Path-Based Routing in Kubernetes (K8s) Cluster

Step 1: Make sure your docker is running and start minikube cluster.
Step 2: Build the images of ms1, ms2, ms3
Step 3: Push these images on docker hub / aws ecr
Step 4: Write ms1-deployment.yml describing the deployment of ms1.
Step 5: Write ms1-service.yml describing the service of ms1.
Step 6: Replicate these for all the other ms.
Step 7: Write the ingress controller.

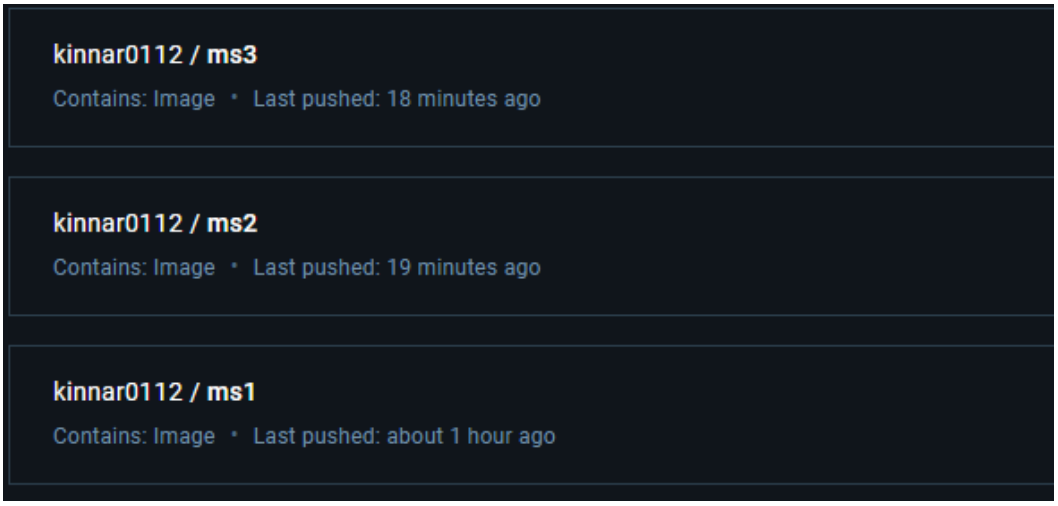
Solution

1. First We validate weather the docker service is running by using command **docker info**

```
Client:
 Version:    25.0.3
 Context:    default
 Debug Mode: false

Server:
 Containers: 29
  Running: 3
  Paused: 4
  Stopped: 22
 Images: 22
 Server Version: 25.0.3
```

2. Started minikube cluster using **minikube start**.
3. Created three separate images for three different microservices i.e. **kinnar0112/ms1:1.0**, **kinnar0112/ms2:1.0,** and **kinnar0112/ms3:1.0** and pushed it into the dokerhub repository.

```
cd microservice1
docker build -t kinnar0112/ms1:1.0 .
docker push kinnar0112/ms1:1.0

cd ../microservice2
docker build -t kinnar0112/ms2:1.0 .
docker push kinnar0112/ms2:1.0

cd ../microservice3
docker build -t kinnar0112/ms3:1.0 .
docker push kinnar0112/ms3:1.0
```



4. Created three different deployment.yml files for microservice1, microservice2 & microservice3 as ms1-deployment.yml, ms2-deployment.yml, and ms3-deployment.yml accordingly.
5. Created three different service.yml files for microservice1, microservice2 & microservice3 as ms1-service.yml, ms2-service.yml, and ms3-service.yml accordingly.
6. These three deployment & service files were applied to the K8s cluster.

```
kubectl apply -f ms1-deployment.yml
kubectl apply -f ms1-service.yml

kubectl apply -f ms2-deployment.yml
kubectl apply -f ms2-service.yml

kubectl apply -f ms3-deployment.yml
kubectl apply -f ms3-service.yml
```

7. Verify the deployment and services.

kubectl get deployments

```
PS D:\kinnarchowdhury\personal\hero_vired\projects\k8s-pathbased_routing> kubectl get deployments
NAME                    READY   UP-TO-DATE   AVAILABLE   AGE
kc-flaskapp-deployment  3/3     3            3           6d
ms1-deployment          3/3     3            3           51m
ms2-deployment          3/3     3            3           32m
ms3-deployment          3/3     3            3           32m
```

kubectl get services

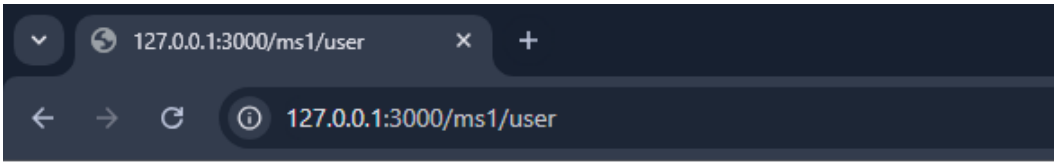
```
PS D:\kinnarchowdhury\personal\hero_vired\projects\k8s-pathbased_routing> kubectl get services
NAME                TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kc-flaskapp-service NodePort    10.106.33.174    <none>        5000:30005/TCP   6d
kubernetes          ClusterIP   10.96.0.1        <none>        443/TCP          6d1h
ms1-service         NodePort    10.104.174.182   <none>        3000:30001/TCP   50m
ms2-service         NodePort    10.100.0.69      <none>        3001:30002/TCP   30m
ms3-service         NodePort    10.97.158.234    <none>        3002:30003/TCP   30m
```

kubectl get pods

```
PS D:\kinnarchowdhury\personal\hero_vired\projects\k8s-pathbased_routing> kubectl get pods
NAME                                    READY   STATUS    RESTARTS        AGE
kc-flaskapp-deployment-6d447d6cb7-dt4lm 1/1     Running   1 (5d22h ago)   6d
kc-flaskapp-deployment-6d447d6cb7-nft4n 1/1     Running   1 (5d22h ago)   6d
kc-flaskapp-deployment-6d447d6cb7-rv26s 1/1     Running   1 (5d22h ago)   6d
ms1-deployment-7458b7b6f6-qdw8q         1/1     Running   0               48m
ms1-deployment-7458b7b6f6-th9ph         1/1     Running   0               48m
ms1-deployment-7458b7b6f6-thppt         1/1     Running   0               48m
ms2-deployment-64c889565b-6npfg         1/1     Running   0               29m
ms2-deployment-64c889565b-wzd75         1/1     Running   0               29m
ms2-deployment-64c889565b-zp9pr         1/1     Running   0               29m
ms3-deployment-5946cd7688-4gfwt         1/1     Running   0               29m
ms3-deployment-5946cd7688-mztf2         1/1     Running   0               29m
ms3-deployment-5946cd7688-q4gqm         1/1     Running   0               29m
```

8. Accessing ms1 service by tunnelling 3000 port.

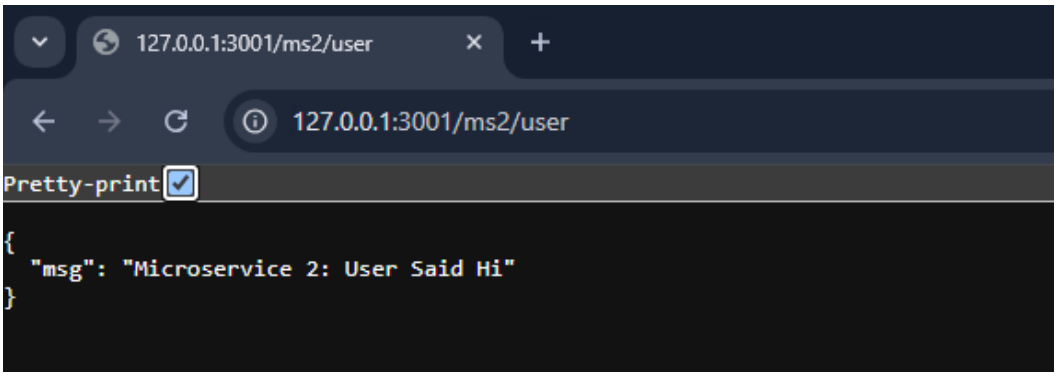
kubectl port-forward service/ms1-service 3000:3000



Hello World from MS1 User!

9. Accessing ms2 service by tunnelling 3001 port.

kubectl port-forward service/ms2-service 3001:3001



10. Accessing ms3 service by tunnelling 3002 port.

kubectl port-forward service/ms3-service 3002:3002