



Red Hat OpenShift Service sur AWS avec FSxN

NetApp Solutions

NetApp
December 19, 2024

Sommaire

- Red Hat OpenShift Service sur AWS avec FSxN..... 1
- Red Hat OpenShift Service sur AWS avec NetApp ONTAP 1
- Red Hat OpenShift Service sur AWS avec NetApp ONTAP..... 11

Red Hat OpenShift Service sur AWS avec FSxN

Red Hat OpenShift Service sur AWS avec NetApp ONTAP

Présentation

Dans cette section, nous allons apprendre à utiliser FSX pour ONTAP en tant que couche de stockage persistant pour les applications qui s'exécutent sur ROSA. Il présente l'installation du pilote NetApp Trident CSI sur un cluster ROSA, le provisionnement d'un système de fichiers FSX pour ONTAP et le déploiement d'un exemple d'application avec état. Il présente également des stratégies de sauvegarde et de restauration des données de votre application. Cette solution intégrée vous permet d'établir une structure de stockage partagée qui s'adapte facilement aux zones de disponibilité, simplifiant ainsi les processus d'évolutivité, de protection et de restauration de vos données à l'aide du pilote Trident CSI.

Prérequis

- ["Compte AWS"](#)
- ["Un compte Red Hat"](#)
- Utilisateur IAM ["avec les autorisations appropriées"](#) pour créer et accéder au cluster ROSA
- ["CLI AWS"](#)
- ["CLI ROSA"](#)
- ["Interface de ligne de commandes OpenShift" \(oc\)](#)
- [Helm 3 "documentation"](#)
- ["UN CLUSTER HCP ROSA"](#)
- ["Accès à la console Web Red Hat OpenShift"](#)

Ce schéma présente le cluster ROSA déployé dans plusieurs zones de disponibilité. Les nœuds maîtres du cluster ROSA, les nœuds d'infrastructure sont dans le VPC de Red Hat, tandis que les nœuds worker sont dans un VPC du compte du client. Nous allons créer un système de fichiers FSX pour ONTAP au sein du même VPC et installer le pilote Trident dans le cluster ROSA, permettant ainsi à tous les sous-réseaux de ce VPC de se connecter au système de fichiers.



Configuration initiale

1. Provision de FSX pour NetApp ONTAP

Créez une FSX multi-AZ pour NetApp ONTAP dans le même VPC que le cluster ROSA. Il existe plusieurs façons de le faire. Les détails de la création de FSxN à l'aide d'une pile CloudFormation sont fournis

A. Clone le référentiel GitHub

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

B. Exécuter la pile CloudFormation exécutez la commande ci-dessous en remplaçant les valeurs des paramètres par vos propres valeurs :

```
$ cd rosa-fsx-netapp-ontap/fsx
```

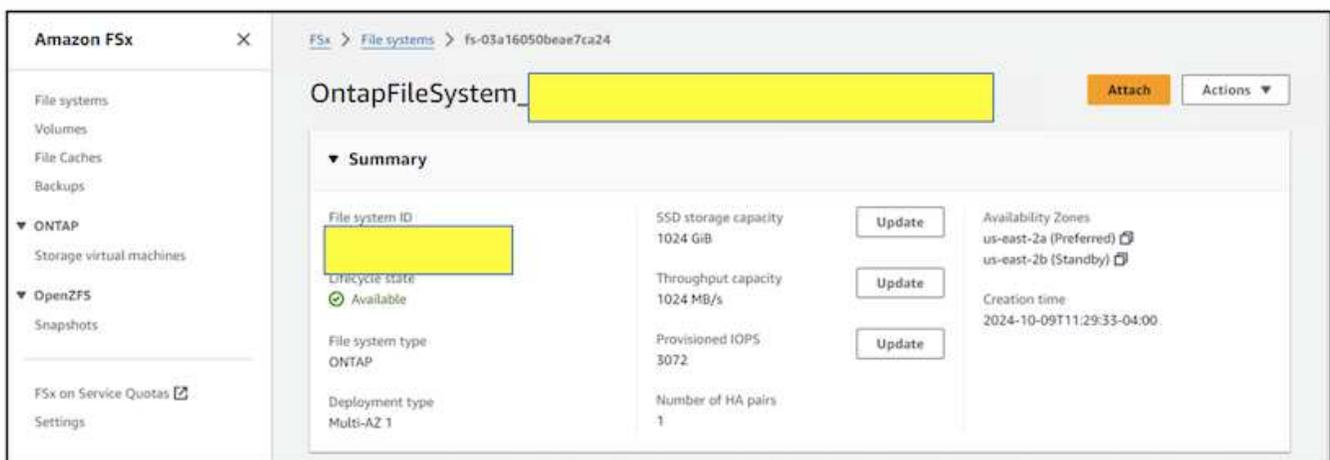
```

$ 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

```

Où : nom-région : identique à la région dans laquelle le cluster ROSA est déployé ID-sous : ID du sous-réseau préféré pour FSxN subnet2_ID : id du sous-réseau de secours pour FSxN ID-VPC : id du VPC où le cluster ROSA est déployé routetable1_ID, routetable2_id : id du sous-réseau de secours pour les tables de contrôle CIDR associées aux sous-réseaux ONTAP sélectionnés. Vous pouvez utiliser 0.0.0.0/0 ou tout autre CIDR approprié pour autoriser tout le trafic à accéder aux ports spécifiques de FSX pour ONTAP. Define Admin password: Un mot de passe pour se connecter à FSxN define SVM password: Un mot de passe pour se connecter à SVM qui sera créé.

Vérifier que votre système de fichiers et votre machine virtuelle de stockage (SVM) ont été créés à l'aide de la console Amazon FSX, voir ci-dessous :



2. installer et configurer le pilote Trident CSI pour le cluster ROSA

A. Ajouter le référentiel Trident Helm

```

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

```

B. installer Trident à l'aide de Helm

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



Selon la version que vous installez, le paramètre de version doit être modifié dans la commande affichée. Reportez-vous au pour connaître le "[documentation](#)" numéro de version correct. Pour des méthodes supplémentaires d'installation de Trident, reportez-vous au Trident "[documentation](#)".

C. Vérifiez que tous les modules Trident sont à l'état d'exécution

```
[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. Configurez le backend Trident CSI pour utiliser FSX for ONTAP (NAS ONTAP)

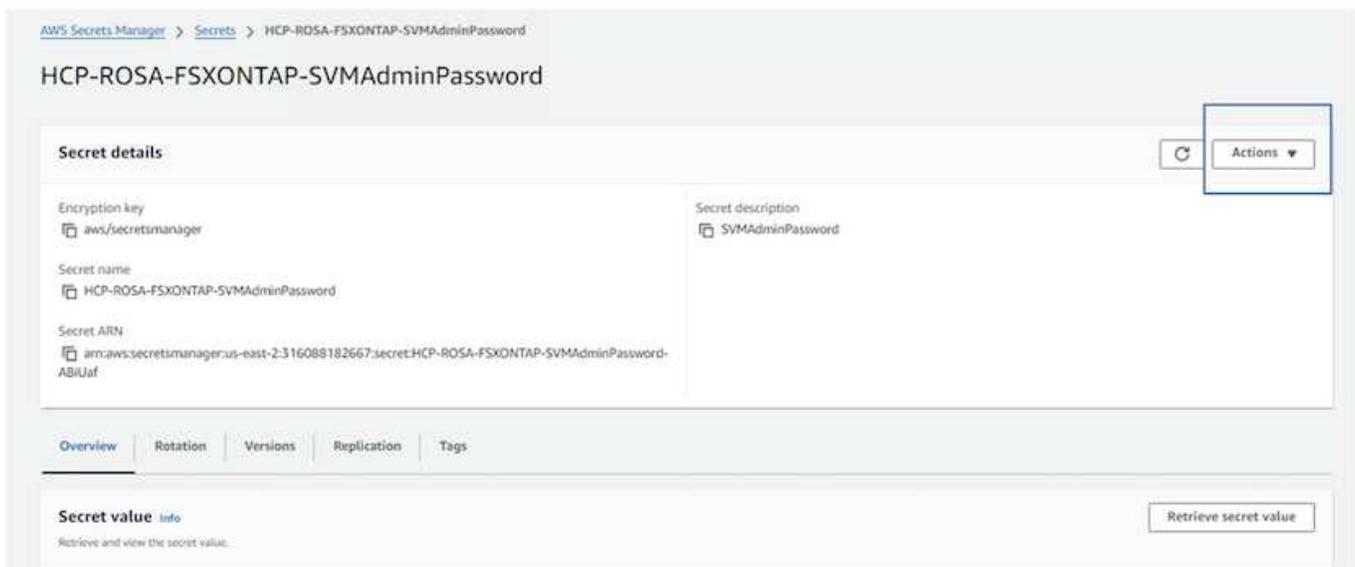
La configuration interne de Trident indique à Trident comment communiquer avec le système de stockage (dans ce cas, FSX pour ONTAP). Pour la création du back-end, nous fournirons les informations d'identification de la machine virtuelle de stockage à connecter, ainsi que les interfaces de données Cluster Management et NFS. Nous utiliserons "[pilote ontap-nas](#)" pour provisionner des volumes de stockage dans le système de fichiers FSX.

a. Tout d'abord, créer un secret pour les informations d'identification du SVM à l'aide du yml suivant

```
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>
```



Vous pouvez également récupérer le mot de passe de SVM créé pour FSxN à partir d’AWS secrets Manager, comme indiqué ci-dessous.



B.Suivant, ajouter le secret pour les informations d’identification du SVM au cluster ROSA en utilisant la commande suivante

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

Vous pouvez vérifier que le secret a été ajouté dans l’espace de noms Trident à l’aide de la commande suivante

```
$ 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. Ensuite, créez l'objet **backend** pour cela, déplacez-vous dans le répertoire **fsx** de votre référentiel Git cloné. Ouvrez le fichier **backend-ONTAP-nas.yaml**. Remplacer ce qui suit : **managementLIF** par le nom DNS de gestion **dataLIF** par le nom DNS NFS du SVM Amazon FSX et **svm** par le nom du SVM. Créez l'objet back-end à l'aide de la commande suivante.

Créez l'objet back-end à l'aide de la commande suivante.

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



Vous pouvez obtenir le nom DNS de gestion, le nom DNS NFS et le nom du SVM depuis la console Amazon FSX, comme indiqué dans la capture d'écran ci-dessous

The screenshot shows the Amazon FSx console interface. On the left, there is a navigation menu with options like File systems, Volumes, File Caches, Backups, ONTAP, OpenZFS, Snapshots, FSx on Service Quotas, and Settings. The main content area is titled 'Summary' and displays various details for an SVM. The SVM ID is highlighted with a blue box: svm-07a733da2584f2045. Other details include Creation time (2024-10-09T11:31:46-04:00), Lifecycle state (Created), Subtype (DEFAULT), and File system ID (fs-03a16050beae7ca24). Below the Summary section, there are tabs for Endpoints, Administration, Volumes, and Tags. The Endpoints section is expanded, showing Management DNS name, NFS DNS name, Management IP address, NFS IP address, and iSCSI IP addresses. The Management DNS name and NFS DNS name are highlighted with a blue box: svm-07a733da2584f2045.fs-03a16050beae7ca24.fsx.us-east-2.amazonaws.com. The Management IP address is 198.19.255.182, the NFS IP address is 198.19.255.182, and the iSCSI IP addresses are 10.10.9.32 and 10.10.26.28.

d. Maintenant, exécutez la commande suivante pour vérifier que l'objet back-end a été créé et que la phase affiche lié et que l'état est réussite

```
[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. Créer une classe de stockage maintenant que le backend Trident est configuré, vous pouvez créer une classe de stockage Kubernetes pour utiliser le back-end. Classe de stockage est un objet de ressource mis à disposition du cluster. Il décrit et classe le type de stockage que vous pouvez demander pour une application.

a. Passez en revue le fichier Storage-class-csi-nas.yaml dans le dossier 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. Créez une classe de stockage dans le cluster ROSA et vérifiez que la classe de stockage Trident-csi a été créée.

```
[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]#
```

Ceci termine l'installation du pilote Trident CSI et sa connectivité au système de fichiers FSX for ONTAP. Vous pouvez désormais déployer un exemple d'application avec état PostgreSQL sur ROSA à l'aide de volumes de fichiers sur FSX pour ONTAP.

c. Vérifiez qu'il n'y a pas de demandes de volume persistant ni de volumes persistants créés à l'aide de la classe de stockage Trident-csi.

```

[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-79949aef-e00d-4d9a-8b54-514ed85fbab2 100Gi RWO gp3-csi <unset> 2d16h
openshift-visualization-os-images centos-stream9-aae11cd5a1 Bound pvc-96b01a44-cb3f-449b-bd7d-39d020499c16 30Gi RWO gp3-csi <unset> 24h
openshift-visualization-os-images centos-stream9-d0244a141a44 Bound pvc-8230e84a-c5e8-452b-bf90-10e04fe102c1 30Gi RWO gp3-csi <unset> 44h
openshift-visualization-os-images fedora-21a0f3e638cd Bound pvc-64f375a8-d377-456d-83a0-268e413ae79c 30Gi RWO gp3-csi <unset> 44h
openshift-visualization-os-images rhel8-0652df0eb359 Bound pvc-2dc6de48-5916-411e-0cb3-99598f50be4c 30Gi RWO gp3-csi <unset> 44h
openshift-visualization-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-0cb3-99598f50be4c 30Gi RWO Delete Bound openshift-visualization-os-images/rhel8-0652df0eb359 gp3-csi <unset>
pvc-64f375a8-d377-456d-83a0-268e413ae79c 30Gi RWO Delete Bound openshift-visualization-os-images/fedora-21a0f3e638cd gp3-csi <unset>
pvc-74949aef-e00d-4d9a-8b54-514ed85fbab2 100Gi RWO Delete Bound openshift-monitoring/prometheus-data-prometheus-k8s-1 gp3-csi <unset>
pvc-8230e84a-c5e8-452b-bf90-10e04fe102c1 30Gi RWO Delete Bound openshift-visualization-os-images/centos-stream9-d0244a141a44 gp3-csi <unset>
pvc-9a4553a5-07e9-440a-8a90-99e384c97624 100Gi RWO Delete Bound openshift-monitoring/prometheus-data-prometheus-k8s-0 gp3-csi <unset>
pvc-96b01a44-cb3f-449b-bd7d-39d020499c16 30Gi RWO Delete Bound openshift-visualization-os-images/centos-stream9-aae11cd5a1 gp3-csi <unset>
pvc-f4374ce7-568d-4afc-b635-0228cf4544d4 30Gi RWO Delete Bound openshift-visualization-os-images/rhel9-2521bd116e64 gp3-csi <unset>
[root@localhost hcp-testing]#

```

d. Vérifiez que les applications peuvent créer des PV à l'aide de Trident CSI.

Créez un PVC à l'aide du fichier pvc-Trident.yaml fourni dans le dossier **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["Créer un PVC test à l'aide de Trident"]

```

5. Déployer un exemple d'application avec état PostgreSQL

a. Utilisez Helm pour installer 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. Vérifiez que le pod d'application est en cours d'exécution et qu'un PVC et un PV sont créés pour l'application.

```

[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                                     4h20m

```

c. Déployer un client PostgreSQL

Utilisez la commande suivante pour obtenir le mot de passe du serveur postgresql installé.

```

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

```

Utilisez la commande suivante pour exécuter un client postgresql et vous connecter au serveur en utilisant le mot de passe

```
$ 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/bitnami/postgresql:16.2.0-debian-11-r1 --env="PGPASSWORD=$POSTGRES_PASSWORD" \  
> --command -- psql --host postgresql -U postgres -d postgres -p 5432  
Warning: would violate PodSecurity "restricted:vl.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityContext.allowPrivilegeEscalation to true), runAsNonRoot != true (pod or container "postgresql-client" must set securityContext.runAsNonRoot to 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.
```

d. Créez une base de données et une table. Créez un schéma pour la table et insérez 2 lignes de données dans la table.

```
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 | first_name | last_name
-----+-----+-----
  1 | John       | Doe
  2 | Jane       | Scott
(2 rows)

```

Red Hat OpenShift Service sur AWS avec NetApp ONTAP

Ce document explique comment utiliser NetApp ONTAP avec le service Red Hat OpenShift sur AWS (ROSA).

Créer un Snapshot de volume

1. Créer un instantané du volume d'app dans cette section, nous allons montrer comment créer un instantané Trident du volume associé à l'app. Il s'agit d'une copie ponctuelle des données d'app. En cas de perte des données de l'application, nous pouvons récupérer les données à partir de cette copie instantanée. REMARQUE : ce snapshot est stocké dans le même agrégat que le volume d'origine dans ONTAP (sur site ou dans le cloud). Par conséquent, en cas de perte de l'agrégat de stockage ONTAP, nous ne pouvons pas restaurer les données d'application à partir de son snapshot.

****a.** Créez un VolumeSnapshotClass Enregistrez le manifeste suivant dans un fichier appelé volume-snapshot-class.yaml

```

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

```

Créez un snapshot à l'aide du manifeste ci-dessus.

```

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

```

b. Ensuite, créez un snapshot Créez un snapshot de la demande de volume existante en créant VolumeSnapshot pour créer une copie ponctuelle de vos données PostgreSQL. Cela crée un snapshot FSX qui ne prend quasiment pas d'espace dans le système de fichiers back-end. Enregistrez le manifeste suivant

dans un fichier appelé 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. Créer l'instantané de volume et confirmer qu'il est créé

Supprimer la base de données pour simuler la perte de données (la perte de données peut se produire pour diverses raisons, nous la simulons simplement en supprimant la base de données)

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc create -f postgresql-volume-snapshot.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. Supprimer la base de données pour simuler la perte de données (la perte de données peut se produire pour diverses raisons, nous la simulons simplement en supprimant la base de données)

```
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=#
```

Restaurer à partir d'une copie Snapshot de volume

1. Restaurer à partir de l'instantané dans cette section, nous allons montrer comment restaurer une application à partir de l'instantané Trident du volume d'application.

a. Créer un clone de volume à partir de l'instantané

Pour restaurer le volume à son état précédent, vous devez créer une nouvelle demande de volume persistant basée sur les données de l'instantané que vous avez pris. Pour ce faire, enregistrez le manifeste suivant dans un fichier nommé `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
```

Créez un clone du volume en créant une demande de volume persistant en utilisant le snapshot comme source à l'aide du manifeste ci-dessus. Appliquez le manifeste et assurez-vous que le clone est créé.

```
[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. Supprimez l'installation postgresql d'origine

```
[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. Créez une nouvelle application postgresql en utilisant le nouveau clone PVC

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

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

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

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

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

To get the password for "postgres" run:

    export POSTGRES_PASSWORD=$(kubectl get secret --namespace postgresql postgresql -o jsonpath="{.data.postgres-password}" | 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
    --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 through
password, 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 prod
ing to your workload needs:
- primary.resources
- readReplicas.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
[root@localhost hcp-testing]#
```

d. Vérifiez que le pod d'application est à l'état d'exécution

```
[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. Vérifiez que le pod utilise le clone comme 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      Activat  
Normal    Started          3m42s  kubelet        Started container postgresql      Go to Set  
[root@localhost hcp-testing]#
```

f) pour vérifier que la base de données a été restaurée comme prévu, revenez à la console du conteneur et affichez les bases de données existantes

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:16.2 --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.allowPrivilegeEscalation to true), runAsNonRoot != true (pod or container "postgresql-client" must set securityContext.runAsNonRoot to 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 |  |  | =c/postgres +
       |          |      |      |              |              |  |  | postgres=Ctc/postgres
 postgres | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 |  |  | =c/postgres +
       |          |      |      |              |              |  |  | postgres=Ctc/postgres
 template0 | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 |  |  |
 template1 | postgres | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 |  |  |
(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)
```

Vidéo de démonstration

[Amazon FSX pour NetApp ONTAP avec Red Hat OpenShift Service sur AWS à l'aide d'Hosted Control plane](#)

D'autres vidéos sur les solutions Red Hat OpenShift et OpenShift sont disponibles ["ici"](#).

Informations sur le copyright

Copyright © 2024 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

Les logiciels dérivés des éléments NetApp protégés par copyright sont soumis à la licence et à l'avis de non-responsabilité suivants :

CE LOGICIEL EST FOURNI PAR NETAPP « EN L'ÉTAT » ET SANS GARANTIES EXPRESSES OU TACITES, Y COMPRIS LES GARANTIES TACITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN USAGE PARTICULIER, QUI SONT EXCLUES PAR LES PRÉSENTES. EN AUCUN CAS NETAPP NE SERA TENU POUR RESPONSABLE DE DOMMAGES DIRECTS, INDIRECTS, ACCESSOIRES, PARTICULIERS OU EXEMPLAIRES (Y COMPRIS L'ACHAT DE BIENS ET DE SERVICES DE SUBSTITUTION, LA PERTE DE JOUISSANCE, DE DONNÉES OU DE PROFITS, OU L'INTERRUPTION D'ACTIVITÉ), QUELLES QU'EN SOIENT LA CAUSE ET LA DOCTRINE DE RESPONSABILITÉ, QU'IL S'AGISSE DE RESPONSABILITÉ CONTRACTUELLE, STRICTE OU DÉLICTELLE (Y COMPRIS LA NÉGLIGENCE OU AUTRE) DÉCOULANT DE L'UTILISATION DE CE LOGICIEL, MÊME SI LA SOCIÉTÉ A ÉTÉ INFORMÉE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NetApp se réserve le droit de modifier les produits décrits dans le présent document à tout moment et sans préavis. NetApp décline toute responsabilité découlant de l'utilisation des produits décrits dans le présent document, sauf accord explicite écrit de NetApp. L'utilisation ou l'achat de ce produit ne concède pas de licence dans le cadre de droits de brevet, de droits de marque commerciale ou de tout autre droit de propriété intellectuelle de NetApp.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevets américains, étrangers ou par une demande en attente.

LÉGENDE DE RESTRICTION DES DROITS : L'utilisation, la duplication ou la divulgation par le gouvernement sont sujettes aux restrictions énoncées dans le sous-paragraphe (b)(3) de la clause Rights in Technical Data-Noncommercial Items du DFARS 252.227-7013 (février 2014) et du FAR 52.227-19 (décembre 2007).

Les données contenues dans les présentes se rapportent à un produit et/ou service commercial (tel que défini par la clause FAR 2.101). Il s'agit de données propriétaires de NetApp, Inc. Toutes les données techniques et tous les logiciels fournis par NetApp en vertu du présent Accord sont à caractère commercial et ont été exclusivement développés à l'aide de fonds privés. Le gouvernement des États-Unis dispose d'une licence limitée irrévocable, non exclusive, non cessible, non transférable et mondiale. Cette licence lui permet d'utiliser uniquement les données relatives au contrat du gouvernement des États-Unis d'après lequel les données lui ont été fournies ou celles qui sont nécessaires à son exécution. Sauf dispositions contraires énoncées dans les présentes, l'utilisation, la divulgation, la reproduction, la modification, l'exécution, l'affichage des données sont interdits sans avoir obtenu le consentement écrit préalable de NetApp, Inc. Les droits de licences du Département de la Défense du gouvernement des États-Unis se limitent aux droits identifiés par la clause 252.227-7015(b) du DFARS (février 2014).

Informations sur les marques commerciales

NETAPP, le logo NETAPP et les marques citées sur le site <http://www.netapp.com/TM> sont des marques déposées ou des marques commerciales de NetApp, Inc. Les autres noms de marques et de produits sont des marques commerciales de leurs propriétaires respectifs.