**Step 2: Installation script: master Node**

|  |
| --- |
| <https://github.com/dev-ops-source/Installation-Scripts.git> |
| sudo hostnamectl set-hostname Master  bash |

**Step 3: Initialize the Kubernetes Master Node:**

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

|  |
| --- |
| mkdir -p $HOME/.kube  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  sudo chown $(id -u):$(id -g) $HOME/.kube/config |

kubectl apply -f <https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml>

kubeadm token create --print-join-command

**Step 4: Worker Node**

sudo kubeadm reset pre-flight checks

Paste generated join command before thatuse **“sudo”**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Cloud fare Part:**

# Add cloudflare gpg key  
sudo mkdir -p --mode=0755 /usr/share/keyrings  
curl -fsSL <https://pkg.cloudflare.com/cloudflare-main.gpg> | sudo tee /usr/share/keyrings/cloudflare-main.gpg >/dev/null

# Add this repo to your apt repositories  
echo 'deb [signed-by=/usr/share/keyrings/cloudflare-main.gpg] <https://pkg.cloudflare.com/cloudflared> any main' | sudo tee /etc/apt/sources.list.d/cloudflared.list

# install cloudflared  
sudo apt-get update && sudo apt-get install cloudflared

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Updated deployment files on Kubeadm.**

*Vim frontend-deployment.yml*

kind: Service

metadata:

name: frontend-service

spec:

selector:

app: frontend

ports:

- protocol: TCP

port: 80

targetPort: 80

type: NodePort

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: frontend-deployment

spec:

replicas: 1

selector:

matchLabels:

app: frontend

template:

metadata:

labels:

app: frontend

spec:

containers:

- name: frontend

image: pmasne98/qst-frontend:m1

imagePullPolicy: Always

ports:

- containerPort: 80

*Vim backend-deployment.yml*

apiVersion: v1

kind: Service

metadata:

name: springboot-service

spec:

selector:

app: backend

ports:

- protocol: TCP

port: 8080

targetPort: 8080

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: backend-deployment

spec:

replicas: 1

selector:

matchLabels:

app: backend

template:

metadata:

labels:

app: backend

spec:

containers:

- name: backend

image: pmasne98/qst-backend:m1

imagePullPolicy: Always

ports:

- containerPort: 8080

env:

- name: MYSQL\_HOST

value: "mysql-service"

- name: MYSQL\_PORT

value: "3306"

- name: MYSQL\_USER

value: "root"

- name: MYSQL\_PASSWORD

value: "root"

*vim mysql-deployment.yml*

apiVersion: v1

kind: Service

metadata:

name: mysql-service

spec:

ports:

- port: 3306

selector:

app: mysql

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mysql:8

ports:

- containerPort: 3306

env:

- name: MYSQL\_ROOT\_PASSWORD

value: "root"

- name: MYSQL\_DATABASE

value: "quantumsoft"

vim default.conf

server {

listen 80;

server\_name \_;

# Serve Angular application

root /usr/share/nginx/html;

index index.html;

location / {

try\_files $uri $uri/ /index.html;

}

# Proxy API requests to Spring Boot service

location /api {

proxy\_pass http://springboot-service:8080;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

# Error handling for Angular routes

error\_page 404 /index.html;

location = /index.html {

allow all;

}

}

Vim Dockerfile: frontend

FROM nginx:alpine

COPY dist/ /usr/share/nginx/html

RUN chmod -R 755 /usr/share/nginx/html

COPY default.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]

Vim Dockerfile: backend

FROM eclipse-temurin:23-jdk-alpine

WORKDIR /app

COPY QuantomSoft-0.0.1-SNAPSHOT\_7.jar QuantomSoft-0.0.1-SNAPSHOT\_7.jar

EXPOSE 8080

CMD ["java", "-jar", "QuantomSoft-0.0.1-SNAPSHOT\_7.jar"]

Note: check config/Config.js

**Ingress:-**

Ingress is a resource that manages external access to services within a cluster.

**Distributing request*:*** *i.e. /qst & /*

<https://kind.sigs.k8s.io/docs/user/ingress> 🡪documentation link for setup of kind cluster.

* kubectl apply -f <https://kind.sigs.k8s.io/examples/ingress/deploy-ingress-nginx.yaml>
* kubectl get pods -n ingress-nginx

apiVersion: networking.k8s.io/v1 🡨----demo file(ingress.yml)

kind: Ingress

metadata:

name: example-ingress

spec:

rules:

- http:

paths:

- pathType: Prefix

path: /frontend

backend:

service:

name: frontend-service

port:

number: 80

- pathType: Prefix

path: /back

backend:

service:

name: backend-service

port:

number: 8080

* kubectl apply -f ingress.yaml
* kubectl get svc -n ingress-nginx

**Annotations:-**

# ingress.yaml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: app-ingress

annotations: <-- ----only this is a part of Annotation

nginx.ingress.kubernetes.io/rewrite-target: /$1

spec:

ingressClassName: nginx

rules:

- host: your-ec2-app.local

http:

paths:

- path: /(.\*)

pathType: Prefix

backend:

service:

name: frontend-service

port:

number: 80

- path: /api/(.\*)

pathType: Prefix

backend:

service:

name: backend-service

port:

number: 8080

**PV And PVC:-**

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mysql:8

ports:

- containerPort: 3306

env:

- name: MYSQL\_ROOT\_PASSWORD

value: "root"

- name: MYSQL\_DATABASE

value: "quantumsoft"

volumeMounts:

- name: mysql-storage

mountPath: /var/lib/mysql

volumes:

- name: mysql-storage

persistentVolumeClaim:

claimName: example-pvc

**PV:**

apiVersion: v1

kind: PersistentVolume

metadata:

name: example-pv

spec:

capacity:

storage: 5Gi

accessModes:

- ReadWriteOnce

persistentVolumeReclaimPolicy: Retain

storageClassName: standard

hostPath:

path: "/mnt/data"

**PVC:**

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: example-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 5Gi

storageClassName: standard

**ConfigMap & Secret:-**

apiVersion: v1

kind: ConfigMap

metadata:

name: mysql-config

data:

MYSQL\_DATABASE: quantumsoft

**Secret:-**

apiVersion: v1

kind: Secret

metadata:

name: mysql-secret

type: Opaque

stringData:

MYSQL\_ROOT\_PASSWORD: root

apiVersion: v1

kind: Service

metadata:

name: mysql-service

spec:

ports:

- port: 3306

selector:

app: mysql

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: mysql

spec:

replicas: 1

selector:

matchLabels:

app: mysql

template:

metadata:

labels:

app: mysql

spec:

containers:

- name: mysql

image: mysql: 8

ports:

- containerPort: 3306

envFrom:

- configMapRef:

name: mysql-config

env:

- name: MYSQL\_ROOT\_PASSWORD <---------- based on sercret

valueFrom:

secretKeyRef:

name: mysql-secret

key: MYSQL\_ROOT\_PASSWORD

- name: MYSQL\_DATABASE <---------- based on Configmap

valueFrom:

configMapKeyRef:

name: myconfigmap

key: MYSQL\_DATABASE

**Resource Quotas & Limits:-**

When you are going to use request quota and Limits in any deployment.yml then mention it in a specification of pods. Below--

|  |  |  |
| --- | --- | --- |
| **Feature** | **Scope** | **Purpose** |
| |  | | --- | | Resource Requests |  |  | | --- | |  | | |  | | --- | | Pod/Container |  |  | | --- | |  | | |  | | --- | | Scheduling decision (minimum resource) |  |  | | --- | |  | |
| |  | | --- | | Resource Limits |  |  | | --- | |  | | |  | | --- | | Pod/Container |  |  | | --- | |  | | |  | | --- | | Enforce upper resource use limit |  |  | | --- | |  | |
| LimitRange | Namespace | |  | | --- | | Set default/min/max values for resources |  |  | | --- | |  | |
| ResourceQuota | Namespace | Restrict total resource usage in the namespace |

**Taints and Tolerations:-**

A **taint** is applied to a **node** to repel pods **unless they tolerate** that taint.

Think of it as:

Don’t schedule any pod on this node… unless the pod explicitly says it's okay.”

cmd= kubectl taint nodes <node-name> key=value:NoSchedule

Cmd= kubectl taint nodes <node-name> key=value:NoSchedule- < -- removing taints

Example:

Kubectl taint node node\_name prod=true: NoSchedule

**Tolerations:-**

Is use while any node is taints still you to run the **Pod** on that particular Node.

**tolerations**:

- **key**: "key1"

**operator**: "Equal"

**value**: "true"

**effect**: "NoSchedule"

**Auto Scaling in k8s:**

**HPA:-**

Below command for installing metrics for monitoring:-

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

kubectl -n kube-system edit deployment metrics-server

find containers :

ports:

- /metrics-server

- --kubelet-insecure-tls

- --kubelet-preferred-address-types=InternalIP,Hostname,ExternalIP

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

kubectl get pods -n kube-system

kubectl top nodes

***used to******manually scale:***

*Syntax:* kubectl scale <resource\_type>/<resource\_name> --replicas=<number>

*example*: kubectl scale deployment backend-deployment --replicas=3

----------add below frontend-deployment.yaml with Resources & Limits

resources:

requests:

cpu: "100m" # Minimum CPU guaranteed

memory: "128Mi" # Minimum memory guaranteed

limits:

cpu: "300m" # Max CPU container can use

memory: "256Mi" # Max memory container can use

**HPA for Frontend-deployment.**

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: frontend-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: frontend-deployment

minReplicas: 1

maxReplicas: 4

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 50

**HPA for Backend-deployment:-**

apiVersion: autoscaling/v2

kind: HorizontalPodAutoscaler

metadata:

name: backend-hpa

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: backend-deployment

minReplicas: 1

maxReplicas: 4

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 60

------Add below at end of the Backendend-deployment.yaml with Resources & Limits for HPA.

resources:

requests:

cpu: "200m"

memory: "256Mi"

limits:

cpu: "500m"

memory: "512Mi"

|  |
| --- |
| Apply updated deployments |

|  |  |
| --- | --- |
| * kubectl apply -f frontend-deployment.yml **and** backend-deployment.yml | |
| Apply HPAs |

|  |
| --- |
| * kubectl apply -f frontend-hpa.yaml **and** backend-hpa.yaml |

* kubectl get hpa

**also you can autoscale with a single command like below.**

kubectl autoscale deployment frontend-deployment \

--cpu-percent=50 \

--min=1 \

--max=4

**Generating load manually for testing:**

**For backend:-**

kubectl run -it --rm load-generator \

--image=busybox \

--restart=Never \

-- /bin/sh -c "while true; do wget -q -O- http://springboot-service.default.svc.cluster.local:8080; done"

**VPA:**

git clone https://github.com/kubernetes/autoscaler.git

cd autoscaler/vertical-pod-autoscaler

./hack/vpa-up.sh

vim vpa.yml

apiVersion: autoscaling.k8s.io/v1

kind: VerticalPodAutoscaler

metadata:

  name: my-vpa

spec:

  targetRef:

    apiVersion: "apps/v1"

    kind: Deployment

    name: backend-deployment

  updatePolicy:

    updateMode: "Auto"

kubectl run -i --tty load-gernerator --image=bosybox -n apache/bin/sh

# while true; do wget -q -O- http://<SERVICE\_NAME>.<NAMESPACE>.svc.cluster.local; done

Example:

while true; do wget -q -O- http://apache-service.apache.svc.cluster.local; done

*Open Duplicate session then:*

watch kubectl get vpa

kubectl top pods

//

git checkout origin/vpa-release-1.0

REGISTRY=registry.k8s.io/autoscaling TAG=1.0.0 ./hack/vpa-process-yamls.sh apply

//

**Helm in Kubernetes?**

**$** curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

**$** chmod 700 get\_helm.sh

**$** ./get\_helm.sh 🡨 apply this command to run shell script

Cmd: helm version

Cmd: helm create name\_helm [ex. Apache-helm] <---- creating helm chart.

Cmd: ls

Cmd: cd apache-helm

Cmd: ls

Cmd: sudo apt install tree

Cmd: tree

Cmd:

Cmd:

Once you have done with the updated values and changes in Values.yml

Cmd:

Cmd: helm package apache-helm/

Cmd: helm install