AWS 上的 Red Hat OpenShift Service 搭配 FSxN

NetApp Solutions

NetApp
December 19, 2024

目錄

AWS 上的 Red Hat OpenShift Service 搭配 FSxN

AWS 上的 Red Hat OpenShift 服務搭配 NetApp ONTAP

總覽

在本節中、我們將說明如何將適用於 ONTAP 的 FSX 作為在 ROSA 上執行之應用程式的持續儲存層。它將顯示在 ROSA 叢集上安裝 NetApp Trident CSI 驅動程式、為 ONTAP 檔案系統提供 FSX 、以及部署可設定狀態的應用程式範例。它也會顯示備份及還原應用程式資料的策略。有了這套整合式解決方案、您就能建立共享儲存架構、輕鬆地在各個 AZs 之間擴充、簡化擴充、保護及還原資料的程序、並使用 Trident CSI 驅動程式。

先決條件

- "AWS帳戶"
- "Red Hat 帳戶"
- IAM 使用者"具有適當權限"可建立及存取 ROSA 叢集
- "AWS CLI"
- "ROSA CLI"
- "OpenShift 命令列介面"（ OC ）
- 船舵 3."文件"
- "HCP ROSA 叢集"
- "存取 Red Hat OpenShift Web 主控台"

此圖顯示部署在多個 AZs 中的 ROSA 叢集。ROSA 叢集的主節點、基礎架構節點位於 Red Hat 的 VPC 中、而工作節點則位於客戶帳戶的 VPC 中。我們將在同一部 VPC 中建立適用於 ONTAP 檔案系統的 FSX 、並在 ROSA 叢集中安裝 Trident 驅動程式、讓此 VPC 的所有子網路都能連線至檔案系統。

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 AZ FSX 。有幾種方法可以做到這一點。我們將提供使用 CloudForgation Stack 建立 FSxN 的詳細資料

- 完整複製 GitHub 儲存庫 **

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

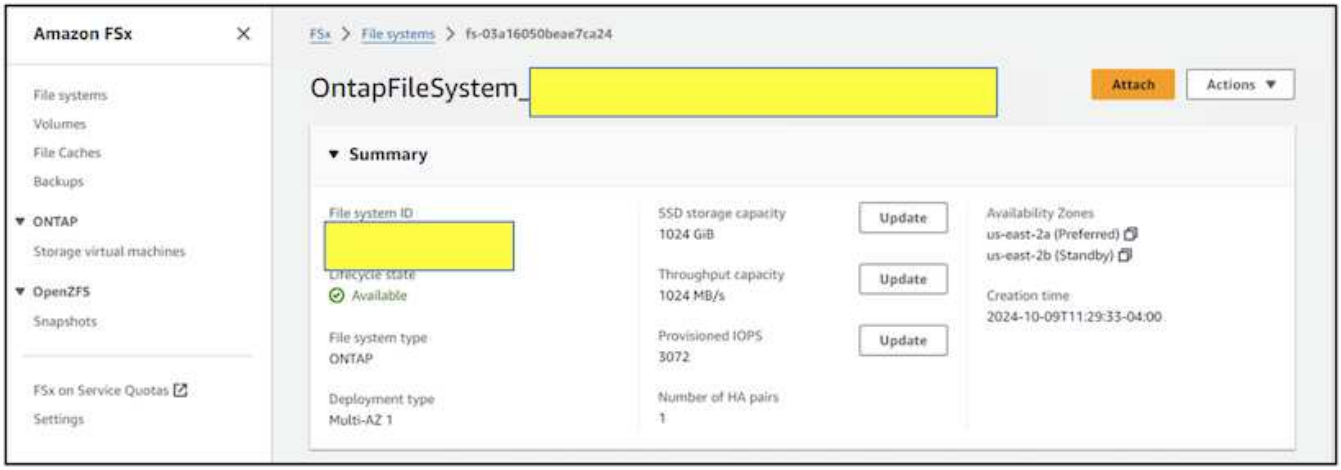
- b 。執行 CloudForomation Stack** 執行下列命令、將參數值取代為您自己的值：

```
$ 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 叢集的區域相同 subnet1_ID ： FSxN 子網路偏好的子網路 ID 2_ID ： FSxN VPC_ID 的待命子網路 ID ：部署 ROSA 叢集的 VPC ID routetable1_ID 、 routetable2_ID ：允許使用 CIDR 子網路 ONTAP 存取的路由表 ID 。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適用於 ONTAP 的特定 FSX 連接埠。定義管理員密碼：登入 FSxN 的密碼定義 SVM 密碼：登入將要建立的 SVM 的密碼。

確認您的檔案系統和儲存虛擬機器（ SVM ）已使用 Amazon FSX 主控台建立、如下所示：



2. 安裝及設定 **ROSA** 叢集的 **Trident CSI** 驅動程式

1. 新增 Trident Helm 儲存庫 **

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

- b 。使用 helm** 安裝 Trident

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

視您安裝的版本而定、需要在所示命令中變更版本參數。請參閱"文件"以取得正確的版本編號。有關安裝 Trident "文件"的其他方法，請參閱 Trident 。

c. 驗證所有 **Trident Pod** 是否都處於運行狀態

```
[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]#
```

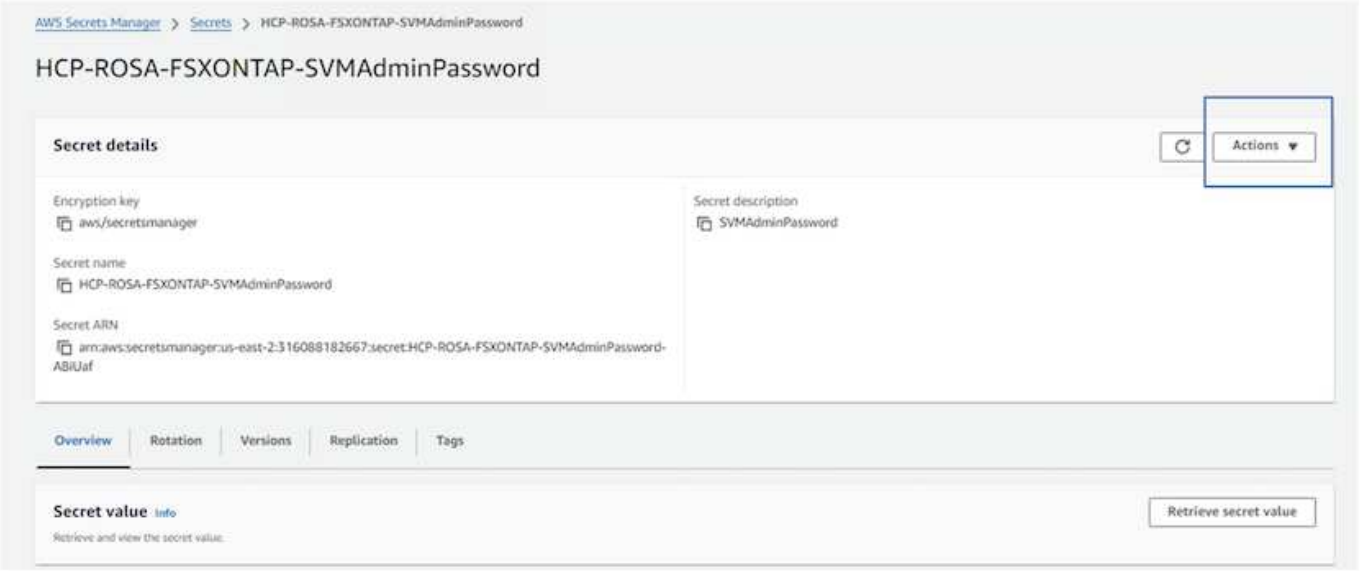
3.將 **Trident CSI** 後端設定為使用適用於 **ONTAP** 的 **FSX** （ **ONTAP NAS** ）

Trident 後端組態會告訴 Trident 如何與儲存系統通訊（在此案例中為 ONTAP 的 FSX ）。為了建立後端、我們會提供要連線的儲存虛擬機器認證、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 檔案系統中配置儲存磁碟區。

a.首先、使用下列 **yaml** 建立 SVM 認證的機密

```
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 擷取為 FSxN 建立的 SVM 密碼、如下所示。

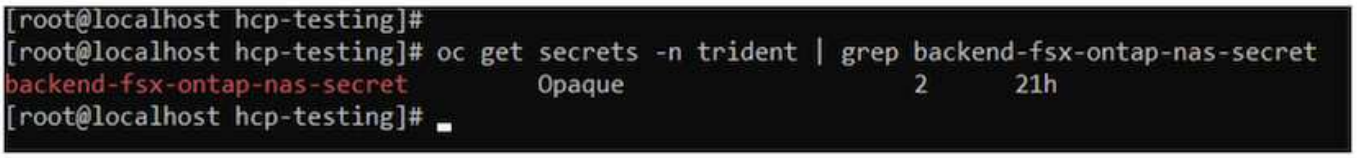


- b.Next ：使用下列命令將 SVM 認證的機密新增至 ROSA 叢集 **

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

您可以使用下列命令來驗證是否已將機密新增至 Trident 命名空間

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

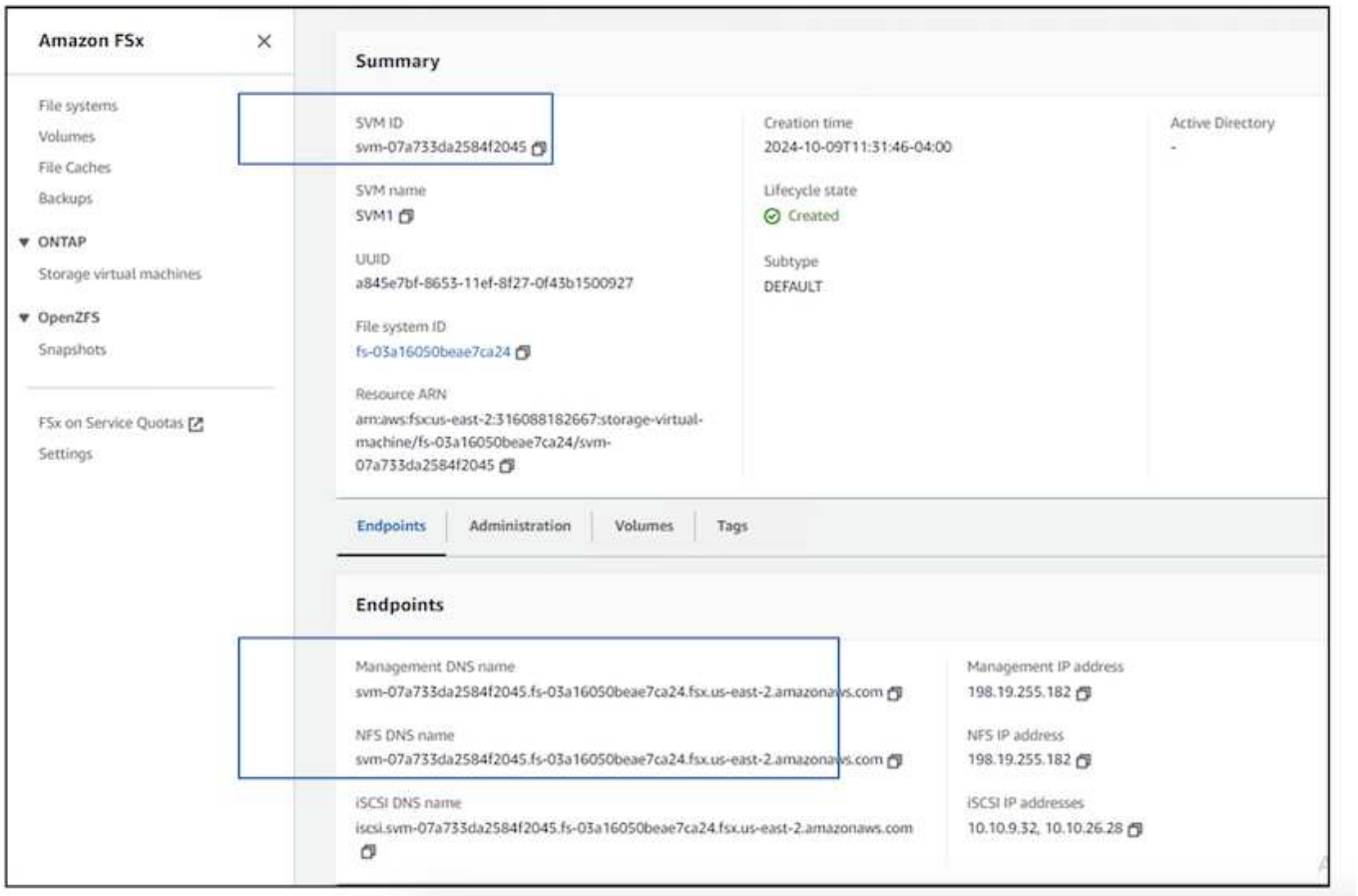


- c.接下來、為此建立後端物件 、移至複製 **Git** 儲存庫的 FSX 目錄。開啟檔案 **ONTAP NAS** 。 **yaml** 。將以下內容替換爲： managementLIF 和管理 **DNS** 名稱 dataLIF ，用 **Amazon FSX SVM** 的 **NFS DNS** 名稱和使用 **SVM** 名稱的 SVM** 。使用下列命令建立後端物件。

使用下列命令建立後端物件。

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

您可以從 Amazon FSX 主控台取得管理 DNS 名稱、 NFS DNS 名稱和 SVM 名稱、如下面的螢幕擷取畫面所示



- d.現在、請執行下列命令、確認已建立後端物件、且 Phase 顯示「界限」和「狀態」為「成功」 **

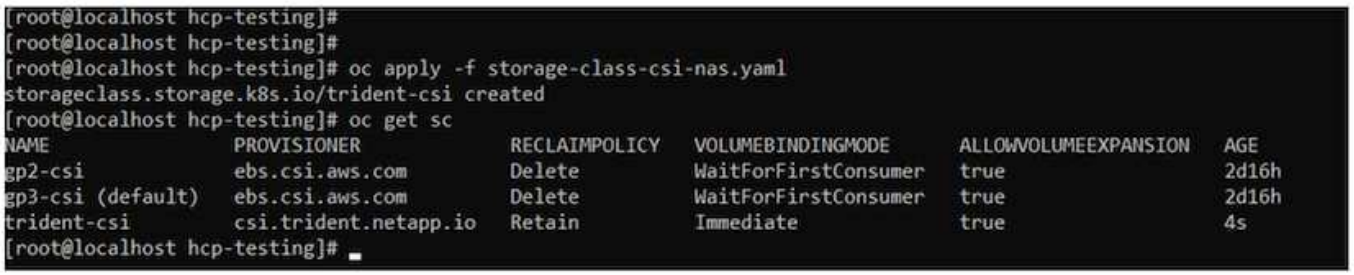
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc apply -f backend-ontap-nas.yaml
tridentbackendconfig.trident.netapp.io/backend-fsx-ontap-nas created
[root@localhost hcp-testing]# oc get tbc -n trident
NAME                   BACKEND NAME   BACKEND UUID                           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 Trident 後端已設定好、您可以建立 Kubernetes 儲存類別以使用後端。儲存類別是可供叢集使用的資源物件。它說明並分類您可以申請應用程式的儲存類型。

a.檢閱 **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、並確認已建立 Trident CSI 儲存類別。 **



這將完成 Trident CSI 驅動程式的安裝、以及其與適用於 ONTAP 檔案系統之 FSX 的連線。現在您可以使用適用於 ONTAP 的 FSX 上的檔案磁碟區、在 ROSA 上部署 PostgreSQL 狀態應用程式範例。

- c.確認沒有使用 Trident 儲存類別建立的 PVCs 和 PVs 。 **

- d.確認應用程式可以使用 Trident CSI 建立 PV 。 **

使用在 **fsx** 文件夾中提供的 Trident ・ yaml 文件創建 PVC 。

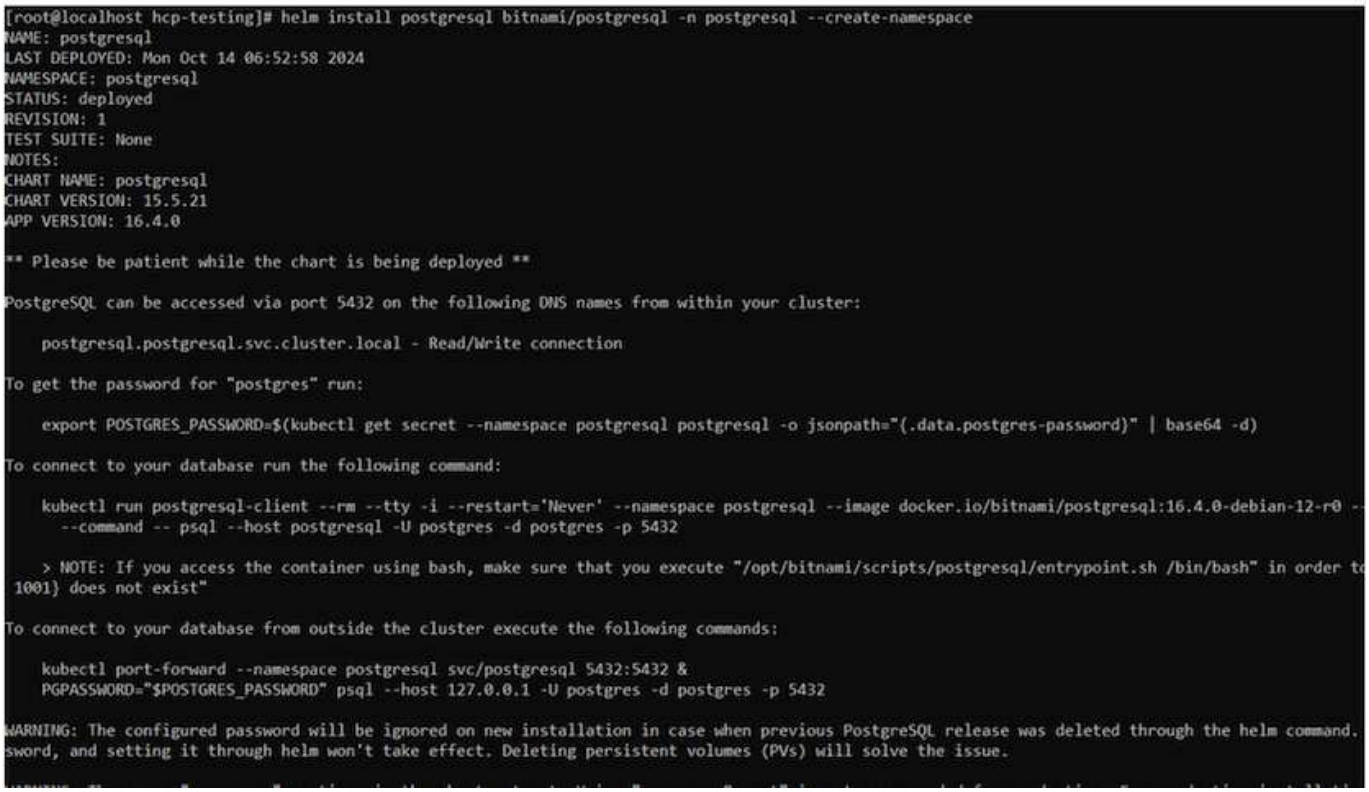
```
pvc-trident.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: trident-csi
```

```
 You can issue the following commands to create a pvc and verify that it
has been created.
image:redhat_openshift_container_rosa_image11.png["使用 Trident 建立測試
PVC"]
```

5.部署 **PostgreSQL** 有狀態應用程式的範例

**a.使用 helm 來安裝 PostgreSQL *

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



- b.確認應用程式 Pod 正在執行、並為應用程式建立了 PVC 和 PV 。 **

- c.部署 PostgreSQL 用戶端 **
- 使用下列命令取得安裝的 PostgreSQL 伺服器密碼。 **

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

- 使用下列命令來執行 PostgreSQL 用戶端、並使用 password** 連線至伺服器

```
$ 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.建立資料庫和資料表。為表格建立架構、並將 2 列資料插入表格。 **

```
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)
```

AWS 上的 Red Hat OpenShift 服務搭配 NetApp ONTAP

本文件將概述如何在 AWS （ ROSA ）上搭配 Red Hat OpenShift 服務使用 NetApp ONTAP 。

建立 Volume Snapshot

1.建立應用程式 **Volume** 的快照 在本節中、我們將示範如何建立與應用程式相關之 Volume 的 Trident 快照。這將是應用程式資料的時間點複本。如果應用程式資料遺失、我們可以從時間點複本恢復資料。附註：此快照儲存在與 ONTAP （內部部署或雲端）中原始磁碟區相同的集合中。因此、如果 ONTAP 儲存集合體遺失、我們就無法從其快照中恢復應用程式資料。

**a.建立 Volume SnapshotClass 將下列資訊清單儲存在名為 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
```

使用上述資訊清單建立快照。

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

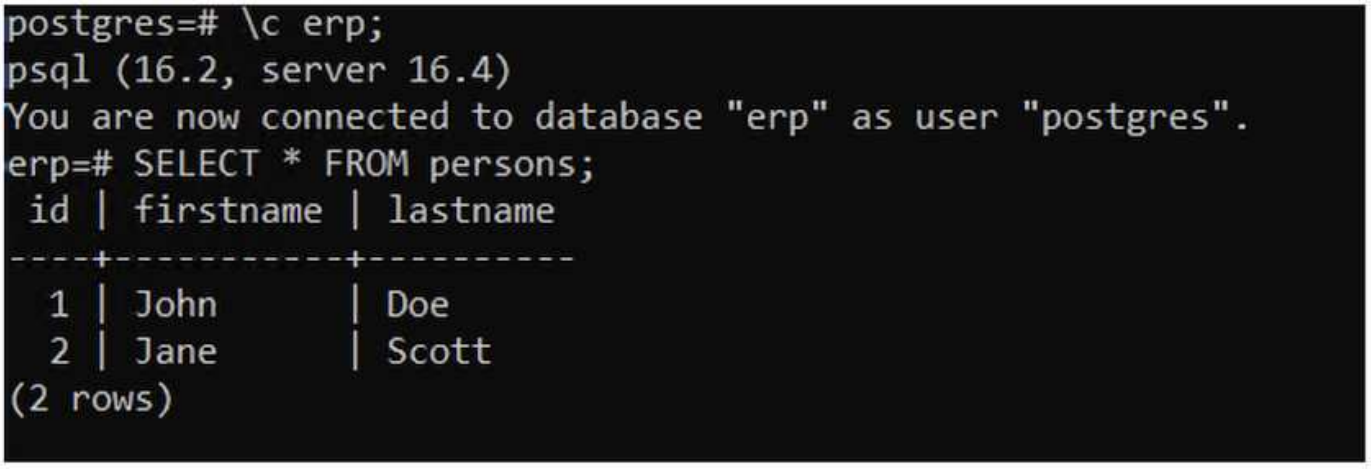
- b.接下來、建立 SnapShot * 建立現有 PVC 的快照、建立 Volume Snapshot 來製作 PostgreSQL 資料的時間點複本。這會建立一個 FSX 快照、幾乎不需要檔案系統後端的空間。將下列資訊清單儲存在名為 volume － snapshot ・ yaml 的檔案中：

```
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 Volume 快照並確認已建立 **

刪除資料庫以模擬資料遺失（資料遺失可能因各種原因而發生、在此我們只是刪除資料庫來模擬資料遺失）

- d.刪除資料庫以模擬資料遺失（資料遺失可能因各種原因而發生、在此我們只是刪除資料庫來模擬資料遺失）**



從 **Volume Snapshot** 還原

1.從 **Snapshot** 還原 在本節中、我們將說明如何從應用程式 Volume 的 Trident 快照還原應用程式。

a.從 **SnapShot** 建立磁碟區複本

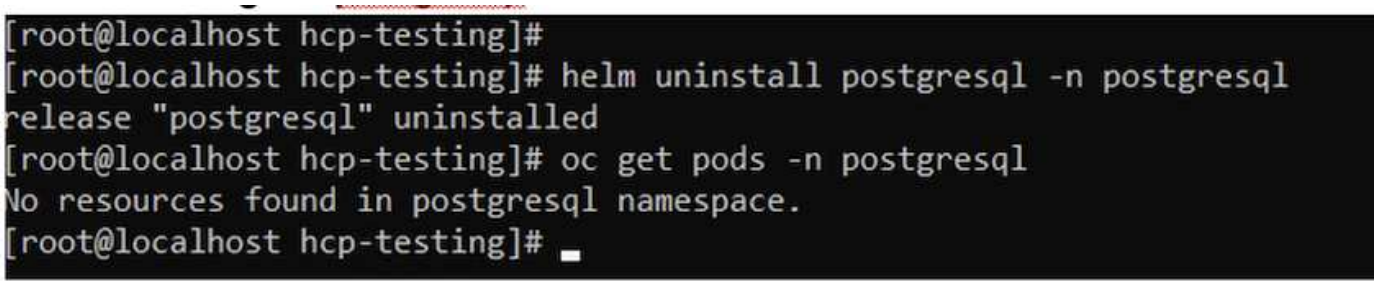
若要將磁碟區還原至先前的狀態、您必須根據所拍攝快照中的資料建立新的 PVC 。若要這麼做、請將下列資訊

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
```

使用上述資訊清單建立 PVC 作為來源、藉此建立磁碟區的複本。套用資訊清單、並確定已建立複本。

- b.刪除原始的 PostgreSQL 安裝 **



- c.使用新的複製 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.確認應用程式 Pod 處於執行中狀態 **

```
[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.確認 Pod 使用複本作為 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如預期還原、請返回容器主控台並顯示現有的資料庫

```
[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)
```

示範影片

Amazon FSX for NetApp ONTAP 搭配使用託管控制平面的 AWS 上的 Red Hat OpenShift 服務

有關 Red Hat OpenShift 和 OpenShift 解決方案的更多影片"請按這裡"、請參閱。