Cloud hybride avec composants gérés par le fournisseur

NetApp public and hybrid cloud solutions

NetApp
August 18, 2025

Sommaire

Cloud hybride avec composants gérés par le fournisseur

Solution NetApp avec charges de travail de la plateforme de conteneurs Red Hat OpenShift gérées

Les clients peuvent être « nés dans le cloud » ou être à un stade de leur parcours de modernisation où ils sont prêts à déplacer certaines charges de travail sélectionnées ou toutes les charges de travail de leurs centres de données vers le cloud. Ils peuvent choisir d'utiliser des conteneurs OpenShift gérés par le fournisseur et un stockage NetApp géré par le fournisseur dans le cloud pour exécuter leurs charges de travail. Ils doivent planifier et déployer les clusters de conteneurs Red Hat OpenShift gérés dans le cloud pour un environnement prêt pour la production pour leurs charges de travail de conteneurs. NetApp propose des offres de stockage entièrement gérées pour les solutions Red Hat gérées dans les trois principaux clouds publics.

- Amazon FSx for NetApp ONTAP (FSx ONTAP)*

FSx ONTAP offre une protection des données, une fiabilité et une flexibilité pour les déploiements de conteneurs dans AWS. Trident sert de fournisseur de stockage dynamique pour consommer le stockage persistant FSx ONTAP pour les applications avec état des clients.

Comme ROSA peut être déployé en mode HA avec des nœuds de plan de contrôle répartis sur plusieurs zones de disponibilité, FSx ONTAP peut également être provisionné avec l'option Multi-AZ qui offre une haute disponibilité et une protection contre les pannes AZ.

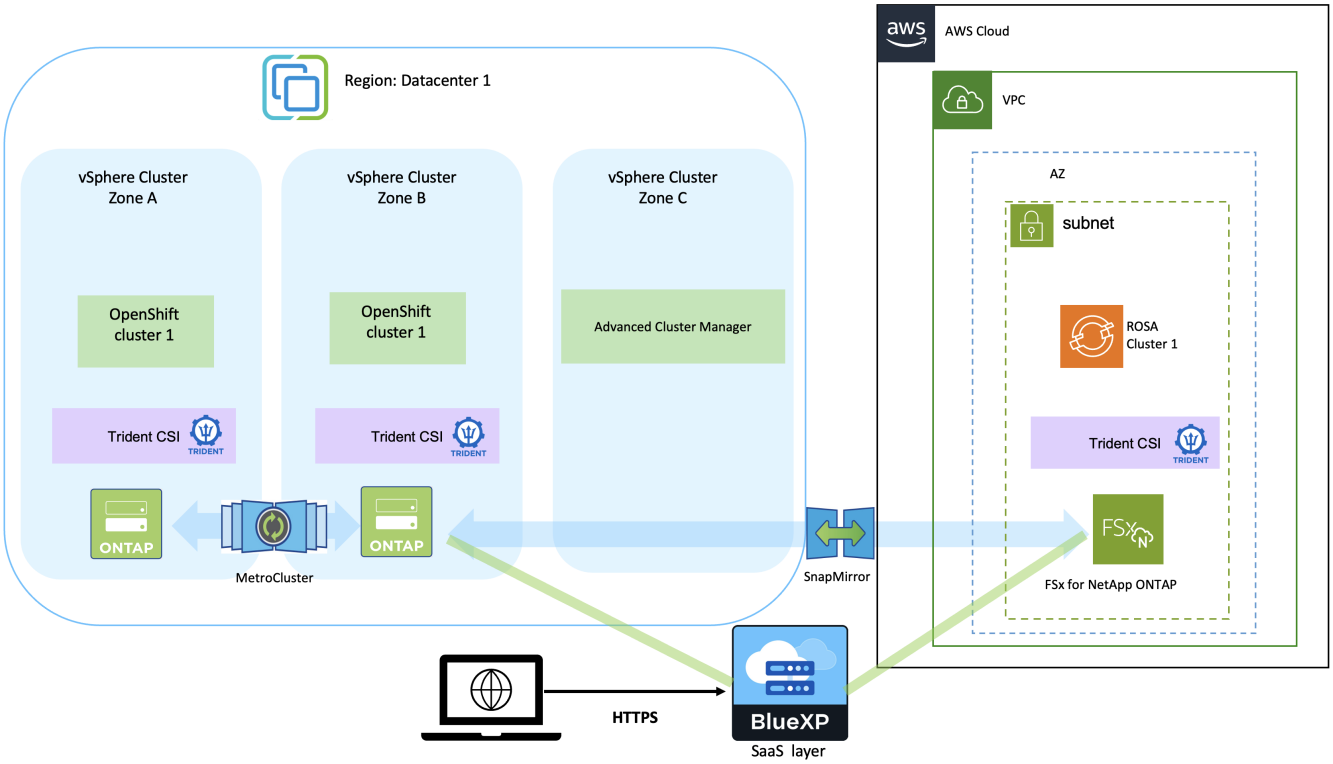
- Google Cloud NetApp Volumes*

Red Hat OpenShift Dedicated est une plateforme d'applications entièrement gérée qui vous permet de créer, de déployer et de faire évoluer rapidement des applications dans le cloud hybride. Google Cloud NetApp Volumes fournit des volumes persistants apportant la suite complète des fonctionnalités de gestion des données d'entreprise d'ONTAP aux déploiements OpenShift dans Google Cloud.

Déployer et configurer la plateforme Managed Red Hat OpenShift Container sur AWS

Cette section décrit un flux de travail de haut niveau pour la configuration des clusters Red Hat OpenShift gérés sur AWS (ROSA). Il montre l'utilisation d' Amazon FSx for NetApp ONTAP (FSx ONTAP) comme backend de stockage par Trident pour fournir des volumes persistants. Des détails sont fournis sur le déploiement de FSx ONTAP sur AWS à l'aide de BlueXP. Des détails sont également fournis sur l'utilisation de BlueXP et d'OpenShift GitOps (Argo CD) pour effectuer des activités de protection et de migration des données pour les applications avec état sur les clusters ROSA.

Voici un diagramme qui illustre les clusters ROSA déployés sur AWS et utilisant FSx ONTAP comme stockage backend.



Cette solution a été vérifiée en utilisant deux clusters ROSA dans deux VPC dans AWS. Chaque cluster ROSA a été intégré à FSx ONTAP à l'aide de Trident. Il existe plusieurs façons de déployer des clusters ROSA et FSx ONTAP dans AWS. Cette description de haut niveau de la configuration fournit des liens de documentation pour la méthode spécifique qui a été utilisée. Vous pouvez vous référer aux autres méthodes dans les liens pertinents fournis dans le"section ressources" .

Le processus de configuration peut être décomposé selon les étapes suivantes :

Installer les clusters ROSA

- Créez deux VPC et configurez la connectivité de peering VPC entre les VPC.
- Référer"ici" pour obtenir des instructions sur l'installation des clusters ROSA.

Installer FSx ONTAP

- Installez FSx ONTAP sur les VPC depuis BlueXP. Référer"ici" pour la création d'un compte BlueXP et pour commencer. Référer"ici" pour l'installation de FSx ONTAP. Référer"ici" pour créer un connecteur dans AWS pour gérer le FSx ONTAP.
- Déployez FSx ONTAP à l'aide d'AWS. Référer"ici" pour le déploiement à l'aide de la console AWS.

Installer Trident sur les clusters ROSA (à l'aide du graphique Helm)

- Utilisez le graphique Helm pour installer Trident sur les clusters ROSA. Consultez le lien de documentation : https://docs.netapp.com/us-en/trident/trident-get-started/kubernetes-deploy-helm.html [ici].

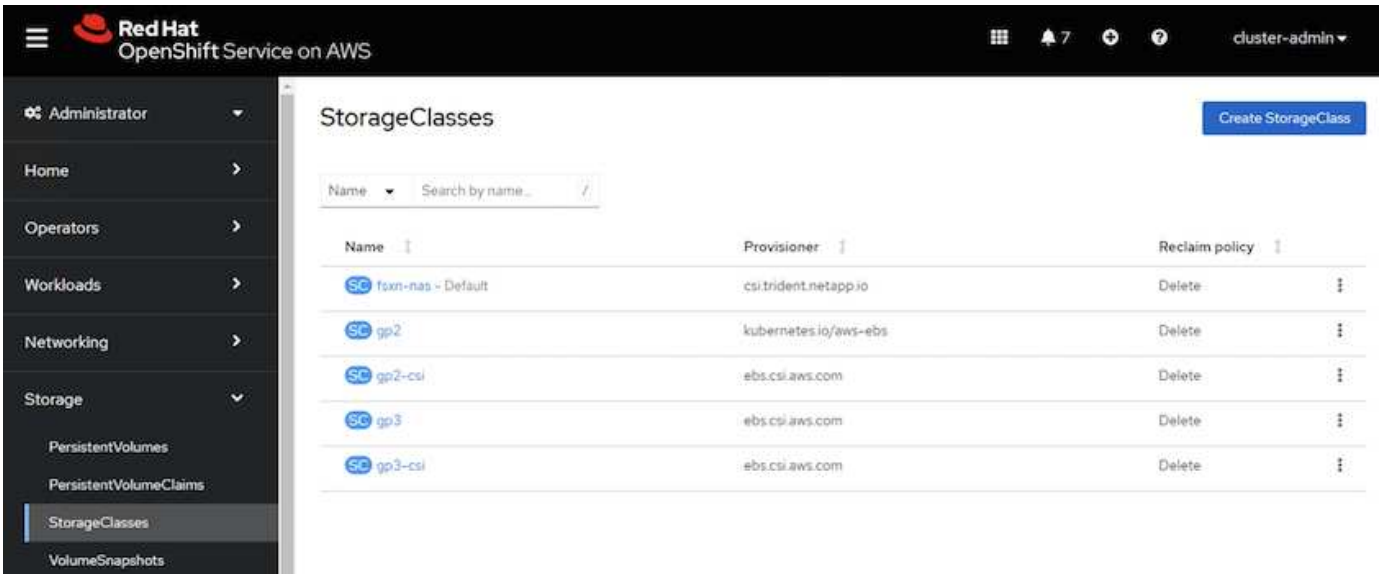
Intégration de FSx ONTAP avec Trident pour les clusters ROSA

OpenShift GitOps peut être utilisé pour déployer Trident CSI sur tous les clusters gérés au fur et à mesure qu'ils sont enregistrés sur ArgoCD à l'aide d'ApplicationSet.

```yaml
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
```

Créer des classes backend et de stockage à l'aide de Trident (pour FSx ONTAP)

- Référer"ici" pour plus de détails sur la création d'un backend et d'une classe de stockage.
- Créez la classe de stockage créée pour FsxN avec Trident CSI par défaut à partir de la console OpenShift. Voir la capture d'écran ci-dessous :



Déployer une application à l'aide d'OpenShift GitOps (Argo CD)

- Installez l'opérateur OpenShift GitOps sur le cluster. Se référer aux instructions"ici" .
- Configurez une nouvelle instance Argo CD pour le cluster. Se référer aux instructions"ici" .

Ouvrez la console d'Argo CD et déployez une application. À titre d'exemple, vous pouvez déployer une application Jenkins à l'aide d'Argo CD avec un graphique Helm. Lors de la création de l'application, les détails

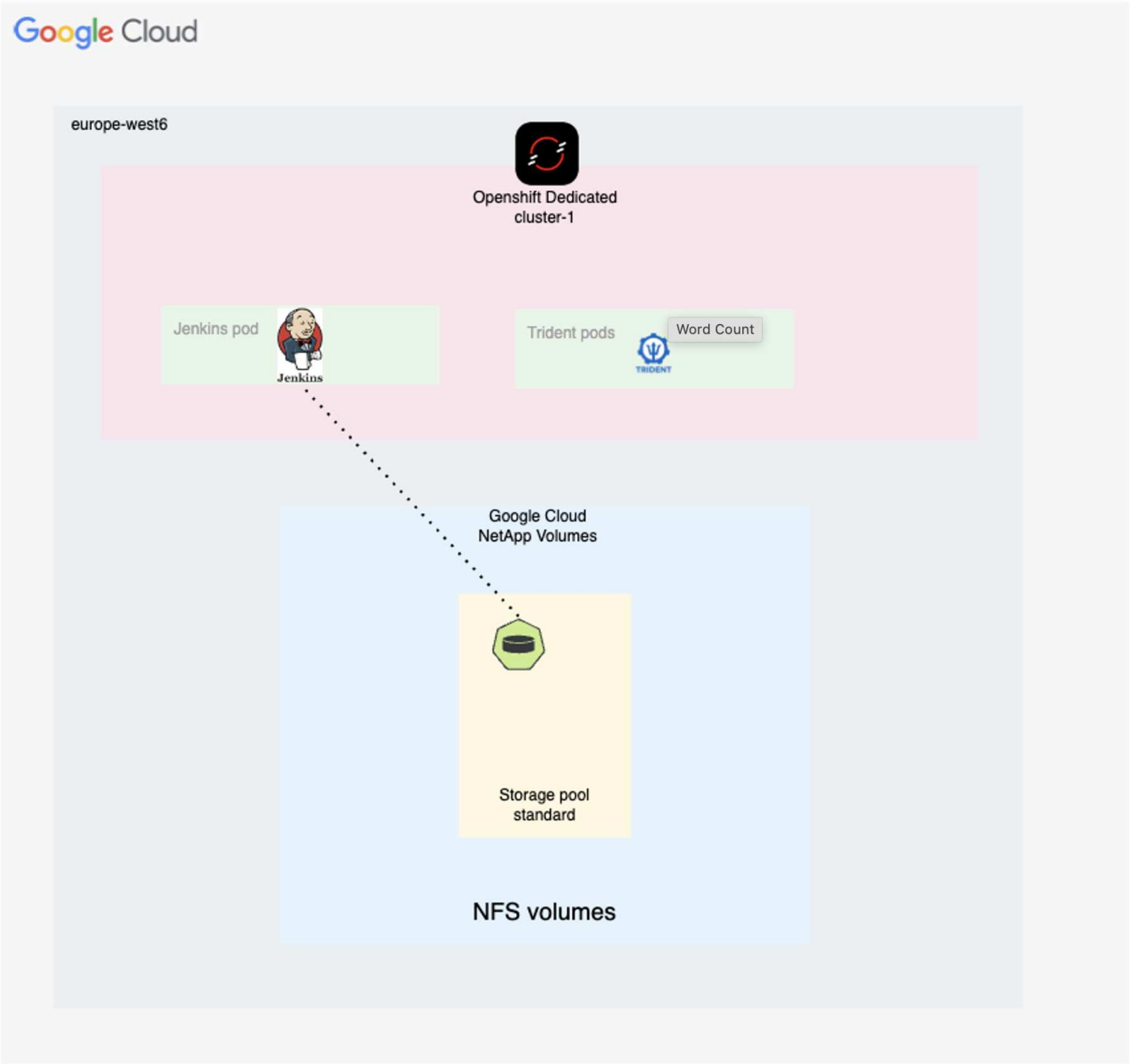
suivants ont été fournis : Projet : cluster par défaut :'https://kubernetes.default.svc' (sans les guillemets) Espace de noms : Jenkins L'URL du graphique Helm :'https://charts.bitnami.com/bitnami' (sans les guillemets)

Paramètres Helm : global.storageClass : fsxn-nas

Déployer et configurer OpenShift Dedicated sur Google Cloud avec Google Cloud NetApp Volumes

Cette section décrit un flux de travail de haut niveau pour la configuration de clusters OpenShift Dedicated (OSD) sur la plate-forme Google Cloud. Il montre NetApp Trident utilisant Google Cloud NetApp Volumes comme backend de stockage pour fournir des volumes persistants pour les applications avec état exécutées avec Kubernetes.

Voici un diagramme qui illustre un cluster OSD déployé sur Google Cloud et utilisant NetApp Volumes comme stockage backend.



Le processus de configuration peut être décomposé selon les étapes suivantes :

Installer des clusters OSD dans Google Cloud

- Si vous souhaitez utiliser un VPC existant pour le cluster, vous devez créer le VPC, deux sous-réseaux, un routeur cloud et deux NAT cloud GCP pour le cluster OSD. Référer"ici" pour les instructions.
- Référer"ici" pour obtenir des instructions sur l'installation de clusters OSD sur GCP à l'aide du modèle de facturation Customer Cloud Subscription (CCS). OSD est également inclus sur Google Cloud Marketplace. Une vidéo montrant comment installer OSD à l'aide de la solution Google Cloud Marketplace est disponible"ici" .

Activer les Google Cloud NetApp Volumes

- Référer"ici" pour plus d'informations sur la configuration de l'accès à Google Cloud NetApp Volumes. Suivez toutes les étapes jusqu'à et y compris
- Créer un pool de stockage. Référer"ici" pour plus d'informations sur la configuration d'un pool de stockage sur Google Cloud NetApp Volumes. Les volumes pour les applications Kubernetes avec état exécutées sur

OSD seront créés dans le pool de stockage.

Installer Trident sur les clusters OSD (à l'aide du graphique Helm)

- Utilisez un graphique Helm pour installer Trident sur des clusters OSD. Référer"ici" pour obtenir des instructions sur la façon d'installer le Helm Chart. La carte de barre peut être trouvée"ici" .

Intégration de NetApp Volumes avec NetApp Trident pour les clusters OSD

Créer des classes backend et de stockage à l'aide de Trident (pour Google Cloud NetApp Volumes)

- Consultez ici pour plus de détails sur la création du backend.
- Si l'une des classes de stockage actuelles dans Kubernetes est marquée comme par défaut, supprimez cette annotation en modifiant la classe de stockage.
- Créez au moins une classe de stockage pour les volumes NetApp avec le provisionneur Trident CSI. Faites d'une seule des classes de stockage la valeur par défaut à l'aide d'une annotation. Cela permettra à un PVC d'utiliser cette classe de stockage lorsqu'elle n'est pas explicitement appelée dans le manifeste PVC. Un exemple avec l'annotation est montré ci-dessous.

```
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
```

Déployer une application à l'aide d'OpenShift GitOps (Argo CD)

- Installez l'opérateur OpenShift GitOps sur le cluster. Se référer aux instructions"ici" .
- Configurez une nouvelle instance Argo CD pour le cluster. Se référer aux instructions"ici" .

Ouvrez la console d'Argo CD et déployez une application. À titre d'exemple, vous pouvez déployer une application Jenkins à l'aide d'Argo CD avec un graphique Helm. Lors de la création de l'application, les détails suivants ont été fournis : Projet : cluster par défaut :'https://kubernetes.default.svc' (sans les guillemets) Espace de noms : Jenkins L'URL du graphique Helm :'https://charts.bitnami.com/bitnami' (sans les guillemets)

Protection des données

Cette page présente les options de protection des données pour les clusters Red Hat OpenShift on AWS (ROSA) gérés à l'aide d' Astra Control Service. Astra Control Service (ACS) fournit une interface utilisateur graphique facile à utiliser avec laquelle vous pouvez ajouter des clusters, définir des applications exécutées sur eux et effectuer des activités de gestion des données prenant en compte les applications. Les fonctions ACS sont également accessibles à l'aide d'une API qui permet l'automatisation des flux de travail.

Astra Control (ACS ou ACC) est alimenté par NetApp Trident. Trident intègre plusieurs types de clusters Kubernetes tels que Red Hat OpenShift, EKS, AKS, SUSE Rancher, Anthos, etc., avec différentes versions de stockage NetApp ONTAP telles que FAS/ AFF, ONTAP Select, CVO, Google Google Cloud NetApp Volumes, Azure NetApp Files et Amazon FSx ONTAP.

Cette section fournit des détails sur les options de protection des données suivantes utilisant ACS :

- Une vidéo montrant la sauvegarde et la restauration d'une application ROSA exécutée dans une région et restaurée dans une autre région.
- Une vidéo montrant un instantané et une restauration d'une application ROSA.
- Détails étape par étape de l'installation d'un cluster ROSA, Amazon FSx ONTAP, à l'aide de NetApp Trident pour l'intégration avec le backend de stockage, l'installation d'une application postgresql sur le cluster ROSA, l'utilisation d'ACS pour créer un instantané de l'application et la restauration de l'application à partir de celle-ci.
- Un blog présentant les détails étape par étape de la création et de la restauration à partir d'un instantané pour une application mysql sur un cluster ROSA avec FSx ONTAP à l'aide d'ACS.

Sauvegarde/restauration à partir d'une sauvegarde

La vidéo suivante montre la sauvegarde d'une application ROSA exécutée dans une région et sa restauration dans une autre région.

Service FSx NetApp ONTAP pour Red Hat OpenShift sur AWS

Instantané/Restauration à partir d'un instantané

La vidéo suivante montre comment prendre un instantané d'une application ROSA et restaurer à partir de l'instantané par la suite.

Instantané/Restauration pour les applications sur les clusters Red Hat OpenShift Service sur AWS (ROSA) avec stockage Amazon FSx ONTAP

Blog

- "Utilisation d' Astra Control Service pour la gestion des données des applications sur les clusters ROSA avec stockage Amazon FSx"

Détails étape par étape pour créer un instantané et le restaurer à partir de celui-ci

Configuration des prérequis

- "compte AWS"
- "Compte Red Hat OpenShift"
- Utilisateur IAM avec"autorisations appropriées" pour créer et accéder au cluster ROSA
- "AWS CLI"
- "ROSA CLI"
- "OpenShift CLI"(oc)
- VPC avec sous-réseaux et passerelles et routes appropriées
- "Cluster ROSA installé"dans le VPC

- "Amazon FSx ONTAP"créé dans le même VPC
- Accès au cluster ROSA depuis"Console OpenShift Hybrid Cloud"

Prochaines étapes

1. Créez un utilisateur administrateur et connectez-vous au cluster.
2. Créez un fichier kubeconfig pour le cluster.
3. Installez Trident sur le cluster.
4. Créez une configuration de backend, de classe de stockage et de classe d'instantané à l'aide du provisionneur Trident CSI.
5. Déployez une application postgresql sur le cluster.
6. Créez une base de données et ajoutez un enregistrement.
7. Ajoutez le cluster dans ACS.
8. Définir l'application dans ACS.
9. Créez un instantané à l'aide d'ACS.
10. Supprimez la base de données dans l'application postgresql.
11. Restaurer à partir d'un instantané à l'aide d'ACS.
12. Vérifiez que votre application a été restaurée à partir de l'instantané.

1. Créez un utilisateur administrateur et connectez-vous au cluster

Accédez au cluster ROSA en créant un utilisateur administrateur avec la commande suivante : (Vous devez créer un utilisateur administrateur uniquement si vous n'en avez pas créé un au moment de l'installation)

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

La commande fournira une sortie qui ressemblera à ce qui suit. Connectez-vous au cluster à l'aide du `oc login` commande fournie dans la sortie.

```
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
```

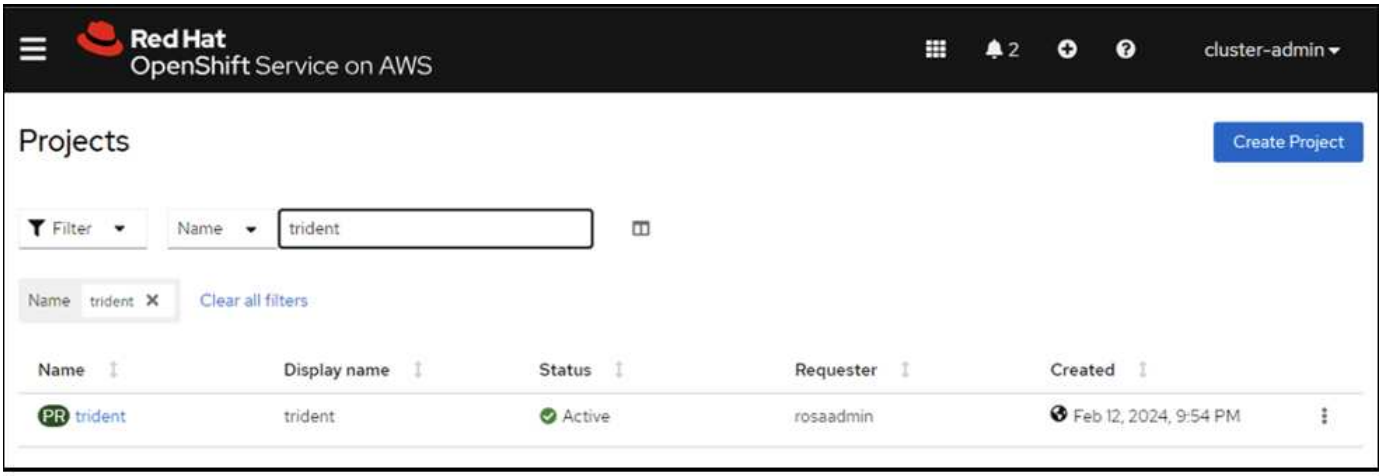
Vous pouvez également vous connecter au cluster à l'aide d'un jeton. Si vous avez déjà créé un utilisateur administrateur au moment de la création du cluster, vous pouvez vous connecter au cluster à partir de la console Red Hat OpenShift Hybrid Cloud avec les informations d'identification de l'utilisateur administrateur. Ensuite, en cliquant sur le coin supérieur droit où s'affiche le nom de l'utilisateur connecté, vous pouvez obtenir le `oc login` commande (token login) pour la ligne de commande.

2. Créer un fichier kubeconfig pour le cluster

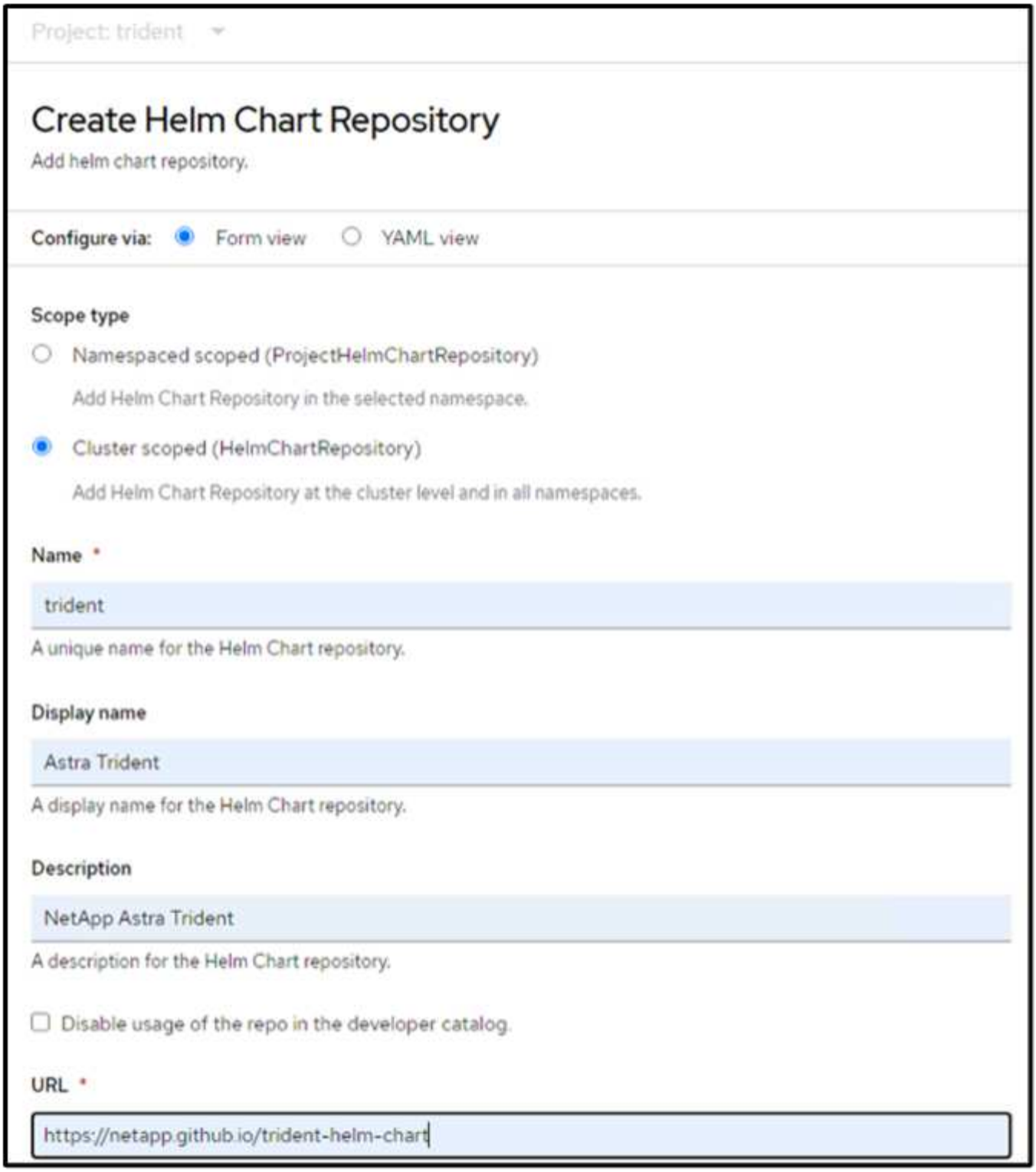
Suivez les procédures"ici" pour créer un fichier kubeconfig pour le cluster ROSA. Ce fichier kubeconfig sera utilisé ultérieurement lorsque vous ajouterez le cluster dans ACS.

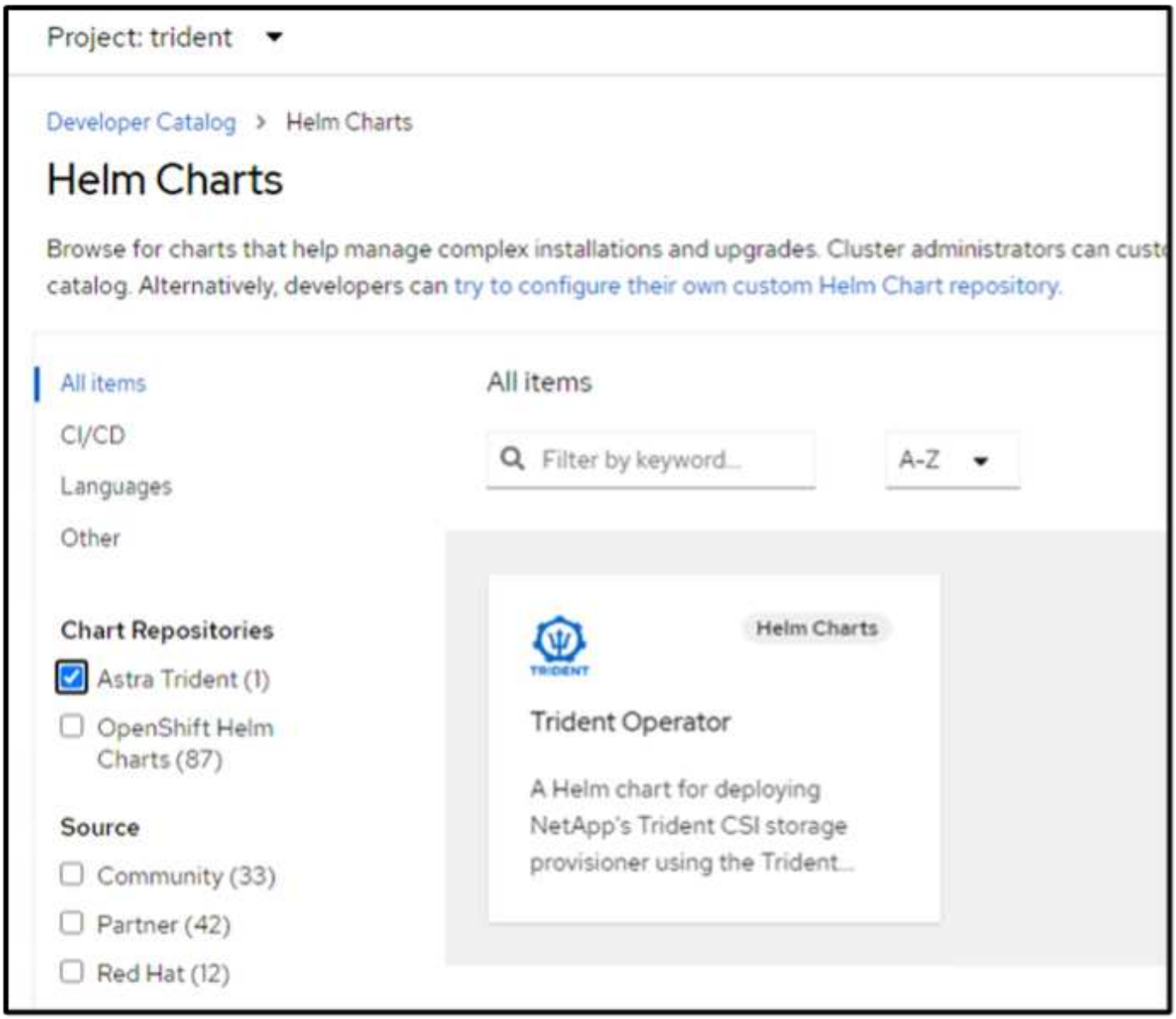
3. Installer Trident sur le cluster

Installez Trident (dernière version) sur le cluster ROSA. Pour ce faire, vous pouvez suivre l'une des procédures indiquées"ici" . Pour installer Trident à l'aide de Helm depuis la console du cluster, créez d'abord un projet appelé Trident.

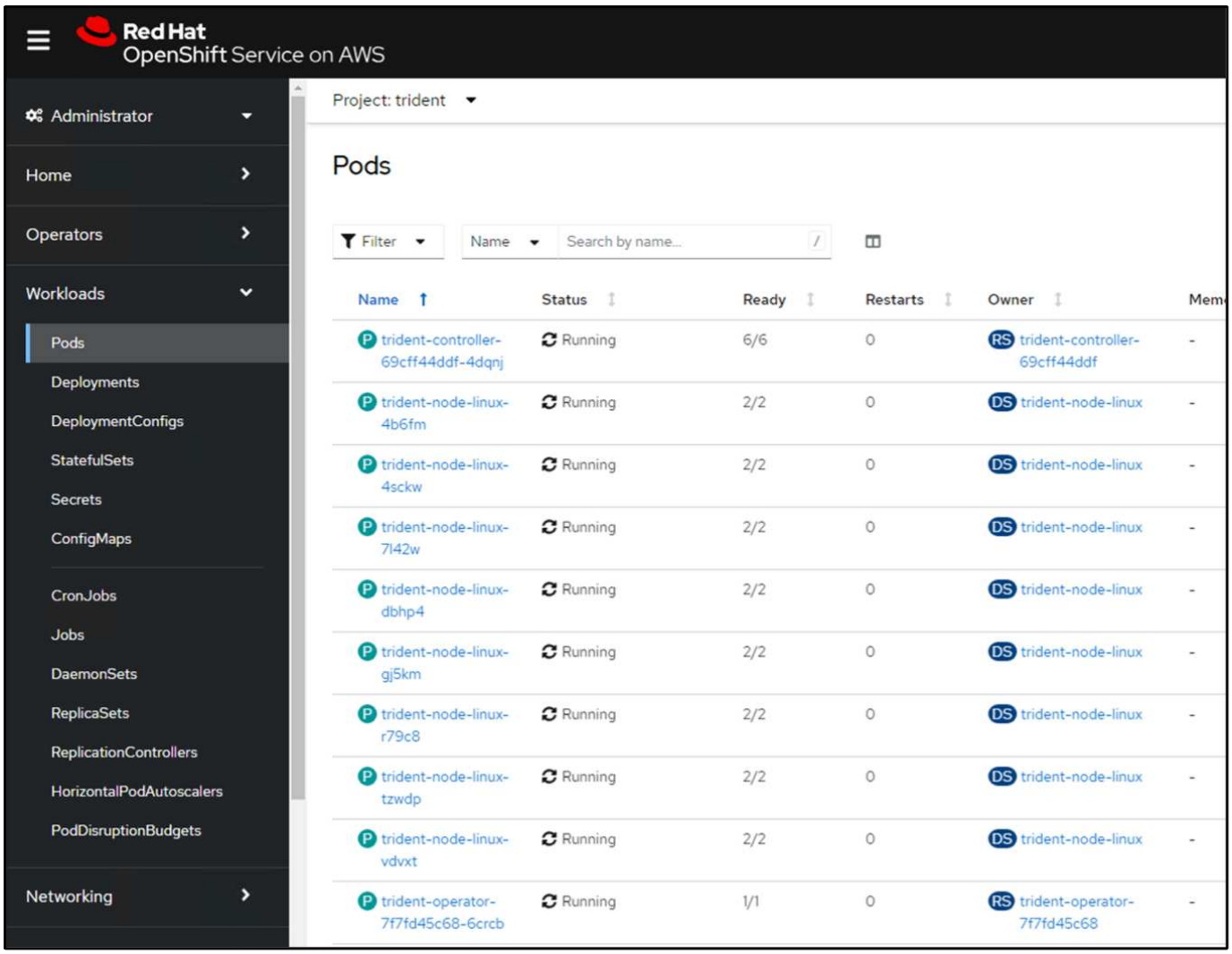


Ensuite, à partir de la vue Développeur, créez un référentiel de graphiques Helm. Pour le champ URL, utilisez `'https://netapp.github.io/trident-helm-chart'` . Créez ensuite une version de barre pour l'opérateur Trident .


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



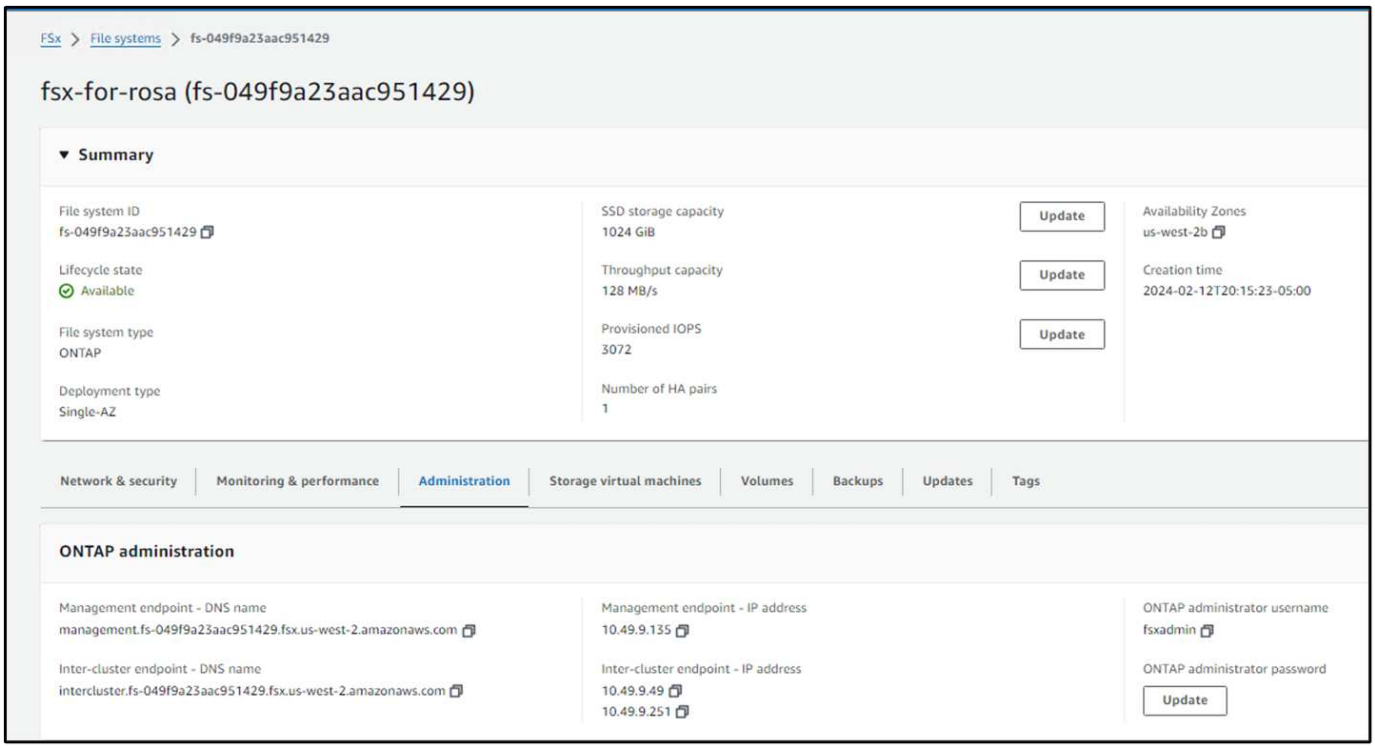
Vérifiez que tous les pods Trident sont en cours d'exécution en revenant à la vue Administrateur sur la console et en sélectionnant les pods dans le projet Trident.



4. Créez une configuration de backend, de classe de stockage et de classe d'instantané à l'aide du provisionneur Trident CSI

Utilisez les fichiers yaml ci-dessous pour créer un objet backend trident, un objet de classe de stockage et l'objet Volumesnapshot. Assurez-vous de fournir les informations d'identification de votre système de fichiers Amazon FSx ONTAP que vous avez créé, le LIF de gestion et le nom du serveur virtuel de votre système de fichiers dans le fichier yaml de configuration pour le backend. Pour obtenir ces détails, accédez à la console AWS pour Amazon FSx et sélectionnez le système de fichiers, accédez à l'onglet Administration. Cliquez également sur Mettre à jour pour définir le mot de passe du `fsxadmin` utilisateur.

Vous pouvez utiliser la ligne de commande pour créer les objets ou les créer avec les fichiers yaml à partir de la console cloud hybride.



- Configuration du backend Trident **

```
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
```

Classe de stockage

```
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
```

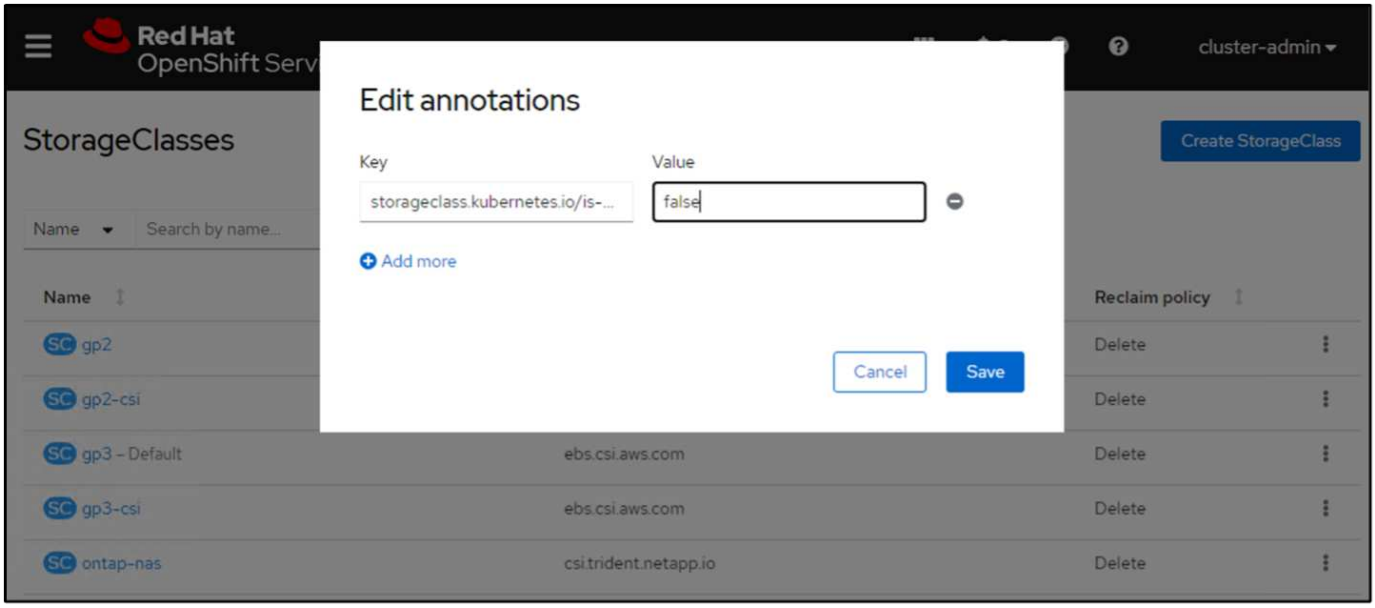
classe d'instantanés

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

Vérifiez que le backend, la classe de stockage et les objets trident-snapshotclass sont créés en exécutant les commandes ci-dessous.

```
[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]$
```

À ce stade, une modification importante que vous devez apporter est de définir ontap-nas comme classe de stockage par défaut au lieu de gp3 afin que l'application postgresql que vous déployez ultérieurement puisse utiliser la classe de stockage par défaut. Dans la console Openshift de votre cluster, sous Stockage, sélectionnez StorageClasses. Modifiez l'annotation de la classe par défaut actuelle pour qu'elle soit fausse et ajoutez l'annotation storageclass.kubernetes.io/is-default-class définie sur true pour la classe de stockage ontap-nas.



5. Déployer une application postgresql sur le cluster

Vous pouvez déployer l'application à partir de la ligne de commande comme suit :

```
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]$
```

Si vous ne voyez pas les pods d'application en cours d'exécution, il peut y avoir une erreur due à des contraintes de contexte de sécurité.

```
[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]$
```

Corrigez l'erreur en modifiant le `runAsUser` et `fsGroup` champs dans `statefuleset.apps/postgresql` objet avec l'uid qui se trouve dans la sortie du `oc get project` commande comme indiqué ci-dessous.

```
[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]$
```

L'application postgresql doit être en cours d'exécution et utiliser des volumes persistants sauvegardés par le stockage Amazon FSx ONTAP .

```
[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]$
```

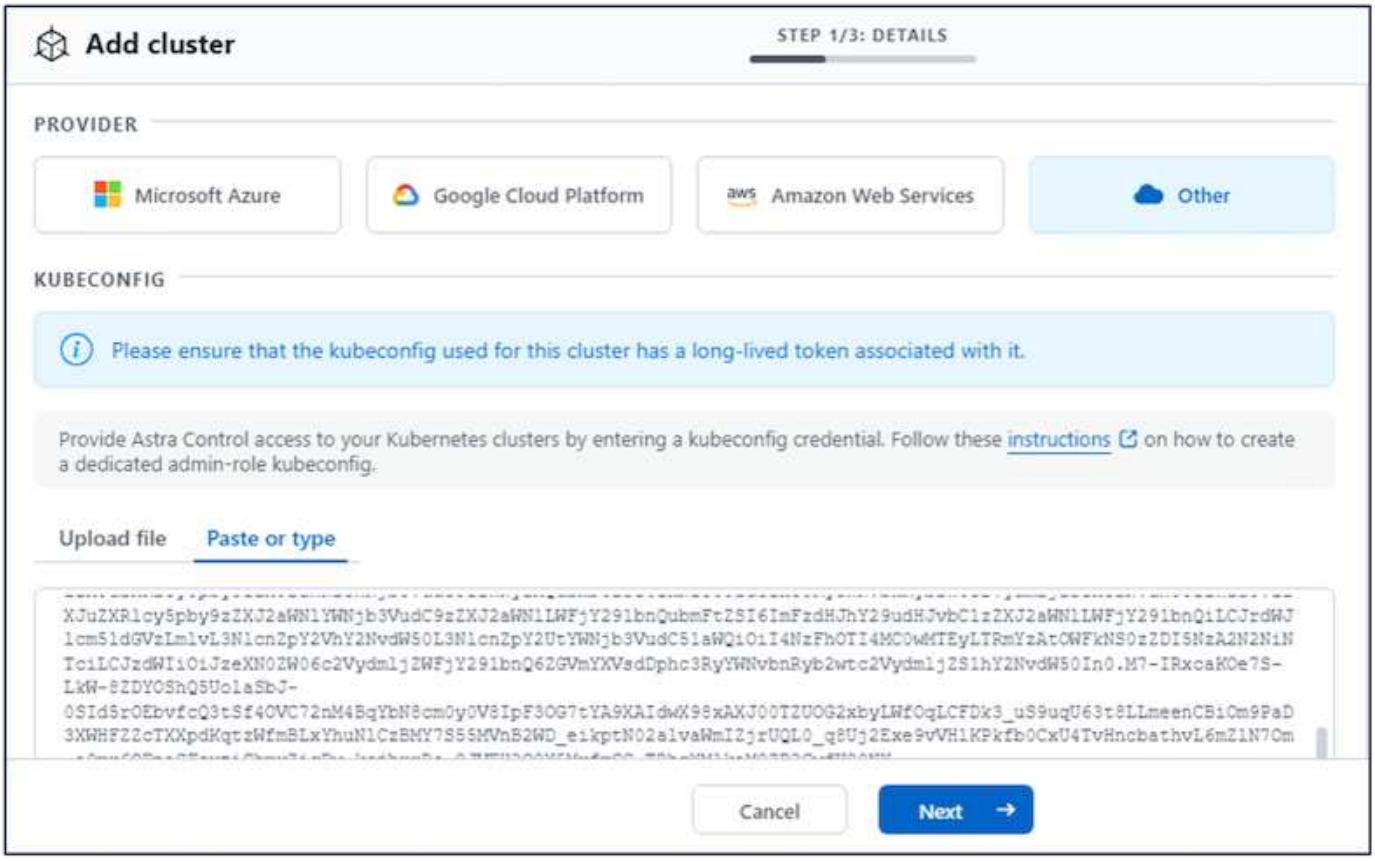
```
[ec2-user@ip-10-49-11-132 storage]$ kubectl get pvc -n postgresql
NAME               STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
data-postgresql-0  Bound    pvc-dd09524a-de75-4825-9424-03a9b91195ca   8Gi        RWO            ontap-nas      4m2s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

6. Créer une base de données et ajouter un enregistrement

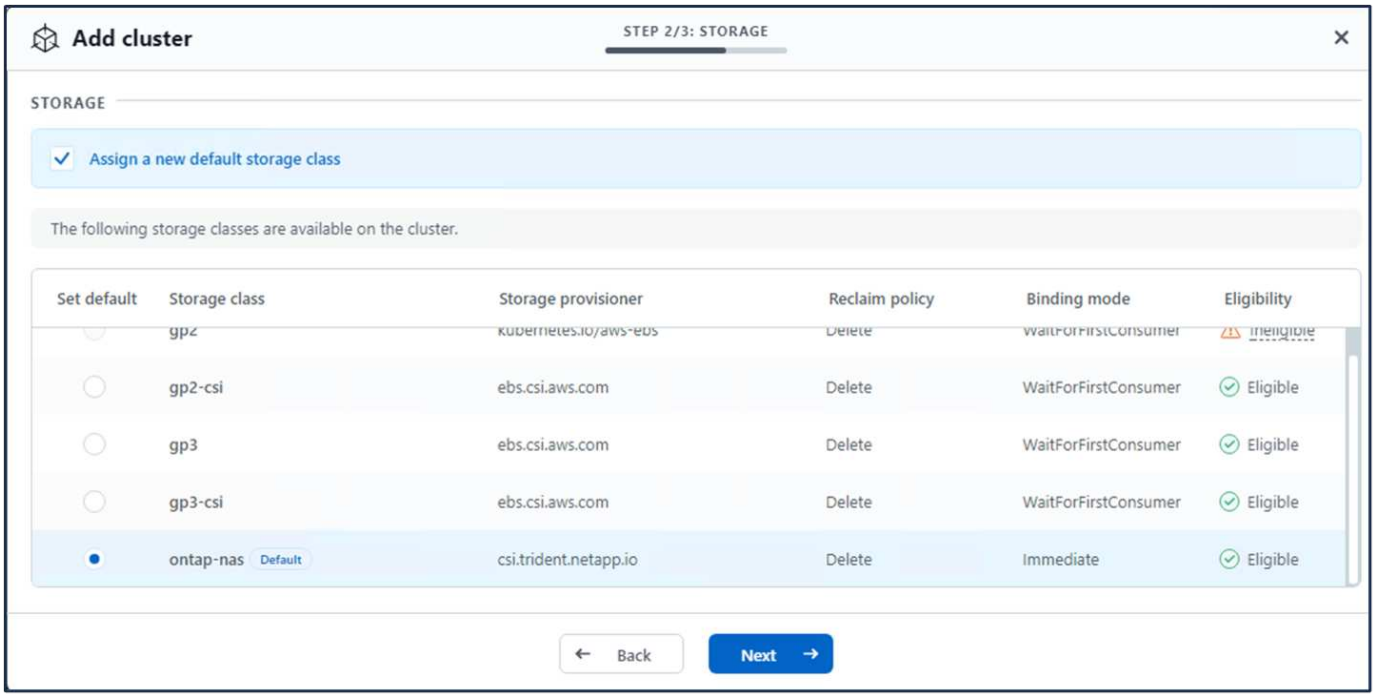
```
[ec2-user@ip-10-49-11-132 astra]$ export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql -o jsonpath="
{.data.postgres-password}" | base64 -d)
[ec2-user@ip-10-49-11-132 astra]$ kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image
docker.io/bitnami/postgresql:16.2.0-debian-11-r1 --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. Ajouter le cluster dans ACS

Connectez-vous à ACS. Sélectionnez le cluster et cliquez sur Ajouter. Sélectionnez autre et téléchargez ou collez le fichier kubeconfig.



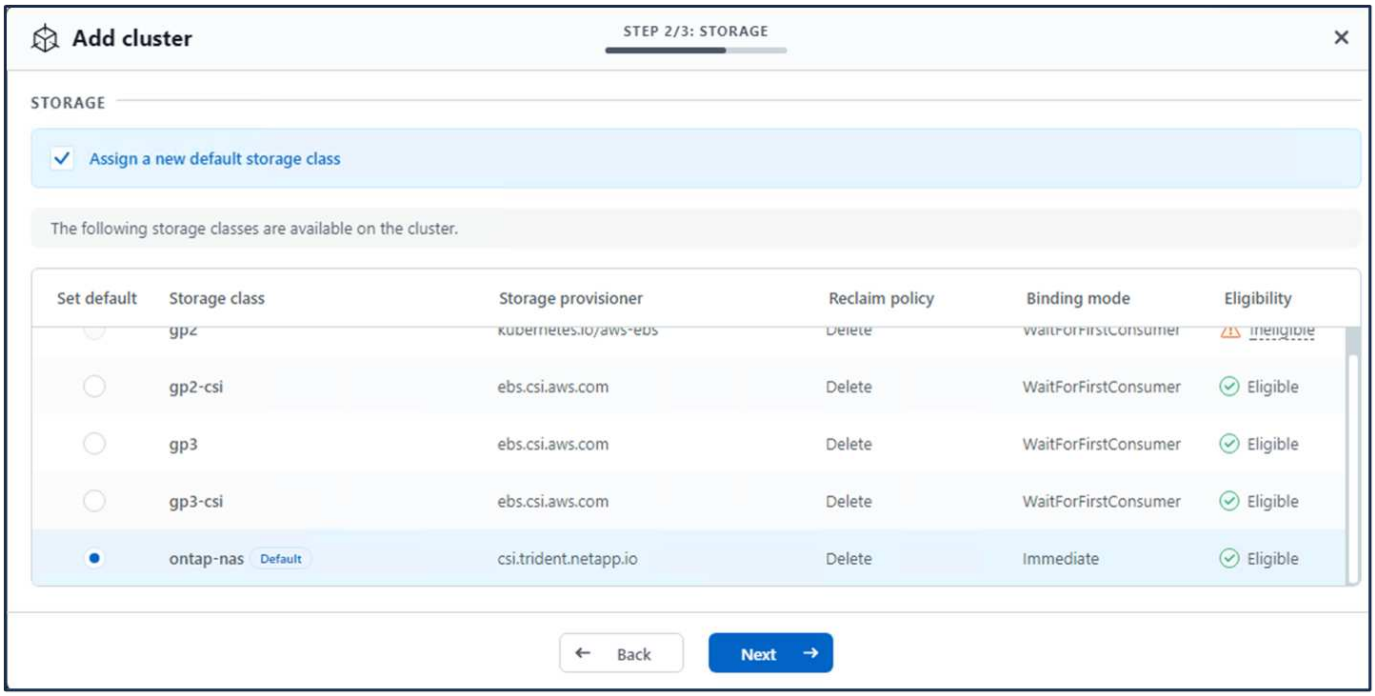
Cliquez sur **Suivant** et sélectionnez ontap-nas comme classe de stockage par défaut pour ACS. Cliquez sur **Suivant**, vérifiez les détails et **Ajoutez** le cluster.



8. Définir l'application dans ACS

Définir l'application postgresql dans ACS. Depuis la page d'accueil, sélectionnez **Applications**, **Définir** et remplissez les détails appropriés. Cliquez sur **Suivant** plusieurs fois, vérifiez les détails et cliquez sur **Définir**.

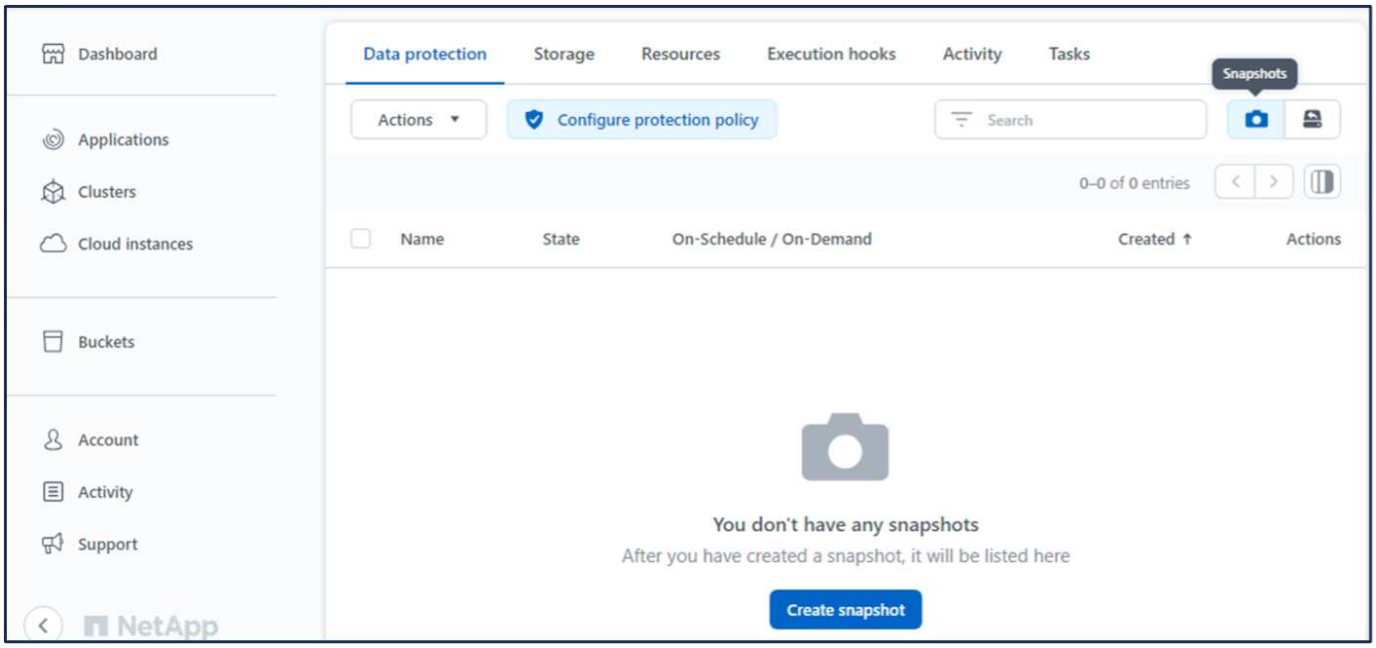
L’application est ajoutée à ACS.

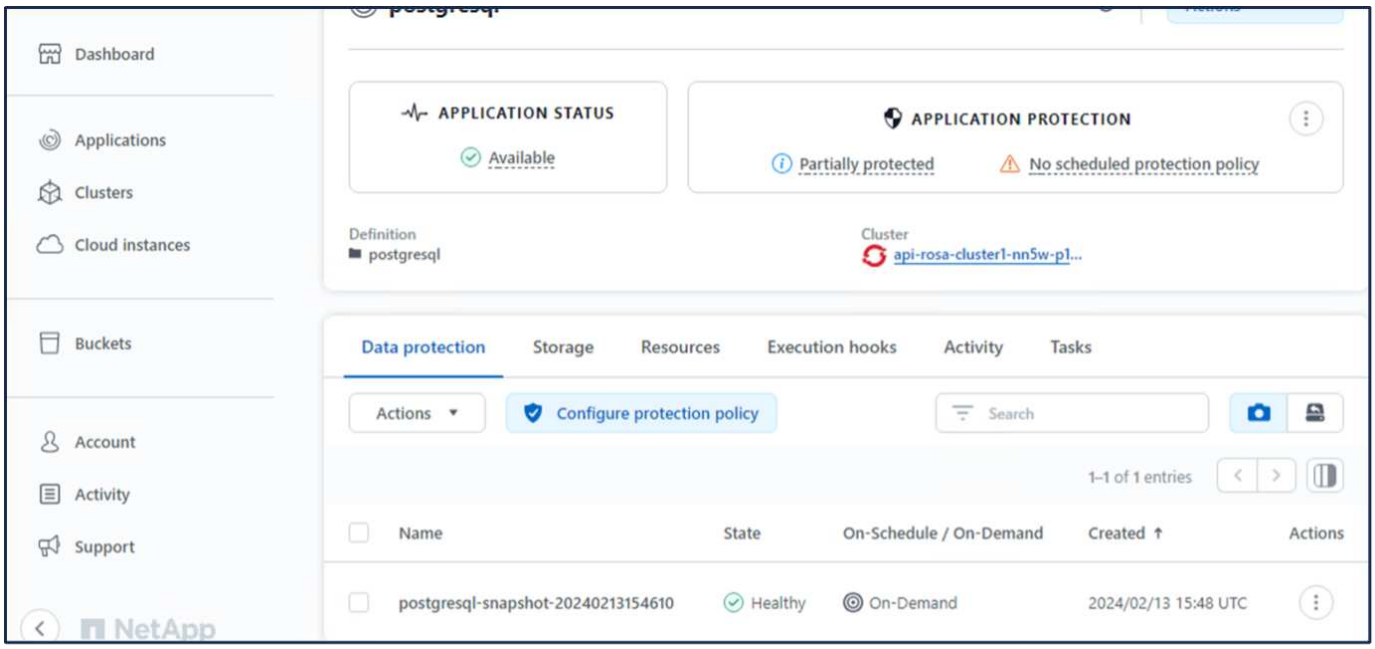


9. Créer un instantané à l’aide d’ACS

Il existe de nombreuses façons de créer un instantané dans ACS. Vous pouvez sélectionner l’application et créer un instantané à partir de la page qui affiche les détails de l’application. Vous pouvez cliquer sur Créer un instantané pour créer un instantané à la demande ou configurer une politique de protection.

Créez un instantané à la demande en cliquant simplement sur **Créer un instantané**, en fournissant un nom, en vérifiant les détails et en cliquant sur **Instantané**. L’état de l’instantané passe à Sain une fois l’opération terminée.





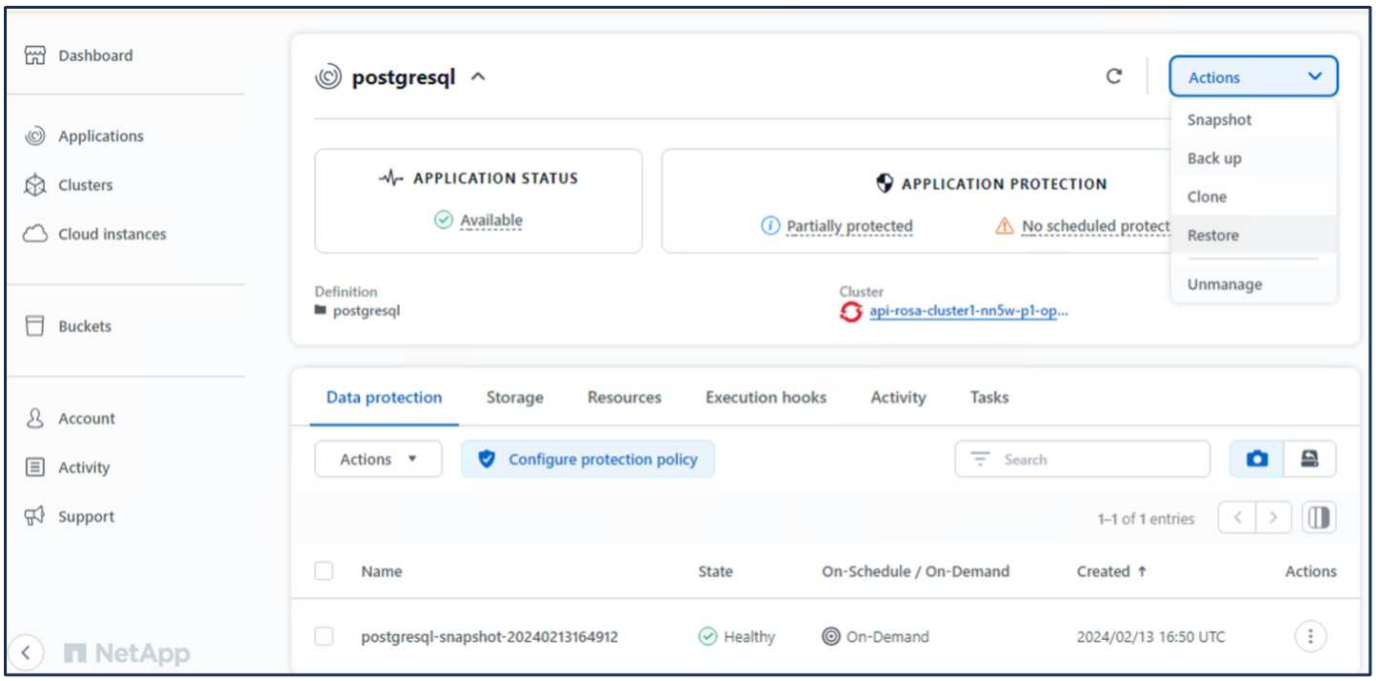
10. Supprimer la base de données dans l'application postgresql

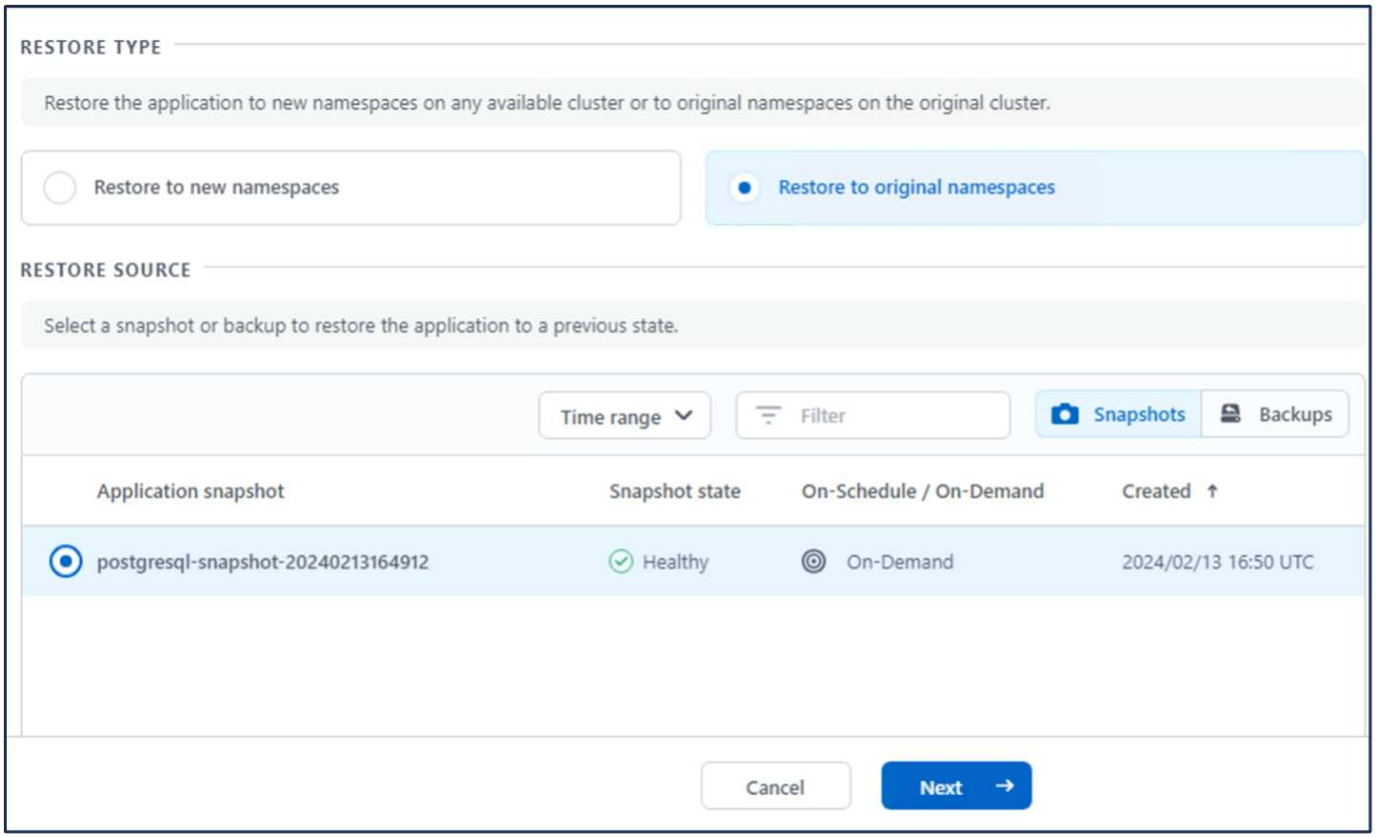
Reconnectez-vous à postgresql, répertoriez les bases de données disponibles, supprimez celle que vous avez créée précédemment et répertoriez-les à nouveau pour vous assurer que la base de données a été supprimée.

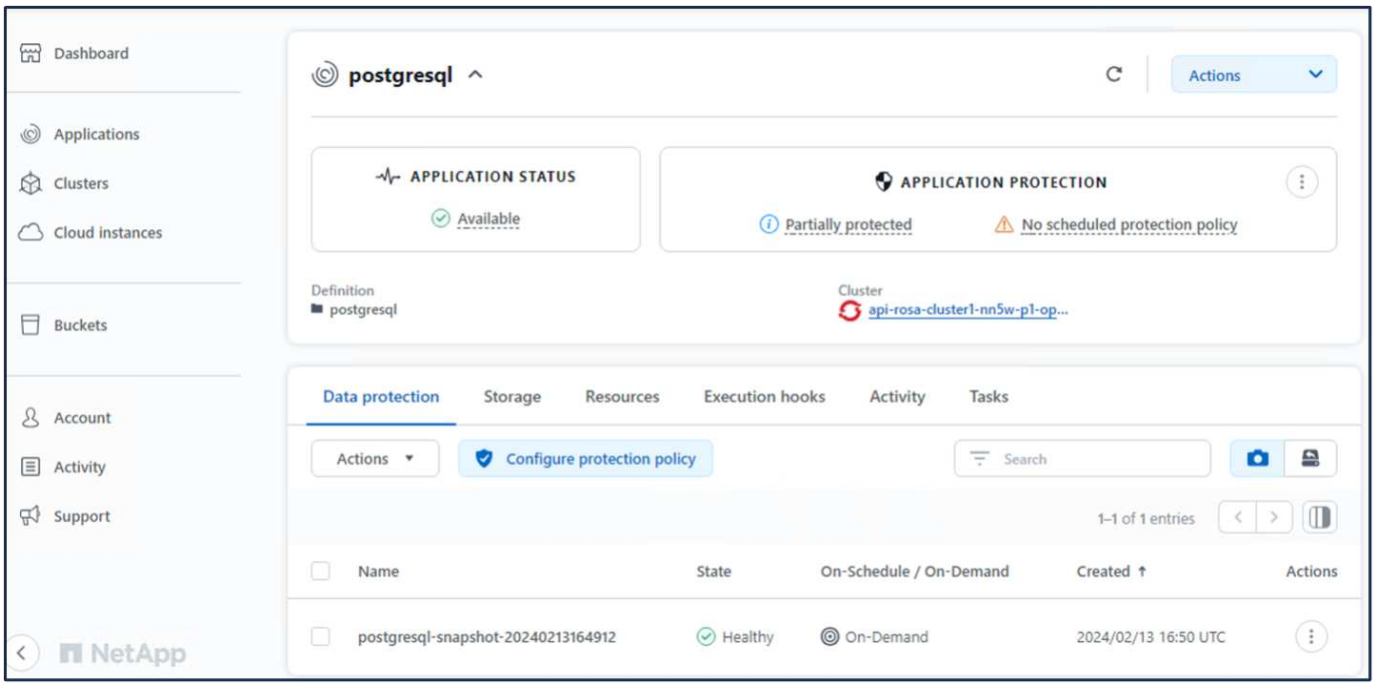
```
postgres=# \l
                                                List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access priv
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+-------------
-
 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. Restaurer à partir d'un instantané à l'aide d'ACS

Pour restaurer l'application à partir d'un instantané, accédez à la page de destination de l'interface utilisateur ACS, sélectionnez l'application et sélectionnez Restaurer. Vous devez choisir un instantané ou une sauvegarde à partir de laquelle effectuer la restauration. (En règle générale, vous en aurez plusieurs créés en fonction d'une politique que vous avez configurée). Faites les choix appropriés dans les deux écrans suivants, puis cliquez sur **Restaurer**. L'état de l'application passe de Restauration à Disponible après avoir été restauré à partir de l'instantané.







12. Vérifiez que votre application a été restaurée à partir de l'instantané

Connectez-vous au client postgresql et vous devriez maintenant voir la table et l'enregistrement dans la table que vous aviez auparavant. C'est ça. En cliquant simplement sur un bouton, votre application a été restaurée à un état antérieur. C'est ainsi que nous facilitons la tâche à nos clients avec 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)
```


Migration des données

Cette page présente les options de migration de données pour les charges de travail de conteneurs sur les clusters Red Hat OpenShift gérés à l'aide de FSx ONTAP pour le stockage persistant.

Migration des données

Le service Red Hat OpenShift sur AWS ainsi qu'Amazon Amazon FSx for NetApp ONTAP (FSx ONTAP) font partie de leur portefeuille de services par AWS. FSx ONTAP est disponible sur les options Single AZ ou Multi-AZ. L'option Multi-Az offre une protection des données contre les pannes de zone de disponibilité. FSx ONTAP peut être intégré à Trident pour fournir un stockage persistant pour les applications sur les clusters ROSA.

Intégration de FSx ONTAP avec Trident à l'aide de la carte Helm

Intégration du cluster ROSA avec Amazon FSx ONTAP

La migration des applications conteneurisées implique :

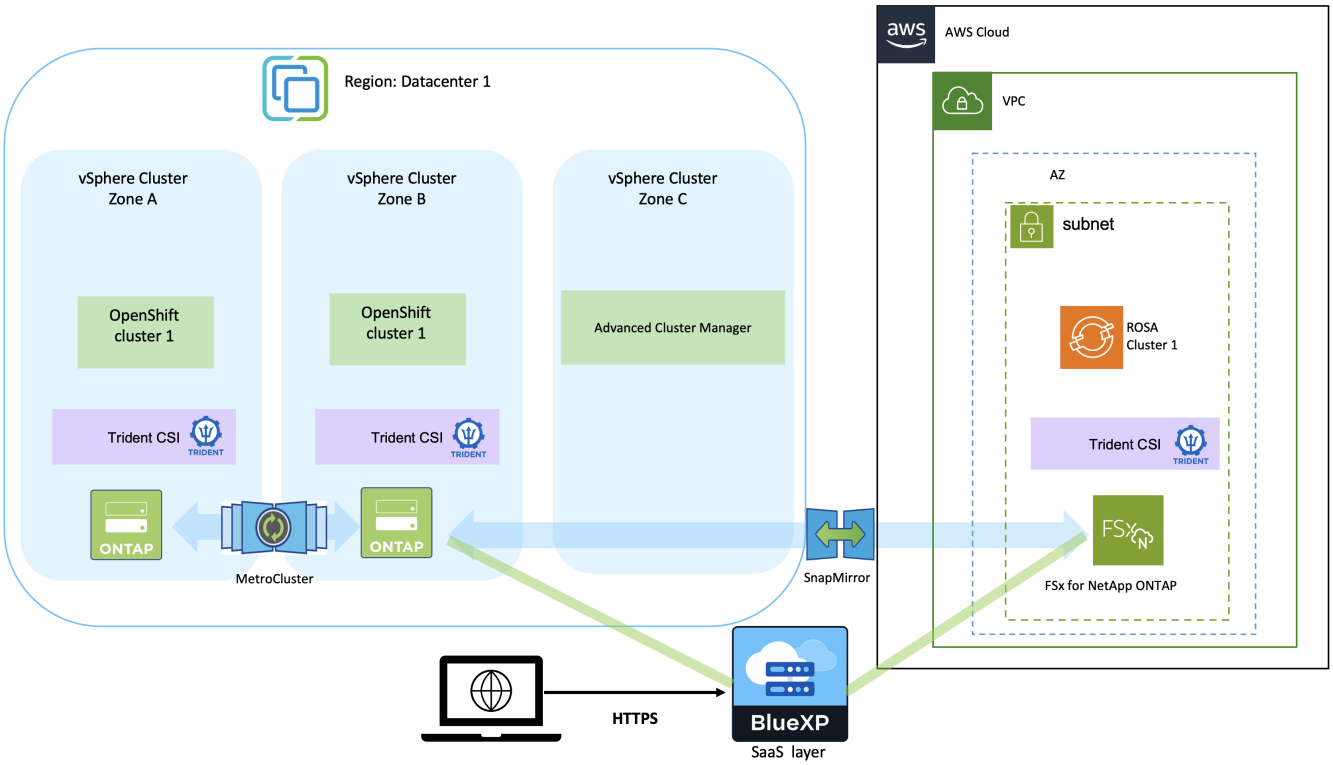
- Volumes persistants : cela peut être réalisé à l'aide de BlueXP. Une autre option consiste à utiliser Trident Protect pour gérer les migrations d'applications de conteneurs depuis l'environnement local vers l'environnement cloud. L'automatisation peut être utilisée dans le même but.
- Métadonnées d'application : cela peut être réalisé à l'aide d'OpenShift GitOps (Argo CD).

Basculement et restauration d'applications sur un cluster ROSA à l'aide de FSx ONTAP pour le stockage persistant

La vidéo suivante est une démonstration des scénarios de basculement et de restauration d'applications à l'aide de BlueXP et d'Argo CD.

Basculement et restauration des applications sur le cluster ROSA

Solution de protection et de migration des données pour les charges de travail des conteneurs OpenShift



Solutions NetApp Hybrid Multicloud supplémentaires pour les charges de travail Red Hat OpenShift

Solutions supplémentaires

Des solutions supplémentaires sont disponibles dans d'autres sections comme suit :

Pour les solutions Red Hat OpenShift Container, voir"ici" .

Pour les solutions de virtualisation Red Hat OpenShift sur site, voir"ici" .