**Deploying a Flask Application on Kubernetes with Auto-Scaling & Load Testing**

**Overview**

This guide walks through deploying a **Flask application** on **Kubernetes**, handling **authentication issues**, enabling **auto-scaling (HPA)**, and performing **load testing**.

**Prerequisites**

Ensure you have the following installed:

* **Docker** (for building images)
* **Kubernetes cluster** (with master-vm, worker1-vm, worker2-vm)
* **Metrics Server** (for auto-scaling)

**1️.Building & Containerizing the Flask Application**

**Flask Application (app.py)**

from flask import Flask, jsonify

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return jsonify(message="Hello, World! This is a Flask app running in Docker.")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

**Issue:** Flask bound to 127.0.0.1 won't be accessible. **Fix:** Use app.run(host="0.0.0.0", port=5000).

**Dockerfile**

FROM python:3.11

WORKDIR /app

COPY . /app

RUN pip install flask

EXPOSE 5000

CMD ["python", "app.py"]

**Build & Push Image**

docker build -t kpkm25/flask-kube .

docker push kpkm25/flask-kube

**2️.Deploying Flask App on Kubernetes**

**Deployment & Service YAML (deployment-service.yaml)**

apiVersion: apps/v1

kind: Deployment

metadata:

name: flask-app

spec:

replicas: 3

selector:

matchLabels:

app: flask-app

template:

metadata:

labels:

app: flask-app

spec:

containers:

- name: flask-container

image: kpkm25/flask-kube:latest

ports:

- containerPort: 5000

resources:

requests:

cpu: "100m"

limits:

cpu: "250m"

imagePullSecrets:

- name: docker-secret

---

apiVersion: v1

kind: Service

metadata:

name: flask-service

spec:

selector:

app: flask-app

ports:

- protocol: TCP

port: 80

targetPort: 5000

type: NodePort

**Issue:** ErrImagePull due to unauthenticated Docker pulls. **Fix:** Authenticate Kubernetes with Docker Hub.

**Apply Deployment**

kubectl apply -f deployment-service.yaml

**3️.Fixing Docker Hub Rate Limits (Authentication Issue)**

**Issue:**

Failed to pull image "curlimages/curl": toomanyrequests: You have reached your unauthenticated pull rate limit.

**Fix:** Authenticate Kubernetes with Docker Hub.

**Solution: Create Docker Secret**

kubectl create secret docker-registry docker-secret \

--docker-server=https://index.docker.io/v1/ \

--docker-username=kpkm25 \

--docker-password=YOUR\_DOCKER\_HUB\_PASSWORD \

--docker-email=YOUR\_EMAIL

**Patch Default Service Account**

kubectl patch serviceaccount default -p '{"imagePullSecrets": [{"name": "docker-secret"}]}'

**Fix applied!** Now Kubernetes will authenticate with Docker Hub and avoid rate limits.

**4️.Installing & Troubleshooting Metrics Server**

kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

**Issue:** x509: certificate signed by unknown authority

**Fix:** Check the logs:  
kubectl logs -n kube-system deployment/metrics-server

Edit metrics-server deployment and add:

kubectl edit deployment -n kube-system metrics-server

In containers description and add:

- --kubelet-insecure-tls

kubectl rollout restart deployment -n kube-system metrics-server

**5️.Enabling HPA (Horizontal Pod Autoscaler)**

kubectl autoscale deployment flask-app --cpu-percent=50 --min=3 --max=10

kubectl get hpa

**6️.Load Testing & Debugging NodePort Issues**

**Testing Service Internally**

kubectl run -it --rm busybox --image=busybox -- /bin/sh

wget -q -O- http://10.97.210.48:80

**Finding NodePort & Testing External Access**

kubectl get svc flask-service

**7️.Simulating Load for HPA**

kubectl run -it --rm load-generator --image=busybox -- /bin/sh

while true; do wget -q -O- http://192.168.147.129:30455; done

**Check Scaling**

kubectl get hpa

kubectl get pods

Outputs:  
