**Kubernetes Networking & Kubernetes Services, Nodeport and Volumes**

Note:

1. Pods have ip & containers have no ip
2. If any container want to communicate any other containers they always hit on the pod ip

***LABS***

1. **Communication between the 2 different pods on same nodes**

**//first create 2 pods separately on same node and run 1 conatiner in each with HTTPD and Ubuntu image with yml files**

**Creating pods**

* nano co1

kind: Pod

apiVersion: v1

metadata:

name: testpod1

spec:

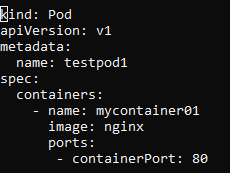
containers:

- name: mycontainer01

image: nginx

ports:

- containerPort: 80



* Kubectl apply –f co1.yml // create the pods
* nano co2

kind: Pod

apiVersion: v1

metadata:

name: testpod2

spec:

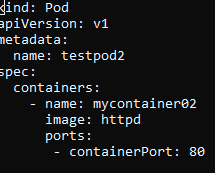
containers:

- name: mycontainer02

image: httpd

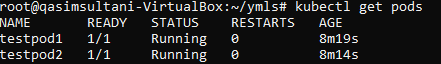
ports:

- containerPort: 80

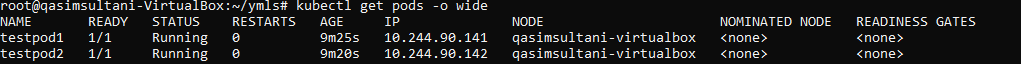


* Kubectl apply –f co2.yml // create the pods

//2 pods are create and running sucesfully

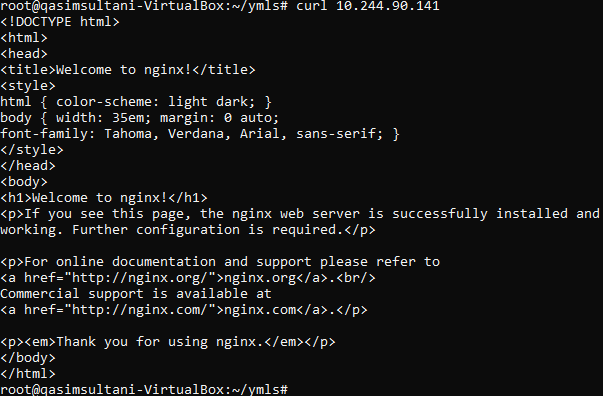


//Now check ip of the pods to communicate with them



//now check and confirm can we communicate with the pods individually for this used curl command

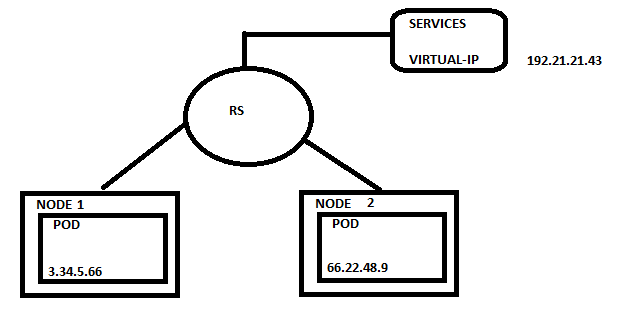
* curl pod-IP





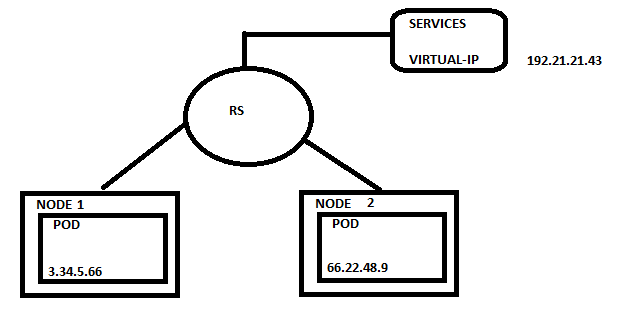
//YOU can see that both apache and nginx are running on both both and we can communicated this successfully

//Till here You can learn how to communicated pods but there is exception what if one container support other container like backend container support frontend container and if the container destroy due to any reason and you have created Replica set so the new container created to support so you can remember the IP of new container ?

  
  
**ANSWER: BIG NO**

**So here is New Object is “SERVICES”**

Services object came with this solution of Virtual-IP, it’s IP came on above the replicaset so that once any container gone down/destroy services object virtual ip is there and once any new container created services virtual ip redirected the container ip into virtual-ip so you have to remember only virtual-ip



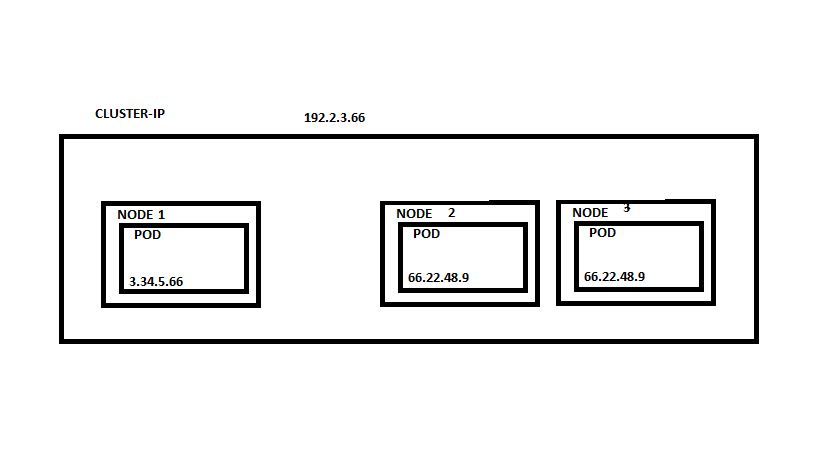
SERVICES can be done with these 4 Tyoes

* Cluster-IP
* Nodeport
* Loadbalancer
* Headlers

We work only Cluster-IP and nodeportthe other 2 methods are already in Cloud so that we cannot dicuss here

1. ***Cluster-IP***

* You cannot access cluster-ip outside the cluster
* You can communicate each pod with in the cluster



***LABS***

**Nano deply.yml**

**kind: Deployment**

**apiVersion: apps/v1**

**metadata:**

**name: mydeployments**

**spec:**

**replicas: 1**

**selector: # tells the controller which pods to watch/belong to**

**matchLabels:**

**name: deployment**

**template:**

**metadata:**

**name: testpod1**

**labels:**

**name: deployment**

**spec:**

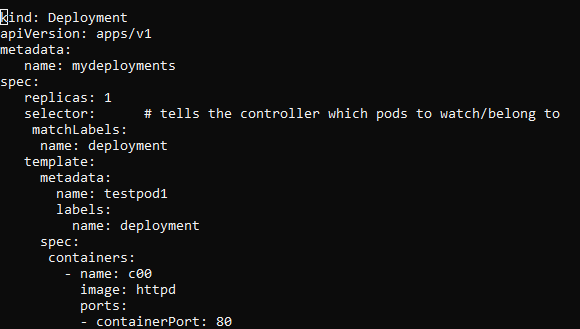
**containers:**

**- name: c00**

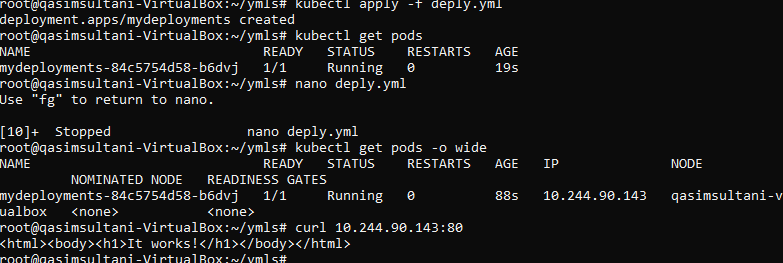
**image: httpd**

**ports:**

**- containerPort: 80**



**//now create pod and check ip**



**//now pods is create next step is create services with selector so that this services is allocated to the pods who have same selecter and labels**

* **Nano services.yml**

**kind: Service # Defines to create Service type Object**

**apiVersion: v1**

**metadata:**

**name: demoservice**

**spec:**

**ports:**

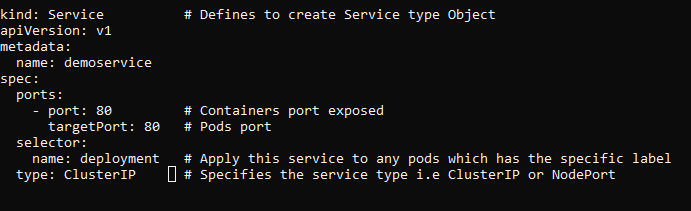
**- port: 80 # Containers port exposed**

**targetPort: 80 # Pods port**

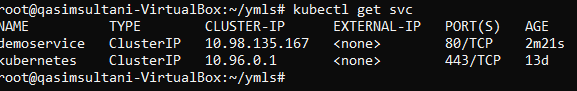
**selector:**

**name: deployment # Apply this service to any pods which has the specific label**

**type: ClusterIP # Specifies the service type i.e ClusterIP or NodePort**

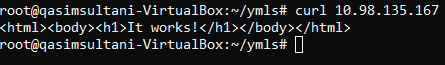


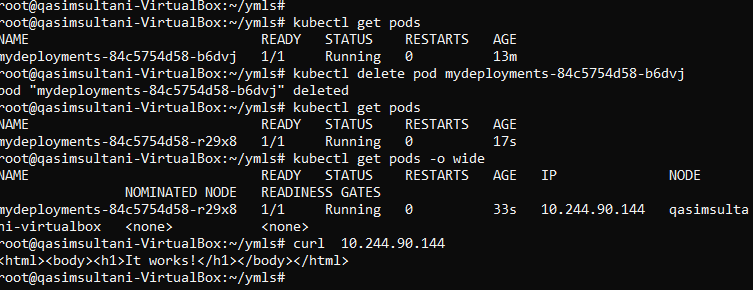
* **Kubectl apply –f services.yml**
* **kubectl get svc //to view services details & IP**



**//k8s is default ignor it**

**//check Virtual-ip is easily available to us**



**//now delete the existing pods and in replicaset new pods create and get again comminucated with virtual ip to check our cluster-ip services working fine.**

**//yes, we can easily access your pods after delete with same virtual-ip**

**//now delete all pods & service**

**kubectl delete -f services.yml //filename of pods & services**

1. ***NODEPORT***

* You can access the pod out side the cluster from internet
* You will get port range from 30,000 – 32,767

//create pods

* nano deployment.yml

**kind: Deployment**

**apiVersion: apps/v1**

**metadata:**

**name: mydeployments**

**spec:**

**replicas: 1**

**selector: # tells the controller which pods to watch/belong to**

**matchLabels:**

**name: deployment**

**template:**

**metadata:**

**name: testpod1**

**labels:**

**name: deployment**

**spec:**

**containers:**

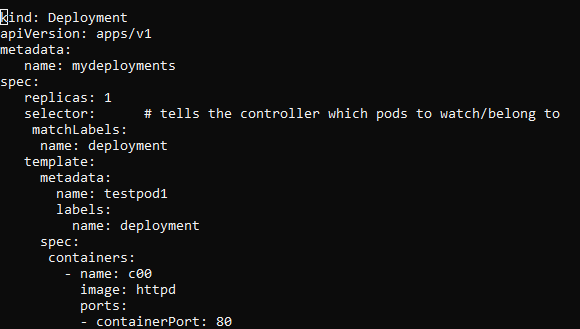
**- name: c00**

**image: httpd**

**ports:**

**- containerPort: 80**

* **Kubectl apply –f deployment.yml**



**//now create services**

* **Nano services.yml**

**kind: Service # Defines to create Service type Object**

**apiVersion: v1**

**metadata:**

**name: demoservice**

**spec:**

**ports:**

**- port: 80 # Containers port exposed**

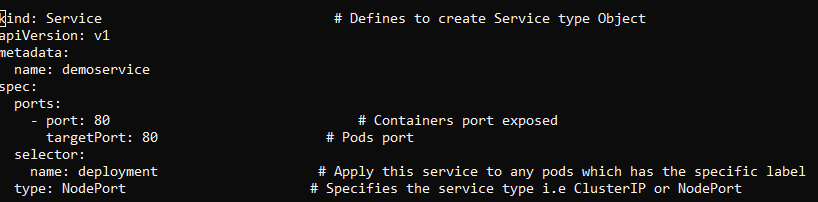
**targetPort: 80 # Pods port**

**selector:**

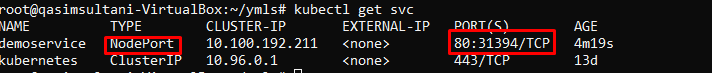
**name: deployment # Apply this service to any pods which has the specific label**

**type: NodePort # Specifies the service type i.e ClusterIP or NodePort**

* **kubectl apply –f services.yml**



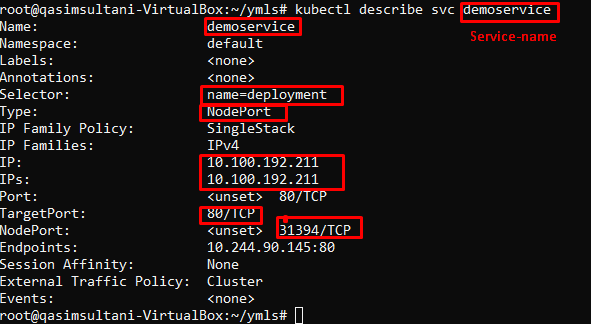
// now check service details



// you will get port range between 30000-32767

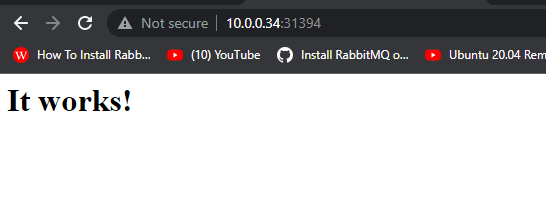
//To check more details about services

* Kubectl describe svc service-name



//to check ip and port in easily acess through the internet just paste the ip and port and it’s shows output. Hence, if you are using AWS cloud just allowed the port from the security group on your machine

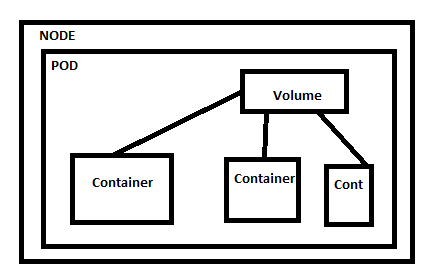
* paste on browser **10.0.0.34:31394 //ip:port**



**//now we can easily view our container from internet**

* **//delete all the files and services   
  kubectl delete -f deployment.yml**

***VOLUMES***



* You cannot acess volume of pod outside the pod

***LAB***

**Create a pod with 2 container which was easily share volume of bot container**

“Volume type emptydir”

nano emptydir.yml

apiVersion: v1

kind: Pod

metadata:

name: myvolemptydir

spec:

containers:

- name: c1

image: centos

command: ["/bin/bash", "-c", "sleep 15000"]

volumeMounts: # Mount definition inside the container

- name: xchange

mountPath: "/tmp/xchange"

- name: c2

image: centos

command: ["/bin/bash", "-c", "sleep 10000"]

volumeMounts:

- name: xchange

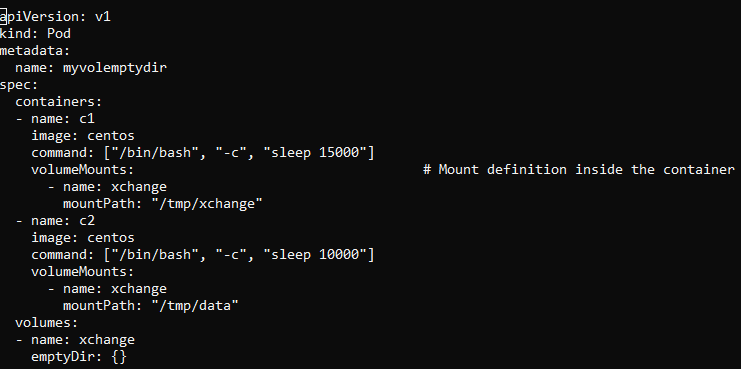
mountPath: "/tmp/data"

volumes:

- name: xchange

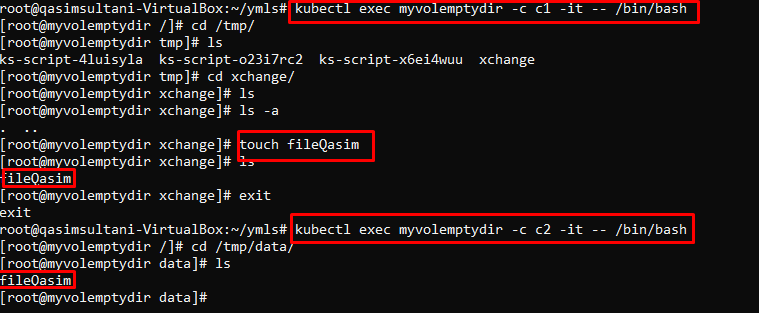
emptyDir: {}

* kubectl apply –f emptydir.yml

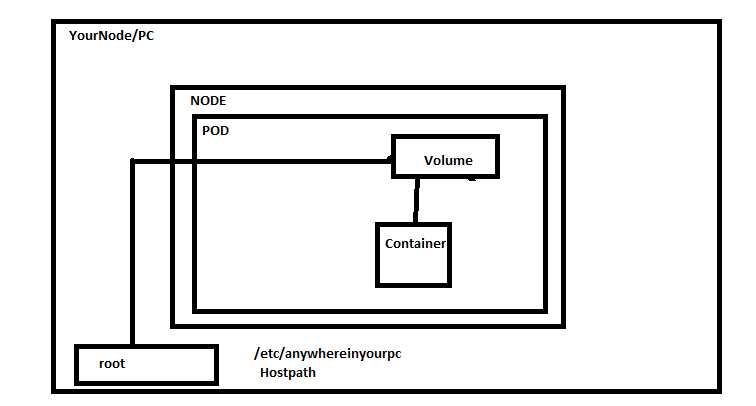


//pods are create now go into the pods and any container create any file on the path which we define in our yml script and view the file from other container from the specific path that share volume

* kubectl exec myvolemptydir -c c1 -it -- /bin/bash



**//now if you want to access pods volume outside the other pods volume we used hostpath to mount volume**



* nano hostpath.yml

apiVersion: v1

kind: Pod

metadata:

name: myvolhostpath

spec:

containers:

- image: centos

name: testc

command: ["/bin/bash", "-c", "sleep 15000"]

volumeMounts:

- mountPath: /tmp/hostpath

name: testvolume

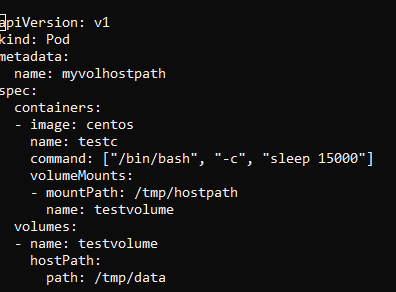
volumes:

- name: testvolume

hostPath:

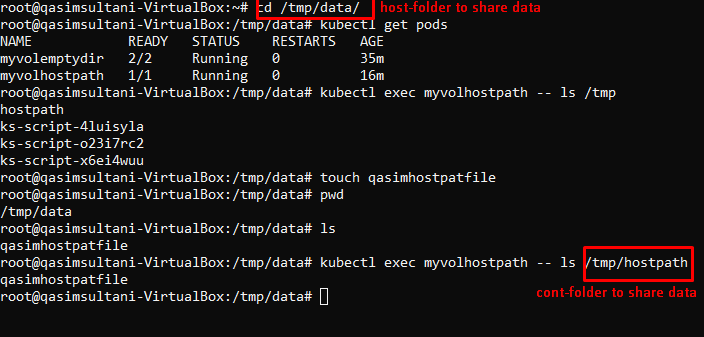
path: /tmp/data

* kubectl apply -f hostpath.yml

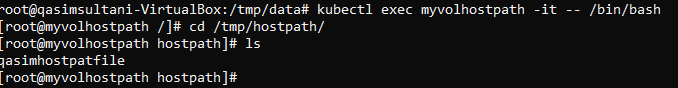


kubectl get pods

// now goto your hostpath loacation folder and enter your conatier & create file



//now o to your container and check file is present there or not



**Thankyou**