Servizio Red Hat OpenShift su AWS con FSxN

NetApp container solutions

NetApp
January 21, 2026

Sommario

Servizio Red Hat OpenShift su AWS con FSxN

Servizio Red Hat OpenShift su AWS con NetApp ONTAP

Panoramica

In questa sezione mostreremo come utilizzare FSx per ONTAP come livello di archiviazione persistente per le applicazioni in esecuzione su ROSA. Verrà illustrata l'installazione del driver NetApp Trident CSI su un cluster ROSA, il provisioning di un file system FSx per ONTAP e la distribuzione di un'applicazione stateful di esempio. Verranno inoltre illustrate le strategie per eseguire il backup e il ripristino dei dati delle applicazioni. Grazie a questa soluzione integrata, è possibile creare un framework di archiviazione condiviso che si adatta senza problemi a tutte le zone di disponibilità, semplificando i processi di ridimensionamento, protezione e ripristino dei dati mediante il driver Trident CSI.

Prerequisiti

- "Account AWS"
- "Un account Red Hat"
- Utente IAM"con le autorizzazioni appropriate" per creare e accedere al cluster ROSA
- "Interfaccia a riga di comando AWS"
- "ROSA CLI"
- "Interfaccia della riga di comando di OpenShift"(oc)
- Timone 3"documentazione"
- "Un cluster HCP ROSA"
- "Accesso alla console web di Red Hat OpenShift"

Questo diagramma mostra il cluster ROSA distribuito in più AZ. I nodi master del cluster ROSA e i nodi infrastrutturali si trovano nella VPC di Red Hat, mentre i nodi worker si trovano in una VPC nell'account del cliente. Creeremo un file system FSx per ONTAP all'interno della stessa VPC e installeremo il driver Trident nel cluster ROSA, consentendo a tutte le subnet di questa VPC di connettersi al file system.

Configurazione iniziale

1. Provisioning FSx per NetApp ONTAP

Creare un FSx multi-AZ per NetApp ONTAP nella stessa VPC del cluster ROSA. Ci sono diversi modi per farlo. Vengono forniti i dettagli sulla creazione di FSxN utilizzando uno stack CloudFormation

a.Clona il repository GitHub

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

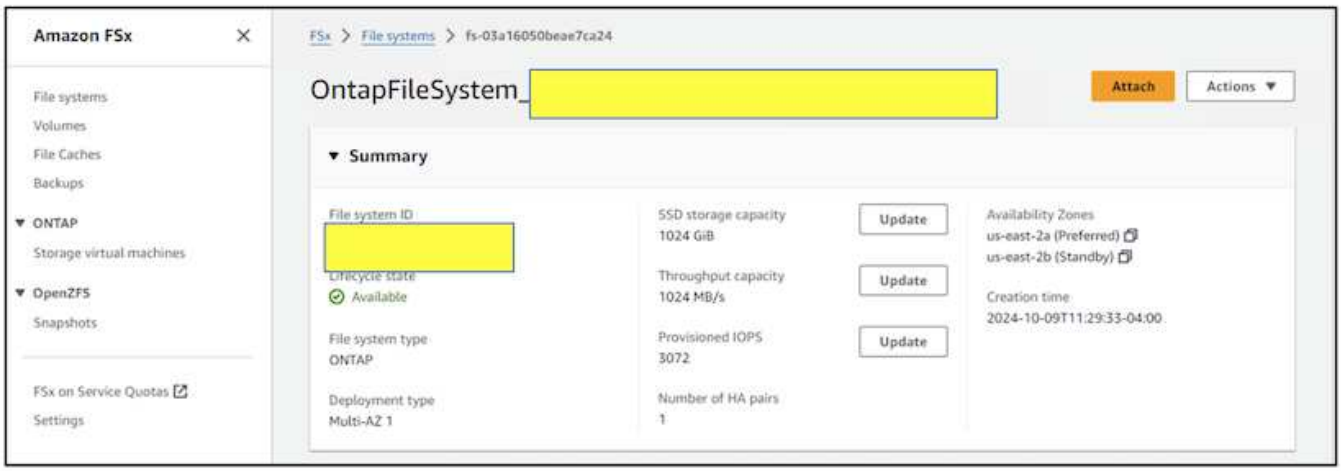
b.Esegui lo stack CloudFormation Esegui il comando seguente sostituendo i valori dei parametri con i tuoi valori:

```
$ 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
```

Dove: region-name: uguale alla regione in cui è distribuito il cluster ROSA subnet1_ID: ID della subnet preferita per FSxN subnet2_ID: ID della subnet di standby per FSxN VPC_ID: ID della VPC in cui è distribuito il cluster ROSA routetable1_ID, routetable2_ID: ID delle tabelle di routing associate alle subnet scelte sopra your_allowed_CIDR: intervallo CIDR consentito per le regole di ingresso dei gruppi di sicurezza FSx for ONTAP per controllare l'accesso. È possibile utilizzare 0.0.0.0/0 o qualsiasi CIDR appropriato per consentire a tutto il traffico di accedere alle porte specifiche di FSx per ONTAP. Definisci password amministratore: una password per accedere a FSxN Definisci password SVM: una password per accedere a SVM che verrà creata.

Verifica che il file system e la macchina virtuale di archiviazione (SVM) siano stati creati utilizzando la console Amazon FSx , mostrata di seguito:



2. Installare e configurare il driver Trident CSI per il cluster ROSA

b.Installa Trident

I nodi worker del cluster ROSA sono preconfigurati con strumenti NFS che consentono di utilizzare protocolli NAS per il provisioning e l'accesso allo storage.

Se invece si desidera utilizzare iSCSI, è necessario preparare i nodi worker per iSCSI. A partire dalla versione Trident 25.02, è possibile preparare facilmente i nodi worker del cluster ROSA (o di qualsiasi cluster OpenShift) per eseguire operazioni iSCSI sullo storage FSxN. Esistono 2 semplici modi per installare Trident 25.02 (o versione successiva) che automatizza la preparazione del nodo worker per iSCSI. 1. utilizzando il node-prep-flag dalla riga di comando tramite lo strumento tridentctl. 2. Utilizzo dell'operatore Trident certificato Red Hat dall'hub degli operatori e personalizzazione dello stesso. 3.Utilizzo del timone.

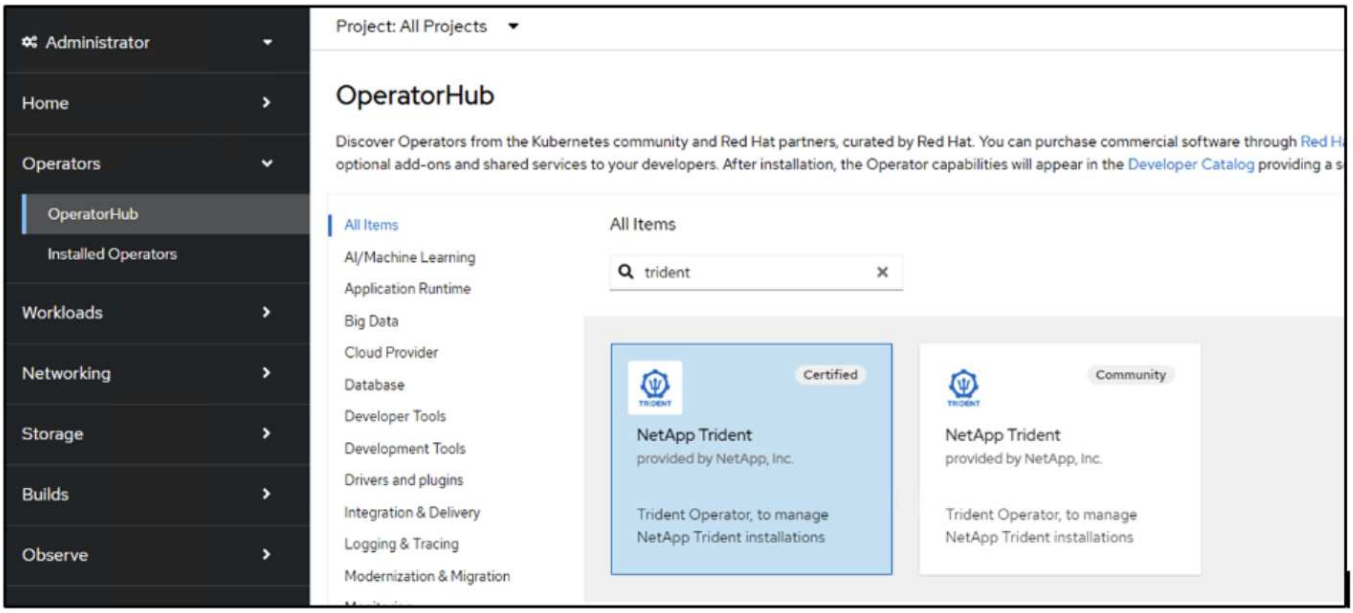
Utilizzando uno qualsiasi dei metodi sopra indicati senza abilitare node-prep sarà possibile utilizzare solo i protocolli NAS per il provisioning dello storage su FSxN.

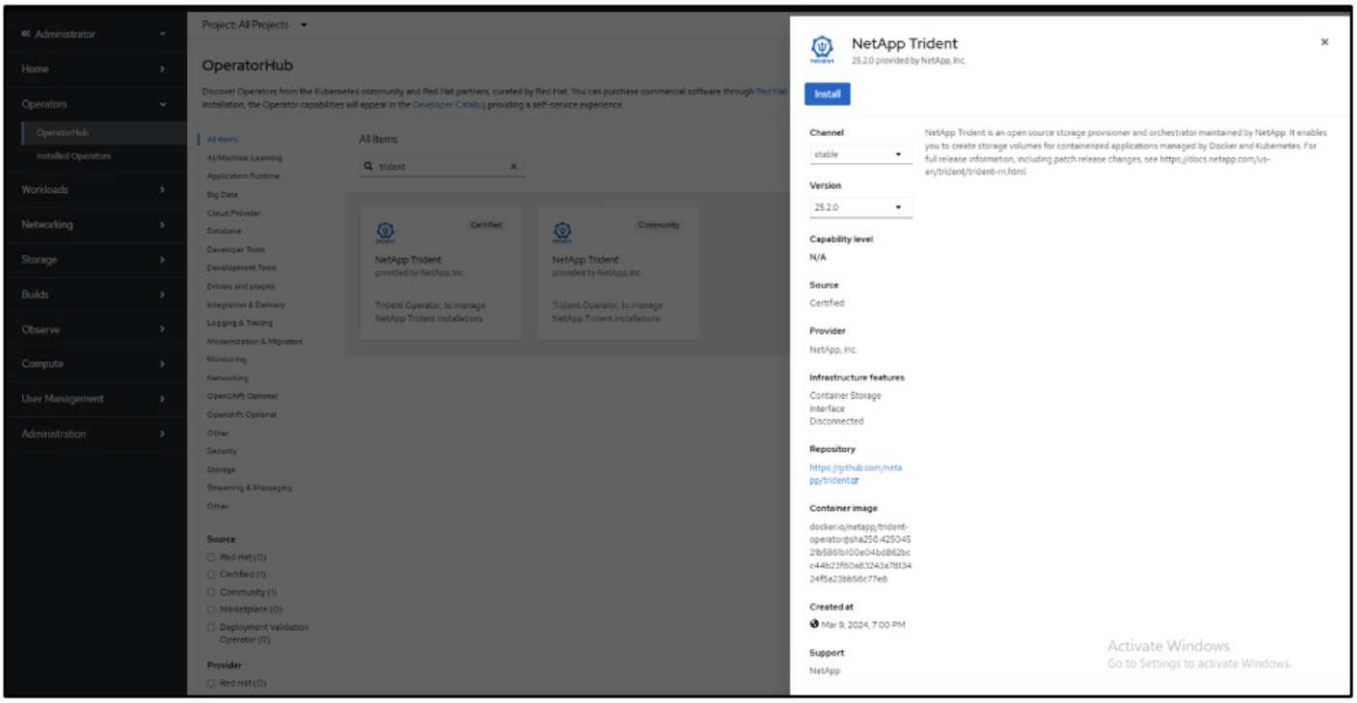
Metodo 1: utilizzare lo strumento tridentctl

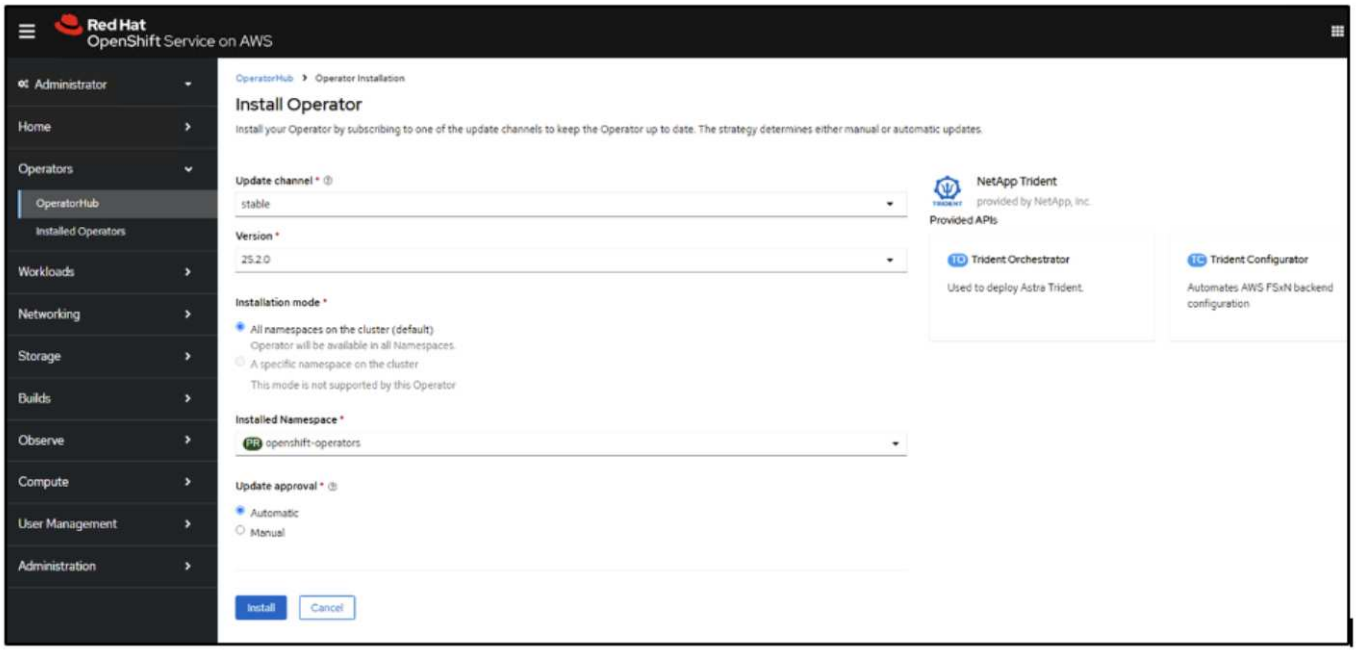
Utilizzare il flag node-prep e installare Trident come mostrato. Prima di inviare il comando di installazione, è necessario aver scaricato il pacchetto di installazione. Fare riferimento a "la documentazione qui" .

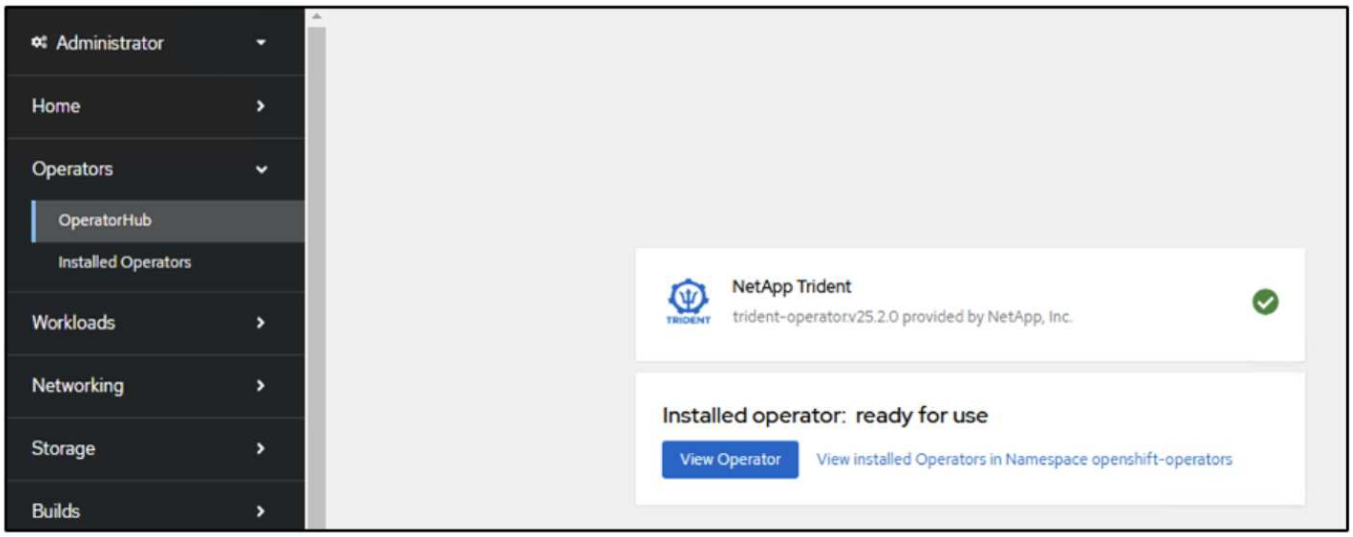
```
#./tridentctl install trident -n trident --node-prep=iscsi
```

Metodo 2: utilizzare l'operatore Trident certificato Red Hat e personalizzarlo Da OperatorHub, individuare l'operatore Trident certificato Red Hat e installarlo.

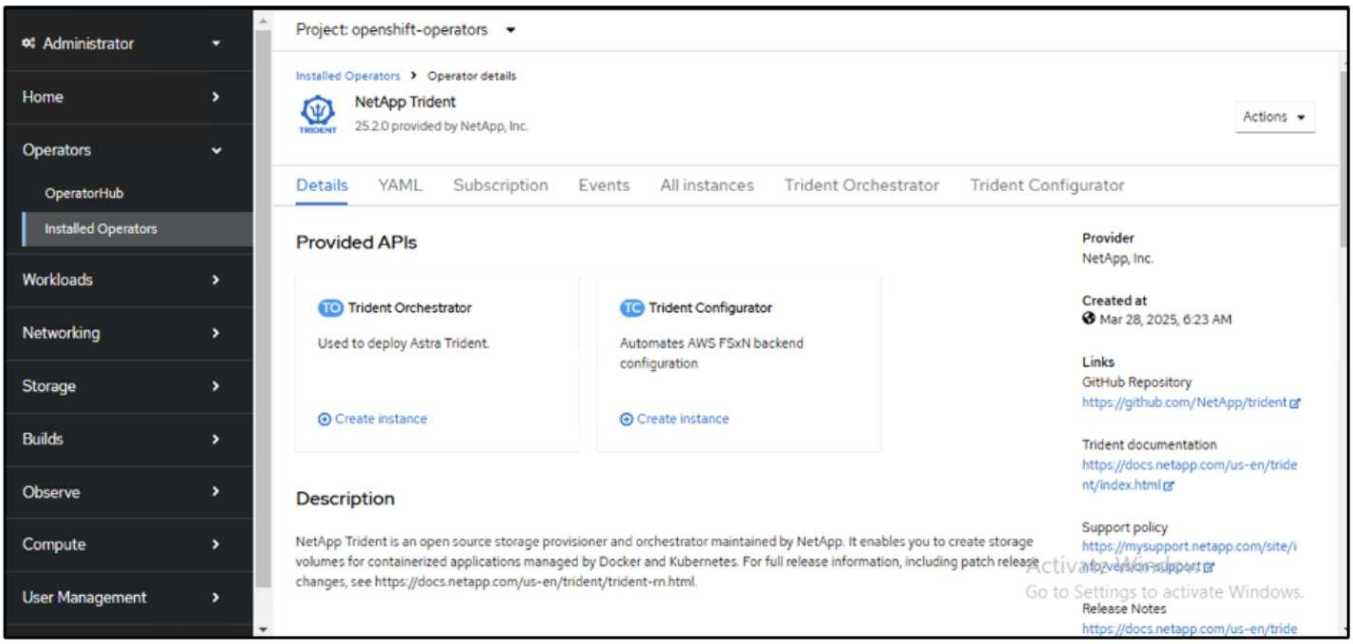


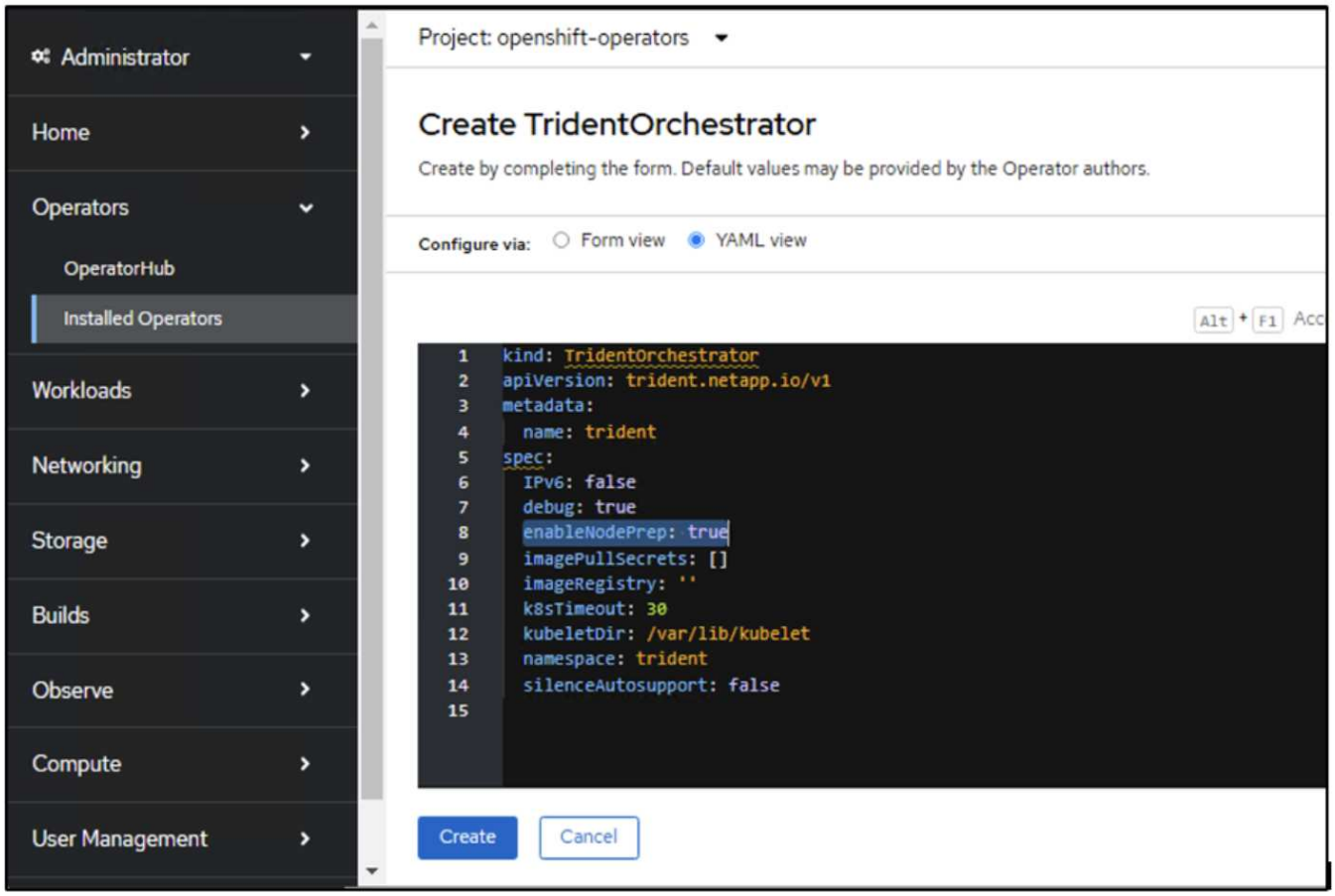

Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
Project: All Projects
OperatorHub
All Items
trident
NetApp Trident
provided by NetApp, Inc.
Trident Operator, to manage NetApp Trident installations
Certified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

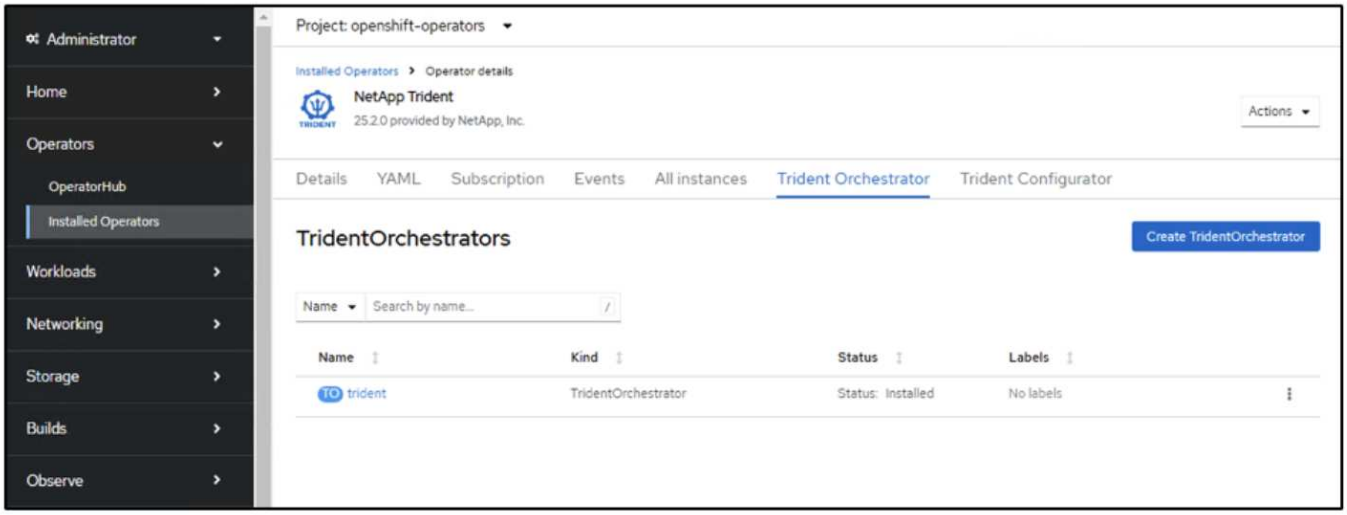

Red Hat
OpenShift Service on AWS
Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
Administration
OperatorHub > Operator Installation
Install Operator
Install your Operator by subscribing to one of the update channels to keep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



Successivamente, crea l'istanza di Trident Orchestrator. Utilizzare la visualizzazione YAML per impostare valori personalizzati o abilitare la preparazione del nodo iscsi durante l'installazione.








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

L’installazione Trident utilizzando uno qualsiasi dei metodi sopra indicati preparerà i nodi worker del cluster ROSA per iSCSI avviando i servizi iscsid e multipathd e impostando quanto segue nel file /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.Verificare che tutti i pod Trident siano in funzione

```
[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]#
```

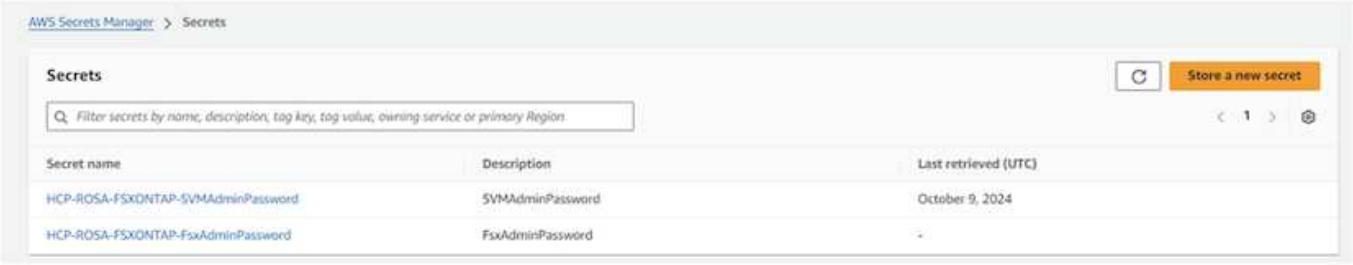
3. Configurare il backend Trident CSI per utilizzare FSx per ONTAP (ONTAP NAS)

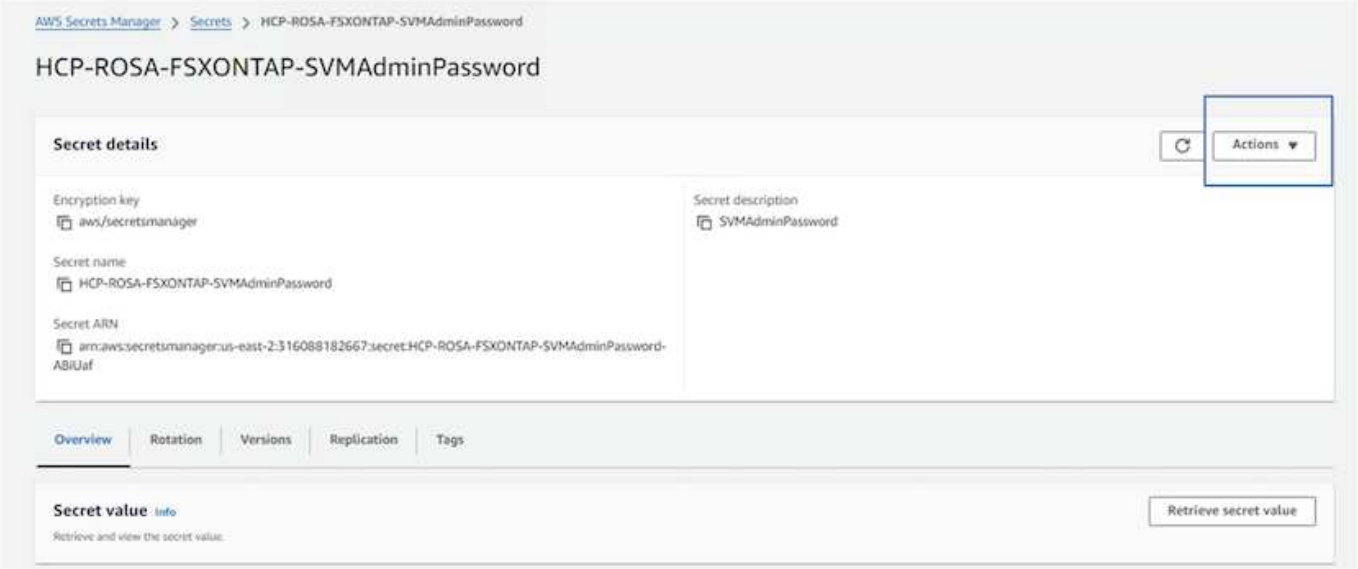
La configurazione back-end Trident indica a Trident come comunicare con il sistema di archiviazione (in questo caso, FSx per ONTAP). Per creare il backend, forniremo le credenziali della macchina virtuale di archiviazione a cui connettersi, insieme alla gestione del cluster e alle interfacce dati NFS. Useremo il"driver ontap-nas" per fornire volumi di archiviazione nel file system FSx.

UN. Per prima cosa, crea un segreto per le credenziali SVM utilizzando il seguente 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>
```

È anche possibile recuperare la password SVM creata per FSxN da AWS Secrets Manager, come mostrato di seguito.





b.Successivamente, aggiungi il segreto per le credenziali SVM al cluster ROSA utilizzando il seguente comando

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

Puoi verificare che il segreto sia stato aggiunto nello spazio dei nomi trident utilizzando il seguente comando

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

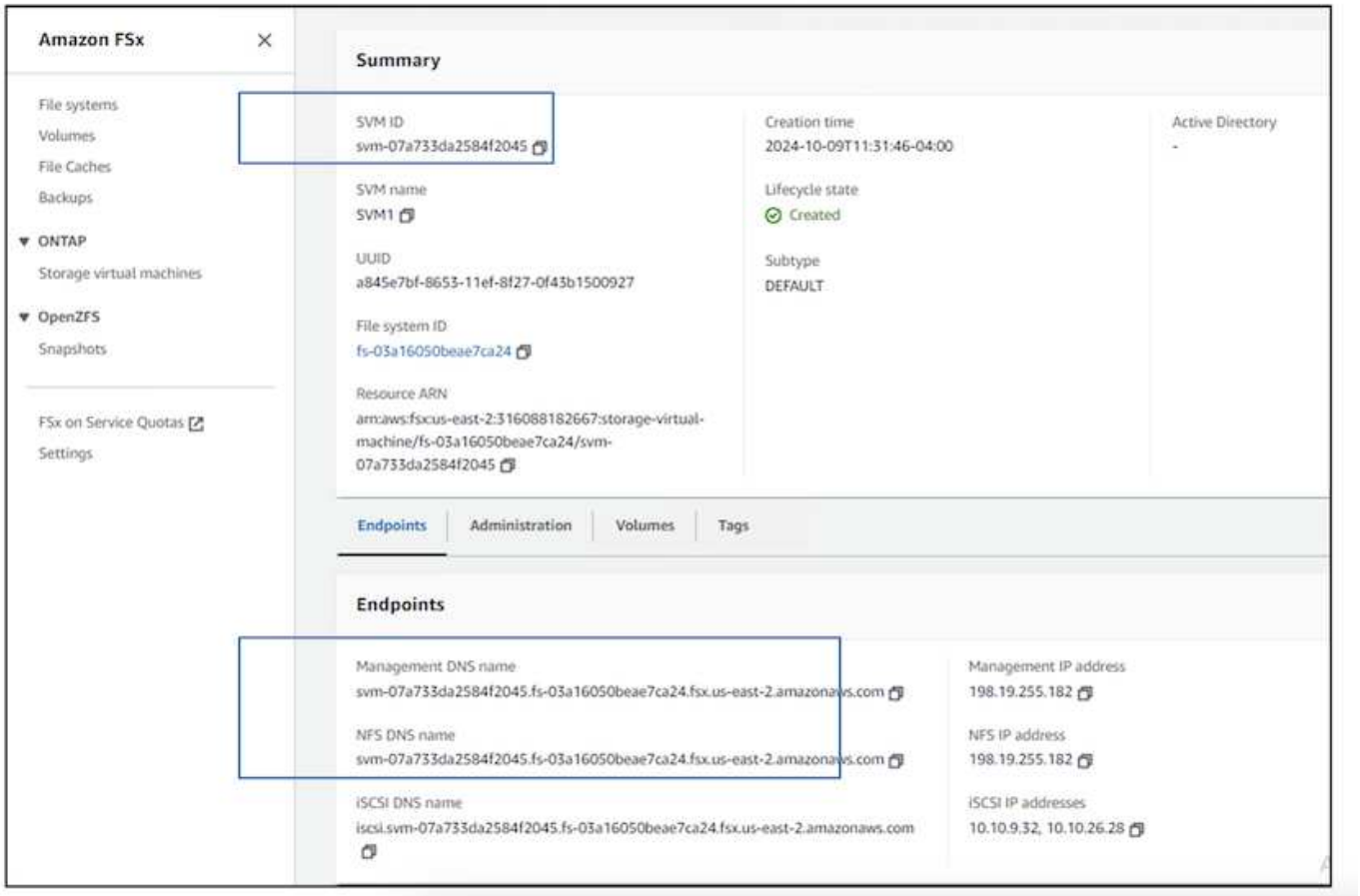
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

C. Successivamente, crea l'oggetto backend. Per farlo, spostati nella directory **fsx** del tuo repository Git clonato. Aprire il file backend-ontap-nas.yaml. Sostituire quanto segue: **managementLIF** con il nome DNS di gestione, **dataLIF** con il nome DNS NFS dell'SVM Amazon FSx e **svm** con il nome dell'SVM. Creare l'oggetto backend utilizzando il seguente comando.

Creare l'oggetto backend utilizzando il seguente comando.

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

È possibile ottenere il nome DNS di gestione, il nome DNS NFS e il nome SVM dalla console Amazon FSx come mostrato nello screenshot qui sotto



D. Ora, esegui il seguente comando per verificare che l'oggetto backend sia stato creato e che Phase mostri Bound e Status sia 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. Crea classe di archiviazione Ora che il backend Trident è configurato, puoi creare una classe di archiviazione Kubernetes per utilizzare il backend. La classe di archiviazione è un oggetto risorsa reso disponibile al cluster. Descrive e classifica il tipo di archiviazione che è possibile richiedere per un'applicazione.

UN. Esaminare il file storage-class-csi-nas.yaml nella cartella 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. Creare una classe di archiviazione nel cluster ROSA e verificare che la classe di archiviazione trident-csi sia stata creata.

```
[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]#
```

Questo completa l'installazione del driver Trident CSI e la sua connettività al file system ONTAP . Ora è possibile distribuire un'applicazione Postgresql con stato di esempio su ROSA utilizzando volumi di file su FSx per ONTAP.

C. Verificare che non siano stati creati PVC e PV utilizzando la classe di archiviazione trident-csi.

D. Verificare che le applicazioni possano creare PV utilizzando Trident CSI.

Creare un PVC utilizzando il file pvc-trident.yaml fornito nella cartella **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["creare PVC di prova
utilizzando Trident"]
```

Per utilizzare iSCSI, è necessario abilitare iSCSI sui nodi worker come mostrato in precedenza e creare un backend iSCSI e una classe di archiviazione. Ecco alcuni esempi di file yaml.

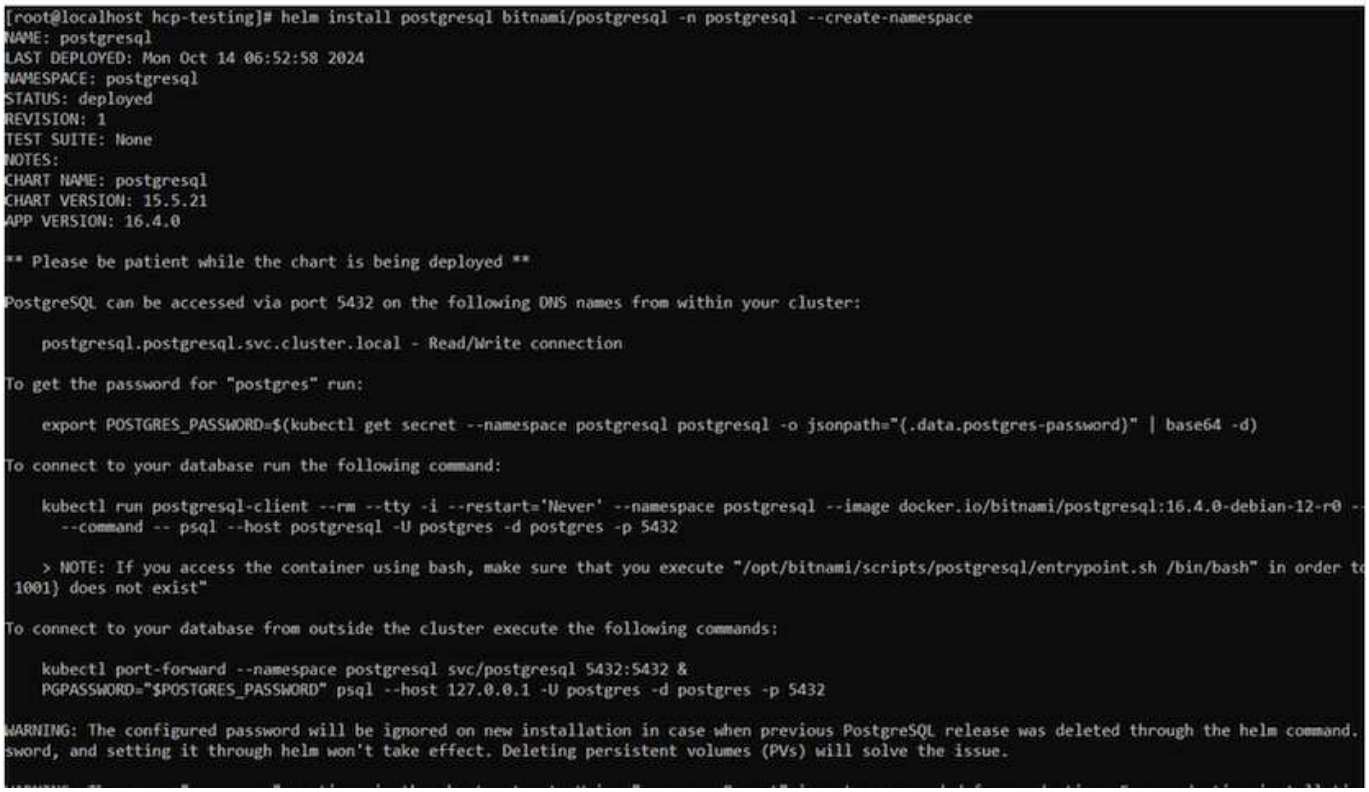
```
cat tbc.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: fsxadmin
  password: <password for the fsxN filesystem>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 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. Distribuisci un'applicazione Postgresql con stato di esempio

UN. Utilizzare helm per installare postgresql

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



B. Verificare che il pod dell'applicazione sia in esecuzione e che siano stati creati un PVC e un PV per l'applicazione.

C. Distribuisci un client Postgresql

Utilizzare il seguente comando per ottenere la password per il server PostgreSQL installato.

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

Utilizzare il seguente comando per eseguire un client PostgreSQL e connettersi al server utilizzando la 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/bitna
$POSTGRES_PASSWORD" \
> --command -- psql --host postgresql -U postgres -d postgres -p 5432
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityC
 capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "
Root=true), seccompProfile (pod or container "postgresql-client" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Loca
If you don't see a command prompt, try pressing enter.
```

D. Crea un database e una tabella. Crea uno schema per la tabella e inserisci 2 righe di dati nella tabella.

```
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)
```

Servizio Red Hat OpenShift su AWS con NetApp ONTAP

Questo documento descriverà come utilizzare NetApp ONTAP con Red Hat OpenShift Service on AWS (ROSA).

Crea snapshot del volume

1. Creare uno snapshot del volume dell'app In questa sezione mostreremo come creare uno snapshot Trident del volume associato all'app. Si tratterà di una copia puntuale dei dati dell'app. Se i dati dell'applicazione vengono persi, possiamo recuperarli da questa copia in un dato momento. NOTA: questo snapshot viene archiviato nello stesso aggregato del volume originale in ONTAP(in locale o nel cloud). Pertanto, se l'aggregato di archiviazione ONTAP viene perso, non è possibile recuperare i dati dell'app dal relativo snapshot.

**UN. Crea un VolumeSnapshotClass Salva il seguente manifesto in un file denominato volume-snapshot-class.yaml

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

Crea uno snapshot utilizzando il manifesto sopra.

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

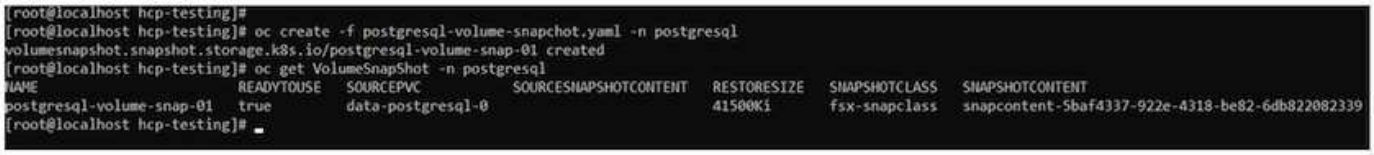
B. Successivamente, crea uno snapshot Crea uno snapshot del PVC esistente creando VolumeSnapshot per acquisire una copia puntuale dei tuoi dati PostgreSQL. In questo modo viene creato uno snapshot FSx che non occupa quasi spazio nel backend del file system. Salvare il seguente manifesto in un file denominato

volume-snapshot.yaml:

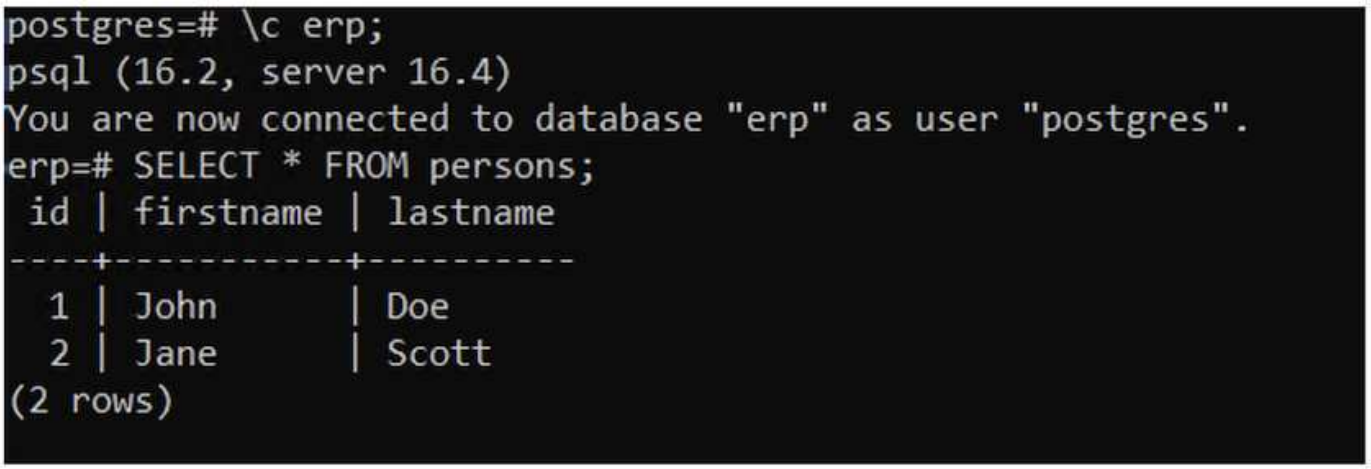
```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
 name: postgresql-volume-snap-01
spec:
 volumeSnapshotClassName: fsx-snapclass
 source:
   persistentVolumeClaimName: data-postgresql-0
```

C. Crea lo snapshot del volume e conferma che è stato creato

Eliminare il database per simulare la perdita di dati (la perdita di dati può verificarsi per diversi motivi, qui la simuliamo semplicemente eliminando il database)



D. Eliminare il database per simulare la perdita di dati (la perdita di dati può verificarsi per diversi motivi, qui la simuliamo semplicemente eliminando il database)



Ripristina da snapshot del volume

1. Ripristino da snapshot In questa sezione mostreremo come ripristinare un'applicazione dallo snapshot Trident del volume dell'app.

UN. Crea un clone del volume dallo snapshot

Per ripristinare il volume allo stato precedente, è necessario creare un nuovo PVC basato sui dati presenti nello snapshot acquisito. Per fare ciò, salva il seguente manifesto in un file denominato 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
```

Creare un clone del volume creando un PVC utilizzando lo snapshot come origine utilizzando il manifesto sopra. Applicare il manifesto e assicurarsi che il clone venga creato.

```
[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. Elimina l'installazione originale di 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. Crea una nuova applicazione postgresql utilizzando il nuovo 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}" | bas

To connect to your database run the following command:

    kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:16
      --command -- psql --host postgresql -U postgres -d postgres -p 5432

    > NOTE: If you access the container using bash, make sure that you execute "/opt/bitnami/scripts/postgresql/entrypoint.sh /bi
 1001} does not exist"

To connect to your database from outside the cluster execute the following commands:

    kubectl port-forward --namespace postgresql svc/postgresql 5432:5432 &
    PGPASSWORD="$POSTGRES_PASSWORD" psql --host 127.0.0.1 -U postgres -d postgres -p 5432

WARNING: The configured password will be ignored on new installation in case when previous PostgreSQL release was deleted through
sword, and setting it through helm won't take effect. Deleting persistent volumes (PVs) will solve the issue.

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

D. Verificare che il pod dell'applicazione sia in esecuzione

```
[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. Verificare che il pod utilizzi il clone come PVC

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

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

f) Per convalidare che il database sia stato ripristinato come previsto, tornare alla console del contenitore e visualizzare i database esistenti

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:
$POSTGRES_PASSWORD"  --command -- psql --host postgresql -U postgres -d postgres -p 5432
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityContext.allowPri
 capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "postgresql-clie
Root=true), seccompProfile (pod or container "postgresql-client" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
If you don't see a command prompt, try pressing enter.

postgres=# \l
                                                List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+-----------------------
 erp       | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
(4 rows)

postgres=# \c erp;
psql (16.2, server 16.4)
You are now connected to database "erp" as user "postgres".
erp=# \dt
        List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persons | table | postgres
(1 row)

erp=# SELECT * FROM PERSONS;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
  2 | Jane      | Scott
(2 rows)
```

Video dimostrativo

Amazon FSx for NetApp ONTAP con Red Hat OpenShift Service su AWS utilizzando Hosted Control Plane

Sono disponibili altri video su Red Hat OpenShift e sulle soluzioni OpenShift"Qui" .