Kubeadm Installation

Apply Below Commands on Both Worker and Master.

```
$sudo su

#apt update -y

#apt-get install docker.io -y

#systemctl start docker

#systemctl enable docker

#curl -fsSL "https://packages.cloud.google.com/apt/doc/apt-key.gpg" | sudo gpg --dearmor -o
/etc/apt/trusted.gpg.d/kubernetes-archive-keyring.gpg

#echo 'deb https://packages.cloud.google.com/apt kubernetes-xenial main' >
/etc/apt/sources.list.d/kubernetes.list

#apt update -y;apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
```

→To connect with cluster execute above commands on master node and worker node respectively.

Master node

```
$sudo su

#kubeadm init
```

→ To start using your cluster, you need to run the following as a regular user:

```
  $mkdir -p $HOME/.kube

  $sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

  $sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

→ Alternatively, if you are the root user, you can run:

```
#export KUBECONFIG=/etc/kubernetes/admin.conf

#kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-
k8s.yaml

#kubeadm token create --print-join-command
```

Note:-Expose port 6443 in the Security group for the Worker to connect to the Master Node

Worker node

#kubeadm reset pre-flight checks

- ➔ Paste the Join command on worker node and append `--v=5` at end
- ➔ To verify cluster connection

On master node

# kubectl get nodes

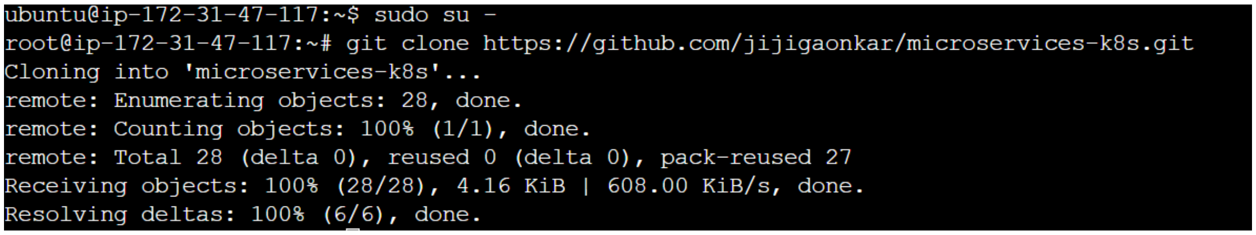
1. Deployment of a Microservices Application on K8s

- Do Mongo Db Deployment

- Do Flask App Deployment

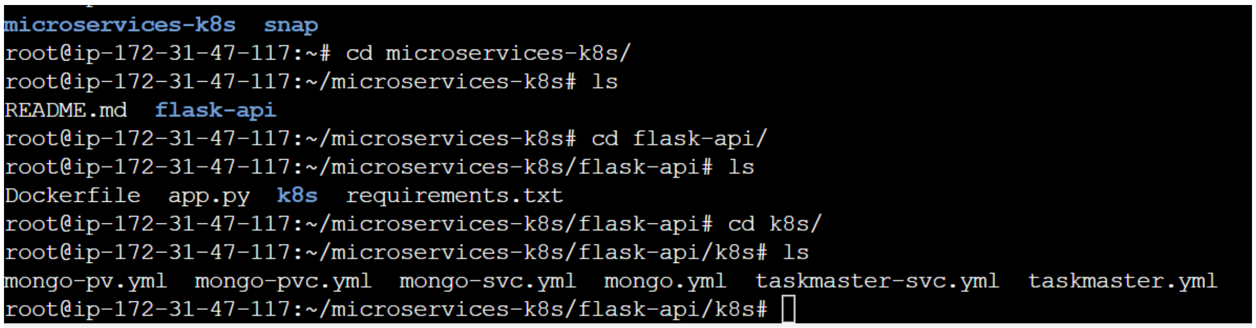
- Connect both using Service Discovery

After installing your kubeadm on the master clone your microservice-k8s in your master local machine
git clone https://github.com/jijigaonkar/microservices-k8s.git



```
ubuntu@ip-172-31-47-117:~$ sudo su -
root@ip-172-31-47-117:~# git clone https://github.com/jijigaonkar/microservices-k8s.git
Cloning into 'microservices-k8s'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 28 (delta 0), reused 0 (delta 0), pack-reused 27
Receiving objects: 100% (28/28), 4.16 KiB | 608.00 KiB/s, done.
Resolving deltas: 100% (6/6), done.
```

After that enter into that repository of microservices-k8s, flask-api, k8s if you list ls that file you will find some yml files which we call manifest file


```
microservices-k8s  snap
root@ip-172-31-47-117:~# cd microservices-k8s/
root@ip-172-31-47-117:~/microservices-k8s# ls
README.md  flask-api
root@ip-172-31-47-117:~/microservices-k8s# cd flask-api/
root@ip-172-31-47-117:~/microservices-k8s/flask-api# ls
Dockerfile  app.py  k8s  requirements.txt
root@ip-172-31-47-117:~/microservices-k8s/flask-api# cd k8s/
root@ip-172-31-47-117:~/microservices-k8s/flask-api/k8s# ls
mongo-pv.yml  mongo-pvc.yml  mongo-svc.yml  mongo.yml  taskmaster-svc.yml  taskmaster.yml
root@ip-172-31-47-117:~/microservices-k8s/flask-api/k8s#
```

Create a Kubernetes deployment and service by running the following command:

#kubectl apply -f <yml files.yml>

Mongo-pv.yml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongo-pv
spec:
  capacity:
    storage: 256Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /tmp/db
```

Mongo-pvc.yml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongo-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 256Mi
```

Mongo-svc.yml

```
apiVersion: v1
kind: Service
metadata:
  labels:
    app: mongo
  name: mongo
spec:
  ports:
    - port: 27017
      targetPort: 27017
  selector:
    app: mongo
```

Mongo.yml

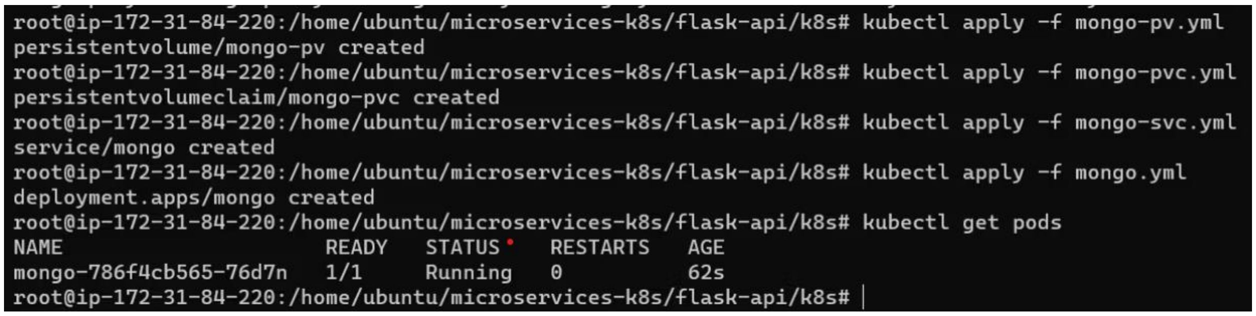
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
  labels:
      app: mongo
spec:
  selector:
    matchLabels:
      app: mongo
  template:
    metadata:
      labels:
        app: mongo
    spec:
      containers:
        - name: mongo
          image: mongo
          ports:
            - contaierPort: 27017
          volumeMounts:
            - name: storage
              mountPath: /data/db
      volumes:
        - name: storage
          persistentVolumeClaim:
```

```
          claimName: mongo-pvc
```

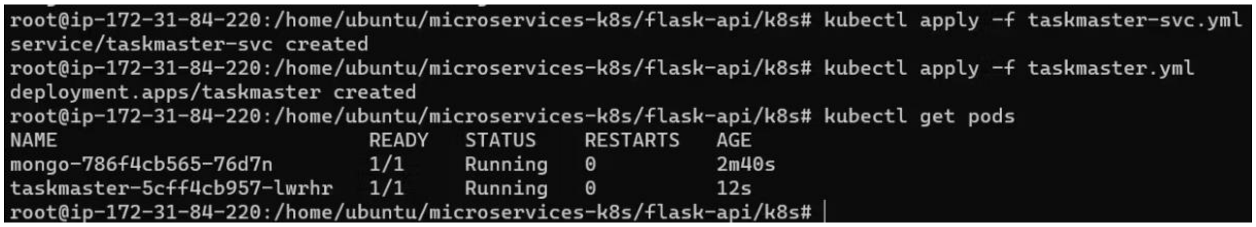
Taskmaster-svc.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: taskmaster
  labels:
    app: taskmaster
spec:
  replicas: 1
  selector:
    matchLabels:
      app: taskmaster
  template:
    metadata:
      labels:
        app: taskmaster
    spec:
      containers:
        - name: taskmaster
          image: nsparthu/microservice-k8s:latest
          ports:
            - containerPort: 5000
          imagePullPolicy: Always
```

After that apply them all using this command kubectl apply -f <yml files.yml>



```
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl apply -f mongo-pv.yml
persistentvolume/mongo-pv created
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl apply -f mongo-pvc.yml
persistentvolumeclaim/mongo-pvc created
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl apply -f mongo-svc.yml
service/mongo created
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl apply -f mongo.yml
deployment.apps/mongo created
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl get pods
NAME                     READY   STATUS    RESTARTS   AGE
mongo-786f4cb565-76d7n   1/1     Running   0          62s
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s#
```

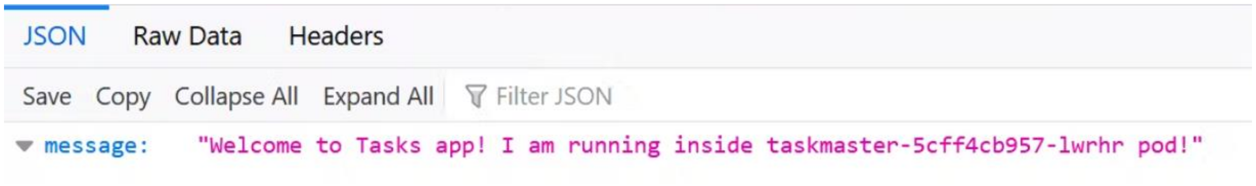


```
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl apply -f taskmaster-svc.yml
service/taskmaster-svc created
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl apply -f taskmaster.yml
deployment.apps/taskmaster created
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s# kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mongo-786f4cb565-76d7n        1/1     Running   0          2m40s
taskmaster-5cff4cb957-lwrhr   1/1     Running   0          12s
root@ip-172-31-84-220:/home/ubuntu/microservices-k8s/flask-api/k8s#
```

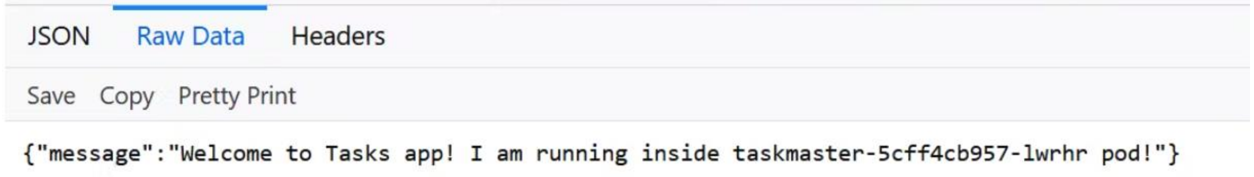
After that go to your AWS account click on master node go to security group and create a port number as per your given in taskmaster.yml

Example:- http://ip.adress:30007

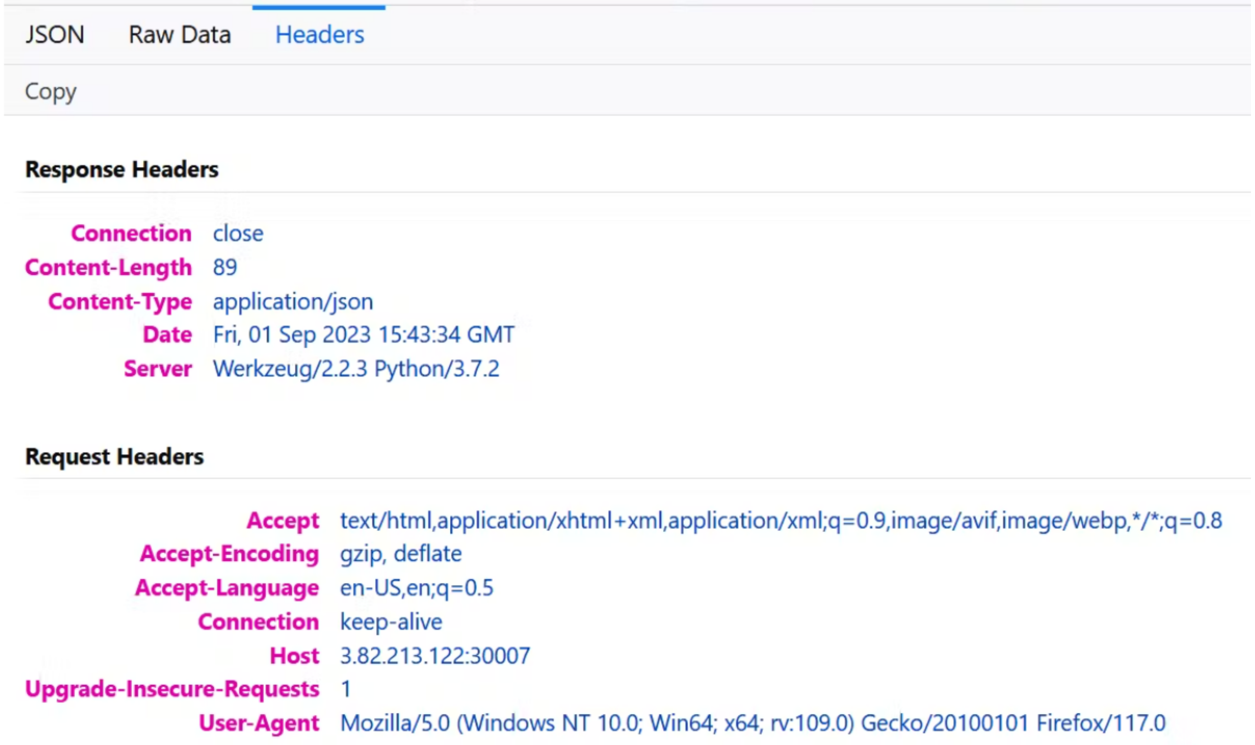
After that open a new tab and past your IP and port number



```
message:    "Welcome to Tasks app! I am running inside taskmaster-5cff4cb957-lwrhr pod!"
```



```
{"message":"Welcome to Tasks app! I am running inside taskmaster-5cff4cb957-lwrhr pod!"}
```


JSON Raw Data Headers

Copy

Response Headers

Connection	close
Content-Length	89
Content-Type	application/json
Date	Fri, 01 Sep 2023 15:43:34 GMT
Server	Werkzeug/2.2.3 Python/3.7.2

Request Headers

Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Encoding	gzip, deflate
Accept-Language	en-US,en;q=0.5
Connection	keep-alive
Host	3.82.213.122:30007
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/117.0

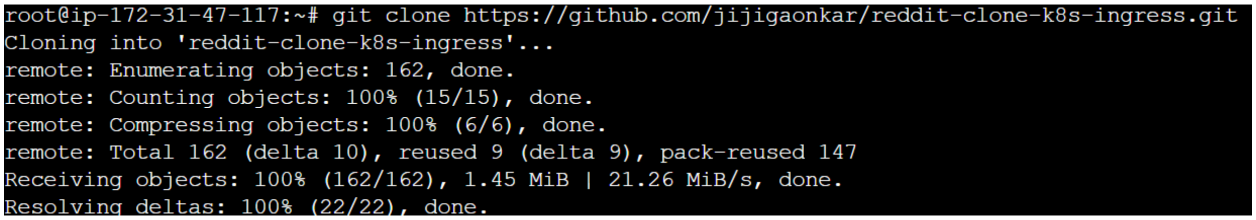
2. Deployment of a Reddit-Clone Application

- Do Deployment of the Reddit Clone app

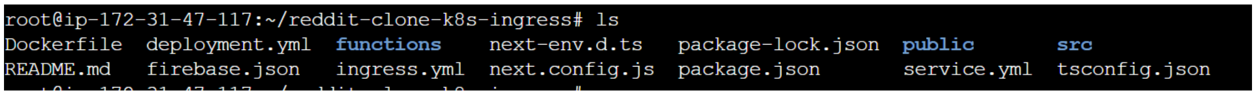
- Write an ingress controller for the same to give a custom route

Same here Also install your kubeadm on the master and Worker then in master clone your Reddit-clone-k8s-ingress in your master local machine

git clone https://github.com/jijigaonkar/reddit-clone-k8s-ingress.git


```
root@ip-172-31-47-117:~# git clone https://github.com/jijigaonkar/reddit-clone-k8s-ingress.git
Cloning into 'reddit-clone-k8s-ingress'...
remote: Enumerating objects: 162, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 162 (delta 10), reused 9 (delta 9), pack-reused 147
Receiving objects: 100% (162/162), 1.45 MiB | 21.26 MiB/s, done.
Resolving deltas: 100% (22/22), done.
```

After that enter into that repository of Reddit-clone-k8s-ingress if you list ls that file you will find some yml files which we call manifest file

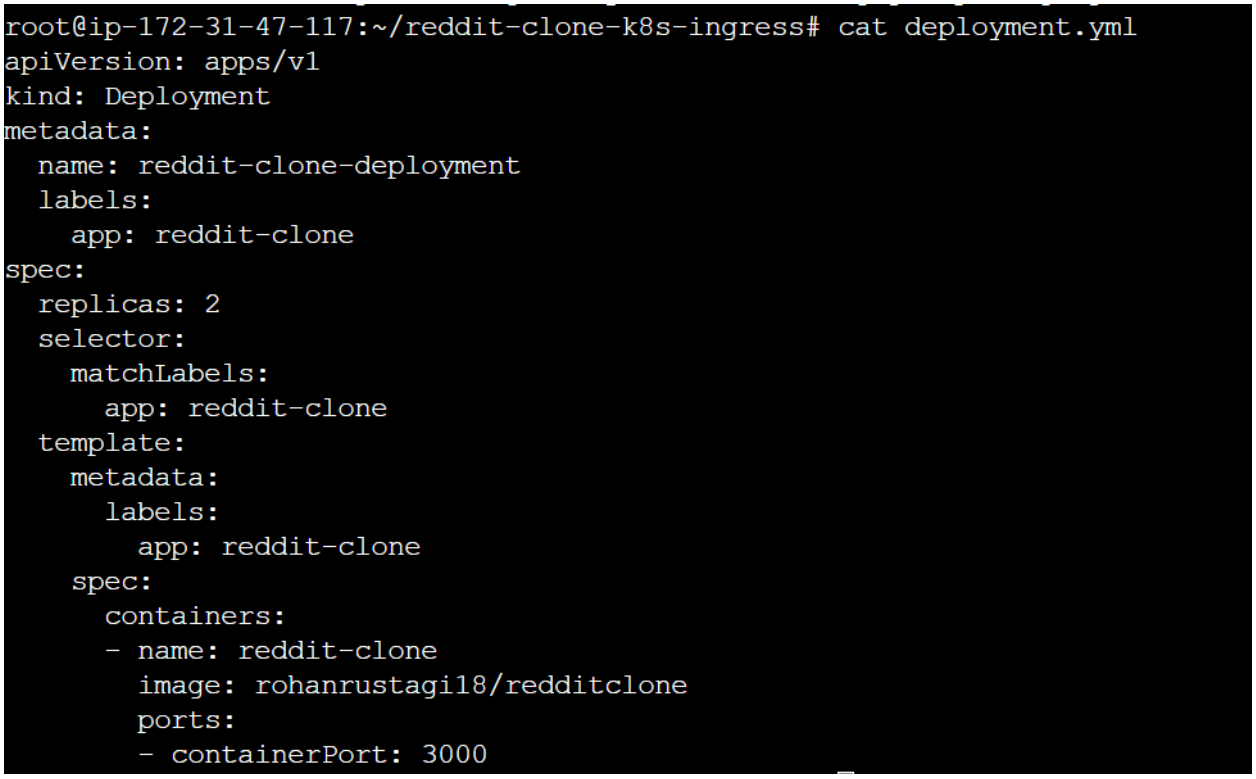


```
root@ip-172-31-47-117:~/reddit-clone-k8s-ingress# ls
Dockerfile   deployment.yml  functions    next-env.d.ts    package-lock.json  public       src
README.md    firebase.json   ingress.yml  next.config.js   package.json       service.yml  tsconfig.json
```

Create a Kubernetes deployment and service by running the following command:

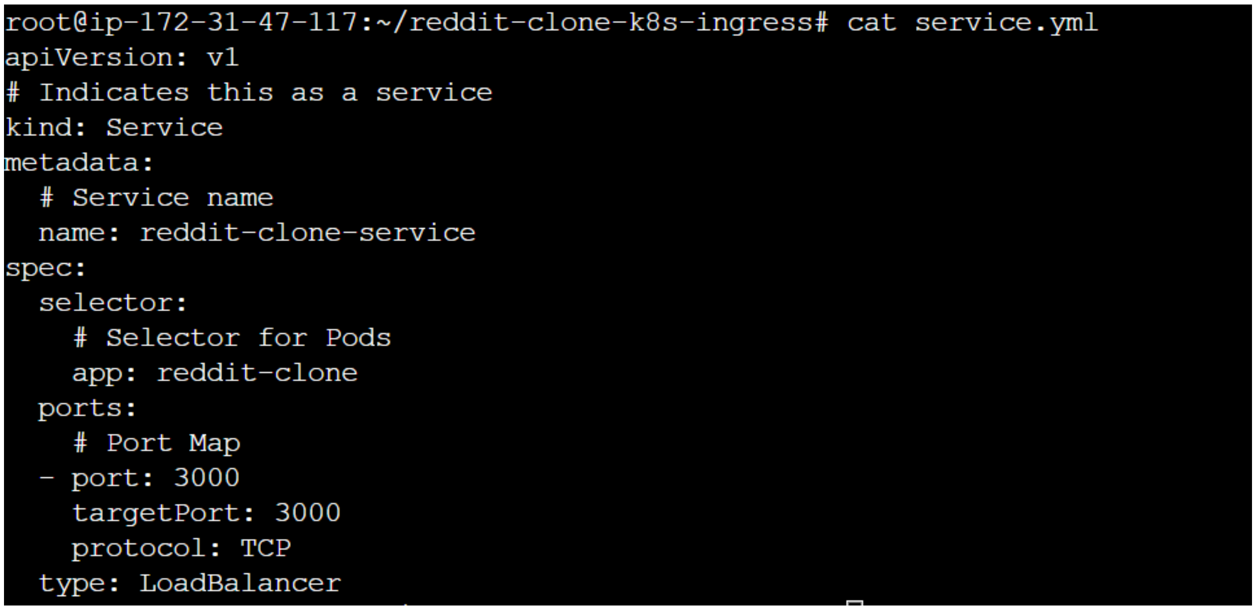
kubectl apply -f <yml files.yml> briefly you can understand in photos. Before adding photos let's see yml file of Reddit-clone-k8s-ingress

Deployment.yml



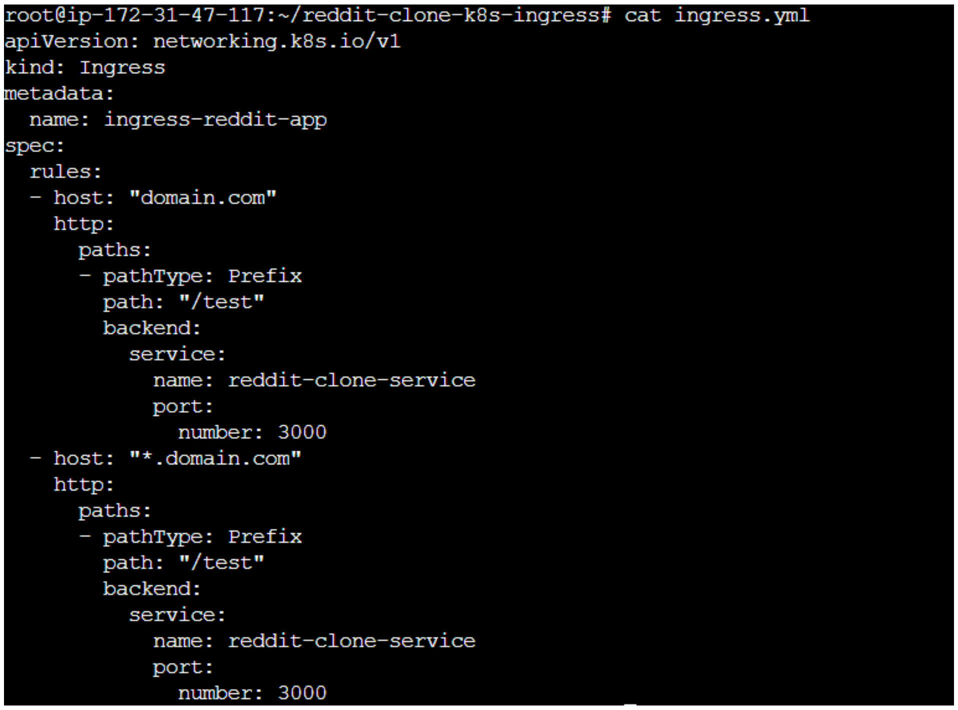
```
root@ip-172-31-47-117:~/reddit-clone-k8s-ingress# cat deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reddit-clone-deployment
  labels:
    app: reddit-clone
spec:
  replicas: 2
  selector:
    matchLabels:
      app: reddit-clone
  template:
    metadata:
      labels:
        app: reddit-clone
    spec:
      containers:
      - name: reddit-clone
        image: rohanrustagi18/redditclone
        ports:
        - containerPort: 3000
```

Service.yml



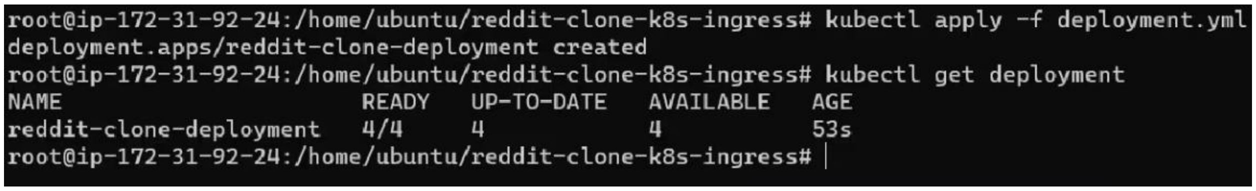
```
root@ip-172-31-47-117:~/reddit-clone-k8s-ingress# cat service.yml
apiVersion: v1
# Indicates this as a service
kind: Service
metadata:
  # Service name
  name: reddit-clone-service
spec:
  selector:
    # Selector for Pods
    app: reddit-clone
  ports:
    # Port Map
  - port: 3000
    targetPort: 3000
    protocol: TCP
  type: LoadBalancer
```

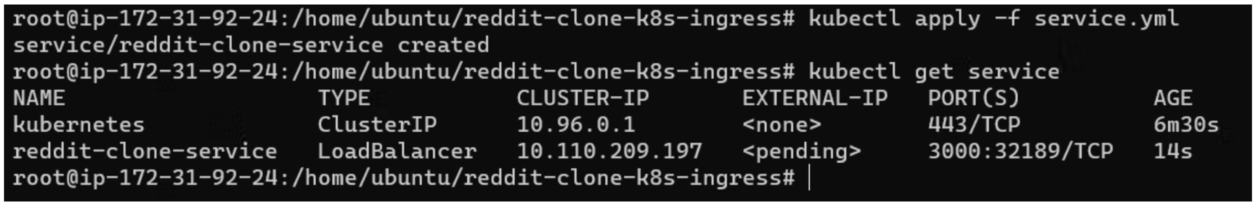
Ingress.yml

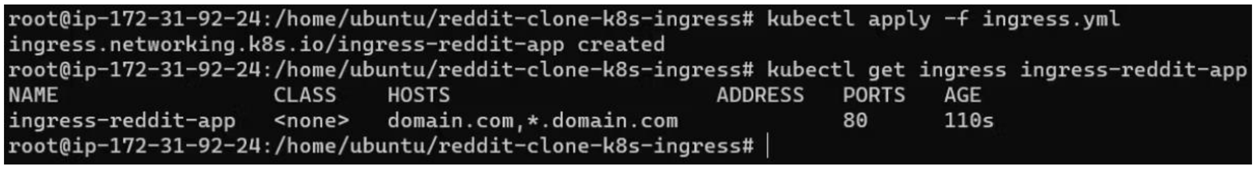


```
root@ip-172-31-47-117:~/reddit-clone-k8s-ingress# cat ingress.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-reddit-app
spec:
  rules:
  - host: "domain.com"
    http:
      paths:
      - pathType: Prefix
        path: "/test"
        backend:
          service:
            name: reddit-clone-service
            port:
              number: 3000
  - host: "*.domain.com"
    http:
      paths:
      - pathType: Prefix
        path: "/test"
        backend:
          service:
            name: reddit-clone-service
            port:
              number: 3000
```

After that apply them all using this command kubectl apply -f <yml files.yml>

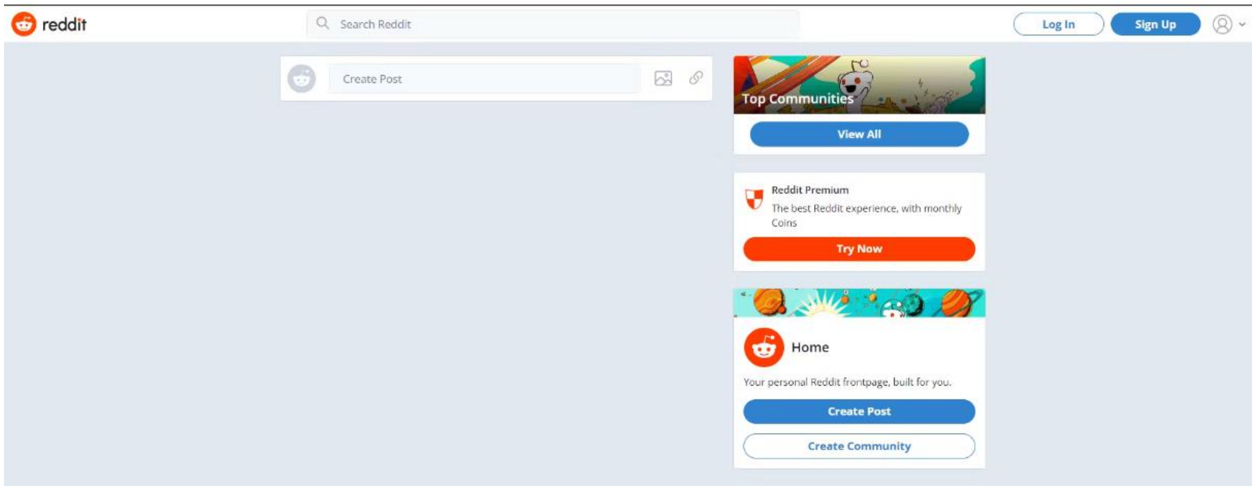

```
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# kubectl apply -f deployment.yml
deployment.apps/reddit-clone-deployment created
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# kubectl get deployment
NAME                      READY   UP-TO-DATE   AVAILABLE   AGE
reddit-clone-deployment   4/4     4            4           53s
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# |
```


```
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# kubectl apply -f service.yml
service/reddit-clone-service created
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# kubectl get service
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
kubernetes             ClusterIP      10.96.0.1        <none>        443/TCP          6m30s
reddit-clone-service   LoadBalancer   10.110.209.197   <pending>     3000:32189/TCP   14s
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# |
```


```
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# kubectl apply -f ingress.yml
ingress.networking.k8s.io/ingress-reddit-app created
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# kubectl get ingress ingress-reddit-app
NAME                 CLASS    HOSTS                        ADDRESS   PORTS   AGE
ingress-reddit-app   <none>   domain.com,*.domain.com                80      110s
root@ip-172-31-92-24:/home/ubuntu/reddit-clone-k8s-ingress# |
```

After that go to your AWS account click on master node go to a security group and create a port number Example:- http://ip.adress:32189

After that open a new tab and past your IP and port number Reddit will open


reddit
Search Reddit
Log In
Sign Up
Create Post
Top Communities
View All
Reddit Premium
The best Reddit experience, with monthly Coins
Try Now
Home
Your personal Reddit frontpage, built for you.
Create Post
Create Community