**ceph storage**

**Ceph Storage** is an open-source, software-defined storage platform that provides **highly scalable object, block, and file-based storage** in a unified system. It’s widely used in cloud-native environments, especially in platforms like **OpenStack**, **Kubernetes**, and **OpenShift**, due to its flexibility, fault tolerance, and performance.

**Core Components of Ceph:**

| **Component** | **Description** |
| --- | --- |
| **Ceph Monitor (MON)** | Maintains the cluster map and state; handles authentication. |
| **Ceph Manager (MGR)** | Provides additional monitoring and interfaces (dashboard, metrics, etc). |
| **Ceph OSD (Object Storage Daemon)** | Handles actual data storage, replication, recovery, and rebalancing. |
| **Ceph MDS (Metadata Server)** | Manages metadata for CephFS (Ceph File System). |
| **RADOS** | Underlying object storage layer that stores data as objects in pools. |

**Ceph Storage Interfaces:**

| **Interface** | **Use Case** |
| --- | --- |
| **RBD** (RADOS Block Device) | Block storage for VMs or applications. |
| **CephFS** | POSIX-compliant shared filesystem. |
| **RGW** (RADOS Gateway) | Object storage compatible with S3 and Swift APIs. |

**Key Features:**

* **Highly Available** – Uses replication or erasure coding for redundancy.
* **Self-Healing** – Automatically detects and repairs failures.
* **Scalable** – Horizontally scales from a few nodes to thousands.
* **Decentralized** – No single point of failure.
* **Flexible Backends** – Works with HDDs, SSDs, or NVMe.
* **Open Source** – Maintained by the Ceph Foundation and community.

**How Ceph stores your data:**

1. You upload a file (say a photo).
2. Ceph **splits it into chunks** and **stores it in different places** (OSDs).
3. It makes **extra copies or uses erasure coding** to ensure **no data loss**.
4. You don’t need to know where it went — Ceph knows and finds it for you.

**Why it's awesome:**

* **No single point of failure** – Nothing breaks the system entirely.
* **Self-healing** – Lost data is recreated automatically.
* **Expandable** – Add more storage anytime, no need to reconfigure everything.

**Where is the data actually stored in Ceph (in Kubernetes)?**

The data is **physically stored on the disks of the worker nodes** (or dedicated storage nodes) where **Ceph OSDs (Object Storage Daemons)** are running.

**🔍 More Clearly:**

When you deploy Ceph in Kubernetes (typically via Rook), here's what happens:

| **Component** | **What it does** | **Where it runs** | **What it uses** |
| --- | --- | --- | --- |
| **Ceph OSDs** | Store your actual data (chunks, blocks, files). | On nodes with available disks (HDDs or SSDs). | Raw disks or partitions mounted on those nodes. |

**Example Setup:**

Let's say you have 3 worker nodes:

| **Node Name** | **Attached Disk** | **OSD Status** |
| --- | --- | --- |
| worker-1 | /dev/sdb (1TB) | Ceph OSD running |
| worker-2 | /dev/sdb (1TB) | Ceph OSD running |
| worker-3 | /dev/sdb (1TB) | Ceph OSD running |

* Ceph will **initialize these disks** and spread the data across them.
* It uses **replication** or **erasure coding** to make sure your data is safe (e.g., 3 copies in 3 OSDs).
* So when your pod writes data, it goes through the **Ceph CSI driver** → **Ceph cluster** → **OSDs on these disks**.
* **Example:**
* You deploy a pod with a PVC → Ceph creates a volume → Actual data blocks are written to /dev/sdb on one or more nodes running Ceph OSDs.

**You can scale Ceph horizontally by adding more storage. This can be done in two ways:**

**1. Add New Disks to Existing Nodes**

* Example: Add a new disk (/dev/sdc) to a node already running a Ceph OSD.
* Rook can **automatically detect the new disk** and create a new OSD on it (if auto-discovery is enabled).
* That disk will now participate in storing Ceph data.
* Ceph will **rebalance** and redistribute data across the old and new OSDs.

**How to integrate it (Rook-Ceph way – community version)?**

**Step-by-Step:**   
  
Prepare your OpenShift cluster   
Make sure it's running, has at least 3 worker nodes, and you can install Operators (from OperatorHub or via CLI).   
  
Install Rook-Ceph Operator   
This Operator is like a robot that installs and manages Ceph for you.   
  
**You can do this using:**   
oc create namespace rook-ceph

git clone <https://github.com/rook/rook.git>   
cd rook/cluster/examples/kubernetes/ceph   
kubectl apply -f common.yaml   
kubectl apply -f operator.yaml

**Create the Ceph Cluster**

This defines how Ceph will run inside OpenShift.   
It creates monitors, OSDs (storage daemons), and other parts of the Ceph system.   
Example:   
  
apiVersion: [ceph.rook.io/v1](https://ceph.rook.io/v1)   
kind: CephCluster   
metadata:   
  name: rook-ceph   
  namespace: rook-ceph   
spec:   
  cephVersion:   
    image: ceph/ceph:v18   
  dataDirHostPath: /var/lib/rook   
  storage:   
    useAllNodes: true   
    useAllDevices: true  
Apply it:   
kubectl apply -f cluster.yaml   
  
**Create StorageClass (SC)**

This tells OpenShift how to create volumes using Ceph.   
**Example SC for block storage (RBD):**

apiVersion: [storage.k8s.io/v1](https://storage.k8s.io/v1)   
kind: StorageClass   
metadata:   
  name: rook-ceph-block   
provisioner: [rook-ceph.rbd.csi.ceph.com](https://rook-ceph.rbd.csi.ceph.com/)   
parameters:   
  clusterID: rook-ceph   
  pool: replicapool   
  imageFormat: "2"   
  imageFeatures: layering   
  [csi.storage.k8s.io/provisioner-secret-name](https://csi.storage.k8s.io/provisioner-secret-name): rook-csi-rbd-provisioner   
  [csi.storage.k8s.io/provisioner-secret-namespace](https://csi.storage.k8s.io/provisioner-secret-namespace): rook-ceph   
  [csi.storage.k8s.io/node-stage-secret-name](https://csi.storage.k8s.io/node-stage-secret-name): rook-csi-rbd-node   
  [csi.storage.k8s.io/node-stage-secret-namespace](https://csi.storage.k8s.io/node-stage-secret-namespace): rook-ceph   
reclaimPolicy: Delete   
allowVolumeExpansion: true   
volumeBindingMode: Immediate  
  
Apply it:   
kubectl apply -f rook-ceph-block-sc.yaml   
  
Use PVC in apps   
Apps can now request storage dynamically.   
  
**Here’s a sample PVC:**

apiVersion: v1   
kind: PersistentVolumeClaim   
metadata:   
  name: ceph-pvc   
spec:   
  accessModes:   
    - ReadWriteOnce   
  storageClassName: rook-ceph-block   
  resources:   
    requests:   
      storage: 5Gi  
  
**And use it in a Deployment:**

apiVersion: apps/v1   
kind: Deployment   
metadata:   
  name: nginx-app   
spec:   
  replicas: 1   
  selector:   
    matchLabels:   
      app: nginx   
  template:   
    metadata:   
      labels:   
        app: nginx   
    spec:   
      containers:   
        - name: nginx   
          image: nginx   
          volumeMounts:   
            - mountPath: "/usr/share/nginx/html"   
              name: ceph-storage   
      volumes:   
        - name: ceph-storage   
          persistentVolumeClaim:   
            claimName: ceph-pvc  
**Monitor and Manage**   
**Use:**   
ceph status – check cluster health   
ceph df – view used space   
Or check the dashboard from Rook-Ceph for a UI.   
  
  
**So What Happens When You Apply This?CephCluster**   
  
Rook-Ceph Operator reads this resource (CephCluster).   
It starts to deploy Ceph components:   
MONs (monitors) — for cluster quorum   
MGR (manager) — for dashboard, metrics, etc.   
OSDs (Object Storage Daemons) — where actual data is stored   
  
It uses every node and every disk (that it can find and is allowed to use).   
Sets up a full Ceph cluster inside your OpenShift environment.   
  
**Caution (in real world):**   
Don't use useAllDevices: true in production unless you are sure which disks Ceph will use — it could wipe data.   
Usually, you define exactly which devices or nodes Ceph should use instead.   
  
  
**If You Want Block Storage (RBD – RADOS Block Device)   
What You Get:**   
Each pod gets its own private volume (like a virtual disk).   
Used for: databases, apps that need ReadWriteOnce storage.   
  
**Changes Needed:**   
Keep the CephCluster YAML as-is.   
Create a StorageClass that uses the RBD CSI driver.   
  
**Example:**   
apiVersion: [storage.k8s.io/v1](https://storage.k8s.io/v1)   
kind: StorageClass   
metadata:   
  name: rook-ceph-block   
provisioner: [rook-ceph.rbd.csi.ceph.com](https://rook-ceph.rbd.csi.ceph.com/)   
parameters:   
  clusterID: rook-ceph   
  pool: replicapool   
  imageFormat: "2"   
  imageFeatures: layering   
  [csi.storage.k8s.io/provisioner-secret-name](https://csi.storage.k8s.io/provisioner-secret-name): rook-csi-rbd-provisioner   
  [csi.storage.k8s.io/provisioner-secret-namespace](https://csi.storage.k8s.io/provisioner-secret-namespace): rook-ceph

[csi.storage.k8s.io/node-stage-secret-name](https://csi.storage.k8s.io/node-stage-secret-name): rook-csi-rbd-node   
  [csi.storage.k8s.io/node-stage-secret-namespace](https://csi.storage.k8s.io/node-stage-secret-namespace): rook-ceph   
reclaimPolicy: Delete   
allowVolumeExpansion: true   
volumeBindingMode: Immediate

**If You Want Shared Storage (CephFS)**   
**What You Get:**   
Multiple pods can read/write to the same volume at the same time.   
Used for: web servers, media processing, shared configs, etc.   
  
**Changes Needed:**   
Keep the CephCluster YAML as-is.   
Also deploy a CephFilesystem resource (to define the CephFS).   
Create a StorageClass that uses the CephFS CSI driver.   
  
**Example: CephFilesystem (needed for CephFS)**   
apiVersion: [ceph.rook.io/v1](https://ceph.rook.io/v1)   
kind: CephFilesystem   
metadata:   
  name: myfs   
  namespace: rook-ceph   
spec:   
  metadataPool:   
    replicated:   
      size: 3   
  dataPools:   
    - replicated:   
        size: 3   
  metadataServer:   
    activeCount: 1   
    activeStandby: true  
  
**Example: StorageClass for CephFS**   
apiVersion: [storage.k8s.io/v1](https://storage.k8s.io/v1)   
kind: StorageClass   
metadata:   
  name: rook-cephfs   
provisioner: [rook-ceph.cephfs.csi.ceph.com](https://rook-ceph.cephfs.csi.ceph.com/)   
parameters:   
  clusterID: rook-ceph   
  fsName: myfs   
  pool: myfs-data0   
  [csi.storage.k8s.io/provisioner-secret-name](https://csi.storage.k8s.io/provisioner-secret-name): rook-csi-cephfs-provisioner   
  [csi.storage.k8s.io/provisioner-secret-namespace](https://csi.storage.k8s.io/provisioner-secret-namespace): rook-ceph   
  [csi.storage.k8s.io/node-stage-secret-name](https://csi.storage.k8s.io/node-stage-secret-name): rook-csi-cephfs-node   
  [csi.storage.k8s.io/node-stage-secret-namespace](https://csi.storage.k8s.io/node-stage-secret-namespace): rook-ceph   
reclaimPolicy: Delete   
allowVolumeExpansion: true   
volumeBindingMode: Immediate

**so:**   
CephCluster YAML = same in both cases.   
For block storage: Use RBD StorageClass.   
For shared storage: Add CephFilesystem + use CephFS StorageClass.   
  
  
**What Happens When a Node’s Storage is Full?**   
**If the Node is Running a Ceph OSD (Object Storage Daemon):   
Ceph OSD fills up:**   
Ceph stops writing to that OSD.   
Cluster enters a HEALTH\_WARN or HEALTH\_ERR state.   
Data rebalancing or recovery may fail.   
  
**Cluster May Get Stuck:**If multiple OSDs are full, Ceph can’t meet replication rules (e.g., 3 copies).   
Writes are blocked cluster-wide, even if some nodes have space left.   
  
**Rook-Ceph Logs and Metrics:**   
You’ll see messages like:   
no space left on device   
OSD nearfull or full

**how to add OSDs (Object Storage Daemons) to a running Rook-Ceph cluster in OpenShift   
What Is an OSD?**   
An OSD is a Ceph process that stores data on a disk.   
Each OSD is tied to a specific block device (like /dev/sdb).   
More OSDs = more capacity and better performance.   
  
**Prerequisites   
Before adding OSDs:**   
The node must already be labeled for Ceph (or useAllNodes must be true).   
You must have new, unformatted raw block devices on your node (e.g., /dev/sdb, /dev/sdc).   
The Rook-Ceph Operator must already be running.   
  
**Method 1: Let Rook Auto-Discover New Devices (If useAllDevices: true is already set)**   
**If your CephCluster looks like this:**   
storage:   
useAllNodes: true   
useAllDevices: true

**Then Rook will:**   
Automatically detect any new raw block device attached to existing nodes.   
Automatically provision them as new OSDs.   
  
**Method 2: Manually Specify Which Devices to Use**   
If you want to control exactly which devices become OSDs, you can edit the CephCluster spec and use nodes section.   
  
storage:   
  useAllNodes: false   
  nodes:   
    - name: worker-node-1   
      devices:   
        - name: "sdb"   
    - name: worker-node-2   
      devices:   
        - name: "sdc"  
  
  
**where:**name = name of the node (get with oc get nodes)   
[devices.name](https://web.telegram.org/a/devices.name) = device name on that node (e.g., /dev/sdb → "sdb")

**Apply the Updated YAML:**   
oc apply -f ceph-cluster.yaml   
  
**Rook will then:**   
Deploy new OSD pods on those nodes.   
Format and initialize the devices.   
Add them to the Ceph cluster.