FSxNを使用したAWSでのRed Hat OpenShiftサービス

NetApp Solutions

NetApp
December 19, 2024

目次

FSxNを使用したAWSでのRed Hat OpenShiftサービス

AWSでのRed Hat OpenShiftサービスと NetApp ONTAP

概要

このセクションでは、ROSAで実行されるアプリケーションの永続的ストレージレイヤとしてFSx for ONTAPを活用する方法を説明します。ROSAクラスタへのNetApp Trident CSIドライバのインストール、FSx for ONTAPファイルシステムのプロビジョニング、サンプルステートフルアプリケーションの導入について説明します。また、アプリケーションデータをバックアップおよび復元するための戦略も示します。この統合ソリューションにより、AZ間で容易に拡張できる共有ストレージフレームワークを確立し、Trident CSIドライバを使用してデータのスケーリング、保護、リストアのプロセスを簡素化できます。

前提条件

- "AWS アカウント"
- "Red Hatアカウント"
- ROSAクラスタを作成してアクセスするためのIAMユーザ"適切な権限を持つ"
- "AWS CLI"
- "ローザCLI"
- "OpenShiftコマンドラインインターフェイス"（OC）
- ヘルム3"ドキュメント"
- "HCP ROSAクラスタ"
- "Red Hat OpenShift Webコンソールへのアクセス"

この図は、複数のAZに展開されたROSAクラスタを示しています。Rosaクラスタのマスターノード、インフラノードはRed HatのVPCにあり、ワーカーノードはお客様のアカウントのVPCにあります。同じVPC内にFSx for ONTAPファイルシステムを作成し、TridentドライバをROSAクラスタにインストールして、このVPCのすべてのサブネットがファイルシステムに接続できるようにします。

初期セットアップ

- 1.FSx for NetApp ONTAPのプロビジョニング**

ROSAクラスタと同じVPC内にマルチAZ FSx for NetApp ONTAPを作成します。これにはいくつかの方法があります。CloudFormationスタックを使用したFSxNの作成の詳細が記載されています。

- a. GitHubリポジトリのクローン**

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

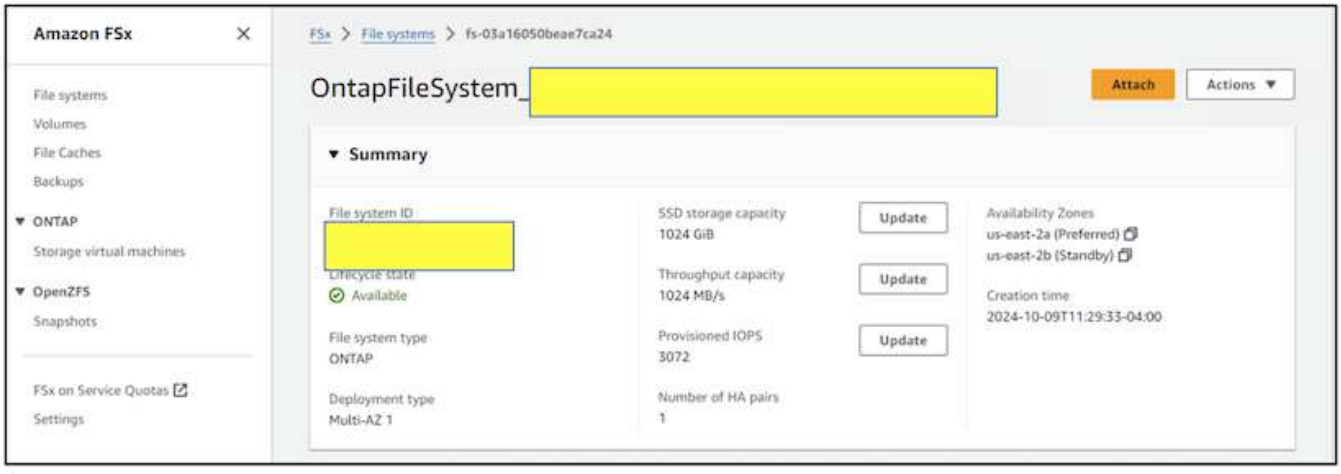
- B. CloudFormationスタックの実行**パラメータ値を独自の値に置き換えて、以下のコマンドを実行します。

```
$ 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
```

WHERE: region-name: ROSAクラスタが展開されているリージョンと同じsubnet1_ID: FSxNサブネットの優先サブネットのid 2_ID: FSxNのスタンバイサブネットのid VPC_ID: ROSAクラスタが展開されているVPCのid routetable1_ID, routetable2_ID: ONTAPすべてのトラフィックがFSx for ONTAPの特定のポートにアクセスできるようにするには、0.0.0.0/0または適切なCIDRを使用します。Define Admin password：FSxNにログインするためのパスワードDefine SVM password：作成されるSVMにログインするためのパスワード。

以下に示すAmazon FSxコンソールを使用して、ファイルシステムとStorage Virtual Machine（SVM）が作成されていることを確認します。



- 2. ROSAクラスタ用のTrident CSIドライバのインストールと設定**
- a. Trident Helmリポジトリの追加**

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

- B. helmを使用してTridentをインストール**

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

インストールするバージョンに応じて、表示されているコマンドでversionパラメータを変更する必要があります。正しいバージョン番号については、を参照して"ドキュメント"ください。Tridentのその他のインストール方法については、『Trident』を参照して"ドキュメント"ください。

- C.すべてのTridentポッドが実行中状態であることを確認します**

```
[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]#
```

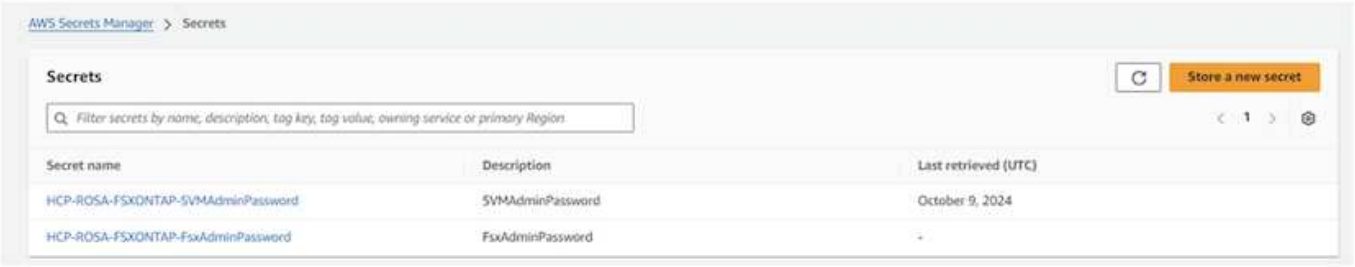
- 3.FSx for ONTAP（ONTAP NAS）を使用するようにTrident CSIバックエンドを構成**

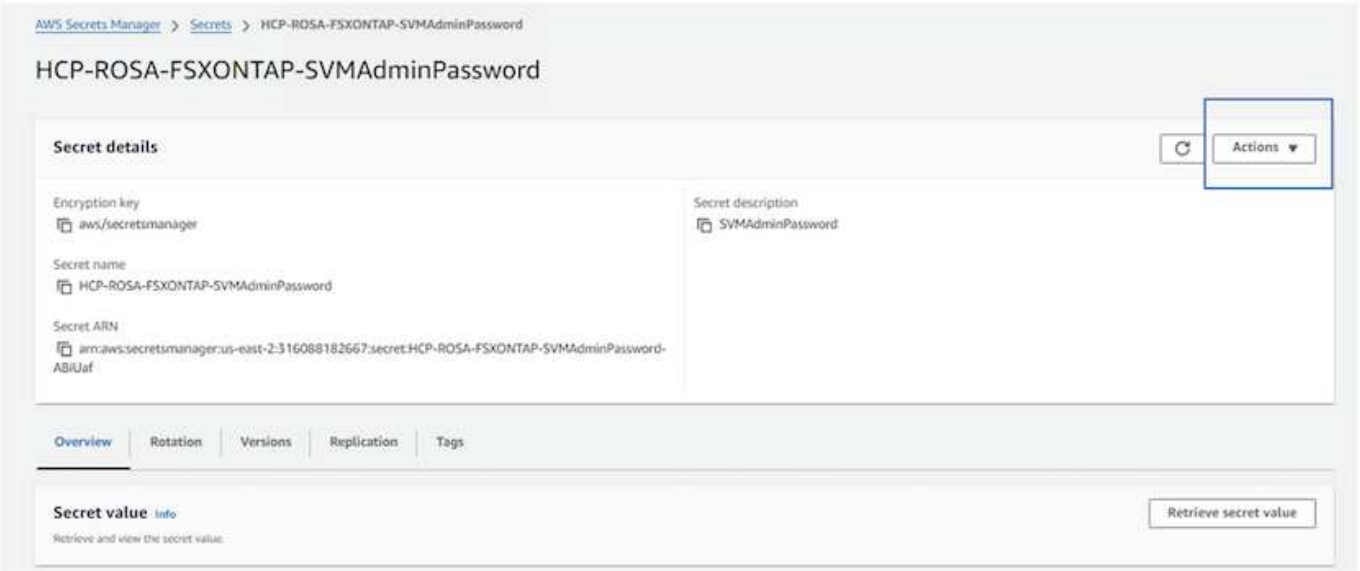
Tridentバックエンド構成では、ストレージシステムとの通信方法（ここではFSx for ONTAP）をTridentに設定します。バックエンドの作成にあたっては、クラスタ管理インターフェイスとNFSデータインターフェイスに接続するSVMのクレデンシャルを指定します。を使用して、FSxファイルシステムでストレージボリュームをプロビジョニングします"ONTAP - NAS ドライバ"。

- a。まず、次のYAMLを使用してSVMクレデンシャルのシークレットを作成します**

```
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>
```

以下に示すように、FSxN用に作成したSVMのパスワードをAWS Secrets Managerから取得することもできます。





- B.次に、次のコマンドを使用してSVMクレデンシャルのシークレットをROSAクラスタに追加します**

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

シークレットがTridentネームスペースに追加されたことを確認するには、次のコマンドを使用します。

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

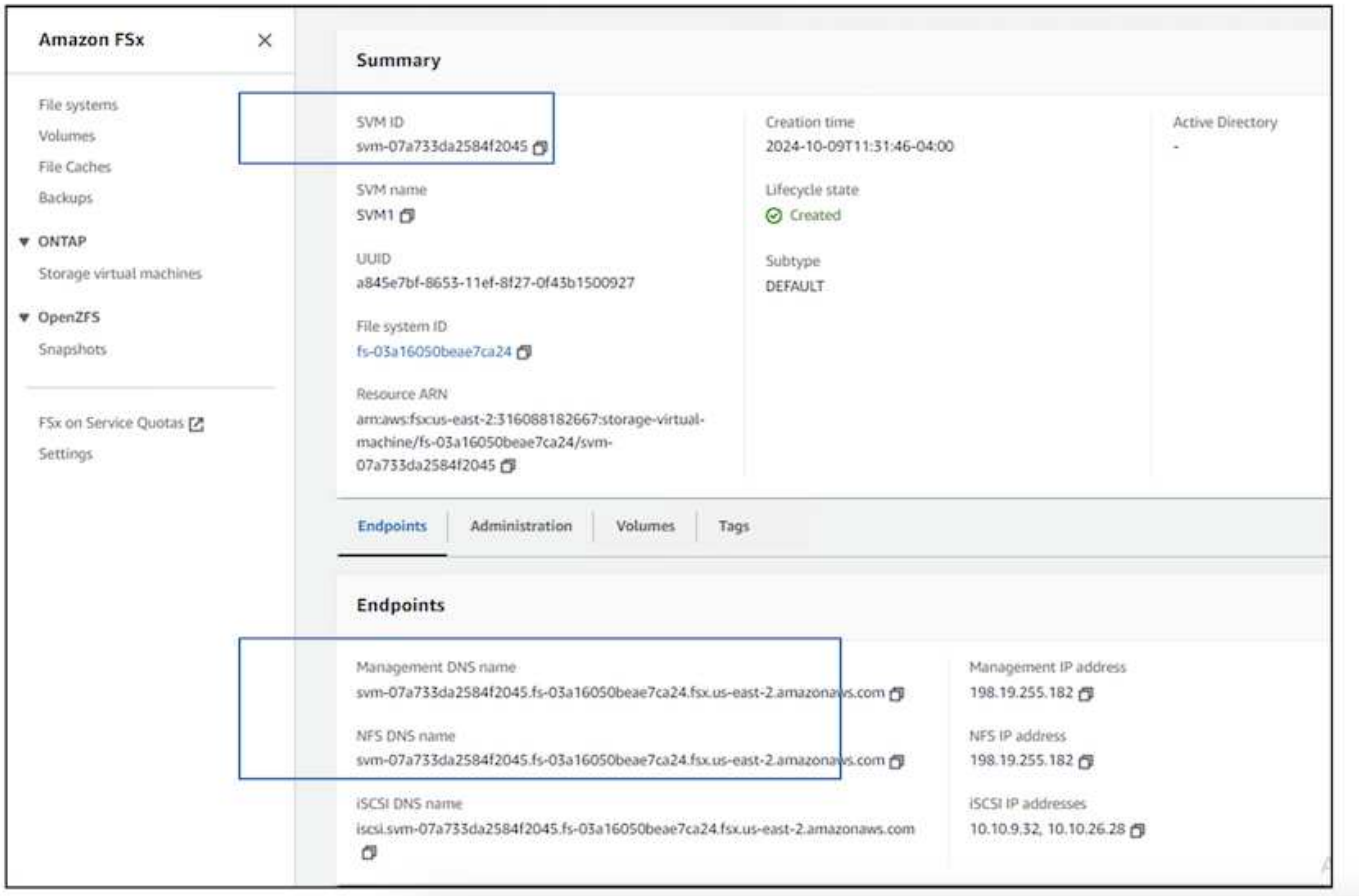
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc get secrets -n trident | grep backend-fsx-ontap-nas-secret
backend-fsx-ontap-nas-secret         Opaque                  2      21h
[root@localhost hcp-testing]#
```

- C。次に、このためのバックエンドオブジェクトを作成し、クローンされた**Git**リポジトリの FSXディレクトリに移動します。**backend-ansc-nas.yaml**ファイルを開きます**ONTAP**。次の項目を、 managementLIFを管理**DNS**名 dataLIFに、**Amazon FSx SVM**の**NFS DNS**名、 SVM**をSVM名に置き換えます。次のコマンドを使用して、バックエンドオブジェクトを作成します。

次のコマンドを使用して、バックエンドオブジェクトを作成します。

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

以下のスクリーンショットに示すように、Amazon FSxコンソールから管理DNS名、NFS DNS名、SVM名を確認できます。



- d。次に、次のコマンドを実行して、バックエンドオブジェクトが作成され、[フェーズ]が[バインド済み]、[ステータス]が[成功]になっていることを確認します**

```
[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.ストレージクラスの作成**これでTridentバックエンドを設定したので、バックエンドを使用するKubernetesストレージクラスを作成できます。ストレージクラスは、クラスタで使用できるリソースオブジェクトです。アプリケーションに対して要求できるストレージのタイプについて説明し、分類します。

- a。FSxフォルダのstorage-class-csi-nas.yamlファイルを確認します。**

```
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.ROSAクラスタでストレージクラスを作成し、Trident CSIストレージクラスが作成されていることを確認します。**

```
[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]#
```

これで、Trident CSIドライバのインストールとFSx for ONTAPファイルシステムへの接続は完了です。FSx for ONTAPのファイルボリュームを使用して、サンプルのPostgreSQLステートフルアプリケーションをROSAに導入できるようになりました。

- C。Trident CSIストレージクラスを使用して作成されたPVCおよびPVCがないことを確認します。**

```
[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-f4374ce7-588d-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-f4374ce7-588d-4afc-b635-0228cf4544d4   30Gi       RWO            Delete           Bound    openshift-virtualization-os-images/rhel9-2521bd116e64                  gp3-csi        <unset>
[root@localhost hcp-testing]#
```

- d。アプリケーションがTrident CSIを使用してPVを作成できることを確認します。**
- fsx**フォルダにあるpvc-pvc.yamlファイルを使用してTridentを作成します。

```
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["Tridentを使用したテストPVC
の作成"]
```

- 5.サンプルのPostgreSQLステートフルアプリケーションの導入**
- a。Helmを使用してPostgreSQLをインストール**

```
$ 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.アプリケーションポッドが実行中であること、およびアプリケーション用にPVCとPVが作成されていることを確認します。**

```
[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。PostgreSQLクライアントの配備**

次のコマンドを使用して、インストールされた**PostgreSQL**サーバーのパスワードを取得します。

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

次のコマンドを使用して**PostgreSQL**クライアントを実行し、パスワードを使用してサーバに接続します

```
$ 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。データベースとテーブルを作成します。テーブルのスキーマを作成し、テーブルに2行のデータを挿入します。**

```
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でのRed Hat OpenShiftサービスとNetApp ONTAP

このドキュメントでは、Red Hat OpenShift Service on AWS（ROSA）でNetApp ONTAPを使用する方法について説明します。

ボリュームSnapshotの作成

- 1.アプリボリュームのスナップショットを作成する**このセクションでは、アプリに関連付けられたボリュームのTridentスナップショットを作成する方法を説明します。これは、アプリデータのポイントインタイムコピーです。アプリケーションデータが失われた場合は、この時点のコピーからデータをリカバリできます。注：このSnapshotは、ONTAP（オンプレミスまたはクラウド）の元のボリュームと同じアグリゲートに格納されます。そのため、ONTAPストレージアグリゲートが失われた場合、スナップショットからアプリデータを回復できません。
- a。VolumeSnapshotClassを作成するvolume-snapshot-class.yamlという名前のファイルに次のマニフェストを保存します。

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

上記のマニフェストを使用してスナップショットを作成します。

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

- B.次に、スナップショットを作成します** VolumeSnapshotを作成して、PostgreSQLデータのポイントインタイムコピーを作成し、既存のPVCのスナップショットを作成します。これにより、ファイルシステム

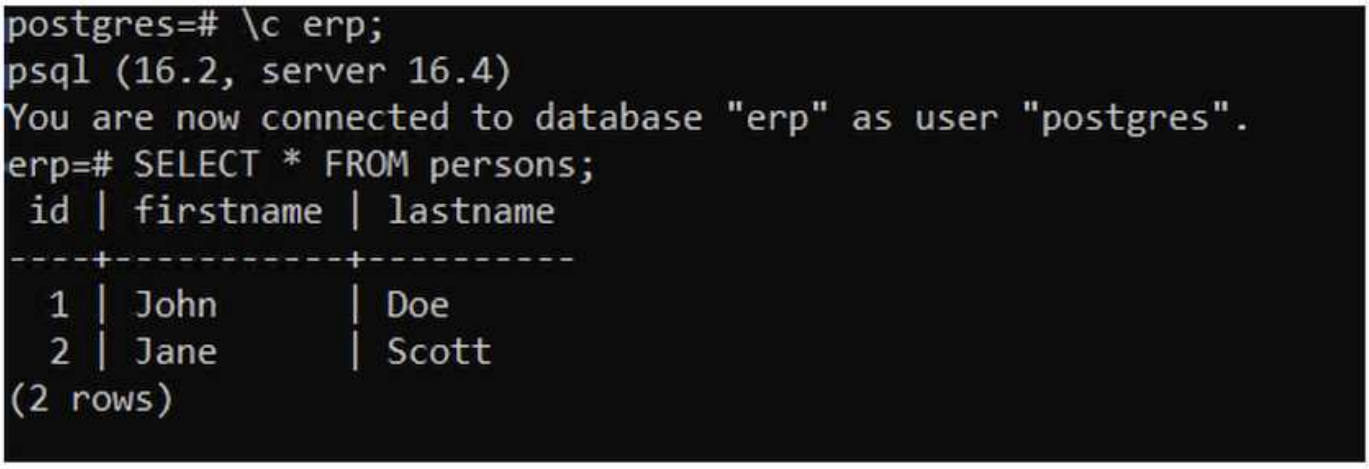
バックエンドにほとんどスペースを消費しないFSxスナップショットが作成されます。次のマニフェストをvolume-snapshot.yamlという名前のファイルに保存します。

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

- C。ボリュームSnapshotを作成し、作成されたことを確認します**

データベースを削除してデータ損失をシミュレートします（データ損失はさまざまな理由で発生する可能性があります。ここでは、データベースを削除してシミュレートしています）。

- d。データベースを削除してデータ損失をシミュレートします（データ損失はさまざまな理由で発生する可能性があります。ここでは、データベースを削除してシミュレートしています）**



ボリューム**Snapshot**からリストア

- 1.スナップショットからの復元**このセクションでは、アプリボリュームのTridentスナップショットから

アプリケーションを復元する方法を説明します。

- a。スナップショットからボリュームクローンを作成**

ボリュームを以前の状態に復元するには、作成したスナップショットのデータに基づいて新しいPVCを作成する必要があります。これを行うには、次のマニフェストをpvc-clone.yamlという名前のファイルに保存します。

```
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
```

上記のマニフェストを使用して、スナップショットをソースとしてPVCを作成し、ボリュームのクローンを作成します。マニフェストを適用し、クローンが作成されたことを確認します。

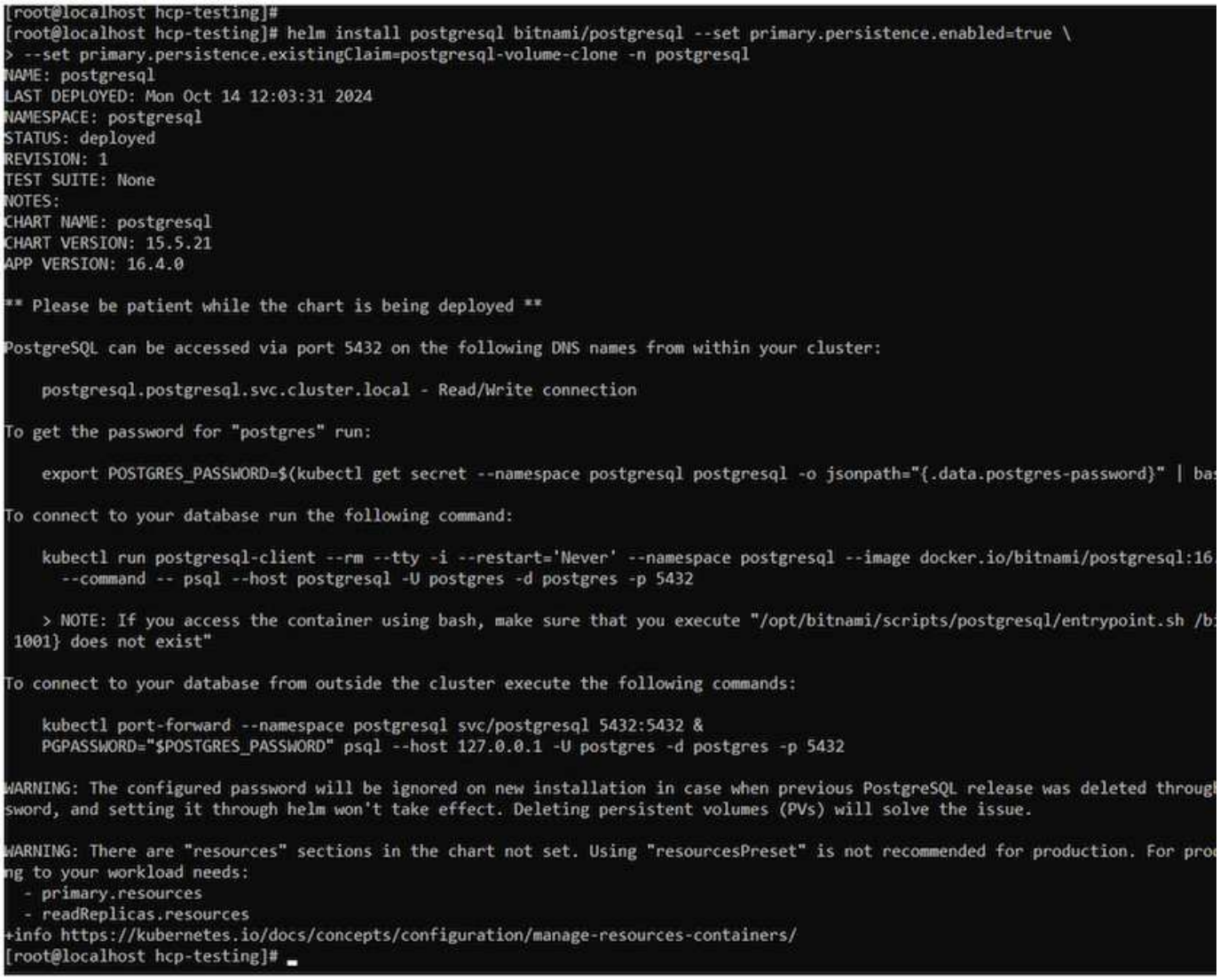
```
[root@localhost hcp-testing]# oc create -f postgresql-pvc-clone.yaml -n postgresql
persistentvolumeclaim/postgresql-volume-clone created
[root@localhost hcp-testing]# oc get pvc -n postgresql
NAME                     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
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.元のPostgreSQLインストールの削除**

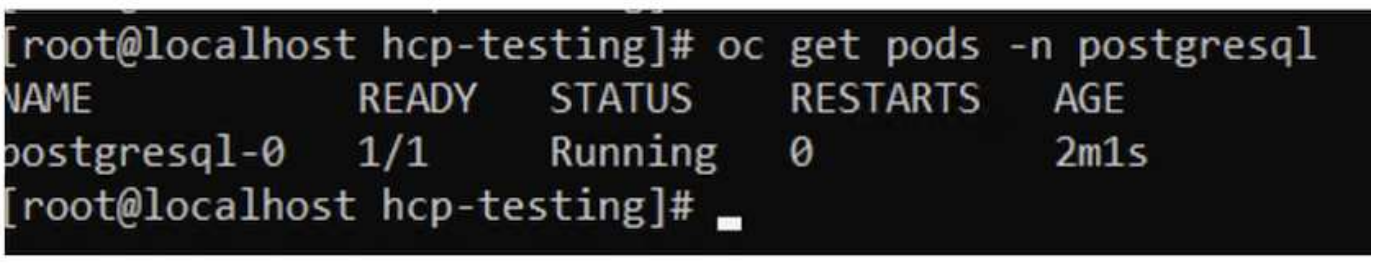
```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# helm uninstall postgresql -n postgresql
release "postgresql" uninstalled
[root@localhost hcp-testing]# oc get pods -n postgresql
No resources found in postgresql namespace.
[root@localhost hcp-testing]#
```

- C。新しいクローンPVCを使用して新しいPostgreSQLアプリケーションを作成する**

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



- d。アプリケーションポッドが実行中状態であることを確認します**

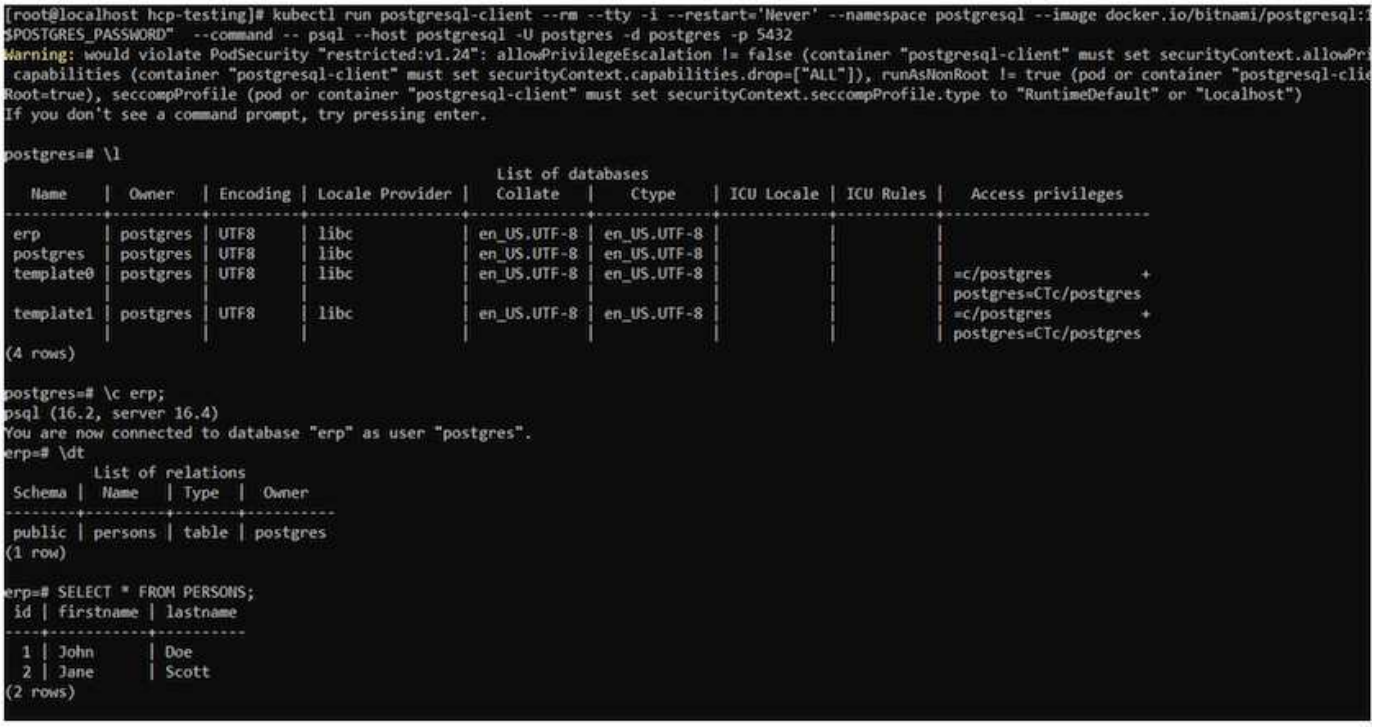


- e。ポッドがクローンをPVCとして使用することを確認します**

```
[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）データベースが想定どおりにリストアされたことを確認するには、コンテナコンソールに戻り、既存のデータベースを表示します。



デモビデオ

Amazon FSx for NetApp ONTAPとAWSでのRed Hat OpenShiftサービス（ホスト型コントロールプレーンを使用）

Red Hat OpenShiftおよびOpenShiftソリューションに関するその他のビデオをご覧ください"こちらをご覧ください"。

著作権に関する情報

商標に関する情報