



FSxN이 포함된 AWS 기반 Red Hat OpenShift Service

NetApp Solutions

NetApp
January 09, 2025

목차

FSxN이 포함된 AWS 기반 Red Hat OpenShift Service	1
NetApp ONTAP가 포함된 AWS 기반 Red Hat OpenShift Service	1
NetApp ONTAP가 포함된 AWS 기반 Red Hat OpenShift Service	11

FSxN이 포함된 AWS 기반 Red Hat OpenShift Service

NetApp ONTAP가 포함된 AWS 기반 Red Hat OpenShift Service

개요

이 섹션에서는 Rosa에서 실행되는 애플리케이션의 영구 스토리지 계층으로 FSx for ONTAP을 활용하는 방법을 알아봅니다. Rosa 클러스터에 NetApp Trident CSI 드라이버 설치, FSx for ONTAP 파일 시스템 프로비저닝 및 샘플 상태 저장 애플리케이션 배포를 보여 줍니다. 또한 응용 프로그램 데이터를 백업하고 복원하기 위한 전략도 보여 줍니다. 이 통합 솔루션을 사용하면 AZ 간에 손쉽게 확장되는 공유 스토리지 프레임워크를 구축하여 Trident CSI 드라이버를 사용하여 데이터 크기 조정, 보호 및 복원 프로세스를 간소화할 수 있습니다.

필수 구성 요소

- ["설치하고 있습니다"](#)
- ["Red Hat 계정"](#)
- IAM 사용자가 ["적절한 권한이 있어야 합니다"](#) ROSA 클러스터를 생성하고 액세스합니다
- ["AWS CLI를 참조하십시오"](#)
- ["로사 CLI"](#)
- ["OpenShift 명령줄 인터페이스" \(OC\)](#)
- [도움 3 "문서화"](#)
- ["HCP ROSA 클러스터"](#)
- ["Red Hat OpenShift 웹 콘솔에 액세스합니다"](#)

이 다이어그램은 여러 AZ에 배포된 ROSA 클러스터를 보여 줍니다. Rosa 클러스터의 마스터 노드, 인프라 노드는 Red Hat의 VPC에 있고, 작업자 노드는 고객 어카운트의 VPC에 있습니다. 동일한 VPC 내에 FSx for ONTAP 파일 시스템을 생성하고 ROSA 클러스터에 Trident 드라이버를 설치하여 이 VPC의 모든 서브넷이 파일 시스템에 연결할 수 있게 합니다.



초기 설정

- 1. NetApp ONTAP용 FSx 프로비저닝**

ROSA 클러스터와 동일한 VPC에서 다중 AZ FSx for NetApp ONTAP를 생성합니다. 이 작업은 여러 가지 방법으로 수행할 수 있습니다. CloudFormation Stack을 사용하여 FSxN을 생성하는 방법에 대한 세부 정보를 제공합니다

- a. GitHub 리포지토리를 복제합니다**

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

- b. CloudFormation Stack을 실행합니다. **매개 변수 값을 사용자 고유의 값으로 바꾸어 아래 명령을 실행합니다.

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

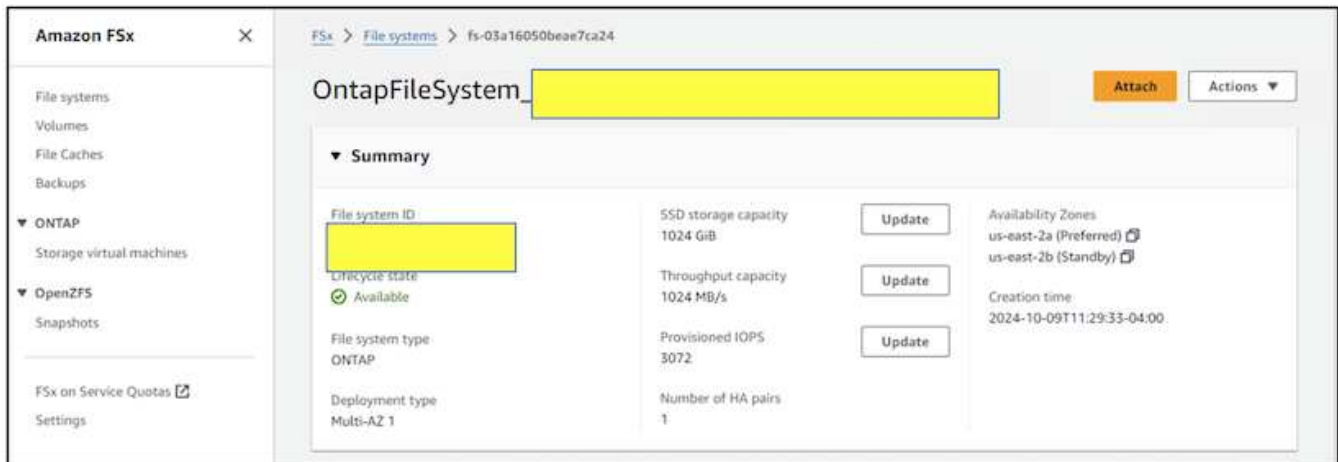
```

$ 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

```

장소: region-name: ROSA 클러스터가 배포된 지역과 동일 subnet1_ID: FSxN subnet2_ID: FSxN VPC_ID용 대기 서버넷의 id: ROSA 클러스터가 배포된 VPC의 id, routable1_ID: CIDR ONTAP에 대한 라우팅 테이블의 ID: CIDR에 대한 액세스 허용 범위 설정 0.0.0.0/0 또는 적절한 CIDR을 사용하여 모든 트래픽이 FSx for ONTAP의 특정 포트에 액세스하도록 허용할 수 있습니다. 관리자 암호 정의: FSxN에 로그인하기 위한 암호 SVM 암호 정의: 생성할 SVM에 로그인하기 위한 암호 정의

아래와 같이 Amazon FSx 콘솔을 사용하여 파일 시스템 및 SVM(스토리지 가상 머신)이 생성되었는지 확인합니다.



- 2. Rosa 클러스터용 Trident CSI 드라이버를 설치 및 구성합니다**
- a. Trident Helm 리포지토리를 추가합니다**

```

$ helm repo add netapp-trident https://netapp.github.io/trident-helm-chart

```

- b. Helm**을 사용하여 Trident를 설치합니다

```
$ helm install trident netapp-trident/trident-operator --version 100.2406.0 --create-namespace --namespace trident
```



설치하는 버전에 따라 표시된 명령에서 version 매개 변수를 변경해야 합니다. 올바른 버전 번호는 ["문서화"](#)참조하십시오. Trident를 설치하는 추가 방법은 Trident를 ["문서화"](#)참조하십시오.

C. 모든 Trident Pod가 실행 중인지 확인합니다

```
[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. Trident CSI 백엔드에서 FSx for ONTAP(ONTAP NAS)** 를 사용하도록 구성합니다

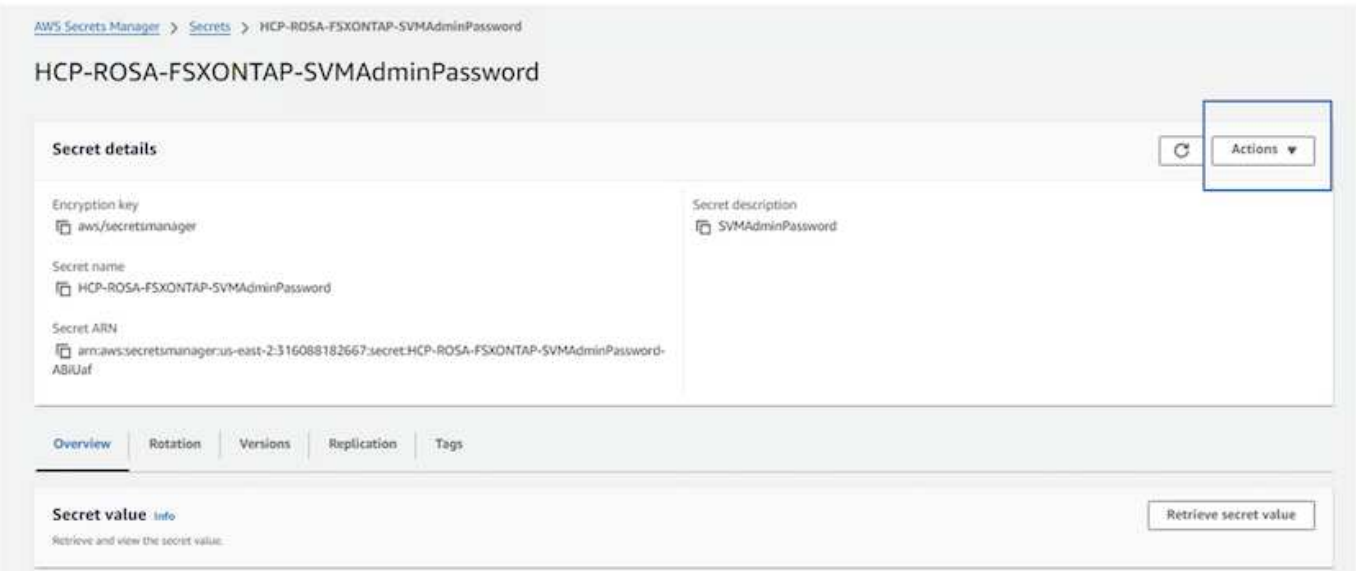
Trident 백 엔드 구성은 Trident에 스토리지 시스템과 통신하는 방법을 알려줍니다(이 경우에는 FSx for ONTAP). 백엔드를 생성하기 위해 클러스터 관리 및 NFS 데이터 인터페이스와 함께 연결할 스토리지 가상 시스템의 자격 증명을 제공합니다. 을 사용하여 ["ONTAP - NAS 드라이버"](#)FSx 파일 시스템에서 스토리지 볼륨을 프로비저닝합니다.

- a. 먼저 다음 YAML** 을 사용하여 SVM 자격 증명에 대한 암호를 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>
```



아래와 같이 AWS Secrets Manager에서 FSxN용으로 생성된 SVM 암호를 검색할 수도 있습니다.



- b. 다음 명령을 사용하여 ROSA 클러스터에 SVM 자격 증명에 대한 암호를 추가합니다**

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

다음 명령을 사용하여 Trident 네임스페이스에 암호가 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]#
```

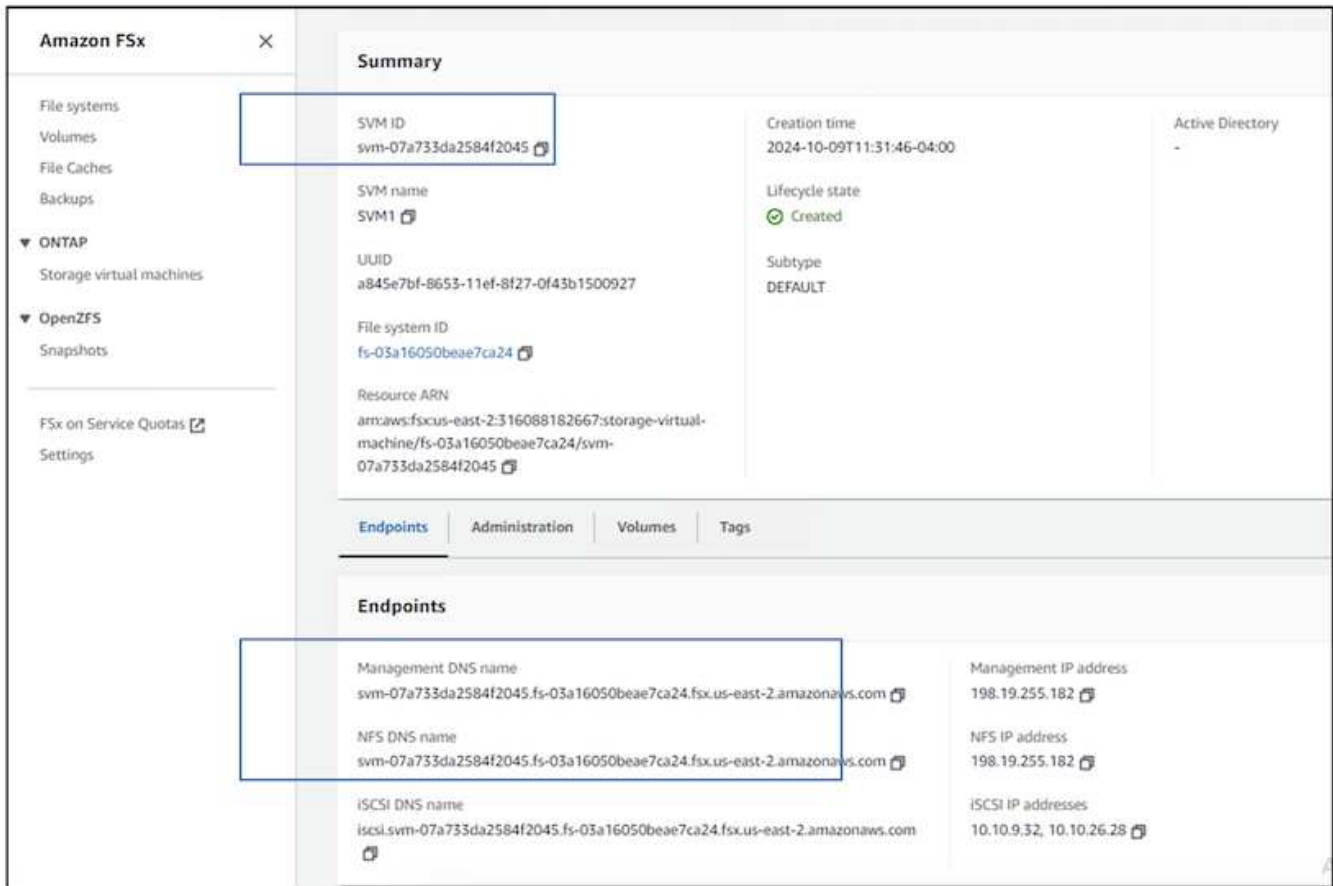
- C. 그런 다음 백엔드 객체를 생성하고, 복제된 Git 리포지토리의 FSX 디렉토리로 이동합니다. Backend-ONTAP-nas.yaml 파일을 엽니다. 다음 항목을 관리 DNS 이름 dataLIF 로, Amazon FSx SVM의 NFS DNS 이름으로, svm** 을 SVM 이름으로 바꿉니다. 다음 명령을 사용하여 백엔드 객체를 생성합니다.

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 확인할 수 있습니다



- d. 이제 다음 명령을 실행하여 백엔드 객체가 생성되었고 Phase가 Bound로 표시되고 Status가 Success** 인지 확인합니다

```
[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 개체입니다. 애플리케이션에 대해 요청할 수 있는 스토리지 유형을 설명하고 분류합니다.
- a. 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

```

- b. Rosa 클러스터에서 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]#

```

Trident CSI 드라이버 설치 및 FSx for ONTAP 파일 시스템에 대한 연결이 완료되었습니다. 이제 FSx for ONTAP의 파일 볼륨을 사용하여 ROSA에 샘플 PostgreSQL 상태 저장 애플리케이션을 배포할 수 있습니다.

- C. Trident-CSI 스토리지 클래스를 사용하여 생성된 PVC 및 PVS가 없는지 확인합니다.**

```

[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get pvc -A

```

NAMESPACE	NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
openshift-monitoring	prometheus-data-prometheus-k8s-0	Bound	pvc-9a4553a5-07e9-440a-8a90-99e384c97624	100Gi	RWO	gp3-csi	<unset>	2d16h
openshift-monitoring	prometheus-data-prometheus-k8s-1	Bound	pvc-7d949aef-e00d-4d9a-8b54-514e885fba62	100Gi	RWO	gp3-csi	<unset>	2d16h
openshift-visualization-os-images	centos-stream9-bae111cd5a1	Bound	pvc-d6bb1444-cb3f-449b-8d7d-39d028496c16	30Gi	RWO	gp3-csi	<unset>	24h
openshift-visualization-os-images	centos-stream9-d82f4a141a4	Bound	pvc-82b0e84a-e5ef-452b-bf90-1aae4fe162c1	30Gi	RWO	gp3-csi	<unset>	44h
openshift-visualization-os-images	fedora-21a0f3e028cd	Bound	pvc-64f375ad-d377-456d-83a0-360e113ae79c	30Gi	RWO	gp3-csi	<unset>	44h
openshift-visualization-os-images	rhel18-0052d4f0eb259	Bound	pvc-2dc6de48-5916-411e-9c3b-99598f50be4c	30Gi	RWO	gp3-csi	<unset>	44h
openshift-visualization-os-images	rhel10-2521bd116e64	Bound	pvc-f4374ce7-568d-4afc-b035-0228c44544d4	30Gi	RWO	gp3-csi	<unset>	44h

```

[root@localhost hcp-testing]# oc get pv

```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	VOLUMEATTRIBUTESCLASS
pvc-2dc6de48-5916-411e-9c3b-99598f50be4c	30Gi	RWO	Delete	Bound	openshift-visualization-os-images/rhel18-0052d4f0eb259	gp3-csi	<unset>
pvc-64f375ad-d377-456d-83a0-360e113ae79c	30Gi	RWO	Delete	Bound	openshift-visualization-os-images/fedora-21a0f3e028cd	gp3-csi	<unset>
pvc-7d949aef-e00d-4d9a-8b54-514e885fba62	100Gi	RWO	Delete	Bound	openshift-monitoring/prometheus-data-prometheus-k8s-1	gp3-csi	<unset>
pvc-82b0e84a-e5ef-452b-bf90-1aae4fe162c1	30Gi	RWO	Delete	Bound	openshift-visualization-os-images/centos-stream9-d82f4a141a4	gp3-csi	<unset>
pvc-9a4553a5-07e9-440a-8a90-99e384c97624	100Gi	RWO	Delete	Bound	openshift-monitoring/prometheus-data-prometheus-k8s-0	gp3-csi	<unset>
pvc-d6bb1444-cb3f-449b-8d7d-39d028496c16	30Gi	RWO	Delete	Bound	openshift-visualization-os-images/centos-stream9-bae111cd5a1	gp3-csi	<unset>
pvc-f4374ce7-568d-4afc-b035-0228c44544d4	30Gi	RWO	Delete	Bound	openshift-visualization-os-images/rhel10-2521bd116e64	gp3-csi	<unset>

```

[root@localhost hcp-testing]#

```

- d. 응용 프로그램이 Trident CSI를 사용하여 PV를 생성할 수 있는지 확인합니다.**

FSX** 풀더에 제공된 PVC-Trident.yaml 파일을 사용하여 PVC를 생성합니다.

```
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_image11.png["Trident를 사용하여 테스트 PVC를 생성합니다"]
```

- 5. 샘플 PostgreSQL 상태 저장 응용 프로그램을 배포합니다**
- a. helm을 사용하여 PostgreSQL **를 설치합니다

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

```
[root@localhost hcp-testing]# helm install postgresql bitnami/postgresql -n postgresql --create-namespace
NAME: postgresql
LAST DEPLOYED: Mon Oct 14 06:52:58 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}" | 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:16.4.0-debian-12-r0 --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
    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,
password, and setting it through helm won't take effect. Deleting persistent volumes (PVs) will solve the issue.
```

- b. 응용 프로그램 포드가 실행 중이고 응용 프로그램에 대해 PVC 및 PV가 생성되었는지 확인합니다.**

```
[root@localhost hcp-testing]# oc get pods -n postgresql
NAME                READY   STATUS    RESTARTS   AGE
postgresql-0        1/1    Running   0           29m

[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

[root@localhost hcp-testing]# oc get pv | grep postgresql
pvc-e3ddd9bd-e6a7-4a4a-b935-f1c090fd8db6   8Gi        RWO             Retain        Bound        postgresql/data-postgresql-0
csi                                     <unset>   4h20m
```

- C. PostgreSQL 클라이언트 배포**

다음 명령을 사용하여 설치된 PostgreSQL 서버의 암호를 가져옵니다.

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

다음 명령을 사용하여 PostgreSQL 클라이언트를 실행하고 암호를 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/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:vl.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityContext.allowPrivilegeEscalation to true), runAsNonRoot != true (pod or container "postgresql-client" must set securityContext.runAsNonRoot to 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.
```

- d. 데이터베이스 및 테이블 만들기 테이블에 대한 스키마를 만들고 테이블에 두 개의 데이터 행을 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 | first_name | last_name
----+-----+-----
  1 | John      | Doe
(1 row)
```

```

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

```

NetApp ONTAP가 포함된 AWS 기반 Red Hat OpenShift Service

이 문서에서는 NetApp ONTAP를 AWS(ROSA)에서 Red Hat OpenShift Service와 함께 사용하는 방법에 대해 설명합니다.

볼륨 스냅샷을 생성합니다

- 1. 앱 볼륨 스냅샷 만들기** 이 섹션에서는 앱과 연결된 볼륨의 Trident 스냅샷을 만드는 방법을 보여 줍니다. 이것은 앱 데이터의 특정 시점 복제본입니다. 애플리케이션 데이터가 손실되면 이 시점 복제본에서 데이터를 복구할 수 있습니다. 참고: 이 스냅샷은 ONTAP(사내 또는 클라우드)의 원래 볼륨과 동일한 애그리게이트에 저장됩니다. 따라서 ONTAP 스토리지 Aggregate가 손실되면 해당 스냅샷에서 앱 데이터를 복구할 수 없습니다.
- a. VolumeSnapshotClass 생성 volume-snapshot-class.yaml이라는 파일에 다음 매니페스트를 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]#

```

- b. 그런 다음 스냅샷을 생성합니다** VolumeSnapshot을 생성하여 PostgreSQL 데이터의 시점 복제본을 생성하여 기존 PVC의 스냅샷을 생성합니다. 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

```

- C. 볼륨 스냅샷을 생성하고 볼륨 스냅샷이 생성되었는지 확인합니다**

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

```

[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc create -f postgresql-volume-snapshot.yaml -n postgresql
volumeSnapshot.snapshot.storage.k8s.io/postgresql-volume-snap-01 created
[root@localhost hcp-testing]# oc get VolumeSnapshot -n postgresql
NAME                                READYTOUSE  SOURCEPVC                SOURCESNAPSHOTCONTENT  RESTORESIZE  SNAPSHOTCLASS  SNAPSHOTCONTENT
postgresql-volume-snap-01          true        data-postgresql-0       41500Ki              fsx-snapclass  snapcontent-5baf4337-922e-4318-be82-6db822082339
[root@localhost hcp-testing]#

```

- d. 데이터 손실을 시뮬레이션하기 위해 데이터베이스를 삭제합니다(데이터 손실은 다양한 이유로 발생할 수 있습니다. 여기서는 데이터베이스를 삭제하여 시뮬레이션합니다.)**

```

postgres=# \c erp;
psql (16.2, server 16.4)
You are now connected to database "erp" as user "postgres".
erp=# SELECT * FROM persons;
 id |  firstname  |  lastname
----+-----+-----
  1 | John       | Doe
  2 | Jane       | Scott
(2 rows)

```

```

postgres=# DROP DATABASE erp;
DROP DATABASE
postgres=# \c erp;
connection to server at "postgresql" (172.30.103.67), port 5432 failed: FATAL: database "erp" does not exist
Previous connection kept
postgres=#

```

볼륨 스냅샷에서 복원합니다

- 1. 스냅샷에서 복원** 이 섹션에서는 앱 볼륨의 Trident 스냅샷에서 응용 프로그램을 복원하는 방법을 보여 줍니다.
- a. 스냅샷에서 볼륨 클론을 생성합니다**

볼륨을 이전 상태로 복원하려면 스냅샷의 데이터를 기반으로 새 PVC를 생성해야 합니다. 이렇게 하려면 다음

매니페스트를 pvc-clone.yaml이라는 파일에 저장합니다

```
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 클론을 생성합니다. 매니페스트를 적용하고 클론이 생성되었는지 확인합니다.

```
[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. 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]#
```

- C. 새 클론 PVC**를 사용하여 새 PostgreSQL 애플리케이션을 생성합니다

```
$ 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}" | 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
    --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"
    so that "/opt/bitnami/scripts/postgresql/entrypoint.sh /bin/bash" 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
WARNING: There are "resources" sections in the chart not set. Using "resourcesPreset" is not recommended for production. For production
ing to your workload needs:
- primary.resources
- readReplicas.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[root@localhost hcp-testing]#

```

- d. 응용 프로그램 포드가 실행 중인지 확인합니다**

```

[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]#

```

- 예 포드가 클론을 PVC**로 사용하는지 확인합니다

```

[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/postgresql to ip-10-0-1-1.us-east-2.compute.internal
  Normal   SuccessfulAttachVolume  3m54s attachdetach-controller   AttachVolume.Attach succeeded for volume pvc-83b9334d-47f181fddac6"
  Normal   AddedInterface   3m43s multus          Add eth0 [10.129.2.126/23] from ovn-kubernetes
  Normal   Pulled           3m43s kubelet         Container image "docker.io/bitnami/postgresql" 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 복원되었는지 확인하려면 컨테이너 콘솔로 돌아가 기존 데이터베이스를 표시합니다

```

[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:16.2 --env 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.allowPrivilegeEscalation to false), capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "postgresql-client" must set securityContext.runAsNonRoot to 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 |             |             |
 template1 | postgres | UTF8     | libc             | en_US.UTF-8 | en_US.UTF-8 |             |             |
(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 | first_name | last_name
----+-----+-----
  1 | John       | Doe
  2 | Jane       | Scott
(2 rows)

```

데모 비디오

Amazon FSx for NetApp ONTAP은 호스팅된 Control Plane을 사용하는 AWS 기반 Red Hat OpenShift Service를 제공합니다

Red Hat OpenShift 및 OpenShift 솔루션에 대한 더 많은 비디오를 찾을 수 ["여기"](#)있습니다.

저작권 정보

Copyright © 2025 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이선스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이선스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이선스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이선스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 <http://www.netapp.com/TM>에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.