



Provisionnement et gestion des volumes

Trident

NetApp
February 02, 2026

Sommaire

Provisionnement et gestion des volumes	1
Provisionner un volume	1
Présentation	1
Créer la PVC	1
Développement des volumes	5
Développez un volume iSCSI	5
Développez un volume FC	9
Développez un volume NFS	13
Importer des volumes	16
Présentation et considérations	16
Importer un volume	17
Exemples	19
Personnaliser les noms et les étiquettes des volumes	26
Avant de commencer	26
Limites	26
Comportements clés des noms de volume personnalisables	27
Exemples de configuration back-end avec modèle de nom et étiquettes	27
Exemples de modèles de noms	28
Points à prendre en compte	29
Partager un volume NFS entre les espaces de noms	29
Caractéristiques	29
Démarrage rapide	30
Configurer les espaces de noms source et de destination	31
Supprimer un volume partagé	32
Utiliser <code>tridentctl get</code> pour interroger des volumes subordonnés	32
Limites	33
Pour en savoir plus	33
Cloner des volumes entre des espaces de noms	33
Prérequis	33
Démarrage rapide	33
Configurer les espaces de noms source et de destination	34
Limites	36
Réplication de volumes à l'aide de SnapMirror	36
Conditions préalables à la réplication	36
Créer une demande de volume persistant en miroir	37
États de réplication des volumes	40
Promotion de la demande de volume persistant secondaire en cas de basculement non planifié	40
Promotion de la demande de volume persistant secondaire lors d'un basculement planifié	40
Restaurer une relation de miroir après un basculement	41
Opérations supplémentaires	41
Mettre à jour les relations miroir lorsque ONTAP est en ligne	42
Mettre à jour les relations en miroir lorsque ONTAP est hors ligne	42
Utiliser la topologie CSI	42

Présentation	43
Étape 1 : création d'un back-end conscient de la topologie	44
Étape 2 : définissez des classes de stockage qui prennent en compte la topologie	46
Étape 3 : création et utilisation d'une demande de volume persistant	47
Mise à jour des systèmes back-end pour inclure supportedTopologies	50
Trouvez plus d'informations	50
Travailler avec des instantanés	50
Présentation	50
Créer un snapshot de volume	51
Créer une demande de volume persistant à partir d'un snapshot de volume	52
Importer un instantané de volume	53
Restaurez les données de volume à l'aide de snapshots	55
Restauration de volumes sur place à partir d'un snapshot	55
Supprimez un volume persistant avec les snapshots associés	57
Déployer un contrôleur de snapshot de volume	57
Liens connexes	58
Travailler avec des instantanés de groupes de volumes	58
Créer des instantanés de groupes de volumes	59
Récupérer des données de volume à l'aide d'un instantané de groupe	60
Restauration de volumes sur place à partir d'un snapshot	61
Supprimer un PV avec des instantanés de groupe associés	61
Déployer un contrôleur de snapshot de volume	61
Liens connexes	62

Provisionnement et gestion des volumes

Provisionner un volume

Créez une demande de volume persistant qui utilise la classe de stockage Kubernetes configurée pour demander l'accès au volume persistant. Vous pouvez ensuite monter le volume persistant sur un pod.

Présentation

Une "[PersistentVolumeClaim](#)" demande de volume persistant est une demande d'accès au volume persistant sur le cluster.

Le PVC peut être configuré pour demander un stockage d'une certaine taille ou d'un certain mode d'accès. À l'aide de la classe de stockage associée, l'administrateur du cluster peut contrôler plus que la taille du volume persistant et le mode d'accès, tels que les performances ou le niveau de service.

Une fois la demande de volume créée, vous pouvez la monter dans un pod.

Créer la PVC

Étapes

1. Créer la PVC.

```
kubectl create -f pvc.yaml
```

2. Vérifiez l'état de la demande de volume persistant.

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-storage	Bound	pv-name	1Gi	RWO		5m

1. Montez le volume dans un pod.

```
kubectl create -f pv-pod.yaml
```



Vous pouvez surveiller la progression à l'aide de `kubectl get pod --watch`.

2. Vérifiez que le volume est monté sur `/my/mount/path`.

```
kubect1 exec -it task-pv-pod -- df -h /my/mount/path
```

3. Vous pouvez maintenant supprimer le Pod. L'application Pod n'existera plus, mais le volume restera.

```
kubect1 delete pod pv-pod
```

Exemples de manifestes

Exemples de manifestes de demande de volume persistant

Ces exemples présentent les options de configuration de base de la PVC.

PVC avec accès RWO

Cet exemple montre une demande de volume persistant de base avec accès RWO associée à une classe de stockage nommée `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC avec NVMe/TCP

Cet exemple présente une demande de volume persistant de base pour NVMe/TCP avec accès RWO associée à une classe de stockage nommée `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

Échantillons de manifeste de pod

Ces exemples présentent les configurations de base pour fixer la demande de volume persistant à un pod.

Configuration de base

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
  - name: storage
    persistentVolumeClaim:
      claimName: pvc-storage
  containers:
  - name: pv-container
    image: nginx
    ports:
    - containerPort: 80
      name: "http-server"
    volumeMounts:
    - mountPath: "/my/mount/path"
      name: storage
```

Configuration NVMe/TCP de base

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
  - name: basic-pvc
    persistentVolumeClaim:
      claimName: pvc-san-nvme
  containers:
  - name: task-pv-container
    image: nginx
    volumeMounts:
    - mountPath: "/my/mount/path"
      name: basic-pvc
```

Reportez-vous ["Kubernetes et objets Trident"](#) à pour plus de détails sur l'interaction des classes de stockage avec les PersistentVolumeClaim paramètres et pour le contrôle de la manière dont Trident provisionne les

volumes.

Développement des volumes

Trident offre aux utilisateurs de Kubernetes la possibilité d'étendre leurs volumes après leur création. Recherchez des informations sur les configurations requises pour étendre les volumes iSCSI, NFS, SMB, NVMe/TCP et FC.

Développez un volume iSCSI

Vous pouvez développer un volume persistant iSCSI à l'aide du mécanisme de provisionnement CSI.



L'extension de volume iSCSI est prise en charge par `ontap-san`, `ontap-san-economy`, `solidfire-san` Pilotes et requiert Kubernetes 1.16 et version ultérieure.

Étape 1 : configurer la classe de stockage pour prendre en charge l'extension de volume

Modifiez la définition de la classe de stockage pour définir le `allowVolumeExpansion` champ à `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Pour une classe de stockage déjà existante, modifiez-la pour l'inclure `allowVolumeExpansion` paramètre.

Étape 2 : créez une demande de volume persistant avec la classe de stockage que vous avez créée

Modifiez la définition de la demande de volume persistant et mettez à jour le `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.

```
cat pvc-ontapsan.yaml
```



```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san

```

Trident crée un volume persistant et l'associe à cette demande de volume persistant.

```

kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY   STATUS    CLAIM                                STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO          Delete          Bound      default/san-pvc  ontap-san                                10s

```

Étape 3 : définissez un pod qui fixe la demande de volume persistant

Reliez le volume persistant à un pod pour qu'il soit redimensionné. Lors du redimensionnement d'un volume persistant iSCSI, deux scénarios sont possibles :

- Si le volume persistant est connecté à un pod, Trident étend le volume sur le back-end de stockage, analyse à nouveau le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un volume persistant non attaché, Trident étend le volume sur le back-end de stockage. Une fois le volume de volume persistant lié à un pod, Trident analyse de nouveau le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille de la demande de volume persistant une fois l'opération d'extension terminée.

Dans cet exemple, un pod est créé et utilise le `san-pvc`.

```

kubectll get pod
NAME          READY   STATUS    RESTARTS   AGE
ubuntu-pod    1/1     Running   0           65s

kubectll describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
               csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod

```

Étape 4 : développez le volume persistant

Pour redimensionner la PV créée de 1Gi à 2Gi, modifiez la définition de la demande de volume persistant et mettez à jour la `spec.resources.requests.storage` À 2Gi.

```
kubectll edit pvc san-pvc
```

```

# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
# ...

```

Étape 5 : valider l'extension

Vous pouvez valider le fonctionnement correct de l'extension en vérifiant la taille de la demande de volume persistant, du volume PV et du volume Trident :

```
kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM          STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi        RWO
Delete              Bound      default/san-pvc  ontap-san    12m

tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID   | STATE | MANAGED |
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san |
| block      | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Développez un volume FC

Vous pouvez étendre un volume persistant FC à l'aide du mécanisme de provisionnement CSI.



L'extension de volume FC est prise en charge par le `ontap-san` pilote et requiert Kubernetes 1.16 et versions ultérieures.

Étape 1 : configurer la classe de stockage pour prendre en charge l'extension de volume

Modifiez la définition de la classe de stockage pour définir le `allowVolumeExpansion` champ à `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Pour une classe de stockage déjà existante, modifiez-la pour l'inclure `allowVolumeExpansion` paramètre.

Étape 2 : créez une demande de volume persistant avec la classe de stockage que vous avez créée

Modifiez la définition de la demande de volume persistant et mettez à jour le `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crée un volume persistant et l'associe à cette demande de volume persistant.

```
kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY   STATUS    CLAIM                                STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO          Delete          Bound     default/san-pvc  ontap-san                10s
```

Étape 3 : définissez un pod qui fixe la demande de volume persistant

Reliez le volume persistant à un pod pour qu'il soit redimensionné. Lors du redimensionnement d'un volume persistant FC, il existe deux scénarios :

- Si le volume persistant est connecté à un pod, Trident étend le volume sur le back-end de stockage, analyse à nouveau le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un volume persistant non attaché, Trident étend le volume sur le back-end de stockage. Une fois le volume de volume persistant lié à un pod, Trident analyse de

nouveau le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille de la demande de volume persistant une fois l'opération d'extension terminée.

Dans cet exemple, un pod est créé et utilise le `san-pvc`.

```
kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
ubuntu-pod    1/1     Running   0           65s

kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
               csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Étape 4 : développez le volume persistant

Pour redimensionner la PV créée de 1Gi à 2Gi, modifiez la définition de la demande de volume persistant et mettez à jour la `spec.resources.requests.storage` À 2Gi.

```
kubectl edit pvc san-pvc
```

```

# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
# ...

```

Étape 5 : valider l'extension

Vous pouvez valider le fonctionnement correct de l'extension en vérifiant la taille de la demande de volume persistant, du volume PV et du volume Trident :

```
kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM          STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi        RWO
Delete              Bound      default/san-pvc  ontap-san    12m

tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+-----+-----+
|          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san |
+-----+-----+-----+-----+-----+-----+
| block | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

Développez un volume NFS

Trident prend en charge l'extension de volume pour les PV NFS provisionnés sur `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, et `azure-netapp-files` backends.

Étape 1 : configurer la classe de stockage pour prendre en charge l'extension de volume

Pour redimensionner un volume persistant NFS, l'administrateur doit d'abord configurer la classe de stockage afin de permettre l'extension du volume en paramétrant le `allowVolumeExpansion` champ à `true`:

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si vous avez déjà créé une classe de stockage sans cette option, vous pouvez simplement modifier la classe

de stockage existante en utilisant `kubectl edit storageclass` pour permettre l'extension de volume.

Étape 2 : créez une demande de volume persistant avec la classe de stockage que vous avez créée

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident doit créer un PV NFS de 20 Mio pour ce PVC :

```
kubectl get pvc
NAME                STATUS    VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb        Bound      pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi
RWO                  ontapnas          9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY      STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi      RWO
Delete              Bound     default/ontapnas20mb  ontapnas
2m42s
```

Étape 3 : développez le volume persistant

Pour redimensionner le PV de 20 Mio nouvellement créé à 1 Gio, modifiez le PVC et définissez `spec.resources.requests.storage` à 1 Gio :

```
kubectl edit pvc ontapnas20mb
```

```

# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...

```

Étape 4 : valider l'extension

Vous pouvez valider le redimensionnement travaillé correctement en vérifiant la taille de la demande de volume persistant, de la valeur PV et du volume Trident :

```
kubectl get pvc ontapnas20mb
```

NAME	STATUS	VOLUME
ontapnas20mb	Bound	pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
RWO	ontapnas	4m44s


```
kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
```

NAME	CAPACITY	ACCESS MODES
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7	1Gi	RWO
Delete	Bound	default/ontapnas20mb
5m35s		ontapnas


```
tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
```

NAME	SIZE	STORAGE CLASS
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7	1.0 GiB	ontapnas
file	c5a6f6a4-b052-423b-80d4-8fb491a14a22	online
		true

Importer des volumes

Vous pouvez importer des volumes de stockage existants en tant que PV Kubernetes en utilisant `tridentctl import` ou en créant une revendication de volume persistant (PVC) avec des annotations d'importation Trident.

Présentation et considérations

Vous pouvez importer un volume dans Trident vers :

- Conteneurisation d'une application et réutilisation de son jeu de données existant
- Utilisez un clone d'un jeu de données pour une application éphémère
- Reconstruction d'un cluster Kubernetes en panne
- Migration des données applicatives pendant la reprise après incident

Considérations

Avant d'importer un volume, consultez les considérations suivantes.

- Trident peut importer des volumes ONTAP de type RW (lecture-écriture) uniquement. Les volumes de type

DP (protection des données) sont des volumes de destination SnapMirror. Vous devez rompre la relation de miroir avant d'importer le volume dans Trident.

- Nous vous suggérons d'importer des volumes sans connexions actives. Pour importer un volume activement utilisé, clonez-le, puis effectuez l'importation.



C'est particulièrement important pour les volumes en mode bloc, car Kubernetes ignorerait la connexion précédente et pourrait facilement relier un volume actif à un pod. Cela peut entraîner une corruption des données.

- Bien que `StorageClass` doit être spécifié sur une demande de volume persistant, Trident n'utilise pas ce paramètre lors de l'importation. Les classes de stockage sont utilisées lors de la création du volume pour sélectionner un pool disponible en fonction des caractéristiques de stockage. Comme le volume existe déjà, aucune sélection de pool n'est requise pendant l'importation. Par conséquent, l'importation n'échouera pas même si le volume existe sur un back-end ou un pool qui ne correspond pas à la classe de stockage spécifiée dans le PVC.
- La taille du volume existant est déterminée et définie dans la PVC. Une fois le volume importé par le pilote de stockage, le volume persistant est créé avec un `SécurRef` dans la demande de volume persistant.
 - La règle de récupération est initialement définie sur `retain` Dans la PV. Une fois que Kubernetes a réussi à relier la demande de volume persistant et le volume persistant, la règle de récupération est mise à jour pour correspondre à la règle de récupération de la classe de stockage.
 - Si la règle de récupération de la classe de stockage est `delete`, Le volume de stockage sera supprimé lorsque le volume persistant est supprimé.
- Par défaut, Trident gère le PVC et renomme le FlexVol volume et le LUN en arrière-plan. Vous pouvez réussir le `--no-manage` l'indicateur pour importer un volume non géré et le `--no-rename` indicateur permettant de conserver le nom du volume.
 - `--no-manage*` - Si vous utilisez le `--no-manage` Trident n'effectue aucune opération supplémentaire sur le PVC ou le PV pendant le cycle de vie des objets. Le volume de stockage n'est pas supprimé lorsque le PV est supprimé, et d'autres opérations telles que le clonage et le redimensionnement de volume sont également ignorées.
 - `--no-rename*` - Si vous utilisez le `--no-rename` Grâce à cet indicateur, Trident conserve le nom de volume existant lors de l'importation de volumes et gère le cycle de vie de ces volumes. Cette option est prise en charge uniquement pour le `ontap-nas`, `ontap-san` (y compris les systèmes ASA r2), et `ontap-san-economy` conducteurs.



Ces options sont utiles si vous souhaitez utiliser Kubernetes pour les charges de travail conteneurisées, mais que vous souhaitez par ailleurs gérer le cycle de vie du volume de stockage en dehors de Kubernetes.

- Une annotation est ajoutée pour la demande de volume persistant et la volume persistant, qui servent un double objectif : indiquer l'importation du volume et gérer la demande de volume persistant. Cette annotation ne doit pas être modifiée ni supprimée.

Importer un volume

Vous pouvez importer un volume en utilisant soit `tridentctl import` soit en créant un PVC avec des annotations d'importation Trident.



Si vous utilisez des annotations PVC, vous n'avez pas besoin de télécharger ou d'utiliser `tridentctl` pour importer le volume.

Utilisation de tridentctl

Étapes

1. Créez un fichier PVC (par exemple, `pvc.yaml`) qui sera utilisé pour créer le PVC. Le fichier PVC doit inclure `name`, `namespace`, `accessModes` et `storageClassName`. Vous pouvez éventuellement spécifier `unixPermissions` dans la définition de votre PVC.

Voici un exemple de spécification minimale :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```



N'indiquez que les paramètres requis. Les paramètres supplémentaires tels que le nom du PV ou la taille du volume peuvent entraîner l'échec de la commande d'importation.

2. Utilisez le `tridentctl import` commande permettant de spécifier le nom du backend Trident contenant le volume et le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume). Le `-f` Un argument est requis pour spécifier le chemin d'accès au fichier PVC.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Utilisation des annotations PVC

Étapes

1. Créez un fichier YAML PVC (par exemple, `pvc.yaml`) avec les annotations d'importation Trident requises. Le fichier PVC doit inclure :
 - `name` et `namespace` dans les métadonnées
 - `accessModes`, `resources.requests.storage`, et `storageClassName` dans les spécifications
 - Annotations :
 - `trident.netapp.io/importOriginalName`: Nom du volume sur le backend
 - `trident.netapp.io/importBackendUUID`: UUID du backend où le volume existe
 - `trident.netapp.io/notManaged` (*Facultatif*) : Définir sur `"true"` pour les volumes non gérés. La valeur par défaut est `"false"`.

Voici un exemple de spécification pour l'importation d'un volume géré :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Appliquez le fichier YAML PVC à votre cluster Kubernetes :

```
kubectl apply -f <pvc-file>.yaml
```

Trident importera automatiquement le volume et le liera au PVC.

Exemples

Consultez les exemples d'importation de volume suivants pour les pilotes pris en charge.

NAS ONTAP et FlexGroup NAS ONTAP

Trident prend en charge l'importation de volumes à l'aide des `ontap-nas` pilotes et `ontap-nas-flexgroup`.



- Trident ne prend pas en charge l'importation de volume à l'aide de `ontap-nas-economy` conducteur.
- Le `ontap-nas` et `ontap-nas-flexgroup` les pilotes n'autorisent pas les noms de volumes dupliqués.

Chaque volume créé avec le `ontap-nas` pilote est un FlexVol volume sur le cluster ONTAP. L'importation de volumes FlexVol avec le `ontap-nas` pilote fonctionne de la même manière. Les volumes FlexVol qui existent déjà sur un cluster ONTAP peuvent être importés en tant que `ontap-nas` demande de volume persistant. De même, les volumes FlexGroup peuvent être importés en tant que `ontap-nas-flexgroup` ESV.

Exemples ONTAP NAS utilisant `tridentctl`

Les exemples suivants montrent comment importer des volumes gérés et non gérés à l'aide de `tridentctl`.

Volume géré

L'exemple suivant importe un volume nommé `managed_volume` sur un système back-end nommé `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	true

Volume non géré

Lors de l'utilisation de l'`--no-manage`argument, Trident ne renomme pas le volume.

L'exemple suivant importe `unmanaged_volume` sur le `ontap_nas` back-end :

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file> --no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	false

Exemples ONTAP NAS utilisant des annotations PVC

Les exemples suivants montrent comment importer des volumes gérés et non gérés à l'aide d'annotations PVC.

Volume géré

L'exemple suivant importe un volume de 1 GiB ontap-nas nommé `ontap_volume1` à partir du backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` avec le mode d'accès RWO défini à l'aide d'annotations PVC :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume non géré

L'exemple suivant importe 1GiB ontap-nas volume nommé `ontap-volume2` depuis le backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` avec le mode d'accès RWO défini à l'aide d'annotations PVC :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```


SAN ONTAP

Trident prend en charge l'importation de volumes à l'aide du `ontap-san` (iSCSI, NVMe/TCP et FC) et `ontap-san-economy` conducteurs.

Trident peut importer des volumes ONTAP SAN FlexVol contenant un seul LUN. Ceci est cohérent avec le `ontap-san` pilote, qui crée un FlexVol volume pour chaque PVC et un LUN dans le FlexVol volume. Trident importe le FlexVol volume et l'associe à la définition PVC. Trident peut importer `ontap-san-economy` volumes contenant plusieurs LUN.

Les exemples suivants montrent comment importer des volumes gérés et non gérés :

Volume géré

Pour les volumes gérés, Trident renomme le FlexVol volume au `pvc-<uuid>` format et le LUN dans le FlexVol volume à `lun0`.

L'exemple suivant importe le `ontap-san-managed` FlexVol volume présent sur le `ontap_san_default` back-end :

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-  
basic-import.yaml -n trident -d
```

```
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
|          NAME          | SIZE | STORAGE CLASS |  
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic          |  
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true      |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+
```

Volume non géré

L'exemple suivant importe `unmanaged_example_volume` sur le `ontap_san` back-end :

```
tridentctl import volume -n trident san_blog unmanaged_example_volume  
-f pvc-import.yaml --no-manage
```

```
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
|          NAME          | SIZE | STORAGE CLASS |  
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+  
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog       |  
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false    |  
+-----+-----+-----+-----+  
+-----+-----+-----+-----+
```

Si des LUN sont mappées à des igroups qui partagent un IQN avec un IQN de nœud Kubernetes, comme dans l'exemple suivant, l'erreur s'affiche : `LUN already mapped to initiator(s) in this group`. Vous devez supprimer l'initiateur ou annuler le mappage de la LUN pour importer le volume.

Vserver	Igroup	Protocol	OS Type	Initiators
svm0	k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3	iscsi	linux	iqn.1994-05.com.redhat:4c2e1cf35e0
svm0	unmanaged-example-igroup	mixed	linux	iqn.1994-05.com.redhat:4c2e1cf35e0

Élément

Trident prend en charge le logiciel NetApp Element et l'importation de volumes NetApp HCI à l'aide du `solidfire-san` pilote.



Le pilote d'élément prend en charge les noms de volume dupliqués. Cependant, Trident renvoie une erreur s'il existe des noms de volumes dupliqués. Pour contourner ce problème, clonez le volume, indiquez un nom de volume unique et importez le volume cloné.

L'exemple suivant importe un `element-managed` volume sur le back-end `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          |  STATE  | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online  | true    |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Azure NetApp Files

Trident prend en charge l'importation de volumes à l'aide du `azure-netapp-files` pilote.



Pour importer un volume Azure NetApp Files, identifiez-le par son chemin d'accès au volume. Le chemin du volume est la partie du chemin d'exportation du volume après le `:/`. Par exemple, si le chemin de montage est `10.0.0.2:/importvol1`, le chemin du volume est `importvol1`.

L'exemple suivant importe un `azure-netapp-files` volume sur le back-end `azurenetaappfiles_40517` avec le chemin de volume `importvol1`.

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-pvc-file> -n trident
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
| PROTOCOL | BACKEND UUID | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage |
| file | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true |
+-----+-----+-----+
+-----+-----+-----+-----+
```

Google Cloud NetApp volumes

Trident prend en charge l'importation de volumes à l'aide du `google-cloud-netapp-volumes` pilote.

L'exemple suivant importe un volume sur le backend `backend-tbc-gcnv1` avec le volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-to-pvc> -n trident
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
| PROTOCOL | BACKEND UUID | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 | 10 GiB | gcnv-nfs-sc-
identity | file | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true |
|
+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+
```

L'exemple suivant importe un `google-cloud-netapp-volumes` volume lorsque deux volumes sont présents dans la même région :

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS
| PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 | 10 GiB | gcnv-nfs-sc-
identity | file      | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Personnaliser les noms et les étiquettes des volumes

Avec Trident, vous pouvez attribuer des noms et des libellés significatifs aux volumes que vous créez. Vous pouvez ainsi identifier et mapper facilement les volumes vers leurs ressources Kubernetes respectives. Vous pouvez également définir des modèles au niveau du back-end pour créer des noms de volume personnalisés et des étiquettes personnalisées. Tous les volumes que vous créez, importez ou clonez seront conformes aux modèles.

Avant de commencer

Prise en charge des noms et des libellés de volumes personnalisables :

- Opérations de création, d'importation et de clonage de volumes.
- Dans le cas du `ontap-nas-economy` pilote, seul le nom du volume Qtree est conforme au modèle de nom.
- Dans le cas du `ontap-san-economy` pilote, seul le nom du LUN est conforme au modèle de nom.

Limites

- Les noms de volumes personnalisés sont compatibles uniquement avec les pilotes ONTAP sur site.
- Les étiquettes personnalisées ne sont prises en charge que pour les `ontap-san`, `ontap-nas`, et `ontap-nas-flexgroup` conducteurs.
- Les noms de volumes personnalisés ne s'appliquent pas aux volumes existants.

Comportements clés des noms de volume personnalisables

- Si une défaillance survient en raison d'une syntaxe incorrecte dans un modèle de nom, la création du back-end échoue. Toutefois, si l'application modèle échoue, le volume sera nommé conformément à la convention de nommage existante.
- Le préfixe de stockage n'est pas applicable lorsqu'un volume est nommé à l'aide d'un modèle de nom de la configuration back-end. Toute valeur de préfixe souhaitée peut être directement ajoutée au modèle.

Exemples de configuration back-end avec modèle de nom et étiquettes

Les modèles de noms personnalisés peuvent être définis au niveau de la racine et/ou du pool.

Exemple de niveau racine

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
      "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.RequestName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Exemple au niveau du pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Exemples de modèles de noms

Exemple 1 :

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{ .config.BackendName }}"
```

Exemple 2 :

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{ slice .volume.RequestName 1 5 }}"
```

Points à prendre en compte

1. Dans le cas des importations en volume, les étiquettes ne sont mises à jour que si le volume existant comporte des étiquettes dans un format spécifique. Par exemple :
`{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Dans le cas des importations de volume gérées, le nom du volume suit le modèle de nom défini au niveau racine dans la définition du back-end.
3. Trident ne prend pas en charge l'utilisation d'un opérateur de tranche avec le préfixe de stockage.
4. Si les modèles ne donnent pas de noms de volumes uniques, Trident ajoute quelques caractères aléatoires pour créer des noms de volumes uniques.
5. Si le nom personnalisé d'un volume économique NAS dépasse 64 caractères, Trident nommera les volumes conformément à la convention de nommage existante. Pour tous les autres pilotes ONTAP, si le nom du volume dépasse la limite de nom, le processus de création du volume échoue.

Partager un volume NFS entre les espaces de noms

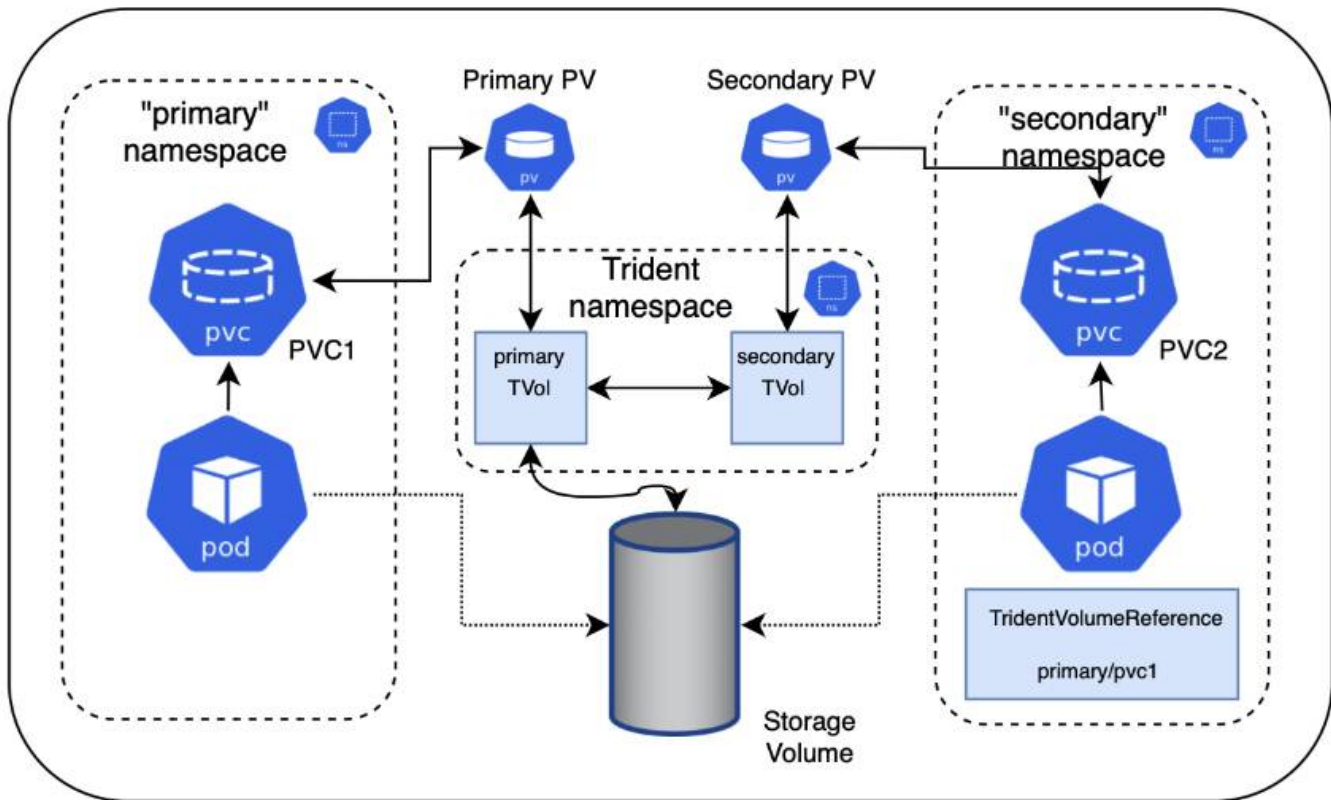
Avec Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

Caractéristiques

Le système TridentVolumeReference CR vous permet de partager en toute sécurité des volumes ReadWriteMany (RWX) NFS sur un ou plusieurs espaces de noms Kubernetes. Cette solution Kubernetes-native présente plusieurs avantages :

- Plusieurs niveaux de contrôle d'accès pour assurer la sécurité
- Fonctionne avec tous les pilotes de volume NFS Trident
- Pas de dépendance à `tridentctl` ou à toute autre fonctionnalité Kubernetes non native

Ce schéma illustre le partage de volumes NFS entre deux espaces de noms Kubernetes.



Démarrage rapide

Vous pouvez configurer le partage de volumes NFS en quelques étapes seulement.

1

Configurez la demande de volume persistant source pour partager le volume

Le propriétaire de l'espace de noms source autorise l'accès aux données dans la demande de volume persistant source.

2

Accorder l'autorisation de créer une demande de modification dans l'espace de noms de destination

L'administrateur de cluster accorde l'autorisation au propriétaire de l'espace de noms de destination pour créer le CR TridentVolumeReference.

3

Créer TridentVolumeReference dans l'espace de noms de destination

Le propriétaire de l'espace de noms de destination crée le CR TridentVolumeReference pour faire référence au PVC source.

4

Créez le PVC subalterne dans l'espace de noms de destination

Le propriétaire de l'espace de noms de destination crée le PVC subalterne pour utiliser la source de données à partir du PVC source.

Configurer les espaces de noms source et de destination

Pour garantir la sécurité, le partage de l'espace de noms croisé nécessite une collaboration et une action du propriétaire de l'espace de noms source, de l'administrateur de cluster et du propriétaire de l'espace de noms de destination. Le rôle utilisateur est désigné dans chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source:** Créez le PVC (`pvc1`) dans l'espace de noms source qui accorde l'autorisation de partager avec l'espace de noms de destination (`namespace2`) à l'aide de l'`shareToNamespace` annotation.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident crée le volume de stockage PV et son volume de stockage NFS back-end.



- Vous pouvez partager le PVC sur plusieurs espaces de noms à l'aide d'une liste délimitée par des virgules. Par exemple :
`trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4.`
- Vous pouvez partager avec tous les espaces de noms à l'aide de `*`. Par exemple :
`trident.netapp.io/shareToNamespace: *`
- Vous pouvez mettre à jour le PVC pour inclure le `shareToNamespace` annotation à tout moment.

2. **Administrateur de cluster :** assurez-vous qu'un RBAC approprié est en place pour accorder l'autorisation au propriétaire de l'espace de noms de destination de créer le CR `TridentVolumeReference` dans l'espace de noms de destination.
3. **Propriétaire de l'espace de noms de destination :** Créez un CR `TridentVolumeReference` dans l'espace de noms de destination qui fait référence à l'espace de noms source `pvc1`.

```

apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1

```

4. **Propriétaire de l'espace de noms de destination:** Créer un PVC (pvc2) dans l'espace de noms de destination (namespace2) à l'aide de l' `shareFromPVC` Annotation pour désigner la PVC source.

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi

```



La taille du PVC de destination doit être inférieure ou égale à la PVC source.

Résultats

Trident lit l' `shareFromPVC` annotation sur la demande de volume de destination et crée le volume persistant de destination en tant que volume subordonné sans ressource de stockage propre qui pointe vers le volume persistant source et partage la ressource de stockage du volume persistant source. La demande de volume persistant et la demande de volume persistant de destination apparaissent comme normales.

Supprimer un volume partagé

Vous pouvez supprimer un volume partagé entre plusieurs namespaces. Trident supprime l'accès au volume sur l'espace de noms source et maintient l'accès aux autres espaces de noms qui partagent le volume. Lorsque tous les espaces de noms qui référencent le volume sont supprimés, Trident supprime le volume.

Utiliser `tridentctl get` pour interroger des volumes subordonnés

À l'aide du `tridentctl` vous pouvez exécuter l' `get` commande pour obtenir des volumes subordonnés. Pour plus d'informations, consultez le lien [../trident-Reference/tridentctl.html](#) `tridentctl` commandes et

options].

```
Usage:
  tridentctl get [option]
```

Alarmes :

- `-h, --help`: Aide pour les volumes.
- `--parentOfSubordinate string`: Limiter la requête au volume source subordonné.
- `--subordinateOf string`: Limiter la requête aux subordonnés du volume.

Limites

- Trident ne peut pas empêcher les espaces de noms de destination d'écrire sur le volume partagé. Nous vous recommandons d'utiliser un verrouillage de fichiers ou d'autres processus pour éviter d'écraser les données du volume partagé.
- Vous ne pouvez pas révoquer l'accès au PVC source en retirant le `shareToNamespace` ou `shareFromNamespace` annotations ou suppression du `TridentVolumeReference` CR. Pour annuler l'accès, vous devez supprimer le PVC subalterne.
- Les snapshots, clones et la mise en miroir ne sont pas possibles sur les volumes subordonnés.

Pour en savoir plus

Pour en savoir plus sur l'accès aux volumes multi-espaces de noms :

- Visitez ["Partage de volumes entre les espaces de noms : dites bonjour à l'accès aux volumes situés à l'échelle d'un espace de noms"](#).
- Regardez la démo sur ["NetAppTV"](#).

Cloner des volumes entre des espaces de noms

Avec Trident, vous pouvez créer de nouveaux volumes à l'aide de volumes ou de volumes namespaces existants à partir d'un autre namespace situé dans le même cluster Kubernetes.

Prérequis

Avant de cloner des volumes, assurez-vous que les systèmes back-end source et de destination sont du même type et ont la même classe de stockage.



Le clonage entre les espaces de noms est pris en charge uniquement pour le `ontap-san` et `ontap-nas` pilotes de stockage. Les clones en lecture seule ne sont pas pris en charge.

Démarrage rapide

Vous pouvez configurer le clonage de volumes en quelques étapes seulement.

1**Configurez la demande de volume persistant source pour cloner le volume**

Le propriétaire de l'espace de noms source autorise l'accès aux données dans la demande de volume persistant source.

2**Accorder l'autorisation de créer une demande de modification dans l'espace de noms de destination**

L'administrateur de cluster accorde l'autorisation au propriétaire de l'espace de noms de destination pour créer le CR TridentVolumeReference.

3**Créer TridentVolumeReference dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée le CR TridentVolumeReference pour faire référence au PVC source.

4**Créer la demande de volume persistant de clone dans le namespace de destination**

Le propriétaire de l'espace de noms de destination crée une demande de volume persistant pour cloner la demande de volume persistant à partir de l'espace de noms source.

Configurer les espaces de noms source et de destination

Pour garantir la sécurité, le clonage de volumes entre espaces de noms nécessite la collaboration et l'action du propriétaire de l'espace de noms source, de l'administrateur du cluster et du propriétaire de l'espace de noms de destination. Le rôle utilisateur est désigné dans chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source** : Créez le PVC (namespace1) (pvc1` dans l'espace de noms source qui accorde l'autorisation de partager avec l'espace de noms de destination (`namespace2) à l'aide de l'`cloneToNamespace`annotation.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident crée le volume PV et son volume de stockage back-end.



- Vous pouvez partager le PVC sur plusieurs espaces de noms à l'aide d'une liste délimitée par des virgules. Par exemple `trident.netapp.io/cloneToNamespace: namespace2, namespace3, namespace4, .`
- Vous pouvez partager sur tous les espaces de noms à l'aide de `*`. Par exemple, `trident.netapp.io/cloneToNamespace: *`
- Vous pouvez mettre à jour la demande de volume persistant pour inclure l'annotation `cloneToNamespace` à tout moment.

2. **Administrateur du cluster** : Assurez-vous que le RBAC approprié est en place pour accorder l'autorisation au propriétaire de l'espace de noms de destination de créer le CR `TridentVolumeReference` dans l'espace de noms de destination(`namespace2`).
3. **Propriétaire de l'espace de noms de destination** : Créez un CR `TridentVolumeReference` dans l'espace de noms de destination qui fait référence à l'espace de noms source `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propriétaire de l'espace de noms de destination** : Créez un PVC (`namespace2`)(`pvc2` dans l'espace de noms de destination en utilisant le `cloneFromPVC` ou `cloneFromSnapshot`, et les `cloneFromNamespace` annotations pour désigner le PVC source.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limites

- Pour les demandes de provisionnement virtuels provisionnées à l'aide de pilotes ONTAP-nas-Economy, les clones en lecture seule ne sont pas pris en charge.

Réplication de volumes à l'aide de SnapMirror

Trident prend en charge les relations de mise en miroir entre un volume source sur un cluster et le volume de destination sur le cluster peering pour la réplication des données à des fins de reprise après incident. Vous pouvez utiliser une définition de ressource personnalisée (CRD) avec espace de noms, appelée relation miroir Trident (TMR), pour effectuer les opérations suivantes :

- Création de relations de symétrie entre les volumes (ESV)
- Supprimez les relations de symétrie entre les volumes
- Rompez les relations de symétrie
- Promotion du volume secondaire en cas d'incident (basculements)
- Transition sans perte des applications d'un cluster à un autre (en cas de basculements ou de migrations planifiés)

Conditions préalables à la réplication

Assurez-vous que les conditions préalables suivantes sont remplies avant de commencer :

Clusters ONTAP

- **Trident** : Trident version 22.10 ou ultérieure doit exister sur les clusters Kubernetes source et destination qui utilisent ONTAP en tant que back-end.
- **Licences** : les licences asynchrones de SnapMirror ONTAP utilisant le bundle protection des données doivent être activées sur les clusters ONTAP source et cible. Pour plus d'informations, reportez-vous à la section "[Présentation des licences SnapMirror dans ONTAP](#)".

À partir de ONTAP 9.10.1, toutes les licences sont livrées sous forme de fichier de licence NetApp (NLF), qui est un fichier unique qui active plusieurs fonctionnalités. Pour plus d'informations, reportez-vous à la section "[Licences incluses avec ONTAP One](#)".



Seule la protection asynchrone SnapMirror est prise en charge.

Peering

- **Cluster et SVM** : les systèmes back-end de stockage ONTAP doivent être peering. Pour plus d'informations, reportez-vous à la section "[Présentation du cluster et de SVM peering](#)".



S'assurer que les noms de SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **Trident et SVM** : les SVM distants peering doivent être disponibles pour Trident sur le cluster destination.

Pilotes pris en charge

NetApp Trident prend en charge la réplication de volumes avec la technologie NetApp SnapMirror à l'aide de

classes de stockage prises en charge par les pilotes suivants : **ontap-nas : NFS** ontap-san : iSCSI
ontap-san : FC ontap-san : NVMe/TCP (nécessite au minimum la version ONTAP 9.15.1)



La réplication de volume via SnapMirror n'est pas prise en charge sur les systèmes ASA r2. Pour plus d'informations sur les systèmes ASA r2, consultez la section ["En savoir plus sur les systèmes de stockage ASA r2"](#).

Créer une demande de volume persistant en miroir

Suivez ces étapes et utilisez les exemples CRD pour créer une relation miroir entre les volumes principal et secondaire.

Étapes

1. Effectuez les étapes suivantes sur le cluster Kubernetes principal :
 - a. Créez un objet StorageClass avec le `trident.netapp.io/replication: true` paramètre.

Exemple

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

- b. Créez une demande de volume persistant avec une classe de stockage précédemment créée.

Exemple

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

- c. Créez une demande de modification MirrorRelationship avec des informations locales.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident récupère les informations internes du volume et l'état de protection des données (DP) actuel du volume, puis remplit le champ d'état de MirrorRelationship.

- d. Obtenir le CR TridentMirrorRelationship pour obtenir le nom interne et la SVM du PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Effectuez les étapes suivantes sur le cluster Kubernetes secondaire :

- a. Créez une classe de stockage avec le paramètre `trident.netapp.io/replication: true`.

Exemple

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

- b. Créez une demande de modification MirrorRelationship avec les informations de destination et de source.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
    - localPVCName: csi-nas
      remoteVolumeHandle:
        "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident crée une relation SnapMirror avec le nom de la stratégie de relation configurée (ou par défaut pour ONTAP) et l'initialise.

- c. Créez une demande de volume persistant avec une classe de stockage précédemment créée pour agir en tant que classe secondaire (destination SnapMirror).

Exemple

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
    - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident vérifie la CRD TridentMirrorRelationship et ne crée pas le volume si la relation n'existe pas. Si la relation existe, Trident s'assurera que le nouveau FlexVol volume est placé sur un SVM peering avec le SVM distant défini dans le MirrorRelationship.

États de réplication des volumes

Une relation de miroir Trident (TMR) est une relation CRD qui représente une extrémité d'une relation de réplication entre les ESV. La TMR de destination a un état qui indique à Trident quel est l'état souhaité. La TMR de destination a les États suivants :

- **Établi** : le PVC local est le volume de destination d'une relation miroir, et il s'agit d'une nouvelle relation.
- **Promu**: Le PVC local est ReadWrite et montable, sans relation de miroir actuellement en vigueur.
- **Rétabli**: Le PVC local est le volume de destination d'une relation miroir et était également auparavant dans cette relation miroir.
 - L'état rétabli doit être utilisé si le volume de destination était déjà en relation avec le volume source car il écrase le contenu du volume de destination.
 - L'état rétabli échouera si le volume n'était pas auparavant dans une relation avec la source.

Promotion de la demande de volume persistant secondaire en cas de basculement non planifié

Effectuez l'étape suivante sur le cluster Kubernetes secondaire :

- Mettez à jour le champ `spec.state` de TridentMirrorRelationship vers `promoted`.

Promotion de la demande de volume persistant secondaire lors d'un basculement planifié

Lors d'un basculement planifié (migration), effectuez les étapes suivantes pour promouvoir la demande de volume persistant secondaire :

Étapes

1. Sur le cluster Kubernetes principal, créez un snapshot de la demande de volume persistant et attendez que le snapshot soit créé.
2. Sur le cluster Kubernetes principal, créez la CR SnapshotInfo pour obtenir des informations internes.

Exemple

```
kind: SnapshotInfo
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  snapshot-name: csi-nas-snapshot
```

3. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.state* du *TridentMirrorRelationship* CR en *promu* et *spec.promotedSnapshotHandle* en tant que nom interne du snapshot.
4. Sur le cluster Kubernetes secondaire, confirmez l'état (champ *status.state*) de *TridentMirrorRelationship* à promu.

Restaurer une relation de miroir après un basculement

Avant de restaurer une relation de symétrie, choisissez le côté que vous voulez faire comme nouveau principal.

Étapes

1. Sur le cluster Kubernetes secondaire, assurez-vous que les valeurs du champ *spec.remoteVolumeHandle* du champ *TridentMirrorRelationship* sont mises à jour.
2. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.mirror* de *TridentMirrorRelationship* sur *reestablished*.

Opérations supplémentaires

Trident prend en charge les opérations suivantes sur les volumes principal et secondaire :

Répliquer la demande de volume persistant primaire sur une nouvelle demande de volume secondaire

Assurez-vous que vous avez déjà un PVC primaire et un PVC secondaire.

Étapes

1. Supprimez les CRD *PersistentVolumeClaim* et *TridentMirrorRelationship* du cluster secondaire (destination) établi.
2. Supprimez le CRD *TridentMirrorRelationship* du cluster principal (source).
3. Créez un nouveau CRD *TridentMirrorRelationship* sur le cluster principal (source) pour le nouveau PVC secondaire (destination) que vous souhaitez établir.

Redimensionner une PVC en miroir, principale ou secondaire

La demande de volume persistant peut être redimensionnée normalement, ONTAP étendra automatiquement les flexvols de destination si la quantité de données dépasse la taille actuelle.

Supprimer la réplication d'une demande de volume persistant

Pour supprimer la réplication, effectuez l'une des opérations suivantes sur le volume secondaire actuel :

- Supprimez `MirrorRelationship` sur le PVC secondaire. Cela interrompt la relation de réplication.
- Ou, mettez à jour le champ `spec.state` à *promu*.

Suppression d'une demande de volume persistant (qui était auparavant mise en miroir)

Trident recherche les ESV répliquées et libère la relation de réplication avant toute tentative de suppression du volume.

Supprimer une TMR

La suppression d'une TMR d'un côté d'une relation symétrique entraîne la transition de la TMR restante vers l'état *promu* avant que Trident ne termine la suppression. Si la TMR sélectionnée pour la suppression est déjà à l'état *promoted*, il n'y a pas de relation miroir existante et la TMR sera supprimée et Trident promouvra la PVC locale en *ReadWrite*. Cette suppression libère les métadonnées `SnapMirror` pour le volume local dans ONTAP. Si ce volume est utilisé dans une relation miroir à l'avenir, il doit utiliser une nouvelle TMR avec un état de réplication *établi* volume lors de la création de la nouvelle relation miroir.

Mettre à jour les relations miroir lorsque ONTAP est en ligne

Les relations miroir peuvent être mises à jour à tout moment après leur établissement. Vous pouvez utiliser les `state: promoted` champs ou `state: reestablished` pour mettre à jour les relations. Lors de la promotion d'un volume de destination en volume `ReadWrite` standard, vous pouvez utiliser *`promotedSnapshotHandle`* pour spécifier un snapshot spécifique dans lequel restaurer le volume actuel.

Mettre à jour les relations en miroir lorsque ONTAP est hors ligne

Vous pouvez utiliser un CRD pour effectuer une mise à jour `SnapMirror` sans que Trident ne dispose d'une connectivité directe au cluster ONTAP. Reportez-vous à l'exemple de format de `TridentActionMirrorUpdate` suivant :

Exemple

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` Reflète l'état du CRD `TridentActionMirrorUpdate`. Il peut prendre une valeur de *succeed*, *In Progress* ou *FAILED*.

Utiliser la topologie CSI

Trident peut créer et attacher de manière sélective des volumes aux nœuds présents

dans un cluster Kubernetes en utilisant le ["Fonction de topologie CSI"](#).

Présentation

Grâce à la fonction de topologie CSI, l'accès aux volumes peut être limité à un sous-ensemble de nœuds, en fonction des régions et des zones de disponibilité. Les fournisseurs cloud permettent aujourd'hui aux administrateurs Kubernetes de frayer des nœuds basés sur une zone. Les nœuds peuvent se trouver dans différentes zones de disponibilité au sein d'une région ou entre différentes régions. Pour faciliter le provisionnement des volumes pour les charges de travail dans une architecture multi-zone, Trident utilise la topologie CSI.



En savoir plus sur la fonction de topologie CSI ["ici"](#).

Kubernetes propose deux modes de liaison de volumes :

- Avec la `VolumeBindingMode` valeur définie sur `Immediate`, Trident crée le volume sans connaissance de la topologie. La liaison de volumes et le provisionnement dynamique sont gérés au moment de la création de la demande de volume persistant. Il s'agit de la valeur par défaut `VolumeBindingMode` et convient aux clusters qui n'appliquent pas de contraintes de topologie. Les volumes persistants sont créés sans dépendance vis-à-vis des exigences de planification du pod demandeur.
- Avec `VolumeBindingMode` réglé sur `WaitForFirstConsumer`, La création et la liaison d'un volume persistant pour une demande de volume persistant sont retardées jusqu'à ce qu'un pod qui utilise la demande de volume persistant soit planifié et créé. De cette façon, les volumes sont créés pour répondre aux contraintes de planification appliquées en fonction des besoins de topologie.



Le `WaitForFirstConsumer` le mode de liaison ne nécessite pas d'étiquettes de topologie. Il peut être utilisé indépendamment de la fonction de topologie CSI.

Ce dont vous avez besoin

Pour utiliser la topologie CSI, vous devez disposer des éléments suivants :

- Un cluster Kubernetes exécutant un ["Version Kubernetes prise en charge"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Les nœuds du cluster doivent avoir des étiquettes qui permettent de prendre en compte la topologie (`topology.kubernetes.io/region`et `topology.kubernetes.io/zone`). Ces libellés **doivent être présents sur les nœuds du cluster** avant l'installation de Trident pour que Trident soit compatible avec la topologie.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/os":"linux","node-role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Étape 1 : création d'un back-end conscient de la topologie

Les systèmes back-end de stockage Trident peuvent être conçus pour provisionner de manière sélective des volumes en fonction des zones de disponibilité. Chaque back-end peut porter un bloc facultatif `supportedTopologies` représentant une liste de zones et de régions prises en charge. Pour les classes de stockage qui utilisent un tel backend, un volume ne sera créé que si une application est planifiée dans une région/zone prise en charge.

Voici un exemple de définition de back-end :

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```



`supportedTopologies` est utilisé pour fournir une liste de régions et de zones par back-end. Ces régions et ces zones représentent la liste des valeurs admissibles qui peuvent être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble des régions et zones fournies dans un back-end, Trident crée un volume sur le back-end.

Vous pouvez définir `supportedTopologies` par pool de stockage également. Voir l'exemple suivant :


```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-centrall
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-centrall
    topology.kubernetes.io/zone: us-centrall-a
  - topology.kubernetes.io/region: us-centrall
    topology.kubernetes.io/zone: us-centrall-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-centrall
        topology.kubernetes.io/zone: us-centrall-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-centrall
        topology.kubernetes.io/zone: us-centrall-b

```

Dans cet exemple, le region et zone les étiquettes correspondent à l'emplacement du pool de stockage. `topology.kubernetes.io/region` et `topology.kubernetes.io/zone` déterminer à partir de où les pools de stockage peuvent être consommés.

Étape 2 : définissez des classes de stockage qui prennent en compte la topologie

Les classes de stockage peuvent être définies en fonction des labels de topologie fournis aux nœuds du cluster, et contenir des informations de topologie. Cela déterminera les pools de stockage qui servent de candidats aux demandes de volume persistant faites et le sous-ensemble de nœuds qui peuvent utiliser les volumes provisionnés par Trident.

Voir l'exemple suivant :

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - key: topology.kubernetes.io/zone
    values:
      - us-east1-a
      - us-east1-b
  - key: topology.kubernetes.io/region
    values:
      - us-east1
parameters:
  fsType: ext4

```

Dans la définition de classe de stockage fournie ci-dessus, `volumeBindingMode` est définie sur `WaitForFirstConsumer`. Les demandes de volume persistant demandées pour cette classe de stockage ne seront pas traitées tant qu'elles ne seront pas référencées dans un pod. Et `allowedTopologies` fournit les zones et la région à utiliser. La `netapp-san-us-east1` classe de stockage crée des ESV sur le `san-backend-us-east1` back-end défini ci-dessus.

Étape 3 : création et utilisation d'une demande de volume persistant

Une fois la classe de stockage créée et mappée à un back-end, vous pouvez désormais créer des demandes de volume persistant.

Voir l'exemple `spec` ci-dessous :

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1

```

La création d'une demande de volume persistant à l'aide de ce manifeste se traduit par les éléments suivants :

```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY    ACCESS MODES    STORAGECLASS
AGE
pvc-san      Pending                                netapp-san-us-east1
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type      Reason              Age   From              Message
  ----      -
  Normal    WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding

```

Pour que Trident puisse créer un volume et le lier à la demande de volume persistant, utilisez la demande de volume persistant dans un pod. Voir l'exemple suivant :

```

apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values:
                  - us-east1-a
                  - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
    - name: voll
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: voll
          mountPath: /data/demo
      securityContext:
        allowPrivilegeEscalation: false

```

Ce podSpec demande à Kubernetes de planifier le pod sur les nœuds présents dans le us-east1 et choisissez parmi les nœuds présents dans le us-east1-a ou us-east1-b zones.

Voir le résultat suivant :

```
kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE READINESS GATES
app-pod-1     1/1     Running   0           19s   192.168.25.131  node2
<none>        <none>
kubectl get pvc -o wide
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS          AGE   VOLUMEMODE
pvc-san       Bound     pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b  300Mi
RWO           netapp-san-us-east1   48s   Filesystem
```

Mise à jour des systèmes back-end pour inclure `supportedTopologies`

Les systèmes back-end pré-existants peuvent être mis à jour pour inclure une liste de `supportedTopologies` à l'aide de `tridentctl backend update`. Cela n'affecte pas les volumes qui ont déjà été provisionnés et ne sera utilisé que pour les demandes de volume virtuel suivantes.

Trouvez plus d'informations

- ["Gestion des ressources pour les conteneurs"](#)
- ["Outil de sélection de nœud"](#)
- ["Affinité et anti-affinité"](#)
- ["Teintes et tolérances"](#)

Travailler avec des instantanés

Les copies Snapshot de volume Kubernetes de volumes persistants (PVS) permettent d'effectuer des copies instantanées de volumes. Vous pouvez créer un snapshot d'un volume créé à l'aide de Trident, importer un snapshot créé en dehors de Trident, créer un volume à partir d'un snapshot existant et restaurer des données de volume à partir de snapshots.

Présentation

La capture instantanée de volume est prise en charge par `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files`, et `google-cloud-netapp-volumes` conducteurs.

Avant de commencer

Vous devez disposer d'un contrôleur de snapshot externe et de définitions de ressources personnalisées (CRD) pour pouvoir utiliser les snapshots. Cela relève de la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshot et les CRD, reportez-vous à la [Déployer un contrôleur de snapshot de volume](#).



Ne créez pas de contrôleur de snapshot si vous créez des snapshots de volume à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshot caché intégré.

Créer un snapshot de volume

Étapes

1. Créer un `VolumeSnapshotClass`. Pour plus d'informations, reportez-vous à la section ["VolumeSnapshotClass"](#).
 - Le driver signale au conducteur Trident CSI.
 - `deletionPolicy` peut être `Delete` ou `Retain`. Lorsqu'il est réglé sur `Retain`, le snapshot physique sous-jacent sur le cluster de stockage est conservé même lorsque `VolumeSnapshot` l'objet a été supprimé.

Exemple

```
cat snap-sc.yaml
```

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

2. Créer un snapshot d'une demande de volume persistant existante.

Exemples

- Dans cet exemple, nous allons créer un snapshot d'un volume persistant existant.

```
cat snap.yaml
```

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- Cet exemple crée un objet de snapshot de volume pour une demande de volume persistant nommée `pvc1` et le nom du snapshot est défini sur `pvc1-snap`. Un instantané `VolumeSnapshot` est similaire à une demande de volume persistant et est associé à une `VolumeSnapshotContent` objet qui représente le snapshot réel.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                                AGE
pvc1-snap                          50s
```

- Vous pouvez identifier le `VolumeSnapshotContent` objet pour le `pvc1-snap` `VolumeSnapshot` en le décrivant. Le `Snapshot Content Name` identifie l'objet `VolumeSnapshotContent` qui sert ce snapshot. Le `Ready To Use` paramètre indique que l'instantané peut être utilisé pour créer une nouvelle demande de volume persistant.

```
kubectl describe volumesnapshots pvc1-snap
Name:          pvc1-snap
Namespace:     default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Créer une demande de volume persistant à partir d'un snapshot de volume

Vous pouvez utiliser `dataSource` Pour créer une demande de volume persistant à l'aide d'un `VolumeSnapshot` nommé `<pvc-name>` comme source des données. Une fois la demande de volume persistant créée, elle peut être connectée à un pod et utilisée comme n'importe quel autre PVC.



La demande de volume sera créée dans le même back-end que le volume source. Reportez-vous à la section ["Base de connaissances : la création d'une demande de volume persistant à partir d'un Snapshot de volume persistant Trident ne peut pas être créée dans un autre back-end"](#).

L'exemple suivant crée la demande de volume persistant à l'aide de `pvc1-snap` comme source de données.

```
cat pvc-from-snap.yaml
```

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

Importer un instantané de volume

Trident prend en charge la "[Processus Snapshot préprovisionné Kubernetes](#)" permettant à l'administrateur du cluster de créer un `VolumeSnapshotContent` objet et d'importer des snapshots créés en dehors de Trident.

Avant de commencer

Trident doit avoir créé ou importé le volume parent du snapshot.

Étapes

1. **Cluster admin:** Créez un `VolumeSnapshotContent` objet qui fait référence au snapshot back-end. Ceci lance le flux de travail de snapshot dans Trident.
 - Spécifiez le nom du snapshot back-end dans annotations comme `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
 - Spécifiez `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` dans `snapshotHandle`. il s'agit de la seule information fournie à Trident par le snapshotter externe dans l'`ListSnapshots`appel.



Le `<volumeSnapshotContentName>` Impossible de toujours faire correspondre le nom du snapshot back-end en raison des contraintes de dénomination CR.

Exemple

L'exemple suivant crée un `VolumeSnapshotContent` objet qui fait référence au snapshot back-end `snap-01`.


```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default

```

2. **Cluster admin:** Créez le VolumeSnapshot CR qui fait référence au VolumeSnapshotContent objet. Cette opération demande l'accès à VolumeSnapshot dans un espace de noms donné.

Exemple

L'exemple suivant crée un VolumeSnapshot CR nommée import-snap qui fait référence au VolumeSnapshotContent nommé import-snap-content.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. **Traitement interne (aucune action requise):** le snapshotter externe reconnaît le nouveau créé VolumeSnapshotContent et exécute l' `ListSnapshots` appel. Trident crée le `TridentSnapshot`.
 - Le snapshotter externe définit le VolumeSnapshotContent à `readyToUse` et le VolumeSnapshot à `true`.
 - Retour Trident `readyToUse=true`.
4. **Tout utilisateur :** Créer un PersistentVolumeClaim pour référencer le nouveau VolumeSnapshot, où `spec.dataSource` (ou `spec.dataSourceRef`) nom est le VolumeSnapshot nom.

Exemple

L'exemple suivant crée un PVC faisant référence à VolumeSnapshot nommé `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Restaurez les données de volume à l'aide de snapshots

Le répertoire des snapshots est masqué par défaut pour faciliter la compatibilité maximale des volumes provisionnés à l'aide de `ontap-nas` et `ontap-nas-economy` pilotes. Activez le `.snapshot` répertoire permettant de restaurer directement les données à partir de snapshots.

Utilisez l'interface de ligne de commandes ONTAP de restauration de snapshot de volume pour restaurer un volume à un état enregistré dans un snapshot précédent.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



Lorsque vous restaurez une copie Snapshot, la configuration de volume existante est écrasée. Les modifications apportées aux données de volume après la création de la copie Snapshot sont perdues.

Restauration de volumes sur place à partir d'un snapshot

Trident assure une restauration rapide des volumes sur place à partir d'un snapshot à l'aide du `TridentActionSnapshotRestore` système CR (TASR). Cette CR fonctionne comme une action Kubernetes impérative et ne persiste pas une fois l'opération terminée.

Trident prend en charge la restauration des instantanés sur le `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes`, et `solidfire-san` conducteurs.

Avant de commencer

Vous devez disposer d'une demande de volume liée et d'un instantané de volume disponible.

- Vérifiez que l'état de la demande de volume persistant est lié.

```
kubectl get pvc
```

- Vérifiez que le snapshot du volume est prêt à être utilisé.

```
kubectl get vs
```

Étapes

1. Créer la CR TASR. Cet exemple crée une CR pour la PVC `pvc1` et l'instantané de volume `pvc1-snapshot`.



Le CR TIR doit se trouver dans un espace de nom où le PVC et le VS existent.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Appliquez la CR pour effectuer une restauration à partir de l'instantané. Cet exemple permet de restaurer des données à partir d'un snapshot `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Résultats

Trident restaure les données à partir du snapshot. Vous pouvez vérifier l'état de la restauration des snapshots :

```
kubectl get tasr -o yaml
```

```

apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""

```



- Dans la plupart des cas, Trident ne réessaiera pas automatiquement l'opération en cas d'échec. Vous devrez effectuer à nouveau l'opération.
- Les utilisateurs Kubernetes sans accès administrateur peuvent avoir à obtenir l'autorisation de l'administrateur pour créer une CR ASR dans l'espace de noms de leur application.

Supprimez un volume persistant avec les snapshots associés

Lors de la suppression d'un volume persistant avec snapshots associés, le volume Trident correspondant est mis à jour et passe à l'état « Suppression ». Supprimez les snapshots de volume pour supprimer le volume Trident.

Déployer un contrôleur de snapshot de volume

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshot et les CRD, vous pouvez les déployer comme suit.

Étapes

1. Création de CRD de snapshot de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Créer le contrôleur de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```



Si nécessaire, ouvrir `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettre à jour namespace à votre espace de noms.

Liens connexes

- ["Snapshots de volume"](#)
- ["VolumeSnapshotClass"](#)

Travailler avec des instantanés de groupes de volumes

Instantanés de groupes de volumes Kubernetes de volumes persistants (PV) : NetApp Trident permet de créer des instantanés de plusieurs volumes (un groupe d'instantanés de volumes). Ces instantanés de groupes de volumes représentent des copies de plusieurs volumes prises simultanément.



VolumeGroupSnapshot est une fonctionnalité bêta de Kubernetes avec des API bêta. Kubernetes 1.32 est la version minimale requise pour VolumeGroupSnapshot.

Créer des instantanés de groupes de volumes

La prise en charge des instantanés de groupes de volumes est assurée par les pilotes de stockage suivants :

- `ontap-san`Pilote - uniquement pour les protocoles iSCSI et FC, pas pour le protocole NVMe/TCP.
- `ontap-san-economy` - uniquement pour le protocole iSCSI.
- `ontap-nas`



La création d'instantanés de groupes de volumes n'est pas prise en charge pour les systèmes de stockage NetApp ASA r2 ou AFX.

Avant de commencer

- Assurez-vous que votre version de Kubernetes est K8s 1.32 ou supérieure.
- Vous devez disposer d'un contrôleur de snapshot externe et de définitions de ressources personnalisées (CRD) pour pouvoir utiliser les snapshots. Cela relève de la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshot externe et les CRD, reportez-vous à [Déployer un contrôleur de snapshot de volume](#) .



Ne créez pas de contrôleur de snapshot si vous créez des snapshots de groupe de volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshot caché intégré.

- Dans le contrôleur de snapshot YAML, définissez le `CSIVolumeGroupSnapshot` Définissez la fonction « true » pour garantir que l'instantané du groupe de volumes est activé.
- Créez les classes d'instantanés de groupe de volumes requises avant de créer un instantané de groupe de volumes.
- Assurez-vous que tous les PVC/volumes se trouvent sur le même SVM pour pouvoir créer VolumeGroupSnapshot.

Étapes

- Créez un VolumeGroupSnapshotClass avant de créer un VolumeGroupSnapshot. Pour plus d'informations, reportez-vous "[Classe d'instantanés de groupe de volumes](#)"à .

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Créez des PVC avec les étiquettes requises à l'aide des classes de stockage existantes ou ajoutez ces étiquettes aux PVC existants.

1. Définissez la clé et la valeur de l'étiquette en fonction de vos besoins

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvcl-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1

```

- Créer un VolumeGroupSnapshot avec la même étiquette (`consistentGroupSnapshot: groupA`) spécifié dans le PVC.

Cet exemple crée un instantané de groupe de volumes :

```

apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA

```

Récupérer des données de volume à l'aide d'un instantané de groupe

Vous pouvez restaurer des volumes persistants individuels à l'aide des snapshots individuels créés dans le cadre du snapshot de groupe de volumes. Il n'est pas possible de restaurer le snapshot de groupe de volumes en tant qu'unité.

Utilisez l'interface de ligne de commandes ONTAP de restauration de snapshot de volume pour restaurer un volume à un état enregistré dans un snapshot précédent.

```

cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive

```



Lorsque vous restaurez une copie Snapshot, la configuration de volume existante est écrasée. Les modifications apportées aux données de volume après la création de la copie Snapshot sont perdues.

Restauration de volumes sur place à partir d'un snapshot

Trident assure une restauration rapide des volumes sur place à partir d'un snapshot à l'aide du `TridentActionSnapshotRestore` système CR (TASR). Cette CR fonctionne comme une action Kubernetes impérative et ne persiste pas une fois l'opération terminée.

Pour plus d'informations, voir ["Restauration de volumes sur place à partir d'un snapshot"](#).

Supprimer un PV avec des instantanés de groupe associés

Lors de la suppression d'un instantané de volume de groupe :

- Vous pouvez supprimer `VolumeGroupSnapshots` dans son intégralité, et non des snapshots individuels du groupe.
- Si des volumes persistants sont supprimés alors qu'un instantané existe pour ce volume persistant, Trident déplacera ce volume vers un état de « suppression » car l'instantané doit être supprimé avant que le volume puisse être supprimé en toute sécurité.
- Si un clone a été créé à l'aide d'un instantané groupé et que le groupe doit ensuite être supprimé, une opération de division sur clone commencera et le groupe ne pourra pas être supprimé tant que la division n'est pas terminée.

Déployer un contrôleur de snapshot de volume

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshot et les CRD, vous pouvez les déployer comme suit.

Étapes

1. Création de CRD de snapshot de volume.

```
cat snapshot-setup.sh
```



```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.yaml
```

2. Créer le contrôleur de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```



Si nécessaire, ouvrir `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettre à jour namespace à votre espace de noms.

Liens connexes

- ["Classe d'instantanés de groupe de volumes"](#)
- ["Snapshots de volume"](#)

Informations sur le copyright

Copyright © 2026 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

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

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

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

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

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

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

Informations sur les marques commerciales

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