Nginx Deployment - k8s

t.me/vishalk17 github.com/vishalk17

—---

Create Persistent volume :

- vi nginx-pv.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nginx-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /data/nginx
```

The `spec` section of the PV resource defines the following properties:

- `capacity`: The capacity of the PV in gigabytes (Gi).
- `accessModes`: The access modes that the PV supports. In this case, the PV supports ReadWriteOnce, which means that it can be mounted as read-write by a single pod.
- `hostPath`: The path on the host machine where the PV is located. In this case, the PV is located at `/data/nginx`.

The `apiVersion` and `kind` fields of the PV resource specify the Kubernetes API version and resource type of the resource. In this case, the API version is `v1` and the resource type is `PersistentVolume`.

The `metadata` field of the PV resource specifies the name of the PV. In this case, the name of the PV is `nginx-pv`.

- vi nginx-pvc.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nginx-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      pv-name: nginx-pv
```

The `spec` section of the PVC resource defines the following properties:

- `accessModes`: The access modes that the PVC requests. In this case, the PVC requests ReadWriteOnce, which means that it can be mounted as read-write by a single pod.
- `resources`: The resources that the PVC requests. In this case, the PVC requests 1 gigabyte (Gi) of storage.
- `selector`: The selector that the PVC uses to match a PV. In this case, the PVC matches PVs that have the label `pv-name: nginx-pv`.

The `apiVersion` and `kind` fields of the PVC resource specify the Kubernetes API version and resource type of the resource. In this case, the API version is `v1` and the resource type is `PersistentVolumeClaim`.

The `metadata` field of the PVC resource specifies the name of the PVC. In this case, the name of the PVC is `nginx-pvc`.

- vi nginx-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx-ingress
spec:
  rules:
  - host: vishalk17.google.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

n Ingress resource is a Kubernetes resource that defines how external traffic is routed to services within a Kubernetes cluster.

The `spec` section of the Ingress resource defines the following properties:

- `rules`: The rules that define how traffic is routed to services. In this case, the Ingress resource defines a single rule that routes traffic to the `nginx-service` service for the `example.com` host.
- `http`: The HTTP configuration for the Ingress resource. In this case, the HTTP configuration specifies that all requests to the `example.com` host should be routed to the `nginx-service` service.
- `paths`: The paths that are matched by the Ingress resource. In this case, the Ingress resource matches all requests to the `/` path.
- `backend`: The backend service that is used to handle requests that match the Ingress resource. In this case, the backend service is the `nginx-service` service.

The `apiVersion` and `kind` fields of the Ingress resource specify the Kubernetes API version and resource type of the resource. In this case, the API version is `networking.k8s.io/v1` and the resource type is `Ingress`.

—--

The `metadata` field of the Ingress resource specifies the name of the Ingress resource. In this case, the name of the Ingress resource is `nginx-ingress`.

- vi nginx-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Sure. The YAML code you provided defines a Service resource named `nginx-service`. A Service resource in Kubernetes is a logical abstraction of a set of Pods. It defines a single point of access for a set of Pods, and provides load balancing and other features.

The `spec` section of the Service resource defines the following properties:

- `selector`: The selector that is used to match Pods. In this case, the selector matches Pods that have the label `app: nginx`.
- `type`: The type of Service. In this case, the Service is of type `LoadBalancer`, which means that it will be exposed as a load balancer.
- `ports`: The ports that are exposed by the Service. In this case, the Service exposes port 80.

The `apiVersion` and `kind` fields of the Service resource specify the Kubernetes API version and resource type of the resource. In this case, the API version is `v1` and the resource type is `Service`.

The `metadata` field of the Service resource specifies the name of the Service resource. In this case, the name of the Service resource is `nginx-service`.

- vi nginx-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      initContainers:
      - name: init-nginx
        image: centos
        command: ["/bin/sh", "-c", "echo 'Hello, World!' > /data/nginx/vishalk17.txt"]
        volumeMounts:
        - name: nginx-pv-volume
          mountPath: /data/nginx
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
        volumeMounts:
        - name: nginx-pv-volume
          mountPath: /data/nginx
      volumes:
      - name: nginx-pv-volume
        persistentVolumeClaim:
          claimName: nginx-pvc
```

The `spec` section of the Deployment resource defines the following properties:

- `replicas`: The number of Pods that should be created by the Deployment. In this case, the Deployment will create 1 Pod.
- `selector`: The selector that is used to match Pods. In this case, the selector matches Pods that have the label `app: nginx`.
- `template`: The template that is used to create Pods. The template defines the Pod's configuration, such as the image that should be used, the ports that should be exposed, and the volumes that should be mounted.

The `initContainers` section of the Deployment resource defines a list of initContainers. InitContainers are Containers that are run before the main Container in a Pod is started. In this case, the initContainer is used to create a file named `vishalk17.txt` in the `/data/nginx` directory.

The `containers` section of the Deployment resource defines a list of containers. Containers are the main components of a Pod. In this case, the container is used to run the Nginx web server. The container is configured to expose port 80.

The `volumes` section of the Deployment resource defines a list of volumes. Volumes are used to store data that is persistent across Pod restarts. In this case, the volume is a PersistentVolumeClaim (PVC) named `nginx-pvc`. The PVC is used to create a persistent volume that can be mounted by the Pods.

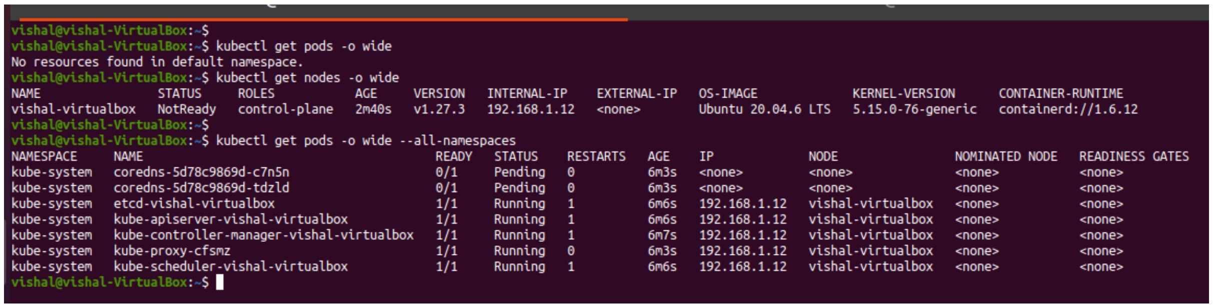
The `apiVersion` and `kind` fields of the Deployment resource specify the Kubernetes API version and resource type of the resource. In this case, the API version is `apps/v1` and the resource type is `Deployment`.

The `metadata` field of the Deployment resource specifies the name of the Deployment resource. In this case, the name of the Deployment resource is `nginx-deployment`.

—---

So, this is the fresh install of k8s network , so we need pod network too.

kubectl get pods -o wide --all-namespaces



```
vishal@vishal-VirtualBox:~$
vishal@vishal-VirtualBox:~$ kubectl get pods -o wide
No resources found in default namespace.
vishal@vishal-VirtualBox:~$ kubectl get nodes -o wide
NAME               STATUS     ROLES           AGE     VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
vishal-virtualbox  NotReady   control-plane   2m40s   v1.27.3   192.168.1.12   <none>        Ubuntu 20.04.6 LTS   5.15.0-76-generic   containerd://1.6.12
vishal@vishal-VirtualBox:~$
vishal@vishal-VirtualBox:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                        READY   STATUS    RESTARTS   AGE    IP             NODE                NOMINATED NODE   READINESS GATES
kube-system   coredns-5d78c9869d-c7n5n                    0/1     Pending   0          6m3s   <none>         <none>              <none>           <none>
kube-system   coredns-5d78c9869d-tdzld                    0/1     Pending   0          6m3s   <none>         <none>              <none>           <none>
kube-system   etcd-vishal-virtualbox                      1/1     Running   1          6m6s   192.168.1.12   vishal-virtualbox   <none>           <none>
kube-system   kube-apiserver-vishal-virtualbox            1/1     Running   1          6m6s   192.168.1.12   vishal-virtualbox   <none>           <none>
kube-system   kube-controller-manager-vishal-virtualbox   1/1     Running   1          6m7s   192.168.1.12   vishal-virtualbox   <none>           <none>
kube-system   kube-proxy-cfsmz                            1/1     Running   0          6m3s   192.168.1.12   vishal-virtualbox   <none>           <none>
kube-system   kube-scheduler-vishal-virtualbox            1/1     Running   1          6m6s   192.168.1.12   vishal-virtualbox   <none>           <none>
vishal@vishal-VirtualBox:~$
```

#You will notice from the previous command, that all the pods are running except one: ‘kube-dns’. For resolving this we will install a # pod network. To install the weave pod network, run the following command:

releases : https://github.com/weaveworks/weave/releases/

installing latest version:

kubectl apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s-1.11.yaml

Now, status should have changed

—---- next step is to install ingress controller and metal lb , if you dont have —

Follow my other guide pdf else search on internet

—--

—---
Final stage:

Apply all manifest files ,

```
kubectl apply -f nginx-pv.yaml
kubectl apply -f nginx-pvc.yaml
```

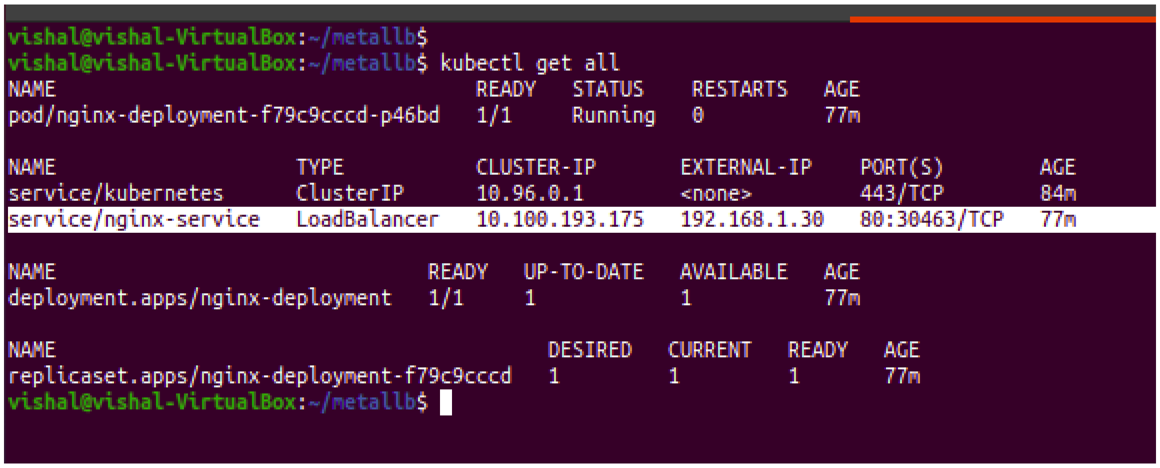
——---

kubectl apply -f nginx-ingress.yaml
kubectl apply -f nginx-service.yaml
kubectl apply -f nginx-deployment.yaml

——---

Let access , nginx through the web ,

As we have configure, nginx ingress controller , it will take responsibility of routing traffic to desired pod. And metallb provide us external ip with which we can access the service



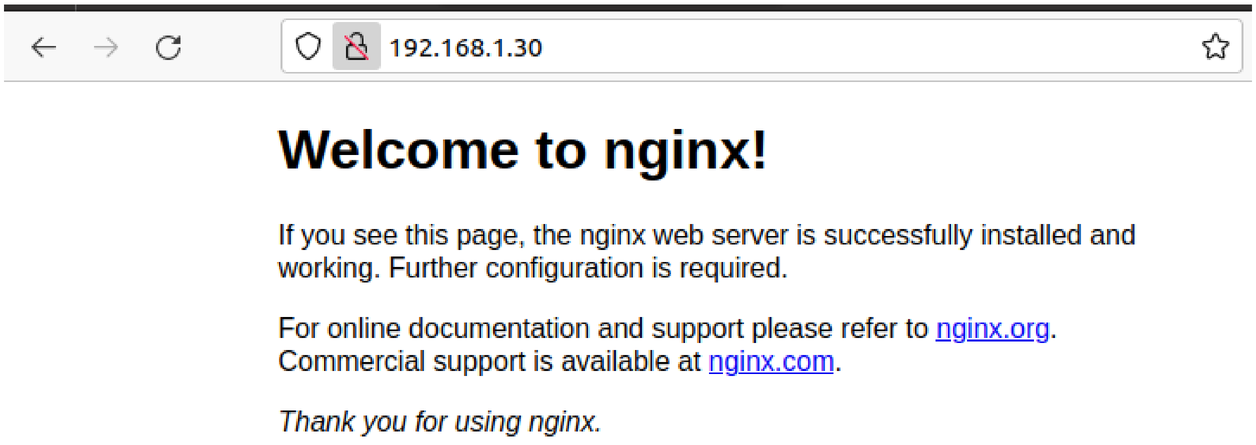
```
vishal@vishal-VirtualBox:~/metallb$
vishal@vishal-VirtualBox:~/metallb$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-f79c9cccd-p46bd    1/1     Running   0          77m

NAME                     TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)        AGE
service/kubernetes       ClusterIP      10.96.0.1        <none>         443/TCP        84m
service/nginx-service    LoadBalancer   10.100.193.175   192.168.1.30   80:30463/TCP   77m

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment    1/1     1            1           77m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-f79c9cccd    1         1         1       77m
vishal@vishal-VirtualBox:~/metallb$
```

Go though this ip ,



192.168.1.30

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Summary:

- Pod network is required in order to communicate nodes with each other
- Nginx ingress controller require for ip routing
- Metalb load balancer provide external ip with with app or service open for the world

Sourcecode :

- https://github.com/vishalk17/devops/tree/main/kubernetes

My devops repo :

- https://github.com/vishalk17/devops

My telegram channel:

- https://t.me/vishalk17_devops

Contact:

Telegram : t.me/vishalk17

vishalk17 My youtube Channel :

- YouTube https://www.youtube.com/@vishalk17