vSphere Kubernetes Service con storage NetApp

NetApp container solutions

NetApp
September 01, 2026

Sommario

vSphere Kubernetes Service con storage NetApp

Scopri VMware vSphere Kubernetes Service con NetApp ONTAP

VMware vSphere Kubernetes Service (VKS) è un servizio Kubernetes gestito offerto da VMware che ti permette di implementare, gestire e scalare applicazioni containerizzate utilizzando Kubernetes. Abbinato a NetApp ONTAP storage, VKS offre persistenza, prestazioni e gestione dei dati enterprise per i carichi di lavoro cloud-native.

VKS è integrato in VMware Cloud Foundation (VCF) e si conforma allo standard CNCF Kubernetes, garantendo la compatibilità con l'ecosistema Kubernetes più ampio e con i relativi strumenti.

Caratteristiche principali del servizio Kubernetes vSphere

1. **Cluster Kubernetes gestiti:** vSphere Kubernetes Service semplifica la distribuzione e la gestione dei cluster Kubernetes. Automatizza attività come la creazione, l'aggiornamento, il dimensionamento e il monitoraggio dei cluster.
2. **Integrazione con l'infrastruttura VMware:** VKS si integra perfettamente con VMware vSphere, NSX-T e altre soluzioni VMware, consentendo alle organizzazioni di sfruttare la propria infrastruttura VMware esistente per i carichi di lavoro Kubernetes.
3. **Supporto multi-cloud e cloud ibrido:** vSphere Kubernetes Service supporta la distribuzione su cloud privati, cloud pubblici (ad esempio, AWS, Azure, Google Cloud) e ambienti cloud ibridi, offrendo alle organizzazioni flessibilità nella gestione delle proprie applicazioni.
4. **Sicurezza integrata:** VKS include funzionalità di sicurezza come il controllo degli accessi in base al ruolo (RBAC), l'applicazione delle policy e l'integrazione con VMware NSX per la sicurezza della rete.
5. **Supporto per l'ecosistema Kubernetes:** VKS supporta l'intera API Kubernetes upstream, garantendo la portabilità tra ambienti e la compatibilità con l'ecosistema Kubernetes più ampio, inclusi strumenti come Istio per il service mesh, Prometheus per il monitoraggio e altri strumenti certificati CNCF. Le organizzazioni possono applicare competenze e integrazioni Kubernetes standard direttamente ai cluster VKS.
6. **Strumenti a misura di sviluppatore:** VKS supporta i flussi di lavoro standard degli sviluppatori Kubernetes, incluse pipeline CI/CD, GitOps e strumenti come kubectl e Helm, consentendo ai team di sviluppo di creare e distribuire applicazioni containerizzate senza dover apprendere interfacce proprietarie.
7. **Scalabilità e alta disponibilità:** VKS offre funzionalità come l'auto-scaling e la tolleranza ai guasti per garantire che le applicazioni rimangano altamente disponibili e dalle performance elevate.
8. **Osservabilità e monitoraggio:** VKS si integra con gli strumenti di monitoraggio standard compatibili con Kubernetes per fornire informazioni in real-time sulle prestazioni del cluster e delle applicazioni.

Casi d'uso per vSphere Kubernetes Service

1. **Modernizzazione delle applicazioni:** Le organizzazioni possono modernizzare le applicazioni legacy containerizzandole e distribuendole su cluster Kubernetes gestiti da VKS.
2. **DevOps Enablement:** VKS supporta i workflow DevOps fornendo automazione, integrazione CI/CD e strumenti adatti agli sviluppatori.
3. **Implementazioni cloud ibride:** Le aziende possono utilizzare VKS per implementare cluster Kubernetes in ambienti on-premises e cloud, garantendo operazioni coerenti.

4. **Architettura a microservizi:** VKS supporta lo sviluppo di applicazioni basate su microservizi, consentendo ai team di creare e scalare sistemi distribuiti.

Guida introduttiva a Storage e vSphere Kubernetes Service

Installa un cluster vSphere Kubernetes Service

Installa un cluster vSphere Kubernetes Service (VKS) nel tuo ambiente VCF 9.1 e configura i nodi worker con le utility di storage appropriate per i tuoi workload.

Informazioni su questa attività

Le utility NFS vengono installate automaticamente nei nodi worker del cluster che crei. Se vuoi creare nodi worker con utility iSCSI o NVMe installate, devi creare un'immagine OVA personalizzata e usare quell'immagine per i nodi worker. L'immagine personalizzata può essere creata usando Docker e lo `vcf` strumento da riga di comando può poi essere usato per creare un file OVA da questa immagine. Questo file OVA può essere caricato nella Content Library in vSphere. Quando crei il cluster VKS, puoi selezionare questa immagine dalla Content Library per i pool di nodi.

Passaggi

1. Crea il cluster VKS:

   Puoi creare un cluster VKS utilizzando l'immagine predefinita del nodo worker oppure un'immagine personalizzata che crei tu. L'immagine predefinita del nodo worker include le utilità NFS preinstallate, mentre l'immagine personalizzata può includere le utilità iSCSI e NVMe. Le seguenti opzioni sono disponibili:

   - **Solo NAS (impostazione predefinita):** Crea un cluster VKS utilizzando l'immagine predefinita del nodo worker, che include le utilità NFS preinstallate.
   - **SAN abilitato (immagine personalizzata):** Crea un'immagine Docker personalizzata che includa le utilità iSCSI e NVMe, genera un file OVA utilizzando il comando `vcf` da riga di comando, carica l'OVA nella vSphere Content Library e poi crea il cluster VKS selezionando quell'immagine personalizzata per i pool di nodi.

   ▼ Mostra le istruzioni per creare un cluster Kubernetes di solo NAS vSphere con l'immagine predefinita del nodo worker

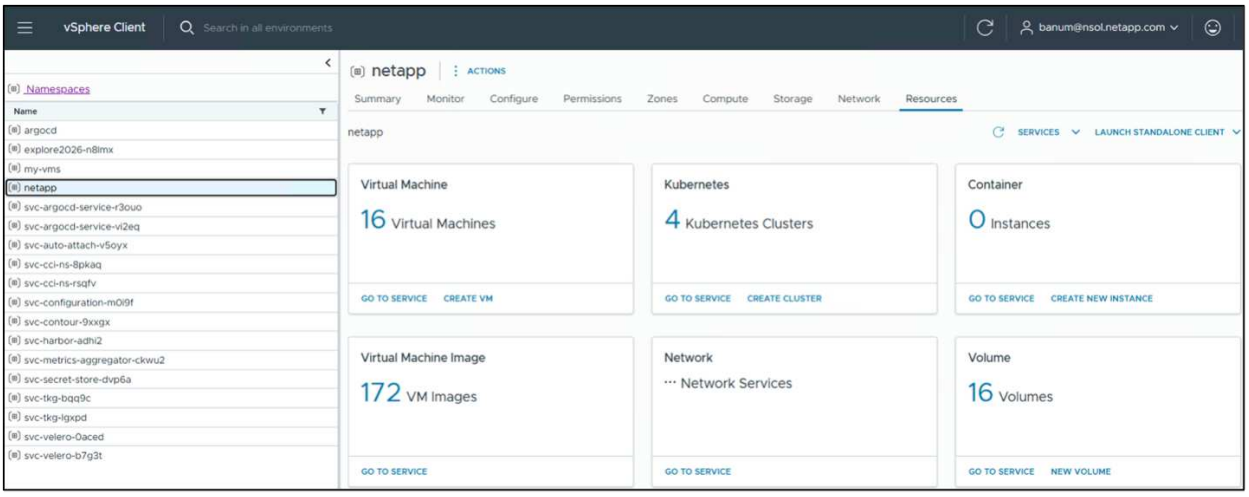
   Crea il cluster VKS utilizzando l'immagine predefinita del nodo worker. Segui le istruzioni per specificare il nome del cluster, la configurazione del pool di nodi e altre impostazioni.

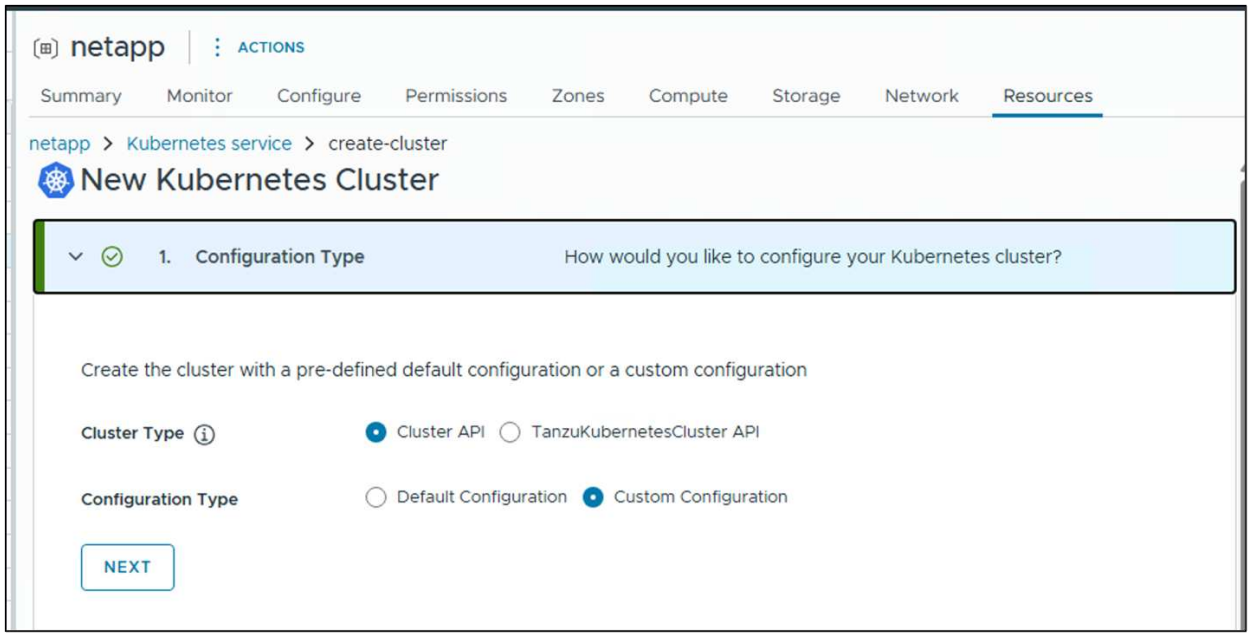
   Dal vSphere Client, vai al namespace in cui vuoi creare il vSphere Kubernetes cluster. Nella scheda **Risorse** di quel namespace, fai clic su **Crea cluster** nella sezione Kubernetes oppure fai clic su **Vai al servizio** e poi su **Crea cluster**.

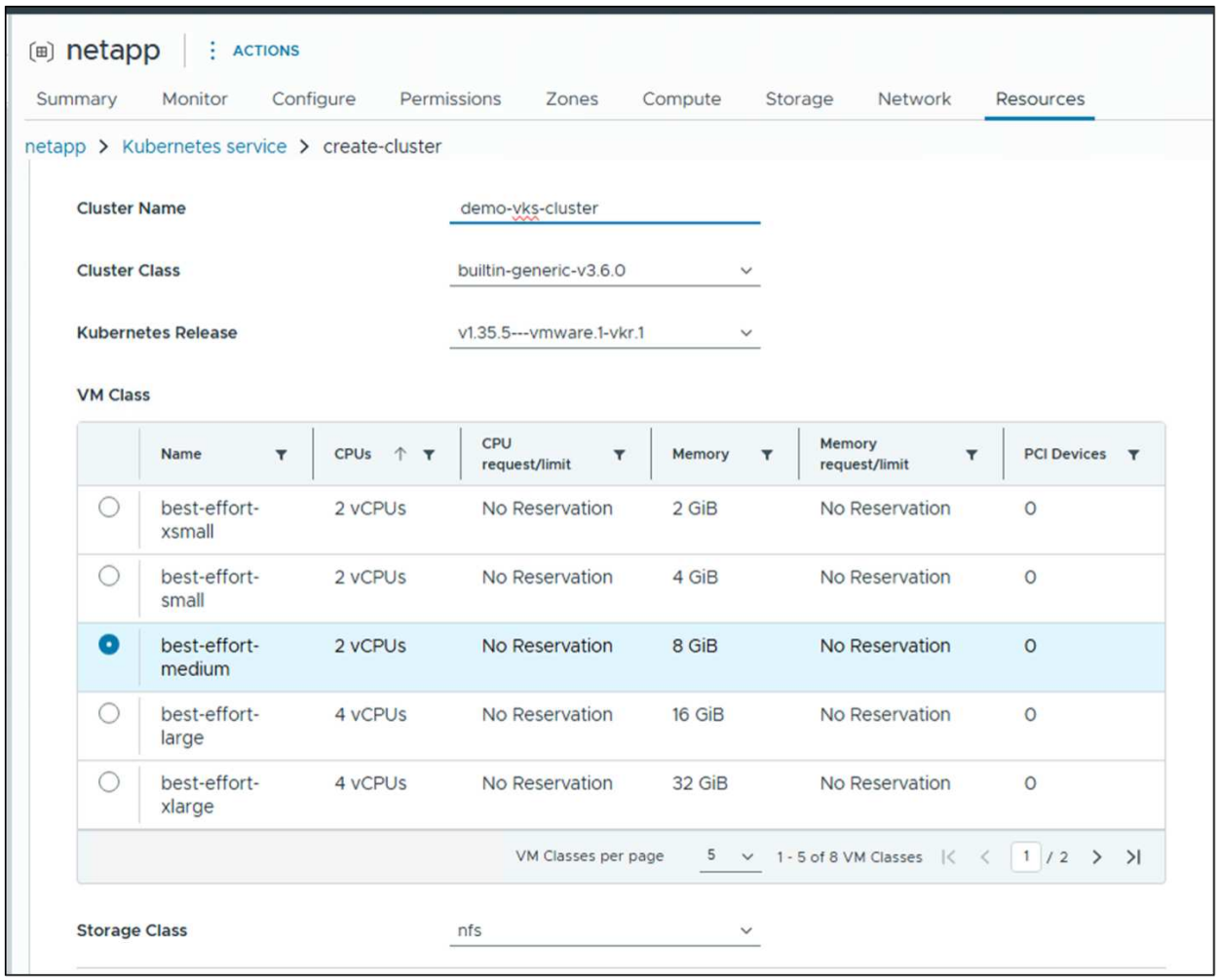
   Compila il modulo con i dettagli del cluster:

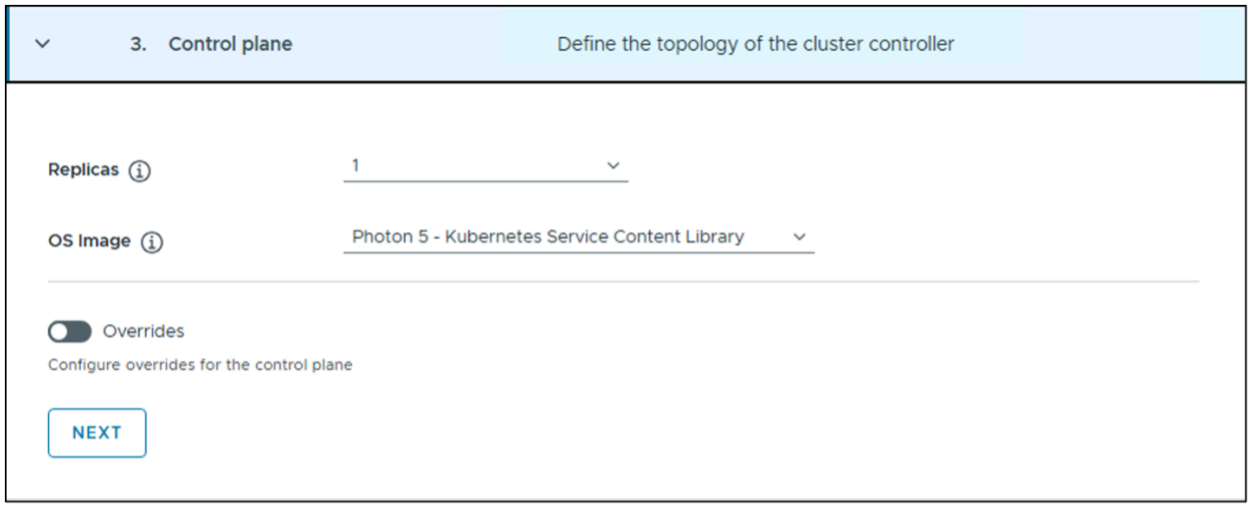
   - Nella sezione 1, **Tipo di configurazione**, seleziona **Personalizzata**.
   - Nella sezione 2, **Impostazioni generali**, inserisci un nome per il cluster e lascia tutte le altre impostazioni ai valori predefiniti.
   - Accetta tutte le impostazioni predefinite per la sezione 3.
   - Nella sezione 4, **Node Pools**, fai clic sui puntini di sospensione (…) accanto al node pool e poi su **Edit**. Aumenta le repliche a 3 e seleziona la versione più recente di Ubuntu per l'immagine del sistema operativo. Fai clic su **Next**, poi su **Review and Confirm**.

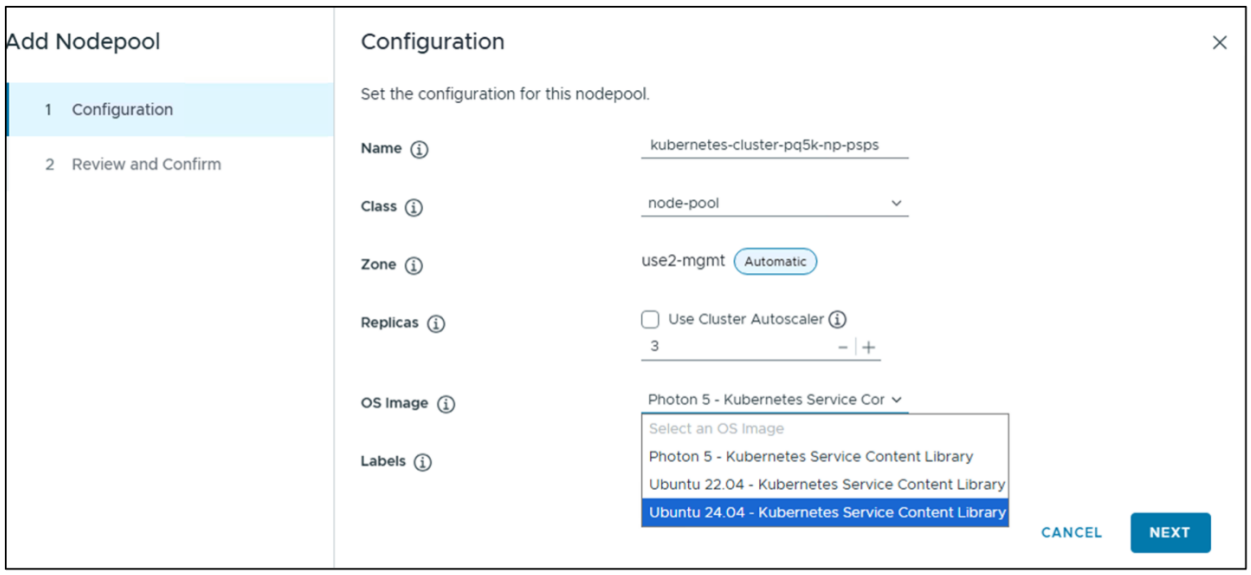
Dopo aver esaminato i dettagli del cluster, fai clic su **Fine**. Il cluster Kubernetes vSphere verrà creato in pochi minuti.

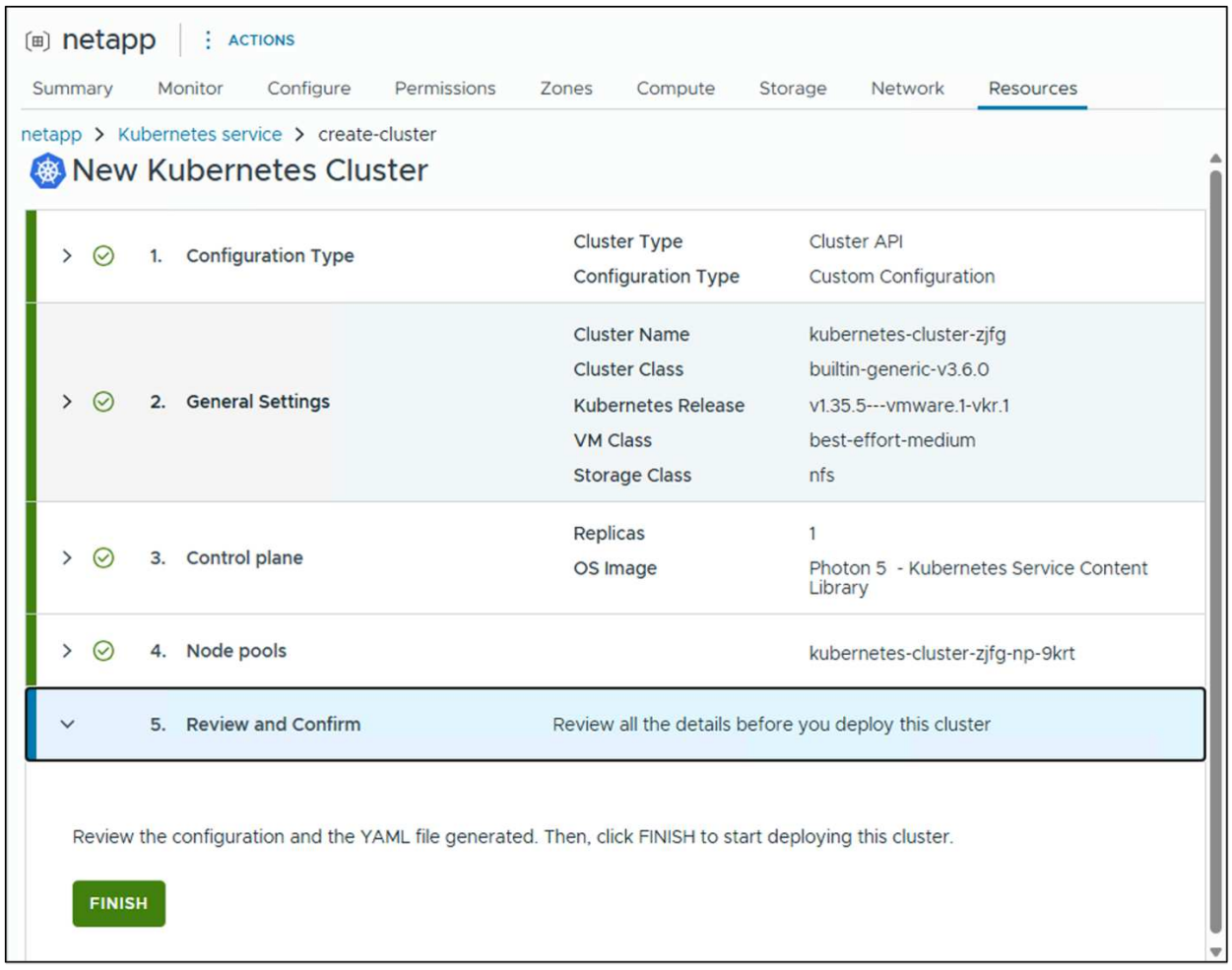


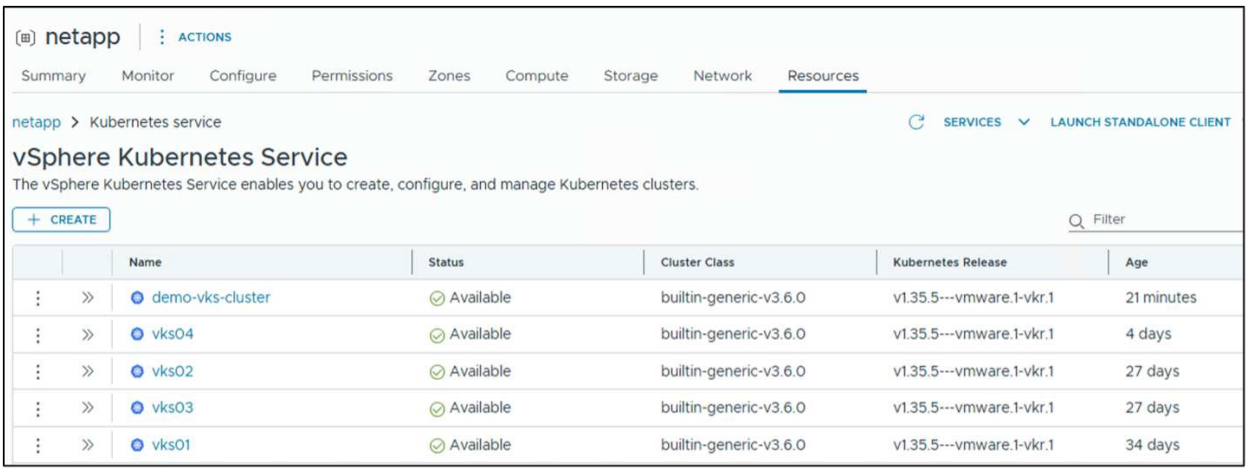



netapp
ACTIONS
Summary
Monitor
Configure
Permissions
Zones
Compute
Storage
Network
Resources
netapp > Kubernetes service > create-cluster
Cluster Name
demo-vks-cluster
Cluster Class
builtin-generic-v3.6.0
Kubernetes Release
v1.35.5---vmware.1-vkr.1
VM Class
Name
CPUs
CPU request/limit
Memory
Memory request/limit
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



▼ **Mostra le istruzioni per creare un'immagine OVA personalizzata per i nodi worker abilitati SAN**

Crea un'immagine Docker con le utilità necessarie installate. L'esempio seguente mostra come creare un'immagine Docker basata su Ubuntu 24.04 con le utilità iSCSI e Multipathing installate.

```
#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"]
```

Crea l'immagine Docker utilizzando:

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

Se il registry Docker non è disponibile, esegui questo comando per avviare un registry locale e caricare l'immagine su di esso:

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

Tagga l'immagine e pubblicala.

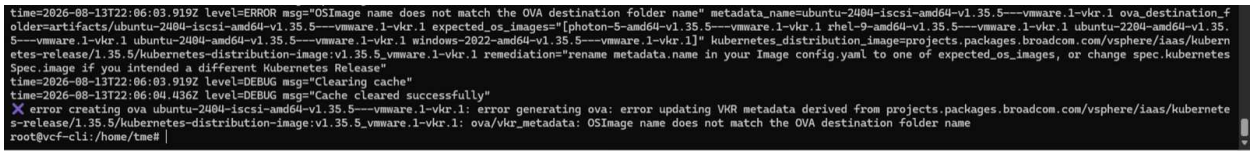
```
docker tag ubuntu-2404:san localhost:5000/ubuntu-2404:san
docker push localhost:5000/ubuntu-2404:san
```

Crea il file di configurazione dell'immagine (salvalo come `ubuntu2404vkr135-san.yml`) e fai riferimento all'immagine:

```
#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
```

Il nome dei metadati deve essere uno di quelli presenti nell'elenco pre-approvato. Se non lo è, riceverai un errore come mostrato qui sotto quando provi a creare il file OVA. L'elenco pre-approvato si trova nella guida della CLI di VCF per il comando `kr bake`.



Crea il file OVA utilizzando la CLI di VCF:

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

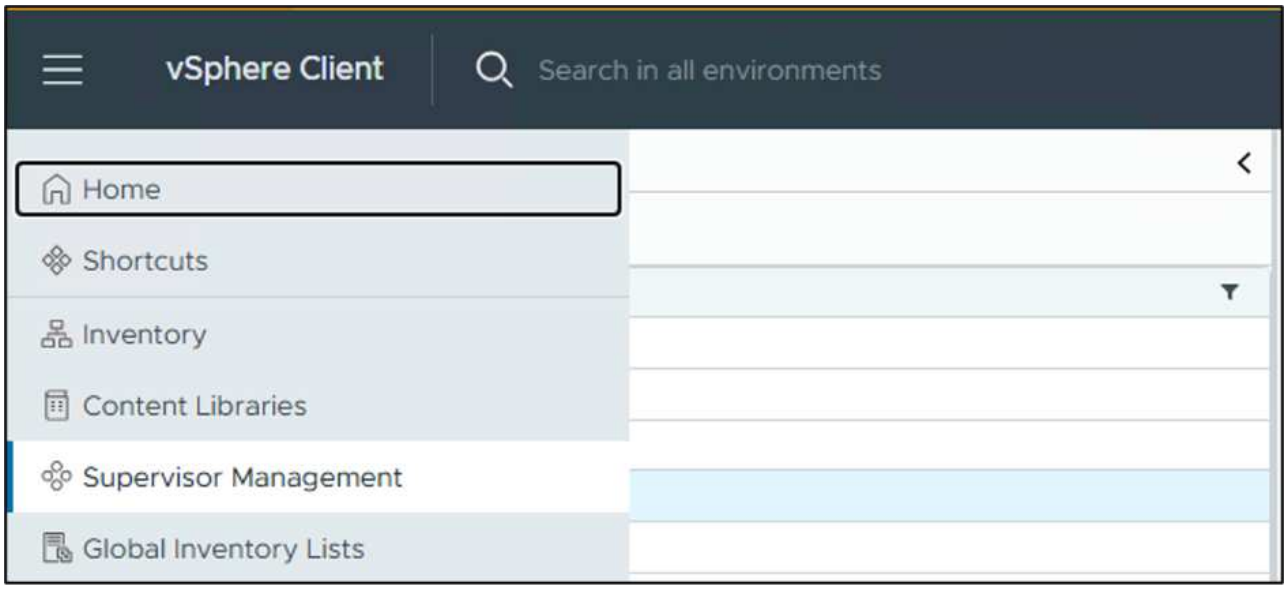
Trasferisci il file OVA tramite SCP a una macchina dalla quale puoi caricarlo nella vSphere Content Library.

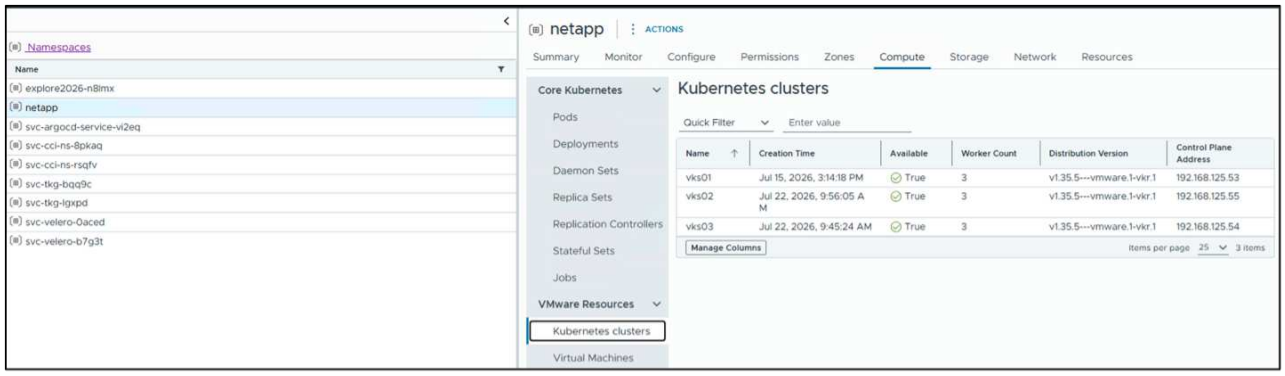
2. Dopo l'installazione del cluster, individua il cluster in vCenter e scarica il file kubeconfig.

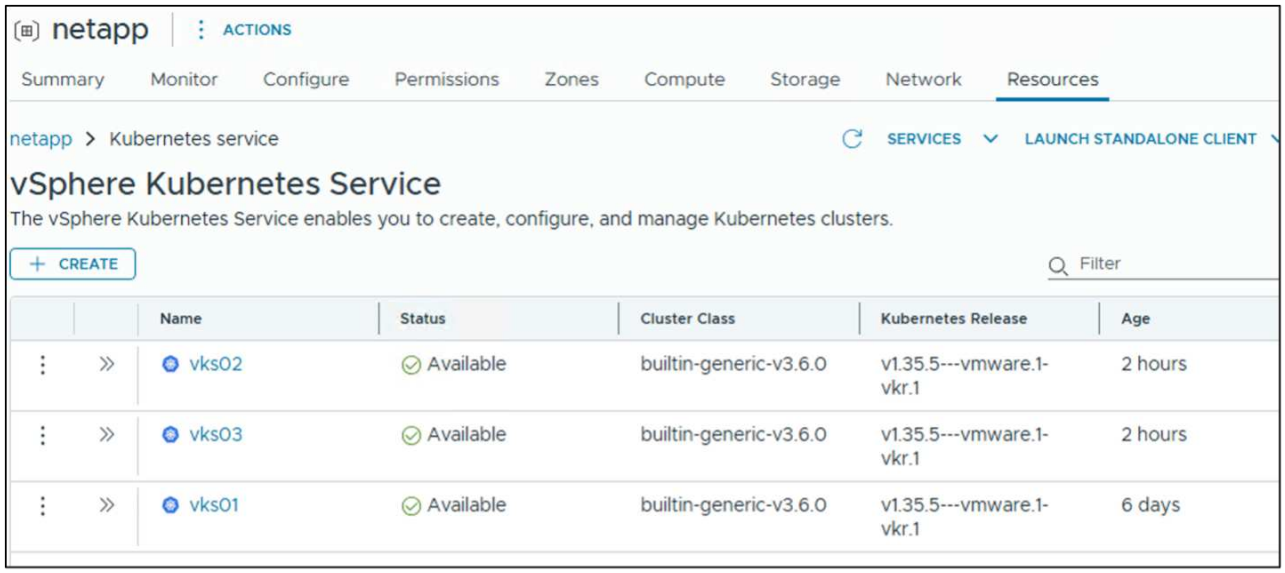
    a. Fai clic su **Gestione supervisori** dal menu principale e seleziona lo spazio dei nomi in cui è stato creato il cluster.

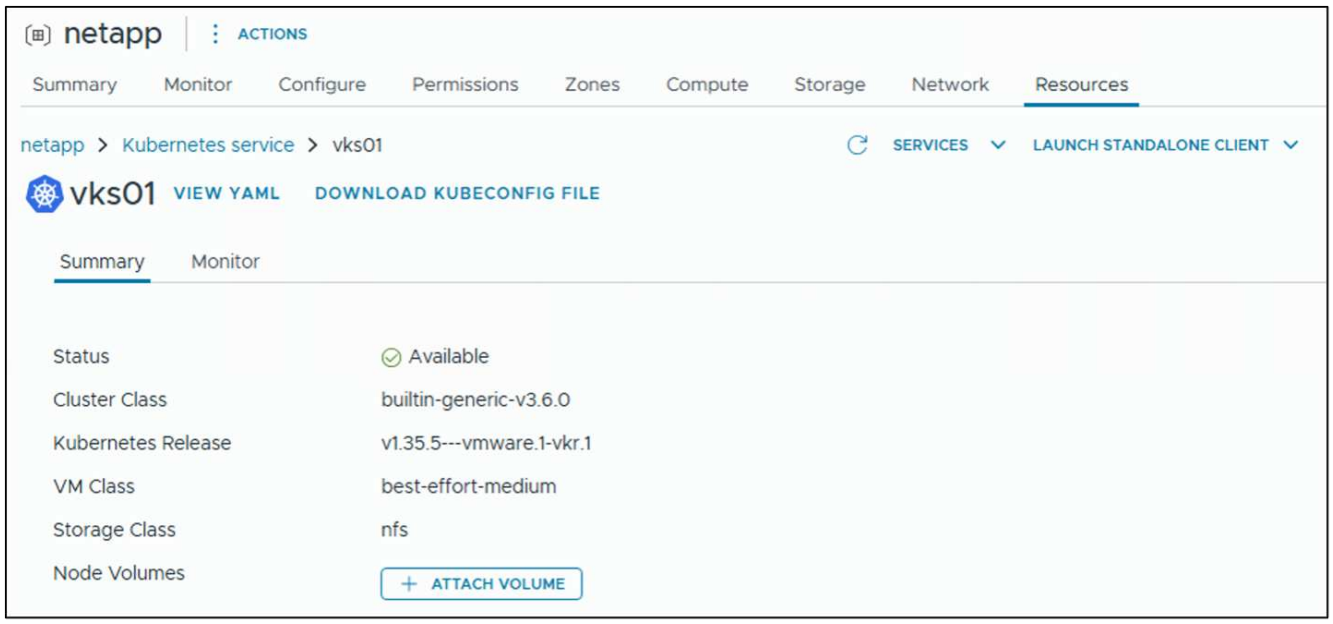
b. Seleziona la scheda **Compute** e poi **Kubernetes Clusters**. Vengono visualizzati tutti i cluster presenti nello spazio dei nomi.

c. Vai alla scheda **Risorse**, seleziona il cluster Kubernetes di cui hai bisogno e scarica il relativo file kubeconfig.









3. Da un host bastion, connettiti al cluster utilizzando il file kubeconfig per gestirlo tramite la CLI.

```
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
```

Video dimostrativo

I seguenti video mostrano due modi per creare un vSphere Kubernetes cluster.

Crea un vSphere Kubernetes cluster su un Supervisor Cluster

Crea un vSphere Kubernetes cluster con l'automazione VCF

Scopri lo storage NetApp per vSphere Kubernetes Service

Utilizzare NetApp come storage per vSphere Kubernetes Service (VKS) è una combinazione potente che sfrutta le solide soluzioni di storage di NetApp per migliorare le prestazioni, la scalabilità e l'affidabilità dei carichi di lavoro Kubernetes. NetApp Trident, un provisioner di storage dinamico open source per Kubernetes, viene utilizzato per integrare lo storage con i carichi di lavoro su vSphere Kubernetes Service.

Principali vantaggi dell'utilizzo dello storage NetApp con VKS

1. **Provisioning dinamico dello storage:** NetApp Trident consente il provisioning dinamico dei volumi di storage, permettendo ai pod di Kubernetes di richiedere storage al volo in base alle proprie esigenze.
2. **Archiviazione persistente:** Le soluzioni NetApp offrono funzionalità di storage persistente, garantendo la durabilità e la disponibilità dei dati anche in caso di riavvio o guasto dei pod.
3. **Prestazioni elevate:** Le soluzioni di storage di NetApp, come ONTAP, offrono storage dalle performance elevate con bassa latenza, ideale per applicazioni ad alta intensità di I/O in esecuzione su Kubernetes.

4. **Scalabilità:** I sistemi di storage NetApp possono scalare per soddisfare le esigenze dei carichi di lavoro Kubernetes in crescita, supportando implementazioni su larga scala.
5. **Gestione dei dati:** NetApp fornisce funzionalità avanzate di gestione dei dati, tra cui snapshot, clonazione e replica dei dati, che possono essere utili per il backup, il disaster recovery e i flussi di lavoro di sviluppo.
6. **Supporto multi-cloud e cloud ibrido:** Le soluzioni di storage di NetApp possono essere implementate in ambienti on-premises, cloud pubblici e cloud ibridi, offrendo flessibilità e coerenza nella gestione dello storage.

NetApp ONTAP panoramica sull'integrazione

NetApp ONTAP offre storage enterprise con funzionalità come deduplica dei dati, compressione ed encryption. Usi NetApp Trident per integrare lo storage NetApp con Kubernetes. NetApp Trident supporta diverse piattaforme di storage NetApp, tra cui ONTAP on-premises, FSx for NetApp ONTAP in AWS, Azure NetApp Files in Azure e Google Cloud NetApp Volumes in Google Cloud.

Usi Trident Protect per gestire la protezione dei dati e il disaster recovery dei tuoi workload Kubernetes. Trident Protect ti permette di creare Snapshot, cloni e replicare i dati su diversi backend di storage, assicurando che i tuoi dati siano al sicuro e recuperabili in caso di guasti.

Caratteristiche principali di Trident

Conformità CSI Garantisce la compatibilità con le più recenti funzionalità e standard di storage di Kubernetes. **Configurazione dichiarativa** Consente agli utenti di definire i requisiti di storage nei file YAML, che vengono poi gestiti automaticamente da Trident. **Driver** Trident supporta i driver per i protocolli NFS, CIFS/SMB, iSCSI, FC e NVMe/TCP supportati da ONTAP. **Supporto ibrido e multi-cloud** Trident supporta i driver per lo storage ONTAP on-premises e per i servizi di storage gestiti negli hyperscaler, ovvero FSx per NetApp ONTAP in AWS, Azure NetApp Files in Azure e Google Cloud NetApp Volumes in Google Cloud. **Gestione avanzata dei dati** Sfrutta le funzionalità di snapshot e clonazione di ONTAP per una protezione efficiente dei dati e un provisioning rapido degli ambienti di test e sviluppo.

Per informazioni complete su Trident, consulta la "Documentazione Trident".

Trident Protect

Trident Protect viene installato separatamente da Trident. Prima della distribuzione, verifica che la piattaforma Kubernetes, il backend di storage, i componenti Snapshot e lo storage a oggetti soddisfino i "Requisiti di Trident Protect".

Caratteristiche principali di Trident Protect

Backup automatici Trident Protect consente backup automatici e basati su policy delle applicazioni Kubernetes, garantendo che vengano sottoposte a backup regolarmente senza intervento manuale.

Snapshot Management Sfrutta le funzionalità di snapshot di ONTAP per creare copie point-in-time frequenti delle applicazioni container e delle VM, fornendo un meccanismo affidabile per il ripristino.

Crittografia del repository di backup Trident Protect supporta la crittografia basata su password per i repository di backup di Restic e Kopia. Configura separatamente la crittografia dei volumi persistenti e la crittografia in transito nei livelli di storage e di rete.

Conformità e audit Fornisce strumenti e funzionalità che aiutano le organizzazioni a soddisfare i requisiti di conformità e a condurre audit periodici delle proprie pratiche di protezione dei dati.

Ripristino di emergenza Si integra con le funzionalità di replica di ONTAP per fornire soluzioni di ripristino di

emergenza robuste, garantendo la continuità operativa anche in caso di eventi catastrofici.

Per informazioni complete su Trident Protect, consulta il "Documentazione Trident Protect".

NetApp ONTAP modelli hardware e protocolli supportati

NetApp ONTAP software è eseguibile su una varietà di modelli hardware, ciascuno progettato per soddisfare diverse esigenze di prestazioni e capacità. Trident estende le sue robuste funzionalità per supportare diversi sistemi NetApp ONTAP, inclusi All Flash FAS (AFF), All SAN Array (ASA) e ASA R2. Questo supporto garantisce che le organizzazioni possano sfruttare appieno il potenziale delle loro soluzioni di storage dalle performance elevate in ambienti containerizzati.

All Flash FAS (AFF) Systems I sistemi AFF offrono prestazioni eccezionali e bassa latenza, rendendoli ideali per applicazioni ad alta richiesta.

All SAN Array (ASA) Systems I sistemi ASA sono progettati per ambienti SAN dedicati, offrendo prestazioni e affidabilità elevate.

ASA R2 Systems I sistemi ASA R2 offrono funzionalità e capacità avanzate per gli ambienti SAN.

AFX NetApp AFX è un'architettura di storage all-flash disaggregata costruita ad hoc, progettata per i carichi di lavoro AI enterprise. Offre NAS dalle performance elevate, consentendo la scalabilità indipendente di calcolo e capacità. I sistemi storage NetApp AFX differiscono dagli altri sistemi basati su ONTAP (ASA, AFF e FAS) nell'implementazione del loro livello di storage. AFX utilizza un file system distribuito che consente prestazioni elevate e scalabilità, mentre gli altri sistemi basati su ONTAP utilizzano un'architettura di storage tradizionale.

1. Protocolli supportati per AFF: NFS, CIFS/SMB, iSCSI, FC, NVMe/TCP
2. Protocolli supportati per ASA: iSCSI, FC, NVMe/TCP
3. Protocolli supportati per ASA R2: iSCSI, FC, NVMe/TCP
4. Protocolli supportati per AFX: a partire da Trident 25.10, è supportato solo il protocollo NFS; il protocollo SMB non è supportato.

Driver Trident per lo storage ONTAP

Trident include i seguenti driver CSI, ognuno dei quali può effettuare il provisioning di un volume nello storage di backend.

Per i protocolli NAS sui modelli hardware supportati

1. ontap-nas: Un PVC corrisponde a un PV, che è un FlexVol volume dedicato.
2. ontap-nas-economy: Ogni PV di cui viene effettuato il provisioning è un qtree, con un numero configurabile di qtree per FlexVol volume (il valore predefinito è 200, configurabile tra 50 e 300).

   Un PVC corrisponde a un PV, che è un qtree su un volume FlexVol condiviso.

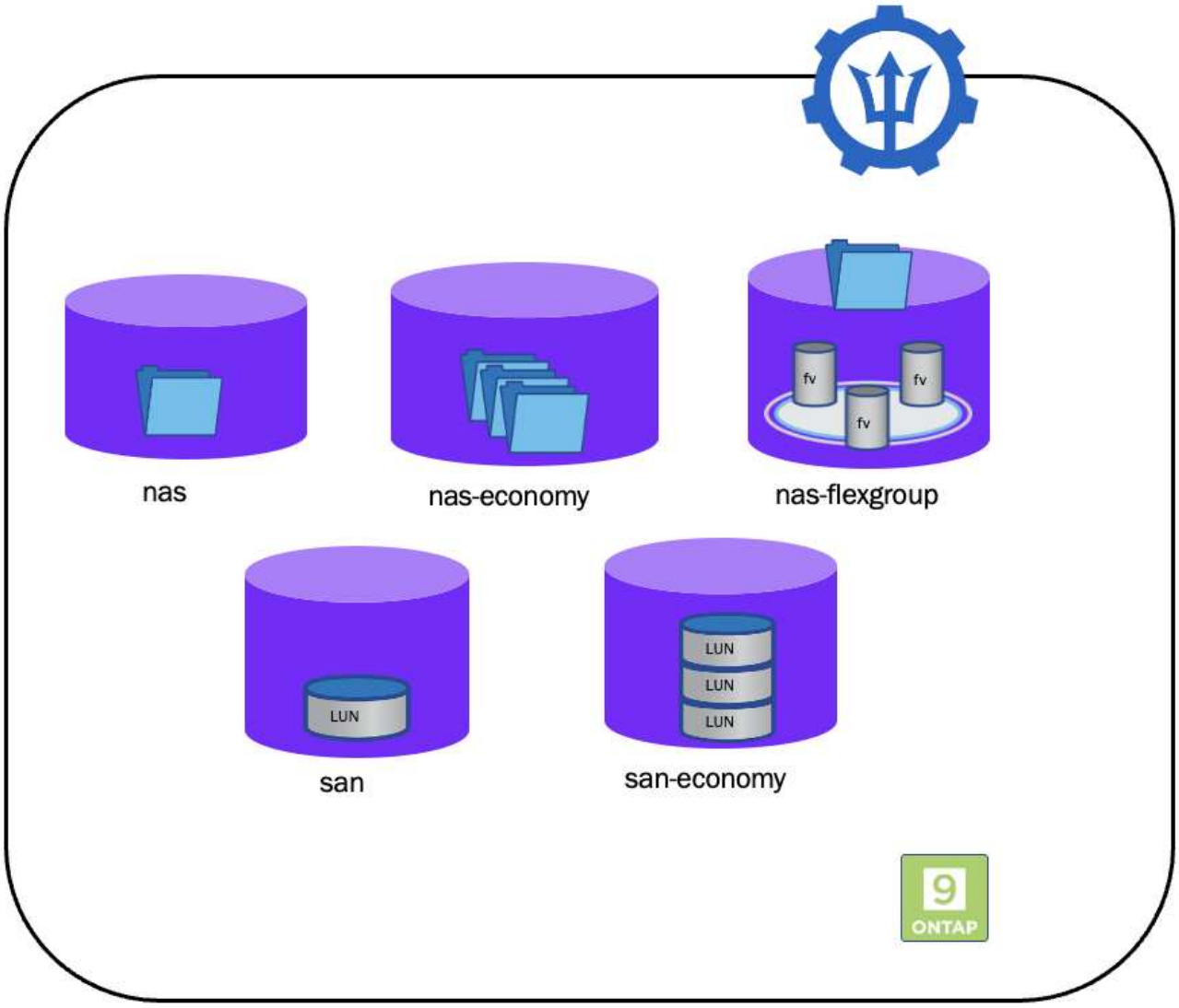
   > Puoi posizionare tutti i dischi delle VM come qtree in un singolo volume. Tuttavia, tieni presente che le funzionalità di Snapshot, Cloni e Replica non sono supportate da Trident. Quando queste funzionalità sono gestite dall'amministratore dello storage, non sono granulari a livello di PV.

1. ontap-nas-flexgroup: Ogni PV di cui è stato effettuato il provisioning è un ONTAP FlexGroup, e vengono utilizzati tutti gli aggregati assegnati a un SVM.

Per i protocolli SAN sui modelli hardware supportati

1. ontap-san: Un PVC corrisponde a un PV, che è un LUN su un volume FlexVol dedicato.
2. ontap-san-economy: Un PVC corrisponde a un PV, che è un LUN su un FlexVol volume condiviso. Puoi avere più LUN nel FlexVol volume condiviso (il valore predefinito è 100, configurabile tra 50 e 200 LUN/PV per FlexVol volume).

Puoi posizionare tutti i dischi delle VM come LUN in un singolo volume. Tuttavia, questo driver supporta Snapshot e Cloni ma non supporta la Replica. Quando la Replica è gestita dall'amministratore dello storage, non è granulare a livello di PV.



Per l'elenco completo delle funzionalità esposte tramite i driver Trident e di quelle che devono essere applicate dall'amministratore dello storage, consulta la sezione "Driver backend ONTAP" nella documentazione di Trident.

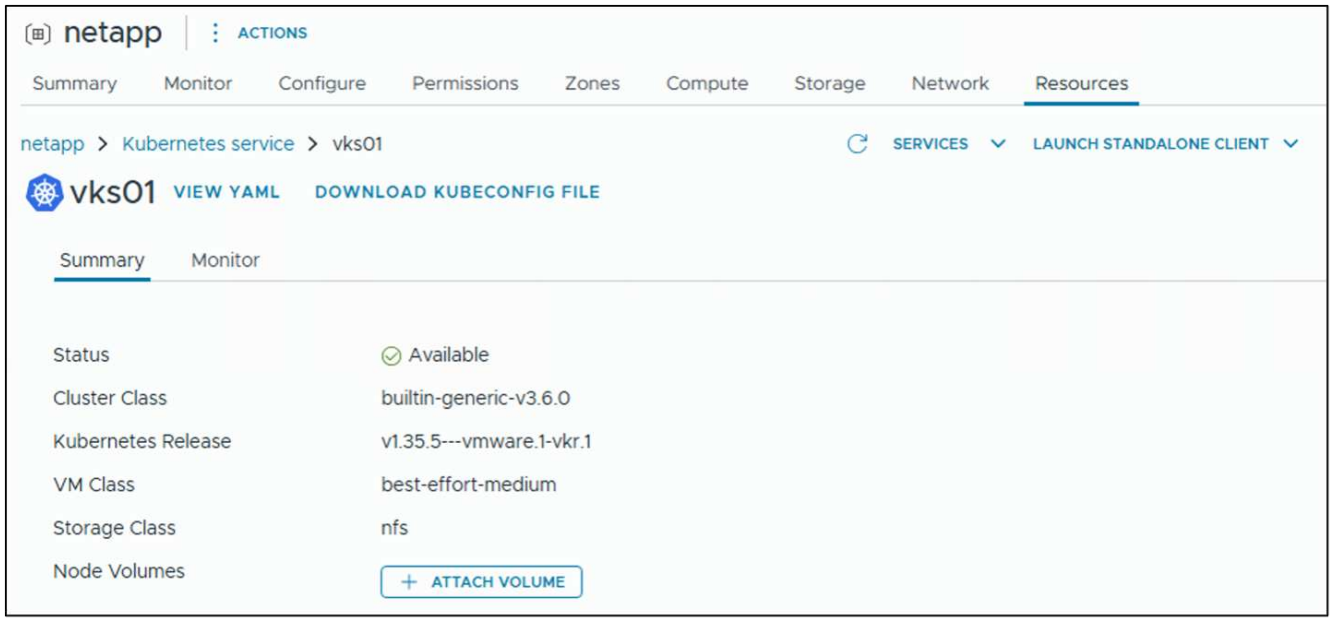
Installa NetApp Trident in un cluster VKS

Installa NetApp Trident come provisioner di storage dinamico nel tuo cluster vSphere Kubernetes Service per integrare lo storage NetApp ONTAP con i tuoi carichi di lavoro

Kubernetes.

Prima di iniziare

- Assicurati che il tuo ONTAP cluster sia configurato e connesso al tuo ambiente VMware Kubernetes.
- Assicurati che il tuo cluster VKS abbia gli strumenti iSCSI o NVMe installati se i tuoi carichi di lavoro utilizzeranno questi protocolli per lo storage.
- Assicurati di avere `kubectl` installato su una macchina dalla quale puoi connetterti al cluster VKS utilizzando il file kubeconfig.



Passaggi

1. Installa l'operatore Trident utilizzando un chart Helm seguendo le istruzioni nella documentazione di Trident:

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

   Prima di eseguire l'installazione di Helm, crea e assegna l'etichetta al `trident` namespace. L'etichetta `enforce=privileged` è necessaria perché Trident esegue carichi di lavoro privilegiati. Senza di essa, il controller di ammissione alla sicurezza dei pod di Kubernetes bloccherà l'avvio dei pod di Trident.

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

   ▼ Mostra esempio

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

   Per abilitare la registrazione di debug, aggiungi `--set tridentDebug=true`:

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

Puoi anche installare Trident manualmente usando il Trident Operator. Consulta la documentazione di Trident per le procedure: "Distribuisci manualmente l'operatore Trident"

Assicurati di aver assegnato al namespace `trident` le etichette di sicurezza del pod richieste prima di installare Trident manualmente.

2. Verificare che tutti i pod di Trident siano in esecuzione nel cluster.

▼ **Mostra esempio**

```
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$
```

Se i pod di Trident non si avviano a causa di un errore di `PodSecurity`ammissione, potrebbe essere che le etichette di sicurezza dei pod non siano state applicate allo `trident`namespace prima dell'installazione. Usa `--overwrite` per riapplicarle, poi verifica che i pod si avviino.

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

Prepara i file di backend dello storage e di configurazione StorageClass per il tuo ambiente

1. Configura un backend Trident per ONTAP definendo i dettagli di connessione e i parametri di storage necessari.
2. Definisci classi di storage in Kubernetes che mappano i backend di storage NetApp. Queste classi specificano parametri come le caratteristiche prestazionali e le politiche di replica.

   Definisci i backend e le classi di storage Trident per i seguenti protocolli, in base alle esigenze dei tuoi carichi di lavoro:

   - NAS (driver ontap-nas)
   - NAS Economy (driver ontap-nas-economy)
   - SAN (driver ontap-san) per iSCSI, FC o protocolli NVMe
   - SAN Economy (driver ontap-san-economy) per iSCSI

NAS

Crea un TridentBackendConfig e un StorageClass per ONTAP NAS per abilitare il provisioning di storage persistente basato su NFS. La configurazione del backend include le credenziali memorizzate in un Secret di Kubernetes e fa riferimento al tuo ONTAP SVM e al LIF di gestione.

Esempio di segreto del backend e file di configurazione con metodo di autenticazione tramite nome utente e password (salva come `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
```

Esempio di segreto del backend e file di configurazione che utilizza l'autenticazione tramite certificato client (salva come `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 (salva come `sc-nas.yaml`):

`mountOptions` è un parametro opzionale che specifica opzioni di montaggio aggiuntive per i volumi NFS. L'opzione `nconnect` consente più connessioni TCP parallele per un singolo mount NFS, il che può migliorare le prestazioni per carichi di lavoro ad alto throughput. Il valore predefinito è 1, ma può essere aumentato fino al massimo consentito dal sistema ONTAP.

ONTAP supporta fino a 16 connessioni per un singolo mount NFS su un client compatibile con nconnect. Trident passa le opzioni di mount direttamente ai nodi worker di Kubernetes, quindi scegli un valore supportato sia dal client NFS del nodo worker che da ONTAP; l'esempio seguente utilizza quattro connessioni.

```
# 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
```

Crea un TridentBackendConfig e una StorageClass per il driver ONTAP NAS Economy per abilitare il provisioning dello storage persistente basato su NFS utilizzando i qtree. La configurazione del backend include le credenziali memorizzate in un Secret di Kubernetes e fa riferimento al tuo ONTAP SVM e al LIF di gestione.

Esempio di segreto del backend e file di configurazione con metodo di autenticazione tramite nome utente e password (salva come `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 (salva come `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
```

Crea un TridentBackendConfig e una StorageClass per ONTAP SAN con iSCSI per abilitare il provisioning dello storage a blocchi basato su iSCSI. La configurazione del backend utilizza il driver ontap-san e include le credenziali memorizzate in un Secret di Kubernetes. Il parametro `sanType` predefinito è il protocollo iSCSI se omesso.

Segreto del backend e file di configurazione del backend che utilizzano l'autenticazione tramite nome utente e password (salva come `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
```

Segreto del backend e file di configurazione del backend che utilizzano il metodo di autenticazione tramite certificato client (salva come `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 (salva come `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
```

Crea un TridentBackendConfig e un StorageClass per ONTAP SAN con NVMe per abilitare il provisioning dello storage a blocchi basato su NVMe. La configurazione del backend utilizza il driver ontap-san e include le credenziali memorizzate in un Secret di Kubernetes. Il `sanType` parametro deve essere impostato su `nvme`.

Segreto del backend e file di configurazione del backend che utilizzano l'autenticazione tramite nome utente e password (salva come `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
```

Segreto del backend e file di configurazione del backend che utilizzano il metodo di autenticazione tramite certificato client (salva come `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 (salva come `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
```

Consulta la documentazione di Trident per ulteriori esempi di file YAML e informazioni sulle modalità di accesso e sulle modalità di volume supportate dai driver SAN e NAS.

- "Esempi che utilizzano driver NAS"
- "Esempi che utilizzano driver SAN"

Crea un file di configurazione VolumeSnapshotClass

Crea una definizione VolumeSnapshotClass. Questa configurazione abilita le operazioni basate su snapshot per i volumi persistenti.

VolumeSnapshotClass definition (salva come `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
```

Applica i file di configurazione al tuo cluster

Applica i file di configurazione che hai creato nei passaggi precedenti al cluster vSphere Kubernetes Service usando `kubectl`. Questo crea i segreti necessari, le configurazioni di backend, le classi di storage e la classe snapshot nel tuo cluster.

Passaggi

1. Applica i file TridentBackendConfig e StorageClass per ciascun protocollo configurato.

```
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. Verificare che le risorse siano state create correttamente.

   Controlla gli oggetti TridentBackendConfig:

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

   Controlla gli oggetti StorageClass:

```
kubectl get storageclass
```

   Verifica VolumeSnapshotClass:

```
kubectl get volumesnapshotclass
```

Imposta le classi di storage e snapshot predefinite di Trident

Imposta Trident StorageClass e VolumeSnapshotClass come predefiniti nel cluster vSphere.

Passaggi

1. Imposta il StorageClass predefinito di Trident.

   Imposta una StorageClass supportata da Trident come predefinita del cluster, in modo che le PersistentVolumeClaims la utilizzino automaticamente quando non viene specificata alcuna classe di storage.

   Assicurati che solo uno StorageClass sia impostato come predefinito. Se un altro StorageClass è già impostato come predefinito, imposta la sua annotazione su `false`.

   Dalla CLI:

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

2. Imposta la VolumeSnapshotClass predefinita di Trident.

   Imposta un VolumeSnapshotClass supportato da Trident come predefinito del cluster per abilitare le operazioni basate su snapshot per i volumi persistenti. Questo garantisce che le VolumeSnapshots utilizzino automaticamente il driver CSI Trident quando non viene specificata alcuna classe di snapshot.

   Assicurati che solo un VolumeSnapshotClass sia impostato come predefinito. Se un altro VolumeSnapshotClass è già impostato come predefinito, imposta la sua annotazione su `false`.

   Dalla CLI:

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

Utilizza Kubernetes Persistent Volume Claims per richiedere storage per i carichi di lavoro

Trident esegue automaticamente il provisioning dei volumi necessari dallo storage NetApp di backend. Consulta "Distribuisci i carichi di lavoro con storage persistente" per esempi di distribuzione di un carico di lavoro con PVC in un cluster VKS.

Distribuisci un'applicazione container con NetApp storage persistente

Distribuisci un'applicazione containerizzata nel tuo cluster vSphere Kubernetes Service utilizzando lo storage persistente NetApp ONTAP. Gli esempi seguenti mostrano come distribuire un database PostgreSQL utilizzando lo storage NAS (ontap-nas) o SAN iSCSI

(ontap-san).

La seguente configurazione YAML imposta POSTGRES_USER / POSTGRES_PASSWORD direttamente nel manifest. In produzione, ti consiglio di usare i Secret di Kubernetes per gestire informazioni sensibili come le credenziali del database.

Distribuisci PostgreSQL con storage NAS (driver ontap-nas)

Puoi distribuire un'applicazione container PostgreSQL supportata da un volume persistente NFS fornito dal driver ontap-nas. Il seguente esempio mostra come creare un PersistentVolumeClaim (PVC) e una Deployment per 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
```

Distribuisci PostgreSQL con storage SAN (driver iSCSI ontap-san)

Usa il seguente YAML per distribuire un'applicazione container PostgreSQL supportata da un volume persistente iSCSI fornito dal driver ontap-san. La `Recreate` strategia di deployment garantisce che il pod venga terminato completamente prima che ne inizi uno nuovo, requisito necessario per i volumi iSCSI ReadWriteOnce e che previene la corruzione dei dati durante i riavvii del pod. Questa configurazione supporta la persistenza dei dati durante l'eliminazione e la ricreazione del pod ed è compatibile con gli snapshot dei volumi Trident e con le operazioni di backup e ripristino.

```
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
```

Convalida la distribuzione del database

Dopo aver distribuito l'applicazione, verifica che i pod siano in esecuzione e che il database sia operativo. Usa i seguenti comandi per controllare lo stato dei pod, dei PVC e dei servizi. Assicurati che i pod siano in esecuzione e che i PVC siano associati ai volumi appropriati.

```
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:~$
```

A seconda del protocollo utilizzato, lo storage di backend è un volume, qtree o LUN. Puoi verificare lo storage fornito nel sistema ONTAP descrivendo il PersistentVolume (PV) per recuperare il nome del volume interno e poi usarlo nei comandi CLI di ONTAP per ottenere i dettagli dello storage. Usa il seguente comando per descrivere il PV e recuperare il nome del volume interno:

```
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>
```

Figura 1. Mostra esempio

```
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>
```

Copia il nome del volume interno dall'output ed esegui il seguente comando per ottenere i dettagli di storage dalla CLI di ONTAP.

Per i volumi 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
```

Per i LUN SAN, esegui il seguente comando per ottenere i dettagli del LUN dalla CLI di 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::>
```

Se hai utilizzato `nconnect` mount options nella classe di storage NAS o NAS Economy, puoi verificare il numero di connessioni TCP attive dal nodo worker al server di storage.

Innanzitutto, elenca le configurazioni del backend Trident per trovare il nome del backend, quindi descrivilo per recuperare il nome del vserver e il LIF dei dati:

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

Accedi quindi al cluster ONTAP ed esegui il seguente comando, sostituendo la data LIF con quella riportata nell'output sopra:

```
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.
```

Figura 2. Mostra esempio

Una volta che i pod sono in stato di esecuzione, connettiti al database e verifica le operazioni sui dati.

Passaggi

1. Inoltra la porta del servizio PostgreSQL alla tua macchina locale:

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

2. Da un terminale diverso, connettiti al database utilizzando psql:

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

3. Crea un database e una tabella, quindi popola la tabella con i dati:

```
CREATE DATABASE employee;
```

Passa al nuovo database (comando meta-psql):

```
\c employee
```

Crea la tabella, inserisci una riga ed esegui una query sui risultati:

```
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;
```

Video dimostrativo

Il seguente video mostra l'installazione di Trident su un cluster Kubernetes vSphere e la distribuzione di un database PostgreSQL con storage persistente utilizzando Trident.

Distribuire carichi di lavoro su un cluster vSphere Kubernetes Service supportato da storage persistente ONTAP usando Trident

Protezione dei dati

Esegui il backup e il ripristino delle applicazioni container in un cluster vSphere Kubernetes Service usando NetApp Console

Esegui il backup e il ripristino delle applicazioni container in un cluster vSphere Kubernetes Service (VKS) utilizzando NetApp Console. Questa procedura copre la creazione e la gestione delle operazioni di backup e ripristino tramite Backup and Recovery Service tramite NetApp Console, utilizzando ONTAP S3 object storage come destinazione del backup.

NetApp Console offre una gestione centralizzata dei servizi di storage e dati in ambienti on-premises e cloud. Offre controllo unificato, semplicità operativa e gestione sicura. Sono disponibili diversi servizi dati e funzionalità di gestione accessibili dall'interfaccia utente su "console.netapp.com". Per dettagli completi su

NetApp Console, vedi "Documentazione di NetApp Console".

Questa sezione si concentra sul servizio dati NetApp Backup and Recovery per i carichi di lavoro Kubernetes, in particolare sulle applicazioni container sui cluster vSphere Kubernetes Service. Il servizio NetApp Backup and Recovery ti permette di individuare, gestire, creare e associare policy di protezione alle tue applicazioni Kubernetes tramite un'unica interfaccia. Per una guida completa, vedi "NetApp Backup and Recovery documentazione".

Flusso di lavoro di backup e ripristino

1. **Assicurati di avere un agente Console**

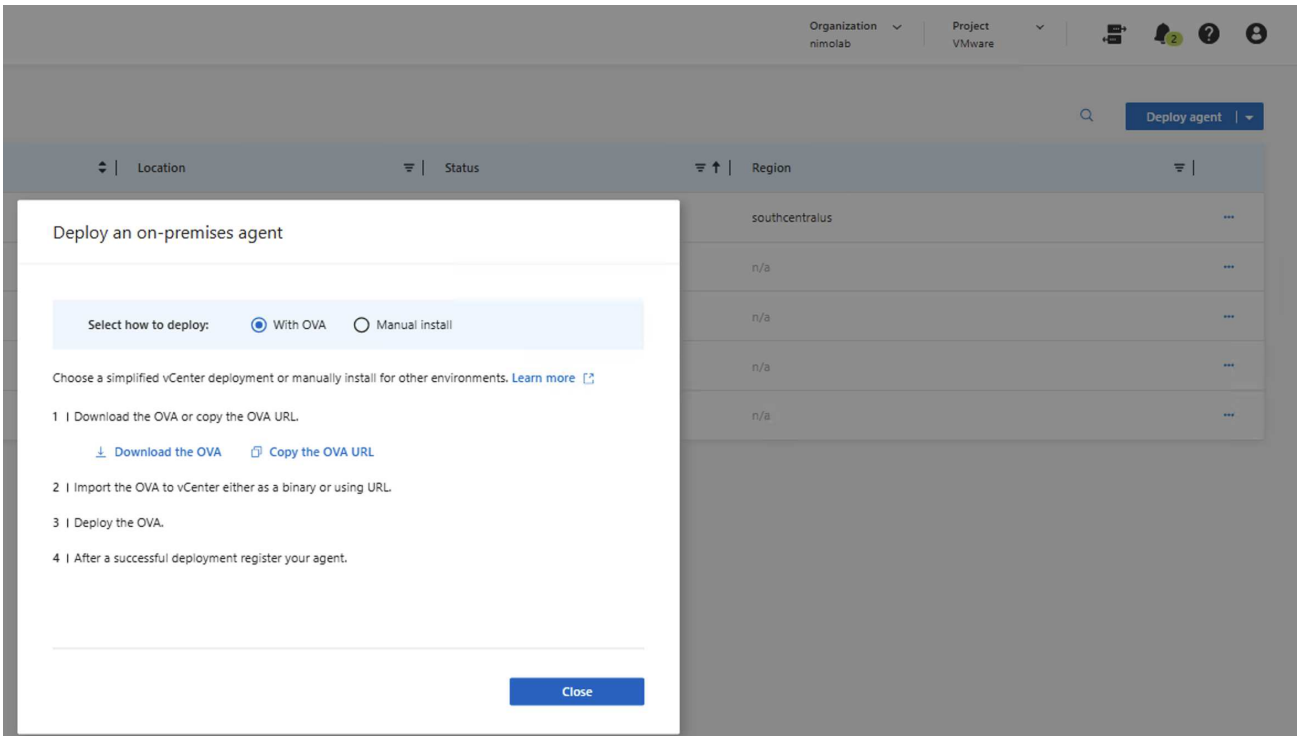
   Assicurati di aver distribuito un agente Console nell'ambiente in cui è in esecuzione il cluster VKS. Per informazioni dettagliate sull'installazione di un agente Console, consulta la "NetApp Console documentazione di configurazione".

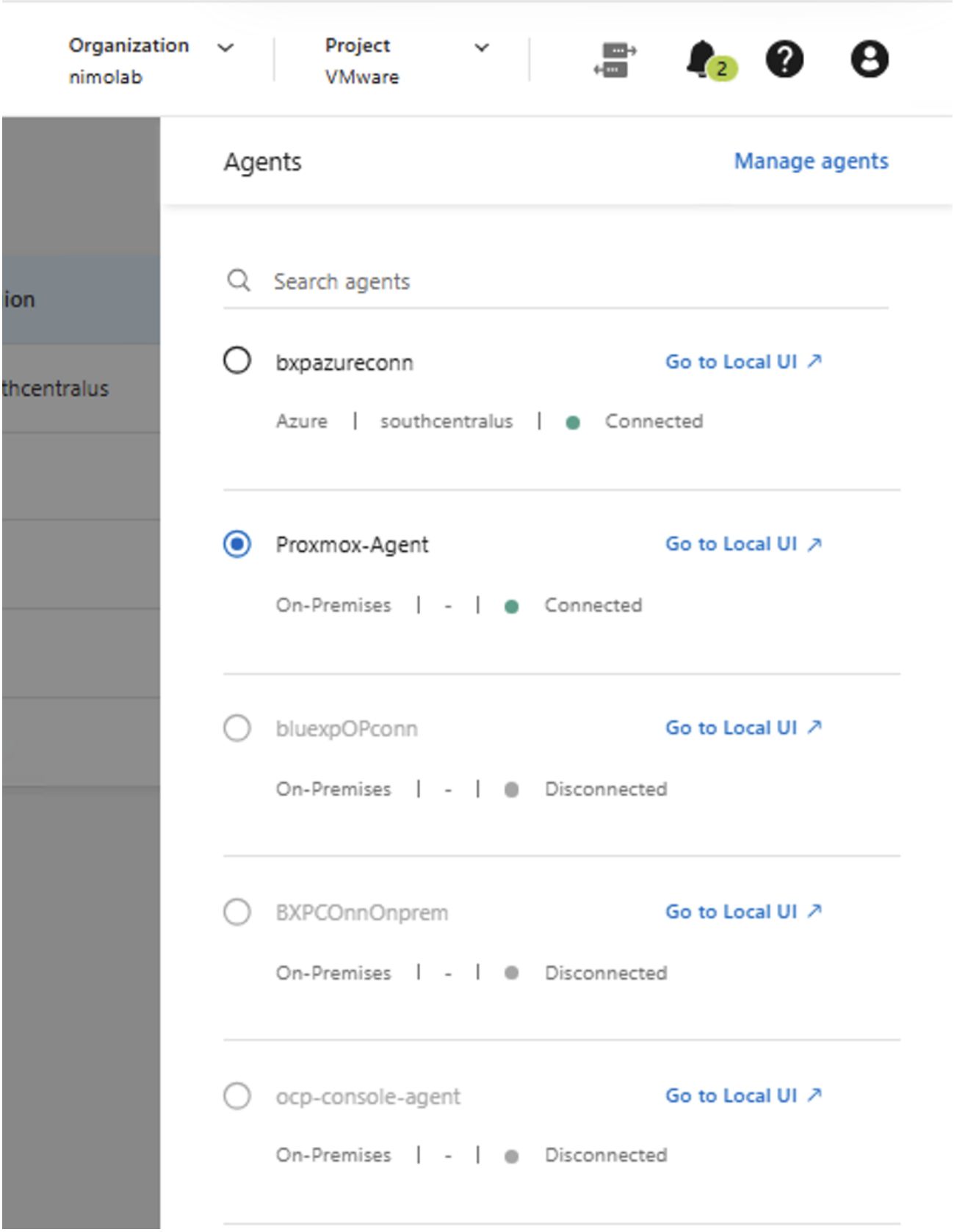
   ▼ **Mostra esempio**

   Nella NetApp Console, fai clic su Deploy Agent come mostrato nello screenshot e distribuisci l'agente nell'ambiente in cui è in esecuzione il cluster VKS.

   In questo esempio, seleziona **On-Premises > Con OVA**, copia l'URL OVA e usa quell'URL in vCenter per distribuire l'agente della Console.

   Registralo utilizzando l'indirizzo IP dell'agente nell'URL. Vedi "la documentazione" per istruzioni dettagliate. Dopo che l'agente è stato registrato, puoi vederlo in NetApp Console come mostrato nello screenshot qui sotto.





2. **Aggiungi il cluster ONTAP a NetApp Console e abilita NetApp Backup and Recovery**

   Il cluster ONTAP primario deve essere aggiunto alla NetApp Console e il servizio di backup e ripristino deve essere abilitato su di esso prima che i carichi di lavoro possano essere protetti. Per istruzioni, vedi "Configura NetApp Backup and Recovery".

3. **Nella NetApp Console, scopri il vSphere Kubernetes cluster**

▼ **Mostra esempio**

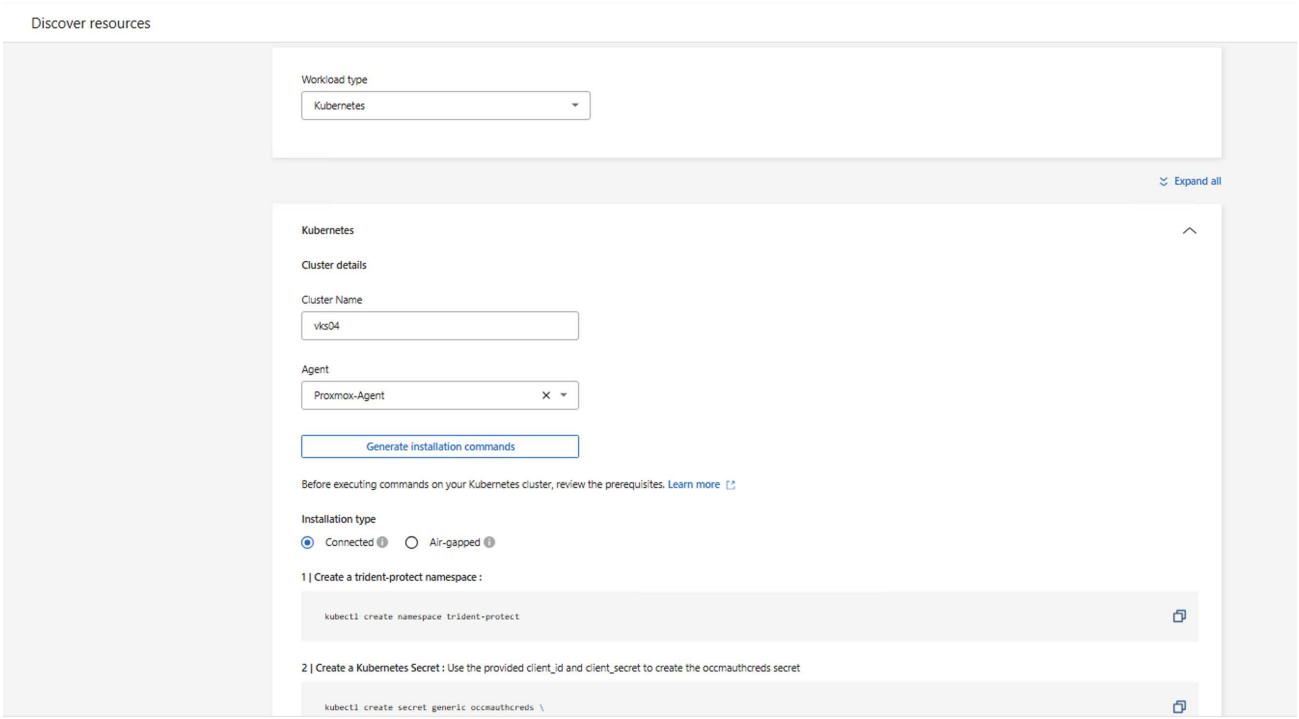
Questo passaggio richiede che il vSphere Kubernetes cluster sia in esecuzione e che l'agente della Console sia distribuito nello stesso ambiente. Segui questi passaggi per individuare il vSphere Kubernetes cluster nella NetApp Console.

a. In NetApp Console, seleziona **Inventory > Kubernetes** e fornisci i dettagli del vSphere Kubernetes cluster.

b. Seleziona l'agente distribuito nello stesso ambiente in cui si trova il cluster Kubernetes vSphere.

c. Copia i comandi forniti per installare Trident Protect ed eseguili da una macchina connessa al tuo vSphere Kubernetes cluster.

d. Dopo aver installato correttamente Trident Protect, fai clic su **Scopri**. NetApp Console rileva il vSphere Kubernetes cluster e tutte le sue risorse tramite l'agente e le visualizza nell'interfaccia utente.



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
```

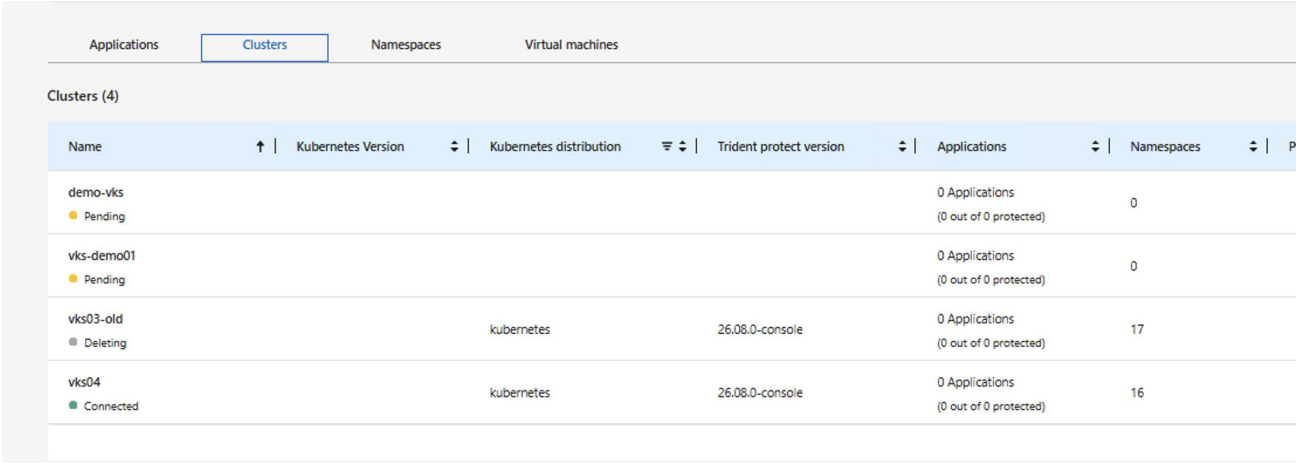
3 | Add/update helm repo :

```
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. **Configura ONTAP S3 come destinazione di backup**

   In questo esempio, utilizzi ONTAP S3 storage a oggetti come destinazione di backup. Puoi anche usare AWS S3, Azure Blob, Google Cloud Storage o StorageGRID come destinazioni di backup.

   Per ulteriori dettagli, consulta "Documentazione di NetApp Console".

   Per aggiungere ONTAP S3 come destinazione di backup nella NetApp Console, avrai bisogno delle seguenti informazioni dal tuo ONTAP cluster.

   ```
   S3 Endpoint:
   ```

```
Access Key:
Secret Key:
Bucket Name:
```

Per ottenere questi valori, procedi come segue nel tuo ONTAP cluster utilizzando System Manager.

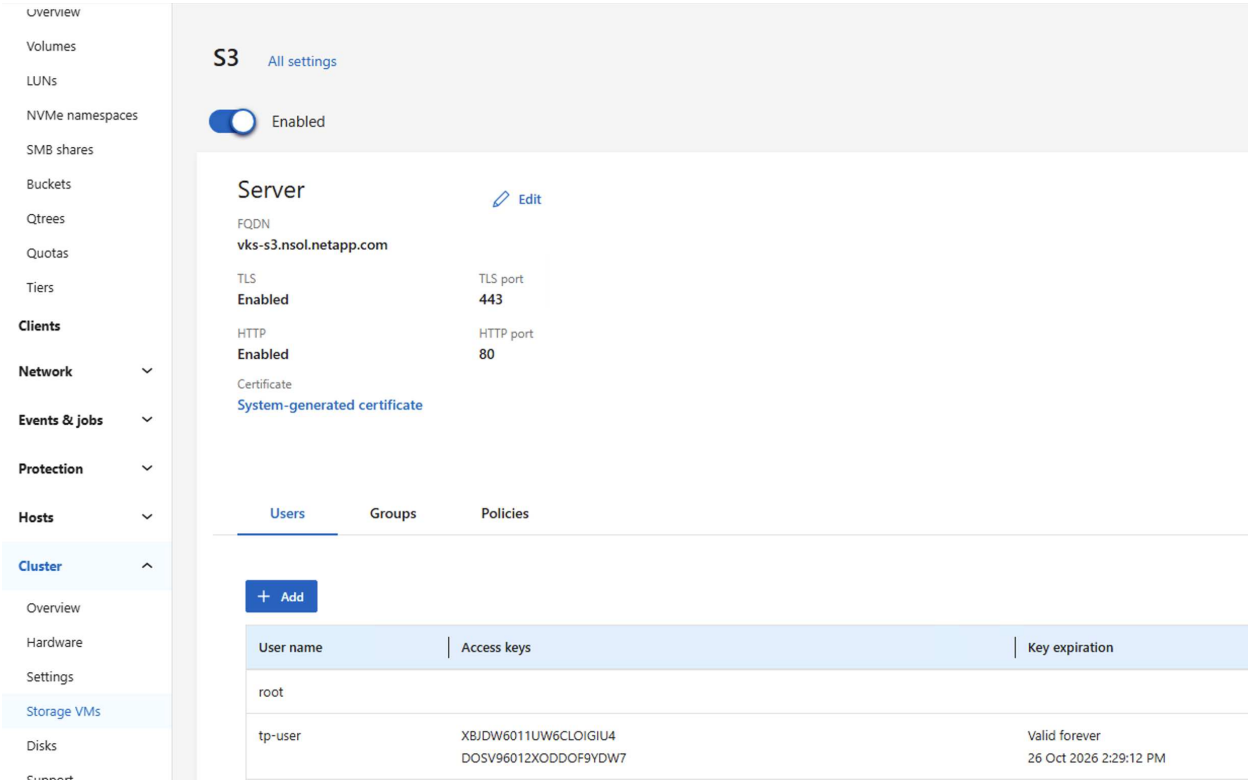
a. Crea un SVM abilitato per S3 nel cluster ONTAP per archiviare i dati di backup. Prendi nota dell'URL dell'endpoint S3 per questo SVM.

b. Nelle impostazioni S3 della SVM, crea un utente e assegnagli i permessi di accesso necessari. Annota la access key e la secret key per questo utente.

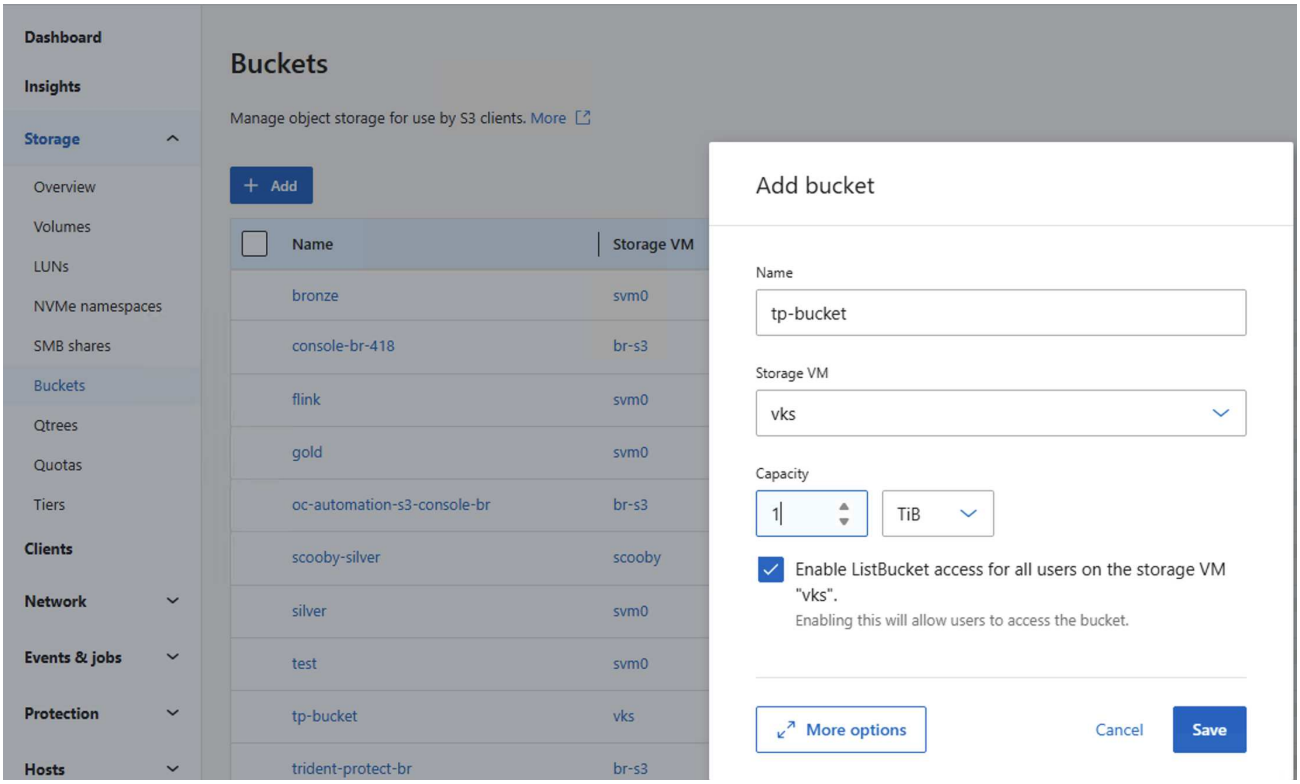
> Trident Protect richiede che l'utente S3 abbia almeno `PutObject`, `GetObject`, `ListBucket` e `DeleteObject` autorizzazioni. Senza queste autorizzazioni, le operazioni di backup e ripristino AppVault non riescono e viene visualizzato un errore di accesso negato. Per dettagli, vedi "Documentazione di Trident Protect AppVault".

▼ **Mostra esempio**

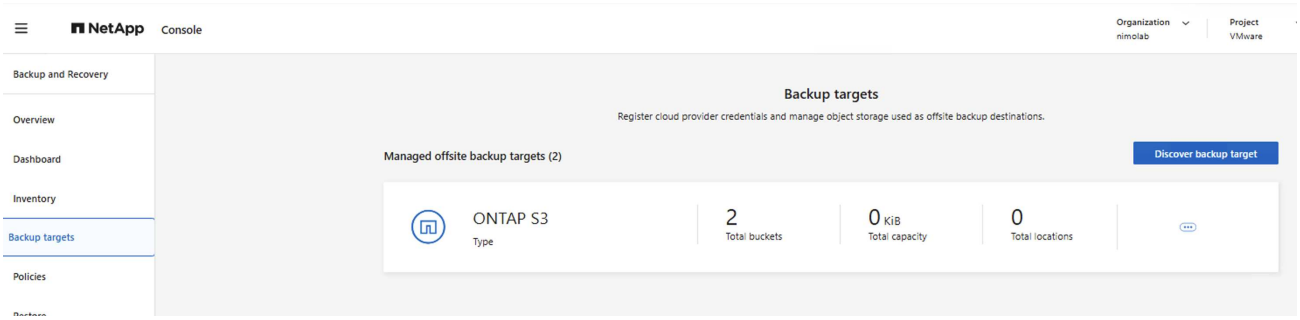
i. **Crea un utente S3 e scarica la chiave di accesso e la chiave segreta**

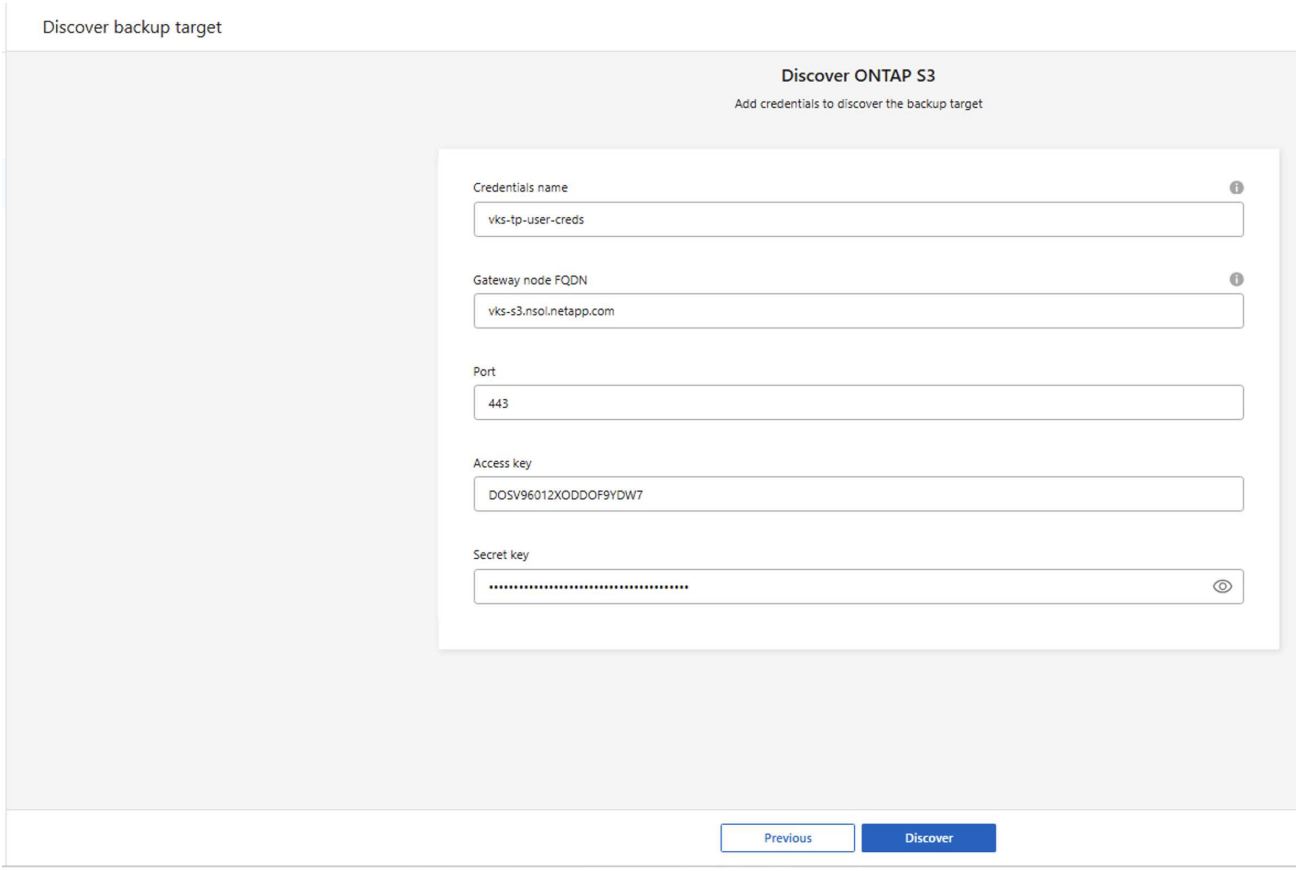


A. Crea un bucket nell'SVM abilitato per S3. Annota il nome del bucket da utilizzare quando aggiungi lo storage a oggetti ONTAP come destinazione di backup in NetApp Console.



Aggiungi lo storage a oggetti ONTAP come destinazione di backup in NetApp Console utilizzando l'endpoint S3, la chiave di accesso, la chiave segreta e il nome del bucket.





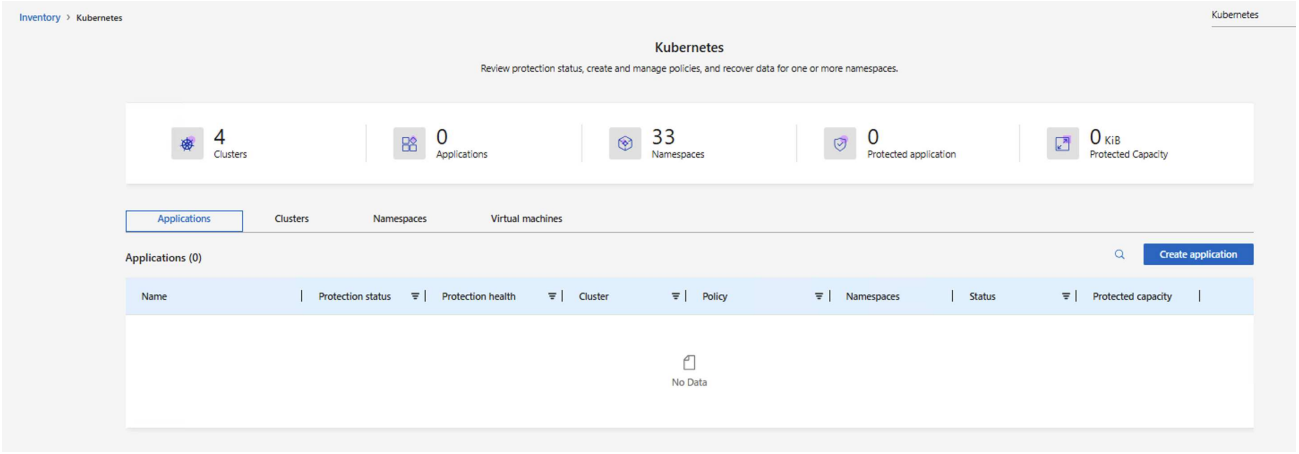
5. **Nella NetApp Console, crea un'applicazione per i carichi di lavoro dei container e proteggila utilizzando i backup**

> Per creare e proteggere un'applicazione Kubernetes, devi avere il ruolo di super admin di NetApp Backup and Recovery o di backup admin di NetApp Backup and Recovery. Per ripristinare un'applicazione Kubernetes, devi avere il ruolo di super admin di NetApp Backup and Recovery o di restore admin di NetApp Backup and Recovery.

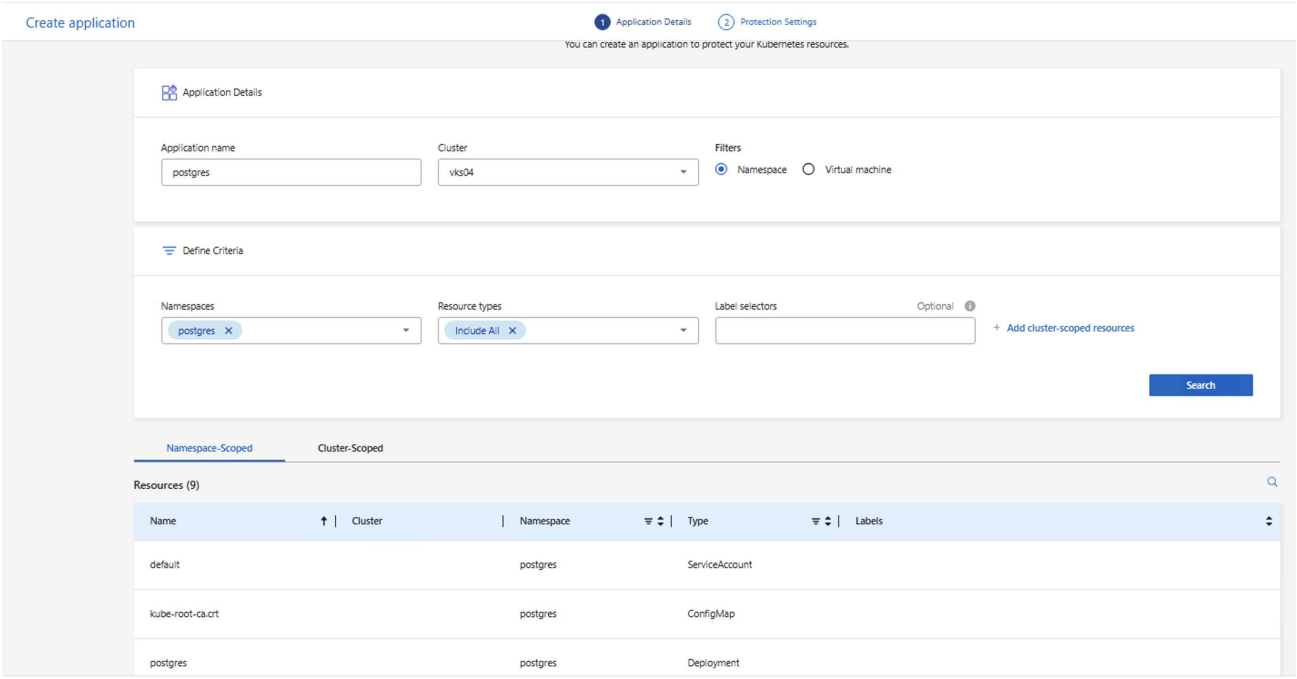
▼ **Mostra esempio**

In questo passaggio, crea un'applicazione in NetApp Console per le applicazioni container nel tuo cluster Kubernetes vSphere che devono essere protette. Puoi usare i namespace per creare un'applicazione che protegga tutte le applicazioni container in un namespace nel cluster Kubernetes vSphere.

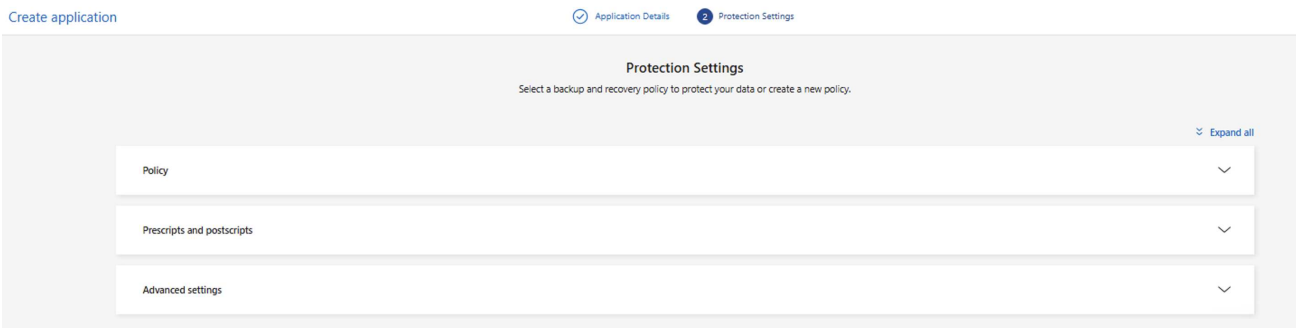
Avrai anche bisogno di una policy di protezione da associare all'applicazione. Puoi creare una nuova policy di protezione o usarne una esistente.



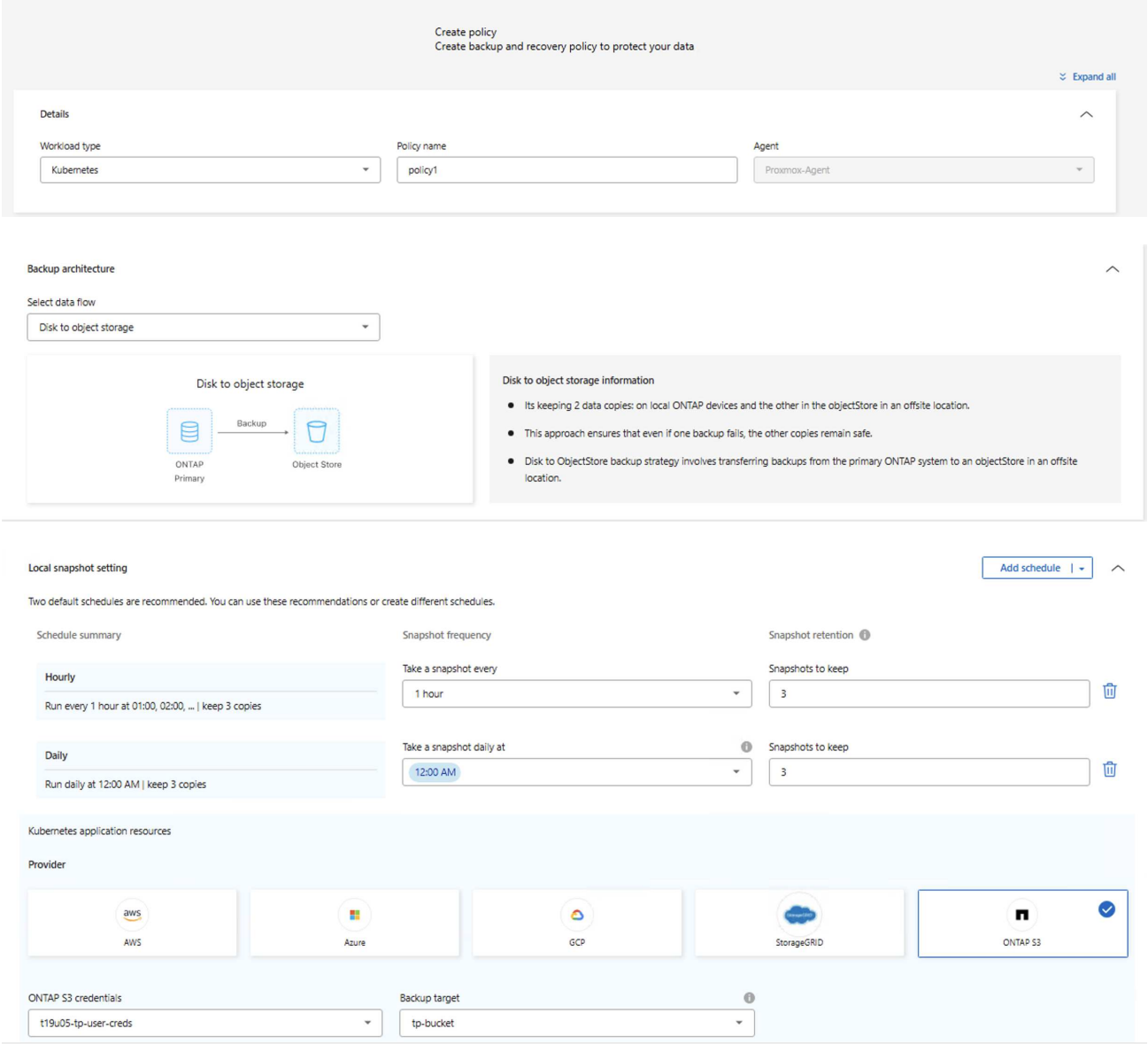
Fornisci i dettagli dell'applicazione e definisci i criteri per l'applicazione. In questo esempio, l'applicazione viene creata per tutte le applicazioni container in un namespace nel cluster Kubernetes vSphere. Fai clic su **Cerca** per vedere l'elenco delle risorse con ambito namespace che fanno parte dell'applicazione. Fai clic su **Avanti** per procedere.

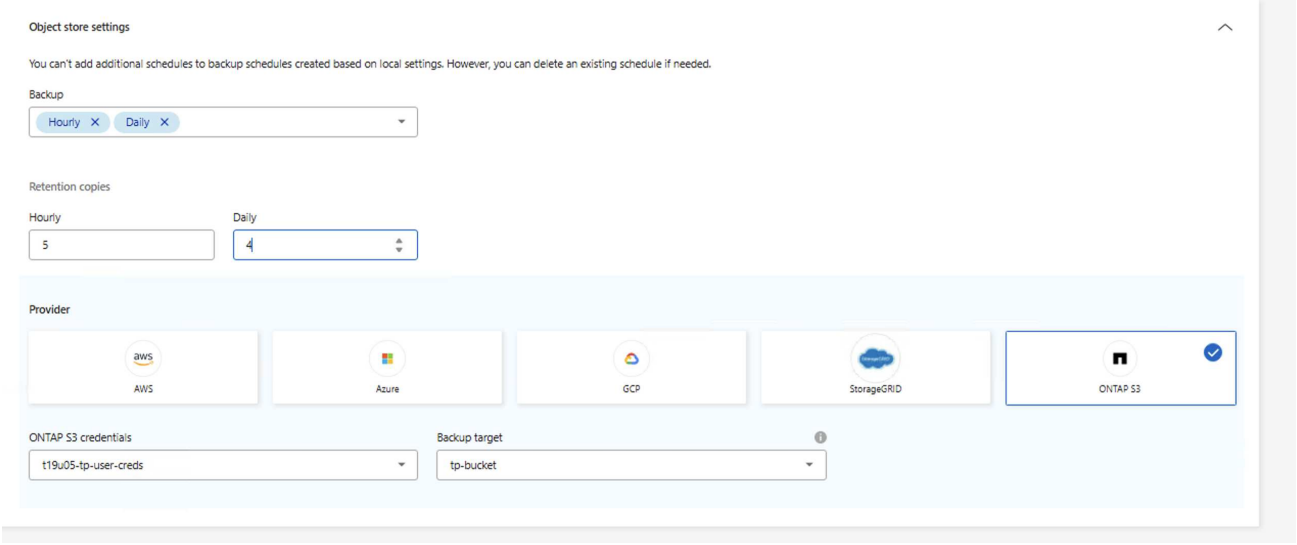


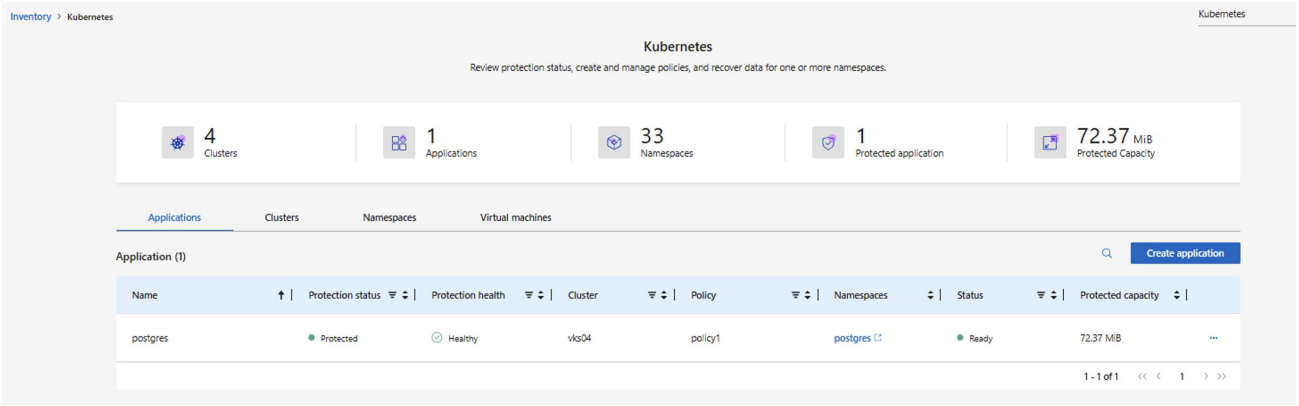
Successivamente, devi fornire le impostazioni di protezione per l'applicazione. Puoi creare una nuova policy di protezione o usarne una esistente. In questo esempio, viene creata una nuova policy di protezione per l'applicazione e ad essa associata. Espandi **Policy** e fai clic su **Crea nuova policy**.



Fornisci un nome per la policy e scegli l'architettura di backup. In questo esempio, viene scelta l'opzione "da disco a object storage". Definisci la pianificazione degli snapshot locali, che controlla la frequenza con cui vengono creati gli snapshot sullo storage primario. Poi configura separatamente le impostazioni di backup dell'object storage: scegli la destinazione del backup (in questo esempio, ONTAP S3), imposta la pianificazione del trasferimento che controlla quando i dati degli snapshot vengono copiati nell'object storage e specifica il numero di copie da conservare. La pianificazione del trasferimento è indipendente da quella degli snapshot, quindi le risorse vengono copiate nell'object storage in base alla pianificazione del trasferimento, non immediatamente quando viene creato ogni snapshot. Puoi anche modificare il numero massimo di tentativi e l'intervallo tra i tentativi, se necessario. Fai clic su **Crea** per procedere. L'applicazione verrà rilevata e la policy di protezione verrà associata ad essa.





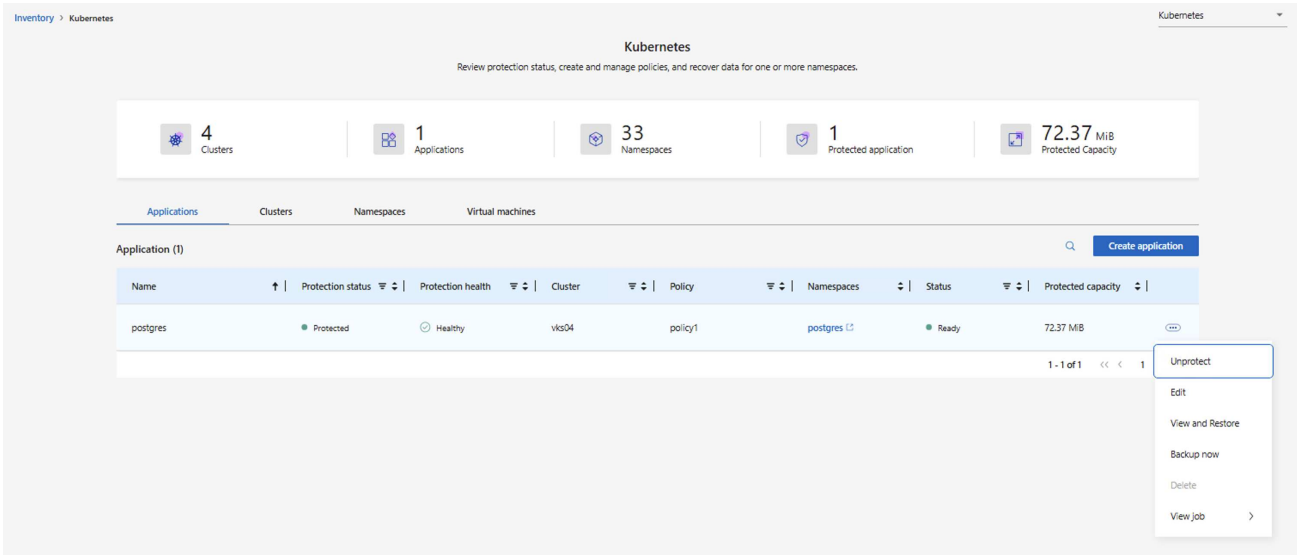


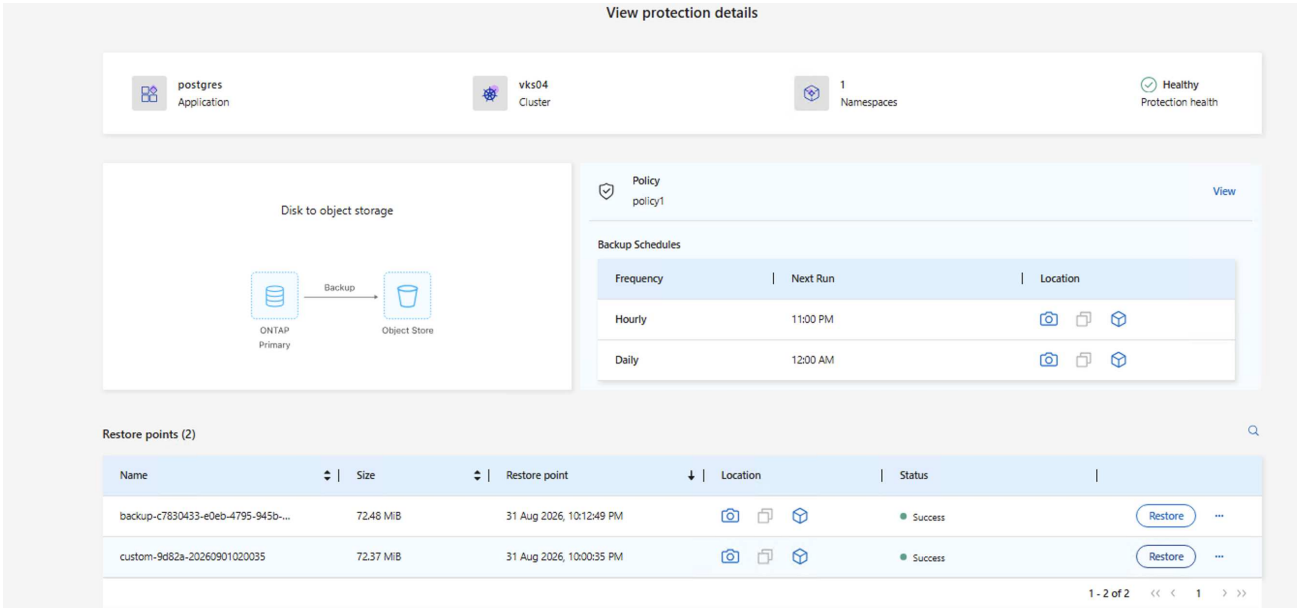
6. **Ripristina l'applicazione da un backup**

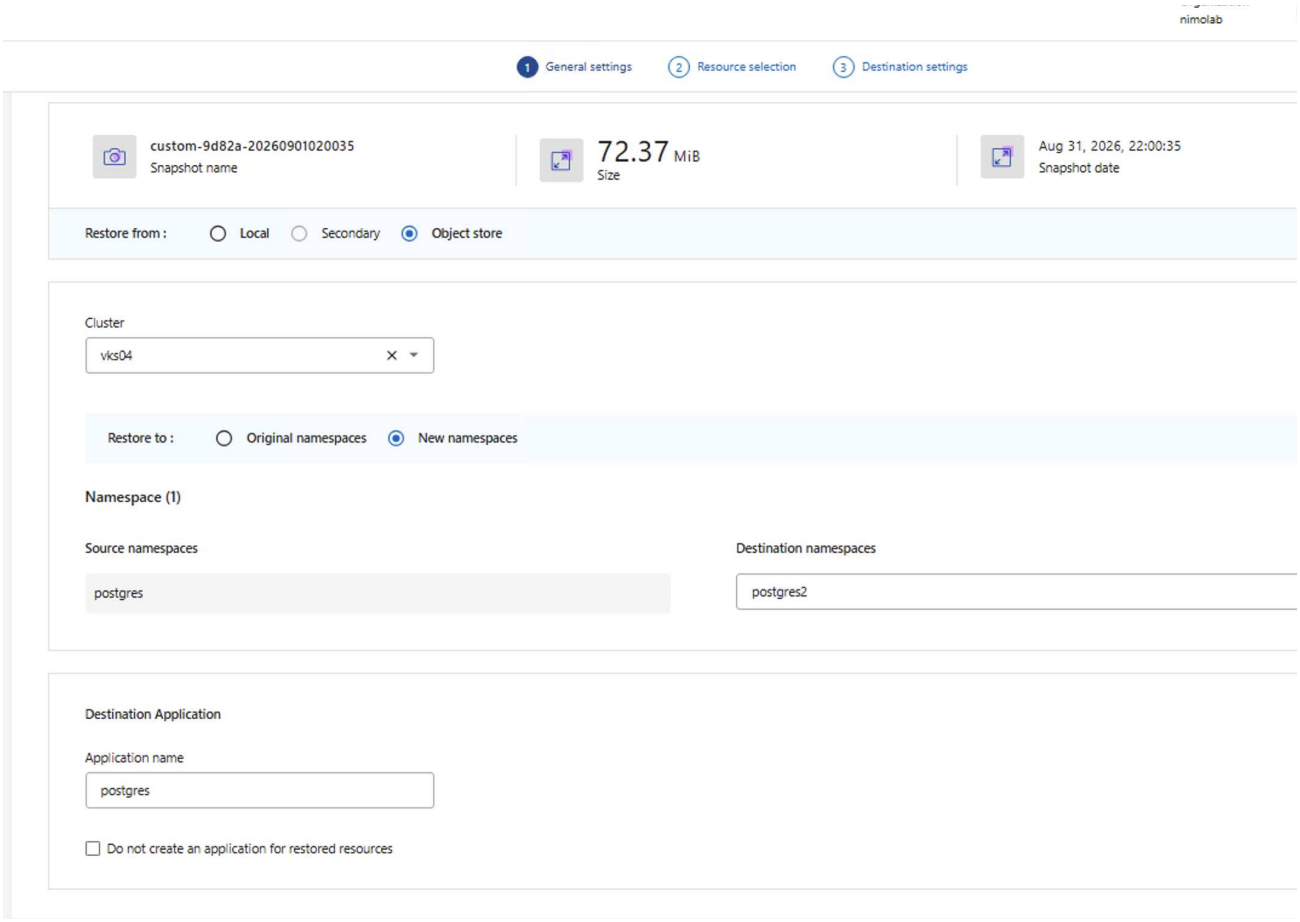
Per ripristinare un'applicazione Kubernetes, devi avere il ruolo di super admin di Backup and Recovery o di restore admin di Backup and Recovery.

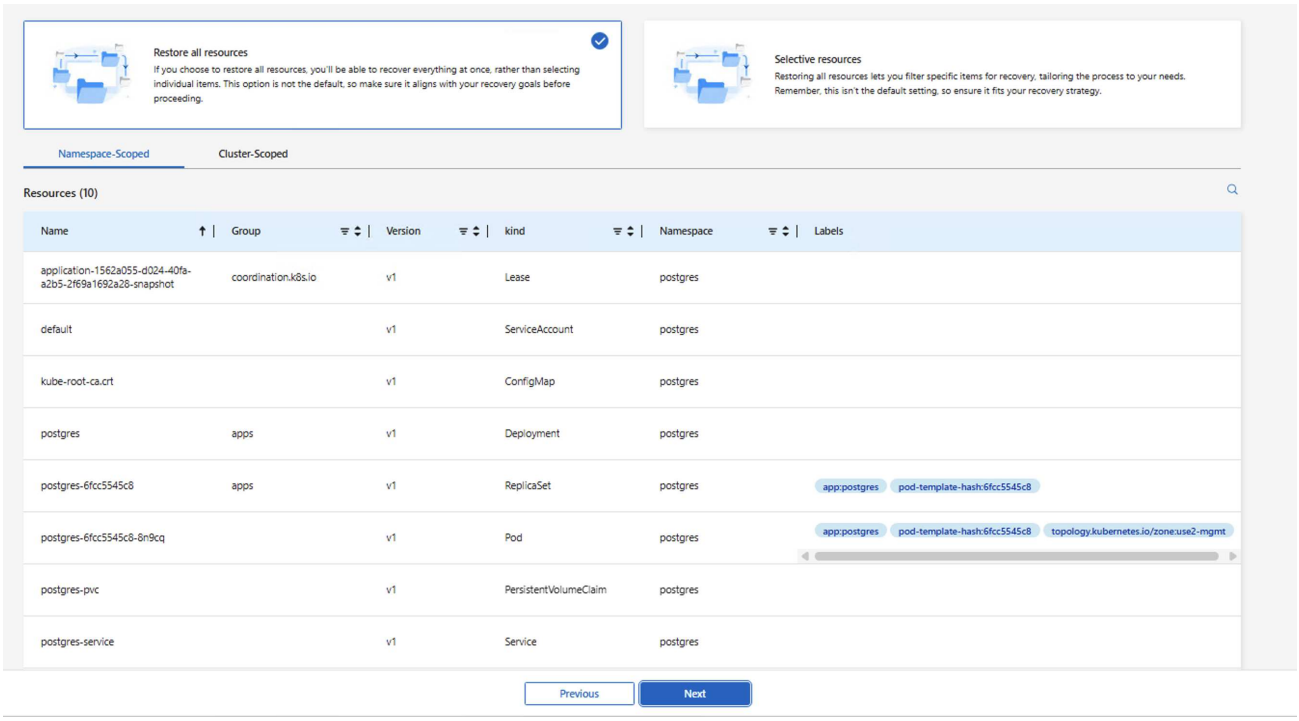
▼ **Mostra esempio**

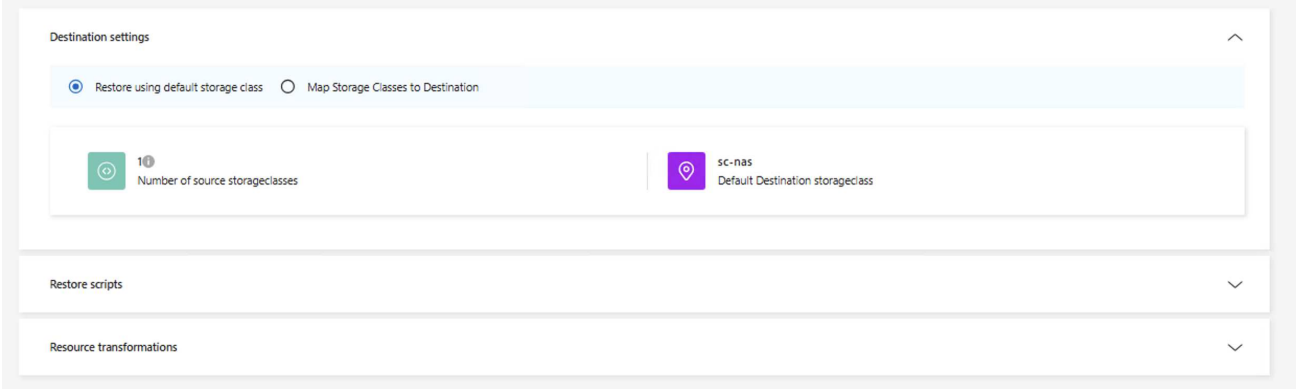
- Puoi scegliere qualsiasi punto di ripristino disponibile — un'istantanea locale, un backup su storage a oggetti o un backup da una destinazione secondaria — per ripristinare l'applicazione.
- Puoi ripristinare l'applicazione nel suo namespace originale o in un namespace diverso.
- Puoi selezionare le risorse da includere nel ripristino.
- Puoi scegliere la classe di storage da usare per le risorse dell'applicazione ripristinate.


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

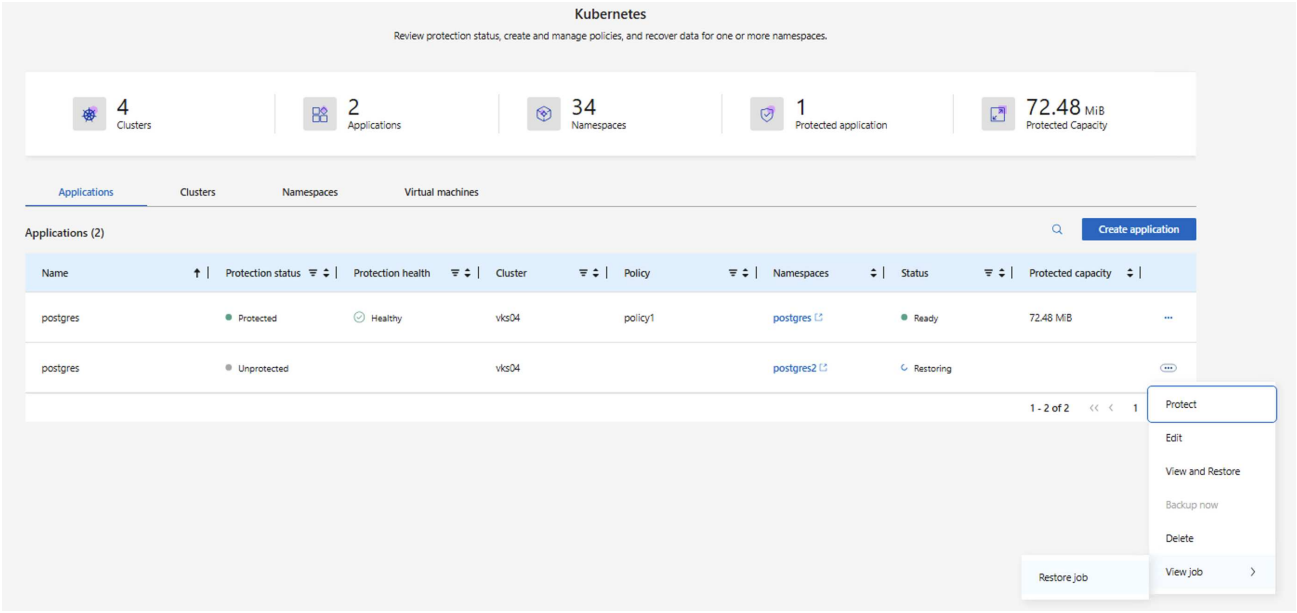

View protection details
postgres
Application
vks04
Cluster
1
Namespaces
Healthy
Protection health
Disk to object storage
Backup
ONTAP
Primary
Object Store
Policy
policy1
View
Backup Schedules
Frequency
Next Run
Location
Hourly
11:00 PM
Daily
12:00 AM
Restore points (2)
Name
Size
Restore point
Location
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

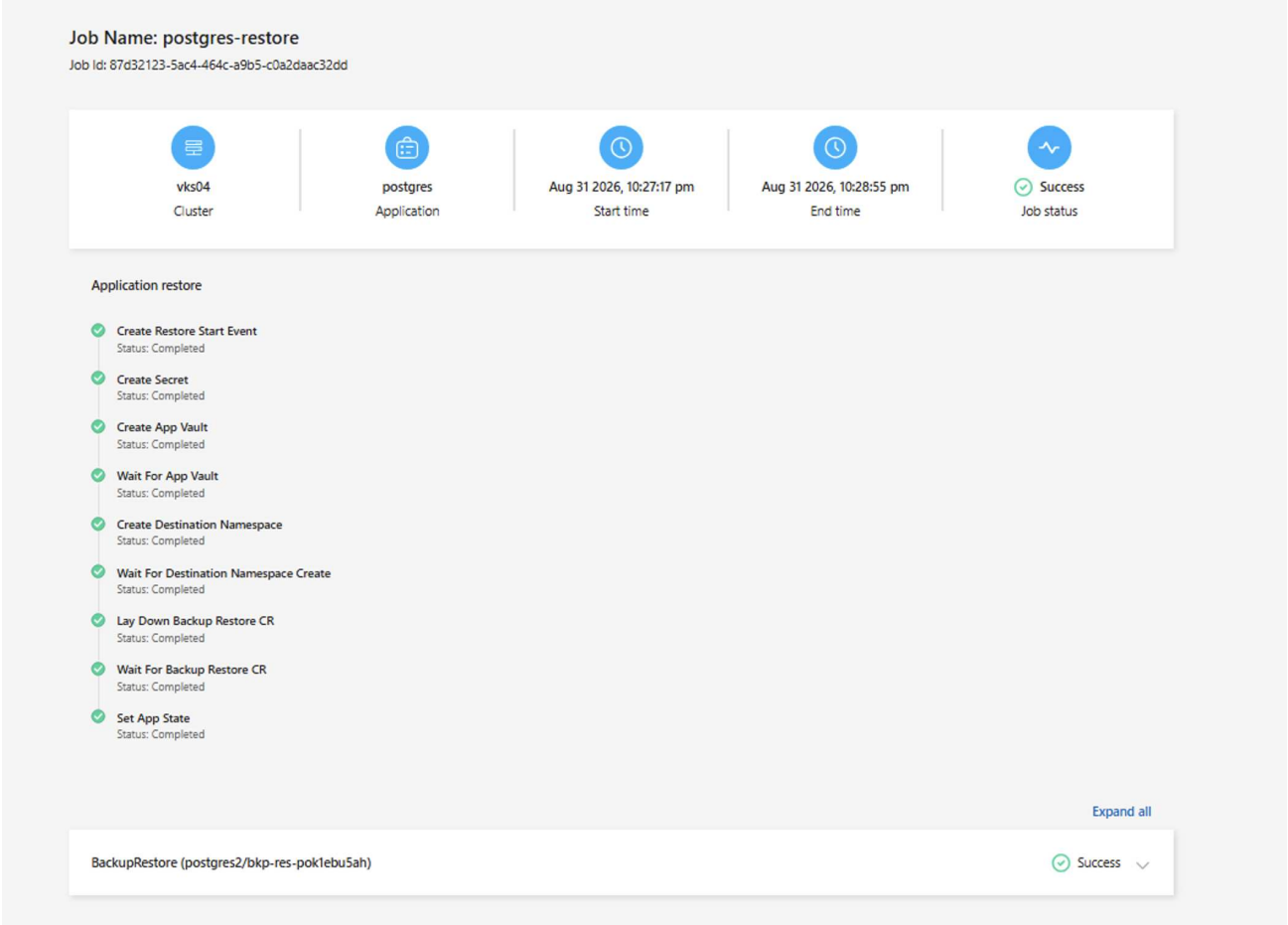

nimolab
1 General settings
2 Resource selection
3 Destination settings
custom-9d82a-20260901020035
Snapshot name
72.37 MiB
Size
Aug 31, 2026, 22:00:35
Snapshot date
Restore from :
Local
Secondary
Object store
Cluster
vks04
Restore to :
Original namespaces
New namespaces
Namespace (1)
Source namespaces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

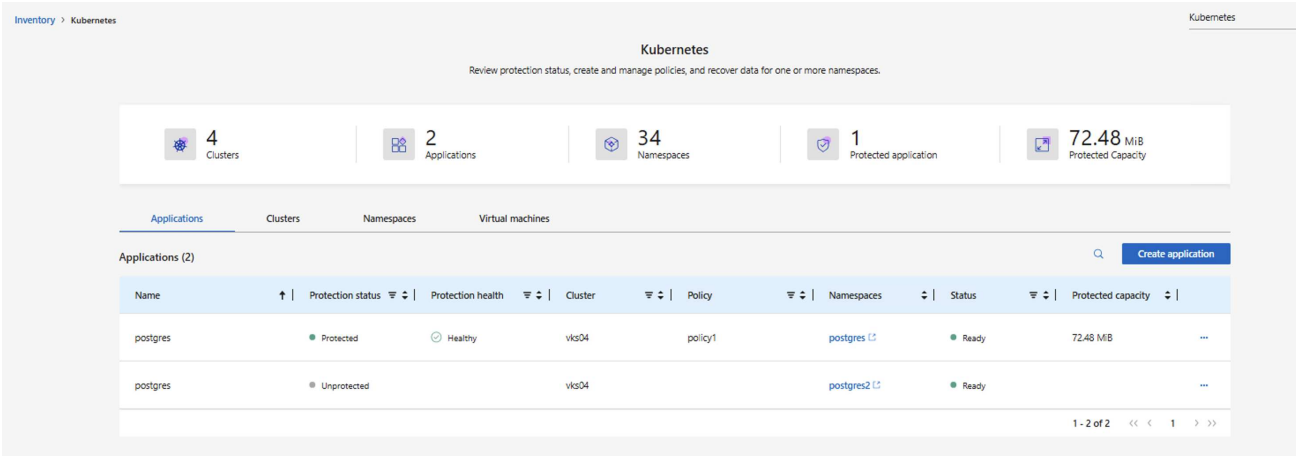

Restore all resources
If you choose to restore all resources, you'll be able to recover everything at once, rather than selecting individual items. This option is not the default, so make sure it aligns with your recovery goals before proceeding.
Selective resources
Restoring all resources lets you filter 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



Puoi guardare l'avanzamento del processo di ripristino nella sezione Monitor di NetApp Console. Dopo che il processo di ripristino è stato completato, puoi vedere le risorse ripristinate nel vSphere Kubernetes cluster.







Esegui il backup e il ripristino delle applicazioni container in un cluster vSphere Kubernetes Service utilizzando Trident Protect

Esegui il backup e il ripristino delle applicazioni container in un cluster vSphere Kubernetes Service (VKS) utilizzando snapshot e backup di Trident Protect. Questa procedura include la creazione di un AppVault utilizzando ONTAP S3 storage a oggetti, la configurazione di Trident Protect per acquisire i dati dell'applicazione, inclusi gli oggetti risorsa Kubernetes e i volumi persistenti, e il ripristino dei dati quando necessario.

Le applicazioni container in esecuzione in un cluster VKS vengono gestite come workload Kubernetes negli spazi dei nomi dei nodi worker. È importante proteggere sia i metadati dell'applicazione che i volumi

persistenti, così che, se vengono persi o danneggiati, puoi recuperarli.

I volumi persistenti delle applicazioni VKS possono essere supportati da ONTAP storage integrato con il cluster VKS usando "Trident CSI". Questa procedura utilizza "Trident Protect" per creare snapshot delle applicazioni, i cui metadati sono archiviati nello storage a oggetti ONTAP mentre i dati degli snapshot dei volumi rimangono nel backend di storage, e backup, i cui dati vengono copiati nello storage a oggetti ONTAP. Puoi ripristinare sia da uno snapshot che da un backup quando necessario.

Trident Protect ti permette di creare snapshot, backup, ripristinare e gestire il disaster recovery delle applicazioni su un cluster Kubernetes. I dati che puoi proteggere con Trident Protect includono gli oggetti risorsa Kubernetes associati alle applicazioni e i relativi volumi persistenti.

Di seguito sono riportate le versioni dei vari componenti utilizzati per gli esempi in questa sezione

- VMware Cloud Foundation (VCF) 9.1 con vSphere Kubernetes Service
- "Trident 26.06"
- "Trident Protect 26.06"
- "ONTAP 9.16"

Installa Trident Protect sul vSphere cluster

▼ **Installa Trident Protect**

Usa Helm per installare Trident Protect sul vSphere Kubernetes Service cluster. Gli esempi in questa sezione usano `tridentctl-protect`, la Trident Protect CLI. Installala seguendo le istruzioni nel "Documentazione CLI di Trident Protect". Metodi aggiuntivi per installare Trident Protect sono documentati nel "Documentazione Trident Protect".

Innanzitutto, crea e assegna un'etichetta al `trident-protect` namespace. L' `enforce=privileged` etichetta è necessaria perché Trident Protect esegue carichi di lavoro privilegiati. Senza di essa, il controller di ammissione alla sicurezza dei pod di Kubernetes impedirà l'avvio dei pod di Trident Protect.

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

Successivamente, aggiungi il repository Helm e installa Trident Protect nello spazio dei nomi.

```
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
```

Se i pod di Trident Protect non si avviano a causa di un `PodSecurity` errore di ammissione, è possibile che le etichette di sicurezza dei pod non siano state applicate allo `trident-protect` namespace prima dell'installazione. Usa `--overwrite` per riapplicarle, poi verifica che i pod si avviino.

```
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$
```

Crea App Vault per lo storage a oggetti

▼ **Crea AppVault**

Prima di creare snapshot e backup per un'applicazione, devi configurare lo storage a oggetti in Trident Protect. Questo si fa creando un AppVault CR. Solo gli amministratori possono creare e configurare un AppVault CR.

Gli oggetti AppVault sono la rappresentazione come risorsa personalizzata di Kubernetes di un bucket di storage. Un AppVault CR contiene le configurazioni necessarie affinché un bucket possa essere utilizzato nelle operazioni di protezione, come backup, snapshot, operazioni di ripristino e replica SnapMirror.

I seguenti passaggi creano un AppVault CR configurato per ONTAP S3: Crea un server di object store S3 nell'SVM del cluster ONTAP. . Crea un bucket nel server di object store. . Crea un utente S3 nell'SVM. Tieni la access key e la secret key in un luogo sicuro.

+ NOTA: Trident Protect richiede che l'utente S3 disponga almeno delle autorizzazioni `PutObject`, `GetObject`, `ListBucket` e `DeleteObject`. Senza queste autorizzazioni, le operazioni di backup e ripristino AppVault non riescono con errori di accesso negato. Per i dettagli, vedi "Documentazione di Trident Protect AppVault". . Nel cluster VKS, crea un secret per archiviare le credenziali ONTAP S3. . Crea un oggetto AppVault per ONTAP S3.

Configura Trident Protect AppVault per ONTAP S3

> ⓘ Il manifesto seguente utilizza HTTPS con la convalida del certificato abilitata, necessaria per la produzione. Se il tuo endpoint ONTAP S3 utilizza un certificato firmato da una CA pubblica già considerata attendibile dal cluster, non è necessaria alcuna configurazione aggiuntiva del certificato. Se utilizza un certificato CA autofirmato o interno, fornisci il certificato CA radice personalizzato usando il `rootCA` campo come mostrato nel manifesto. Guarda la variante solo per laboratorio alla fine di questa sezione se ti serve una configurazione HTTP per test isolati non di produzione.

```
# 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 solo per laboratorio (HTTP, senza convalida del certificato)

Usa solo le seguenti `s3` impostazioni in un ambiente di laboratorio isolato dove non sono presenti dati sensibili. Non usare queste impostazioni in produzione.

```
    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$
```

Crea un'applicazione Trident Protect

▼ **Crea un'applicazione Trident Protect**

Questo esempio illustra la protezione dei dati per l'applicazione postgres di esempio, installata nel namespace postgres. Per dettagli sull'installazione, vedi "Distribuisci i carichi di lavoro VKS utilizzando NetApp Storage".

> ⓘ I PVC di esempio di PostgreSQL richiedono la modalità di accesso RWO. La modalità di accesso deve essere specificata esplicitamente nel manifesto del PVC; Trident non seleziona automaticamente una modalità di accesso in base al protocollo storage. RWX è supportato per i PVC basati su NAS, e i PVC basati su SAN possono supportare RWX con una configurazione aggiuntiva. Per ulteriori informazioni, consulta la "Documentazione Trident Protect".

Nell'esempio, il namespace postgres contiene un'applicazione e tutte le risorse del namespace vengono incluse durante la creazione della Trident Protect Application.

```
# 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
```

Proteggi l'app creando un backup

▼ **Crea backup**

Crea un backup su richiesta

Crea un backup per la postgres-app creata in precedenza che includa tutte le risorse presenti nel namespace postgres. Specifica il nome dell'appvault in cui verranno archiviati i backup.

```
# 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
```

Crea backup secondo una pianificazione

Crea una pianificazione per i backup specificando la granularità e il numero di backup da conservare.

```
# 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
```

Ripristino da backup

▼ **Ripristina dai backup**

Ripristina l'applicazione nello stesso namespace

In questo esempio, il backup postgres-backup-on-demand contiene il backup per postgres-app.

Prima di eliminare l'applicazione, verifica che il backup da cui intendi eseguire il ripristino sia nello stato `Completed`. Un backup in corso o non riuscito non può essere utilizzato per ripristinare l'applicazione.

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

Dopo aver confermato che lo stato del backup è `Completed`, elimina l'applicazione postgres e assicurati che i PVC e gli oggetti pod siano eliminati dal namespace "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.
```

Ora, crea un oggetto di ripristino in-place del backup.

```
# 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
```

Verifica che il deployment dell'applicazione postgres, il servizio, il pod e i PVC siano stati ripristinati.

```
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
```

Ripristina l'applicazione in un namespace diverso

Innanzitutto, crea un nuovo namespace in cui vuoi ripristinare l'app, in questo esempio postgres2. Ora è disponibile un backup orario creato dalla pianificazione per la postgres-app. Usa questo backup per ripristinare l'applicazione nel nuovo namespace 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
```

Per ottenere il percorso del backup, usa il comando `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
```

Verifica che gli oggetti dell'applicazione postgres e i PVC siano creati nel nuovo namespace 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$
```

Proteggi l'app utilizzando le Snapshot

▼ Crea Snapshot

Crea uno snapshot su richiesta Crea uno snapshot per l'app e specifica l'AppVault dove i metadati dello snapshot verranno archiviati. I dati dello snapshot del volume non vengono copiati nello storage a oggetti: rimangono nel backend di storage.

```
# 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$
```

Crea una pianificazione per gli snapshot Crea una pianificazione per gli snapshot. Specifica la granularità e il numero di snapshot da conservare.

```
# 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
```

Ripristina da Snapshot

▼ Ripristina da Snapshot

Ripristina l'applicazione da uno Snapshot nello stesso namespace

Prima di eliminare l'applicazione, verifica che lo snapshot da cui vuoi eseguire il ripristino sia nello stato `Completed`. Uno snapshot in corso o non riuscito non può essere usato per ripristinare l'applicazione.

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

Dopo aver confermato che lo stato dello Snapshot è `Completed`, elimina gli oggetti dell'applicazione e i PVC per postgres dal namespace 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$ |
```

Crea un oggetto snapshot-in-place-restore dallo 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
```

Verifica che gli oggetti e i PVC dell'applicazione siano creati nel namespace 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$
```

Ripristina l'applicazione da uno Snapshot in un namespace diverso

Elimina l'applicazione nel namespace postgres2 precedentemente ripristinata dal backup.

```
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$
```

Crea l'oggetto di ripristino dello snapshot dallo snapshot e fornisci la mappatura dello spazio dei nomi.

```
# 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
```

Verifica che gli oggetti e i PVC dell'applicazione siano ripristinati nel 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$
```

Informazioni sul copyright

Informazioni sul marchio commerciale