FSxN을 사용한 AWS의 Red Hat OpenShift 서비스

NetApp container solutions

NetApp
January 21, 2026

목차

FSxN을 사용한 AWS의 Red Hat OpenShift 서비스

NetApp ONTAP 사용한 AWS의 Red Hat OpenShift 서비스

개요

이 섹션에서는 ROSA에서 실행되는 애플리케이션을 위한 영구 저장소 계층으로 FSx for ONTAP 을 활용하는 방법을 보여드리겠습니다. ROSA 클러스터에 NetApp Trident CSI 드라이버를 설치하는 방법, FSx for ONTAP 파일 시스템을 프로비저닝하는 방법, 샘플 상태 저장 애플리케이션을 배포하는 방법을 보여줍니다. 또한 애플리케이션 데이터를 백업하고 복원하는 전략도 알려드립니다. 이 통합 솔루션을 사용하면 여러 AZ에 걸쳐 손쉽게 확장 가능한 공유 스토리지 프레임워크를 구축하고 Trident CSI 드라이버를 사용하여 데이터의 확장, 보호 및 복원 프로세스를 간소화할 수 있습니다.

필수 조건

- "AWS 계정"
- "Red Hat 계정"
- IAM 사용자"적절한 권한이 있는 경우" ROSA 클러스터를 생성하고 액세스하려면
- "AWS CLI"
- "로사 CLI"
- "OpenShift 명령줄 인터페이스"(오케이)
- 헬멧 3"선적 서류 비치"
- "HCP ROSA 클러스터"
- "Red Hat OpenShift 웹 콘솔에 액세스"

이 다이어그램은 여러 AZ에 배포된 ROSA 클러스터를 보여줍니다. ROSA 클러스터의 마스터 노드, 인프라 노드는 Red Hat의 VPC에 있고, 워커 노드는 고객 계정의 VPC에 있습니다. 동일한 VPC 내에 ONTAP 파일 시스템용 FSx를 만들고 ROSA 클러스터에 Trident 드라이버를 설치하여 이 VPC의 모든 서브넷이 파일 시스템에 연결할 수 있도록 합니다.

초기 설정

1. NetApp ONTAP 에 대한 **FSx** 제공

ROSA 클러스터와 동일한 VPC에 NetApp ONTAP 용 다중 AZ FSx를 만듭니다. 이를 수행하는 방법은 여러 가지가 있습니다. CloudFormation Stack을 사용하여 FSxN을 만드는 방법에 대한 세부 정보가 제공됩니다.

a.GitHub 저장소 복제

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

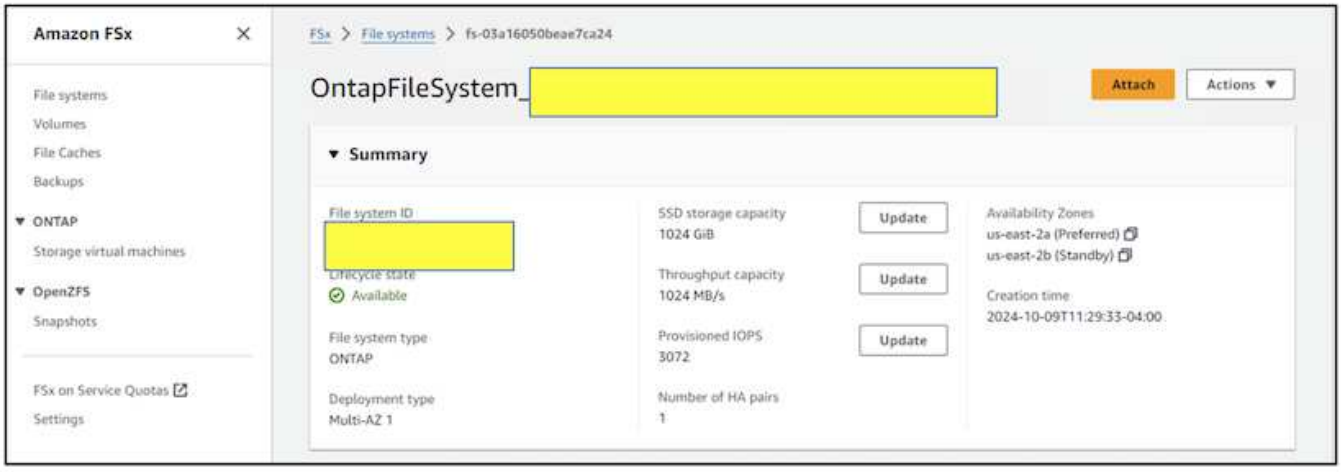
b. CloudFormation 스택 실행 매개변수 값을 사용자 정의 값으로 바꿔서 아래 명령을 실행합니다.

```
$ 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
```

여기서: region-name: ROSA 클러스터가 배포된 지역과 동일 subnet1_ID: FSxN에 대한 기본 서브넷의 ID subnet2_ID: FSxN에 대한 대기 서브넷의 ID VPC_ID: ROSA 클러스터가 배포된 VPC의 ID routetable1_ID, routetable2_ID: 위에서 선택한 서브넷과 연관된 경로 테이블의 ID your_allowed_CIDR: FSx for ONTAP 보안 그룹에 허용되는 CIDR 범위 액세스를 제어하기 위한 수신 규칙. 0.0.0.0/0 또는 적절한 CIDR을 사용하여 모든 트래픽이 FSx for ONTAP 의 특정 포트에 액세스하도록 허용할 수 있습니다. 관리자 비밀번호 정의: FSxN에 로그인하기 위한 비밀번호입니다. SVM 비밀번호 정의: 생성될 SVM에 로그인하기 위한 비밀번호입니다.

아래에 표시된 Amazon FSx 콘솔을 사용하여 파일 시스템과 스토리지 가상 머신(SVM)이 생성되었는지 확인하세요.



2. ROSA 클러스터용 **Trident CSI** 드라이버 설치 및 구성

b. Trident

ROSA 클러스터 워커 노드에는 스토리지 프로비저닝 및 액세스를 위해 NAS 프로토콜을 사용할 수 있는 NFS 도구가 미리 구성되어 있습니다.

대신 iSCSI를 사용하려면 iSCSI에 대한 작업자 노드를 준비해야 합니다. Trident 25.02 릴리스부터 ROSA 클러스터

(또는 모든 OpenShift 클러스터)의 작업자 노드를 쉽게 준비하여 FSxN 스토리지에서 iSCSI 작업을 수행할 수 있습니다. iSCSI를 위한 워커 노드 준비를 자동화하는 Trident 25.02(또는 이후 버전)를 설치하는 쉬운 방법은 2가지가 있습니다. 1. tridentctl 도구를 사용하여 명령줄에서 node-prep-flag를 사용합니다. 2. 운영자 허브에서 Red Hat 인증 Trident 운영자를 사용하고 사용자 정의합니다. 3. Helm 사용하기.

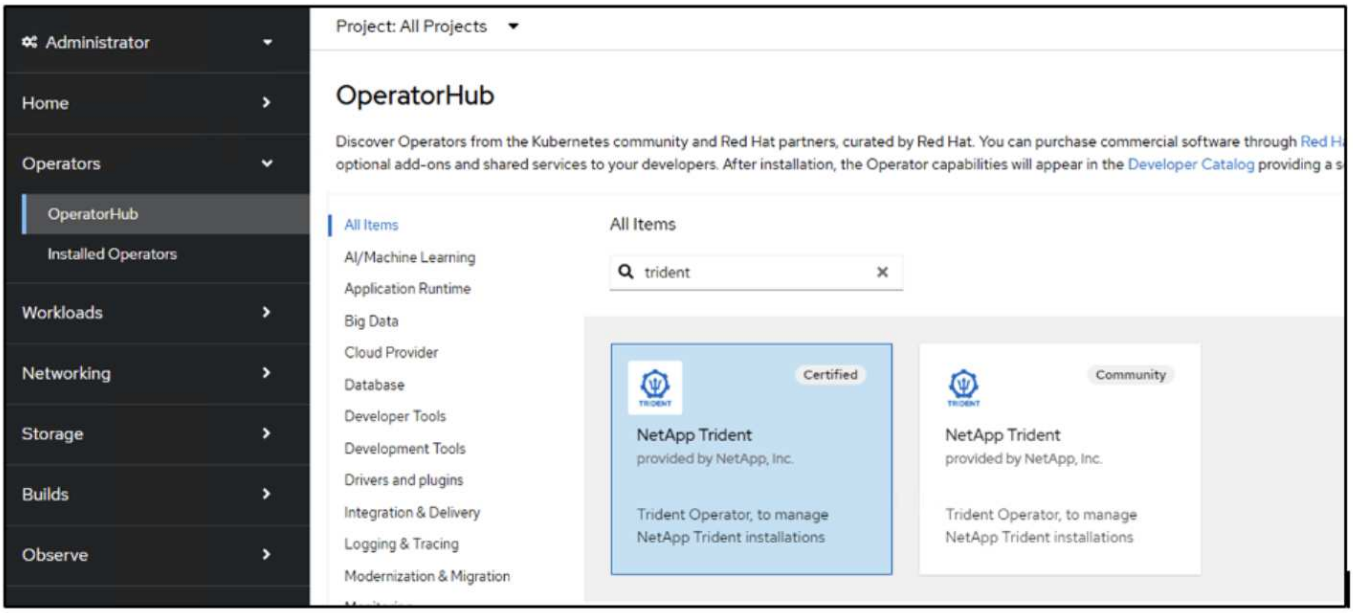
위의 방법을 node-prep을 활성화하지 않고 사용하면 FSxN에서 스토리지 프로비저닝에 NAS 프로토콜만 사용할 수 있습니다.

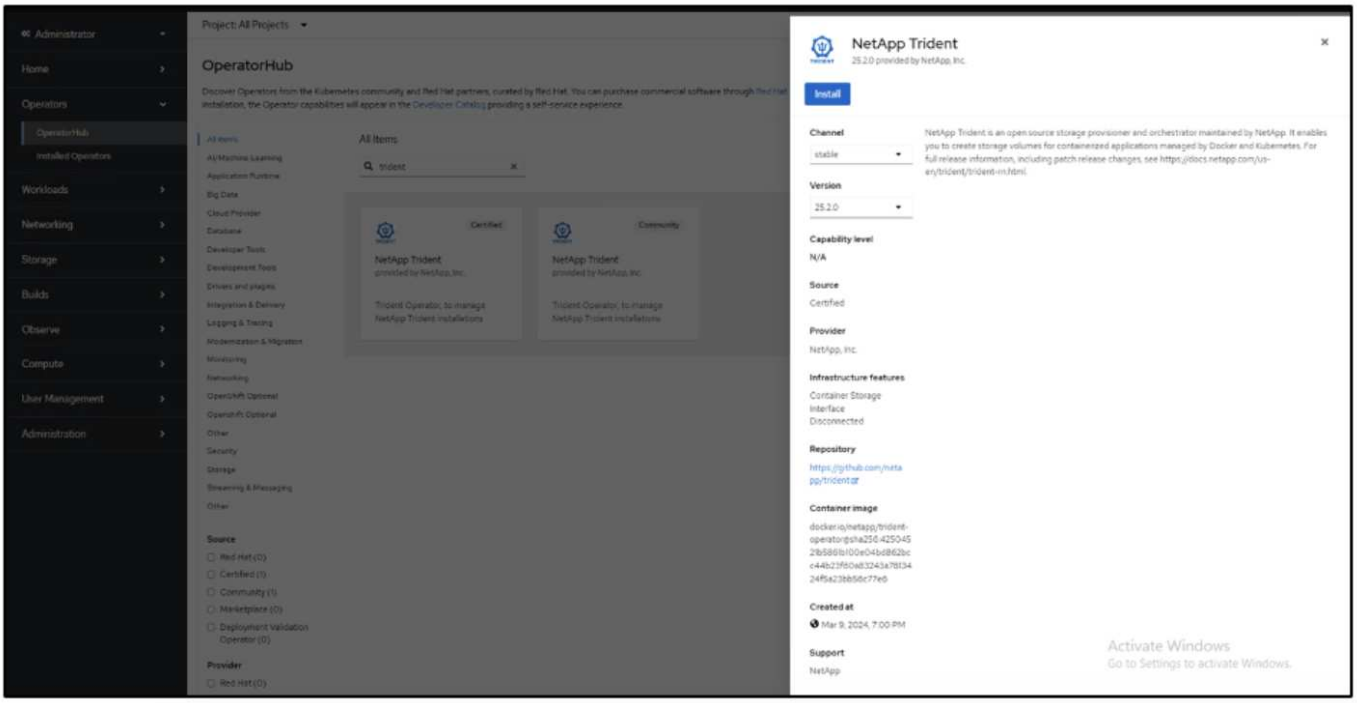
방법 **1: tridentctl** 도구 사용

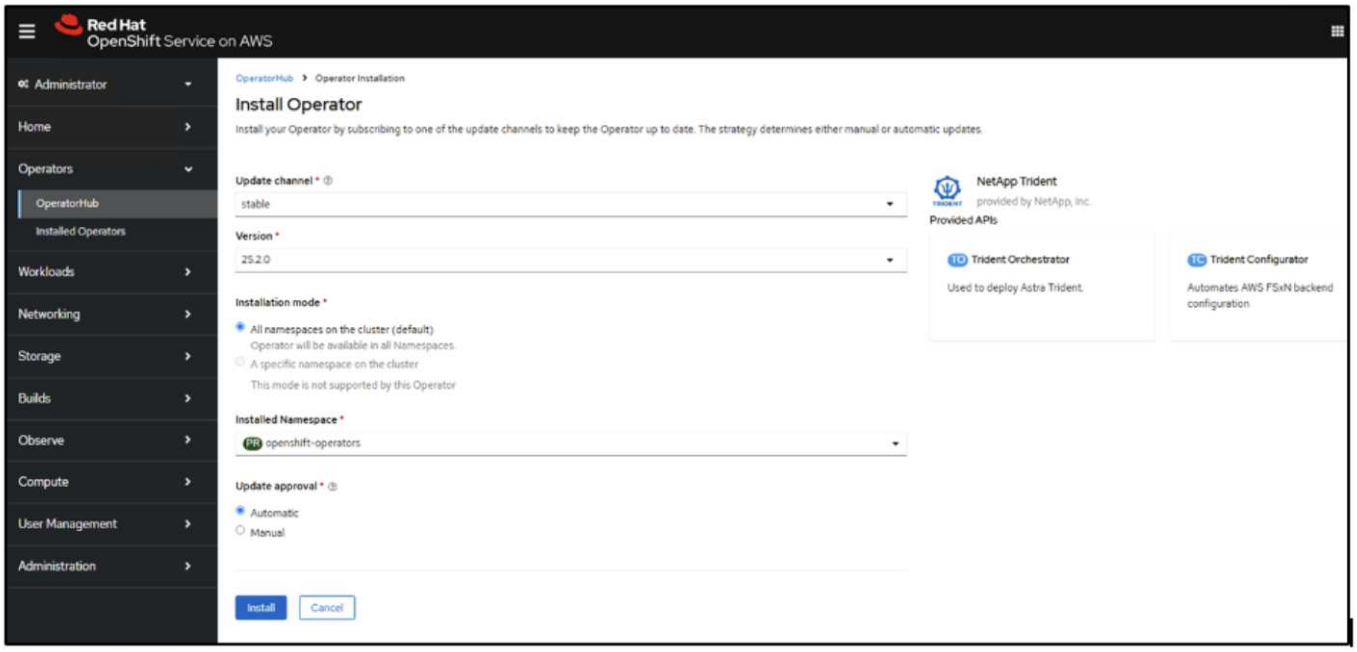
node-prep 플래그를 사용하여 표시된 대로 Trident 설치합니다. 설치 명령을 실행하기 전에 설치 프로그램 패키지를 다운로드해야 합니다. "여기 문서" .

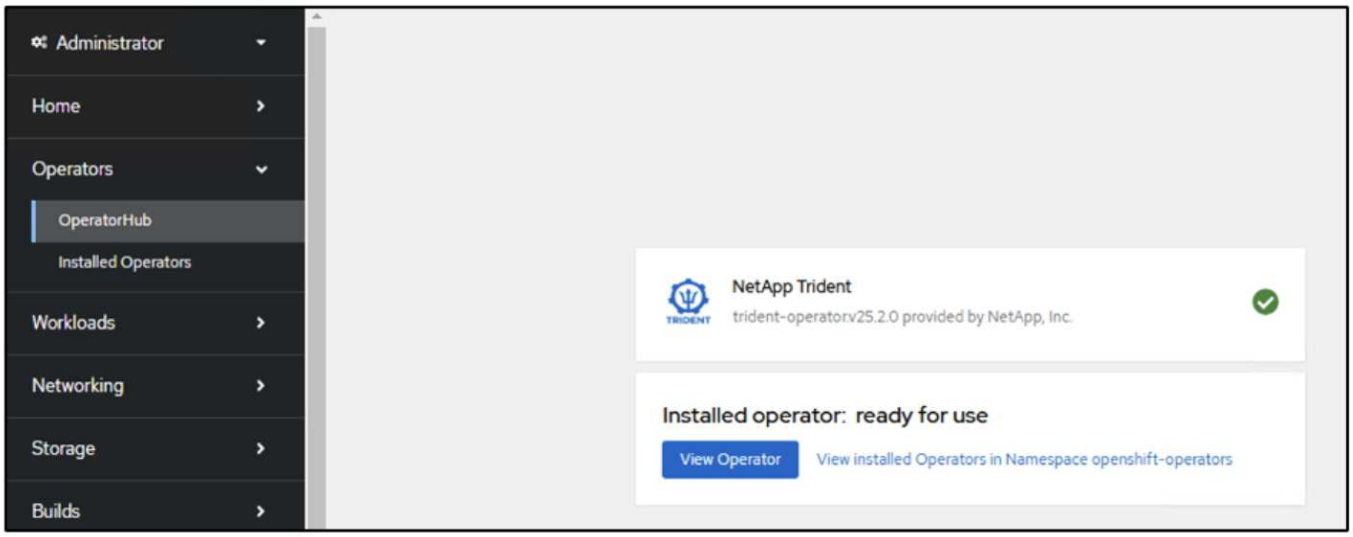
```
#./tridentctl install trident -n trident --node-prep=iscsi
```

방법 **2: Red Hat** 인증 **Trident Operator**를 사용하여 사용자 지정 OperatorHub에서 Red Hat 인증 Trident Operator를 찾아 설치합니다.

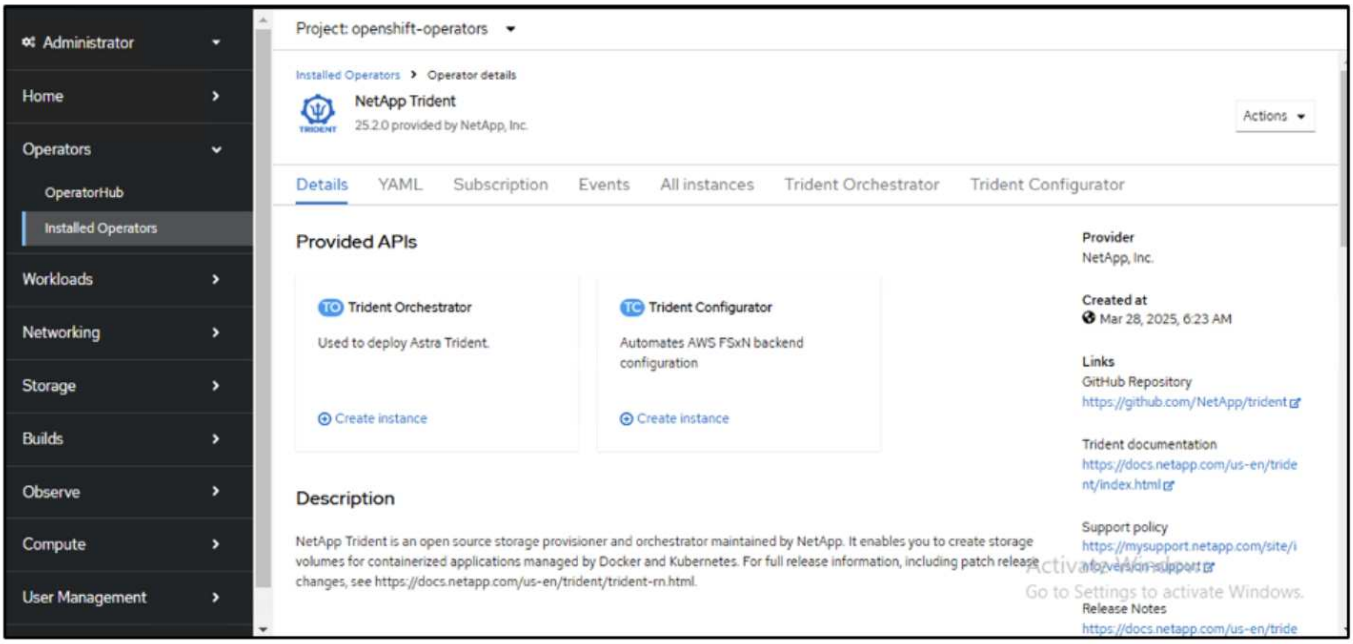


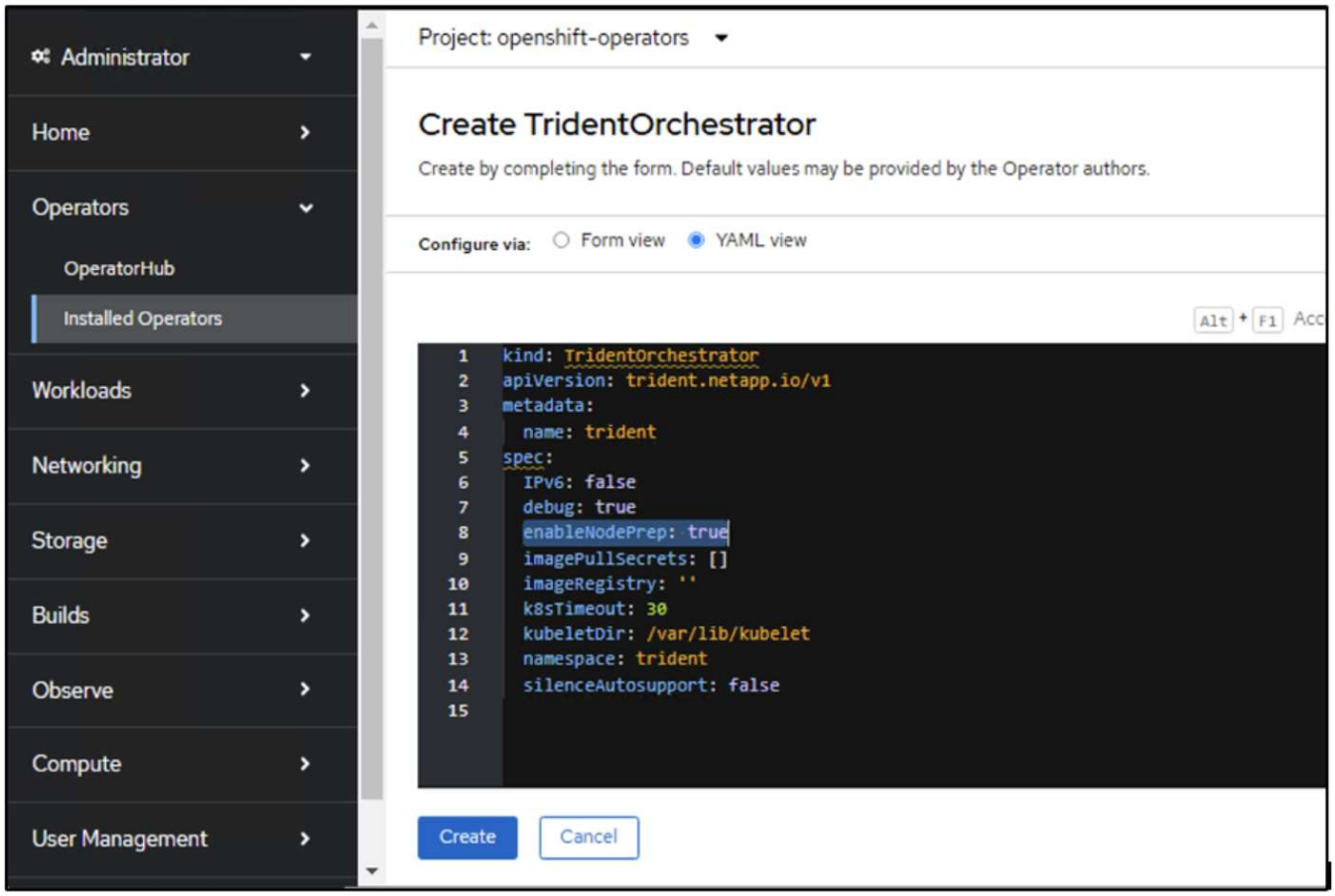

Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: All Projects
OperatorHub
All Items
trident
NetApp Trident
provided by NetApp, Inc.
Trident Operator, to manage NetApp Trident installations
Certified
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.

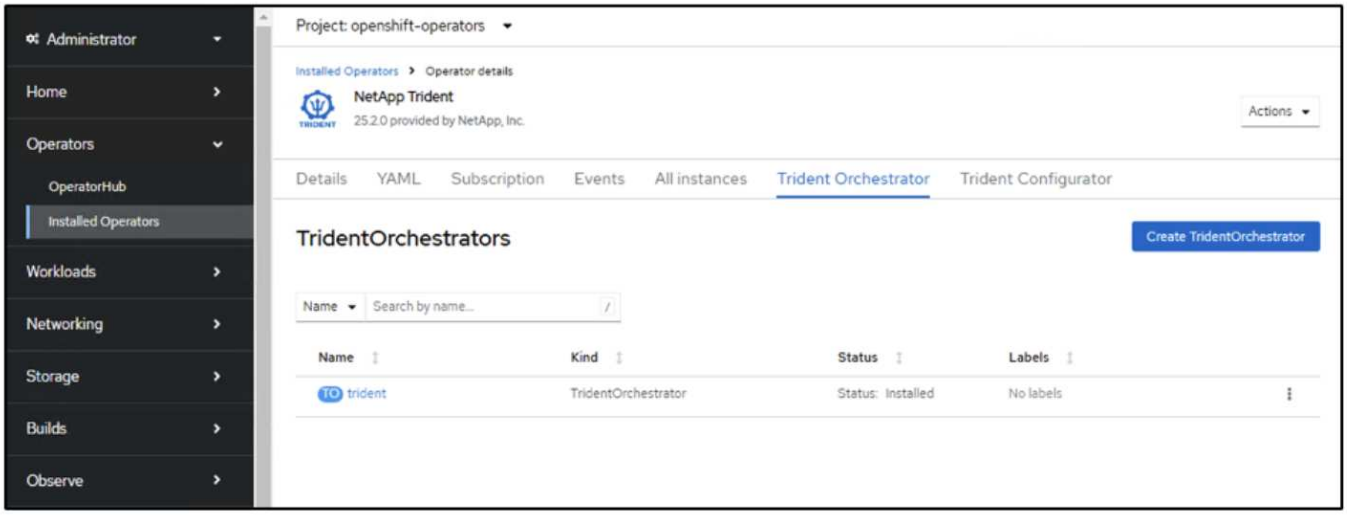

Red Hat
OpenShift Service on AWS
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OperatorHub > Operator Installation
Install Operator
Install your Operator by subscribing to one of the update channels to keep 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



다음으로, Trident Orchestrator 인스턴스를 생성합니다. YAML 뷰를 사용하여 사용자 정의 값을 설정하거나 설치 중에 iSCSI 노드 준비를 활성화합니다.








```
[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]#
```

위의 방법 중 하나를 사용하여 Trident 설치하면 iscsid 및 multipathd 서비스를 시작하고 /etc/multipath.conf 파일에 다음을 설정하여 ROSA 클러스터 워커 노드가 iSCSI를 위해 준비됩니다.

```
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. 모든 **Trident** 포드가 실행 상태인지 확인하세요**.**

```
[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]#
```

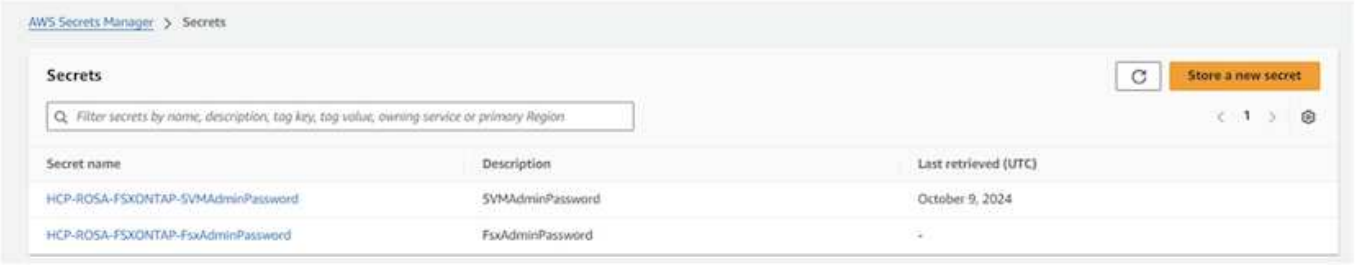
3. ONTAP (ONTAP NAS)에 대한 FSx를 사용하도록 Trident CSI 백엔드를 구성합니다.

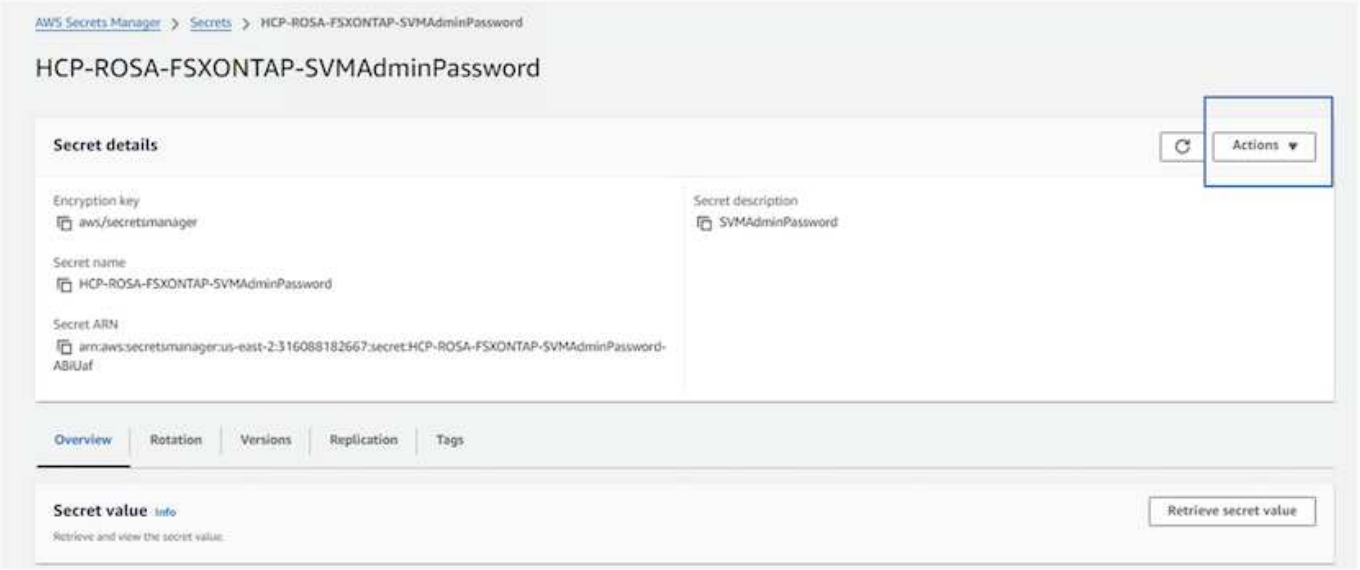
Trident 백엔드 구성은 Trident 스토리지 시스템(이 경우 FSx for ONTAP)과 통신하는 방법을 알려줍니다. 백엔드를 생성하려면 클러스터 관리 및 NFS 데이터 인터페이스와 함께 연결할 스토리지 가상 머신의 자격 증명을 제공합니다. 우리는 사용할 것입니다"ontap-nas 드라이버" FSx 파일 시스템에서 스토리지 볼륨을 프로비저닝합니다.

에이**.** 먼저 다음 **yaml**을 사용하여 SVM 자격 증명에 대한 비밀을 만듭니다.

```
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>
```

아래에 표시된 것처럼 AWS Secrets Manager에서 FSxN에 대해 생성된 SVM 비밀번호를 검색할 수도 있습니다.





b. 다음으로, 다음 명령을 사용하여 **ROSA** 클러스터에 **SVM** 자격 증명에 대한 비밀을 추가합니다.

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

다음 명령을 사용하여 비밀이 트라이던트 네임스페이스에 추가되었는지 확인할 수 있습니다.

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

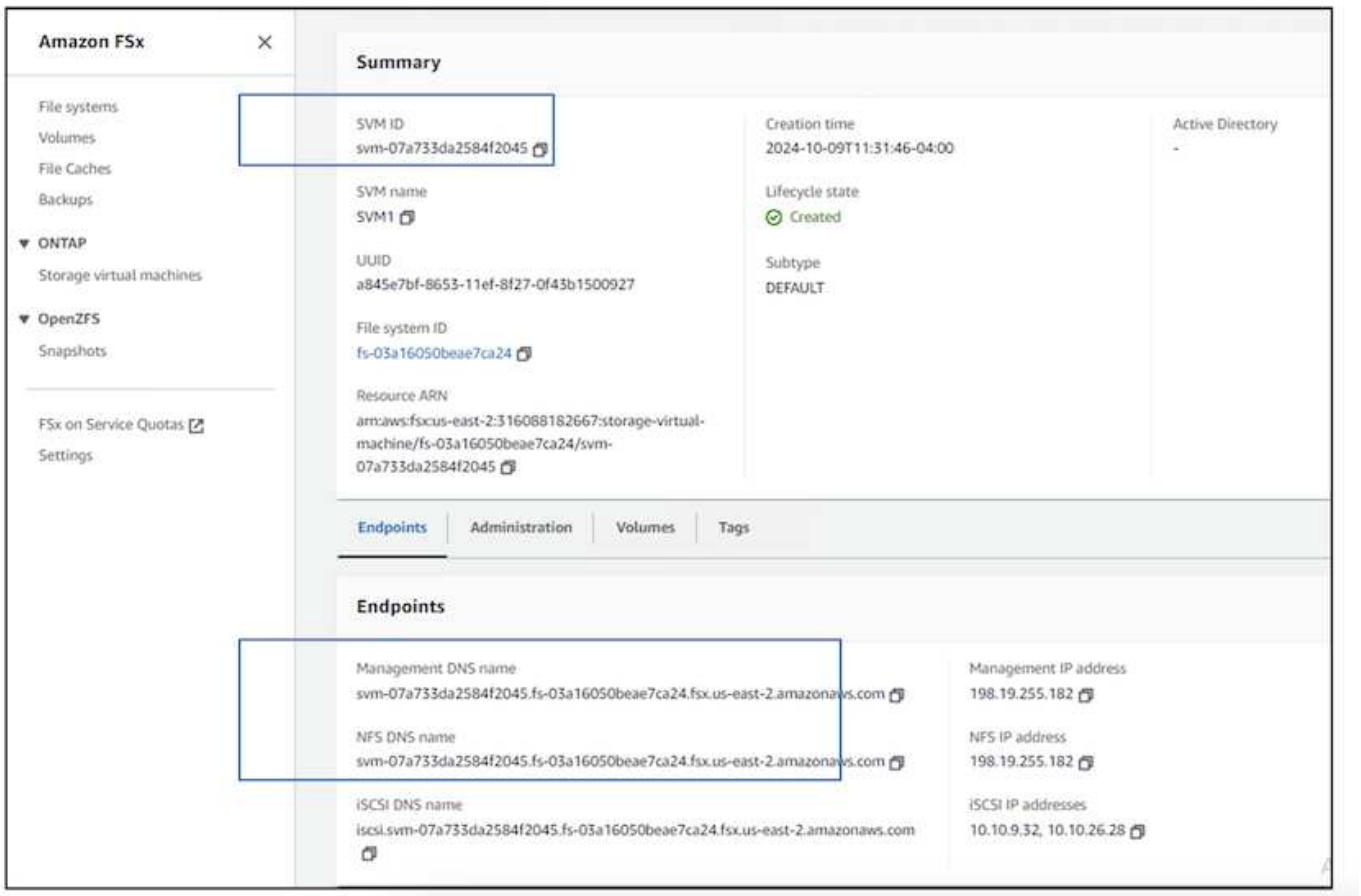
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

기음. 다음으로 백엔드 객체를 만듭니다. 이를 위해 복제된 **Git** 저장소의 fsx 디렉토리로 이동합니다. **backend-ontap-nas.yaml** 파일을 엽니다. 다음을 바꾸세요. managementLIF를 관리 **DNS** 이름으로, dataLIF를 **Amazon FSx SVM**의 **NFS DNS** 이름으로, svm**을 SVM 이름으로 바꾸세요. 다음 명령을 사용하여 백엔드 객체를 생성합니다.

다음 명령을 사용하여 백엔드 객체를 생성합니다.

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

아래 스크린샷에 표시된 것처럼 Amazon FSx 콘솔에서 관리 DNS 이름, NFS DNS 이름 및 SVM 이름을 얻을 수 있습니다.



**디. 이제 다음 명령을 실행하여 백엔드 개체가 생성되었고 Phase가 Bound를 표시하고 Status가 Success인지 확인하세요.

```
[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. 스토리지 클래스 생성 이제 Trident 백엔드가 구성되었으므로 백엔드를 사용할 Kubernetes 스토리지 클래스를 생성할 수 있습니다. 스토리지 클래스는 클러스터에서 사용할 수 있는 리소스 객체입니다. 이는 애플리케이션에 대해 요청할 수 있는 저장 유형을 설명하고 분류합니다.

에이**. fsx** 폴더에서 **storage-class-csi-nas.yaml** 파일을 검토합니다**.**

```
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
```

비**. ROSA** 클러스터에 스토리지 클래스를 생성하고 **trident-csi** 스토리지 클래스가 생성되었는지 확인합니다**.**

```
[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]#
```

이것으로 Trident CSI 드라이버 설치와 FSx for ONTAP 파일 시스템에 대한 연결이 완료되었습니다. 이제 FSx for ONTAP 의 파일 볼륨을 사용하여 ROSA에 샘플 Postgresql 상태 저장 애플리케이션을 배포할 수 있습니다.

기음**. trident-csi** 스토리지 클래스를 사용하여 생성된 **PVC** 및 **PV**가 없는지 확인합니다**.**

디. **Trident CSI**를 사용하여 애플리케이션이 **PV**를 생성할 수 있는지 확인합니다.

fsx 폴더에 제공된 pvc-trident.yaml 파일을 사용하여 PVC를 만듭니다.

```
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["Trident 사용하여 테스트 PVC
만들기"]
```

iSCSI를 사용하려면 이전에 보여준 것처럼 작업자 노드에서 iSCSI를 활성화해야 하며 iSCSI 백엔드와 스토리지 클래스를 생성해야 합니다. 다음은 몇 가지 YAML 파일 샘플입니다.

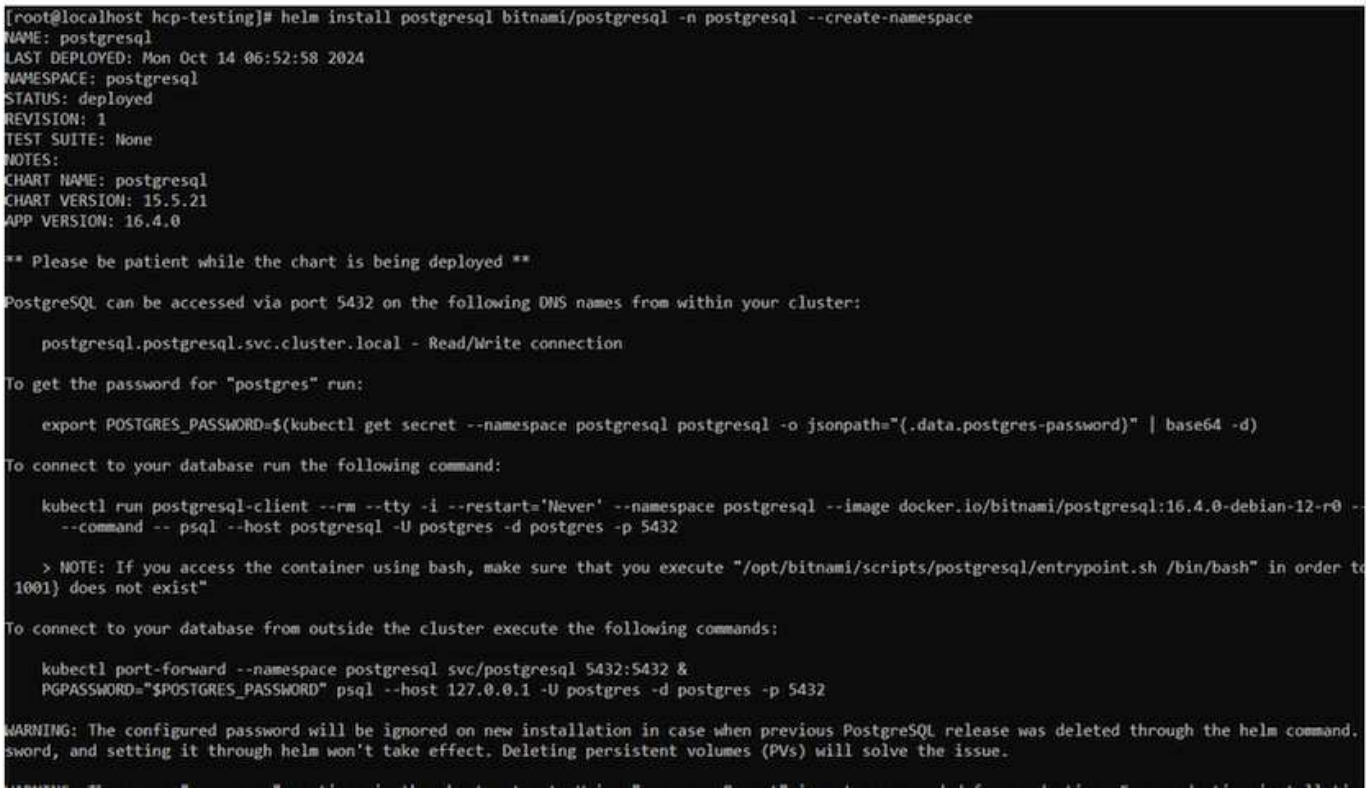
```
cat tbc.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: fsxadmin
  password: <password for the fsxN filesystem>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 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. 샘플 **Postgresql** 상태 저장 애플리케이션 배포

**에이. helm을 사용하여 postgresql을 설치하세요.

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



비. 애플리케이션 포드가 실행 중인지, 그리고 애플리케이션에 대한 **PVC**와 **PV**가 생성되었는지 확인하세요.

기음. **Postgresql** 클라이언트 배포

설치된 **postgresql** 서버의 비밀번호를 얻으려면 다음 명령을 사용하세요.

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

다음 명령을 사용하여 **postgresql** 클라이언트를 실행하고 비밀번호를 사용하여 서버에 연결합니다.

```
$ 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.
```

디. 데이터베이스와 테이블을 만듭니다. 테이블에 대한 스키마를 만들고 테이블에 **2**행의 데이터를 삽입합니다.

```
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)
```

NetApp ONTAP 사용한 AWS의 Red Hat OpenShift 서비스

이 문서에서는 AWS의 Red Hat OpenShift 서비스(ROSA)와 함께 NetApp ONTAP 사용하는 방법을 설명합니다.

볼륨 스냅샷 생성

1. 앱 볼륨의 스냅샷 만들기 이 섹션에서는 앱과 연관된 볼륨의 트라이던트 스냅샷을 만드는 방법을 보여줍니다. 이는 앱 데이터의 특정 시점 복사본입니다. 애플리케이션 데이터가 손실된 경우 해당 시점의 복사본으로부터 데이터를 복구할 수 있습니다. 참고: 이 스냅샷은 ONTAP 의 원본 볼륨과 동일한 집계에 저장됩니다(온프레미스 또는 클라우드). 따라서 ONTAP 스토리지 집계가 손실되면 스냅샷에서 앱 데이터를 복구할 수 없습니다.

**에이. VolumeSnapshotClass를 만듭니다. 다음 매니페스트를 volume-snapshot-class.yaml이라는 파일에 저장합니다.

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

위의 매니페스트를 사용하여 스냅샷을 만듭니다.

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

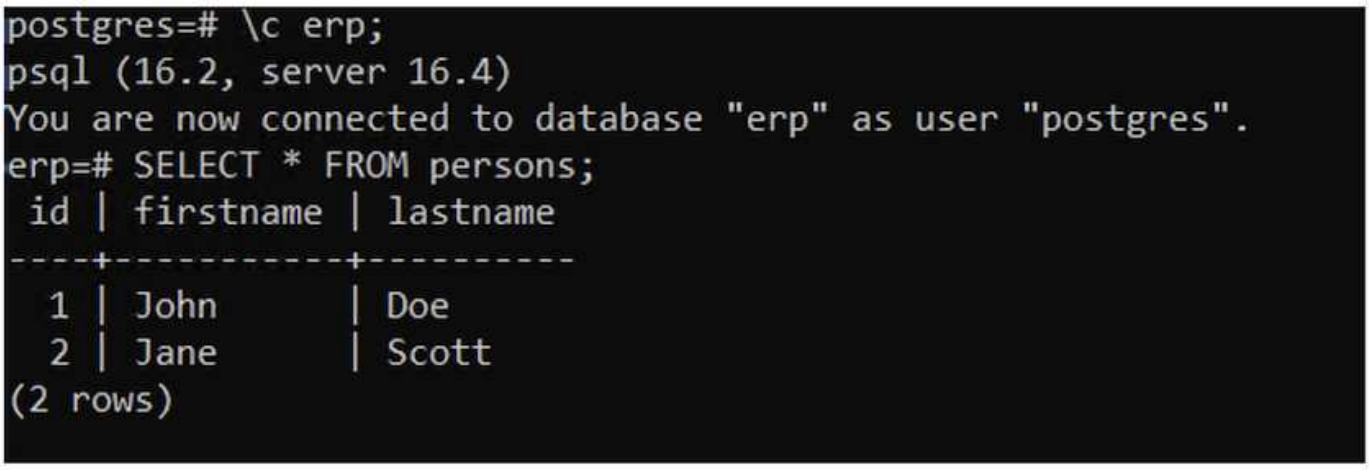
**비. 다음으로 스냅샷을 만듭니다. VolumeSnapshot을 만들어 Postgresql 데이터의 특정 시점 복사본을 만들어 기존 PVC의 스냅샷을 만듭니다. 이렇게 하면 파일 시스템 백엔드에서 공간을 거의 차지하지 않는 FSx 스냅샷이 생성됩니다. 다음 매니페스트를 volume-snapshot.yaml이라는 파일에 저장합니다.

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

기음. 볼륨 스냅샷을 생성하고 생성되었는지 확인합니다.

데이터 손실을 시뮬레이션하기 위해 데이터베이스를 삭제합니다(데이터 손실은 다양한 이유로 발생할 수 있지만 여기서는 데이터베이스를 삭제하여 시뮬레이션합니다)

디. 데이터 손실을 시뮬레이션하기 위해 데이터베이스를 삭제합니다**(**데이터 손실은 다양한 이유로 발생할 수 있지만 여기서는 데이터베이스를 삭제하여 시뮬레이션합니다**)**



볼륨 스냅샷에서 복원

1. 스냅샷에서 복원 이 섹션에서는 앱 볼륨의 트라이던트 스냅샷에서 애플리케이션을 복원하는 방법을 보여드립니다.

에이. 스냅샷에서 볼륨 복제본을 만듭니다.

볼륨을 이전 상태로 복원하려면 스냅샷의 데이터를 기반으로 새 PVC를 만들어야 합니다. 이렇게 하려면 다음 매니페스트를 pvc-clone.yaml이라는 파일에 저장하세요.

```
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
```

위의 매니페스트를 사용하여 스냅샷을 소스로 사용하여 PVC를 생성하여 볼륨의 복제본을 만듭니다. 매니페스트를 적용하고 복제본이 생성되었는지 확인하세요.

```
[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]#
```

**비. 원래 postgresql 설치를 삭제합니다.

```
[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]#
```

**기음. 새로운 복제 PVC를 사용하여 새 postgresql 애플리케이션을 만듭니다.

```
$ 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}" | ba

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 /b
 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 throug
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 pro
ng to your workload needs:
  - primary.resources
  - readReplicas.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[root@localhost hcp-testing]#
```

**디. 애플리케이션 포드가 실행 상태인지 확인하세요.

```
[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]#
```

**이자형. 포드가 복제본을 PVC로 사용하는지 확인하세요.

```
[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) 데이터베이스가 예상대로 복구되었는지 확인하려면 컨테이너 콘솔로 돌아가서 기존 데이터베이스를 표시합니다.

```
[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)
```

데모 비디오

AWS에서 호스팅 제어 평면을 사용하여 Red Hat OpenShift 서비스를 갖춘 Amazon FSx for NetApp ONTAP

Red Hat OpenShift 및 OpenShift 솔루션에 대한 추가 비디오를 찾을 수 있습니다."여기" .