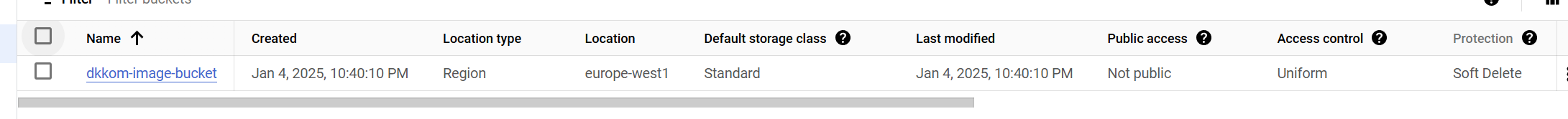
*Europe-west4 subnet*

#### Google Cloud Storage

I have two use cases for this service. The first one is that I use it store all the images and file uploads from the users of my application.

The second is for storing the state of my terraform runs.

This service is great because it gives me a close to a full managed filed system with great access and huge storage capacity and scalability.



A screenshot of a computer

Description automatically generated

*Cloud storage overview*

#### Secret Manager

Almost all my services interact with an external source that requires some form of credentials to authenticate with, it is very important that these credentials are not exposed or hardcoded in any place publicly accessible. For this I store most of my important secrets in this manager where I can later easily integrate with my Kubernetes cluster allowing access to whichever service requires it.

A screenshot of a computer

Description automatically generated

*Secret manager panel*

#### Memory storage

This service is a wrapper on a managed Redis server. I make use of Redis a lot throughout my services, so it makes sense to make use of this service. This service not only gives a fully managed and secure Redis instance, but also gives the capability to create a complete Redis cluster allowing making it very scalable if necessary.

A screenshot of a computer

Description automatically generated

*Redis instance in memory store*

#### Google Cloud SQL (MySQL version)

Similar reason to Memory storage, I make use of this service for its out-of-the-box security and management. Not only that but also its very easy to vertical scale, create full managed backups and create different instances. For this service I not only have normal database server but also a read replica. However, these two are shared across multiple services for cost’s sake because otherwise each service would have its database or a read replica if necessary.

A close-up of a card

Description automatically generated

*MySQL instance with the read replica*

#### GKE (Google Kubernetes Engine)

This is by far the most important piece of the architecture. I make use of Google’s managed Kubernetes cluster to deploy almost all components of my application. The cluster first starts with an ingress controller attached to a load balancer routing all the incoming traffic to the correct places. Within the cluster I have a custom deployment for each service present. Currently this cluster also has one component that in a real production environment I wouldn’t have like this.

This being the Kafka instance. Google also provides a Kafka fully managed service, but for sake of costs I didn’t use it. However, it would have been a crucial part of the infrastructure.

The configuration of this cluster can be found on the terraform document and for the security details they can be found on the Security Document.

A screenshot of a computer

Description automatically generated

*Cluster basic overview*

#### IAM

IAM roles/accounts isn’t a resource like the others, but I consider it to be an important part of the application because this allows me to create custom accounts for accessing the necessary resources and nothing more which contributes a lot for security.

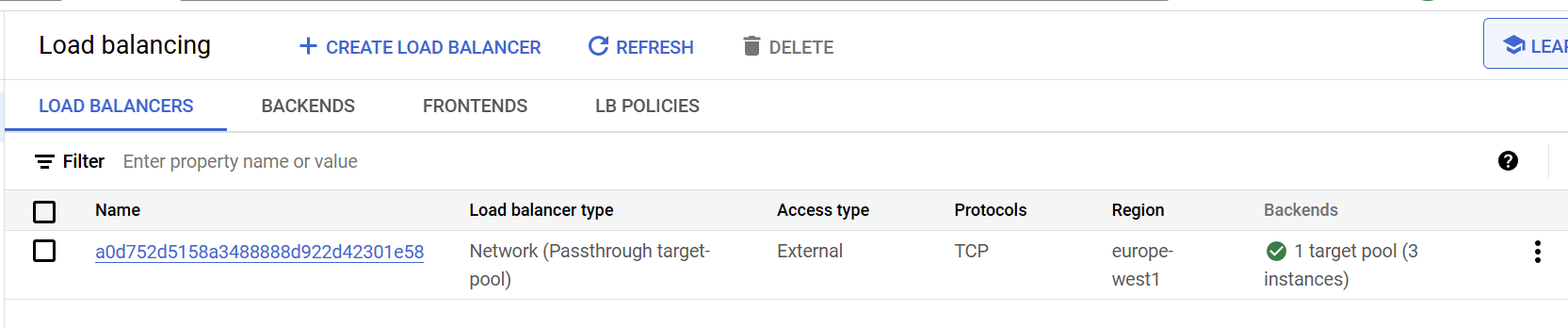
A screenshot of a computer

Description automatically generated

*IAM Accounts example*

#### Load balancer

Load balancer is created automatically when I create the ingress controller for the cluster, but since the load balancer itself it’s also a provided resource it makes sense to be mentioned.



*Load balancer*

#### Cloud Artifact Registry

The artifact registry is where I store all my docker images. These images are used by the Kubernetes cluster to create the pods. For this resource I also have image scanning enabled which is better explained in the Security Document.

*A screenshot of a computer

Description automatically generated*

*Artifact Registry dashboard*

#### Cloud Tracer

Cloud tracer is Google’s implementation of a tracing tool. The setup for this tracer is explained on the CI/CD Document, but as a sum up, I have an open telemetry agent that I connect to my java application that periodically sends metric data to a collector and this collector sends them to Google’s servers. This tracer was incredibly helpful when I was performing performance tests.

A screenshot of a computer

Description automatically generated

*Tracer Dashboard*

#### Cloud Profiler

Like the Cloud tracer, cloud profiling is tool google provides for profiling my services and look for inconsistencies at a lower level. More details can be found on the CI/CD document. This was also a very interesting tool to investigate even though most results values originate from frameworks and not my code.

A screenshot of a computer

Description automatically generated

*Profiler Dashboard*

#### Cloud Logs

Cloud logs is Google’s centralized logging solution. Google captures logs from all my google cloud resources and my own services inside Kubernetes. These logs are then available in custom dashboards.

A screenshot of a computer

Description automatically generated

*Logs explorer dashboard to view and filter all logs of my environment*

#### Cloud monitoring

Cloud monitoring by itself is the combination of the logs, traces, profiling and some others. However, I’m mentioning it because within this monitoring environment I can create custom dashboards to see any metrics I want and organize in any way I want which is very useful.

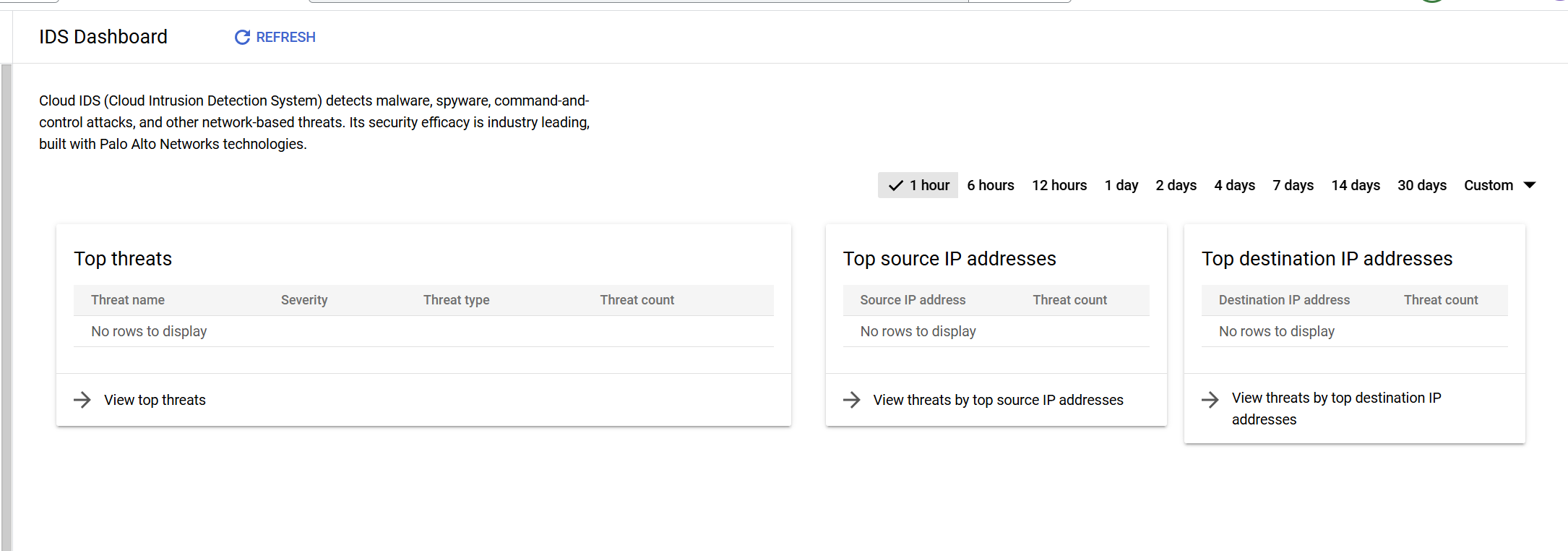
A screenshot of a computer

Description automatically generated

*Custom monitoring dashboards*

#### Cloud IDS (Intrusion Detection system)

This is google’s implementation on an IDS. I go more into detail on my Security Document, but basically, I have an IDS agent on my VPC network that mirrors the packets analyzes them and flags any malware, spyware, etc.



*IDS Dashboard*

*A screenshot of a computer

Description automatically generated*

*IDS Mirrored networks logs*

#### Cloud Firewall

As the name suggests this the firewall that comes by default when I have a VPC network. I’m mentioning it because the firewall has some custom rules for some of my components like the turn server and my Scylla Cluster.

A screenshot of a computer

Description automatically generated

*Firewall Rules dashboard*

#### Cloud compute engine

Compute engine is google’s Virtual machine solution. This is indirectly used by Kubernetes because the node pool is just a group of virtual machines. The reason I’m specifying this is because I don’t have just the automatically generated nodes, but also have some custom VM’s that run my turn server and my Scylla DB cluster.

A screenshot of a computer

Description automatically generated

*All running VM instances of my production environment*

# Conclusion

Before this semester I never touched a cloud platform, so I think I made very good progress in the field. I managed to explore most of the main resources a provider like google can offer and managed to integrate all of them with my system. Although this was fun to do there are certain solutions that I would prefer to integrate myself or have a more hybrid approach, this mainly being the observability part. I would have liked to implement more complex and complete observability like Prometheus and Grafana. I say this specially because I think using the java agents(profiling and tracing) on the application makes the application start very slow in comparison. Maybe using Prometheus would make the entire performance of the pod better, but this is just speculation. Also, as a side note I realized that a cloud platform is probably one of the most complex software systems in the world and trying to think about how they do some of the things they do is a very interesting mental exercise.