**SOCK SHOP APP DEPLOYMENT WITH KUBERNETES**

**Overview**

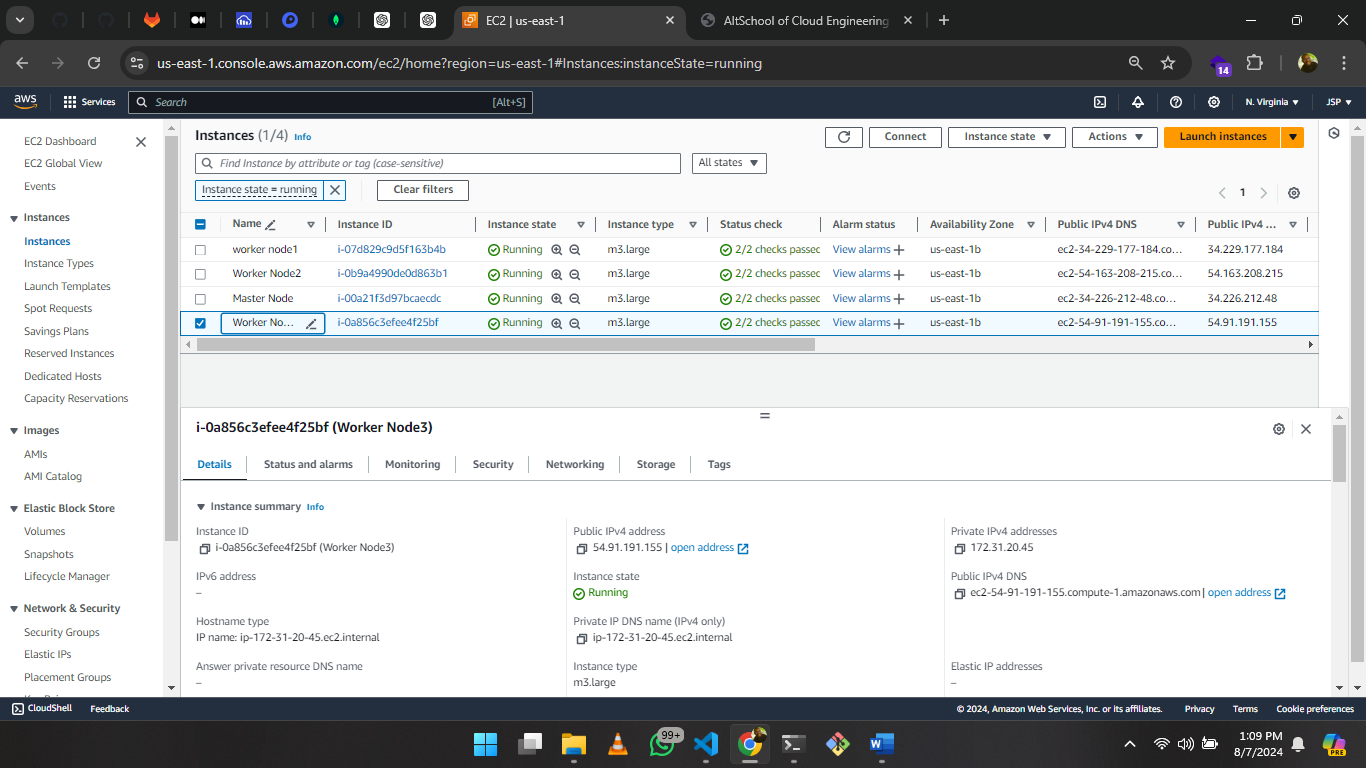
This project demonstrates deploying the Sock Shop microservices application on a Kubernetes cluster using Weave Net and Weave Scope for networking and monitoring, respectively. It also includes setting up Prometheus for monitoring, Alertmanager for alerting, and Fluentd with the ELK stack for logging. The deployment process is automated using a Jenkins pipeline.

**INFRASTURE SETUP**

The infrastructure setup for deploying the Sock Shop application on Kubernetes involves several key steps, all automated through a Jenkins pipeline. Here's a brief overview:

I used the terraform script to create the infrastructure using the command:

terraform apply deploy/kubernetes/terraform/

****

Infrastructure configured creating a three(3) worker nodes and one(1) Master node.

After a successful server creation, I went on to configure kubectl, kubeadm and kubelet on the worker node using the following commands;

 master\_ip=$(terraform output -json | jq -r '.master\_address.value')

    scp -i ~/.ssh/deploy-docs-k8s.pem -o StrictHostKeyChecking=no -rp deploy/kubernetes/manifests ubuntu@$master\_ip:/tmp/

    master\_ip=$(terraform output -json | jq -r '.master\_address.value')

    ssh -i ~/.ssh/deploy-docs-k8s.pem

    ubuntu@$master\_ip sudo kubeadm init > k8s-init.log

    ssh -i ~/.ssh/deploy-docs-k8s.pem ubuntu@$master\_ip "sudo cp /etc/kubernetes/admin.conf /home/ubuntu/"

    ssh -i ~/.ssh/deploy-docs-k8s.pem ubuntu@$master\_ip "sudo chown \$(id -u):\$(id -g) \$HOME/admin.conf"

    grep -e --token k8s-init.log > join.cmd

    ssh -i ~/.ssh/deploy-docs-k8s.pem ubuntu@$master\_ip KUBECONFIG=\$HOME/admin.conf kubectl apply -f https://git.io/weave-kube-1.6

Master node configured in the cluster successfully.

The next step was to to join the worker node to cluster. I was able to do that usingdo that using the following command;

 node\_addresses=$(terraform output -json | jq -r '.node\_addresses.value|@sh' | sed -e "s/'//g" )

    for node in $node\_addresses; do

        ssh -i ~/.ssh/deploy-docs-k8s.pem -o StrictHostKeyChecking=no ubuntu@$node sudo `cat join.cmd`

    done

Setup WEAVE in our hosted environment on AWS

* SSH into the server, then run;

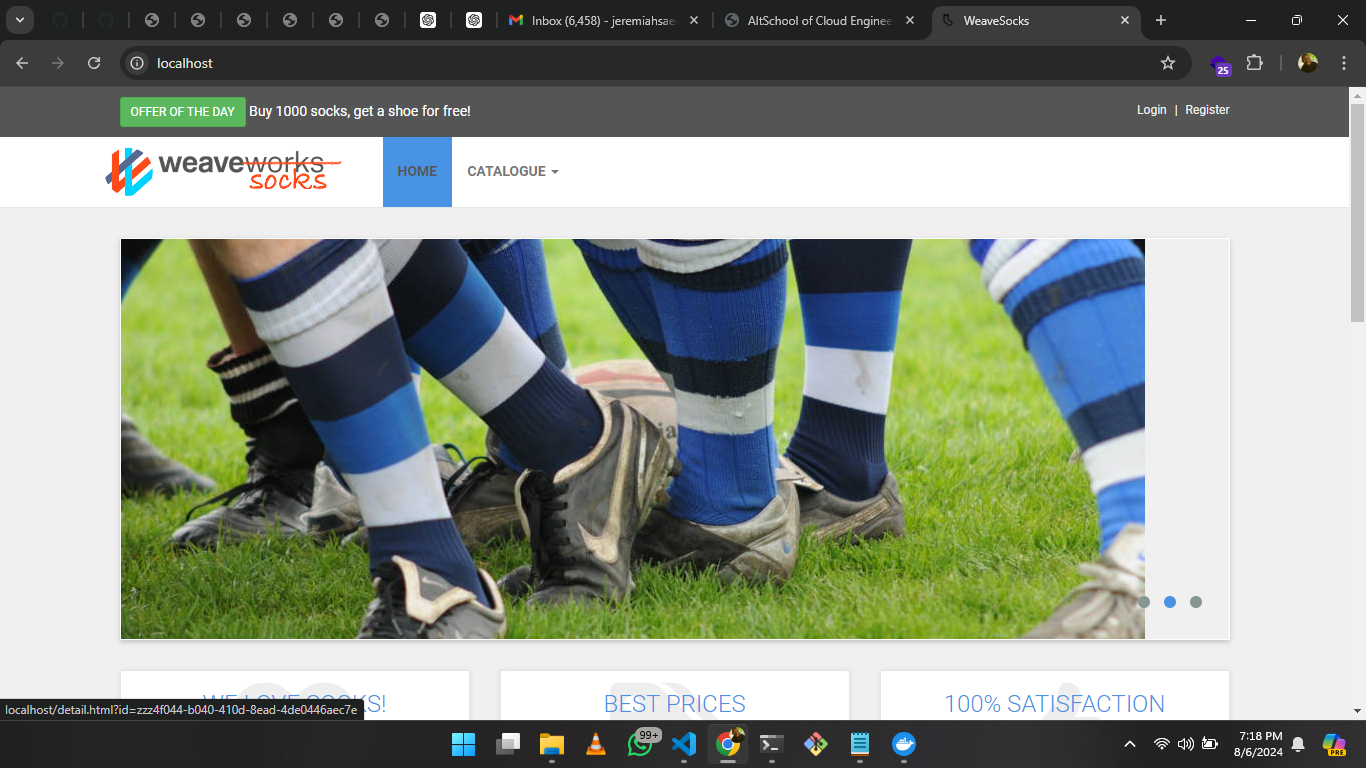
ssh -i ~/.ssh/deploy-docs-k8s.pem ubuntu@$master\_ip kubectl apply -f 'https://cloud.weave.works/launch/k8s/weavescope.yaml?service-token=<token>'

Next in other to deploy the sock shop application I SSH into the master node and ran this command;

 master\_ip=$(terraform output -json | jq -r '.master\_address.value')

    ssh -i ~/.ssh/deploy-docs-k8s.pem ubuntu@$master\_ip KUBECONFIG=\$HOME/admin.conf kubectl apply -f /tmp/manifests/sock-shop-ns.yaml -f /tmp/manifests

Sock shop app was deployed successfully



MONITORING AND ALERTS

In this project, monitoring and alerting are implemented using Prometheus, Alertmanager, and Grafana. This setup ensures that the Sock Shop application is efficiently monitored, and any critical issues are promptly alerted.

To successfully accomplish this task I needed to deploy Prometheus and Grafana.

 kubectl create -f ./deploy/kubernetes/manifests-monitoring

Prometheus and Grafana was successfully deployed and accessible in the below mentioned ports

$ minikube service list | grep prometheus

| monitoring  | prometheus           | http://192.168.99.100:31090 |

```

**#### Grafana**

```

$ minikube service list | grep grafana

| monitoring  | grafana              | http://192.168.99.100:31300 |

After the deployment of Prometheus and Grafana, I used Grafana as a data visualization tool to oversee everythin and to also collect data from the sock shop application.

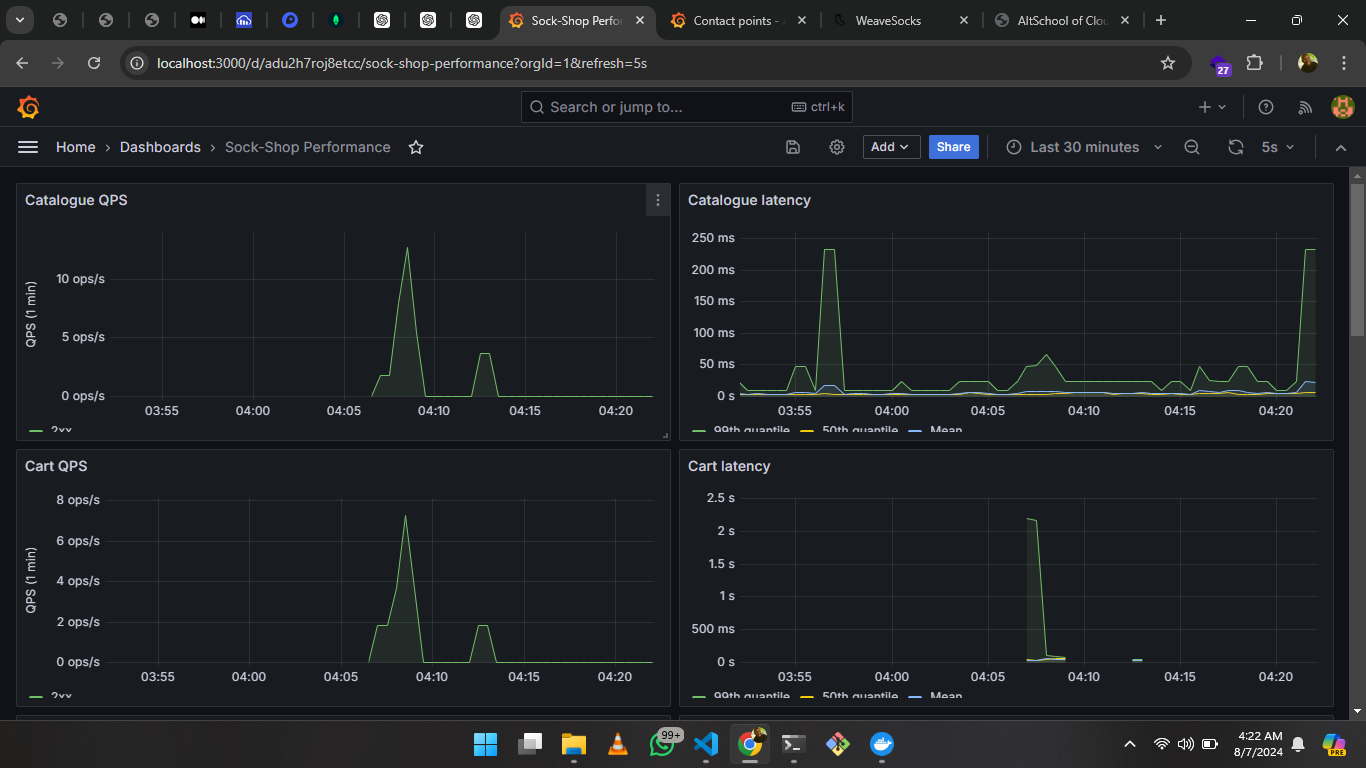


FIG 1.3

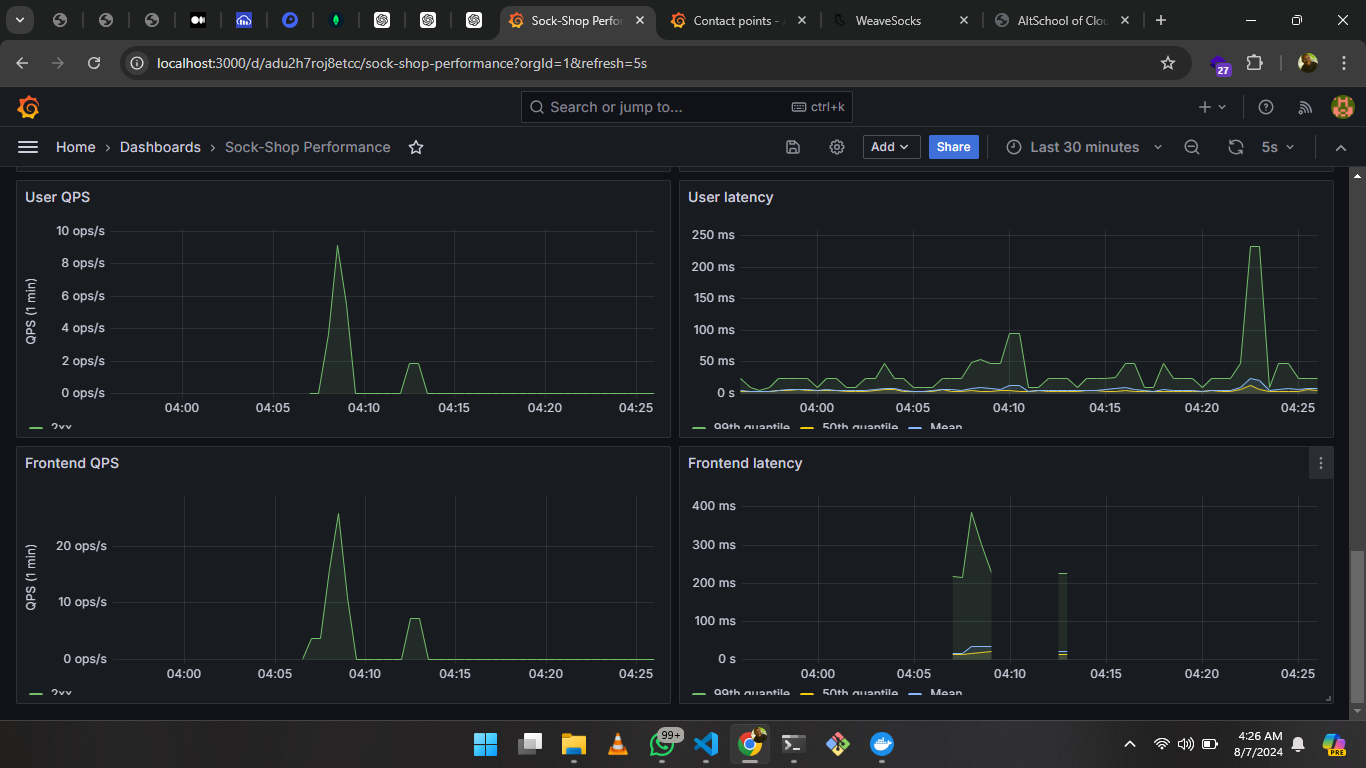


FIG 1.4

Above are images of some of the deliverables monitored by Grafana

Configured Alert rules in the sense that when theres a problem in the cluster an alert will be sent via email.

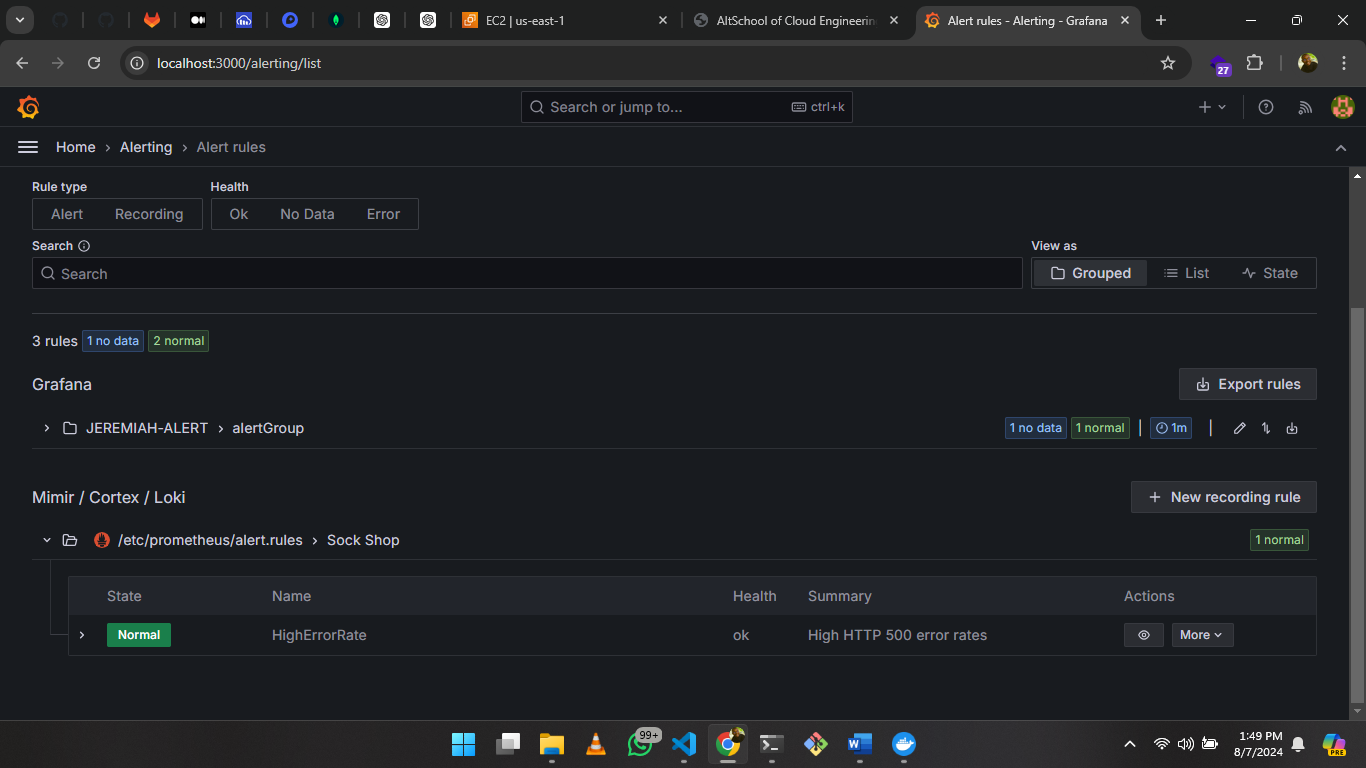


FIG 1.5

Deployed an ingress controller to manage external request

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/cloud/deploy.yaml

**LOGGING**

**Used** Prometheus to collect collect logs for the socks shop application by passing in some queries

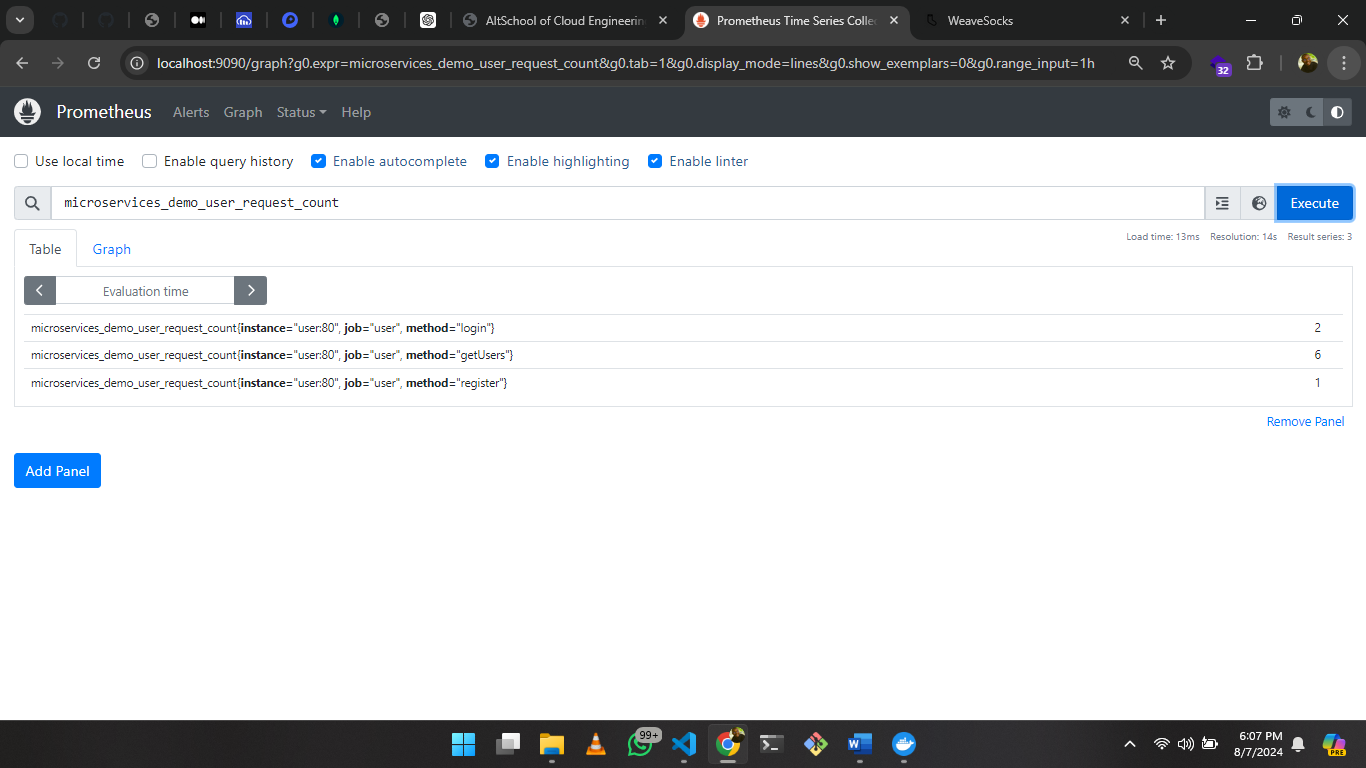


FIG 1.6

Logs collected in table form

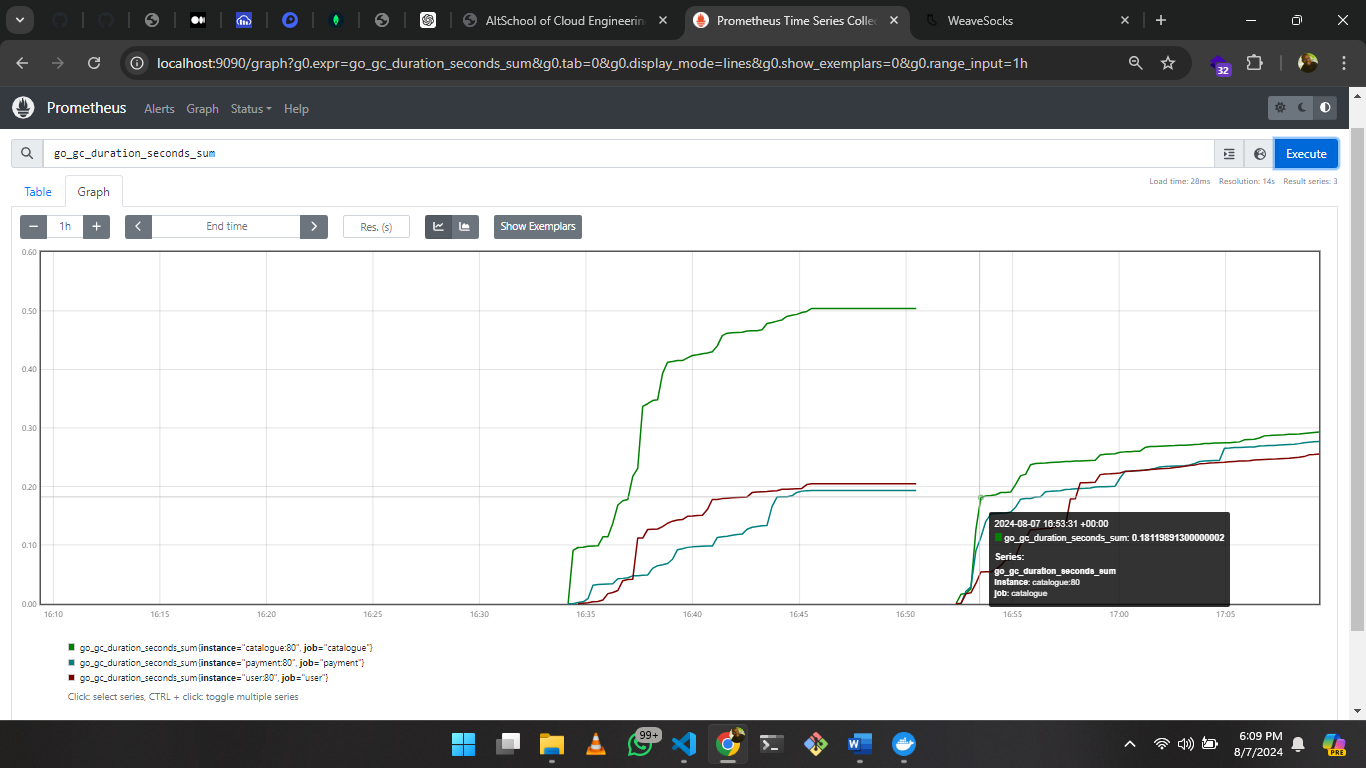


FIG 1.7

Logs collected in graphical format

After the whole deployment I cleaned up the entire work space by running the commands below;

 terraform destroy -force deploy/kubernetes/terraform/

    aws ec2 delete-key-pair -\-key-name deploy-docs-k8s

    rm -f ~/.ssh/deploy-docs-k8s.pem

    rm -f terraform.tfstate

    rm -f terraform.tfstate.backup

    rm -f k8s-init.log

    rm -f join.cmd

PROBLEMS FACED  
Amidst gaining a lot of experience with this project I realized that some of the scripts given in the project was outdated, therefore some functions couldn’t work.

I was able to re-write some of them to get the maximum result required.

CONCLUSION

By implementing this setup, I gained a robust understanding, observability stack that helps maintain the health and performance of a microservices application, leading to improved reliability and faster issue resolution.