Red Hat OpenShift Service su AWS con FSxN

NetApp Solutions

NetApp
January 09, 2025

Sommario

Red Hat OpenShift Service su AWS con FSxN

Servizio Red Hat OpenShift su AWS con NetApp ONTAP

Panoramica

In questa sezione, mostreremo come utilizzare FSX per ONTAP come layer di storage persistente per le applicazioni eseguite su ROSA. Mostra 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. Oltre a mostrare le strategie per il backup e il ripristino dei dati dell'applicazione. Con questa soluzione integrata, è possibile stabilire un framework di storage condiviso che scala facilmente tra le zone di disponibilità, semplificando i processi di scalabilità, protezione e ripristino dei dati utilizzando il driver Trident CSI.

Prerequisiti

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

Questo diagramma mostra il cluster ROSA implementato in più zone di disponibilità. I nodi master del cluster ROSA, i nodi infrastruttura si trovano nel VPC di Red Hat, mentre i nodi di lavoro si trovano in un VPC nell'account del cliente . Creeremo un file system FSX per ONTAP con lo stesso VPC e installeremo il driver Trident nel cluster ROSA, permettendo a tutte le subnet di questo VPC di connettersi al file system.

Setup iniziale

1. Esegui il provisioning di FSX per NetApp ONTAP

Creare un FSX multi-AZ per NetApp ONTAP nello stesso VPC del cluster ROSA. Ci sono diversi modi per farlo. Vengono forniti i dettagli per la creazione di FSxN utilizzando uno stack di CloudFormation

A.Clona il repository di GitHub

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

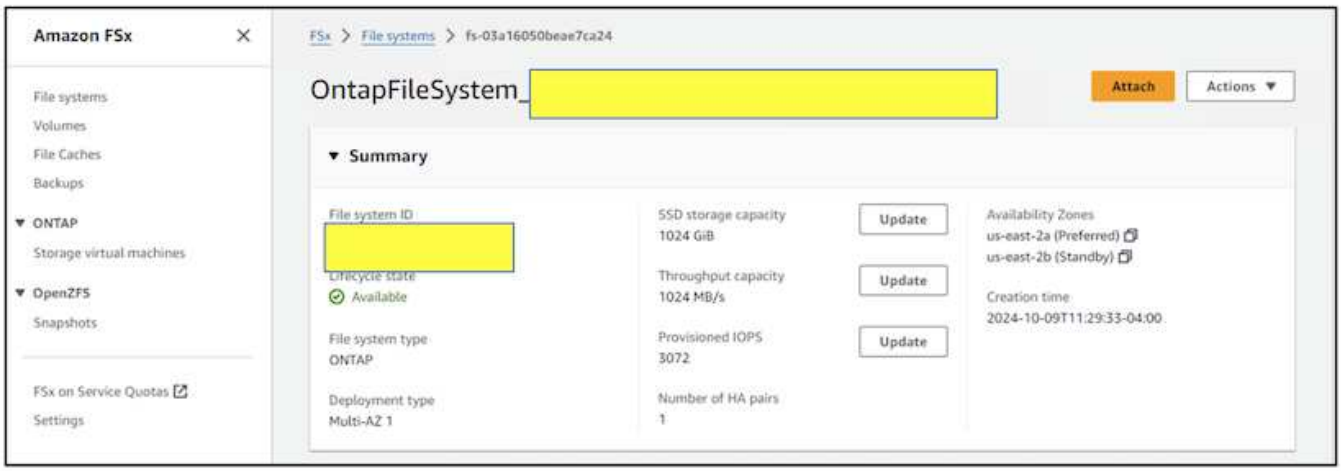
B.Run the CloudFormation Stack Esegui il comando qui sotto sostituendo i valori dei parametri con i tuoi valori:

```
$ 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 : nome-regione: Uguale alla regione in cui viene distribuito il cluster ROSA subnet1_ID : id della subnet preferita per FSxN subnet2_ID: id della subnet di standby per FSxN VPC_ID: id del VPC in cui viene distribuito il cluster ROSA routetable1_ID, routetable2_ID: ID delle tabelle di instradamento associate alle sottoreti scelte sopra il tuo_permesso_CIDR: Permesso l'intervallo di sicurezza CIDR per l'ingresso per i gruppi di controllo ONTAP. 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 Storage Virtual Machine (SVM) siano stati creati utilizzando la console Amazon FSX, illustrata di seguito:



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

A.aggiungere il repository Helm di Trident

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

B.installare Trident utilizzando helm

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

A seconda della versione installata, il parametro della versione dovrà essere modificato nel comando mostrato. Fare riferimento alla "documentazione"per il numero di versione corretto. Per ulteriori metodi di installazione di Trident, consultare Trident "documentazione".

C.verificare che tutti i pod Trident siano in stato di funzionamento

```
[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]#
```

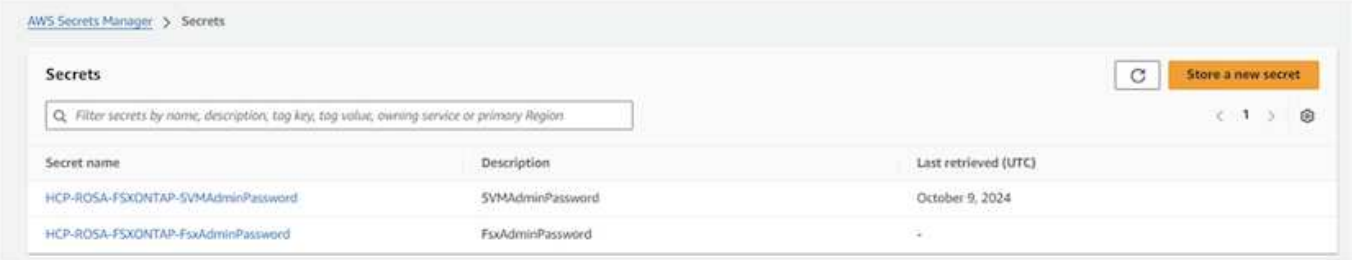
3. Configurare il backend Trident CSI per utilizzare FSX for ONTAP (ONTAP NAS)

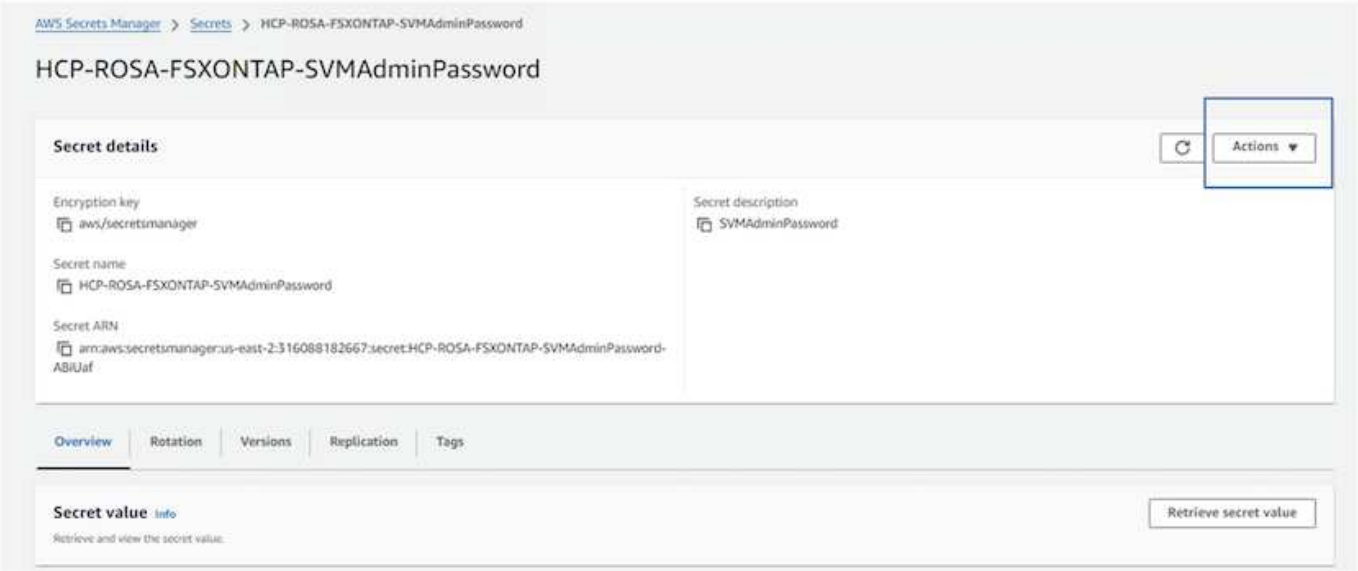
La configurazione back-end Trident indica a Trident come comunicare con il sistema storage (in questo caso FSX per ONTAP). Per la creazione del backend, forniremo le credenziali della Storage Virtual Machine a cui connettersi, insieme alle interfacce dati di Cluster Management e NFS. Utilizzeremo "driver ontap-nas"per il provisioning dei volumi di storage nel file system FSX.

a. Innanzitutto, creare 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>
```

Puoi anche recuperare la password SVM creata per FSxN da AWS Secrets Manager, come illustrato di seguito.





B.Avanti, aggiungere il segreto per le credenziali SVM al cluster ROSA utilizzando il seguente comando

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

È possibile 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
```

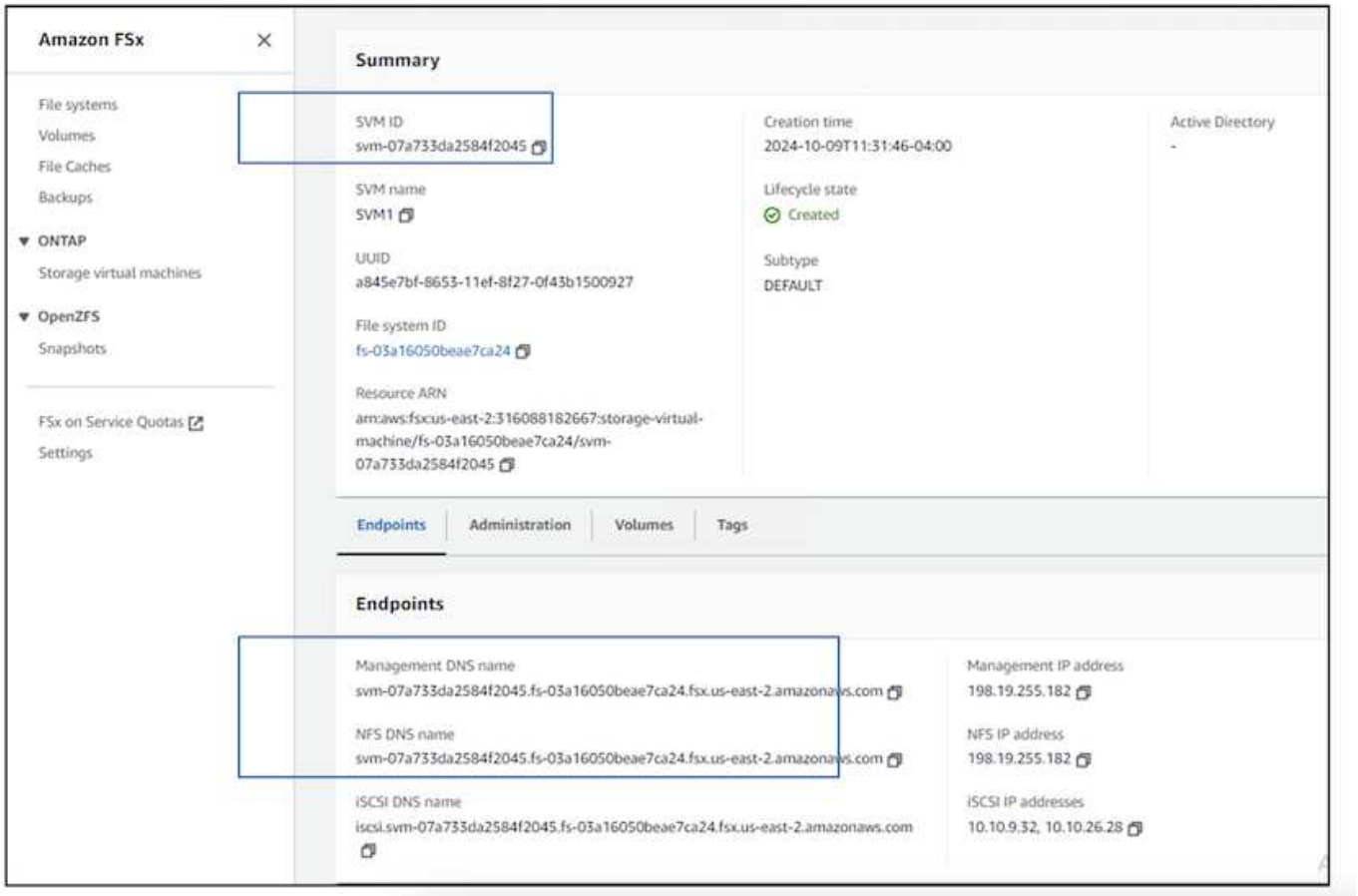
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

c. Successivamente, creare l'oggetto backend per questo, spostarsi nella directory **fsx** del 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 della SVM Amazon FSX e **svm** con il nome svm. Creare l'oggetto backend utilizzando il seguente comando.

Creare l'oggetto backend utilizzando il seguente comando.

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

Puoi ottenere il nome del DNS di gestione, il nome del DNS NFS e il nome della SVM dalla Amazon FSX Console, come mostrato nella screenshot seguente



d. A questo punto, eseguire il comando seguente per verificare che l'oggetto backend sia stato creato e che la fase sia associata e che lo stato sia riuscito

```
[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. Creare classe di storage ora che il back-end Trident è configurato, è possibile creare una classe di storage Kubernetes per utilizzare il back-end. La classe storage è un oggetto risorsa reso disponibile al cluster. Descrive e classifica il tipo di storage che puoi richiedere per un'applicazione.

a. 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 la 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]#
```

L'installazione del driver Trident CSI e la sua connettività al file system FSX per ONTAP vengono completate. Ora puoi implementare un'applicazione stateful PostgreSQL di esempio su ROSA usando i volumi di file su FSX per ONTAP.

c. Verificare che non siano stati creati PVC e PVC 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**.

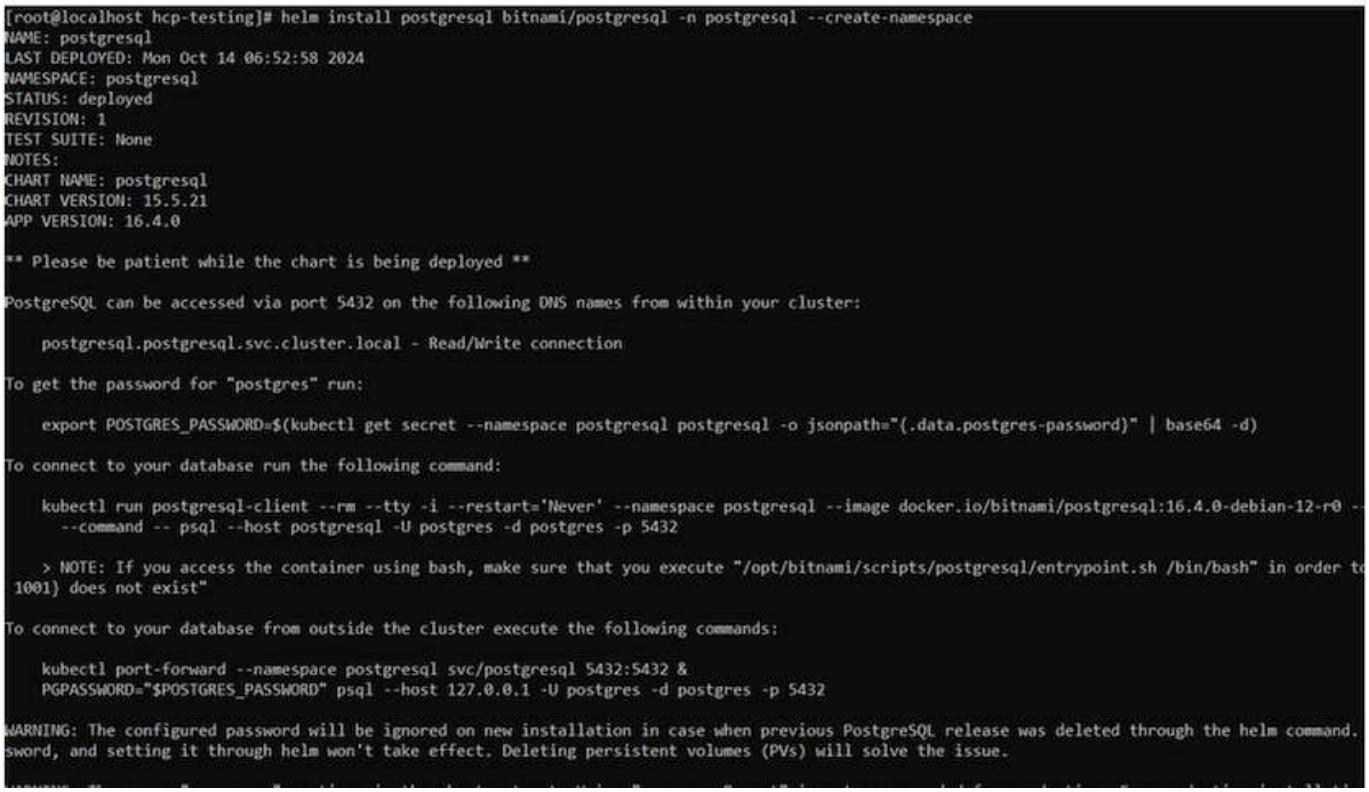
```
pvc-trident.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: trident-csi
```

```
 You can issue the following commands to create a pvc and verify that it
has been created.
image:redhat_openshift_container_rosa_image11.png["Creare un PVC di test
con Trident"]
```

5. Distribuire un'applicazione stateful PostgreSQL di esempio

a. Utilizzare helm per installare postgresql

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



b. Verificare che il pod applicazioni sia in esecuzione e che siano stati creati PVC e PV per l'applicazione.

c. Distribuire 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. Creare un database e una tabella. Creare uno schema per la tabella e inserire 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 spiega come utilizzare NetApp ONTAP con il servizio Red Hat OpenShift su AWS (ROSA).

Creare snapshot del volume

1. Creare un'istantanea del volume dell'app in questa sezione, verrà mostrato come creare un'istantanea Trident del volume associato all'app. Si tratta di una copia temporizzata dei dati dell'app. In caso di perdita dei dati dell'applicazione, siamo in grado di ripristinarli da questa copia point-in-time. NOTA: Questo snapshot viene memorizzato nello stesso aggregato del volume originale in ONTAP (on-premise o nel cloud). Pertanto, in caso di perdita dell'aggregato di storage ONTAP, non è possibile ripristinare i dati dell'applicazione dalla relativa istantanea.

**a. Creare un VolumeSnapshotClass salvare 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
```

Creare un'istantanea utilizzando il manifesto riportato sopra.

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

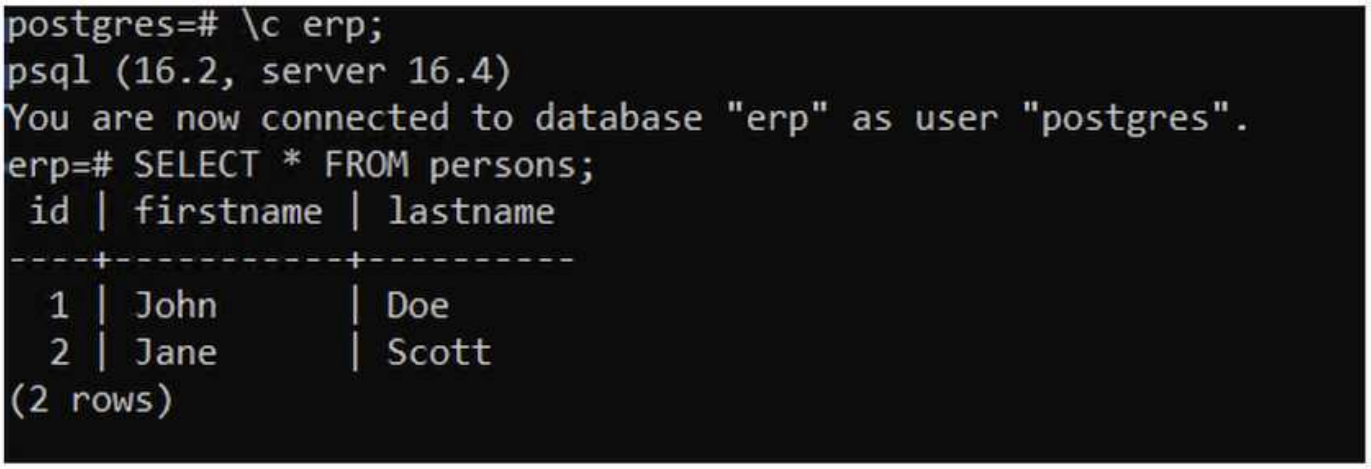
b. Creare quindi un'istantanea creare un'istantanea del PVC esistente creando VolumeSnapshot per acquisire una copia point-in-time dei dati PostgreSQL. Questo crea uno snapshot FSX che non occupa quasi spazio nel backend del filesystem. Salvare il seguente manifesto in un file chiamato volume-snapshot.yaml:

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

c. Creare lo snapshot del volume e confermarne la creazione

Eliminare il database per simulare la perdita di dati (la perdita di dati può verificarsi per una serie di motivi, in questo caso viene semplicemente simulata eliminando il database)

d. Eliminare il database per simulare la perdita di dati (la perdita di dati può verificarsi per una serie di motivi, qui stiamo solo simulando eliminando il database)



Ripristino da Snapshot volume

1. Ripristino da istantanea in questa sezione, verrà illustrato come ripristinare un'applicazione dallo snapshot Trident del volume dell'applicazione.

a. Creare un clone del volume dallo snapshot

Per ripristinare lo stato precedente del volume, è necessario creare un nuovo PVC in base ai dati nello snapshot acquisito. A tale scopo, salvare il manifesto seguente 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 riportato sopra. Applicare il manifesto e assicurarsi che il clone sia stato 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. Eliminare l’installazione postgresql originale

```
[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. Creare una nuova applicazione postgresql utilizzando il nuovo PVC clone

```
$ 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. Verificare che il pod applicazioni 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 verificare che il database sia stato ripristinato come previsto, tornare alla console 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 per NetApp ONTAP con il servizio Red Hat OpenShift su AWS usando Hosted Control Plane

Ulteriori video sulle soluzioni Red Hat OpenShift e OpenShift sono disponibili "qui".

Informazioni sul copyright

Informazioni sul marchio commerciale