

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 necessary commands in that file for creating.

1.A.A) KUBEADM INIT

File name:- kubeadmInit.sh

```
#!/bin/sh
```

#kubeadm init

```
sudo kubeadm init --kubernetes-version=v1.18.0 --pod-network-cidr=10.244.0.0/16 --  
control-plane-endpoint=192.168.1.103 --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-
```

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
  namespace: kube-system
```

```
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:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: traefik-ingress
subjects:
- kind: ServiceAccount
  name: traefik-ingress
  namespace: kube-system

```

COMMAND - \$ kubectl create -f traefik-service-acc.yaml
 \$ kubectl create -f traefik-cr.yaml
 \$ kubectl create -f traefik-crb.yaml

Deployment file for traefik ingress controller

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
  labels:
    k8s-app: traefik-ingress-lb # selector v
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: traefik-ingress-lb
  template:
    metadata:
      labels:
        k8s-app: traefik-ingress-lb
        name: traefik-ingress-lb
    spec:
      serviceAccountName: traefik-ingress
      terminationGracePeriodSeconds: 60
      containers:
        - image: traefik:v1.7
          name: traefik-ingress-lb
          ports:
            - name: http
              containerPort: 80
            - name: admin
              containerPort: 8080
          args:
            - --api
            - --kubernetes
            - --logLevel=INFO
---
apiVersion: v1
kind: Service
metadata:
  name: traefik-ingress-service
  namespace: kube-system
spec:
  selector:
    k8s-app: traefik-ingress-lb
  type: NodePort
  ports:
    - protocol: TCP
      name: web
      port: 80
    - protocol: TCP
      port: 8080
      name: admin

```

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

Ingress Resource:-

```
apiVersion: v1
kind: Service
metadata:
  name: traefik-web-ui
  namespace: kube-system
spec:
  selector:
    k8s-app: traefik-ingress-lb
  ports:
    - name: web
      port: 8080
      targetPort: 8080
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: traefik-ingress-resource
  namespace: kube-system
spec:
  rules:
    - host: traefik-controller.com
      http:
        paths:
          - path: /
            backend:
              serviceName: traefik-web-ui
              servicePort: web
```

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



PROVIDERS HEALTH

V1.7.24 / MAROILLES DOCUMENTATION

Q Filter by name or id ...

kubernetes

1 FRONTENDS

traefik-controller.com/	
Main	Details
Route Rule	
PathPrefix:/	
Host:traefik-controller.com	
Entry Points	
http	
Backend	
traefik-controller.com/	

1 BACKENDS

traefik-controller.com/	
Main	Details
Server	
Weight	
http://192.168.136.4:8080	1

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 docker-repo/php-mysql-image:v3 .
          $ docker push docker-repo/php-mysql-image:v3
```

2.2 Create the persistent volume for application

1. persistent volume:

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

2. persistent Volume Claim

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: task-pv-claim
  namespace: kube-system
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

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

```
# php application deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-deploy
  namespace: kube-system
  labels:
    app: php-app
spec:
  selector:
    matchLabels:
      app: php-app
  template:
    metadata:
      labels:
        app: php-app
    spec:
      containers:
        - name: php-app-container
          image: nirmalcontainer/php-mysql-image:v3
          # imagePullPolicy: Never
          ports:
            - containerPort: 80
```

mysql-deploy

```
---
#mysql deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql-deploy
  namespace: kube-system
  labels:
    app: mysql-app
spec:
  selector:
    matchLabels:
      app: mysql-app
  template:
    metadata:
      labels:
        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
```

phpmyadmin-deploy

```
---
# phpmyadmin deploy
apiVersion: apps/v1
kind: Deployment
metadata:
  name: phpmyadmin-deploy
  namespace: kube-system
  labels:
    app: phpmyadmin-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: phpmyadmin-app
  template:
    metadata:
      labels:
        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
```

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

```
# php application
apiVersion: v1
kind: Service
metadata:
  name: php-svc
  namespace: kube-system
spec:
  type: NodePort
  selector:
    app: php-app
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
      name: http
```

mysql-svc

```
---
apiVersion: v1
kind: Service
metadata:
  name: mysql-svc
  namespace: kube-system
  labels:
    app: mysql-app
spec:
  selector:
    app: mysql-app
  type: NodePort
  ports:
    - port: 3306
      protocol: TCP
      targetPort: 3306
      name: mysqlhttp
```

phpmyadmin-svc

```
---
# phpmyadmin service
apiVersion: v1
kind: Service
metadata:
  name: phpmyadmin-svc
  namespace: kube-system
  labels:
spec:
  type: NodePort
  selector:
    app: phpmyadmin-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      name: phpmyadminhttp
```

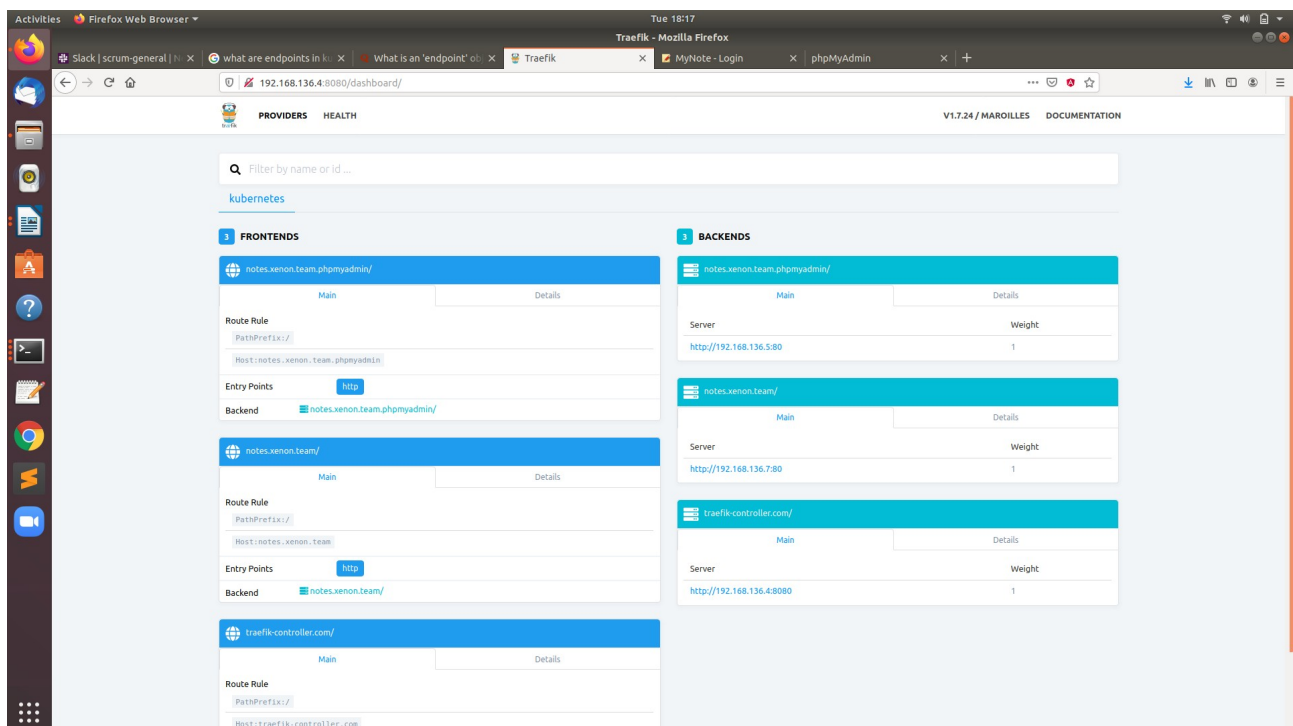
2.5 Create ingress-Resource file

```
# this is ingress resource file used to hit traefik
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: php-ingress
  namespace: kube-system
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: notes.xenon.team
    http:
      paths:
      - path: /
        backend:
          serviceName: php-svc
          servicePort: http

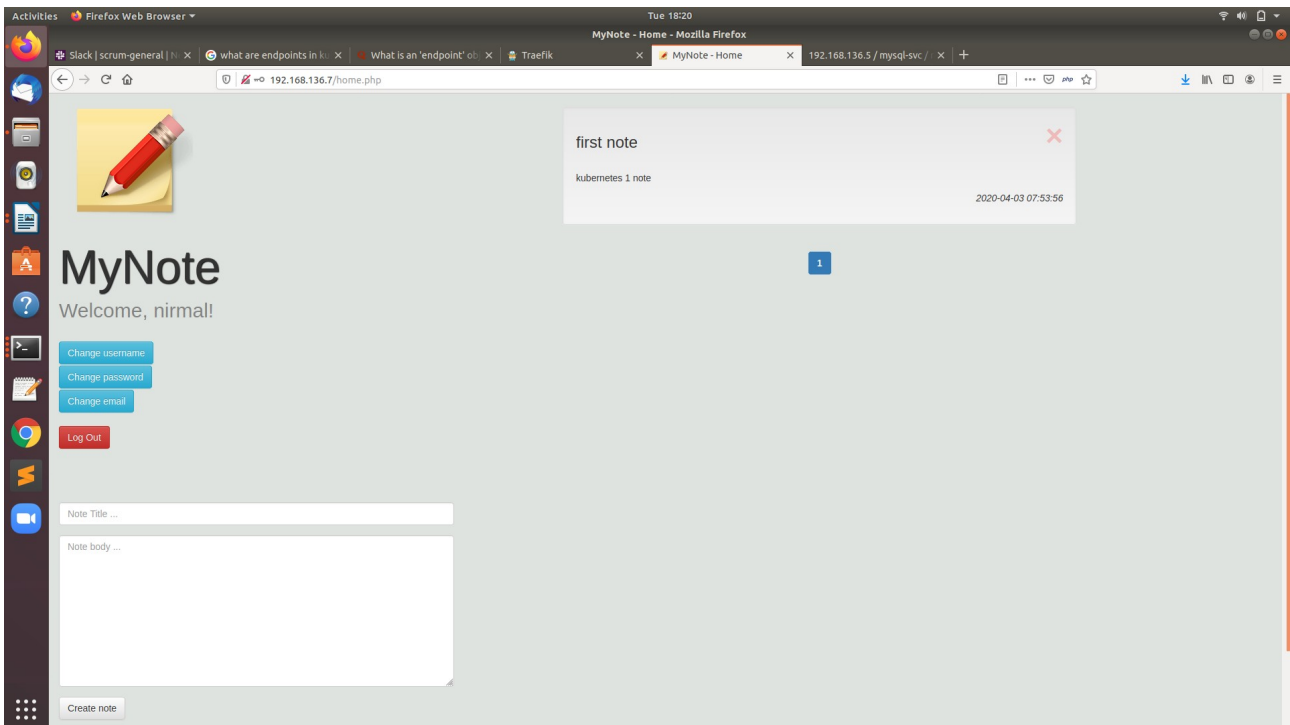
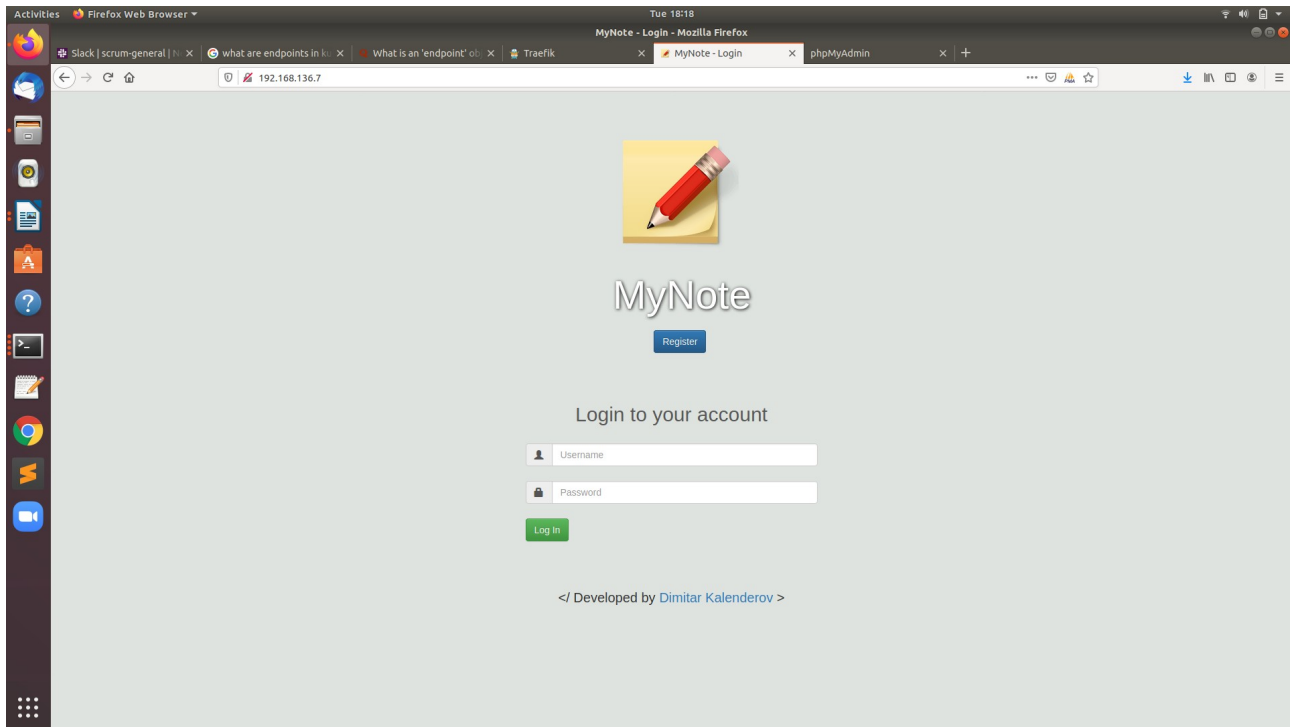
# - host: notes.xenon.team.mysql
#   http:
#     paths:
#     - path: /
#       backend:
#         serviceName: mysql-svc
#         servicePort: mysqlhttp
  - host: notes.xenon.team.phpmyadmin
    http:
      paths:
      - path: /
        backend:
          serviceName: phpmyadmin-svc
          servicePort: phpmyadminhttp
```

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

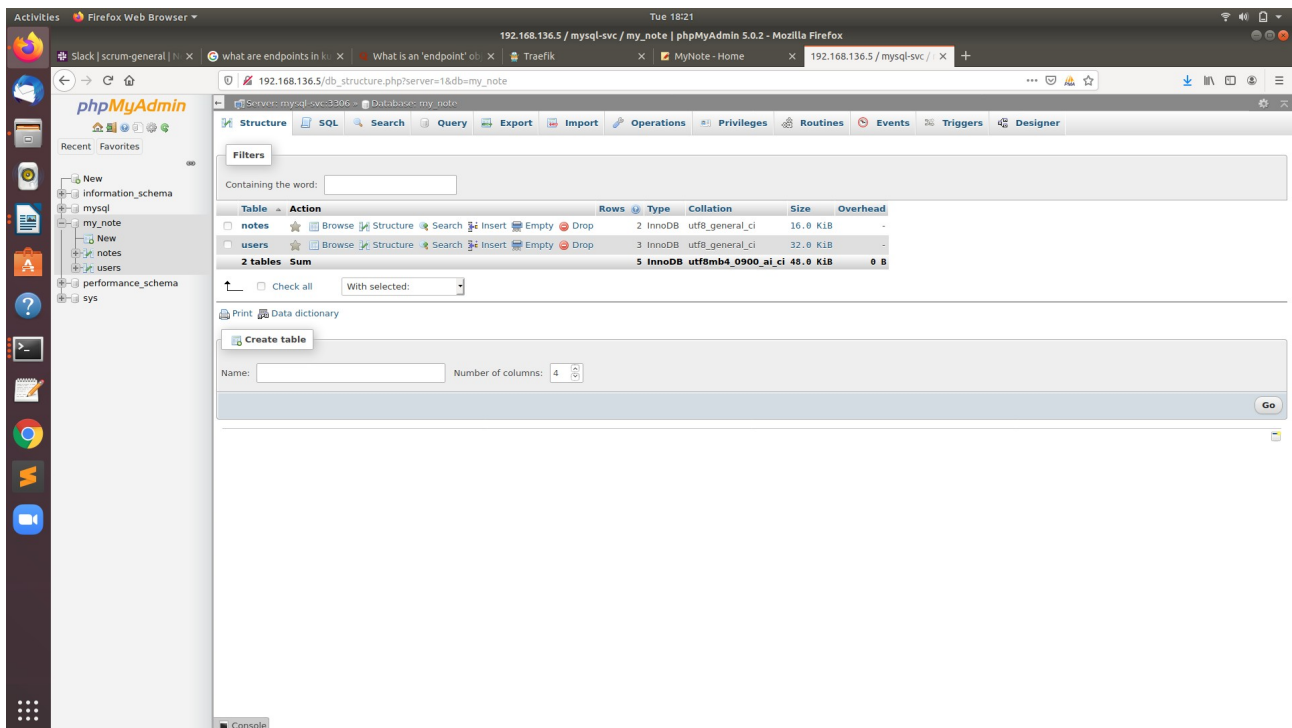
SCREENSHOTS – traefik-ingress-controller



php-app UI screen shots



Mysql Database



Task to be done via Helm3

Step 1 – create namespace monitoring - **\$ kubectl create namespace monitoring.**

Step 2 – create helm chart : **\$ helm install metrics stable/prometheus-operator --namespace monitoring.**

Step 3 - get all objects under monitoring namespace - **\$ kubectl get all -n monitoring**

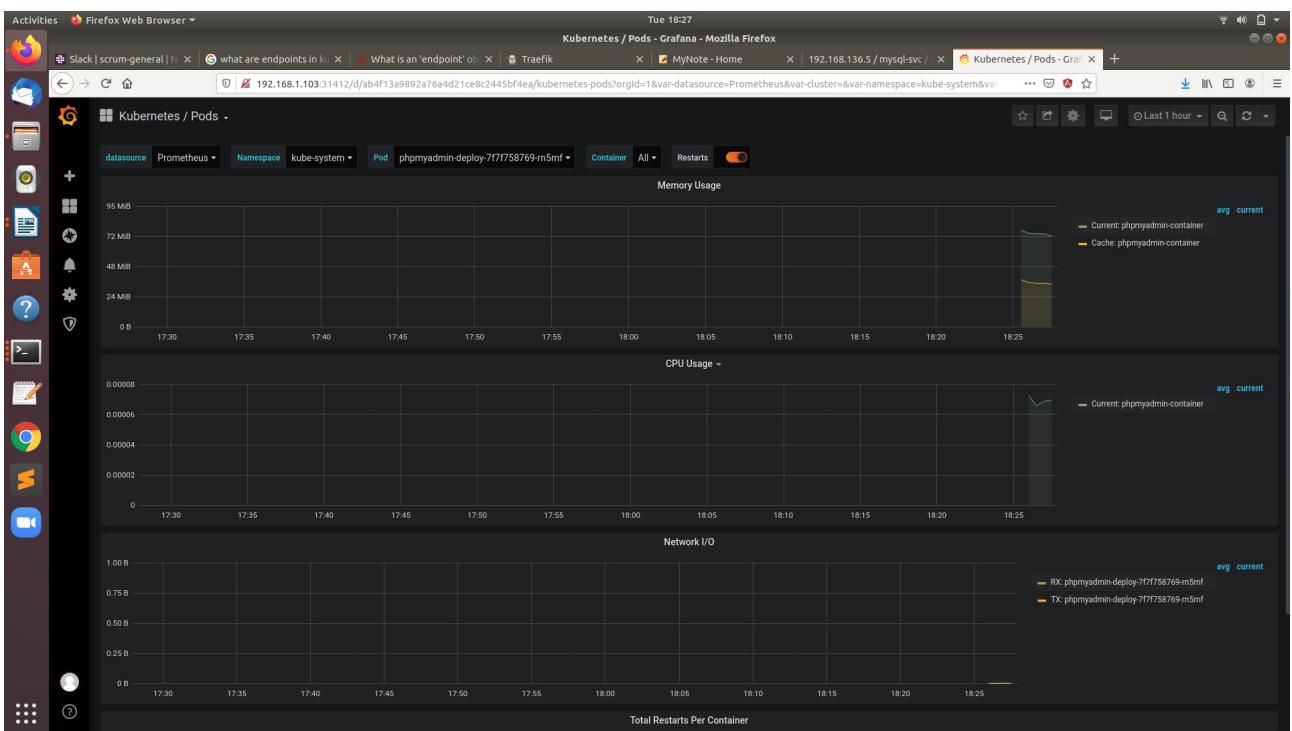
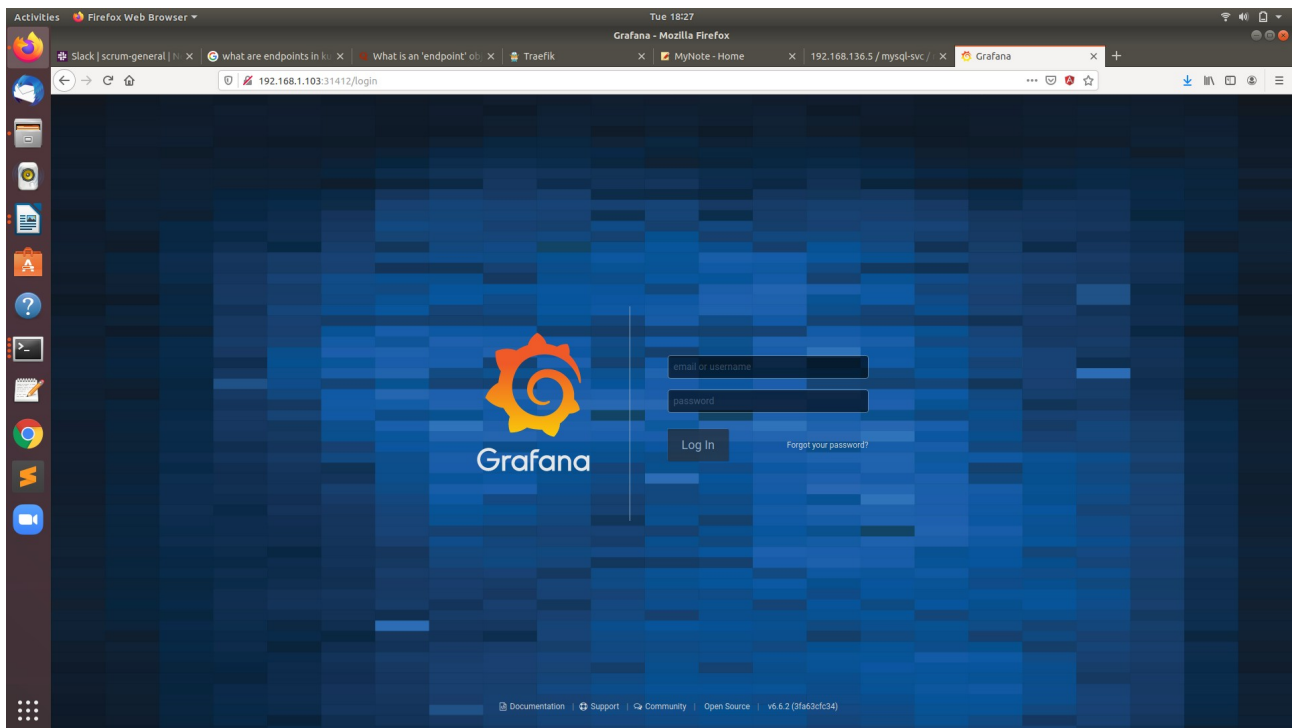
Step 4 - Now edit the grafana service. Change its type from **ClusterIP** to **NodePort** - **\$ kubectl edit service/metrics-grafana -n monitoring**

Step 5 – Goto browser and write URL = IP address of worker machine on which chart installed with service nodeport like : - **192.168.1.1:32412**

Next page consist screenshots of this

username – admin

password - prom-operator



GITHUB REPO link : - <https://github.com/nirmalsingh7950/finaltaskkonkubernetes>