Réaliser des opérations de volume

Astra Trident

NetApp
April 16, 2024

Sommaire

Réaliser des opérations de volume

Découvrez les fonctionnalités d'Astra Trident pour la gestion de vos volumes.

- "Utiliser la topologie CSI"
- "Travailler avec des instantanés"
- "Développement des volumes"
- "Importer des volumes"

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". 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` réglez 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 qui en fait la demande.
- Avec `VolumeBindingMode` réglez 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:"1e11e4a2108024935ecfcb2912226cedeafd99df",
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:"1e11e4a2108024935ecfcb2912226cedeafd99df",
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","kube
rnetes.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","kube
rnetes.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","kube
rnetes.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 :

```
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 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

Vous pouvez créer des copies Snapshot de volume Kubernetes VolumeSnapshot (copie de volume) de volumes persistants pour conserver des copies instantanées de volumes Trident. Vous pouvez également créer un nouveau volume, également appelé *clone*, à partir d'un snapshot de volume existant. 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). 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 Déploiement d'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.

Étape 1 : créer un `VolumeSnapshotClass`

Cet exemple crée une classe de snapshot de volume.

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

Le `driver` Indique le conducteur CSI d'Astra Trident. `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.

Pour plus d'informations, consultez le lien :../trident-Reference/objects.html#kubernetes-volumesnapshotclass-objects[`VolumeSnapshotClass`].

Étape 2 : création d'un snapshot d'un volume persistant existant

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
```

Dans cet exemple, le snapshot est créé pour une demande de volume persistant nommée `pvc1` et le nom du snapshot est défini sur `pvc1-snap`.

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

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

Cela a créé un `VolumeSnapshot` objet. Un instantané VolumeSnapshot est similaire à une demande de volume persistant et est associé à une `VolumeSnapshotContent` objet qui représente le snapshot réel.

Il est possible d'identifier le `VolumeSnapshotContent` objet pour le `pvc1-snap` VolumeSnapshot en le décrivant.

```
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
.
.
```

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.

Étape 3 : création de demandes de volume persistant à partir de copies Snapshot VolumeCas

Dans cet exemple, vous créez une demande de volume persistant à l'aide d'un snapshot :

```
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
```

`dataSource` La montre que la demande de volume persistant doit être créée à l'aide d'un Snapshot VolumeSnapshot nommé `pvc1-snap` comme source des données. Cela demande à Astra Trident de créer un volume persistant à partir du snapshot. Une fois la demande de volume persistant créée, elle peut être connectée à un pod et utilisée comme n'importe quel autre PVC.

> 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 ». Pour supprimer le volume Astra Trident, il est nécessaire de supprimer les snapshots du volume.

Déploiement d'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.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. Créer le contrôleur snapshot dans l'espace de noms souhaité. Modifiez les manifestes YAML ci-dessous pour modifier l'espace de noms.

```
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
```

Liens connexes

- "Snapshots de volume"
- "VolumeSnapshotClass"

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.

Présentation

L'extension d'un volume persistant iSCSI comprend les étapes suivantes :

- Modification de la définition de classe de stockage pour définir le `allowVolumeExpansion` champ à

`true`.

- Modification de la définition de PVC et mise à jour de `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.
- Pour redimensionner le volume persistant, vous devez le connecter à un pod. 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.

L'exemple ci-dessous montre le fonctionnement de l'extension de PV iSCSI.

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

```
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

```
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

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

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

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

Étape 4 : développez le volume persistant

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

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

Étape 5 : valider l'extension

Vous pouvez valider le 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 |
PROTOCOL |             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 1Go :

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

Étape 4 : valider l'extension

Vous pouvez valider le redimensionnement 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
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 |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| 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`.

Pilotes prenant en charge l'importation de volumes

Ce tableau décrit les pilotes qui prennent en charge l'importation de volumes et la version dans laquelle ils ont été introduits.

Conducteur	Relâchez
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04

Conducteur	Relâchez
gcp-cvs	19.04
ontap-san	19.04

Pourquoi importer des volumes ?

L'importation d'un volume dans Trident est possible dans plusieurs champs d'application :

- Conaerisation d'une application et réutilisation de son jeu de données existant
- L'utilisation d'un clone du jeu de données pour une application éphémère
- Reconstruction d'un cluster Kubernetes défaillant
- Migration des données applicatives pendant la reprise sur incident

Comment fonctionne l'importation ?

Le fichier de demande de volume persistant est utilisé par le processus d'importation des volumes pour créer la demande de volume persistant. Au minimum, le fichier PVC doit inclure les champs Nom, espace de noms, Access modes et storageClassName, comme indiqué dans l'exemple suivant.

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

Le `tridentctl` le client est utilisé pour importer un volume de stockage existant. Trident importe le volume en persistant des métadonnées de volume et en créant la demande de volume persistant et le volume persistant.

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

Pour importer un volume de stockage, spécifiez le nom du back-end Astra Trident contenant le volume, ainsi que le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume, chemin du volume CVS). Le volume de stockage doit autoriser l'accès en lecture/écriture et être accessible par le back-end Trident spécifié de l'Astra. Le `-f` L'argument de chaîne est requis et spécifie le chemin d'accès au fichier PVC YAML ou JSON.

Lorsque Astra Trident reçoit la demande de volume d'importation, la taille du volume existant est déterminée et définie dans le volume persistant. 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.

Lorsqu'un volume est importé avec le `--no-manage` Argument, Trident n'effectue aucune opération supplémentaire sur la demande de volume persistant ou la volume persistant pour le cycle de vie des objets. Trident ignore les événements PV et PVC pour `--no-manage` Objets, le volume de stockage n'est pas supprimé lors de la suppression du volume persistant. D'autres opérations, telles que le clone de volume et le redimensionnement des volumes, 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.

Trident 19.07 et les versions ultérieures traitent la pièce jointe des volumes persistants et monte le volume dans le cadre de son importation. Pour les importations utilisant les versions antérieures d'Astra Trident, il n'y aura aucune opération dans le chemin d'accès aux données et l'importation de volume ne vérifiera pas si le volume peut être monté. En cas d'erreur lors de l'importation de volumes (par exemple, la classe de stockage n'est pas correcte), vous pouvez effectuer une restauration en changeant la règle de récupération du volume persistant à `retain`, Suppression de la demande de volume persistant et nouvelle tentative de la commande d'importation du volume.

`ontap-nas` et `ontap-nas-flexgroup` importations

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.

> Pour être importé par Trident, un volume ONTAP doit être de type rw. Si un volume est de type dp, il s'agit d'un volume de destination SnapMirror ; vous devez rompre la relation du miroir avant d'importer le volume dans Trident.

> Le `ontap-nas` 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.

Par exemple, pour importer un volume nommé `managed_volume` sur un système back-end nommé `ontap_nas`, utilisez la commande suivante :

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

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

Pour importer un volume nommé `unmanaged_volume` (sur le `ontap_nas backend`), que Trident ne gère pas, utilisez la commande suivante :

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

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

Lorsque vous utilisez le `--no-manage` Argument, Trident ne renomme pas le volume ni ne valide si le volume a été monté. L'opération d'importation du volume échoue si le volume n'a pas été monté manuellement.

> Un bogue existant précédemment avec l'importation de volumes avec Unixpermissions personnalisées a été corrigé. Vous pouvez spécifier unixpermissions dans votre définition de PVC ou configuration back-end et demander à Astra Trident d'importer le volume en conséquence.

`ontap-san` importer

Astra Trident peut également importer des volumes FlexVol SAN de ONTAP contenant un seul LUN. 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. Vous pouvez utiliser le `tridentctl import` commande de la même manière que dans les autres cas :

- Inclure le nom du `ontap-san` back-end.
- Indiquez le nom de la FlexVol à importer. N'oubliez pas que cette FlexVol ne contient qu'une seule LUN qui doit être importée.
- Fournir le chemin de la définition de PVC qui doit être utilisée avec le `-f` drapeau.
- Vous avez le choix entre gérer ou non le volume persistant. Par défaut, Trident gère le volume de volume persistant et renomme la FlexVol et la LUN en back-end. Pour importer en tant que volume non géré, passez le `--no-manage` drapeau.

> Lors de l'importation d'un non géré `ontap-san` Volume, vérifiez que la LUN de la FlexVol est nommée `lun0` et est mappée sur un groupe initiateur avec les initiateurs souhaités. Astra Trident le gère automatiquement pour une importation gérée.

Astra Trident va ensuite importer le FlexVol et l'associer à la définition de la demande de volume persistant. Astra Trident renomme également le FlexVol avec le `pvc-<uuid>` Formatez et la LUN au sein du FlexVol à `lun0`.

> Il est recommandé d'importer des volumes qui n'ont pas de connexions actives existantes. Pour importer un volume activement utilisé, commencez par cloner le volume, puis procédez à l'importation.

Exemple

Pour importer `ontap-san-managed` FlexVol présent sur le `ontap_san_default` back-end, exécutez le `tridentctl import` sous forme de commande :

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

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

> Pour être importé par Astra Trident, un volume ONTAP doit être de type rw. Si un volume est de type dp, il s'agit d'un volume de destination SnapMirror. Vous devez rompre la relation du miroir avant d'importer le volume dans Astra Trident.

`element` importer

Vous pouvez importer le logiciel NetApp Element/les volumes NetApp HCI dans votre cluster Kubernetes avec

Trident. Vous avez besoin du nom de votre back-end Astra Trident, ainsi que du nom unique du volume et du fichier PVC comme arguments pour le `tridentctl import` commande.

```
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    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Le pilote d'élément prend en charge les noms de volume dupliqués. S'il existe des noms de volume dupliqués, le processus d'importation de volume de Trident renvoie une erreur. Pour contourner ce problème, clonez le volume et fournissez un nom de volume unique. Importez ensuite le volume cloné.

`gcp-cvs` importer

Pour importer un volume sauvegardé par NetApp Cloud Volumes Service dans GCP, identifiez le volume par son chemin d'accès au volume et non son nom.

Pour importer un `gcp-cvs` volume sur le back-end appelé `gcpcvs_YEppr` avec le chemin de volume de `adroit-jolly-swift`, utilisez la commande suivante :

```
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    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Le chemin du volume correspond à la partie du chemin d'exportation du volume après :/. Par exemple, si le chemin d'exportation est `10.0.0.1:/adroit-jolly-swift`, le chemin du volume est `adroit-jolly-swift`.

`azure-netapp-files` importer

Pour importer un `azure-netapp-files` volume sur le back-end appelé `azurenetappfiles_40517` avec le chemin de volume `importvol1`, exécutez la commande suivante :

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

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

Le chemin de volume du volume ANF est présent dans le chemin de montage après :/. Par exemple, si le chemin de montage est `10.0.0.2:/importvol1`, le chemin du volume est `importvol1`.

Informations sur le copyright

Informations sur les marques commerciales