**Step 1: Set Up a 3-Node Kubernetes Cluster**

1. **Set up Kubernetes Cluster**: You can create a Kubernetes cluster with 3 nodes on a AWS Cloud
2. **Install software of all servers**: kubeadm,and kubectl on mater node

**Using kubeadm :**

1. Initialize the master node:

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

1. Configure kubectl for the root user:

mkdir -p $HOME/.kube

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

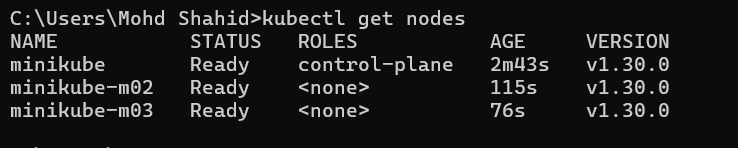
sudo chown $(id -u):$(id -g) $HOME/.kube/config

1. Install a Pod network add-on (like Calico):

kubectl apply -f https://docs.projectcalico.org/v3.14/manifests/calico.yaml

1. Join the other nodes: On each worker node, run the join command (from the output of kubeadm init) to join them to the cluster. If you lost this command, you can regenerate it using:

kubeadm token create --print-join-command



**1. Deploy an Apache2 Deployment with 2 Replicas**

Create a file named apache2-deployment.yaml with the following content:

yaml

Copy code

apiVersion: apps/v1

kind: Deployment

metadata:

name: apache-deployment

labels:

app: apache

spec:

replicas: 2

selector:

matchLabels:

app: apache

template:

metadata:

labels:

app: apache

spec:

containers:

- name: apache

image: httpd:latest

ports:

- containerPort: 80

Create the Apache service with a file named apache2-service.yaml:

yaml

Copy code

apiVersion: v1

kind: Service

metadata:

name: apache-service

spec:

type: ClusterIP

selector:

app: apache

ports:

- port: 80

targetPort: 80

**Apply these configurations:**

kubectl apply -f apache2-deployment.yaml

kubectl apply -f apache2-service.yaml

**2. Containerize and Deploy the GitHub Application**

**a. Containerize the Code**

1. **Clone the GitHub repository:**

git clone https://github.com/hshar/website.git

cd website

1. **Create a Dockerfile in the repository root directory:**

# Use the official Apache HTTP server image from Docker Hub

FROM httpd:latest

COPY ./website/ /usr/local/apache2/htdocs/

EXPOSE 80

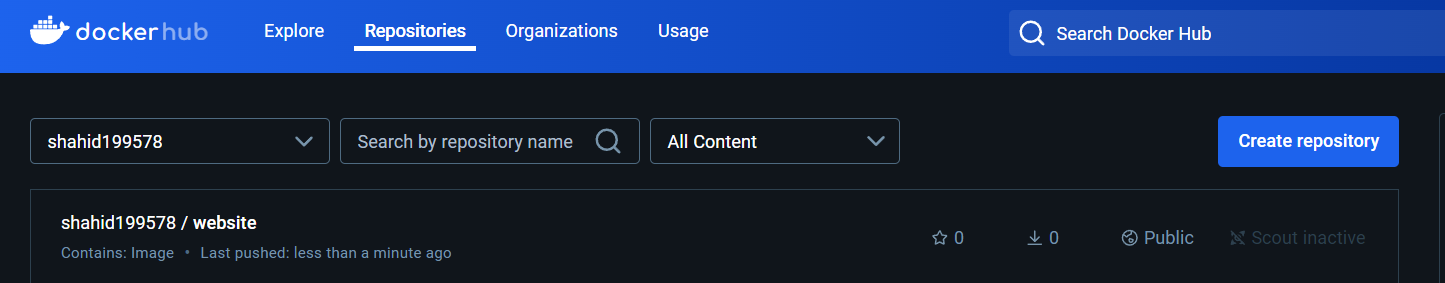
1. **Build the Docker image:**

docker build -t shahid199578/website:latest .

1. **Push the Docker image to Docker Hub:**

docker login

docker push shahid/website:latest

****

**b. Deploy the Containerized Application on Kubernetes**

1. **Deploy an Apache2 Deployment with 2 Replicas**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: Apache2-deployment

  labels:

    app: apache2

spec:

  replicas: 2

  selector:

    matchLabels:

      app: apache2

  template:

    metadata:

      labels:

        app: apache2

    spec:

      containers:

      - name: Apache2

        image: shahid199578/website

        ports:

        - containerPort: 80

1. **Create a Service YAML file named apache-service.yaml:**

apiVersion: v1

kind: Service

metadata:

  name: apache2-service

spec:

  type: ClusterIP

  selector:

    app: apache

  ports:

    - port: 80

      targetPort: 80

1. **Apply these configurations:**

kubectl apply -f apache-deployment.yaml

kubectl apply -f apache-service.yaml

**3. Set Up Ingress**

1. **Create an Ingress Resource YAML file named ingress.yaml:**

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: my-ingress

spec:

  rules:

  - host: localhost

    http:

      paths:

      - path: /apache

        pathType: Prefix

        backend:

          service:

            name: apache-service

            port:

              number: 80

      - path: /custom

        pathType: Prefix

        backend:

          service:

            name: apache2-service

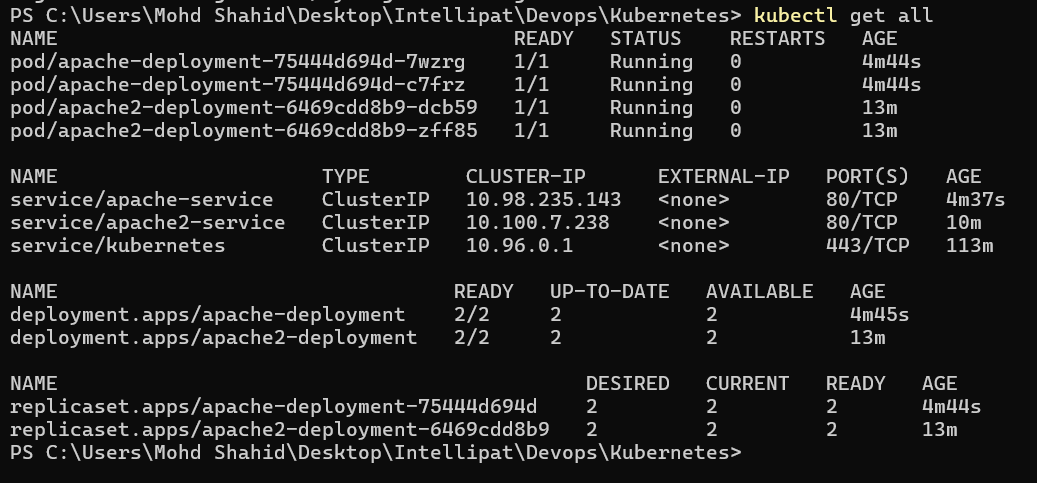
            port:

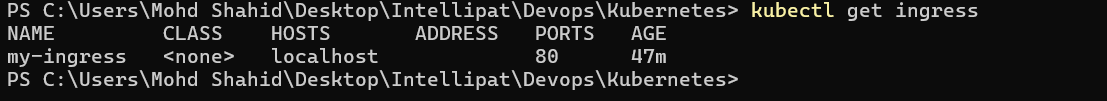
              number: 80

1. **Apply the Ingress configuration:**

kubectl apply -f ingress.yaml

1. **Verify Everything**

****

****

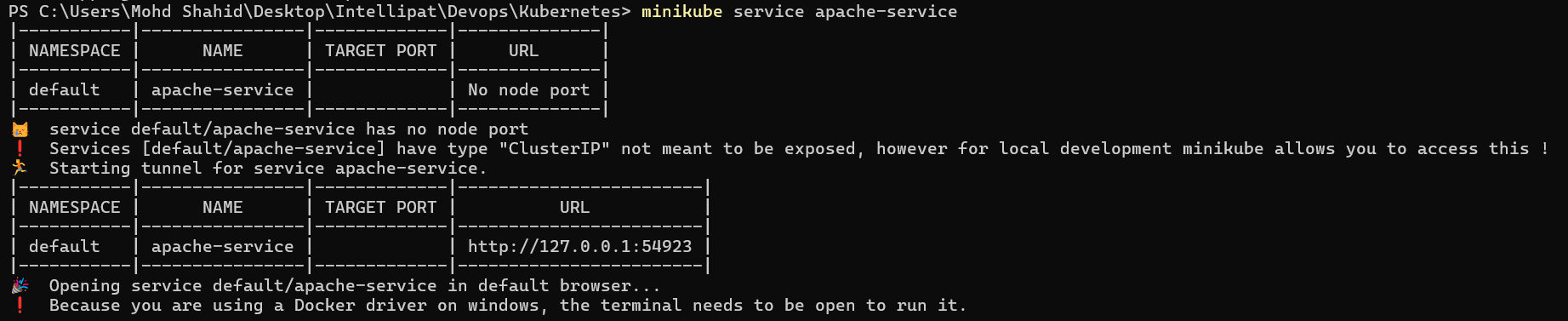
**Update /etc/hosts (if necessary) for local testing:**

Add the following entry to your /etc/hosts file (or C:\Windows\System32\drivers\etc\hosts on Windows):

Copy code

127.0.0.1 localhost

1. **Access the services:**
   * Visit http://localhost/apache to see the Apache service.





* + Visit http://localhost/custom to see the GitHub application.

