



Serviço Red Hat OpenShift na AWS com FSxN

NetApp container solutions

NetApp
January 21, 2026

Índice

Serviço Red Hat OpenShift na AWS com FSxN	1
Serviço Red Hat OpenShift na AWS com NetApp ONTAP	1
Visão geral	1
Pré-requisitos	1
Configuração inicial	2
Serviço Red Hat OpenShift na AWS com NetApp ONTAP	17
Criar instantâneo de volume	17
Restaurar a partir do instantâneo de volume	18
Vídeo de demonstração	22

Serviço Red Hat OpenShift na AWS com FSxN

Serviço Red Hat OpenShift na AWS com NetApp ONTAP

Visão geral

Nesta seção, mostraremos como utilizar o FSx para ONTAP como uma camada de armazenamento persistente para aplicativos em execução no ROSA. Ele mostrará a instalação do driver NetApp Trident CSI em um cluster ROSA, o provisionamento de um FSx para o sistema de arquivos ONTAP e a implantação de um aplicativo com estado de exemplo. Ele também mostrará estratégias para fazer backup e restaurar os dados do seu aplicativo. Com esta solução integrada, você pode estabelecer uma estrutura de armazenamento compartilhado que pode ser facilmente dimensionada entre AZs, simplificando os processos de dimensionamento, proteção e restauração de seus dados usando o driver Trident CSI.

Pré-requisitos

- "[Conta AWS](#)"
- "[Uma conta Red Hat](#)"
- Usuário IAM"[com permissões apropriadas](#)" para criar e acessar o cluster ROSA
- "[CLI da AWS](#)"
- "[ROSA CLI](#)"
- "[Interface de linha de comando OpenShift](#)"(oc)
- Leme 3"[documentação](#)"
- "[Um cluster HCP ROSA](#)"
- "[Acesso ao console web do Red Hat OpenShift](#)"

Este diagrama mostra o cluster ROSA implantado em várias AZs. Os nós mestres e os nós de infraestrutura do cluster ROSA estão na VPC da Red Hat, enquanto os nós de trabalho estão em uma VPC na conta do cliente. Criaremos um FSx para o sistema de arquivos ONTAP dentro da mesma VPC e instalaremos o driver Trident no cluster ROSA, permitindo que todas as sub-redes dessa VPC se conectem ao sistema de arquivos.



Configuração inicial

1. Provisão FSx para NetApp ONTAP

Crie um FSx multi-AZ para o NetApp ONTAP na mesma VPC que o cluster ROSA. Há várias maneiras de fazer isso. Os detalhes da criação do FSxN usando uma pilha CloudFormation são fornecidos

a. Clone o repositório GitHub

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

b. Execute o CloudFormation Stack

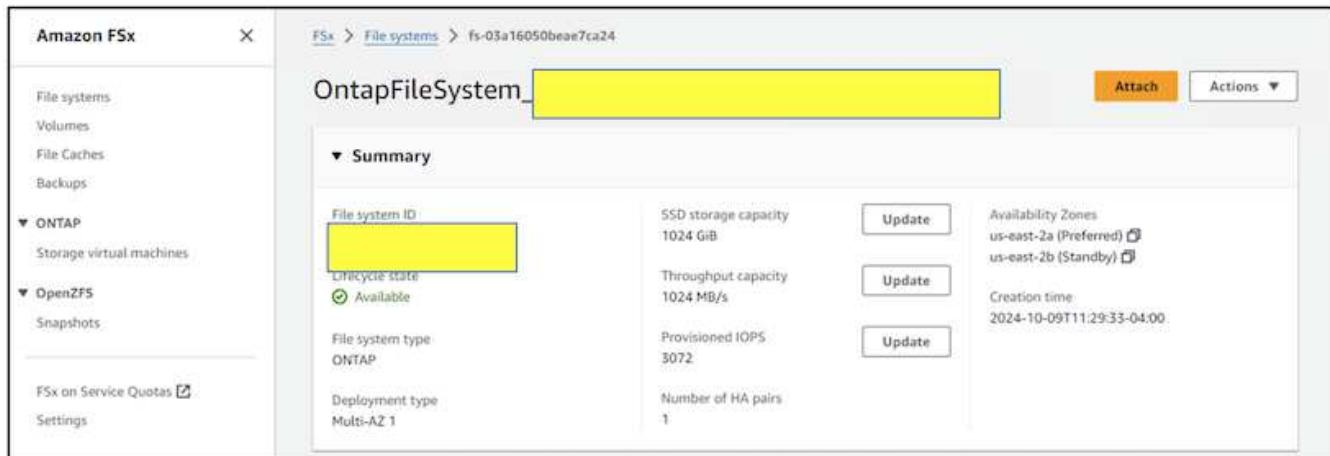
Execute o comando abaixo substituindo os valores dos parâmetros pelos seus próprios valores:

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

```
$ aws cloudformation create-stack \
--stack-name ROSA-FSxONTAP \
--template-body file://./FSxONTAP.yaml \
--region <region-name> \
--parameters \
ParameterKey=Subnet1ID,ParameterValue=[subnet1_ID] \
ParameterKey=Subnet2ID,ParameterValue=[subnet2_ID] \
ParameterKey=myVpc,ParameterValue=[VPC_ID] \
ParameterKey=FSxONTAPRouteTable,ParameterValue=[routetable1_ID,routetable2_ID] \
ParameterKey=FileSystemName,ParameterValue=ROSA-myFSxONTAP \
ParameterKey=ThroughputCapacity,ParameterValue=1024 \
ParameterKey=FSxAllowedCIDR,ParameterValue=[your_allowed_CIDR] \
ParameterKey=FsxAdminPassword,ParameterValue=[Define Admin password] \
ParameterKey=SvmAdminPassword,ParameterValue=[Define SVM password] \
--capabilities CAPABILITY_NAMED_IAM
```

Onde: region-name: o mesmo que a região onde o cluster ROSA está implantado subnet1_ID: id da sub-rede preferencial para FSxN subnet2_ID: id da sub-rede em espera para FSxN VPC_ID: id da VPC onde o cluster ROSA está implantado routetable1_ID, routetable2_ID: ids das tabelas de rota associadas às sub-redes escolhidas acima your_allowed_CIDR: intervalo CIDR permitido para o FSx para grupos de segurança ONTAP regras de entrada para controlar o acesso. Você pode usar 0.0.0.0/0 ou qualquer CIDR apropriado para permitir que todo o tráfego acesse as portas específicas do FSx para ONTAP. Definir senha do administrador: uma senha para fazer login no FSxN Definir senha do SVM: uma senha para fazer login no SVM que será criado.

Verifique se seu sistema de arquivos e máquina virtual de armazenamento (SVM) foram criados usando o console do Amazon FSx , mostrado abaixo:



2. Instale e configure o driver Trident CSI para o cluster ROSA

b. Instalar o Trident

Os nós de trabalho do cluster ROSA vêm pré-configurados com ferramentas nfs que permitem usar protocolos NAS para provisionamento e acesso ao armazenamento.

Se você quiser usar iSCSI, precisará preparar os nós de trabalho para iSCSI. A partir da versão Trident 25.02, você pode preparar facilmente os nós de trabalho do cluster ROSA (ou qualquer cluster OpenShift) para executar operações iSCSI no armazenamento FSxN. Há duas maneiras fáceis de instalar o Trident 25.02 (ou posterior) que automatiza a preparação do nó de trabalho para iSCSI. 1. usando o node-prep-flag na linha de comando usando a ferramenta tridentctl. 2. Usando o operador Trident certificado pela Red Hat do hub do operador e personalizando-o. 3.Usando o Helm.



Usar qualquer um dos métodos acima sem habilitar o node-prep permitirá que você use apenas protocolos NAS para provisionar armazenamento no FSxN.

Método 1: Use a ferramenta tridentctl

Use o sinalizador node-prep e instale o Trident conforme mostrado. Antes de emitir o comando de instalação, você deve ter baixado o pacote do instalador. Consulte "[a documentação aqui](#)" .

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

Método 2: usar o operador Red Hat Certified Trident e personalizar No OperatorHub, localize o operador Red Hat Certified Trident e instale-o.

The screenshot shows the Red Hat OpenShift OperatorHub interface. On the left is a sidebar with navigation links: Home, Operators (selected), OperatorHub (highlighted in blue), Installed Operators, Workloads, Networking, Storage, Builds, and Observe. The main 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 H... optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a s...". Below this is a search bar with the text "trident". Two operator cards are displayed: "NetApp Trident" (Certified, provided by NetApp, Inc.) and "NetApp Trident" (Community, provided by NetApp, Inc.). Both cards describe the Trident Operator as managing NetApp Trident installations.

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

All Items

Search: trident

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@sha256:4250452b588fb100e04a048d62dc444b2f6fb0e3243a7813424f5a23b50c77e6

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

Support: NetApp

Activate Windows
Go to Settings to activate Windows.

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

The screenshot shows the OperatorHub interface. On the left, a sidebar menu includes Home, Operators (selected), OperatorHub, Installed Operators, Workloads, Networking, Storage, and Builds. The main area displays the NetApp Trident operator details. It features a logo, the name "NetApp Trident", the version "trident-operatorv25.2.0 provided by NetApp, Inc.", and a green checkmark indicating it is installed. Below this, a message says "Installed operator: ready for use". There are two buttons: "View Operator" and "View installed Operators in Namespace openshift-operators".

Em seguida, crie a instância do Trident Orchestrator. Use a visualização YAML para definir quaisquer valores personalizados ou habilitar a preparação do nó iscsi durante a instalação.

The screenshot shows the "Operator details" page for the NetApp Trident operator. The top navigation bar includes "Project: openshift-operators", "Actions", and tabs for Details (selected), YAML, Subscription, Events, All instances, Trident Orchestrator, and Trident Configurator. The "Provided APIs" section lists "Trident Orchestrator" and "Trident Configurator", each with a "Create instance" button. The "Description" section provides a brief overview of the operator's purpose and links to documentation and support policies. A "Links" section includes a GitHub Repository link. A "Trident documentation" link is also present.

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						
Name	Search by name...					
Name	Kind	Status	Labels			
trident	TridentOrchestrator	Status: Installed	No labels			

Create TridentOrchestrator

```
[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-rnrnnn            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]#
```

A instalação do Trident usando qualquer um dos métodos acima preparará os nós de trabalho do cluster ROSA para iSCSI iniciando os serviços iscsid e multipathd e definindo o seguinte no arquivo /etc/multipath.conf

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

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

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

c.Verifique se todos os pods Trident estão em execução

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

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

A configuração de back-end do Trident informa ao Trident como se comunicar com o sistema de armazenamento (neste caso, FSx para ONTAP). Para criar o backend, forneceremos as credenciais da máquina virtual de armazenamento para conexão, juntamente com o gerenciamento de cluster e as interfaces de dados NFS. Nós usaremos o "driver ontap-nas" para provisionar volumes de armazenamento no sistema de arquivos FSx.

um. Primeiro, crie um segredo para as credenciais do SVM usando o seguinte 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>

```



Você também pode recuperar a senha do SVM criada para o FSxN no AWS Secrets Manager, conforme mostrado abaixo.

The screenshot shows the AWS Secrets Manager interface with the following details:

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

The screenshot shows the "Secret details" page for the secret "HCP-ROSA-FSXONTAP-SVMAdminPassword". The details are as follows:

- Encryption key:** aws/secretsmanager
- Secret name:** HCP-ROSA-FSXONTAP-SVMAdminPassword
- Secret ARN:** arn:aws:secretsmanager:us-east-2:316088182667:secret:HCP-ROSA-FSXONTAP-SVMAdminPassword-ABiUaf
- Secret description:** SVMAdminPassword

At the bottom, there are tabs for Overview, Rotation, Versions, Replication, and Tags. A "Secret value" section contains a "Retrieve secret value" button.

b. Em seguida, adicione o segredo para as credenciais do SVM ao cluster ROSA usando o seguinte comando

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

Você pode verificar se o segredo foi adicionado no namespace trident usando o seguinte comando

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

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

c. Em seguida, crie o objeto de backend Para isso, vá para o diretório **fsx** do seu repositório Git clonado. Abra o arquivo **backend-ontap-nas.yaml**. Substitua o seguinte: **managementLIF** pelo nome DNS de gerenciamento, **dataLIF** pelo nome DNS NFS do Amazon FSx SVM e **svm** pelo nome do SVM. Crie o objeto de backend usando o seguinte comando.

Crie o objeto de backend usando o seguinte comando.

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



Você pode obter o nome DNS de gerenciamento, o nome DNS NFS e o nome SVM no console Amazon FSx, conforme mostrado na captura de tela abaixo

The screenshot shows the AWS FSx console interface. On the left, there's a sidebar with navigation links: File systems, Volumes, File Caches, Backups, ONTAP (Storage virtual machines), OpenZFS (Snapshots), FSx on Service Quotas, and Settings. The main area has a title bar 'Amazon FSx' and a 'Summary' tab selected. Under the 'File systems' section, the 'SVM ID' is highlighted. In the center, there are details for the SVM: 'SVM name' (SVM1), 'UUID' (a845e7bf-8653-11ef-8f27-0f43b1500927), 'File system ID' (fs-03a16050beae7ca24), and 'Resource ARN' (arn:aws:fsx:us-east-2:316088182667:storage-virtual-machine/fs-03a16050beae7ca24/svm-07a733da2584f2045). To the right, there are fields for 'Creation time' (2024-10-09T11:31:46-04:00), 'Lifecycle state' (Created), and 'Subtype' (DEFAULT). Below the summary, there are tabs for Endpoints, Administration, Volumes, and Tags. The 'Endpoints' tab is active, showing the 'Management DNS name' (svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com), 'NFS DNS name' (svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com), 'iSCSI DNS name' (iscsi.svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com), 'Management IP address' (198.19.255.182), 'NFS IP address' (198.19.255.182), and 'iSCSI IP addresses' (10.10.9.32, 10.10.26.28).

d. Agora, execute o seguinte comando para verificar se o objeto de backend foi criado e se a Fase está mostrando Bound e o Status é 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. Criar classe de armazenamento Agora que o backend do Trident está configurado, você pode criar uma classe de armazenamento do Kubernetes para usar o backend. A classe de armazenamento é um objeto de recurso disponibilizado ao cluster. Ele descreve e classifica o tipo de armazenamento que você pode solicitar para um aplicativo.

um. Revise o arquivo storage-class-csi-nas.yaml na pasta fsx.

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

b. Crie uma classe de armazenamento no cluster ROSA e verifique se a classe de armazenamento trident-csi foi criada.

```
[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]#
```

Isso conclui a instalação do driver Trident CSI e sua conectividade com o FSx para o sistema de arquivos ONTAP . Agora você pode implantar um aplicativo Postgresql com estado de exemplo no ROSA usando volumes de arquivo no FSx para ONTAP.

c. Verifique se não há PVCs e PVs criados usando a classe de armazenamento trident-csi.

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	VOLUMEATTRIBUTESCLASS	AGE
openshift-monitoring-prometheus-data-prometheus-kbs-0	Bound	pvc-9a455a5-07e9-44ba-8a90-99e384c97624	100Gi	RwO	gp3-csi	<unset>	2d16h
openshift-monitoring-prometheus-data-prometheus-kbs-1	Bound	pvc-79949aef-e80d-4d9a-8f54-514e885fbab2	100Gi	RwO	gp3-csi	<unset>	2d16h
openshift-virtualization-os-images-centos-stream9-bae11cd5a1	Bound	pvc-debb1444-cb3f-449b-8d7d-39d028496c16	30Gi	RwO	gp3-csi	<unset>	24h
openshift-virtualization-os-images-centos-stream9-d82f4a141a44	Bound	pvc-82bbbe84a-e5ef-452b-bf98-1eaef0e162c1	30Gi	RwO	gp3-csi	<unset>	44h
openshift-virtualization-os-images-fedora-21afbf628bcd	Bound	pvc-64f375ad-d377-45d6-83a8-388e113ae79c	30Gi	RwO	gp3-csi	<unset>	44h
openshift-virtualization-os-images-rhel8-0b52df0eb259	Bound	pvc-2dc6de48-5916-411e-9cb3-9959bf50be4c	30Gi	RwO	gp3-csi	<unset>	44h
openshift-virtualization-os-images-rhel9-2521bd116e64	Bound	pvc-f4374ce7-568d-4afc-bd35-0228cf4544d4	30Gi	RwO	gp3-csi	<unset>	44h

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	VOLUMEATTRIBUTESCLASS
pvc-64f375ad-d377-45d6-83a8-388e113ae79c	30Gi	RwO	Delete	Bound	openshift-virtualization-os-images/fedora-21afbf628bcd	gp3-csi	<unset>
pvc-7d949aef-e00d-4d9a-8b54-514e885fbab2	100Gi	RwO	Delete	Bound	openshift-monitoring/prometheus-data-prometheus-kbs-1	gp3-csi	<unset>
pvc-82bbbe84a-e5ef-452b-bf98-1eaef0e162c1	30Gi	RwO	Delete	Bound	openshift-virtualization-os-images/centos-stream9-d82f4a141a44	gp3-csi	<unset>
pvc-9a455a5-07e9-44ba-8a90-99e384c97624	100Gi	RwO	Delete	Bound	openshift-monitoring/prometheus-data-prometheus-kbs-0	gp3-csi	<unset>
pvc-debb1444-cb3f-449b-8d7d-39d028496c16	30Gi	RwO	Delete	Bound	openshift-virtualization-os-images/centos-stream9-bae11cd5a1	gp3-csi	<unset>
pvc-f4374ce7-568d-4afc-bd35-0228cf4544d4	30Gi	RwO	Delete	Bound	openshift-virtualization-os-images/rhel9-2521bd116e64	gp3-csi	<unset>

d. Verifique se os aplicativos podem criar PV usando o Trident CSI.

Crie um PVC usando o arquivo pvc-trident.yaml fornecido na pasta **fsx**.

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

You can issue the following commands to create a pvc and verify that it has been created.

image:redhat-openshift-container-rosa-011.png["criar PVC de teste usando Trident"]



Para usar o iSCSI, você deve ter habilitado o iSCSI nos nós de trabalho, conforme mostrado anteriormente, e precisa criar um backend iSCSI e uma classe de armazenamento. Aqui estão alguns arquivos yaml de exemplo.

```

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. Implantar um aplicativo Postgresql com estado de exemplo

um. Use o helm para instalar o 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. Verifique se o pod do aplicativo está em execução e se um PVC e um PV foram criados para o aplicativo.

```
[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. Implantar um cliente Postgresql

Use o seguinte comando para obter a senha do servidor postgresql que foi instalado.

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

Use o seguinte comando para executar um cliente postgresql e conectar-se ao servidor usando a senha

```
$ 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. Crie um banco de dados e uma tabela. Crie um esquema para a tabela e insira 2 linhas de dados na tabela.

```
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)

```

Serviço Red Hat OpenShift na AWS com NetApp ONTAP

Este documento descreverá como usar o NetApp ONTAP com o Red Hat OpenShift Service na AWS (ROSA).

Criar instantâneo de volume

1. Criar um instantâneo do volume do aplicativo Nesta seção, mostraremos como criar um instantâneo do Trident do volume associado ao aplicativo. Esta será uma cópia de um momento específico dos dados do aplicativo. Se os dados do aplicativo forem perdidos, podemos recuperar os dados desta cópia de momento. OBSERVAÇÃO: este instantâneo é armazenado no mesmo agregado que o volume original no ONTAP (no local ou na nuvem). Portanto, se o agregado de armazenamento ONTAP for perdido, não poderemos recuperar os dados do aplicativo a partir do seu snapshot.

**um. Crie uma VolumeSnapshotClass Salve o seguinte manifesto em um arquivo chamado volume-snapshot-class.yaml

```

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

```

Crie um snapshot usando o manifesto acima.

```

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

```

b. Em seguida, crie um snapshot Crie um snapshot do PVC existente criando o VolumeSnapshot para fazer uma cópia de um momento específico dos seus dados do Postgresql. Isso cria um instantâneo do FSx que ocupa quase nenhum espaço no backend do sistema de arquivos. Salve o seguinte manifesto em um arquivo

chamado 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. Crie o instantâneo do volume e confirme que ele foi criado

Exclua o banco de dados para simular a perda de dados (a perda de dados pode ocorrer por vários motivos, aqui estamos apenas simulando isso excluindo o banco de dados)

```
[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-5baef4337-922e-4318-be82-6db822082339
[root@localhost hcp-testing]#
```

d. Exclua o banco de dados para simular a perda de dados (a perda de dados pode ocorrer por vários motivos, aqui estamos apenas simulando isso excluindo o banco de dados)

```
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=#
Activate Window
Go to Settings to ad
```

Restaurar a partir do instantâneo de volume

1. Restaurar do Snapshot Nesta seção, mostraremos como restaurar um aplicativo a partir do snapshot do Trident do volume do aplicativo.

um. Crie um clone de volume a partir do snapshot

Para restaurar o volume ao seu estado anterior, você deve criar um novo PVC com base nos dados do instantâneo que você tirou. Para fazer isso, salve o seguinte manifesto em um arquivo chamado 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
```

Crie um clone do volume criando um PVC usando o snapshot como fonte usando o manifesto acima. Aplique o manifesto e certifique-se de que o clone foi criado.

```
[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. Excluir a instalação original do postgresql

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

c. Crie um novo aplicativo postgresql usando o novo 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]#
[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 helm. In this case, 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 production environments, please refer to your workload needs:
  - primary.resources
  - readReplicas.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[root@localhost hcp-testing]#
```

d. Verifique se o pod do aplicativo está em execução

```
[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. Verifique se o pod usa o clone como seu 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  
3-934d-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  
" already present on machine  
  Normal Created           3m42s kubelet         Created container postgresql  
  Normal Started          3m42s kubelet         Started container postgresql  
[root@localhost hcp-testing]#
```

f) Para validar se o banco de dados foi restaurado conforme o esperado, volte ao console do contêiner e mostre os bancos de dados existentes

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:  
POSTGRES_PASSWORD" --command -- psql --host postgresql -U postgres -d postgres -p 5432  
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityContext.allowPr  
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 | en_US.UTF-8 | en_US.UTF-8 |  
postgres | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 | en_US.UTF-8 | en_US.UTF-8 |  
template0 | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 | en_US.UTF-8 | en_US.UTF-8 |  
template1 | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 | 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 | firstname | lastname  
---+---+---+  
 1 | John | Doe  
 2 | Jane | Scott  
(2 rows)
```

Vídeo de demonstração

[Amazon FSx for NetApp ONTAP com Red Hat OpenShift Service na AWS usando o Hosted Control Plane](#)

Mais vídeos sobre o Red Hat OpenShift e as soluções OpenShift podem ser encontrados ["aqui"](#).

Informações sobre direitos autorais

Copyright © 2026 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTE DOCUMENTO. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSAENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTE SOFTWARE, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em <http://www.netapp.com/TM> são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.