K8s 08

1) Create and Test a Kubernetes Pod with an EmptyDir Volume

--Yaml file to create a Kubernetes Pod with an EmptyDir Volume.

Here We create volume inside a pod to store data related to container.

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
spec:
  containers:
  - name: firstcontainer
    image: nginx
    volumeMounts:
      - mountPath: /data  #Directory inside container
        name: first-volume   #any logical name
  volumes:
    - name: first-volume
      emptyDir: {}   #Blank Object
```

--run the yaml then pod got created

```
root@master:~# vi emptydir.yaml
root@master:~# kubectl create -f emptydir.yaml
pod/firstpod created
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          7s
```

Now test:

--now login to the firstpod

Then do ls now can see data directory

```
root@master:~# kubectl exec -it firstpod bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@firstpod:/# ls
bin  boot  data  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
```

Now cd data and create some files in that directory

```
root@firstpod:/# cd data/
root@firstpod:/data# touch a b c d e f
```

Stop nginx service to terminate your container

```
root@firstpod:/data# cd
root@firstpod:~# service nginx stop
command terminated with exit code 137
```

-now exit and check pods again then we can see pod will create a new container and try to login into container check status

```
root@master:~# exit
logout
ubuntu@master:~$ kubectl get pods
NAME       READY   STATUS    RESTARTS      AGE
firstpod   1/1     Running   1 (106s ago)   10m
ubuntu@master:~$
```

Again login and check also Files in directory /data will be visible

Note:

Problem with empty dir is if the pod got deleted then we will be loosing all our data.

2) Configure a HostPath Volume in Kubernetes and Validate Data Persistence

--Yaml file to create a Kubernetes Pod with an HostPath Volume.

Here We create volume on hostpath, means volume will be created outside pod.

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: firstpod
spec:
  containers:
  - name: firstcontainer
    image: nginx
    volumeMounts:
      - mountPath: /data  #Directory inside container
        name: first-volume    #any logical name
  volumes:
    - name: first-volume
      hostPath:
        path: /tmp/data    # Path inside host machine (Minikube)
```

-run yaml then pod got created

```
root@master:~# vi hostpath.yaml
root@master:~# kubectl create -f hostpath.yaml
pod/firstpod created
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          8s
```

-then check on which node pod got created

```
root@master:~# kubectl get pods -o wide
NAME       READY   STATUS    RESTARTS   AGE   IP               NODE                NOMINATED NODE   READINESS GATES
firstpod   1/1     Running   0          37s   192.168.201.231  ip-172-31-13-158    <none>           <none>
```

Now test:

-Login to container and create some random files in /data location.

```
root@master:~# kubectl exec -it firstpod bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl e
root@firstpod:/# ls
bin  boot  data  dev  docker-entrypoint.d  docker-entrypoint.sh  etc  home  lib  lib64  media  mn
root@firstpod:/# cd data/
root@firstpod:/data# touch created_inside_pod
root@firstpod:/data# ls
created_inside_pod
root@firstpod:/data#
```

Check in worker-02

```
root@worker-02:~# ls
created_inside_pod  snap
```

Now delete the pod and create a new pod.

```
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          19m
root@master:~# kubectl delete pods --all
pod "firstpod" deleted
```

Now create new pod

```
root@master:~# kubectl create -f hostpath.yaml
pod/firstpod created
```

Login to new pod and checked if files are available in data directory.

```
root@master:~# kubectl exec -it firstpod bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@firstpod:/# cd data/
root@firstpod:/data# ls
created_inside_pod
root@firstpod:/data#
```

Note:

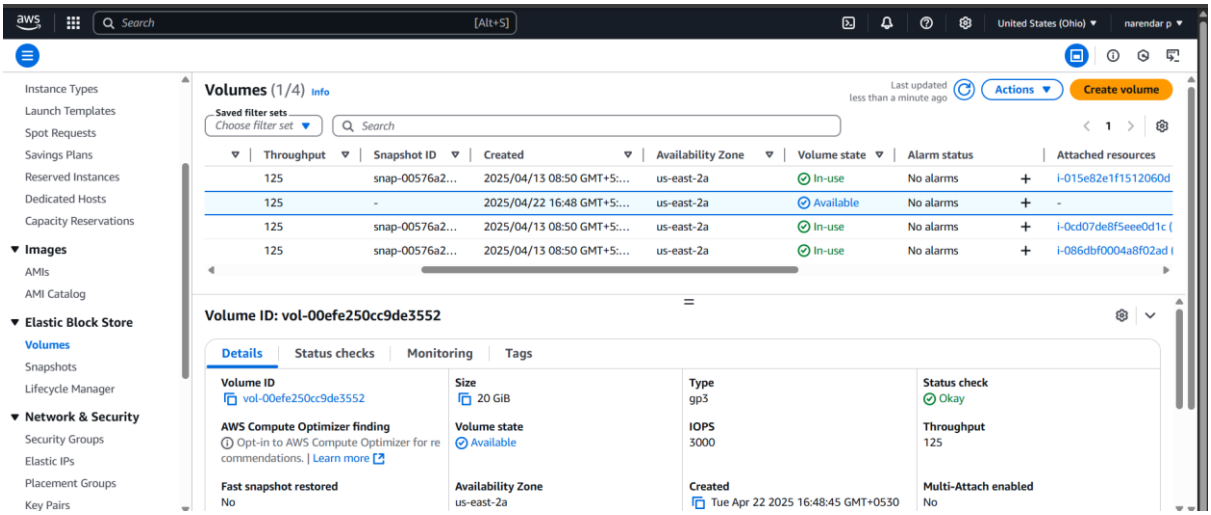
If we have multiple nodes then we will other nodes will not be able to access the volume created on node.

3) Deploy an Amazon EBS Volume Using Persistent Volume and Persistent Volume Claim (PVC)

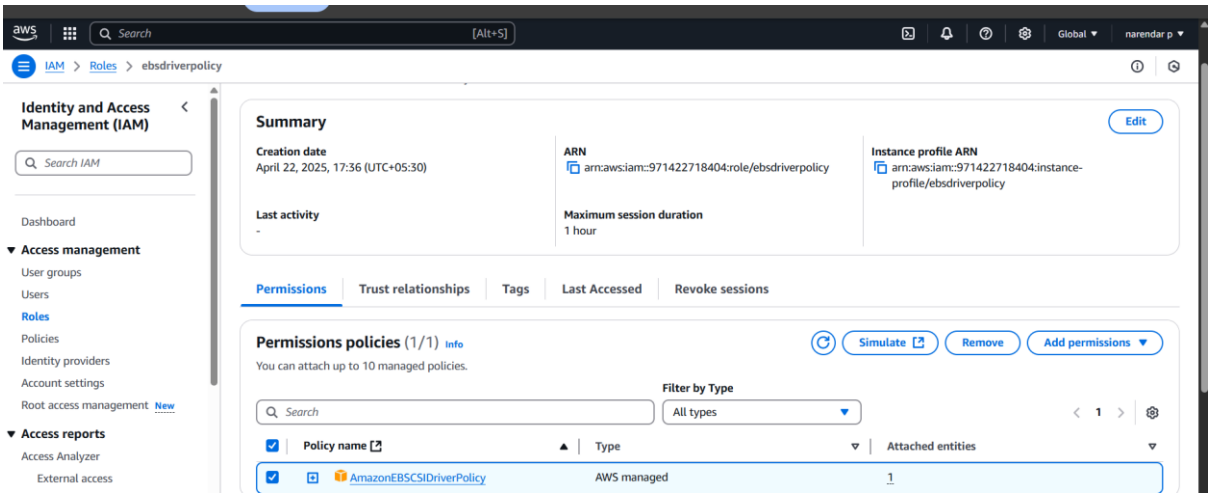
Here If we have multinode k8s cluster,then in this case we need to keep our volume outside the cluster.

If our pod is created on other node then the volume should also be moved to that node.

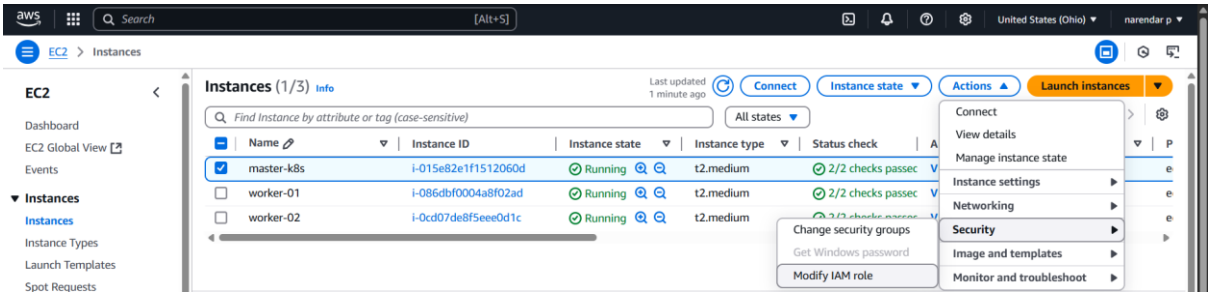
--create an Amazon EBS Volume



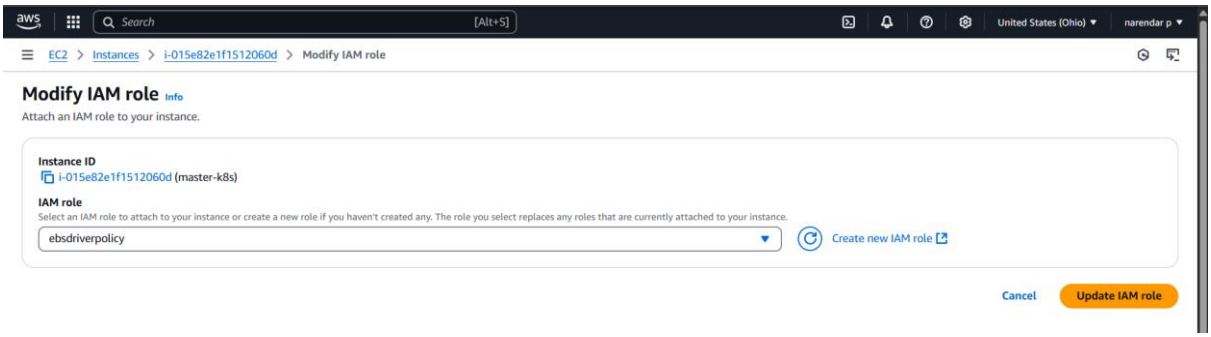
--create an iam role

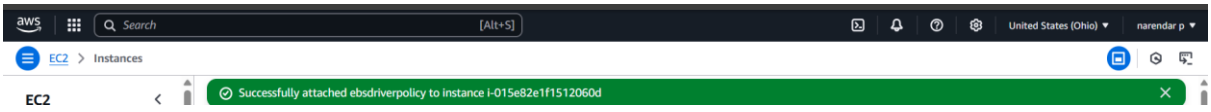


--now go to ec2 select master-security-modifyIAM role



Then select creted iam role and click on update iam role





Do this worker nodes also

--now install EBS CSI driver on k8s masternode by using below url

git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git

```
root@master:~# git clone https://github.com/kubernetes-sigs/aws-ebs-csi-driver.git
Cloning into 'aws-ebs-csi-driver'...
remote: Enumerating objects: 33956, done.
remote: Counting objects: 100% (4718/4718), done.
remote: Compressing objects: 100% (944/944), done.
remote: Total 33956 (delta 4010), reused 3778 (delta 3772), pack-reused 29238 (from 2)
Receiving objects: 100% (33956/33956), 28.60 MiB | 2.98 MiB/s, done.
Resolving deltas: 100% (19655/19655), done.
```

Then execute these

cd aws-ebs-csi-driver

kubectl apply -k deploy/kubernetes/overlays/stable/

kubectl get pods -n kube-system **#To verify the driver is installed and running.**

we can see ebs pods are created

```
root@master:~# cd aws-ebs-csi-driver
root@master:~/aws-ebs-csi-driver# kubectl apply -k deploy/kubernetes/overlays/stable/
serviceaccount/ebs-csi-controller-sa created
serviceaccount/ebs-csi-node-sa created
role.rbac.authorization.k8s.io/ebs-csi-leases-role created
clusterrole.rbac.authorization.k8s.io/ebs-csi-node-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-attacher-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-resizer-role created
clusterrole.rbac.authorization.k8s.io/ebs-external-snapshotter-role created
rolebinding.rbac.authorization.k8s.io/ebs-csi-leases-rolebinding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-attacher-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-node-getter-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-provisioner-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-resizer-binding created
clusterrolebinding.rbac.authorization.k8s.io/ebs-csi-snapshotter-binding created
deployment.apps/ebs-csi-controller created
poddisruptionbudget.policy/ebs-csi-controller created
daemonset.apps/ebs-csi-node created
csidriver.storage.k8s.io/ebs.csi.aws.com created
root@master:~/aws-ebs-csi-driver# kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS       AGE
calico-kube-controllers-658d97c59c-xxs7b   1/1     Running   8 (3h3m ago)   8d
calico-node-496nd                     1/1     Running   8 (3h3m ago)   8d
calico-node-cp9gc                     1/1     Running   0              8d
calico-node-xsfcf                     1/1     Running   8 (3h3m ago)   8d
coredns-76f75df574-6lts2              1/1     Running   7 (3h3m ago)   7d3h
coredns-76f75df574-nvn9b              1/1     Running   0              177m
ebs-csi-controller-6d89bf7f66-l2ck6   6/6     Running   0              64s
ebs-csi-controller-6d89bf7f66-q5n26   6/6     Running   0              64s
ebs-csi-node-2h7hj                    3/3     Running   0              64s
ebs-csi-node-cxnvs                    3/3     Running   0              64s
ebs-csi-node-wvcf9                    3/3     Running   0              64s
etcd-master                           1/1     Running   1 (159m ago)   8d
kube-apiserver-master                 1/1     Running   1 (159m ago)   8d
kube-controller-manager-master        1/1     Running   0              8d
kube-proxy-lcq4p                      1/1     Running   8 (3h3m ago)   8d
kube-proxy-m682x                      1/1     Running   0              8d
kube-proxy-pqkkc                      1/1     Running   8 (3h3m ago)   8d
kube-scheduler-master                 1/1     Running   0              8d
metrics-server-75bf97fcc9-qmn68       0/1     Running   1 (3h3m ago)   25h
root@master:~/aws-ebs-csi-driver#
```

--Now we need to create one Persistent volume for our EBS

root@master: ~

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-ebs-pv
spec:
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: aws-ebs
  awsElasticBlockStore:
    volumeID: vol-00efe250cc9de3552      #EBS volume ID
    fsType: ext4
```

Run yaml then pv created

```
root@master:~# kubectl create -f pv-ebs.yaml
persistentvolume/my-ebs-pv created
root@master:~#
```

```
root@master:~# kubectl get pv
NAME        CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
my-ebs-pv   10Gi       RWO            Retain           Available           aws-ebs        <unset>                          64s
```

--now create PVC

```
root@master: ~
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-ebs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
  storageClassName: aws-ebs
```

Now run yaml then pvc got created

```
root@master:~# vi pvc-ebs.yaml
root@master:~# kubectl create -f pvc-ebs.yaml
persistentvolumeclaim/my-ebs-pvc created
```

```
root@master:~# kubectl get pvc
NAME         STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
my-ebs-pvc   Bound    my-ebs-pv   10Gi       RWO            aws-ebs        <unset>                 61s
```

--Now Create pod with PVC

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
    - name: my-container
      image: nginx
      volumeMounts:
        - name: data-volume
          mountPath: /data            #data where we can write data
  volumes:
    - name: data-volume
      persistentVolumeClaim:
        claimName: my-ebs-pvc        #PVC Name
```

Run the yaml then pod got created

```
root@master:~# kubectl apply -f pod.yaml
pod/my-pod created
```

```
root@master:~# kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
firstpod   1/1     Running   0          93m
my-pod     1/1     Running   0          46s
```

Now test:

Check pod is creted on which node

```
NAME       READY   STATUS    RESTARTS   AGE     IP               NODE                NOMINATED NODE   READINESS GATES
firstpod   1/1     Running   0          96m     192.168.201.232  ip-172-31-13-158    <none>           <none>
my-pod     1/1     Running   0          3m44s   192.168.201.238  ip-172-31-13-158    <none>           <none>
```

Now login to pod

Cd data

Crete one file and exit

```
root@master:~# kubectl exec -it my-pod bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@my-pod:/# cd data/
root@my-pod:/data# ls
lost+found
root@my-pod:/data# touch create_in_worker-02-pod
root@my-pod:/data# exit
```

Once pod is created then we can check the volume as it will show in use status.

Delete the pod and try to see the status of volume

```
root@master:~# kubectl delete pods --all
pod "firstpod" deleted
```

```
root@master:~# kubectl get pvc my-ebs-pvc
NAME         STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
my-ebs-pvc   Bound    my-ebs-pv  10Gi       RWO            aws-ebs        <unset>                 17m
```

Create new pod again and this time if it is scheduled on another node(worker-02) then volume will be attached to that node.

```
root@master:~# kubectl apply -f pod.yaml
pod/my-pod created
```

```
root@master:~# kubectl get pods -o wide
NAME     READY   STATUS    RESTARTS   AGE   IP               NODE               NOMINATED NODE   READINESS GATES
my-pod   1/1     Running   0          38s   192.168.201.237  ip-172-31-13-158   <none>           <none>
```

Now login and check volume attached to that node.

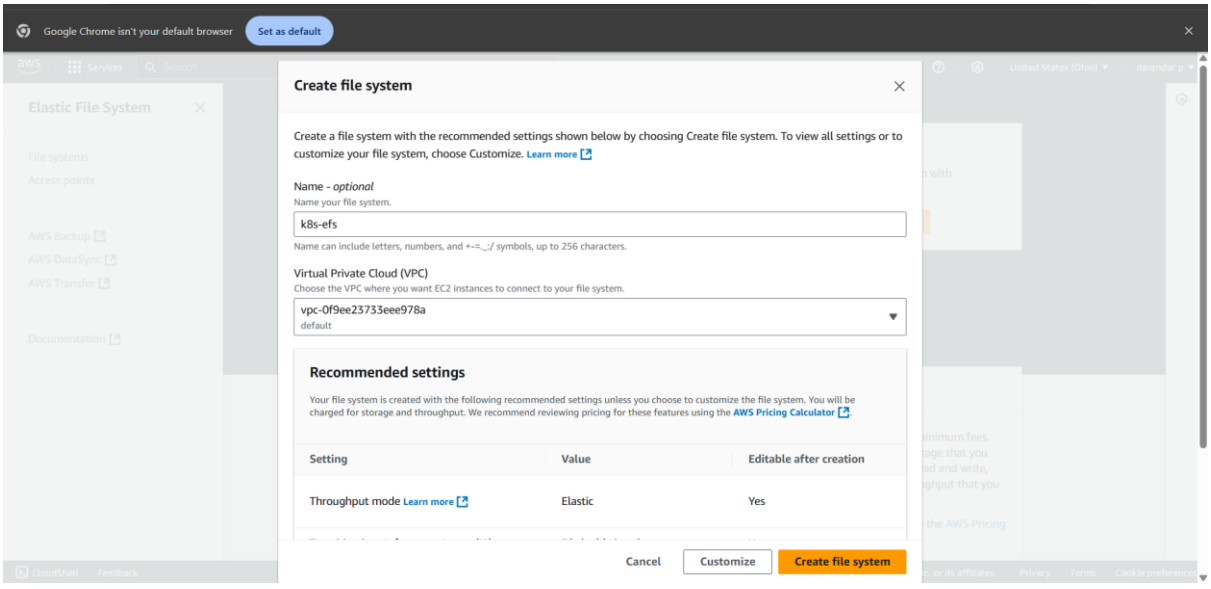
```
root@master:~# kubectl exec -it my-pod bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@my-pod:/# cd data/
root@my-pod:/data# ls
create_in_worker-02-pod  lost+found
root@my-pod:/data#
```

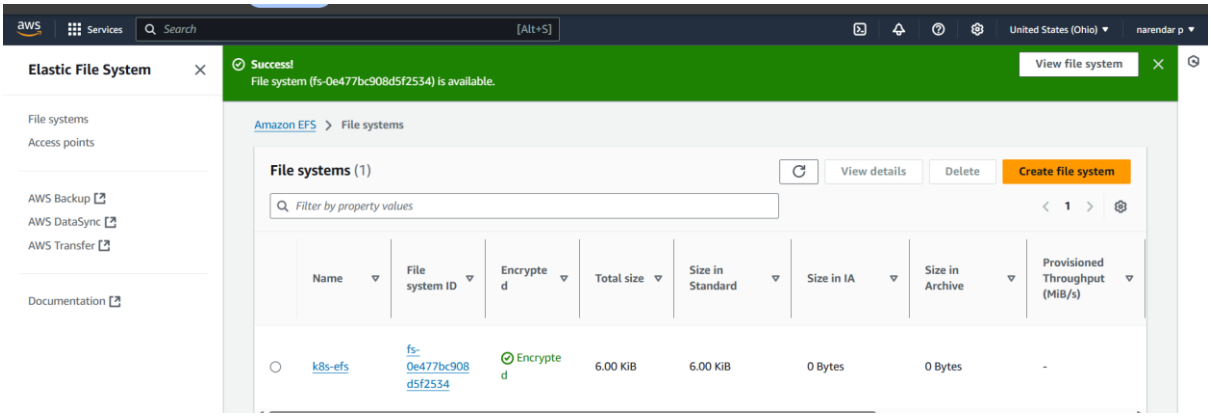
Note:

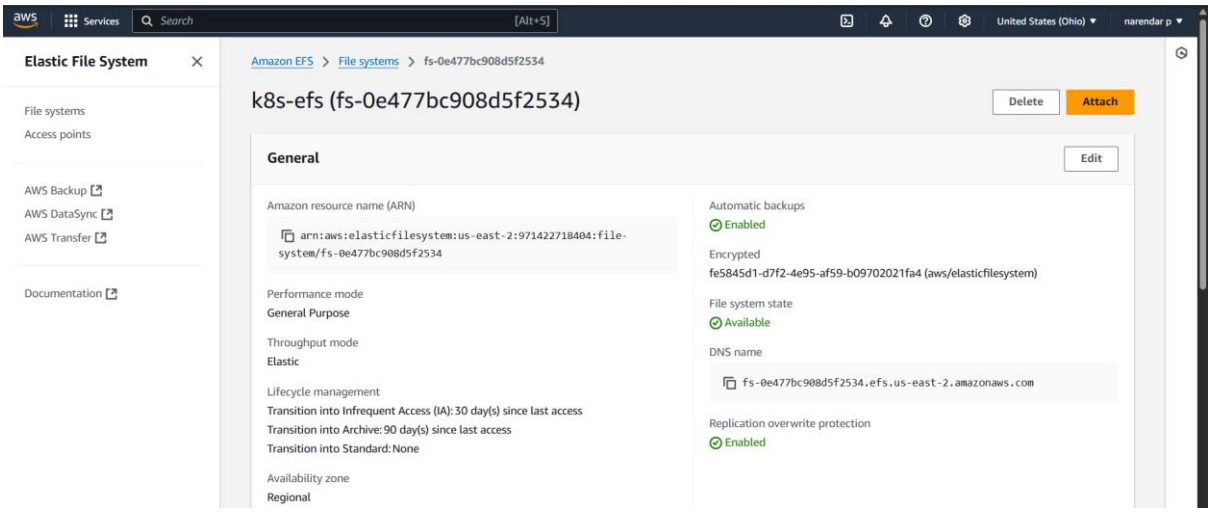
Here we can map multiple pods to the volumes but we cannot map two pods to same volume at same point of time.

4) Set Up an Amazon EFS Volume and Attach it to Multiple Pods

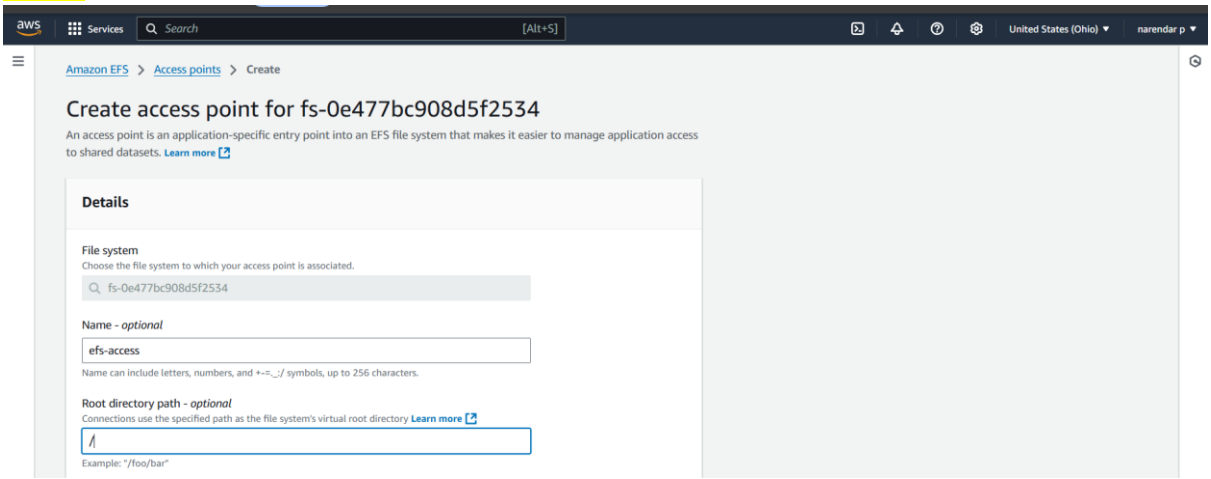
Step 1: Create an Amazon EFS File System

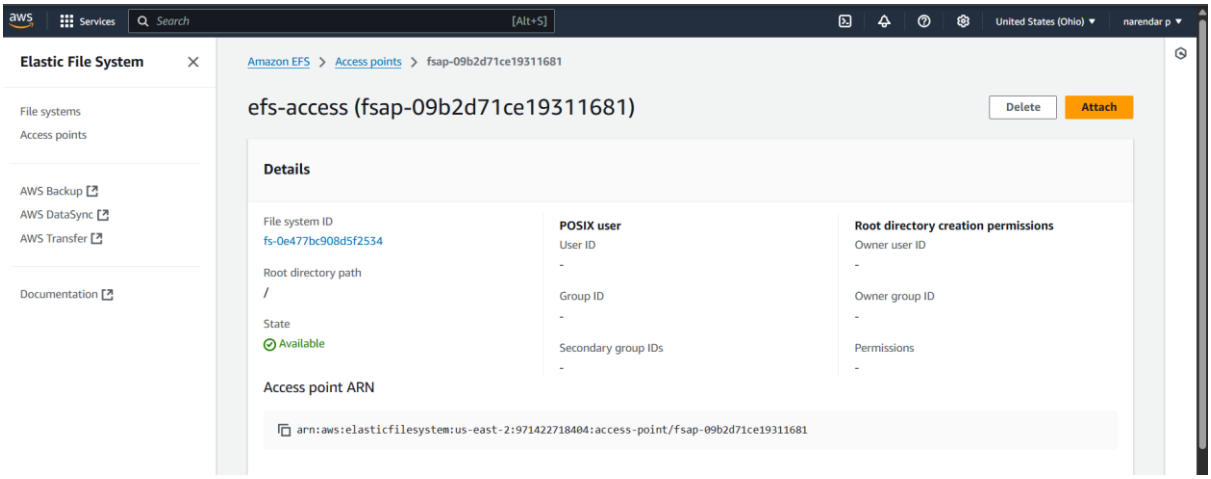






Step 2: Create an Access Point





Step 3: Install the Amazon EFS CSI Driver

kubectl apply -k "github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/ecr/?ref=release-1.7"

```
root@master:~# kubectl apply -k "github.com/kubernetes-sigs/aws-efs-csi-driver/deploy/kubernetes/overlays/stable/ecr/?ref=release-1.7"
# Warning: 'bases' is deprecated. Please use 'resources' instead. Run 'kustomize edit fix' to update your Kustomization automatically.
serviceaccount/efs-csi-controller-sa created
serviceaccount/efs-csi-node-sa created
clusterrole.rbac.authorization.k8s.io/efs-csi-external-provisioner-role created
clusterrole.rbac.authorization.k8s.io/efs-csi-external-provisioner-role-describe-secrets created
clusterrole.rbac.authorization.k8s.io/efs-csi-node-role created
rolebinding.rbac.authorization.k8s.io/efs-csi-provisioner-binding-describe-secrets created
clusterrolebinding.rbac.authorization.k8s.io/efs-csi-node-binding created
clusterrolebinding.rbac.authorization.k8s.io/efs-csi-provisioner-binding created
deployment.apps/efs-csi-controller created
daemonset.apps/efs-csi-node created
csidriver.storage.k8s.io/efs.csi.aws.com created
```

Step 4: Create a StorageClass

Yaml file

root@master: ~

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: efs-sc
provisioner: efs.csi.aws.com
```

Run yaml

```
root@master:~# kubectl apply -f efs-sc.yaml
storageclass.storage.k8s.io/efs-sc created
```

Step 5: Create a Persistent Volume (PV)

Yaml file

root@master: ~

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: efs-pv
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: efs-sc
  csi:
    driver: efs.csi.aws.com
    volumeHandle: fs-0e477bc908d5f2534::fsap-09b2d71ce19311681 # Format: <FileSystemId>::<AccessPointId>
```

Run yaml then PV got created

```
root@master:~# kubectl apply -f efs-pv.yaml
persistentvolume/efs-pv created
root@master:~# kubectl get pv
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
efs-pv   5Gi        RWX            Retain           Available           efs-sc         <unset>                          19s
```

Step 6: Create a Persistent Volume Claim (PVC)

Yaml file

root@master: ~

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: efs-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: efs-sc
  resources:
    requests:
      storage: 5Gi
```

Run the yaml file then pvc got created

```
root@master:~# kubectl apply -f efs-pvc.yaml
persistentvolumeclaim/efs-pvc created
```

```
root@master:~# kubectl get pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
efs-pvc   Bound    efs-pv   5Gi        RWX            efs-sc         <unset>                 46s
```

Step 7: Use PVC in Multiple Pods

root@master: ~

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod1
spec:
  containers:
  - name: app
    image: busybox
    command: [ "sleep", "3600" ]
    volumeMounts:
    - name: efs-vol
      mountPath: /mnt/efs
  volumes:
  - name: efs-vol
    persistentVolumeClaim:
      claimName: efs-pvc
```

Run yaml file

kubectl apply -f pod.yaml

```
root@master:~# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
efs-pod   1/1     Running   0          47m
```

Test now:

Now login to pod and create some random files in efs-mount directory.

```
root@master:~# kubectl exec -it efs-pod -- bash
root@efs-pod:/# cd /efs-mount
```

```
root@master:~# kubectl exec -it efs-pod -- bash
root@efs-pod:/# cd /efs-mount
echo "Hello from EFS!" > hello.txt
touch random-file-{1..5}.txt

root@efs-pod:/efs-mount# ls
hello.txt  random-file-1.txt  random-file-2.txt  random-file-3.txt  random-file-4.txt  random-file-5.txt
```

Now Create one more pod and check if the files are available or not

In above yaml file just edit name to pod2 and apply and create one more pod

```
root@master:~# kubectl apply -f pod.yaml
pod/efs-pod2 created
```

Again login and check the files

```
root@master:~# kubectl exec -it efs-pod2 -- bash
root@efs-pod2:/# ls
bin  boot  dev  docker-entrypoint.d  docker-entrypoint.sh  efs-mount  etc  home  lib  lib64  media  mnt  opt  proc  root  run
root@efs-pod2:/# cd efs-mount/
root@efs-pod2:/efs-mount# ls
hello.txt  random-file-1.txt  random-file-2.txt  random-file-3.txt  random-file-4.txt  random-file-5.txt
root@efs-pod2:/efs-mount#
```

We can now connect multiple pods to same volume

Note:

Here we can map multiple pods to same EFS volume.

EFS cost is 3time higher to EBS.

5) Implement and Test Liveness and Readiness Probes in a Kubernetes Pod

----yaml file to create pod with liveness probe

root@master: ~

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-app
    image: nginx            #Image Name
    ports:
    - containerPort: 80
    livenessProbe:
      httpGet:              #get request or we can use tcpSocket, or exec
        path: /              #health check path
        port: 80
      initialDelaySeconds: 15    #It will wait 15 second before applying first liveness
      periodSeconds: 10          #Every 10 seconds the liveness will be checked.
```

--run the yaml

kubectl apply -f pod.yaml

```
root@master:~# vi pod.yaml
root@master:~# kubectl apply -f pod.yaml
pod/my-pod created
```

-pod created

```
root@master:~# kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
my-pod    1/1     Running   0          8s
```

-describe the pod

kubectl describe pod my-pod

we can see liveness of pod

```
Status:           Running
IP:               192.168.201.229
IPs:
  IP:  192.168.201.229
Containers:
  my-app:
    Container ID:   containerd://b1281635a727dc8fca880e4ac7838f38771ae6d95a1ccdcfbd753bf431794ed9
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:5ed8fcc66f4ed123c1b2560ed708dc148755b6e4cbd8b943fab094f
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 22 Apr 2025 09:45:42 +0000
    Ready:          True
    Restart Count:  0
    Liveness:       http-get http://:80/ delay=15s timeout=1s period=10s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-48ccg (ro)
                    node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  30s   default-scheduler  Successfully assigned default/my-pod to ip-172-31-13-158
  Normal  Pulling    29s   kubelet            Pulling image "nginx"
  Normal  Pulled     29s   kubelet            Successfully pulled image "nginx" in 215ms (215ms including waiting)
  Normal  Created    29s   kubelet            Created container: my-app
  Normal  Started    29s   kubelet            Started container my-app
```

Now Test Liveness probe of a my-pod

So create one yaml with wrong container ports

```
root@master: ~
apiVersion: v1
kind: Pod
metadata:
  name: first-pod
spec:
  containers:
  - name: my-app
    image: nginx            #Image Name
    ports:
    - containerPort: 8080
    livenessProbe:
      httpGet:              #get request or we can use tcpSocket, or exec
        path: /              #health check path
        port: 8080
      initialDelaySeconds: 5    #It will wait 15 second before applying first liveness
      periodSeconds: 5          #Every 10 seconds the liveness will be checked.
```

--run yaml file

First-Pod created

```
root@master:~# vi pod.yaml
root@master:~# kubectl apply -f pod.yaml
pod/first-pod created
root@master:~# kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
first-pod    1/1     Running   0          10s
my-pod       1/1     Running   0          12m
```

-describe first-pod

We can see Container my-app failed liveness probe, will be restarted

```
                    node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  42s                default-scheduler  Successfully assigned default/first-pod to ip-172-31-13-158
  Normal   Pulled     41s                kubelet            Successfully pulled image "nginx" in 228ms (228ms including waiting)
  Normal   Pulled     22s                kubelet            Successfully pulled image "nginx" in 226ms (226ms including waiting)
  Normal   Pulling    2s (x3 over 42s)   kubelet            Pulling image "nginx"
  Warning  Unhealthy  2s (x6 over 32s)   kubelet            Liveness probe failed: Get "http://192.168.201.230:8080/": dial tcp 192
used
  Normal   Killing    2s (x2 over 22s)   kubelet            Container my-app failed liveness probe, will be restarted
  Normal   Created    1s (x3 over 41s)   kubelet            Created container: my-app
  Normal   Started    1s (x3 over 41s)   kubelet            Started container my-app
  Normal   Pulled     1s                 kubelet            Successfully pulled image "nginx" in 439ms (439ms including waiting)
```

-check pods now

It restarted got failed

```
root@master:~# kubectl get pods
NAME        READY   STATUS             RESTARTS      AGE
first-pod   0/1     CrashLoopBackOff   6 (57s ago)   5m22s
my-pod      1/1     Running            0             17m
```

----yaml file to create pod with liveness probe

root@master: ~

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod1
spec:
  containers:
  - name: my-app
    image: nginx
    ports:
    - containerPort: 80
    readinessProbe:
      httpGet:
        path: /index.html
        port: 80
      initialDelaySeconds: 10
      periodSeconds: 5
```

--run yaml file

My-Pod1 created

```
root@master:~# vi pod1.yaml
root@master:~# kubectl apply -f pod1.yaml
pod/my-pod1 created
root@master:~# vi pod1.yaml
root@master:~# kubectl get pods
NAME        READY   STATUS             RESTARTS        AGE
first-pod   0/1     CrashLoopBackOff   7 (2m54s ago)   10m
my-pod      1/1     Running            0               22m
my-pod1     1/1     Running            0               81s
```

-describe my-pod1

We can see readiness of a pod

```
Containers:
  my-app:
    Container ID:   containerd://99f65d1c8c4391ff85aa4369a2d437e0350ebb27d7f6a4c93158b03f0d47d152
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:5ed8fcc66f4ed123c1b2560ed708dc148755b6e4cbd8b943fab094f2c6bfa91e
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 22 Apr 2025 10:06:50 +0000
    Ready:          True
    Restart Count:  0
    Readiness:      http-get http://:80/index.html delay=10s timeout=1s period=5s #success=1 #failure=3
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-955qm (ro)
```

Now test

For that Login to the pod and delete the index.html file

```
root@master:~# kubectl exec -it my-pod1 bash
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
root@my-pod1:/# cd /usr/share/nginx/html
root@my-pod1:/usr/share/nginx/html# rm -rf *
root@my-pod1:/usr/share/nginx/html#
```

Now check pods can see pod is not ready not serve the traffic

```
root@master:~# kubectl get pods
NAME         READY   STATUS             RESTARTS        AGE
first-pod    0/1     CrashLoopBackOff   9 (5m9s ago)    18m
my-pod       1/1     Running            0               30m
my-pod1      0/1     Running            0               9m21s
```

-describe my-pod1

Readiness probe failed

```
Events:
  Type     Reason     Age                 From               Message
  ----     ------     ----                ----               -------
  Normal   Scheduled  9m54s               default-scheduler  Successfully assigned default/my-pod1 to ip-172-31-13-158
  Normal   Pulling    9m53s               kubelet            Pulling image "nginx"
  Normal   Pulled     9m53s               kubelet            Successfully pulled image "nginx" in 207ms (207ms including waiting)
  Normal   Created    9m53s               kubelet            Created container: my-app
  Normal   Started    9m53s               kubelet            Started container my-app
  Warning  Unhealthy  4s (x18 over 84s)   kubelet            Readiness probe failed: HTTP probe failed with statuscode: 404
```