vSphere Kubernetes Service avec stockage NetApp

NetApp container solutions

NetApp
September 01, 2026

Sommaire

vSphere Kubernetes Service avec stockage NetApp

Découvrez le service Kubernetes de VMware vSphere avec NetApp ONTAP

VMware vSphere Kubernetes Service (VKS) est une offre Kubernetes managée de VMware qui permet aux organisations de déployer, gérer et faire évoluer des applications conteneurisées à l'aide de Kubernetes. Associé au stockage NetApp ONTAP, VKS offre une persistance, des performances et une gestion des données de niveau entreprise pour les charges de travail cloud-native.

VKS est intégré à VMware Cloud Foundation (VCF) et conforme à la norme CNCF Kubernetes en amont, assurant ainsi la compatibilité avec l'écosystème Kubernetes plus large et les outils associés.

Principales caractéristiques du service Kubernetes vSphere

1. **Clusters Kubernetes gérés :** vSphere Kubernetes Service simplifie le déploiement et la gestion des clusters Kubernetes. Il automatise des tâches telles que la création, la mise à niveau, le dimensionnement et la surveillance des clusters.
2. **Intégration avec l'infrastructure VMware :** VKS s'intègre parfaitement à VMware vSphere, NSX-T et autres solutions VMware, permettant aux organisations de tirer parti de leur infrastructure VMware existante pour les charges de travail Kubernetes.
3. **Prise en charge du multicloud et du cloud hybride :** vSphere Kubernetes Service prend en charge le déploiement sur des clouds privés, des clouds publics (par exemple, AWS, Azure, Google Cloud) et des environnements de cloud hybride, offrant ainsi aux organisations une flexibilité dans la gestion de leurs applications.
4. **Sécurité intégrée :** VKS inclut des fonctionnalités de sécurité telles que le contrôle d'accès basé sur les rôles (RBAC), l'application de politiques et l'intégration avec VMware NSX pour la sécurité du réseau.
5. **Prise en charge de l'écosystème Kubernetes :** VKS prend en charge l'intégralité de l'API Kubernetes en amont, garantissant ainsi la portabilité entre les environnements et la compatibilité avec l'écosystème Kubernetes au sens large, notamment avec des outils comme Istio pour le service mesh, Prometheus pour la supervision, et d'autres outils certifiés CNCF. Les organisations peuvent appliquer directement les compétences et intégrations Kubernetes standard aux clusters VKS.
6. **Outils conviviaux pour les développeurs :** VKS prend en charge les flux de travail standard des développeurs Kubernetes, notamment les pipelines CI/CD, les pratiques GitOps et les outils comme kubectl et Helm, permettant aux équipes de développement de créer et de déployer des applications conteneurisées sans avoir à apprendre des interfaces propriétaires.
7. **Évolutivité et haute disponibilité :** VKS offre des fonctionnalités telles que la mise à l'échelle automatique et la tolérance aux pannes pour garantir que les applications restent hautement disponibles et performantes.
8. **Observabilité et surveillance :** VKS s'intègre aux outils de surveillance standard compatibles avec Kubernetes pour fournir des informations en temps réel sur la performance de l'application.

Cas d'utilisation du service Kubernetes vSphere

1. **Modernisation des applications :** Les organisations peuvent moderniser leurs applications existantes en les conteneurisant et en les déployant sur des clusters Kubernetes gérés par VKS.
2. **DevOps Enablement :** VKS prend en charge les workflows DevOps en fournissant l'automatisation, l'intégration CI/CD et des outils adaptés aux développeurs.
3. **Déploiements de cloud hybride :** Les entreprises peuvent utiliser VKS pour déployer des clusters Kubernetes dans des environnements sur site et cloud, permettant ainsi des opérations cohérentes.
4. **Architecture de microservices :** VKS prend en charge le développement d'applications basé sur les microservices, permettant aux équipes de construire et de faire évoluer des systèmes distribués.

Premiers pas avec le stockage et le service Kubernetes vSphere

Installez un cluster vSphere Kubernetes Service

Installez un cluster vSphere Kubernetes Service (VKS) dans votre environnement VCF 9.1 et configurez les nœuds de travail avec les utilitaires de stockage appropriés pour vos charges de travail.

À propos de cette tâche

Les utilitaires NFS sont automatiquement installés sur les nœuds de travail du cluster que vous créez. Si vous souhaitez créer des nœuds de travail avec les utilitaires iSCSI ou NVMe installés, vous devez créer une image OVA personnalisée et utiliser cette image pour les nœuds de travail. L'image personnalisée peut être créée à l'aide de Docker, et l'outil de ligne de commandes `vcf` peut ensuite être utilisé pour créer un fichier OVA à partir de cette image. Ce fichier OVA peut être téléchargé dans la Content Library de vSphere. Lors de la création du cluster VKS, cette image de la Content Library peut être sélectionnée pour les pools de nœuds.

Étapes

1. Créer le cluster VKS :

   Vous pouvez créer un cluster VKS à l'aide de l'image de nœud de travail par défaut ou d'une image personnalisée que vous créez. L'image de nœud de travail par défaut inclut les utilitaires NFS préinstallés, tandis que l'image personnalisée peut inclure les utilitaires iSCSI et NVMe. Les options suivantes sont disponibles :

   - **NAS uniquement (par défaut) :** Créez un cluster VKS en utilisant l'image de nœud de travail par défaut, qui inclut des utilitaires NFS préinstallés.
   - **SAN activé (image personnalisée) :** Créez une image Docker personnalisée incluant les utilitaires iSCSI et NVMe, générez un fichier OVA à l'aide de l'outil en ligne de commande `vcf`, téléchargez l'OVA dans la vSphere Content Library, puis créez le cluster VKS en sélectionnant cette image personnalisée pour les pools de nœuds.

     ▼ Afficher les instructions pour créer un cluster de service Kubernetes NAS uniquement vSphere avec l'image de nœud de travail par défaut

     Créez le cluster VKS à l'aide de l'image de nœud de travail par défaut. Suivez les instructions pour spécifier le nom du cluster, la configuration du pool de nœuds et les autres paramètres.

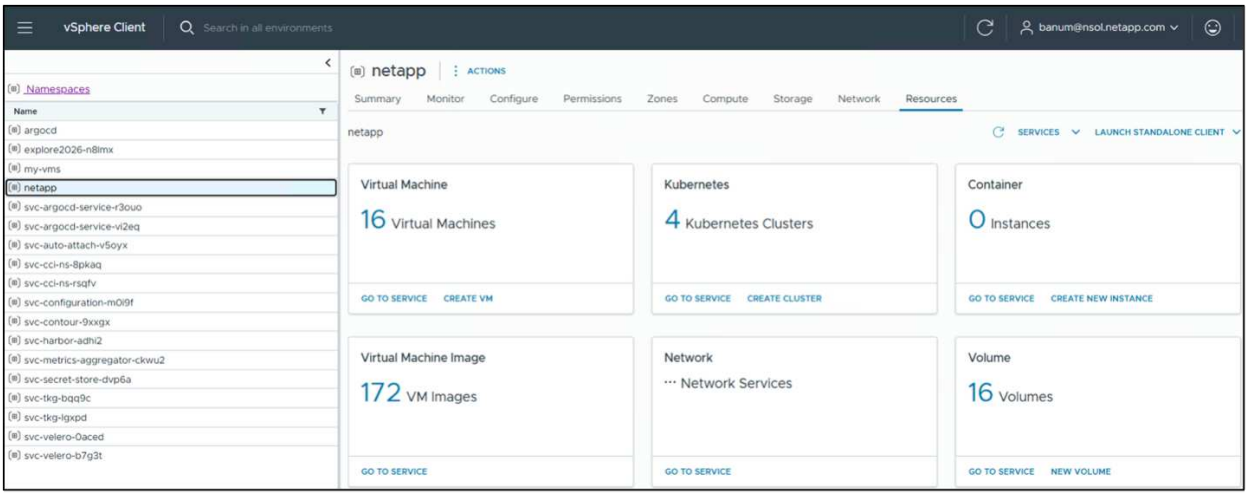
     Depuis le client vSphere, accédez à l'espace de noms où vous souhaitez créer le cluster Kubernetes vSphere. Dans l'onglet **Ressources** de cet espace de noms, cliquez sur **Créer un**

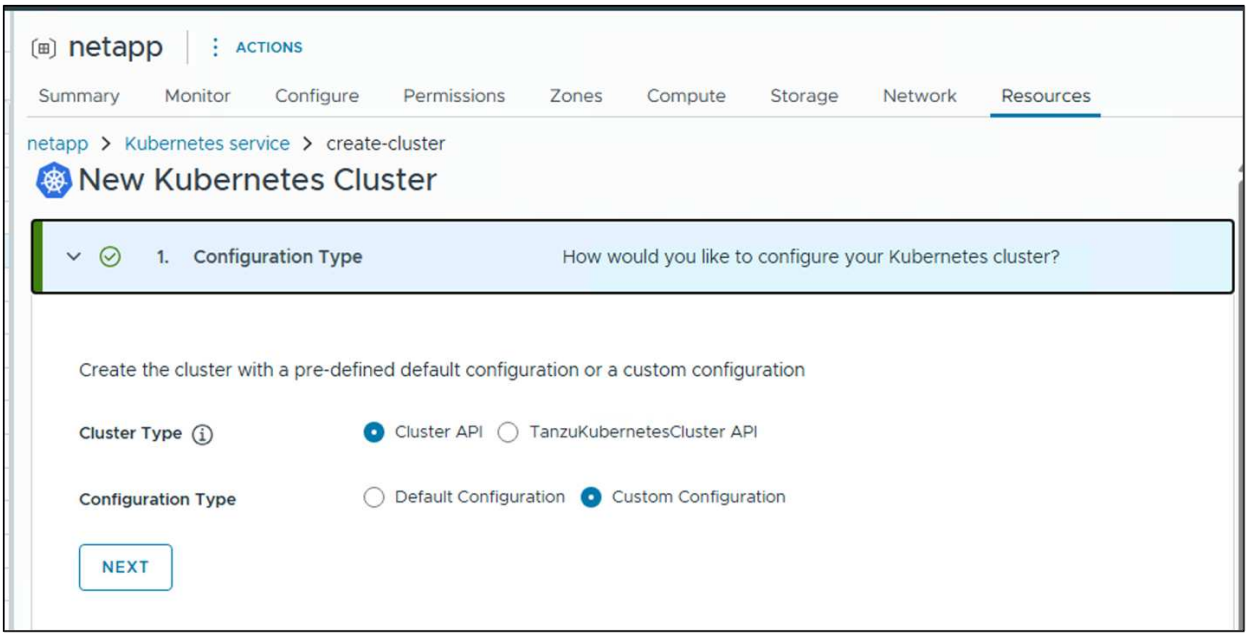
cluster dans la section Kubernetes, ou cliquez sur **Accéder au service** puis sur **Créer un cluster**.

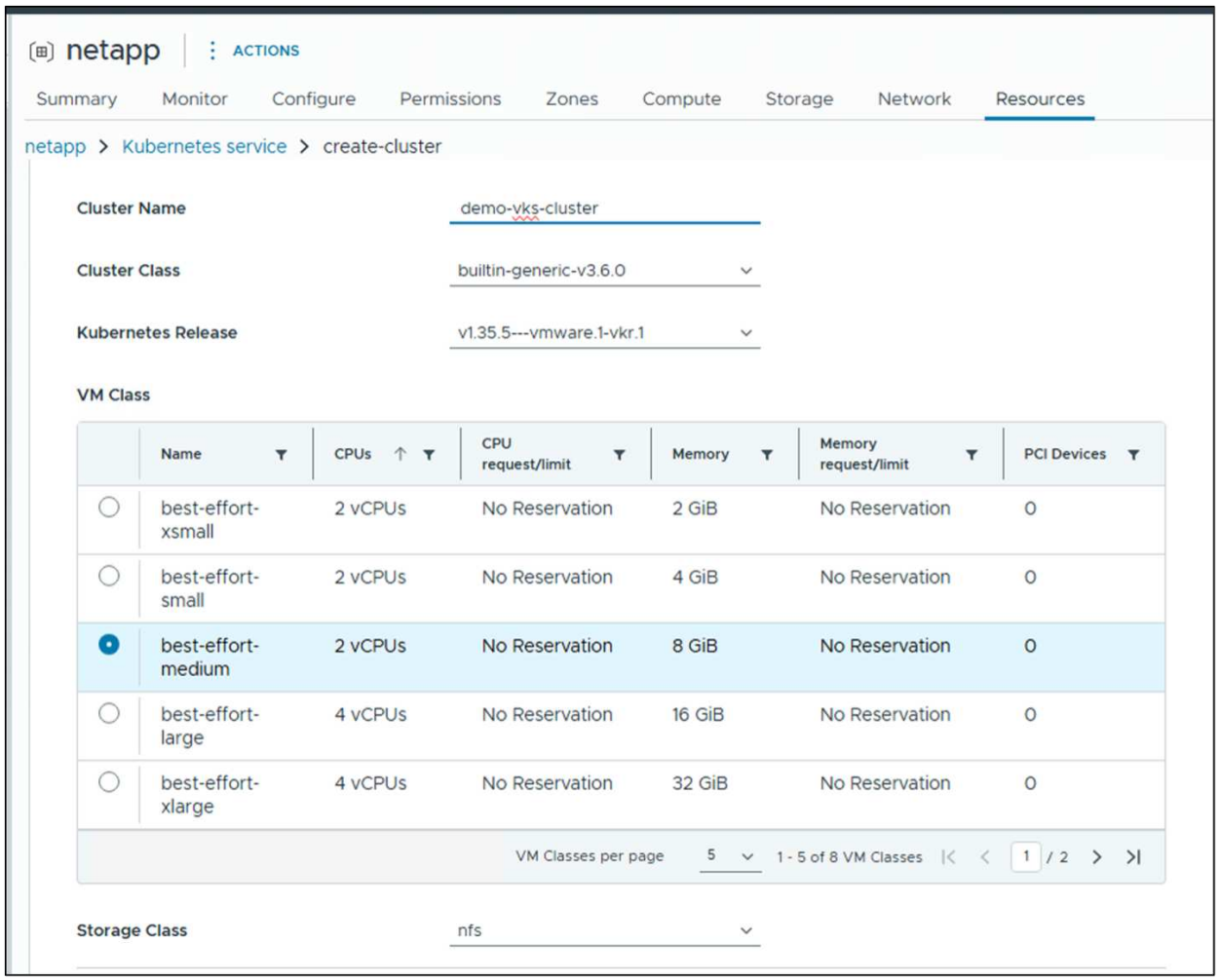
Remplissez le formulaire avec les détails du cluster :

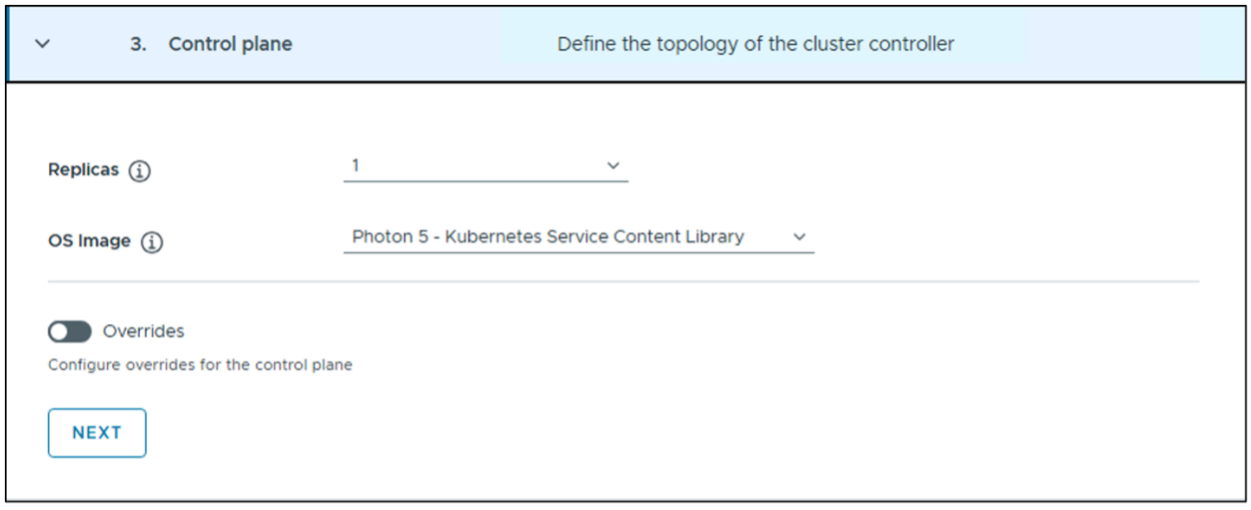
- Dans la section 1, **Type de configuration**, sélectionnez **Personnalisé**.
- Dans la section 2, **Paramètres généraux**, indiquez un nom pour le cluster et laissez tous les autres paramètres à leurs valeurs par défaut.
- Acceptez toutes les valeurs par défaut pour la section 3.
- Dans la section 4, **Node Pools**, cliquez sur les points de suspension (…) à côté du node pool, puis sur **Edit**. Augmentez le nombre de replicas à 3 et sélectionnez la dernière version d'Ubuntu pour l'image du système d'exploitation. Cliquez sur **Next**, puis sur **Review and Confirm**.

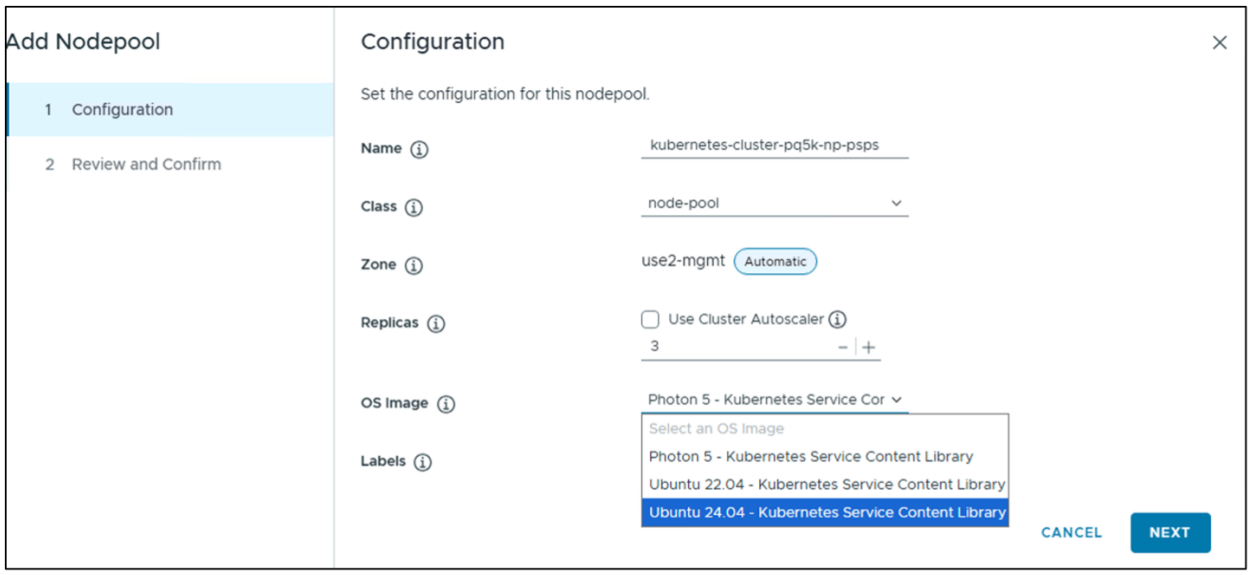
Après avoir vérifié les détails du cluster, cliquez sur **Terminer**. Le cluster Kubernetes vSphere sera créé dans quelques minutes.

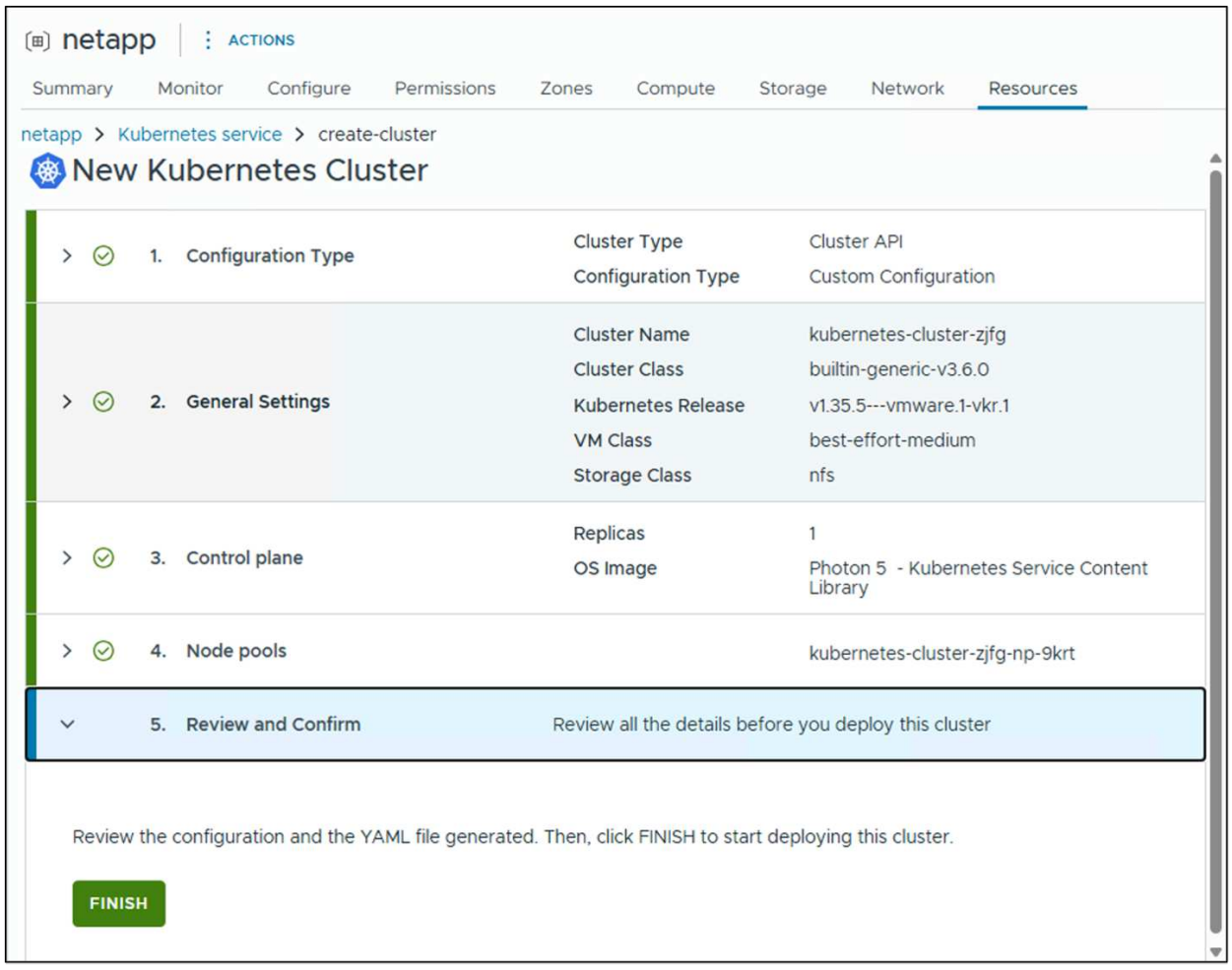


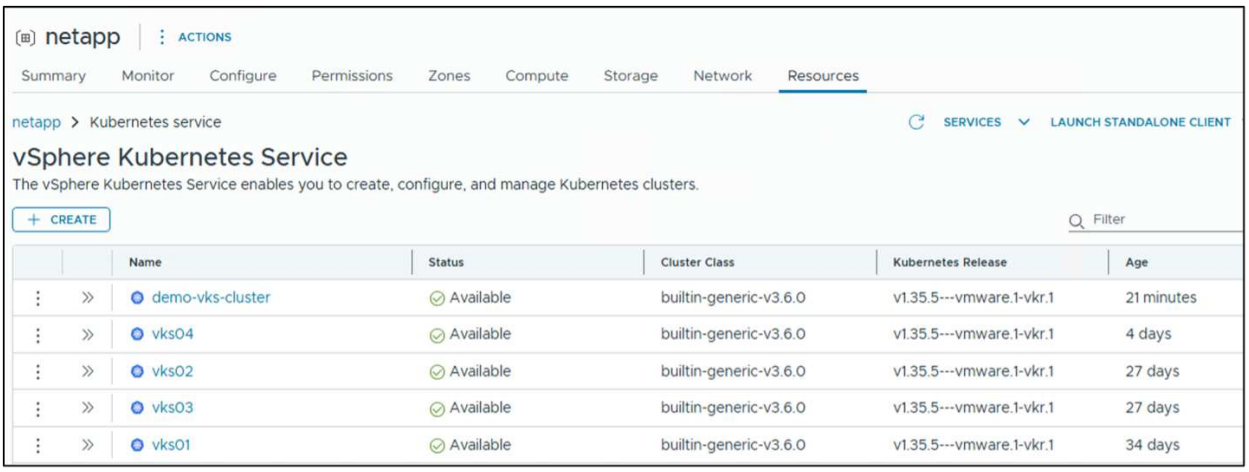



netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
Cluster Name
demo-vks-cluster
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
Name
CPUs
CPU request/limit
Memory
Memory request/limit
PCI Devices
best-effort-xsmall
2 vCPUs
No Reservation
2 GiB
No Reservation
0
best-effort-small
2 vCPUs
No Reservation
4 GiB
No Reservation
0
best-effort-medium
2 vCPUs
No Reservation
8 GiB
No Reservation
0
best-effort-large
4 vCPUs
No Reservation
16 GiB
No Reservation
0
best-effort-xlarge
4 vCPUs
No Reservation
32 GiB
No Reservation
0
VM Classes per page
5
1 - 5 of 8 VM Classes
1 / 2
Storage Class
nfs


3. Control plane
Define the topology of the cluster controller
Replicas
1
OS Image
Photon 5 - Kubernetes Service Content Library
Overrides
Configure overrides for the control plane
NEXT


Add Nodepool
1 Configuration
2 Review and Confirm
Configuration
Set the configuration for this nodepool.
Name
kubernetes-cluster-pq5k-np-psps
Class
node-pool
Zone
use2-mgmt
Automatic
Replicas
Use Cluster Autoscaler
3
OS Image
Photon 5 - Kubernetes Service Cor
Select an OS Image
Photon 5 - Kubernetes Service Content Library
Ubuntu 22.04 - Kubernetes Service Content Library
Ubuntu 24.04 - Kubernetes Service Content Library
Labels
CANCEL
NEXT


netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
New Kubernetes Cluster
1. Configuration Type
Cluster Type
Cluster API
Configuration Type
Custom Configuration
2. General Settings
Cluster Name
kubernetes-cluster-zjfg
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
best-effort-medium
Storage Class
nfs
3. Control plane
Replicas
1
OS Image
Photon 5 - Kubernetes Service Content Library
4. Node pools
kubernetes-cluster-zjfg-np-9krt
5. Review and Confirm
Review all the details before you deploy this cluster
Review the configuration and the YAML file generated. Then, click FINISH to start deploying this cluster.
FINISH



▼ Afficher les instructions pour créer une image OVA personnalisée pour les nœuds de travail compatibles SAN

Créez une image Docker avec les utilitaires requis installés. L'exemple suivant illustre la création d'une image Docker basée sur Ubuntu 24.04 avec les utilitaires iSCSI et Multipathing installés.

```
#san-dockerfile.txt
FROM ubuntu:24.04
# Prevent interactive prompts during package installation
ENV DEBIAN_FRONTEND=noninteractive
# Install required system packages for iSCSI and Multipathing
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
    nvme-cli \
    kmod \
    linux-tools-common \
    open-iscsi \
    lsscsi \
    sg3-utils \
    multipath-tools \
    scsitools  \
    chrony \
    networkd-dispatcher &&\
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*
# Set iSCSI session scanning to manual as required by Trident
RUN sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
# Configure device-mapper-multipath with Trident recommendations
RUN printf "defaults {\n    user_friendly_names yes\n
find_multipaths no\n}\n" > /etc/multipath.conf
RUN mkdir -p /etc/networkd-dispatcher/off.d /etc/networkd-
dispatcher/routable.d/
CMD ["bash"]
```

Créez l'image Docker en utilisant :

```
docker build -f san-dockerfile.txt -t ubuntu-2404:san .
```

Si le registre Docker n'est pas disponible, exécutez cette commande pour démarrer un registre local et y transférer l'image :

```
docker run -d -p 5000:5000 --restart=always --name registry
registry:2
```

Étiquetez l'image et poussez-la.

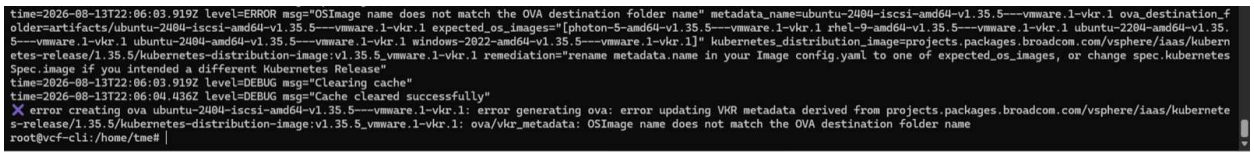
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

Créez le fichier de configuration de l'image (enregistrez-le sous `ubuntu2404vkr135-san.yml`) et référencez l'image :

```
#ubuntu2404vkr135-san.yml
apiVersion: imageconfiguration.vmware.com/v1alpha1
kind: Image
metadata:
    name: ubuntu-2404-amd64-v1.35.5---vmware.1-vkr.1
spec:
    osSpec:
        name: ubuntu
        version: "24.04"
        image: "localhost:5000/ubuntu-2404:san"
    kubernetesSpec:
        image:
projects.packages.broadcom.com/vsphere/iaas/kubernetes-
release/1.35.5/kubernetes-distribution-image:v1.35.5_vmware.1-vkr.1
    diskSize: 20Gi
    repositorySpec:
        debs:
            - name: noble
              url: http://archive.ubuntu.com/ubuntu
              suites:
                  - noble
              components:
                  - main
                  - restricted
                  - universe
```

```
                - multiverse
            - name: noble-security
              url: http://security.ubuntu.com/ubuntu
              suites:
                - noble-security
              components:
                - main
                - restricted
                - universe
                - multiverse
    packages:
      present:                    # Packages to install
        - name: open-iscsi
        - name: lsscsi
        - name: sg3-utils
        - name: multipath-tools
        - name: scsitools
        - name: nvme-cli
    services:
      - name: multipath-tools.service
        status: enabled
      - name: open-iscsi.service
        status: enabled
```

Le nom des métadonnées doit figurer dans la liste pré-approuvée. Dans le cas contraire, vous obtiendrez une erreur comme indiqué ci-dessous lorsque vous tenterez de créer le fichier OVA. La liste pré-approuvée est disponible dans l'aide de la CLI VCF pour la commande `kr bake`.



Créez le fichier OVA à l'aide de la ligne de commandes VCF CLI :

```
vcf kr bake -f <path/to/ubuntu2404vkr135-san.yml> --log-level DEBUG
--log-format plain
```

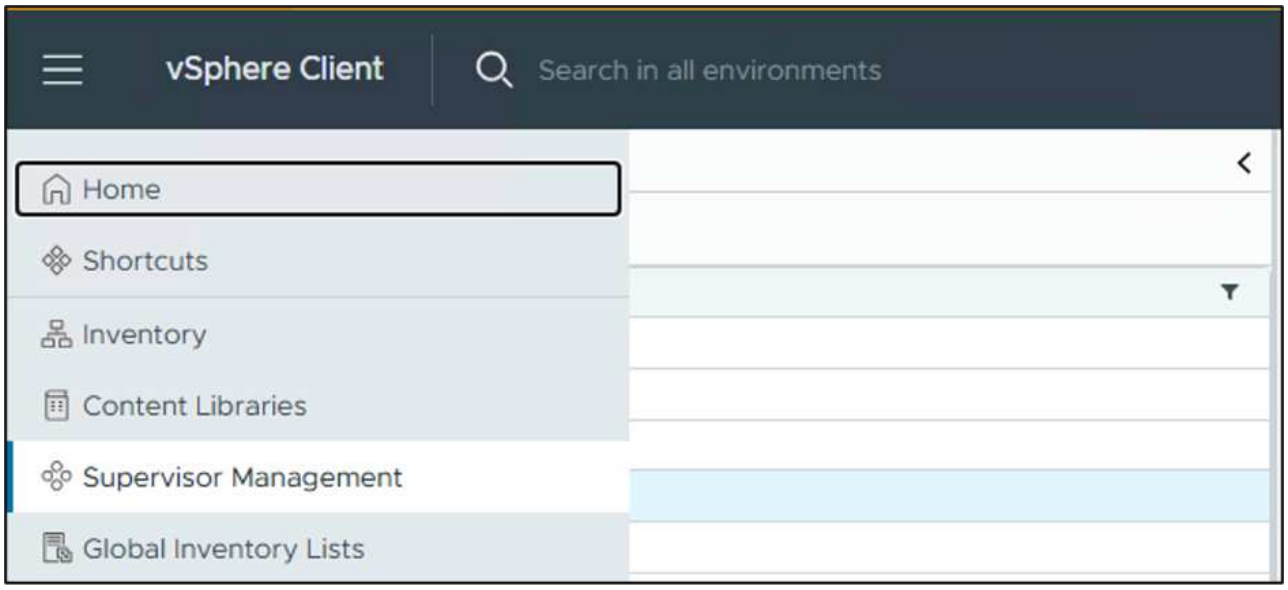
Transférez le fichier OVA à l'aide de SCP vers une machine à partir de laquelle vous pourrez le télécharger dans la vSphere Content Library.

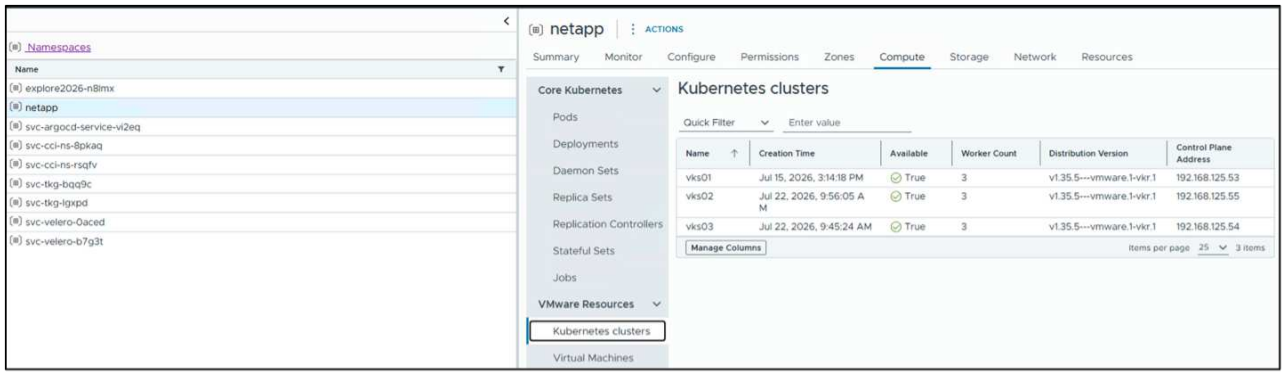
2. Une fois le cluster installé, localisez-le dans vCenter et téléchargez le fichier kubeconfig.

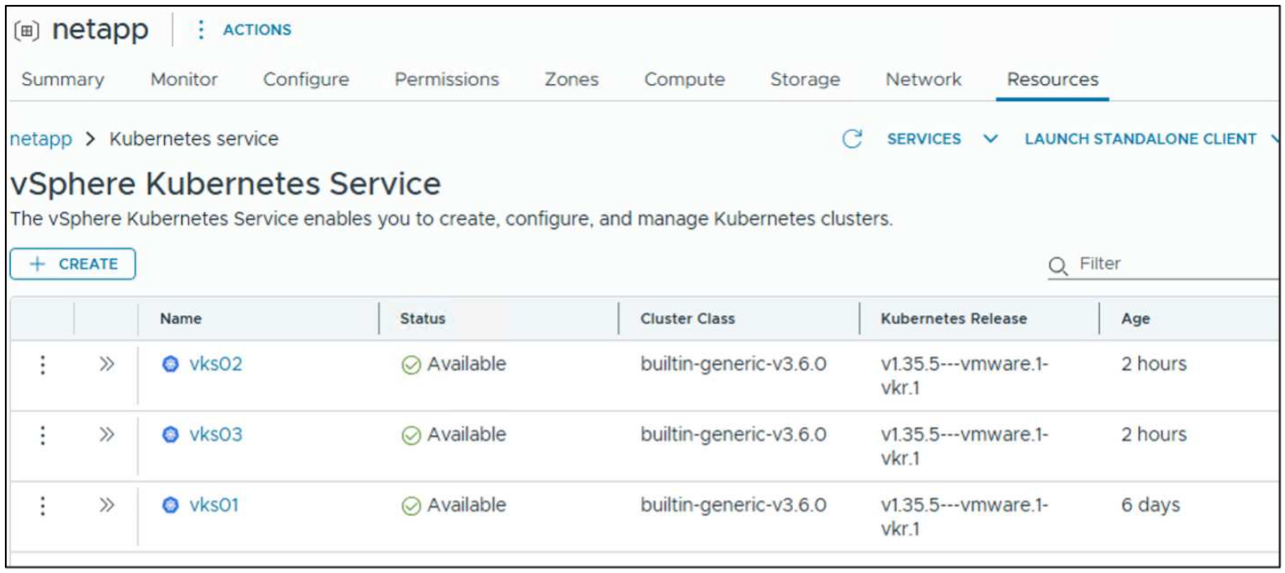
    a. Cliquez sur **Gestion du superviseur** dans le menu principal et sélectionnez l'espace de noms où votre cluster a été créé.

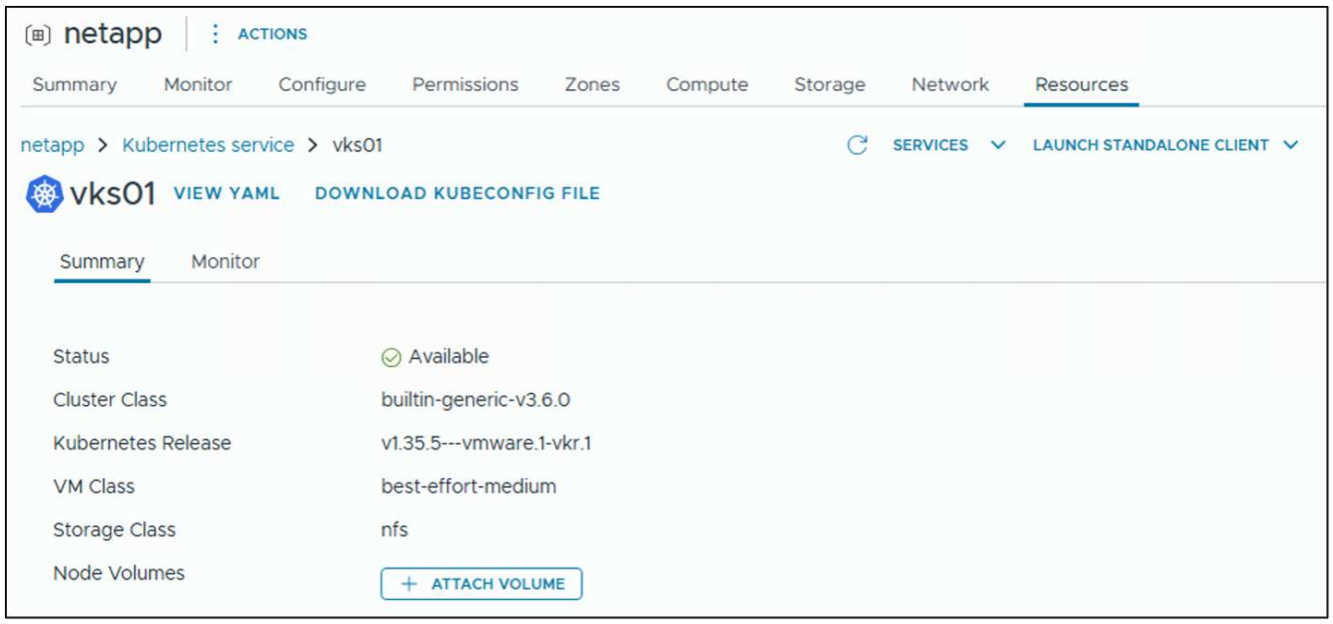
b. Sélectionnez l'onglet **Calcul**, puis **Clusters Kubernetes**. Tous les clusters de l'espace de noms s'affichent.

c. Accédez à l'onglet **Resources**, sélectionnez le cluster Kubernetes dont vous avez besoin et téléchargez son fichier kubeconfig.









3. Depuis un hôte bastion, connectez-vous au cluster en utilisant le fichier kubeconfig pour le gérer depuis la ligne de commandes.

```
vksbastion@vks:~$ kubectl get nodes
NAME                                    STATUS   ROLES           AGE     VERSION
vks01-dhwbg-rpxst                       Ready    control-plane   6d20h   v1.35.5+vmware.1
vks01-vks01-np-pt27-764sx-cdd5q-g5qnw   Ready    <none>          3h45m   v1.35.5+vmware.1
vks01-vks01-np-pt27-764sx-cdd5q-ltj5s   Ready    <none>          6d20h   v1.35.5+vmware.1
vks01-vks01-np-pt27-764sx-cdd5q-npxmj   Ready    <none>          6d20h   v1.35.5+vmware.1
```

Démonstration vidéo

Les vidéos suivantes présentent deux façons de créer un cluster Kubernetes vSphere.

Créer un cluster Kubernetes vSphere sur un cluster superviseur

Créer un cluster Kubernetes vSphere avec VCF Automation

Découvrez le stockage NetApp pour le service Kubernetes vSphere

L'utilisation de NetApp comme stockage pour vSphere Kubernetes Service (VKS) constitue une combinaison puissante qui exploite les solutions de stockage robustes de NetApp pour améliorer la performance, la scalabilité et la fiabilité des charges de travail Kubernetes. NetApp Trident, un provisionneur de stockage dynamique open source pour Kubernetes, est utilisé pour intégrer le stockage aux charges de travail sur vSphere Kubernetes Service.

Principaux avantages de l'utilisation du stockage NetApp avec VKS

1. **Provisionnement dynamique du stockage :** NetApp Trident permet le provisionnement dynamique des volumes de stockage, permettant aux pods Kubernetes de demander du stockage à la volée en fonction de leurs besoins.
2. **Stockage persistant :** Les solutions NetApp offrent des capacités de stockage persistant, garantissant la durabilité et la disponibilité des données en cas de redémarrage ou de panne des pods.

3. **Haute performance :** Les solutions de stockage de NetApp, telles que ONTAP, offrent un stockage haute performance avec une faible latence, ce qui est idéal pour les applications gourmandes en E/S exécutées sur Kubernetes.
4. **Évolutivité :** Les systèmes de stockage NetApp peuvent évoluer pour répondre aux besoins croissants des charges de travail Kubernetes, prenant en charge les déploiements à grande échelle.
5. **Gestion des données :** NetApp fournit des fonctionnalités avancées de gestion des données, notamment les instantanés, le clonage et la réplication des données, qui peuvent être utiles pour la sauvegarde, la reprise après sinistre et les flux de travail de développement.
6. **Prise en charge du multicloud et du cloud hybride :** NetApp propose des solutions de stockage pouvant être déployées sur site, dans le cloud public et dans des environnements de cloud hybride, offrant flexibilité et cohérence dans la gestion du stockage.

NetApp ONTAP aperçu de l'intégration

NetApp ONTAP offre un stockage de niveau entreprise avec des fonctionnalités telles que la déduplication des données, la compression et le chiffrement. Vous utilisez NetApp Trident pour intégrer le stockage NetApp à Kubernetes. NetApp Trident prend en charge différentes plateformes de stockage NetApp, notamment ONTAP sur site, FSx for NetApp ONTAP dans AWS, Azure NetApp Files dans Azure et Google Cloud NetApp Volumes dans Google Cloud.

Vous utilisez Trident Protect pour gérer la protection des données et la reprise après sinistre de vos charges de travail Kubernetes. Trident Protect vous permet de créer des instantanés, des clones et de répliquer des données sur différents systèmes de stockage, garantissant ainsi que vos données sont sûres et récupérables en cas de panne.

Fonctionnalités clés de Trident

Conformité CSI Garantit la compatibilité avec les dernières fonctionnalités et normes de stockage Kubernetes. **Configuration déclarative** Permet aux utilisateurs de définir les exigences de stockage dans des fichiers YAML, qui sont ensuite automatiquement gérés par Trident. **Pilotes** Trident prend en charge les pilotes pour les protocoles NFS, CIFS/SMB, iSCSI, FC et NVMe/TCP pris en charge par ONTAP. **Prise en charge du cloud hybride et multicloud** Trident prend en charge les pilotes pour le stockage ONTAP sur site et les services de stockage gérés chez les hyperscalers, à savoir FSx for NetApp ONTAP sur AWS, Azure NetApp Files sur Azure et Google Cloud NetApp Volumes sur Google Cloud. **Gestion avancée des données** Tirant parti des fonctionnalités de snapshot et de clonage d'ONTAP pour une protection efficace des données et un provisionnement rapide des environnements de test et de développement.

Pour plus de détails sur Trident, veuillez vous référer au "Documentation Trident".

Trident Protect

Trident Protect s'installe séparément de Trident. Avant le déploiement, vérifiez que votre plateforme Kubernetes, votre système de stockage, vos composants de snapshots et votre stockage objet répondent aux "Exigences de Trident Protect".

Principales caractéristiques de Trident Protect

Sauvegardes automatisées Trident Protect permet des sauvegardes automatisées, pilotées par des politiques, des applications Kubernetes, garantissant qu'elles sont régulièrement sauvegardées sans intervention manuelle.

Gestion des instantanés Elle exploite les capacités d'instantanés d'ONTAP pour créer des copies fréquentes et ponctuelles des applications conteneurisées et des machines virtuelles, fournissant ainsi un mécanisme

fiable de récupération.

Chiffrement du référentiel de sauvegarde Trident Protect prend en charge le chiffrement basé sur mot de passe pour les référentiels de sauvegarde Restic et Kopia. Configurez séparément le chiffrement des volumes persistants et le chiffrement en transit dans les couches de stockage et de réseau.

Conformité et audit Fournit des outils et des fonctionnalités qui aident les organisations à respecter les exigences de conformité et à effectuer des audits réguliers de leurs pratiques en matière de protection des données.

Reprise après sinistre S'intègre aux fonctionnalités de réplication d'ONTAP pour fournir des solutions de reprise après sinistre robustes, garantissant la continuité des activités même en cas d'événements catastrophiques.

Pour plus de détails sur Trident Protect, veuillez vous référer au "Documentation Trident Protect".

NetApp Modèles de matériel et protocoles ONTAP pris en charge

NetApp ONTAP fonctionne sur différents modèles de matériel, chacun conçu pour répondre à des exigences spécifiques en matière de performances et de capacité. Trident étend ses fonctionnalités robustes pour prendre en charge divers systèmes NetApp ONTAP, notamment All Flash FAS (AFF), All SAN Array (ASA) et ASA R2. Cette prise en charge garantit que les organisations peuvent exploiter pleinement le potentiel de leurs solutions de stockage haute performance dans des environnements conteneurisés.

Systèmes All Flash FAS (AFF) Les systèmes AFF offrent des performances exceptionnelles et une faible latence, ce qui les rend idéaux pour les applications à forte demande.

All SAN Array (ASA) Systems Les systèmes ASA sont conçus pour les environnements SAN dédiés, offrant des performances et une fiabilité robustes.

ASA R2 Systems Les systèmes ASA R2 offrent des fonctionnalités et des capacités améliorées pour les environnements SAN.

AFX NetApp AFX est une architecture de stockage 100 % Flash, désagrégée et spécialement conçue pour les charges de travail d'IA en entreprise. Elle offre un NAS haute performance, permettant une mise à l'échelle indépendante de la puissance de calcul et de la capacité. Les systèmes de stockage NetApp AFX diffèrent des autres systèmes basés sur ONTAP (ASA, AFF et FAS) par l'implémentation de leur couche de stockage. AFX utilise un système de fichiers distribué qui permet des performances et une évolutivité élevées, tandis que les autres systèmes basés sur ONTAP utilisent une architecture de stockage traditionnelle.

1. Protocoles pris en charge par AFF : NFS, CIFS/SMB, iSCSI, FC, NVMe/TCP
2. Protocoles pris en charge pour ASA : iSCSI, FC, NVMe/TCP
3. Protocoles pris en charge pour ASA R2 : iSCSI, FC, NVMe/TCP
4. Protocoles pris en charge pour AFX : à partir de Trident 25.10, seul le protocole NFS est pris en charge ; le protocole SMB n'est pas pris en charge.

Pilotes Trident pour le stockage ONTAP

Trident inclut les pilotes CSI suivants, chacun capable de provisionner un volume dans le stockage backend.

Pour les protocoles NAS sur les modèles matériels pris en charge

1. ontap-nas : Un PVC correspond à un PV, qui est un volume FlexVol dédié.

2. ontap-nas-economy : Chaque PV provisionné est un qtree, avec un nombre configurable de qtrees par FlexVol volume (la valeur par défaut est de 200, configurable entre 50 et 300).

   Un PVC correspond à un PV, qui est un qtree sur un volume FlexVol partagé.

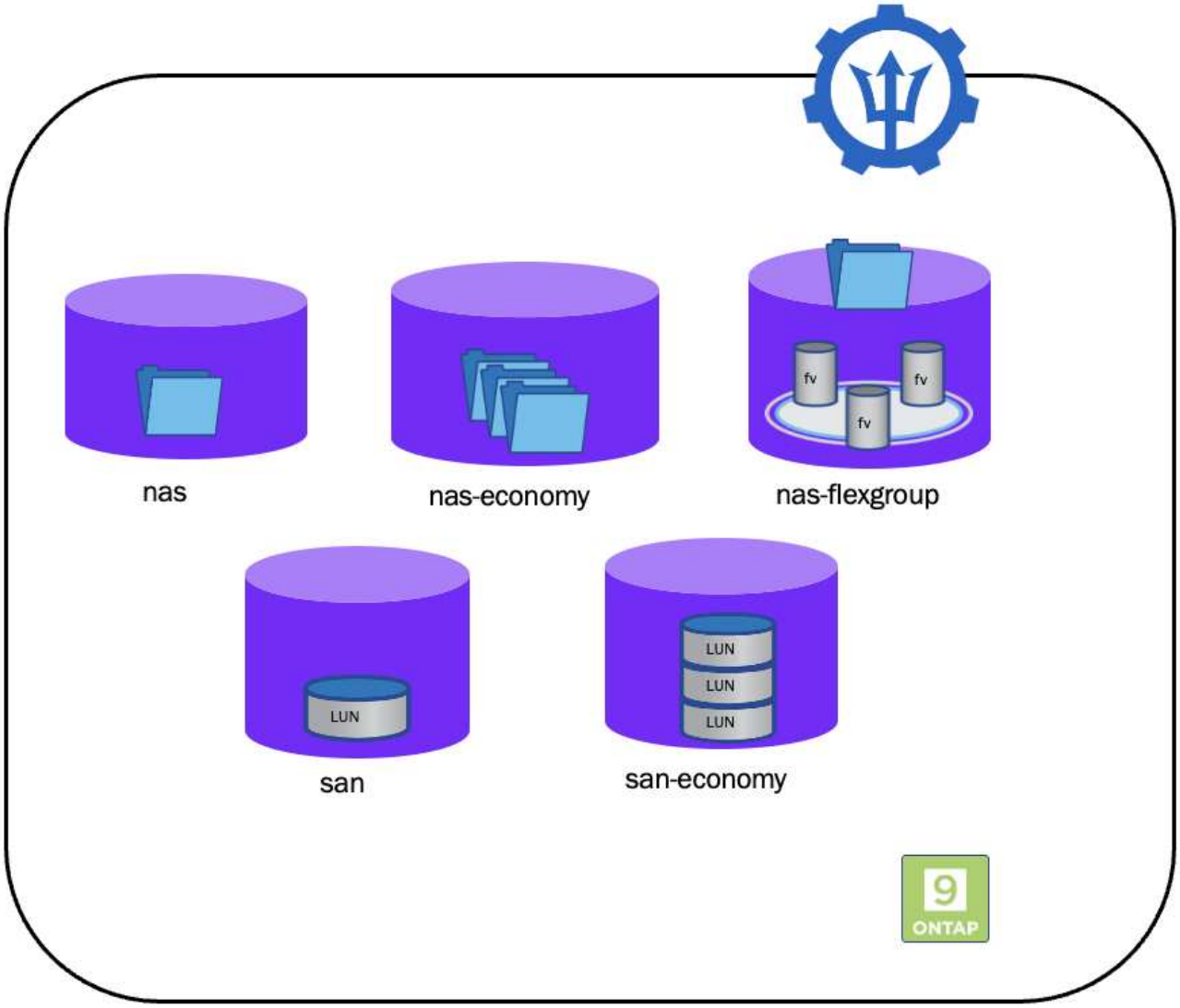
> Vous pouvez placer tous les disques des machines virtuelles sous forme de qtrees dans un seul volume. Cependant, veuillez noter que les fonctionnalités de Snapshots, Clones et Replication ne sont pas prises en charge par Trident. Lorsque ces fonctionnalités sont gérées par l'administrateur de stockage, elles ne sont pas granulaires au niveau des volumes persistants (PV).

1. ontap-nas-flexgroup : Chaque PV provisionné est un ONTAP FlexGroup, et tous les agrégats affectés à un SVM sont utilisés.

Pour les protocoles SAN sur les modèles matériels pris en charge

1. ontap-san : Un PVC correspond à un PV, qui est un LUN sur un volume FlexVol dédié.
2. ontap-san-economy : Un PVC correspond à un PV, qui est un LUN sur un volume FlexVol partagé. Vous pouvez avoir plusieurs LUN dans le volume FlexVol partagé (100 par défaut, configurable entre 50 et 200 LUN/PV par volume FlexVol).

> Vous pouvez placer tous les disques des machines virtuelles sous forme de LUN dans un seul volume. Cependant, ce pilote prend en charge les Snapshots et les Clones, mais ne prend pas en charge la Réplication. Lorsque la Réplication est gérée par l'administrateur de stockage, elle n'est pas granulaire au niveau du PV.



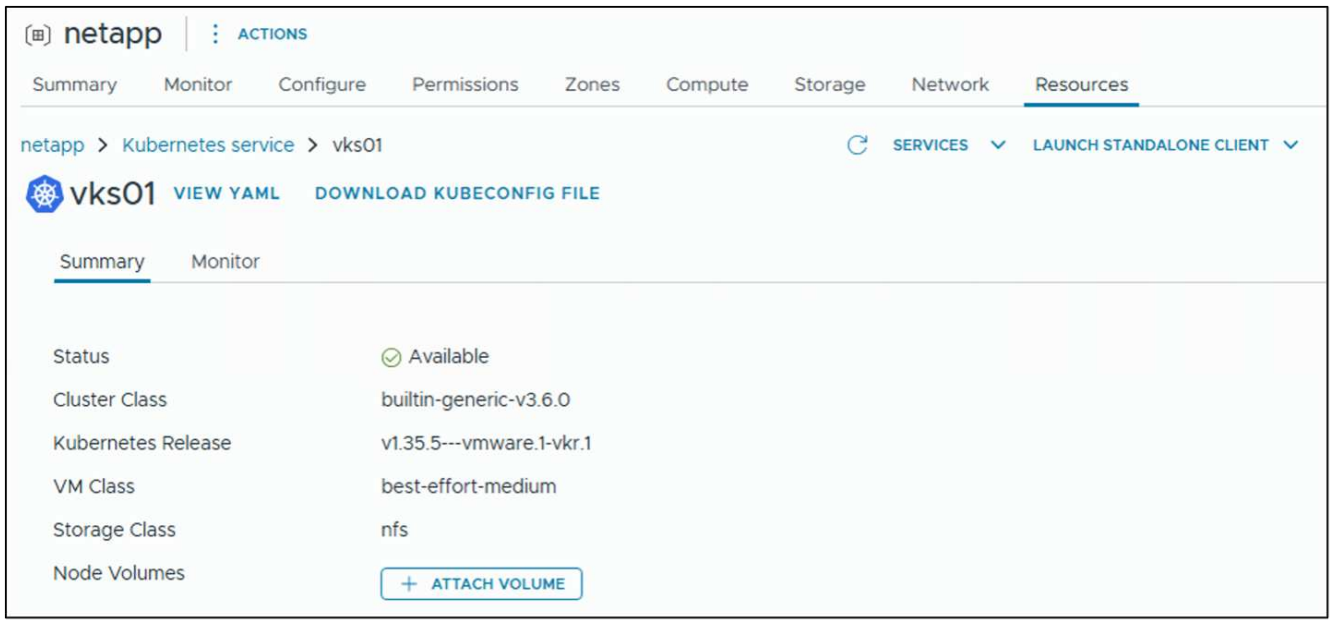
Pour obtenir la liste complète des fonctionnalités exposées par les pilotes Trident et celles qui doivent être appliquées par l'administrateur de stockage, consultez la section "Pilotes backend ONTAP" dans la documentation Trident.

Installer NetApp Trident dans un cluster VKS

Installez NetApp Trident comme provisionneur de stockage dynamique dans votre cluster vSphere Kubernetes Service pour intégrer le stockage NetApp ONTAP à vos charges de travail Kubernetes.

Avant de commencer

- Assurez-vous que votre cluster ONTAP est configuré et connecté à votre environnement VMware Kubernetes.
- Assurez-vous que votre cluster VKS dispose des outils iSCSI ou NVMe installés si vos charges de travail utiliseront ces protocoles pour le stockage.
- Assurez-vous d'avoir `kubectl` installé sur une machine à partir de laquelle vous pouvez vous connecter au cluster VKS à l'aide du fichier kubeconfig.



Étapes

1. Installez l'opérateur Trident à l'aide d'un chart Helm en suivant les instructions de la documentation Trident :

   "https://docs.netapp.com/us-en/trident/trident-install/kubernetes-deploy-helm.html"

   Avant d'exécuter l'installation Helm, créez et étiquetez l'`trident`espace de noms. L'étiquette `enforce=privileged` est requise car Trident exécute des charges de travail privilégiées. Sans cela, le contrôleur d'admission de la sécurité des pods Kubernetes empêchera le démarrage des pods Trident.

   ```
   kubectl create namespace trident
   kubectl label namespace trident pod-
   security.kubernetes.io/enforce=privileged
   ```

   ▼ Afficher un exemple

   ```
   helm install trident netapp-trident/trident-operator --version
   100.2606.0 --namespace trident
   ```

   Pour activer la journalisation de débogage, ajoutez `--set tridentDebug=true` :

   ```
   helm install trident netapp-trident/trident-operator --version
   100.2606.0 --namespace trident --set tridentDebug=true
   ```

   Vous pouvez également installer Trident manuellement à l'aide de l'opérateur Trident. Consultez la documentation Trident pour connaître la procédure : "Déployez manuellement l'opérateur Trident"

Assurez-vous d'avoir étiqueté l'espace de noms `trident` avec les étiquettes de sécurité de pod requises

avant d'installer Trident manuellement.

2. Confirmez que tous les pods Trident sont en cours d'exécution dans le cluster.

▼ **Afficher un exemple**

```
kubectl get pods -n trident
```

```
vksbastion@vks:~/demo-vks$ kubectl get pods -n trident
NAME                                  READY   STATUS    RESTARTS        AGE
trident-controller-67f7cd9bf8-4b289   6/6     Running   0               3d21h
trident-node-linux-46c86              2/2     Running   0               3d21h
trident-node-linux-bh29q              2/2     Running   0               11d
trident-node-linux-fcvjw              2/2     Running   1 (3d20h ago)   3d20h
trident-node-linux-gkhls              2/2     Running   0               3d20h
trident-operator-5cbf7f857-h5hx8      1/1     Running   0               3d20h
vksbastion@vks:~/demo-vks$
```

Si les pods Trident ne démarrent pas et qu'une erreur d' `PodSecurity`admission se produit, il est possible que les étiquettes de sécurité des pods n'aient pas été appliquées à l' `trident`espace de noms avant l'installation. Utilisez `--overwrite` pour les réappliquer, puis vérifiez que les pods démarrent.

```
kubectl label namespace trident pod-
security.kubernetes.io/enforce=privileged --overwrite
```

Préparez les fichiers de configuration du système de stockage et de StorageClass pour votre environnement

1. Configurez un backend Trident pour ONTAP en définissant les détails de connexion et les paramètres de stockage nécessaires.
2. Définissez des classes de stockage dans Kubernetes qui correspondent aux backends de stockage NetApp. Ces classes spécifient des paramètres tels que les caractéristiques de performance et les politiques de réplication.

   Définissez les backends Trident et les classes de stockage pour les protocoles suivants, selon les besoins de vos charges de travail :

   - NAS (pilote ontap-nas)
   - NAS Economy (pilote ontap-nas-economy)
   - SAN (pilote ontap-san) pour les protocoles iSCSI, FC ou NVMe
   - SAN Economy (pilote ontap-san-economy) pour iSCSI

   NAS

   Créez un TridentBackendConfig et un StorageClass pour ONTAP NAS afin d'activer le provisionnement de stockage persistant basé sur NFS. La configuration du backend inclut des informations d'identification stockées dans un secret Kubernetes et fait référence à votre ONTAP SVM et à votre LIF de gestion.

Exemple de secret backend et de fichier de configuration backend utilisant la méthode d'authentification par nom d'utilisateur et mot de passe (enregistrez sous `tbc-nas.yaml`) :

```
# tbc-nas.yaml
apiVersion: v1
kind: Secret
metadata:
  name: tbc-nas-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using
this backend configuration>
  defaults:
    nameTemplate: "{{ .config.StoragePrefix }}_{{
.volume.Namespace }}_{{ .volume.RequestName }}"
  credentials:
    name: tbc-nas-secret
```

Exemple de secret backend et de fichier de configuration backend utilisant la méthode d'authentification par certificat client (enregistrez sous `tbc-nas.yaml`) :

```
# tbc-nas.yaml
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key value>
```

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using
this backend configuration>
  credentials:
    name: ontap-nas-secret
  clientCertificate: <client certificate>
```

StorageClass definition (enregistrer sous `sc-nas.yaml`):

`mountOptions` est un paramètre optionnel qui spécifie des options de montage supplémentaires pour les volumes NFS. L' `nconnect`option permet plusieurs connexions TCP parallèles pour un même montage NFS, ce qui peut améliorer les performances pour les charges de travail à haut débit. La valeur par défaut est 1, mais elle peut être augmentée jusqu'à la limite maximale autorisée par le système ONTAP.

ONTAP prend en charge jusqu'à 16 connexions pour un seul montage NFS sur un client compatible nconnect. Trident transmet directement les options de montage aux nœuds de travail Kubernetes, alors choisissez une valeur prise en charge à la fois par le client NFS du nœud de travail et par ONTAP ; l'exemple suivant utilise quatre connexions.

```
# sc-nas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
mountOptions:
```

```
- nconnect=4
```

Créez un TridentBackendConfig et un StorageClass pour le pilote ONTAP NAS Economy afin d'activer le provisionnement de stockage persistant basé sur NFS à l'aide de qtrees. La configuration du backend inclut des informations d'identification stockées dans un Secret Kubernetes et fait référence à votre ONTAP SVM et à votre LIF de gestion.

Exemple de secret backend et de fichier de configuration backend utilisant la méthode d'authentification par nom d'utilisateur et mot de passe (enregistrez sous `tbc-nas-economy.yaml`) :

```
# tbc-nas-economy.yaml
apiVersion: v1
kind: Secret
metadata:
  name: tbc-nas-economy-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-nas-economy
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: <ONTAP management LIF>
  backendName: tbc-nas-eco
  svm: <your SVM name>
  storagePrefix: <your prefix for all volume names created using this
backend configuration>
  defaults:
    nameTemplate: "{{ .config.StoragePrefix }}_{{ .volume.Namespace
}}_{{ .volume.RequestName }}"
  credentials:
    name: tbc-nas-economy-secret
```

StorageClass definition (enregistrer sous `sc-nas-economy.yaml`):

```
# sc-nas-economy.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```
metadata:
  name: sc-nas-economy
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas-economy"
  media: "ssd"
  provisioningType: "thin"
allowVolumeExpansion: true
mountOptions:
- nconnect=4
```

Créez un TridentBackendConfig et un StorageClass pour ONTAP SAN avec iSCSI afin d'activer le provisionnement de stockage bloc basé sur iSCSI. La configuration du backend utilise le pilote ontap-san et inclut des informations d'identification stockées dans un Secret Kubernetes. Le `sanType` paramètre utilise par défaut le protocole iSCSI s'il est omis.

Secret backend et fichier de configuration backend utilisant l'authentification par nom d'utilisateur et mot de passe (enregistrer sous `tbc-iscsi.yaml`) :

```
# tbc-iscsi.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-iscsi-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-iscsi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <ONTAP management LIF>
  backendName: ontap-iscsi
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-iscsi-secret
```

Secret backend et fichier de configuration backend utilisant la méthode d'authentification par certificat client (enregistrez sous `tbc-iscsi.yaml`) :

```
# tbc-iscsi.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-iscsi-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-iscsi
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  managementLIF: <ONTAP management LIF>
  backendName: ontap-iscsi
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-iscsi-secret
  clientCertificate: <client certificate>
```

StorageClass definition (enregistrer sous `sc-iscsi.yaml`):

```
# sc-iscsi.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-iscsi
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

Créez un TridentBackendConfig et un StorageClass pour ONTAP SAN avec NVMe afin d'activer le provisionnement de stockage bloc basé sur NVMe. La configuration du backend utilise le pilote ontap-san et inclut des informations d'identification stockées dans un Secret Kubernetes. Le `sanType` paramètre

doit être défini sur `nvme`.

Secret backend et fichier de configuration backend utilisant l'authentification par nom d'utilisateur et mot de passe (enregistrer sous `tbc-nvme.yaml`) :

```
# tbc-nvme.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-nvme-secret
  namespace: trident
type: Opaque
data:
  username: "<base64-encoded cluster admin username>"
  password: "<base64-encoded cluster admin password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: ontap-nvme
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  sanType: nvme
  managementLIF: <ONTAP management LIF>
  backendName: ontap-nvme
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-nvme-secret
```

Secret backend et fichier de configuration backend utilisant la méthode d'authentification par certificat client (enregistrez sous `tbc-nvme.yaml`) :

```
# tbc-nvme.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-nvme-secret
  namespace: trident
type: Opaque
stringData:
  clientPrivateKey: <client private key>
---
apiVersion: trident.netapp.io/v1
```

```
kind: TridentBackendConfig
metadata:
  name: ontap-nvme
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  sanType: nvme
  managementLIF: <ONTAP management LIF>
  backendName: ontap-nvme
  svm: <your SVM name>
  credentials:
    name: backend-tbc-ontap-nvme-secret
  clientCertificate: <client certificate>
```

StorageClass definition (enregistrer sous `sc-nvme.yaml`):

```
# sc-nvme.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nvme
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  media: "ssd"
  provisioningType: "thin"
  fsType: ext4
  snapshots: "true"
allowVolumeExpansion: true
```

Consultez la documentation Trident pour obtenir des exemples supplémentaires de fichiers YAML et des informations sur les modes d'accès et les modes de volume pris en charge par les pilotes SAN et NAS.

- "Exemples utilisant des pilotes NAS"
- "Exemples utilisant des pilotes SAN"

Créez un fichier de configuration VolumeSnapshotClass

Créez une définition VolumeSnapshotClass. Cette configuration active les opérations basées sur les instantanés pour les volumes persistants.

VolumeSnapshotClass definition (enregistrer sous `snapshot-class.yaml`):

```
# snapshot-class.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: trident-snapshotclass
driver: csi.trident.netapp.io
deletionPolicy: Retain
```

Appliquez les fichiers de configuration à votre cluster

Appliquez les fichiers de configuration que vous avez créés lors des étapes précédentes au cluster vSphere Kubernetes Service en utilisant `kubectl`. Cela crée les secrets nécessaires, les configurations backend, les classes de stockage et la classe de snapshot dans votre cluster.

Étapes

1. Appliquez les fichiers TridentBackendConfig et StorageClass pour chaque protocole que vous avez configuré.

```
kubectl apply -f tbc-nas.yaml -n trident
kubectl apply -f sc-nas.yaml

kubectl apply -f tbc-nas-economy.yaml -n trident
kubectl apply -f sc-nas-economy.yaml

kubectl apply -f tbc-iscsi.yaml -n trident
kubectl apply -f sc-iscsi.yaml

kubectl apply -f tbc-nvme.yaml -n trident
kubectl apply -f sc-nvme.yaml

kubectl apply -f snapshot-class.yaml
```

2. Vérifiez que les ressources ont été créées avec succès.

   Vérifiez les objets TridentBackendConfig :

```
kubectl get tbc -n trident
```

   Vérifiez les objets StorageClass :

```
kubectl get storageclass
```

   Vérifiez VolumeSnapshotClass :

```
kubectl get volumesnapshotclass
```

Définissez les classes de stockage et d'instantanés Trident par défaut

Définissez les classes StorageClass et VolumeSnapshotClass de Trident comme valeurs par défaut dans le cluster vSphere Kubernetes Service.

Étapes

1. Définissez le Trident StorageClass par défaut.

   Définissez un StorageClass pris en charge par Trident comme valeur par défaut du cluster afin que les PersistentVolumeClaims l'utilisent automatiquement lorsqu'aucune classe de stockage n'est spécifiée.

   Assurez-vous qu'une seule StorageClass soit définie par défaut. Si une autre StorageClass est déjà définie par défaut, définissez son annotation sur `false`.

   Depuis l'interface de ligne de commande :

   ```
   kubectl patch storageclass <storage class name> -p '{"metadata":
   {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
   ```

2. Définissez la VolumeSnapshotClass Trident par défaut.

   Définissez un VolumeSnapshotClass pris en charge par Trident comme valeur par défaut du cluster afin d'activer les opérations basées sur les instantanés pour les volumes persistants. Cela garantit que les VolumeSnapshots utilisent automatiquement le pilote CSI Trident lorsqu'aucune classe d'instantané n'est spécifiée.

   Assurez-vous qu'une seule VolumeSnapshotClass soit définie par défaut. Si une autre VolumeSnapshotClass est déjà définie par défaut, définissez son annotation sur `false`.

   Depuis l'interface de ligne de commande :

   ```
   kubectl patch volumesnapshotclass <snapshot class name> --type=merge -p
   '{"metadata":{"annotations":{"snapshot.storage.kubernetes.io/is-default-
   class":"true"}}}'
   ```

Utilisez les revendications de volumes persistants Kubernetes pour demander du stockage pour les charges de travail

Trident provisionne automatiquement les volumes nécessaires à partir du stockage NetApp backend. Consultez "Déployez des charges de travail avec un stockage persistant" pour des exemples de déploiement d'une charge de travail avec des PVC dans un cluster VKS.

Déployez une application conteneurisée avec un stockage persistant NetApp

Déployez une application conteneurisée dans votre cluster vSphere Kubernetes Service en utilisant le stockage persistant NetApp ONTAP. Les exemples suivants illustrent le

déploiement d'une base de données PostgreSQL en utilisant soit le stockage NAS (ontap-nas), soit le stockage SAN iSCSI (ontap-san).

La configuration YAML suivante définit directement POSTGRES_USER / POSTGRES_PASSWORD dans le manifeste. En production, il est recommandé d'utiliser les Secrets Kubernetes pour gérer les informations sensibles telles que les identifiants de base de données.

Déployez PostgreSQL avec un stockage NAS (pilote ontap-nas)

Vous pouvez déployer une application conteneurisée PostgreSQL reposant sur un volume persistant NFS provisionné par le pilote ontap-nas. L'exemple suivant illustre comment créer un PersistentVolumeClaim (PVC) et un déploiement pour PostgreSQL.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pvc
spec:
  storageClassName: sc-nas # Specify the StorageClass for dynamic
provisioning
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      securityContext:
        fsGroup: 999  # Ensures proper permissions for the mounted
volume (PostgreSQL default user is 999)
      containers:
        - name: postgres
          image: postgres:15
          securityContext:
```

```
            allowPrivilegeEscalation: false
            runAsNonRoot: true
            runAsUser: 999  # PostgreSQL default user ID
            capabilities:
              drop:
                - ALL
            seccompProfile:
              type: RuntimeDefault
          env:
            - name: POSTGRES_DB
              value: "postgres"
            - name: POSTGRES_USER
              value: "<username>"
            - name: POSTGRES_PASSWORD
              value: "<password>"
            - name: PGDATA
              value: "/var/lib/postgresql/data/pgdata"
          ports:
            - containerPort: 5432
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: postgres-storage
              subPath: postgres-data  # Ensures data is stored in a
subdirectory to avoid permission issues
          resources:
            requests:
              memory: "512Mi"
              cpu: "250m"
            limits:
              memory: "1Gi"
              cpu: "500m"
      volumes:
        - name: postgres-storage
          persistentVolumeClaim:
            claimName: postgres-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  type: ClusterIP
  ports:
    - port: 5432
      targetPort: 5432
  selector:
```

```
    app: postgres
```

Déployez PostgreSQL avec un stockage SAN (pilote iSCSI ontap-san)

Utilisez le YAML suivant pour déployer une application conteneurisée PostgreSQL reposant sur un volume persistant iSCSI provisionné par le pilote ontap-san. La stratégie de déploiement `Recreate` garantit que le pod est entièrement arrêté avant qu'un nouveau ne démarre, ce qui est requis pour les volumes iSCSI ReadWriteOnce et prévient la corruption des données lors des redémarrages de pod. Cette configuration prend en charge la persistance des données lors des suppressions et recréations de pods, et est compatible avec les snapshots de volumes Trident ainsi qu'avec les opérations de sauvegarde et de restauration.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-block-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: sc-iscsi # Must map to an ontap-san driver
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres-block-app
spec:
  replicas: 1
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      # 1. POD LEVEL SECURITY CONTEXT
      securityContext:
        runAsNonRoot: true
        runAsUser: 999       # Official Postgres image default user ID
        runAsGroup: 999      # Official Postgres image default group ID
        fsGroup: 999
        seccompProfile:
```

```
          type: RuntimeDefault
      containers:
        - name: postgres
          image: postgres:15
          # 2. CONTAINER LEVEL SECURITY CONTEXT
          securityContext:
            allowPrivilegeEscalation: false
            capabilities:
              drop:
                - ALL
          env:
            - name: POSTGRES_DB
              value: "postgres"
            - name: POSTGRES_USER
              value: "<username>"
            - name: POSTGRES_PASSWORD
              value: "<password>"
            - name: PGDATA
              value: /var/lib/postgresql/data/pgdata
          ports:
            - containerPort: 5432
              name: postgres
          volumeMounts:
            - mountPath: /var/lib/postgresql/data
              name: postgres-block-storage
              subPath: postgres-data
          resources:
            requests:
              memory: "512Mi"
              cpu: "250m"
            limits:
              memory: "1Gi"
              cpu: "500m"
      volumes:
        - name: postgres-block-storage
          persistentVolumeClaim:
            claimName: postgres-block-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
spec:
  type: ClusterIP
  ports:
    - port: 5432
```

```
    targetPort: 5432
  selector:
    app: postgres
```

Valider le déploiement de la base de données

Après le déploiement de l'application, vérifiez que les pods sont en cours d'exécution et que la base de données est opérationnelle. Utilisez les commandes suivantes pour vérifier l'état des pods, des PVC et des services. Assurez-vous que les pods sont en cours d'exécution et que les PVC sont liés aux volumes appropriés.

```
kubectl get all -n <namespace>
kubectl get pvc -n <namespace>
```

```
vksbastion@vks:~$ kubectl get all -n postgres
NAME                                    READY   STATUS    RESTARTS   AGE
pod/postgres-block-app-7c9859c558-bxhbg   1/1     Running   0          20h

NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.100.154.170   <none>        5432/TCP   25h

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres-block-app   1/1     1            1           25h

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-block-app-7c9859c558   1         1         1       25h
vksbastion@vks:~$ kubectl get pvc -n postgres
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
postgres-block-pvc   Bound    pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536   20Gi       RWO            sc-iscsi       <unset>                 25h
vksbastion@vks:~$
```

Selon le protocole utilisé, le stockage backend est un volume, un qtree ou un LUN. Vous pouvez vérifier le stockage provisionné dans le système ONTAP en décrivant le PersistentVolume (PV) pour récupérer le nom du volume interne, puis en l'utilisant dans les commandes de l'interface de ligne de commande ONTAP pour obtenir les détails du stockage. Utilisez la commande suivante pour décrire le PV et récupérer le nom du volume interne :

```
kubectl describe pv/<pv-name>
```

```
vksbastion@vks:~/demo-vks$ kubectl describe pv/pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
Name:            pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.kubernetes.io/provisioner-deletion-secret-name:
                 volume.kubernetes.io/provisioner-deletion-secret-namespace:
Finalizers:      [external-provisioner.volume.kubernetes.io/finalizer kubernetes.io/pv-protection external-attacher/csi-trident-ne
tapp-io]
StorageClass:    sc-nas
Status:          Bound
Claim:           default/postgres-pvc
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        10Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            csi.trident.netapp.io
    FSType:
    VolumeHandle:      pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
    ReadOnly:          false
    VolumeAttributes:      backendUUID=8d6b9527-91dc-4847-b5cc-6aa7307cf1ba
                           internalName=trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77
                           name=pvc-4782b054-5aff-49ae-bc40-08cdfd30db77
                           protocol=file
                           storage.kubernetes.io/csiProvisionerIdentity=1787189141467-7183-csi.trident.netapp.io
Events:                <none>
```

Figure 1. Afficher un exemple

```
vksbastion@vks:~/vks02$ kubectl describe pv/pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
Name:            pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.kubernetes.io/provisioner-deletion-secret-name:
                 volume.kubernetes.io/provisioner-deletion-secret-namespace:
Finalizers:      [external-provisioner.volume.kubernetes.io/finalizer kubernetes.io/pv-protection external-attacher/csi-trident-ne
tapp-io]
StorageClass:    sc-iscsi
Status:          Bound
Claim:           postgres/postgres-block-pvc
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        20Gi
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume source)
    Driver:            csi.trident.netapp.io
    FSType:            ext4
    VolumeHandle:      pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
    ReadOnly:          false
    VolumeAttributes:      backendUUID=01dd718d-04a4-4c43-95fb-dc52f56e8624
                           internalName=trident_pvc_b992a5cf_1127_4b6c_a775_80ab64e4b536
                           name=pvc-b992a5cf-1127-4b6c-a775-80ab64e4b536
                           protocol=block
                           storage.kubernetes.io/csiProvisionerIdentity=1786638282639-3882-csi.trident.netapp.io
Events:                <none>
```

Copiez le nom du volume interne à partir de la sortie et exécutez la commande suivante pour obtenir les détails de stockage à partir de l'interface de ligne de commande ONTAP.

Pour les volumes NAS :

```
volume show -vserver <vserver> -volume <internal-volume-name>
```

```
NSOL-NetApp-A70-T19U05::> volume show -vserver vks -volume trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77

                                Vserver Name: vks
                                 Volume Name: trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77
                              Aggregate Name: NSOL_NetApp_A70_T19U05a_NVME_SSD_1
    List of Aggregates for FlexGroup Constituents: NSOL_NetApp_A70_T19U05a_
                                               NVME_SSD_1
                            Encryption Type: volume
               List of Nodes Hosting the Volume: NSOL-NetApp-A70-T19U05a
                                 Volume Size: 10GB
                          Volume Data Set ID: 2179
                   Volume Master Data Set ID: 2155359667
                                Volume State: online
                                Volume Style: flex
                           Extended Volume Style: flexvol
                         FlexCache Endpoint Type: none
                          Is Cluster-Mode Volume: true
                           Is Constituent Volume: false
                  Number of Constituent Volumes: -
                                Export Policy: default
                                      User ID: 0
                                     Group ID: 0
                                Security Style: unix
                               UNIX Permissions: ---rwxrwxrwx
                                   Junction Path: /trident_pvc_4782b054_5aff_49ae_bc40_08cdfd30db77
                            Junction Path Source: RW_volume
```

Pour les LUN SAN, exécutez la commande suivante pour obtenir les détails du LUN à partir de l'interface de ligne de commande ONTAP :

```
lun show -vserver <vserver> -volume <internal-volume-name>
```

```
NSOL-NetApp-A70-T19U05::> lun show -vserver vks -volume trident_pvc_b992a5cf_1127_4b
6c_a775_80ab64e4b536
Vserver   Path                            State   Mapped   Type        Size
--------- ------------------------------ ------- -------- -------- --------
vks       /vol/trident_pvc_b992a5cf_1127_4b6c_a775_80ab64e4b536/lun0
                                          online  mapped   linux        20GB

NSOL-NetApp-A70-T19U05::>
```

Si vous avez utilisé les options de montage `nconnect` dans la classe de stockage NAS ou NAS Economy, vous pouvez vérifier le nombre de connexions TCP actives du nœud de travail vers le serveur de stockage.

Commencez par lister les configurations du backend Trident pour trouver le nom du backend, puis décrivez-le pour récupérer le nom du vserver et le LIF de données :

```
kubectl get tbc -n trident
kubectl describe tbc <backend-name> -n trident
```

Connectez-vous ensuite au cluster ONTAP et exécutez la commande suivante, en remplaçant la LIF de données par celle de la sortie ci-dessus :

```
network connections active show -local-address <data-lif> -local-port 2049
```

```
NSOL-NetApp-A70-T19U05::> network connections active show -local-address 192.168.164.214 -local-port 2049
Vserver     Interface              Remote
Name        Name:Local Port        Host:Port                    Protocol/Service
----------- ---------------------- ---------------------------- ----------------
Node: NSOL-NetApp-A70-T19U05b
vks         lif_VKS_4608:2049      192.168.178.41:821           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:843           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:745           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:866           TCP/nfs
vks         lif_VKS_4608:2049      192.168.178.54:759           TCP/nfs
5 entries were displayed.
```

Figure 2. Afficher un exemple

Une fois les pods en état de fonctionnement, connectez-vous à la base de données et vérifiez les opérations sur les données.

Étapes

1. Redirigez le port du service PostgreSQL vers votre machine locale :

```
kubectl port-forward svc/postgres-service 5432:5432 -n <namespace>
```

2. Depuis un autre terminal, connectez-vous à la base de données en utilisant psql :

```
psql "postgresql://<username>:<password>@127.0.0.1:5432/postgres"
```

3. Créez une base de données et une table, puis remplissez la table avec des données :

```
CREATE DATABASE employee;
```

Passez à la nouvelle base de données (méta-commande psql) :

```
\c employee
```

Créez la table, insérez une ligne et interrogez les résultats :

```
CREATE TABLE employees (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    department VARCHAR(50),
    hire_date DATE DEFAULT CURRENT_DATE,
    salary NUMERIC(10, 2)
);

INSERT INTO employees (first_name, last_name, email, department, salary)
VALUES ('John', 'Doe', 'john.doe@example.com', 'Engineering', 85000.00);
```

```
SELECT * FROM employees;
```

Démonstration vidéo

La vidéo suivante montre comment installer Trident sur un cluster Kubernetes vSphere et déployer une base de données PostgreSQL avec stockage persistant à l'aide de Trident.

Déploiement de charges de travail sur un cluster vSphere Kubernetes Service reposant sur un stockage persistant ONTAP à l'aide de Trident

Protection des données

Sauvegardez et restaurez des applications conteneurisées dans un cluster vSphere Kubernetes Service à l'aide de la NetApp Console

Sauvegardez et restaurez des applications conteneurisées dans un cluster vSphere Kubernetes Service (VKS) à l'aide de la NetApp Console. Cette procédure couvre la création et la gestion des opérations de sauvegarde et de restauration via le service Backup and Recovery Service via la NetApp Console, en utilisant le stockage d'objets ONTAP S3 comme cible de sauvegarde.

NetApp Console permet la gestion centralisée du stockage et des services de données dans les environnements sur site et cloud. Elle offre un contrôle unifié, une simplicité opérationnelle et une gestion sécurisée. Plusieurs services de données et fonctionnalités de gestion sont accessibles depuis l'interface utilisateur à "console.netapp.com". Pour des informations complètes sur NetApp Console, consultez "Documentation NetApp Console".

Cette section est consacrée au service de données NetApp Backup and Recovery pour les charges de travail Kubernetes, en particulier les applications conteneurisées sur les clusters vSphere Kubernetes Service. Le service NetApp Backup and Recovery vous permet de découvrir, gérer, créer et associer des politiques de protection à vos applications Kubernetes à l'aide d'une interface unique. Pour un guide complet, consultez "Documentation NetApp Backup and Recovery".

Flux de travail de sauvegarde et de restauration

1. **Assurez-vous de disposer d'un agent Console**

   Assurez-vous qu'un agent Console est déployé dans l'environnement où votre cluster VKS est en cours d'exécution. Pour plus d'informations sur l'installation d'un agent Console, consultez la "NetApp Console Documentation d'installation".

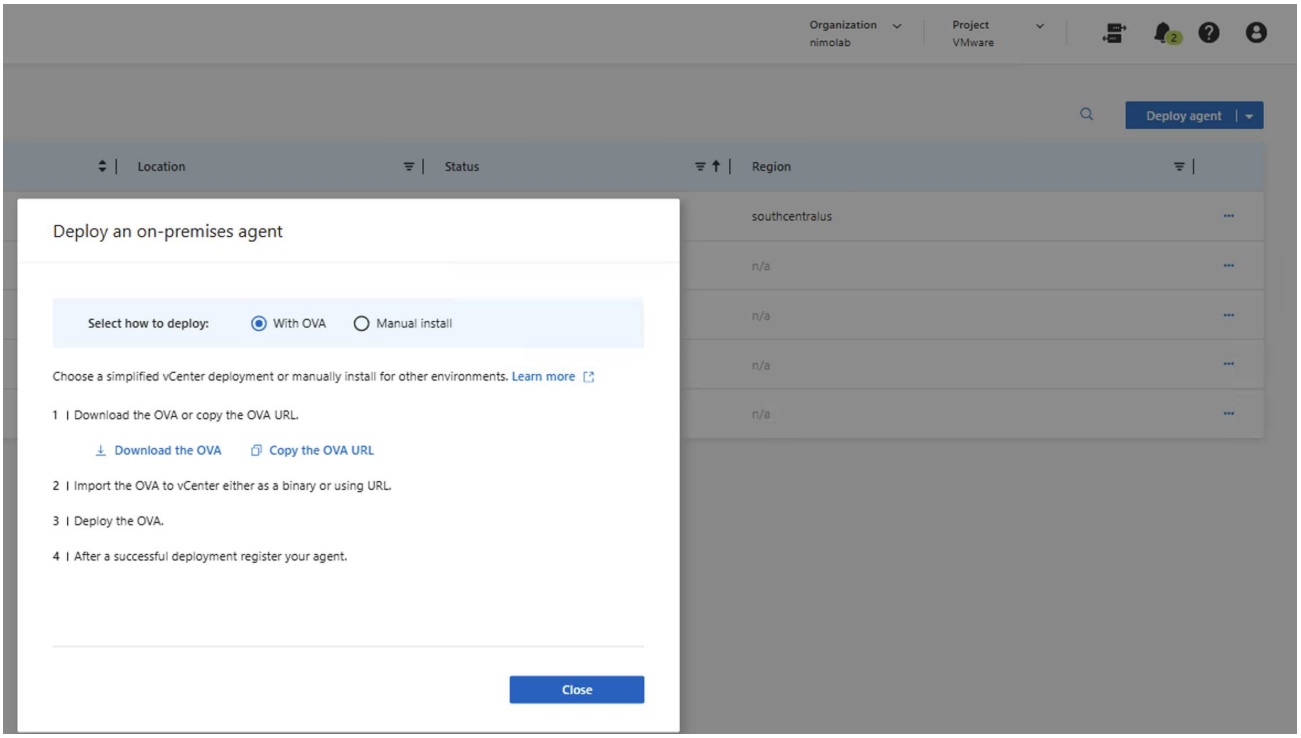
   ▼ **Afficher un exemple**

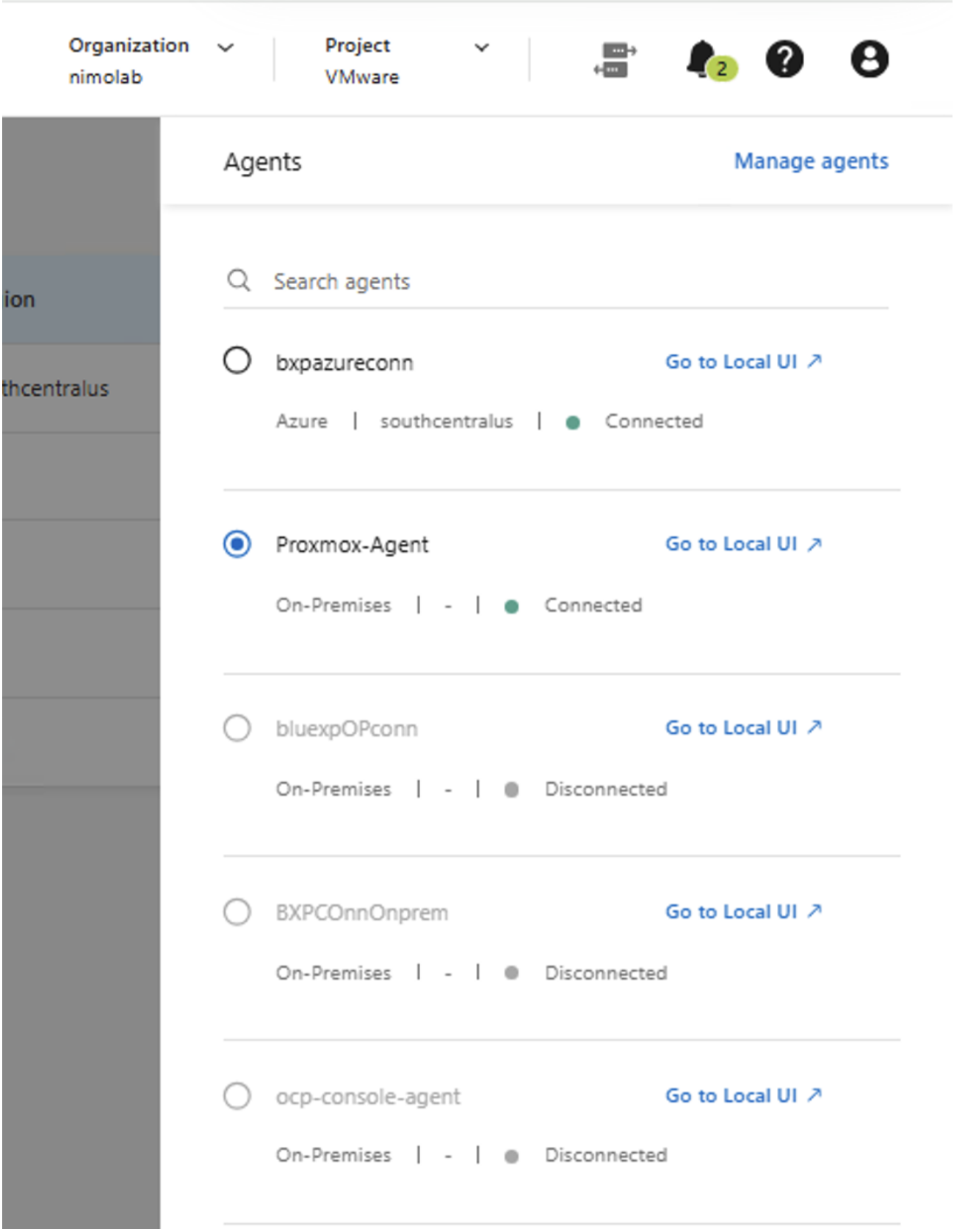
   Dans NetApp Console, cliquez sur Déployer l'agent comme indiqué dans la capture d'écran, puis déployez l'agent dans l'environnement où le cluster VKS est en cours d'exécution.

   Dans cet exemple, sélectionnez **Sur site > Avec OVA**, copiez l'URL OVA et utilisez cette URL dans vCenter pour déployer l'agent Console.

   Enregistrez-le en utilisant l'adresse IP de l'agent dans l'URL. Consultez "la documentation" pour des instructions détaillées. Une fois l'agent enregistré, vous pouvez le voir dans NetApp Console comme

illustré dans la capture d'écran ci-dessous.





2. **Ajoutez le cluster ONTAP à la NetApp Console et activez NetApp Backup and Recovery**

   Le cluster ONTAP principal doit être ajouté à la NetApp Console et le service NetApp Backup and Recovery doit y être activé avant que les charges de travail puissent être protégées. Pour obtenir des instructions, consultez "Configurer NetApp Backup and Recovery".

3. **Dans la NetApp Console, découvrez le cluster Kubernetes vSphere**

▼ **Afficher un exemple**

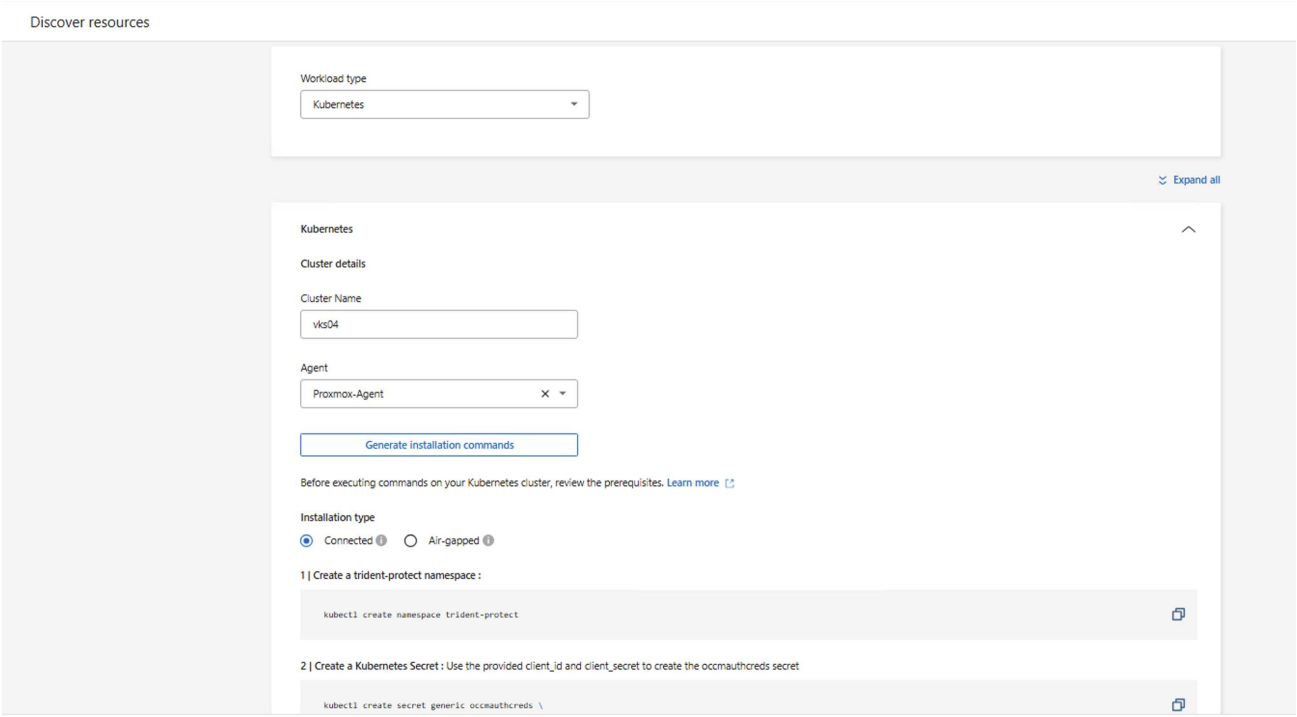
Cette étape nécessite que le cluster Kubernetes vSphere soit en cours d'exécution et que l'agent Console soit déployé dans le même environnement. Suivez ces étapes pour découvrir le cluster Kubernetes vSphere dans la NetApp Console.

a. Dans la NetApp Console, sélectionnez **Inventory > Kubernetes** et fournissez les détails du cluster Kubernetes vSphere.

b. Sélectionnez l'agent déployé dans le même environnement où se trouve le cluster Kubernetes vSphere.

c. Copiez les commandes fournies pour installer Trident Protect et exécutez-les depuis une machine connectée à votre cluster Kubernetes vSphere.

d. Une fois Trident Protect installé avec succès, cliquez sur **Découvrir**. NetApp Console détecte le cluster Kubernetes vSphere et toutes ses ressources via l'agent, puis les affiche dans l'interface utilisateur.



1 | Create a trident-protect namespace :

```
kubectl create namespace trident-protect
```

2 | Create a Kubernetes Secret : Use the provided client_id and client_secret to create the occmauthcreds secret

```
kubectl create secret generic occmauthcreds \
--namespace=trident-protect \
--from-literal=client_id=PZBjqkcEVZeeg-fJECfCRm42rO1EOKrq \
--from-literal=client_secret=5-XbW8MoCnQKb7ImJSIdTC0pef5W4klpmTqeLTeDU5s8ArZWYJuup7OuNdgyBJbz
```

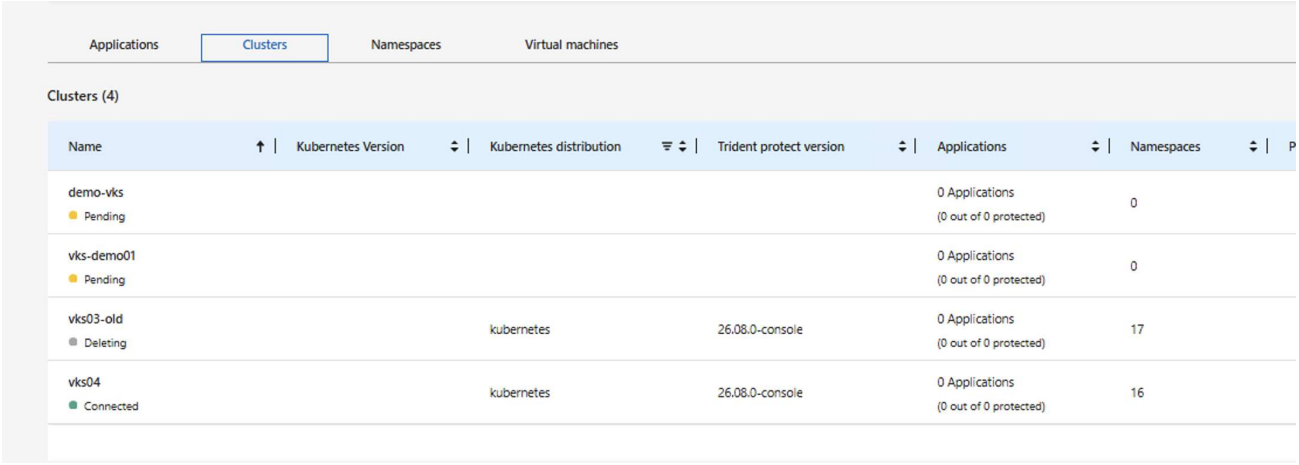
3 | Add/update helm repo :

```
helm repo add --force-update netapp-trident-protect https://netapp.github.io/trident-protect-helm-chart
```

4 | Install or Upgrade Trident Protect and Trident Protect Connector

```
helm upgrade --install trident-protect \
netapp-trident-protect/trident-protect-console \
--version 100.2608.0-console \
--namespace trident-protect \
--set clusterName=vks04 \
--set clusterID=966e65b1-c36a-441b-b23b-99db28c33151 \
--set trident-protect.cbs.accountID=account-B6oEAubL \
--set trident-protect.cbs.agentID=kgl3b3JQQR7e0NxidFg1wH3XockAb8p7clients \
--set trident-protect.cbs.proxySecretName=occmauthcreds \
--set trident-protect.cbs.proxyHostIP=https://10.192.113.198
```

```
vksbastion@vks:~/vks04$ kubectl get pods -n trident-protect
NAME                                                READY   STATUS    RESTARTS   AGE
trident-protect-connector-86bf87f7d7-76llf          1/1     Running   0          29s
trident-protect-controller-manager-8567cf76b9-lxxqz   1/1     Running   0          29s
vksbastion@vks:~/vks04$
```



4. **Configurer ONTAP S3 comme cible de sauvegarde**

   Dans cet exemple, nous utilisons le stockage d'objets ONTAP S3 comme cible de sauvegarde. Vous pouvez également utiliser AWS S3, Azure Blob, Google Cloud Storage ou StorageGRID comme cibles de sauvegarde.

   Pour plus de détails, consultez "Documentation NetApp Console".

   Pour ajouter ONTAP S3 comme cible de sauvegarde dans NetApp Console, vous aurez besoin des informations suivantes provenant de votre cluster ONTAP.

```
S3 Endpoint:
Access Key:
Secret Key:
Bucket Name:
```

Pour obtenir ces valeurs, procédez comme suit dans votre cluster ONTAP à l'aide de System Manager.

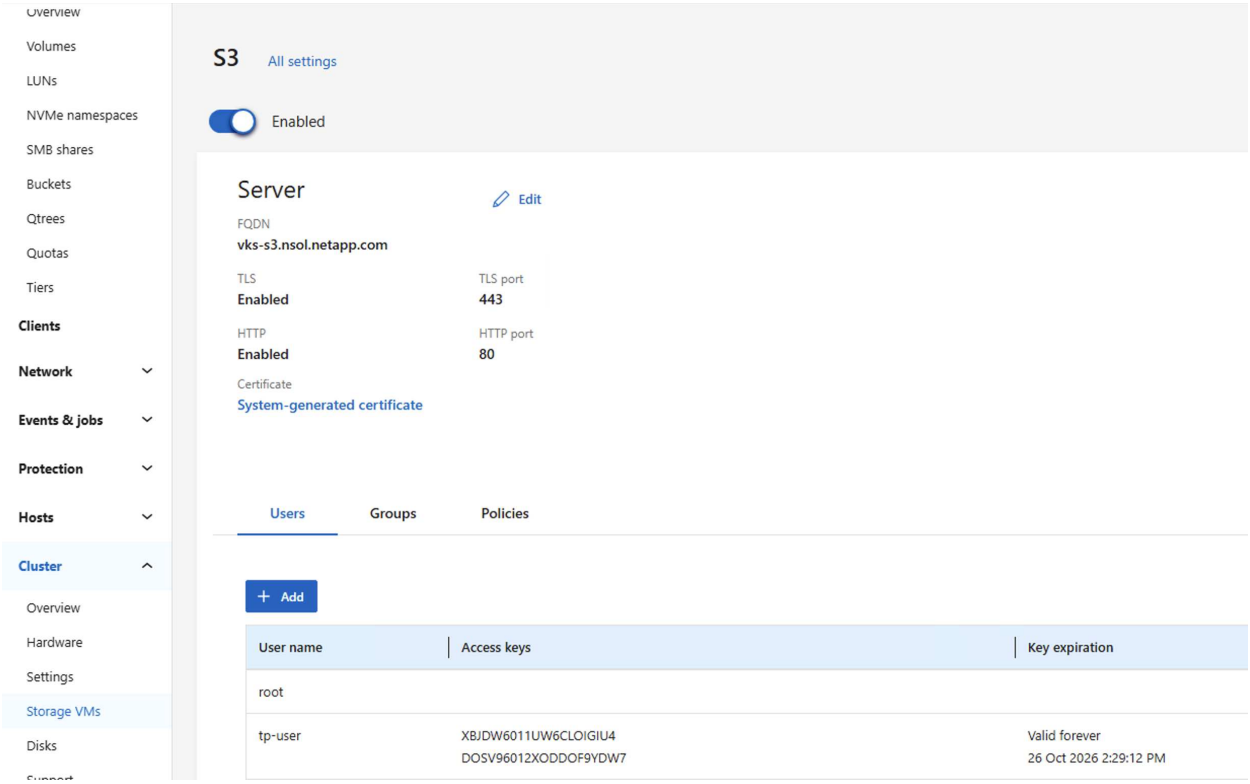
a. Créez une SVM compatible S3 dans le cluster ONTAP pour stocker les données de sauvegarde. Notez l'URL du point de terminaison S3 de cette SVM.

b. Dans les paramètres S3 de la SVM, créez un utilisateur et attribuez-lui les droits d'accès requis. Notez la clé d'accès et la clé secrète de cet utilisateur.

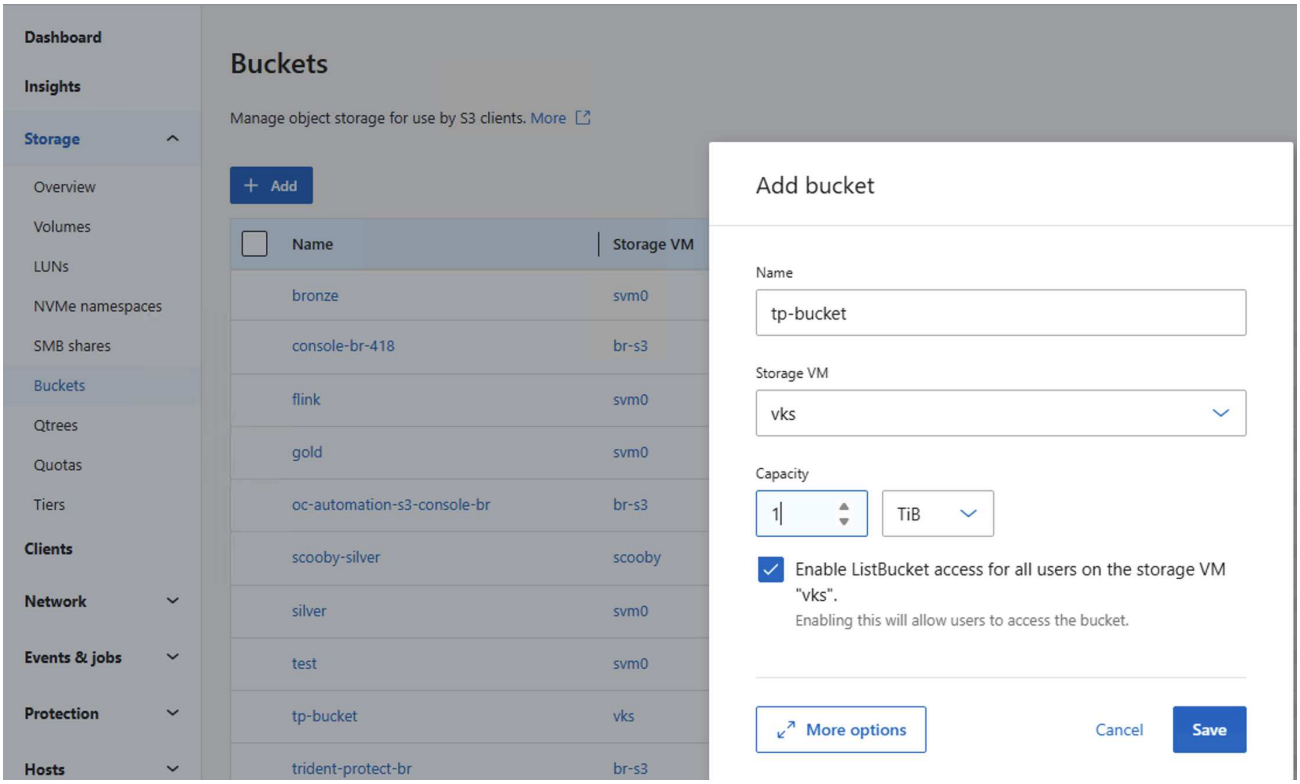
> Trident Protect exige que l'utilisateur S3 dispose au minimum des autorisations `PutObject`, `GetObject`, `ListBucket` et `DeleteObject`. Sans ces autorisations, les opérations de sauvegarde et de restauration AppVault échouent avec des erreurs d'accès refusé. Pour plus de détails, consultez "Documentation de Trident Protect AppVault".

▼ Afficher un exemple

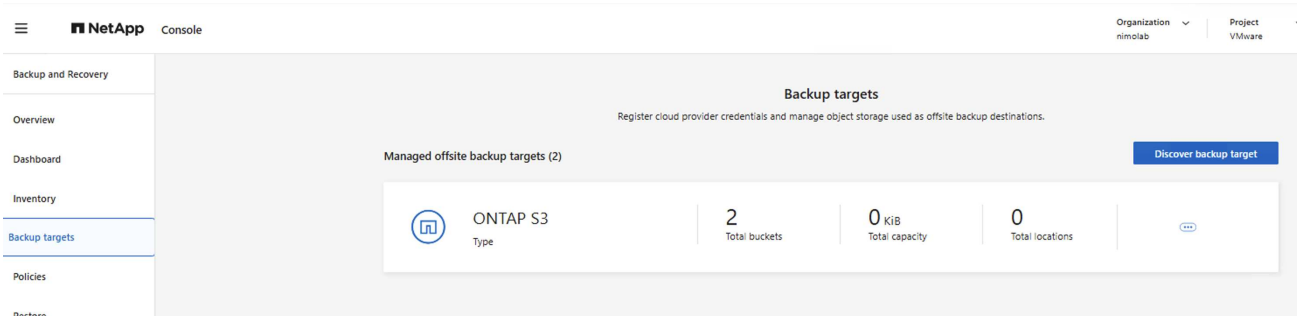
i. **Créez un utilisateur S3 et téléchargez la clé d'accès et la clé secrète**

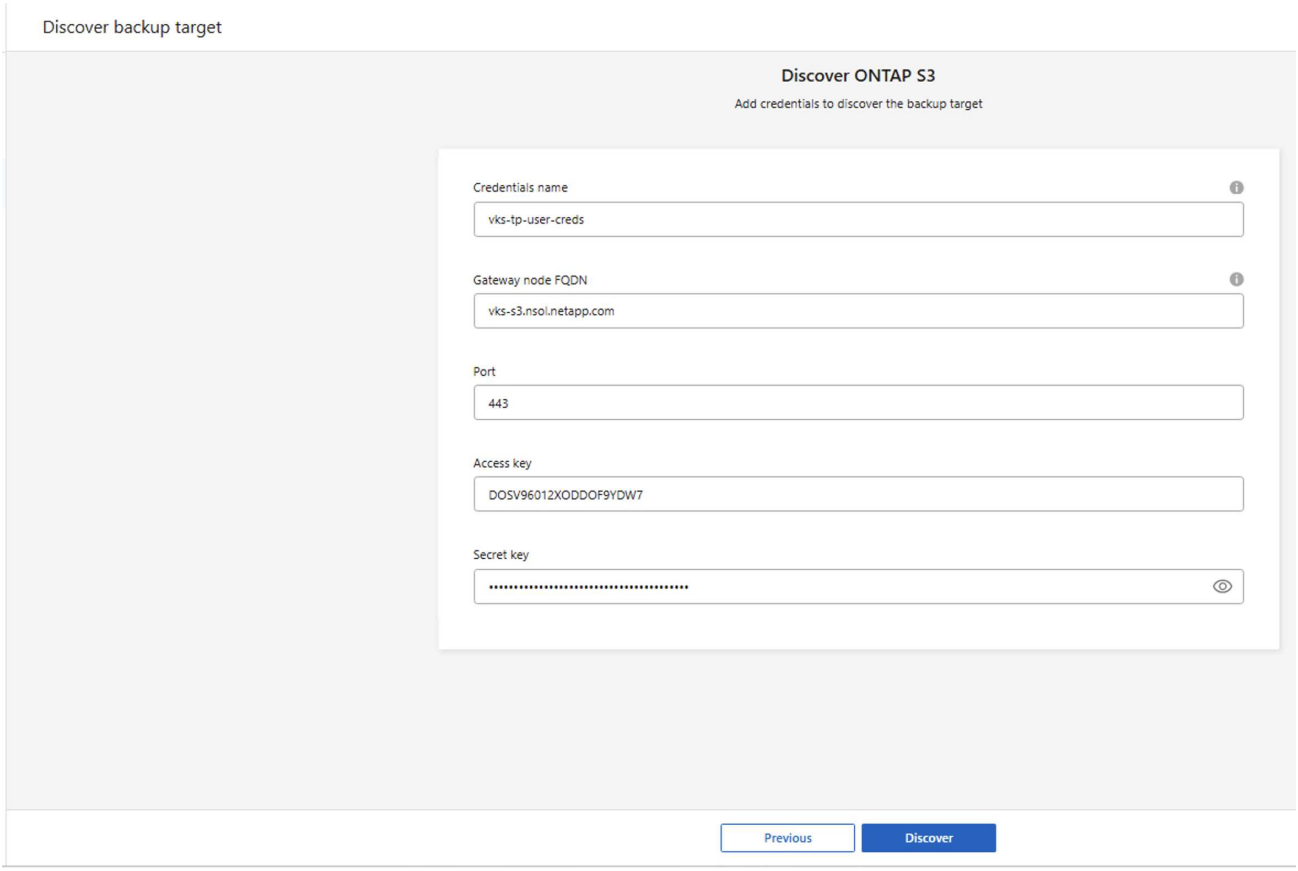


A. Créez un compartiment dans la SVM compatible S3. Notez le nom du compartiment à utiliser lors de l'ajout du stockage d'objets ONTAP S3 comme cible de sauvegarde dans la NetApp Console.



Ajoutez le stockage d'objets ONTAP S3 comme cible de sauvegarde dans la NetApp Console en utilisant le point de terminaison S3, la clé d'accès, la clé secrète et le nom du compartiment.





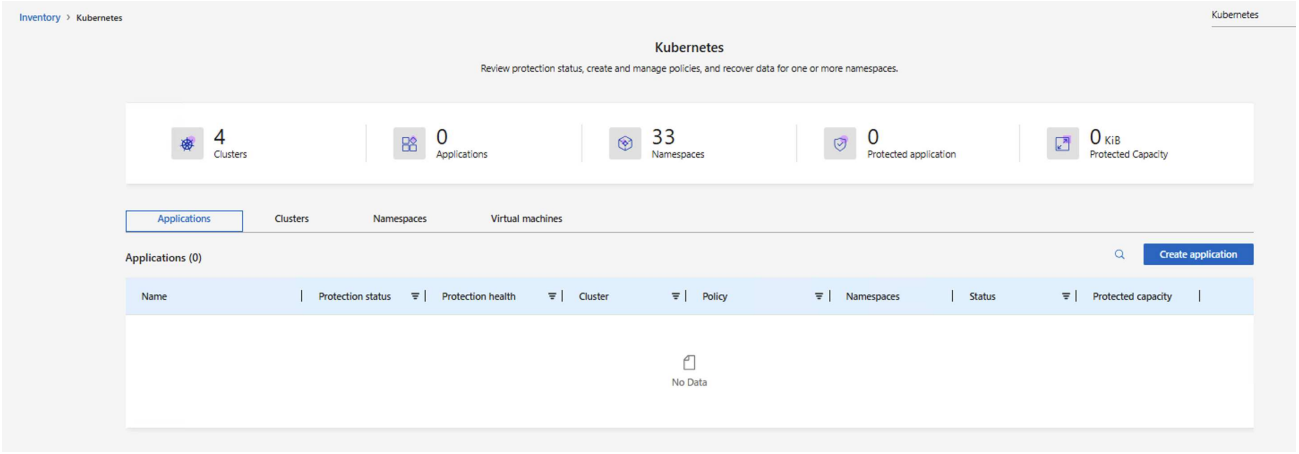
5. **Dans la NetApp Console, créez une application pour les charges de travail conteneurisées et protégez-la à l'aide de sauvegardes.**

> Pour créer et protéger une application Kubernetes, vous devez disposer du rôle de super administrateur Backup and Recovery ou d'administrateur de sauvegarde Backup and Recovery. Pour restaurer une application Kubernetes, vous devez disposer du rôle de super administrateur Backup and Recovery ou d'administrateur de restauration Backup and Recovery.

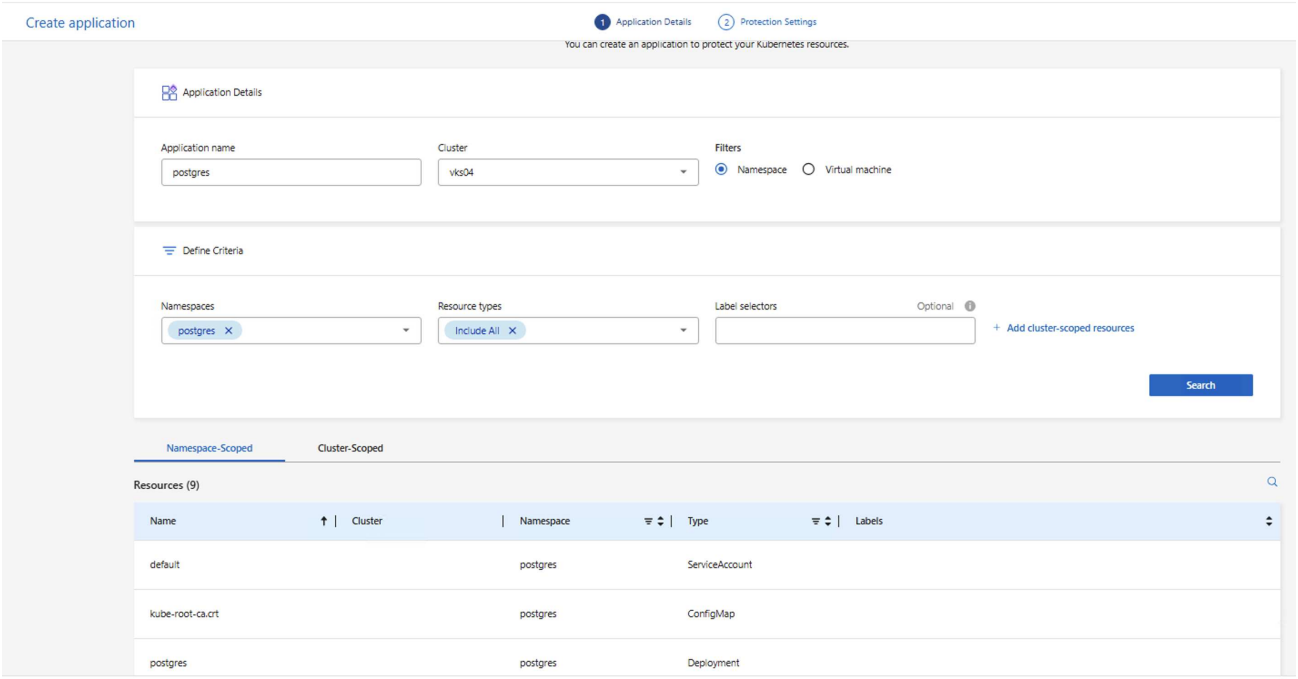
▼ Afficher un exemple

Dans cette étape, créez une application dans la NetApp Console pour les applications conteneurisées de votre cluster Kubernetes vSphere qui doivent être protégées. Vous pouvez utiliser des espaces de noms pour créer une application afin de protéger toutes les applications conteneurisées dans un espace de noms du cluster Kubernetes vSphere.

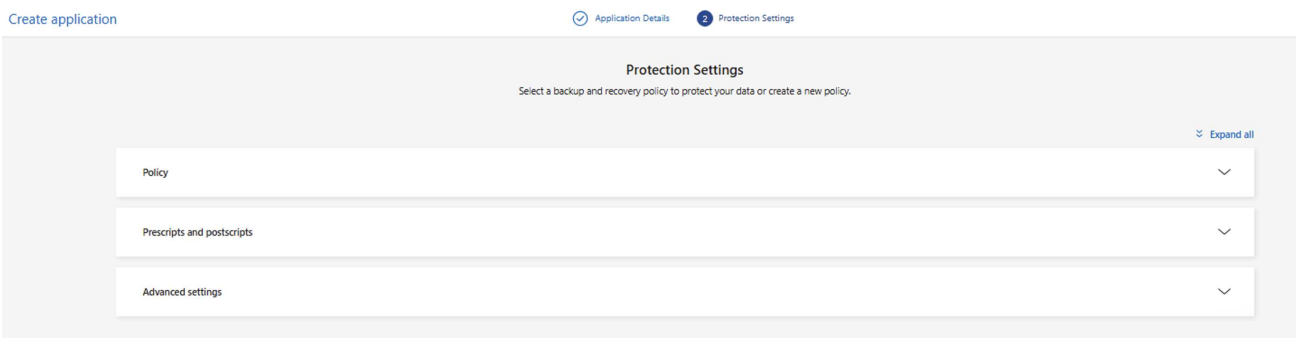
Vous aurez également besoin d'une stratégie de protection à associer à l'application. Vous pouvez créer une nouvelle stratégie de protection ou en utiliser une existante.



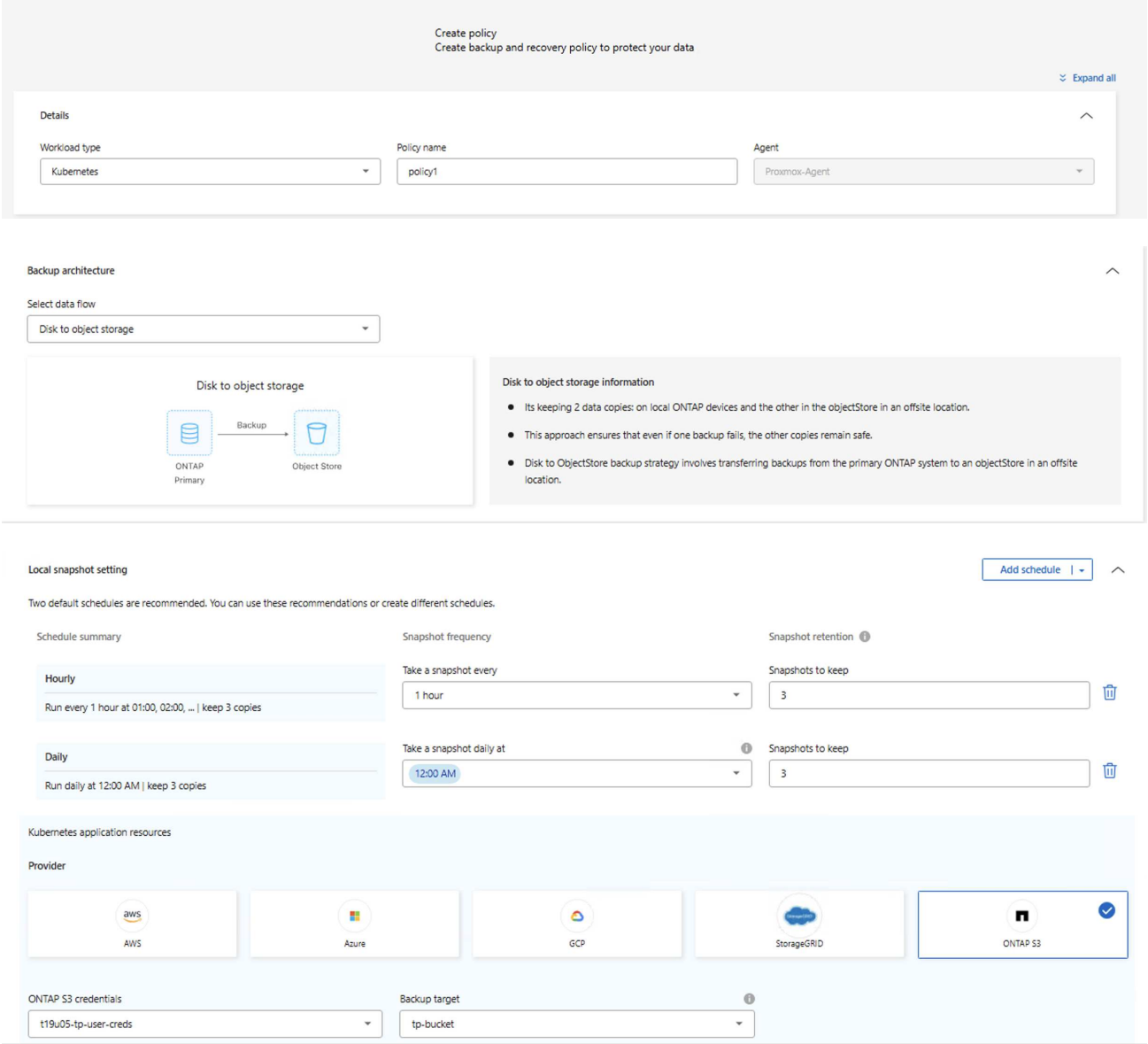
Fournissez les détails de l'application et définissez les critères de l'application. Dans cet exemple, l'application est créée pour toutes les applications conteneurisées dans un espace de noms du cluster Kubernetes vSphere. Cliquez sur **Rechercher** pour afficher la liste des ressources à portée d'espace de noms qui font partie de l'application. Cliquez sur **Suivant** pour continuer.

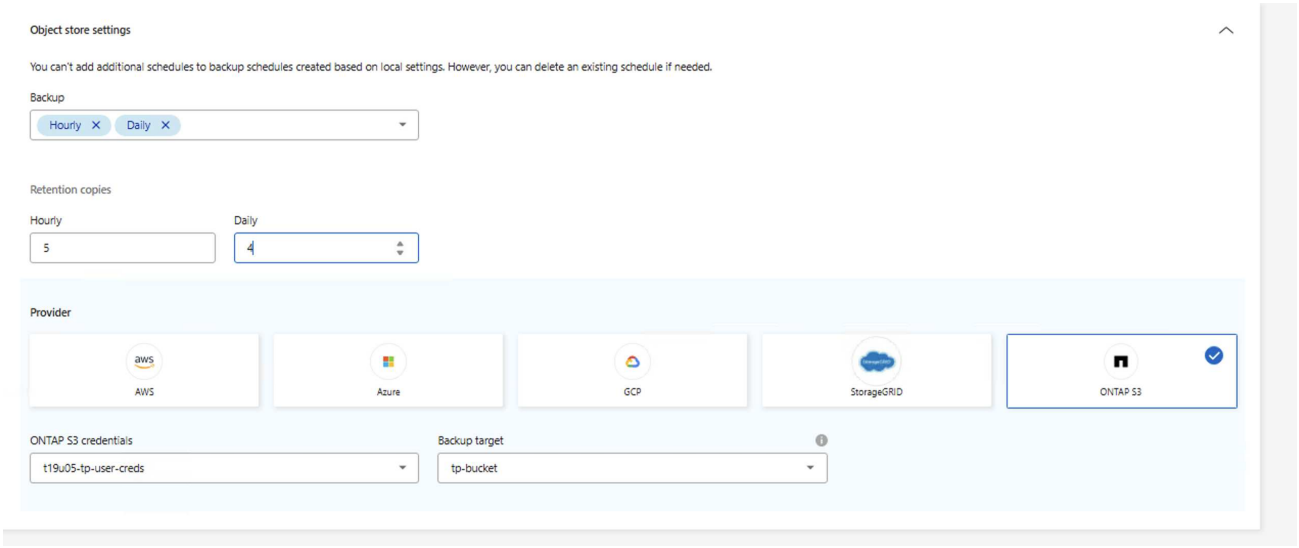


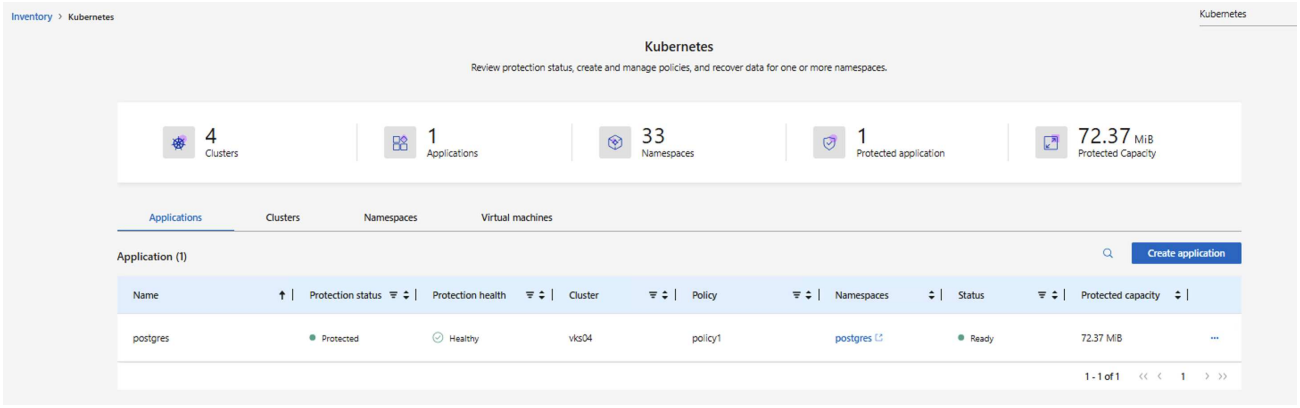
Ensuite, vous devez définir les paramètres de protection de l'application. Vous pouvez créer une nouvelle stratégie de protection ou utiliser une stratégie existante. Dans cet exemple, une nouvelle stratégie de protection est créée pour l'application et lui est associée. Développez **Policy** et cliquez sur **Create new policy**.



Attribuez un nom à la stratégie et choisissez l'architecture de sauvegarde. Dans cet exemple, la sauvegarde de disque vers stockage objet est sélectionnée. Définissez la planification des instantanés locaux, qui contrôle la fréquence à laquelle les instantanés sont pris sur le stockage principal. Configurez ensuite séparément les paramètres de sauvegarde du stockage objet : choisissez la cible de sauvegarde (dans cet exemple, ONTAP S3), définissez la planification des transferts qui contrôle le moment où les données d'instantané sont copiées vers le stockage objet, et spécifiez le nombre de copies de rétention. La planification des transferts est indépendante de la planification des instantanés, ainsi les ressources sont copiées vers le stockage objet selon la planification des transferts, et non immédiatement après la prise de chaque instantané. Vous pouvez également modifier le nombre maximal de tentatives et l'intervalle entre les tentatives si nécessaire. Cliquez sur **Créer** pour continuer. L'application sera découverte et la stratégie de protection lui sera associée.





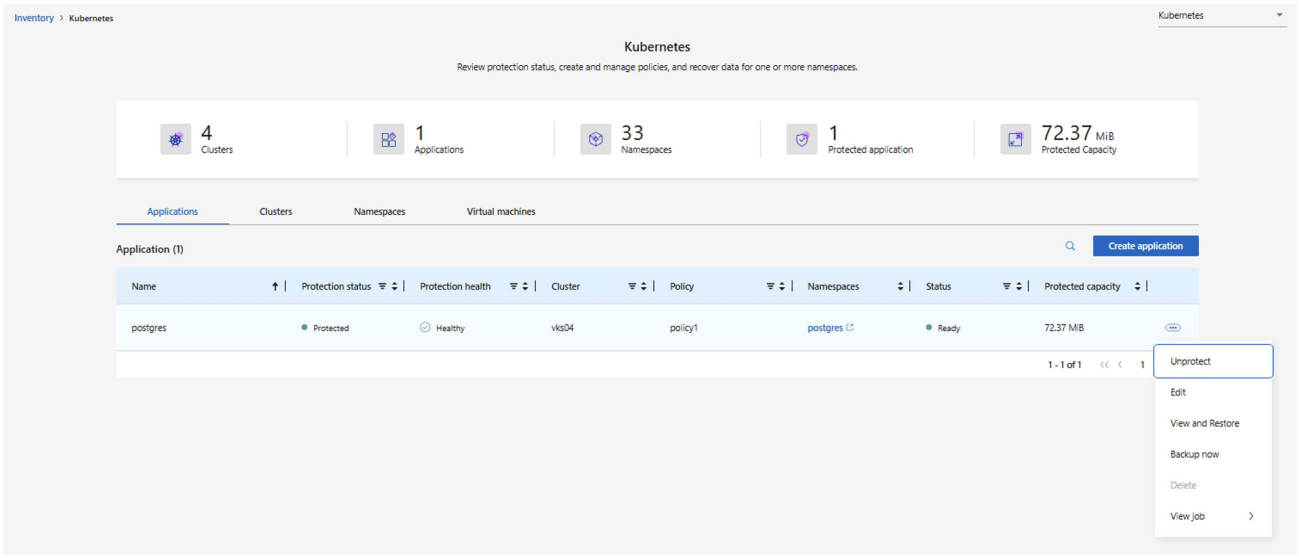


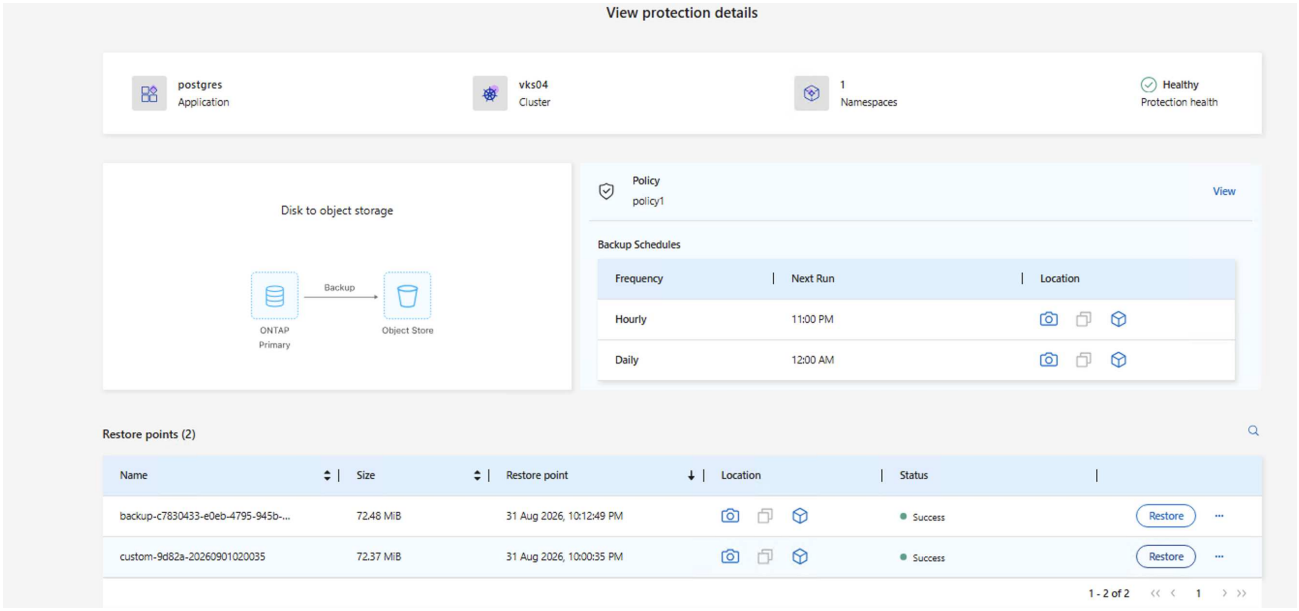
6. **Restaurez l’application à partir d’une sauvegarde**

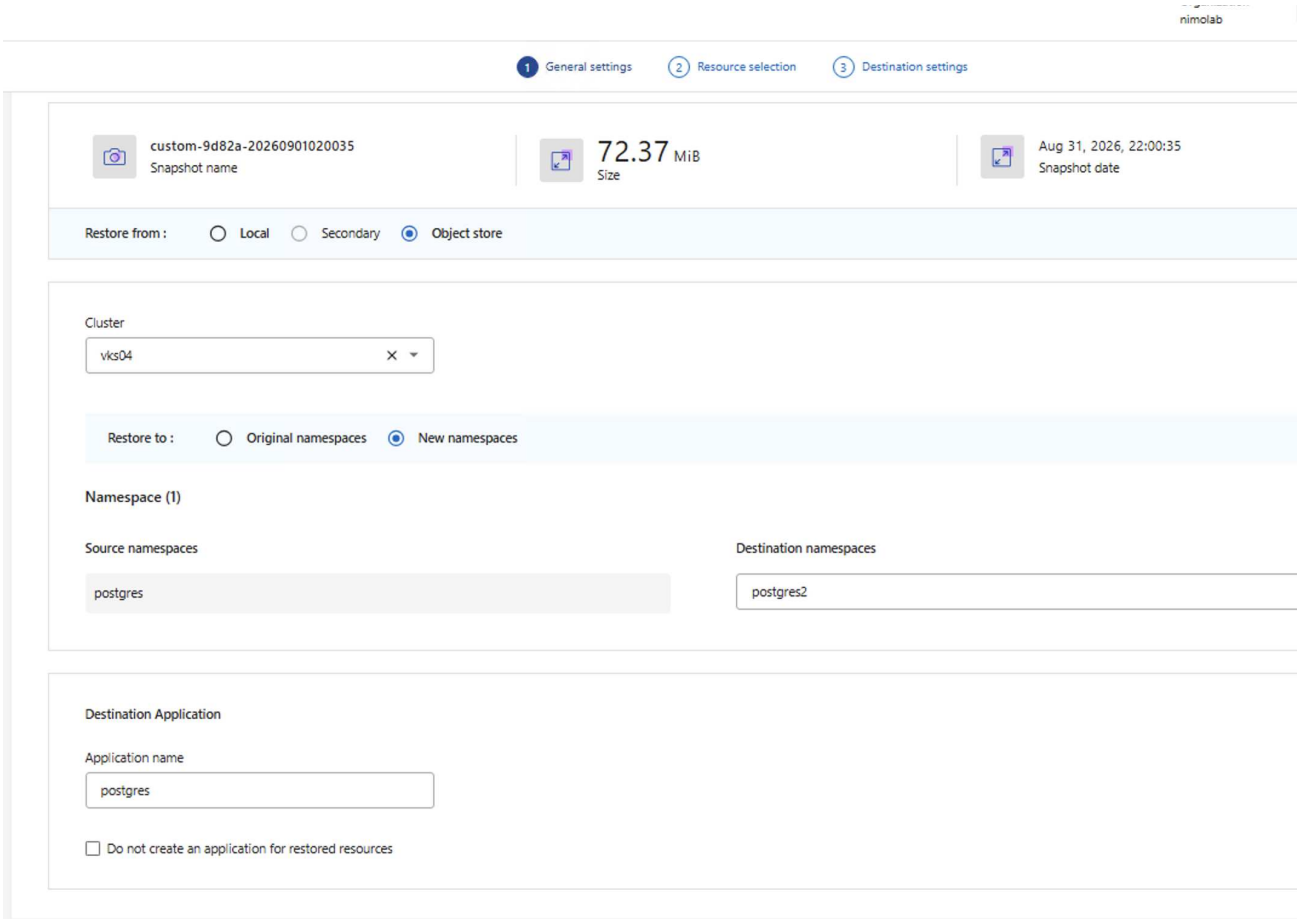
Vous avez besoin du rôle de super administrateur NetApp Backup and Recovery ou du rôle d’administrateur de restauration NetApp Backup and Recovery pour restaurer une application Kubernetes.

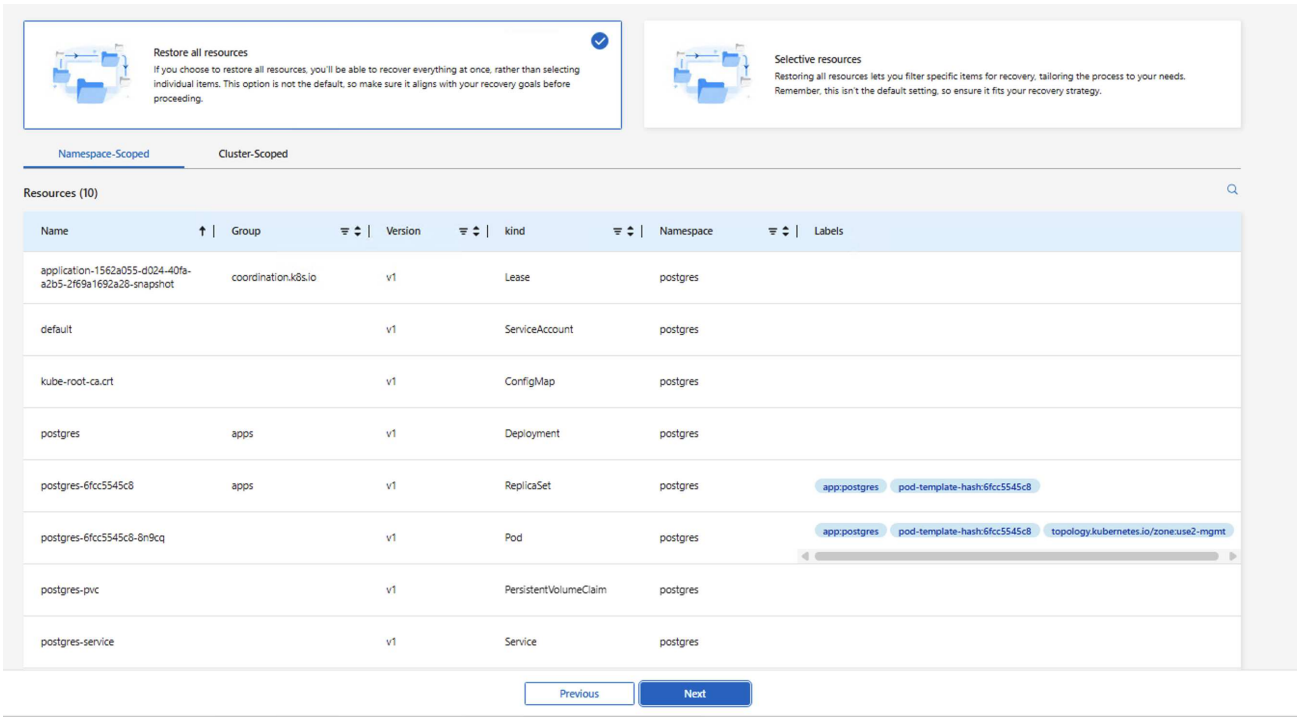
▼ Afficher un exemple

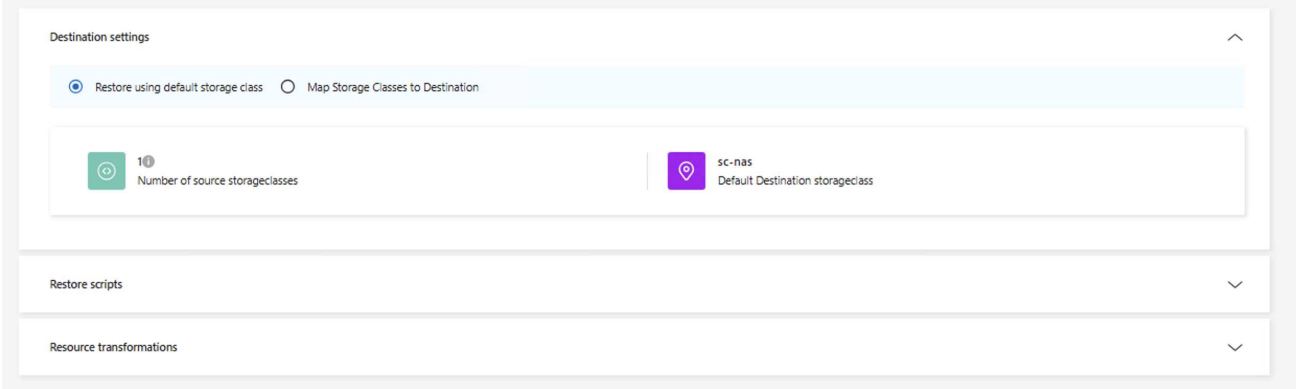
- Vous pouvez choisir n’importe quel point de restauration disponible — un instantané local, une sauvegarde de stockage d’objets ou une sauvegarde provenant d’une cible secondaire — pour restaurer l’application.
- Vous pouvez restaurer l’application dans son espace de noms d’origine ou dans un espace de noms différent.
- Vous pouvez sélectionner les ressources à inclure dans la restauration.
- Vous pouvez choisir la classe de stockage à utiliser pour les ressources d’application restaurées.


Inventory › Kubernetes
Kubernetes
Kubernetes
Review protection status, create and manage policies, and recover data for one or more namespaces.
4 Clusters
1 Applications
33 Namespaces
1 Protected application
72.37 MiB Protected Capacity
Applications
Clusters
Namespaces
Virtual machines
Application (1)
Create application
Name
Protection status
Protection health
Cluster
Policy
Namespaces
Status
Protected capacity
postgres
Protected
Healthy
vks04
policy1
postgres
Ready
72.37 MiB
1 - 1 of 1
Unprotect
Edit
View and Restore
Backup now
Delete
View job

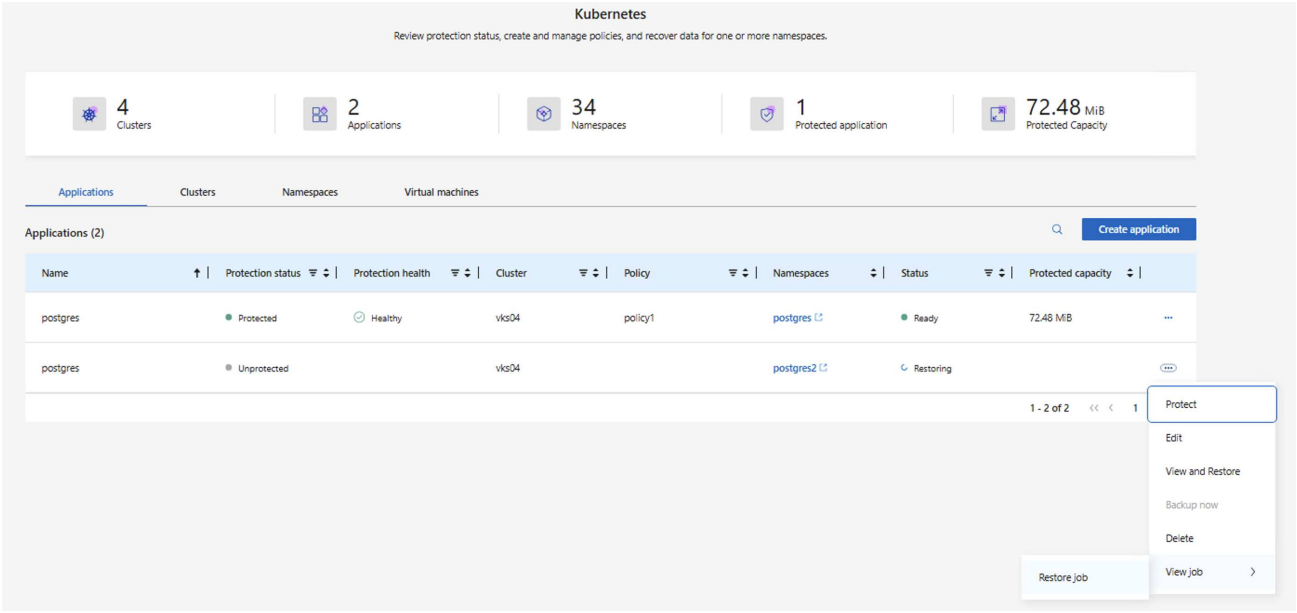

View protection details
postgres
Application
vks04
Cluster
1
Namespaces
Healthy
Protection health
Disk to object storage
Backup
ONTAP Primary
Object Store
Policy
policy1
View
Backup Schedules
Frequency
Next Run
Location
Hourly
11:00 PM
Daily
12:00 AM
Restore points (2)
Name
Size
Restore point
Location
Status
backup-c7830433-e0eb-4795-945b-...
72.48 MiB
31 Aug 2026, 10:12:49 PM
Success
Restore
custom-9d82a-20260901020035
72.37 MiB
31 Aug 2026, 10:00:35 PM
Success
Restore
1 - 2 of 2

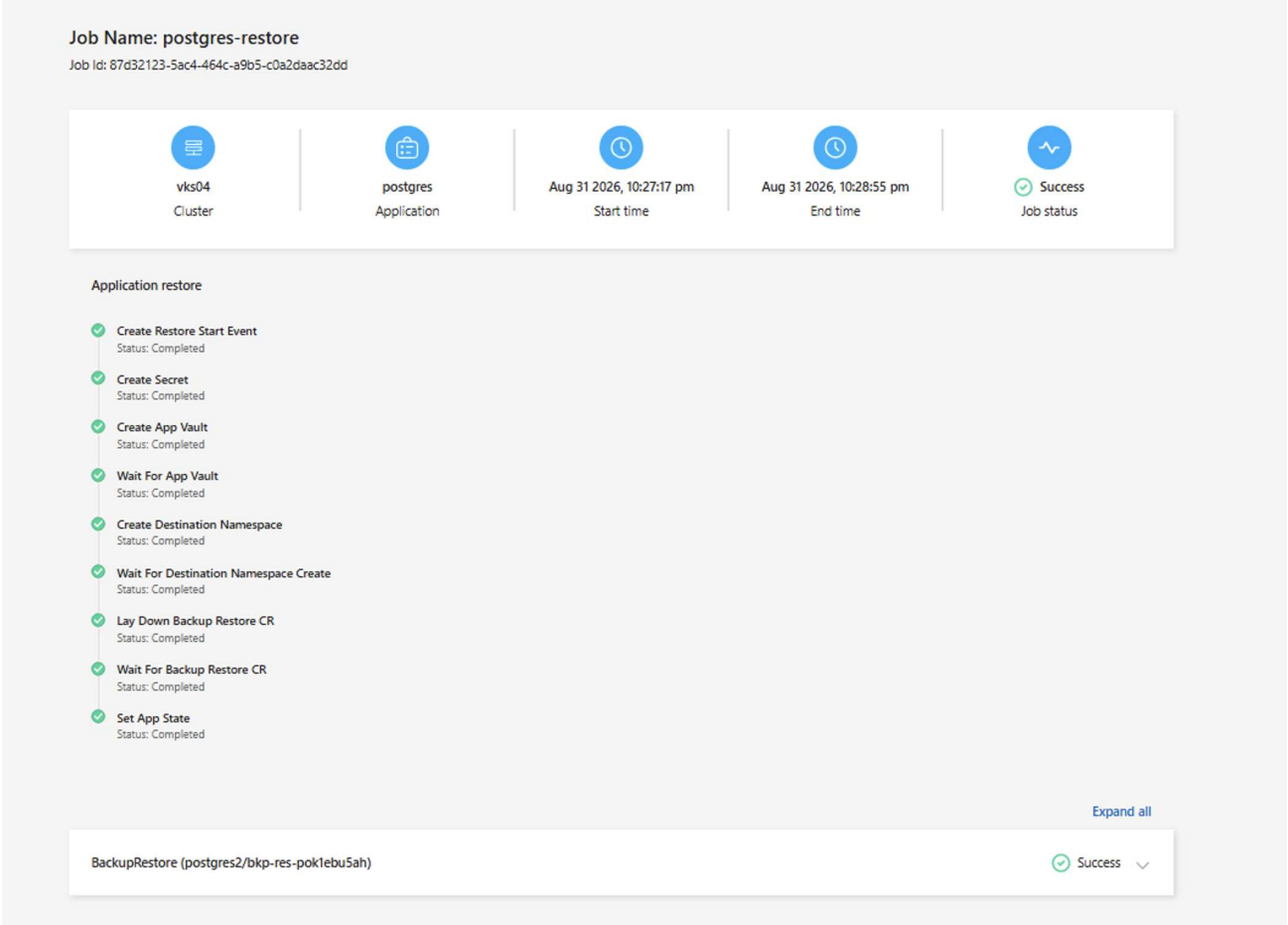

nimolab
1 General settings
2 Resource selection
3 Destination settings
custom-9d82a-20260901020035
Snapshot name
72.37 MiB
Size
Aug 31, 2026, 22:00:35
Snapshot date
Restore from :
Local
Secondary
Object store
Cluster
vks04
Restore to :
Original namespaces
New namespaces
Namespace (1)
Source namespaces
postgres
Destination namespaces
postgres2
Destination Application
Application name
postgres
Do not create an application for restored resources

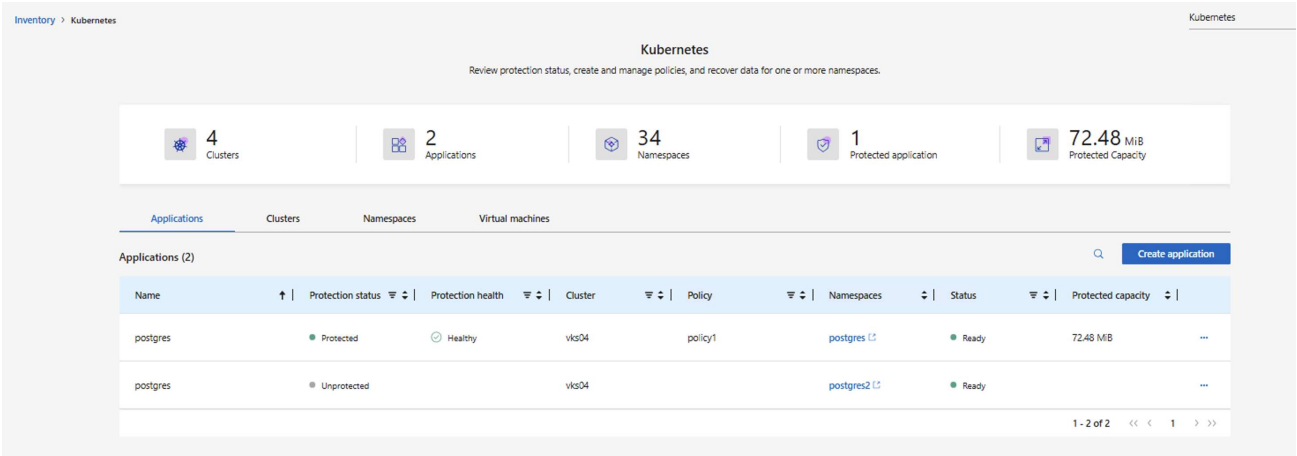

Restore all resources
If you choose to restore all resources, you'll be able to recover everything at once, rather than selecting individual items. This option is not the default, so make sure it aligns with your recovery goals before proceeding.
Selective resources
Restoring all resources lets you filter specific items for recovery, tailoring the process to your needs. Remember, this isn't the default setting, so ensure it fits your recovery strategy.
Namespace-Scoped
Cluster-Scoped
Resources (10)
Name
Group
Version
kind
Namespace
Labels
application-1562a055-d024-40fa-a2b5-2f69a1692a28-snapshot
coordination.k8s.io
v1
Lease
postgres
default
v1
ServiceAccount
postgres
kube-root-ca.crt
v1
ConfigMap
postgres
postgres
apps
v1
Deployment
postgres
postgres-6fcc5545c8
apps
v1
ReplicaSet
postgres
app:postgres
pod-template-hash:6fcc5545c8
postgres-6fcc5545c8-8n9cq
v1
Pod
postgres
app:postgres
pod-template-hash:6fcc5545c8
topology.kubernetes.io/zone:use2-mgmt
postgres-pvc
v1
PersistentVolumeClaim
postgres
postgres-service
v1
Service
postgres
Previous
Next



Vous pouvez suivre la progression du travail de restauration dans la section Surveillance de la NetApp Console. Après l'achèvement du travail de restauration, vous pouvez voir les ressources restaurées dans le cluster Kubernetes vSphere.







Sauvegardez et restaurez des applications conteneurisées dans un cluster vSphere Kubernetes Service à l'aide de Trident Protect

Sauvegardez et restaurez des applications conteneurisées dans un cluster vSphere Kubernetes Service (VKS) à l'aide des snapshots et sauvegardes Trident Protect. Cette procédure comprend la création d'un AppVault à l'aide du stockage d'objets ONTAP S3, la configuration de Trident Protect pour capturer les données applicatives, y compris les objets de ressources Kubernetes et les volumes persistants, ainsi que la restauration des données en cas de besoin.

Les applications conteneurisées exécutées dans un cluster VKS sont gérées comme des charges de travail

Kubernetes dans les espaces de noms des nœuds de travail. Il est important de protéger à la fois les métadonnées de l'application et les volumes persistants afin que, s'ils sont perdus ou corrompus, vous puissiez les récupérer.

Les volumes persistants des applications VKS peuvent être pris en charge par le stockage ONTAP intégré au cluster VKS à l'aide de "Trident CSI". Cette procédure utilise "Trident Protect" pour créer des instantanés d'application, dont les métadonnées sont stockées dans le stockage objet ONTAP tandis que les données des instantanés de volume restent sur le stockage principal, et des sauvegardes, dont les données sont copiées dans le stockage objet ONTAP. Vous pouvez restaurer à partir d'un instantané ou d'une sauvegarde en cas de besoin.

Trident Protect permet la création de snapshots, la sauvegarde, la restauration et la reprise après sinistre des applications sur un cluster Kubernetes. Les données pouvant être protégées avec Trident Protect incluent les objets de ressources Kubernetes associés aux applications et leurs volumes persistants.

Voici les versions des différents composants utilisés dans les exemples de cette section

- VMware Cloud Foundation (VCF) 9.1 avec le service Kubernetes vSphere
- "Trident 26.06"
- "Trident Protect 26.06"
- "ONTAP 9.16"

Installez Trident Protect sur le cluster Kubernetes vSphere

▼ Installer Trident Protect

Utilisez Helm pour installer Trident Protect sur le cluster vSphere Kubernetes Service. Les exemples de cette section utilisent `tridentctl-protect` l'interface de ligne de commande Trident Protect (CLI). Installez-le en suivant les instructions dans le "Documentation de l'interface de ligne de commande Trident Protect". D'autres méthodes d'installation de Trident Protect sont documentées dans le "Documentation Trident Protect".

Commencez par créer et étiqueter l `trident-protect`espace de noms. L' `enforce=privileged`étiquette est requise car Trident Protect exécute des charges de travail privilégiées. Sans elle, le contrôleur d'admission de la sécurité des pods Kubernetes empêchera le démarrage des pods Trident Protect.

```
kubectl create namespace trident-protect
kubectl label namespace trident-protect pod-
security.kubernetes.io/enforce=privileged
```

Ajoutez ensuite le dépôt Helm et installez Trident Protect dans l'espace de noms.

```
helm repo add netapp-trident-protect https://netapp.github.io/trident-
protect-helm-chart
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2606.0 --namespace
trident-protect
```

Si les pods Trident Protect ne démarrent pas avec une erreur d'admission `PodSecurity`, il

est possible que les étiquettes de sécurité des pods n'aient pas été appliquées à l'espace de noms `trident-protect` avant l'installation. Utilisez `--overwrite` pour les réappliquer, puis vérifiez que les pods démarrent.

```
kubectl label namespace trident-protect pod-
security.kubernetes.io/enforce=privileged --overwrite
```

```
kubectl get pods -n trident-protect
```

```
vksbastion@vks:~/demo-vks$ kubectl get pods -n trident-protect
NAME                                                READY   STATUS    RESTARTS   AGE
autosupportbundle-e4cb1693-6640-4fcd-bddf-b413ff41f37c-znvsw   1/1     Running   0          16h
trident-protect-controller-manager-5f64ff594f-cl8cp            1/1     Running   0          3h50m
vksbastion@vks:~/demo-vks$
```

```
vksbastion@vks:~/demo-vks$ tridentctl-protect version -n trident-protect
26.06.0
vksbastion@vks:~/demo-vks$
```

Créer un coffre-fort d'applications pour le stockage d'objets

▼ Créer AppVault

Avant de créer des instantanés et des sauvegardes pour une application, vous devez configurer le stockage objet dans Trident Protect. Cela se fait en créant un CR AppVault. Seuls les administrateurs peuvent créer et configurer un CR AppVault.

Les objets AppVault sont la représentation de ressource personnalisée Kubernetes d'un bucket de stockage. Une ressource personnalisée AppVault contient les configurations nécessaires pour qu'un bucket soit utilisé dans les opérations de protection, telles que les sauvegardes, les snapshots, les opérations de restauration et la réplication SnapMirror.

Les étapes suivantes créent une AppVault CR configurée pour ONTAP S3 : . Créez un serveur de stockage d'objets S3 dans la SVM du cluster ONTAP. . Créez un compartiment dans le serveur de stockage d'objets. . Créez un utilisateur S3 dans la SVM. Conservez la clé d'accès et la clé secrète en lieu sûr.

+ REMARQUE : Trident Protect exige que l'utilisateur S3 dispose au moins des autorisations `PutObject`, `GetObject`, `ListBucket` et `DeleteObject`. Sans ces autorisations, les opérations de sauvegarde et de restauration AppVault échouent avec des erreurs d'accès refusé. Pour plus de détails, consultez "Documentation de Trident Protect AppVault". . Dans le cluster VKS, créez un secret pour stocker les identifiants ONTAP S3. . Créez un objet AppVault pour ONTAP S3.

Configurer Trident Protect AppVault pour ONTAP S3

Le manifeste ci-dessous utilise HTTPS avec validation de certificat activée, ce qui est requis pour la production. Si votre point de terminaison ONTAP S3 utilise un certificat signé par une autorité de certification publique déjà approuvée par le cluster, aucune configuration de certificat supplémentaire n'est nécessaire. S'il utilise un certificat auto-signé ou un certificat d'autorité de certification interne, fournissez le certificat d'autorité de certification racine personnalisé à l'aide du champ `rootCA` comme indiqué dans le manifeste. Consultez la variante réservée aux tests en laboratoire à la fin de cette section si vous avez besoin d'une

configuration HTTP pour des tests isolés hors production.

```
# alias tp='tridentctl-protect'

# cat appvault-secret.yaml
apiVersion: v1
data:
  accessKeyID: <base64 encoded access key>
  secretAccessKey: <base64 encoded secret access key>
# Alternatively, remove the data section above and use:
# stringData:
#   accessKeyID: "<access key of S3>"
#   secretAccessKey: "<secret access key of S3>"
kind: Secret
metadata:
  name: ontap-s3-appvault-secret1
  namespace: trident-protect
type: Opaque

# cat appvault.yaml
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-appvault1
  namespace: trident-protect
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: trident-protect
      endpoint: <lif for S3 access>
      secure: "true"
      skipCertValidation: "false"
      # If your ONTAP S3 endpoint uses a certificate signed by a public
CA
      # already trusted by the cluster, no additional certificate config
is needed.
      # If it uses a self-signed or internal CA certificate, provide the
      # root CA certificate:
      # rootCA: <root CA certificate>
```

```
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: ontap-s3-appvault-secret1
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: ontap-s3-appvault-secret1
  providerType: OntapS3

# kubectl create -f appvault-secret.yaml -n trident-protect
# kubectl create -f appvault.yaml -n trident-protect
```

Variante réservée au laboratoire (HTTP, sans validation de certificat)

Utilisez uniquement les paramètres suivants `s3` dans un environnement de laboratoire isolé où aucune donnée sensible n'est présente. N'utilisez pas ces paramètres en production.

```
    s3:
      bucketName: trident-protect
      endpoint: <lif for S3 access>
      secure: "false"
      skipCertValidation: "true"
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get appvault -n trident-protect
NAME                STATE       ERROR   MESSAGE   AGE
ontap-s3-appvault1  Available                     3d14h
vks-appvault        Available                     5d15h
vksbastion@vks:~/demo-vks/tp$
```

Créer une application Trident Protect

▼ Créer une application Trident Protect

Cet exemple traite de la protection des données pour l'application postgres d'exemple, qui est installée dans l'espace de noms postgres. Pour plus de détails sur l'installation, voir "Déployez des charges de travail VKS à l'aide du stockage NetApp".

> ⓘ Les exemples de PVC PostgreSQL requièrent le mode d'accès RWO. Le mode d'accès doit être explicitement spécifié dans le manifeste du PVC ; Trident ne sélectionne pas automatiquement un mode d'accès en fonction du protocole de stockage. RWX est pris en charge pour les PVC reposant sur NAS, et les PVC reposant sur SAN peuvent prendre en charge RWX moyennant une configuration supplémentaire. Pour plus d'informations, consultez la "Documentation Trident Protect".

Dans cet exemple, l'espace de noms postgres contient une application et toutes les ressources de cet espace de noms sont incluses lors de la création de l'application Trident Protect.

```
# alias tp='tridentctl-protect'
# tp create app postgres-app --namespaces postgres -n postgres --dry-run
> postgres-app.yaml

# cat postgres-app.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: postgres-app
spec:
  includedNamespaces:
  - namespace: postgres
  resourceFilter: {}
# kubectl create -f postgres-app.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get application -n postgres
NAME           PROTECTION STATE   AGE
postgres-app   None               3d13h
```

Protégez l'application en créant une sauvegarde

▼ **Créer des sauvegardes**

Créer une sauvegarde à la demande

Créez une sauvegarde pour l'application postgres-app créée précédemment, incluant toutes les ressources dans l'espace de noms postgres. Indiquez le nom de l'appvault où les sauvegardes seront stockées.

```
# cat postgres-backup-on-demand.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: postgres-backup-on-demand
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  cleanupSnapshot: true
  replicateSnapshot: false
```

```
kubectl create -f postgres-backup-on-demand.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-backup-on-demand.yaml -n postgres
backup.protect.trident.netapp.io/postgres-backup-on-demand created
vksbastion@vks:~/demo-vks/tp$ kubectl get backups -n postgres
NAME                        APP           RECLAIM POLICY   STATE     ERROR   AGE
postgres-backup-on-demand   postgres-app   Retain           Running           12s
vksbastion@vks:~/demo-vks/tp$ kubectl get backups -n postgres -w
NAME                        APP           RECLAIM POLICY   STATE     ERROR   AGE
postgres-backup-on-demand   postgres-app   Retain           Running           29s
postgres-backup-on-demand   postgres-app   Retain           Running           82s
postgres-backup-on-demand   postgres-app   Retain           Running           82s
postgres-backup-on-demand   postgres-app   Retain           Running           82s
postgres-backup-on-demand   postgres-app   Retain           Running           83s
postgres-backup-on-demand   postgres-app   Retain           Running           83s
postgres-backup-on-demand   postgres-app   Retain           Running           83s
postgres-backup-on-demand   postgres-app   Retain           Completed          83s
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get application -n postgres
NAME           PROTECTION STATE   AGE
postgres-app   Partial            3d13h
```

Créer des sauvegardes selon un planning

Créez un calendrier pour les sauvegardes en spécifiant la granularité et le nombre de sauvegardes à conserver.

```
# tp create schedule backup-schedule1 --app postgres-app --appvault
ontap-s3-appvault1 --granularity Hourly --minute 45 --backup-retention 1
-n postgres --dry-run>postgres-backup-schedule1.yaml

#cat postgres-backup-schedule1.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: backup-schedule1
  namespace: postgres
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  backupRetention: "1"
  dayOfMonth: ""
  dayOfWeek: ""
  enabled: true
  granularity: Hourly
  hour: ""
  minute: "45"
  recurrenceRule: ""
  replicationRetention: "0"
  runImmediately: false
  snapshotRetention: "0"
# kubectl create -f postgres-backup-schedule1.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-backup-schedule1.yaml -n postgres
schedule.protect.trident.netapp.io/backup-schedule1 created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get schedule -n postgres
NAME               ENABLED   GRANULARITY   STATE   ERROR   AGE
backup-schedule1   true      Hourly                        10m
```

Restauration à partir d'une sauvegarde

▼ **Restaurer à partir de sauvegardes**

Restaurez l'application dans le même espace de noms

Dans cet exemple, la sauvegarde postgres-backup-on-demand contient la sauvegarde pour la postgres-app.

Avant de supprimer l'application, vérifiez que la sauvegarde à partir de laquelle vous prévoyez de restaurer est dans l'état `Completed`. Une sauvegarde en cours ou ayant échoué ne peut pas être utilisée pour restaurer l'application.

```
kubectl get backup -n postgres
```

Après avoir confirmé que l'état de la sauvegarde est `Completed`, supprimez l'application postgres et assurez-vous que les PVC et les objets pod sont supprimés de l'espace de noms "postgres".

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all -n postgres
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-6gw9h   1/1     Running   0          3d13h

NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.101.183.142   <none>        5432/TCP   3d13h

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           3d13h

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       3d13h
vksbastion@vks:~/demo-vks/tp$ kubectl delete deployment/postgres -n postgres
deployment.apps "postgres" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl get all -n postgres
NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.101.183.142   <none>        5432/TCP   3d13h
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get pvc -n postgres
NAME           STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   V
OLUMEATTRIBUTESCLASS   AGE
postgres-pvc   Bound    pvc-43f275d5-6063-4751-98d5-3a36b07d0bf0   10Gi       RWO            sc-nas         <
unset>                 3d14h
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres
No resources found in postgres namespace.
```

Créez maintenant un objet de restauration sur place.

```
# tp create bir postgres-app-restore --backup postgres/postgres-backup-
on-demand -n postgres --dry-run>postgres-app-bir.yaml
```

```
# cat postgres-app-bir.yaml
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  annotations:
    protect.trident.netapp.io/max-parallel-restore-jobs: "25"
  name: postgres-app-restore
  namespace: postgres
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/backups/postgres-backup-on-demand_63efc9d1-92d7-45ce-84fe-
846ce7db7b29
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  resourceFilter: {}
  runArchivedExecHooks: true

# kubectl create -f postgres-app-bir.yaml -n postgres
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-bir.yaml -n postgres
backupinplacerestore.protect.trident.netapp.io/postgres-app-restore created
```

Vérifiez que le déploiement de l'application postgres, le service, le pod et les PVC sont restaurés.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-9pv2m   1/1     Running   0          2m1s

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.107.38.167   <none>        5432/TCP   2m1s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           2m1s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       2m1s

NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS MO
DES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-191af706-64ac-4f09-921a-ff9bf6d86a68   10Gi       RWO
      sc-nas         <unset>                 2m6s
```

Restaurez l'application dans un espace de noms différent

Commencez par créer un nouvel espace de noms dans lequel vous souhaitez restaurer l'application, dans cet exemple postgres2. Une sauvegarde horaire créée par la planification est désormais disponible pour postgres-app. Utilisez cette sauvegarde pour restaurer l'application dans le nouvel espace de noms postgres2.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get backup -n postgres
NAME                              APP            RECLAIM POLICY   STATE       ERROR   AGE
hourly-cbd11-20260831104500   postgres-app   Retain           Completed           14m
postgres-backup-on-demand     postgres-app   Retain           Completed           51m
```

```
# tp create backuprestore --appvault ontap-s3-appvault1 --path postgres-
app_314bbaf6-2ce3-4065-b3c1-ab85c7bb7c7c/backups/hourly-cbd11-
20260831124500_2b57bda9-5cb0-4c8f-ae7a-20f94c23c3b9 --namespace-mapping
postgres:postgres2 -n postgres2 --dry-run>postgres-app-postgres2-br.yaml
```

Pour obtenir le chemin d'accès à la sauvegarde, utilisez la commande `kubectl get backups <backup-name> -n postgres -o jsonpath='{.status.appArchivePath}'`.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get backups -n postgres
NAME                          APP            RECLAIM POLICY   STATE       ERROR   AGE
hourly-cbd11-20260831124500   postgres-app   Retain           Completed           13m
postgres-backup-on-demand     postgres-app   Retain           Completed           170m
vksbastion@vks:~/demo-vks/tp$ tp get backups -n postgres -o yaml | grep hourly-cbd11-20260831124500
    name: hourly-cbd11-20260831124500
    appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-ab85c7bb7c7c/backups/hourly-cbd11-20260831124500_2b5
7bda9-5cb0-4c8f-ae7a-20f94c23c3b9
vksbastion@vks:~/demo-vks/tp$
```

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  annotations:
    protect.trident.netapp.io/max-parallel-restore-jobs: "25"
  name: postgres-app-z2woek
  namespace: postgres2
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/backups/hourly-cbd11-20260831124500_2b57bda9-5cb0-4c8f-
ae7a-20f94c23c3b9
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  namespaceMapping:
  - destination: postgres2
    source: postgres
  resourceFilter: {}
  runArchivedExecHooks: true
  skipApplicationCreation: false

# kubectl create -f postgres-app-postgres2-br.yaml -n postgres2
```

```
vksbastion@vks:~/demo-vks/tp$ tp create backuprestore --appvault ontap-s3-appvault1 --path postgres-app_314bb
af6-2ce3-4065-b3c1-ab85c7bb7c7c/backups/hourly-cbd11-20260831124500_2b57bda9-5cb0-4c8f-ae7a-20f94c23c3b9 --na
mespace-mapping postgres:postgres2 -n postgres2 --dry-run>postgres-app-postgres2-br.yaml
vksbastion@vks:~/demo-vks/tp$
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-postgres2-br.yaml -n postgres2
backuprestore.protect.trident.netapp.io/postgres-app-btguyt created
```

Vérifiez que les objets d'application postgres et les PVC sont créés dans le nouvel espace de noms

postgres2.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-858dt   1/1     Running   0          72s

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.109.5.180   <none>        5432/TCP   72s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           72s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       72s

NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MO
DES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-5893851c-c152-439f-a147-c4ae3581cc6f   10Gi       RWO
      sc-nas         <unset>                 78s
vksbastion@vks:~/demo-vks/tp$
```

Protégez l'application à l'aide de snapshots

▼ Créer des instantanés

Créer un instantané à la demande Créez un instantané pour l'application et spécifiez le AppVault où les métadonnées de l'instantané seront stockées. Les données de l'instantané de volume ne sont pas copiées vers le stockage objet — elles restent sur le système de stockage principal.

```
# tp create snapshot postgres-app-snapshot-ondemand --app postgres-app
--appvault ontap-s3-appvault1 -n postgres --dry-run>postgres-app
-snapshot-ondemand.yaml

# cat postgres-app-snapshot-ondemand.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  name: postgres-app-snapshot-ondemand
  namespace: postgres
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  cleanupSnapshot: false
  completionTimeout: 0s
  volumeSnapshotsCreatedTimeout: 0s
  volumeSnapshotsReadyToUseTimeout: 0s

# kubectl create -f postgres-app-snapshot-ondemand.yaml
snapshot.protect.trident.netapp.io/postgres-app-snapshot-ondemand
created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-snapshot-ondemand.yaml
snapshot.protect.trident.netapp.io/postgres-app-snapshot-ondemand created
vksbastion@vks:~/demo-vks/tp$
vksbastion@vks:~/demo-vks/tp$ kubectl get snapshots -n postgres
NAME                              APP            RECLAIM POLICY   STATE       ERROR   AGE
postgres-app-snapshot-ondemand    postgres-app   Delete           Completed           20s
vksbastion@vks:~/demo-vks/tp$
```

Créer une planification pour les instantanés Créez une planification pour les instantanés. Spécifiez la granularité et le nombre d'instantanés à conserver.

```
# tp create schedule snapshot-schedule1 --app postgres-app --appvault
ontap-s3-appvault1 --granularity Hourly --minute 55 --snapshot-retention
1 -n postgres --dry-run>postgres-app-snapshot-schedule1.yaml

# cat postgres-app-snapshot-schedule1.yaml
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: snapshot-schedule1
  namespace: postgres
spec:
  appVaultRef: ontap-s3-appvault1
  applicationRef: postgres-app
  backupRetention: "0"
  dayOfMonth: ""
  dayOfWeek: ""
  enabled: true
  granularity: Hourly
  hour: ""
  minute: "55"
  recurrenceRule: ""
  replicationRetention: "0"
  runImmediately: false
  snapshotRetention: "1"

# kubectl create -f postgres-app-snapshot-schedule1.yaml
schedule.protect.trident.netapp.io/snapshot-schedule1 created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl create -f postgres-app-snapshot-schedule1.yaml -n postgres
schedule.protect.trident.netapp.io/snapshot-schedule1 created
vksbastion@vks:~/demo-vks/tp$ kubectl get schedule -n postgres
NAME                 ENABLED   GRANULARITY   STATE   ERROR   AGE
backup-schedule1     true      Hourly                        3h37m
snapshot-schedule1   true      Hourly                        10s
vksbastion@vks:~/demo-vks/tp$ kubectl get snapshots -n postgres -w
NAME                             APP            RECLAIM POLICY   STATE       ERROR   AGE
postgres-app-snapshot-ondemand   postgres-app   Delete           Completed           32m
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete                               0s
hourly-f1dd9-20260831135500      postgres-app   Delete           Running             0s
hourly-f1dd9-20260831135500      postgres-app   Delete           Running             0s
hourly-f1dd9-20260831135500      postgres-app   Delete           Running             0s
hourly-f1dd9-20260831135500      postgres-app   Delete           Running             0s
^Cvksbastion@vks:~/demo-vks/tp$ kubectl get snapshots -n postgres
NAME                             APP            RECLAIM POLICY   STATE       ERROR   AGE
hourly-f1dd9-20260831135500      postgres-app   Delete           Completed           15s
postgres-app-snapshot-ondemand   postgres-app   Delete           Completed           33m
```

Restaurer à partir d’un instantané

▼ **Restaurer à partir d’un instantané**

Restaurez l’application à partir d’un instantané dans le même espace de noms

Avant de supprimer l’application, vérifiez que l’instantané à partir duquel vous prévoyez de restaurer l’application est dans l’état `Completed`. Un instantané en cours ou ayant échoué ne peut pas être utilisé pour restaurer l’application.

```
kubectl get snapshot -n postgres
```

Après avoir confirmé que l’état du snapshot est `Completed`, supprimez les objets d’application et les PVC pour postgres de l’espace de noms postgres.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get deployment,svc,pvc -n postgres
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           3h14m

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.107.38.167   <none>        5432/TCP   3h14m

NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-191af706-64ac-4f09-921a-ff9bf6d86a68   10Gi       RWO            sc-nas
<unset>                 3h14m
vksbastion@vks:~/demo-vks/tp$ kubectl delete deployment/postgres -n postgres
deployment.apps "postgres" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete service/postgres-service -n postgres
service "postgres-service" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete pvc/postgres-pvc -n postgres
persistentvolumeclaim "postgres-pvc" deleted from postgres namespace
vksbastion@vks:~/demo-vks/tp$ kubectl get deployment,svc,pvc -n postgres
No resources found in postgres namespace.
vksbastion@vks:~/demo-vks/tp$ |
```

Créez un objet de restauration sur place à partir du snapshot.

```
# tp create sir postgres-restore-from-snapshot --snapshot
```

```
postgres/hourly-f1dd9-20260831135500 -n postgres --dry-run > postgres-
sir.yaml

# cat postgres-sir.yaml
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: postgres-restore-from-snapshot
  namespace: postgres
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/snapshots/20260831135500_hourly-f1dd9-
20260831135500_36c2bd2a-9405-4424-88a7-75e6bbc5b315
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  resourceFilter: {}
  runArchivedExecHooks: true

# kubectl create -f postgres-sir.yaml
snapshotinplacerestore.protect.trident.netapp.io/postgres-restore-from-
snapshot created
```

Vérifiez que les objets et les PVC de l'application sont créés dans l'espace de noms postgres.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-wrppb   1/1     Running   0          43s

NAME                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.101.142.54   <none>        5432/TCP   43s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           43s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       43s

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc    Bound    pvc-40f55cd3-ba5c-40b6-94ba-52a4e6767b33   10Gi       RWO            sc-nas
<unset>                 49s
vksbastion@vks:~/demo-vks/tp$
```

Restaurer l'application à partir d'un instantané vers un espace de noms différent

Supprimez l'application dans l'espace de noms postgres2 précédemment restaurée à partir de la sauvegarde.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-858dt   1/1     Running   0          65m

NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.109.5.180   <none>        5432/TCP   65m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           65m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       65m

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc    Bound    pvc-5893851c-c152-439f-a147-c4ae3581cc6f   10Gi       RWO            sc-nas
<unset>                 65m
vksbastion@vks:~/demo-vks/tp$ kubectl delete deployment/postgres -n postgres2
deployment.apps "postgres" deleted from postgres2 namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete service/postgres-service -n postgres2
service "postgres-service" deleted from postgres2 namespace
vksbastion@vks:~/demo-vks/tp$ kubectl delete pvc/postgres-pvc -n postgres2
persistentvolumeclaim "postgres-pvc" deleted from postgres2 namespace
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
No resources found in postgres2 namespace.
vksbastion@vks:~/demo-vks/tp$
```

Créez l'objet de restauration d'instantané à partir de l'instantané et fournissez le mappage d'espace de noms.

```
# tp create sr postgres-sr --snapshot postgres/hourly-f1dd9-
20260831135500 --namespace-mapping postgres:postgres2 -n postgres2 --dry
-run>postgres-sr.yaml

# cat postgres-sr.yaml
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: postgres-sr
  namespace: postgres2
spec:
  appArchivePath: postgres-app_314bbaf6-2ce3-4065-b3c1-
ab85c7bb7c7c/snapshots/20260831135500_hourly-f1dd9-
20260831135500_36c2bd2a-9405-4424-88a7-75e6bbc5b315
  appVaultRef: ontap-s3-appvault1
  cleanUpAdditionalExecHooks: true
  cleanUpArchivedExecHooks: false
  namespaceMapping:
  - destination: postgres2
    source: postgres
  resourceFilter: {}
  runArchivedExecHooks: true
  skipApplicationCreation: false

# kubectl create -f postgres-sr.yaml
snapshotrestore.protect.trident.netapp.io/postgres-sr created
```

```
vksbastion@vks:~/demo-vks/tp$ kubectl get snapshotrestore  -n postgres2
NAME           STATE       ERROR   AGE
postgres-sr    Completed           44s
```

Vérifiez que les objets et les PVC de l'application sont restaurés dans le namespace postgres2.

```
vksbastion@vks:~/demo-vks/tp$ kubectl get all,pvc -n postgres2
NAME                            READY   STATUS    RESTARTS   AGE
pod/postgres-6fcc5545c8-ql8h4   1/1     Running   0          3m29s

NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/postgres-service   ClusterIP   10.107.103.215   <none>        5432/TCP   3m29s

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/postgres   1/1     1            1           3m29s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/postgres-6fcc5545c8   1         1         1       3m29s

NAME                                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS
VOLUMEATTRIBUTESCLASS   AGE
persistentvolumeclaim/postgres-pvc   Bound    pvc-11e41614-4706-4bc7-ab77-da57b3baf754   10Gi       RWO            sc-nas
<unset>                 3m33s
vksbastion@vks:~/demo-vks/tp$
```

Informations sur le copyright

Informations sur les marques commerciales