Kubernets

Tuesday, June 14, 2022 2:25 PM

Start the minikube locally on windows machine
INSTALL minikube application
INSTALL kubectl application
Virtual box required

Go to the minikube location open cmd
Run:- minikube start --driver=virtualbox --no-vtx-check

NOW YOU CAN USE IT FOR Kubernetess

1. Create a web service using any publicly available image (e.g. nginx) and deploy it on Kubernetes with 2 replicas

Create a deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

kubectl apply -f deployment.yaml

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (main)
$ ./kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   2/2     2            2           44s

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (main)
$ ./kubectl get pods
NAME                               READY   STATUS    RESTARTS   AGE
nginx-deployment-9456bbbf9-gb8f8   1/1     Running   0          49s
nginx-deployment-9456bbbf9-xlgn2   1/1     Running   0          49s
```

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (main)
$ ./kubectl get replicaset
NAME                         DESIRED   CURRENT   READY   AGE
nginx-deployment-9456bbbf9   2         2         2       4m30s
```

Clean up

kubectl delete deployment nginx-deployment

2. Update the web service to store the logs on the Host using persistent volume and volume claim

- You, as cluster administrator, create a PersistentVolume backed by physical storage. You do not associate the volume with any Pod.
- You, now taking the role of a developer / cluster user, create a PersistentVolumeClaim that is automatically bound to a suita ble PersistentVolume.
- You create a Pod that uses the above PersistentVolumeClaim for storage.

Create one index.html file for testing
Open shell minikube ssh
/mnt/data here create one index.html file created
sudo mkdir /mnt/data
sudo sh -c "echo 'Hello from Kubernetes storage' > /mnt/data/index.html"
cat /mnt/data/index.html

Create a PersistentVolume

Create pods-storage-pv-volume.yaml

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

kubectl apply -f pods-storage-pv-volume.yaml
kubectl get pv task-pv-volume

Create a PersistentVolumeClaim

Create pods-storage-pv-claim.yaml

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

kubectl apply -f pods-storage-pv-claim.yaml
kubectl get pvc task-pv-claim

create a Pod that uses your PersistentVolumeClaim as a volume.

Create with-pv-deployment.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: task-pv-claim
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
```

kubectl apply -f with-pv-deployment.yaml
kubectl get pod task-pv-pod
kubectl exec -it task-pv-pod -- /bin/bash

```
apt update
apt install curl
curl http://localhost/
```

```
C:\Users\AVVSFS744\Downloads\New folder>kubectl exec -it task-pv-pod -- /bin/bash
root@task-pv-pod:/# apt update
Get:1 http://security.debian.org/debian-security bullseye-security InRelease [44.1 k
B]
Get:2 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:3 http://security.debian.org/debian-security bullseye-security/main amd64 Packag
```

```
C:\Users\AVVSFS744\Downloads\New folder>kubectl exec -it task-pv-pod -- /bin/bash
root@task-pv-pod:/# apt update
Get:1 http://security.debian.org/debian-security bullseye-security InRelease [44.1 k
B]
Get:2 http://deb.debian.org/debian bullseye InRelease [116 kB]
Get:3 http://security.debian.org/debian-security bullseye-security/main amd64 Packag
es [154 kB]
Get:4 http://deb.debian.org/debian bullseye-updates InRelease [39.4 kB]
Get:5 http://deb.debian.org/debian bullseye/main amd64 Packages [8182 kB]
Get:6 http://deb.debian.org/debian bullseye-updates/main amd64 Packages [2592 B]
Fetched 8539 kB in 9s (973 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
root@task-pv-pod:/# apt install curl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.74.0-1.3+deb11u1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
root@task-pv-pod:/# curl http://localhost/
Hello from Kubernetes storage
root@task-pv-pod:/#
```

If you see that message, you have successfully configured a Pod to use storage from a PersistentVolumeClaim.

Clean up

Delete the Pod, the PersistentVolumeClaim and the PersistentVolume:

kubectl delete pod task-pv-pod
kubectl delete pvc task-pv-claim
kubectl delete pv task-pv-volume

3. Expose the web service so it can be accessed outside the cluster

We need to create one service here for expose the application

- **NodePort** – This is the most basic option of exposing your service to be accessible outside of your cluster, on a specific port (called the **NodePort**) on every node in the cluster. We will illustrate this option shortly.

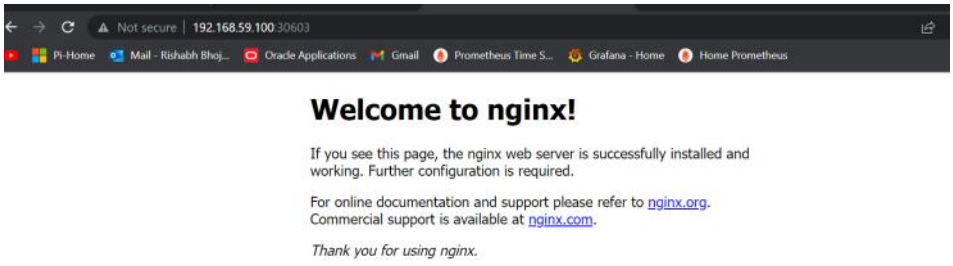
kubectl create service nodeport nginx --tcp=80:80
kubectl get svc

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (main)
$ ./kubectl get svc
NAME         TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
kubernetes   ClusterIP   10.96.0.1      <none>        443/TCP        5h42m
nginx        NodePort    10.110.87.3    <none>        80:30603/TCP   8m56s
```

Decribe...

```
$ ./kubectl describe svc nginx
Name:                     nginx
Namespace:                default
Labels:                   app=nginx
Annotations:              <none>
Selector:                 app=nginx
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.110.87.3
IPs:                      10.110.87.3
Port:                     80-80  80/TCP
TargetPort:               80/TCP
NodePort:                 80-80  30603/TCP
Endpoints:                172.17.0.3:80,172.17.0.4:80
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

curl http://<external-ip>:<port>



4. Create another service for the database using any publicly available database images (e.g. MySQL) and deploy it on Kubernetes Cluster

- Create a PersistentVolume referencing a disk in your environment.
- Create a MySQL Deployment.
- Expose MySQL to other pods in the cluster at a known DNS name.

► Create the application-mysql-mysql-deployment.yaml
Include the service/deployement

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
  - port: 3306
  selector:
    app: mysql
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
          # Use secret in real usage
        - name: MYSQL_ROOT_PASSWORD
          value: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```

► Create application-mysql-mysql-pv.yaml
Include pv/pvc

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mysql-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

Then,
Apply both :-
kubectl apply -f application-mysql-mysql-pv.yaml
kubectl apply -f application-mysql-mysql-deployment.yaml

kubectl get pods

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
mysql-6879db89b4-9v69s   1/1     Running   0          12s

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$
```

kubectl describe pvc mysql-pv-claim

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl describe pvc mysql-pv-claim
Name:          mysql-pv-claim
Namespace:     default
StorageClass:  manual
Status:        Bound
Volume:        mysql-pv-volume
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      20Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       mysql-6879db89b4-9v69s
Events:
  Type     Reason              Age   From                         Message
  ----     ------              ----  ----                         -------
  Warning  ProvisioningFailed  2m5s  persistentvolume-controller  storageclass.storage.k8s.io "manual" n
```

Then

kubectl run -it --rm --image=mysql:5.6 --restart=Never mysql-client -- mysql -h mysql -ppassword>

This command creates a new Pod in the cluster running a MySQL client and connects it to the server through the Service. If it connects, you know your stateful MySQL database is up and running.

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl run -it --rm --image=mysql:5.6 --restart=Never mysql-client -- mysql -h mysql -ppassword
Unable to use a TTY - input is not a terminal or the right kind of file
If you don't see a command prompt, try pressing enter.
Error attaching, falling back to logs:
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2021, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
mysql-6879db89b4-9v69s   1/1     Running   0          8m8s
mysql-client             1/1     Running   0          10s
```

Clean up

kubectl delete deployment,svc mysql
kubectl delete pvc mysql-pv-claim
kubectl delete pv mysql-pv-volume
kubectl delete pod mysql-client

5. Expose the web service so it can be accessed only within the cluster

ClusterIP (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster.
This Service-type generally exposes the service on an internal IP, reachable only within the cluster, and possibly only within the cluster -node,
On privious application MySql we use ClusteIP = none on Service file.

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ cat  application-mysql-mysql-deployment.yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql
spec:
  ports:
  - port: 3306
  selector:
    app: mysql
  clusterIP: None
```

SO IT CAN ONLY BE ACCESSED WITHIN THE CLUSTER

6. Create a configMap to store

- location/URL of database service.
- Port number where the database service is available

Update the web service to include these configurations as environment variables
DB_HOST

DB_PORT

7. Create a secret to store the database user and password. Update the web service to mount the secret at /etc/secret

We create 2 deployment
MongoDB and MongoEXPRESS

First we create mongodb deployment.yaml

- With one internal service
- Create config map
- And secrete
- Use EVN enviromental variable to use secret and configmap

mongo.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo
        ports:
        - containerPort: 27017
```

then we create mongoEXPRESS deployment.yaml

- With external service so it can be accessible outside
- Use EVN enviromental variable to use secret and configmap

mongo-express.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo-express
  labels:
    app: mongo-express
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo-express
  template:
    metadata:
      labels:
        app: mongo-express
    spec:
      containers:
      - name: mongo-express
        image: mongo-express
        ports:
        - containerPort: 8081
        env:
        - name: ME_CONFIG_MONGODB_ADMINUSERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: mongo-root-username
        - name: ME_CONFIG_MONGODB_ADMINPASSWORD
          valueFrom:
            secretKeyRef:
```

Deploy Both,
You can check kubectl logs <mongodb pod>

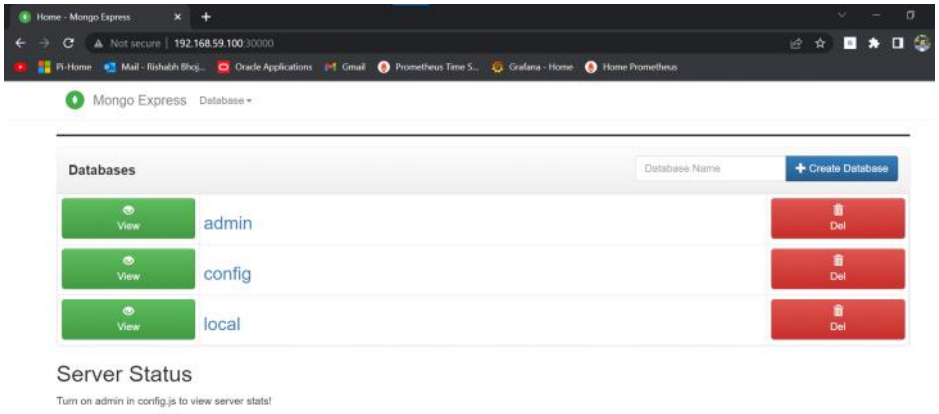
So it can show it's available to serve

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
mongo-express       1/1     1            1           63m
mongodb-deployment  1/1     1            1           155m

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get pods
NAME                                 READY   STATUS    RESTARTS        AGE
mongo-express-68c4748bd6-24r67       1/1     Running   2 (4m59s ago)   63m
mongodb-deployment-7bb6c6c4c7-xpnc2  1/1     Running   2 (5m55s ago)   155m

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get svc
NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
kubernetes            ClusterIP      10.96.0.1       <none>        443/TCP          5d7h
mongo-express-service LoadBalancer   10.100.33.247   <pending>     8081:30000/TCP   63m
mongodb-service       ClusterIP      10.111.243.38   <none>        27017/TCP        155m

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
```



8. Create another web service with 1 replica and expose it only within the cluster

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ cat deployment-nginx-1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

ClusterIP – This Service-type generally exposes the service on an internal IP, reachable only within the cluster, and possibly only within the cluster-nodes
As the **clusterIP** service is by default injected so the application can be exposed within the cluster only

kubectl get pods
kubectl exec -it task-pv-pod -- /bin/bash

```
apt update
apt install curl
curl http://localhost/
```

```
root@nginx-deployment-9456bbbf9-n2j4z:/# curl http://localhost/
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@nginx-deployment-9456bbbf9-n2j4z:/#
```

9. Deploy an nginx-ingress controller and create ingress rules

We use,
minikube addons enable ingress
commond to create ingress controller
Kubectl get pod -n kube-system

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ minikube addons enable ingress
  - Using image k8s.gcr.io/ingress-nginx/controller:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
  - Using image k8s.gcr.io/ingress-nginx/kube-webhook-certgen:v1.1.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

create ingress.yaml for mongo application

mongo-ingress.yaml

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: mongodb-ingress
spec:
  rules:
    - host: myapp.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: mongo-express-service
                port:
                  number: 8081
```

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get ingress
NAME              CLASS   HOSTS       ADDRESS          PORTS   AGE
mongodb-ingress   nginx   myapp.com   192.168.59.100   80      64s

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl describe ingress mongodb-ingress
Name:             mongodb-ingress
Labels:           <none>
Namespace:        default
Address:          192.168.59.100
Ingress Class:    nginx
Default backend:  <default>
Rules:
  Host        Path  Backends
  ----        ----  --------
  myapp.com
              /   mongo-express-service:8081 (172.17.0.5:8081)
Annotations:  <none>
Events:
  Type    Reason  Age                  From                      Message
  ----    ------  ----                 ----                      -------
  Normal  Sync    5m1s (x2 over 5m49s)  nginx-ingress-controller  Scheduled for sync
```

Imp:- Add the following line to the bottom of the /etc/hosts file on your computer (you will need administrator access):

```
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
#       192.168.59.100  myapp.com
~
~
~
~
~
```

11. A helm chart to deploy everything

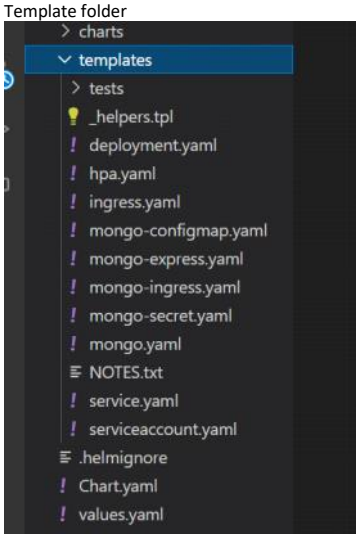
Install helm

Create a helm chart

`helm create mongoapp`

All deployment required file present in

Template folder



Apply the `helm install <name> <chartname>`

```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./helm install appmongo mongoapp
NAME: appmongo
LAST DEPLOYED: Tue Jun 21 10:20:59 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=mongoapp,app.kubernetes.io/instance=appmongo" -o jsonpath="{.items[0].metadata.name}")
  export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT
```

Check using,

`kubectl get all`

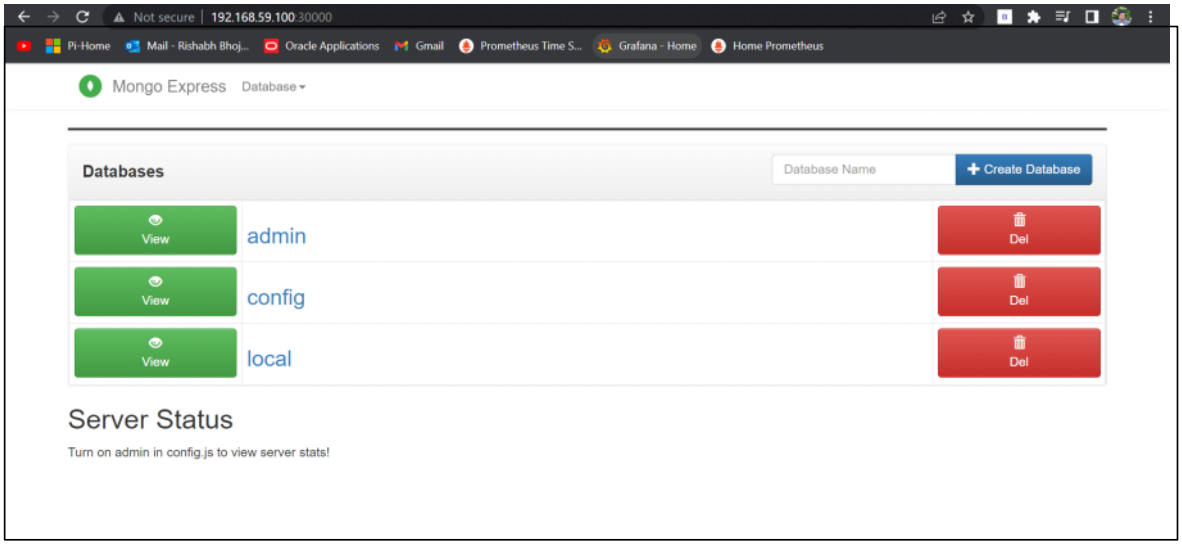
```
GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$ ./kubectl get all
NAME                                      READY   STATUS    RESTARTS   AGE
pod/appmongo-mongoapp-5fcb4f7699-7m9rx    1/1     Running   0          70s
pod/mongo-express-68c4748bd6-tkkhk        1/1     Running   0          70s
pod/mongodb-deployment-7bb6c6c4c7-z9d8k   1/1     Running   0          70s

NAME                            TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/appmongo-mongoapp       ClusterIP      10.96.4.55       <none>        80/TCP           70s
service/kubernetes              ClusterIP      10.96.0.1        <none>        443/TCP          6d21h
service/mongo-express-service   LoadBalancer   10.98.10.38      <pending>     8081:30000/TCP   70s
service/mongodb-service         ClusterIP      10.106.181.251   <none>        27017/TCP        70s

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/appmongo-mongoapp    1/1     1            1           70s
deployment.apps/mongo-express        1/1     1            1           70s
deployment.apps/mongodb-deployment   1/1     1            1           70s

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/appmongo-mongoapp-5fcb4f7699    1         1         1       70s
replicaset.apps/mongo-express-68c4748bd6        1         1         1       70s
replicaset.apps/mongodb-deployment-7bb6c6c4c7   1         1         1       70s

GMX+AVVSFS744@DESKTOP-IF5GKU3 MINGW64 ~/Downloads/New folder (master)
$
```



APPLICATION RUNNING BY INSTALLING ONE HELM CHART ONLY