使用 FSxN 在 AWS 上提供 Red Hat OpenShift 服務

NetApp container solutions

NetApp
January 21, 2026

目錄

使用 FSxN 在 AWS 上提供 Red Hat OpenShift 服務

使用NetApp ONTAP在 AWS 上提供 Red Hat OpenShift 服務

概況

在本節中，我們將展示如何利用 FSx for ONTAP作為在 ROSA 上運行的應用程式的持久性儲存層。它將展示在 ROSA 叢集上安裝NetApp Trident CSI 驅動程式、為ONTAP檔案系統配置 FSx 以及部署範例有狀態應用程式。它還將顯示備份和恢復應用程式資料的策略。透過此整合解決方案，您可以建立一個可輕鬆跨可用區擴展的共享儲存框架，從而簡化使用Trident CSI 驅動程式擴充、保護和復原資料的流程。

先決條件

- "AWS 帳號"
- "Red Hat 帳號"
- IAM 用戶"具有適當的權限"建立並存取ROSA集群
- "AWS CLI"
- "ROSA CLI"
- "OpenShift 命令列介面"（oc）
- Helm 3"文件"
- "HCP ROSA 集群"
- "造訪 Red Hat OpenShift Web 控制台"

此圖顯示了部署在多個 AZ 中的 ROSA 集群。 ROSA叢集的主節點、基礎設施節點位於Red Hat的VPC中，而工作節點位於客戶帳戶中的VPC。我們將在同一個 VPC 內建立一個 FSx for ONTAP檔案系統，並在 ROSA 叢集中安裝Trident驅動程序，以允許該 VPC 的所有子網路連接到該檔案系統。

初始設定

1.為NetApp ONTAP配置 FSx

在與 ROSA 叢集相同的 VPC 中為NetApp ONTAP建立多可用區 FSx。有幾種方法可以做到這一點。提供了使用 CloudFormation Stack 建立 FSxN 的詳細信息

a.克隆 **GitHub** 儲存庫

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

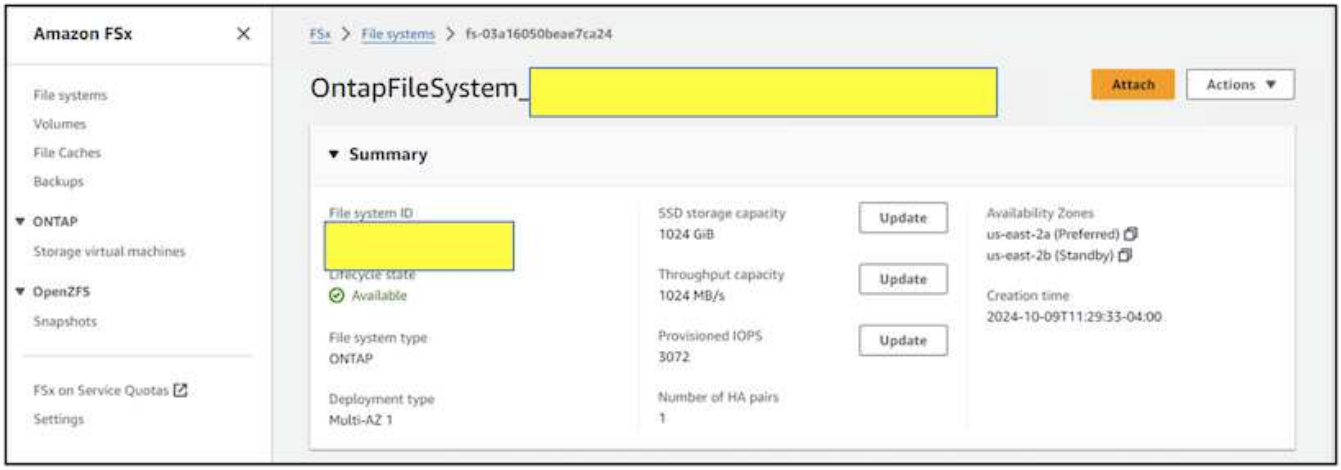
b.執行 **CloudFormation** 堆疊 透過將參數值替換為您自己的值來執行以下命令：

```
$ 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
```

其中： region-name：與部署 ROSA 群集的區域相同 subnet1_ID：FSxN 的首選子網的 id subnet2_ID：FSxN 的備用子網的 id VPC_ID：部署 ROSA 群集的 VPC 的 id routetable1_ID、routetable2_our:CI 叢集的 VPC 的路由表 編號ONTAP安全群組入口規則允許的 CIDR 範圍，用於控制存取。您可以使用 0.0.0.0/0 或任何適當的 CIDR 來允許所有流量存取 FSx for ONTAP的特定連接埠。定義管理者密碼：登入 FSxN 的密碼 定義 SVM 密碼：登入將要建立的 SVM 的密碼。

使用Amazon FSx控制台驗證您的檔案系統和儲存虛擬機器 (SVM) 是否已創建，如下所示：



2.為 **ROSA** 叢集安裝並設定**Trident CSI** 驅動程式

b.安裝**Trident**

ROSA 叢集工作節點預先配置了 nfs 工具，使您能夠使用 NAS 協定進行儲存配置和存取。

如果您想使用 iSCSI，則需要為 iSCSI 準備工作節點。從Trident 25.02 版本開始，您可以輕鬆準備 ROSA 叢集（或任何 OpenShift 叢集）的工作節點以在 FSxN 儲存體上執行 iSCSI 作業。有兩種簡單的方法可以安裝Trident 25.02（或更高版本），以自動為 iSCSI 準備工作節點。 1. 使用 tridentctl 工具從命令列使用 node-prep-flag。2.使用操作員中心的 Red Hat 認證的Trident操作員並對其進行自訂。 3.使用 Helm。

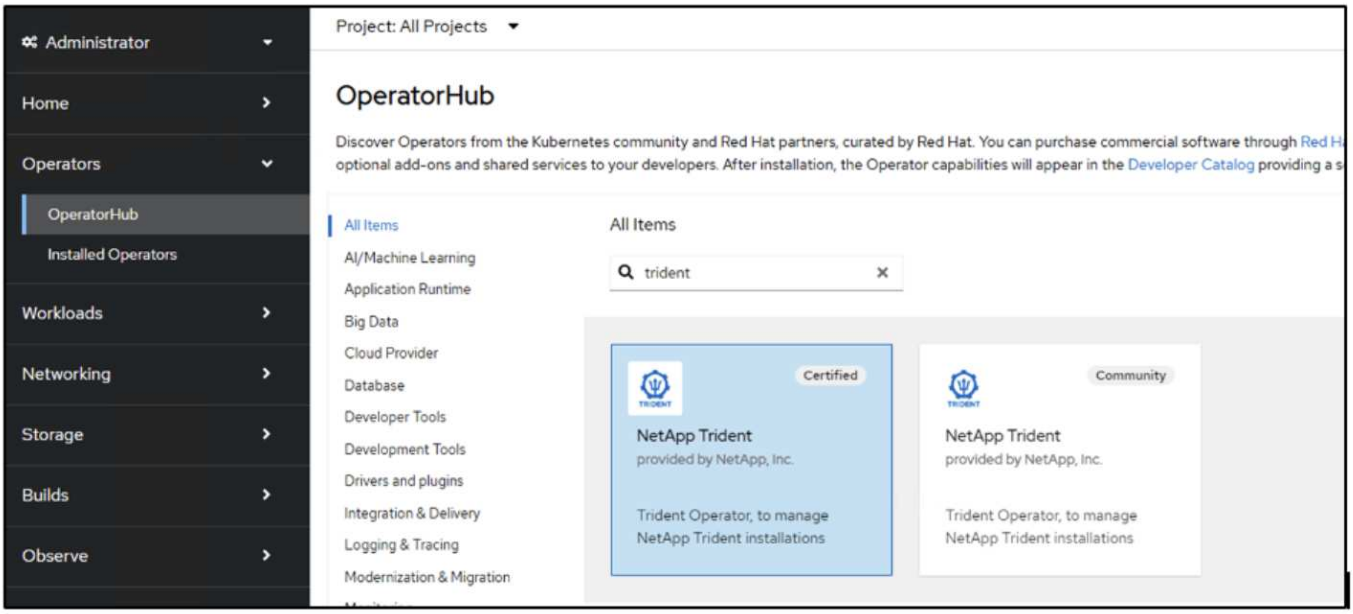
使用上述任何一種方法而不啟用節點準備將允許您僅使用 NAS 協定在 FSxN 上配置儲存。

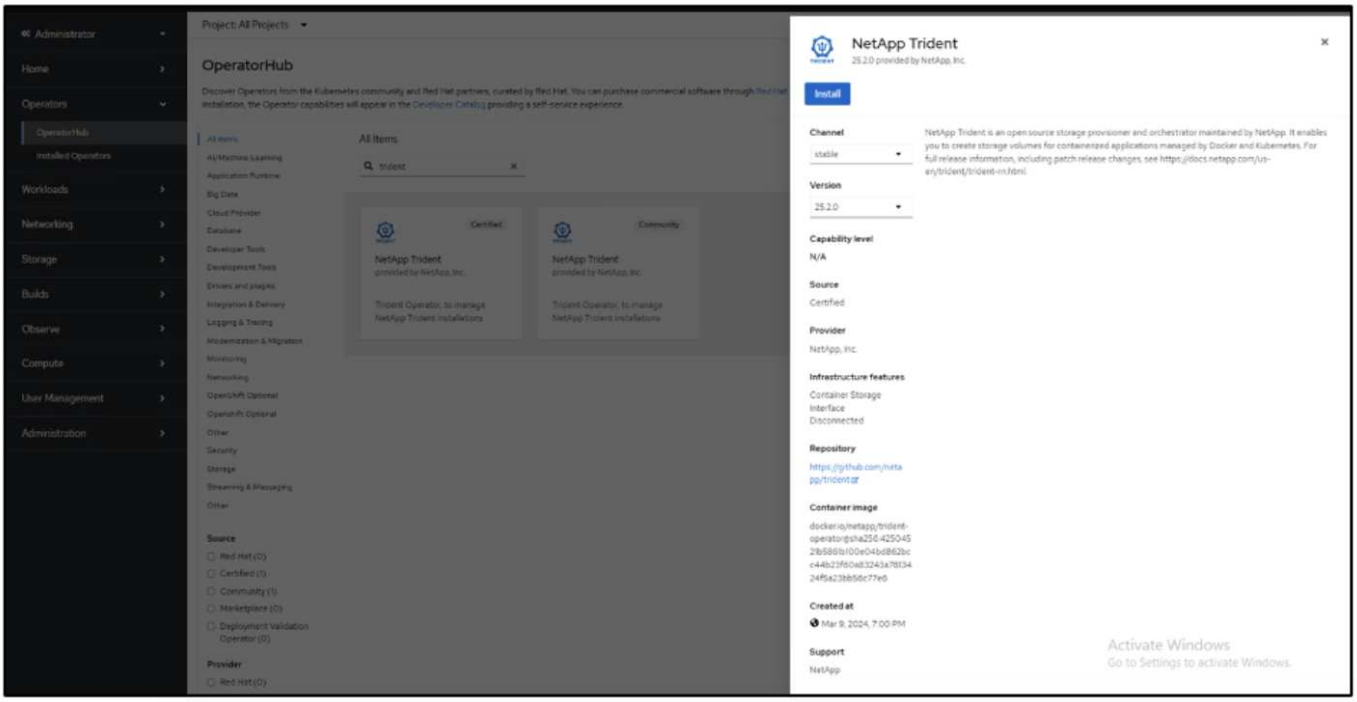
方法一：使用 **tridentctl** 工具

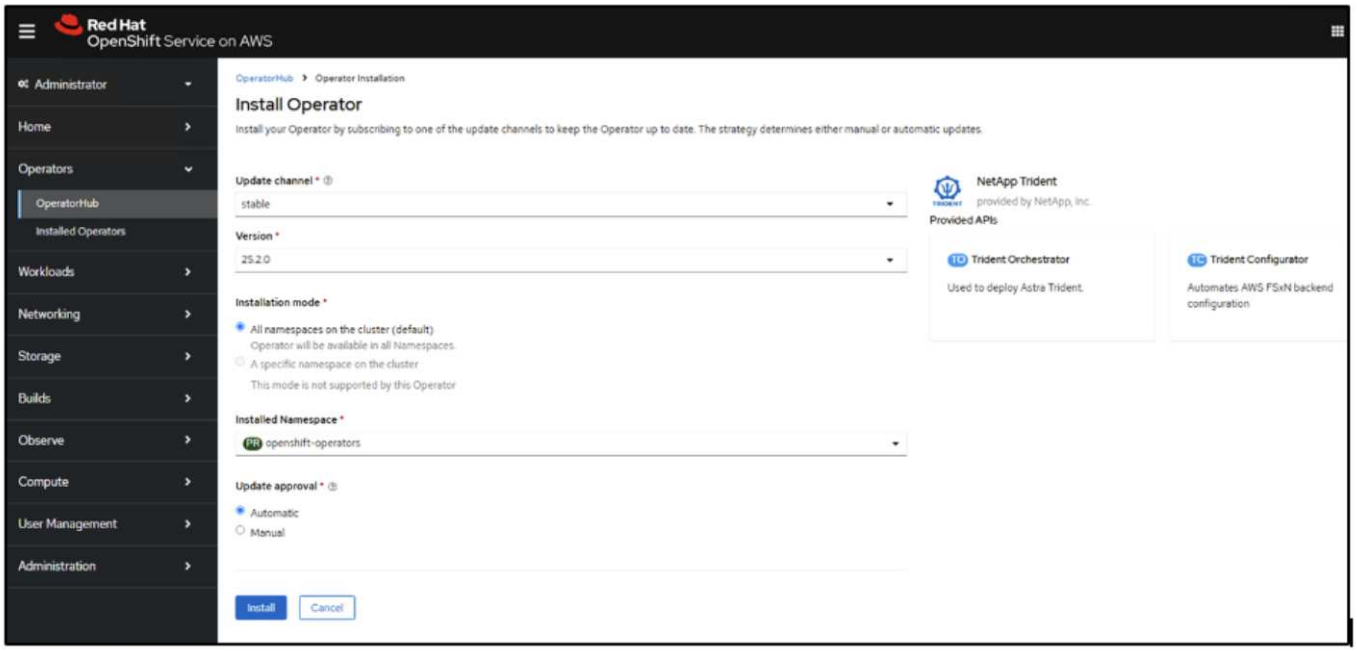
使用 node-prep 標誌並安裝Trident ，如圖所示。在發出安裝命令之前，您應該已經下載了安裝程式包。請參閱"此處的文檔" 。

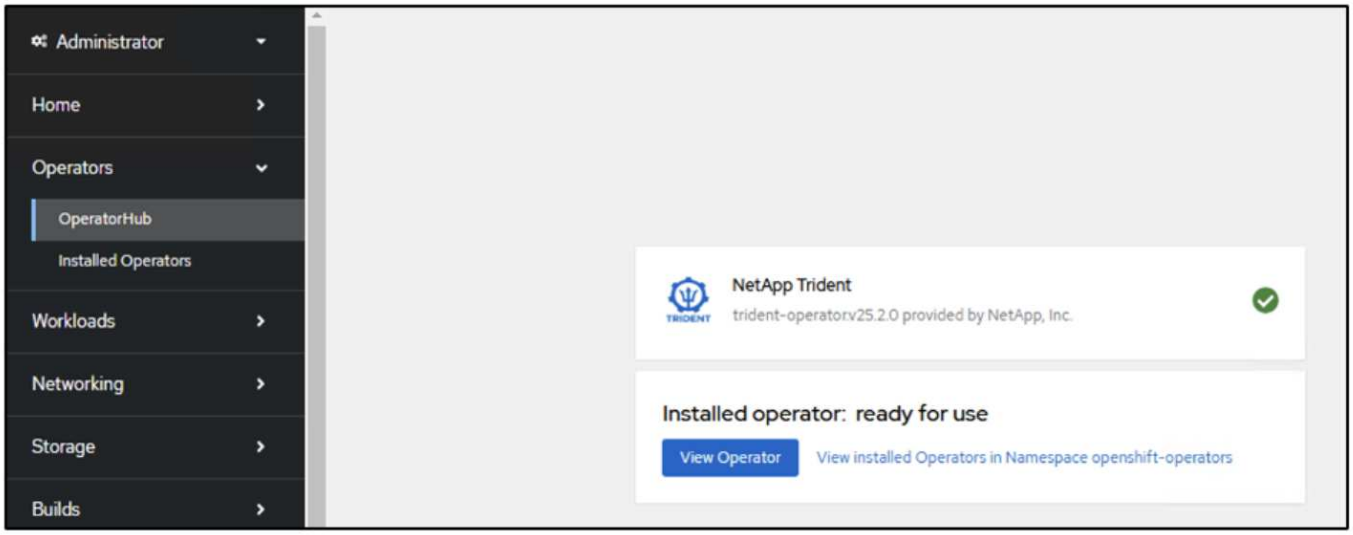
```
#./tridentctl install trident -n trident --node-prep=iscsi
```

方法 **2**：使用 **Red Hat** 認證的**Trident Operator** 並自訂 從 OperatorHub 找到 Red Hat 認證的Trident Operator 並安裝它。

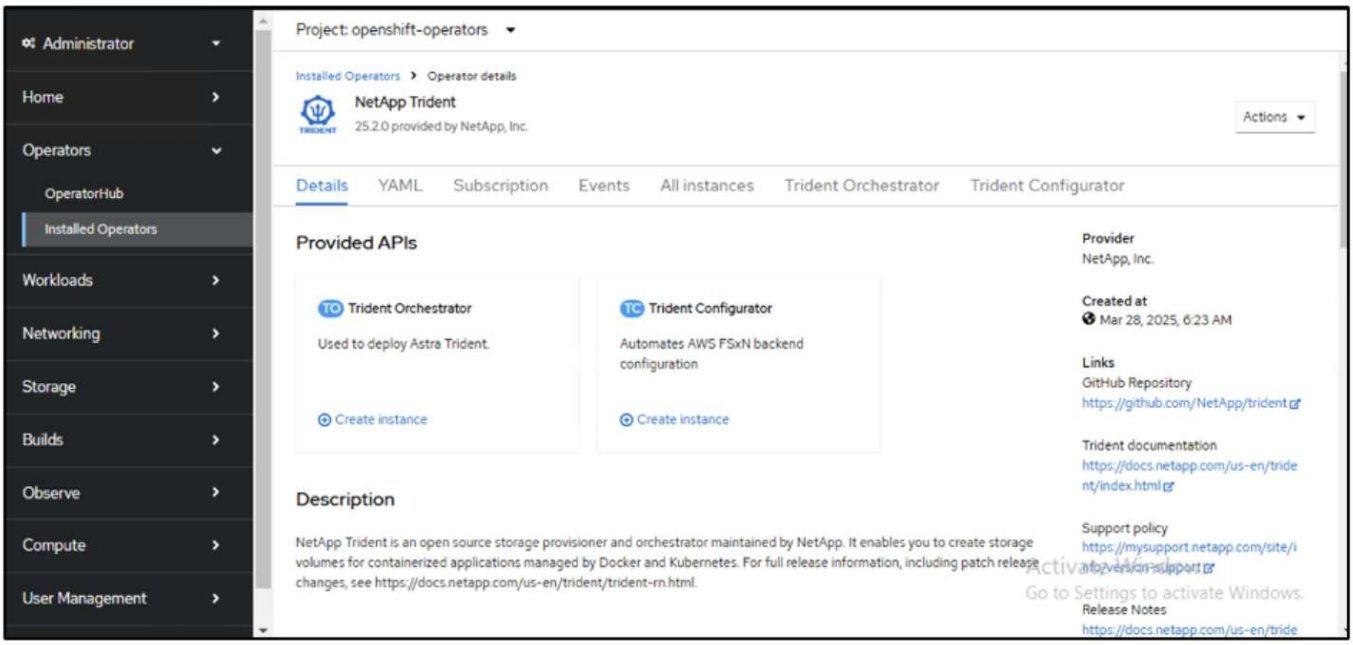

Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Project: All Projects
OperatorHub
Discover Operators from the Kubernetes community and Red Hat partners, curated by Red Hat. You can purchase commercial software through Red H
optional add-ons and shared services to your developers. After installation, the Operator capabilities will appear in the Developer Catalog providing a s
All Items
AI/Machine Learning
Application Runtime
Big Data
Cloud Provider
Database
Developer Tools
Development Tools
Drivers and plugins
Integration & Delivery
Logging & Tracing
Modernization & Migration
All Items
trident
Certified
NetApp Trident
provided by NetApp, Inc.
Trident Operator, to manage NetApp Trident installations
Community
NetApp Trident
provided by NetApp, Inc.
Trident Operator, to manage NetApp Trident installations

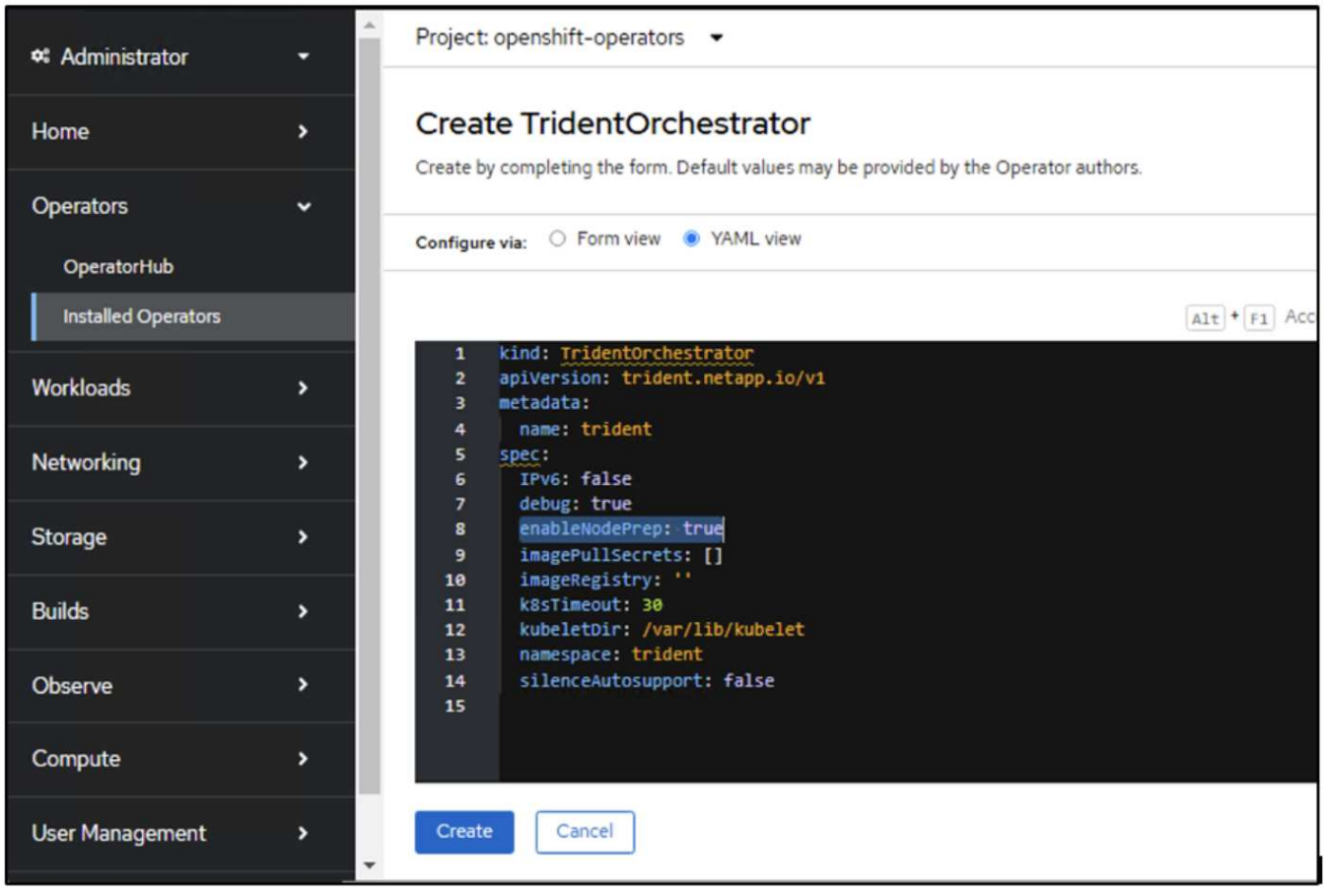

NetApp Trident
Install
Channel
stable
Version
25.2.0
Capability level
N/A
Source
Certified
Provider
NetApp, Inc.
Infrastructure features
Container Storage Interface
Disconnected
Repository
Container image
Created at
Mar 9, 2024, 7:00 PM
Support
NetApp
Activate Windows

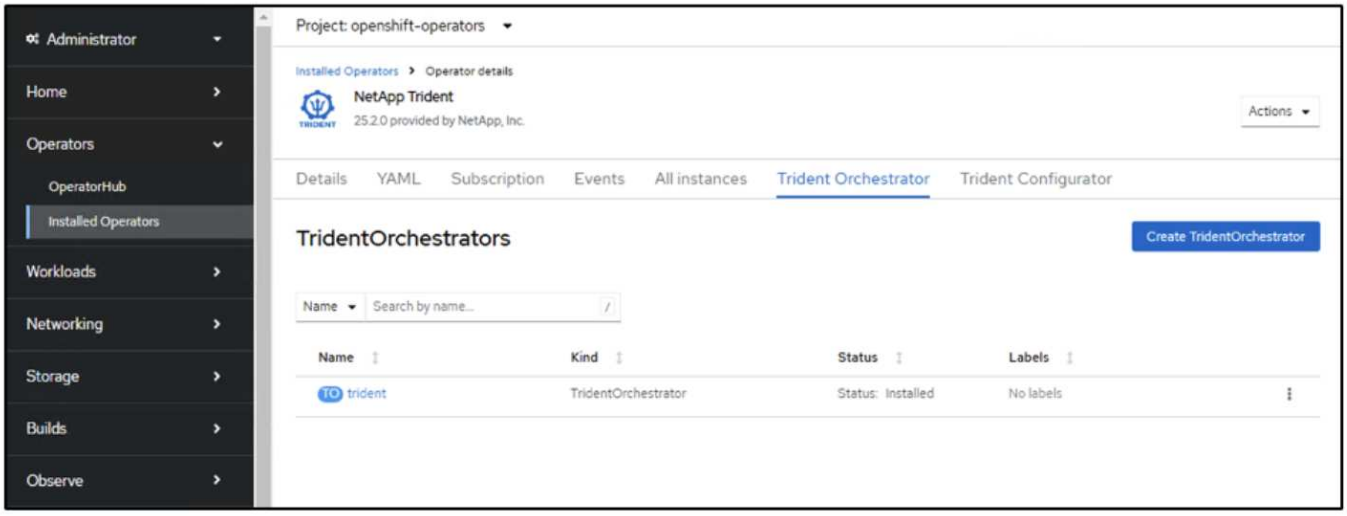




接下來，建立Trident Orchestrator 實例。使用 YAML 視圖設定任何自訂值或在安裝期間啟用 iscsi 節點準備。


Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
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
Provided APIs
Trident Orchestrator
Used to deploy Astra Trident.
Create instance
Trident Configurator
Automates AWS FSxN backend configuration
Create instance
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
Release Notes
Description
NetApp Trident is an open source storage provisioner and orchestrator maintained by NetApp. It enables you to create storage volumes for containerized applications managed by Docker and Kubernetes. For full release information, including patch release changes, see https://docs.netapp.com/us-en/trident/trident-rn.html.


Administrator
Home
Operators
OperatorHub
Installed Operators
Workloads
Networking
Storage
Builds
Observe
Compute
User Management
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
Alt + F1
kind: TridentOrchestrator
apiVersion: trident.netapp.io/v1
metadata:
name: trident
spec:
IPv6: false
debug: true
enableNodePrep: true
imagePullSecrets: []
imageRegistry: ''
k8sTimeout: 30
kubeletDir: /var/lib/kubelet
namespace: trident
silenceAutosupport: false
Create
Cancel




```
[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]#
```

使用上述任何一種方法安裝Trident將透過啟動 iscsid 和 multipathd 服務並在 /etc/multipath.conf 檔案中設定以下內容，為 iSCSI 準備 ROSA 叢集工作節點

```
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.驗證所有**Trident pod** 是否處於運作狀態

```
[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]#
```

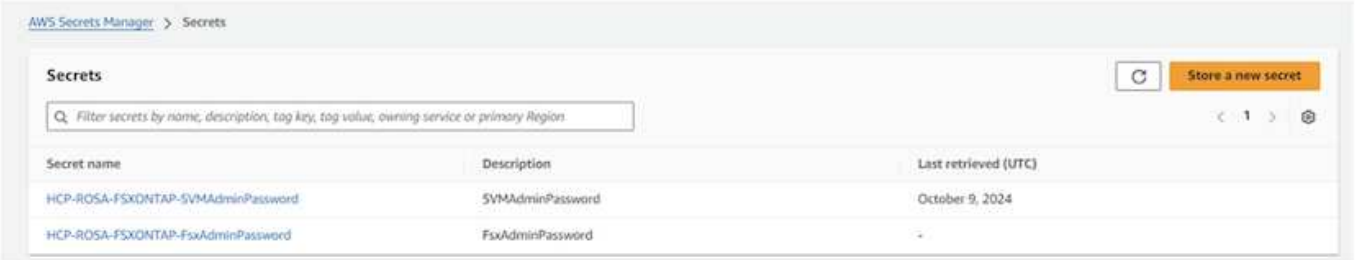
3.設定Trident CSI 後端以使用 FSx for ONTAP (ONTAP NAS)

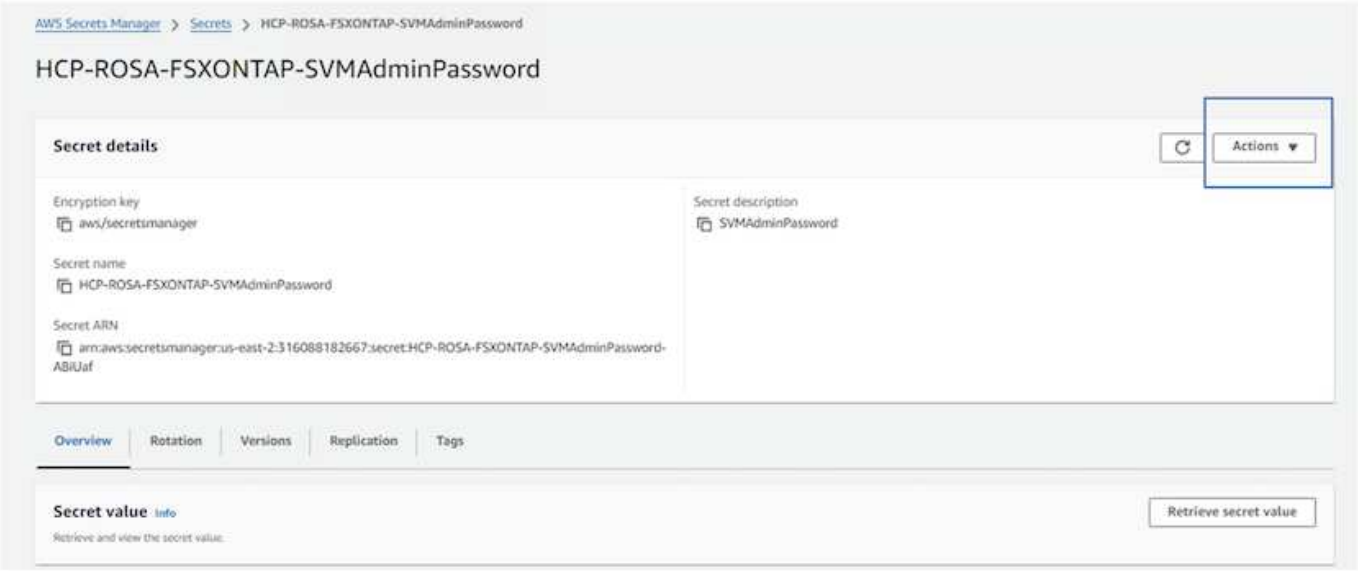
Trident後端設定告訴Trident如何與儲存系統（在本例中為 FSx for ONTAP）通訊。為了建立後端，我們將提供要連接的儲存虛擬機器的憑證，以及叢集管理和 NFS 資料介面。我們將使用"ontap-nas驅動程式"在 FSx 檔案系統中設定儲存磁碟區。

一個。首先，使用以下 **yaml** 為 SVM 憑證建立金鑰

```
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>
```

您也可以從 AWS Secrets Manager 檢索為 FSxN 建立的 SVM 密碼，如下所示。





b.接下來，使用以下命令將 **SVM** 憑證的金鑰新增至 **ROSA** 叢集

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

您可以使用下列指令驗證該秘密是否已新增至 trident 命名空間中

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

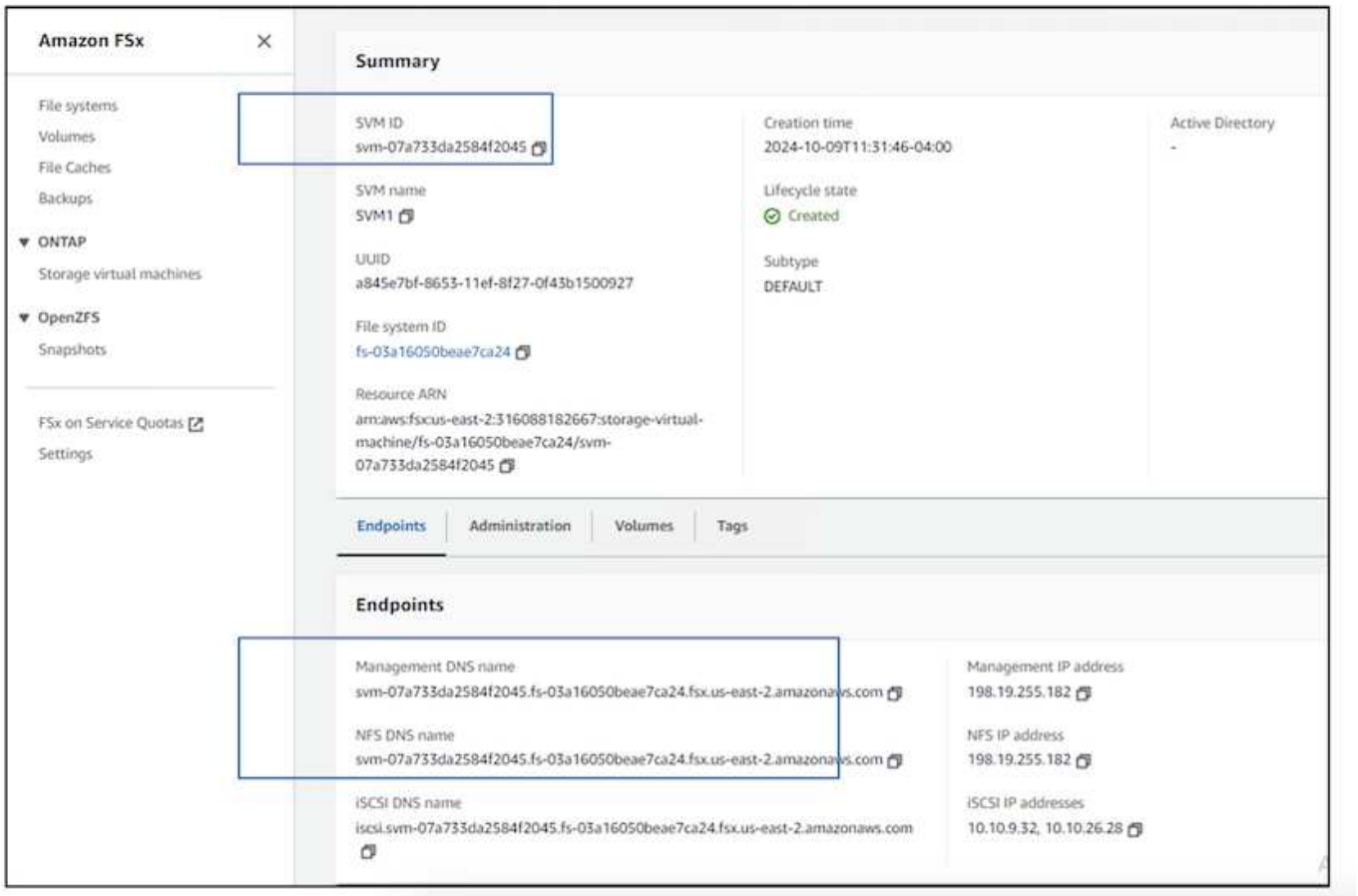
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

c.接下來，建立後端物件為此，請進入複製的 Git 儲存庫的 **fsx** 目錄。開啟檔案 backend-ontap-nas.yaml。取代以下內容：將 **managementLIF** 替換為管理 DNS 名稱，將 **dataLIF** 替換為Amazon FSx SVM 的 NFS DNS 名稱，將 **svm** 替換為 SVM 名稱。使用以下命令建立後端物件。

使用以下命令建立後端物件。

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

您可以從Amazon FSx控制台取得管理 DNS 名稱、NFS DNS 名稱和 SVM 名稱，如下面的螢幕截圖所示



d.現在，執行以下命令來驗證後端物件是否已創建，並且階段顯示為綁定且狀態為成功

```
[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.建立儲存類別 現在已經設定了Trident後端，您可以建立一個 Kubernetes 儲存類別來使用該後端。儲存類別是叢集可用的資源物件。它描述並分類了您可以為應用程式請求的儲存類型。

一個。查看 **fsx** 資料夾中的檔案 **storage-class-csi-nas.yaml**。

```
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.在 **ROSA** 叢集中建立儲存類別並驗證 **trident-csi** 儲存類別是否已建立。

```
[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]#
```

這完成了Trident CSI 驅動程式的安裝及其與 FSx for ONTAP檔案系統的連線。現在，您可以使用 FSx for ONTAP上的檔案磁碟區在 ROSA 上部署範例 Postgresql 有狀態應用程式。

c.驗證沒有使用 **trident-csi** 儲存類別建立的 **PVC** 和 **PV**。

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get pvc -A
NAMESPACE                       NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
openshift-monitoring            prometheus-data-prometheus-k8s-0    Bound    pvc-9a4553a5-07e9-440a-8a90-99e384c97624   100Gi      RWO            gp3-csi        <unset>                 2d16h
openshift-monitoring            prometheus-data-prometheus-k8s-1    Bound    pvc-7d949aef-e00d-4d9a-8b54-514ed85fbab2   100Gi      RWO            gp3-csi        <unset>                 2d16h
openshift-virtualization-os-images   centos-stream9-bae111cdd5a1    Bound    pvc-deb61444-cb3f-449b-8d7d-39d028496c16   30Gi       RWO            gp3-csi        <unset>                 24h
openshift-virtualization-os-images   centos-stream9-d82f4a141a44    Bound    pvc-82b0e84a-e5ef-452b-bf90-1eae4fe162c1   30Gi       RWO            gp3-csi        <unset>                 44h
openshift-virtualization-os-images   fedora-21a6f3e628cd            Bound    pvc-64f375ad-d377-456d-83a0-368e413ae79c   30Gi       RWO            gp3-csi        <unset>                 44h
openshift-virtualization-os-images   rhel8-0b52df0eb259             Bound    pvc-2dc6de48-5916-411e-9cb3-99598f50be4c   30Gi       RWO            gp3-csi        <unset>                 44h
openshift-virtualization-os-images   rhel9-2521bd116e64             Bound    pvc-f4374ce7-568d-4afc-b635-0228cf4544d4   30Gi       RWO            gp3-csi        <unset>                 44h
[root@localhost hcp-testing]# oc get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                                                          STORAGECLASS   VOLUMEATTRIBUTESCLASS
pvc-2dc6de48-5916-411e-9cb3-99598f50be4c   30Gi       RWO            Delete           Bound    openshift-virtualization-os-images/rhel8-0b52df0eb259          gp3-csi        <unset>
pvc-64f375ad-d377-456d-83a0-368e413ae79c   30Gi       RWO            Delete           Bound    openshift-virtualization-os-images/fedora-21a6f3e628cd         gp3-csi        <unset>
pvc-7d949aef-e00d-4d9a-8b54-514ed85fbab2   100Gi      RWO            Delete           Bound    openshift-monitoring/prometheus-data-prometheus-k8s-1          gp3-csi        <unset>
pvc-82b0e84a-e5ef-452b-bf90-1eae4fe162c1   30Gi       RWO            Delete           Bound    openshift-virtualization-os-images/centos-stream9-d82f4a141a44 gp3-csi        <unset>
pvc-9a4553a5-07e9-440a-8a90-99e384c97624   100Gi      RWO            Delete           Bound    openshift-monitoring/prometheus-data-prometheus-k8s-0          gp3-csi        <unset>
pvc-deb61444-cb3f-449b-8d7d-39d028496c16   30Gi       RWO            Delete           Bound    openshift-virtualization-os-images/centos-stream9-bae111cdd5a1 gp3-csi        <unset>
pvc-f4374ce7-568d-4afc-b635-0228cf4544d4   30Gi       RWO            Delete           Bound    openshift-virtualization-os-images/rhel9-2521bd116e64          gp3-csi        <unset>
[root@localhost hcp-testing]#
```

d.驗證應用程式是否可以使用**Trident CSI** 建立 **PV**。

使用 **fsx** 資料夾中提供的 pvc-trident.yaml 檔案建立 PVC。

```
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["使用Trident建立測試 PVC"]
```

要使用 iSCSI，您應該已經在工作節點上啟用了 iSCSI（如前所示），並且需要建立 iSCSI 後端和儲存類別。以下是一些範例 yaml 檔案。

```
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.部署範例 **Postgresql** 有狀態應用程式

一個。使用 **helm** 安裝 **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 postgresql -U postgres -d postgres -p 5432

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

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

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

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

b.驗證應用程式 **pod** 是否正在運行，以及是否為該應用程式建立了 **PVC** 和 **PV**。

```
[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.部署 **Postgresql** 客戶端

使用下列指令取得已安裝的 **postgresql** 伺服器的密碼。

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

使用以下命令運行 **postgresql** 用戶端並使用密碼連接到伺服器

```
$ 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.建立一個資料庫和一個表。為表建立一個模式，並在表中插入 **2** 行資料。

```
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)
```

使用NetApp ONTAP在 AWS 上提供 Red Hat OpenShift 服務

本文檔將概述如何將NetApp ONTAP與 AWS 上的 Red Hat OpenShift 服務 (ROSA) 結合使用。

建立磁碟區快照

1.建立應用程式磁碟區的快照 在本節中，我們將展示如何建立與應用程式關聯的磁碟區的 trident 快照。這將是應用程式資料的時間點副本。如果應用程式資料遺失，我們可以從該時間點的副本恢復資料。注意：此快照與ONTAP中的原始磁碟區儲存在同一個聚合中（本地或雲端）。因此，如果ONTAP儲存聚合遺失，我們將無法從其快照中復原應用程式資料。

**一個。建立 VolumeSnapshotClass 將以下清單保存在名為volume-snapshot-class.yaml的檔案中

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

使用上述清單建立快照。

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

b.接下來，建立一個快照透過建立 VolumeSnapshot 來建立現有 PVC 的快照，以取得 Postgresql 資料的時間點副本。這會建立一個幾乎不佔用檔案系統後端空間的 FSx 快照。將以下清單保存在名為volume-snapshot.yaml的檔案中：

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

c.建立磁碟區快照並確認已建立

刪除資料庫模擬資料遺失（資料遺失可能因為多種原因發生，這裡我們只是透過刪除資料庫來模擬）

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc create -f postgresql-volume-snapchot.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.刪除資料庫以模擬資料遺失（資料遺失可能由於多種原因而發生，這裡我們只是透過刪除資料庫來模擬）

```
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=#
```

從磁碟區快照還原

1.從快照還原 在本節中，我們將展示如何從應用程式磁碟區的 trident 快照還原應用程式。

一個。從快照建立磁碟區克隆

若要將磁碟區還原到先前的狀態，您必須根據所拍攝快照中的資料建立新的 PVC。為此，請將以下清單儲存在名為 pvc-clone.yaml 的檔案中

```
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
```

使用上述清單，透過使用快照作為來源建立 PVC 來建立磁碟區的複製。應用清單並確保創建克隆。

```
[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.刪除原始 **postgresql** 安裝

```
[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.使用新的克隆 **PVC** 建立新的 **postgresql** 應用程式

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

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# helm install postgresql bitnami/postgresql --set primary.persistence.enabled=true \
> --set primary.persistence.existingClaim=postgresql-volume-clone -n postgresql
NAME: postgresql
LAST DEPLOYED: Mon Oct 14 12:03:31 2024
NAMESPACE: postgresql
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: postgresql
CHART VERSION: 15.5.21
APP VERSION: 16.4.0

** Please be patient while the chart is being deployed **

PostgreSQL can be accessed via port 5432 on the following DNS names from within your cluster:

    postgresql.postgresql.svc.cluster.local - Read/Write connection

To get the password for "postgres" run:

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

To connect to your database run the following command:

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

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

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

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

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

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

d.驗證應用程式 **pod** 處於運行狀態

```
[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.驗證 **Pod** 是否使用克隆作為其 **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) 若要驗證資料庫是否已如預期恢復，請返回容器控制台並顯示現有資料庫

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:1
$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)
```

示範影片

使用託管控制平面在 AWS 上將Amazon FSx for NetApp ONTAP與 Red Hat OpenShift 服務結合使用

可以找到更多關於 Red Hat OpenShift 和 OpenShift 解決方案的視頻"這裡"。