



## **Red Hat OpenShift Service on AWS with FSxN**

NetApp container solutions

NetApp  
January 21, 2026

# Table of Contents

Red Hat OpenShift Service on AWS with FSxN .....	1
Red Hat OpenShift Service on AWS with NetApp ONTAP .....	1
Overview .....	1
Prerequisites .....	1
Initial Setup .....	2
Red Hat OpenShift Service on AWS with NetApp ONTAP .....	16
Create Volume Snapshot .....	16
Restore from Volume Snapshot .....	18
Demo video .....	22

# Red Hat OpenShift Service on AWS with FSxN

## Red Hat OpenShift Service on AWS with NetApp ONTAP

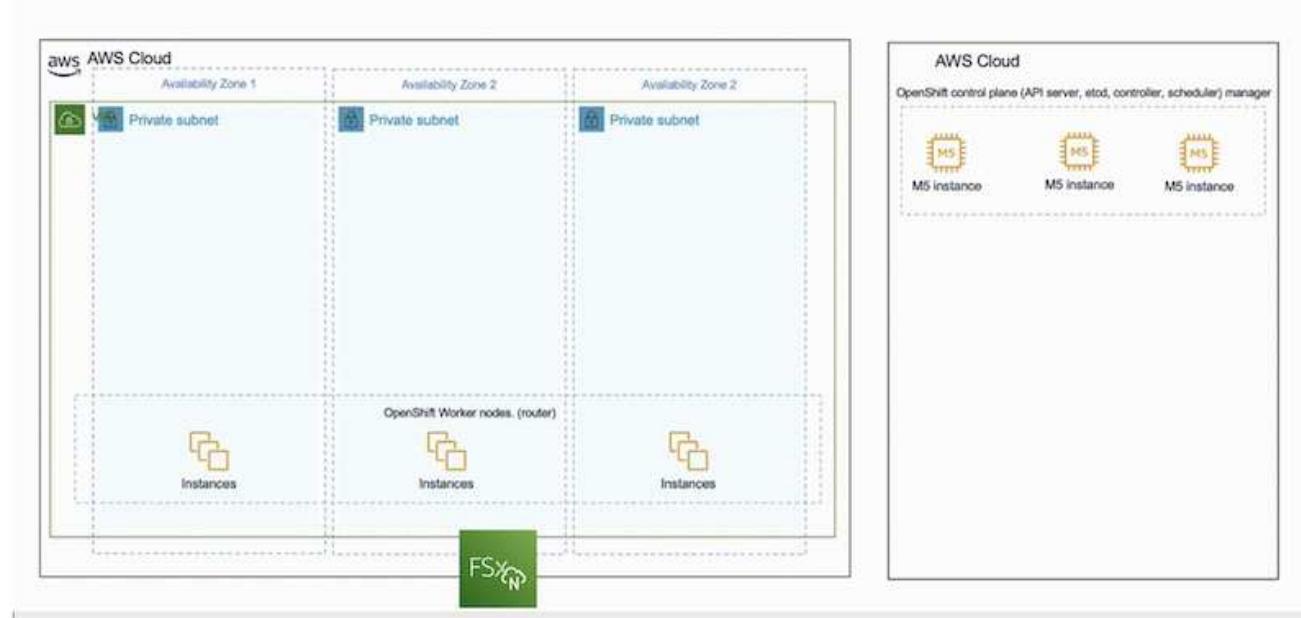
### Overview

In this section, we will show how to utilize FSx for ONTAP as a persistent storage layer for applications running on ROSA. It will show the installation of the NetApp Trident CSI driver on a ROSA cluster, the provisioning of an FSx for ONTAP file system, and the deployment of a sample stateful application. It will also show strategies for backing up and restoring your application data. With this integrated solution, you can establish a shared storage framework that effortlessly scales across AZs, simplifying the processes of scaling, protecting, and restoring your data using the Trident CSI driver.

### Prerequisites

- [AWS account](#)
- [A Red Hat account](#)
- IAM user [with appropriate permissions](#) to create and access ROSA cluster
- [AWS CLI](#)
- [ROSA CLI](#)
- [OpenShift command-line interface \(oc\)](#)
- Helm 3 [documentation](#)
- [A HCP ROSA cluster](#)
- [Access to Red Hat OpenShift web console](#)

This diagram shows the ROSA cluster deployed in multiple AZs. ROSA cluster's master nodes, infrastructure nodes are in Red Hat's VPC, while the worker nodes are in a VPC in the customer's account . We'll create an FSx for ONTAP file system within the same VPC and install the Trident driver in the ROSA cluster, allowing all the subnets of this VPC to connect to the file system.



## Initial Setup

### 1. Provision FSx for NetApp ONTAP

Create a multi-AZ FSx for NetApp ONTAP in the same VPC as the ROSA cluster. There are several ways to do this. The details of creating FSxN using a CloudFormation Stack are provided

#### a. Clone the GitHub repository

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

#### b. Run the CloudFormation Stack

Run the command below by replacing the parameter values with your own values:

```
$ 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
```

Where :

region-name: same as the region where the ROSA cluster is deployed

subnet1\_ID : id of the Preferred subnet for FSxN

subnet2\_ID: id of the Standby subnet for FSxN

VPC\_ID: id of the VPC where the ROSA cluster is deployed

routetable1\_ID, routetable2\_ID: ids of the route tables associated with the subnets chosen above

your\_allowed\_CIDR: allowed CIDR range for the FSx for ONTAP security groups ingress rules to control access. You can use 0.0.0.0/0 or any appropriate CIDR to allow all

traffic to access the specific ports of FSx for ONTAP.

Define Admin password: A password to login to FSxN

Define SVM password: A password to login to SVM that will be created.

Verify that your file system and storage virtual machine (SVM) has been created using the Amazon FSx console, shown below:

File system ID	SSD storage capacity	Availability Zones
OntapFileSystem	1024 GiB	us-east-2a (Preferred)  us-east-2b (Standby)
	Throughput capacity	Creation time
	1024 MB/s	2024-10-09T11:29:33-04:00
	Provisioned IOPS	
	3072	
	Deployment type	
	Multi-AZ 1	
	Number of HA pairs	
	1	

## 2. Install and configure Trident CSI driver for the ROSA cluster

## b. Install Trident

ROSA cluster worker nodes come pre-configured with nfs tools that enable you to use NAS protocols for storage provisioning and access.

If you would like to use iSCSI instead, you need to prepare the worker nodes for iSCSI. Starting from Trident 25.02 release, you can easily prepare the worker nodes of the ROSA cluster (or any OpenShift cluster) to perform iSCSI operations on FSxN storage.

There are 2 easy ways of installing Trident 25.02 (or later) that automates worker node preparation for iSCSI.

1. using the node-prep-flag from the command line using tridentctl tool.
2. Using the Red Hat certified Trident operator from the operator hub and customizing it.
3. Using Helm.



Using any of the above methods without enabling the node-prep will allow you to only use NAS protocols for provisioning storage on FSxN.

### Method 1: Use tridentctl tool

Use the node-prep flag and install Trident as shown.

Prior to issuing the install command, you should have downloaded installer package. Refer to [the documentation here](#).

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

### Method 2: Use the Red Hat Certified Trident Operator and customize

From the OperatorHub, locate the Red Hat certified Trident operator and install it.

The screenshot shows the Red Hat OpenShift OperatorHub interface. The left sidebar has a dark theme with categories: Home, Operators (selected), Workloads, Networking, Storage, Builds, and Observe. The Operators category has a sub-menu with OperatorHub (selected) and Installed Operators. The main content area has a header 'OperatorHub' and a sub-header 'Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a simple way to use them in your applications.' Below this is a search bar with the query 'trident'. Two operators are listed: 'NetApp Trident' (Certified, provided by NetApp, Inc.) and 'Trident Operator' (Community, provided by NetApp, Inc.).

Project: All Projects

OperatorHub

Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red Hat installation; the Operator capabilities will appear in the Developer Catalog providing a self-service experience.

All Items

Search: trident

NetApp Trident (Certified)

Trident Operator (Community)

NetApp Trident (Certified) provided by NetApp, Inc.

Trident Operator, to manage NetApp Trident installations

NetApp Trident (Community) provided by NetApp, Inc.

Trident Operator, to manage NetApp Trident installations

Channel: stable

Version: 25.2.0

Capability level: N/A

Source: Certified

Provider: NetApp, Inc.

Infrastructure features: Container Storage Interface, Disconnected

Repository: https://github.com/netapp/pytrident

Container image: docker.io/netapp/trident-operator:sha256:4250452b588fb100e04a048d626c444b276fb0e3243a7813424f5a23fb50c77e6

Created at: Mar 9, 2024, 7:00 PM

Support: NetApp

Activate Windows  
Go to Settings to activate Windows.

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

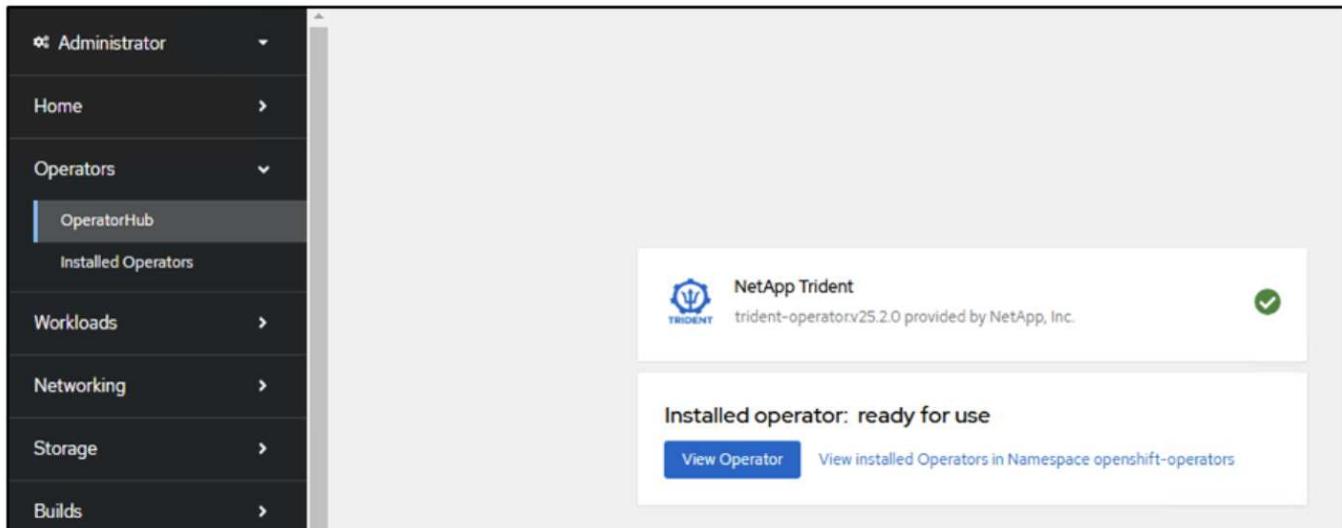
Install Cancel

NetApp Trident  
provided by NetApp, Inc.

Provided APIs

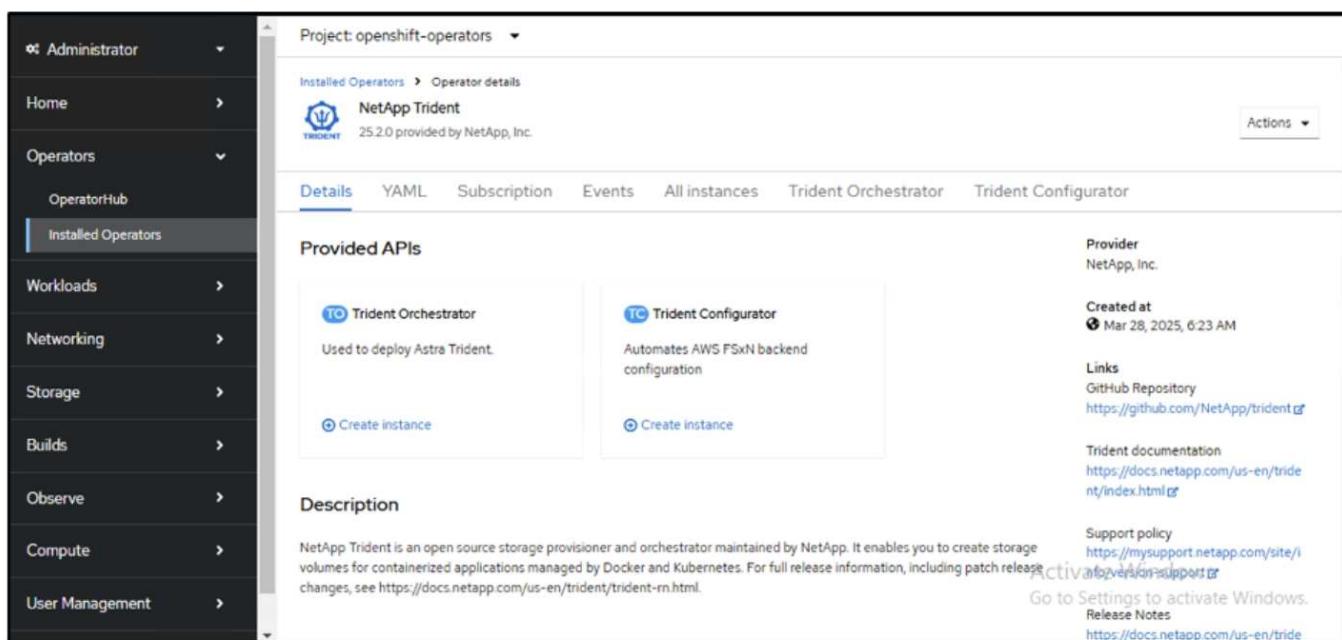
Trident Orchestrator  
Used to deploy Astra Trident.

Trident Configurator  
Automates AWS FSxN backend configuration



The screenshot shows the OperatorHub interface. The left sidebar has a dropdown for 'Administrator' and links for 'Home', 'Operators' (selected), 'OperatorHub' (selected), 'Installed Operators', 'Workloads', 'Networking', 'Storage', and 'Builds'. The main content area shows a card for 'NetApp Trident' with a 'TRIDENT' logo, the name 'NetApp Trident', the version 'trident-operatorv25.2.0 provided by NetApp, Inc.', and a green checkmark icon. Below the card, a message says 'Installed operator: ready for use' with 'View Operator' and 'View installed Operators in Namespace openshift-operators' buttons.

Next, create the Trident Orchestrator instance. Use the YAML view to set any custom values or enable iscsi node prep during installation.



The screenshot shows the 'Operator details' page for 'NetApp Trident' in the 'openshift-operators' project. The left sidebar is identical to the previous screenshot. The main content area shows the 'Details' tab selected. It displays the 'Provided APIs' section with two items: 'Trident Orchestrator' (used to deploy Astra Trident) and 'Trident Configurator' (automates AWS FSxN backend configuration). Each item has a 'Create instance' button. The 'Description' section contains a detailed description of the NetApp Trident operator. On the right side, there are sections for 'Provider' (NetApp, Inc.), 'Created at' (Mar 28, 2025, 6:23 AM), 'Links' (GitHub Repository: <https://github.com/NetApp/trident>), 'Trident documentation' (<https://docs.netapp.com/us-en/trident/index.html>), 'Support policy' (<https://mysupport.netapp.com/site/info/activeversion-support>), and 'Release Notes' (<https://docs.netapp.com/us-en/tride>).

Project: openshift-operators

## Create TridentOrchestrator

Create by completing the form. Default values may be provided by the Operator authors.

Configure via:  Form view  YAML view

```

1  kind: TridentOrchestrator
2  apiVersion: trident.netapp.io/v1
3  metadata:
4    name: trident
5  spec:
6    IPv6: false
7    debug: true
8    enableNodePrep: true
9    imagePullSecrets: []
10   imageRegistry: ''
11   k8sTimeout: 30
12   kubeletDir: /var/lib/kubelet
13   namespace: trident
14   silenceAutosupport: false
15

```

**Create** **Cancel**

Project: openshift-operators

Installed Operators > Operator details

**NetApp Trident**  
25.2.0 provided by NetApp, Inc.

Actions ▾

Details YAML Subscription Events All instances **Trident Orchestrator** Trident Configurator

**TridentOrchestrators**

**Create TridentOrchestrator**

Name	Kind	Status	Labels
trident	TridentOrchestrator	Status: Installed	No labels

```
[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-rnrrnn            2/2    Running   0          38s
trident-node-linux-t9bxj             2/2    Running   0          38s
trident-node-linux-vqv19             2/2    Running   0          38s
[root@localhost RedHat]#
```

Installing Trident using any of the above methods will prepare the ROSA cluster worker nodes for iSCSI by starting the iscsid and multipathd services and setting the following in /etc/multipath.conf file

```
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# 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. Verify that all Trident pods are in the running state

```
[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. Configure the Trident CSI backend to use FSx for ONTAP (ONTAP NAS)

The Trident back-end configuration tells Trident how to communicate with the storage system (in this case, FSx for ONTAP). For creating the backend, we will provide the credentials of the Storage Virtual machine to connect to, along with the Cluster Management and the NFS data interfaces. We will use the [ontap-nas driver](#) to provision storage volumes in FSx file system.

#### a. First, create a secret for the SVM credentials using the following 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>

```



You can also retrieve the SVM password created for FSxN from the AWS Secrets Manager as shown below.

Secret name	Description	Last retrieved (UTC)
HCP-ROSA-FSXONTAP-SVMAdminPassword	SVMAdminPassword	October 9, 2024
HCP-ROSA-FSXONTAP-FsxAdminPassword	FsxAdminPassword	

**b.Next, add the secret for the SVM credentials to the ROSA cluster using the following command**

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

You can verify that the secret has been added in the trident namespace using the following command

```
$ 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. Next, create the backend object

For this, move into the **fsx** directory of your cloned Git repository. Open the file `backend-ontap-nas.yaml`. Replace the following:

**managementLIF** with the Management DNS name

**dataLIF** with the NFS DNS name of the Amazon FSx SVM and

**svm** with the SVM name. Create the backend object using the following command.

Create the backend object using the following command.

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



You can get the Management DNS name, NFS DNS name and the SVM name from the Amazon FSx Console as shown in the screenshot below

The screenshot shows the Amazon FSx console interface. On the left, there's a sidebar with navigation links: 'Amazon FSx', 'File systems', 'Volumes', 'File Caches', 'Backups', 'ONTAP' (expanded to show 'Storage virtual machines'), 'OpenZFS' (expanded to show 'Snapshots'), 'FSx on Service Quotas', and 'Settings'. The main area is titled 'Summary' and contains the following details for an SVM:

SVM ID	svm-07a733da2584f2045	Creation time	2024-10-09T11:31:46-04:00	Active Directory
SVM name	SVM1	Lifecycle state	Created	
UUID	a845e7bf-8653-11ef-8f27-0f43b1500927	Subtype	DEFAULT	
File system ID	fs-03a16050beae7ca24			
Resource ARN	arn:aws:fsx:us-east-2:316088182667:storage-virtual-machine/fs-03a16050beae7ca24/svm-07a733da2584f2045			

Below the summary, there are tabs for 'Endpoints', 'Administration', 'Volumes', and 'Tags'. The 'Endpoints' tab is selected and shows the following details:

Management DNS name	svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com	Management IP address	198.19.255.182
NFS DNS name	svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com	NFS IP address	198.19.255.182
iSCSI DNS name	iscsi.svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com	iSCSI IP addresses	10.10.9.32, 10.10.26.28

### d. Now, run the following command to verify that the backend object has been created and Phase is

## showing Bound and Status is Success

```
[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. Create Storage Class

Now that the Trident backend is configured, you can create a Kubernetes storage class to use the backend. Storage class is a resource object made available to the cluster. It describes and classifies the type of storage that you can request for an application.

### a. Review the file **storage-class-csi-nas.yaml** in the **fsx** folder.

```
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. Create Storage Class in ROSA cluster and verify that **trident-csi** storage class has been created.

```
[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]#
```

This completes the installation of Trident CSI driver and its connectivity to FSx for ONTAP file system. Now you can deploy a sample Postgresql stateful application on ROSA using file volumes on FSx for ONTAP.

### c. Verify that there are no PVCs and PVs created using the **trident-csi** storage class.

NAME	NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
openshift-monitoring	prometheus-data-prometheus-kbs-0	Bound	pvc-9a455a5-07e9-440a-8a90-99e384c97624	100Gi	RwO	gp3-csi	<unset>	2d16h
openshift-monitoring	prometheus-data-prometheus-kbs-1	Bound	pvc-79949af-e80d-409a-8b54-514e085fbab2	100Gi	RwO	gp3-csi	<unset>	2d16h
openshift-virtualization-os-images	centos-stream9-bae11cd5a1	Bound	pvc-debb1444-cb3f-449b-8d7d-30a028496c16	300Gi	RwO	gp3-csi	<unset>	24h
openshift-virtualization-os-images	centos-stream9-d22f4a141a44	Bound	pvc-82b8e8aa-e5ef-452b-bf98-1eaef0e162c1	300Gi	RwO	gp3-csi	<unset>	44h
openshift-virtualization-os-images	fedora-21af6f3e628cd	Bound	pvc-64f375ad-d177-45d9-83a0-358e113ae79c	300Gi	RwO	gp3-csi	<unset>	44h
openshift-virtualization-os-images	rhel8-0b52d0eb259	Bound	pvc-2dc6de48-5916-411e-9cb3-9998f50be4c	300Gi	RwO	gp3-csi	<unset>	44h
openshift-virtualization-os-images	rhel9-2521bd116e64	Bound	pvc-f4374ce7-568d-4afc-bd35-0228cf4544d4	300Gi	RwO	gp3-csi	<unset>	44h

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	VOLUMEATTRIBUTESCLASS
pvc-64f375ad-d177-45d9-83a0-358e113ae79c	300Gi	RwO	Delete	Bound	openshift-virtualization-os-images/rhel8-0b52d0eb259	gp3-csi	<unset>
pvc-7d949aef-e00d-4d9a-8b54-514e085fbab2	100Gi	RwO	Delete	Bound	openshift-monitoring/prometheus-data-prometheus-kbs-1	gp3-csi	<unset>
pvc-82b8e8aa-e5ef-452b-bf98-1eaef0e162c1	300Gi	RwO	Delete	Bound	openshift-virtualization-os-images/centos-stream9-d22f4a141a44	gp3-csi	<unset>
pvc-9a4553a5-07e9-440a-8a90-99e384c97624	100Gi	RwO	Delete	Bound	openshift-monitoring/prometheus-data-prometheus-kbs-0	gp3-csi	<unset>
pvc-debb1444-cb3f-449b-8d7d-30a028496c16	300Gi	RwO	Delete	Bound	openshift-virtualization-os-images/centos-stream9-bae11cd5a1	gp3-csi	<unset>
pvc-f4374ce7-568d-4afc-bd35-0228cf4544d4	300Gi	RwO	Delete	Bound	openshift-virtualization-os-images/rhel9-2521bd116e64	gp3-csi	<unset>

#### d. Verify that applications can create PV using Trident CSI.

Create a PVC using the pvc-trident.yaml file provided in the **fsx** folder.

```
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.

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
basic	Bound	pvc-adb709b8-fe12-4d4e-9a6b-2afb345bad29	10Gi	RwX	trident-csi	<unset>	9s



To use iSCSI, you should have enabled iSCSI on the worker nodes as shown previously and you need to create an iSCSI backend and storage class. Here are some sample yaml files.

```

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. Deploy a sample Postgresql stateful application

### a. Use helm to install postgresql

```

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

**b. Verify that the application pod is running, and a PVC and PV is created for the application.**

```
[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
[root@localhost hcp-testing]#
```

**c. Deploy a Postgresql client**

Use the following command to get the password for the postgresql server that was installed.

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

Use the following command to run a postgresql client and connect to the server using the password

```
$ 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:v1.24": allowPrivilegeEscalation != false (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 "Local")
If you don't see a command prompt, try pressing enter.
```

d. Create a database and a table. Create a schema for the table and insert 2 rows of data into the table.

```
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)
```

## Red Hat OpenShift Service on AWS with NetApp ONTAP

This document will outline how to use NetApp ONTAP with the Red Hat OpenShift Service on AWS (ROSA).

### Create Volume Snapshot

#### 1. Create a Snapshot of the app volume

In this section, we will show how to create a trident snapshot of the volume associated with the app. This will be a point in time copy of the app data. If the application data is lost, we can recover the data from this point in time copy.

NOTE: This snapshot is stored in the same aggregate as the original volume in ONTAP(on-premises or in the cloud). So if the ONTAP storage aggregate is lost, we cannot recover the app data from its snapshot.

#### **\*\*a. Create a VolumeSnapshotClass**

Save the following manifest in a file called volume-snapshot-class.yaml

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

Create a snapshot by using the above manifest.

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

#### **b. Next, create a snapshot**

Create a snapshot of the existing PVC by creating VolumeSnapshot to take a point-in-time copy of your Postgresql data. This creates an FSx snapshot that takes almost no space in the filesystem backend. Save the following manifest in a file called 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
```

#### **c. Create the volume snapshot and confirm that it is created**

Delete the database to simulate the loss of data (data loss can happen due to a variety of reasons, here we are just simulating it by deleting the database)

```
[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. Delete the database to simulate the loss of data (data loss can happen due to a variety of reasons, here we are just simulating it by deleting the database)

```
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#
```

## Restore from Volume Snapshot

### 1. Restore from Snapshot

In this section, we will show how to restore an application from the trident snapshot of the app volume.

#### a. Create a volume clone from the snapshot

To restore the volume to its previous state, you must create a new PVC based on the data in the snapshot you took. To do this, save the following manifest in a file named 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
```

Create a clone of the volume by creating a PVC using the snapshot as the source using the above manifest. Apply the manifest and ensure that the clone is created.

```
[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
data-postgresql-0          Bound   pvc-e3ddd9bd-e6a7-4a4a-b935-f1c090fd8db6  8Gi    RWO
postgresql-volume-clone   Bound   pvc-b38fbc54-55dc-47e8-934d-47f181fddac6  8Gi    RWO
[root@localhost hcp-testing]#
```

**b. Delete the original postgresql installation**

```
[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. Create a new postgresql application using the new clone PVC**

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

```
[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 --decode)

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

d. Verify that the application pod is in the running state

```
[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]#
```

e. Verify that the pod uses the clone as its 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-kubernetes
  Normal Pulled          3m43s kubelet         Container image "docker.io/bitnami/postgres:10.0" already present on machine
  Normal Created          3m42s kubelet         Created container postgresql
  Normal Started          3m42s kubelet         Started container postgresql
[root@localhost hcp-testing]# 

```

f) To validate that the database has been restored as expected, go back to the container console and show the existing databases

```

[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:10.0 --env POSTGRES_PASSWORD="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 |             |             | =c/postgres      +
   postgres | postgres | UTF8    | libc            | en_US.UTF-8 | en_US.UTF-8 |             |             | =c/postgres      +
   template0 | postgres | UTF8    | libc            | en_US.UTF-8 | en_US.UTF-8 |             |             | =c/postgres      +
   template1 | postgres | UTF8    | libc            | en_US.UTF-8 | en_US.UTF-8 |             |             | =c/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)

```

## Demo video

[Amazon FSx for NetApp ONTAP wth Red Hat OpenShift Service on AWS using Hosted Control Plane](#)

More videos on Red Hat OpenShift and OpenShift solutions can be found [here](#).

## Copyright information

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—with prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

**LIMITED RIGHTS LEGEND:** Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

## Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.