Final Task

By:- Nirmal Singh

Task 1 :- A) Bootstrap kubernetes cluster on your laptop using kubeadm

task 1.A.A) I have created the shell script and put all neccessary commands in that file for creating.

1.A.A) KUBEADM INIT

File name:- kubeadmInit.sh

```
#!/bin/sh

#kubeadm init
sudo kubeadm init  --ignore-preflight-errors=IsPrivilegedUser,preflight

#create directory
mkdir -p $HOME/.kube

#copy admin.conf file
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

#add to the super user group
sudo chown $(id -u):$(id -g) $HOME/.kube/config


# create pod network
sudo kubectl apply -f https://docs.projectcalico.org/v3.11/manifests/calico.yaml

# make master node as worker node
kubectl taint nodes --all node-role.kubernetes.io/master-

#Create namespace
kubectl create ns nirmal

# RBAC
kubectl create -f traefik-rbac.yaml
kubectl create -f traefik-service-acc.yaml
```

COMMAND - **$ bash kubeadmInit.sh**

Task 1.A.B) KUBEADM RESET

File name:- kubeadmReset.sh

```
#!/bin/sh

#Reset the kubeadm
sudo kubeadm reset --ignore-preflight-errors=IsPrivilegedUser,preflight -f

#flush out the iptables
sudo iptables -F

sudo rm /etc/cni/net.d/*

sudo ipvsadm --clear

rm /home/nirmalsingh/.kube/config
```

COMMAND - **$ bash kubeadmReset.sh**

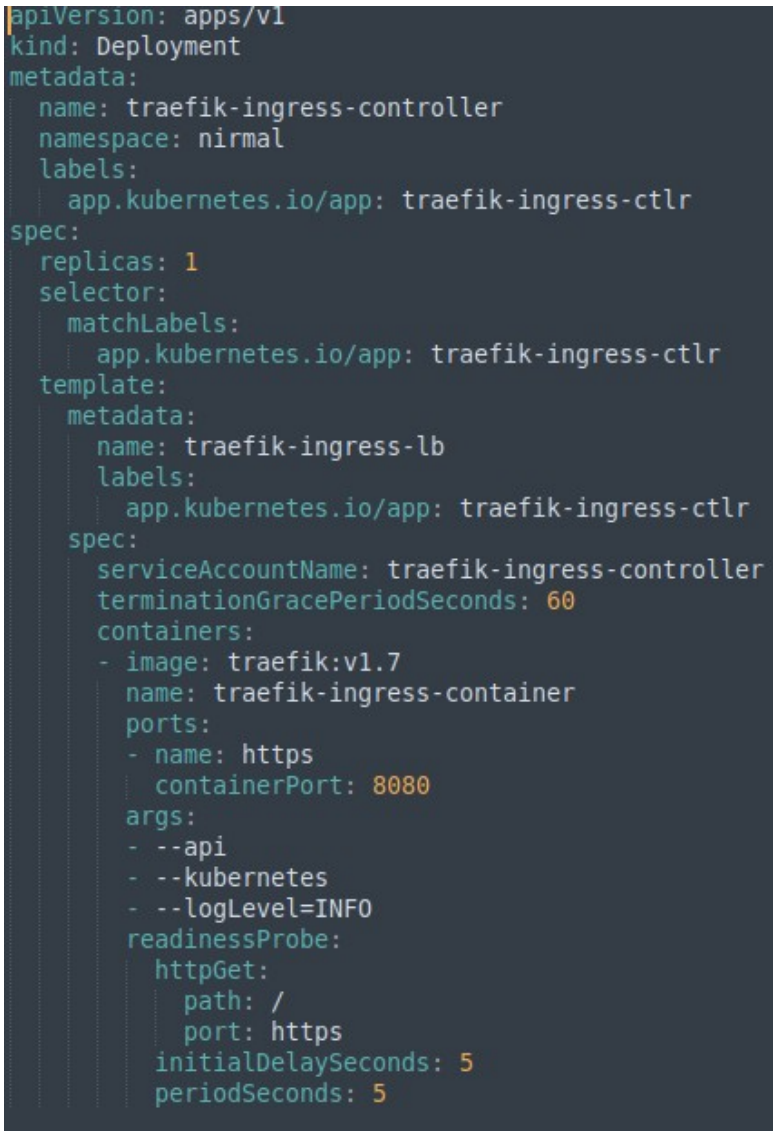
Task 1 . B) Deploy traefik ingress controller on your K8 cluster (you can use helm for this).

1.B.A) **CREATE SERVICE ACCOUNT**

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: traefik-ingress-controller
  namespace: nirmal
```

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: traefik-ingress
rules:
  - apiGroups:
      - ""
    resources:
      - services
      - endpoints
      - secrets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - extensions
    resources:
      - ingresses
    verbs:
      - get
      - list
      - watch
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: traefik-ingress
roleRef:
  kind: ClusterRole
  apiGroup: rbac.authorization.k8s.io
  name: traefik-ingress
subjects:
- kind: ServiceAccount
  name: traefik-ingress-controller
  namespace: nirmal
```

Deployment file for traefik ingress controller



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: traefik-ingress-controller
  namespace: nirmal
  labels:
    app.kubernetes.io/app: traefik-ingress-ctlr
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/app: traefik-ingress-ctlr
  template:
    metadata:
      name: traefik-ingress-lb
      labels:
        app.kubernetes.io/app: traefik-ingress-ctlr
    spec:
      serviceAccountName: traefik-ingress-controller
      terminationGracePeriodSeconds: 60
      containers:
      - image: traefik:v1.7
        name: traefik-ingress-container
        ports:
        - name: https
          containerPort: 8080
        args:
        - --api
        - --kubernetes
        - --logLevel=INFO
        readinessProbe:
          httpGet:
            path: /
            port: https
          initialDelaySeconds: 5
          periodSeconds: 5
```

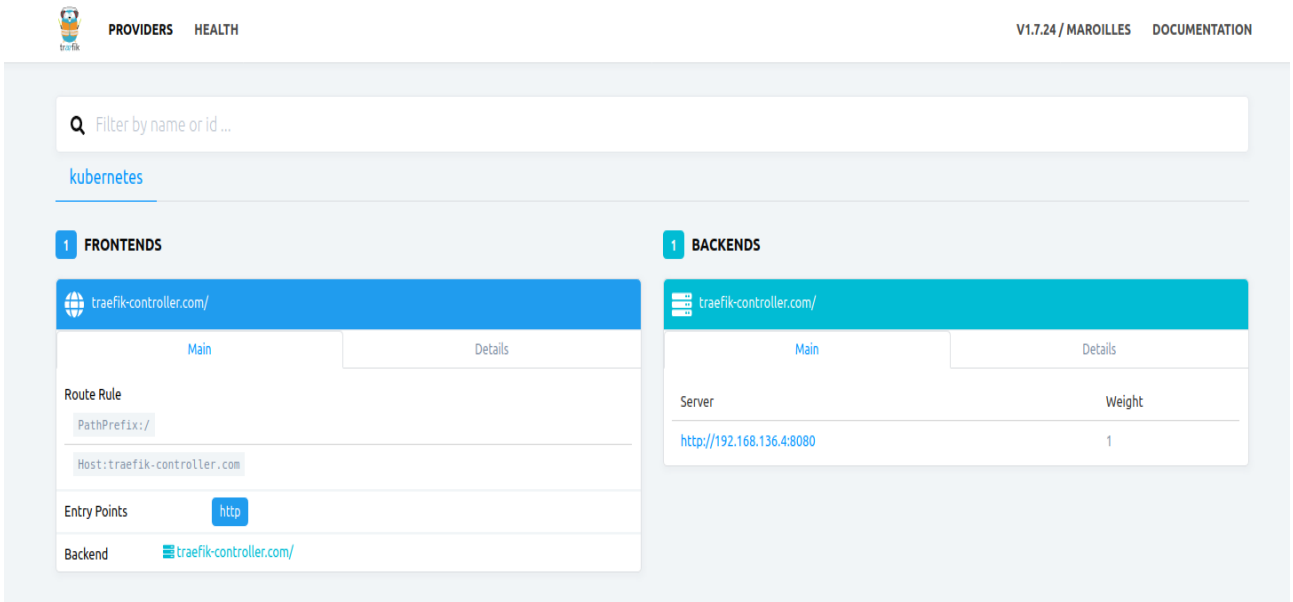
COMMAND - **$ kubectl create -f traefik-deployment.yaml**

Ingress Resource

```yaml
apiVersion: v1
kind: Service
metadata:
  name: traefik
  namespace: nirmal
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/path: "/metrics"
    prometheus.io/port: "80"
spec:
  selector:
   app.kubernetes.io/app: traefik-ingress-ctlr
  ports:
  - name: https
    port: 80
    targetPort: https
    protocol: TCP
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: ingress-resource
  namespace: nirmal
spec:
  rules:
  - host: xenon.ctl
    http:
      paths:
      - path: /
        backend:
          serviceName: traefik
          servicePort: https
```

COMMAND - **$ kubectl create -f traefik-ingress.yaml**

Verify the cluster/ingress controller is operational or not, once things seems good follow below guidelines: 192.168.1.103:8080



Task 2:
Dockerize the App mentioned by the URL https://github.com/M1TKO/my-note-webapp and deploy it on Kubernetes using following guidelines

A. Database should be external (deploy external DB on Kubernetes)
B. app should use persistent volumes (hostpath would work here for us)
C. ingress name to access via web should be notes.xenon.team
D. app should always scheduled by tolerating the taint
E. Demonstrate usage of Readiness and Liveness probe via your application

Task 2) Create docker file : Dockerfile

```
FROM php:7.3.3-apache

WORKDIR /var/www/html

COPY . .

RUN apt-get update && apt-get upgrade -y

RUN docker-php-ext-install mysqli pdo_mysql
```

2.1 Now build and push image to the docker hub

COMMAND - $ docker build -t image-name:version .
$ docker tag <imageID> docker-repo/image-name:version
$ docker push docker-repo/php-mysql-image:v 3

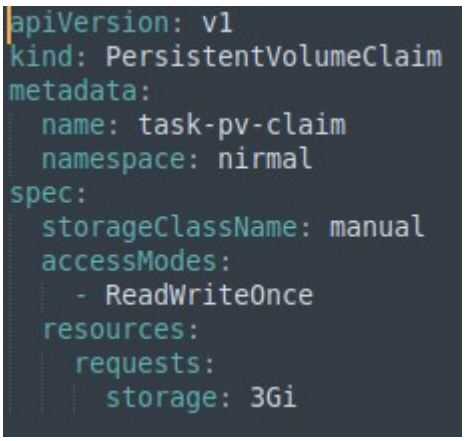
2.2 Create the persistent volume for application

1. persistent volume:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  namespace: nirmal
  labels:
    app.kubernetes.io/type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

2. persistent Volume Claim

a



COMMAND - $ kubectl create -f pv-volume.yaml
$ kubectl create -f pv-claim.yaml

2.3 Create Deployment for php app, mysql and phpmyadmin in single file :

webserver.yaml

php-deploy

```yaml
# php application  deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-deploy
  namespace: nirmal
  labels:
    app.kubernetes.io/app: php-app
spec:
  selector:
    matchLabels:
      app.kubernetes.io/app: php-app
  template:
    metadata:
      labels:
        app.kubernetes.io/app: php-app
    spec:
      containers:
        - name: php-app-container
          image: nirmalcontainer/php-mysql-image:v3
          # imagePullPolicy: Never
          ports:
            - containerPort: 80
          args:
            - --kubernetes
            - --logLevel=DEBUG
          # readinessProbe:
          #   httpGet:
          #     path: /
          #     port: 80
          #   initialDelaySeconds: 2
          #   periodSeconds: 2
      tolerations:
      - key: "key"
        value: "mosquito"
        effect: "NoSchedule"
```

mysql-deploy

```yaml
#mysql deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deploy
  namespace: nirmal
  labels:
    app.kubernetes.io/app: mysql-app
spec:
  selector:
    matchLabels:
      app.kubernetes.io/app: mysql-app
  template:
    metadata:
      labels:
        app.kubernetes.io/app: mysql-app
    spec:
      volumes:
        - name: task-pv-storage
          persistentVolumeClaim:
            claimName: task-pv-claim
      containers:
        - image: mysql:8.0
          name: mysql-app-container
          imagePullPolicy: Always
          env:
            - name: MYSQL_ROOT_PASSWORD
              value: rootpass
            - name: MYSQL_DATABASE
              value: my_note
            - name: MYSQL_USER
              value: root
            - name: MYSQL_PASSWORD
              value:
          args: ["--default-authentication-plugin=mysql_native_password"]
          ports:
            - containerPort: 3306
              name: http
          volumeMounts:
            - mountPath: "/var/lib/mysql"
              name: task-pv-storage
      tolerations:
      - key: "key"
        value: "mosquito"
        effect: "NoSchedule"
```

phpmyadmin deploy

```yaml
# phpmyadmin deploy
apiVersion: apps/v1
kind: Deployment
metadata:
  name: phpmyadmin-deploy
  namespace: nirmal
  labels:
    app.kubernetes.io/app: phpmyadmin-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app.kubernetes.io/app: phpmyadmin-app
  template:
    metadata:
      labels:
        app.kubernetes.io/app: phpmyadmin-app
    spec:
      containers:
        - name: phpmyadmin-container
          image: phpmyadmin/phpmyadmin
          ports:
            - containerPort: 80
          env:
            - name: PMA_HOST
              value: mysql-svc
            - name: PMA_PORT
              value: "3306"
            - name: MYSQL_ROOT_PASSWORD
              value: rootpass
      tolerations:
      - key: "key"
        value: "mosquito"
        effect: "NoSchedule"
```

COMMAND **- $ kubectl create -f webserver.yaml**

2.4 Create Service for each deployment :

webserver-svc.yaml

COMMAND - **$ kubectl create -f webserver-svc.yaml**

php-svc

```yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/path: "/metrics"
    prometheus.io/port: "80"
  name: php-svc
  namespace: nirmal
  labels:
    app.kubernetes.io/app: php-app
spec:
  selector:
    app.kubernetes.io/app: php-app
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      name: http
```

mysql-svc

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql-svc
  namespace: nirmal
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/path: "/metrics"
    prometheus.io/port: "3306"
spec:
  selector:
    app.kubernetes.io/app: mysql-app
  ports:
  - port: 3306
    protocol: TCP
    targetPort: 3306
    name: mysqlhttp
```

phpmyadmin-svc

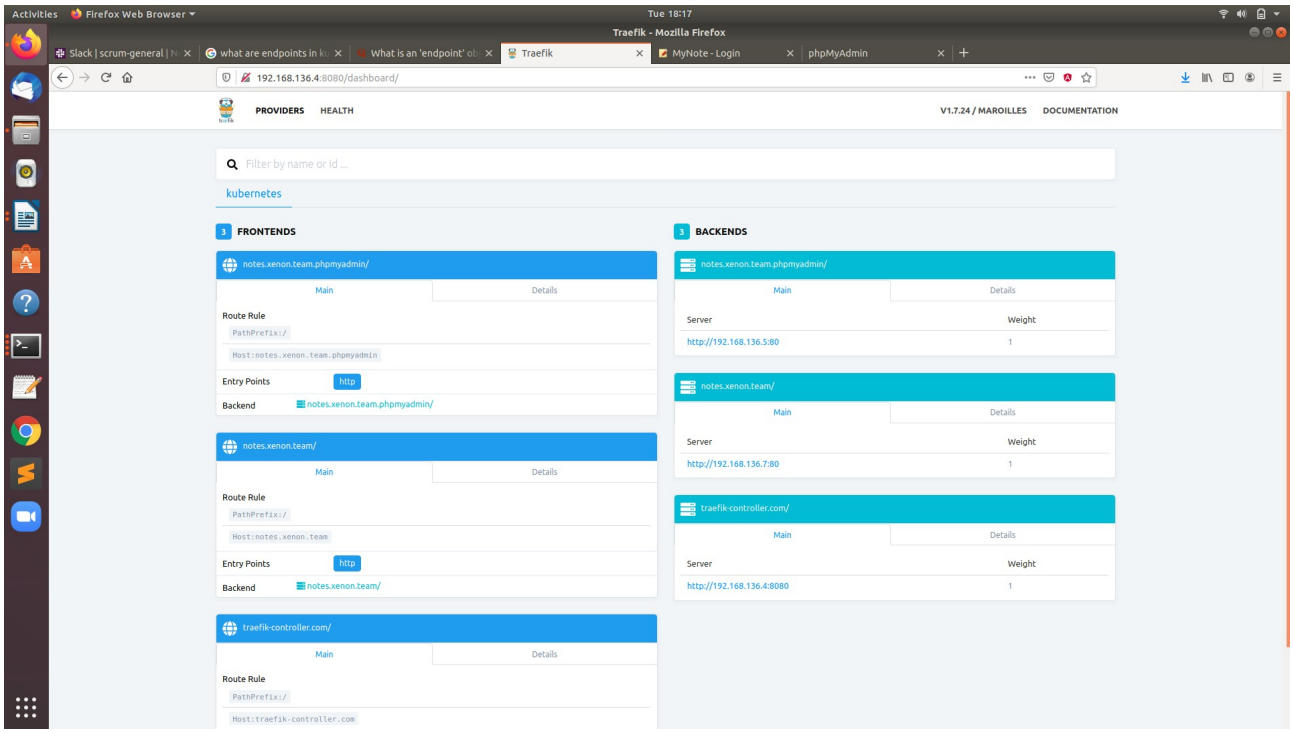
```yaml
# phpmyadmin service
apiVersion: v1
kind: Service
metadata:
  name: phpmyadmin-svc
  namespace: nirmal
  annotations:
    prometheus.io/scrape: "true"
    prometheus.io/path: "/metrics"
    prometheus.io/port: "80"
spec:
  selector:
    app.kubernetes.io/app: phpmyadmin-app
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    name: phpmyadminhttp
```

2.5 Create ingress-Resource file

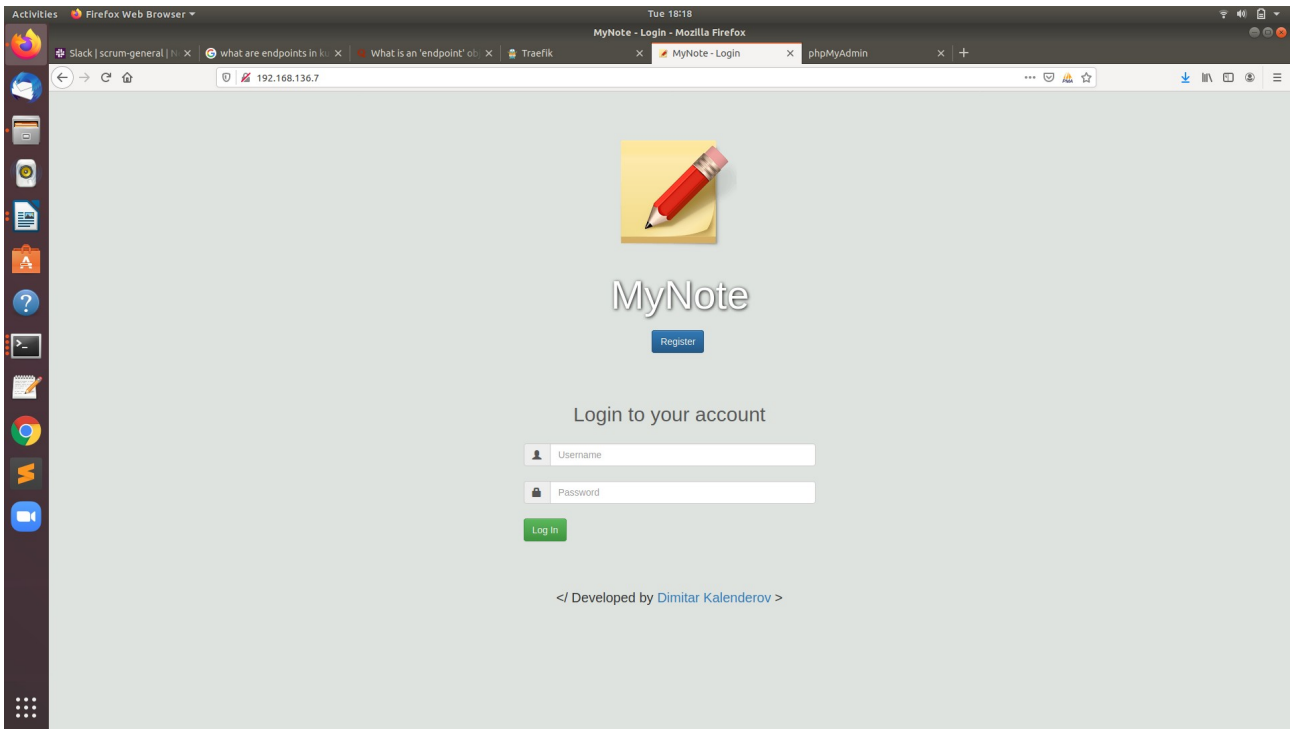
```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: php-ingress
  namespace: nirmal
spec:
  rules:
  - host: notes.xenon.team
    http:
      paths:
      - path: /
        backend:
          serviceName: php-svc
          servicePort: 80
  - host: notes.xenon.team.phpmyadmin
    http:
      paths:
      - path: /
        backend:
          serviceName: phpmyadmin-svc
          servicePort: 80
```

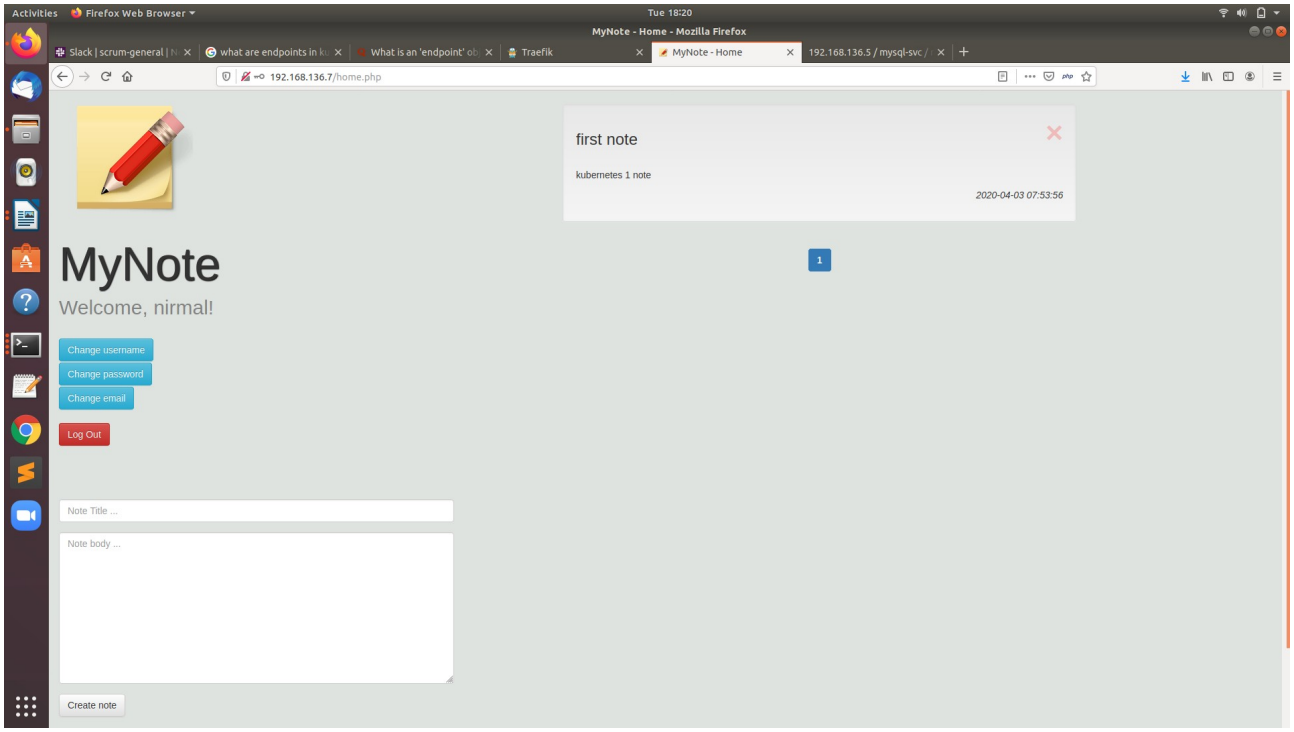
COMMAND - **$ kubectl create -f php-mysql-ingress.yaml**

SCREENSHOTS – traefik-ingress-controller

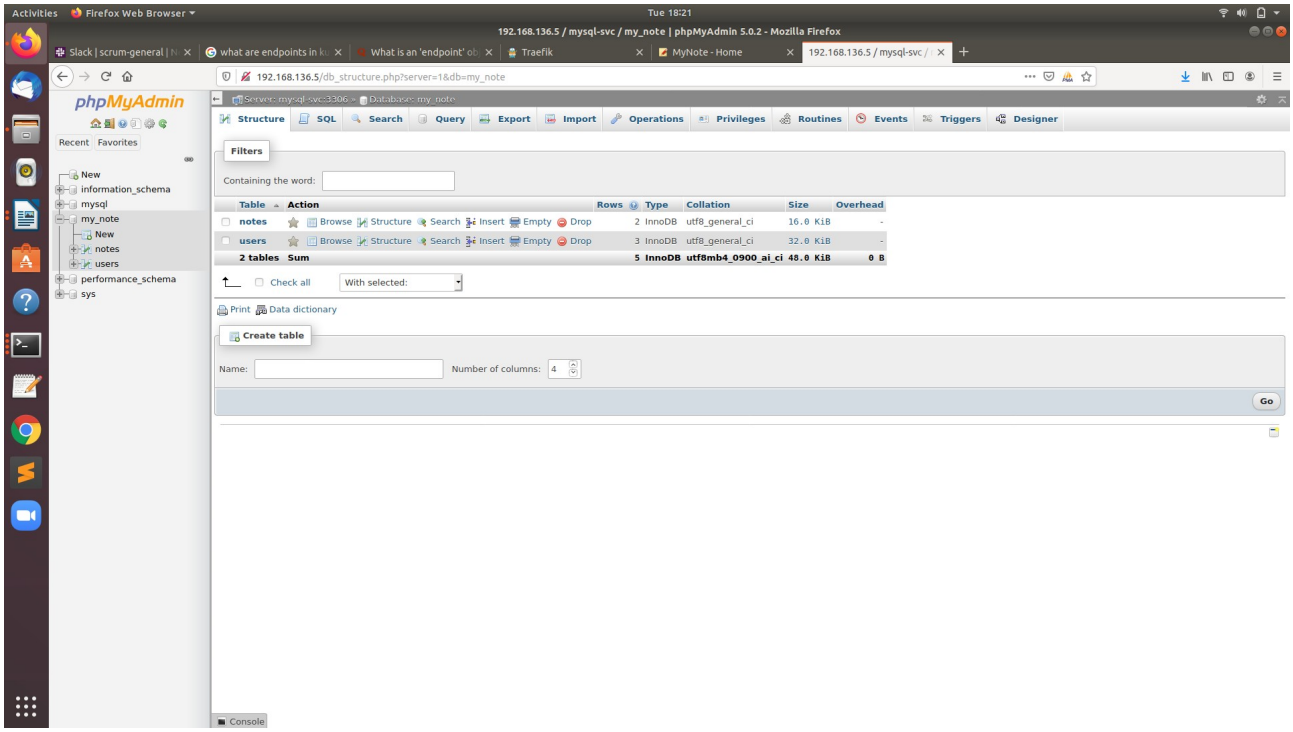


php-app UI screen shots





Mysql Database

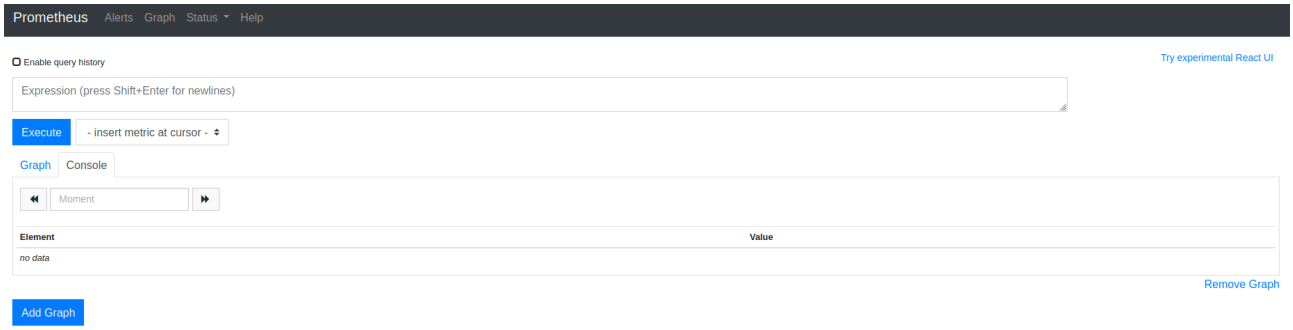


Task to be done via Helm3

Step 1 – Create Persistent volume first.
Step 2 – create helm chart : **$ helm install promgraf promgraf/ -n nirmal**
Step 3 - Set prometheus at nodeport 32322 and grafana at 32323

Next page consist screenshots of this

Prometheus



grafana

username – admin
password - nirmal