For provisioning block storage in OpenShift Data Foundation (ODF) using Ceph, you typically use the ocs-storagecluster-ceph-rbd StorageClass. The provisioner for this StorageClass is openshift-storage.rbd.csi.ceph.com.

Here's a summary:

* **StorageClass**: ocs-storagecluster-ceph-rbd
* **Provisioner**: openshift-storage.rbd.csi.ceph.com

This setup leverages the Ceph RADOS Block Device (RBD) as the backend storage for block storage provisioning in ODF.

NooBaa is a data management and storage platform that provides a unified approach to managing data across multiple cloud and on-premises environments. It is particularly useful for Kubernetes environments and can act as an object storage provider, allowing applications to store and retrieve data in a scalable and efficient manner.

Steps to Set Up NooBaa as a Storage Provisioner in Kubernetes

1. Prerequisites:

- A running Kubernetes cluster.

- `kubectl` configured to interact with your cluster.

2. Install the NooBaa Operator:

NooBaa uses an Operator to manage its deployment on Kubernetes. Install it using the following commands:

kubectl create -f https://github.com/noobaa/noobaa-operator/releases/download/v5.9.0/noobaa-operator.yaml

3. Create a NooBaa Instance:

After the operator is installed, create a NooBaa instance by applying a custom resource definition (CRD):

kubectl create -f https://github.com/noobaa/noobaa-operator/releases/download/v5.9.0/noobaa-cr.yaml

4. Verify the Installation:

Check the status of the NooBaa installation:

kubectl get noobaa

kubectl get pods -n noobaa

5. Access NooBaa Management Console:

NooBaa provides a web-based management console. You can access it by setting up a port-forward:

kubectl port-forward service/noobaa-mgmt 30000:8080 -n noobaa

Then, navigate to `http://localhost:30000` in your web browser.

6. Create a StorageClass:

Define a StorageClass to use NooBaa as the provisioner:

yaml

apiVersion: storage.k8s.io/v1

kind: StorageClass

metadata:

name: noobaa-storage

provisioner: noobaa.io/obc

parameters:

bucketclass: noobaa-default-bucket-class

Apply the StorageClass:

kubectl apply -f noobaa-storageclass.yaml

7. Create a PersistentVolumeClaim (PVC):

Create a PVC that uses the NooBaa StorageClass:

yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: noobaa-pvc

spec:

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 5Gi

storageClassName: noobaa-storage

Apply the PVC:

kubectl apply -f noobaa-pvc.yaml

8. Verify PVC and Pod Usage:

Verify that the PVC is bound and create a Pod that uses this PVC:

yaml

apiVersion: v1

kind: Pod

metadata:

name: noobaa-test-pod

spec:

containers:

- name: noobaa-test-container

image: busybox

command: ["/bin/", "-c", "while true; do echo hello; sleep 10;done"]

volumeMounts:

- mountPath: "/mnt/noobaa"

name: noobaa-volume

volumes:

- name: noobaa-volume

persistentVolumeClaim:

claimName: noobaa-pvc

Apply the Pod configuration:

kubectl apply -f noobaa-test-pod.yaml

9. Verify Data Persistence:

Exec into the Pod to verify that data can be written and read from the mounted volume:

kubectl exec -it noobaa-test-pod -- /bin/

cd /mnt/noobaa

echo "Hello NooBaa" > testfile.txt

cat testfile.txt

These steps will help you set up NooBaa as a storage provisioner in your Kubernetes cluster, enabling you to leverage NooBaa's capabilities for object storage.