Day 4 - Advanced Kubernetes [PV & PVC]

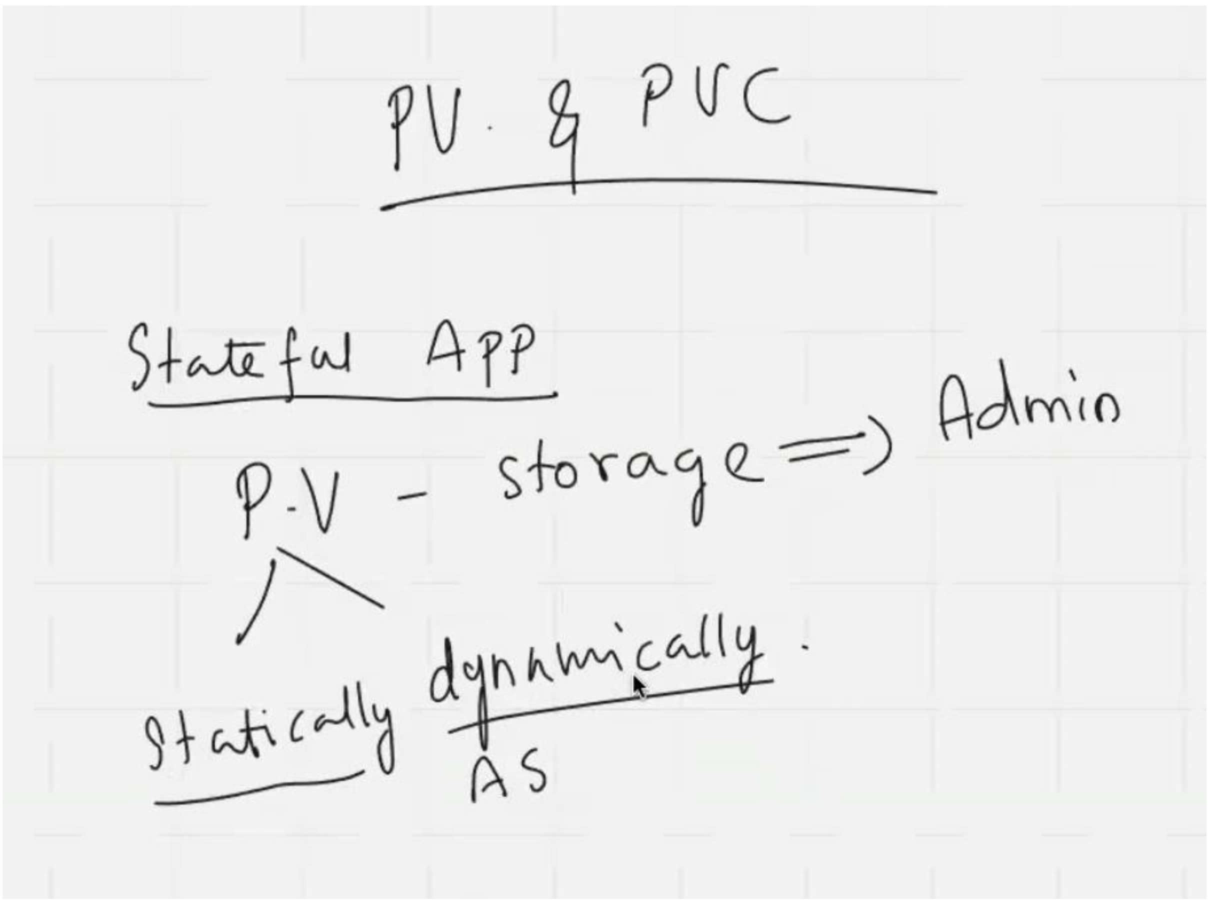
- Mainly we use volume in order to persist. Your containers data, because your container can anytime it can be deleted, or it stop working. So whenever it goes deleted, we have to set. We have to retain the data of your application.
- So, in order to retain the data in case, if you're create running a Mysql container in my sql, you have your entire database. **for example**, the user data or the registration details or the product details, everything will be there inside your application. Only so not maintain or persist your data that belongs to the application. We go with volumes.
- So if you're using docker container, if you're only using docker container, you can go with volumes you can attach, you can simply create a volume and attach a volume to your docker container. So even though when your container gets deleted, you can use this volume, you can launch a new container. You can attach this volume. So what will happen? You can restore the data from the previous container.
- Same way here in the Kubernetes cluster, if you. If your application is a stateful application / dynamic application which generates some sort of data here in Kubernetes.
- Okay, you're making some say. **for example**, you have a login page where you enter your username and password. So you're persisting the username and password for the authentication. So that has to be stored by the back end to your database. So that is a stateful application.
- so, this persistent volume will connect with the PVC.
- PVC Is nothing but persistent volume.
- State less application/static application, it will not get any user data. It does not have to persist any data. No transaction will be happening just only the information will be displayed like your blogs or or your news website, or any website where just an information is shared.
- So for stateful application we have to use. If your stateful application is deployed in the Kubernetes cluster, you have to configure Pv and PVC.
- So mainly pv is nothing but a piece of storage. Persistent volume is nothing but a piece of storage.

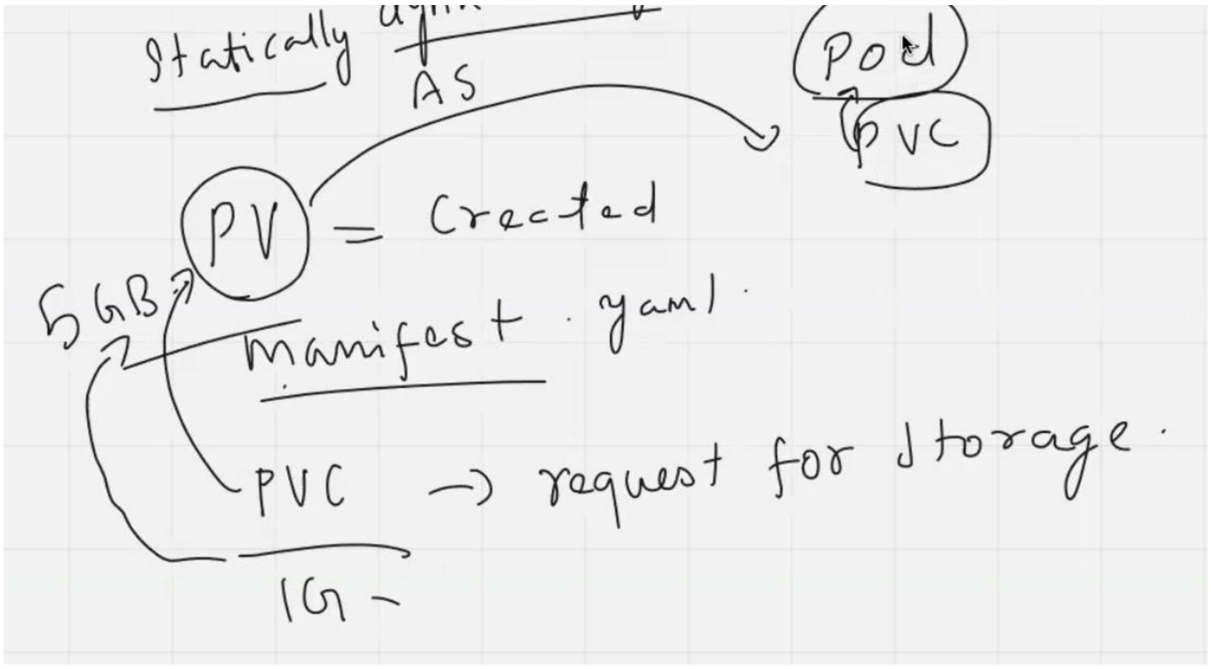
- This storage is created or provisioned by a cluster administrator. So, your cloud administrator, or Kubernetes administrator or cluster administrator.

Create the PV:

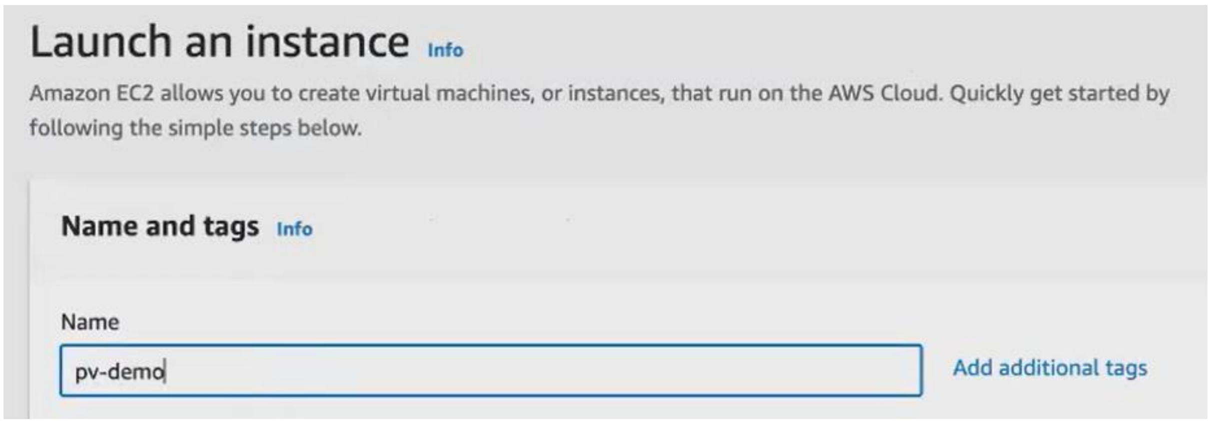
- we will be creating the persistent volume. You 1st create the volume you mentioned the capacity, storage capacity. How many Gbs are required?
- What is the access more for this volume, whether you have to provide a read access, or only read, write, access, or only read only access. So, all those things can be provided. So based on the access mode you give for Pv, a user can access this. Pv, if you create a access more with read, write once, only one time your user can read and write into this. Pv.
- if you set the access code as read, only user can only read this Pv. So, he can only read the Pv. He will not be able to put any data ,write any data to this pv.
- whatever it is required, and where you want to create the Pv**.**
- we will be creating the Pv.
    - create a manifest file for creating deployment
    - create a services file for create a manifest file.
    - Create .Yaml file for Pv creation.
- So, in that Yaml file you will be mentioning you have to specify the capacity you have to specify the access mode, and you have to specify the host path.
- host path is nothing but you have to specify the path on your host machine where the volume or where the Pv should be located.
- **PVC:** **[**request for your volume]
- **Persistent Volume Claim** in Kubernetes.
- It is a request for storage.
- So when you create Pv. Who will create the cloud administrator will create the Pv. So he will create Pv. And he will keep it in his infrastructure.
- When, if your developer or someone, the team, requires a certain amount of storage, so what they will do they will write a PVC. They will create a PVC. Is nothing but request for your volume.
- **PVC Is like you're requesting for some volume.**

- **So, my PVC I can claim up to 5 Gb only I cannot claim beyond 5 Gb. So, I can only claim 5 Gb Or less than 5 Gb.**
- **While creating pvc write**
    - manifest file
- So in the manifest file when creating your Pvc, you have to mention the access .
- how you have to mention access for your PVC, so it should exactly match with the access of your Pv**.**
- After creating your PVC. We will be binding the PVC with your PV. After binding it. After binding a PVC With Pv. We have to attach or mount this volume to Pv.(i.e)To the pod or to the deployment
- Pod creates the container **.inside the pod template, I have to mention the PVC.**





Launch an instance for pv:



Launch an instance Info

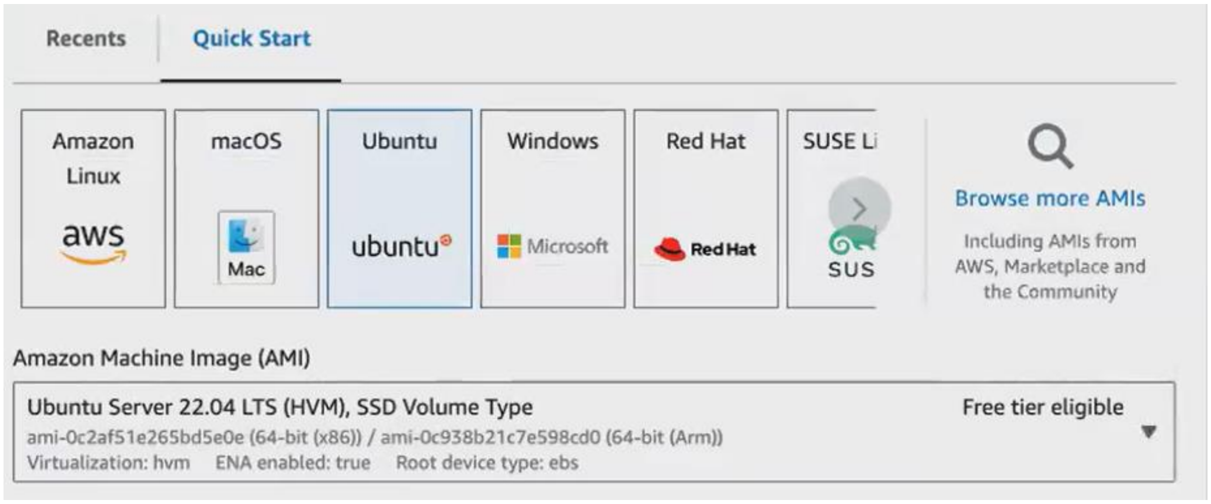
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name

pv-demo

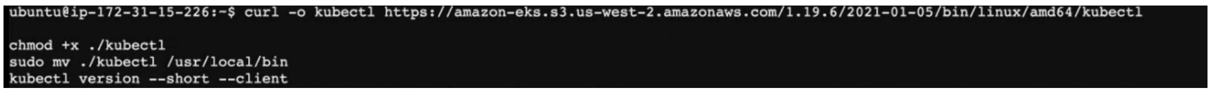
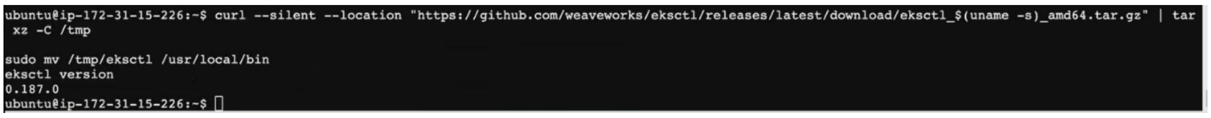
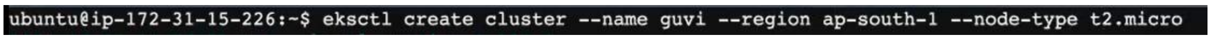
Add additional tags



Amazon Machine Image (AMI)

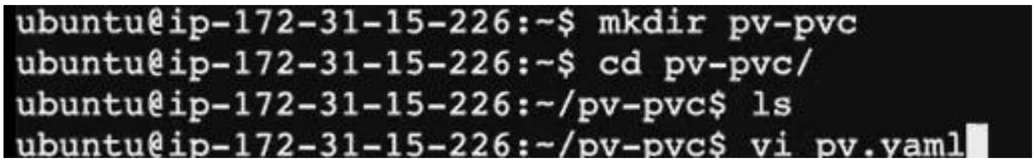
Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-0c2af51e265bd5e0e (64-bit (x86)) / ami-0c938b21c7e598cd0 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Launch instance and connect it and launch a cluster

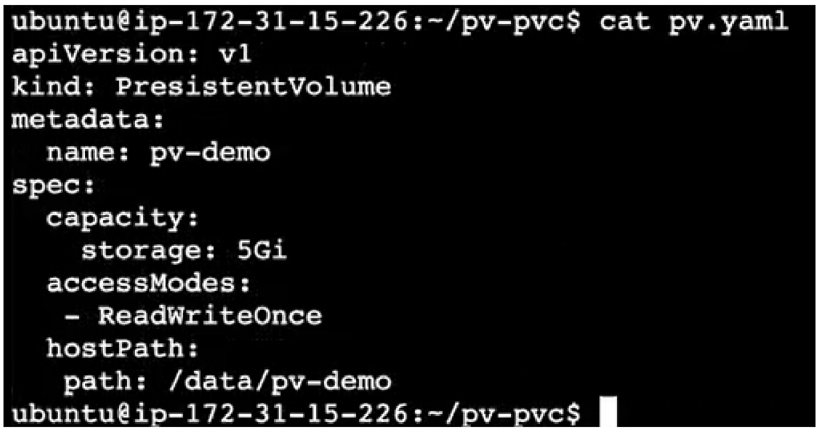
1. Sudo apt update
2. Sudo apt install awscli -y
3. 
```
ubuntu@ip-172-31-15-226:~$ curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin
kubectl version --short --client
```
4. 
```
ubuntu@ip-172-31-15-226:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar
 xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin
eksctl version
0.187.0
ubuntu@ip-172-31-15-226:~$
```
5. Aws configure [give access key, secret access key]
6. 
```
ubuntu@ip-172-31-15-226:~$ eksctl create cluster --name guvi --region ap-south-1 --node-type t2.micro
```

Duplicate an instance and write files for pv amd pvc

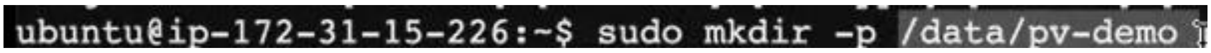
File for PV:


```
ubuntu@ip-172-31-15-226:~$ mkdir pv-pvc
ubuntu@ip-172-31-15-226:~$ cd pv-pvc/
ubuntu@ip-172-31-15-226:~/pv-pvc$ ls
ubuntu@ip-172-31-15-226:~/pv-pvc$ vi pv.yaml
```

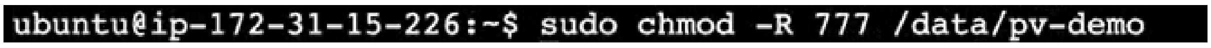
- Pv will be created by your administrators.


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ cat pv.yaml
apiVersion: v1
kind: PresistentVolume
metadata:
  name: pv-demo
spec:
  capacity:
    storage: 5Gi
  accessModes:
   - ReadWriteOnce
  hostPath:
   path: /data/pv-demo
ubuntu@ip-172-31-15-226:~/pv-pvc$
```

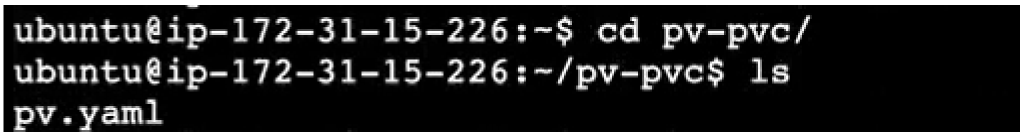
In this path my volume created. so, give permission for this


```
ubuntu@ip-172-31-15-226:~$ sudo mkdir -p /data/pv-demo
```

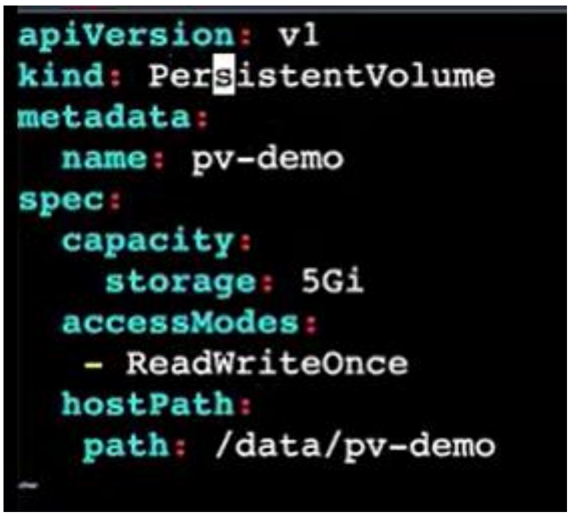
- -R means to all the Directory within this, even to the sub directories and to the files that belong to this particular directory. Okay to all the directories recursively, this function will be applied.
- 777 means all your read, write, and execute, permission will be recursively given to all the user.



```
ubuntu@ip-172-31-15-226:~$ sudo chmod -R 777 /data/pv-demo
```

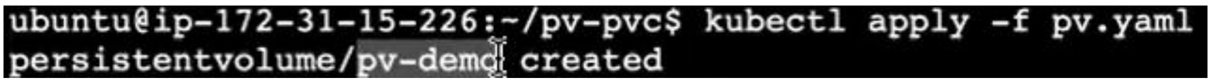


```
ubuntu@ip-172-31-15-226:~$ cd pv-pvc/
ubuntu@ip-172-31-15-226:~/pv-pvc$ ls
pv.yaml
```



```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-demo
spec:
  capacity:
    storage: 5Gi
  accessModes:
   - ReadWriteOnce
  hostPath:
   path: /data/pv-demo
~
```

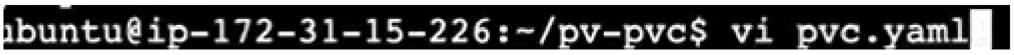
After created cluster then only execute this file.



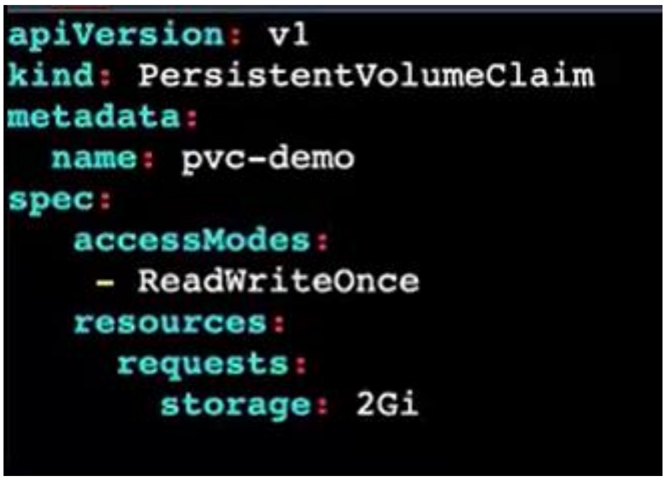
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl apply -f pv.yaml
persistentvolume/pv-demo created
```

File for PVC:

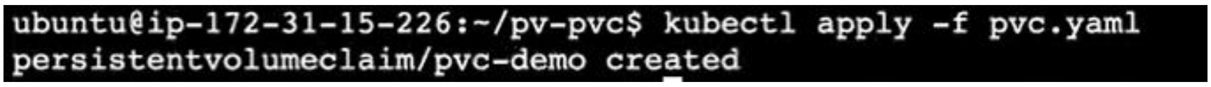
PVC Will be created by your Devops engineers or your development team, who requires the storage.



```
ubuntu@ip-172-31-15-226:~/pv-pvc$ vi pvc.yaml
```

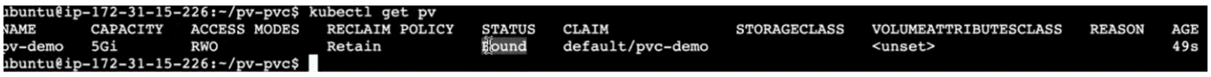


```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-demo
spec:
   accessModes:
    - ReadWriteOnce
   resources:
     requests:
       storage: 2Gi
```



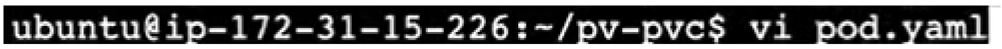
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl apply -f pvc.yaml
persistentvolumeclaim/pvc-demo created
```

Once pvc mounted with pv ,status as BOUND



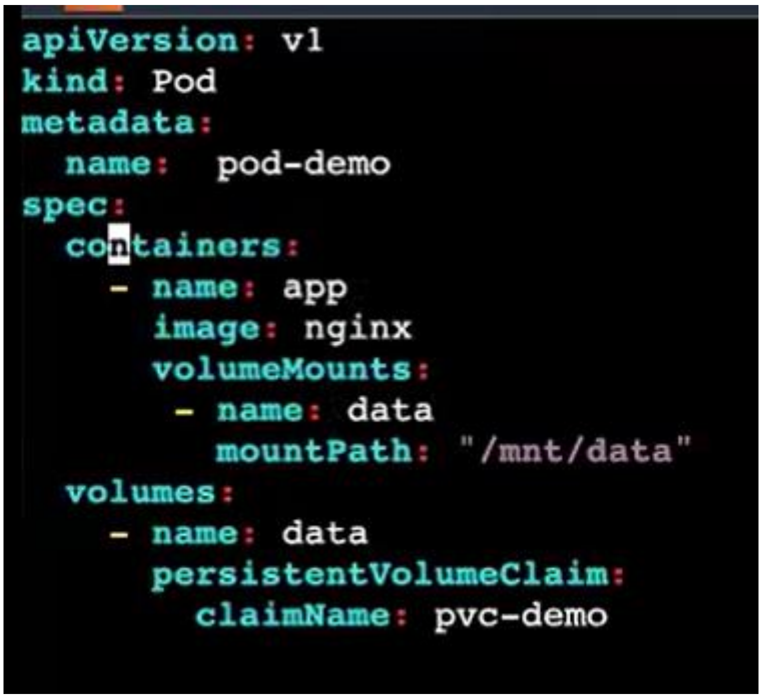
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl get pv
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pv-demo    5Gi        RWO            Retain           Bound    default/pvc-demo                  <unset>                          49s
ubuntu@ip-172-31-15-226:~/pv-pvc$
```

File for POD:

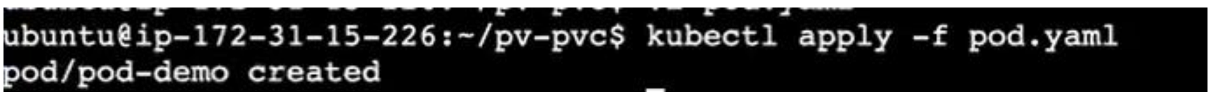


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ vi pod.yaml
```

In this running the nginx application. In real time use some real-time application like get user data.

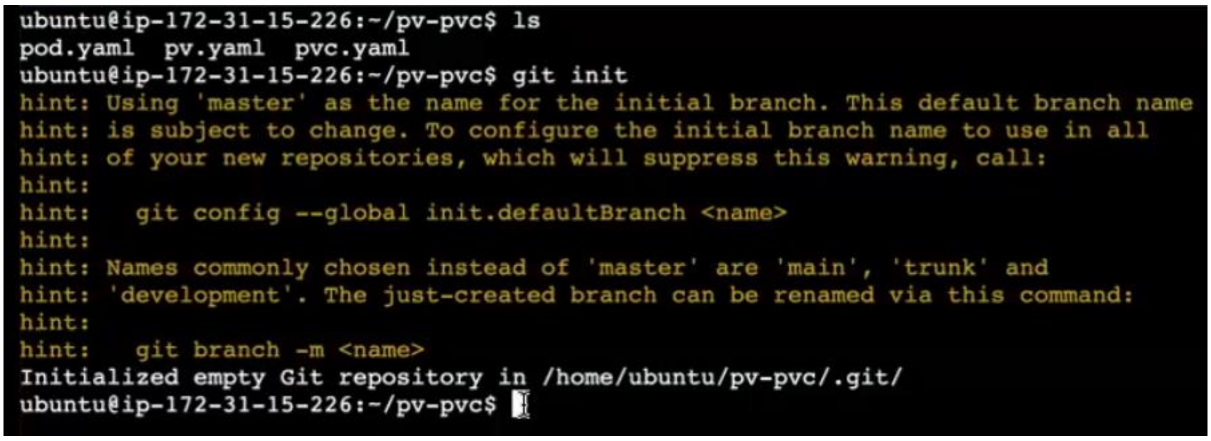


```
apiVersion: v1
kind: Pod
metadata:
  name:  pod-demo
spec:
  containers:
    - name: app
      image: nginx
      volumeMounts:
        - name: data
          mountPath: "/mnt/data"
  volumes:
    - name: data
      persistentVolumeClaim:
        claimName: pvc-demo
```

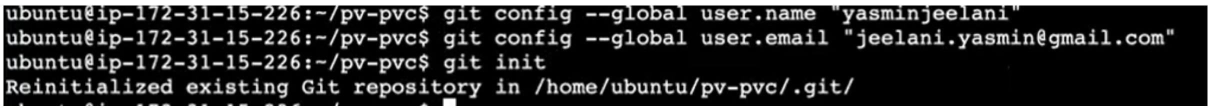


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl apply -f pod.yaml
pod/pod-demo created
```

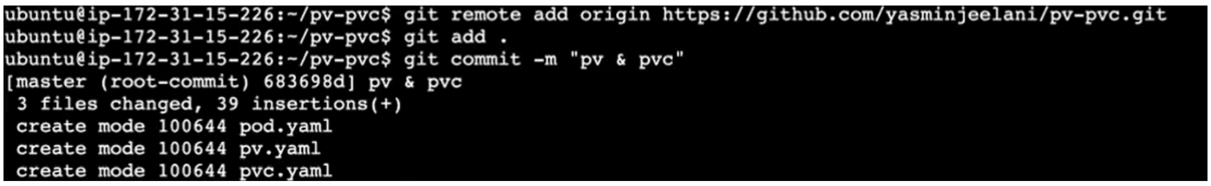
All code can put it in github:



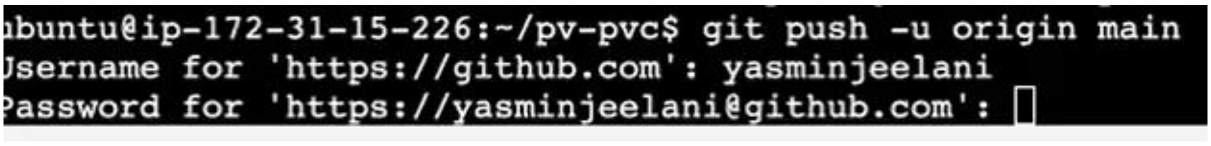
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ ls
pod.yaml  pv.yaml  pvc.yaml
ubuntu@ip-172-31-15-226:~/pv-pvc$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ubuntu/pv-pvc/.git/
ubuntu@ip-172-31-15-226:~/pv-pvc$
```



```
ubuntu@ip-172-31-15-226:~/pv-pvc$ git config --global user.name "yasminjeelani"
ubuntu@ip-172-31-15-226:~/pv-pvc$ git config --global user.email "jeelani.yasmin@gmail.com"
ubuntu@ip-172-31-15-226:~/pv-pvc$ git init
Reinitialized existing Git repository in /home/ubuntu/pv-pvc/.git/
```



```
ubuntu@ip-172-31-15-226:~/pv-pvc$ git remote add origin https://github.com/yasminjeelani/pv-pvc.git
ubuntu@ip-172-31-15-226:~/pv-pvc$ git add .
ubuntu@ip-172-31-15-226:~/pv-pvc$ git commit -m "pv & pvc"
[master (root-commit) 683698d] pv & pvc
 3 files changed, 39 insertions(+)
 create mode 100644 pod.yaml
 create mode 100644 pv.yaml
 create mode 100644 pvc.yaml
```

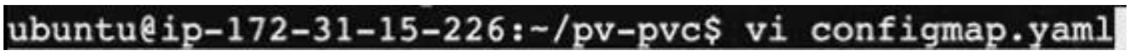


```
ıbuntu@ip-172-31-15-226:~/pv-pvc$ git push -u origin main
Jsername for 'https://github.com': yasminjeelani
Password for 'https://yasminjeelani@github.com':
```

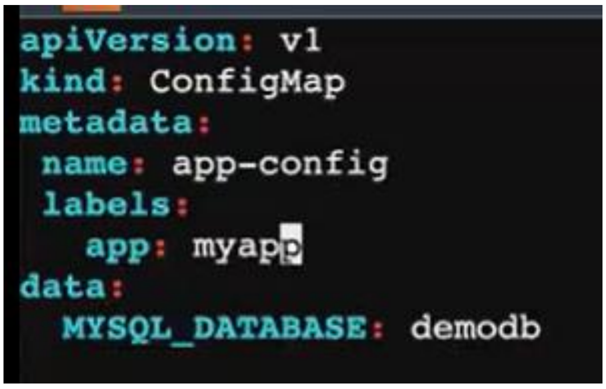
Config map:

- every application will have some kind of configuration data for it. Okay, if you're dealing with an SQL application. So, while launching a SQL application, you have to provide some variables like username password.
- you have to configure before setting up the application, you have to configure the username and password.
- So, by the start of the application, you have to provide certain data. Those data are called as configuration data.
- Okay, so the data that you provide in the initial stage of starting the application, we call those data as a configuration data.

- using config map, what we are doing, we are separating this configuration data from our application, your application and the configuration data is separated. So, you have your application running inside your container.
- Now, the configuration data will be maintained separately, using config map. So, whenever it is required. You can attach this configuration to your application.
- what we will be doing we will be creating an object called config map to hold non sensitive. So, which is not very crucial. So even if it is exposed to the user, it's of so not a big deal.
- So, what we do, we will be holding this config map will be holding non sensitive configuration data of your application.
- So, you're like separating your container. Where your application is running from the configuration details. Okay? So, whenever you want to change your configuration, you can only change the config map. You can directly go and alter or modify your config map. Object rather than going and changing the container itself. Container data itself. Okay? So, I will be creating the config map. I will create it in the same.

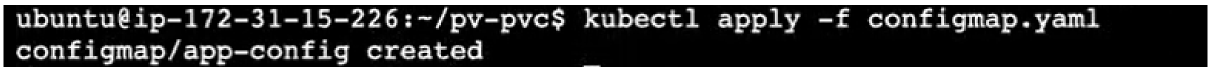


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ vi configmap.yaml
```



```
apiVersion: v1
kind: ConfigMap
metadata:
 name: app-config
 labels:
   app: myapp
data:
  MYSQL_DATABASE: demodb
```

Write all the files then execute files.



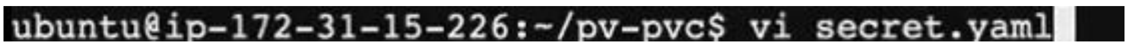
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl apply -f configmap.yaml
configmap/app-config created
```

- You can use this config map in your deployment. Okay? So, while creating your deployment

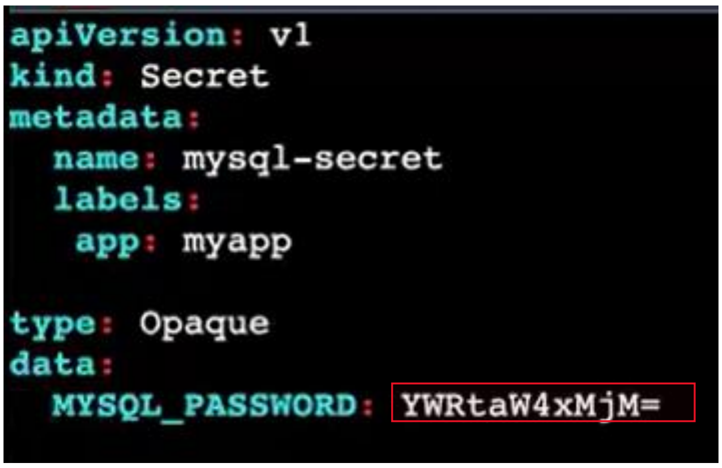
- If I've used the configuration of my SQL. So, suppose you're creating the deployment of my SQL application. In in that deployment, we will be passing this configuration details.

secrets

- secrets are similar to your config map.
- But in secret, if you are maintaining some sensitive data like your password or the username. You can put it inside your secret.
- So, what will happen? Your Kubernetes will encode the data. Everything will be encrypted if you place some data inside the secret, the data will be encoded.
- **our environmental variable or the configuration data both are same.**

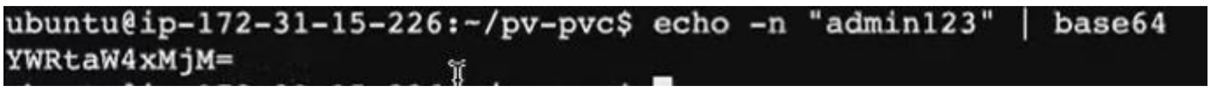


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ vi secret.yaml
```

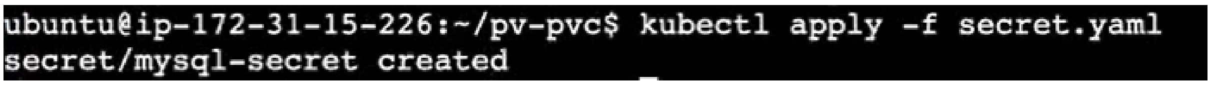


```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
  labels:
   app: myapp

type: Opaque
data:
  MYSQL_PASSWORD: YWRtaW4xMjM=
```

This is encrypted password. Get this by using the below command



```
ubuntu@ip-172-31-15-226:~/pv-pvc$ echo -n "admin123" | base64
YWRtaW4xMjM=
```



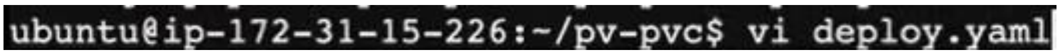
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl apply -f secret.yaml
secret/mysql-secret created
```

- you can give the password. So, before giving the password.
- Let me go in encrypt the password. So, as I told you, your secret can store the value in an encrypted manner here.
- **we will encrypt it using base 64 algorithm.**

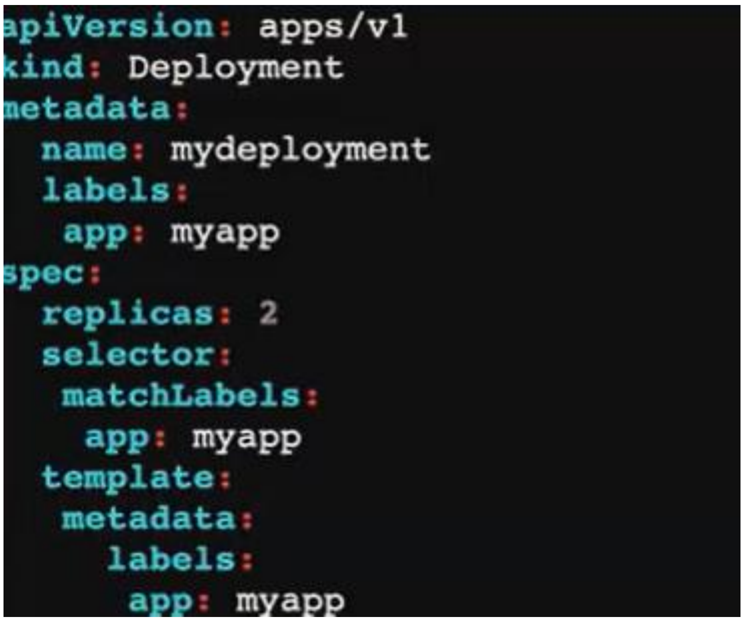
- So, I want to encrypt this password. So you can give echo – “admin123” | base 64 algorithm.
- So, using which algorithm, I want to encrypt it. Using the base 64 algorithm, I want to encrypt my string encrypt. Okay, so we are going to import the string.

deployment file for launching my SQL Application

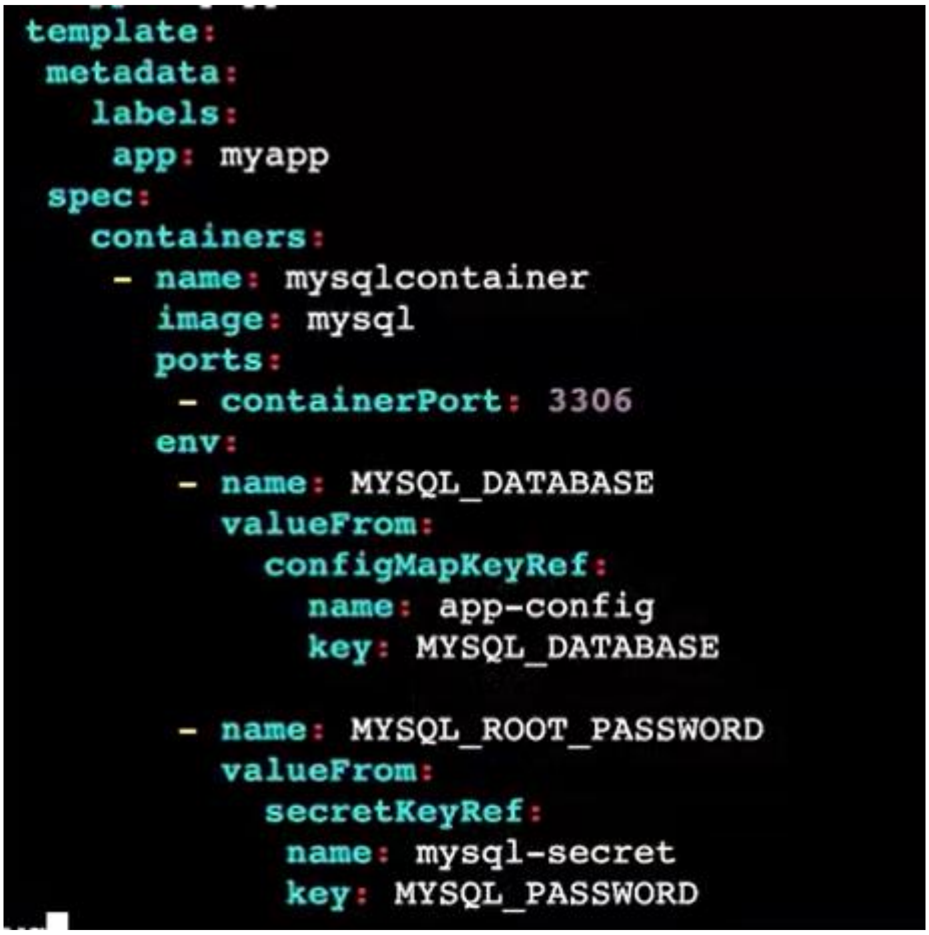
We will create a deployment file for launching a my SQL Application. So by launching your Mysql application, your application requires some details. Some environmental detail has to be passed like your username password database. So those values we will be Instead of hard coding those values inside your deployment file, we will be passing it via config map and secrets.



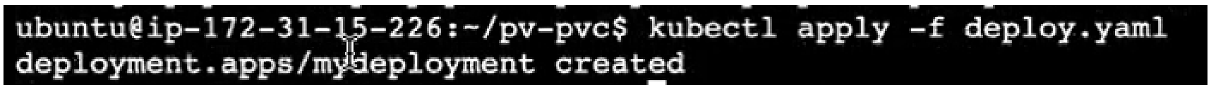
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ vi deploy.yaml
```



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeployment
  labels:
   app: myapp
spec:
  replicas: 2
  selector:
   matchLabels:
     app: myapp
  template:
   metadata:
     labels:
      app: myapp
```

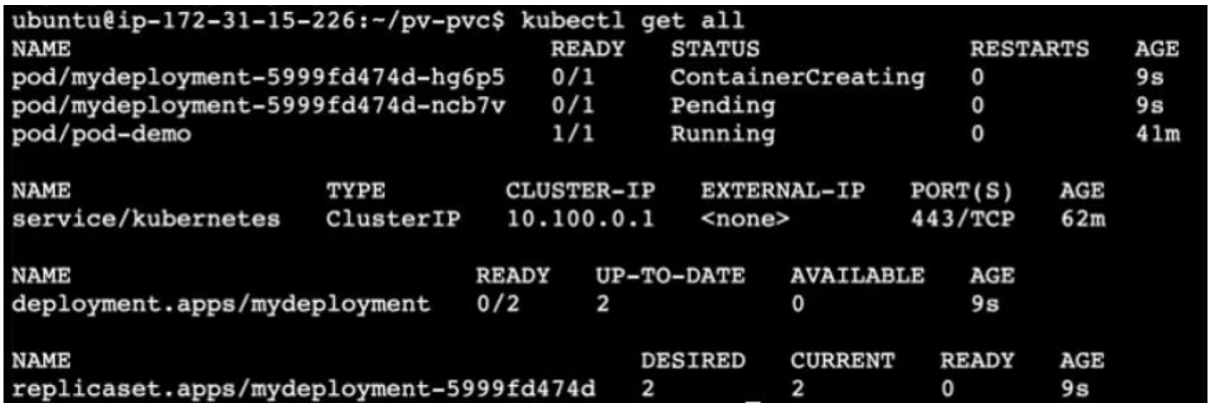


```
template:
 metadata:
   labels:
     app: myapp
 spec:
   containers:
     - name: mysqlcontainer
       image: mysql
       ports:
        - containerPort: 3306
       env:
        - name: MYSQL_DATABASE
          valueFrom:
            configMapKeyRef:
              name: app-config
              key: MYSQL_DATABASE

        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
             name: mysql-secret
             key: MYSQL_PASSWORD
```

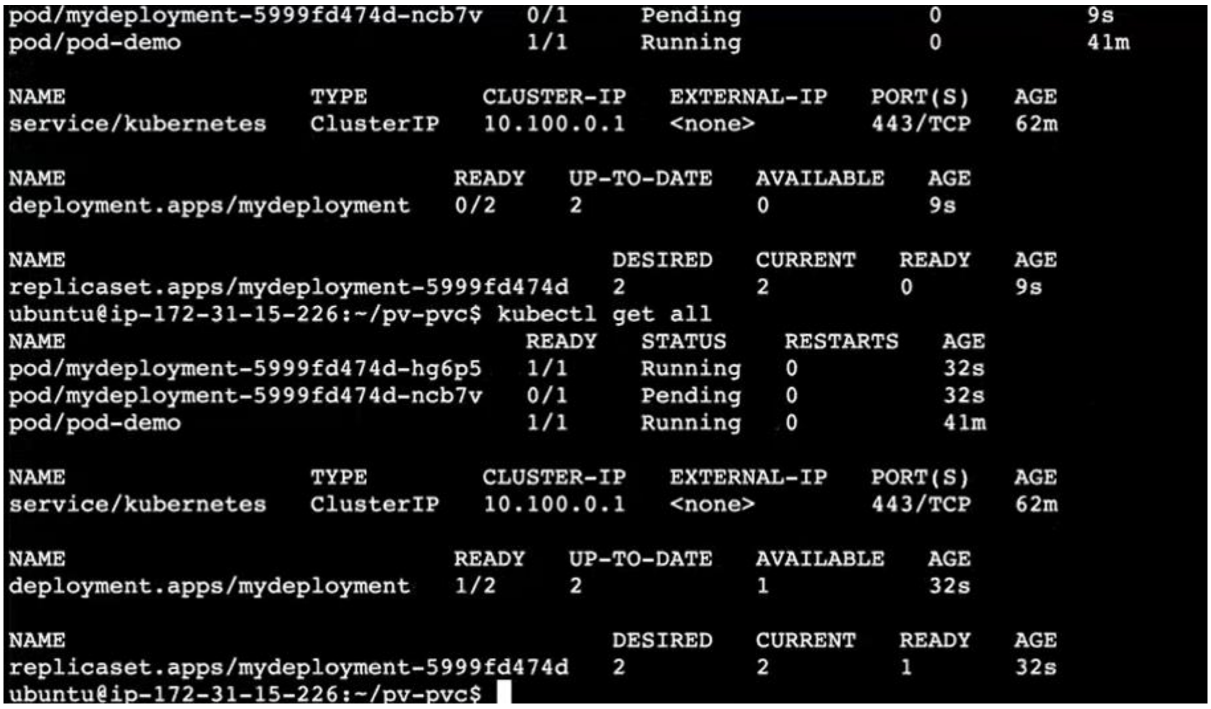


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl apply -f deploy.yaml
deployment.apps/mydeployment created
```

Kubectl get all



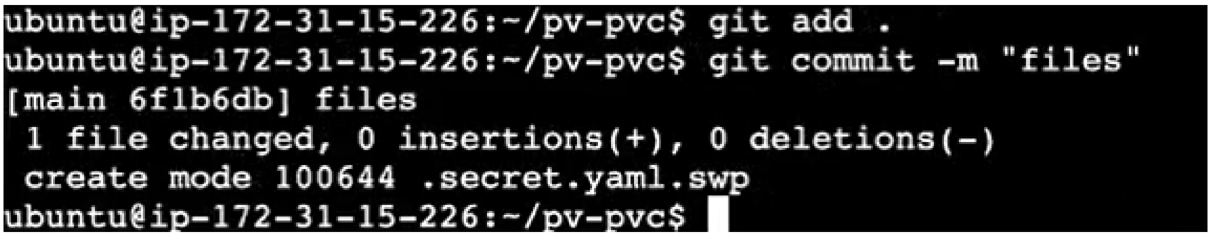
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl get all
NAME                                READY   STATUS              RESTARTS   AGE
pod/mydeployment-5999fd474d-hg6p5   0/1     ContainerCreating   0          9s
pod/mydeployment-5999fd474d-ncb7v   0/1     Pending             0          9s
pod/pod-demo                        1/1     Running             0          41m

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   62m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeployment   0/2     2            0           9s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeployment-5999fd474d   2         2         0       9s
```



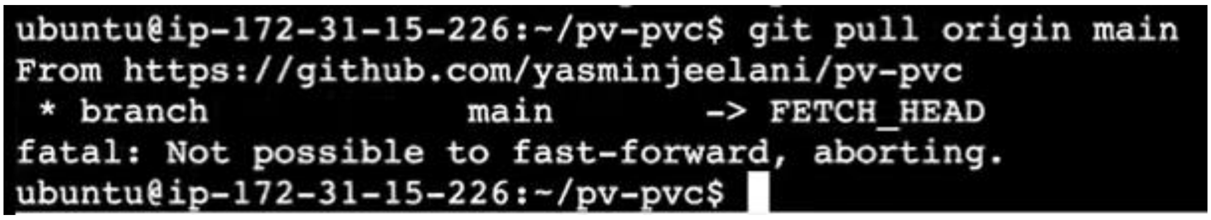
```
pod/mydeployment-5999fd474d-ncb7v    0/1     Pending              0           9s
pod/pod-demo                         1/1     Running              0           41m

NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes    ClusterIP   10.100.0.1    <none>        443/TCP   62m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeployment   0/2     2            0           9s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeployment-5999fd474d   2         2         0       9s
ubuntu@ip-172-31-15-226:~/pv-pvc$ kubectl get all
NAME                                 READY   STATUS    RESTARTS   AGE
pod/mydeployment-5999fd474d-hg6p5    1/1     Running   0          32s
pod/mydeployment-5999fd474d-ncb7v    0/1     Pending   0          32s
pod/pod-demo                         1/1     Running   0          41m

NAME                  TYPE        CLUSTER-IP    EXTERNAL-IP   PORT(S)   AGE
service/kubernetes    ClusterIP   10.100.0.1    <none>        443/TCP   62m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mydeployment   1/2     2            1           32s

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/mydeployment-5999fd474d   2         2         1       32s
ubuntu@ip-172-31-15-226:~/pv-pvc$
```

How to push all codes into github

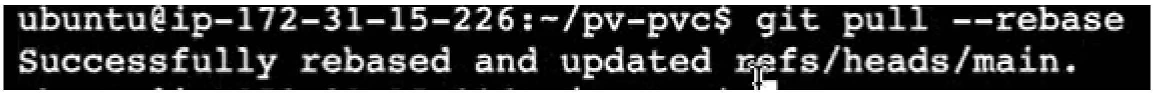
https://github.com/yasminjeelani/pv-pvc



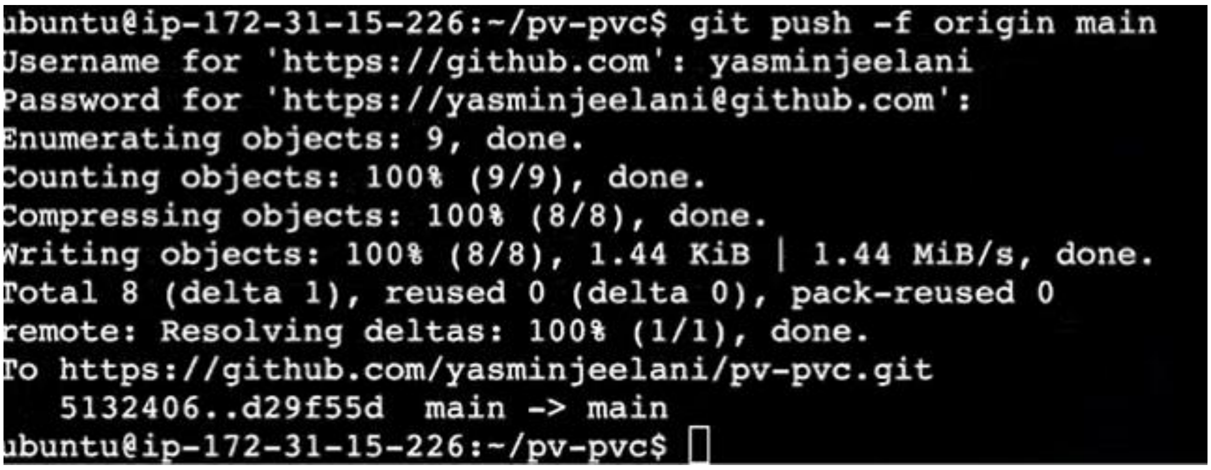
```
ubuntu@ip-172-31-15-226:~/pv-pvc$ git add .
ubuntu@ip-172-31-15-226:~/pv-pvc$ git commit -m "files"
[main 6f1b6db] files
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 .secret.yaml.swp
ubuntu@ip-172-31-15-226:~/pv-pvc$
```



```
ubuntu@ip-172-31-15-226:~/pv-pvc$ git pull origin main
From https://github.com/yasminjeelani/pv-pvc
 * branch            main       -> FETCH_HEAD
fatal: Not possible to fast-forward, aborting.
ubuntu@ip-172-31-15-226:~/pv-pvc$
```

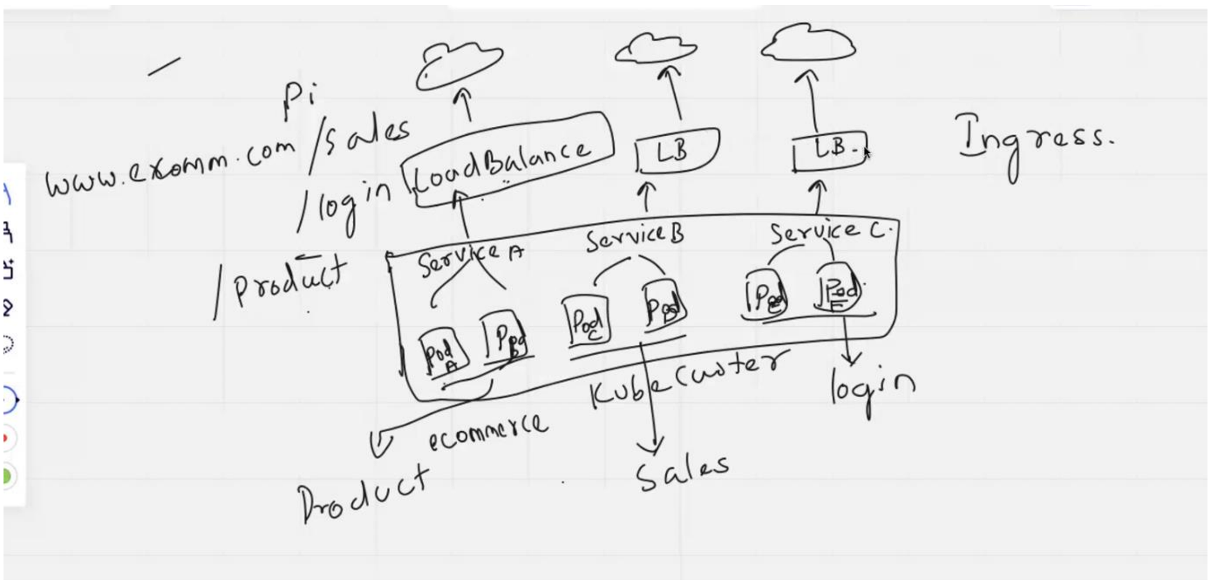


```
ubuntu@ip-172-31-15-226:~/pv-pvc$ git pull --rebase
Successfully rebased and updated refs/heads/main.
```



```
ubuntu@ip-172-31-15-226:~/pv-pvc$ git push -f origin main
Username for 'https://github.com': yasminjeelani
Password for 'https://yasminjeelani@github.com':
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (8/8), 1.44 KiB | 1.44 MiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/yasminjeelani/pv-pvc.git
   5132406..d29f55d  main -> main
ubuntu@ip-172-31-15-226:~/pv-pvc$ 
```

Ingress



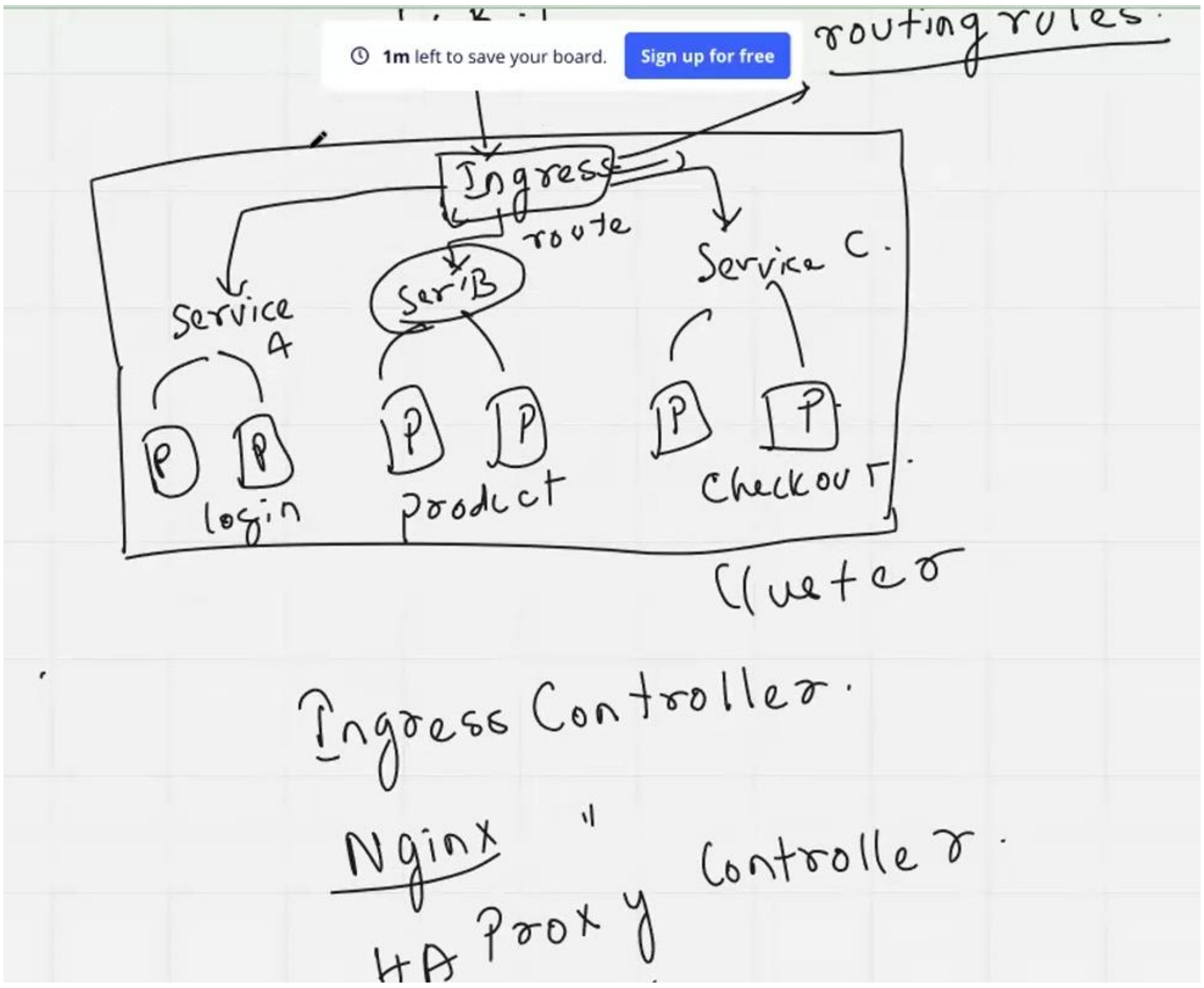
- www.ecommerce.com website. they have 3 pages.
    1. Sales
    2. Login
    3. Product
- Each page has different services (A, B, C). in Kubernetes cluster running pod .in each pod running different services.
- Each services create load balancer.
- In real time, 100's of load balancer created. AWS charged for this.
- Ingress can remove the load balancer usage of multiple load balancer.
- So using ingress what you can do, you can only use one load balancer, you can have multiple services for your application for accessing all the service.

- You don't need different load Balancer. If you have 10 service, you have to create 10 load balancers. To avoid that you can, you can use ingress.
- So how the ingress will be so we will be creating ingress. So, ingress is one of the Kubernetes services. So, it is also a Kubernetes service, only that will that will be inside your Kubernetes cluster. So, I have my cluster inside my cluster. I have my pods.
- So, every pod is hosting a service. So, every port will be connected to the service.
- we connected to a service A service B and service C.
- So, my login page my product page, everything is a different service and checkout page. Everything is a different service which is running inside your pod. Okay, each service is running inside your pod.
- Now, every pod is connected to a service file and with the help of service object, you'll be able to communicate with the Internet. So there what you have, you have a load balancer.
- So load balance. And now what you're going to do my user ? No, he will only configure one load balancer.
- So using this load balancer, using single load balance, how you can connect with all the services we will be using a tool called ingress.
- In E commerce.com, he wants to access the product page. So what it will hit the load Balancer. Your traffic will hit the load Balancer. load Balancer will route the traffic to your ingress. So now I want to go to this ingress.
- **this ingress like a intelligent load balancer.**
- It will check the Dns, or it will check the URL. So now I want to route my traffic. Your load balancer will not know where to route the traffic, whether to service a service B or service C.
- now, but your ingress will exactly know where to route the traffic. Now I want to. This is my URL. I want to route the traffic to the product service. So where is my product service it is running in the service B.
- so my ingress will route the traffic to my product page application or the product service.
- so ingress will take care of routing the traffic based on the path.
- ingress can do
    1. path-based routing
    2. host-based routing

- So we mostly, we will only go with the path based routing so based on the specified path. Your ingress can identify the application, and it will route the traffic to the respective service.
- So here in between, how does your ingress will know based? How does your ingress will check all these things? We will be right.We'll be creating something called this routing rules.
- So you we'll be creating ingress object, and while creating the ingress object, We will be creating, or we will be attaching the routing rules based on the routing rules. Your ingress will route around the traffic to the respective services.
- you using ingress you will be writing.
- You'll be using ingress as I told you, you'll only be using a load balancer.
- your load Balancer will commonly route the traffic, so your load balancer will not know whether to route the traffic to the product page or product service or the checkout service. It will route the traffic to the ingress will check the DNS or the URL endpoint.
- It will check the endpoint based on the routing rule, it will route the traffic to the particular service.
- So we will be writing rules in the ingress file. You have to mention the rules inside the ingress file, based on what path you have to route the traffic to clear.

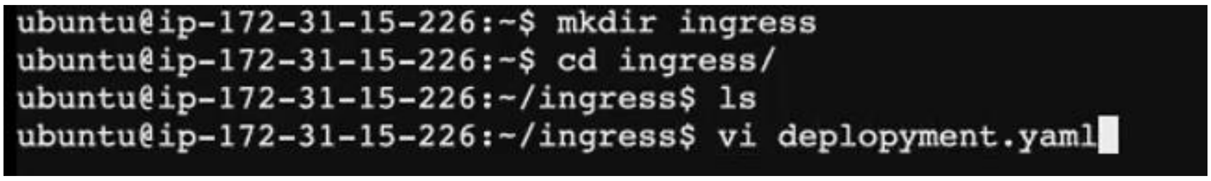
INGRESS CONTROLLER

- So in between. So yeah, ingress cannot run. Say independently. So we will be installing something called as ingress controller.
- with the help of Ingress Controller, the traffic will be routed.
- so here the Ingress Controller, we don't have a dedicated ingress Controller for Kubernetes .
- It will be using the 3rd party ingress controller. So we use inginx ingress controller, H proxy, ingress controller
    1. Nginx ingress controller.
    2. HA proxy ingress controller.
    3. Traffic ingress controller.
- for web-based application. We use Nginx ingress controller/ HA proxy ingress controller. Don't get confused with your nginx web server that you were using that is different and this is different.

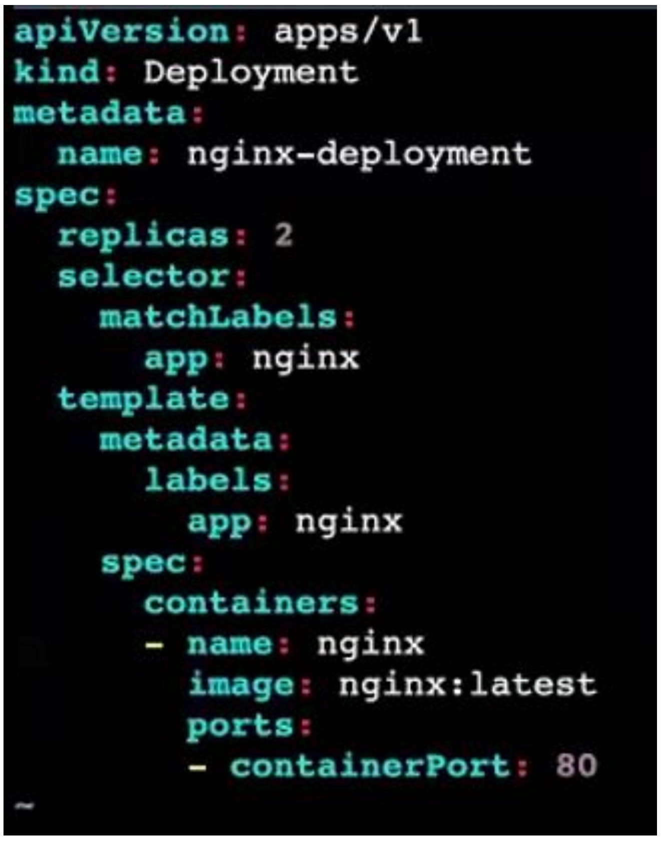


- so we will have to install this nginx ingress Controller. After installing this ingress controller, you have to create the ingress object.
- while creating the ingress object. You'll be giving all the routing rules. You'll specify all the routing rules create while creating the ingress object. After creating these 2, we will be attaching it to our services.
- Further, it gets attached to your deployment.
- So how do you create your ingress object? So obviously in Kubernetes? Every object we create use a yaml file to create the object in your Kubernetes cluster same way. Here we will be writing ingress, dot Yaml file. We'll be writing an ingress manifest file.
- Everything mentioned in this manifest file.

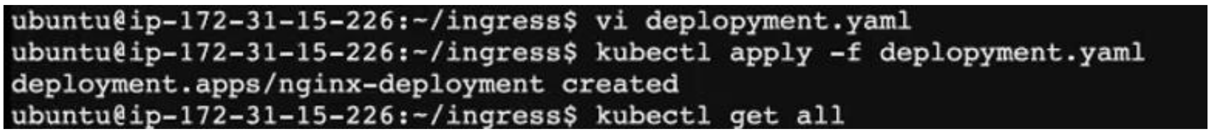
Create deployment file



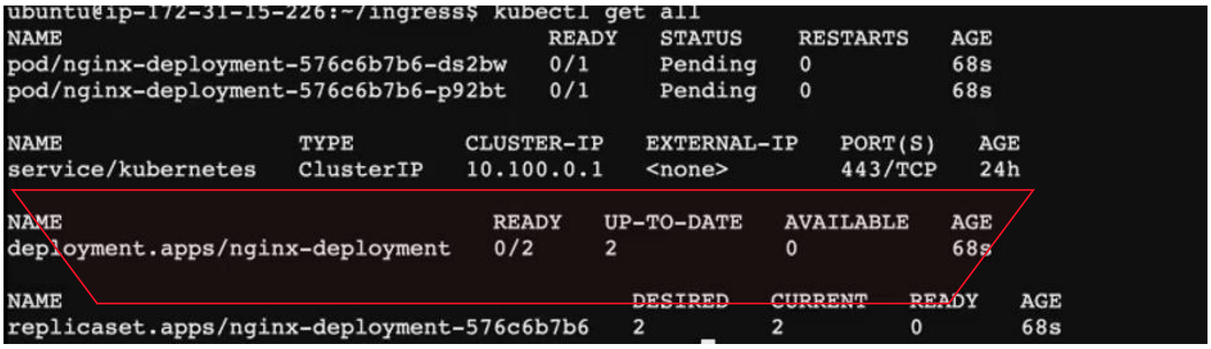
```
ubuntu@ip-172-31-15-226:~$ mkdir ingress
ubuntu@ip-172-31-15-226:~$ cd ingress/
ubuntu@ip-172-31-15-226:~/ingress$ ls
ubuntu@ip-172-31-15-226:~/ingress$ vi deploypment.yaml
```



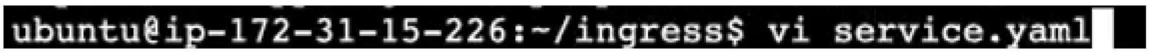
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
~
```



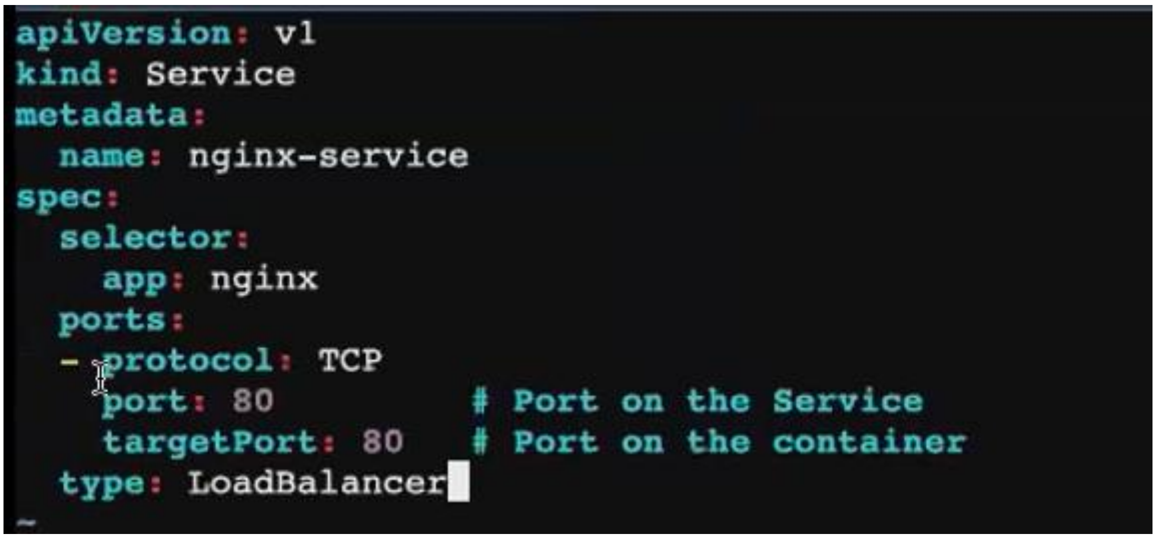
```
ubuntu@ip-172-31-15-226:~/ingress$ vi deployment.yaml
ubuntu@ip-172-31-15-226:~/ingress$ kubectl apply -f deploypment.yaml
deployment.apps/nginx-deployment created
ubuntu@ip-172-31-15-226:~/ingress$ kubectl get all
```



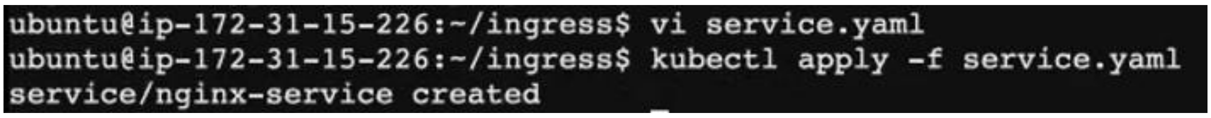
```
ubuntu@ip-172-31-15-226:~/ingress$ kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-576c6b7b6-ds2bw    0/1     Pending   0          68s
pod/nginx-deployment-576c6b7b6-p92bt    0/1     Pending   0          68s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   24h

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   0/2     2            0           68s

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-576c6b7b6    2         2         0       68s
```



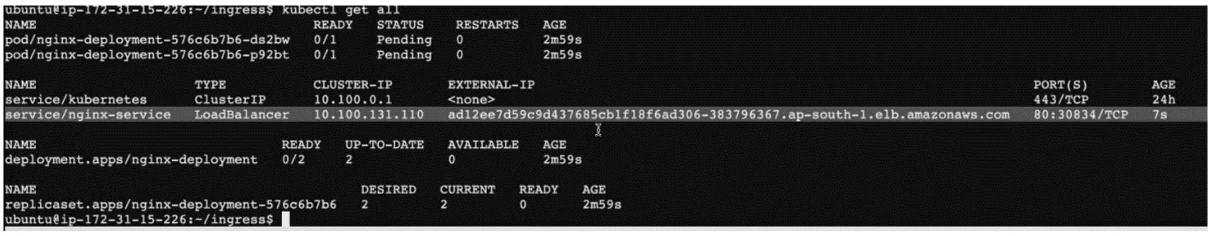
```
ubuntu@ip-172-31-15-226:~/ingress$ vi service.yaml
```



```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80          # Port on the Service
    targetPort: 80    # Port on the container
  type: LoadBalancer
~
```

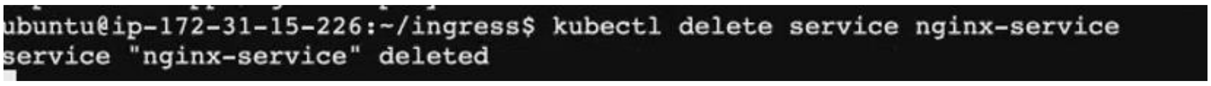


```
ubuntu@ip-172-31-15-226:~/ingress$ vi service.yaml
ubuntu@ip-172-31-15-226:~/ingress$ kubectl apply -f service.yaml
service/nginx-service created
```



```
ubuntu@ip-172-31-15-226:~/ingress$ kubectl get all
NAME                                   READY   STATUS    RESTARTS   AGE
pod/nginx-deployment-576c6b7b6-ds2bw   0/1     Pending   0          2m59s
pod/nginx-deployment-576c6b7b6-p92bt   0/1     Pending   0          2m59s

NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP                                                                PORT(S)        AGE
service/kubernetes      ClusterIP      10.100.0.1       <none>                                                                     443/TCP        24h
service/nginx-service   LoadBalancer   10.100.131.110   ad12ee7d59c9d437685cb1f18f6ad306-383796367.ap-south-1.elb.amazonaws.com   80:30834/TCP   7s

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-deployment   0/2     2            0           2m59s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-deployment-576c6b7b6   2         2         0       2m59s
ubuntu@ip-172-31-15-226:~/ingress$
```

we will delete the service file.I don't want this. I'm going to use my Ingress Controller. I'm going to create a service without the load balancer and via Ingress Controller. I will be accessing my application through load balancer.



```
ubuntu@ip-172-31-15-226:~/ingress$ kubectl delete service nginx-service
service "nginx-service" deleted
```

Some topics bending

Day 1 - Monitoring Tool -Prometheus & Grafana

HELM

- Helm is a package management tool for your Kubernetes cluster
- package management tool for your Kubernetes cluster.
- In ubuntu package management tool is apt, in Linux package management tool is yum.
- if I had to install any application, if I had to install any tools or services inside my Kubernetes cluster. You can go do it with the help of Helm.
- So, helm is package management tool. It acts as a package management tool mainly to install tools or software's inside the Kubernetes cluster.

HELM CHARTS

- Helm charts are mainly used for complex application.
- If you have a huge application which has got hundreds of micro services and becomes tedious for you to manage or create deployment files manually service files manually. If you're if you're having 100 services, you have to create 100 deployment files. Okay, in the real time, it takes a lot of time.
- you have to manage every single deployment file manually.
- Instead of doing that, you can use helm charts.
- So, helm charts are mainly used for managing complex application for deploying complex applications.
- When you create helm chart, will by default, it will give you a template for deployment service ingress.
- Okay, all the basic objects will be created by your helm charts itself.
- Okay, you just have to do certain modifications here and there in your helm chart.
- So you can use helm can be used separately.
- You can use helm for installing tools or packages inside your kubernetes cluster, and you have helm charts.
- helm charts are like pre-configured Kubernetes Resources.
- Inter ques[uses of helm charts] **mainly for automating the deployment process managing your deployment or objects inside the Kubernetes cluster.**

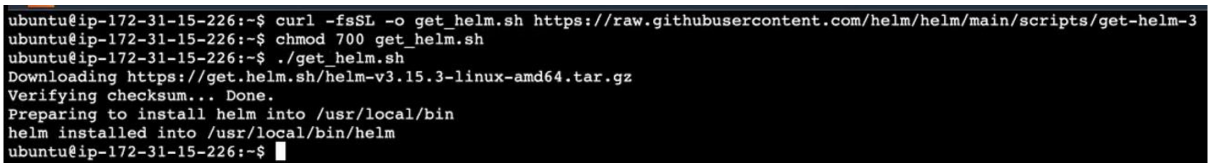
Install helm

https://helm.sh/docs/intro/install/ [installation document for helm]

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



```
ubuntu@ip-172-31-15-226:~$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
ubuntu@ip-172-31-15-226:~$ chmod 700 get_helm.sh
ubuntu@ip-172-31-15-226:~$ ./get_helm.sh
Downloading https://get.helm.sh/helm-v3.15.3-linux-amd64.tar.gz
Verifying checksum... Done.
Preparing to install helm into /usr/local/bin
helm installed into /usr/local/bin/helm
ubuntu@ip-172-31-15-226:~$
```

Command explanation:

- use the curl command what we do, we download the files from the Internet.
- So, I'm downloading the files from this website. Okay, I'm just downloading the files from this website using my curl. Come, and I'm downloading it. After downloading it. I'm giving the permission 700 so that I can execute open to user can execute this. This is a script file. I have to execute this file to install helm.
- So, I change the access. I change the mode I give executable permission, and then I'm executing it. After executing it, you will be able to download the helm in your local machine.

how to create helm charts

some topics bending