|  |  |  |
| --- | --- | --- |
| Kubernetes Storage Concept | KIND Implementation/Demo | EKS Implementation/Hint |
| Volumes (emptyDir) | Direct usage in Pod YAML. | Direct usage in Pod YAML. Behaves identically. Use for temporary, scratch space. |
| PersistentVolume (PV) | hostPath for static provisioning (local file system). | Dynamically provisioned. Represents AWS EBS volumes, EFS file systems, or FSx file systems. |
| PersistentVolumeClaim (PVC) | Request for storage, binds to hostPath PV. | Request for storage, binds to dynamically created AWS storage. |
| StorageClass | Uses KIND's default standard (which is a hostPath provisioner). | Crucial for dynamic provisioning. Points to AWS CSI drivers (e.g., ebs.csi.aws.com, efs.csi.aws.com). Defines storage types (gp2, gp3, io1 for EBS), encryption, etc. |
| CSI | Manually deploy a hostpath-csi driver to understand its mechanics. | The standard for AWS storage integration. AWS EBS CSI driver and AWS EFS CSI driver are commonly used. Requires IAM roles for service accounts for provisioning AWS resources. |

**Small Hint for EKS:** For stateful applications like databases in EKS, you'd use a StorageClass pointing to the **EBS CSI driver**. You'd also likely use a **StatefulSet** instead of a Deployment, as StatefulSets provide stable network identities and ordered, graceful deployment/scaling for Pods, crucial for databases. Each Pod in a StatefulSet gets its own PVC (and thus its own EBS volume), which is ideal for isolating data.

----------------------------------------------------------------------------------------

Eg:

# storageclass-gp3.yaml

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: **gp3-sc**

provisioner: ebs.csi.aws.com # This is the key: the EBS CSI driver

parameters:

type: gp3 # General Purpose SSD, cost-effective

fsType: ext4

encrypted: "true" # Always good practice for encryption

volumeBindingMode: WaitForFirstConsumer # Important for better scheduling

reclaimPolicy: Delete # Default for dynamic provisioning

**Key steps in EKS for CSI drivers:**

1. **Enable IAM OIDC provider for your EKS cluster:** This allows Kubernetes service accounts to assume IAM roles for interacting with AWS services (like creating EBS volumes).
2. **Create an IAM Policy** with the necessary permissions for the CSI driver (e.g., ec2:CreateVolume, ec2:AttachVolume for EBS).
3. **Create an IAM Role** and associate the policy with it.
4. **Create a Kubernetes Service Account** and annotate it with the IAM Role ARN (eks.amazonaws.com/role-arn).
5. **Deploy the AWS EBS CSI driver** (either via EKS Add-on or by applying the official manifests from github.com/kubernetes-sigs/aws-ebs-csi-driver). The driver's pods will run with the service account that has the necessary IAM permissions.
6. Define StorageClass resources like the gp3-sc example shown earlier, referencing ebs.csi.aws.com as the provisioner.

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-csi-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 500Mi

storageClassName: **gp3-sc** # Use the StorageClass provisioned by the CSI driver

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-csi-deployment

spec:

replicas: 1

selector:

matchLabels:

app: nginx-csi

template:

metadata:

labels:

app: nginx-csi

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

volumeMounts:

- name: csi-storage

mountPath: /usr/share/nginx/html

volumes:

- name: csi-storage

persistentVolumeClaim:

claimName: my-csi-pvc