



Eeguire operazioni sui volumi

Astra Trident

NetApp
April 16, 2024

Sommario

- Eeguire operazioni sui volumi 1
 - Utilizzare la topologia CSI 1
 - Lavorare con le istantanee 8
 - Espandere i volumi 13
 - Importa volumi 20

Eseguire operazioni sui volumi

Scopri le funzionalità offerte da Astra Trident per la gestione dei volumi.

- ["Utilizzare la topologia CSI"](#)
- ["Lavorare con le istantanee"](#)
- ["Espandere i volumi"](#)
- ["Importa volumi"](#)

Utilizzare la topologia CSI

Astra Trident può creare e collegare in modo selettivo volumi ai nodi presenti in un cluster Kubernetes utilizzando ["Funzionalità topologia CSI"](#). Utilizzando la funzionalità topologia CSI, l'accesso ai volumi può essere limitato a un sottoinsieme di nodi, in base alle aree geografiche e alle zone di disponibilità. I provider di cloud oggi consentono agli amministratori di Kubernetes di generare nodi basati su zone. I nodi possono essere collocati in diverse zone di disponibilità all'interno di una regione o in diverse regioni. Per facilitare il provisioning dei volumi per i carichi di lavoro in un'architettura multi-zona, Astra Trident utilizza la topologia CSI.



Scopri di più sulla funzionalità topologia CSI ["qui"](#).

Kubernetes offre due esclusive modalità di binding del volume:

- Con `VolumeBindingMode` impostare su `Immediate`, Astra Trident crea il volume senza alcuna consapevolezza della topologia. Il binding dei volumi e il provisioning dinamico vengono gestiti quando viene creato il PVC. Questa è l'impostazione predefinita `VolumeBindingMode` ed è adatto per i cluster che non applicano vincoli di topologia. I volumi persistenti vengono creati senza alcuna dipendenza dai requisiti di pianificazione del pod richiedente.
- Con `VolumeBindingMode` impostare su `WaitForFirstConsumer`, La creazione e il binding di un volume persistente per un PVC viene ritardata fino a quando un pod che utilizza il PVC viene pianificato e creato. In questo modo, i volumi vengono creati per soddisfare i vincoli di pianificazione imposti dai requisiti di topologia.



Il `WaitForFirstConsumer` la modalità di binding non richiede etichette di topologia. Questo può essere utilizzato indipendentemente dalla funzionalità topologia CSI.

Di cosa hai bisogno

Per utilizzare la topologia CSI, è necessario disporre di quanto segue:

- Un cluster Kubernetes con 1.17 o versione successiva.

```

$ kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}

```

- I nodi nel cluster devono essere dotati di etichette che introducano la consapevolezza della topologia (topology.kubernetes.io/region e topology.kubernetes.io/zone). Queste etichette **devono essere presenti sui nodi del cluster** prima dell'installazione di Astra Trident affinché Astra Trident sia consapevole della topologia.

```

$ 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"}]

```

Fase 1: Creazione di un backend compatibile con la topologia

I backend di storage Astra Trident possono essere progettati per eseguire il provisioning selettivo dei volumi in base alle zone di disponibilità. Ogni backend può portare un optional `supportedTopologies` blocco che rappresenta un elenco di zone e regioni che devono essere supportate. Per `StorageClasses` che utilizzano tale backend, un volume viene creato solo se richiesto da un'applicazione pianificata in una regione/zona supportata.

Di seguito viene riportato un esempio di definizione di backend:

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "xxxxxxxxxxxx",
  "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` viene utilizzato per fornire un elenco di regioni e zone per backend. Queste regioni e zone rappresentano l'elenco dei valori consentiti che possono essere forniti in una `StorageClass`. Per `StorageClasses` che contengono un sottoinsieme delle regioni e delle zone fornite in un backend, Astra Trident creerà un volume sul backend.

È possibile definire `supportedTopologies` anche per pool di storage. Vedere il seguente esempio:

```

{"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "nas-backend-us-central1",
"managementLIF": "172.16.238.5",
"svm": "nfs_svm",
"username": "admin",
"password": "Netapp123",
"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"}
    ]
  }
]
}

```

In questo esempio, il `region` e `zone` le etichette indicano la posizione del pool di storage. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` stabilire da dove possono essere consumati i pool di storage.

Fase 2: Definire StorageClasses che siano compatibili con la topologia

In base alle etichette della topologia fornite ai nodi del cluster, è possibile definire StorageClasses in modo da contenere informazioni sulla topologia. In questo modo verranno determinati i pool di storage che fungono da candidati per le richieste PVC effettuate e il sottoinsieme di nodi che possono utilizzare i volumi forniti da Trident.

Vedere il seguente esempio:

```
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"
```

Nella definizione di StorageClass sopra riportata, `volumeBindingMode` è impostato su `WaitForFirstConsumer`. I PVC richiesti con questa classe di storage non verranno utilizzati fino a quando non saranno referenziati in un pod. Inoltre, `allowedTopologies` fornisce le zone e la regione da utilizzare. Il `netapp-san-us-east1` StorageClass crea PVC su `san-backend-us-east1` backend definito sopra.

Fase 3: Creare e utilizzare un PVC

Con StorageClass creato e mappato a un backend, è ora possibile creare PVC.

Vedere l'esempio spec sotto:

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

La creazione di un PVC utilizzando questo manifesto comporta quanto segue:

```

$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
$ kubectl get pvc
NAME          STATUS      VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
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
  ----          -
  Normal        WaitForFirstConsumer 6s    persistentvolume-controller
  waiting for first consumer to be created before binding
  Message
  -----

```

Affinché Trident crei un volume e lo legghi al PVC, utilizza il PVC in un pod. Vedere il seguente esempio:


```

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

```

Questo podSpec indica a Kubernetes di pianificare il pod sui nodi presenti in us-east1 e scegliere tra i nodi presenti in us-east1-a oppure us-east1-b zone.

Vedere il seguente output:

```

$ 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

```

Aggiorna i back-end da includere `supportedTopologies`

I backend preesistenti possono essere aggiornati per includere un elenco di `supportedTopologies` utilizzo di `tridentctl backend update`. Ciò non influisce sui volumi già sottoposti a provisioning e verrà utilizzato solo per i PVC successivi.

Trova ulteriori informazioni

- ["Gestire le risorse per i container"](#)
- ["NodeSelector"](#)
- ["Affinità e anti-affinità"](#)
- ["Contamini e pedaggi"](#)

Lavorare con le istantanee

A partire dalla versione 20.01 di Astra Trident, è possibile creare snapshot di PVS nel livello Kubernetes. È possibile utilizzare queste snapshot per mantenere copie point-in-time dei volumi creati da Astra Trident e pianificare la creazione di volumi aggiuntivi (cloni). Lo snapshot del volume è supportato da `ontap-nas`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, e `azure-netapp-files` driver.



Questa funzione è disponibile da Kubernetes 1.17 (beta) ed è GA da 1.20. Per informazioni sulle modifiche apportate al passaggio dalla versione beta a quella GA, vedere ["il blog di release"](#). Con la laurea in GA, il v1 La versione API è stata introdotta ed è compatibile con le versioni precedenti `v1beta1` snapshot.

Di cosa hai bisogno

- La creazione di snapshot di volumi richiede la creazione di uno snapshot controller esterno e di definizioni di risorse personalizzate (CRD). Questa è la responsabilità dell'utilizzo di Kubernetes orchestrator (ad esempio: Kubeadm, GKE, OpenShift).

Se la distribuzione Kubernetes non include lo snapshot controller e i CRD, è possibile implementarli come segue.

1. Creare CRD snapshot di volume.

Per Kubernetes 1.20 e versioni successive, utilizzare gli snapshot CRD v1 con i componenti snapshot della

versione 5.0 o superiore. Per Kubernetes versioni 1.18 e 1.19, utilizzare v1beta1 con i componenti snapshot v3.0.3.

componenti v5.0

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

componenti v3.0.3

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-
snapshotter/v3.0.3/client/config/crd/snapshot.storage.k8s.io_volumes
napshots.yaml
```

2. Creare lo snapshot controller nello spazio dei nomi desiderato. Modificare i manifesti YAML riportati di seguito per modificare lo spazio dei nomi.

Per Kubernetes 1.20 e versioni successive, utilizzare la versione 5.0 o superiore. Per Kubernetes versioni 1.18 e 1.19 utilizzare v3.0.3



Non creare un controller di snapshot se si configurano snapshot di volumi on-demand in un ambiente GKE. GKE utilizza un controller di snapshot integrato e nascosto.

controller v5.0

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-5.0/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-5.0/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

controller v3.0.3

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v3.0.3/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/v3.0.3/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```



CSI Snapshotter offre un "convalida di webhook" per aiutare gli utenti a convalidare le istantanee v1beta1 esistenti e confermare che si tratta di oggetti di risorse validi. Il webhook validante etichetta automaticamente gli oggetti snapshot non validi e impedisce la creazione di oggetti non validi futuri. Il webhook di convalida viene implementato da Kubernetes orchestrator. Consultare le istruzioni per implementare manualmente il webhook di convalida "qui". Trova esempi di manifesti di snapshot non validi "qui".

Nell'esempio riportato di seguito vengono illustrati i costrutti necessari per l'utilizzo delle snapshot e viene illustrato come creare e utilizzare le istantanee.

Fase 1: Impostare un VolumeSnapshotClass

Prima di creare un'istanza del volume, impostare un collegamento: `../trident-reference/objects.html[VolumeSnapshotClass^]`.

```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.18 and
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Il driver Indica il driver CSI di Astra Trident. `deletionPolicy` può essere `Delete` oppure `Retain`. Quando è impostato su `Retain`, lo snapshot fisico sottostante sul cluster di storage viene conservato anche quando `VolumeSnapshot` oggetto eliminato.

Fase 2: Creare un'istantanea di un PVC esistente

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.18 and
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvcl-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvcl
```

Lo snapshot è in fase di creazione per un PVC denominato `pvcl` e il nome dello snapshot è impostato su `pvcl-snap`.

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

$ kubectl get volumesnapshots
NAME                AGE
pvcl-snap           50s
```

Questo ha creato un `VolumeSnapshot` oggetto. Un'istantanea `VolumeSnapshot` è analoga a un PVC ed è associata a un `VolumeSnapshotContent` oggetto che rappresenta lo snapshot effettivo.

È possibile identificare `VolumeSnapshotContent` oggetto per `pvcl-snap` `VolumeSnapshot` descrivendolo.

```

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

```

Il Snapshot Content Name identifica l'oggetto VolumeSnapshotContent che fornisce questa snapshot. Il Ready To Use Il parametro indica che l'istantanea può essere utilizzata per creare un nuovo PVC.

Fase 3: Creazione di PVC da VolumeSnapshots

Vedere l'esempio seguente per creare un PVC utilizzando uno 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: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

`dataSource` Mostra che il PVC deve essere creato utilizzando un VolumeSnapshot denominato `pvc1-snap` come origine dei dati. Questo indica ad Astra Trident di creare un PVC dall'istantanea. Una volta creato, il PVC può essere collegato a un pod e utilizzato come qualsiasi altro PVC.



Quando si elimina un volume persistente con snapshot associate, il volume Trident corrispondente viene aggiornato a uno stato di eliminazione. Per eliminare il volume Astra Trident, è necessario rimuovere le snapshot del volume.

Trova ulteriori informazioni

- ["Snapshot dei volumi"](#)
- `VolumeSnapshotClass`

Espandere i volumi

Astra Trident offre agli utenti Kubernetes la possibilità di espandere i propri volumi dopo la loro creazione. Informazioni sulle configurazioni richieste per espandere i volumi iSCSI e NFS.

Espandere un volume iSCSI

È possibile espandere un volume persistente iSCSI (PV) utilizzando il provisioning CSI.



L'espansione del volume iSCSI è supportata da `ontap-san`, `ontap-san-economy`, `solidfire-san` Driver e richiede Kubernetes 1.16 e versioni successive.

Panoramica

L'espansione di un PV iSCSI include i seguenti passaggi:

- Modifica della definizione di `StorageClass` per impostare `allowVolumeExpansion` campo a `true`.
- Modifica della definizione PVC e aggiornamento di `spec.resources.requests.storage` per riflettere le nuove dimensioni desiderate, che devono essere superiori alle dimensioni originali.
- Il collegamento del PV deve essere collegato a un pod per poter essere ridimensionato. Esistono due scenari quando si ridimensiona un PV iSCSI:
 - Se il PV è collegato a un pod, Astra Trident espande il volume sul backend dello storage, esegue di nuovo la scansione del dispositivo e ridimensiona il file system.
 - Quando si tenta di ridimensionare un PV non collegato, Astra Trident espande il volume sul backend dello storage. Dopo aver associato il PVC a un pod, Trident esegue nuovamente la scansione del dispositivo e ridimensiona il file system. Kubernetes aggiorna quindi le dimensioni del PVC dopo il completamento dell'operazione di espansione.

L'esempio seguente mostra come funziona l'espansione del PVS iSCSI.

Fase 1: Configurare `StorageClass` per supportare l'espansione dei volumi

```

$ 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

```

Per un StorageClass già esistente, modificarlo per includere allowVolumeExpansion parametro.

Fase 2: Creare un PVC con la StorageClass creata

```

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

```

Astra Trident crea un volume persistente (PV) e lo associa a questo PVC (Persistent Volume Claim).

```

$ 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

```


Fase 3: Definire un pod che colleghi il PVC

In questo esempio, viene creato un pod che utilizza `san-pvc`.

```
$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
centos-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:    centos-pod
```

Fase 4: Espandere il PV

Per ridimensionare il PV creato da 1 Gi a 2 Gi, modificare la definizione PVC e aggiornare `spec.resources.requests.storage` A 2 Gi.

```

$ 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
  ...

```

Fase 5: Convalidare l'espansione

È possibile verificare che l'espansione funzioni correttamente controllando le dimensioni del volume PVC, PV e 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    |
+-----+-----+-----+
+-----+-----+-----+-----+

```

Espandere un volume NFS

Astra Trident supporta l'espansione dei volumi per NFS PVS su cui è stato eseguito il provisioning `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `gcp-cvs`, e `azure-netapp-files` back-end.

Fase 1: Configurare StorageClass per supportare l'espansione dei volumi

Per ridimensionare un PV NFS, l'amministratore deve prima configurare la classe di storage per consentire l'espansione del volume impostando `allowVolumeExpansion` campo a `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

```

Se è già stata creata una classe di storage senza questa opzione, è possibile modificare semplicemente la classe di storage esistente utilizzando `kubectl edit storageclass` per consentire l'espansione del volume.

Fase 2: Creare un PVC con la StorageClass creata

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

Astra Trident deve creare un PV NFS 20MiB per questo PVC:

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

$ kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY   ACCESS MODES   STORAGECLASS   REASON   AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi      RWO            ontapnas      Delete   2m42s
```

Fase 3: Espandere il PV

Per ridimensionare il PV 20MiB appena creato in 1GiB, modificare il PVC e impostare `spec.resources.requests.storage` A 1 GB:

```
$ 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
  ...
```

Fase 4: Convalidare l'espansione

È possibile verificare che il ridimensionamento funzioni correttamente controllando le dimensioni del volume PVC, PV e 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     |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+

```

Importa volumi

È possibile importare volumi di storage esistenti come PV Kubernetes utilizzando `tridentctl import`.

Driver che supportano l'importazione di volumi

Questa tabella illustra i driver che supportano l'importazione di volumi e la release in cui sono stati introdotti.

Driver	Rilasciare
ontap-nas	19.04
ontap-nas-flexgroup	19.04
solidfire-san	19.04
azure-netapp-files	19.04

Driver	Rilasciare
gcp-cvs	19.04
ontap-san	19.04

Perché è necessario importare i volumi?

Esistono diversi casi di utilizzo per l'importazione di un volume in Trident:

- La creazione di un'applicazione e il riutilizzo del set di dati esistente
- Utilizzo di un clone di un set di dati per un'applicazione temporanea
- Ricostruzione di un cluster Kubernetes guasto
- Migrazione dei dati delle applicazioni durante il disaster recovery

Come funziona l'importazione?

Il file PVC (Persistent Volume Claim) viene utilizzato dal processo di importazione del volume per creare il PVC. Come minimo, il file PVC deve includere i campi name, namespace, accessModes e storageClassName, come illustrato nell'esempio seguente.

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

Il `tridentctl` client viene utilizzato per importare un volume di storage esistente. Trident importa il volume persistendo i metadati del volume e creando PVC e PV.

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

Per importare un volume di storage, specificare il nome del backend Astra Trident contenente il volume, nonché il nome che identifica in modo univoco il volume nello storage (ad esempio: ONTAP FlexVol, Volume elemento, percorso del volume CVS). Il volume di storage deve consentire l'accesso in lettura/scrittura ed essere accessibile dal backend Astra Trident specificato. Il `-f` L'argomento string è obbligatorio e specifica il percorso del file YAML o JSON PVC.

Quando Astra Trident riceve la richiesta del volume di importazione, le dimensioni del volume esistente vengono determinate e impostate nel PVC. Una volta importato il volume dal driver di storage, il PV viene creato con un ClaimRef sul PVC. La policy di recupero viene inizialmente impostata su `retain` Nel PV. Dopo

che Kubernetes ha eseguito il binding con PVC e PV, la policy di recupero viene aggiornata in modo da corrispondere alla policy di recupero della classe di storage. Se il criterio di recupero della classe di storage è `delete`, il volume di storage viene cancellato quando il PV viene cancellato.

Quando viene importato un volume con `--no-manage` Argomento: Trident non esegue operazioni aggiuntive sul PVC o sul PV per il ciclo di vita degli oggetti. Perché Trident ignora gli eventi PV e PVC per `--no-manage` Oggetti, il volume di storage non viene cancellato quando il PV viene cancellato. Vengono ignorate anche altre operazioni, come il clone del volume e il ridimensionamento del volume. Questa opzione è utile se si desidera utilizzare Kubernetes per carichi di lavoro containerizzati, ma altrimenti si desidera gestire il ciclo di vita del volume di storage al di fuori di Kubernetes.

Al PVC e al PV viene aggiunta un'annotazione che serve a doppio scopo per indicare che il volume è stato importato e se il PVC e il PV sono gestiti. Questa annotazione non deve essere modificata o rimossa.

Trident 19.07 e versioni successive gestiscono il collegamento di PVS e montano il volume come parte dell'importazione. Per le importazioni che utilizzano versioni precedenti di Astra Trident, non verranno eseguite operazioni nel percorso dei dati e l'importazione del volume non verificherà se il volume può essere montato. Se si commette un errore con l'importazione del volume (ad esempio, StorageClass non è corretta), è possibile ripristinare modificando la policy di recupero sul PV in `retain`, Eliminando PVC e PV e riprovando il comando di importazione del volume.

ontap-nas e. ontap-nas-flexgroup **importazioni**

Ogni volume creato con `ontap-nas` Driver è un FlexVol sul cluster ONTAP. Importazione di FlexVol con `ontap-nas` il driver funziona allo stesso modo. Un FlexVol già presente in un cluster ONTAP può essere importato come `ontap-nas` PVC. Allo stesso modo, è possibile importare i volumi FlexGroup come `ontap-nas-flexgroup` PVC.



Un volume ONTAP deve essere di tipo `rw` per essere importato da Trident. Se un volume è di tipo `dp`, si tratta di un volume di destinazione SnapMirror; è necessario interrompere la relazione di mirroring prima di importare il volume in Trident.



Il `ontap-nas` il driver non può importare e gestire `qtree`. Il `ontap-nas e. ontap-nas-flexgroup` i driver non consentono nomi di volumi duplicati.

Ad esempio, per importare un volume denominato `managed_volume` su un backend denominato `ontap_nas`, utilizzare il seguente comando:


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

Per importare un volume denominato `unmanaged_volume` (su `ontap_nas` backend), che Trident non gestirà, utilizzare il seguente comando:

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

Quando si utilizza `--no-manage` Argomento: Trident non rinomina il volume né convalida se il volume è stato montato. L'operazione di importazione del volume non riesce se il volume non è stato montato manualmente.



È stato risolto un bug esistente in precedenza relativo all'importazione di volumi con UnixPermissions personalizzati. È possibile specificare `unixPermissions` nella definizione PVC o nella configurazione backend e richiedere ad Astra Trident di importare il volume di conseguenza.

ontap-san importa

Astra Trident può anche importare SAN FlexVol ONTAP che contengono una singola LUN. Ciò è coerente con `ontap-san` Driver, che crea un FlexVol per ogni PVC e un LUN all'interno di FlexVol. È possibile utilizzare `tridentctl import` comando nello stesso modo degli altri casi:

- Includere il nome di `ontap-san` back-end.

- Specificare il nome del FlexVol da importare. Tenere presente che questo FlexVol contiene un solo LUN che deve essere importato.
- Fornire il percorso della definizione PVC che deve essere utilizzata con `-f` allarme.
- Scegli tra gestire il PVC o non gestirlo. Per impostazione predefinita, Trident gestirà il PVC e rinominerà il FlexVol e il LUN sul backend. Per importare come volume non gestito, passare a. `--no-manage` allarme.



Quando si importa un non gestito `ontap-san` Assicurarsi che il LUN nel FlexVol sia denominato `lun0` ed è mappato ad un igroup con gli iniziatori desiderati. Astra Trident gestisce automaticamente questa operazione per un'importazione gestita.

Astra Trident importa il FlexVol e lo associa alla definizione del PVC. Astra Trident rinomina anche FlexVol in `pvc-<uuid>` E il LUN all'interno di FlexVol a. `lun0`.



Si consiglia di importare volumi che non dispongono di connessioni attive. Se si desidera importare un volume utilizzato attivamente, clonare prima il volume, quindi eseguire l'importazione.

Esempio

Per importare `ontap-san-managed` FlexVol presente su `ontap_san_default` eseguire il backend `tridentctl import` comando come:

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



Per poter essere importato da Astra Trident, un volume ONTAP deve essere di tipo `rw`. Se un volume è di tipo `dp`, si tratta di un volume di destinazione `SnapMirror`; è necessario interrompere la relazione di mirroring prima di importare il volume in Astra Trident.

element importa

Con Trident è possibile importare il software NetApp Element/volumi NetApp HCI nel cluster Kubernetes. È necessario il nome del backend Astra Trident e il nome univoco del volume e del file PVC come argomenti per `tridentctl import` comando.

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



Il driver Element supporta nomi di volumi duplicati. Se sono presenti nomi di volumi duplicati, il processo di importazione dei volumi di Trident restituisce un errore. Come soluzione alternativa, clonare il volume e fornire un nome di volume univoco. Quindi importare il volume clonato.

gcp-cvs importa



Per importare un volume supportato da NetApp Cloud Volumes Service in GCP, identificare il volume in base al percorso del volume anziché al nome.

Per importare un `gcp-cvs` volume sul backend chiamato `gcpcvs_YEppr` con il percorso del volume di `adroit-jolly-swift`, utilizzare il seguente comando:

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



Il percorso del volume è la parte del percorso di esportazione del volume dopo `./`. Ad esempio, se il percorso di esportazione è `10.0.0.1:/adroit-jolly-swift`, il percorso del volume è `adroit-jolly-swift`.

azure-netapp-files importa

Per importare un azure-netapp-files volume sul backend chiamato `azurenetaappfiles_40517` con il percorso del volume `importvoll1`, eseguire il seguente comando:

```
$ tridentctl import volume azurenetaappfiles_40517 importvoll1 -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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



Il percorso del volume per il volume ANF è presente nel percorso di montaggio dopo `:/`. Ad esempio, se il percorso di montaggio è `10.0.0.2:/importvoll1`, il percorso del volume è `importvoll1`.

Informazioni sul copyright

Copyright © 2024 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIALIZZABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEGUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina <http://www.netapp.com/TM> sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.