Case Study 1: High Availability Microservices Deployment

Scenario:

A growing **e-commerce startup** faces frequent website crashes during peak traffic. They need a scalable solution.

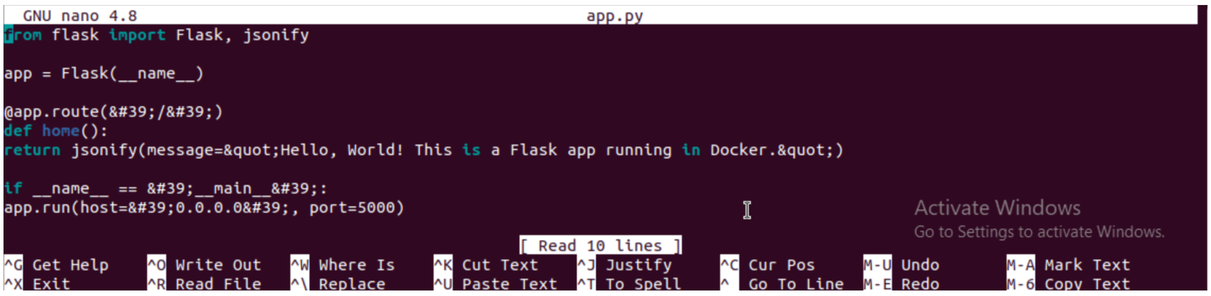
Solution:

Deploy **Product & Cart services** with Kubernetes
Implement **Horizontal Pod Autoscaler (HPA)** to scale dynamically.

Plugins Installation:

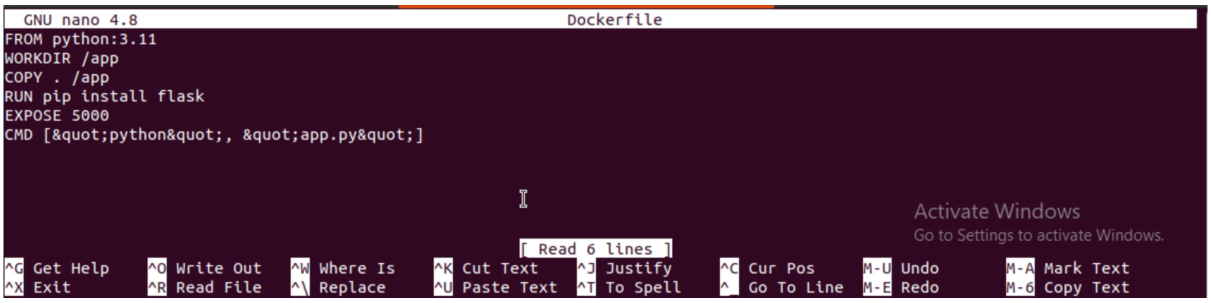
- **Docker** (for building images)
- **Kubernetes cluster** (with master-vm, worker1-vm, worker2-vm)
- **Metrics Server** (for auto-scaling)

Step 1: Building & Containerizing the Flask Application.



```
  GNU nano 4.8                                       app.py
from flask import Flask, jsonify

app = Flask(__name__)

@app.route(&#39;/&#39;)
def home():
return jsonify(message=&quot;Hello, World! This is a Flask app running in Docker.&quot;)

if __name__ == &#39;__main__&#39;:
app.run(host=&#39;0.0.0.0&#39;, port=5000)

                                                  [ Read 10 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^  Go To Line M-E Redo      M-6 Copy Text
```

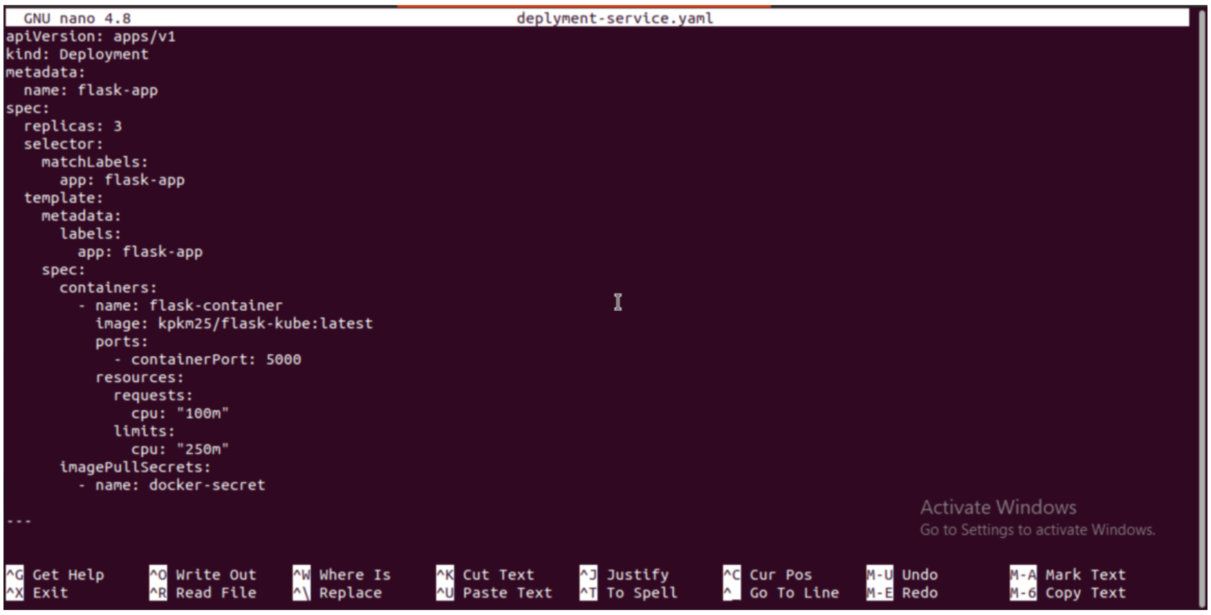
Step 2: Dockerfile



```
  GNU nano 4.8                                     Dockerfile
FROM python:3.11
WORKDIR /app
COPY . /app
RUN pip install flask
EXPOSE 5000
CMD [&quot;python&quot;, &quot;app.py&quot;]

                                                  [ Read 6 lines ]
^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo      M-A Mark Text
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^  Go To Line M-E Redo      M-6 Copy Text
```

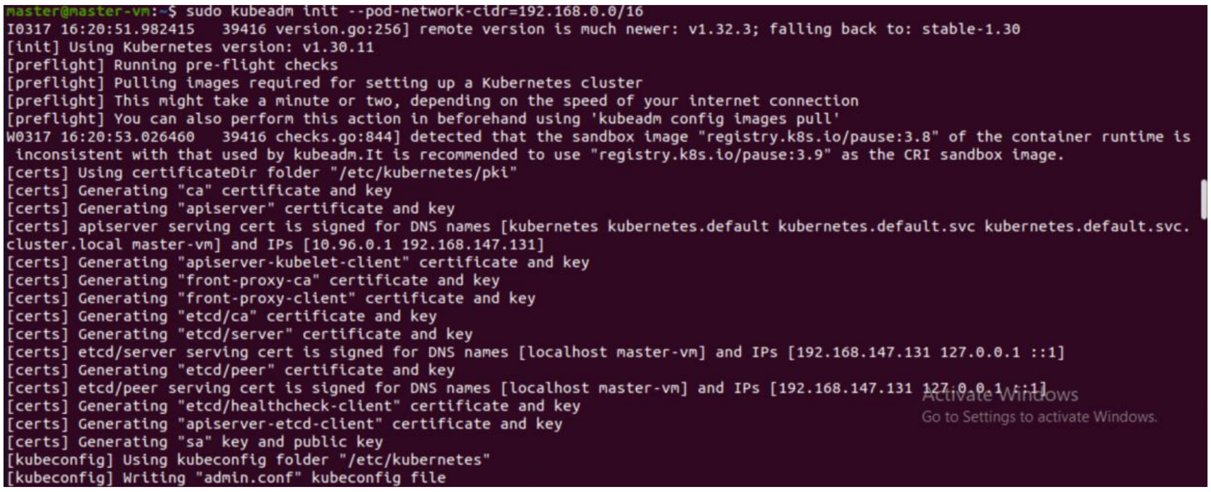
Step 3: Deploying Flask App on Kubernetes.

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



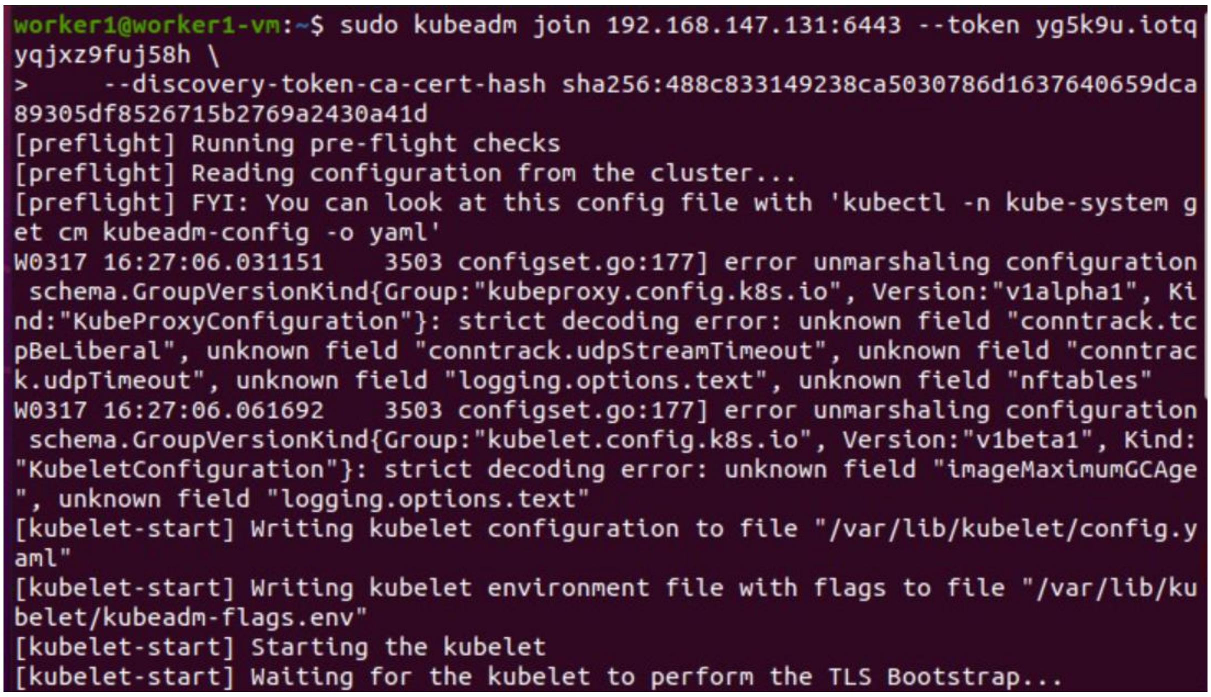
```
  GNU nano 4.8                          deplyment-service.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: flask-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: flask-app
  template:
    metadata:
      labels:
        app: flask-app
    spec:
      containers:
        - name: flask-container
          image: kpkm25/flask-kube:latest
          ports:
            - containerPort: 5000
          resources:
            requests:
              cpu: "100m"
            limits:
              cpu: "250m"
      imagePullSecrets:
        - name: docker-secret

---

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos     M-U Undo       M-A Mark Text
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spell    ^_ Go To Line  M-E Redo       M-6 Copy Text
```

Step 4: Initially connecting the Master Node with the worker node 1. Using the below Token.



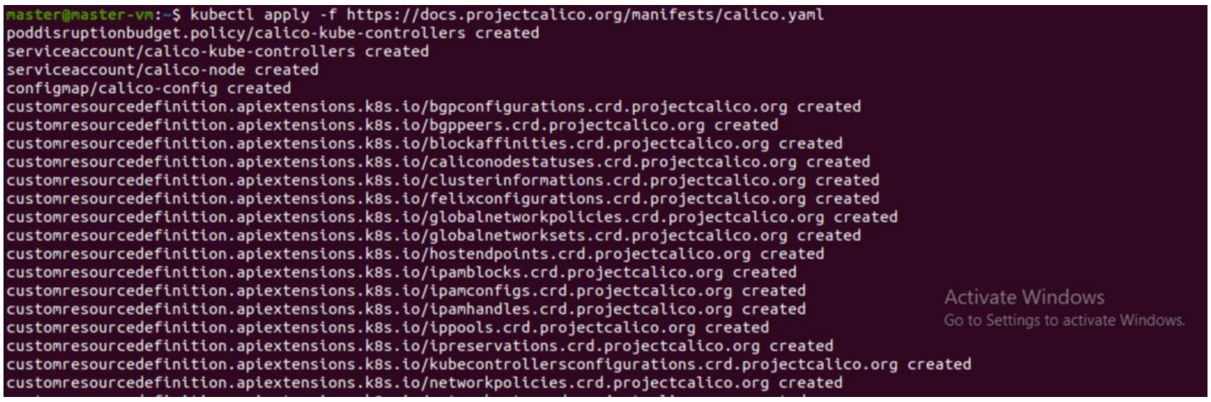
```
master@master-vm:~$ sudo kubeadm init --pod-network-cidr=192.168.0.0/16
I0317 16:20:51.982415   39416 version.go:256] remote version is much newer: v1.32.3; falling back to: stable-1.30
[init] Using Kubernetes version: v1.30.11
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0317 16:20:53.026460   39416 checks.go:844] detected that the sandbox image "registry.k8s.io/pause:3.8" of the container runtime is
 inconsistent with that used by kubeadm.It is recommended to use "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.
cluster.local master-vm] and IPs [10.96.0.1 192.168.147.131]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [localhost master-vm] and IPs [192.168.147.131 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [localhost master-vm] and IPs [192.168.147.131 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
```

Step 5: Worker Node 1 is connected.



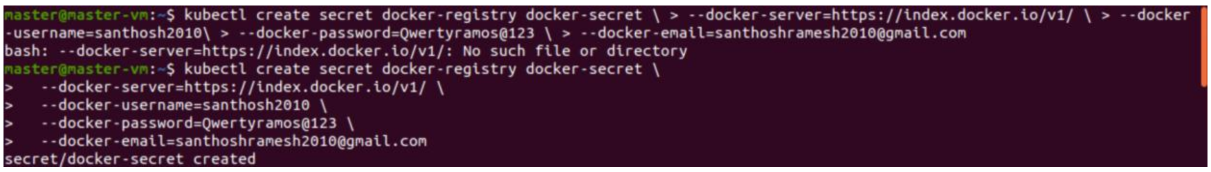
```
worker1@worker1-vm:~$ sudo kubeadm join 192.168.147.131:6443 --token yg5k9u.iotq
yqjxz9fuj58h \
>     --discovery-token-ca-cert-hash sha256:488c833149238ca5030786d1637640659dca
89305df8526715b2769a2430a41d
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system g
et cm kubeadm-config -o yaml'
W0317 16:27:06.031151    3503 configset.go:177] error unmarshaling configuration
 schema.GroupVersionKind{Group:"kubeproxy.config.k8s.io", Version:"v1alpha1", Ki
nd:"KubeProxyConfiguration"}: strict decoding error: unknown field "conntrack.tc
pBeLiberal", unknown field "conntrack.udpStreamTimeout", unknown field "conntrac
k.udpTimeout", unknown field "logging.options.text", unknown field "nftables"
W0317 16:27:06.061692    3503 configset.go:177] error unmarshaling configuration
 schema.GroupVersionKind{Group:"kubelet.config.k8s.io", Version:"v1beta1", Kind:
"KubeletConfiguration"}: strict decoding error: unknown field "imageMaximumGCAge
", unknown field "logging.options.text"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.y
aml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/ku
belet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
```

Step 6: Applying the Deployment**.**



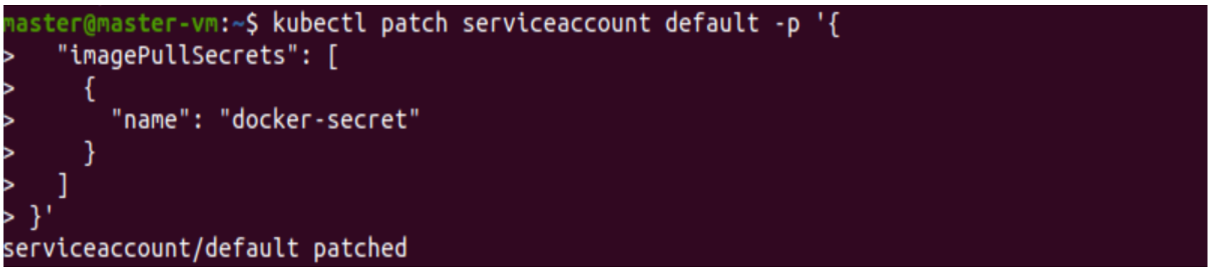
```
master@master-vm:~$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
poddisruptionbudget.policy/calico-kube-controllers created
serviceaccount/calico-kube-controllers created
serviceaccount/calico-node created
configmap/calico-config created
customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
```

Step 7: Fixing Docker Hub Rate Limits (Authentication Issue).



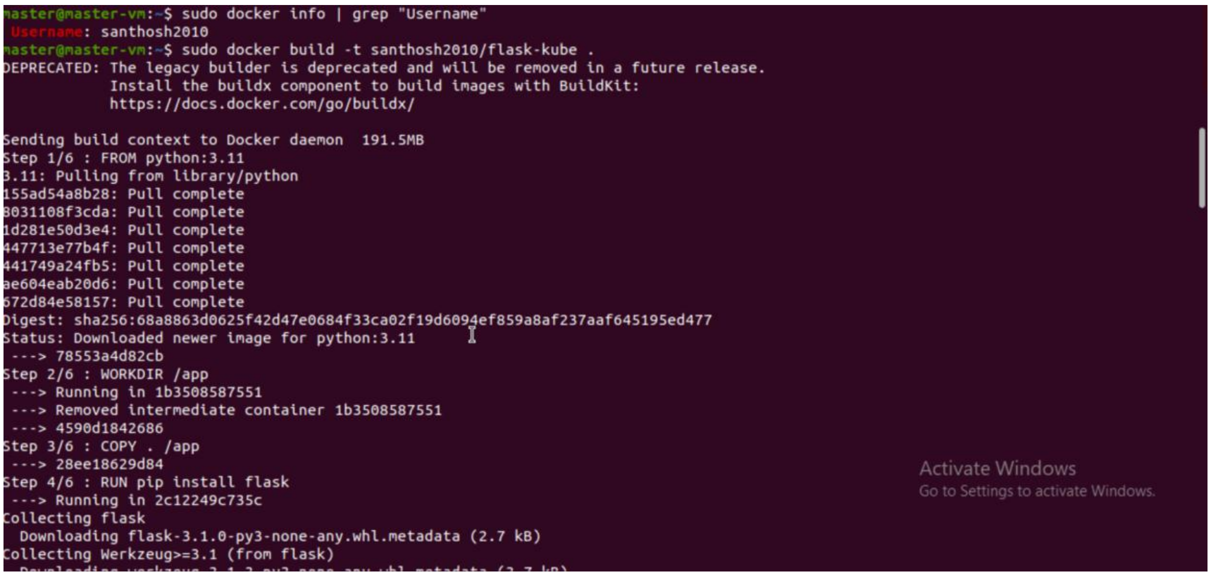
```
master@master-vm:~$ kubectl create secret docker-registry docker-secret \ > --docker-server=https://index.docker.io/v1/ \ > --docker
-username=santhosh2010\ > --docker-password=Qwertyramos@123 \ > --docker-email=santhoshramesh2010@gmail.com
bash: --docker-server=https://index.docker.io/v1/: No such file or directory
master@master-vm:~$ kubectl create secret docker-registry docker-secret \
>   --docker-server=https://index.docker.io/v1/ \
>   --docker-username=santhosh2010 \
>   --docker-password=Qwertyramos@123 \
>   --docker-email=santhoshramesh2010@gmail.com
secret/docker-secret created
```

Step 8: Patch Default Service Account.



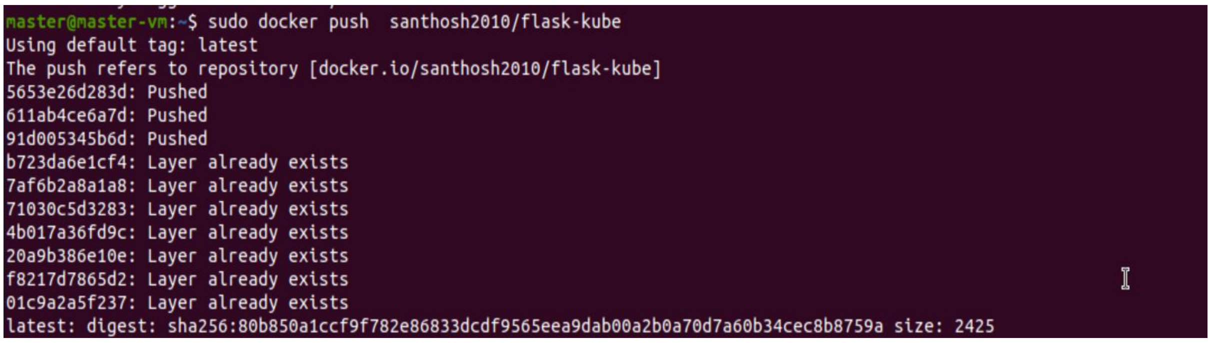
```
master@master-vm:~$ kubectl patch serviceaccount default -p '{
>   "imagePullSecrets": [
>     {
>       "name": "docker-secret"
>     }
>   ]
> }'
serviceaccount/default patched
```

Step 9: Building the Docker Image.



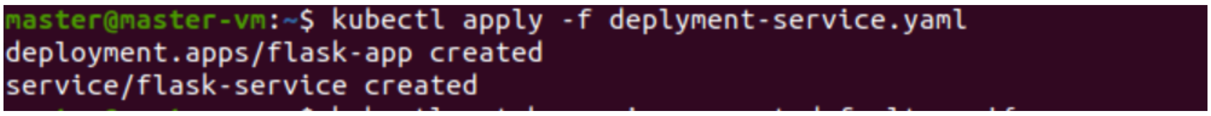
```
master@master-vm:~$ sudo docker info | grep "Username"
 Username: santhosh2010
master@master-vm:~$ sudo docker build -t santhosh2010/flask-kube .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  191.5MB
Step 1/6 : FROM python:3.11
3.11: Pulling from library/python
155ad54a8b28: Pull complete
8031108f3cda: Pull complete
1d281e50d3e4: Pull complete
447713e77b4f: Pull complete
441749a24fb5: Pull complete
ae604eab20d6: Pull complete
672d84e58157: Pull complete
Digest: sha256:68a8863d0625f42d47e0684f33ca02f19d6094ef859a8af237aaf645195ed477
Status: Downloaded newer image for python:3.11
 ---> 78553a4d82cb
Step 2/6 : WORKDIR /app
 ---> Running in 1b3508587551
 ---> Removed intermediate container 1b3508587551
 ---> 4590d1842686
Step 3/6 : COPY . /app
 ---> 28ee18629d84
Step 4/6 : RUN pip install flask
 ---> Running in 2c12249c735c
Collecting flask
  Downloading flask-3.1.0-py3-none-any.whl.metadata (2.7 kB)
Collecting Werkzeug>=3.1 (from flask)
```

Step 10: Pushing the Docker Image.



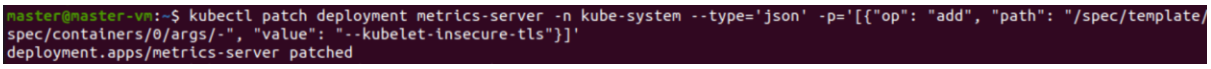
```
master@master-vm:~$ sudo docker push  santhosh2010/flask-kube
Using default tag: latest
The push refers to repository [docker.io/santhosh2010/flask-kube]
5653e26d283d: Pushed
611ab4ce6a7d: Pushed
91d005345b6d: Pushed
b723da6e1cf4: Layer already exists
7af6b2a8a1a8: Layer already exists
71030c5d3283: Layer already exists
4b017a36fd9c: Layer already exists
20a9b386e10e: Layer already exists
f8217d7865d2: Layer already exists
01c9a2a5f237: Layer already exists
latest: digest: sha256:80b850a1ccf9f782e86833dcdf9565eea9dab00a2b0a70d7a60b34cec8b8759a size: 2425
```

Step 11: Installing & Troubleshooting Metrics Server.



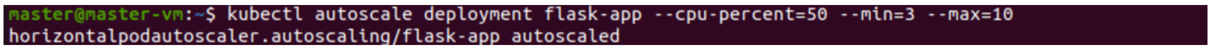
```
master@master-vm:~$ kubectl apply -f deplyment-service.yaml
deployment.apps/flask-app created
service/flask-service created
```

Step 12: Installing & Troubleshooting Metrics Server.



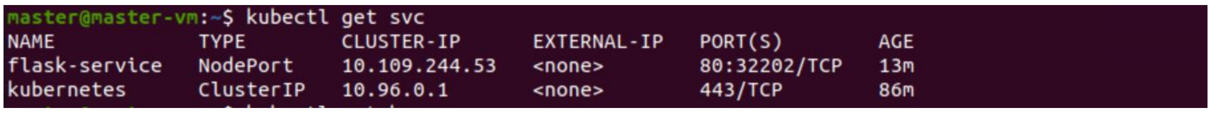
```
master@master-vm:~$ kubectl patch deployment metrics-server -n kube-system --type='json' -p='[{"op": "add", "path": "/spec/template/
spec/containers/0/args/-", "value": "--kubelet-insecure-tls"}]'
deployment.apps/metrics-server patched
```

Step 13: Enabling HPA (Horizontal Pod Auto-scaler).



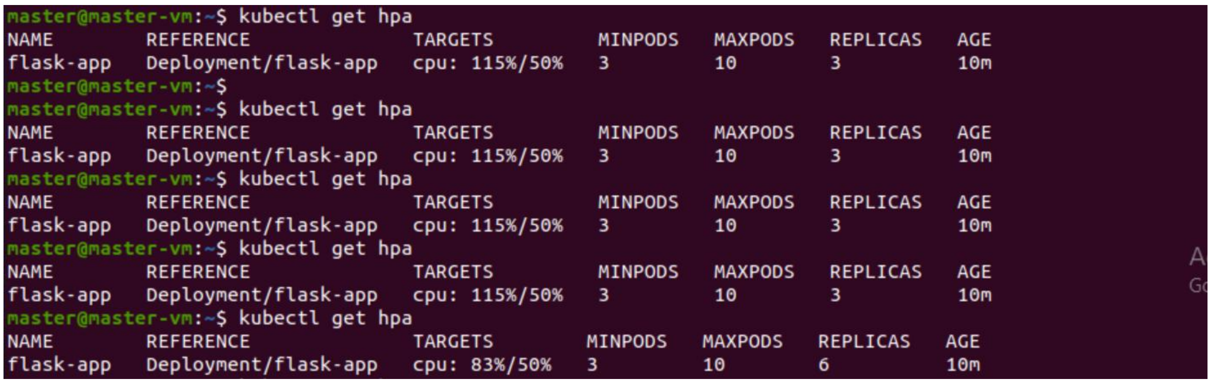
```
master@master-vm:~$ kubectl autoscale deployment flask-app --cpu-percent=50 --min=3 --max=10
horizontalpodautoscaler.autoscaling/flask-app autoscaled
```

Step 14: Seeing the Service of the Deployment.



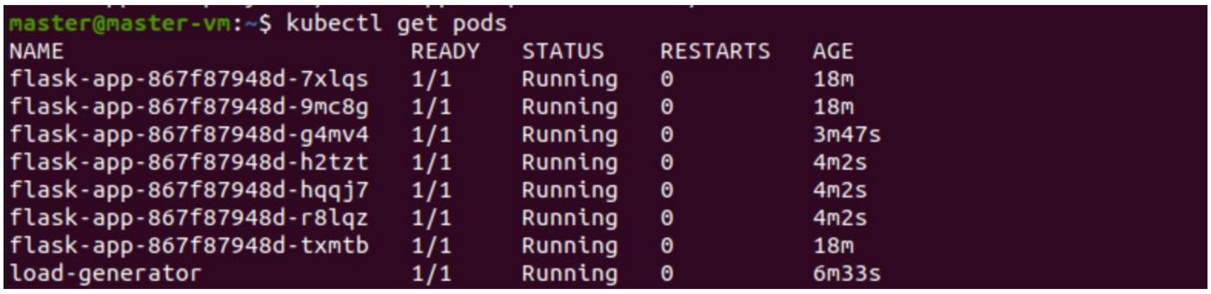
```
master@master-vm:~$ kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
flask-service   NodePort    10.109.244.53   <none>        80:32202/TCP   13m
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        86m
```

Step 15: Simulating Load for HPA.



```
master@master-vm:~$ kubectl get hpa
NAME        REFERENCE              TARGETS        MINPODS   MAXPODS   REPLICAS   AGE
flask-app   Deployment/flask-app   cpu: 115%/50%   3         10        3          10m
master@master-vm:~$
master@master-vm:~$ kubectl get hpa
NAME        REFERENCE              TARGETS        MINPODS   MAXPODS   REPLICAS   AGE
flask-app   Deployment/flask-app   cpu: 115%/50%   3         10        3          10m
master@master-vm:~$ kubectl get hpa
NAME        REFERENCE              TARGETS        MINPODS   MAXPODS   REPLICAS   AGE
flask-app   Deployment/flask-app   cpu: 115%/50%   3         10        3          10m
master@master-vm:~$ kubectl get hpa
NAME        REFERENCE              TARGETS        MINPODS   MAXPODS   REPLICAS   AGE
flask-app   Deployment/flask-app   cpu: 115%/50%   3         10        3          10m
master@master-vm:~$ kubectl get hpa
NAME        REFERENCE              TARGETS       MINPODS   MAXPODS   REPLICAS   AGE
flask-app   Deployment/flask-app   cpu: 83%/50%   3         10        6          10m
```

Step 16: Checking the Scaling.



```
master@master-vm:~$ kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
flask-app-867f87948d-7xlqs   1/1     Running   0          18m
flask-app-867f87948d-9mc8g   1/1     Running   0          18m
flask-app-867f87948d-g4mv4   1/1     Running   0          3m47s
flask-app-867f87948d-h2tzt   1/1     Running   0          4m2s
flask-app-867f87948d-hqqj7   1/1     Running   0          4m2s
flask-app-867f87948d-r8lqz   1/1     Running   0          4m2s
flask-app-867f87948d-txmtb   1/1     Running   0          18m
load-generator               1/1     Running   0          6m33s
```