**Deployment Of My Portfolio React app in GKE**

Contents

[**Use Case:** 1](#_Toc183277779)

[Deploy IaC via terraform 1](#_Toc183277780)

[Iac Terraform Folder Structure: 1](#_Toc183277781)

[**Image Building and Pushing to Artifact Registry:** 2](#_Toc183277782)

[**Introduction to app:** 2](#_Toc183277783)

[**Deploy Kubernetes manifests:** 3](#_Toc183277784)

[**Deploy Prometheus and Grafana using Helm:** 4](#_Toc183277785)

[**Deployment Screenshots in GCP:** 7](#_Toc183277786)

# **Use Case:**

* Deploy a react application in GKE and configure Prometheus and Grafana.

Terraform for Infrastructure.

**Pre-requestees:**

Install Node – v16.20.2

Install npm – 8.19.4

Install Terraform – v1.9.8

Docker

gcloud CLI

Clone the below repo:

<https://github.com/venkata-suresh-bukka/my-portfolio.git>

Branch: new-gcp-env

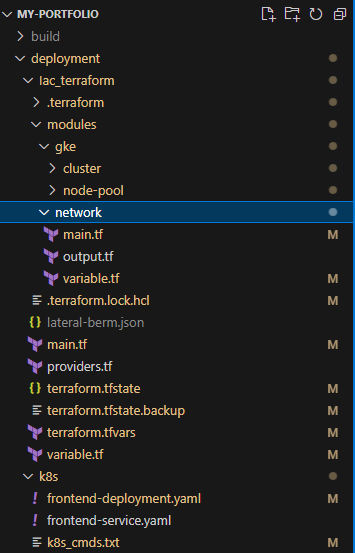
Login to the GCP using below command:

gcloud auth login

# **Deploy IaC via terraform**

1. VPC
2. Subnet
3. Firewalls
4. GKE
5. Nodepool

# **Iac Terraform Folder Structure:**



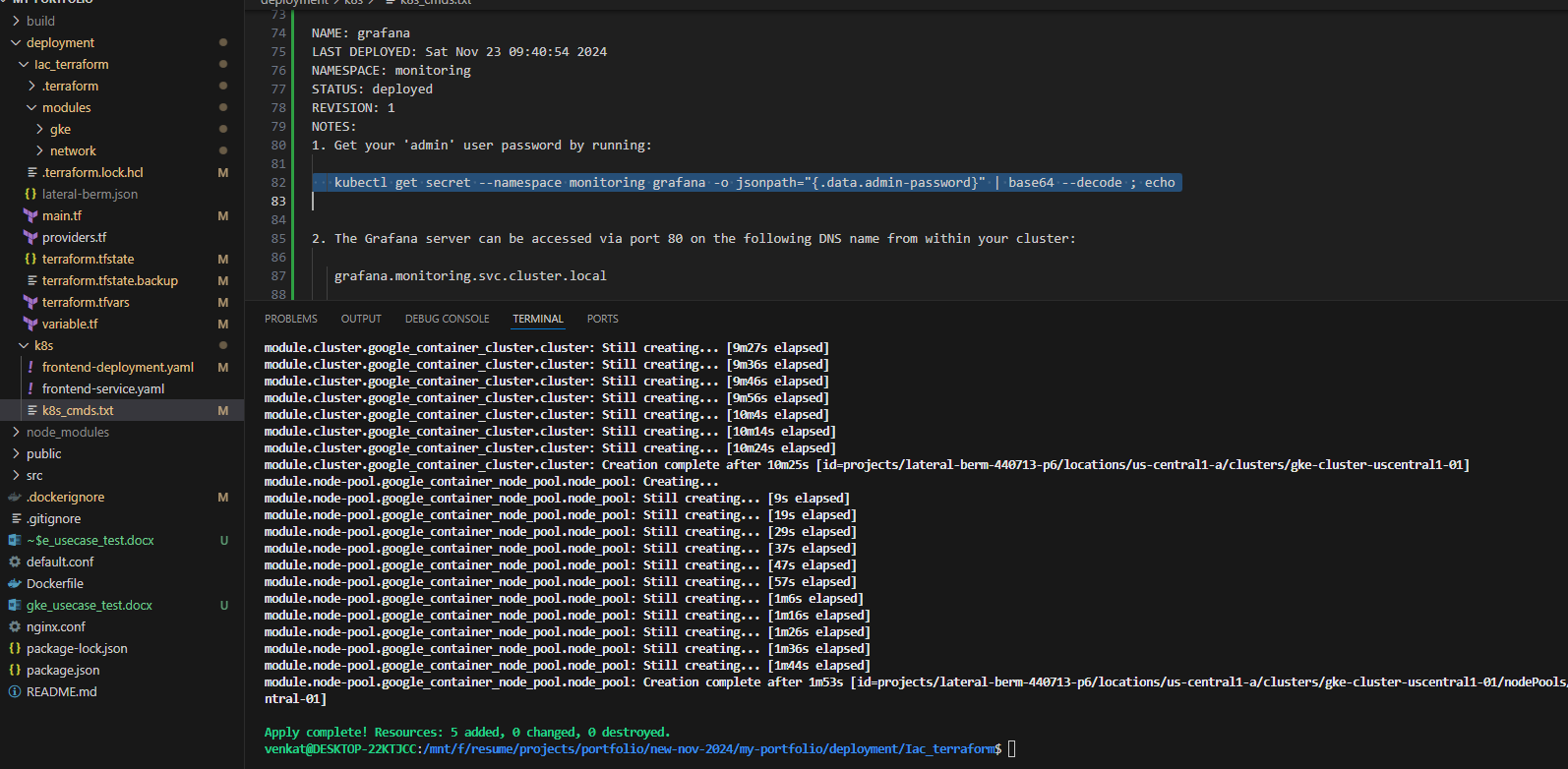
For this use-case modules have been used.

After cloning the repo go-ahead and perform below commands to do the Infra deployment.

* terraform validate
* terraform plan
* terraform apply -auto-approve

**Note**: My gcp account has antifactory repo already so didn’t included in terraform.

Once deployment is done connect to the gke cluster.



# **Image Building and Pushing to Artifact Registry:**

## **Introduction to app:**

A portfolio website which is build using react js.

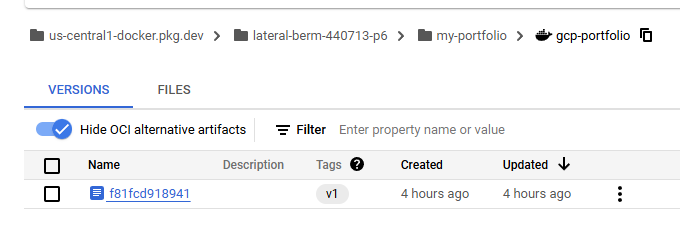


**Step 1:**

Go to the main folder where we can find Dockerfile and build the react application docker image.

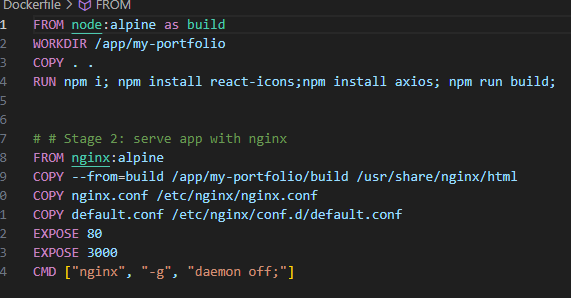
Command:

docker build -t us-central1-docker.pkg.dev/<project-id>/<repo-name>/<image-name>:v1 .



Nginx has been used as revers proxy and static files are being served by it.

Dockerfile:



**Step 2:**

Once image is build push it to artifact registry.

**Command**:

docker push us-central1-docker.pkg.dev/<project-id>/<repo-name>/<image-name>:v1

# **Deploy Kubernetes manifests:**

**Pre-requests:**

Create a secret in the cluster with permissions to pull images from artifact registry to the cluster.

**Command**:

kubectl create secret docker-registry <secret-name> \

--docker-server=us-central1-docker.pkg.dev \

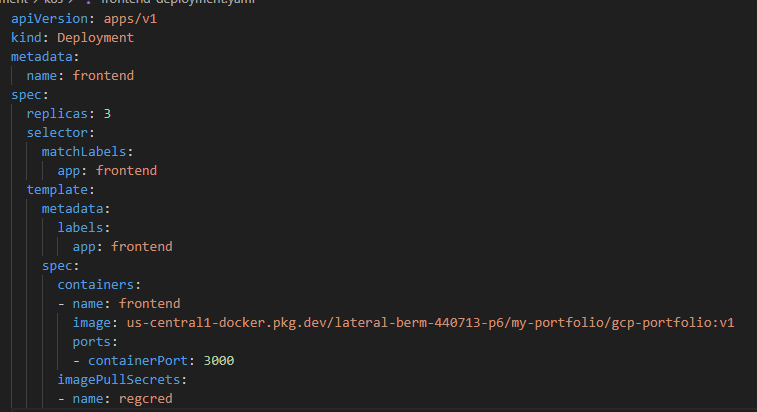
--docker-username=\_json\_key \

--docker-password="$(cat <service-account>.json)"

Step 1:

Apply deployment file with below command.

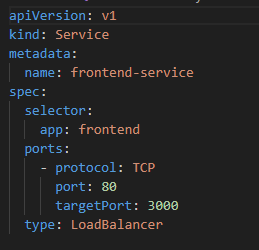
kubectl apply -f frontend-deployment.yaml



**Step 2:**

Create service

kubectl apply -f frontend-service.yaml -- type is LoadBalancer

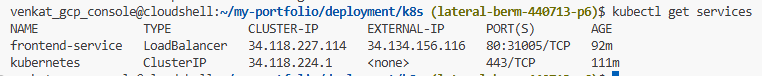


Check the deployment & service with below commands:

Kubectl get pods



Kubectl get svc/services



Now the app can be accessed via External IP of the Load Balancer



# **Deploy Prometheus and Grafana using Helm:**

Step 1:

Install Helm

curl https://get.helm.sh/helm-v3.10.3-linux-amd64.tar.gz -o helm-v3.10.3-linux-md64.tar.gz

tar -zxvf helm-v3.10.3-linux-amd64.tar.gz

sudo mv linux-amd64/helm /usr/local/bin/helm

**Reference**: <https://phoenixnap.com/kb/install-helm>

**Step 2:**

**Create a separate namespace for monitoring and**

kubectl create namespace monitoring

**Add Prometheus, Grafana repos to Helm**

helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>

helm repo add grafana <https://grafana.github.io/helm-charts>

helm repo update

**Run below commands to deploy premetheus and Grafana as services with Type LoadBalancer**

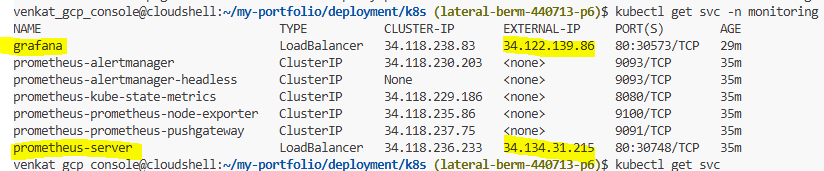
helm upgrade --install prometheus prometheus-community/prometheus --namespace monitoring -f prom.yaml

helm upgrade --install grafana grafana/grafana --namespace monitoring -f grafana.yaml

**Now check the services availability**

kubectl get svc prometheus -n monitoring

kubectl get svc grafana -n monitoring



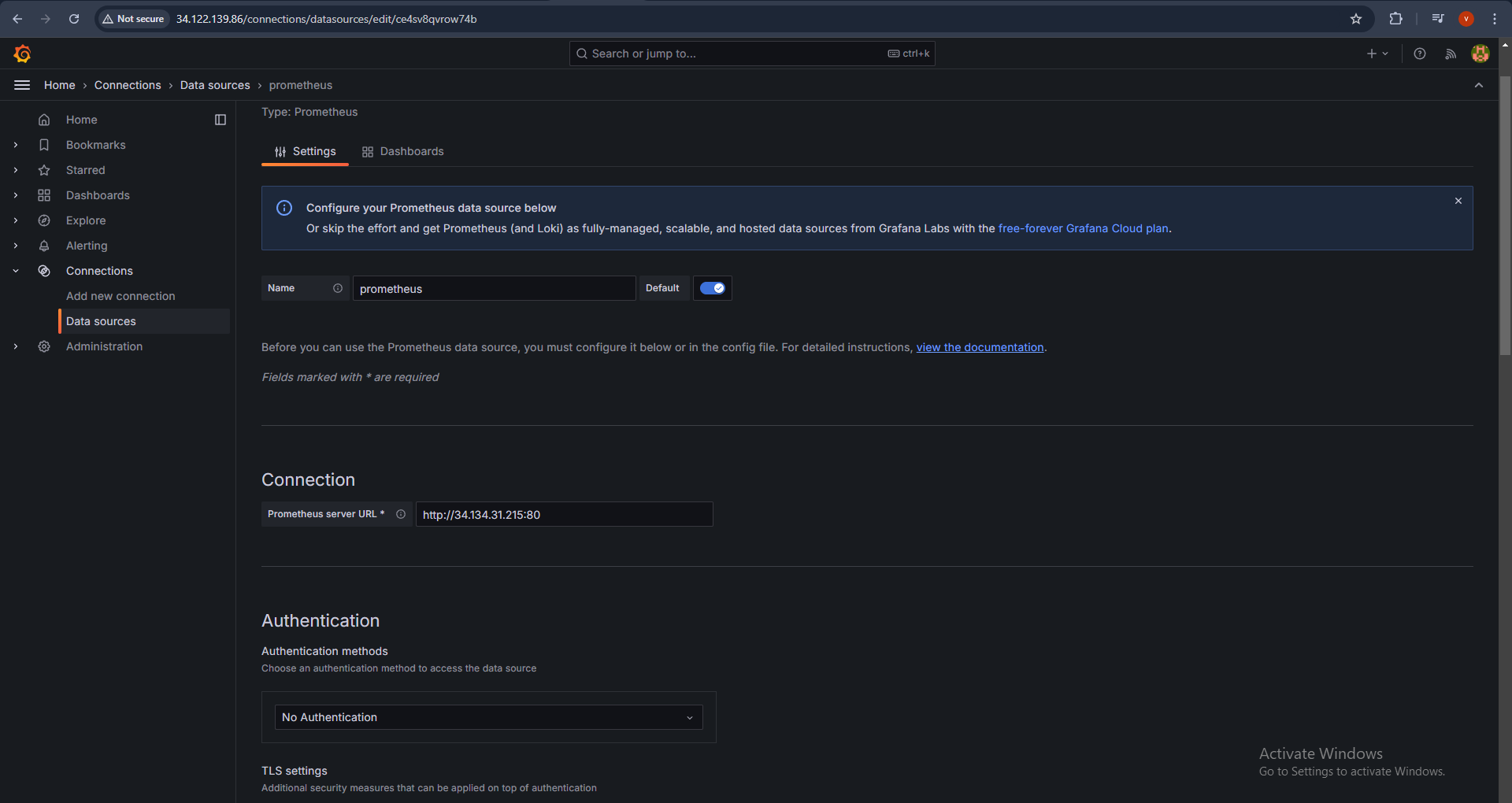
**Access from browser:**

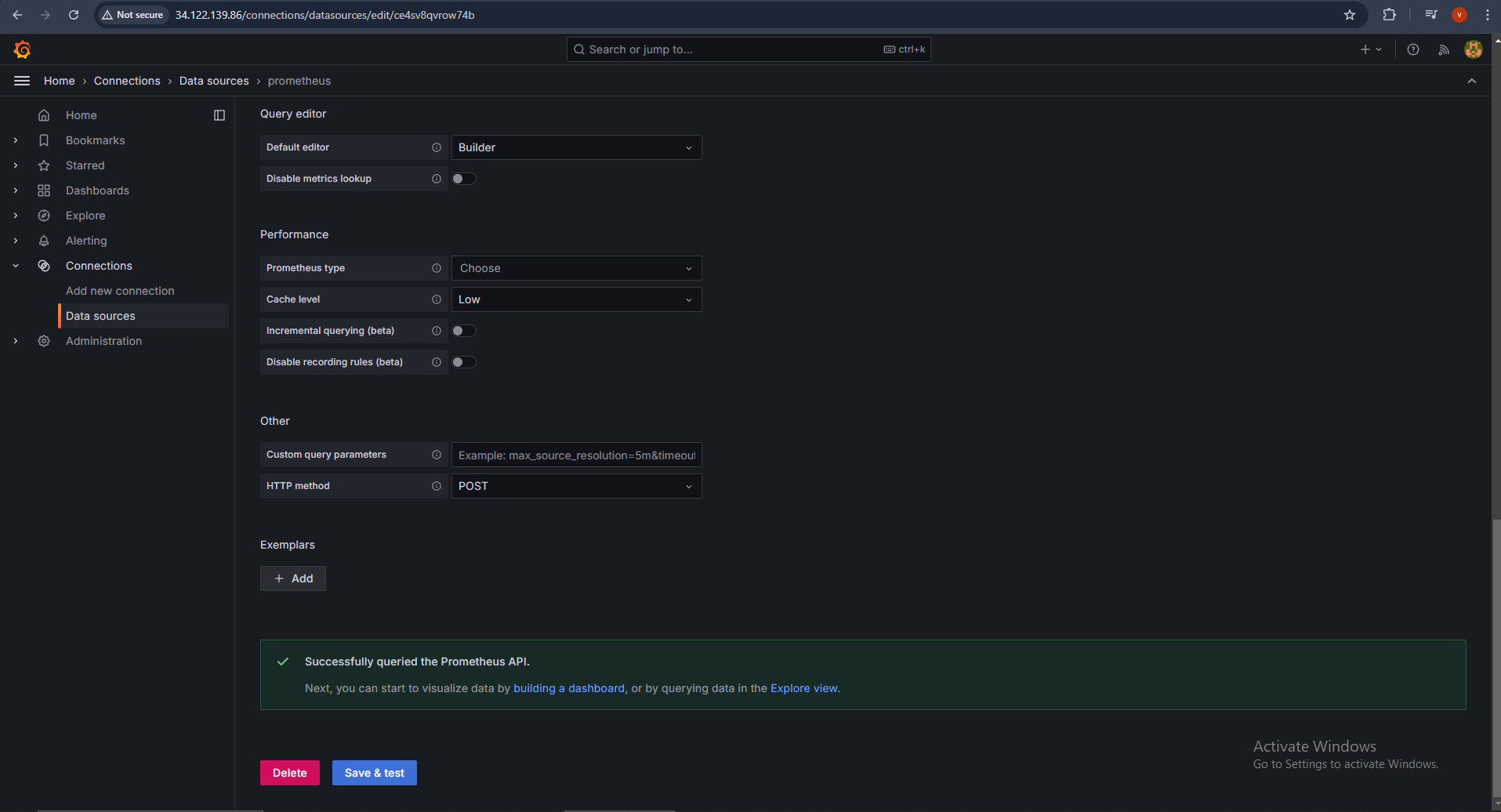
We can get Grafana password from this command

kubectl get secret --namespace <namespace-name> grafana -o jsonpath="{.data.admin-password}" | base64 --decode ; echo

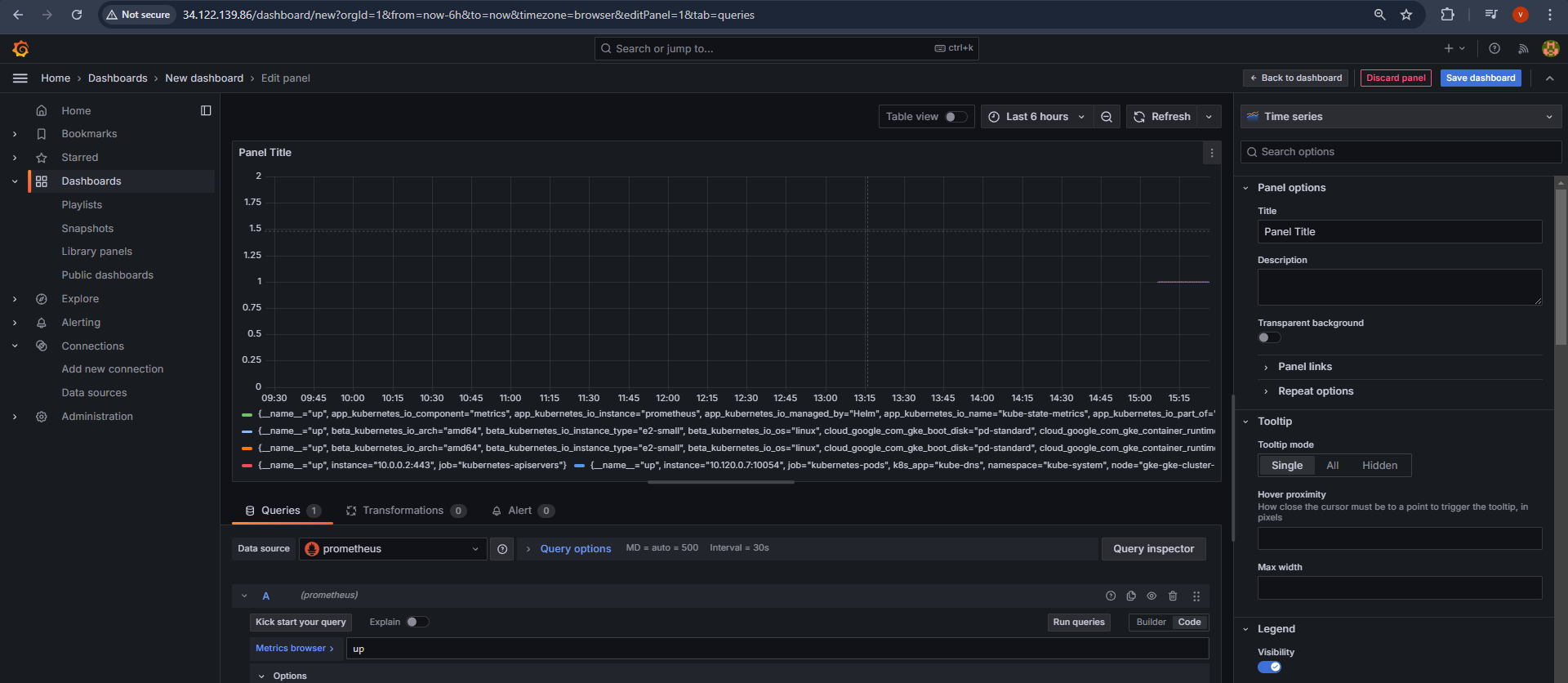
**Grafana**

After logging go to dashboard and connect to the Prometheus server ip in connection details

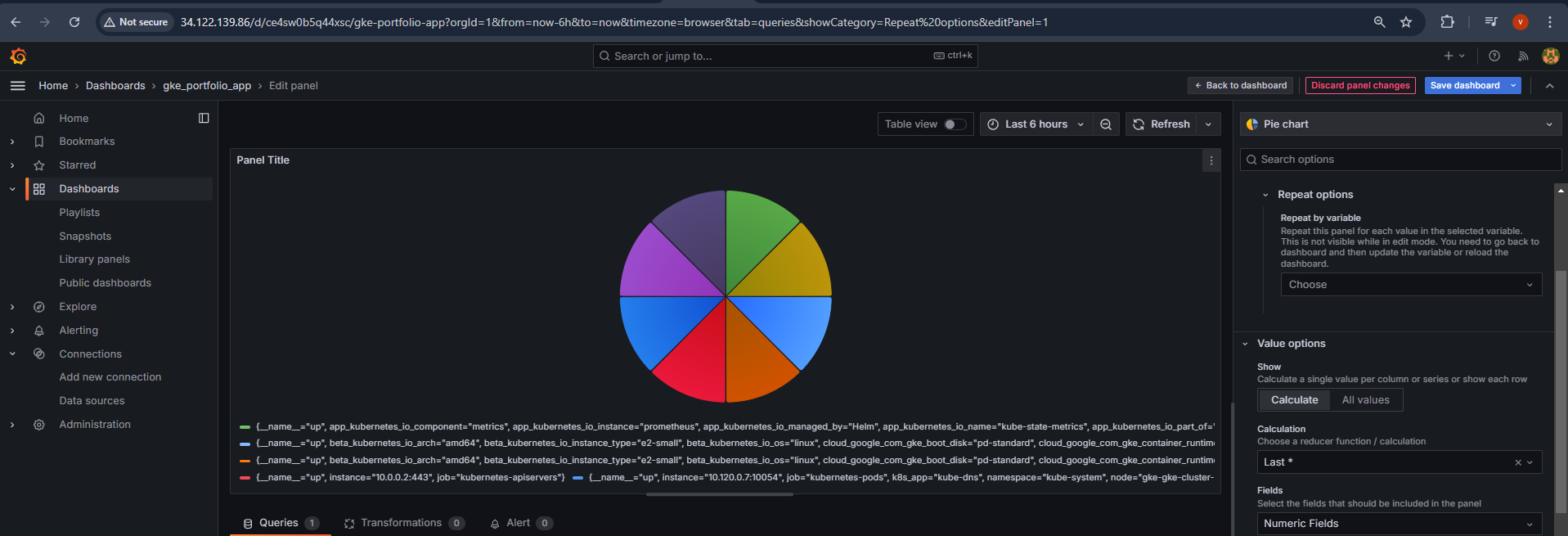




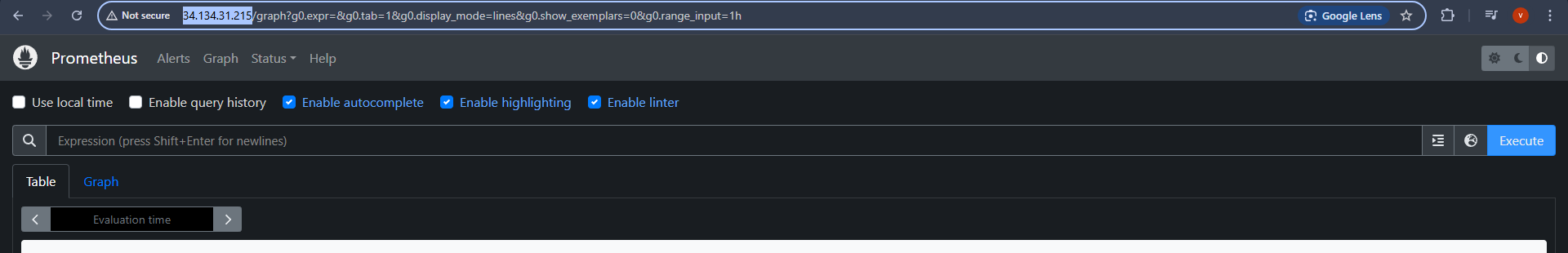
Using sample query “up” in Grafana we see the below info



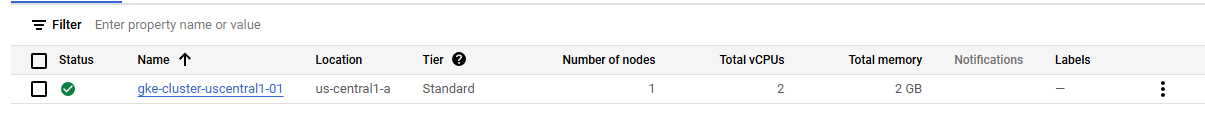
Panel – Pie chart



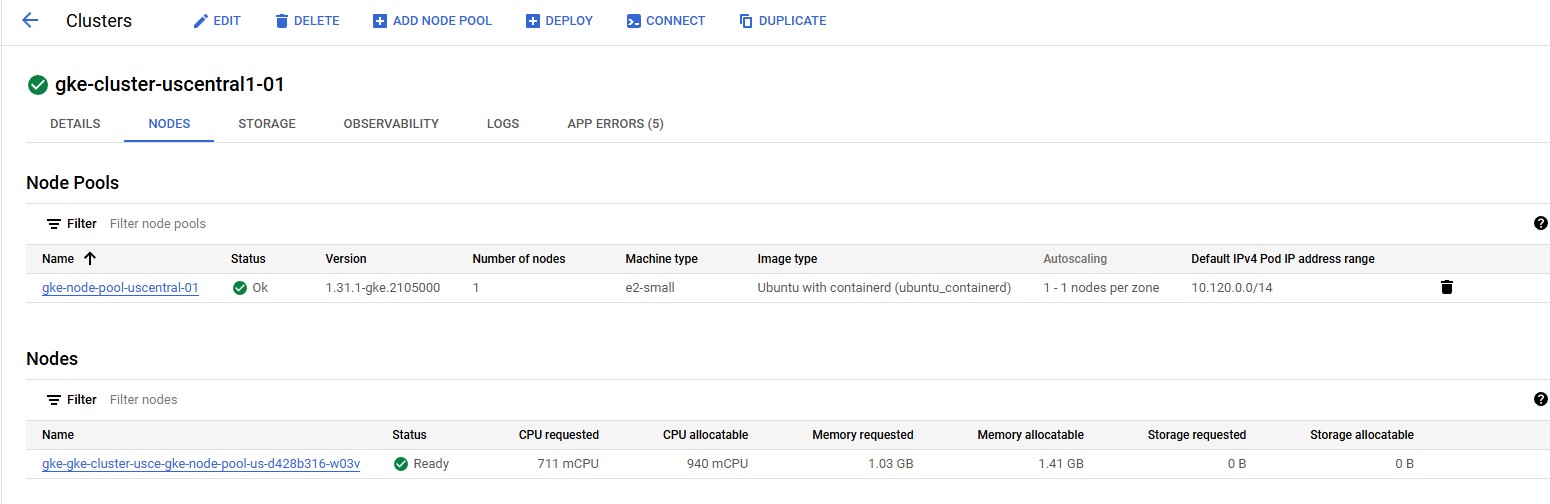
**Prometheus**



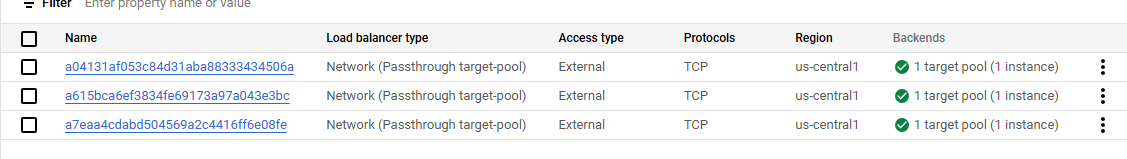
# **Deployment Screenshots in GCP:**



**Node pools**



**Load Balancers for React app, Grafana and Prometheus**



**Note**: As cost is high for GKE I’ve cleaned the deployment and deployed in cloud run for application sample.

<https://gcp-portfolio-524836044132.us-central1.run.app/>