使用FSxN在AWS上运行Red Hat OpenShift服务

NetApp Solutions

NetApp
December 19, 2024

目录

使用FSxN在AWS上运行Red Hat OpenShift服务

借助NetApp ONTAP在AWS上运行Red Hat OpenShift服务

概述

在本节中、我们将介绍如何将FSx for ONTAP用作ROSA上运行的应用程序的永久性存储层。其中将显示在ROSA集群上安装NetApp Trident CSI驱动程序、配置FSx for ONTAP文件系统以及部署有状态应用程序示例的过程。同时、还会显示备份和还原应用程序数据的策略。借助这款集成解决方案、您可以建立一个共享存储框架、轻松地跨多个应用程序进行扩展、从而简化使用Trident CSI驱动程序扩展、保护和还原数据的过程。

前提条件

- "AWS 帐户"
- "Red Hat帐户"
- IAM用户"具有适当的权限"、用于创建和访问ROSA集群
- "AWS命令行界面"
- "罗莎命令行界面"
- "OpenShift命令行界面"(OC)
- Helm 3"文档。"
- "HCP ROSA集群"
- "访问Red Hat OpenShift Web控制台"

此图显示了部署在多个澳大利亚地区的ROSA集群。罗莎集群的主节点、基础架构节点位于Red Hat的VPC中，而工作节点位于客户帐户的VPC中。我们将在同一个VPC中创建一个FSx for ONTAP文件系统、并在ROSA集群中安装Trident驱动程序、从而允许此VPC的所有子网连接到文件系统。

初始设置

1.为**FSX**配置NetApp ONTAP

在与ROSA集群相同的VPC中创建适用于NetApp ONTAP的多可用性FSx。可通过多种方法实现此操作。提供了使用CloudFormation堆栈创建FSxN的详细信息

a.Clone the GitHub re듬 뢺 处

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

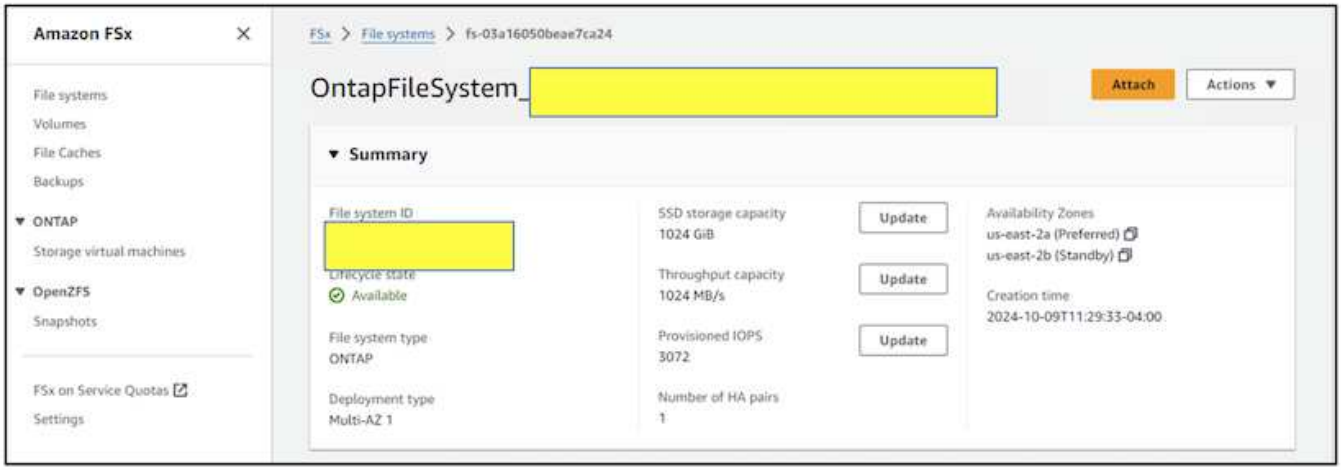
b.Run the CloudFormation Stack 通过将参数值替换为您自己的值来运行以下命令：

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

```
$ aws cloudformation create-stack \
  --stack-name ROSA-FSXONTAP \
  --template-body file://./FSxONTAP.yaml \
  --region <region-name> \
  --parameters \
  ParameterKey=Subnet1ID,ParameterValue=[subnet1_ID] \
  ParameterKey=Subnet2ID,ParameterValue=[subnet2_ID] \
  ParameterKey=myVpc,ParameterValue=[VPC_ID] \
ParameterKey=FSxONTAPRouteTable,ParameterValue=[routetable1_ID,routetable2
_ID] \
  ParameterKey=FileSystemName,ParameterValue=ROSA-myFSxONTAP \
  ParameterKey=ThroughputCapacity,ParameterValue=1024 \
  ParameterKey=FSxAllowedCIDR,ParameterValue=[your_allowed_CIDR] \
  ParameterKey=FsxAdminPassword,ParameterValue=[Define Admin password] \
  ParameterKey=SvmAdminPassword,ParameterValue=[Define SVM password] \
  --capabilities CAPABILITY_NAMED_IAM
```

其中：region-name：sas the region where the ROSA cluster is才会部署subnet1_ID：FSxN subnet2_ID的首选子网的ID：FSxN的备用子网的ID VPC_ID：部署ROSA集群的ONTAP的VPC的ID routetable1_ID 、routetable2_ID：与所选CIDR规则关联的路由表的ID：允许对所选Sx访问的子网进行控制。您可以使用0.0.0.0/0或任何适当的CIDR允许所有流量访问FSx for ONTAP的特定端口。定义管理员密码：用于登录到FSxN的密码定义SVM密码：用于登录到要创建的SVM的密码。

验证是否已使用Amazon FSx控制台创建文件系统和Storage Virtual Machine (SVM)、如下所示：



2.为ROSA群集安装和配置Trident CSI驱动程序

a.Add the Trident Helm re旨 在处

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

b.Install Trident using Helm

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

根据您安装的版本、需要在显示的命令中更改version参数。有关正确的版本号、请参见"文档。"。有关安装Trident的其他方法，请参阅《Trident》"文档。"。

c.Verify that all Trident Pod are in the running state

```
[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]#
```

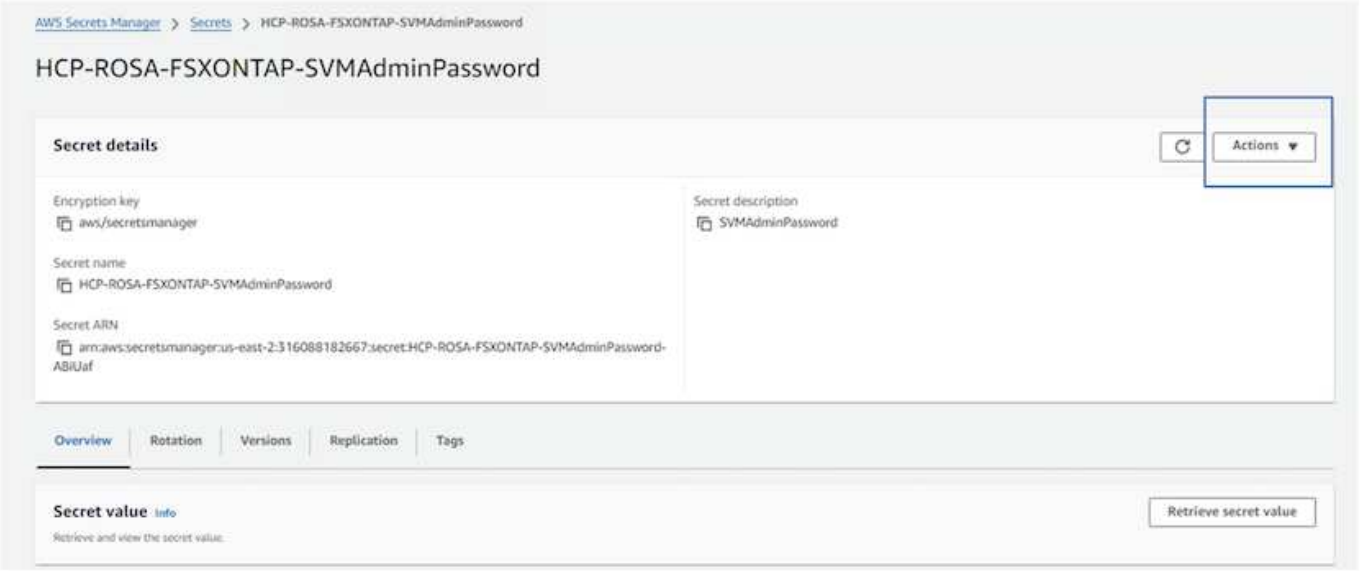
3.配置Trident CSI后端以使用FSx for ONTAP (ONTAP NAS)

Trident后端配置指示Trident如何与存储系统通信(在本示例中为FSx for ONTAP)。要创建后端、我们将提供要连接到的Storage Virtual Machine的凭据、以及集群管理和NFS数据接口。我们将使用"ontap-NAS 驱动程序"在FSx文件系统中配置存储卷。

**…首先、使用以下YAML"为SVM凭据创建一个密钥

```
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>
```

您也可以从AWS机密管理器检索为FSxN创建的SVM密码、如下所示。

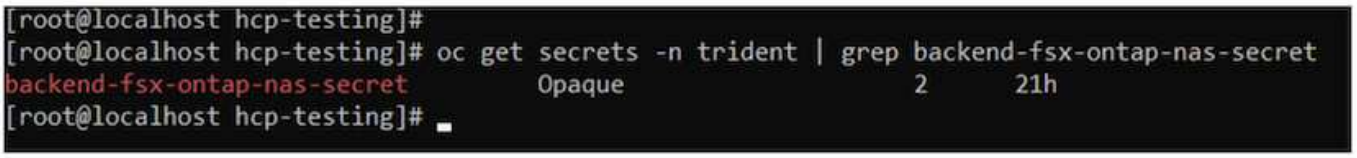


b.Next，使用以下命令将**SVM**凭据的密钥添加到**ROSA**集群中

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

您可以使用以下命令验证是否已在Trident命名空间中添加此密钥

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

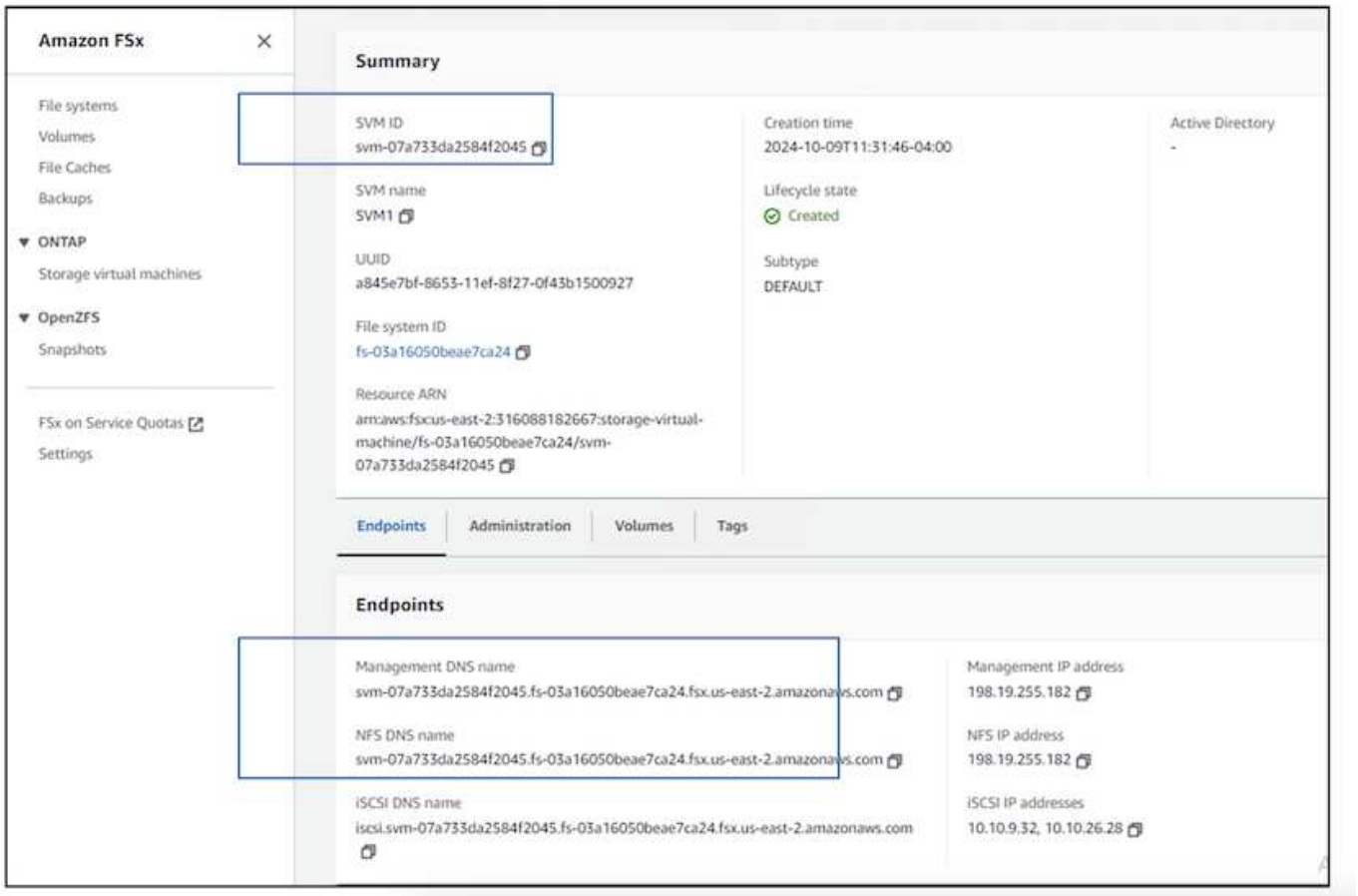


- C.接下来，创建后端对象为此，移至克隆的**Git**储存库的FSX目录。打开文件**backend-ams-naS.yaml ONTAP**。将以下内容： ManagementLIF替换为管理**DNS**名称 dataLIF，替换为**Amazon FSx SVM**的**NFS DNS**名称，并将 SVM**替换为SVM名称。使用以下命令创建后端对象。

使用以下命令创建后端对象。

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

您可以从Amazon FSx控制台获取管理DNS名称、NFS DNS名称和SVM名称、如以下屏幕截图所示



*现在，运行以下命令以验证是否已创建后端对象，并且**Phase (**阶段)显示**bound and Status (**绑定)为**Success (**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.创建存储类配置Trident后端后，您可以创建一个Kubbernetes存储类以使用后端。存储类是可供集群使用的资源对象。它介绍并分类您可以为应用程序请求的存储类型。

…查看**FSx**文件夹中的**storage-class-CSI－NAS．yaml**文件。

```
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集群中创建存储类、并验证是否Trident已创建ROSA-CSI存储类。**

```
[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]#
```

至此、Trident CSI驱动程序的安装完成、并完成了它与FSx for ONTAP文件系统的连接。现在、您可以使用FSx for ONTAP上的文件卷在ROSA上部署示例PostgreSQL有状态应用程序。

- C.确认没有使用PVC-sI存储类创建Trident和PV。**

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

*验证应用程序是否可以使用**Trident Csi.**创建PV

使用**FSX**文件夹中提供的PVC-AML.YAML文件创建Trident。

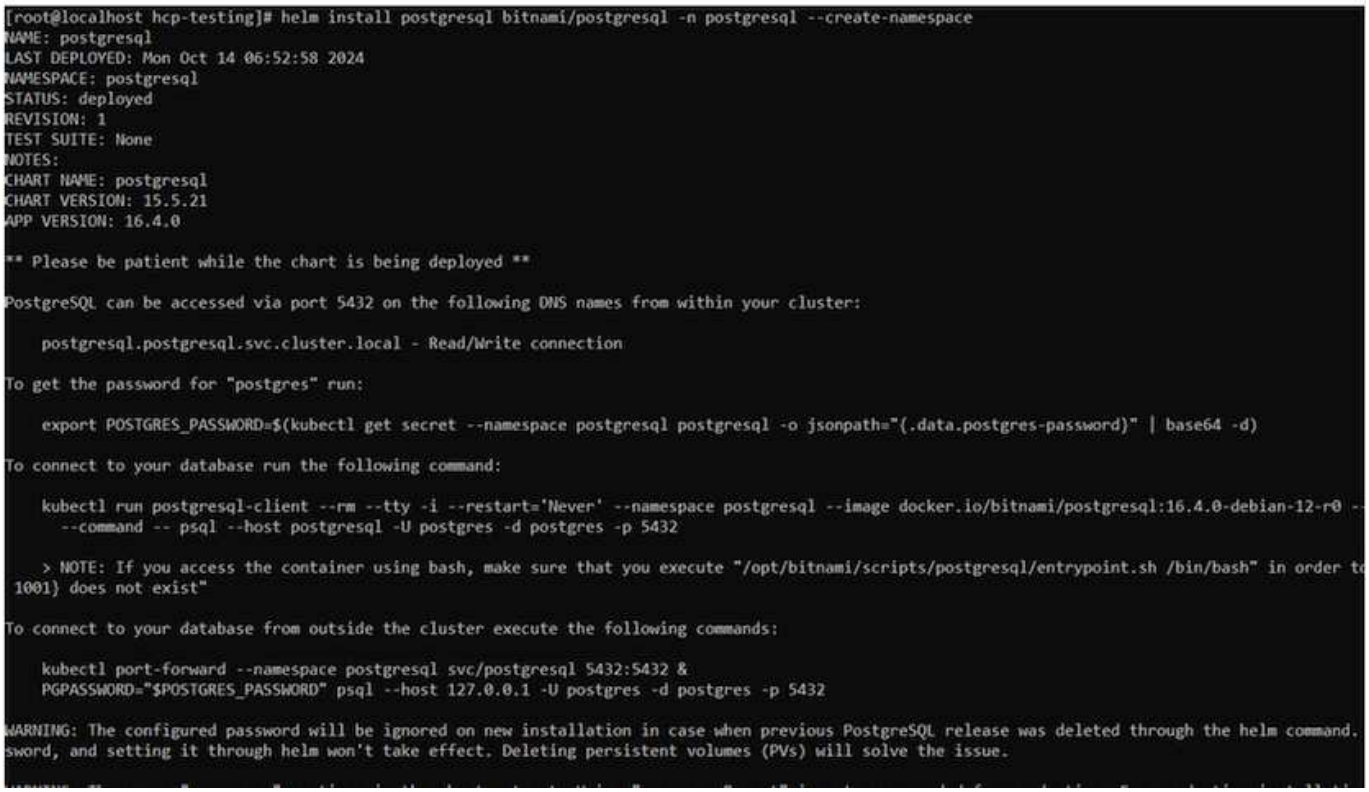
```
pvc-trident.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: basic
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: trident-csi
```

```
 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创建测试PVC"]
```

5.部署示例**PostgreSQL**有状态应用程序

…使用**Helm**安装**PostgreSQL**

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



- b.确认应用程序POD正在运行，并且为应用程序创建了PVC和PV。**

- C.部署PostgreSQL客户机**

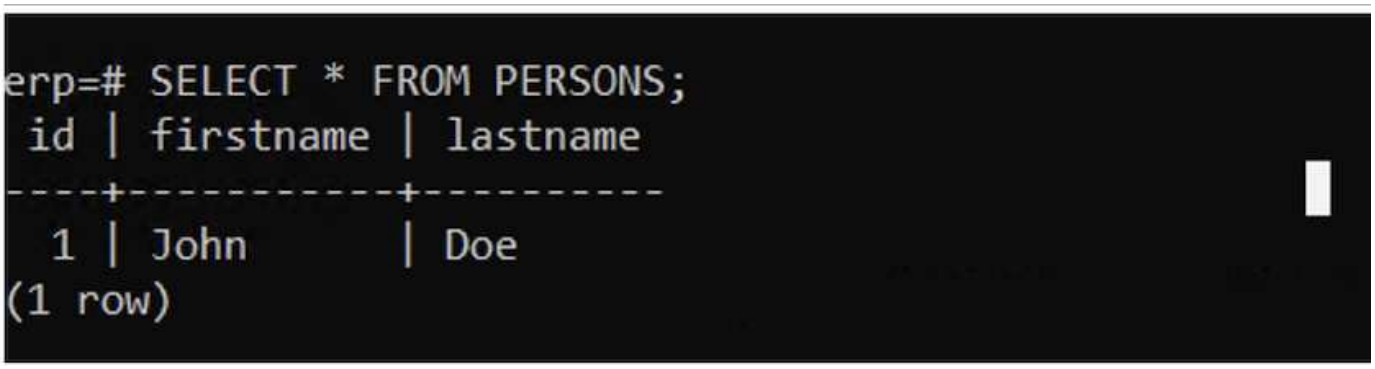
使用以下命令获取已安装的**PostgreSQL**服务器的口令。

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

使用以下命令运行**PostgreSQL**客户机，并使用口令连接到服务器

```
$ 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
```

*创建数据库和表。创建表的纲要并将**2**行数据插入表中。



```
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)
```

借助NetApp ONTAP在AWS上运行Red Hat OpenShift服务

本文档将概述如何将NetApp ONTAP与基于AWS的Red Hat OpenShift服务(ROSA)结合使用。

创建卷快照

1.创建应用程序卷的快照在本节中，我们将介绍如何创建与应用程序关联的卷的Trident快照。这将是应用程序数据的时间点副本。如果应用程序数据丢失、我们可以从此时间点副本恢复数据。注意：此快照与ONTAP中的原始卷存储在同一聚合中(内部或云中)。因此、如果ONTAP存储聚合丢失、我们将无法从其快照中恢复应用程序数据。

**…创建卷快照类将以下清单保存在名为volume-snapshot-class.yaml的文件中

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

使用上述清单创建快照。

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

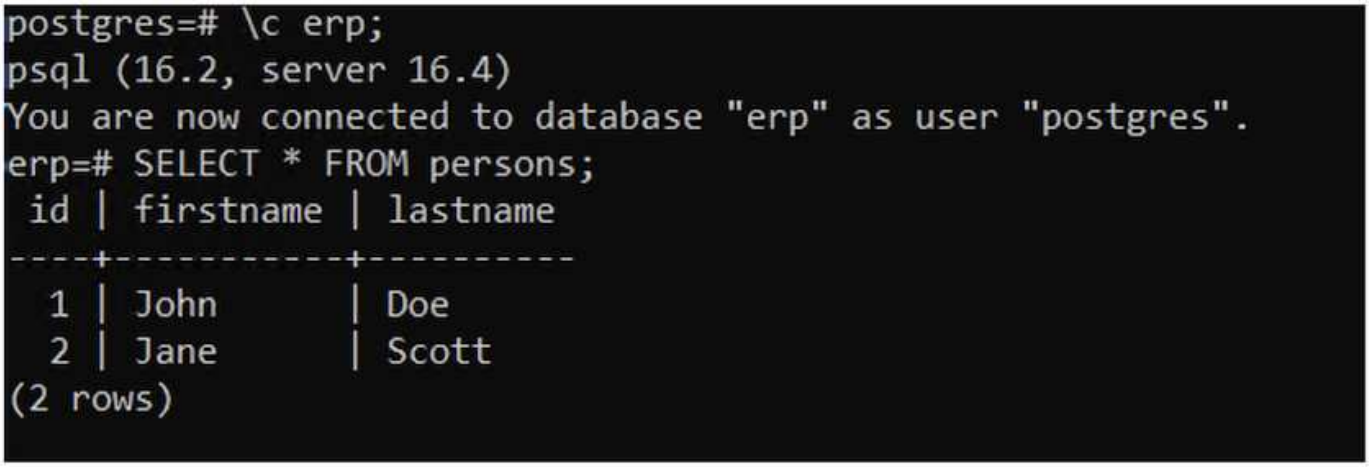
- b.接下来，创建快照**通过创建卷快照创建现有PVC的快照，以创建PostgreSQL数据的时间点副本。这将创建FSx快照、该快照在文件系统后端几乎不占用任何空间。将以下清单保存在名为volume-Snapshot．yaml的文件中：

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

- C.创建卷快照并确认已创建**

删除数据库以模拟数据丢失(由于各种原因可能会发生数据丢失、此处我们只是通过删除数据库来模拟它)

*删除数据库以模拟数据丢失(由于各种原因可能会发生数据丢失，此处我们只是通过删除数据库来模拟它)



从卷快照还原

1.从**Snap**照 恢复在本节中，我们将介绍如何从应用程序卷的Trident快照恢复应用程序。

...从快照创建卷克隆

要将卷还原到其先前状态、您必须根据创建的快照中的数据创建一个新的PVC。为此、请将以下清单保存在名为pvC-CLONE .yaml的文件中

```
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
```

通过使用快照作为源并使用上述清单创建PVC来创建卷的克隆。应用清单并确保已创建克隆。

```
[root@localhost hcp-testing]# oc create -f postgresql-pvc-clone.yaml -n postgresql
persistentvolumeclaim/postgresql-volume-clone created
[root@localhost hcp-testing]# oc get pvc -n postgresql
NAME                     STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
data-postgresql-0        Bound    pvc-e3ddd9bd-e6a7-4a4a-b935-f1c090fd8db6   8Gi        RWO            trident-csi
postgresql-volume-clone  Bound    pvc-b38fbc54-55dc-47e8-934d-47f181fddac6   8Gi        RWO            trident-csi
[root@localhost hcp-testing]#
```

- b.删除初始PostgreSQL安装**

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

- C.使用新的克隆PVC**创建新的PostgreSQL应用程序

```
$ 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]#
```

*验证应用程序**POD**是否处于**running**状态

```
[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]#
```

验证**POD**是否使用克隆作为其**PVC**

```
[root@localhost hcp-testing]#
[root@localhost hcp-testing]# oc describe pod/postgresql-0 -n postgresql
```

```
  ContainersReady             True
  PodScheduled                True
Volumes:
  empty-dir:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
    SizeLimit:  <unset>
  dshm:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:     Memory
    SizeLimit:  <unset>
  data:
    Type:       PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName:  postgresql-volume-clone
    ReadOnly:   false
QoS Class:      Burstable
Node-Selectors: <none>
Tolerations:    node.kubernetes.io/memory-pressure:NoSchedule op=Exists
                node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason                  Age    From                     Message
  ----    ------                  ----   ----                     -------
  Normal  Scheduled               3m55s  default-scheduler        Successfully assigned postgresql/postgres
.us-east-2.compute.internal
  Normal  SuccessfulAttachVolume  3m54s  attachdetach-controller  AttachVolume.Attach succeeded for volume
8-934d-47f181fddac6"
  Normal  AddedInterface          3m43s  multus                   Add eth0 [10.129.2.126/23] from ovn-kuber
  Normal  Pulled                  3m43s  kubelet                  Container image "docker.io/bitnami/postgr
r0" already present on machine
  Normal  Created                 3m42s  kubelet                  Created container postgresql
  Normal  Started                 3m42s  kubelet                  Started container postgresql
[root@localhost hcp-testing]#
```

f)要验证数据库是否已按预期还原、请返回容器控制台并显示现有数据库

```
[root@localhost hcp-testing]# kubectl run postgresql-client --rm --tty -i --restart='Never' --namespace postgresql --image docker.io/bitnami/postgresql:1
$POSTGRES_PASSWORD"  --command -- psql --host postgresql -U postgres -d postgres -p 5432
Warning: would violate PodSecurity "restricted:v1.24": allowPrivilegeEscalation != false (container "postgresql-client" must set securityContext.allowPri
 capabilities (container "postgresql-client" must set securityContext.capabilities.drop=["ALL"]), runAsNonRoot != true (pod or container "postgresql-clie
Root=true), seccompProfile (pod or container "postgresql-client" must set securityContext.seccompProfile.type to "RuntimeDefault" or "Localhost")
If you don't see a command prompt, try pressing enter.

postgres=# \l
                                                List of databases
   Name    |  Owner   | Encoding | Locale Provider |   Collate   |    Ctype    | ICU Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+-------------+-------------+------------+-----------+-----------------------
 erp       | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 postgres  | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           |
 template0 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.UTF-8 | en_US.UTF-8 |            |           | =c/postgres          +
           |          |          |                 |             |             |            |           | postgres=CTc/postgres
(4 rows)

postgres=# \c erp;
psql (16.2, server 16.4)
You are now connected to database "erp" as user "postgres".
erp=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persons | table | postgres
(1 row)

erp=# SELECT * FROM PERSONS;
 id | firstname | lastname
----+-----------+----------
  1 | John      | Doe
  2 | Jane      | Scott
(2 rows)
```

演示视频

Amazon FSx for NetApp ONTAP使用托管控制平台在AWS上运行Red Hat OpenShift服务

有关Red Hat OpenShift和OpenShift解决方案的更多视频，请参见"此处"。