



# **Provisionnement et gestion des volumes**

Astra Trident

NetApp  
January 14, 2026

# Sommaire

Provisionnement et gestion des volumes	1
Provisionner un volume	1
Présentation	1
Créer le volume persistant et la demande de volume persistant	4
Développement des volumes	5
Développez un volume iSCSI	5
Développez un volume NFS	10
Importer des volumes	13
Présentation et considérations	13
Importer un volume	14
Exemples	15
Partager un volume NFS entre les espaces de noms	20
Caractéristiques	20
Démarrage rapide	20
Configurer les espaces de noms source et de destination	21
Supprimer un volume partagé	23
Utiliser <code>tridentctl get</code> pour interroger des volumes subordonnés	23
Limites	23
Pour en savoir plus	23
Réplication de volumes à l'aide de SnapMirror	23
Conditions préalables à la réplication	24
Créer une demande de volume persistant en miroir	24
États de réplication des volumes	27
Promotion de la demande de volume persistant secondaire en cas de basculement non planifié	28
Promotion de la demande de volume persistant secondaire lors d'un basculement planifié	28
Restaurer une relation de miroir après un basculement	28
Opérations supplémentaires	29
Mettre à jour les relations miroir lorsque ONTAP est en ligne	29
Mettre à jour les relations en miroir lorsque ONTAP est hors ligne	29
Activez le mécanisme de provisionnement Astra Control	30
Utiliser la topologie CSI	39
Présentation	39
Étape 1 : création d'un back-end conscient de la topologie	41
Étape 2 : définissez des classes de stockage qui prennent en compte la topologie	43
Étape 3 : création et utilisation d'une demande de volume persistant	44
Mise à jour des systèmes back-end pour inclure <code>supportedTopologies</code>	47
Trouvez plus d'informations	47
Travailler avec des instantanés	47
Présentation	47
Créer un snapshot de volume	48
Créer une demande de volume persistant à partir d'un snapshot de volume	49
Importer un instantané de volume	50

Restaurez les données de volume à l'aide de snapshots .....	52
Restauration de volumes sur place à partir d'un snapshot .....	52
Supprimez un volume persistant avec les snapshots associés .....	54
Déployer un contrôleur de snapshot de volume .....	54
Liens connexes .....	55

# Provisionnement et gestion des volumes

## Provisionner un volume

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

### Présentation

A "*Volume persistant*" (PV) est une ressource de stockage physique provisionnée par l'administrateur du cluster sur un cluster Kubernetes. Le "*PersistentVolumeClaim*" (PVC) est une demande d'accès au volume persistant sur le cluster.

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

Après avoir créé le volume persistant et la demande de volume persistant, vous pouvez monter le volume dans un pod.

### Exemples de manifestes

#### Exemple de manifeste de volume persistant

Cet exemple de manifeste montre un volume persistant de base de 10Gi associé à StorageClass `basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

## Exemples de manifestes de demande de volume persistant

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

### PVC avec accès RWO

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

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

### PVC avec NVMe/TCP

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

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

## Échantillons de manifeste de pod

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

### Configuration de base

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

## Configuration NVMe/TCP de base

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    resources: {}
    volumeMounts:
    - mountPath: "/usr/share/nginx/html"
      name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
  - name: task-pv-storage
    persistentVolumeClaim:
      claimName: pvc-san-nvme
```

## Créer le volume persistant et la demande de volume persistant

### Étapes

1. Créer la PV.

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

2. Vérifiez l'état du PV.

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. Créer la PVC.

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

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

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-storage	Bound	pv-name	2Gi	RWO		5m

5. Montez le volume dans un pod.

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



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

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

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

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

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

Reportez-vous à la section "[Kubernetes et objets Trident](#)" pour plus d'informations sur l'interaction des classes de stockage avec le `PersistentVolumeClaim` Et paramètres de contrôle du provisionnement des volumes par Astra Trident.

## Développement des volumes

Astra Trident permet aux utilisateurs de Kubernetes d'étendre leurs volumes après leur création. Trouvez des informations sur les configurations requises pour développer les volumes iSCSI et NFS.

### Développez un volume iSCSI

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



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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

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

```

#### Étape 4 : développez le volume persistant

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

```

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

```

## Étape 5 : valider l'extension

Vous pouvez valider le bon fonctionnement de l'extension en contrôlant la taille de la demande de volume persistant, du volume persistant et du volume Astra Trident :

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

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

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

## Développez un volume NFS

Astra Trident prend en charge l'extension de volume pour les volumes persistants NFS provisionnés sur ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, gcp-cvs, et azure-netapp-files systèmes back-end.

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

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

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

Si vous avez déjà créé une classe de stockage sans cette option, vous pouvez simplement modifier la classe de stockage existante en utilisant `kubectl edit storageclass` pour permettre l'extension de volume.

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

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

Astra Trident doit créer un volume persistant NFS 20MiB pour cette demande de volume persistant :

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

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

## Étape 3 : développez le volume persistant

Pour redimensionner le volume persistant 20MiB nouvellement créé à 1 Gio, modifiez la demande de volume persistant et définissez-la `spec.resources.requests.storage` À 1 Gio :

```

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

```

#### Étape 4 : validation de l'extension

Vous pouvez valider le redimensionnement correctement en contrôlant la taille de la demande de volume persistant, de la volume persistant et du volume Astra Trident :

```
kubectl get pvc ontapnas20mb
```

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

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

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

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

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

## Importer des volumes

Vous pouvez importer des volumes de stockage existants sous forme de volume persistant Kubernetes à l'aide de `tridentctl import`.

### Présentation et considérations

Vous pouvez importer un volume dans Astra Trident et :

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

### Considérations

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

- ASTRA Trident peut importer des volumes ONTAP de type RW (lecture-écriture) uniquement. Les volumes de type DP (protection des données) sont des volumes de destination SnapMirror. Vous devez rompre la



relation de miroir avant d'importer le volume dans Astra Trident.

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



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

- Cependant `StorageClass` Doit être spécifié sur une demande de volume persistant, Astra Trident n'utilise pas ce paramètre lors de l'importation. Les classes de stockage sont utilisées lors de la création du volume pour sélectionner un pool disponible en fonction des caractéristiques de stockage. Comme le volume existe déjà, aucune sélection de pool n'est requise pendant l'importation. Par conséquent, l'importation n'échouera pas même si le volume existe sur un back-end ou un pool qui ne correspond pas à la classe de stockage spécifiée dans le PVC.
- La taille du volume existant est déterminée et définie dans la PVC. Une fois le volume importé par le pilote de stockage, le volume persistant est créé avec un `SécurRef` dans la demande de volume persistant.
  - La règle de récupération est initialement définie sur `retain` Dans la PV. Une fois que Kubernetes a réussi à relier la demande de volume persistant et le volume persistant, la règle de récupération est mise à jour pour correspondre à la règle de récupération de la classe de stockage.
  - Si la règle de récupération de la classe de stockage est `delete`, Le volume de stockage sera supprimé lorsque le volume persistant est supprimé.
- Par défaut, Astra Trident gère la demande de volume persistant et renomme le `FlexVol` et le `LUN` sur le back-end. Vous pouvez passer le `--no-manage` indicateur pour importer un volume non géré. Si vous utilisez `--no-manage`, Astra Trident n'effectue aucune opération supplémentaire sur la demande de volume persistant ou la demande de volume persistant pour le cycle de vie des objets. Le volume de stockage n'est pas supprimé lorsque le volume persistant est supprimé et d'autres opérations telles que le clone de volume et le redimensionnement de volume sont également ignorées.



Cette option est utile si vous souhaitez utiliser Kubernetes pour des workloads conteneurisés, mais que vous souhaitez gérer le cycle de vie du volume de stockage en dehors de Kubernetes.

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

## Importer un volume

Vous pouvez utiliser `tridentctl import` pour importer un volume.

### Étapes

1. Création du fichier de demande de volume persistant (par exemple, `pvc.yaml`) Qui sera utilisé pour créer la PVC. Le fichier PVC doit inclure `name`, `namespace`, `accessModes`, et `storageClassName`. Vous pouvez également spécifier `unixPermissions` Dans votre définition de 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'incluez pas de paramètres supplémentaires tels que le nom du volume persistant ou la taille du volume. Cela peut entraîner l'échec de la commande d'importation.

2. Utilisez le `tridentctl import` Commande permettant de spécifier le nom du système back-end Astra Trident contenant le volume et le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume, chemin Cloud Volumes Service). Le `-f` L'argument est requis pour spécifier le chemin d'accès au fichier PVC.

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

## Exemples

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

### NAS ONTAP et FlexGroup NAS ONTAP

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



- Le `ontap-nas-economy` le pilote ne peut pas importer et gérer les qtrees.
- Le `ontap-nas` et `ontap-nas-flexgroup` les pilotes n'autorisent pas les noms de volumes dupliqués.

Chaque volume créé avec le `ontap-nas` Le pilote est un FlexVol sur le cluster ONTAP. Importation de volumes FlexVol avec `ontap-nas` le pilote fonctionne de la même manière. Une FlexVol qui existe déjà sur un cluster ONTAP peut être importée en tant que `ontap-nas` PVC. De même, les volumes FlexGroup peuvent être importés en tant que `ontap-nas-flexgroup` ESV.

### Exemples NAS de ONTAP

Voici un exemple d'importation de volume géré et de volume non géré.

## Volume géré

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

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

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

## Volume non géré

Lorsque vous utilisez le `--no-manage` Argument, Astra Trident ne renomme pas le volume.

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

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

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

## SAN ONTAP

ASTRA Trident prend en charge l'importation de volumes à l'aide du `ontap-san` conducteur. L'importation de volume n'est pas prise en charge à l'aide du `ontap-san-economy` conducteur.

ASTRA Trident peut importer des volumes ONTAP SAN FlexVols qui contiennent un LUN unique. Ceci est cohérent avec le `ontap-san` Pilote, qui crée un FlexVol pour chaque demande de volume persistant et une LUN au sein de la FlexVol. ASTRA Trident importe le FlexVol et l'associe à la définition de l'ESV.

## Exemples de SAN ONTAP

Voici un exemple d'importation de volume g  r   et de volume non g  r  .

### Volume g  r  

Pour les volumes g  r  s, Astra Trident renomme le syst  me FlexVol en `pvc-<uuid>`. Formatez et la LUN au sein du FlexVol    `lun0`.

L'exemple suivant importe le `ontap-san-managed` FlexVol pr  sent sur le `ontap_san_default` back-end :

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

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

### Volume non g  r  

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

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

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

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

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

## Élément

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



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

## Exemple d'élément

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

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

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

## Google Cloud Platform

ASTRA Trident prend en charge l'importation de volumes à l'aide du `gcp-cvs` conducteur.



Pour importer un volume soutenu par NetApp Cloud Volumes Service dans Google Cloud Platform, identifiez le volume par son chemin d'accès au volume. Le chemin du volume est la partie du chemin d'exportation du volume après le `:/`. Par exemple, si le chemin d'exportation est `10.0.0.1:/adroit-jolly-swift`, le chemin du volume est `adroit-jolly-swift`.

## Exemple de Google Cloud Platform

L'exemple suivant importe un `gcp-cvs` volume sur le back-end `gcpcvs_YEppr` avec le chemin de volume de `adroit-jolly-swift`.

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-file> -n trident
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |      BACKEND UUID      | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true      |
+-----+-----+-----+
+-----+-----+-----+-----+
```

## Azure NetApp Files

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



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

## Exemple Azure NetApp Files

L'exemple suivant importe un `azure-netapp-files` volume sur le back-end `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      |
+-----+-----+-----+
+-----+-----+-----+-----+
```

# Partager un volume NFS entre les espaces de noms

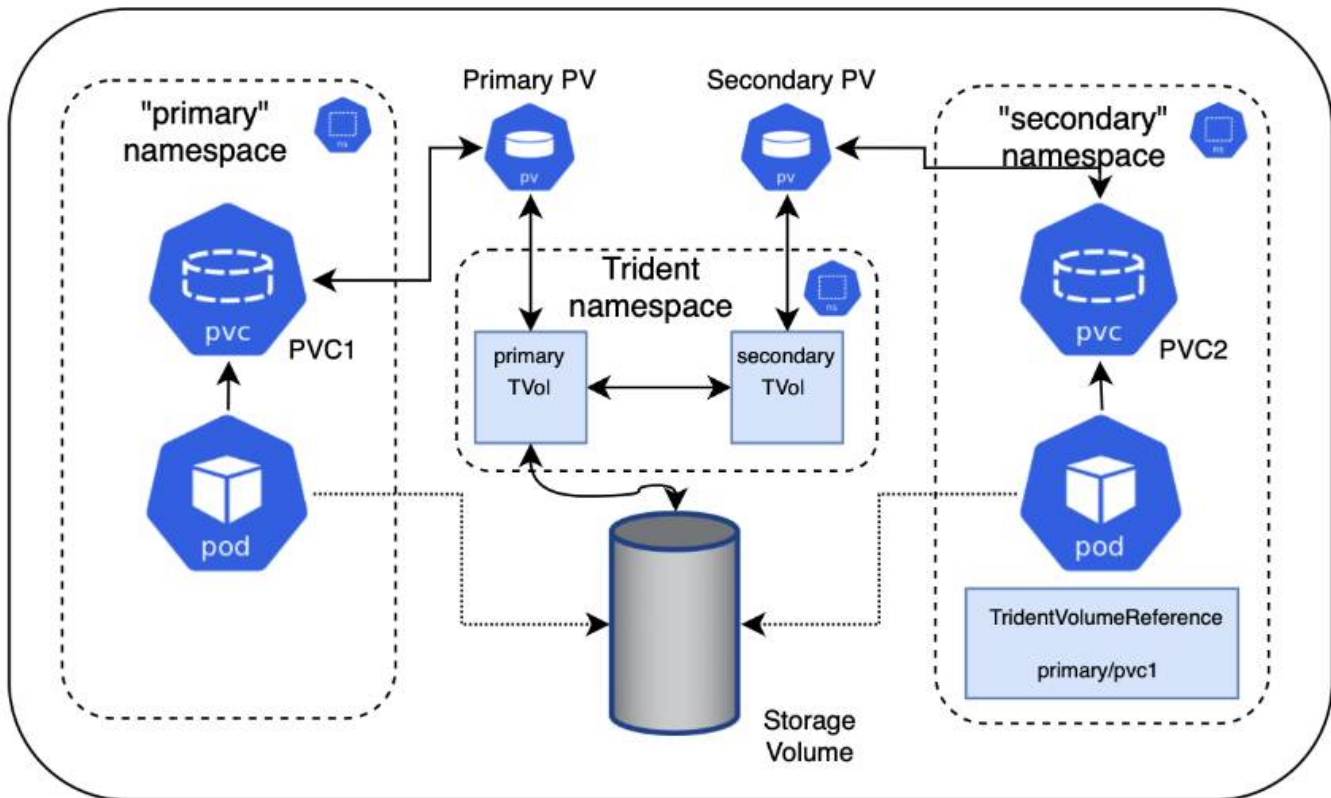
Avec Astra Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

## Caractéristiques

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

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

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



## Démarrage rapide

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

**1**

### Configurez la demande de volume persistant source pour partager le volume

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

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

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

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

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

**4****Créer le PVC subalterne dans l'espace de noms de destination**

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

**Configurer les espaces de noms source et de destination**

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

**Étapes**

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

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

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





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

2. **Cluster admin:** Créez le rôle personnalisé et kubeconfig pour accorder l'autorisation au propriétaire de l'espace de noms de destination de créer le CR TridentVolumeReference dans l'espace de noms de destination.
3. **Propriétaire de l'espace de noms de destination :** Créez un CR TridentVolumeReference dans l'espace de noms de destination qui fait référence à l'espace de noms source pvc1.

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

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

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



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

## Résultats

Découvrez Astra Trident `shareFromPVC` Annotation sur la demande de volume persistant de destination et création du volume persistant de destination en tant que volume subalterne, sans ressource de stockage correspondant au volume persistant source et partage la ressource de stockage PV source. La demande de volume persistant et la demande de volume persistant de destination apparaissent comme normales.

## Supprimer un volume partagé

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

## Utiliser `tridentctl get` pour interroger des volumes subordonnés

À l'aide du `tridentctl` vous pouvez exécuter l' `get` commande pour obtenir des volumes subordonnés. Pour plus d'informations, consultez le lien [../trident-Reference/tridentctl.html](https://trident-reference.sigs.k8s.io/tridentctl.html) `tridentctl` commandes et options].

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

Alarmes :

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

## Limites

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

## Pour en savoir plus

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

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

## Réplication de volumes à l'aide de SnapMirror

À l'aide d'Astra Control Provisioner, vous pouvez créer des relations de miroir entre un

volume source sur un cluster et le volume de destination sur le cluster peering pour la réplication des données pour la reprise après incident. Vous pouvez utiliser une définition de ressource personnalisée (CRD) avec un espace de nom pour effectuer les opérations suivantes :

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

## Conditions préalables à la réplication

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

### Clusters ONTAP

- **Astra Control Provisioner** : Astra Control Provisioner version 23.10 ou ultérieure doit exister sur les clusters Kubernetes source et de destination qui utilisent ONTAP en tant que backend.
- **Licences** : les licences asynchrones de SnapMirror ONTAP utilisant le bundle protection des données doivent être activées sur les clusters ONTAP source et cible. Pour plus d'informations, reportez-vous à la section ["Présentation des licences SnapMirror dans ONTAP"](#) .

### Peering

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



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

- **Astra Control Provisioner et SVM** : les SVM distants à peering doivent être disponibles pour Astra Control Provisioner sur le cluster de destination.

### Pilotes pris en charge

- La réplication de volume est prise en charge pour les pilotes `ontap-nas` et `ontap-san`.

## Créer une demande de volume persistant en miroir

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

### Étapes

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

### Exemple

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

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

### Exemple

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

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

### Exemple

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

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

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

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

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

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

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

#### Exemple

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

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

### Exemple

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

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

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

### Exemple

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

ASTRA Control Provisioner vérifiera le CRD TridentMirrorRelationship et ne créera pas le volume si la relation n'existe pas. Si la relation existe, Astra Control Provisioner s'assurera que le nouveau volume FlexVol est placé sur un SVM peering avec le SVM distant défini dans le MirrorRelationship.

## États de réplication des volumes

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

- **Établi** : le PVC local est le volume de destination d'une relation miroir, et il s'agit d'une nouvelle relation.
- **Promu** : Le PVC local est ReadWrite et montable, sans relation de miroir actuellement en vigueur.

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

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

Effectuez l'étape suivante sur le cluster Kubernetes secondaire :

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

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

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

### Étapes

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

### Exemple

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

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

## Restaurer une relation de miroir après un basculement

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

### Étapes

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

## Opérations supplémentaires

ASTRA Control Provisioner prend en charge les opérations suivantes sur les volumes principal et secondaire :

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

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

#### Étapes

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

### Redimensionner une PVC en miroir, principale ou secondaire

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

### Supprimer la réplication d'une demande de volume persistant

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

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

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

Le mécanisme de provisionnement Astra Control vérifie si des demandes de volume persistant sont répliquées et libère la relation de réplication avant toute tentative de suppression du volume.

### Supprimer une TMR

La suppression d'une TMR d'un côté d'une relation symétrique entraîne la transition de la TMR restante vers l'état *promu* avant que Astra Control Provisioner ne termine la suppression. Si la TMR sélectionnée pour la suppression est déjà à l'état *promoted*, il n'y a pas de relation miroir existante et la TMR sera supprimée et Astra Control Provisioner 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 doit utiliser une nouvelle TMR avec un état de réplication *établi* volume lors de la création de la nouvelle relation miroir.

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

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

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

Vous pouvez utiliser un CRD pour effectuer une mise à jour SnapMirror sans qu'Astra Control ne dispose



d'une connectivité directe au cluster ONTAP. Reportez-vous à l'exemple de format de TridentActionMirrorUpdate suivant :

### Exemple

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

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

## Activez le mécanisme de provisionnement Astra Control

Les versions 23.10 et ultérieures de Trident comprennent l'option d'utiliser Astra Control Provisioner qui permet aux utilisateurs d'Astra Control sous licence d'accéder à des fonctionnalités avancées de provisionnement du stockage. ASTRA Control Provisioner offre cette fonctionnalité étendue en plus des fonctionnalités standard d'Astra Trident CSI. Vous pouvez utiliser cette procédure pour activer et installer Astra Control Provisioner.

Votre abonnement à Astra Control Service inclut automatiquement la licence pour l'utilisation d'Astra Control Provisioner.

Dans les prochaines mises à jour d'Astra Control, Astra Control Provisioner remplacera Astra Trident en tant que mécanisme de provisionnement et d'orchestration du stockage, et sera obligatoire pour l'utilisation d'Astra Control. Pour cette raison, il est fortement recommandé aux utilisateurs d'Astra Control d'activer Astra Control Provisioner. ASTRA Trident continuera à rester open source et sera publié, maintenu, pris en charge et mis à jour avec le nouveau CSI et d'autres fonctionnalités de NetApp.

### Comment savoir si j'ai besoin d'activer Astra Control provisionner ?

Si vous ajoutez un cluster à Astra Control Service qui ne possède pas encore Astra Trident, le cluster sera marqué comme `Eligible`. Après "[Ajoutez le cluster à Astra Control](#)", Astra Control Provisioner sera automatiquement activé.

Si votre cluster n'est pas marqué `Eligible`, il sera marqué `Partially eligible` pour l'une des raisons suivantes :

- Il utilise une version plus ancienne d'Astra Trident
- Il utilise une Astra Trident 23.10 qui ne dispose pas encore de l'option de provisionnement activée
- Il s'agit d'un type de cluster qui ne permet pas l'activation automatique

`Partially eligible` Dans certains cas, utilisez ces instructions pour activer manuellement le mécanisme de provisionnement Astra Control pour votre cluster.

**CLUSTER**

Choose an Azure Kubernetes Service cluster to enable application data management and the Astra Control storage operator.

⚠ Some Kubernetes cluster(s) below have private networking. [Learn more](#)

Cluster	Location	Eligibility
<input type="radio"/> sandbox-ragnarok-aks-02 ✓	centraluseup	✓ Eligible
<input type="radio"/> sandbox-ragnarok-aks-03	centraluseup	ⓘ Partially eligible
<input type="radio"/> sandbox-rstephe2-aks-01	centraluseup	✓ Eligible

Configuration required  
Failed to detect Astra Control Provisioner on cluster

← Back   Next →

### Avant d'activer Astra Control Provisioner

Si vous disposez d'une Astra Trident sans Astra Control Provisioner et que vous souhaitez activer Astra Control Provisioner, effectuez d'abord les opérations suivantes :

- **Si vous avez installé Astra Trident, vérifiez que sa version se trouve dans une fenêtre à quatre versions :** Vous pouvez effectuer une mise à niveau directe vers Astra Trident 24.02 avec Astra Control Provisioner si votre Astra Trident se trouve dans une fenêtre à quatre versions de la version 24.02. Par exemple, vous pouvez effectuer une mise à niveau directe d'Astra Trident 23.04 vers la version 24.02.
- **Confirmez que votre cluster a une architecture système AMD64 :** l'image Astra Control Provisioner est fournie dans les architectures CPU AMD64 et ARM64, mais seul AMD64 est pris en charge par Astra Control.

### Étapes

1. Accédez au registre d'images NetApp Astra Control :
  - a. Connectez-vous à l'interface utilisateur d'Astra Control Service et enregistrez votre ID de compte Astra Control.
    - i. Sélectionnez l'icône de figure en haut à droite de la page.
    - ii. Sélectionnez **accès API**.
    - iii. Notez votre ID de compte.

- b. A partir de la même page, sélectionnez **générer jeton API** et copiez la chaîne de jeton API dans le presse-papiers et enregistrez-la dans votre éditeur.
- c. Connectez-vous au registre Astra Control à l'aide de la méthode de votre choix :

```
docker login cr.astra.netapp.io -u <account-id> -p <api-token>
```

```
crane auth login cr.astra.netapp.io -u <account-id> -p <api-token>
```

2. (Registres personnalisés uniquement) Suivez ces étapes pour déplacer l'image vers votre registre personnalisé. Si vous n'utilisez pas de registre, suivez les étapes de l'opérateur Trident dans le [section suivante](#).



Vous pouvez utiliser Podman à la place de Docker pour les commandes suivantes. Si vous utilisez un environnement Windows, PowerShell est recommandé.

## Docker

- a. Extrayez l'image Astra Control Provisioner du Registre :



L'image extraite ne prend pas en charge plusieurs plates-formes et ne prend en charge que la même plate-forme que l'hôte qui a extrait l'image, comme Linux AMD64.

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
--platform <cluster platform>
```

Exemple :

```
docker pull cr.astra.netapp.io/astra/trident-acp:24.02.0
--platform linux/amd64
```

- b. Marquer l'image :

```
docker tag cr.astra.netapp.io/astra/trident-acp:24.02.0
<my_custom_registry>/trident-acp:24.02.0
```

- c. Envoyez l'image vers votre registre personnalisé :

```
docker push <my_custom_registry>/trident-acp:24.02.0
```

## Grue

- a. Copiez le manifeste Astra Control Provisioner dans votre registre personnalisé :

```
crane copy cr.astra.netapp.io/astra/trident-acp:24.02.0
<my_custom_registry>/trident-acp:24.02.0
```

3. Déterminez si la méthode d'installation d'Astra Trident d'origine utilisait un.
4. Activez Astra Control Provisioner dans Astra Trident en utilisant la méthode d'installation que vous avez utilisée initialement :

## Opérateur Astra Trident

- a. "Téléchargez le programme d'installation d'Astra Trident et extrayez-le".
- b. Si vous n'avez pas encore installé Astra Trident ou si vous avez supprimé l'opérateur de votre déploiement Astra Trident d'origine :

- i. Créez le CRD :

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.1
6.yaml
```

- ii. Créez l'espace de noms trident (`kubectl create namespace trident`) ou confirmez que l'espace de noms trident existe toujours (`kubectl get all -n trident`). Si l'espace de noms a été supprimé, créez-le à nouveau.

- c. Mettez à jour Astra Trident vers la version 24.02.0 :



Pour les clusters exécutant Kubernetes 1.24 ou version antérieure, utilisez `bundle_pre_1_25.yaml`. Pour les clusters exécutant Kubernetes 1.25 ou version ultérieure, utilisez `bundle_post_1_25.yaml`.

```
kubectl -n trident apply -f trident-installer/deploy/<bundle-
name.yaml>
```

- d. Vérifiez que Astra Trident est en cours d'exécution :

```
kubectl get torc -n trident
```

Réponse :

NAME	AGE
trident	21m

- e. si vous avez un registre qui utilise des secrets, créez un secret à utiliser pour extraire l'image Astra Control Provisioner :

```
kubectl create secret docker-registry <secret_name> -n trident
--docker-server=<my_custom_registry> --docker-username=<username>
--docker-password=<token>
```

- f. Modifiez la CR TridentOrchestrator et apportez les modifications suivantes :

```
kubectl edit torc trident -n trident
```

- i. Définissez un emplacement de Registre personnalisé pour l'image Astra Trident ou extrayez-le du Registre Astra Control ou (`tridentImage: <my_custom_registry>/trident:24.02.0` `tridentImage: netapp/trident:24.02.0`).
- ii. Activez le mécanisme de provisionnement Astra Control (`enableACP: true`).
- iii. Définissez l'emplacement de registre personnalisé pour l'image Astra Control Provisioner ou extrayez-le du registre Astra Control ou (`acpImage: <my_custom_registry>/trident-acp:24.02.0` `acpImage: cr.astra.netapp.io/astra/trident-acp:24.02.0`).
- iv. Si vous avez établi [secrets d'extraction d'image](#) précédemment dans cette procédure, vous pouvez les définir ici (`imagePullSecrets: - <secret_name>`). Utilisez le même nom secret que celui que vous avez établi lors des étapes précédentes.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: <registry>/trident:24.02.0
  enableACP: true
  acpImage: <registry>/trident-acp:24.02.0
  imagePullSecrets:
    - <secret_name>
```

- g. Enregistrez et quittez le fichier. Le processus de déploiement commence automatiquement.
- h. Vérifiez que l'opérateur, le déploiement et les réplicateurs sont créés.

```
kubectl get all -n trident
```



Il ne doit y avoir que **une instance** de l'opérateur dans un cluster Kubernetes. Ne créez pas plusieurs déploiements de l'opérateur Astra Trident.

- i. Vérifiez que le `trident-acp` conteneur est en cours d'exécution et que `acpVersion` l'état est `24.02.0 Installed`:

```
kubectl get torc -o yaml
```

Réponse :

```

status:
  acpVersion: 24.02.0
  currentInstallationParams:
    ...
    acpImage: <registry>/trident-acp:24.02.0
    enableACP: "true"
    ...
  ...
status: Installed

```

### tridentctl

- "Téléchargez le programme d'installation d'Astra Trident et extrayez-le".
- "Si vous disposez d'une Astra Trident, désinstallez-la du cluster qui l'héberge".
- Installer Astra Trident avec Astra Control Provisioner activé (`--enable-acp=true`):

```

./tridentctl -n trident install --enable-acp=true --acp
-image=mycustomregistry/trident-acp:24.02

```

- Vérifiez que le mécanisme de provisionnement Astra Control a été activé :

```

./tridentctl -n trident version

```

Réponse :

```

+-----+-----+-----+ | SERVER
VERSION | CLIENT VERSION | ACP VERSION | +-----+
+-----+-----+-----+ | 24.02.0 | 24.02.0 | 24.02.0. |
+-----+-----+-----+

```

### Gouvernail

- Si vous avez installé Astra Trident 23.07.1 ou une version antérieure, "désinstaller" l'opérateur et d'autres composants.
- Si votre cluster Kubernetes s'exécute sur la version 1.24 ou antérieure, supprimez la psp :

```

kubectl delete psp tridentoperatorpod

```

- Ajout du référentiel Astra Trident Helm :

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

d. Mettre à jour le graphique Helm :

```
helm repo update netapp-trident
```

Réponse :

```
Hang tight while we grab the latest from your chart
repositories...
...Successfully got an update from the "netapp-trident" chart
repository
Update Complete. ☐Happy Helming!☐
```

e. Répertoire les images :

```
./tridentctl images -n trident
```

Réponse :

```
| v1.28.0           | netapp/trident:24.02.0|
|                  | docker.io/netapp/trident-
autosupport:24.02|
|                  | registry.k8s.io/sig-storage/csi-
provisioner:v4.0.0|
|                  | registry.k8s.io/sig-storage/csi-
attacher:v4.5.0|
|                  | registry.k8s.io/sig-storage/csi-
resizer:v1.9.3|
|                  | registry.k8s.io/sig-storage/csi-
snapshotter:v6.3.3|
|                  | registry.k8s.io/sig-storage/csi-node-
driver-registrar:v2.10.0 |
|                  | netapp/trident-operator:24.02.0 (optional)
```

f. Vérifier que trident-Operator 24.02.0 est disponible :

```
helm search repo netapp-trident/trident-operator --versions
```

Réponse :



NAME	CHART VERSION	APP VERSION	
DESCRIPTION			
netapp-trident/trident-operator	100.2402.0	24.02.0	A

g. Utilisez `helm install` et exécutez l'une des options suivantes qui incluent ces paramètres :

- Un nom pour votre emplacement de déploiement
- Version d'Astra Trident
- Nom de l'image Astra Control Provisioner
- Indicateur d'activation du provisionneur
- (Facultatif) Un chemin de registre local. Si vous utilisez un registre local, votre "[Images Trident](#)" peut se trouver dans un registre ou dans des registres différents, mais toutes les images CSI doivent se trouver dans le même registre.
- Espace de noms Trident

### Options

- Images sans registre

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=cr.astra.netapp.io/astra/trident-
acp:24.02.0 --set enableACP=true --set operatorImage=netapp/trident-
operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

- Images dans un ou plusieurs registres

```
helm install trident netapp-trident/trident-operator --version
100.2402.0 --set acpImage=<your-registry>:<acp image> --set
enableACP=true --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=docker.io/netapp/trident-autosupport:24.02
--set tridentImage=netapp/trident:24.02.0 --namespace trident
```

Vous pouvez utiliser `helm list` pour vérifier les détails de l'installation tels que le nom, l'espace de noms, le graphique, l'état, la version de l'application, et numéro de révision.

Si vous rencontrez des problèmes pour déployer Trident à l'aide d'Helm, exécutez cette commande pour désinstaller complètement Astra Trident :

```
./tridentctl uninstall -n trident
```

Ne pas "[Retirez complètement les CRD Astra Trident](#)" dans le cadre de votre désinstallation avant de tenter à nouveau d'activer Astra Control Provisioner.

## Résultat

La fonctionnalité Astra Control Provisioner est activée et vous pouvez utiliser toutes les fonctions disponibles pour la version que vous exécutez.

Une fois Astra Control Provisioner installé, le cluster qui héberge le provisionneur dans l'interface utilisateur Astra Control affiche un `ACP version` champ plutôt que `Trident version` et le numéro de version installée actuel.

~ CLUSTER STATUS

✓ Available

Version v1.24.9+rke2r2	Managed 2024/03/15 17:32 UTC	Kube-system namespace UID <div></div>	ACP Version <div></div>
Private route identifier <div>...</div>	Cloud instance private	Default bucket astra-bucket1 (inherited)	

Overview

Namespaces

Storage

Activity

## Pour en savoir plus

- "[Documentation sur les mises à niveau d'Astra Trident](#)"

# Utiliser la topologie CSI

Astra Trident peut créer et relier de façon sélective des volumes aux nœuds présents dans un cluster Kubernetes en utilisant le "[Fonction de topologie CSI](#)".

## Présentation

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



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

Kubernetes propose deux modes de liaison de volumes :

- Avec `VolumeBindingMode` réglé sur `Immediate`, Astra Trident crée le volume sans la reconnaissance de la topologie. La liaison de volumes et le provisionnement dynamique sont gérés au moment de la création de la demande de volume persistant. Il s'agit de la valeur par défaut `VolumeBindingMode` et convient aux clusters qui n'appliquent pas les contraintes de topologie. Les volumes persistants sont créés

sans dépendance vis-à-vis des exigences de planification du pod demandeur.

- Avec `VolumeBindingMode` réglé sur `WaitForFirstConsumer`, La création et la liaison d'un volume persistant pour une demande de volume persistant sont retardées jusqu'à ce qu'un pod qui utilise la demande de volume persistant soit planifié et créé. De cette façon, les volumes sont créés pour répondre aux contraintes de planification appliquées en fonction des besoins de topologie.



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

### Ce dont vous avez besoin

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

- Un cluster Kubernetes exécutant un **"Version Kubernetes prise en charge"**

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

- Les nœuds du cluster doivent avoir des étiquettes qui permettent la prise en charge de la topologie (`topology.kubernetes.io/region` et `topology.kubernetes.io/zone`). Ces étiquettes **doivent être présentes sur les nœuds du cluster** avant d'installer Astra Trident pour qu'Astra Trident soit compatible avec la topologie.

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

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

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

Voici un exemple de définition de back-end :

## YAML

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

## JSON

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



`supportedTopologies` sert à fournir une liste de régions et de zones par backend. Ces régions et ces zones représentent la liste des valeurs admissibles qui peuvent être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble de régions et de zones qu'il fournit en back-end, Astra Trident crée un volume en interne.

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
    region: Iowa-DC
    zone: Iowa-DC-A
    supportedTopologies:
    - topology.kubernetes.io/region: us-central1
      topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
    region: Iowa-DC
    zone: Iowa-DC-B
    supportedTopologies:
    - topology.kubernetes.io/region: us-central1
      topology.kubernetes.io/zone: us-central1-b

```

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

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

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

Voir l'exemple suivant :

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  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 de classe de stockage décrite ci-dessus, `volumeBindingMode` est défini sur `WaitForFirstConsumer`. Les demandes de volume persistant demandées pour cette classe de stockage ne seront pas traitées tant qu'elles ne seront pas référencées dans un pod. Et, `allowedTopologies` fournit les zones et la région à utiliser. Le `netapp-san-us-east1` `StorageClass` crée des ESV sur le `san-backend-us-east1` système back-end défini ci-dessus.

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

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

Voir l'exemple `spec` ci-dessous :

```

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

```

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

```

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

```

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



```

apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
        - weight: 1
          preference:
            matchExpressions:
              - key: topology.kubernetes.io/zone
                operator: In
                values:
                  - us-east1-a
                  - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
    - name: vol1
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: vol1
          mountPath: /data/demo
      securityContext:
        allowPrivilegeEscalation: false

```

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

Voir le résultat suivant :

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

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

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

## Trouvez plus d'informations

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

## Travailler avec des instantanés

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

## Présentation

Le snapshot de volume est pris en charge par `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, et `azure-netapp-files` pilotes.

### Avant de commencer

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

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



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

## Créer un snapshot de volume

### Étapes

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

### Exemple

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

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

### Exemples

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

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

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

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

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

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

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

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

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



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

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

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

## Importer un instantané de volume

ASTRA Trident prend en charge le "[Processus Snapshot préprovisionné Kubernetes](#)" pour permettre à l'administrateur du cluster de créer un VolumeSnapshotContent Objet et importation de snapshots créés en dehors d'Astra Trident.

### Avant de commencer

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

### Étapes

1. **Cluster admin:** Créer un VolumeSnapshotContent objet qui fait référence au snapshot back-end. Cela lance le workflow de snapshot dans Astra Trident.
  - Spécifiez le nom du snapshot back-end dans annotations comme `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
  - Spécifiez `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` dans `snapshotHandle`. Il s'agit des seules informations fournies à Astra Trident par le Snapshot externe du `ListSnapshots` appel.



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

### Exemple

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

```

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

```

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

#### Exemple

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

```

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

```

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

#### Exemple

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

```

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

```

## Restaurez les données de volume à l'aide de snapshots

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

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

```

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

```



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

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



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

## Restauration de volumes sur place à partir d'un snapshot

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

ASTRA Control Provisioner prend en charge la restauration Snapshot sur `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files` `gcp-cvs`, et `solidfire-san` pilotes.

### Avant de commencer

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

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

```
kubectl get pvc
```

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

```
kubectl get vs
```

### Étapes

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

```
cat tasr-pvc1-snapshot.yaml

apiVersion: v1
kind: TridentActionSnapshotRestore
metadata:
  name: this-doesnt-matter
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

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

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

tridentactionsnapshotrestore.trident.netapp.io/this-doesnt-matter
created
```

### Résultats

ASTRA Control Provisioner restaure les données à partir du snapshot. Vous pouvez vérifier l'état de la restauration des snapshots.



```
kubectl get tasr -o yaml

apiVersion: v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: this-doesnt-matter
    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, Astra Control Provisioner ne réessaiera pas automatiquement l'opération en cas de panne. Vous devrez effectuer à nouveau l'opération.
- Les utilisateurs Kubernetes sans accès administrateur peuvent avoir à obtenir l'autorisation de l'administrateur pour créer une CR ASR dans l'espace de noms de leur application.

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

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

## Supprimez un volume persistant avec les snapshots associés

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

## Déployer un contrôleur de snapshot de volume

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

## Étapes

### 1. Création de CRD de snapshot de volume.

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

### 2. Créer le contrôleur de snapshot.

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



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

## Liens connexes

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

## 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.