Servicio Red Hat OpenShift en AWS con FSxN

NetApp container solutions

NetApp
January 21, 2026

Tabla de contenidos

Servicio Red Hat OpenShift en AWS con FSxN

Servicio Red Hat OpenShift en AWS con NetApp ONTAP

Descripción general

En esta sección, mostraremos cómo utilizar FSx para ONTAP como una capa de almacenamiento persistente para aplicaciones que se ejecutan en ROSA. Se mostrará la instalación del controlador NetApp Trident CSI en un clúster ROSA, el aprovisionamiento de un sistema de archivos FSx para ONTAP y la implementación de una aplicación con estado de muestra. También mostrará estrategias para realizar copias de seguridad y restaurar los datos de su aplicación. Con esta solución integrada, puede establecer un marco de almacenamiento compartido que se escala sin esfuerzo entre zonas de disponibilidad, lo que simplifica los procesos de escalamiento, protección y restauración de sus datos mediante el controlador Trident CSI.

Prerrequisitos

- "Cuenta de AWS"
- "Una cuenta de Red Hat"
- Usuario de IAM"con los permisos apropiados" para crear y acceder al clúster ROSA
- "CLI de AWS"
- "ROSA CLI"
- "Interfaz de línea de comandos de OpenShift"(jefe)
- Timón 3"documentación"
- "Un grupo HCP ROSA"
- "Acceso a la consola web de Red Hat OpenShift"

Este diagrama muestra el clúster ROSA implementado en múltiples AZ. Los nodos maestros y los nodos de infraestructura del clúster ROSA están en la VPC de Red Hat, mientras que los nodos de trabajo están en una VPC en la cuenta del cliente. Crearemos un sistema de archivos FSx para ONTAP dentro de la misma VPC e instalaremos el controlador Trident en el clúster ROSA, lo que permitirá que todas las subredes de esta VPC se conecten al sistema de archivos.

Configuración inicial

1. Aprovisionamiento de FSx para NetApp ONTAP

Cree un FSx multi-AZ para NetApp ONTAP en la misma VPC que el clúster ROSA. Hay varias maneras de hacer esto. Se proporcionan los detalles de la creación de FSxN utilizando una pila CloudFormation

a. Clonar el repositorio de GitHub

```
$ git clone https://github.com/aws-samples/rosa-fsx-netapp-ontap.git
```

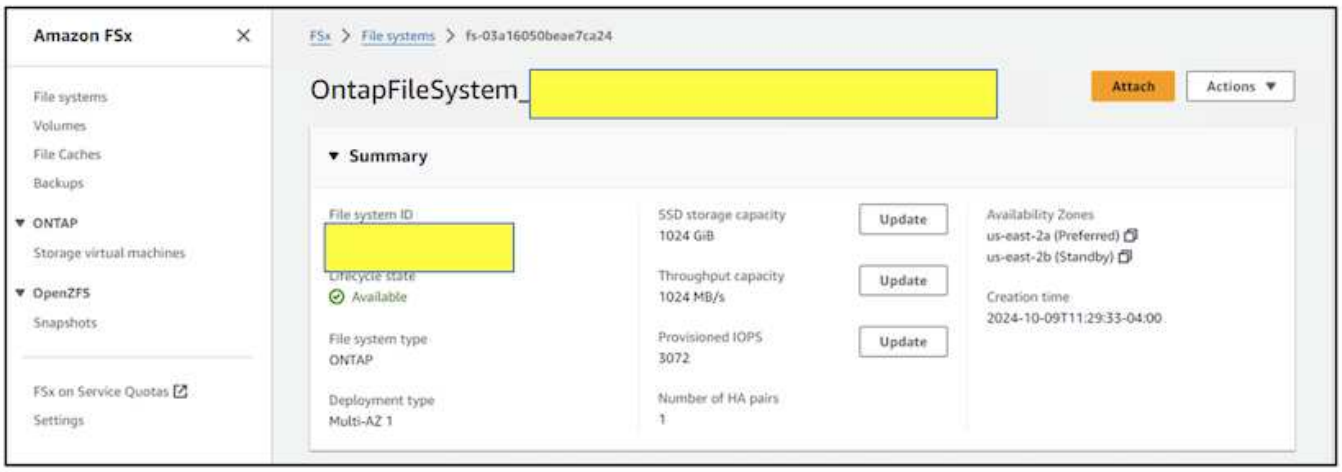
b.Ejecute la pila CloudFormation Ejecute el siguiente comando reemplazando los valores de los parámetros con sus propios valores:

```
$ cd rosa-fsx-netapp-ontap/fsx
```

```
$ aws cloudformation create-stack \
  --stack-name ROSA-FSXONTAP \
  --template-body file://./FSxONTAP.yaml \
  --region <region-name> \
  --parameters \
  ParameterKey=Subnet1ID,ParameterValue=[subnet1_ID] \
  ParameterKey=Subnet2ID,ParameterValue=[subnet2_ID] \
  ParameterKey=myVpc,ParameterValue=[VPC_ID] \
ParameterKey=FSxONTAPRouteTable,ParameterValue=[routetable1_ID,routetable2
_ID] \
  ParameterKey=FileSystemName,ParameterValue=ROSA-myFSxONTAP \
  ParameterKey=ThroughputCapacity,ParameterValue=1024 \
  ParameterKey=FSxAllowedCIDR,ParameterValue=[your_allowed_CIDR] \
  ParameterKey=FsxAdminPassword,ParameterValue=[Define Admin password] \
  ParameterKey=SvmAdminPassword,ParameterValue=[Define SVM password] \
  --capabilities CAPABILITY_NAMED_IAM
```

Dónde: region-name: igual que la región donde está implementado el clúster ROSA subnet1_ID: id de la subred preferida para FSxN subnet2_ID: id de la subred en espera para FSxN VPC_ID: id de la VPC donde está implementado el clúster ROSA routetable1_ID, routetable2_ID: id de las tablas de rutas asociadas con las subredes elegidas anteriormente your_allowed_CIDR: rango de CIDR permitido para las reglas de ingreso de los grupos de seguridad de FSx for ONTAP para controlar el acceso. Puede utilizar 0.0.0.0/0 o cualquier CIDR apropiado para permitir que todo el tráfico acceda a los puertos específicos de FSx para ONTAP. Definir contraseña de administrador: una contraseña para iniciar sesión en FSxN Definir contraseña de SVM: una contraseña para iniciar sesión en SVM que se creará.

Verifique que su sistema de archivos y su máquina virtual de almacenamiento (SVM) se hayan creado mediante la consola de Amazon FSx , como se muestra a continuación:



2. Instale y configure el controlador Trident CSI para el clúster ROSA

b.Instalar Trident

Los nodos de trabajo del clúster ROSA vienen preconfigurados con herramientas NFS que le permiten utilizar protocolos NAS para el aprovisionamiento y acceso al almacenamiento.

Si desea utilizar iSCSI en su lugar, deberá preparar los nodos de trabajo para iSCSI. A partir de la versión Trident 25.02, puede preparar fácilmente los nodos de trabajo del clúster ROSA (o cualquier clúster OpenShift) para realizar operaciones iSCSI en el almacenamiento FSxN. Hay dos formas sencillas de instalar Trident 25.02 (o posterior) que automatiza la preparación del nodo de trabajo para iSCSI. 1. usando el indicador node-prep-flag desde la línea de comando usando la herramienta tridentctl. 2. Utilizar el operador Trident certificado por Red Hat desde el centro de operadores y personalizarlo. 3.Usando Helm.

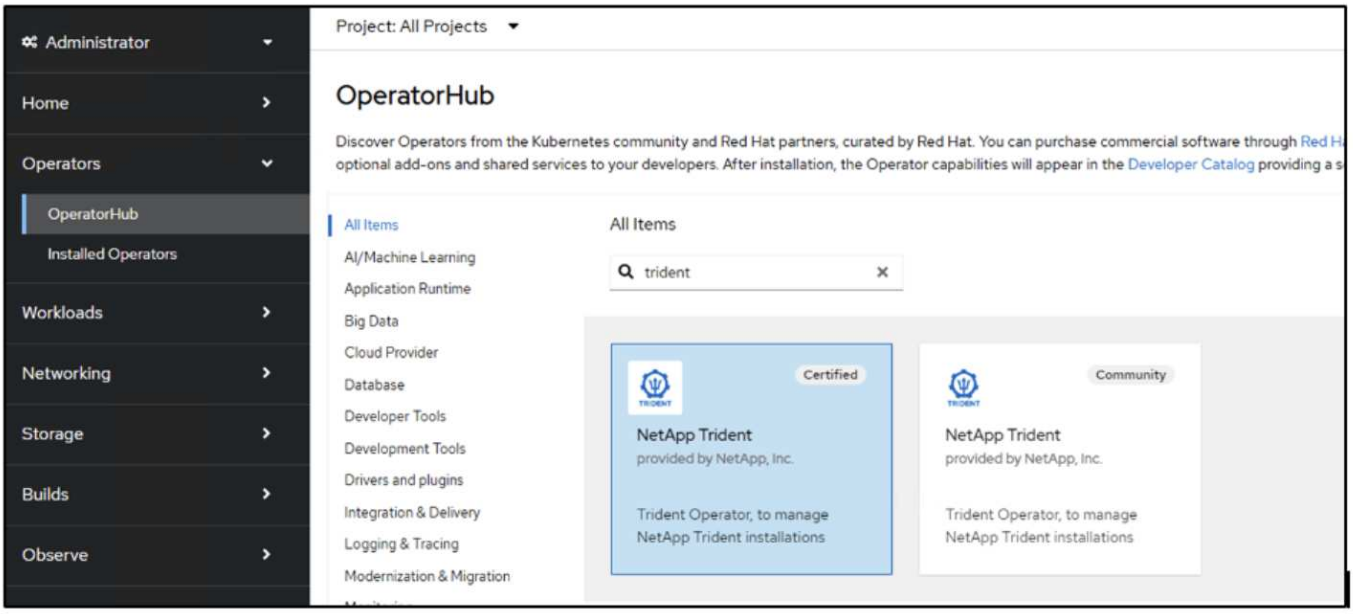
Si utiliza cualquiera de los métodos anteriores sin habilitar la preparación del nodo, podrá utilizar únicamente protocolos NAS para aprovisionar almacenamiento en FSxN.

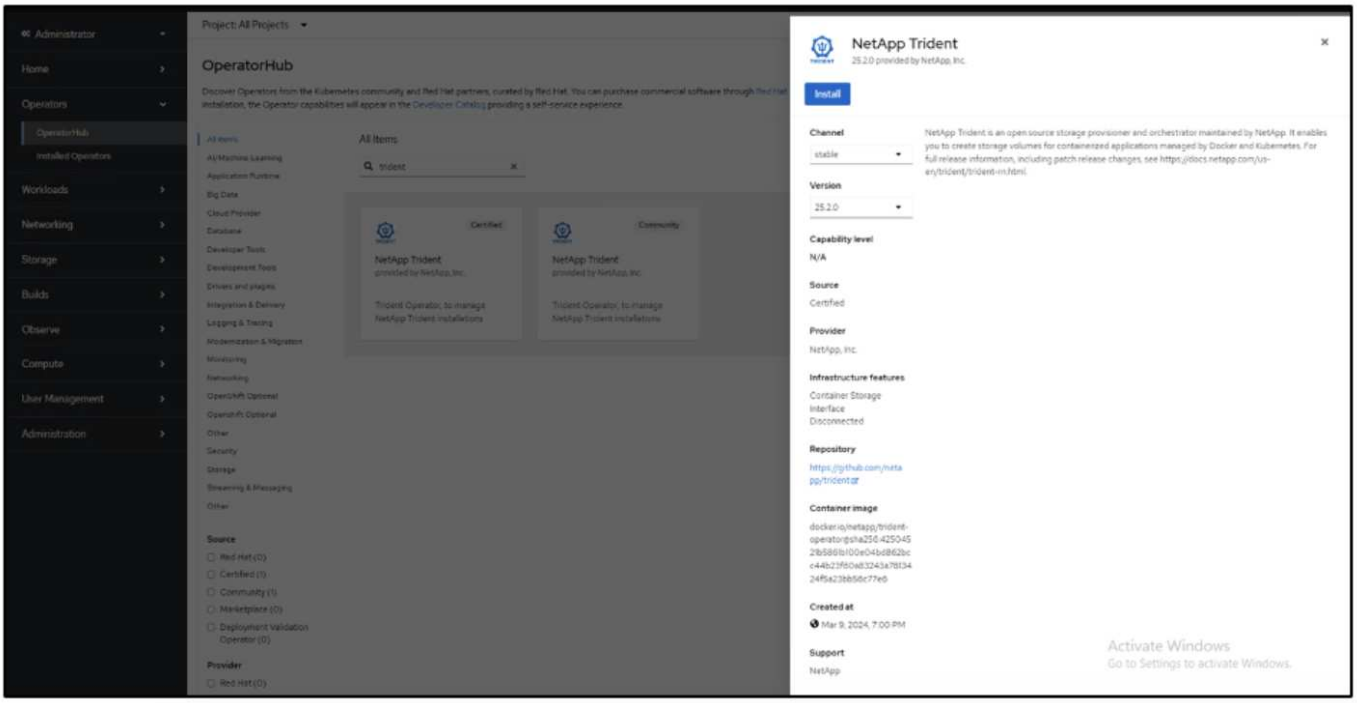
Método 1: utilizar la herramienta tridentctl

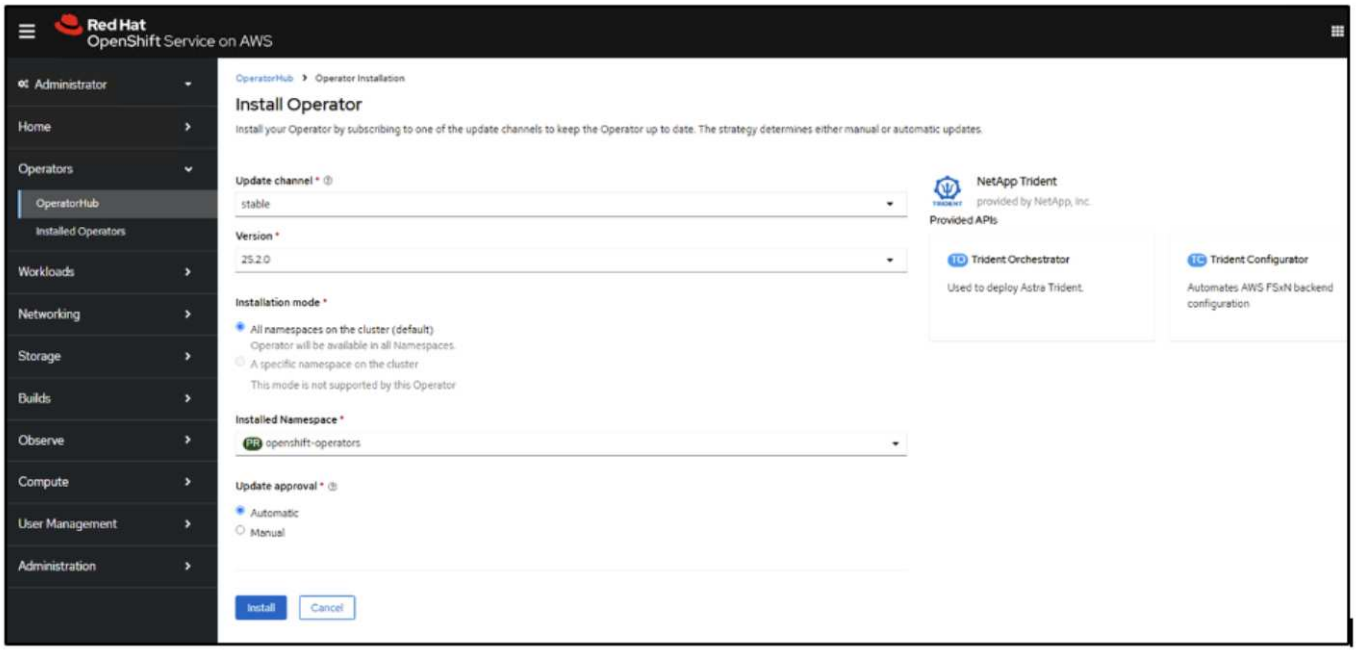
Utilice el indicador node-prep e instale Trident como se muestra. Antes de emitir el comando de instalación, debe haber descargado el paquete de instalación. Consulte "La documentación aquí" .

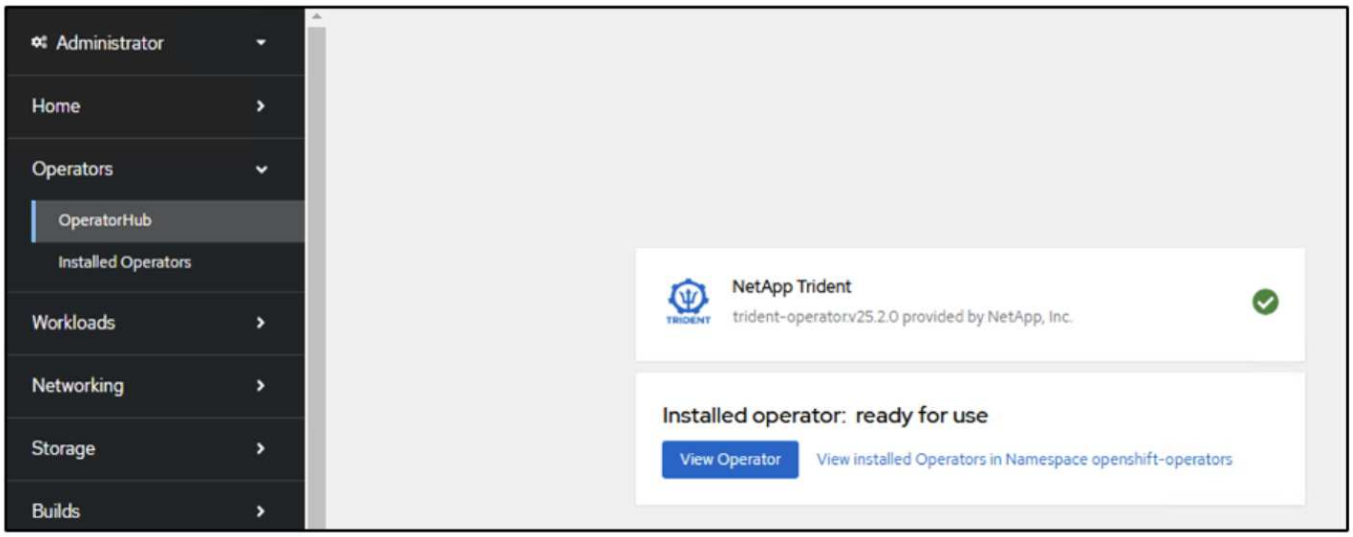
```
#./tridentctl install trident -n trident --node-prep=iscsi
```

Método 2: utilizar el operador Trident certificado por Red Hat y personalizarlo Desde OperatorHub, busque el operador Trident certificado por Red Hat e instálelo.

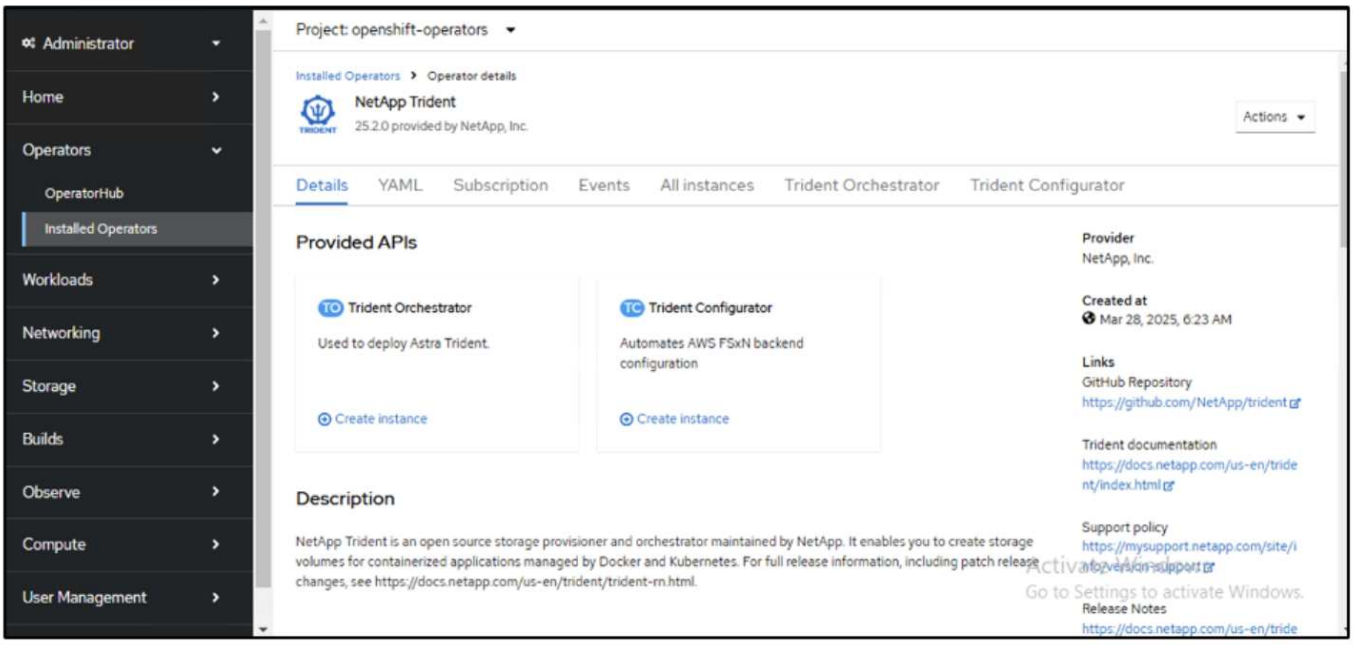


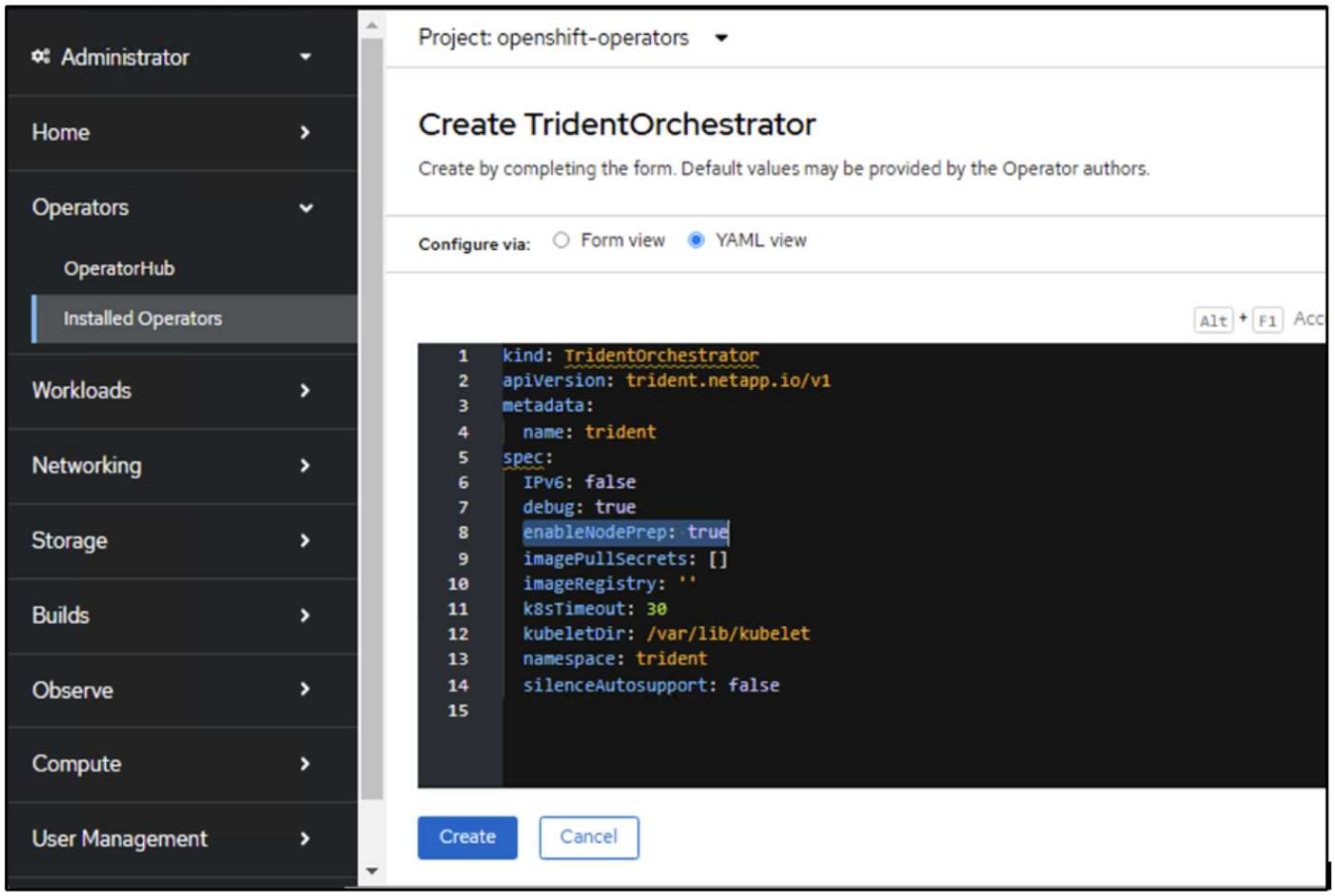

Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: All Projects
OperatorHub
All Items
trident
NetApp Trident
provided by NetApp, Inc.
Trident Operator, to manage NetApp Trident installations
Certified
Community
NetApp Trident
25.2.0 provided by NetApp, Inc.
Install
Channel
stable
Version
25.2.0
Capability level
N/A
Source
Certified
Provider
NetApp, Inc.
Infrastructure features
Container Storage Interface
Disconnected
Repository
Container image
Created at
Mar 9, 2024, 7:00 PM
Support
NetApp
Activate Windows
Go to Settings to activate Windows.

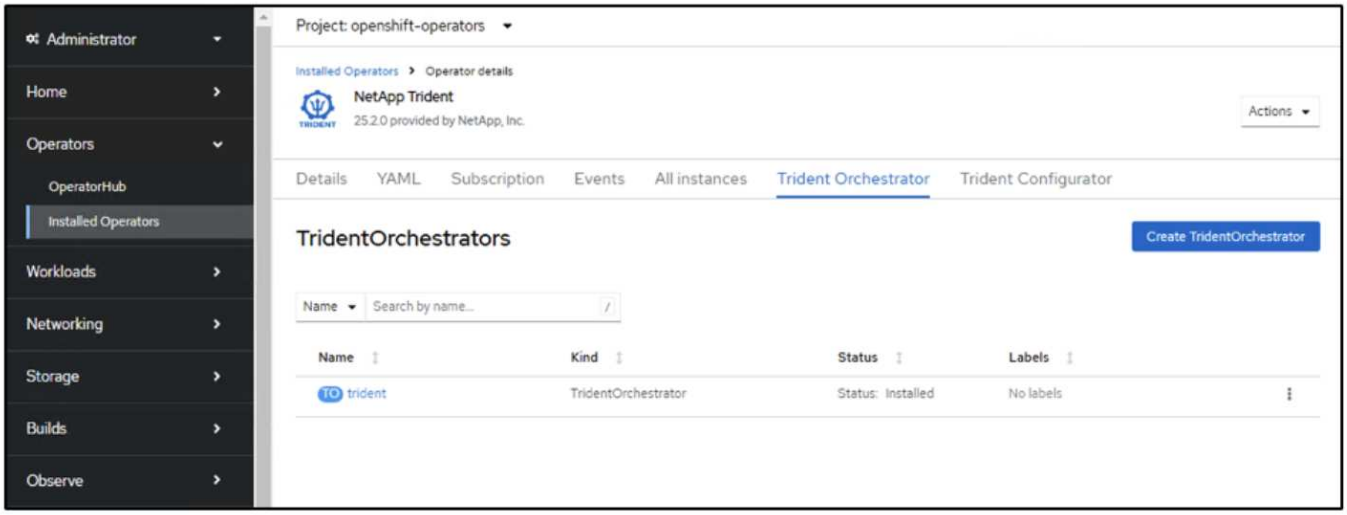

Red Hat
OpenShift Service on AWS
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OperatorHub > Operator Installation
Install Operator
Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.
Update channel *
stable
Version *
25.2.0
Installation mode *
All namespaces on the cluster (default)
Operator will be available in all Namespaces.
A specific namespace on the cluster
This mode is not supported by this Operator
Installed Namespace *
openshift-operators
Update approval *
Automatic
Manual
Install
Cancel
NetApp Trident
provided by NetApp, Inc.
Provided APIs
Trident Orchestrator
Used to deploy Astra Trident.
Trident Configurator
Automates AWS FSxN backend configuration



A continuación, cree la instancia de Trident Orchestrator. Utilice la vista YAML para establecer valores personalizados o habilitar la preparación del nodo iscsi durante la instalación.








```
[root@localhost RedHat]# oc get pods -n trident
NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-86f89c855d-8w2jx   6/6     Running   0          38s
trident-node-linux-rnrnn              2/2     Running   0          38s
trident-node-linux-t9bxj              2/2     Running   0          38s
trident-node-linux-vqvl9              2/2     Running   0          38s
[root@localhost RedHat]#
```

La instalación de Trident mediante cualquiera de los métodos anteriores preparará los nodos de trabajo del clúster ROSA para iSCSI iniciando los servicios iscsid y multipathd y configurando lo siguiente en el archivo /etc/multipath.conf

```
sh-5.1#
sh-5.1# systemctl status iscsid
● iscsid.service - Open-iSCSI
     Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled; preset: disabled)
     Active: active (running) since Fri 2025-03-21 18:28:13 UTC; 3 days ago
TriggeredBy: ● iscsid.socket
       Docs: man:iscsid(8)
             man:iscsiuio(8)
             man:iscsiadm(8)
   Main PID: 23224 (iscsid)
     Status: "Ready to process requests"
      Tasks: 1 (limit: 1649420)
     Memory: 3.2M
        CPU: 109ms
     CGroup: /system.slice/iscsid.service
             └─23224 /usr/sbin/iscsid -f
sh-5.1#
```

```
sh-5.1#
sh-5.1# systemctl status multipathd
● multipathd.service - Device-Mapper Multipath Device Controller
     Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-03-21 18:20:50 UTC; 3 days ago
TriggeredBy: ● multipathd.socket
   Main PID: 1565 (multipathd)
     Status: "up"
      Tasks: 7
     Memory: 62.4M
        CPU: 33min 51.363s
     CGroup: /system.slice/multipathd.service
             └─1565 /sbin/multipathd -d -s
```

```
sh-5.1#
sh-5.1# cat /etc/multipath.conf
defaults {
    find_multipaths     no
    user_friendly_names yes
}
blacklist {
}
blacklist_exceptions {
    device {
        vendor  NETAPP
        product LUN
    }
}
sh-5.1#
```

c.Verifique que todos los pods Trident estén en estado de ejecución

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get pods -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-f5f6796f-vd2sk   6/6     Running   0          19h
trident-node-linux-4svgz            2/2     Running   0          19h
trident-node-linux-dj9j4            2/2     Running   0          19h
trident-node-linux-jlshh            2/2     Running   0          19h
trident-node-linux-sqthw            2/2     Running   0          19h
trident-node-linux-ttj9c            2/2     Running   0          19h
trident-node-linux-vmjr5            2/2     Running   0          19h
trident-node-linux-wvqsf            2/2     Running   0          19h
trident-operator-545869857c-kgc7p   1/1     Running   0          19h
[root@localhost hcp-testing]#
```

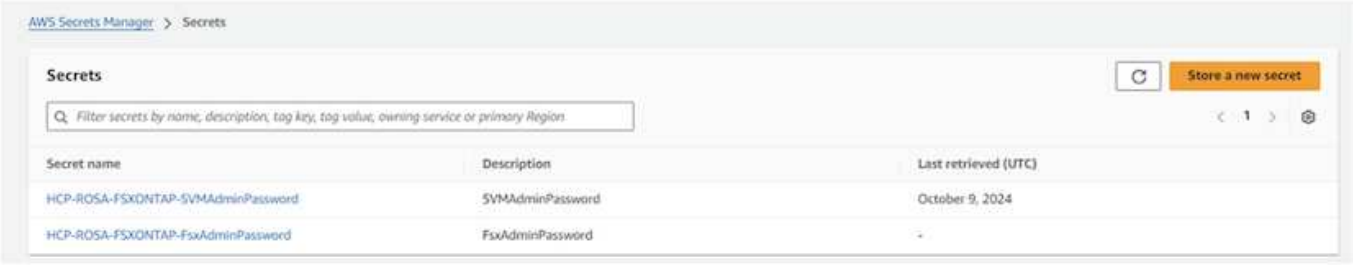
3. Configurar el backend Trident CSI para usar FSx para ONTAP (ONTAP NAS)

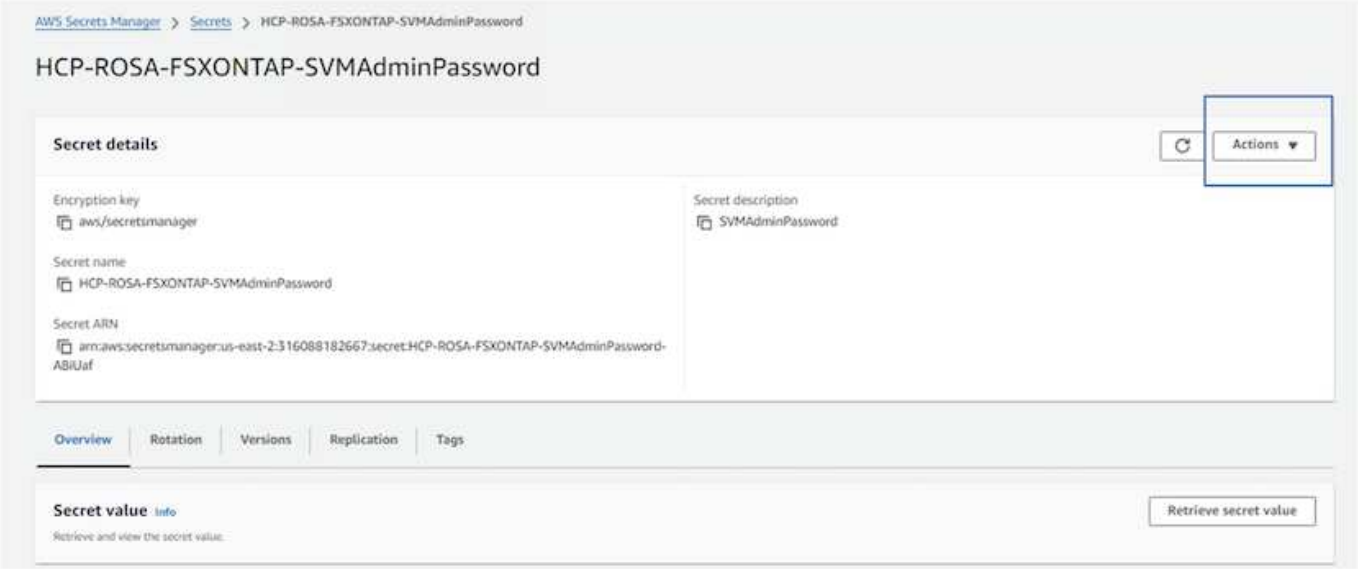
La configuración del back-end de Trident le dice a Trident cómo comunicarse con el sistema de almacenamiento (en este caso, FSx para ONTAP). Para crear el backend, proporcionaremos las credenciales de la máquina virtual de almacenamiento a la que conectarse, junto con las interfaces de administración de clúster y de datos NFS. Usaremos el"controlador ontap-nas" para aprovisionar volúmenes de almacenamiento en el sistema de archivos FSx.

a. Primero, cree un secreto para las credenciales de SVM usando el siguiente yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-fsx-ontap-nas-secret
  namespace: trident
type: Opaque
stringData:
  username: vsadmin
  password: <value provided for Define SVM password as a parameter to the
Cloud Formation Stack>
```

También puede recuperar la contraseña SVM creada para FSxN desde AWS Secrets Manager como se muestra a continuación.





b. A continuación, agregue el secreto de las credenciales de SVM al clúster ROSA usando el siguiente comando

```
$ oc apply -f svm_secret.yaml
```

Puede verificar que el secreto se haya agregado en el espacio de nombres trident usando el siguiente comando

```
$ oc get secrets -n trident |grep backend-fsx-ontap-nas-secret
```

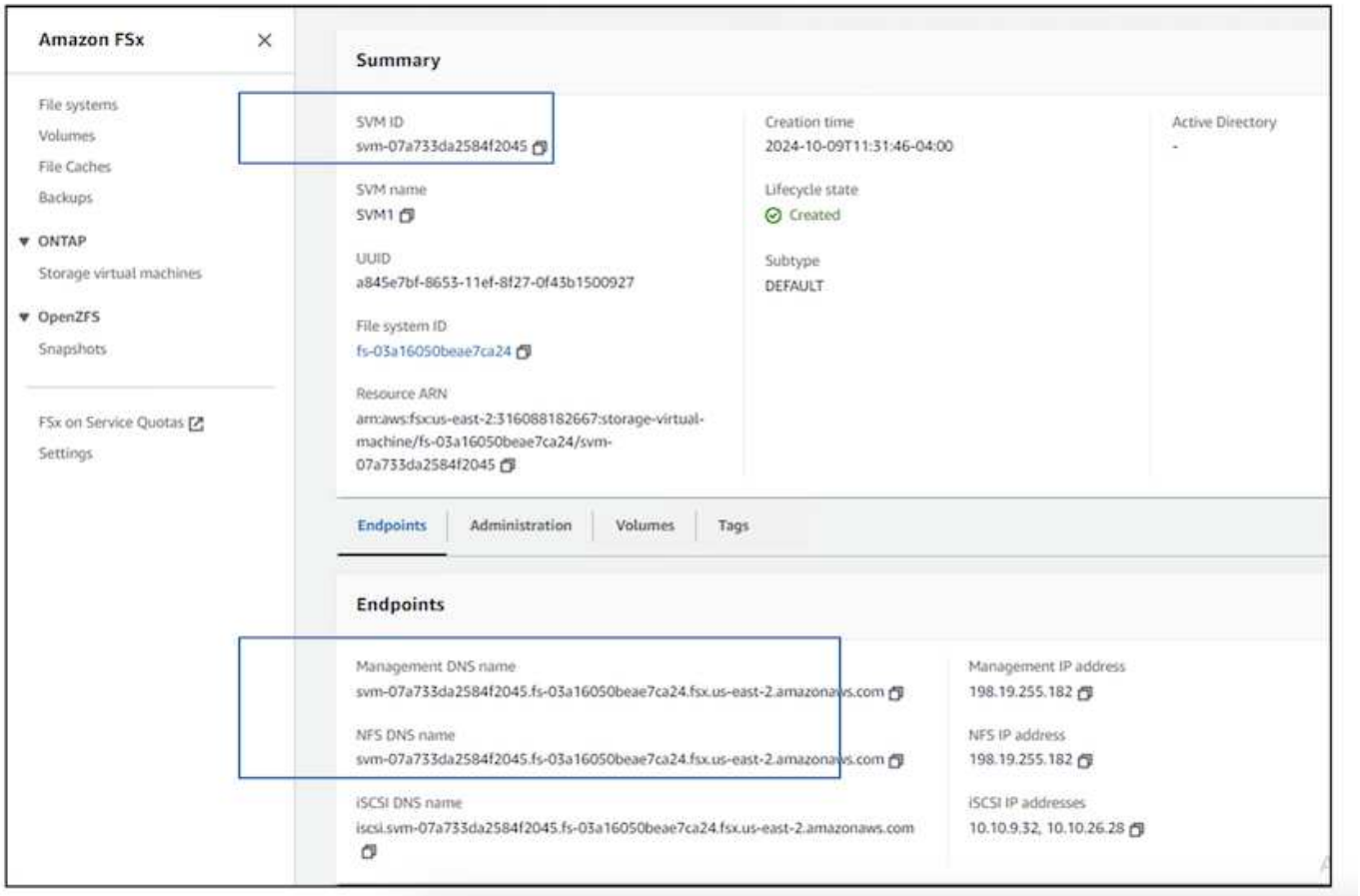
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

do. A continuación, crea el objeto backend. Para ello, dirígete al directorio **fsx** de tu repositorio Git clonado. Abra el archivo backend-ontap-nas.yaml. Reemplace lo siguiente: **managementLIF** con el nombre de DNS de administración, **dataLIF** con el nombre de DNS NFS de la SVM de Amazon FSx y **svm** con el nombre de la SVM. Cree el objeto backend usando el siguiente comando.

Cree el objeto backend usando el siguiente comando.

```
$ oc apply -f backend-ontap-nas.yaml
```

Puede obtener el nombre DNS de administración, el nombre DNS de NFS y el nombre SVM desde la consola de Amazon FSx como se muestra en la siguiente captura de pantalla.



**d. Ahora, ejecute el siguiente comando para verificar que se haya creado el objeto backend y que la fase muestre "Bound" y el estado sea "Éxito".

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc apply -f backend-ontap-nas.yaml
tridentbackendconfig.trident.netapp.io/backend-fsx-ontap-nas created
[root@localhost hcp-testing]# oc get tbc -n trident
NAME                  BACKEND NAME   BACKEND UUID                           PHASE   STATUS
backend-fsx-ontap-nas   fsx-ontap      acc65405-56be-4719-999d-27b448a50e29   Bound   Success
[root@localhost hcp-testing]#
```

4. Crear una clase de almacenamiento Ahora que el backend de Trident está configurado, puede crear una clase de almacenamiento de Kubernetes para usar el backend. La clase de almacenamiento es un objeto de recurso disponible para el clúster. Describe y clasifica el tipo de almacenamiento que puedes solicitar para una aplicación.

a. Revise el archivo storage-class-csi-nas.yaml en la carpeta fsx.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: trident-csi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: True
reclaimPolicy: Retain
```

b. Cree una clase de almacenamiento en el clúster ROSA y verifique que se haya creado la clase de almacenamiento trident-csi.

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc apply -f storage-class-csi-nas.yaml
storageclass.storage.k8s.io/trident-csi created
[root@localhost hcp-testing]# oc get sc
NAME              PROVISIONER             RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
gp2-csi           ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   2d16h
gp3-csi (default)   ebs.csi.aws.com         Delete          WaitForFirstConsumer   true                   2d16h
trident-csi       csi.trident.netapp.io   Retain          Immediate              true                   4s
[root@localhost hcp-testing]#
```

Esto completa la instalación del controlador Trident CSI y su conectividad al sistema de archivos FSx para ONTAP . Ahora puede implementar una aplicación con estado de Postgresql de muestra en ROSA usando volúmenes de archivos en FSx para ONTAP.

do. Verifique que no haya PVC ni PV creados utilizando la clase de almacenamiento trident-csi.

d. Verifique que las aplicaciones puedan crear PV usando Trident CSI.

Cree un PVC utilizando el archivo pvc-trident.yaml proporcionado en la carpeta **fsx**.

```
pvc-trident.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: trident-csi
```

```
 You can issue the following commands to create a pvc and verify that it
has been created.
image:redhat-openshift-container-rosa-011.png["crear una prueba de PVC
usando Trident"]
```

Para utilizar iSCSI, debe haber habilitado iSCSI en los nodos de trabajo como se mostró anteriormente y debe crear un backend iSCSI y una clase de almacenamiento. A continuación se muestran algunos archivos yaml de muestra.

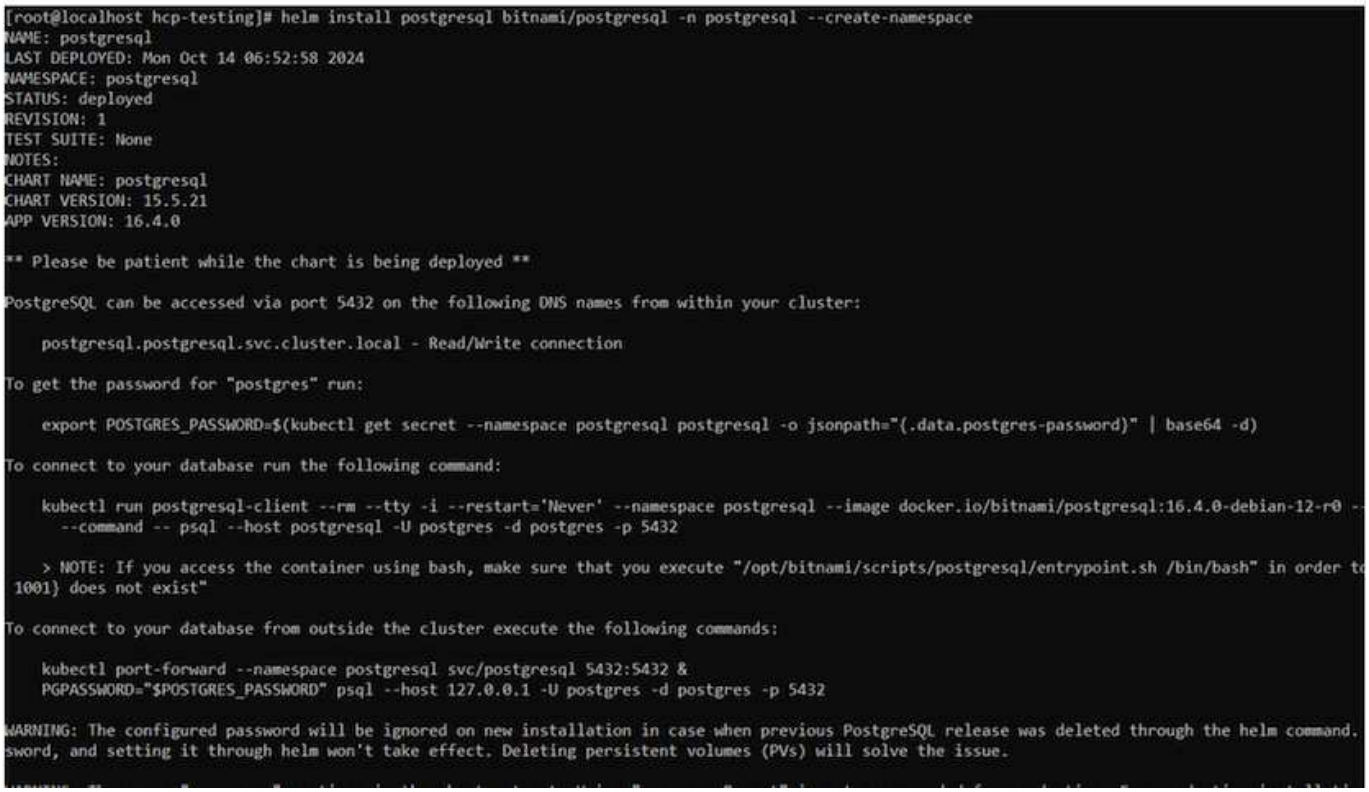
```
cat tbc.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: fsxadmin
  password: <password for the fsxN filesystem>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <management lif of fsxN filesystem>
  backendName: backend-tbc-ontap-san
  svm: svm_FSxNForROSAiSCSI
  credentials:
    name: backend-tbc-ontap-san-secret

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

5. Implementar una aplicación con estado de PostgreSQL de muestra

a. Utilice Helm para instalar PostgreSQL

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



b. Verifique que el pod de la aplicación se esté ejecutando y que se haya creado un PVC y un PV para la aplicación.

do. Implementar un cliente Postgresql

Utilice el siguiente comando para obtener la contraseña del servidor postgresql que se instaló.

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

Utilice el siguiente comando para ejecutar un cliente postgresql y conectarse al servidor usando la contraseña

```
$ 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
```

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitna
$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 securityC
 capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "
Root=true), seccompProfile (pod or container "postgresql-client" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Loca
If you don't see a command prompt, try pressing enter.
```

d. Crea una base de datos y una tabla. Cree un esquema para la tabla e inserte 2 filas de datos en la tabla.

```
postgres=# CREATE DATABASE erp;
CREATE DATABASE
postgres=# \c erp
psql (16.2, server 16.4)
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)
```

```
erp=# INSERT INTO PERSONS VALUES(2,'Jane','Scott');
INSERT 0 1
erp=# SELECT * from PERSONS;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
  2 | Jane      | Scott
(2 rows)
```

Servicio Red Hat OpenShift en AWS con NetApp ONTAP

Este documento describirá cómo utilizar NetApp ONTAP con Red Hat OpenShift Service en AWS (ROSA).

Crear instantánea de volumen

1. Crear una instantánea del volumen de la aplicación En esta sección, mostraremos cómo crear una instantánea de Trident del volumen asociado con la aplicación. Esta será una copia de un punto en el tiempo de los datos de la aplicación. Si se pierden los datos de la aplicación, podemos recuperarlos desde esta copia del momento. NOTA: Esta instantánea se almacena en el mismo agregado que el volumen original en ONTAP(en las instalaciones o en la nube). Entonces, si se pierde el agregado de almacenamiento de ONTAP , no podemos recuperar los datos de la aplicación desde su instantánea.

**a. Crear una VolumeSnapshotClass Guarde el siguiente manifiesto en un archivo llamado volume-snapshot-class.yaml

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

Cree una instantánea utilizando el manifiesto anterior.

```
[root@localhost hcp-testing]# oc create -f volume-snapshot-class.yaml
volumesnapshotclass.snapshot.storage.k8s.io/fsx-snapclass created
[root@localhost hcp-testing]#
```

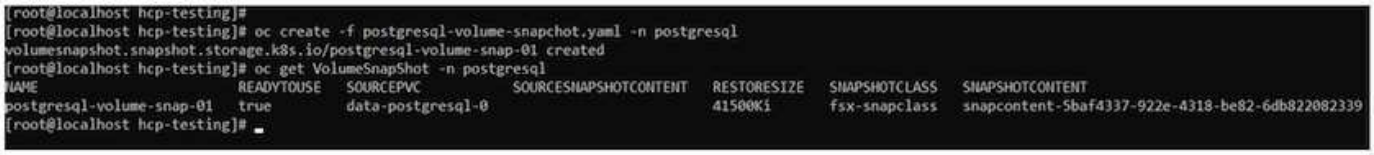
b. A continuación, cree una instantánea Cree una instantánea de la PVC existente creando VolumeSnapshot para tomar una copia de un punto en el tiempo de sus datos de Postgresql. Esto crea una instantánea de FSx que casi no ocupa espacio en el backend del sistema de archivos. Guarde el siguiente

manifiesto en un archivo llamado volume-snapshot.yaml:

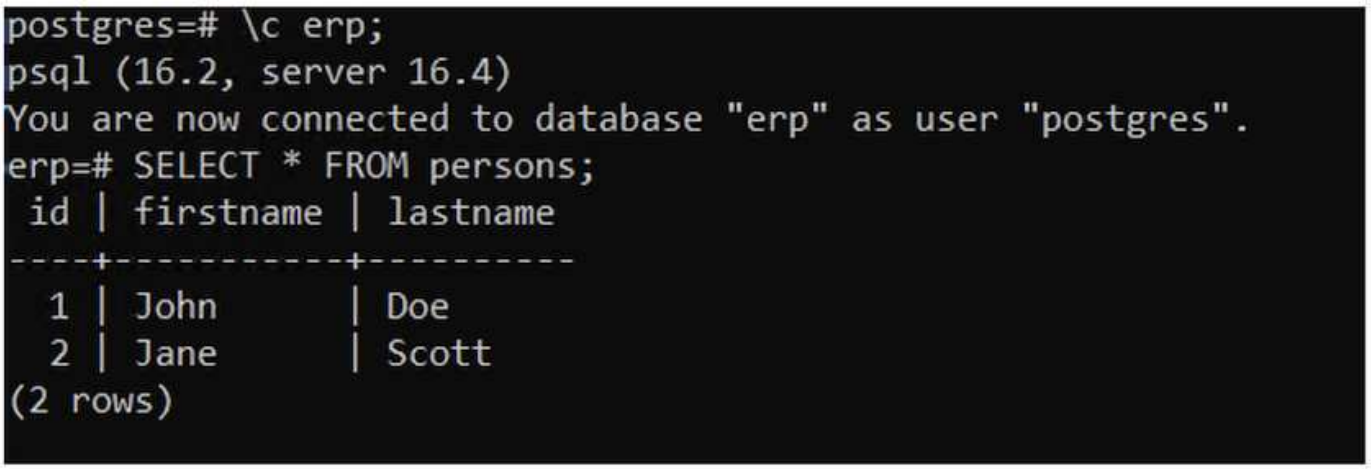
```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
 name: postgresql-volume-snap-01
spec:
 volumeSnapshotClassName: fsx-snapclass
 source:
   persistentVolumeClaimName: data-postgresql-0
```

do. Cree la instantánea del volumen y confirme que se ha creado

Eliminar la base de datos para simular la pérdida de datos (la pérdida de datos puede ocurrir debido a una variedad de razones, aquí solo la estamos simulando eliminando la base de datos)



d. Eliminar la base de datos para simular la pérdida de datos (la pérdida de datos puede ocurrir debido a una variedad de razones, aquí solo la estamos simulando eliminando la base de datos)



Restaurar desde una instantánea de volumen

1. Restaurar desde una instantánea En esta sección, mostraremos cómo restaurar una aplicación desde la instantánea trident del volumen de la aplicación.

a. Crear un clon de volumen a partir de la instantánea

Para restaurar el volumen a su estado anterior, debe crear un nuevo PVC basado en los datos de la instantánea que tomó. Para ello, guarde el siguiente manifiesto en un archivo llamado pvc-clone.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
 name: postgresql-volume-clone
spec:
 accessModes:
   - ReadWriteOnce
 storageClassName: trident-csi
 resources:
   requests:
     storage: 8Gi
 dataSource:
   name: postgresql-volume-snap-01
   kind: VolumeSnapshot
   apiGroup: snapshot.storage.k8s.io
```

Cree un clon del volumen creando una PVC usando la instantánea como fuente usando el manifiesto anterior. Aplique el manifiesto y asegúrese de que se cree el clon.

```
[root@localhost hcp-testing]# oc create -f postgresql-pvc-clone.yaml -n postgresql
persistentvolumeclaim/postgresql-volume-clone created
[root@localhost hcp-testing]# oc get pvc -n postgresql
NAME                     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-postgresql-0        Bound    pvc-e3ddd9bd-e6a7-4a4a-b935-f1c090fd8db6   8Gi        RWO            trident-csi
postgresql-volume-clone  Bound    pvc-b38fbc54-55dc-47e8-934d-47f181fddac6   8Gi        RWO            trident-csi
[root@localhost hcp-testing]#
```

b. Eliminar la instalación original de postgresql

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# helm uninstall postgresql -n postgresql
release "postgresql" uninstalled
[root@localhost hcp-testing]# oc get pods -n postgresql
No resources found in postgresql namespace.
[root@localhost hcp-testing]#
```

do. Cree una nueva aplicación postgresql usando el nuevo PVC clonado

```
$ helm install postgresql bitnami/postgresql --set
primary.persistence.enabled=true --set
primary.persistence.existingClaim=postgresql-volume-clone -n postgresql
```

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# helm install postgresql bitnami/postgresql --set primary.persistence.enabled=true \
> --set primary.persistence.existingClaim=postgresql-volume-clone -n postgresql
NAME: postgresql
LAST DEPLOYED: Mon Oct 14 12:03:31 2024
NAMESPACE: postgresql
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: postgresql
CHART VERSION: 15.5.21
APP VERSION: 16.4.0

** 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}" | bas

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.
      --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 /bi
 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
sword, and setting it through helm won't take effect. Deleting persistent volumes (PVs) will solve the issue.

WARNING: There are "resources" sections in the chart not set. Using "resourcesPreset" is not recommended for production. For prod
ng to your workload needs:
  - primary.resources
  - readReplicas.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[root@localhost hcp-testing]#
```

d. Verifique que el pod de aplicación esté en estado de ejecución

```
[root@localhost hcp-testing]# oc get pods -n postgresql
NAME           READY   STATUS    RESTARTS   AGE
postgresql-0   1/1     Running   0          2m1s
[root@localhost hcp-testing]#
```

mi. Verifique que el pod use el clon como su PVC

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc describe pod/postgresql-0 -n postgresql
```

```
  ContainersReady             True
  PodScheduled                True
Volumes:
  empty-dir:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
    SizeLimit:  <unset>
  dshm:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:     Memory
    SizeLimit:  <unset>
  data:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  postgresql-volume-clone
    ReadOnly:   false
QoS Class:      Burstable
Node-Selectors: <none>
Tolerations:    node.kubernetes.io/memory-pressure:NoSchedule op=Exists
                node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason                  Age    From                     Message
  ----    ------                  ----   ----                     -------
  Normal  Scheduled               3m55s  default-scheduler        Successfully assigned postgresql/postgres
.us-east-2.compute.internal
  Normal  SuccessfulAttachVolume  3m54s  attachdetach-controller  AttachVolume.Attach succeeded for volume
8-934d-47f181fddac6"
  Normal  AddedInterface          3m43s  multus                   Add eth0 [10.129.2.126/23] from ovn-kuber
  Normal  Pulled                  3m43s  kubelet                  Container image "docker.io/bitnami/postgr
r0" already present on machine
  Normal  Created                 3m42s  kubelet                  Created container postgresql
  Normal  Started                 3m42s  kubelet                  Started container postgresql
[root@localhost hcp-testing]#
```

f) Para validar que la base de datos se haya restaurado como se esperaba, regrese a la consola del contenedor y muestre las bases de datos existentes.

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:
$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.allowPri
 capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "postgresql-clie
Root=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=# \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;
psql (16.2, server 16.4)
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
  2 | Jane      | Scott
(2 rows)
```

Vídeo de demostración

Amazon FSx for NetApp ONTAP con Red Hat OpenShift Service en AWS mediante plano de control alojado

Puede encontrar más vídeos sobre Red Hat OpenShift y las soluciones OpenShift"aquí" .