Hybrid cloud with provider-managed components

NetApp public and hybrid cloud solutions

NetApp
February 04, 2026

Table of Contents

Hybrid cloud with provider-managed components

NetApp Solution with Managed Red Hat OpenShift Container platform workloads

Customers may be "born in the cloud" or may be at a point in their modernization journey when they are ready to move some select workloads or all workloads from their data centers to the cloud. They may choose to use provider-managed OpenShift containers and provider-managed NetApp storage in the cloud for running their workloads. They should plan and deploy the Managed Red Hat OpenShift container clusters in the cloud for a successful production-ready environment for their container workloads. NetApp provides fully managed storage offerings for Managed Red Hat solutions in all three leading public clouds.

Amazon FSx for NetApp ONTAP (FSx ONTAP)

FSx ONTAP delivers data protection, reliability, and flexibility for container deployments in AWS. Trident serves as the dynamic storage provisioner to consume the persistent FSx ONTAP storage for customers' stateful applications.

As ROSA can be deployed in HA mode with control plane nodes spread across multiple availability zones, FSx ONTAP can also be provisioned with Multi-AZ option which provides high availability and protect against AZ failures.

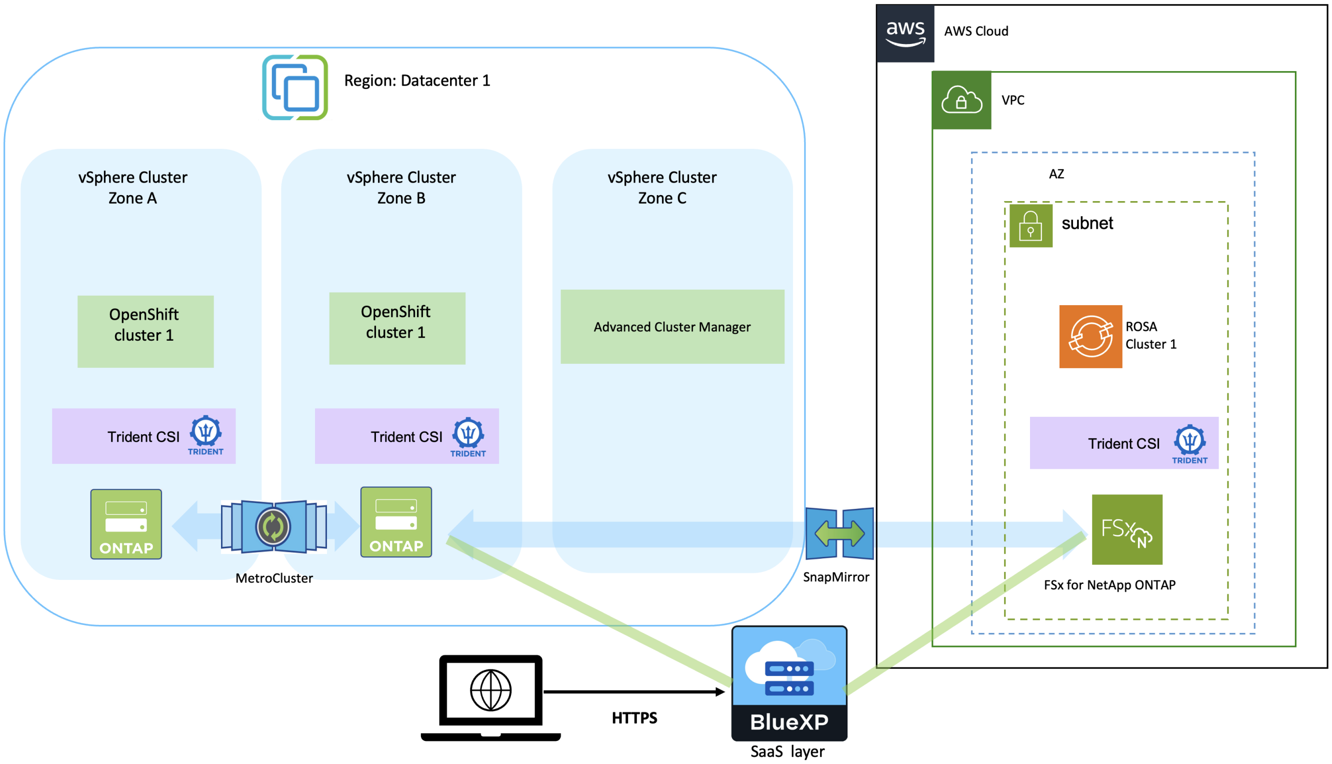
Google Cloud NetApp Volumes

Red Hat OpenShift Dedicated is a fully managed application platform that enables you to quickly build, deploy, and scale applications across the hybrid cloud. Google Cloud NetApp Volumes provides persistent volumes bringing the full suite of ONTAP’s enterprise data management capabilities to OpenShift deployments in Google Cloud.

Deploy and configure the Managed Red Hat OpenShift Container platform on AWS

This section describes a high-level workflow of setting up the Managed Red Hat OpenShift clusters on AWS(ROSA). It shows the use of managed Amazon FSx for NetApp ONTAP (FSx ONTAP) as the storage backend by Trident to provide persistent volumes. Details are provided about the deployment of FSx ONTAP on AWS using BlueXP. Also, details are provided about the use of BlueXP and OpenShift GitOps (Argo CD) to perform data protection and migration activities for the stateful applications on ROSA clusters.

Here is a diagram that depicts the ROSA clusters deployed on AWS and using FSx ONTAP as the backend storage.



This solution was verified by using two ROSA clusters in two VPCs in AWS. Each ROSA cluster was integrated with FSx ONTAP using Trident. There are several ways of deploying ROSA clusters and FSx ONTAP in AWS. This high-level description of the setup provides documentation links for the specific method that was used. You can refer to the other methods in the relevant links provided in the resources section.

The setup process can be broken down into the following steps:

Install ROSA clusters

- Create two VPCs and set up VPC peering connectivity between the VPCs.
- Refer here for instructions to install ROSA clusters.

Install FSx ONTAP

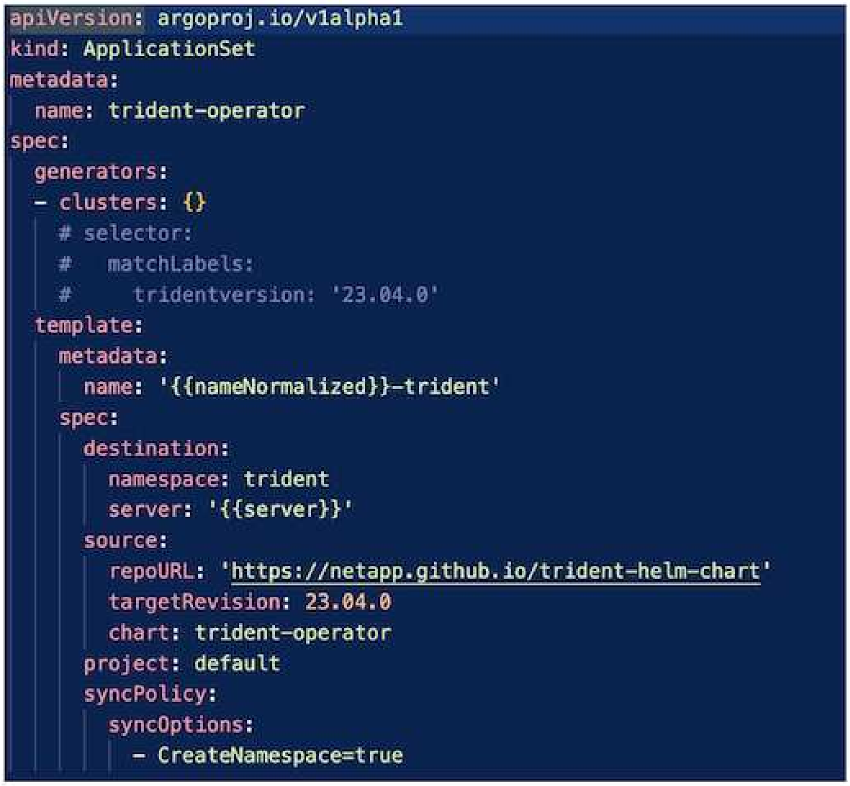
- Install FSx ONTAP on the VPCs from BlueXP.
  Refer here for BlueXP account creation and to get started.
  Refer here for installing FSx ONTAP.
  Refer here for creating a connector in AWS to manage the FSx ONTAP.
- Deploy FSx ONTAP using AWS.
  Refer here for deployment using AWS console.

Install Trident on ROSA clusters (using Helm chart)

- Use Helm chart to install Trident on ROSA clusters.
  Refer to the documentation xref:./openshift/ here.

Integration of FSx ONTAP with Trident for ROSA clusters

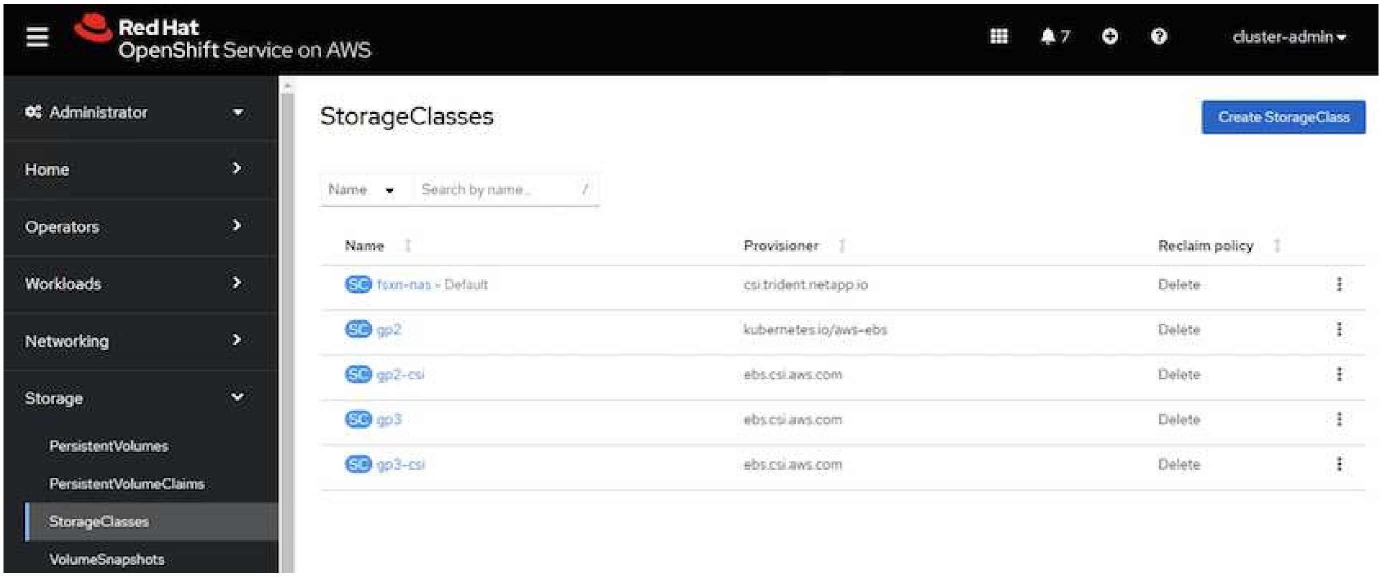
OpenShift GitOps can be utilized to deploy Trident CSI to all managed clusters as they get registered to ArgoCD using ApplicationSet.



```
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: trident-operator
spec:
  generators:
  - clusters: {}
    # selector:
    #   matchLabels:
    #     tridentversion: '23.04.0'
  template:
    metadata:
      name: '{{nameNormalized}}-trident'
    spec:
      destination:
        namespace: trident
        server: '{{server}}'
      source:
        repoURL: 'https://netapp.github.io/trident-helm-chart'
        targetRevision: 23.04.0
        chart: trident-operator
      project: default
      syncPolicy:
        syncOptions:
          - CreateNamespace=true
```

Create backend and storage classes using Trident (for FSx ONTAP)

- Refer here for details about creating backend and storage class.
- Make the storage class created for FsxN with Trident CSI as default from OpenShift Console. See screenshot below:



StorageClasses

Name	Provisioner	Reclaim policy
SC fsxn-nas - Default	csi.trident.netapp.io	Delete
SC gp2	kubernetes.io/aws-ebs	Delete
SC gp2-csi	ebs.csi.aws.com	Delete
SC gp3	ebs.csi.aws.com	Delete
SC gp3-csi	ebs.csi.aws.com	Delete

Deploy an application using OpenShift GitOps (Argo CD)

- Install OpenShift GitOps operator on the cluster. Refer to instructions here.
- SetUp a new Argo CD instance for the cluster. Refer to instructions here.

Open the console of Argo CD and deploy an app.

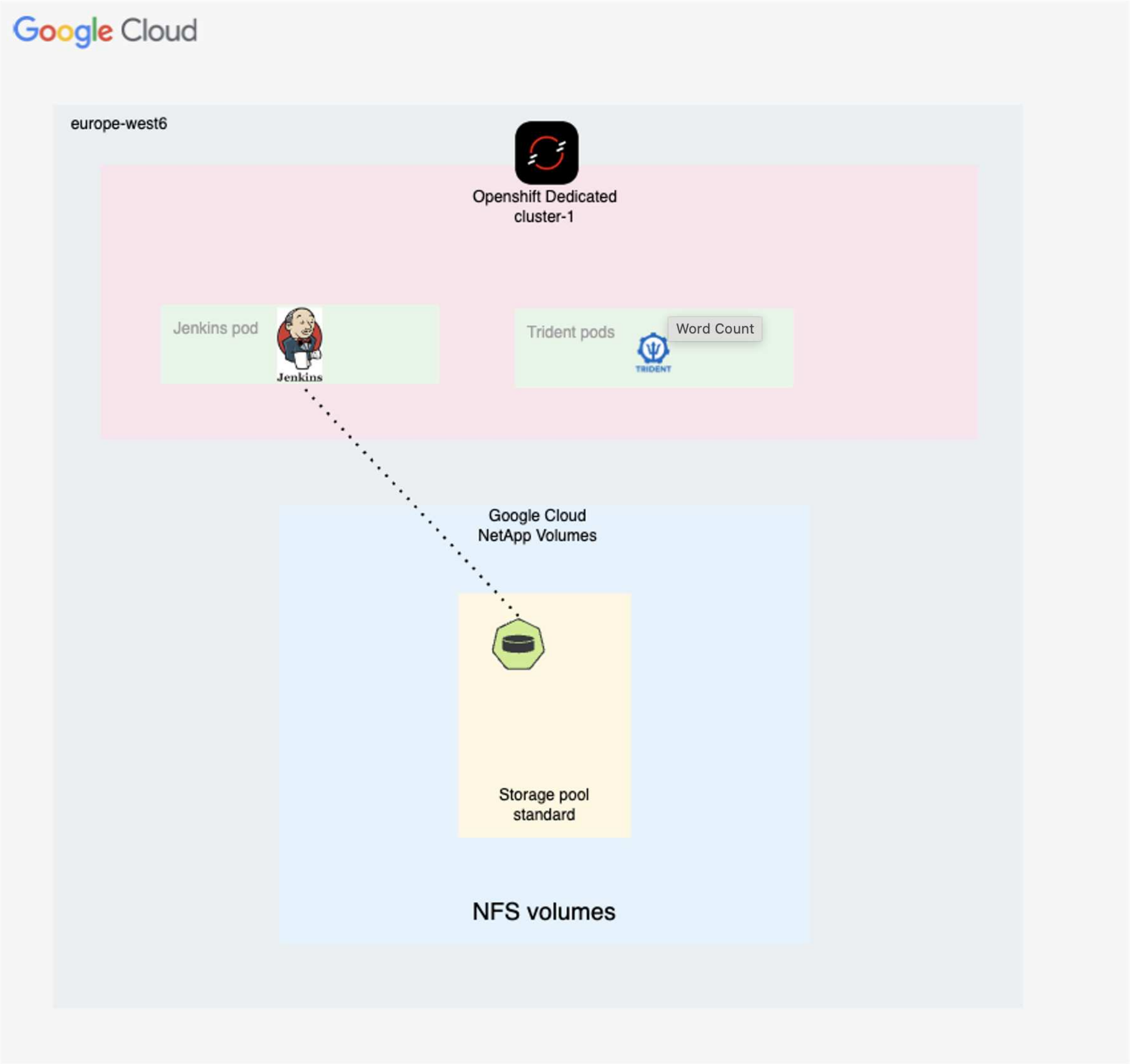
As an example, you can deploy a Jenkins App using Argo CD with a Helm Chart.
When creating the application, the following details were provided:
Project: default
cluster: 'https://kubernetes.default.svc' (without the quotes)
Namespace: Jenkins
The url for the Helm Chart: 'https://charts.bitnami.com/bitnami' (without the quotes)

Helm Parameters:
global.storageClass: fsxn-nas

Deploy and configure OpenShift Dedicated on Google Cloud with Google Cloud NetApp Volumes

This section describes a high-level workflow of setting up OpenShift Dedicated (OSD) clusters on the Google Cloud platform. It shows NetApp Trident using Google Cloud NetApp Volumes as the storage backend to provide persistent volumes for stateful applications running with Kubernetes.

Here is a diagram that depicts an OSD cluster deployed on Google Cloud and using NetApp Volumes as the backend storage.



The setup process can be broken down into the following steps:

Install OSD clusters in Google Cloud

- If you wish to use an existing VPC for the cluster, you must create the VPC, two subnets, a cloud router, and two GCP cloud NATs for the OSD cluster. Refer here for instructions.
- Refer here for instructions to install OSD clusters on GCP using the Customer Cloud Subscription (CCS) billing model. OSD is also included on Google Cloud Marketplace. A video showing how to install OSD using the Google Cloud Marketplace solution is located here.

Enable Google Cloud NetApp Volumes

- Refer here for information on setting up access to Google Cloud NetApp Volumes. Follow all the steps up to and including
- Create a storage pool. Refer here for information on how to set up a storage pool on Google Cloud NetApp Volumes. Volumes for the stateful Kubernetes applications running on OSD will be created within the storage pool.

Install Trident on OSD clusters (using Helm chart)

- Use a Helm chart to install Trident on OSD clusters. Refer here for instructions on how to install the Helm Chart. The helm chart may be found here.

Integration of NetApp Volumes with NetApp Trident for OSD clusters

Create backend and storage classes using Trident (for Google Cloud NetApp Volumes)

- Refer here for details about creating backend.
- If any of the current storage classes in kubernetes are marked as default, remove that annotation by editing the storage class.
- Create at least one storage class for NetApp volumes with the Trident CSI provisioner. Make exactly one of the storage classes the default using an annotation. This will allow a PVC to use this storage class when it is not explicitly called out in the PVC manifest. An example with the annotation is shown below.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-standard-k8s
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

Deploy an application using OpenShift GitOps (Argo CD)

- Install OpenShift GitOps operator on the cluster. Refer to instructions here.
- SetUp a new Argo CD instance for the cluster. Refer to instructions here.

Open the console of Argo CD and deploy an app.
As an example, you can deploy a Jenkins App using Argo CD with a Helm Chart.
When creating the application, the following details were provided:
Project: default
cluster: 'https://kubernetes.default.svc' (without the quotes)
Namespace: Jenkins
The url for the Helm Chart: 'https://charts.bitnami.com/bitnami' (without the quotes)

Data protection

This page shows the data protection options for Managed Red Hat OpenShift on AWS (ROSA) clusters using Astra Control Service. Astra Control Service (ACS) provides an easy-to-use graphical user-interface with which you can add clusters, define applications running on them, and perform application aware data management activities. ACS functions can also be accessed using an API that allows for automation of workflows.

Powering Astra Control (ACS or ACC) is NetApp Trident. Trident integrates several types of Kubernetes clusters such as Red Hat OpenShift, EKS, AKS, SUSE Rancher, Anthos etc., with various flavors of NetApp ONTAP storage such as FAS/AFF, ONTAP Select, CVO, Google Google Cloud NetApp Volumes, Azure NetApp Files and Amazon FSx ONTAP.

This section provides details for the following data protection options using ACS:

- A video showing Backup and Restore of a ROSA application running in one region and restoring to another region.
- A video showing Snapshot and Restore of a ROSA application.
- Step-by-step details of installing a ROSA cluster, Amazon FSx ONTAP, using NetApp Trident to integrate with storage backend, installing a postgresql application on ROSA cluster, using ACS to create a snapshot of the application and restoring the application from it.
- A blog showing step-by-step details of creating and restoring from a snapshot for a mysql application on a ROSA cluster with FSx ONTAP using ACS.

Backup/Restore from Backup

The following video shows the backup of a ROSA application running in one region and restoring to another region.

FSx NetApp ONTAP for Red Hat OpenShift Service on AWS

Snapshot/Restore from snapshot

The following video shows taking a snapshot of a ROSA application and restoring from the snapshot after.

Snapshot/Restore for Applications on Red Hat OpenShift Service on AWS (ROSA)clusters with Amazon FSx ONTAP storage

Blog

- Using Astra Control Service for data management of apps on ROSA clusters with Amazon FSx storage

Step-by-Step Details to create snapshot and restore from it

Prerequisite setup

- AWS account
- Red Hat OpenShift account
- IAM user with appropriate permissions to create and access ROSA cluster
- AWS CLI
- ROSA CLI
- OpenShift CLI(oc)
- VPC with subnets and appropriate gateways and routes
- ROSA Cluster installed into the VPC
- Amazon FSx ONTAP created in the same VPC
- Access to the ROSA cluster from OpenShift Hybrid Cloud Console

Next Steps

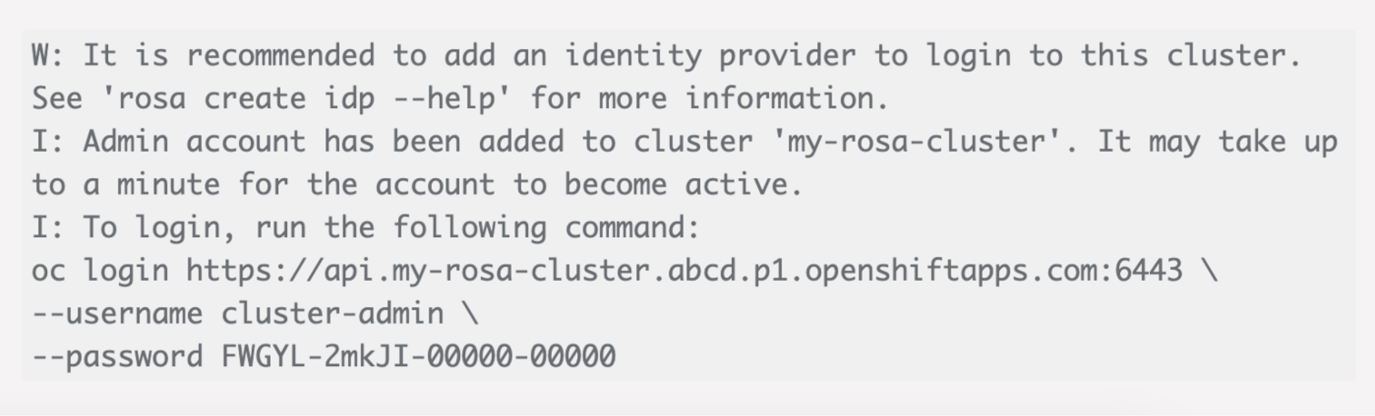
1. Create an admin user and login to the cluster.
2. Create a kubeconfig file for the cluster.
3. Install Trident on the cluster.
4. Create a backend, storage class and snapshot class configuration using the Trident CSI provisioner.
5. Deploy a postgresql application on the cluster.
6. Create a database and add a record.
7. Add the cluster into ACS.
8. Define the application in ACS.
9. Create a snapshot using ACS.
10. Delete the database in the postgresql application.
11. Restore from a snapshot using ACS.
12. Verify your app has been restored form the snapshot.

1. Create an admin user and login to the cluster

Access the ROSA cluster by creating an admin user with the following command : (You need to create an admin user only if you did not create one at the time of installation)

```
rosa create admin --cluster=<cluster-name>
```

The command will provide an output that will look like the following. Login to the cluster using the `oc login` command provided in the output.



```
W: It is recommended to add an identity provider to login to this cluster.
See 'rosa create idp --help' for more information.
I: Admin account has been added to cluster 'my-rosa-cluster'. It may take up
to a minute for the account to become active.
I: To login, run the following command:
oc login https://api.my-rosa-cluster.abcd.p1.openshiftapps.com:6443 \
--username cluster-admin \
--password FWGYL-2mkJI-00000-00000
```

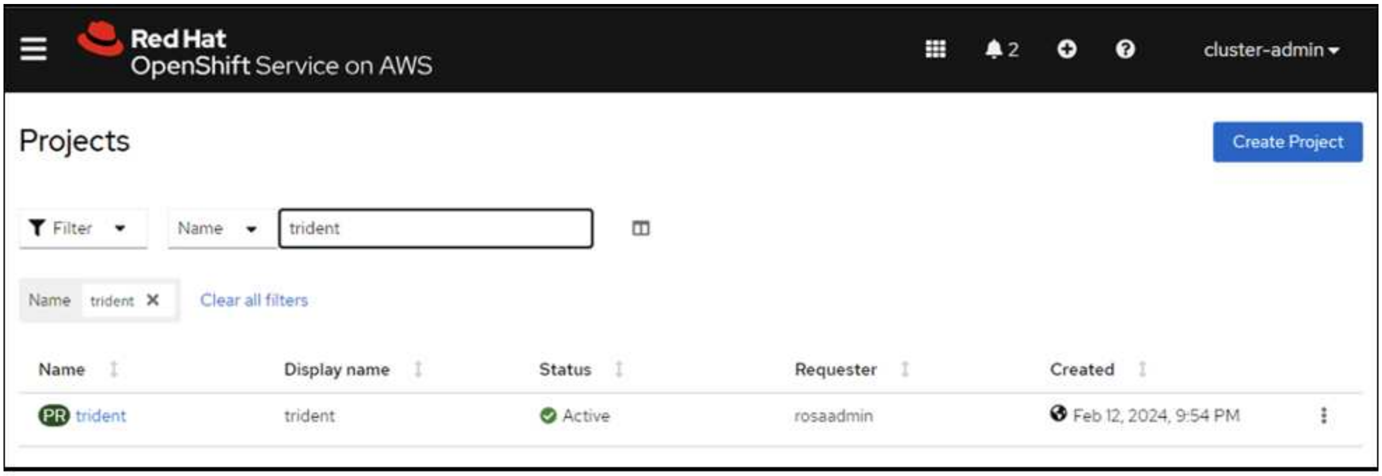
> ⓘ You can also login to the cluster using a token. If you already created an admin-user at the time of cluster creation, you can login to the cluster from the Red Hat OpenShift Hybrid Cloud console with the admin-user credentials. Then by clicking on the top right corner where it displays the name of the logged in user, you can obtain the `oc login` command (token login) for the command line.

2. Create a kubeconfig file for the cluster

Follow the procedures here to create a kubeconfig file for the ROSA cluster. This kubeconfig file will be used later when you add the cluster into ACS.

3. Install Trident on the cluster

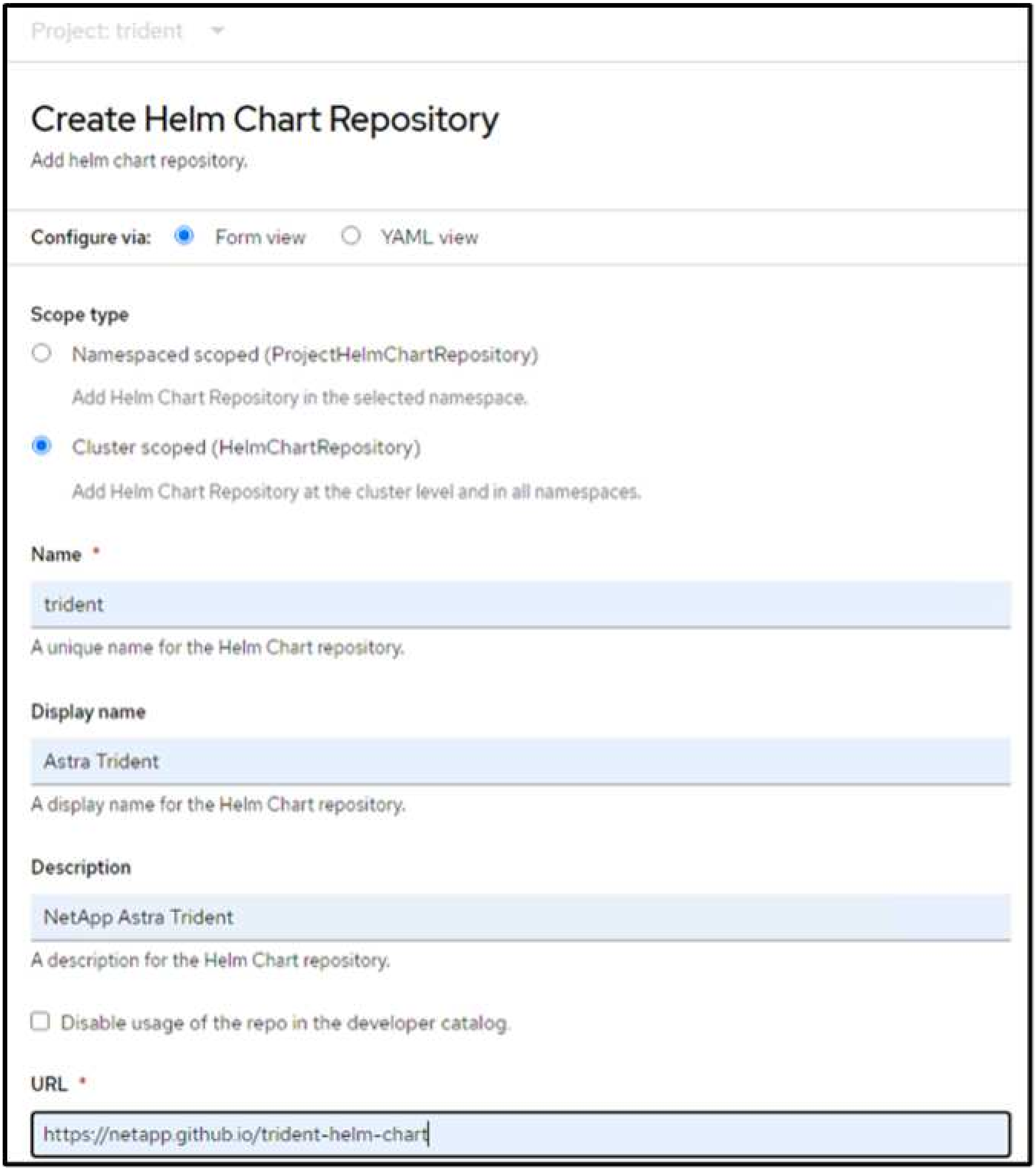
Install Trident (latest version) on the ROSA cluster. To do this, you can follow any one of the procedures given here. To install Trident using helm from the console of the cluster, first create a project called Trident.



Name: trident

Name	Display name	Status	Requester	Created
PR trident	trident	Active	rosaadmin	Feb 12, 2024, 9:54 PM

Then from the Developer view, create a Helm chart repository. For the URL field use `'https://netapp.github.io/trident-helm-chart'`. Then create a helm release for Trident operator.



Project: trident

Create Helm Chart Repository

Add helm chart repository.

Configure via: ● Form view ○ YAML view

Scope type

○ Namespaced scoped (ProjectHelmChartRepository)

Add Helm Chart Repository in the selected namespace.

● Cluster scoped (HelmChartRepository)

Add Helm Chart Repository at the cluster level and in all namespaces.

Name *

trident

A unique name for the Helm Chart repository.

Display name

Astra Trident

A display name for the Helm Chart repository.

Description

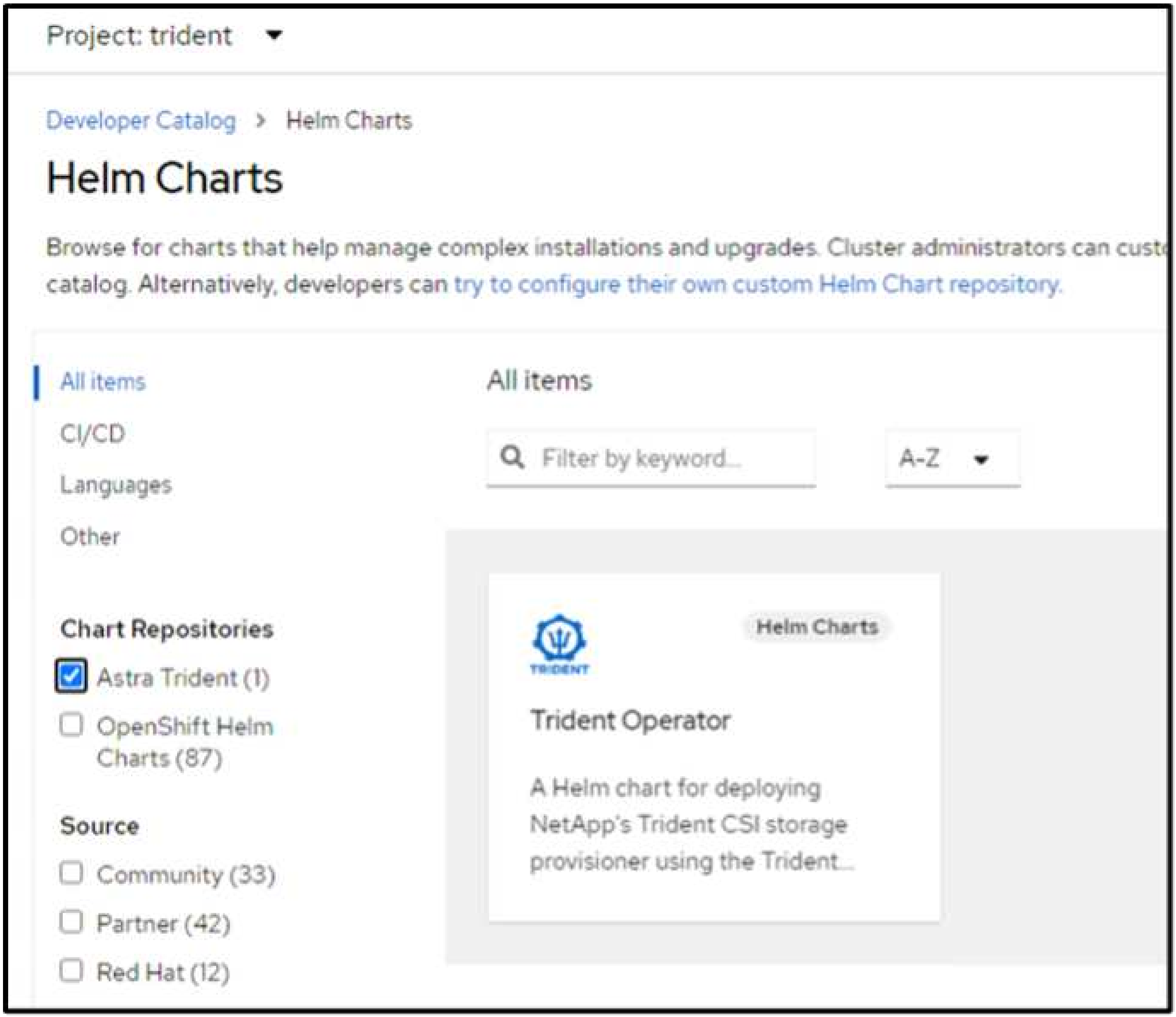
NetApp Astra Trident

A description for the Helm Chart repository.

□ Disable usage of the repo in the developer catalog.

URL *

https://netapp.github.io/trident-helm-chart



Project: trident ▾

Developer Catalog > Helm Charts

Helm Charts

Browse for charts that help manage complex installations and upgrades. Cluster administrators can cust
catalog. Alternatively, developers can try to configure their own custom Helm Chart repository.

All items
CI/CD
Languages
Other

Chart Repositories

☑ Astra Trident (1)
□ OpenShift Helm Charts (87)

Source

□ Community (33)
□ Partner (42)
□ Red Hat (12)

All items

Filter by keyword...

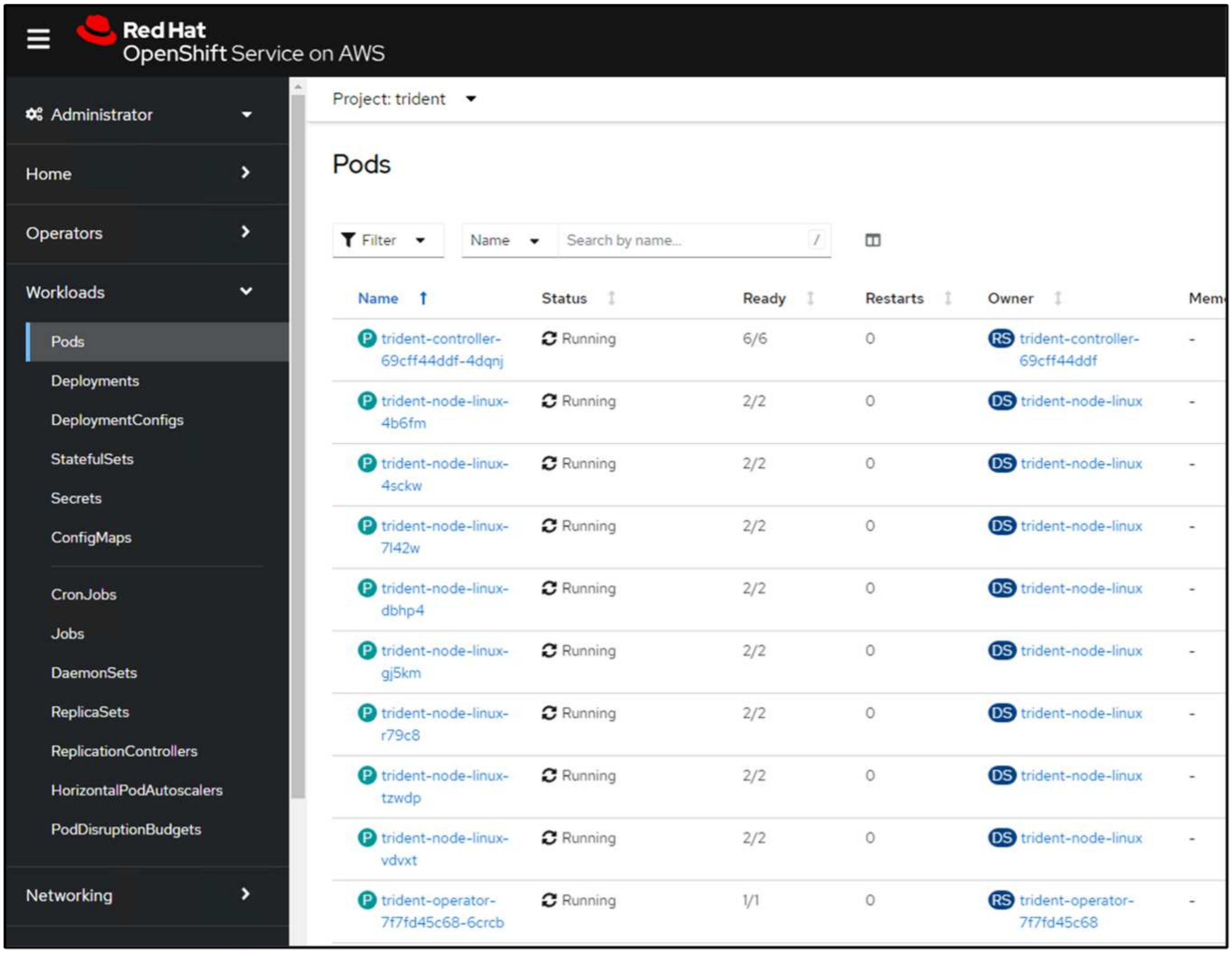
A-Z ▾

Helm Charts

Trident Operator

A Helm chart for deploying NetApp's Trident CSI storage provisioner using the Trident...

Verify all trident pods are running by going back to the Administrator view on the console and selecting pods in the trident project.



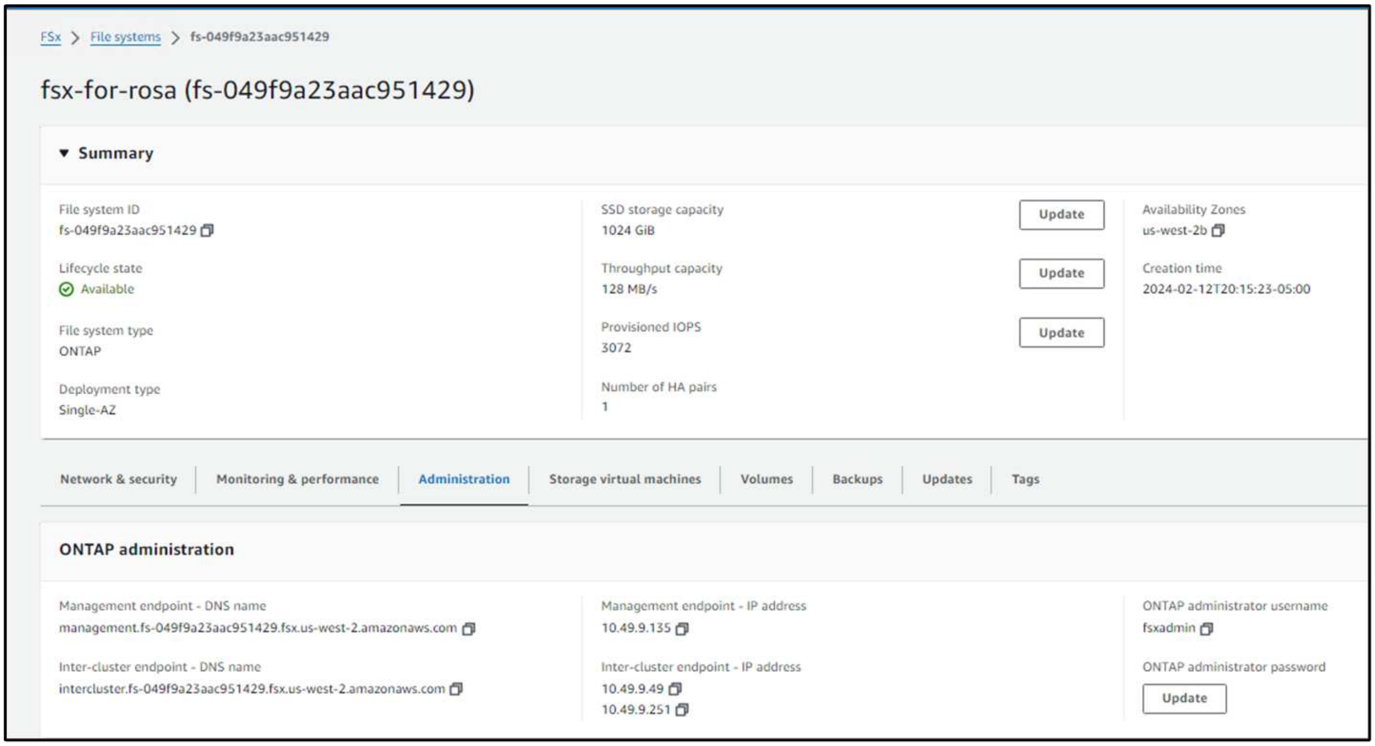
Pods

Name	Status	Ready	Restarts	Owner	Mem
P trident-controller-69cff44ddf-4dqnj	Running	6/6	0	RS trident-controller-69cff44ddf	-
P trident-node-linux-4b6fm	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-4sckw	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-7l42w	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-dbhp4	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-gj5km	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-r79c8	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-tzwdp	Running	2/2	0	DS trident-node-linux	-
P trident-node-linux-vdvxt	Running	2/2	0	DS trident-node-linux	-
P trident-operator-7f7fd45c68-6crcb	Running	1/1	0	RS trident-operator-7f7fd45c68	-

4. Create a backend, storage class and snapshot class configuration using the Trident CSI provisioner

Use the yaml files shown below to create a trident backend object, storage class object and the Volumesnapshot object. Be sure to provide the credentials to your Amazon FSx ONTAP file system you created, the management LIF and the vserver name of your file system in the configuration yaml for the backend. To get those details, go to the AWS console for Amazon FSx and select the file system, navigate to the Administration tab. Also, click on update to set the password for the `fsxadmin` user.

> ⓘ You can use the command line to create the objects or create them with the yaml files from the hybrid cloud console.



fsx-for-rosa (fs-049f9a23aac951429)

Summary

File system ID: fs-049f9a23aac951429

Lifecycle state: Available

File system type: ONTAP

Deployment type: Single-AZ

SSD storage capacity: 1024 GiB

Throughput capacity: 128 MB/s

Provisioned IOPS: 3072

Number of HA pairs: 1

Availability Zones: us-west-2b

Creation time: 2024-02-12T20:15:23-05:00

ONTAP administration

Management endpoint - DNS name: management.fs-049f9a23aac951429.fsx.us-west-2.amazonaws.com

Inter-cluster endpoint - DNS name: intercluster.fs-049f9a23aac951429.fsx.us-west-2.amazonaws.com

Management endpoint - IP address: 10.49.9.135

Inter-cluster endpoint - IP address: 10.49.9.49
10.49.9.251

ONTAP administrator username: fsxadmin

ONTAP administrator password

Trident Backend Configuration

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-nas-secret
type: Opaque
stringData:
  username: fsxadmin
  password: <password>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <management lif>
  backendName: ontap-nas
  svm: fsx
  credentials:
    name: backend-tbc-ontap-nas-secret
```

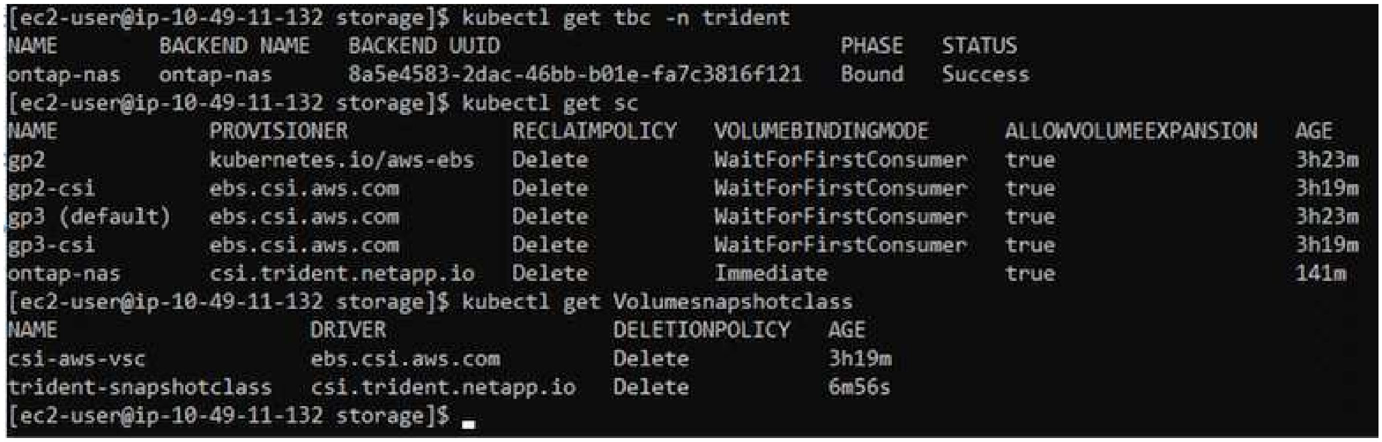
Storage Class

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
```

snapshot class

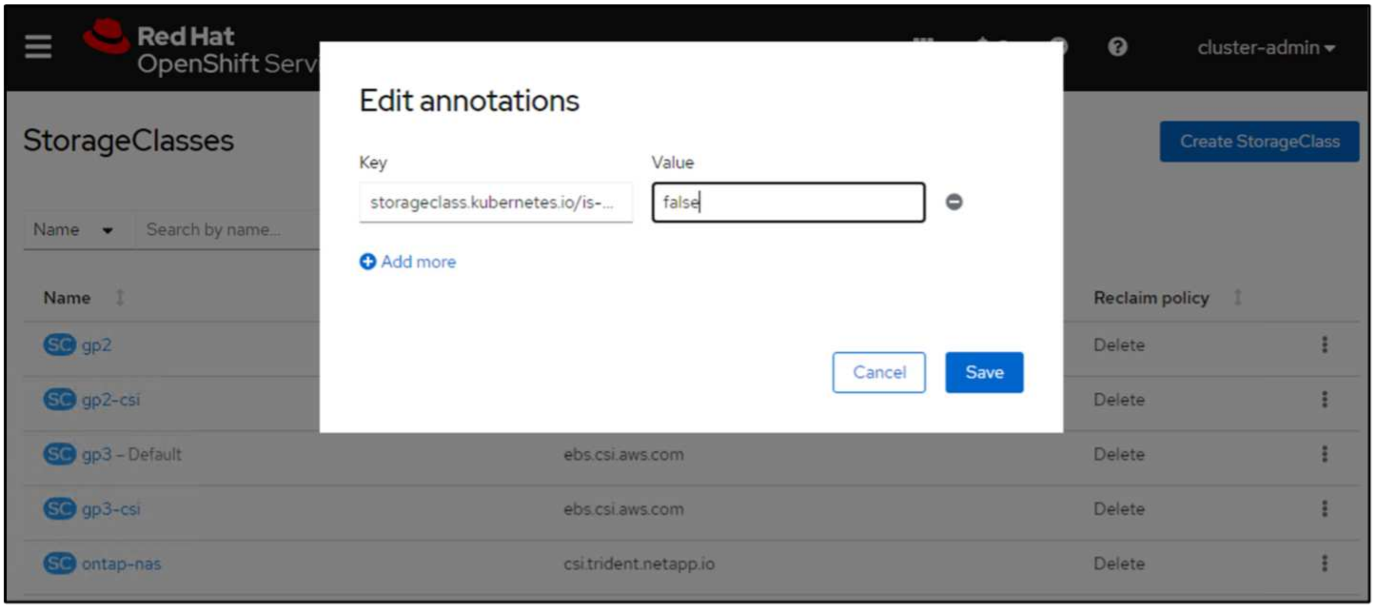
```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: trident-snapshotclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Verify that the backend, storage class and the trident-snapshotclass objects are created by issuing the commands shown below.



```
[ec2-user@ip-10-49-11-132 storage]$ kubectl get tbc -n trident
NAME        BACKEND NAME   BACKEND UUID                           PHASE   STATUS
ontap-nas   ontap-nas      8a5e4583-2dac-46bb-b01e-fa7c3816f121   Bound   Success
[ec2-user@ip-10-49-11-132 storage]$ kubectl get sc
NAME            PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
gp2             kubernetes.io/aws-ebs   Delete          WaitForFirstConsumer   true                   3h23m
gp2-csi         ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   3h19m
gp3 (default)   ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   3h23m
gp3-csi         ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   3h19m
ontap-nas       csi.trident.netapp.io   Delete          Immediate              true                   141m
[ec2-user@ip-10-49-11-132 storage]$ kubectl get Volumesnapshotclass
NAME                    DRIVER                  DELETIONPOLICY   AGE
csi-aws-vsc             ebs.csi.aws.com         Delete           3h19m
trident-snapshotclass   csi.trident.netapp.io   Delete           6m56s
[ec2-user@ip-10-49-11-132 storage]$ 
```

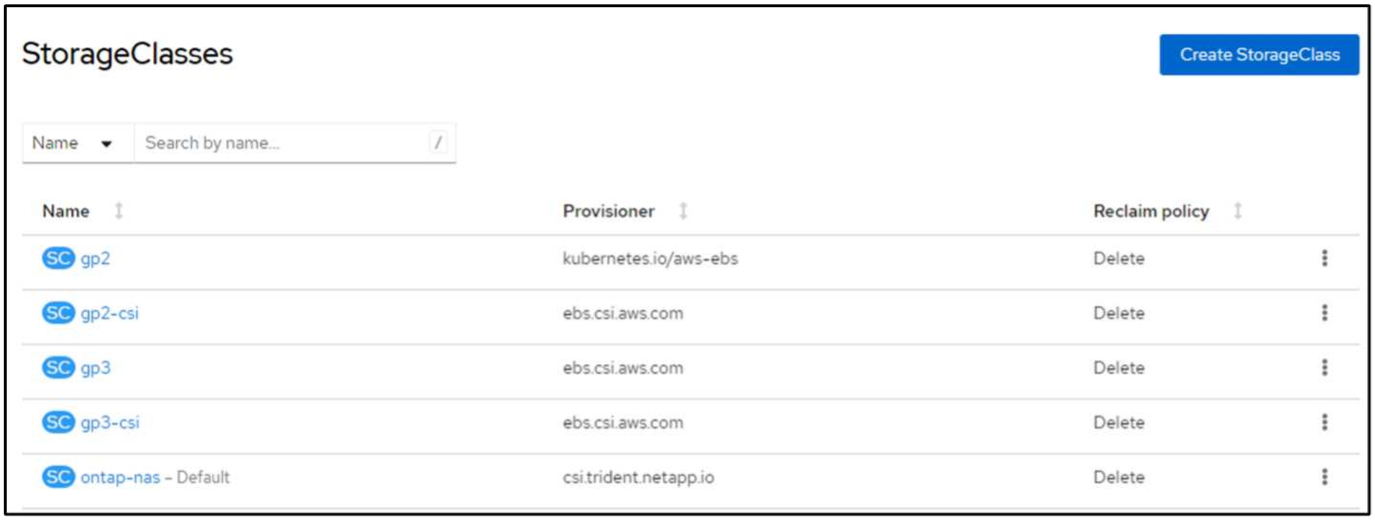
At this time, an important modification you need to make is to set ontap-nas as the default storage class instead of gp3 so that the postgresql app you deploy later can use the default storage class. In the Openshift console of your cluster, under Storage select StorageClasses. Edit the annotation of the current default class to be false and add the annotation storageclass.kubernetes.io/is-default-class set to true for the ontap-nas storage class.



Edit annotations

Key	Value
storageclass.kubernetes.io/is-...	false

Name		Reclaim policy
SC gp2		Delete
SC gp2-csi		Delete
SC gp3 – Default	ebs.csi.aws.com	Delete
SC gp3-csi	ebs.csi.aws.com	Delete
SC ontap-nas	csi.trident.netapp.io	Delete

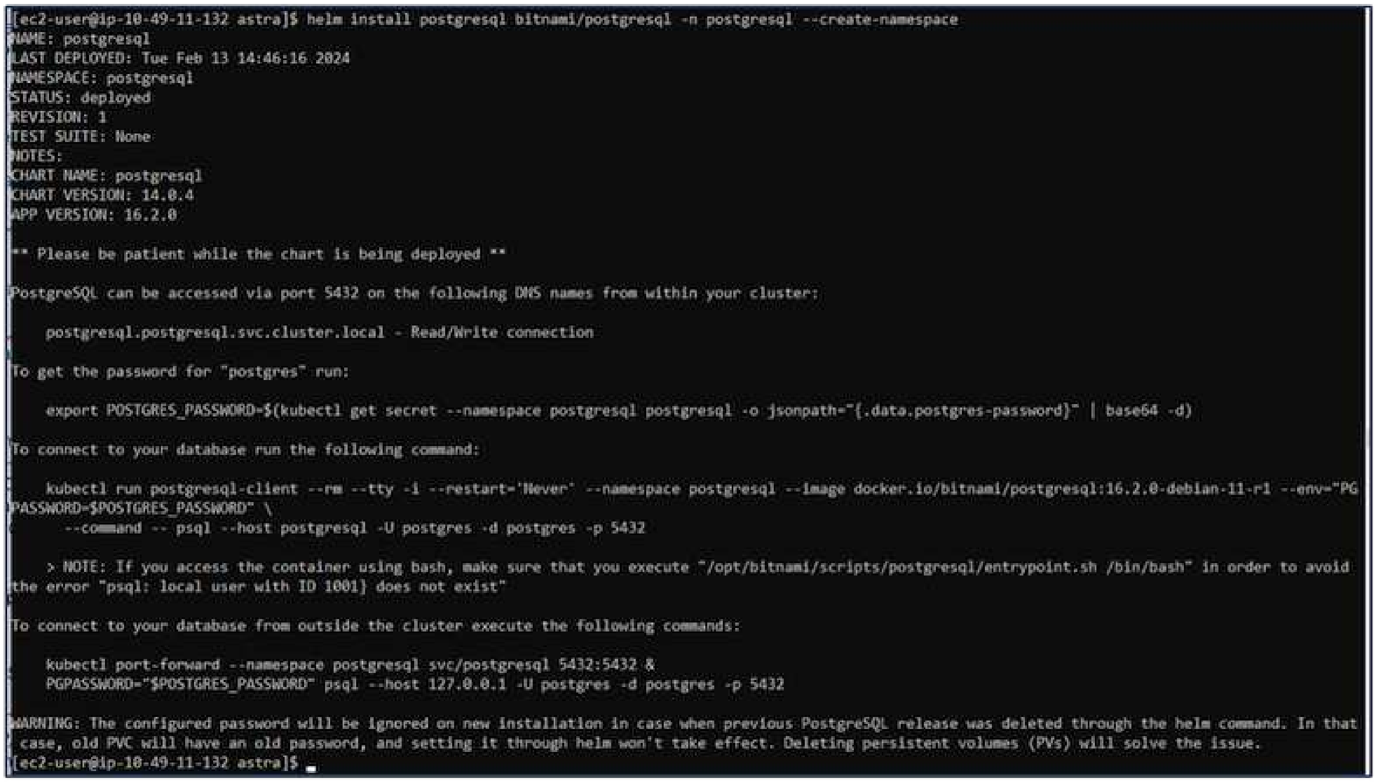

StorageClasses

Name	Provisioner	Reclaim policy
SC gp2	kubernetes.io/aws-ebs	Delete
SC gp2-csi	ebs.csi.aws.com	Delete
SC gp3	ebs.csi.aws.com	Delete
SC gp3-csi	ebs.csi.aws.com	Delete
SC ontap-nas – Default	csi.trident.netapp.io	Delete

5. Deploy a postgresql application on the cluster

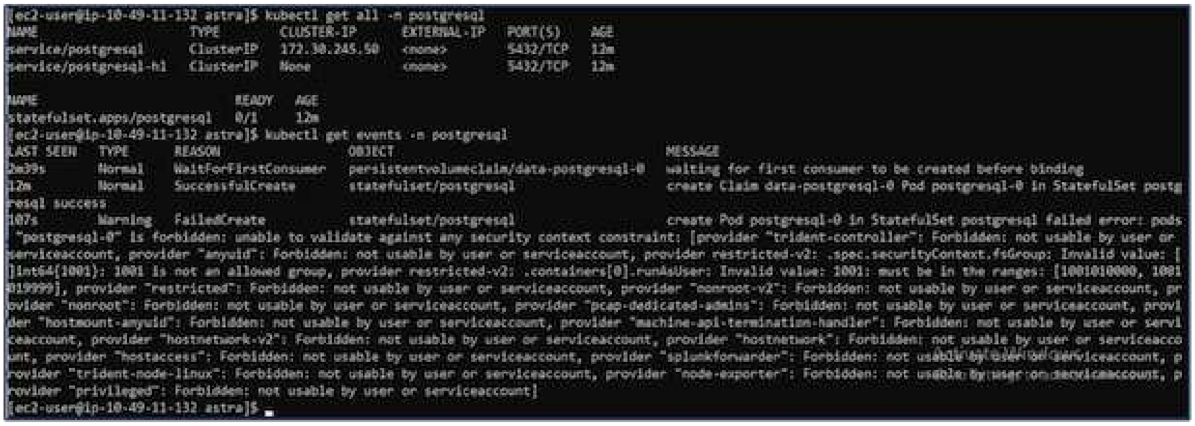
You can deploy the application from the command line as follows:

```
helm install postgresql bitnami/postgresql -n postgresql --create-namespace
```



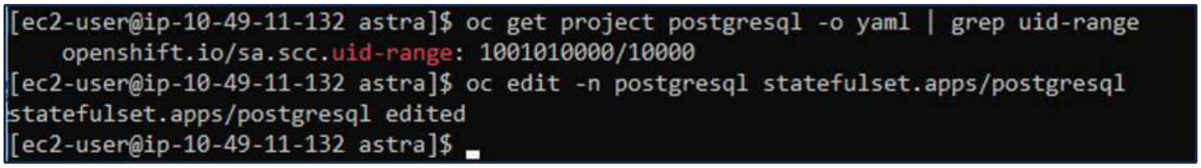
```
[ec2-user@ip-10-49-11-132 astra]$ helm install postgresql bitnami/postgresql -n postgresql --create-namespace
NAME: postgresql
LAST DEPLOYED: Tue Feb 13 14:46:16 2024
NAMESPACE: postgresql
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: postgresql
CHART VERSION: 14.0.4
APP VERSION: 16.2.0

** Please be patient while the chart is being deployed **

PostgreSQL can be accessed via port 5432 on the following DNS names from within your cluster:

    postgresql.postgresql.svc.cluster.local - Read/Write connection

To get the password for "postgres" run:

    export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql -o jsonpath="{.data.postgres-password}" | base64 -d)

To connect to your database run the following command:

    kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:16.2.0-debian-11-r1 --env="PG
PASSWORD=$POSTGRES_PASSWORD" \
      --command -- psql --host postgresql -U postgres -d postgres -p 5432

    > NOTE: If you access the container using bash, make sure that you execute "/opt/bitnami/scripts/postgresql/entrypoint.sh /bin/bash" in order to avoid
the error "psql: local user with ID 1001} does not exist"

To connect to your database from outside the cluster execute the following commands:

    kubectl port-forward --namespace postgresql svc/postgresql 5432:5432 &
    PGPASSWORD="$POSTGRES_PASSWORD" psql --host 127.0.0.1 -U postgres -d postgres -p 5432

WARNING: The configured password will be ignored on new installation in case when previous PostgreSQL release was deleted through the helm command. In that
 case, old PVC will have an old password, and setting it through helm won't take effect. Deleting persistent volumes (PVs) will solve the issue.
[ec2-user@ip-10-49-11-132 astra]$
```

If you do not see the application pods running, then there might be an error caused due to security context constraints.



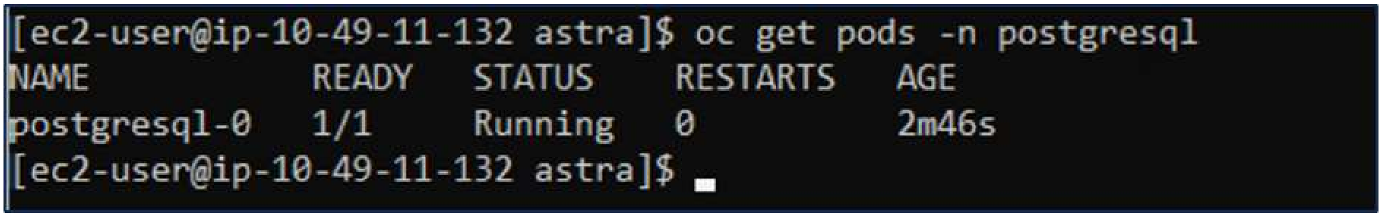
```
[ec2-user@ip-10-49-11-132 astra]$ kubectl get all -n postgresql
NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgresql       ClusterIP   172.30.245.50   <none>        5432/TCP   12m
service/postgresql-hl    ClusterIP   None            <none>        5432/TCP   12m

NAME                          READY   AGE
statefulset.apps/postgresql   0/1     12m
[ec2-user@ip-10-49-11-132 astra]$ kubectl get events -n postgresql
LAST SEEN   TYPE      REASON                  OBJECT                                  MESSAGE
2m39s       Normal    WaitForFirstConsumer    persistentvolumeclaim/data-postgresql-0   waiting for first consumer to be created before binding
12m         Normal    SuccessfulCreate        statefulset/postgresql                  create Claim data-postgresql-0 Pod postgresql-0 in StatefulSet postg
resql success
107s        Warning   FailedCreate            statefulset/postgresql                  create Pod postgresql-0 in StatefulSet postgresql failed error: pods
 "postgresql-0" is forbidden: unable to validate against any security context constraint: [provider "trident-controller": Forbidden: not usable by user or
serviceaccount, provider "anyuid": Forbidden: not usable by user or serviceaccount, provider restricted-v2: .spec.securityContext.fsGroup: Invalid value: [
]int64{1001}: 1001 is not an allowed group, provider restricted-v2: .containers[0].runAsUser: Invalid value: 1001: must be in the ranges: [1001010000, 1001
019999], provider "restricted": Forbidden: not usable by user or serviceaccount, provider "nonroot-v2": Forbidden: not usable by user or serviceaccount, pr
ovider "nonroot": Forbidden: not usable by user or serviceaccount, provider "pcap-dedicated-admins": Forbidden: not usable by user or serviceaccount, provi
der "hostmount-anyuid": Forbidden: not usable by user or serviceaccount, provider "machine-api-termination-handler": Forbidden: not usable by user or servi
ceaccount, provider "hostnetwork-v2": Forbidden: not usable by user or serviceaccount, provider "hostnetwork": Forbidden: not usable by user or serviceacco
unt, provider "hostaccess": Forbidden: not usable by user or serviceaccount, provider "splunkforwarder": Forbidden: not usable by user or serviceaccount, p
rovider "trident-node-linux": Forbidden: not usable by user or serviceaccount, provider "node-exporter": Forbidden: not usable by user or serviceaccount, p
rovider "privileged": Forbidden: not usable by user or serviceaccount]
[ec2-user@ip-10-49-11-132 astra]$
```

Fix the error by editing the `runAsUser` and `fsGroup` fields in `statefuleset.apps/postgresql` object with the uid that is in the output of the `oc get project` command as shown below.

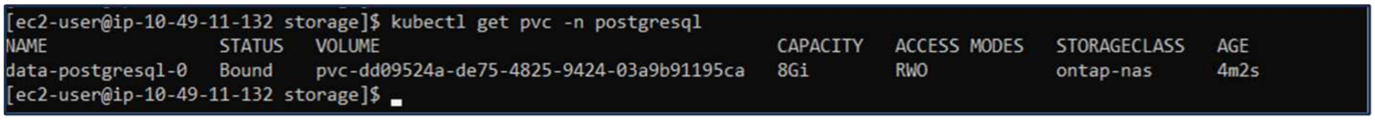


```
[ec2-user@ip-10-49-11-132 astra]$ oc get project postgresql -o yaml | grep uid-range
    openshift.io/sa.scc.uid-range: 1001010000/10000
[ec2-user@ip-10-49-11-132 astra]$ oc edit -n postgresql statefulset.apps/postgresql
statefulset.apps/postgresql edited
[ec2-user@ip-10-49-11-132 astra]$
```

postgresql app should be running and using persistent volumes backed by Amazon FSx ONTAP storage.

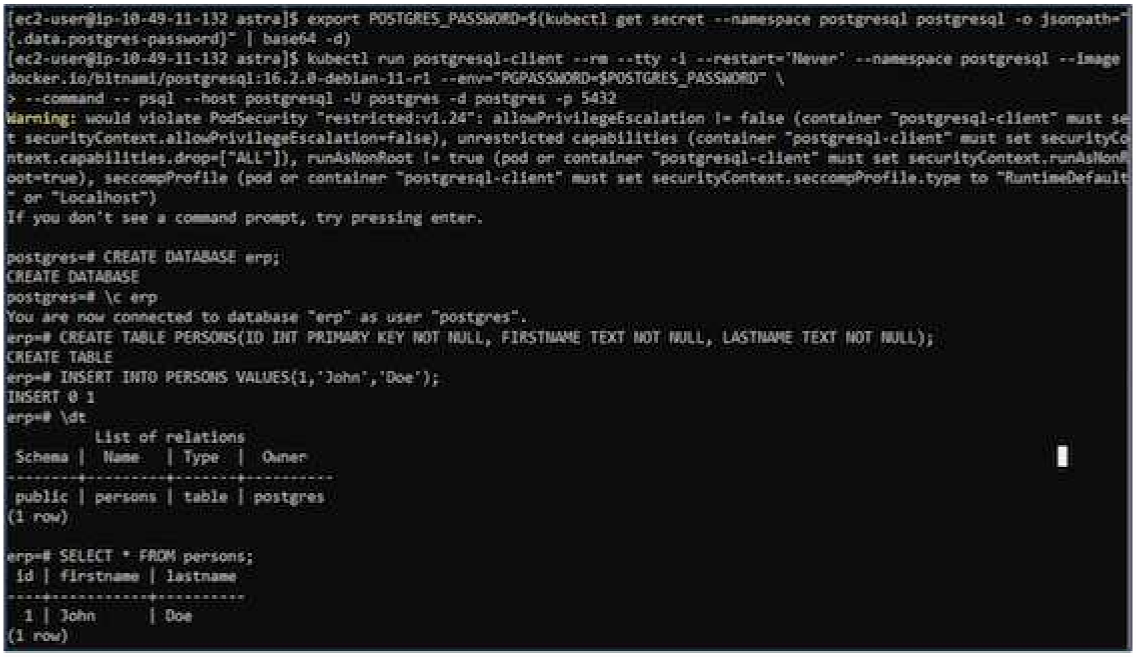


```
[ec2-user@ip-10-49-11-132 astra]$ oc get pods -n postgresql
NAME           READY   STATUS    RESTARTS   AGE
postgresql-0   1/1     Running   0          2m46s
[ec2-user@ip-10-49-11-132 astra]$ _
```



```
[ec2-user@ip-10-49-11-132 storage]$ kubectl get pvc -n postgresql
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-postgresql-0   Bound    pvc-dd09524a-de75-4825-9424-03a9b91195ca   8Gi        RWO            ontap-nas      4m2s
[ec2-user@ip-10-49-11-132 storage]$ _
```

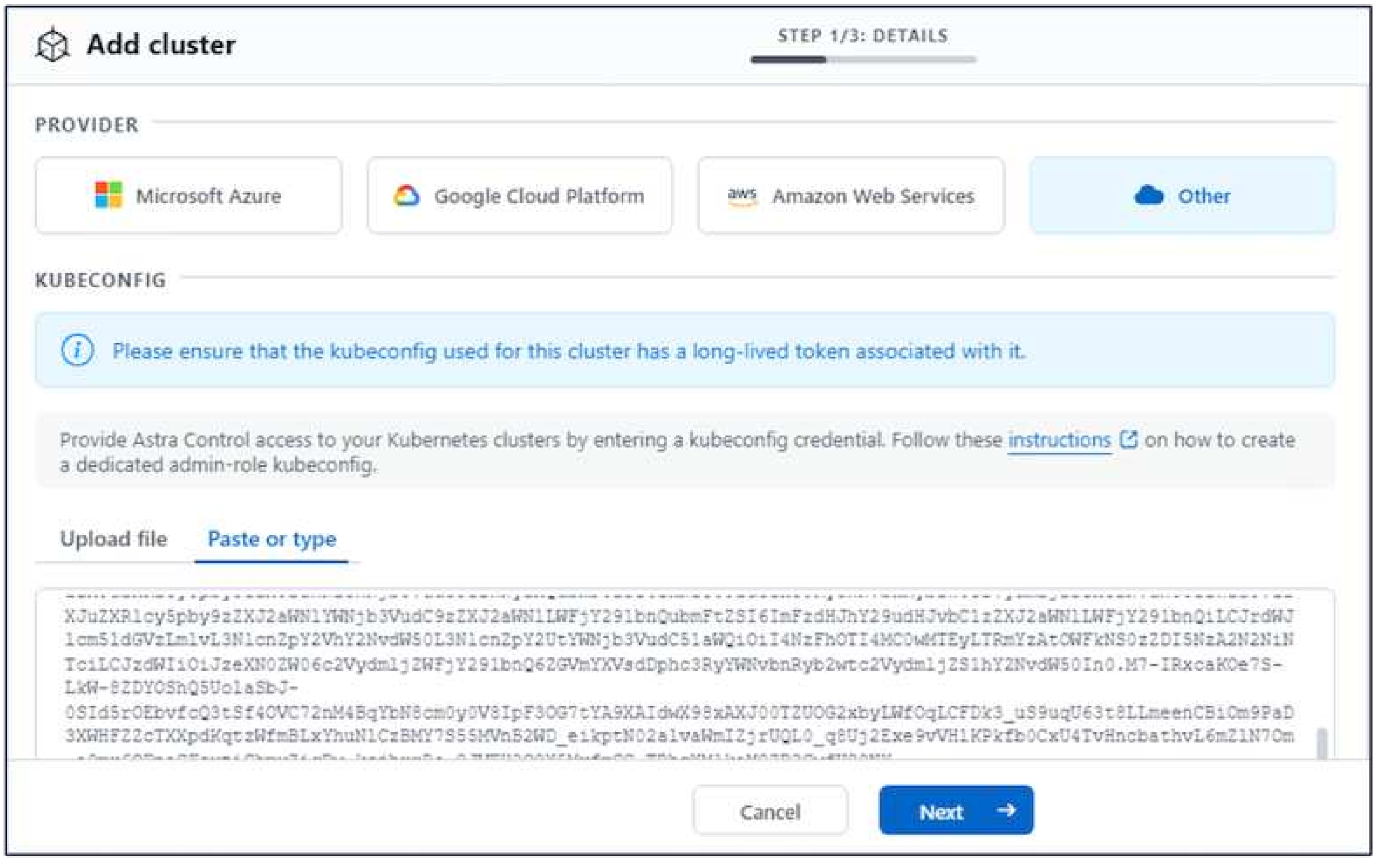
6. Create a database and add a record



```
[ec2-user@ip-10-49-11-132 astra]$ export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql -o jsonpath="
{.data.postgres-password}" | base64 -d)
[ec2-user@ip-10-49-11-132 astra]$ kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image
docker.io/bitnami/postgresql:16.2.0-debian-11-r1 --env="PGPASSWORD=$POSTGRES_PASSWORD" \
> --command -- psql --host postgresql -U postgres -d postgres -p 5432
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must se
t securityContext.allowPrivilegeEscalation=false), unrestricted capabilities (container "postgresql-client" must set securityCo
ntext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "postgresql-client" must set securityContext.runAsNonR
oot=true), seccompProfile (pod or container "postgresql-client" must set securityContext.seccompProfile.type to "RuntimeDefault
" or "Localhost")
If you don't see a command prompt, try pressing enter.

postgres=# CREATE DATABASE erp;
CREATE DATABASE
postgres=# \c erp
You are now connected to database "erp" as user "postgres".
erp=# CREATE TABLE PERSONS(ID INT PRIMARY KEY NOT NULL, FIRSTNAME TEXT NOT NULL, LASTNAME TEXT NOT NULL);
CREATE TABLE
erp=# INSERT INTO PERSONS VALUES(1,'John','Doe');
INSERT 0 1
erp=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persons | table | postgres
(1 row)

erp=# SELECT * FROM persons;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
(1 row)
```

7. Add the cluster into ACS

Log in to ACS. Select cluster and click on Add. Select other and upload or paste the kubeconfig file.

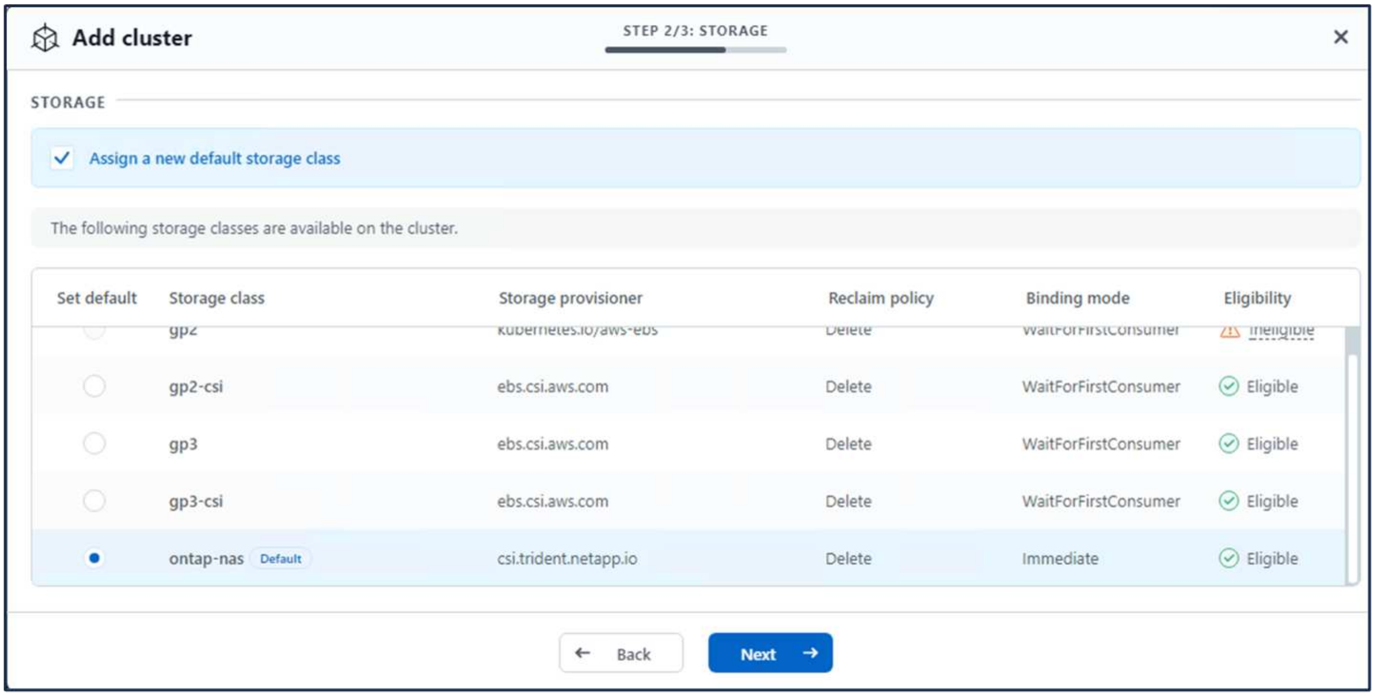


Please ensure that the kubeconfig used for this cluster has a long-lived token associated with it.

Provide Astra Control access to your Kubernetes clusters by entering a kubeconfig credential. Follow these instructions on how to create a dedicated admin-role kubeconfig.

```
XJuZXRlcy5pby9zZXJ2aWN1YWNjb3VudC9zZXJ2aWN1LWFjY291bnQubmFtZSI6ImFzdHJhY29udHJvbC1zZXJ2aWN1LWFjY291bnQiLCJrdWJ
1cm51dGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiI4NzFhOTI4MC0wMTEyLTRmYzAtOWFkNS0zZDI5NzA2N2NiN
TciLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDphc3RyYWNvbnRyb2wtc2VydmljZS1hY2NvdW50In0.M7-IRxcaKOe7S-
LkW-8ZDYOShQ5UolaSbJ-
0SId5rOEbvfcQ3tSf4OVC72nM4BqYbN8cm0y0V8IpF3OG7tYA9XAIdwX98xAXJ00TZUOG2xbyLWfOqLCFDk3_uS9uqU63t8LLmeenCBiOm9PaD
3XWHFZ2cTXXpdKqtzWfmBLxYhuN1CzBMY7S55MVnB2WD_eikptN02alvaWmIZjrUQL0_q8Uj2Exe9vVH1KPkfb0CxU4TvHncbathvL6mZ1N7Om
```

Click **Next** and select ontap-nas as the default storage class for ACS. Click **Next**, review the details and **Add** the cluster.



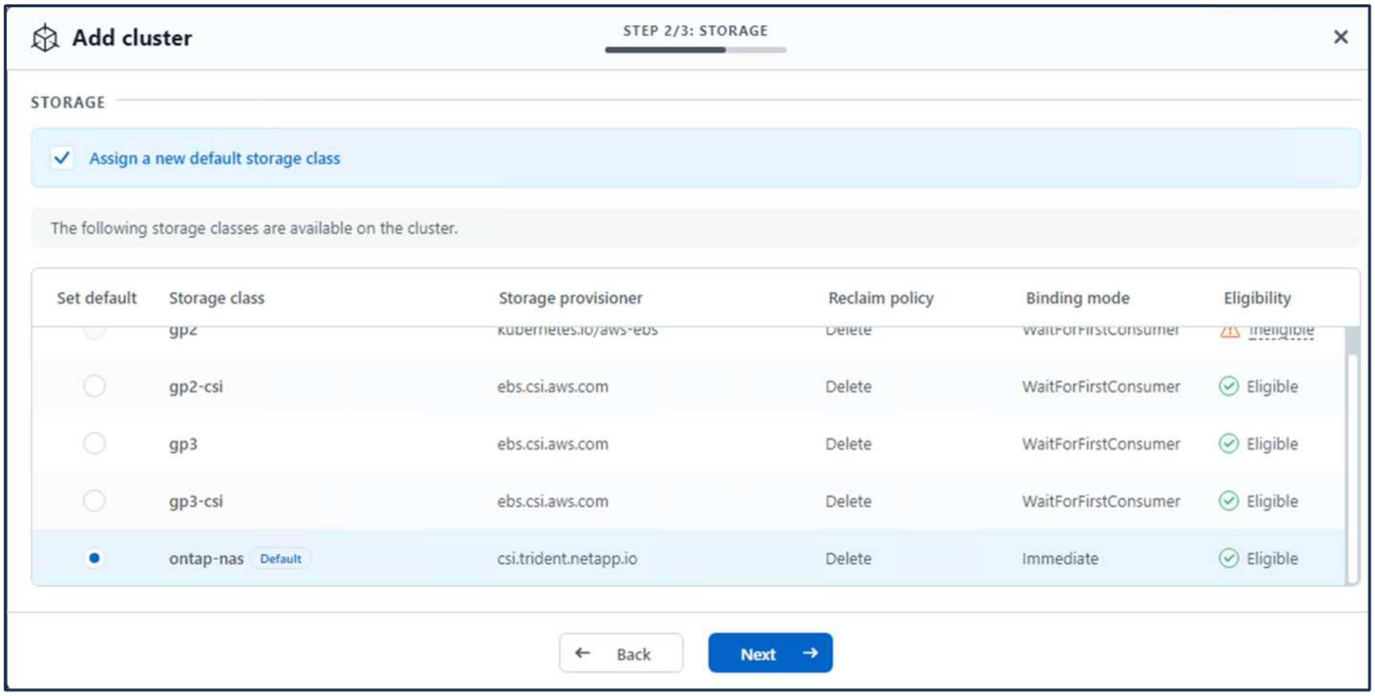
☑ Assign a new default storage class

The following storage classes are available on the cluster.

Set default	Storage class	Storage provisioner	Reclaim policy	Binding mode	Eligibility
○	gp2	kubernetes.io/aws-ebs	Delete	WaitForFirstConsumer	⚠ Ineligible
○	gp2-csi	ebs.csi.aws.com	Delete	WaitForFirstConsumer	✓ Eligible
○	gp3	ebs.csi.aws.com	Delete	WaitForFirstConsumer	✓ Eligible
○	gp3-csi	ebs.csi.aws.com	Delete	WaitForFirstConsumer	✓ Eligible
●	ontap-nas Default	csi.trident.netapp.io	Delete	Immediate	✓ Eligible

8. Define the application in ACS

Define the postgresql application in ACS. From the landing page, select **Applications**, **Define** and fill in the appropriate details. Click **Next** a couple of times, Review the details and click **Define**. The application gets added to ACS.



Add cluster

STEP 2/3: STORAGE

STORAGE

☑ Assign a new default storage class

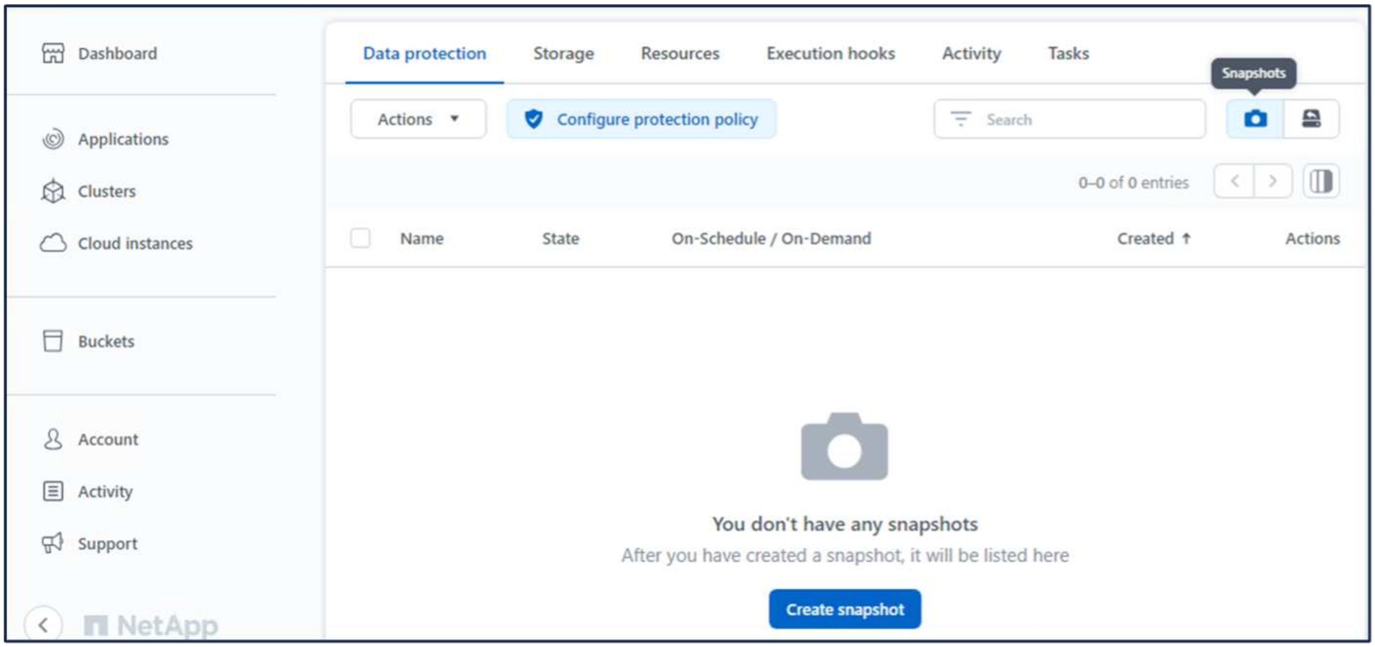
The following storage classes are available on the cluster.

Set default	Storage class	Storage provisioner	Reclaim policy	Binding mode	Eligibility
○	gp2	kubernetes.io/aws-ebs	Delete	WaitForFirstConsumer	⚠ Ineligible
○	gp2-csi	ebs.csi.aws.com	Delete	WaitForFirstConsumer	✓ Eligible
○	gp3	ebs.csi.aws.com	Delete	WaitForFirstConsumer	✓ Eligible
○	gp3-csi	ebs.csi.aws.com	Delete	WaitForFirstConsumer	✓ Eligible
●	ontap-nas Default	csi.trident.netapp.io	Delete	Immediate	✓ Eligible

9. Create a snapshot using ACS

There are many ways to create a snapshot in ACS. You can select the application and create a snapshot from the page that shows the details of the application. You can click on Create snapshot to create an on-demand snapshot or configure a protection policy.

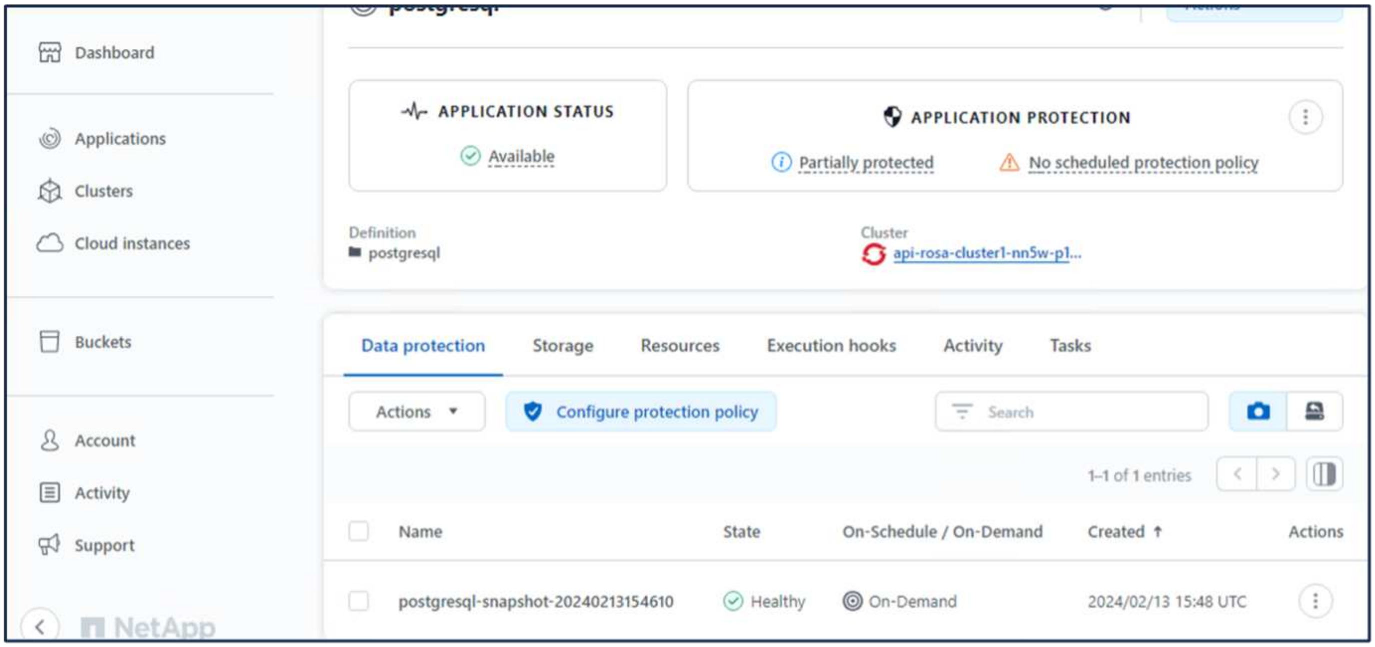
Create an on-demand snapshot by simply clicking on **Create snapshot**, providing a name, reviewing the details, and clicking on **Snapshot**. The snapshot state changes to Healthy after the operation is completed.



☐	Name	State	On-Schedule / On-Demand	Created ↑	Actions

You don't have any snapshots

After you have created a snapshot, it will be listed here



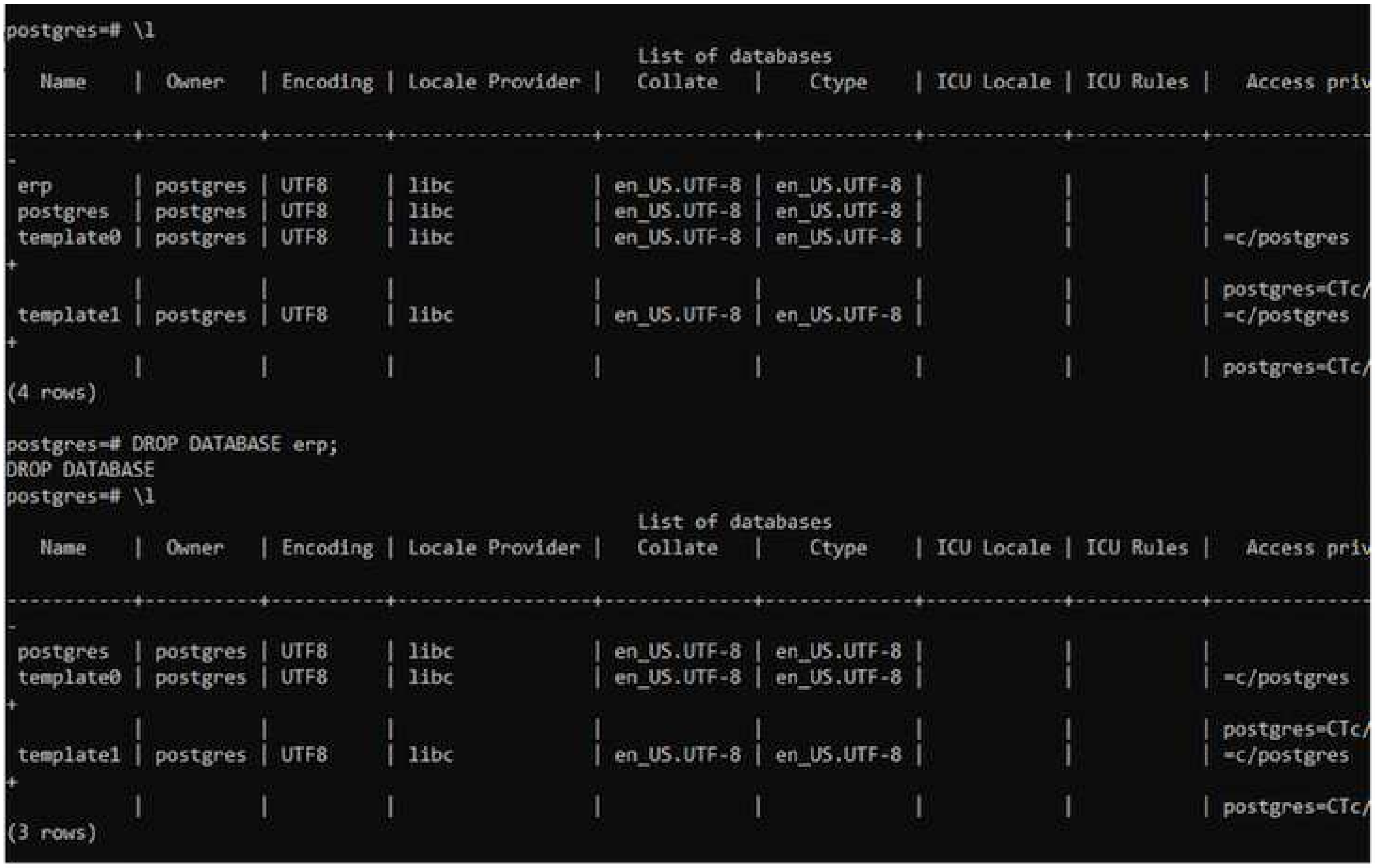
Definition
postgresql

Cluster
api-rosa-cluster1-nn5w-p1...

Name	State	On-Schedule / On-Demand	Created ↑	Actions
postgresql-snapshot-20240213154610	Healthy	On-Demand	2024/02/13 15:48 UTC	⋮

10. Delete the database in the postgresql application

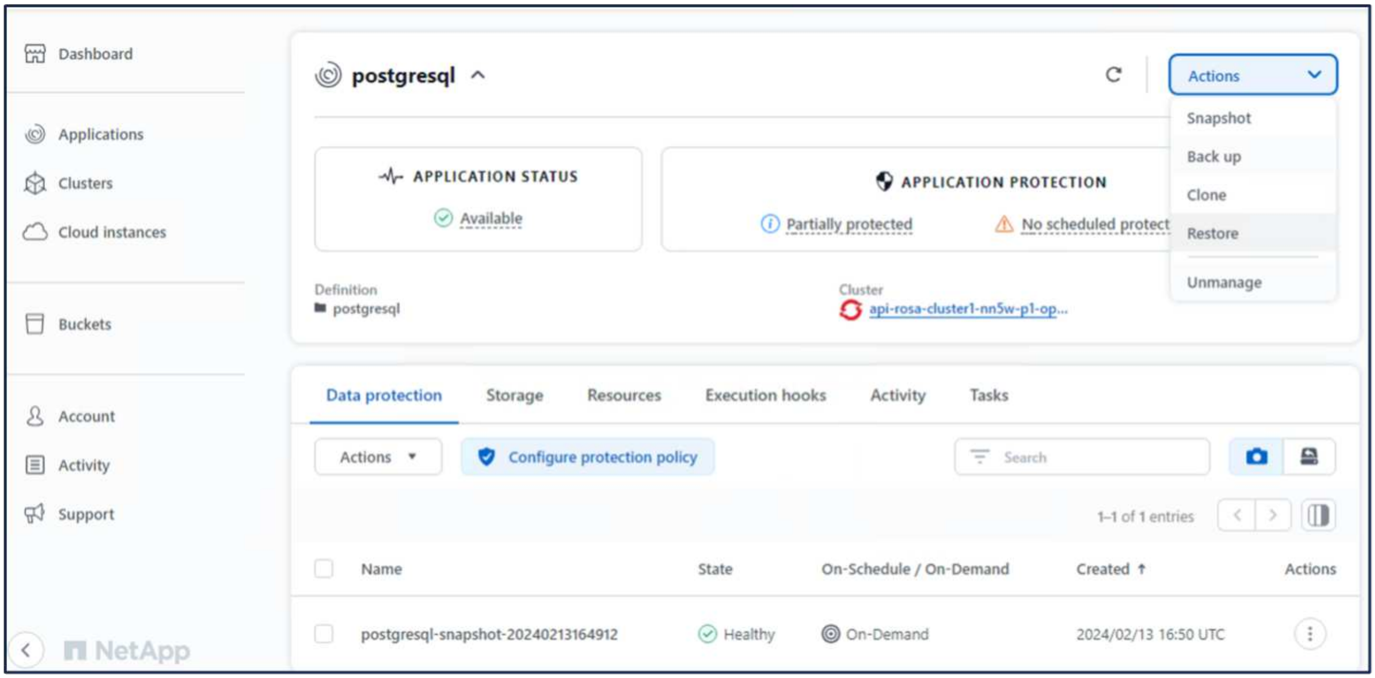
Log back into postgresql, list the available databases, delete the one you created previously and list again to ensure that the database has been deleted.



```
postgres=# \l
                                                     List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access priv
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+--------------
-
 erp       | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres
+
           |          |          |                 |             |             |            |           | postgres=CTc/
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres
+
           |          |          |                 |             |             |            |           | postgres=CTc/
(4 rows)

postgres=# DROP DATABASE erp;
DROP DATABASE
postgres=# \l
                                                     List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access priv
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+--------------
-
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres
+
           |          |          |                 |             |             |            |           | postgres=CTc/
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres
+
           |          |          |                 |             |             |            |           | postgres=CTc/
(3 rows)
```

11. Restore from a snapshot using ACS

To restore the application from a snapshot, go to ACS UI landing page, select the application and select Restore. You need to pick a snapshot or a backup from which to restore. (Typically, you would have multiple created based on a policy that you have configured). Make appropriate choices in the next couple of screens and then click on **Restore**. The application status moves from Restoring to Available after it has been restored from the snapshot.



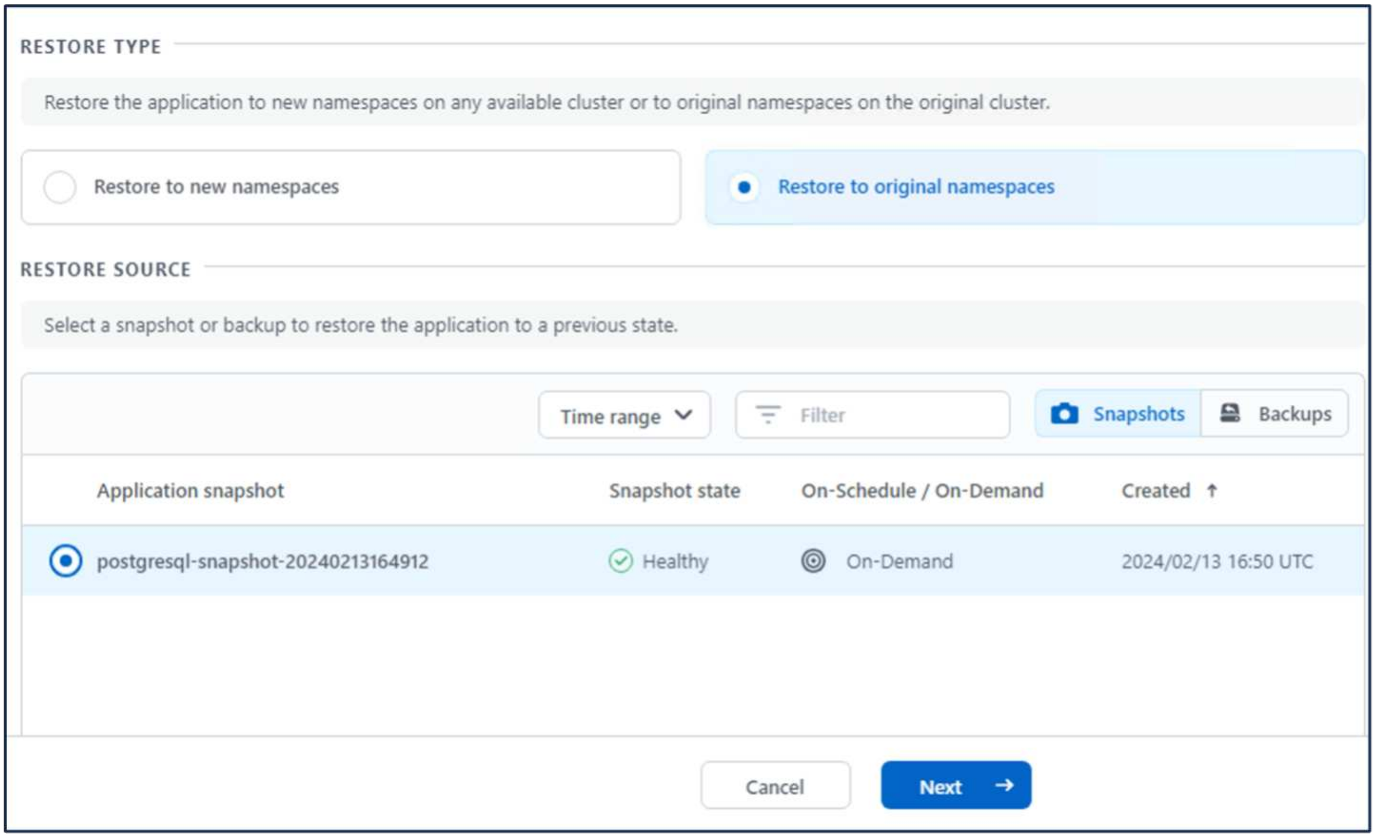
APPLICATION STATUS: Available

APPLICATION PROTECTION: Partially protected; No scheduled protect

Definition: postgresql

Cluster: api-rosa-cluster1-nn5w-p1-op...

	Name	State	On-Schedule / On-Demand	Created ↑	Actions
□	postgresql-snapshot-20240213164912	Healthy	On-Demand	2024/02/13 16:50 UTC	⋮


RESTORE TYPE

Restore the application to new namespaces on any available cluster or to original namespaces on the original cluster.

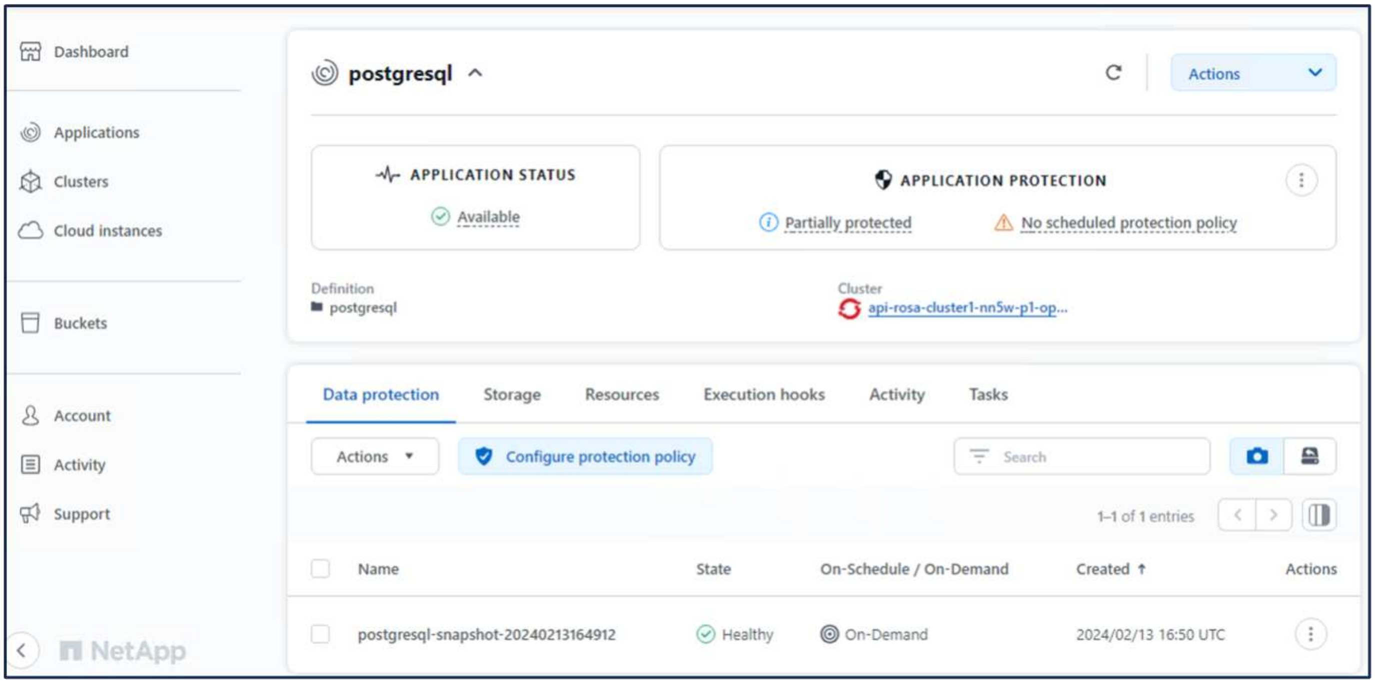
○ Restore to new namespaces

● Restore to original namespaces

RESTORE SOURCE

Select a snapshot or backup to restore the application to a previous state.

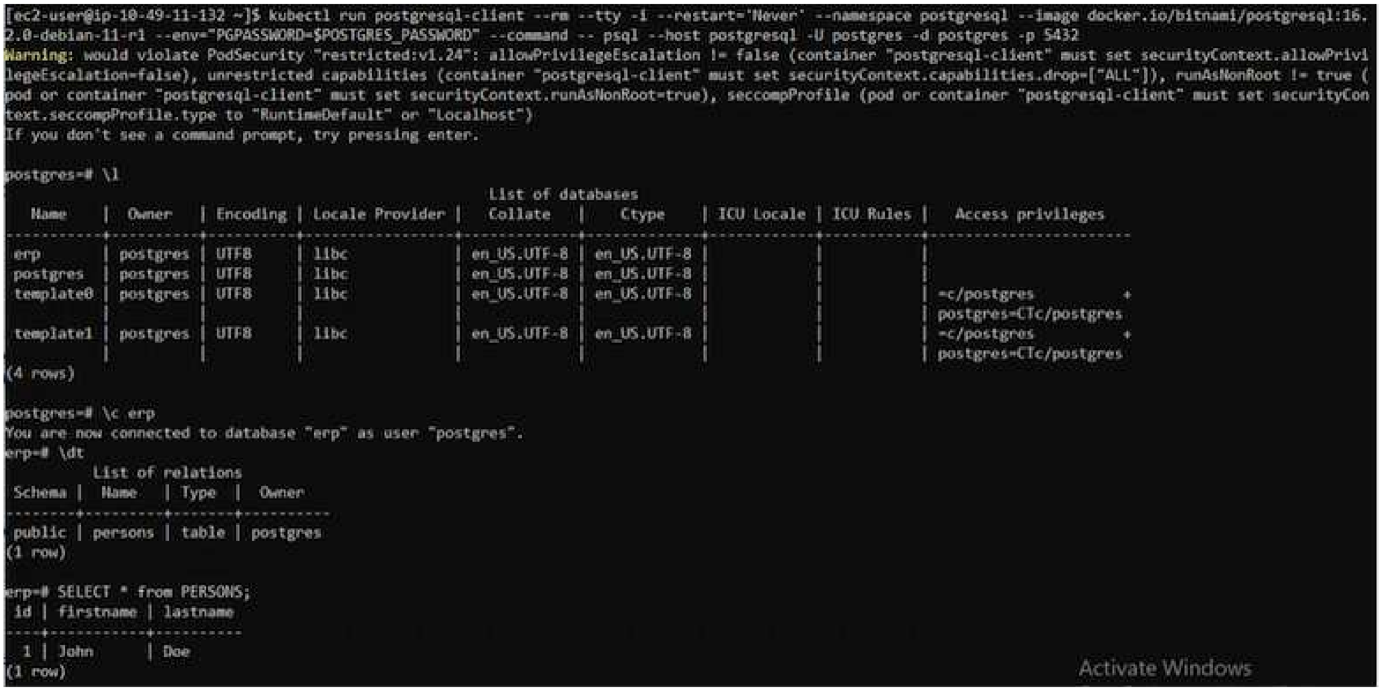
	Application snapshot	Snapshot state	On-Schedule / On-Demand	Created ↑
●	postgresql-snapshot-20240213164912	Healthy	On-Demand	2024/02/13 16:50 UTC



□	Name	State	On-Schedule / On-Demand	Created ↑	Actions
□	postgresql-snapshot-20240213164912	✓ Healthy	On-Demand	2024/02/13 16:50 UTC	⋮

12. Verify your app has been restored from the snapshot

Login to the postgresql client and you should now see the table and the record in the table that you previously had. That’s it. Just by clicking a button, your application has been restored to a previous state. That is how easy we make it for our customers with Astra Control.



```
[ec2-user@ip-10-49-11-132 ~]$ kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:16.
2.0-debian-11-r1 --env="PGPASSWORD=$POSTGRES_PASSWORD" --command -- psql --host postgresql -U postgres -d postgres -p 5432
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityContext.allowPrivi
legeEscalation=false), unrestricted capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (
pod or container "postgresql-client" must set securityContext.runAsNonRoot=true), seccompProfile (pod or container "postgresql-client" must set securityCon
text.seccompProfile.type to "RuntimeDefault" or "Localhost")
If you don't see a command prompt, try pressing enter.

postgres=# \l
                                                       List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+-----------------------
 erp       | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
(4 rows)

postgres=# \c erp
You are now connected to database "erp" as user "postgres".
erp=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persons | table | postgres
(1 row)

erp=# SELECT * from PERSONS;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
(1 row)
```

Data migration

This page shows the data migration options for container workloads on Managed Red Hat OpenShift clusters using FSx ONTAP for persistent storage.

Data Migration

Red Hat OpenShift service on AWS as well as Amazon FSx for NetApp ONTAP (FSx ONTAP) are part of their service portfolio by AWS. FSx ONTAP is available on Single AZ or Multi-AZ options.
Multi-Az option provides data protection from availability zone failure.
FSx ONTAP can be integrated with Trident to provide persistent storage for applications on ROSA clusters.

Integration of FSx ONTAP with Trident using Helm chart

ROSA Cluster Integration with Amazon FSx ONTAP

The migration of container applications involves:

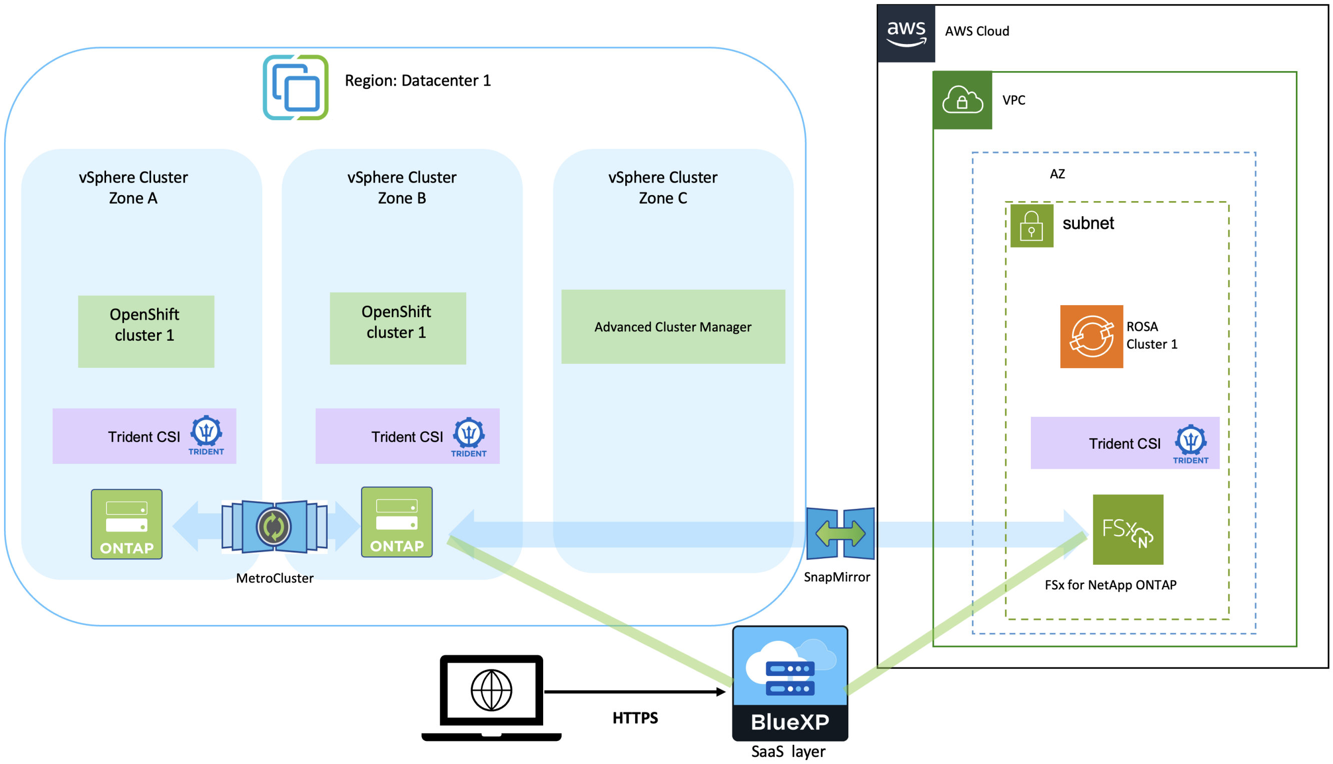
- Persistent volumes: this can be accomplished using BlueXP.
  Another option is to use Trident Protect to handle container application migrations from on-premises to the cloud environment. Automation can be used for the same purpose.
- Application metadata: this can be accomplished using OpenShift GitOps (Argo CD).

Failover and Fail-back of applications on ROSA cluster using FSx ONTAP for persistent storage

The following video is a demonstration of application failover and fail-back scenarios using BlueXP and Argo CD.

Failover and Fail-back of applications on ROSA cluster

Data protection and migration solution for OpenShift Container workloads



Additional NetApp Hybrid Multicloud solutions for Red Hat OpenShift workloads

Additional solutions

Additional solutions are available in other sections as follows:

For Red Hat OpenShift Container solutions, see here.

For Red Hat OpenShift Virtualization solutions on premises, see here.

Copyright information

Trademark information