



Provisionner et gérer les volumes

Trident

NetApp
October 30, 2026

Sommaire

Provisionner et gérer les volumes	1
Provisionner un volume Trident	1
Aperçu	1
Créer le PVC	1
Déplacer un volume	3
Exigences	4
Comment ça marche	4
Déplacer un volume	4
Étendre les volumes Trident	7
Étendre un volume iSCSI	7
Étendre un volume FC	11
Étendre un volume NFS	14
Comprendre les limites du sous-système RWX NVMe	17
Comprendre la limite de 64 nœuds	17
Comprendre les modèles de sous-système NVMe	17
Identifier les symptômes d'erreur	18
Résoudre les erreurs de limite du sous-système	18
Mettre à niveau Trident pour appliquer le modèle de super-sous-système	18
Évolutivité du contrôleur	18
Concepts clés et définitions	19
Comportement par défaut (mode série)	19
Prise en charge de l'évolutivité du contrôleur	19
Comment enableConcurrency fonctionne	20
Avant d'activer	20
Activer la scalabilité du contrôleur	20
Comportement de concurrence	22
Avertissements et limitations	23
Recommandations	23
Extension automatique du volume	23
Exigences	24
Limitations	24
Provisionnement de volumes avec stratégie d'extension automatique	24
Créer une politique Autogrow	25
Créer une politique Autogrow	25
États de la stratégie	26
Associer une politique à une StorageClass	26
Précédence de la politique	27
Exemples de configuration	28
Gérer les stratégies Autogrow	30
Afficher les politiques Autogrow	30
Mettre à jour une politique Autogrow	31
Supprimer une politique Autogrow	32
Surveiller l'utilisation de la politique Autogrow	32

Protocoles pris en charge	33
Backends pris en charge	33
Limitations connues	33
Foire aux questions	33
Importer des volumes Trident	39
Aperçu et considérations	39
Importer un volume	40
Exemples	42
Exemples d'économie SAN ONTAP	48
Personnaliser les noms et les étiquettes des volumes Trident	51
Avant de commencer	51
Limitations	51
Comportements clés des noms de volumes personnalisables	51
Exemples de configuration backend avec modèle de nom et labels	51
Exemples de modèles de noms	53
Points à considérer	53
Partager un volume NFS Trident entre espaces de noms	53
Caractéristiques	53
Démarrage rapide	54
Configurez les espaces de noms source et de destination	55
Supprimer un volume partagé	56
Utilisez <code>tridentctl get</code> pour interroger les volumes subordonnés	56
Limitations	57
Pour plus d'informations	57
Cloner des volumes Trident entre des espaces de noms	57
Prérequis	57
Démarrage rapide	57
Configurez les espaces de noms source et de destination	58
Limitations	59
Répliquer les volumes à l'aide de SnapMirror	60
Prérequis de réplication	60
Créer un PVC en miroir	61
États de réplication du volume	64
Promouvoir le PVC secondaire lors d'un basculement imprévu	64
Promouvoir le PVC secondaire lors d'un basculement planifié	64
Rétablir une relation miroir après un basculement	65
Opérations supplémentaires	65
Mettez à jour les relations miroir lorsque ONTAP est en ligne	66
Mettre à jour les relations miroir lorsque ONTAP est hors ligne	66
Utiliser la topologie CSI	66
Aperçu	66
Étape 1 : Créez un backend tenant compte de la topologie	68
Étape 2 : Définir les StorageClasses qui sont conscients de la topologie	69
Étape 3 : Créer et utiliser un PVC	70
Mettre à jour les backends pour inclure <code>supportedTopologies</code>	73

Pour plus d'informations	73
Travailler avec les instantanés Trident	73
Aperçu	73
Créer un instantané de volume	73
Créer un PVC à partir d'un instantané de volume	75
Importer un instantané de volume	76
Récupérer les données du volume à l'aide de snapshots	78
Restauration sur place du volume à partir d'un instantané	78
Supprimez un PV avec les instantanés associés	80
Déployer un contrôleur d'instantané de volume	80
Liens associés	81
Utilisation des instantanés de groupes de volumes Trident	81
Créer des instantanés de groupes de volumes	81
Récupérer les données du volume à l'aide d'un instantané de groupe	83
Restauration sur place du volume à partir d'un instantané	83
Supprimer un PV avec des snapshots de groupe associés	83
Déployer un contrôleur d'instantané de volume	84
Liens associés	85

Provisionner et gérer les volumes

Provisionner un volume Trident

Créez un PersistentVolumeClaim (PVC) qui utilise le StorageClass Kubernetes configuré pour demander l'accès au PV. Vous pouvez ensuite monter le PV sur un pod.

Aperçu

Un "*PersistentVolumeClaim*" (PVC) est une demande d'accès au PersistentVolume sur le cluster.

Le PVC peut être configuré pour demander un espace de stockage d'une certaine taille ou un certain mode d'accès. En utilisant la StorageClass associée, l'administrateur du cluster peut contrôler plus que la taille et le mode d'accès du PersistentVolume—comme les performances ou le niveau de service.

Après avoir créé le PVC, vous pouvez monter le volume dans un pod.

Créer le PVC

Étapes

1. Créez le PVC.

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

2. Vérifiez l'état du PVC.

```
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 en utilisant `kubectl get pod --watch`.

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

```
kubectl 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.

```
kubectl delete pod pv-pod
```

Exemples de manifestes

▼ Exemples de manifestes PersistentVolumeClaim

Ces exemples montrent les options de configuration de base des PVC.

PVC avec accès RWO

Cet exemple montre un PVC de base avec un accès RWO qui est associé à un StorageClass nommé 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 montre un PVC de base pour NVMe/TCP avec accès RWO qui est associé à un StorageClass nommé protection-gold.

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

▼ Exemples de manifestes Pod

Ces exemples montrent des configurations de base pour attacher le PVC 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
```

Consultez ["Objets Kubernetes et Trident"](#) pour plus de détails sur la manière dont les classes de stockage interagissent avec le PersistentVolumeClaim et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Déplacer un volume

À partir de Trident 26.06, un administrateur de stockage peut déplacer sans interruption

un volume iSCSI vers un autre nœud ONTAP et un autre agrégat au sein du même cluster ONTAP, tandis que la charge de travail qui l'utilise continue de fonctionner. Vous demandez le déplacement de manière déclarative avec une `TridentVolumeMove` ressource personnalisée ou l' `tridentctl` équivalent, et Trident coordonne ONTAP et l'hôte afin que les E/S ne soient jamais interrompues.

Exigences

Le déplacement de volume s'applique au `ontap-san` driver uniquement pour les volumes iSCSI. Les protocoles NVMe et NAS ne sont pas pris en charge.

Avant de déplacer un volume, assurez-vous que les conditions suivantes sont remplies :

- Le backend utilise le ``ontap-san`` pilote avec l'API REST ONTAP.
- Le cluster ONTAP comporte au moins deux nœuds.
- Vous possédez l'un des ensembles d'identifiants suivants :
 - ONTAP 9.19.1 ou version ultérieure si vous utilisez le rôle ONTAP personnalisé dédié.
 - Identifiants d'administrateur de cluster sur les versions ONTAP antérieures.

Comment ça marche

Trident déplace le volume (LUN) entre deux emplacements au sein du même cluster ONTAP. Avant le déplacement des données, Trident ajoute de nouveaux chemins iSCSI vers la cible, en plus des chemins existants, afin que le périphérique multipath hôte ne perde jamais l'accès. Une fois le déplacement terminé par ONTAP, Trident supprime les chemins source obsolètes.

Trident gère le déplacement à travers une série d'états ordonnés visibles dans la `TridentVolumeMove` ressource, ce qui vous permet de suivre la progression du déplacement.

Déplacer un volume

Identifiez le volume et les emplacements

1. Trouvez le nom du PV à déplacer. Le nom du PV correspond également au nom de la ressource à déplacer.
2. À partir du cluster ONTAP, notez le nœud et l'agrégat actuels (source) pour ce volume, puis choisissez le nœud et l'agrégat de destination (cible).

Démarrez le déplacement

Vous pouvez utiliser l'une ou l'autre méthode. Les deux créent la même `TridentVolumeMove` ressource, dont le nom est celui du PV.

Utilisation de `tridentctl`

```
tridentctl create volume-move \  
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \  
  --source-node ontap-node-01 \  
  --target-node  ontap-node-02
```

```
--source-pool aggr1_node01 \  
--target-node ontap-node-02 \  
--target-pool aggr1_node02 \  
-n trident
```

tvm est un alias pour volume-move. Ajoutez `--delete-after-success 10m` pour supprimer automatiquement la ressource après la réussite du déplacement.

Utilisation d'une ressource TridentVolumeMove

```
apiVersion: trident.netapp.io/v1  
kind: TridentVolumeMove  
metadata:  
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671    # the PV to move  
  namespace: trident  
spec:  
  sourceNode: ontap-node-01  
  sourcePool: aggr1_node01  
  targetNode: ontap-node-02  
  targetPool: aggr1_node02  
  # deleteAfterSuccess: 10m    # optional
```

Appliquez la ressource :

```
kubectl apply -f tridentvolumemove.yaml
```

Champs de spécification

sourceNode/ targetNode

Noms des nœuds ONTAP (et non des UUID).

sourcePool/ targetPool

Noms d'agrégats ONTAP.

deleteAfterSuccess

Durée facultative, par exemple 30s ou 10m. Définissez-la sur 0 pour supprimer immédiatement la ressource, ou omettez-la pour conserver la ressource.

Surveillez le déplacement

Suivez la progression en utilisant le `tvm` nom court ou `tridentvolumemove` :

```
kubectl get tridentvolumemove -n trident
```

Pour consulter l'état actuel et la progression par nœud, décrivez la ressource :

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

Le volume peut avoir des attachements simultanés pendant l'état `Moving`, car ONTAP peut prendre un certain temps pour effectuer le déplacement d'un volume important. Pour garantir un déplacement sans interruption, Trident assure la coordination entre le contrôleur Trident et le nœud Trident. Chaque état exécute les étapes suivantes dans l'ordre, se terminant soit par `Succeeded` soit par `Failed`.

État	Ce qui se passe
Pending	Trident accepte et valide la demande de déplacement et recueille les informations nécessaires avant que toute modification du stockage ne commence.
ControllerStaging	Trident prépare l'emplacement cible. Il associe les interfaces de stockage du nœud cible au LUN et met à jour Kubernetes afin qu'un redémarrage ultérieur du nœud le reconnecte à l'emplacement correct.
NodeStaging	Sur chaque hôte sur lequel le volume est monté, Trident ajoute les nouveaux chemins de stockage du nœud cible aux côtés des chemins existants, afin que les E/S du pod se poursuivent sur les deux ensembles de chemins.
Moving	ONTAP copie et transfère les données du volume vers l'agrégat cible. Cette phase est la plus longue et ONTAP la gère entièrement tandis que Trident interroge jusqu'à la fin de l'opération. De nouvelles publications sont autorisées pendant cette phase.
ControllerUnstaging	Le déplacement des données est terminé. Trident met à jour Kubernetes pour supprimer les chemins sources de l'état souhaité, puis dissocie les interfaces de stockage du nœud source du LUN.
NodeUnstaging	Sur chaque hôte, Trident supprime les chemins sources obsolètes. L'hôte ne conserve alors que les chemins cibles, et le périphérique multipath continue de servir les E/S sans interruption.
Succeeded	Le volume est entièrement transféré et en ligne sur la cible. Trident conserve la <code>TridentVolumeMove</code> ressource, ou la supprime automatiquement si vous définissez <code>deleteAfterSuccess</code> .
Failed	Une erreur irrécupérable a interrompu le déplacement ou empêché son démarrage. Trident réinitialise l'état du volume pour permettre la reprise des opérations normales et conserve la ressource avec un message d'erreur. Consultez le message, corrigez la cause racine, supprimez la ressource et réessayez.

La `.status.attachments` liste indique la progression de chaque nœud connecté dans son champ `state`:

État d'attachement	Signification
Pending	Nœud identifié comme attaché ; le travail n'a pas encore commencé.
Bridged	De nouveaux chemins cibles sont ajoutés en parallèle des chemins existants.
Migrated	Le trafic est désormais acheminé via les nouveaux chemins.

État d'attachement	Signification
Cleaned	Les chemins sources obsolètes ont été supprimés du nœud.
Detached	Le nœud a été détaché en cours de déplacement et exclu de toute considération ultérieure pendant que Trident évalue si la connexion est prête à passer à l'étape suivante.
Failed	Erreur terminale sur ce nœud. Consultez le message.

Nettoyez la ressource de déplacement

Si vous définissez `deleteAfterSuccess`, Trident supprime automatiquement la ressource. Sinon, vous pouvez la supprimer manuellement. Avant de demander un autre déplacement sur le même volume, assurez-vous qu'aucune ressource `TridentVolumeMove` n'existe pour ce volume, ni dans un état `Succeeded` ni dans un état `Failed`.



Trident ne supprime jamais automatiquement les déplacements ayant échoué. Trident conserve une `Failed` ressource à titre d'audit. Après avoir corrigé la cause première, supprimez la ressource manuellement avant de réessayer.

Étendre les volumes Trident

Trident permet aux utilisateurs de Kubernetes d'étendre leurs volumes après leur création. Consultez les informations relatives aux configurations requises pour étendre les volumes iSCSI, NFS, SMB, NVMe/TCP et FC.

Étendre un volume iSCSI

Vous pouvez étendre un volume persistant iSCSI (PV) à l'aide du provisionneur CSI.



L'extension de volume iSCSI est prise en charge par les `ontap-san`, `ontap-san-economy`, `solidfire-san` pilotes et nécessite Kubernetes 1.16 et versions ultérieures.

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Modifiez la définition `StorageClass` pour définir le champ `allowVolumeExpansion` sur `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 un StorageClass déjà existant, modifiez-le pour inclure le `allowVolumeExpansion` paramètre.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

Modifiez la définition du PVC 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 (PV) et l'associe à cette Persistent Volume Claim (PVC).

```
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 : Définir un pod auquel se fixe le PVC

Fixez le PV à un pod pour qu'il soit redimensionné. Il existe deux scénarios lors du redimensionnement d'un PV iSCSI :

- Si le PV est attaché à un pod, Trident étend le volume sur le backend de stockage, rescanne le périphérique et redimensionne le système de fichiers.

- Lors d'une tentative de redimensionnement d'un PV non attaché, Trident étend le volume sur le stockage backend. Après que le PVC est lié à un pod, Trident rescanner le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille du PVC après que l'opération d'extension a été effectuée avec succès.

Dans cet exemple, un pod est créé qui 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 : Développer le PV

Pour redimensionner le PV créé de 1Gi à 2Gi, modifiez la définition du PVC et mettez à jour le `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'expansion

Vous pouvez vérifier que l'extension a fonctionné correctement en contrôlant la taille du PVC, du 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  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Étendre un volume FC

Vous pouvez étendre un volume persistant (PV) FC en utilisant le provisionneur CSI.



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

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Modifiez la définition StorageClass pour définir le champ `allowVolumeExpansion` sur `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 un StorageClass déjà existant, modifiez-le pour inclure le `allowVolumeExpansion` paramètre.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

Modifiez la définition du PVC 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 (PV) et l'associe à cette Persistent Volume Claim (PVC).

```
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 : Définir un pod auquel se fixe le PVC

Fixez le PV à un pod pour qu'il soit redimensionné. Il existe deux scénarios lors du redimensionnement d'un PV FC :

- Si le PV est attaché à un pod, Trident étend le volume sur le backend de stockage, rescanne le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un PV non attaché, Trident étend le volume sur le stockage backend. Après que le PVC est lié à un pod, Trident rescanne le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille du PVC après que l'opération d'extension a été effectuée avec succès.

Dans cet exemple, un pod est créé qui 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 : Développer le PV

Pour redimensionner le PV créé de 1Gi à 2Gi, modifiez la définition du PVC et mettez à jour le `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'expansion

Vous pouvez vérifier que l'extension a fonctionné correctement en contrôlant la taille du PVC, du 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      |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

Étendre 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 le StorageClass pour prendre en charge l'extension de volume

Pour redimensionner un volume persistant NFS, l'administrateur doit d'abord configurer la classe de stockage pour autoriser l'extension de volume en définissant le `allowVolumeExpansion` champ sur `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 autoriser l'extension de volume.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

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

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

Trident devrait créer un PV NFS de 20 MiB 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   STORAGECLASS   REASON   AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi        RWO             ontapnas       2m42s
Delete            Bound       default/ontapnas20mb
```

Étape 3 : Développer le PV

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

Vous pouvez vérifier que le redimensionnement a fonctionné correctement en contrôlant la taille du PVC, du PV et du volume Trident :

```

kubectl get pvc ontapnas20mb
NAME          STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS   AGE
ontapnas20mb  Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO          ontapnas      4m44s

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  1Gi        RWO
Delete          Bound    default/ontapnas20mb  ontapnas
5m35s

```

```
tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  |      | STATE  | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapas      |
+-----+-----+-----+-----+
| file          | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Comprendre les limites du sous-système RWX NVMe

ReadWriteMany (RWX) volumes utilisant le protocole NVMe ont une limite de scalabilité de 64 nœuds par volume. Ce qui suit inclut les limitations, explique l'architecture du sous-système NVMe concernée et décrit les étapes de résolution requises.

Comprendre la limite de 64 nœuds

Si vous prévoyez d'utiliser des volumes ReadWriteMany (RWX) avec le protocole NVMe, un seul volume NVMe RWX ne peut pas être monté par plus de 64 nœuds dans un cluster Kubernetes.

Ne planifiez pas de charges de travail qui montent le même RWX NVMe PersistentVolumeClaim sur plus de 64 nœuds.

Cette limitation s'applique uniquement aux volumes RWX qui utilisent le protocole NVMe.

Comprendre les modèles de sous-système NVMe

Modèle de sous-système par volume (Trident releases earlier than 26.02)

Dans les versions de Trident antérieures à 26.02, les volumes NVMe RWX sont provisionnés à l'aide d'un modèle de sous-système par volume. Chaque volume NVMe RWX est associé à son propre sous-système NVMe dédié sur ONTAP.

Ce modèle est simple, mais il a une limite d'évolutivité inférieure. Dans les grands clusters Kubernetes, les limites des contrôleurs de sous-système sont rapidement atteintes car chaque volume RWX consomme un sous-système dédié.

Modèle de super-sous-système (introduit dans Trident 26.02)

À partir de Trident 26.02, les volumes RWX NVMe utilisent un modèle de super-sous-système partagé. Plusieurs volumes RWX NVMe partagent le même sous-système NVMe.

Chaque super-sous-système prend en charge jusqu'à 1024 espaces de noms (volumes). Ce modèle améliore considérablement l'évolutivité des charges de travail RWX et réduit la probabilité d'atteindre les limites du sous-système ONTAP.

Chaque volume RWX NVMe prend en charge jusqu'à 64 nœuds.

Identifier les symptômes d'erreur

Si vous créez ou attachez des volumes RWX NVMe à grande échelle, vous pourriez observer des erreurs similaires aux suivantes :

```
Maximum number of controllers reached. No more controllers can be created.
```

Cette erreur indique que la limite du contrôleur du sous-système ONTAP NVMe a été atteinte.

Résoudre les erreurs de limite du sous-système

Pour dépasser les limitations des sous-systèmes par volume et profiter du modèle de super-sous-système, mettez à niveau vers Trident 26.02 ou une version ultérieure.

Mettre à niveau Trident pour appliquer le modèle de super-sous-système

Pour appliquer le modèle de super-sous-système aux volumes RWX NVMe :

1. Mettez à niveau Trident vers la version 26.02 ou ultérieure.
2. Réduisez à zéro tous les pods qui utilisent des volumes RWX NVMe.
3. Vérifiez qu'aucune charge de travail n'utilise activement les volumes RWX NVMe.
4. Augmentez à nouveau le nombre de pods.

Cette séquence de redémarrage garantit que les volumes RWX NVMe sont attachés à l'aide du modèle super-sous-système.

- Cette limitation s'applique uniquement aux volumes RWX qui utilisent le protocole NVMe.
- La limite de 64 nœuds s'applique par volume RWX NVMe.
- Les autres modes d'accès et les autres protocoles ne sont pas affectés.

Évolutivité du contrôleur

Trident améliore la scalabilité du contrôleur grâce à une concurrence accrue sur plusieurs pilotes de stockage. Lorsque vous activez la scalabilité du contrôleur, le contrôleur Trident traite les opérations de stockage en parallèle au lieu de les sérialiser, ce qui augmente le débit dans les environnements Kubernetes comportant de nombreuses opérations simultanées.

Avant le déploiement, déterminez quels pilotes Trident prennent en charge la mise à l'échelle du contrôleur à la disponibilité générale et quels pilotes sont disponibles en version préliminaire technique dans Trident 26.06. Cela vous aide à prendre des décisions éclairées concernant le déploiement et à gérer les risques. La mise à l'échelle du contrôleur est désactivée par défaut.

Concepts clés et définitions

Évolutivité du contrôleur

La scalabilité du contrôleur fait référence à la capacité du contrôleur Trident à traiter plusieurs opérations de stockage en parallèle, au lieu de les sérialiser derrière un seul verrou. Ces opérations comprennent la création, la suppression et le redimensionnement de volumes ; la création et la suppression d'instantanés ; la publication et la dépublication de volumes ; et la gestion du backend.

Lorsque vous activez la scalabilité du contrôleur, les opérations sur différents volumes et backends s'exécutent simultanément. Cela augmente le débit et réduit le temps d'exécution global dans les environnements comportant un grand nombre d'opérations PersistentVolumeClaim et VolumeSnapshot simultanées.

Comportement par défaut (mode série)

Par défaut, le contrôleur Trident traite les opérations une à une. Chaque requête de création, de suppression, de redimensionnement ou de capture instantanée est terminée avant que Trident ne lance la suivante. Le mode série est le mode par défaut pris en charge pour toutes les installations et ne nécessite aucune configuration.

Le mode séquentiel est suffisant pour la plupart des charges de travail. N'activez la mise à l'échelle du contrôleur que lorsque le volume d'opérations crée un engorgement en mode séquentiel.

Prise en charge de l'évolutivité du contrôleur

Trident prend en charge l'évolutivité du contrôleur à différents niveaux de maturité pour différents pilotes de stockage.

Disponibilité générale

Les pilotes suivants prennent en charge la mise à l'échelle du contrôleur à la disponibilité générale dans Trident 26.06 :

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san-economy`
- `google-cloud-netapp-volumes`
- `azure-netapp-files`
- `solidfire-san`



Les `google-cloud-netapp-volumes` et `google-cloud-netapp-volumes-san` pilotes sont différents. Trident prend uniquement en charge `google-cloud-netapp-volumes`. N'utilisez pas `google-cloud-netapp-volumes-san` dans les configurations ou exemples de backend.

Aperçu technique

Le pilote suivant prend en charge la mise à l'échelle du contrôleur en tant que fonctionnalité en avant-première

technique dans Trident 26.06 :

- `asa-r2` (SAN et NVMe)

Ce pilote présente les limitations suivantes :

- La gestion de la concurrence des contrôleurs est disponible uniquement à des fins d'évaluation et de test.
- Le comportement peut changer d'une version à l'autre.
- NetApp ne recommande pas l'utilisation en environnement de production.

Comment `enableConcurrency` fonctionne

Lorsque vous définissez `enableConcurrency` sur `true`, Trident applique un traitement simultané à tous les backends gérés par le contrôleur. Ce paramètre s'applique à tous les backends en même temps. Vous ne pouvez pas l'activer pour des backends ou des pilotes individuels.

Chaque backend configuré doit utiliser un pilote figurant dans la liste de disponibilité générale ou de préversion technique. Si un backend utilise un pilote non pris en charge, Trident ne démarre pas et Trident n'ajoute pas de backend utilisant un pilote non pris en charge.

Pour rétablir le démarrage, supprimez ou reconfigurez tout backend utilisant un pilote non pris en charge ou désactivez la scalabilité du contrôleur.

Avant d'activer



Avant d'activer la mise à l'échelle du contrôleur, vérifiez que chaque backend configuré utilise un pilote figurant dans la liste de disponibilité générale ou de préversion technique. Si un backend utilise un pilote non pris en charge, Trident ne démarre pas après l'activation de cette fonctionnalité.

Utilisez le tableau suivant pour décider d'activer ou non la mise à l'échelle du contrôleur.

Si votre environnement	Ensuite
Utilise uniquement les pilotes pris en charge et rencontre un retard dans les opérations du contrôleur	Activez la scalabilité du contrôleur.
Utilise n'importe quel pilote non pris en charge	Conservez le mode série par défaut. N'activez pas la mise à l'échelle du contrôleur.
Gère un faible volume d'opérations sans arriérés	Conservez le mode série par défaut.

Activer la scalabilité du contrôleur

L'option de configuration `enableConcurrency` contrôle la scalabilité du contrôleur. Vous devez activer explicitement cette option lors de l'installation de Trident ou lors de la mise à jour d'un déploiement existant.

Déploiement de l'opérateur Trident

Pour activer la scalabilité du contrôleur avec l'opérateur Trident, définissez `enableConcurrency` sur `true` dans la `TridentOrchestrator` ressource personnalisée (CR).

Nouvelle installation

Créez ou modifiez le `TridentOrchestrator` CR et définissez `enableConcurrency` sur `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Appliquez le CR :

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Installation existante

Corrigez le `TridentOrchestrator` CR existant pour activer l'évolutivité du contrôleur :

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Vérifiez que Trident a appliqué le paramètre :

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Déploiement Helm

Pour activer la scalabilité du contrôleur avec Helm, définissez la valeur `enableConcurrency` sur `true`.

Nouvelle installation

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Installation existante

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Vous pouvez également définir `enableConcurrency` sur `true` dans un fichier `values.yaml` personnalisé :

```
# values.yaml
enableConcurrency: true
```

Installez ou mettez à jour ensuite avec le fichier de valeurs :

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

déploiement tridentctl

Pour activer la scalabilité du contrôleur avec `tridentctl`, transmettez le drapeau `--enable-concurrency` lors de l'installation.

Nouvelle installation

```
tridentctl install -n trident --enable-concurrency
```

Installation existante

Pour activer la mise à l'échelle du contrôleur sur un déploiement `tridentctl` existant, désinstallez Trident et réinstallez-le avec l'indicateur :

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Vérifiez que la mise à l'échelle du contrôleur est activée

Après avoir activé la mise à l'échelle du contrôleur, vérifiez que le contrôleur Trident s'exécute avec la concurrence activée. Consultez les journaux du pod du contrôleur :

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Le résultat comprend une entrée de journal confirmant que la concurrence est activée.

Comportement de concurrence

Lorsque la mise à l'échelle du contrôleur est activée, le contrôleur Trident adopte le comportement suivant :

- Trident remplace le verrou global unique par un verrouillage précis, par ressource.
- Trident sérialise les opérations qui modifient la même ressource afin de maintenir la cohérence des données.

- Les opérations qui ne font que lire une ressource s'exécutent simultanément avec d'autres opérations de lecture sur cette ressource.
- Trident limite le nombre de requêtes API ONTAP simultanées à 20 par LIF de gestion afin d'éviter la surcharge des systèmes de stockage backend.
- Si plusieurs backends partagent le même LIF de gestion, ils partagent cette limite de 20 requêtes.

Avertissements et limitations

Les considérations suivantes s'appliquent à l'évolutivité du contrôleur dans Trident 26.06 :

- La mise à l'échelle du contrôleur ne prend en charge que les pilotes figurant dans les listes de disponibilité générale et d'aperçu technique. Pour plus de détails, consultez [Avant d'activer](#).
- Le contrôleur Trident gère la concurrence en interne. Cette version ne propose aucune limite de concurrence configurable par l'utilisateur.
- Le débit global dépend du pilote de stockage utilisé, de la réactivité du backend et des performances du serveur API Kubernetes.
- Une forte concurrence peut augmenter la charge sur les systèmes de stockage backend.
- Le comportement de mise à l'échelle du contrôleur n'est pas identique pour tous les pilotes.
- Le pilote en préversion technique peut présenter des performances incohérentes en cas de charge élevée et son comportement peut changer d'une version à l'autre.
- Le débogage des opérations concurrentes peut s'avérer plus complexe en raison de l'exécution parallèle. Les métriques et les journaux peuvent afficher les résultats des opérations entrelacées.

Recommandations

Avant d'activer la mise à l'échelle du contrôleur, complétez les étapes suivantes :

1. Confirmez que chaque backend configuré utilise un pilote figurant dans la liste de disponibilité générale ou de préversion technique.
2. Testez la modification sur un cluster hors production avant de l'appliquer en production.
3. Vérifiez que la mise à l'échelle du contrôleur est activée après avoir appliqué la modification.

Appliquez les recommandations générales suivantes lorsque vous utilisez la mise à l'échelle du contrôleur :

- Utilisez des pilotes à disponibilité générale pour les environnements de production nécessitant une évolutivité élevée.
- Évaluez le pilote en préversion technique uniquement dans des environnements hors production.
- Surveillez les performances du backend et du contrôleur lorsque vous opérez à grande échelle.
- Ne présumez pas de l'ordre des opérations dans les scripts d'automatisation.

Extension automatique du volume

L'extension automatique des volumes permet aux Persistent Volumes provisionnés par Trident d'augmenter automatiquement leur capacité lorsque celle-ci atteint un seuil défini. Cette fonctionnalité réduit les coûts d'exploitation et contribue à prévenir les interruptions d'application dues à la saturation de la capacité. L'extension automatique du volume est

mise en œuvre à l'aide des Autogrow Policies. Une Autogrow Policy définit :

- Le seuil d'utilisation qui déclenche l'expansion
- La quantité dont le volume augmente
- La taille maximale que le volume peut atteindre



Les volumes augmentent automatiquement lorsque le seuil d'utilisation défini est dépassé. Les volumes ne sont jamais réduits automatiquement.

Exigences

Avant de configurer l'extension automatique du volume, assurez-vous que les exigences suivantes sont remplies :

- Trident 26.02 ou version ultérieure
- Autorisations de contrôle d'accès basées sur les rôles pour créer `TridentAutogrowPolicy` des ressources personnalisées
- `StorageClasses` configuré avec `allowVolumeExpansion: true`
- Protocoles ONTAP pris en charge :
 - Système de fichiers réseau (NFS)
 - Internet Small Computer Systems Interface (iSCSI)
 - Non-Volatile Memory Express (NVMe)
 - Protocole Fibre Channel (FCP)

Limitations

- Les volumes de blocs bruts ONTAP Non-Volatile Memory Express antérieurs à ONTAP 9.16.1 ne prennent pas en charge l'extension automatique.
- Pour les volumes de réseau de stockage, si `growthAmount` est inférieur ou égal à 50 mébioctets, Trident augmente automatiquement la valeur à 51 mébioctets avant le redimensionnement, à condition que la taille résultante ne dépasse pas `maxSize`.
- Dans les environnements brownfield, l'extension automatique peut ne pas fonctionner pour certains volumes existants en raison du comportement de migration de la publication des volumes.
- Lorsqu'un volume atteint `maxSize`, aucune expansion supplémentaire ne se produit.
- Protocoles pris en charge pour l'extension automatique du volume :
 - Système de fichiers réseau (NFS)
 - Internet Small Computer Systems Interface (iSCSI)
 - Non-Volatile Memory Express (NVMe)
 - Protocole Fibre Channel (FCP)

Provisionnement de volumes avec stratégie d'extension automatique

La politique Autogrow peut être configurée à deux niveaux :

- Niveau de classe de stockage : définit la valeur par défaut pour tous les volumes (à l'aide d'une annotation)
- Niveau PVC : Remplace la valeur par défaut de la classe de stockage (en utilisant une annotation)

Créer une politique Autogrow

Les politiques Autogrow permettent l'expansion automatique des volumes lorsque ceux-ci atteignent un seuil de capacité défini.

Assurez-vous d'avoir :

- Trident 26.02 ou version ultérieure installé
- Autorisations de contrôle d'accès basées sur les rôles pour créer `TridentAutogrowPolicy` des ressources
- Compréhension des exigences de croissance de la charge de travail

Une politique Autogrow définit comment les volumes s'étendent automatiquement lorsqu'ils atteignent un seuil de capacité défini.

Vous pouvez créer des stratégies Autogrow à tout moment de votre flux de travail :

- Avant que les StorageClasses et les volumes ne soient créés
- Après que les StorageClasses existent
- Une fois les volumes provisionnés

Cette flexibilité vous permet d'introduire une extension automatique sans recréer les ressources existantes.

Spécifications de la politique Autogrow

Les politiques d'autogrow sont des ressources personnalisées Kubernetes définies comme suit :

Champ	Description	Format	Obligatoire	Exemple	Défaut
nom	Identifiant unique de la politique	Chaîne	Oui	production-db-policy	Aucune
usedThreshold	Pourcentage de capacité qui déclenche l'expansion	Chaîne de pourcentage	Oui	"80%"	Aucune
growthAmount	Montant à augmenter lorsque le seuil est atteint	Pourcentage ou taille	Non	« 10% » ou « 5Gi »	"10%"
maxSize	Limite de taille maximale du volume	quantité Kubernetes	Non	"500Gi"	Illimité

Créer une politique Autogrow

Étapes

1. Créez un fichier YAML qui définit votre Autogrow Policy :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Appliquez la politique à votre cluster :

```
kubectl apply -f autogrow-policy.yaml
```

3. Vérifiez que la policy a été créée :

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Résultat attendu

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
standard-autogrow	80%	10%	Success

États de la stratégie

Après avoir créé une politique, Trident valide la spécification et lui attribue l'un des états suivants :

État	Description	Action requise
Succès	La stratégie est validée et prête à l'emploi.	Aucun.
Échec	Des erreurs de validation ont été détectées.	Examinez et corrigez la spécification.
Suppression	Suppression en cours.	Attendez la fin.

Associer une politique à une StorageClass

Vous pouvez associer une stratégie d'autogrow à un StorageClass en utilisant l'annotation `trident.netapp.io/autogrowPolicy`. Tous les volumes provisionnés à partir de ce StorageClass héritent de la stratégie.



Le StorageClass doit avoir `allowVolumeExpansion: true`.

Étapes

1. Créez ou modifiez un StorageClass avec l'annotation de stratégie d'extension automatique :

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true

```

2. Appliquez le StorageClass :

```
kubectl apply -f storageclass.yaml
```

3. Vérifiez l'annotation :

```

kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'

```

Résultat attendu

```
production-db-policy
```

Précédence de la politique

Lorsque des annotations de stratégie d'extension automatique sont définies à la fois sur un StorageClass et un PVC, Trident applique les règles de priorité suivantes :

1. **L'annotation PVC est prioritaire.** Si un PVC définit `trident.netapp.io/autogrowPolicy`, cette valeur est toujours utilisée.
2. **StorageClass annotation s'applique uniquement lorsque le PVC ne comporte aucune annotation.**
3. **Si aucune des deux ne comporte l'annotation,** aucune Autogrow Policy n'est appliquée.

Annotation StorageClass	Annotation PVC	Comportement efficace
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	Non défini	Utilisations standard-agp.
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: logs-policy</code>	Utilise logs-policy (les PVC remplacent StorageClass).

Annotation StorageClass	Annotation PVC	Comportement efficace
trident.netapp.io/autogrowPolicy: standard-agp	trident.netapp.io/autogrowPolicy: "none"	Aucune politique d'autogrow (PVC désactive l'autogrow).
Non défini	trident.netapp.io/autogrowPolicy: dev-policy	Utilisations dev-policy.
Non défini	Non défini	Aucune politique d'autocroissance.

Exemples de configuration

Les exemples suivants montrent les configurations courantes d'Autogrow Policy pour différents cas d'usage.

Politique conservatrice pour les bases de données de production

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Stockage de journaux avec incréments de croissance fixes

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Politique minimale avec valeurs par défaut

Lorsque vous omettez growthAmount et maxSize, Trident utilise les valeurs par défaut (10% (croissance, taille illimitée) :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
```

```
name: logs-policy
spec:
  usedThreshold: "85%"
```

Stratégie avec une valeur personnalisée maxSize et la valeur par défaut growthAmount

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Croissance agressive avec maxSize illimité

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Politique avec des pourcentages fractionnaires

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```



Les pourcentages fractionnaires sont pris en charge. Si vous spécifiez plus de trois décimales, Trident arrondit la valeur à trois décimales.

NAS StorageClass avec Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```

metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true

```

SAN StorageClass avec Autogrow

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true

```

Gérer les stratégies Autogrow

Une fois les stratégies d'extension automatique créées, vous pouvez les consulter, les modifier et les supprimer. Vous pouvez également surveiller les volumes qui utilisent une stratégie donnée.

Afficher les politiques Autogrow

Lister toutes les politiques

Utilisez `kubectl` pour lister toutes les stratégies d'extension automatique de votre cluster :

```
kubectl get tridentautogrowpolicy
```

Vous pouvez également utiliser `tridentctl` :

```
tridentctl get autogrowpolicy
```

Afficher les détails de la politique

Pour consulter la spécification complète et l'état d'une politique :

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Pour afficher une règle avec ses volumes associés au format YAML :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Mettre à jour une politique Autogrow

Vous pouvez modifier une politique existante pour changer son seuil, son taux de croissance ou sa taille maximale. Les modifications prennent effet immédiatement pour tous les volumes qui utilisent la politique.



Les modifications affectent tous les volumes utilisant cette stratégie. Dans la mesure du possible, testez les modifications d'abord dans un environnement hors production.

Étapes

1. Modifier la politique :

```
kubectl edit tridentautogrowpolicy production-db-policy
```

2. Modifiez les `spec` champs selon vos besoins :

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount:  "20%"      # Changed from 10%
  maxSize:       "1Ti"      # Changed from 500Gi
```

3. Enregistrez et quittez. Les modifications prennent effet immédiatement.

Considérations relatives à la mise à jour

- **Effet immédiat** : Tous les volumes qui utilisent cette politique adoptent les nouveaux paramètres lors de la prochaine évaluation de croissance.
- **Aucun redémarrage du volume requis** : Les modifications s'appliquent à la prochaine opération de croissance.
- **Testez d'abord** : Validez les modifications dans un environnement non production lorsque cela est possible.
- **Communiquer les changements** : Informez les équipes lorsque vous modifiez des politiques partagées.

Supprimer une politique Autogrow

Les politiques d'extension automatique utilisent la protection du finaliseur pour empêcher toute suppression accidentelle pendant que les volumes les utilisent activement.

Étapes

1. Supprimez la règle :

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Si des volumes utilisent encore cette stratégie, la suppression entre dans un état `Deleting`. Vérifiez quels volumes sont concernés :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Supprimez la stratégie de chaque volume concerné. Choisissez l'une des options suivantes :

- **Option A : Désactiver explicitement la croissance automatique** en définissant l'annotation sur "none":

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

- **Option B: Supprimer complètement l'annotation :**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Comportement de suppression

Scénario	Comportement
Aucun volume n'utilise la politique	La règle est supprimée immédiatement.
Les volumes utilisent la politique	La stratégie entre dans <code>Deleting</code> état. Un finaliseur bloque la finalisation jusqu'à ce que tous les volumes soient supprimés.
Tous les volumes sont supprimés de la stratégie	Les finalizers sont supprimés et la policy est supprimée.

Surveiller l'utilisation de la politique Autogrow

Vérifiez les volumes à l'aide d'une règle

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Découvrez quelle politique un volume utilise

```
kubectl get pvc database-pvc -o  
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Surveiller les événements de stratégie

```
kubectl get events --field-selector  
involvedObject.kind=TridentAutogrowPolicy
```

Protocoles pris en charge

Autogrow prend en charge les protocoles de stockage suivants :

- NFS
- iSCSI
- FCP
- NVMe



Pour les volumes SAN, si la valeur configurée `growthAmount` est de 50 Mio ou moins, Trident augmente automatiquement la marge de croissance à 51 Mo pour l'opération de redimensionnement, tant que la taille résultante ne dépasse pas `maxSize`.

Backends pris en charge

Autogrow prend en charge les backends ONTAP. Autogrow prend également en charge Google Cloud NetApp Volumes (GCNV) pour les volumes NAS et SAN.

Limitations connues

- **Volumes de blocs bruts NVMe ONTAP** : Les volumes créés avec des versions d'ONTAP antérieures à 9.16.1 ne prennent pas en charge l'autogrow.
- **Volumes existants (déploiements brownfield)** : L'extension automatique peut ne pas fonctionner pour les volumes existants, même si une stratégie d'extension automatique valide est appliquée. Cette limitation est due à une migration en cours des publications de volumes. Pour confirmer que la migration est terminée, vérifiez les journaux du contrôleur Trident pour les messages "Migration completed".

Foire aux questions

À quel moment Trident évalue le seuil ?

Trident surveille en permanence l'utilisation du volume. Lorsque la capacité utilisée dépasse le `usedThreshold`, Trident crée une demande de redimensionnement interne et augmente le volume de la

valeur configurée `growthAmount`.

Par exemple, cette politique déclenche une extension à 80% de la capacité et augmente le volume de 10% à chaque fois, jusqu'à un maximum de 500 GiB :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Puis-je appliquer une stratégie après que les volumes ont déjà été provisionnés ?

Oui. Vous pouvez créer une stratégie d'autogrow à tout moment et l'appliquer aux PVC existants en ajoutant ou en mettant à jour l'annotation `trident.netapp.io/autogrowPolicy`. Vous n'avez pas besoin de recréer le PVC ni le StorageClass.

Pour appliquer une politique à un PVC existant :

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Pour appliquer une politique à un StorageClass existant :

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Que se passe-t-il si je configure une stratégie d'autogrow à la fois sur le StorageClass et sur le PVC ?

L'annotation PVC est toujours prioritaire. Si un PVC possède l'annotation `trident.netapp.io/autogrowPolicy`, Trident utilise cette valeur, quelle que soit la valeur spécifiée par la StorageClass. Consultez ["Précédence de la politique"](#) pour plus de détails.

Par exemple, étant donné ce StorageClass :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
annotations:
```

```
trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

Et ce PVC qui annule la politique StorageClass :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident utilise logs-policy pour database-pvc, pas standard-agp.

Comment désactiver l'autogrow pour un volume spécifique ?

Définissez l'annotation PVC sur "none". Ce paramètre remplace toute stratégie de niveau StorageClass pour ce volume :

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Vous pouvez vérifier que la croissance automatique est désactivée :

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Résultat attendu

```
none
```

Que se passe-t-il lorsqu'un volume atteint maxSize?

Trident cesse d'étendre le volume. Il ne crée plus de demandes de redimensionnement pour ce volume, même si l'utilisation continue d'augmenter au-delà de `usedThreshold`.

Par exemple, avec cette politique, Trident cesse d'augmenter le volume une fois qu'il atteint 100 Gio :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Pour autoriser une croissance illimitée, omettez `maxSize` ou définissez-la sur 0 :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Puis-je modifier une stratégie sans redémarrer les volumes ?

Oui. Lorsque vous mettez à jour une stratégie, tous les volumes qui utilisent cette stratégie adoptent les nouveaux paramètres lors de la prochaine évaluation de la croissance. Aucun redémarrage de volume n'est requis.

Pour mettre à jour une politique en place :

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifiez les champs si nécessaire :

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Enregistrez et quittez. Vérifiez la politique mise à jour :

```
kubectl get tridentautogrowpolicy production-db-policy
```

Résultat attendu

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
production-db-policy	75%	20%	Success

Pourquoi ma stratégie est-elle en état d'échec ?

Un `Failed` état indique que la spécification de la politique contient des erreurs de validation. Exécutez la commande suivante pour afficher les détails de l'erreur :

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Les causes fréquentes incluent une valeur invalide `usedThreshold` (doit être comprise entre 1 et 99 %), une `growthAmount` qui dépasse `maxSize`, ou un format de quantité Kubernetes invalide. Corrigez la spécification et réappliquez :

```
kubectl apply -f autogrow-policy.yaml
```

Pourquoi ne puis-je pas supprimer une policy ?

Les politiques utilisent une protection de finalisation. Si des volumes utilisent encore la politique, la suppression entre dans un état `Deleting` et attend que vous retiriez tous les volumes de la politique.

Identifiez les volumes concernés :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Supprimez ensuite l'annotation de chaque PVC :

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Après que vous avez supprimé tous les volumes, Trident libère le finaliseur et supprime la stratégie.

Quels backends sont pris en charge par autogrow ?

Autogrow prend en charge les backends ONTAP. Il prend également en charge Google Cloud NetApp Volumes (GCNV) pour les volumes NAS et SAN.

La fonctionnalité d'extension automatique prend en charge les protocoles NFS, iSCSI, FCP et NVMe. Les volumes de blocs bruts NVMe nécessitent ONTAP 9.16.1 ou une version ultérieure.

Dans les déploiements brownfield, les volumes existants peuvent nécessiter la migration de la publication du volume avant que l'extension automatique ne prenne effet. Pour vérifier l'état de la migration, consultez les journaux du contrôleur Trident :

```
kubectl logs -l app=trident-controller -n trident | grep "Migration completed"
```

Les exemples suivants de StorageClass montrent l'extension automatique configurée pour les backends NAS et SAN :

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Quel est le montant minimal de croissance pour les volumes SAN ?

Pour les volumes SAN, la marge de croissance minimale effective est de 51 Mo. Si vous configurez une `growthAmount` de 50 Mio ou moins, Trident augmente automatiquement la croissance à 51 Mo pour l'opération de redimensionnement.

Par exemple, cette politique définit une `growthAmount` de "40Mi", mais Trident applique une croissance de 51 Mo à tout volume SAN qui l'utilise :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Pour éviter ce réglage automatique, définissez `growthAmount` sur une valeur supérieure à 50 Mio :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importer des volumes Trident

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.

Aperçu et considérations

Vous pouvez importer un volume dans Trident pour :

- Conteneurisez une application et réutilisez son ensemble de données existant
- Utilisez un clone d'un ensemble de données pour une application éphémère
- Reconstruire un cluster Kubernetes défaillant
- Migrer les données d'application lors d'une reprise après sinistre

Considérations

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

- Trident peut importer uniquement les volumes ONTAP de type RW (lecture-écriture). 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 suggérons d'importer les volumes sans connexion active. Pour importer un volume utilisé activement, clonez le volume, puis effectuez l'importation.



Ceci est particulièrement important pour les volumes de blocs, car Kubernetes ignorerait la connexion précédente et pourrait facilement associer un volume actif à un pod. Cela peut entraîner une corruption des données.

- Bien que `StorageClass` doive être spécifié sur un PVC, Trident ne l'utilise pas lors de l'importation. Les classes de stockage sont utilisées lors de la création du volume pour sélectionner parmi les pools disponibles en fonction des caractéristiques de stockage. Parce que le volume existe déjà, aucune sélection de pool n'est requise lors de l'importation. Par conséquent, l'importation ne sera pas un échec même si le volume existe sur un backend 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 le PVC. Après que le volume a été importé par le pilote de stockage, le PV est créé avec un `ClaimRef` vers le PVC.
 - La politique de récupération est initialement définie sur `retain` dans le PV. Après que Kubernetes a correctement lié le PVC et le PV, la politique de récupération est mise à jour pour correspondre à la politique de récupération de la Storage Class.
 - Si la politique de récupération de la classe de stockage est `delete`, Trident supprime le volume de stockage lorsque le PV est supprimé.
- Par défaut, Trident gère le PVC et renomme le volume `FlexVol` ainsi que le LUN ou le `qtree` sur le backend. Vous pouvez passer le `--no-manage` indicateur pour importer un volume non géré et le `--no-rename` indicateur pour conserver le nom du volume.
 - `--no-manage*` - Si vous utilisez le `--no-manage` flag, Trident n'effectue aucune opération supplémentaire sur le PVC ou le PV pendant tout 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 de volume et le redimensionnement de volume sont également ignorées.
 - `--no-rename*` - Si vous utilisez le `--no-rename`flag`, Trident conserve le nom du volume existant lors de l'importation des volumes et gère le cycle de vie des volumes. Cette option est prise en charge uniquement pour les pilotes ``ontap-nas`, `ontap-san` (y compris les systèmes ASA r2), `ontap-san-economy`, et `ontap-nas-economy` (importation `qtree`).



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 aux PVC et PV et sert à la fois à indiquer que le volume a été importé et si les PVC et PV sont gérés. 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 la commande `tridentctl import` pour 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). L'argument `-f` est obligatoire 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 volumes suivants pour les pilotes pris en charge.

ONTAP NAS et ONTAP NAS FlexGroup

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



- Le `ontap-nas-economy` pilote gère le stockage au niveau du qtree, il ne prend donc pas en charge l'importation d'un volume FlexVol seul. Pour importer des qtrees avec le `ontap-nas-economy` pilote, consultez [Importer des qtrees](#) (Tech Preview). Lorsque Trident importe un qtree, il importe également le volume parent FlexVol en tant que bucket contenant ce qtree.
- Les `ontap-nas` et `ontap-nas-flexgroup` pilotes n'autorisent pas les noms de volume en double.

Chaque volume créé avec le `ontap-nas` driver est un volume FlexVol sur le cluster ONTAP. L'importation de volumes FlexVol avec le `ontap-nas` driver fonctionne de la même manière. Un volume FlexVol qui existe déjà sur un cluster ONTAP peut être importé comme un `ontap-nas` PVC. De même, les volumes FlexGroup peuvent être importés comme des `ontap-nas-flexgroup` PVC.

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 backend 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` backend :

```
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 1Gi 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>
```

Importer des qtrees



L'importation Qtree est une fonctionnalité en avant-première technique de Trident 26.06. Les fonctionnalités en avant-première technique ne sont pas prises en charge pour une utilisation en production.

À partir de Trident 26.06, vous pouvez importer des qtrees ONTAP existants dans Trident à l'aide du `ontap-nas-economy` driver. Identifiez le qtree par son volume FlexVol et le nom du qtree au format `flexvol/qtree`, ainsi que par l'UUID du backend qui le contient. Les considérations générales d'importation dans [Aperçu et considérations](#) s'appliquent également à l'importation de qtrees.

Importer un qtree à l'aide de `tridentctl`

Spécifiez le qtree au format `flexvol/qtree` comme nom du volume. L'exemple suivant importe le qtree `qtreet1` du volume FlexVol `flexvol1` sur le backend `ontap_nas_economy`:

```
tridentctl import volume ontap_nas_economy flexvol1/qtreet1 -f <path-to-pvc-file>
```

Pour importer le qtree sans le renommer, ajoutez l'option `--no-rename`. Pour importer le qtree sans que Trident gère son cycle de vie, ajoutez l'option `--no-manage`.

Importer un qtree à l'aide d'annotations PVC

Créez un PVC avec les annotations suivantes :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtreet1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

Pour conserver les noms de volume FlexVol et de qtree existants, ajoutez l'annotation `trident.netapp.io/importNoRename: "true"`. Pour importer le qtree comme non géré, ajoutez l'annotation `trident.netapp.io/notManaged: "true"`.

Comportement de dénomination et de gestion

Importation gérée (par défaut)

Trident renomme le qtree pour qu'il corresponde au nom du PVC et renomme le volume FlexVol selon la

convention d'appellation `ontap-nas-economy`. Le chemin de jonction ne change pas.

--no-rename

Trident conserve les noms de volumes FlexVol® et de qtrees existants. Les noms doivent déjà respecter les conventions de nommage de Trident, sinon Trident ne peut pas gérer les opérations de suppression ou de montage pour le qtree.

--no-manage

Trident fait référence au qtree mais n'en gère pas le cycle de vie. Les noms doivent respecter les conventions d'appellation de Trident, et Trident n'utilise ni ne gère les règles d'export automatiques pour le qtree.

Comportement des règles d'export

Si le backend utilise des règles d'export automatiques, le qtree importé utilise le modèle de règles d'export par qtree, dans lequel chaque qtree possède sa propre règle d'export. Lors de l'importation, Trident attribue au qtree une règle d'export vide. Au moment de la publication, Trident met à jour le volume FlexVol avec la règle d'export du backend si l'export automatique est activé et si la règle d'export FlexVol existante est `default` ou vide.



La mise à jour des règles d'export du volume FlexVol peut perturber les autres qtrees du même volume FlexVol qui sont actuellement montés. Importez les qtrees uniquement lorsque les qtrees du volume FlexVol sont démontés.



Ne faites pas pointer deux clusters Kubernetes vers la même SVM active et n'importez pas un qtree déjà géré par un autre cluster. Si les deux clusters gèrent le même volume, des opérations conflictuelles peuvent corrompre ou détruire les données.

ONTAP SAN

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

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

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

Volume géré

Pour les volumes gérés, Trident renomme le volume FlexVol au format `pvc-<uuid>` et le LUN à l'intérieur du volume FlexVol en `lun0`.

L'exemple suivant importe le `ontap-san-managed` volume FlexVol qui est présent sur le `ontap_san_default` backend :

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

```

+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
| 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` backend :

```

tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog       |
+-----+-----+-----+-----+
| block      | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false        |
+-----+-----+-----+-----+
+-----+-----+-----+

```

Si vous avez des LUN mappés à des igroups qui partagent un IQN avec l'IQN d'un nœud Kubernetes, comme illustré dans l'exemple suivant, vous recevrez l'erreur : `LUN already mapped to initiator(s) in this group`. Vous devrez supprimer l'initiateur ou dissocier le 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

Exemples d'économie SAN ONTAP

Les exemples suivants montrent comment importer des volumes gérés et non gérés pour le backend `ontap-san-economy`.

Volume géré

Lors de l'importation d'un volume géré, Trident prend possession du FlexVol et le renomme. Vous devez tenir compte de ce renommage lors de l'importation de plusieurs LUN provenant du même FlexVol.

L'exemple suivant importe `lun1` du FlexVol `toimport` en tant que volume géré nommé `vol-managed-saneco` :

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Après l'importation `lun1`, Trident renomme le FlexVol (par exemple, en `trident_lun_pool_xyz`). Pour importer d'autres LUNs du même FlexVol, utilisez le nouveau nom FlexVol :

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```



Le `ontap-san-economy` backend importe une LUN à la fois. Vous pouvez automatiser plusieurs importations à l'aide d'un script.

Volume non géré

Lorsque vous importez un volume non géré, Trident ne prend pas possession du FlexVol®. Cependant, le FlexVol® et la LUN doivent suivre les conventions de nommage Trident.

Format de nommage FlexVol®

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` est la valeur de `storagePrefix` dans votre configuration backend. La valeur par défaut est `trident`.
- `RANDOMSTRING` est n'importe quelle chaîne de caractères que vous choisirez.

Exigence de dénomination des LUN

Le LUN doit être nommé `lun0`.

Exemple

Si votre `storagePrefix` est `xyz`, le chemin complet vers le LUN est :

```
trident_lun_pool_xyz_randomstring/lun0
```

Élément

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



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

L'exemple suivant importe un `element-managed` volume sur le backend `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` driver.



Pour importer un volume Azure NetApp Files, identifiez le volume par son chemin de volume. Le chemin de 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 de volume est `importvol1`.

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

```
tridentctl import volume azurenetaappfiles_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` driver.

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 Trident

Avec Trident, vous pouvez attribuer des noms et des étiquettes explicites aux volumes que vous créez. Cela vous aide à identifier et à associer facilement les volumes à leurs ressources Kubernetes (PVC) respectives. Vous pouvez également définir des modèles au niveau du backend pour créer des noms et des étiquettes de volumes personnalisés ; tous les volumes que vous créez, importez ou clonez respecteront ces modèles.

Avant de commencer

Prise en charge des noms et étiquettes de volume personnalisables :

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

Limitations

- 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` pilotes.
- Les noms de volumes personnalisés ne s'appliquent pas aux volumes existants.

Comportements clés des noms de volumes personnalisables

- En cas d'échec dû à une syntaxe incorrecte dans un modèle de nom, la création du backend échoue. Cependant, si l'application du modèle échoue, le volume sera nommé selon 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 issu de la configuration du backend. Toute valeur de préfixe souhaitée peut être ajoutée directement au modèle.

Exemples de configuration backend avec modèle de nom et labels

Des modèles de noms personnalisés peuvent être définis au niveau racine et/ou au niveau 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.Requ
estName}}"
},
"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 }}"
      }
    }
  ]
}

```

```

}}_{{ .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 considérer

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

Partager un volume NFS Trident entre 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.

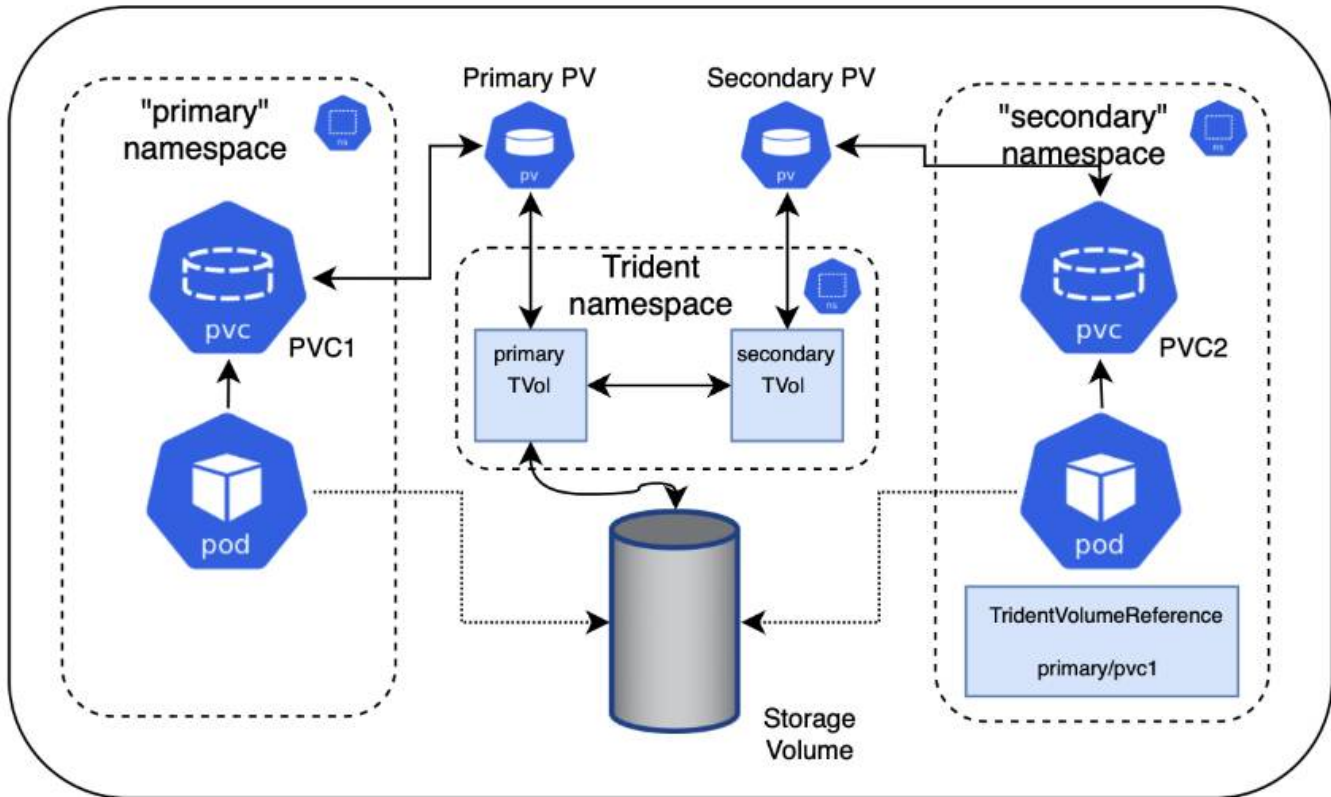
Caractéristiques

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

- Plusieurs niveaux de contrôle d'accès pour garantir la sécurité

- Fonctionne avec tous les pilotes de volumes NFS Trident
- Aucune dépendance à tridentctl ni à toute autre fonctionnalité non native de Kubernetes

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



Démarrage rapide

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

1

Configurez le PVC source pour partager le volume

Le propriétaire de l'espace de noms source accorde l'autorisation d'accéder aux données du PVC source.

2

Autoriser la création d'un CR dans l'espace de noms de destination

L'administrateur du cluster autorise le propriétaire de l'espace de noms de destination à créer la `TridentVolumeReference` CR.

3

Créer `TridentVolumeReference` dans l'espace de noms de destination

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

4

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

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

Configurez les espaces de noms source et de destination

Pour garantir la sécurité, le partage entre espaces de noms nécessite la collaboration et l'intervention 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 de l'utilisateur est défini à chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source** : Créez le PVC (`pvc1` dans l'espace de noms source qui autorise le partage avec l'espace de noms de destination (`namespace2` en utilisant l'annotation `shareToNamespace`.

```
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 PV et son volume de stockage NFS backend.



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

2. **Administrateur du cluster** : Assurez-vous qu'un contrôle d'accès basé sur les rôles (RBAC) approprié est en place pour accorder au propriétaire de l'espace de noms de destination l'autorisation de créer la `TridentVolumeReference` CR dans l'espace de noms de destination.
3. **Propriétaire de l'espace de noms de destination** : Créez une 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 (pvc2 dans l'espace de noms de destination (namespace2 en utilisant l'annotation `shareFromPVC` pour désigner le 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 celle du PVC source.

Résultats

Trident lit l'annotation `shareFromPVC` sur le PVC de destination et crée le PV de destination comme un volume subordonné sans ressource de stockage propre qui pointe vers le PV source et partage la ressource de stockage du PV source. Le PVC et le PV de destination apparaissent liés normalement.

Supprimer un volume partagé

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

Utilisez `tridentctl get` pour interroger les volumes subordonnés

À l'aide de l'[`tridentctl`] utilitaire, vous pouvez exécuter la commande `get` pour obtenir les volumes subordonnés. Pour plus d'informations, consultez le lien : [../trident-reference/trident-cl.html](https://trident-reference.netapp.io/trident-cl.html) [`tridentctl` commandes et options].

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

Drapeaux:

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

Limitations

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

Pour plus d'informations

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

- Regardez la démo sur ["NetAppTV"](#).

Cloner des volumes Trident entre des espaces de noms

Avec Trident, vous pouvez créer de nouveaux volumes à l'aide de volumes existants ou de volumesnapshots provenant d'un espace de noms différent au sein du même cluster Kubernetes.

Prérequis

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



Le clonage entre espaces de noms est pris en charge uniquement pour les `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 volume en quelques étapes seulement.



Configurez le PVC source pour cloner le volume

Le propriétaire de l'espace de noms source accorde l'autorisation d'accéder aux données du PVC source.

2

Autoriser la création d'un CR dans l'espace de noms de destination

L'administrateur du cluster autorise le propriétaire de l'espace de noms de destination à créer la TridentVolumeReference CR.

3

Créer TridentVolumeReference dans l'espace de noms de destination

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

4

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

Le propriétaire de l'espace de noms de destination crée un PVC pour cloner le PVC de l'espace de noms source.

Configurez 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'intervention 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 de l'utilisateur est défini à chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source** : Créez le PVC (pvc1) dans l'espace de noms source (namespace1) qui autorise le partage avec l'espace de noms de destination (namespace2) en utilisant l'annotation `cloneToNamespace`.

```
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 PV et son volume de stockage backend.



- Vous pouvez partager le PVC avec plusieurs espaces de noms à l'aide d'une liste séparée par des virgules. Par exemple `trident.netapp.io/cloneToNamespace: namespace2, namespace3, namespace4`.

- Vous pouvez partager avec tous les espaces de noms en utilisant *. Par exemple, `trident.netapp.io/cloneToNamespace: *`
- Vous pouvez mettre à jour le PVC pour inclure l'annotation `cloneToNamespace` à tout moment.

- 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 la `TridentVolumeReference` CR dans l'espace de noms de destination (`namespace2`).
- Propriétaire de l'espace de noms de destination :** Créez une `TridentVolumeReference` CR dans l'espace de noms de destination qui fait référence au PVC source `pvc1`.

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

- Propriétaire de l'espace de noms de destination :** Créez un PVC (`pvc2`) dans l'espace de noms de destination (`namespace2`) en utilisant les `cloneFromPVC` ou `cloneFromSnapshot`, et `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
```

Limitations

- Pour les PVC provisionnés à l'aide des pilotes `ontap-nas-economy`, les clones en lecture seule ne sont pas pris en charge.

Répliquer les volumes à l'aide de SnapMirror

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

- Créer des relations de miroir entre les volumes (PVCs)
- Supprimer les relations de miroir entre les volumes
- Briser les relations miroir
- Promouvoir le volume secondaire pendant des conditions de sinistre (basculements)
- Effectuer une transition sans perte des applications d'un cluster à l'autre (lors de basculements ou de migrations planifiés)

Prérequis de réplication

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

clusters ONTAP

- **Trident** : la version 22.10 ou ultérieure de Trident doit être présente sur les clusters Kubernetes source et de destination qui utilisent ONTAP comme backend.
- **Licences** : Les licences asynchrones ONTAP SnapMirror utilisant le Data Protection bundle doivent être activées sur les clusters ONTAP source et de destination. Consultez ["SnapMirror aperçu des licences dans ONTAP"](#) pour plus d'informations.

À compter de ONTAP 9.10.1, toutes les licences sont fournies sous forme de fichier de licence NetApp (NLF), qui est un fichier unique permettant d'activer plusieurs fonctionnalités. Consultez ["Licences incluses avec ONTAP One"](#) pour plus d'informations.



Seule la protection asynchrone SnapMirror est prise en charge.

Interconnexion

- **Cluster et SVM** : Les backends de stockage ONTAP doivent être appariés. Consultez ["Aperçu du peering de cluster et de SVM"](#) pour plus d'informations.



Assurez-vous que les noms SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

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

Pilotes pris en charge

NetApp Trident prend en charge la réplication de volumes avec la technologie NetApp SnapMirror en utilisant des classes de stockage prises en charge par les pilotes suivants : **ontap-nas: NFS** **ontap-san: iSCSI** **ontap-san: FC** **ontap-san: NVMe/TCP** (nécessite la version ONTAP 9.15.1 au minimum)



La réplication de volumes à l'aide de SnapMirror n'est pas prise en charge pour les systèmes

ASA r2. Pour des informations sur les systèmes ASA r2, voir ["En savoir plus sur les systèmes de stockage ASA r2"](#).

Créer un PVC en miroir

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

Étapes

1. Effectuez les étapes suivantes sur le cluster Kubernetes principal :
 - a. Créez un objet StorageClass 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:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

- b. Créez un PVC avec le StorageClass précédemment créé.

Exemple

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

- c. Créez un CR 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 actuel de protection des données (DP) du volume, puis remplit le champ d'état du MirrorRelationship.

- d. Obtenez le TridentMirrorRelationship CR pour obtenir le nom interne et le 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 un StorageClass 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 un CR MirrorRelationship avec les informations de destination et de source.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Le TMR spec comporte quatre champs de niveau supérieur `state`, qui définit l'état de relation souhaité (voir [États de réplication du volume](#)) `volumeMappings`, qui associe les PVC locaux à leurs descripteurs de volume distants, ainsi que les champs facultatifs `replicationPolicy` et `replicationSchedule` décrits ci-dessous.

Les champs `replicationPolicy` et `replicationSchedule` sont facultatifs. Définissez `replicationPolicy` sur le nom d'une stratégie SnapMirror asynchrone sur ONTAP et `replicationSchedule` sur le nom d'une planification ONTAP. Si l'un des champs est omis, Trident utilise la stratégie de réplication ou la planification correspondante configurée sur le backend. Définissez ces champs lorsque vous créez la relation ; vous ne pouvez pas modifier la stratégie ni la planification une fois la relation établie. Pour utiliser des paramètres différents, supprimez et recréez le TMR.

Trident crée et initialise une relation SnapMirror à l'aide de la politique et de la planification de réplication configurées.

1. Créez un PVC avec la StorageClass précédemment créée pour servir de SnapMirror destination secondaire.

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 vérifiera la présence de la `TridentMirrorRelationship` CRD et ne créera pas le volume si la relation n'existe pas. Si la relation existe, Trident s'assurera que le nouveau volume `FlexVol` est placé sur une SVM appairée avec la SVM distante définie dans la `MirrorRelationship`.

États de réplication du volume

Une Trident Mirror Relationship (TMR) est une CRD qui représente une extrémité d'une relation de réplication entre des PVC. La TMR de destination possède un état, qui indique à Trident quel est l'état souhaité. La TMR de destination possède 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 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 a déjà été 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 en relation avec la source.

Promouvoir le PVC secondaire lors d'un basculement imprévu

Effectuez l'étape suivante sur le cluster Kubernetes secondaire :

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

Promouvoir le PVC secondaire lors d'un basculement planifié

Lors d'un basculement (migration) planifié, effectuez les étapes suivantes pour promouvoir le PVC secondaire :

Étapes

1. Sur le cluster Kubernetes principal, créez un instantané du PVC et attendez que l'instantané soit créé.
2. Sur le cluster Kubernetes principal, créez le `TridentTridentSnapshotInfoCR` pour obtenir les détails internes.

Exemple

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

- a. Sur le cluster Kubernetes secondaire, mettez à jour le champ `spec.state` de la ressource personnalisée `TridentMirrorRelationship` à `promoted` et `spec.volumeMappings[].promotedSnapshotHandle` pour qu'il corresponde à l'`internalName` du snapshot.
3. Sur le cluster Kubernetes secondaire, confirmez l'état (champ `status.state`) de `TridentMirrorRelationship` sur `promu`.

Rétablir une relation miroir après un basculement

Avant de rétablir une relation miroir, choisissez le côté que vous souhaitez définir comme nouveau principal.

Étapes

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

Opérations supplémentaires

Trident prend en charge les opérations suivantes sur les volumes primaires et secondaires :

Répliquer le PVC primaire vers un nouveau PVC secondaire

Assurez-vous de disposer déjà d'un PVC primaire et d'un PVC secondaire.

Étapes

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

Redimensionner un PVC miroir, principal ou secondaire

Le PVC peut être redimensionné normalement, ONTAP étendra automatiquement les flevxols de destination si la quantité de données dépasse la taille actuelle.

Supprimer la réplication d'un PVC

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

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

Supprimer un PVC (qui était précédemment mis en miroir)

Trident vérifie la présence de PVC répliqués et libère la relation de réplication avant de tenter de supprimer le volume.

Supprimer un TMR

La suppression d'un TMR sur l'un des côtés d'une relation miroir entraîne la transition du TMR restant à l'état `promoted` avant que Trident ne finalise la suppression. Si le TMR sélectionné pour suppression est déjà à l'état `promoted`, il n'existe aucune relation miroir et le TMR sera supprimé et Trident promouvra le PVC local à `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 devra utiliser un nouveau TMR avec un état de réplication de volume `established` lors de la création de la nouvelle relation miroir.

Mettez à jour les relations miroir lorsque ONTAP est en ligne

Les relations de miroir peuvent être mises à jour à tout moment après leur établissement. Vous pouvez utiliser le `state: promoted` ou `state: reestablished` champ pour mettre à jour les relations. Lors de la promotion d'un volume de destination vers un volume ReadWrite standard, vous pouvez utiliser `promotedSnapshotHandle` pour spécifier un instantané précis auquel restaurer le volume actuel.

Mettre à jour les relations miroir lorsque ONTAP est hors ligne

Vous pouvez utiliser une CRD pour effectuer une mise à jour SnapMirror sans que Trident ait de connexion directe au cluster ONTAP. Consultez l'exemple de format suivant de `TridentActionMirrorUpdate` :

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 de la `TridentActionMirrorUpdate` CRD. Elle peut prendre une valeur parmi *Succeeded*, *In Progress* ou *Failed*.

Utiliser la topologie CSI

Trident peut créer et attacher sélectivement des volumes aux nœuds présents dans un cluster Kubernetes en utilisant le "[Fonctionnalité de topologie CSI](#)".

Aperçu

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



Apprenez-en davantage sur la fonctionnalité de topologie CSI "[ici](#)".

Kubernetes propose deux modes de liaison de volumes uniques :

- Avec `VolumeBindingMode` défini sur `Immediate`, Trident crée le volume sans aucune prise en compte de la topologie. La liaison de volume et le provisionnement dynamique sont gérés lors de la création du PVC. C'est le comportement par défaut `VolumeBindingMode` et il convient aux clusters qui n'imposent pas de contraintes de topologie. Les volumes persistants sont créés sans dépendre des exigences de planification du pod demandeur.
- Avec `VolumeBindingMode` défini sur `WaitForFirstConsumer`, la création et la liaison d'un volume persistant pour un PVC sont différées jusqu'à ce qu'un pod utilisant ce PVC soit planifié et créé. De cette

façon, les volumes sont créés pour répondre aux contraintes de planification imposées par les exigences de topologie.



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

Ce dont vous aurez besoin

Pour utiliser la topologie CSI, vous avez besoin 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:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
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:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Les nœuds du cluster doivent comporter des étiquettes permettant de prendre en compte la topologie (`topology.kubernetes.io/region` et `topology.kubernetes.io/zone`). Ces étiquettes **doivent être présentes sur les nœuds du cluster** avant l'installation de Trident pour que Trident puisse prendre en compte 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 : Créez un backend tenant compte de la topologie

Les backends de stockage Trident peuvent être conçus pour provisionner sélectivement des volumes en fonction des zones de disponibilité. Chaque backend peut comporter un bloc `supportedTopologies` optionnel qui représente une liste de zones et de régions prises en charge. Pour les StorageClasses qui utilisent un tel backend, un volume ne serait créé que si la demande provient d'une application planifiée dans une région ou une zone prise en charge.

Voici un exemple de définition de backend :

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 backend. Ces régions et zones représentent la liste des valeurs autorisées qui peuvent être fournies dans un `StorageClass`. Pour les `StorageClasses` qui contiennent un sous-ensemble des régions et zones fournies dans un backend, Trident crée un volume sur le backend.

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

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b

```

Dans cet exemple, les `region` et `zone` étiquettes représentent l'emplacement du pool de stockage. `topology.kubernetes.io/region` et `topology.kubernetes.io/zone` indiquent où les pools de stockage peuvent être utilisés.

Étape 2 : Définir les `StorageClasses` qui sont conscients de la topologie

En fonction des étiquettes de topologie fournies aux nœuds du cluster, il est possible de définir des `StorageClasses` contenant des informations de topologie. Cela déterminera les pools de stockage qui servent

de candidats pour les requêtes PVC effectuées, ainsi que le sous-ensemble de nœuds pouvant utiliser les volumes provisionnés par Trident.

Voir l'exemple suivant :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - 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 StorageClass fournie ci-dessus, volumeBindingMode est définie sur WaitForFirstConsumer. Les PVC demandés avec cette StorageClass ne seront pas traités tant qu'ils ne seront pas référencés dans un pod. Et, allowedTopologies fournit les zones et la région à utiliser. La netapp-san-us-east1 StorageClass crée des PVC sur le backend san-backend-us-east1 défini ci-dessus.

Étape 3 : Créer et utiliser un PVC

Une fois le StorageClass créé et associé à un backend, vous pouvez désormais créer des PVC.

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'un PVC à l'aide de ce manifeste donnerait le résultat suivant :

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME          STATUS      VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending                                netapp-san-us-east1
2s
kubectl 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 au PVC, utilisez le PVC 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 indique à Kubernetes de planifier le pod sur les nœuds présents dans la us-east1 région, et de choisir parmi tous les nœuds présents dans la us-east1-a ou us-east1-b zones.

Voir la sortie suivante :

```

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

```

Mettre à jour les backends pour inclure `supportedTopologies`

Les backends préexistants peuvent être mis à jour pour inclure une liste de `supportedTopologies` en utilisant `tridentctl backend update`. Cela n'affectera pas les volumes déjà provisionnés et ne sera utilisé que pour les PVC ultérieurs.

Pour plus d'informations

- ["Gérer les ressources pour les conteneurs"](#)
- ["nodeSelector"](#)
- ["Affinité et anti-affinité"](#)
- ["Souillures et tolérances"](#)

Travailler avec les instantanés Trident

Les snapshots de volumes persistants (PV) Kubernetes permettent de créer des copies à un instant précis des volumes. Vous pouvez créer un snapshot d'un volume créé avec Trident, importer un snapshot créé en dehors de Trident, créer un nouveau volume à partir d'un snapshot existant, et récupérer des données de volume à partir de snapshots.

Aperçu

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` pilotes.

Avant de commencer

Vous devez disposer d'un contrôleur de snapshots externe et de Custom Resource Definitions (CRDs) pour travailler avec les snapshots. C'est la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

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



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

Créer un instantané de volume

Étapes

1. Créez un `VolumeSnapshotClass`. Pour plus d'informations, consultez ["VolumeSnapshotClass"](#).
 - Le `driver` pointe vers le pilote Trident CSI.
 - `deletionPolicy` peut être `Delete` ou `Retain`. Lorsqu'il est défini sur `Retain`, l'instantané physique sous-jacent sur le cluster de stockage est conservé même lorsque l'objet `VolumeSnapshot` est 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 instantané d'un PVC existant.

Exemples

- Cet exemple crée un instantané d'un PVC 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 d'instantané de volume pour un PVC nommé `pvc1` et le nom de l'instantané est défini sur `pvc1-snap`. Un `VolumeSnapshot` est analogue à un PVC et est associé à un `VolumeSnapshotContent` objet qui représente l'instantané 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 l' `VolumeSnapshotContent` objet pour le `pvc1-snap` `VolumeSnapshot` en le décrivant. Le `Snapshot Content Name` identifie l'objet `VolumeSnapshotContent` qui sert cet instantané. Le `Ready To Use` paramètre indique que l'instantané peut être utilisé pour créer un nouveau PVC.

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

Créer un PVC à partir d'un instantané de volume

Vous pouvez utiliser `dataSource` pour créer un PVC en utilisant un VolumeSnapshot nommé `<pvc-name>` comme source des données. Après la création du PVC, il peut être attaché à un pod et utilisé comme n'importe quel autre PVC.



Le PVC sera créé dans le même backend que le volume source. Reportez-vous à ["KB : La création d'un PVC à partir d'un instantané de PVC Trident ne peut pas être effectuée dans un autre backend"](#).

L'exemple suivant crée le PVC en utilisant `pvcl-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: pvc1-snap
kind: VolumeSnapshot
apiGroup: snapshot.storage.k8s.io
```

Importer un instantané de volume

Trident prend en charge la "[Processus de snapshot pré-provisionné Kubernetes](#)" pour permettre à l'administrateur du cluster de créer un `VolumeSnapshotContent` objet et d'importer des instantanés créés en dehors de Trident.

Avant de commencer

Trident doit avoir créé ou importé le volume parent de la capture.

Étapes

1. **Administrateur du cluster** : Créez un `VolumeSnapshotContent` objet qui référence l'instantané du backend. Cela lance le workflow d'instantané dans Trident.
 - Spécifiez le nom de l'instantané du backend 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 générateur de snapshots externe dans l'appel `ListSnapshots`.



Le `<volumeSnapshotContentName>` ne peut pas toujours correspondre au nom de l'instantané du backend en raison des contraintes de dénomination des CR.

Exemple

L'exemple suivant crée un `VolumeSnapshotContent` objet qui référence l'instantané du backend `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. **Administrateur du cluster** : Créez la VolumeSnapshot CR qui référence l'`VolumeSnapshotContent` objet. Cela demande l'autorisation d'utiliser l'`VolumeSnapshot` dans un espace de noms donné.

Exemple

L'exemple suivant crée un VolumeSnapshot CR nommé `import-snap` qui référence le 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 gestionnaire de snapshots externe reconnaît le nouvellement créé VolumeSnapshotContent et exécute l'appel `ListSnapshots`. Trident crée le `TridentSnapshot`.
 - Le dispositif de capture d'écran externe définit le VolumeSnapshotContent sur `readyToUse` et le VolumeSnapshot sur `true`.
 - Trident retourne `readyToUse=true`.
4. **Tout utilisateur** : Créez un `PersistentVolumeClaim` pour référencer le nouveau VolumeSnapshot, où le `spec.dataSource` (ou `spec.dataSourceRef`) nom est le VolumeSnapshot nom.

Exemple

L'exemple suivant crée un PVC faisant référence au 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
```

Récupérer les données du volume à l'aide de snapshots

Le répertoire des instantanés est masqué par défaut afin d'assurer une compatibilité maximale des volumes provisionnés à l'aide des `ontap-nas` et `ontap-nas-economy` pilotes. Activez le répertoire `.snapshot` pour récupérer directement les données à partir des instantanés.

Utilisez la commande `ONTAP volume snapshot restore` pour restaurer un volume à un état enregistré dans un instantané précédent.

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



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

Restauration sur place du volume à partir d'un instantané

Trident permet une restauration rapide et directe des volumes à partir d'un instantané grâce à la `TridentActionSnapshotRestore` (TASR) CR. Cette CR fonctionne comme une action Kubernetes impérative et n'est pas conservée après la fin de l'opération.

Trident prend en charge la restauration d'instantanés sur les `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` et `solidfire-san` pilotes.

Avant de commencer

Vous devez disposer d'une PVC reliée et d'un instantané de volume disponible.

- Vérifiez que le statut du PVC est `bound`.

```
kubectl get pvc
```

- Vérifiez que l'instantané du volume est prêt à être utilisé.

```
kubectl get vs
```

Étapes

1. Créez la TASR CR. Cet exemple crée une CR pour PVC `pvc1` et volume snapshot `pvc1-snapshot`.



Le TASR CR doit se trouver dans un espace de noms 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 restaurer à partir de l'instantané. Cet exemple restaure à partir de l'instantané pvc1.

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

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

Résultats

Trident restaure les données à partir de l'instantané. Vous pouvez vérifier l'état de la restauration de l'instantané :

```
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 relancera pas automatiquement l'opération en cas d'échec. Vous devrez effectuer l'opération à nouveau.
- Les utilisateurs de Kubernetes ne disposant pas d'un accès administrateur peuvent devoir obtenir l'autorisation de l'administrateur pour créer un TASR CR dans l'espace de noms de leur application.

Supprimez un PV avec les instantanés associés

Lors de la suppression d'un volume persistant avec des snapshots associés, le volume Trident correspondant passe à l'état « Suppression en cours ». Supprimez les snapshots du volume pour supprimer le volume Trident.

Déployer un contrôleur d'instantané de volume

Si votre distribution Kubernetes n'inclut pas le snapshot controller et les CRDs, vous pouvez les déployer comme suit.

Étapes

1. Créer des CRD d'instantané 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.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Créez 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, ouvrez `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettez à jour namespace pour votre espace de noms.

Liens associés

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

Utilisation des instantanés de groupes de volumes Trident

Les instantanés de groupes de volumes Kubernetes des volumes persistants (PV) NetApp Trident offrent la possibilité de créer des instantanés de plusieurs volumes (un groupe d'instantanés de volumes). Cet instantané de groupe de volumes représente des copies de plusieurs volumes prises au même moment.



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 avec les pilotes de stockage suivants :

- `ontap-san` driver - 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 NetApp ASA r2 ou les systèmes de stockage AFX.

Avant de commencer

- Assurez-vous que votre version de Kubernetes est K8s 1.32 ou supérieure.
- Vous devez disposer d'un contrôleur de snapshots externe et de Custom Resource Definitions (CRDs) pour travailler avec les snapshots. C'est la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

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



Ne créez pas de contrôleur de snapshots si vous créez des snapshots de groupes de

volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshots intégré et masqué.

- Dans le fichier YAML du contrôleur d'instantané, définissez la `CSIVolumeGroupSnapshot` feature gate sur « true » pour garantir que l'instantané du groupe de volumes est activé.
- Créez les classes d'instantané 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, consultez "[VolumeGroupSnapshotClass](#)".

```
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 en utilisant les classes de stockage existantes, ou ajoutez ces étiquettes aux PVC existants.

L'exemple suivant crée le PVC en utilisant `pvc1-group-snap` comme source de données et l'étiquette `consistentGroupSnapshot: groupA`. Définissez la clé et la valeur de l'étiquette en fonction de vos besoins.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Créez un VolumeGroupSnapshot avec la même étiquette (`consistentGroupSnapshot: groupA`) spécifiée 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 les données du volume à l'aide d'un instantané de groupe

Vous pouvez restaurer des volumes persistants individuels à l'aide des instantanés individuels qui ont été créés dans le cadre de l'instantané de groupe de volumes. Vous ne pouvez pas restaurer l'instantané de groupe de volumes en tant qu'unité.

Utilisez la commande ONTAP volume snapshot restore pour restaurer un volume à un état enregistré dans un instantané précédent.

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



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

Restauration sur place du volume à partir d'un instantané

Trident permet une restauration rapide et directe des volumes à partir d'un instantané grâce à la `TridentActionSnapshotRestore` (TASR) CR. Cette CR fonctionne comme une action Kubernetes impérative et n'est pas conservée après la fin de l'opération.

Pour plus d'informations, consultez ["Restauration sur place du volume à partir d'un instantané"](#).

Supprimer un PV avec des snapshots de groupe associés

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

- Vous pouvez supprimer VolumeGroupSnapshots dans leur ensemble, pas les instantanés individuels du groupe.
- Si des PersistentVolumes sont supprimés alors qu'un instantané existe pour ce PersistentVolume, Trident déplacera ce volume vers un état « en cours 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 le clone commencera et le groupe ne pourra pas être supprimé tant que la division n'est pas terminée.

Déployer un contrôleur d'instantané de volume

Si votre distribution Kubernetes n'inclut pas le snapshot controller et les CRDs, vous pouvez les déployer comme suit.

Étapes

1. Créer des CRD d'instantané 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. Créez 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, ouvrez `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettez à jour namespace pour votre espace de noms.

Liens associés

- ["VolumeGroupSnapshotClass"](#)
- ["Instantanés 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.