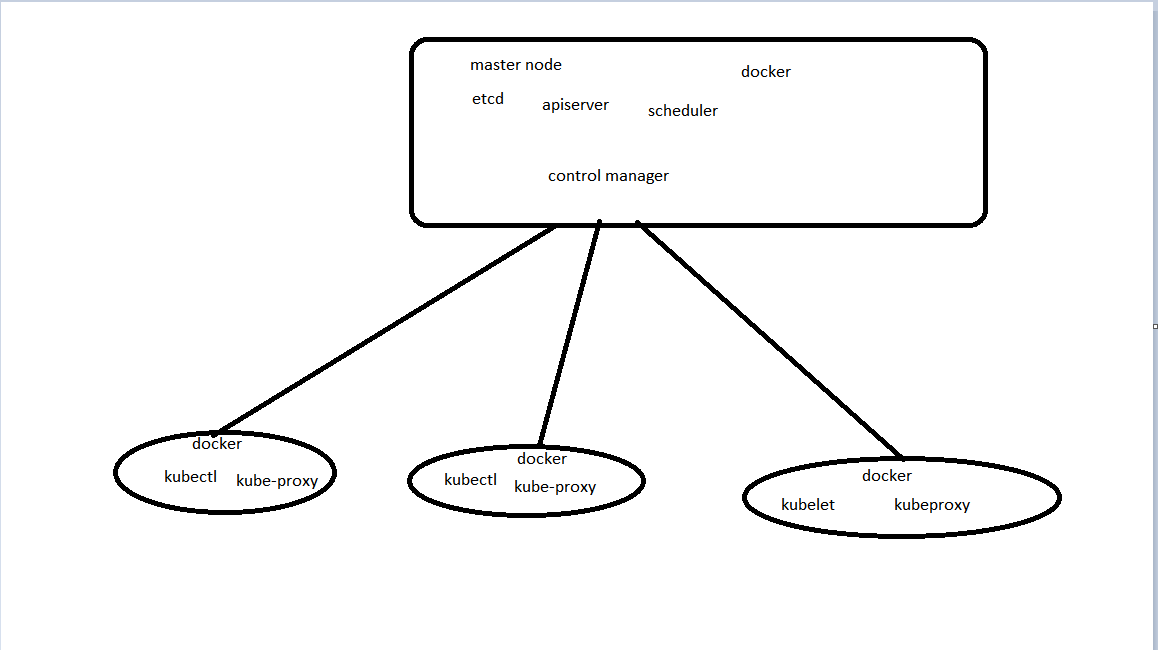
Introduction to kubernetes:



Etcd:database server for kubernetis used to manage cluster related details.used to call api services.

Api server:point of contact from where we can communicate with cluster different part of the cluster api server is exposed outside of the kuber server. to perform something on the kuber.

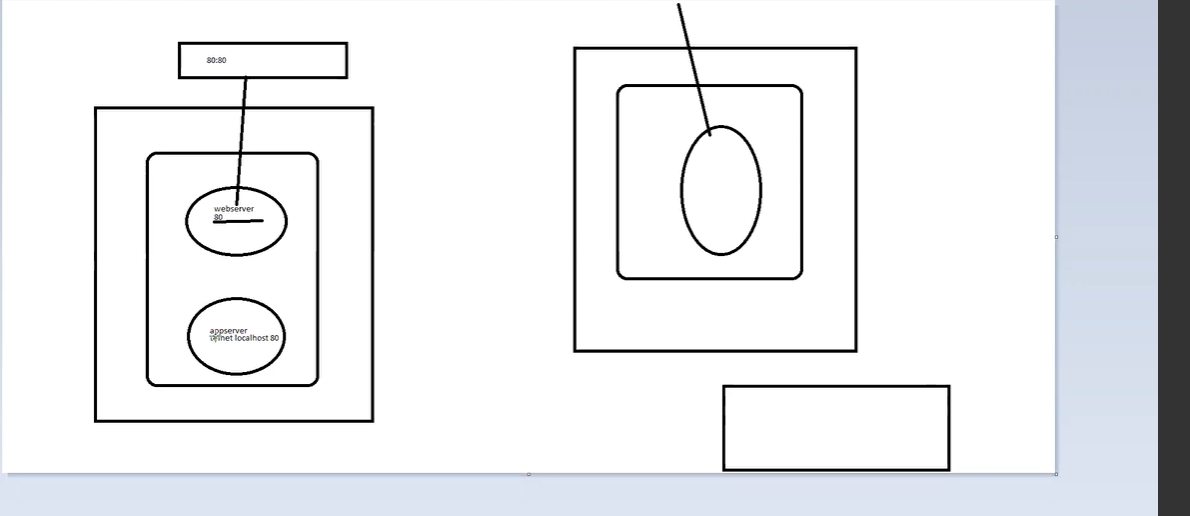
Scheduler: scheduling process and all the resource management dynamic resource management

Control manager:manager of all the controllers like pod controller,endpoint controller,replication controller.

Slave node:

Kubelet

Kubeproxy:



Proxy pass : this is used to communicate between webserver and appserver we will configure the webserver as this will read the data and get the data to appserver.

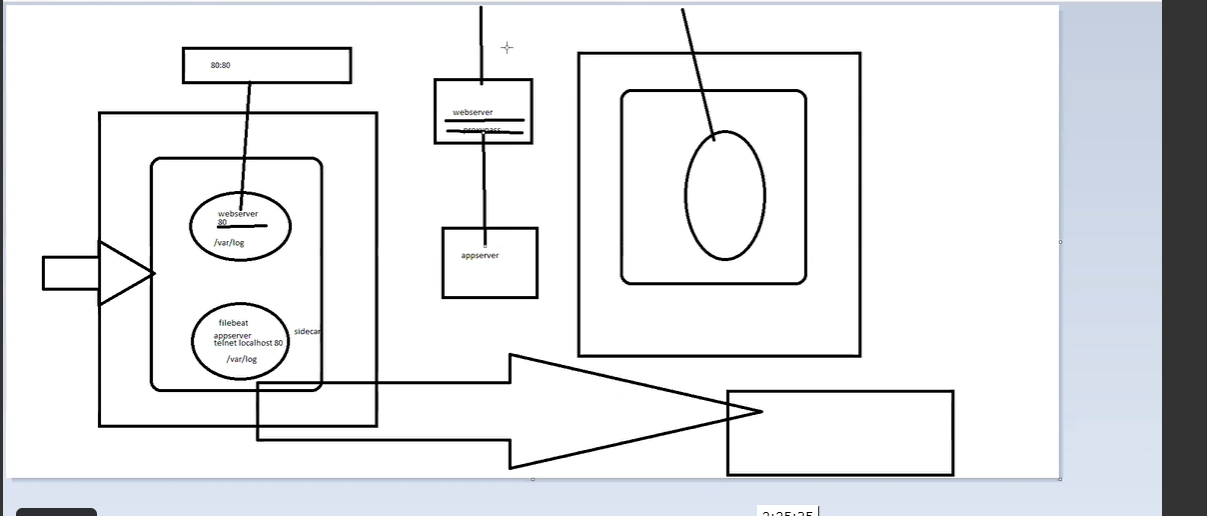
Note: Both the containers will talk to each other.

Container-------🡪pod

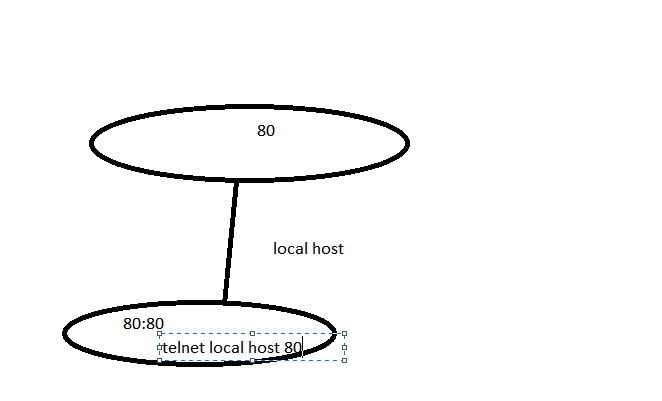
2nd container is called as side cart

Proxy pass:routing traffic from webserver to app server

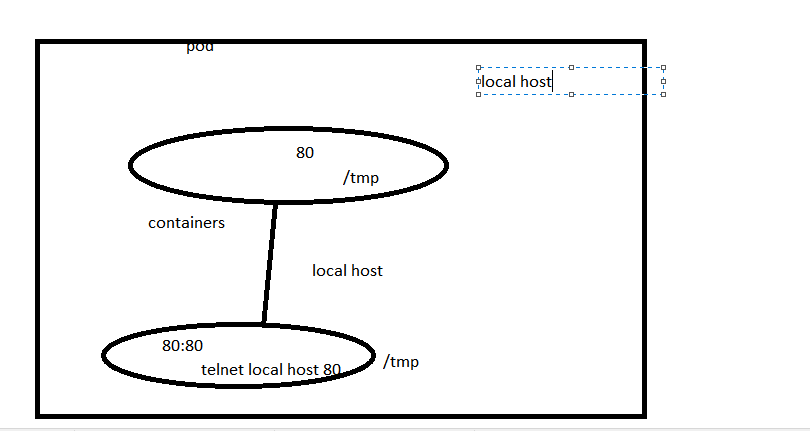
Webserver and app server is in a single pod.

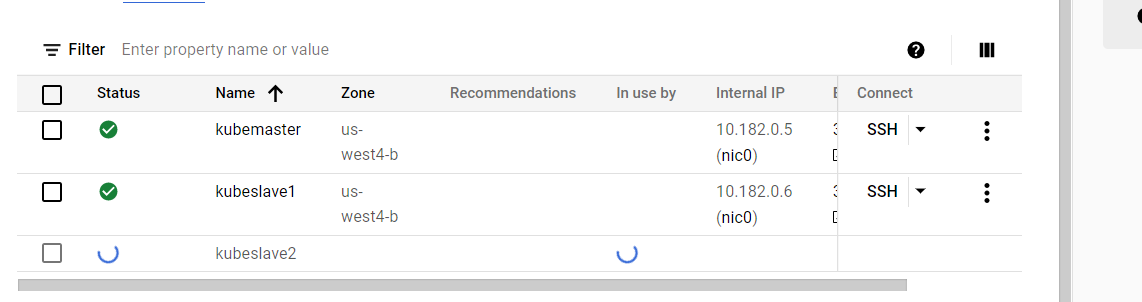


Pods has containers.



Both the containers will talk to each other via local host.





Configuring master:

root@kubemaster-1:~# history

1 sudo apt-get update

2 sudo apt-get install ca-certificates curl gnupg lsb-release

3 sudo apt-get install ca-certificates curl gnupg lsb-release

4 sudo mkdir -p /etc/apt/keyrings

5 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

6 echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

7 sudo apt-get update

8 sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

9 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

11 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

12 apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

13 apt-get install kubeadm=1.20.0-00 kubelet=1.20.0-00 kubectl -y

Docker will create its own network but it is not happening in kubernetis.

Subnet to create private network of your pods and containers.

root@kubemaster-1:~# kubeadm init --pod-network-cidr=10.244.0.0/16

I0727 15:17:30.051170 8861 version.go:251] remote version is much newer: v1.24.3; falling back to: stable-1.20

[init] Using Kubernetes version: v1.20.15

[preflight] Running pre-flight checks

[WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/

[WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.17. Latest validated version: 19.03

[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'

[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 [kubemaster-1 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.182.0.9]

[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 [kubemaster-1 localhost] and IPs [10.182.0.9 127.0.0.1 ::1]

[certs] Generating "etcd/peer" certificate and key

[certs] etcd/peer serving cert is signed for DNS names [kubemaster-1 localhost] and IPs [10.182.0.9 127.0.0.1 ::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

[kubeconfig] Writing "kubelet.conf" kubeconfig file

[kubeconfig] Writing "controller-manager.conf" kubeconfig file

[kubeconfig] Writing "scheduler.conf" kubeconfig file

[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"

[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"

[kubelet-start] Starting the kubelet

[control-plane] Using manifest folder "/etc/kubernetes/manifests"

[control-plane] Creating static Pod manifest for "kube-apiserver"

[control-plane] Creating static Pod manifest for "kube-controller-manager"

[control-plane] Creating static Pod manifest for "kube-scheduler"

[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"

[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s

[apiclient] All control plane components are healthy after 10.504313 seconds

[upload-config] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

[kubelet] Creating a ConfigMap "kubelet-config-1.20" in namespace kube-system with the configuration for the kubelets in the cluster

[upload-certs] Skipping phase. Please see --upload-certs

[mark-control-plane] Marking the node kubemaster-1 as control-plane by adding the labels "node-role.kubernetes.io/master=''" and "node-role.kubernetes.io/control-plane='' (deprecated)"

[mark-control-plane] Marking the node kubemaster-1 as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]

[bootstrap-token] Using token: rjlx9a.g4ob7cjib4krw5fy

[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles

[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to get nodes

[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials

[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster

[bootstrap-token] Creating the "cluster-info" ConfigMap in the "kube-public" namespace

[kubelet-finalize] Updating "/etc/kubernetes/kubelet.conf" to point to a rotatable kubelet client certificate and key

[addons] Applied essential addon: CoreDNS

[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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(admin.conf)-🡪 this will contain the token which will allow you to communicate with the cluster

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

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.182.0.9:6443 --token rjlx9a.g4ob7cjib4krw5fy \

--discovery-token-ca-cert-hash sha256:e3579dddc1499cd5a476d2b80c90f8743cd29e94711900df276a9ad6d7a263fc

root@kubemaster-1:~#

root@kubemaster-1:~#

flannel-🡪this is a overlay network.

Create some name space while initializing the project

root@kubemaster-1:~# kubectl get ns

NAME STATUS AGE

default Active 7m26s

kube-node-lease Active 7m28s

kube-public Active 7m28s

kube-system Active 7m28s----🡪 all the kubernetis related services.

root@kubemaster-1:~#

root@kubemaster-1:~# kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

coredns-74ff55c5b-gclmj 0/1 Pending 0 9m16s

coredns-74ff55c5b-mrkq9 0/1 Pending 0 9m16s

etcd-kubemaster-1 1/1 Running 0 9m29s

kube-apiserver-kubemaster-1 1/1 Running 0 9m29s

kube-controller-manager-kubemaster-1 1/1 Running 0 9m29s

kube-proxy-9r9dt 1/1 Running 0 9m16s

kube-scheduler-kubemaster-1 1/1 Running 0 9m29s

root@kubemaster-1:~# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

namespace/kube-flannel created

clusterrole.rbac.authorization.k8s.io/flannel created

clusterrolebinding.rbac.authorization.k8s.io/flannel created

serviceaccount/flannel created

configmap/kube-flannel-cfg created

daemonset.apps/kube-flannel-ds created

root@kubemaster-1:~# kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

coredns-74ff55c5b-gclmj 0/1 Pending 0 12m

coredns-74ff55c5b-mrkq9 0/1 Pending 0 12m

etcd-kubemaster-1 1/1 Running 0 12m

kube-apiserver-kubemaster-1 1/1 Running 0 12m

kube-controller-manager-kubemaster-1 1/1 Running 0 12m

kube-proxy-9r9dt 1/1 Running 0 12m

kube-scheduler-kubemaster-1 1/1 Running 0 12m

root@kubemaster-1:~# kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

coredns-74ff55c5b-gclmj 0/1 ContainerCreating 0 12m

coredns-74ff55c5b-mrkq9 0/1 ContainerCreating 0 12m

etcd-kubemaster-1 1/1 Running 0 12m

kube-apiserver-kubemaster-1 1/1 Running 0 12m

kube-controller-manager-kubemaster-1 1/1 Running 0 12m

kube-proxy-9r9dt 1/1 Running 0 12m

kube-scheduler-kubemaster-1 1/1 Running 0 12m

root@kubemaster-1:~# kubectl get pods -n kube-system

NAME READY STATUS RESTARTS AGE

coredns-74ff55c5b-gclmj 1/1 Running 0 12m

coredns-74ff55c5b-mrkq9 1/1 Running 0 12m

etcd-kubemaster-1 1/1 Running 0 12m

kube-apiserver-kubemaster-1 1/1 Running 0 12m

kube-controller-manager-kubemaster-1 1/1 Running 0 12m

kube-proxy-9r9dt 1/1 Running 0 12m

kube-scheduler-kubemaster-1 1/1 Running 0 12m

root@kubemaster-1:~#

root@kubemaster-1:~# kubectl cluster-info

Kubernetes control plane is running at https://10.182.0.9:6443

KubeDNS is running at https://10.182.0.9:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

root@kubemaster-1:~#

my api server is exposed in 6443

root@kubemaster-1:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

kubemaster-1 Ready control-plane,master 14m v1.20.0

root@kubemaster-1:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

kubemaster-1 Ready control-plane,master 26m v1.20.0

kubeslave1 Ready <none>

root@kubemaster-1:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

kubemaster-1 Ready control-plane,master 45m v1.20.0

kubeslave-2 Ready <none> 56s v1.20.0

kubeslave1 Ready <none> 19m v1.20.0

swerm -🡪service kubernetis-🡪deployment.

root@kubemaster-1:~# kubectl get nodes

NAME STATUS ROLES AGE VERSION

kubemaster-1 Ready control-plane,master 84m v1.20.0

kubeslave-2 Ready <none> 40m v1.20.0

kubeslave1 Ready <none> 59m v1.20.0

root@kubemaster-1:~# kubectl run --image=nginx nginx

pod/nginx created

root@kubemaster-1:~# kubectl get pods -n default

NAME READY STATUS RESTARTS AGE

nginx 1/1 Running 0 13s

root@kubemaster-1:~#

root@kubemaster-1:~# kubectl create namespace test

namespace/test created

root@kubemaster-1:~# kubectl get ns

NAME STATUS AGE

default Active 88m

kube-flannel Active 75m

kube-node-lease Active 88m

kube-public Active 88m

kube-system Active 88m

test Active 8s

root@kubemaster-1:~#’

name space is nothing but a specific name given to the application or a team.

root@kubemaster:~# vim namespace.yaml

root@kubemaster:~# cat namespace.yaml

apiVersion: v1

kind: Namespace

metadata:

name : demo

labels:

apps: web-based

annotations:

type: demo

root@kubemaster:~# kubectl create -f namespace.yaml

namespace/demo created

root@kubemaster:~#

root@kubemaster:~# kubectl delete -f namespace.yaml

root@kubemaster:~# kubectl get ns

NAME STATUS AGE

default Active 25m

demo Active 55s

kube-flannel Active 12m

kube-node-lease Active 25m

kube-public Active 25m

kube-system Active 25m

test Active 6m52s

root@kubemaster:~# vim deployment.yaml

root@kubemaster:~# kubectl create -f deployment.yaml

deployment.apps/httpd created

root@kubemaster:~# cat deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: httpd

labels:

app: httpd

namespace: demo

annotations:

monitoring: "true"

spec:

replicas: 4

selector:

matchLabels:

app: httpd

template:

metadata:

labels:

app: httpd

spec:

containers:

- image: httpd

name: httpd

ports:

- containerPort: 80

root@kubemaster:~# kubectl get deployment -n demo

NAME READY UP-TO-DATE AVAILABLE AGE

httpd 4/4 4 4 13m

root@kubemaster:~#

root@kubemaster:~# kubectl get pods -n demo -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

httpd-57fc687dcc-r5dh9 1/1 Running 0 14m 10.244.1.4 kubeslave1 <none> <none>

httpd-57fc687dcc-vg66l 1/1 Running 0 14m 10.244.1.3 kubeslave1 <none> <none>

httpd-57fc687dcc-w77b9 1/1 Running 0 14m 10.244.2.2 kubeslave2 <none> <none>

httpd-57fc687dcc-wg7wv 1/1 Running 0 14m 10.244.2.3 kubeslave2 <none> <none>

in kubernetis we need to create a service then only we can access it.

root@kubemaster:~# kubectl get pods -n demo -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

httpd-57fc687dcc-r5dh9 1/1 Running 0 19m 10.244.1.4 kubeslave1 <none> <none>

httpd-57fc687dcc-vg66l 1/1 Running 0 19m 10.244.1.3 kubeslave1 <none> <none>

httpd-57fc687dcc-w77b9 1/1 Running 0 19m 10.244.2.2 kubeslave2 <none> <none>

httpd-57fc687dcc-wg7wv 1/1 Running 0 19m 10.244.2.3 kubeslave2 <none> <none>

root@kubemaster:~# kubectl delete pods httpd-57fc687dcc-r5dh9 -n demo

pod "httpd-57fc687dcc-r5dh9" deleted

root@kubemaster:~#

root@kubemaster:~# kubectl get pods -n demo -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

httpd-57fc687dcc-2l2zk 1/1 Running 0 15s 10.244.1.5 kubeslave1 <none> <none>

httpd-57fc687dcc-vg66l 1/1 Running 0 20m 10.244.1.3 kubeslave1 <none> <none>

httpd-57fc687dcc-w77b9 1/1 Running 0 20m 10.244.2.2 kubeslave2 <none> <none>

httpd-57fc687dcc-wg7wv 1/1 Running 0 20m 10.244.2.3 kubeslave2 <none> <none>

root@kubemaster:~# kubectl describe deployment httpd -n demo

Name: httpd

Namespace: demo

CreationTimestamp: Thu, 28 Jul 2022 07:34:59 +0000

Labels: app=httpd

Annotations: deployment.kubernetes.io/revision: 1

monitoring: true

Selector: app=httpd

Replicas: 4 desired | 4 updated | 4 total | 4 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=httpd

Containers:

httpd:

Image: httpd

Port: 80/TCP

Host Port: 0/TCP

Environment: <none>

Mounts: <none>

Volumes: <none>

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: httpd-57fc687dcc (4/4 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 21m deployment-controller Scaled up replica set httpd-57fc687dcc to 4

root@kubemaster:~#

Creating service………..creating service………..

root@kubemaster:~# kubectl create service nodeport httpd --tcp 80:80

service/httpd created

root@kubemaster:~# kubectl get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

httpd NodePort 10.110.154.30 <none> 80:31843/TCP 9s

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 68m

root@kubemaster:~# kubectl create service nodeport httpd --tcp 80:80 -n demo

service/httpd created

root@kubemaster:~# kubectl get service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

httpd NodePort 10.110.154.30 <none> 80:31843/TCP 37s

kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 68m

root@kubemaster:~# kubectl get service -n demo

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

httpd NodePort 10.97.174.253 <none> 80:31249/TCP 19s

root@kubemaster:~# kubectl get pods -n demo -o wide

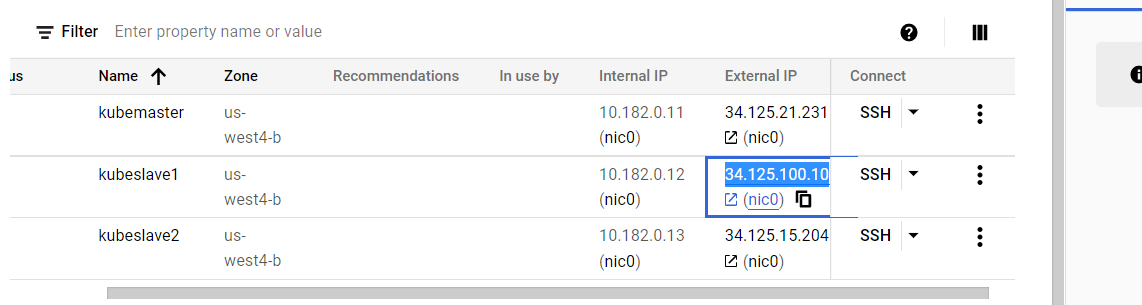
NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

httpd-57fc687dcc-2l2zk 1/1 Running 0 5m36s 10.244.1.5 kubeslave1 <none> <none>

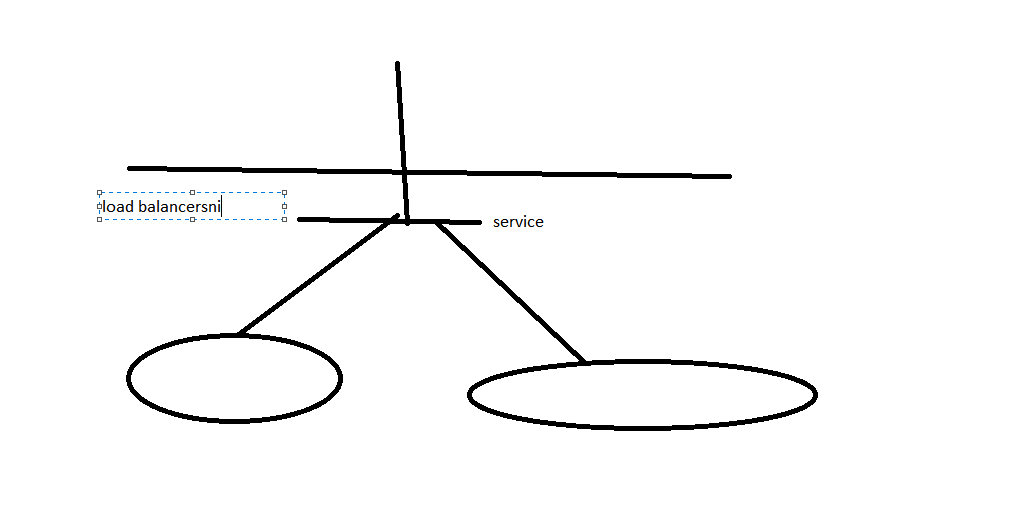
httpd-57fc687dcc-vg66l 1/1 Running 0 25m 10.244.1.3 kubeslave1 <none> <none>

httpd-57fc687dcc-w77b9 1/1 Running 0 25m 10.244.2.2 kubeslave2 <none> <none>

httpd-57fc687dcc-wg7wv 1/1 Running 0 25m 10.244.2.3 kubeslave2 <none> <none>



Note:nodeport is the way to expose application directly to the host mechine.



Restrictions:

Creating paristant volume:

root@kubemaster:~# mkdir /data

root@kubemaster:~# vi pv.yaml

root@kubemaster:~# vi pv.yaml

root@kubemaster:~# vi pv.yml

root@kubemaster:~# cat pv.yml

apiVersion: v1

kind: PersistentVolume

metadata:

name: myvol

namespace: demo

spec:

storageClassName: manual

capacity:

storage: 2Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/data"

root@kubemaster:~#

root@kubemaster:~# kubectl create -f pv.yml

persistentvolume/myvol created

root@kubemaster:~# kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

myvol 2Gi RWO Retain Available manual 11s

root@kubemaster:~#

if we are having large data we cant use them all at once we need to create a persistent volume ex:if you are having 100GB of data in that we need to use only 10GB for my application them we need to create a persistent volume.

persistentvolume/myvol created

root@kubemaster:~# kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

myvol 2Gi RWO Retain Available manual 11s

root@kubemaster:~# vi pvc.yml

root@kubemaster:~# kubectl create -f pvc.yml

persistentvolumeclaim/my-volpvc created

root@kubemaster:~# kubectl get pv

NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE

myvol 2Gi RWO Retain Bound demo/my-volpvc manual 8m8s

root@kubemaster:~# kubectl get pvc -n demo

NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE

my-volpvc Bound myvol 2Gi RWO manual 41s

we can see that the capacity of 2GB which was assigned to name space demo only this two gb can be used by the demo name space.

Volume is the part of cluster where as pvc is the part of name space.no other can use this volume other then us

If we need to assign this pv to multiple pvc multiple applications all the application should be deployed to a single name space.

|  |
| --- |
| apiVersion: apps/v1 |
|  | kind: Deployment |
|  | metadata: |
|  | name: nginx |
|  | labels: |
|  | app: nginx |
|  | namespace: demo |
|  | annotations: |
|  | monitoring: "true" |
|  | spec: |
|  | replicas: 4 |
|  | selector: |
|  | matchLabels: |
|  | app: nginx |
|  | template: |
|  | metadata: |
|  | labels: |
|  | app: nginx |
|  | spec: |
|  | containers: |
|  | - image: nginx:latest |
|  | name: nginx |
|  | ports: |
|  | - containerPort: 80 |
|  | resources: |
|  | limits: |
|  | memory: "2Gi" |
|  | cpu: "1000m" |
|  | requests: |
|  | memory: "1Gi" |
|  | cpu: "500m" |
|  | volumeMounts: |
|  | - name: pvc1 |
|  | mountPath: /tmp |
|  | volumes: |
|  | - name: pvc1 |
|  | persistentVolumeClaim: |
|  | claimName: my-volpvc |

Above is highlighting the resource limitation.

|  |
| --- |
| resources: |
| limits: |
|  |
| memory: "2Gi" |
| cpu: "1000m" |
| requests: |
| memory: "1Gi" |
| cpu: "500m"  request is the resources needed to start the pod and resources are the max limit to start the pod.  If it is passing the limit the pod get restarts. |

|  |
| --- |
| volumeMounts: |
| - name: pvc1 |
| mountPath: /tmp |
| volumes: |
| - name: pvc1 |
| persistentVolumeClaim: |
| claimName: my-volpvc |

root@kubemaster:~# kubectl create -f deployment.yml

deployment.apps/nginx created

root@kubemaster:~# kubectl get deployment -n demo -o wide

NAME READY UP-TO-DATE AVAILABLE AGE CONTAINERS IMAGES SELECTOR

httpd 4/4 4 4 4h13m httpd httpd app=httpd

nginx 4/4 4 4 38s nginx nginx:latest app=nginx

root@kubemaster:~#

root@kubemaster:~# kubectl describe deployment nginx -n demo

Name: nginx

Namespace: demo

CreationTimestamp: Thu, 28 Jul 2022 11:47:52 +0000

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 1

monitoring: true

Selector: app=nginx

Replicas: 4 desired | 4 updated | 4 total | 4 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:latest

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 1

memory: 2Gi

Requests:

cpu: 500m

memory: 1Gi

Environment: <none>

Mounts:

/tmp from pvc1 (rw)

Volumes:

pvc1:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: my-volpvc

ReadOnly: false

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: nginx-55c9c7dc96 (4/4 replicas created)

Events:

Type Reason Age From Message

Normal ScalingReplicaSet 94s deployment-controller Scaled up replica set nginx-55c9c7dc96 to 4

root@kubemaster:~#

root@kubemaster:~# kubectl apply -f deployment.yml

Warning: resource deployments/nginx is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.

deployment.apps/nginx configured

root@kubemaster:~# kubectl describe deployment nginx -n demo

Name: nginx

Namespace: demo

CreationTimestamp: Thu, 28 Jul 2022 11:47:52 +0000

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 2

monitoring: true

Selector: app=nginx

Replicas: 4 desired | 2 updated | 5 total | 3 available | 2 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:latest

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 1

memory: 2Gi

Requests:

cpu: 500m

memory: 1Gi

Environment: <none>

Mounts:

/tmp from pvc1 (rw)

Volumes:

pvc1:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: my-volpvc1

ReadOnly: false

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True ReplicaSetUpdated

OldReplicaSets: nginx-55c9c7dc96 (3/3 replicas created)

NewReplicaSet: nginx-685f8bcb9c (2/2 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 12m deployment-controller Scaled up replica set nginx-55c9c7dc96 to 4

Normal ScalingReplicaSet 14s deployment-controller Scaled up replica set nginx-685f8bcb9c to 1

Normal ScalingReplicaSet 14s deployment-controller Scaled down replica set nginx-55c9c7dc96 to 3

Normal ScalingReplicaSet 14s deployment-controller Scaled up replica set nginx-685f8bcb9c to 2

root@kubemaster:~# cat deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

labels:

app: nginx

namespace: demo

annotations:

monitoring: "true"

spec:

replicas: 4

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- image: nginx:latest

name: nginx

ports:

- containerPort: 80

resources:

limits:

memory: "2Gi"

cpu: "1000m"

requests:

memory: "1Gi"

cpu: "500m"

volumeMounts:

- name: pvc1

mountPath: /tmp

volumes:

- name: pvc1

persistentVolumeClaim:

claimName: my-volpvc1

root@kubemaster:~#

root@kubemaster:~# kubectl describe nginx -n demo

error: the server doesn't have a resource type "nginx"

root@kubemaster:~# kubectl describe deployment nginx -n demo

Name: nginx

Namespace: demo

CreationTimestamp: Thu, 28 Jul 2022 11:47:52 +0000

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 1

monitoring: true

Selector: app=nginx

Replicas: 4 desired | 4 updated | 4 total | 4 available | 0 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:latest

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 1

memory: 2Gi

Requests:

cpu: 500m

memory: 1Gi

Environment: <none>

Mounts:

/tmp from pvc1 (rw)

Volumes:

pvc1:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: my-volpvc

ReadOnly: false

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True NewReplicaSetAvailable

OldReplicaSets: <none>

NewReplicaSet: nginx-55c9c7dc96 (4/4 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 94s deployment-controller Scaled up replica set nginx-55c9c7dc96 to 4

root@kubemaster:~# kubectl create -f deployment.yml

Error from server (AlreadyExists): error when creating "deployment.yml": deployments.apps "nginx" already exists

root@kubemaster:~# vi deployment.yml

root@kubemaster:~# kubectl apply -f deployment.yml

Warning: resource deployments/nginx is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.

deployment.apps/nginx configured

root@kubemaster:~# kubectl describe deployment nginx -n demo

Name: nginx

Namespace: demo

CreationTimestamp: Thu, 28 Jul 2022 11:47:52 +0000

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 2

monitoring: true

Selector: app=nginx

Replicas: 4 desired | 2 updated | 5 total | 3 available | 2 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:latest

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 1

memory: 2Gi

Requests:

cpu: 500m

memory: 1Gi

Environment: <none>

Mounts:

/tmp from pvc1 (rw)

Volumes:

pvc1:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: my-volpvc1

ReadOnly: false

Conditions:

Type Status Reason

---- ------ ------

Available True MinimumReplicasAvailable

Progressing True ReplicaSetUpdated

OldReplicaSets: nginx-55c9c7dc96 (3/3 replicas created)

NewReplicaSet: nginx-685f8bcb9c (2/2 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 12m deployment-controller Scaled up replica set nginx-55c9c7dc96 to 4

Normal ScalingReplicaSet 14s deployment-controller Scaled up replica set nginx-685f8bcb9c to 1

Normal ScalingReplicaSet 14s deployment-controller Scaled down replica set nginx-55c9c7dc96 to 3

Normal ScalingReplicaSet 14s deployment-controller Scaled up replica set nginx-685f8bcb9c to 2

root@kubemaster:~# cat deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

labels:

app: nginx

namespace: demo

annotations:

monitoring: "true"

spec:

replicas: 4

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- image: nginx:latest

name: nginx

ports:

- containerPort: 80

resources:

limits:

memory: "2Gi"

cpu: "1000m"

requests:

memory: "1Gi"

cpu: "500m"

volumeMounts:

- name: pvc1

mountPath: /tmp

volumes:

- name: pvc1

persistentVolumeClaim:

claimName: my-volpvc1

root@kubemaster:~# kubectl delete -f deployment.yml

deployment.apps "nginx" deleted

root@kubemaster:~# kubectl create -f deployment.yml

deployment.apps/nginx created

root@kubemaster:~# kubectl describe deployment nginx -n demo

Name: nginx

Namespace: demo

CreationTimestamp: Thu, 28 Jul 2022 12:03:51 +0000

Labels: app=nginx

Annotations: deployment.kubernetes.io/revision: 1

monitoring: true

Selector: app=nginx

Replicas: 4 desired | 4 updated | 4 total | 0 available | 4 unavailable

StrategyType: RollingUpdate

MinReadySeconds: 0

RollingUpdateStrategy: 25% max unavailable, 25% max surge

Pod Template:

Labels: app=nginx

Containers:

nginx:

Image: nginx:latest

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 1

memory: 2Gi

Requests:

cpu: 500m

memory: 1Gi

Environment: <none>

Mounts:

/tmp from pvc1 (rw)

Volumes:

pvc1:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: my-volpvc1

ReadOnly: false

Conditions:

Type Status Reason

---- ------ ------

Available False MinimumReplicasUnavailable

Progressing True ReplicaSetUpdated

OldReplicaSets: <none>

NewReplicaSet: nginx-685f8bcb9c (4/4 replicas created)

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal ScalingReplicaSet 2m4s deployment-controller Scaled up replica set nginx-685f8bcb9c to 4

root@kubemaster:~#

root@kubemaster:~# kubectl get pods -n demo

NAME READY STATUS RESTARTS AGE

httpd-57fc687dcc-2l2zk 1/1 Running 1 4h11m

httpd-57fc687dcc-vg66l 1/1 Running 1 4h31m

httpd-57fc687dcc-w77b9 1/1 Running 1 4h31m

httpd-57fc687dcc-wg7wv 1/1 Running 1 4h31m

nginx-685f8bcb9c-kgmw9 0/1 Pending 0 2m54s

nginx-685f8bcb9c-nqmzn 0/1 Pending 0 2m54s

nginx-685f8bcb9c-rrlns 0/1 Pending 0 2m54s

nginx-685f8bcb9c-x79m4 0/1 Pending 0 2m54s

root@kubemaster:~# kubectl describe pods nginx-685f8bcb9c-rrlns -n demo

Name: nginx-685f8bcb9c-rrlns

Namespace: demo

Priority: 0

Node: <none>

Labels: app=nginx

pod-template-hash=685f8bcb9c

Annotations: <none>

Status: Pending

IP:

IPs: <none>

Controlled By: ReplicaSet/nginx-685f8bcb9c

Containers:

nginx:

Image: nginx:latest

Port: 80/TCP

Host Port: 0/TCP

Limits:

cpu: 1

memory: 2Gi

Requests:

cpu: 500m

memory: 1Gi

Environment: <none>

Mounts:

/tmp from pvc1 (rw)

/var/run/secrets/kubernetes.io/serviceaccount from default-token-swtps (ro)

Conditions:

Type Status

PodScheduled False

Volumes:

pvc1:

Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)

ClaimName: my-volpvc1

ReadOnly: false

default-token-swtps:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-swtps

Optional: false

QoS Class: Burstable

Node-Selectors: <none>

Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s

node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Warning FailedScheduling 30m default-scheduler 0/3 nodes are available: 3 persistentvolumeclaim "my-volpvc1" not found.

Warning FailedScheduling 27m (x3 over 30m) default-scheduler 0/3 nodes are available: 3 persistentvolumeclaim "my-volpvc1" not found.

We are not having this volume defined above my-volpvc1 so we are seeing the above errors.

root@kubemaster:~# cat deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx

labels:

app: nginx

namespace: demo

annotations:

monitoring: "true"

spec:

replicas: 4

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- image: nginx:latest

name: nginx

ports:

- containerPort: 80

resources:

limits:

memory: "2Gi"

cpu: "1000m"

requests:

memory: "1Gi"

cpu: "500m"

volumeMounts:

- name: pvc1

mountPath: /tmp

volumes:

- name: pvc1

persistentVolumeClaim:

claimName: my-volpvc

root@kubemaster:~# kubectl apply -f deployment.yml

Warning: resource deployments/nginx is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.

deployment.apps/nginx configured

root@kubemaster:~# kubectl get pods -n demo

NAME READY STATUS RESTARTS AGE

httpd-57fc687dcc-2l2zk 1/1 Running 2 5h28m

httpd-57fc687dcc-vg66l 1/1 Running 2 5h48m

httpd-57fc687dcc-w77b9 1/1 Running 2 5h48m

httpd-57fc687dcc-wg7wv 1/1 Running 2 5h48m

nginx-55c9c7dc96-gxnvw 1/1 Running 0 33s

nginx-55c9c7dc96-mfv9j 1/1 Running 0 34s

nginx-55c9c7dc96-smvj5 1/1 Running 0 36s

nginx-55c9c7dc96-vqm7s 1/1 Running 0 36s

note:what ever er are defaining in deployment config that should be there in the env then only pods will come up.

We can deploy it using command line tool or template.

root@instance-1:~# kubectl get SVC -n demo

This will give the details of the ports where it got exposed

root@instance-1:~# kubectl get SVC -n demo -o wide

This is used to check the selector details

root@instance-1:~# kubectl describe pods httpd-57fc687dcc-4bp7j -n demo

Name: httpd-57fc687dcc-4bp7j

Namespace: demo

Priority: 0

Node: slave1/10.182.0.15

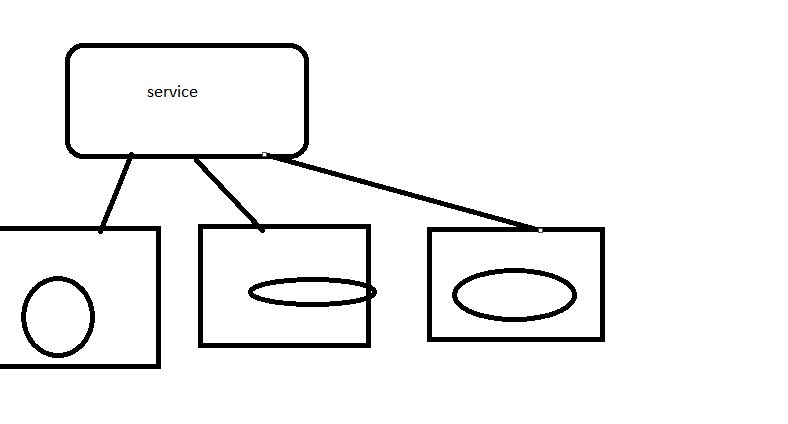
Start Time: Fri, 29 Jul 2022 04:52:59 +0000

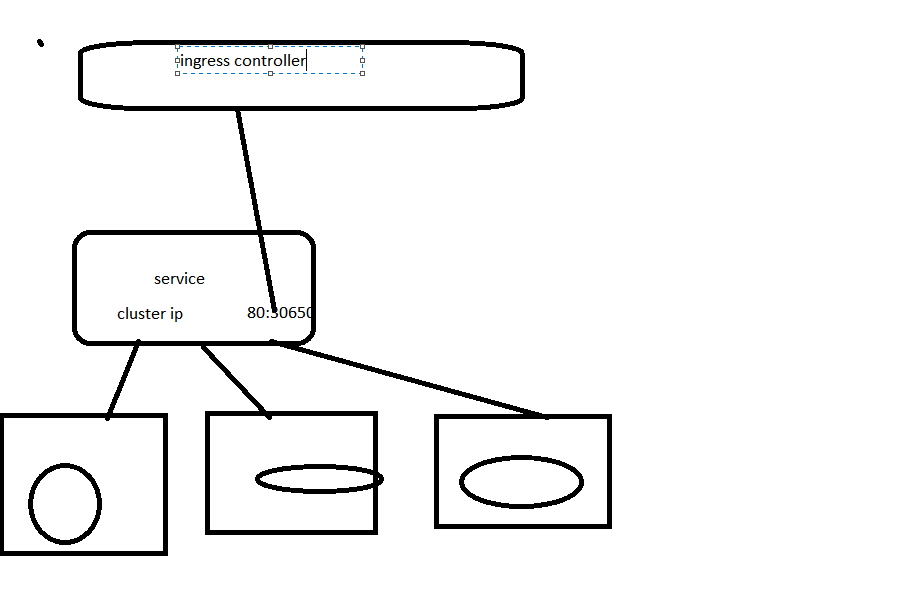
Labels: app=httpd

pod-template-hash=57fc687dcc

app label --🡪 service will check the all the pods with the label app label and send the request to certain pod.

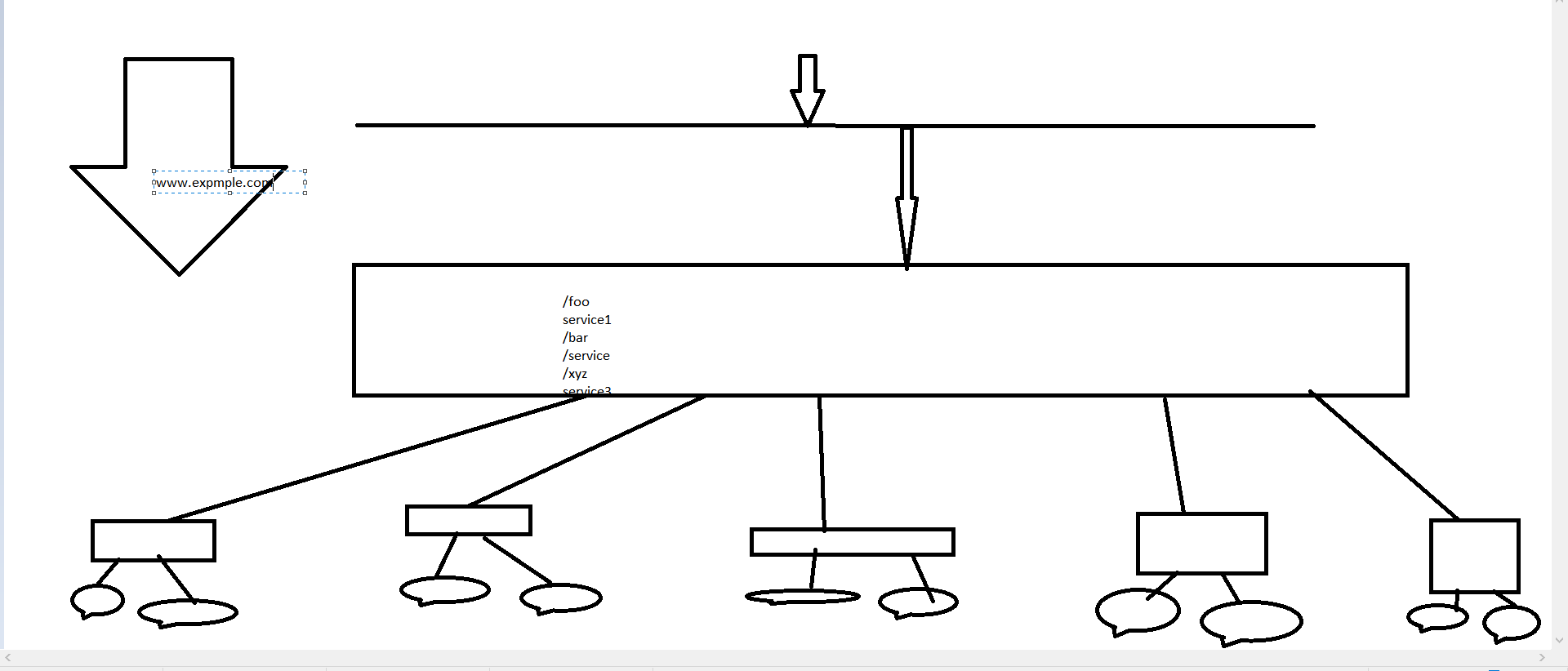
|  |
| --- |
| kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommended/kubernetes-dashboard.yaml |
|  | 129 kubectl get pods -n kube-system |
|  | 130 vim admin.yml |
|  | 131 kubectl create -f admin.yml |
|  | 132 kubectl edit svc kubernetes-dashboard -n kube-system |
|  | 133 kubectl get svc -n kube-system |
|  | 134 kubectl get pods -n kube-system -o wide |
|  | 135 kubectl logs kubernetes-dashboard-65ff5d4cc8-tshkj -n kube-system |
|  | 136 kubectl logs kubernetes-dashboard-65ff5d4cc8-tshkj -n kube-system -f |
|  | 137 kubectl get secret -n kube-system |
|  | 138 kubectl -n kube-system describe secret admin-user-token-lmrjd |
|  | 139 kubectl get pods -n demo |
|  | 140 kubectl get deployment -n demo |
|  | 141 kubectl describe deployment httpd -n demo |
|  | 142 kubectl get deployment -n demo |
|  | 143\* |
|  | 144 kubectl get deployment -n demo |
|  | 145 kubectl create -f deployment.yml |
|  | 146 kubectl get deployment -n demo |
|  | 147 kubectl get node |
|  | 148 kubectl get quota -n demo |
|  | 149 kubectl delete quota mem-cpu-demo |
|  | 150 kubectl delete quota mem-cpu-demo -n demo |
|  | 151 kubectl get deployment -n demo |
|  | 152 kubectl delete -f deployment.yml |
|  | 153 kubectl get deployment -n demo |
|  | 154 kubectl create -f deployment.yml |
|  | 155 kubectl get deployment -n demo |
|  | 156 kubectl autoscale --help |
|  | 157 kubectl autoscale deployment nginx --cpu-percent=50 --min=1 --max=10 |
|  | 158 kubectl autoscale deployment nginx --cpu-percent=50 --min=1 --max=10 -n demo |
|  | 159 kubectl get hpa |
|  | 160 kubectl get hpa -n demo |

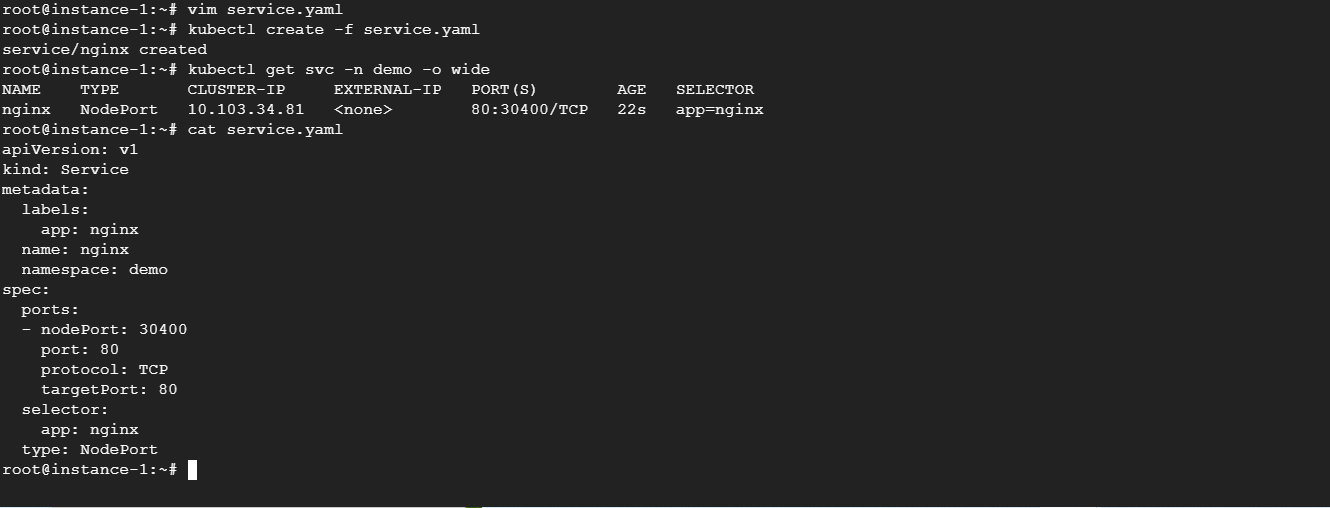




Whenever we are creating the application internally it will create a service and it will round robin and connect the connections to that application.

Service is Labels: app=httpd it will check and it will reflect the traffic to this specific application





Note: we cant link two pods we need to do it via service.

root@instance-1:~# kubectl create -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.34.1/deploy/static/provider/baremetal/deploy.yaml

namespace/ingress-nginx created

serviceaccount/ingress-nginx created

configmap/ingress-nginx-controller created

clusterrole.rbac.authorization.k8s.io/ingress-nginx created

clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created

role.rbac.authorization.k8s.io/ingress-nginx created

rolebinding.rbac.authorization.k8s.io/ingress-nginx created

service/ingress-nginx-controller-admission created

service/ingress-nginx-controller created

deployment.apps/ingress-nginx-controller created

Warning: admissionregistration.k8s.io/v1beta1 ValidatingWebhookConfiguration is deprecated in v1.16+, unavailable in v1.22+; use admissionregistration.k8s.io/v1 ValidatingWebhookConfiguration

validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created

clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created

clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created

job.batch/ingress-nginx-admission-create created

job.batch/ingress-nginx-admission-patch created

role.rbac.authorization.k8s.io/ingress-nginx-admission created

rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created

serviceaccount/ingress-nginx-admission created

root@instance-1:~# kubectl get ns

NAME STATUS AGE

default Active 122m

demo Active 74m

ingress-nginx Active 17s

kube-flannel Active 79m

kube-node-lease Active 122m

kube-public Active 122m

kube-system Active 122m

root@instance-1:~# kubectl get pods -n ingress-nginx

NAME READY STATUS RESTARTS AGE

ingress-nginx-admission-create-7pq5j 0/1 Completed 0 30s

ingress-nginx-admission-patch-4r92w 0/1 Completed 0 30s

ingress-nginx-controller-7bc44b4bb-vtgfs 0/1 ContainerCreating 0 40s

root@instance-1:~# kubectl get pods svc -n ingress-nginx

Error from server (NotFound): pods "svc" not found

root@instance-1:~# kubectl get pods SVC -n ingress-nginx

Error from server (NotFound): pods "SVC" not found

root@instance-1:~# kubectl get SVC -n ingress-nginx

error: the server doesn't have a resource type "SVC"

root@instance-1:~# kubectl get SVC -n ingress-nginx

error: the server doesn't have a resource type "SVC"

root@instance-1:~# kubectl get svc -n ingress-nginx

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

ingress-nginx-controller NodePort 10.99.189.7 <none> 80:32500/TCP,443:32660/TCP 109s

ingress-nginx-controller-admission ClusterIP 10.106.63.96 <none> 443/TCP 109s

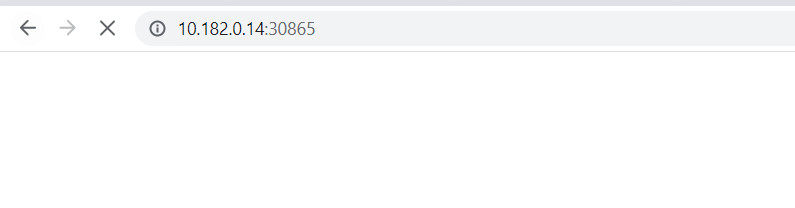
root@instance-1:~# kubectl get svc -n ingress-nginx -o wide

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

ingress-nginx-controller NodePort 10.99.189.7 <none> 80:32500/TCP,443:32660/TCP 2m6s app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx,app.kubernetes.io/name=ingress-nginx

ingress-nginx-controller-admission ClusterIP 10.106.63.96 <none> 443/TCP 2m6s app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx,app.kubernetes.io/name=ingress-nginx

root@instance-1:~#



Setting up the rules in ingress:

root@instance-1:~# vim nginx-igress-rule.yml

root@instance-1:~# vim nginx-igress-rule.yml

root@instance-1:~# kubectl create -f nginx-igress-rule.yml

Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress

ingress.extensions/nginx-ingress created

root@instance-1:~# cat nginx-igress-rule.yml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: nginx-ingress

namespace: demo

annotations:

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

spec:

rules:

- http:

paths:

- backend:

serviceName: httpd

servicePort: 80

path: /

- backend:

serviceName: nginx

servicePort: 80

path: /foo

root@instance-1:~#

below is for nginx:

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

name: nginx-ingress

namespace: demo

annotations:

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

spec:

rules:

- http:

paths:

- backend:

serviceName: httpd

servicePort: 80

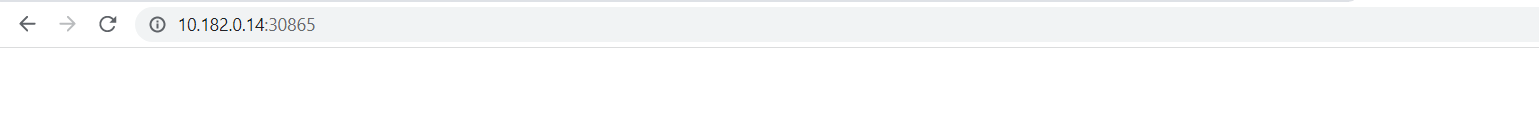
path: /

- backend:

serviceName: nginx

servicePort: 80

path: /foo



configuring slave: root@instance-1:~# kubectl get pods -n demo

NAME READY STATUS RESTARTS AGE

httpd-57fc687dcc-4bp7j 1/1 Running 0 82m

httpd-57fc687dcc-4tbkx 1/1 Running 0 82m

httpd-57fc687dcc-nvzjm 1/1 Running 0 82m

httpd-57fc687dcc-pklr4 1/1 Running 0 82m

nginx-55c9c7dc96-7sktg 0/1 Pending 0 90m

nginx-55c9c7dc96-j4gtt 0/1 Pending 0 90m

nginx-55c9c7dc96-j7bjv 0/1 Pending 0 90m

nginx-649d445686-4rt6c 0/1 Pending 0 65m

nginx-649d445686-zgdks 0/1 Pending 0 65m

root@instance-1:~# kubectl -n demo exec -it httpd-57fc687dcc-4bp7j -- /bin/bash

root@httpd-57fc687dcc-4bp7j:/usr/local/apache2# cd htdocs/

root@httpd-57fc687dcc-4bp7j:/usr/local/apache2/htdocs# cat index.html

<html><body><h1>It works!</h1></body></html>

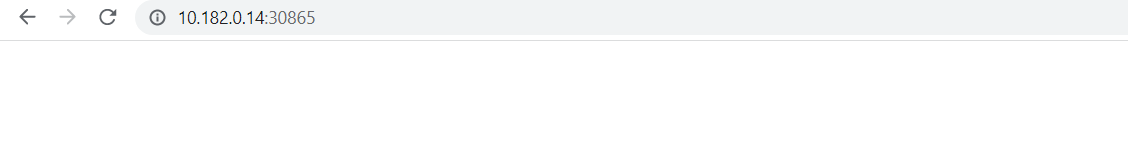
root@httpd-57fc687dcc-4bp7j:/usr/local/apache2/htdocs# vim index.html

root@httpd-57fc687dcc-4bp7j:/usr/local/apache2/htdocs# cat index.html

<html><body><h1>pod1</h1></body></html>

root@httpd-57fc687dcc-4bp7j:/usr/local/apache2/htdocs#

Above step needs to be followed for the pod2.



root@instance-1:~# kubectl get ingress -n demo

NAME CLASS HOSTS ADDRESS PORTS AGE

nginx-ingress <none> \* 10.182.0.16 80 10m

root@instance-1:~# kubectl describe ingress nginx-igres -n demo

Error from server (NotFound): ingresses.networking.k8s.io "nginx-igres" not found

root@instance-1:~#

root@instance-1:~# kubectl describe ingress nginx-ingres -n demo

Name: nginx-ingress

Labels: <none>

Namespace: demo

Address: 10.182.0.16

Ingress Class: <none>

Default backend: <default>

Rules:

Host Path Backends

---- ---- --------

\*

/ httpd:80 (<error: endpoints "httpd" not found>)

/foo nginx:80 (<none>)

Annotations: nginx.ingress.kubernetes.io/rewrite-target: /

Events:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal CREATE 11m nginx-ingress-controller Ingress demo/nginx-ingress

Normal UPDATE 11m nginx-ingress-controller Ingress demo/nginx-ingress

root@instance-1:~# kubectl get pods -n demo -o wide

NAME READY STATUS RESTARTS AGE IP NODE NOMINATED NODE READINESS GATES

httpd-57fc687dcc-4bp7j 1/1 Running 0 91m 10.244.1.3 slave1 <none> <none>

httpd-57fc687dcc-4tbkx 1/1 Running 0 91m 10.244.1.4 slave1 <none> <none>

httpd-57fc687dcc-nvzjm 1/1 Running 0 91m 10.244.2.4 slave2 <none> <none>

httpd-57fc687dcc-pklr4 1/1 Running 0 91m 10.244.2.3 slave2 <none> <none>

nginx-55c9c7dc96-7sktg 0/1 Pending 0 100m <none> <none> <none> <none>

nginx-55c9c7dc96-j4gtt 0/1 Pending 0 100m <none> <none> <none> <none>

nginx-55c9c7dc96-j7bjv 0/1 Pending 0 100m <none> <none> <none> <none>

nginx-649d445686-4rt6c 0/1 Pending 0 74m <none> <none> <none> <none>

nginx-649d445686-zgdks 0/1 Pending 0 74m <none> <none> <none> <none>

root@instance-1:~#

Note: bath backend and the ip address of the pods are same.

Quota:

|  |
| --- |
| root@kmaster:~# cat quota.yml |
|  | apiVersion: v1 |
|  | kind: ResourceQuota |
|  | metadata: |
|  | name: mem-cpu-demo |
|  | namespace: demo |
|  | spec: |
|  | hard: |
|  | requests.cpu: "1" |
|  | requests.memory: 1Gi |
|  | limits.cpu: "2" |
|  | limits.memory: 2Gi |

root@instance-1:~# kubectl create -f quota.yml

resourcequota/mem-cpu-demo created

root@instance-1:~# kubectl get quota -n demo

NAME AGE REQUEST LIMIT

mem-cpu-demo 15s requests.cpu: 2500m/1, requests.memory: 5Gi/1Gi limits.cpu: 5/2, limits.memory: 10Gi/2Gi

root@instance-1:~#

kubectl logs httpd- -n demo –f-------🡪 this is for checking the logs in the running format.

root@kubeslave1:~# history

1 pwd

2 clear

3 sudo apt-get update

4 sudo apt-get install ca-certificates curl gnupg lsb-release

5 sudo mkdir -p /etc/apt/keyrings

6 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

7 echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

8 sudo apt-get update

9 sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

10 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

11 apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

12 docker ps

13 kubeadm join 10.182.0.9:6443 --token rjlx9a.g4ob7cjib4krw5fy --discovery-token-ca-cert-hash sha256:e3579dddc1499cd5a476d2b80c90f8743cd29e94711900df276a9ad6d7a263fc

14 apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

15 kubeadm join 10.182.0.9:6443 --token rjlx9a.g4ob7cjib4krw5fy --discovery-token-ca-cert-hash sha256:e3579dddc1499cd5a476d2b80c90f8743cd29e94711900df276a9ad6d7a263fc

16 apt-get install kubeadm=1.20.0-00 kubelet=1.20.0-00 kubectl -y

17 kubeadm join 10.182.0.9:6443 --token rjlx9a.g4ob7cjib4krw5fy --discovery-token-ca-cert-hash sha256:e3579dddc1499cd5a476d2b80c90f8743cd29e94711900df276a9ad6d7a263fc

18 histoey

19 history

Configuring slave2:

root@kubeslave-2:~# history

1 pwd

2 sudo apt-get update

3 sudo apt-get install ca-certificates curl gnupg lsb-release

4 sudo mkdir -p /etc/apt/keyrings

5 curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

6 echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

7 sudo apt-get update

8 sudo apt*-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin*

9 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

10 apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

11 apt-get install kubeadm=1.20.0-00 kubelet=1.20.0-00 kubectl -y

12 kubeadm join 10.182.0.9:6443 --token rjlx9a.g4ob7cjib4krw5fy --discovery-token-ca-cert-hash sha256:e3579dddc1499cd5a476d2b80c90f8743cd29e94711900df276a9ad6d7a263fc

13 history