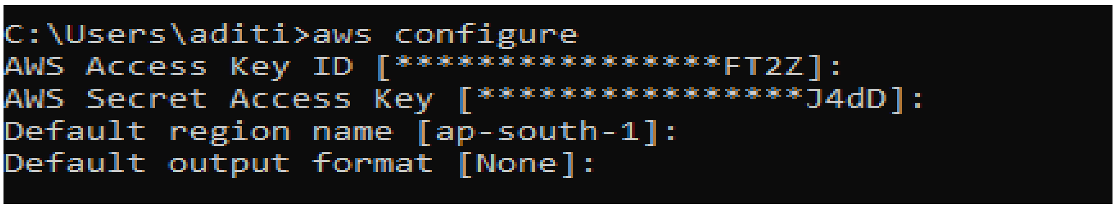
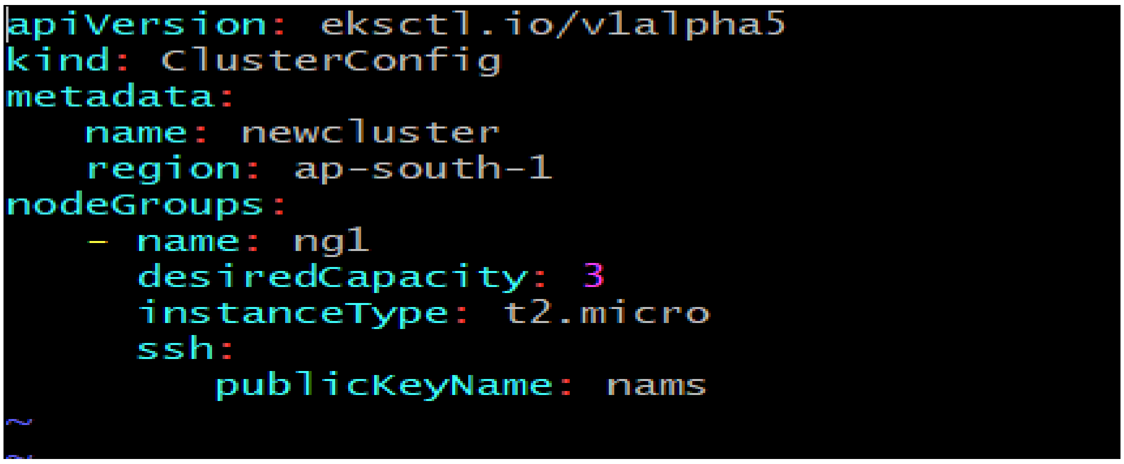
WORDPRESS & MYSQL SERVER SETUP ON EKS

1. **Download aws-cli v2 & eksctl in your system**
2. **Create an IAM user in AWS account, with AdministratorAccess power**
3. **Configure aws**



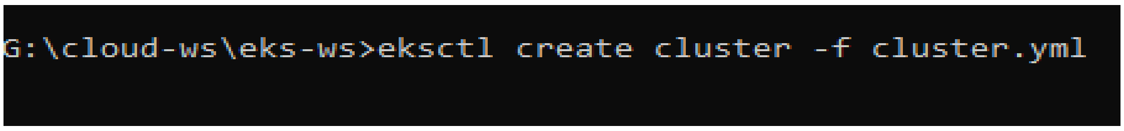
```
C:\Users\aditi>aws configure
AWS Access Key ID [****************FT2Z]:
AWS Secret Access Key [****************J4dD]:
Default region name [ap-south-1]:
Default output format [None]:
```

4. **Create *cluster.yml* file**

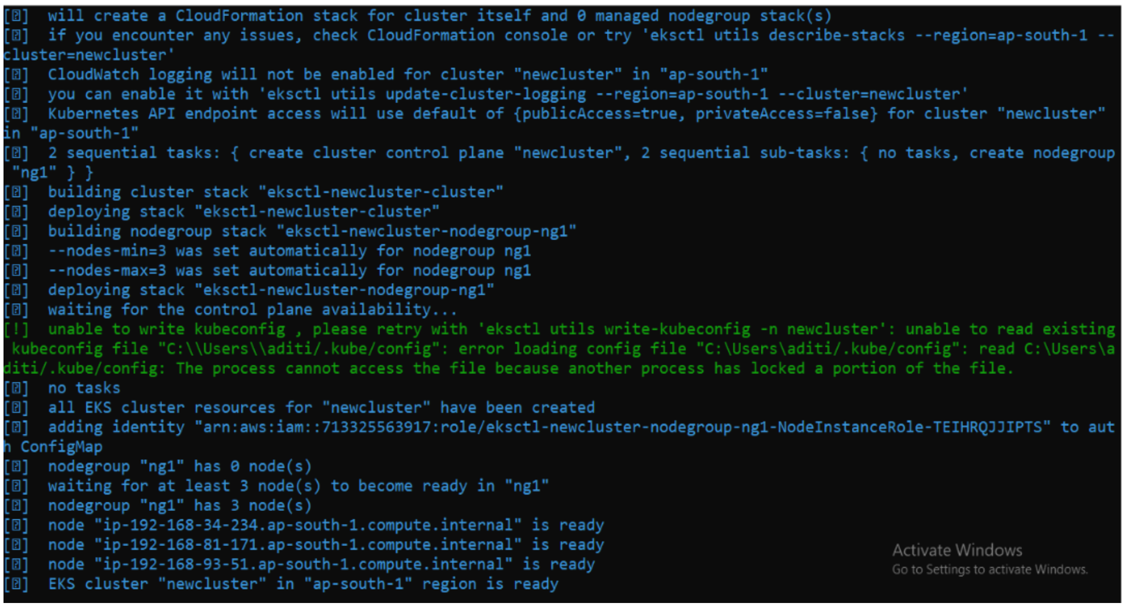


```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
   name: newcluster
   region: ap-south-1
nodeGroups:
   - name: ng1
     desiredCapacity: 3
     instanceType: t2.micro
     ssh:
        publicKeyName: nams
~
~
```

5. **Run *cluster.yml* for creating Kubernetes cluster on EKS**

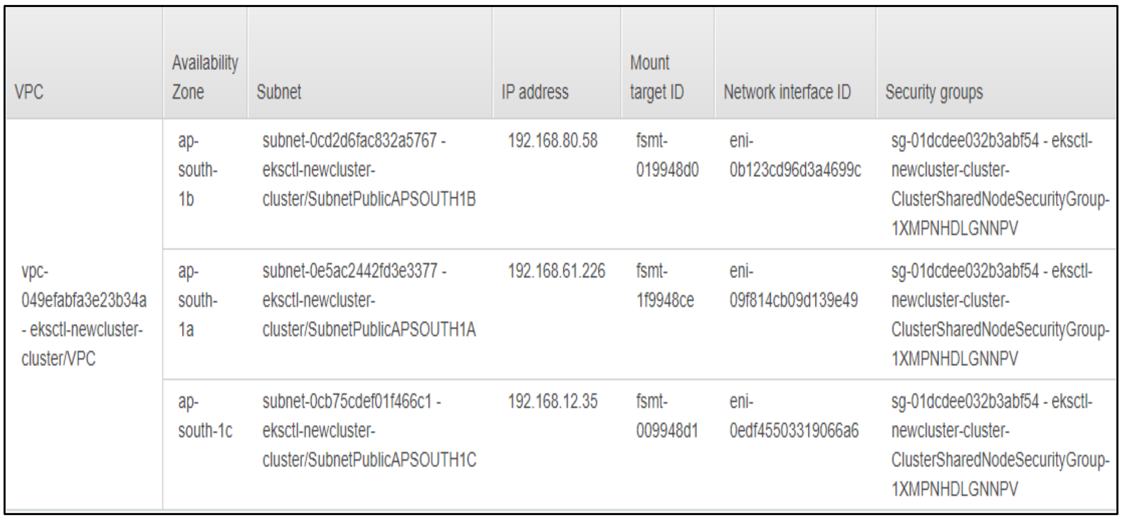


```
G:\cloud-ws\eks-ws>eksctl create cluster -f cluster.yml
```



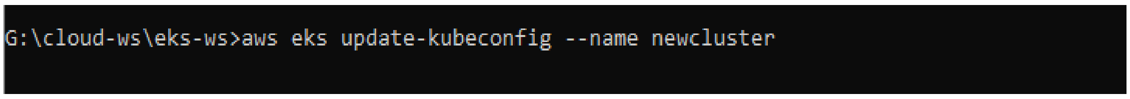
```
[⍰]  will create a CloudFormation stack for cluster itself and 0 managed nodegroup stack(s)
[⍰]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south-1 --
cluster=newcluster'
[⍰]  CloudWatch logging will not be enabled for cluster "newcluster" in "ap-south-1"
[⍰]  you can enable it with 'eksctl utils update-cluster-logging --region=ap-south-1 --cluster=newcluster'
[⍰]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "newcluster"
in "ap-south-1"
[⍰]  2 sequential tasks: { create cluster control plane "newcluster", 2 sequential sub-tasks: { no tasks, create nodegroup
 "ng1" } }
[⍰]  building cluster stack "eksctl-newcluster-cluster"
[⍰]  deploying stack "eksctl-newcluster-cluster"
[⍰]  building nodegroup stack "eksctl-newcluster-nodegroup-ng1"
[⍰]  --nodes-min=3 was set automatically for nodegroup ng1
[⍰]  --nodes-max=3 was set automatically for nodegroup ng1
[⍰]  deploying stack "eksctl-newcluster-nodegroup-ng1"
[⍰]  waiting for the control plane availability...
[!]  unable to write kubeconfig , please retry with 'eksctl utils write-kubeconfig -n newcluster': unable to read existing
 kubeconfig file "C:\\Users\\aditi/.kube/config": error loading config file "C:\Users\aditi/.kube/config": read C:\Users\a
diti/.kube/config: The process cannot access the file because another process has locked a portion of the file.
[⍰]  no tasks
[⍰]  all EKS cluster resources for "newcluster" have been created
[⍰]  adding identity "arn:aws:iam::713325563917:role/eksctl-newcluster-nodegroup-ng1-NodeInstanceRole-TEIHRQJJIPTS" to aut
h ConfigMap
[⍰]  nodegroup "ng1" has 0 node(s)
[⍰]  waiting for at least 3 node(s) to become ready in "ng1"
[⍰]  nodegroup "ng1" has 3 node(s)
[⍰]  node "ip-192-168-34-234.ap-south-1.compute.internal" is ready
[⍰]  node "ip-192-168-81-171.ap-south-1.compute.internal" is ready
[⍰]  node "ip-192-168-93-51.ap-south-1.compute.internal" is ready
[⍰]  EKS cluster "newcluster" in "ap-south-1" region is ready
```

6. Create an EFS in AWS using EKS cluster vpc



VPC	Availability Zone	Subnet	IP address	Mount target ID	Network interface ID	Security groups
vpc-049efabfa3e23b34a - eksctl-newcluster-cluster/VPC	ap-south-1b	subnet-0cd2d6fac832a5767 - eksctl-newcluster-cluster/SubnetPublicAPSOUTH1B	192.168.80.58	fsmt-019948d0	eni-0b123cd96d3a4699c	sg-01dcdee032b3abf54 - eksctl-newcluster-cluster-ClusterSharedNodeSecurityGroup-1XMPNHDLGNNPV
	ap-south-1a	subnet-0e5ac2442fd3e3377 - eksctl-newcluster-cluster/SubnetPublicAPSOUTH1A	192.168.61.226	fsmt-1f9948ce	eni-09f814cb09d139e49	sg-01dcdee032b3abf54 - eksctl-newcluster-cluster-ClusterSharedNodeSecurityGroup-1XMPNHDLGNNPV
	ap-south-1c	subnet-0cb75cdef01f466c1 - eksctl-newcluster-cluster/SubnetPublicAPSOUTH1C	192.168.12.35	fsmt-009948d1	eni-0edf45503319066a6	sg-01dcdee032b3abf54 - eksctl-newcluster-cluster-ClusterSharedNodeSecurityGroup-1XMPNHDLGNNPV

7. Download *kubectl* in your system and update kube-config file with new EKS cluster

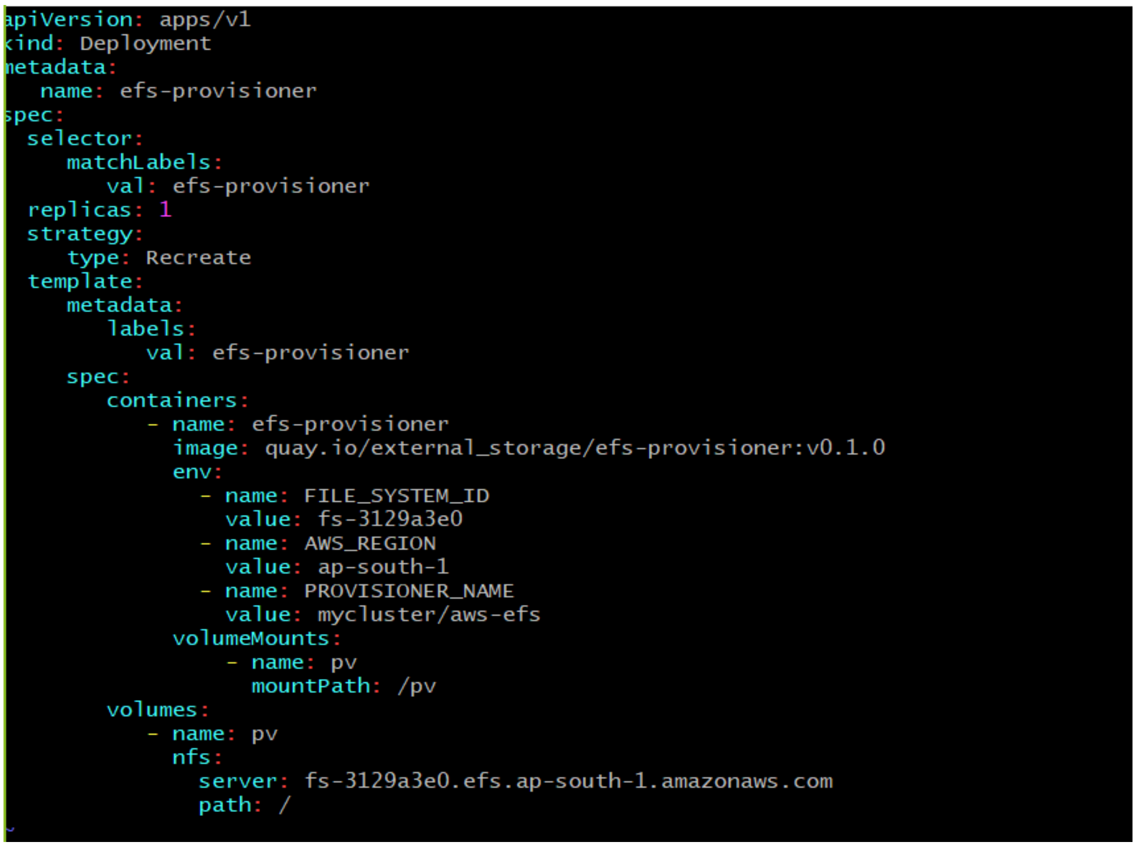


```
G:\cloud-ws\eks-ws>aws eks update-kubeconfig --name newcluster
```

8. Create provisioner file for EFS as *efs-provisioner.yml*

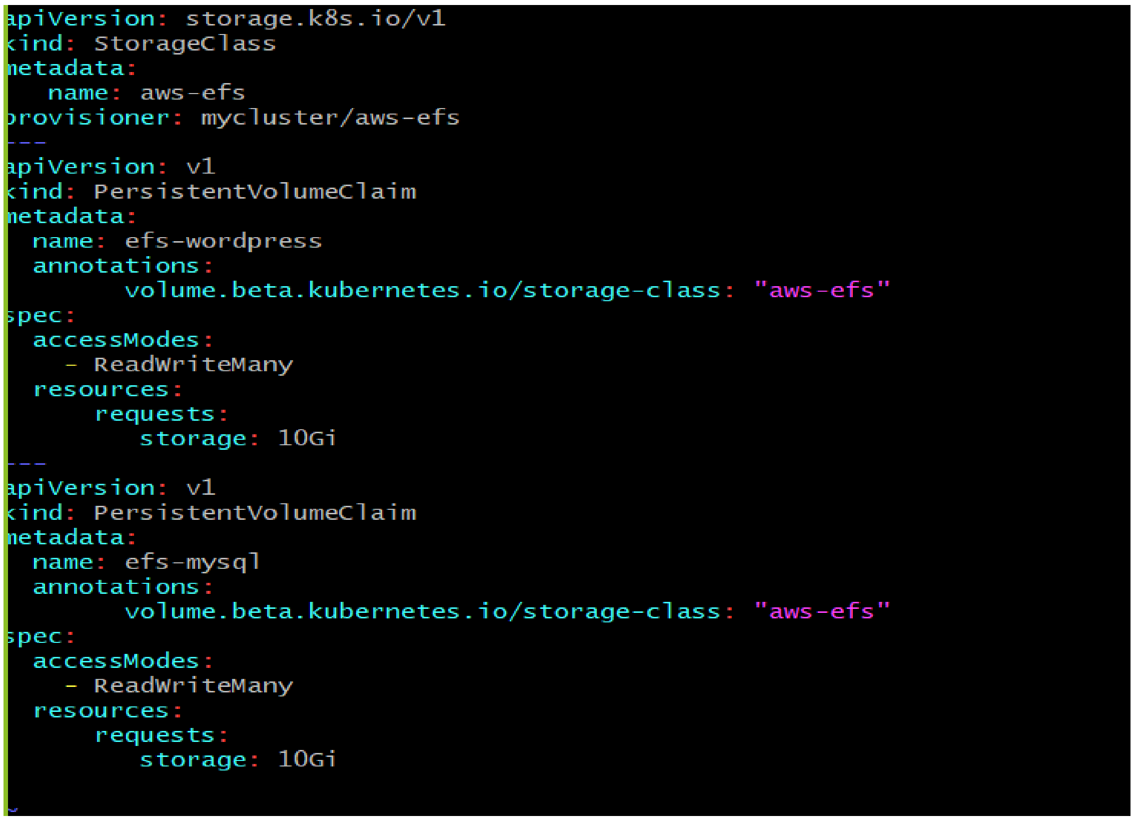
This file will create a deployment which will help us to get access of EFS

- change the variable “FILE_SYSTEM_ID” & “server”, according to your efs storage



```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: efs-provisioner
spec:
  selector:
     matchLabels:
        val: efs-provisioner
  replicas: 1
  strategy:
     type: Recreate
  template:
     metadata:
        labels:
           val: efs-provisioner
     spec:
        containers:
           - name: efs-provisioner
             image: quay.io/external_storage/efs-provisioner:v0.1.0
             env:
               - name: FILE_SYSTEM_ID
                 value: fs-3129a3e0
               - name: AWS_REGION
                 value: ap-south-1
               - name: PROVISIONER_NAME
                 value: mycluster/aws-efs
             volumeMounts:
                 - name: pv
                   mountPath: /pv
        volumes:
           - name: pv
             nfs:
               server: fs-3129a3e0.efs.ap-south-1.amazonaws.com
               path: /
```

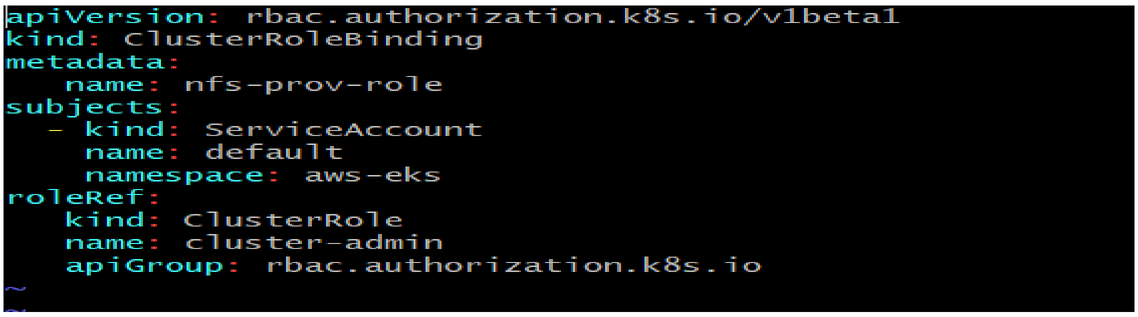
9. **Create "storage class", "pvc" using *sc.yml* for taking storage from EFS**



```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
   name: aws-efs
provisioner: mycluster/aws-efs
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: efs-wordpress
  annotations:
        volume.beta.kubernetes.io/storage-class: "aws-efs"
spec:
  accessModes:
    - ReadWriteMany
  resources:
       requests:
           storage: 10Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: efs-mysql
  annotations:
        volume.beta.kubernetes.io/storage-class: "aws-efs"
spec:
  accessModes:
    - ReadWriteMany
  resources:
       requests:
           storage: 10Gi
```

10. Create *rbac.yml*

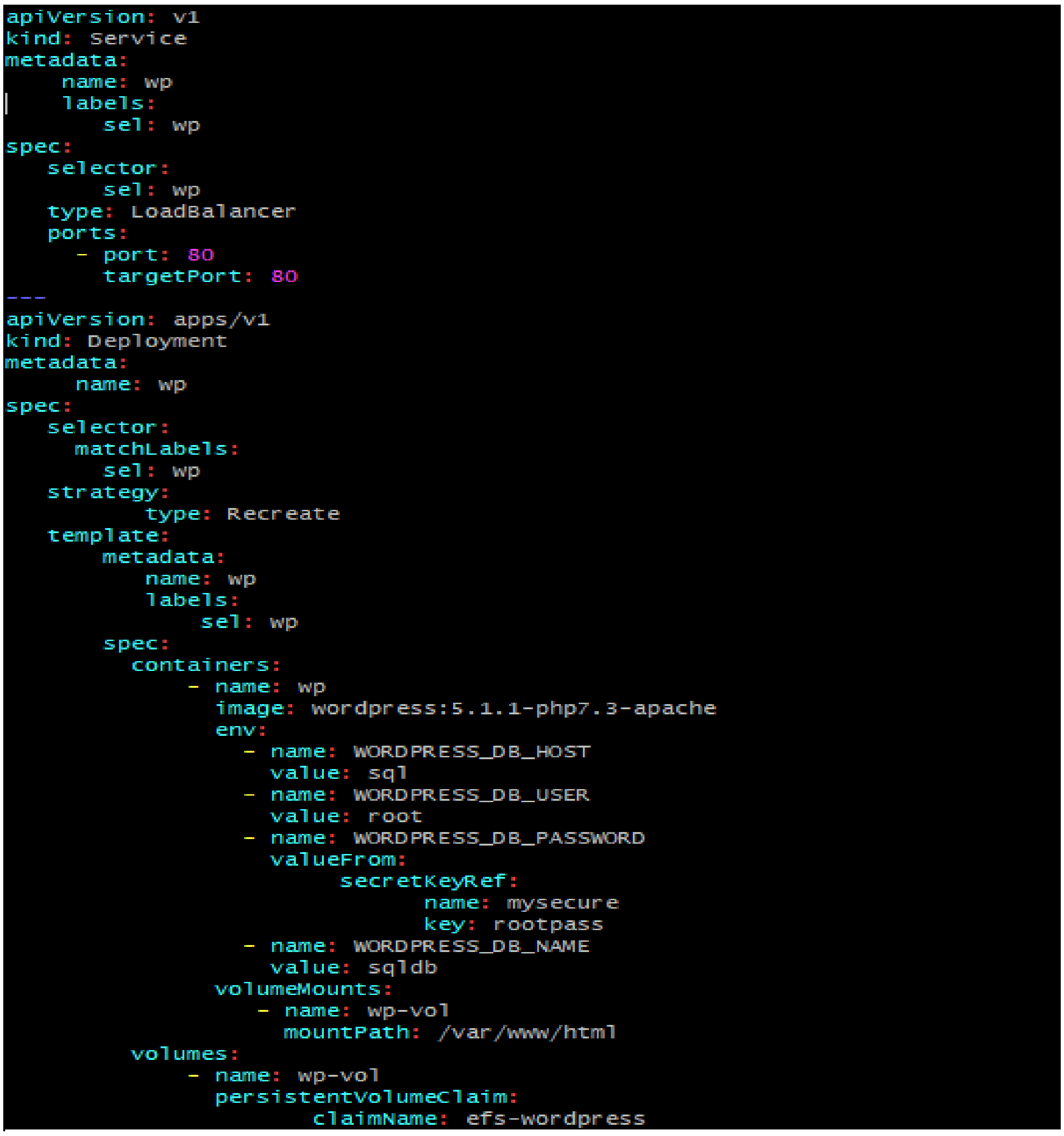
This file helps in security & access of cluster



```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
   name: nfs-prov-role
subjects:
  - kind: ServiceAccount
    name: default
    namespace: aws-eks
roleRef:
   kind: ClusterRole
   name: cluster-admin
   apiGroup: rbac.authorization.k8s.io
~
~
```

11. Create a file *wpsdeploy.yml*

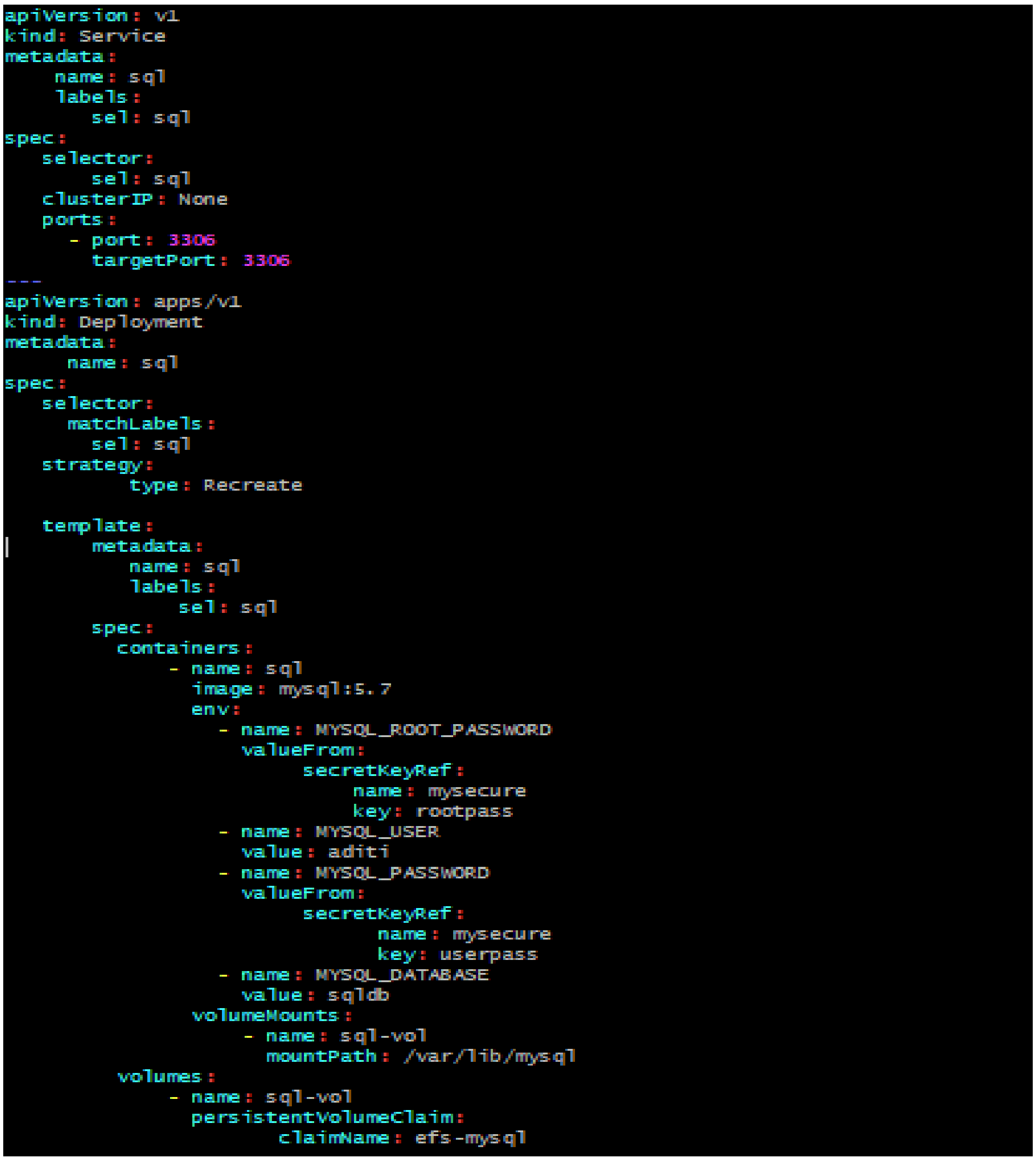
This file will create wordpress deployment with ELB of AWS



```
apiVersion: v1
kind: Service
metadata:
    name: wp
    labels:
       sel: wp
spec:
   selector:
       sel: wp
   type: LoadBalancer
   ports:
     - port: 80
       targetPort: 80
---
apiVersion: apps/v1
kind: Deployment
metadata:
     name: wp
spec:
   selector:
     matchLabels:
       sel: wp
   strategy:
          type: Recreate
   template:
       metadata:
          name: wp
          labels:
              sel: wp
       spec:
         containers:
             - name: wp
               image: wordpress:5.1.1-php7.3-apache
               env:
                 - name: WORDPRESS_DB_HOST
                   value: sql
                 - name: WORDPRESS_DB_USER
                   value: root
                 - name: WORDPRESS_DB_PASSWORD
                   valueFrom:
                        secretKeyRef:
                              name: mysecure
                              key: rootpass
                 - name: WORDPRESS_DB_NAME
                   value: sqldb
               volumeMounts:
                  - name: wp-vol
                    mountPath: /var/www/html
         volumes:
             - name: wp-vol
               persistentVolumeClaim:
                      claimName: efs-wordpress
```

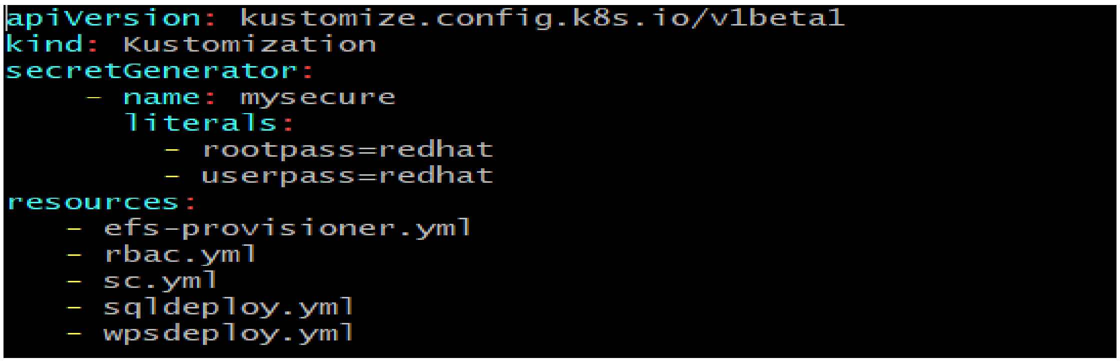
12. Create a file *sqldeploy.yml*

This file will create a mysql database deployment for wordpress server



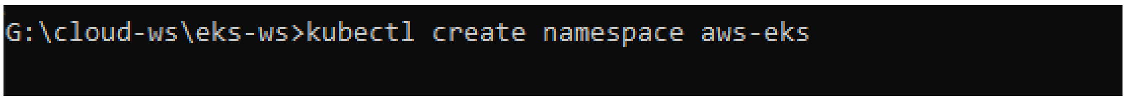
```
apiVersion: v1
kind: Service
metadata:
    name: sql
    labels:
       sel: sql
spec:
   selector:
       sel: sql
   clusterIP: None
   ports:
     - port: 3306
       targetPort: 3306
---
apiVersion: apps/v1
kind: Deployment
metadata:
     name: sql
spec:
   selector:
     matchLabels:
       sel: sql
   strategy:
          type: Recreate

   template:
       metadata:
          name: sql
          labels:
              sel: sql
       spec:
         containers:
             - name: sql
               image: mysql:5.7
               env:
                 - name: MYSQL_ROOT_PASSWORD
                   valueFrom:
                        secretKeyRef:
                            name: mysecure
                            key: rootpass
                 - name: MYSQL_USER
                   value: aditi
                 - name: MYSQL_PASSWORD
                   valueFrom:
                        secretKeyRef:
                              name: mysecure
                              key: userpass
                 - name: MYSQL_DATABASE
                   value: sqldb
               volumeMounts:
                   - name: sql-vol
                     mountPath: /var/lib/mysql
         volumes:
             - name: sql-vol
               persistentVolumeClaim:
                      claimName: efs-mysql
```

13. Create *kustomization.yml* for binding all the files created above



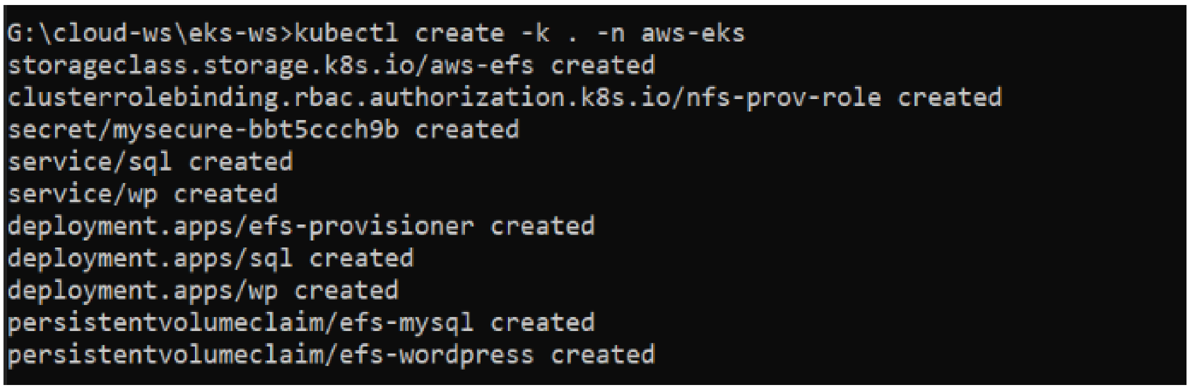
```
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization
secretGenerator:
    - name: mysecure
      literals:
        - rootpass=redhat
        - userpass=redhat
resources:
   - efs-provisioner.yml
   - rbac.yml
   - sc.yml
   - sqldeploy.yml
   - wpsdeploy.yml
```

14. Create a namespace *aws-eks* for deploying everything in a single umbrella



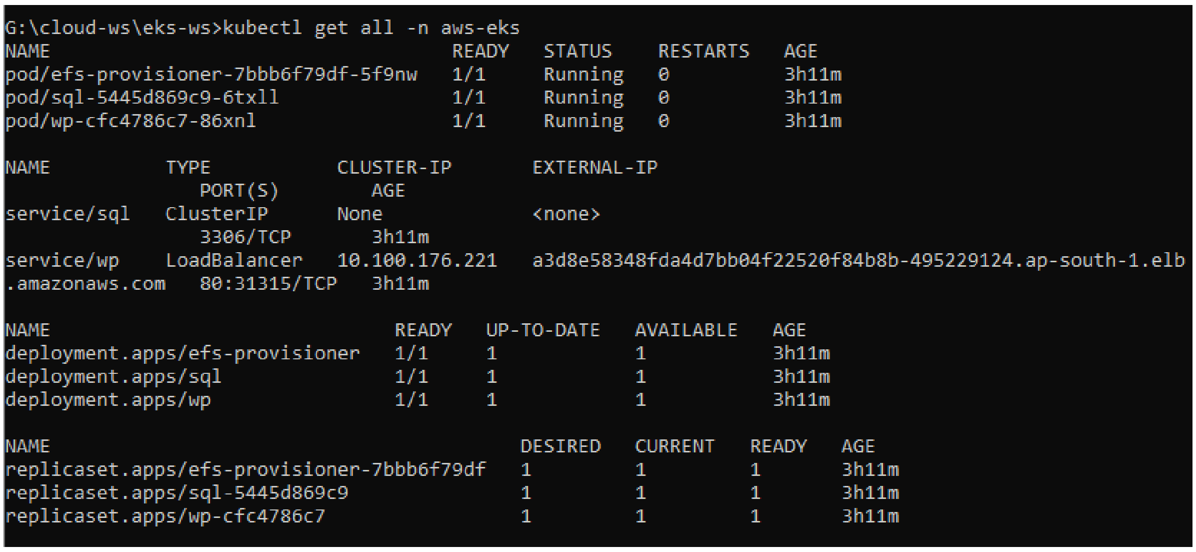
```
G:\cloud-ws\eks-ws>kubectl create namespace aws-eks
```

15. Run *kustomization.yml* in created namespace for finalizing the setup



```
G:\cloud-ws\eks-ws>kubectl create -k . -n aws-eks
storageclass.storage.k8s.io/aws-efs created
clusterrolebinding.rbac.authorization.k8s.io/nfs-prov-role created
secret/mysecure-bbt5ccch9b created
service/sql created
service/wp created
deployment.apps/efs-provisioner created
deployment.apps/sql created
deployment.apps/wp created
persistentvolumeclaim/efs-mysql created
persistentvolumeclaim/efs-wordpress created
```

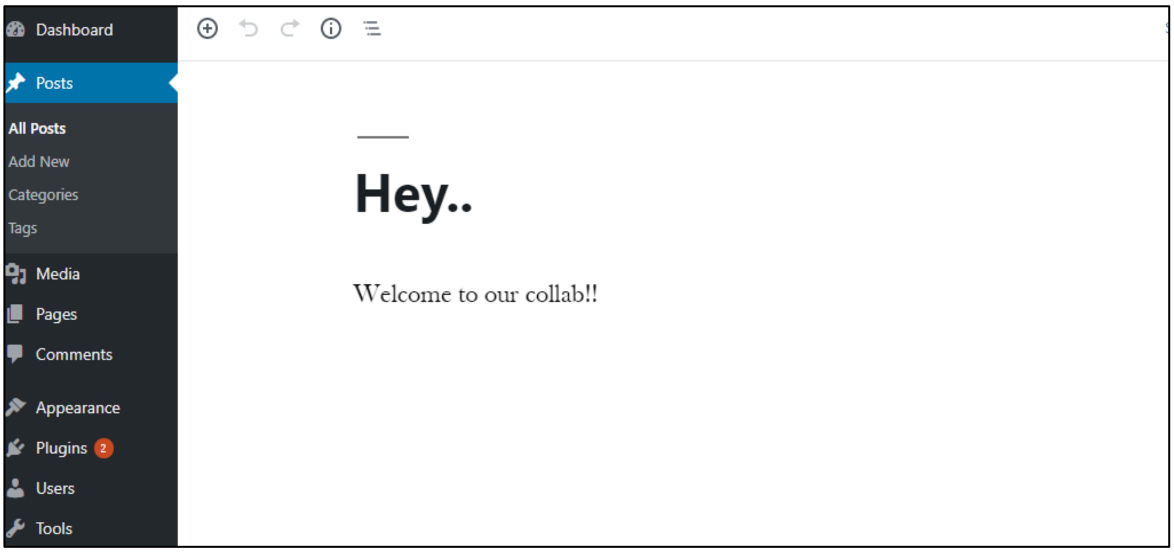
16. Hurrayy!! Your wordpress setup is ready



```
G:\cloud-ws\eks-ws>kubectl get all -n aws-eks
NAME                                  READY   STATUS    RESTARTS   AGE
pod/efs-provisioner-7bbb6f79df-5f9nw   1/1     Running   0          3h11m
pod/sql-5445d869c9-6txll               1/1     Running   0          3h11m
pod/wp-cfc4786c7-86xnl                 1/1     Running   0          3h11m

NAME          TYPE           CLUSTER-IP       EXTERNAL-IP
                 PORT(S)        AGE
service/sql   ClusterIP      None             <none>
                 3306/TCP       3h11m
service/wp    LoadBalancer   10.100.176.221   a3d8e58348fda4d7bb04f22520f84b8b-495229124.ap-south-1.elb
.amazonaws.com   80:31315/TCP   3h11m

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/efs-provisioner   1/1     1            1           3h11m
deployment.apps/sql               1/1     1            1           3h11m
deployment.apps/wp                1/1     1            1           3h11m

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/efs-provisioner-7bbb6f79df   1         1         1       3h11m
replicaset.apps/sql-5445d869c9               1         1         1       3h11m
replicaset.apps/wp-cfc4786c7                 1         1         1       3h11m
```

17. Go to ELB in AWS and copy DNS

Use DNS in your browser and access wordpress



Hey..

Welcome to our collab!!