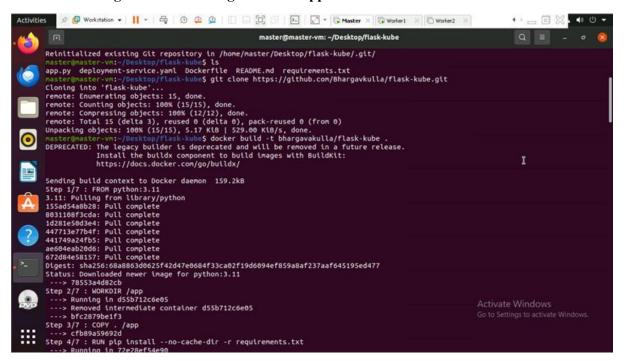# Kubernetes Project-1 Deploying a Flask Application on Kubernetes with Auto-Scaling & Load Testing

STEP-1. Building & Containerizing the Flask Application

STEP-2. Deploying Flask App on Kubernetes

STEP-3. Fixing Docker Hub Rate Limits (Authentication Issue)

STEP-4. Installing & Troubleshooting Metrics Server

```
master@master-vm:~/Desktop/flask-kube$ kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "docker-secret"}]}'
serviceaccount/default patched
master@master-vm:~/Desktop/flask-kube$ kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/c
omponents.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
master@master-vm:~/Desktop/flask-kube$
```
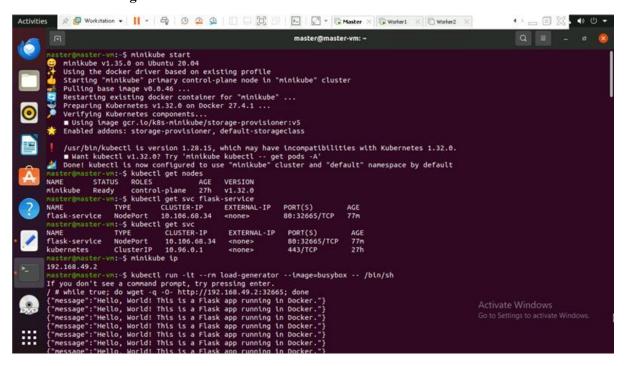
STEP-5. Enabling HPA (Horizontal Pod Autoscaler)

```
master@master-vm:~/Desktop/flask-kube$ kubectl autoscale deployment flask-app --cpu-percent=50 --min=3 --max=10
horizontalpodautoscaler.autoscaling/flask-app autoscaled
master@master-vm:~/Desktop/flask-kube$ kubectl get hpa
NAME        REFERENCE              TARGETS            MINPODS   MAXPODS   REPLICAS   AGE
flask-app   Deployment/flask-app   cpu: <unknown>/50%   3        10        0          3s
master@master-vm:~/Desktop/flask-kube$ kubectl run -it --rm busybox --image=busybox -- /bin/sh
If you don't see a command prompt, try pressing enter.
/ #
/ # ls
bin   dev   etc   home   lib   lib64   proc   root   sys   tmp   usr   var
/ # pwd
/
/ # echo "hi bhargav"
hi bhargav
/ # exit
Session ended, resume using 'kubectl attach busybox -c busybox -i -t' command when the pod is running
pod "busybox" deleted
master@master-vm:~/Desktop/flask-kube$
```

STEP-6. Load Testing & Debugging NodePort Issues

```
master@master-vm:~/Desktop/flask-kube$ wget -q -O- http://10.97.210.48:80
master@master-vm:~/Desktop/flask-kube$ kubectl get svc flask-service
NAME            TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
flask-service   NodePort   10.106.68.34    <none>        80:32665/TCP   25m
```

STEP-7. Simulating Load for HPA

```
master@master-vm:~$ minikube start
😄  minikube v1.35.0 on Ubuntu 20.04
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
    Verifying Kubernetes components...
    ■ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: storage-provisioner, default-storageclass

❗  /usr/bin/kubectl is version 1.28.15, which may have incompatibilities with Kubernetes 1.32.0.
    ■ Want kubectl v1.32.0? Try 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
master@master-vm:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   27h   v1.32.0
master@master-vm:~$ kubectl get svc flask-service
NAME            TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
flask-service   NodePort   10.106.68.34    <none>        80:32665/TCP   77m
master@master-vm:~$ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
flask-service   NodePort    10.106.68.34    <none>        80:32665/TCP   77m
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        27h
master@master-vm:~$ minikube ip
192.168.49.2
master@master-vm:~$ kubectl run -it --rm load-generator --image=busybox -- /bin/sh
If you don't see a command prompt, try pressing enter.
/ # while true; do wget -q -O- http://192.168.49.2:32665; done
{"message":"Hello, World! This is a Flask app running in Docker."}
{"message":"Hello, World! This is a Flask app running in Docker."}
{"message":"Hello, World! This is a Flask app running in Docker."}
{"message":"Hello, World! This is a Flask app running in Docker."}
{"message":"Hello, World! This is a Flask app running in Docker."}
{"message":"Hello, World! This is a Flask app running in Docker."}
```

OUTPUT: