使用 **Astra Trident**

Astra Trident

NetApp
April 16, 2024

目录

使用 Astra Trident

准备工作节点

Kubernetes集群中的所有工作节点都必须能够挂载为Pod配置的卷。要准备工作节点、必须根据您选择的驱动程序安装NFS或iSCSI工具。

选择合适的工具

如果要组合使用驱动程序、则应安装NFS和iSCSI工具。

NFS工具

如果您使用的是以下命令、请安装NFS工具： `ontap-nas`， `ontap-nas-economy`， `ontap-nas-flexgroup`， `azure-netapp-files`， `gcp-cvs`

iSCSI工具

如果您使用的是以下命令、请安装iSCSI工具： `ontap-san`， `ontap-san-economy`， `solidfire-san`

默认情况下、最新版本的RedHat CoreOS会安装NFS和iSCSI。

节点服务发现

Astra Trident会尝试自动检测节点是否可以运行iSCSI或NFS服务。

节点服务发现可识别已发现的服务、但无法保证服务已正确配置。相反、如果没有发现的服务、则无法保证卷挂载将失败。

查看事件

Astra Trident会为节点创建事件以标识发现的服务。要查看这些事件、请运行：

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

查看发现的服务

Astra Trident标识为Trident节点CR上的每个节点启用的服务。要查看发现的服务、请运行：

```
tridentctl get node -o wide -n <Trident namespace>
```

NFS volumes

使用适用于您的操作系统的命令安装NFS工具。确保NFS服务已在启动期间启动。

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

安装NFS工具后重新启动工作节点、以防止在将卷连接到容器时失败。

iSCSI 卷

Astra Trident可以自动建立iSCSI会话、扫描LUN、发现多路径设备、对其进行格式化并将其挂载到Pod。

iSCSI自我修复功能

对于ONTAP 系统、Astra Trident每五分钟运行一次iSCSI自我修复、以便：

1. *确定*所需的iSCSI会话状态和当前的iSCSI会话状态。
2. 将所需状态与当前状态进行比较、以确定所需的修复。Astra Trident可确定修复优先级以及提前修复的时间。
3. 需要执行*修复*才能将当前iSCSI会话状态恢复为所需的iSCSI会话状态。

自我修复活动日志位于中 `trident-main` 相应的Demonset Pod上的容器。要查看日志、必须已设置 `debug` 在Astra Trident安装期间设置为"true"。

Astra Trident iSCSI自我修复功能有助于防止：

- 在网络连接问题描述 之后可能发生的陈旧或运行不正常的iSCSI会话。如果会话已过时、则Astra Trident会等待七分钟、然后再注销以重新建立与门户的连接。

  例如、如果在存储控制器上轮换了CHAP密钥、而网络断开了连接、则旧的(*stal*) CHAP密钥可能会持续存在。自修复功能可以识别此问题、并自动重新建立会话以应用更新后的CHAP密码。

- 缺少iSCSI会话
- 缺少LUN

安装iSCSI工具

使用适用于您的操作系统的命令安装iSCSI工具。

开始之前

- Kubernetes 集群中的每个节点都必须具有唯一的 IQN 。* 这是必要的前提条件 * 。

- 如果使用RHCOS 4.5或更高版本或其他与RHEL兼容的Linux分发版、请与结合使用 `solidfire-san` 驱动程序和Element OS 12：5或更早版本、请确保中的CHAP身份验证算法设置为MD5 `/etc/iscsi/iscsid.conf`。Element 12.7提供了符合FIPS的安全CHAP算法SHA1、SHA-256和SHA3-256。

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- 使用运行RHEL/RedHat CoreOS和iSCSI PV的工作节点时、请指定 `discard` StorageClass中的mountOption、用于执行实时空间回收。请参见 "RedHat 的文档"。

RHEL 8+

1. 安装以下系统软件包：

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. 检查 iscsi-initiator-utils 版本是否为 6.2.0.877-2.el7 或更高版本：

```
rpm -q iscsi-initiator-utils
```

3. 将扫描设置为手动：

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 启用多路径：

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

确保 `etc/multipath.conf` 包含 `find_multipaths no` 下 `defaults`。

5. 请确保 `iscsid` 和 `multipathd` 正在运行：

```
sudo systemctl enable --now iscsid multipathd
```

6. 启用并启动 `iscsi`：

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. 安装以下系统软件包：

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. 检查 open-iscsi 版本是否为 2.0.877-5ubuntu2.10 或更高版本（对于双子系统）或 2.0.877-7.1ubuntu6.1 或更高版本（对于 Focal ）：

```
dpkg -l open-iscsi
```

3. 将扫描设置为手动：

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. 启用多路径：

```
sudo tee /etc/multipath.conf <<-'EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

确保 `etc/multipath.conf` 包含 `find_multipaths no` 下 `defaults`。

5. 请确保 `open-iscsi` 和 `multipath-tools` 已启用且正在运行：

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

对于Ubuntu 18.04、您必须使用发现目标端口 `iscsiadm` 启动前 `open-iscsi` 以启动iSCSI守护进程。您也可以修改 `iscsi` 要启动的服务 `iscsid` 自动。

安装iSCSI工具后重新启动工作节点、以防止在将卷连接到容器时失败。

配置后端

后端定义了 Astra Trident 与存储系统之间的关系。它告诉 Astra Trident 如何与该存储系统进行通信，以及 Astra Trident 如何从该存储系统配置卷。

Astra Trident会自动从后端提供符合存储类定义的要求的存储池。了解如何为存储系统配置后端。

- "配置 Azure NetApp Files 后端"

- "配置适用于 Google 云平台的 Cloud Volumes Service 后端"
- "配置 NetApp HCI 或 SolidFire 后端"
- "使用 ONTAP 或 Cloud Volumes ONTAP NAS 驱动程序配置后端"
- "使用 ONTAP 或 Cloud Volumes ONTAP SAN 驱动程序配置后端"
- "将 Astra Trident 与适用于 NetApp ONTAP 的 Amazon FSX 结合使用"

Azure NetApp Files

配置 **Azure NetApp Files** 后端

您可以将Azure NetApp Files (ANF)配置为Astra Trident的后端。您可以使用ANF后端连接NFS和SMB卷。

- "准备"
- "配置选项和示例"

注意事项

- Azure NetApp Files 服务不支持小于100 GB的卷。如果请求的卷较小，则 Astra Trident 会自动创建 100 GB 的卷。
- Astra Trident仅支持将SMB卷挂载到Windows节点上运行的Pod。
- Astra Trident不支持Windows ARM 架构。

准备配置**Azure NetApp Files** 后端

在配置Azure NetApp Files 后端之前、您需要确保满足以下要求。

> 如果您是首次使用Azure NetApp Files 或在新位置使用、则需要进行一些初始配置来设置Azure NetApp Files 并创建NFS卷。请参见 "Azure：设置Azure NetApp Files 并创建NFS卷"。

NFS和**SMB**卷的前提条件

配置和使用 "Azure NetApp Files" 后端，您需要满足以下要求：

- 一个容量池。请参见 "Microsoft：为Azure NetApp Files 创建容量池"。
- 委派给Azure NetApp Files 的子网。请参见 "Microsoft：将子网委派给Azure NetApp Files"。
- `subscriptionID` 从启用了Azure NetApp Files 的Azure订阅。
- `tenantID`， `clientID`，和 `clientSecret` 从 "应用程序注册" 在 Azure Active Directory 中，具有足够的 Azure NetApp Files 服务权限。应用程序注册应使用以下任一项：
  - 所有者或贡献者角色 "由Azure预定义"。
  - 答 "自定义贡献者角色" 订阅级别 (`assignableScopes`)、并具有以下权限、这些权限仅限于Astra Trident所需的权限。创建自定义角色后、 "使用Azure门户分配角色"。

```
{
    "id": "/subscriptions/<subscription-
```

```
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
    "properties": {
        "roleName": "custom-role-with-limited-perms",
        "description": "custom role providing limited permissions",
        "assignableScopes": [
            "/subscriptions/<subscription-id>"
        ],
        "permissions": [
            {
                "actions": [

"Microsoft.NetApp/netAppAccounts/capacityPools/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/read
",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/writ
e",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/dele
te",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/rea
d",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/wri
te",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/del
ete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/Get
Metadata/action",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets/r
ead",
                    "Microsoft.Network/virtualNetworks/read",
```

```
                    "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations
/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations
/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations
/delete",
                    "Microsoft.Features/features/read",
                    "Microsoft.Features/operations/read",
                    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
                ],
                "notActions": [],
                "dataActions": [],
                "notDataActions": []
            }
        ]
    }
}
```

- Azure `location` 至少包含一个 "委派子网"。自Trident 22.01日开始 `location` 参数是后端配置文件顶层的必填字段。在虚拟池中指定的位置值将被忽略。

SMB卷的其他要求

要创建SMB卷、您必须具有：

- 已配置Active Directory并连接到Azure NetApp Files。请参见 "Microsoft：创建和管理Azure NetApp Files的Active Directory连接"。
- 一个Kubernetes集群、其中包含一个Linux控制器节点以及至少一个运行Windows Server 2019的Windows工作节点。Astra Trident仅支持将SMB卷挂载到Windows节点上运行的Pod。
- 至少一个包含Active Directory凭据的Astra Trident密钥、以便Azure NetApp Files 可以向Active Directory进行身份验证。以生成密钥 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- 配置为Windows服务的CSI代理。配置 `csi-proxy`、请参见 "GitHub：CSI代理" 或 "GitHub：适用于Windows的CSI代理" 适用于在Windows上运行的Kubernetes节点。

Azure NetApp Files 后端配置选项和示例

了解ANF的NFS和SMB后端配置选项、并查看配置示例。

Astra Trident会使用后端配置(子网、虚拟网络、服务级别和位置)在请求的位置提供的容量池上创建ANF卷、并与请求的服务级别和子网匹配。

> Astra Trident 不支持手动 QoS 容量池。

后端配置选项

ANF后端提供了这些配置选项。

参数	Description	Default
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	"Azure-netapp-files"
`backendName`	自定义名称或存储后端	驱动程序名称 + "_" + 随机字符
`subscriptionID`	Azure 订阅中的订阅 ID	
`tenantID`	应用程序注册中的租户 ID	
`clientID`	应用程序注册中的客户端 ID	
`clientSecret`	应用程序注册中的客户端密钥	
`serviceLevel`	其中一个 `Standard`， `Premium`` 或 ``Ultra`	"" （随机）
`location`	要创建新卷的 Azure 位置的名称	
`resourceGroups`	用于筛选已发现资源的资源组列表	"[]" （无筛选器）
`netappAccounts`	用于筛选已发现资源的 NetApp 帐户列表	"[]" （无筛选器）
`capacityPools`	用于筛选已发现资源的容量池列表	"[]" （无筛选器，随机）
`virtualNetwork`	具有委派子网的虚拟网络的名称	""
`subnet`	委派给的子网的名称 `Microsoft.Netapp/volumes`	""
`networkFeatures`	一个卷的一组vNet功能可能是 `Basic` 或 `Standard`。网络功能并非在所有地区都可用、可能需要在订阅中启用。指定 `networkFeatures` 如果未启用此功能、则会导致卷配置失败。	""

参数	Description	Default
`nfsMountOptions`	精细控制 NFS 挂载选项。SMB卷已忽略。要使用NFS 4.1挂载卷、请包括 `nfsvers=4` 在逗号分隔的挂载选项列表中选择NFS v4.1。存储类定义中设置的挂载选项会覆盖后端配置中设置的挂载选项。	"nfsvers=3"
`limitVolumeSize`	如果请求的卷大小超过此值，则配置失败	"" （默认情况下不强制实施）
`debugTraceFlags`	故障排除时要使用的调试标志。示例、 `\{"api": false, "method": true, "discovery": true}`。除非您正在进行故障排除并需要详细的日志转储，否则请勿使用此功能。	空
`nasType`	配置NFS或SMB卷创建。选项包括 `nfs`， `smb` 或为空。默认情况下、将设置为空会将NFS卷设置为空。	`nfs`

有关网络功能的详细信息、请参见 "配置Azure NetApp Files 卷的网络功能"。

所需权限和资源

如果在创建PVC时收到"未找到容量池"错误、则您的应用程序注册可能没有关联的所需权限和资源(子网、虚拟网络、容量池)。如果启用了调试、则Astra Trident将记录创建后端时发现的Azure资源。验证是否正在使用适当的角色。

的值 `resourceGroups`， `netappAccounts`， `capacityPools`， `virtualNetwork`，和 `subnet` 可以使用短名称或完全限定名称来指定。在大多数情况下、建议使用完全限定名称、因为短名称可以与多个同名资源匹配。

。 `resourceGroups`， `netappAccounts`，和 `capacityPools` 值是指筛选器、用于将发现的一组资源限制为此存储后端可用的资源、并且可以以任意组合方式指定。完全限定名称采用以下格式：

Type	格式。
Resource group	< 资源组 >
NetApp 帐户	< 资源组 >/< NetApp 帐户 >
容量池	< 资源组 >/< NetApp 帐户 >/< 容量池 >
虚拟网络	< 资源组 >/< 虚拟网络 >
Subnet	< 资源组 >/< 虚拟网络 >/< 子网 >

卷配置

您可以通过在配置文件的特殊部分中指定以下选项来控制默认卷配置。请参见 [示例配置] 了解详细信息。

参数	Description	Default
exportRule	新卷的导出规则。 exportRule 必须是以CIDR表示法表示的任意IPv4地址或IPv4子网组合的逗号分隔列表。SMB卷已忽略。	"0.0.0.0/0
snapshotDir	控制 .snapshot 目录的可见性	false
size	新卷的默认大小	"100 克 "
unixPermissions	新卷的UNIX权限(4个八进制数字)。SMB卷已忽略。	"" （预览功能，需要在订阅中列入白名单）

示例配置

示例 **1**： 最低配置

这是绝对的最低后端配置。使用此配置， Astra Trident 会发现在已配置位置委派给 ANF 的所有 NetApp 帐户，容量池和子网，并随机将新卷放置在其中一个池和子网上。因为 nasType 省略 nfs 默认情况下适用、后端将为NFS卷配置。

当您刚开始使用 ANF 并尝试执行相关操作时，此配置是理想的选择，但实际上，您希望为所配置的卷提供更多范围界定。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
```

示例 **2**： 使用容量池筛选器的特定服务级别配置

此后端配置会将卷放置在Azure中 `eastus` 位置 `Ultra` 容量池。Astra Trident 会自动发现该位置委派给 ANF 的所有子网，并随机在其中一个子网上放置一个新卷。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

示例 **3**： 高级配置

此后端配置进一步将卷放置范围缩小为一个子网，并修改了某些卷配置默认值。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

示例**4**：虚拟池配置

此后端配置可在一个文件中定义多个存储池。如果您有多个容量池支持不同的服务级别，并且您希望在 Kubernetes 中创建表示这些服务级别的存储类，则此功能非常有用。虚拟池标签用于根据区分池 `performance`。

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
    performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
    performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
    performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

存储类定义

以下内容 `StorageClass` 定义是指上述存储池。

使用的示例定义 `parameter.selector` 字段

使用 `parameter.selector` 您可以为每个指定 `StorageClass` 用于托管卷的虚拟池。卷将在选定池中定义各个方面。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

SMB卷的示例定义

使用 `nasType`， `node-stage-secret-name`，和 `node-stage-secret-namespace`、您可以指定SMB卷并提供所需的Active Directory凭据。

示例**1**：默认命名空间上的基本配置

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

示例**2**：每个命名空间使用不同的密钥

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

示例**3**：每个卷使用不同的密钥

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: `smb` 支持SMB卷的池的筛选器。 `nasType: `nfs` 或 `nasType: `null` NFS池的筛选器。

创建后端

创建后端配置文件后，运行以下命令：

```
tridentctl create backend -f <backend-file>
```

如果后端创建失败，则后端配置出现问题。您可以运行以下命令来查看日志以确定发生原因：

```
tridentctl logs
```

确定并更正配置文件中的问题后，您可以再次运行 create 命令。

为Google Cloud后端配置Cloud Volumes Service

了解如何使用提供的示例配置将适用于Google Cloud的NetApp Cloud Volumes Service 配置为Astra Trident安装的后端。

了解适用于Google Cloud的Cloud Volumes Service 的Astra Trident支持

Astra Trident可以在两个卷中创建Cloud Volumes Service 卷之一 "服务类型"：

- **CVS-Performance**：默认的Astra Trident服务类型。这种性能优化的服务类型最适合重视性能的生产工作负载。CVS-Performance服务类型是一种硬件选项、支持的卷大小至少为100 GiB。您可以选择一个 "三个服务级别"：
  - `standard`
  - `premium`
  - `extreme`
- * CVS*：CVS服务类型提供高区域可用性、性能级别限制为中等。CVS服务类型是一个软件选项、可使用存储池支持小至1 GiB的卷。存储池最多可包含50个卷、其中所有卷都共享池的容量和性能。您可以选择一个 "两个服务级别"：
  - `standardsw`
  - `zoneredundantstandardsw`

您需要的内容

以配置和使用 "适用于 Google Cloud 的 Cloud Volumes Service" 后端，您需要满足以下要求：

- 配置了NetApp Cloud Volumes Service 的Google Cloud帐户
- Google Cloud 帐户的项目编号
- Google Cloud服务帐户 `netappcloudvolumes.admin` role

- Cloud Volumes Service 帐户的API密钥文件

后端配置选项

每个后端都会在一个 Google Cloud 区域中配置卷。要在其他区域创建卷，您可以定义其他后端。

参数	Description	Default
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	"GCP-CVS"
`backendName`	自定义名称或存储后端	驱动程序名称 + "_" + API 密钥的一部分
`storageClass`	用于指定CVS服务类型的可选参数。使用 … `software` 选择CVS服务类型。否则、Astra Trident将采用CVS-Performance服务类型(`hardware`）。	
`storagePools`	仅限CVS服务类型。用于指定用于创建卷的存储池的可选参数。	
`projectNumber`	Google Cloud 帐户项目编号。此值可在Google Cloud门户主页上找到。	
`hostProjectNumber`	如果使用共享VPC网络、则为必需项。在此情景中、`projectNumber` 是服务项目、和 `hostProjectNumber` 是主机项目。	
`apiRegion`	Astra Trident创建Cloud Volumes Service 卷的Google云区域。创建跨区域Kubernetes集群时、在中创建的卷 `apiRegion` 可用于在多个Google Cloud地区的节点上计划的工作负载。跨区域流量会产生额外成本。	
`apiKey`	Google Cloud服务帐户的API密钥 `netappcloudvolumes.admin` 角色。它包括 Google Cloud 服务帐户专用密钥文件的 JSON 格式的内容（逐字复制到后端配置文件）。	
`proxyURL`	代理服务器需要连接到CVS帐户时的代理URL。代理服务器可以是 HTTP 代理，也可以是 HTTPS 代理。对于 HTTPS 代理，将跳过证书验证，以允许在代理服务器中使用自签名证书。不支持启用了身份验证的代理服务器。	
`nfsMountOptions`	精细控制 NFS 挂载选项。	"nfsvers=3"

参数	Description	Default
limitVolumeSize	如果请求的卷大小超过此值、则配置失败。	"" （默认情况下不强制实施）
serviceLevel	新卷的CVS-Performance或CVS服务级别。CVS-Performance值为 standard， premium`或 `extreme。CVS值为 standardsw 或 zoneredundantstandardsw。	CVS-Performance默认值为"standard"。CVS默认值为"standardsw"。
network	用于Cloud Volumes Service 卷的Google云网络。	default
debugTraceFlags	故障排除时要使用的调试标志。示例、 \{"api":false, "method":true}。除非您正在进行故障排除并需要详细的日志转储，否则请勿使用此功能。	空
allowedTopologies	要启用跨区域访问、请定义StorageClass allowedTopologies 必须包括所有地区。例如： - key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

卷配置选项

您可以在中控制默认卷配置 defaults 部分。

参数	Description	Default
exportRule	新卷的导出规则。必须是以 CIDR 表示法表示的任意 IPv4 地址或 IPv4 子网组合的逗号分隔列表。	"0.0.0.0/0
snapshotDir	访问 .snapshot 目录	false
snapshotReserve	为快照预留的卷百分比	"" （接受 CVS 默认值为 0 ）
size	新卷的大小。CVS性能最小值为100 GiB。CVS最小值为1 GiB。	CVS-Performance服务类型默认为"100GiB"。CVS服务类型未设置默认值、但至少需要1 GiB。

CVS-Performance服务类型示例

以下示例提供了CVS-Performance服务类型的示例配置。

示例 **1**： 最低配置

这是使用默认CVS-Performance服务类型以及默认"标准"服务级别的最小后端配置。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
```

```
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

示例**2**：服务级别配置

此示例说明了后端配置选项、包括服务级别和卷默认值。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
```

```
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

示例3：虚拟池配置

此示例使用 `storage` 配置虚拟池和 `StorageClasses` 这是指它们。请参见 [存储类定义] 以查看存储类的定义方式。

此处为设置的所有虚拟池设置了特定的默认值 `snapshotReserve` 5%和 `exportRule` 到0.0.0.0/0。虚拟池在中进行定义 `storage` 部分。每个虚拟池都定义了自己的虚拟池 `serviceLevel`、并且某些池会覆盖默认值。虚拟池标签用于根据区分池 `performance` 和 `protection`。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
```

```
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
    performance: extreme
    protection: extra
  serviceLevel: extreme
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
    performance: extreme
    protection: standard
  serviceLevel: extreme
- labels:
    performance: premium
    protection: extra
  serviceLevel: premium
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
- labels:
    performance: premium
    protection: standard
  serviceLevel: premium
- labels:
    performance: standard
  serviceLevel: standard
```

以下StorageClass定义适用于虚拟池配置示例。使用 `parameters.selector`、您可以为每个StorageClass指定用于托管卷的虚拟池。卷将在选定池中定义各个方面。

存储类示例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
```

```
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

- 第一个StorageClass (`cvs-extreme-extra-protection`)映射到第一个虚拟池。这是唯一一个可提供极高性能且 Snapshot 预留为 10% 的池。
- 最后一个StorageClass (`cvs-extra-protection`)调用提供10%快照预留的任何存储池。Astra Trident决定选择哪个虚拟池、并确保满足快照预留要求。

CVS服务类型示例

以下示例提供了CVS服务类型的示例配置。

示例**1**：最低配置

这是使用的最低后端配置 `storageClass` 指定CVS服务类型和默认值 `standardsw` 服务级别。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

示例**2**：存储池配置

此示例后端配置使用 `storagePools` 配置存储池。

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMgJm2nHzFq2X0rqVMaHghI6ATm4DOuWx8XGWKTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IUp9CAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFhlLlljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcD/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a5OOPm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJDlhkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqswoYuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiyy7Tzg8AUqlH1Ou83XbV428jvg7kDhO7PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqP1yHNNEoRmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHoGN890pZOkCVSrqju6aUef/5KYlFCt8ew
    F/+aIxM2iQSvmWQYOvVCnhuY/F2GFaQ7d0om3decuwI0CX/xy7PjHMkLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbYMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDWnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElbz2AqnzQ3MN9uECgYAqdvdVPnKGfvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+lTImdBBFga
    NWOC6f3r2xbGXHhaWSl+nobpTuvlo56ZRJVvVk7lFMsiddzMuHH8pxfgNJemwA4P
    ThDHCejv035NNV6KyoO0tA==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

下一步是什么?

创建后端配置文件后，运行以下命令：

```
tridentctl create backend -f <backend-file>
```

如果后端创建失败，则后端配置出现问题。您可以运行以下命令来查看日志以确定发生原因：

```
tridentctl logs
```

确定并更正配置文件中的问题后，您可以再次运行 create 命令。

配置 NetApp HCI 或 SolidFire 后端

了解如何在 Astra Trident 安装中创建和使用 Element 后端。

您需要的内容

- 运行 Element 软件的受支持存储系统。
- NetApp HCI/SolidFire 集群管理员或租户用户的凭据，可用于管理卷。
- 所有 Kubernetes 工作节点都应安装适当的 iSCSI 工具。请参见 "工作节点准备信息"。

您需要了解的信息

。 `solidfire-san` 存储驱动程序支持两种卷模式：文件和块。。 `Filesystem` volumemode、Astra Trident 会创建卷并创建文件系统。文件系统类型由 StorageClass 指定。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`solidfire-san`	iSCSI	块	rwo ， rox ， rwx	无文件系统。原始块设备。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
solidfire-san	iSCSI	块	rwo ， rox ， rwx	无文件系统。原始块设备。
solidfire-san	iSCSI	文件系统	工单， ROX	xfs， ext3， ext4
solidfire-san	iSCSI	文件系统	工单， ROX	xfs， ext3， ext4

Astra Trident 在用作增强型 CSI 配置程序时使用 CHAP 。如果您使用的是 CHAP （这是 CSI 的默认设置），则无需进行进一步准备。建议显式设置 `UseCHAP` 可选择对非CSI Trident使用 CHAP。否则，请参见 "此处"。

只有适用于 Astra Trident 的传统非 CSI 框架才支持卷访问组。如果配置为在 CSI 模式下运行，则 Astra Trident 将使用 CHAP 。

如果两者都不是 `AccessGroups` 或 `UseCHAP` 设置后、将应用以下规则之一：

- 如果为默认值 `trident` 检测到访问组、使用访问组。
- 如果未检测到访问组，并且 Kubernetes 版本为 1.7 或更高版本，则会使用 CHAP 。

后端配置选项

有关后端配置选项，请参见下表：

参数	Description	Default
version		始终为 1
storageDriverName	存储驱动程序的名称	始终为 "solidfire-san"
backendName	自定义名称或存储后端	SolidFire + 存储（ iSCSI ） IP 地址
Endpoint	使用租户凭据的 SolidFire 集群的 MVIP	
SVIP	存储（ iSCSI ） IP 地址和端口	
labels	要应用于卷的一组任意 JSON 格式的标签。	"
TenantName	要使用的租户名称（如果未找到，则创建）	
InitiatorIFace	将 iSCSI 流量限制为特定主机接口	default
UseCHAP	使用 CHAP 对 iSCSI 进行身份验证	true
AccessGroups	要使用的访问组 ID 列表	查找名为 "trident " 的访问组的 ID
Types	QoS 规范	
limitVolumeSize	如果请求的卷大小超过此值，则配置失败	" （默认情况下不强制实施）

参数	Description	Default
debugTraceFlags	故障排除时要使用的调试标志。示例 ｛"api"： false ， "method "： true ｝	空

请勿使用 debugTraceFlags 除非您正在进行故障排除并需要详细的日志转储。

示例**1**：的后端配置 solidfire-san 具有三种卷类型的驱动程序

此示例显示了一个后端文件，该文件使用 CHAP 身份验证并使用特定 QoS 保证对三种卷类型进行建模。然后、您很可能会使用定义存储类来使用其中的每一种 IOPS storage class参数。

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
```

示例**2**：的后端和存储类配置 solidfire-san 具有虚拟池的驱动程序

此示例显示了使用虚拟池配置的后端定义文件以及引用这些池的StorageClasses。

在配置时、Astra Trident会将存储池上的标签复制到后端存储LUN。为了方便起见、存储管理员可以按标签为每个虚拟池和组卷定义标签。

在下面所示的示例后端定义文件中、为所有存储池设置了特定的默认值、这些存储池设置了 `type` 在Silver。虚拟池在中进行定义 `storage` 部分。在此示例中，某些存储池设置了自己的类型，而某些池将覆盖上述默认值。

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
    performance: gold
    cost: '4'
  zone: us-east-1a
  type: Gold
- labels:
    performance: silver
    cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
    performance: bronze
    cost: '2'
  zone: us-east-1c
```

```
  type: Bronze
- labels:
    performance: silver
    cost: '1'
  zone: us-east-1d
```

以下StorageClass定义引用了上述虚拟池。使用 `parameters.selector` 字段中、每个StorageClass都会调用可用于托管卷的虚拟池。卷将在选定虚拟池中定义各个方面。

第一个StorageClass (`solidfire-gold-four`)将映射到第一个虚拟池。这是唯一一个可通过提供金牌性能的池 `Volume Type QoS` 金牌。最后一个StorageClass (`solidfire-silver`)调用提供银牌性能的任何存储池。Astra Trident将决定选择哪个虚拟池、并确保满足存储要求。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

了解更多信息

- "卷访问组"

使用 ONTAP SAN 驱动程序配置后端

了解如何使用 ONTAP 和 Cloud Volumes ONTAP SAN 驱动程序配置 ONTAP 后端。

- "准备"
- "配置和示例"

> Astra Control可为使用创建的卷提供无缝保护、灾难恢复和移动性(在Kubernetes集群之间移动卷) `ontap-nas`， `ontap-nas-flexgroup`，和 `ontap-san` 驱动程序。请参见 "Astra Control复制前提条件" 了解详细信息。
>
> - 您必须使用 `ontap-nas` 适用于需要数据保护、灾难恢复和移动性的生产工作负载。
> - 使用 … `ontap-san-economy` 预期的卷使用量应远远高于ONTAP 支持的容量。
> - 使用 … `ontap-nas-economy` 仅当预期的卷使用量应远高于ONTAP 支持的容量时、以及 `ontap-san-economy` 无法使用驱动程序。
> - 请勿使用 `ontap-nas-economy` 预测数据保护、灾难恢复或移动性的需求。

用户权限

Astra Trident应以ONTAP 或SVM管理员身份运行、通常使用 `admin` 集群用户或 `vsadmin` SVM用户或具有相同角色的其他名称的用户。对于适用于NetApp ONTAP 的Amazon FSX部署、Astra Trident应使用集群以ONTAP 或SVM管理员身份运行 `fsxadmin` 用户或 `vsadmin` SVM用户或具有相同角色的其他名称的用户。。 `fsxadmin` 用户是集群管理员用户的有限替代用户。

> 如果您使用 `limitAggregateUsage` 参数、需要集群管理员权限。在将适用于NetApp ONTAP 的Amazon FSx与Astra Trident结合使用时、会显示 `limitAggregateUsage` 参数不适用于 `vsadmin` 和 `fsxadmin` 用户帐户。如果指定此参数，配置操作将失败。

虽然可以在 ONTAP 中创建 Trident 驱动程序可以使用的限制性更强的角色，但我们不建议这样做。大多数新版本的 Trident 都会调用需要考虑的其他 API ，从而使升级变得困难且容易出错。

准备使用ONTAP SAN驱动程序配置后端

了解如何准备使用 ONTAP SAN 驱动程序配置 ONTAP 后端。对于所有 ONTAP 后端， Astra Trident 需要至少为 SVM 分配一个聚合。

请记住，您还可以运行多个驱动程序，并创建指向其中一个驱动程序的存储类。例如、您可以配置 `san-dev` 使用的类 `ontap-san` 驱动程序和A `san-default` 使用的类 `ontap-san-economy` 一个。

所有Kubernetes工作节点都必须安装适当的iSCSI工具。请参见 "此处" 有关详细信息：

身份验证

Astra Trident 提供了两种对 ONTAP 后端进行身份验证的模式。

- Credential Based ： 具有所需权限的 ONTAP 用户的用户名和密码。建议使用预定义的安全登录角色、例如 `admin` 或 `vsadmin` 以确保与ONTAP 版本的最大兼容性。
- 基于证书： Astra Trident 还可以使用后端安装的证书与 ONTAP 集群进行通信。此处，后端定义必须包含客户端证书，密钥和可信 CA 证书的 Base64 编码值（如果使用）（建议）。

您可以更新现有后端、以便在基于凭据的方法和基于证书的方法之间移动。但是、一次仅支持一种身份验证方法。要切换到其他身份验证方法、必须从后端配置中删除现有方法。

> 如果您尝试同时提供*凭据和证书*、则后端创建将失败、并显示一条错误、指出配置文件中提供了多种身份验证方法。

启用基于凭据的身份验证

Astra Trident 需要 SVM 范围 / 集群范围的管理员的凭据才能与 ONTAP 后端进行通信。建议使用标准的预定义角色、例如 `admin` 或 `vsadmin`。这样可以确保与未来的 ONTAP 版本向前兼容，这些版本可能会使功能 API 公开供未来的 Astra Trident 版本使用。可以创建自定义安全登录角色并将其用于 Astra Trident ，但不建议使用。

后端定义示例如下所示：

YAML

版本：1 backendName：ExampleBackend storageDriverName：ONTAP SAN管理LIF：10.0.0.1 SVM ：SVM_NFS用户名：vsadmin密码：password

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

请注意，后端定义是凭据以纯文本格式存储的唯一位置。创建后端后，用户名 / 密码将使用 Base64 进行编码并存储为 Kubernetes 密钥。创建或更新后端是唯一需要了解凭据的步骤。因此，这是一项仅由管理员执行的操作，由 Kubernetes 或存储管理员执行。

启用基于证书的身份验证

新的和现有的后端可以使用证书并与 ONTAP 后端进行通信。后端定义需要三个参数。

- clientCertificate ： 客户端证书的 Base64 编码值。

- clientPrivateKey： 关联私钥的 Base64 编码值。
- trustedCACertifate： 受信任 CA 证书的 Base64 编码值。如果使用可信 CA ，则必须提供此参数。如果不使用可信 CA ，则可以忽略此设置。

典型的工作流包括以下步骤。

步骤

1. 生成客户端证书和密钥。生成时，将公用名（ Common Name ， CN ）设置为要作为身份验证的 ONTAP 用户。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. 将可信 CA 证书添加到 ONTAP 集群。此问题可能已由存储管理员处理。如果未使用可信 CA ，则忽略。

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. 在 ONTAP 集群上安装客户端证书和密钥（从步骤 1 开始）。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. 确认ONTAP 安全登录角色支持 `cert` 身份验证方法。

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. 使用生成的证书测试身份验证。将 <SVM 管理 LIF> 和 <SVM 名称 > 替换为管理 LIF IP 和 ONTAP 名称。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. 使用 Base64 对证书，密钥和可信 CA 证书进行编码。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 使用从上一步获得的值创建后端。

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

更新身份验证方法或轮换凭据

您可以更新现有后端以使用其他身份验证方法或轮换其凭据。这两种方式都适用：使用用户名 / 密码的后端可以更新为使用证书；使用证书的后端可以更新为基于用户名 / 密码的后端。为此、您必须删除现有身份验证方法并添加新的身份验证方法。然后、使用更新后的backend.json文件、该文件包含要执行的所需参数 `tridentctl backend update`。

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

轮换密码时，存储管理员必须先在 ONTAP 上更新用户的密码。然后进行后端更新。轮换证书时，可以向用户添加多个证书。之后，后端将更新以使用新证书，然后可以从 ONTAP 集群中删除旧证书。

更新后端不会中断对已创建卷的访问，也不会影响在之后建立的卷连接。成功的后端更新表明， Astra Trident 可以与 ONTAP 后端进行通信并处理未来的卷操作。

指定 **igroup**

Astra Trident 使用 igroup 来控制对其配置的卷（ LUN ）的访问。在为后端指定 igroup 时，管理员有两种选择：

- Astra Trident 可以自动为每个后端创建和管理 igroup 。条件 `igroupName` 不包含在后端定义中、Astra Trident将创建一个名为的igroup `trident-<backend-UUID>` 在 SVM 上。这将确保每个后端都有一个专用的 igroup ，并处理 Kubernetes 节点 IQN 的自动添加 / 删除。
- 或者，也可以在后端定义中提供预先创建的 igroup 。可以使用完成此操作 `igroupName` config参数。Astra Trident 会将 Kubernetes 节点 IQN 添加 / 删除到已有的 igroup 中。

用于具有的后端 `igroupName` 定义的、 `igroupName` 可以使用删除 `tridentctl backend update` 使用Astra Trident自动处理igroup。这样不会中断对已连接到工作负载的卷的访问。未来的连接将使用创建的 igroup Astra Trident 进行处理。

为 Astra Trident 的每个唯一实例指定一个 igroup 是一个最佳实践，对 Kubernetes 管理员和存储管理员都很有用。CSI Trident 可自动向 igroup 添加和删除集群节点 IQN ，从而极大地简化了其管理。在 Kubernetes 环境（以及 Astra Trident 安装）中使用相同的 SVM 时，使用专用的 igroup 可确保对一个 Kubernetes 集群所做的更改不会影响与另一个 Kubernetes 集群关联的 igroup 。此外，还必须确保 Kubernetes 集群中的每个节点都具有唯一的 IQN 。如上所述，Astra Trident 会自动处理 IQN 的添加和删除。在多个主机之间重复使用 IQN 可能会导致出现主机相互错误并拒绝访问 LUN 的不希望出现的情况。

如果将 Astra Trident 配置为充当 CSI 配置程序，则 Kubernetes 节点 IQN 会自动添加到 igroup 中或从 igroup 中删除。将节点添加到Kubernetes集群后、 `trident-csi` DemonSet部署POD (`trident-csi-xxxxx` 在23.01或之前的版本中 `trident-node<operating system>-xxxx` 在23.01及更高版本中)在新添加的节点上注册可将卷连接到的新节点。节点 IQN 也会添加到后端的 igroup 中。在对节点进行隔离，清空并从 Kubernetes 中删除时，可以执行一组类似的步骤来删除 IQN 。

如果 Astra Trident 未作为 CSI 配置程序运行，则必须手动更新 igroup ，以包含 Kubernetes 集群中每个工作节点的 iSCSI IQN 。需要将加入 Kubernetes 集群的节点的 IQN 添加到 igroup 中。同样，必须从 igroup 中删除从 Kubernetes 集群中删除的节点的 IQN 。

使用双向 **CHAP** 对连接进行身份验证

Astra Trident可以使用双向CHAP对iSCSI会话进行身份验证 `ontap-san` 和 `ontap-san-economy` 驱动程序。这需要启用 `useCHAP` 选项。设置为时 `true`、Astra Trident会将SVM的默认启动程序安全性配置为双向CHAP、并从后端文件设置用户名和密码。NetApp 建议使用双向 CHAP 对连接进行身份验证。请参见以下配置示例：

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
igroupName: trident
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

。 `useCHAP` 参数是一个布尔选项、只能配置一次。默认情况下，此参数设置为 false 。将其设置为 true 后，无法将其设置为 false 。

此外 `useCHAP=true`， `chapInitiatorSecret`， `chapTargetInitiatorSecret`， `chapTargetUsername`，和 `chapUsername` 后端定义中必须包含字段。在创建后端后、可以运行来更改这些密码 `tridentctl update`。

工作原理

通过设置 `useCHAP` 为true、存储管理员指示Astra Trident在存储后端配置CHAP。其中包括：

- 在 SVM 上设置 CHAP ：
  - 如果SVM的默认启动程序安全类型为none (默认设置)*和*卷中没有已存在的LUN、则Astra Trident会将默认安全类型设置为 `CHAP` 然后继续配置CHAP启动程序以及目标用户名和密码。
  - 如果 SVM 包含 LUN ，则 Astra Trident 不会在 SVM 上启用 CHAP 。这样可以确保对 SVM 上已存在的 LUN 的访问不受限制。
- 配置 CHAP 启动程序以及目标用户名和密码；必须在后端配置中指定这些选项（如上所示）。
- 管理向添加启动程序的操作 `igroupName` 在后端提供。如果未指定、则默认为 `trident`。

创建后端后、Astra Trident将创建相应的 `tridentbackend` CRD并将CHAP密钥和用户名存储为Kubernetes密钥。此后端由 Astra Trident 创建的所有 PV 都将通过 CHAP 进行挂载和连接。

轮换凭据并更新后端

您可以通过更新中的CHAP参数来更新CHAP凭据 `backend.json` 文件这需要更新CHAP密码并使用 `tridentctl update` 命令以反映这些更改。

> 更新后端的CHAP密码时、必须使用 `tridentctl` 更新后端。请勿通过 CLI/ONTAP UI 更新存储集群上的凭据，因为 Astra Trident 将无法选取这些更改。

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

现有连接将不受影响；如果凭据由 SVM 上的 Astra Trident 更新，则这些连接将继续保持活动状态。新连接将使用更新后的凭据，现有连接将继续保持活动状态。断开并重新连接旧的 PV 将导致它们使用更新后的凭据。

ONTAP SAN配置选项和示例

了解如何在您的 Astra Trident 安装中创建和使用 ONTAP SAN 驱动程序。本节提供了后端配置示例以及有关如何将后端映射到 StorageClasses 的详细信息。

后端配置选项

有关后端配置选项，请参见下表：

参数	Description	Default
`version`		始终为 1

参数	Description	Default
`storageDriverName`	存储驱动程序的名称	"ontap-nas"， "ontap-nas-economy-"， "ontap-nas-flexgroup"， "ontap-san "，"ontap-san-economy-"
`backendName`	自定义名称或存储后端	驱动程序名称 + "_" + dataLIF
`managementLIF`	集群或SVM管理LIF的IP地址要进行无缝MetroCluster 切换、必须指定SVM管理LIF。可以指定完全限定域名(FQDN)。如果使用安装了Astra Trident、则可以将设置为使用IPv6地址 `--use-ipv6` 标志。IPv6地址必须用方括号定义、例如：[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]。	"10.0.0.1 "， "2001： 1234：abcd：：： fefe] "
`dataLIF`	协议 LIF 的 IP 地址。请勿为**iSCSI**指定。 Astra Trident使用 "ONTAP选择性LUN映射" 发现建立多路径会话所需的iSCI LIF。如果出现、则会生成警告 `dataLIF` 已明确定义。	由SVM派生
`useCHAP`	使用CHAP对iSCSI的ONTAP SAN驱动程序进行身份验证[布尔值]。设置为 `true` 让Astra Trident为后端给定的SVM配置并使用双向CHAP作为默认身份验证。请参见 "准备使用ONTAP SAN驱动程序配置后端" 了解详细信息。	false
`chapInitiatorSecret`	CHAP 启动程序密钥。如果为、则为必需项 `useCHAP=true`	"
`labels`	要应用于卷的一组任意 JSON 格式的标签	"
`chapTargetInitiatorSecret`	CHAP 目标启动程序密钥。如果为、则为必需项 `useCHAP=true`	"
`chapUsername`	入站用户名。如果为、则为必需项 `useCHAP=true`	"
`chapTargetUsername`	目标用户名。如果为、则为必需项 `useCHAP=true`	"
`clientCertificate`	客户端证书的 Base64 编码值。用于基于证书的身份验证	"
`clientPrivateKey`	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证	"
`trustedCACertificate`	受信任 CA 证书的 Base64 编码值。可选。用于基于证书的身份验证。	"
`username`	与ONTAP 集群通信所需的用户名。用于基于凭据的身份验证。	"

参数	Description	Default
password	与ONTAP 集群通信所需的密码。用于基于凭据的身份验证。	"
svm	要使用的 Storage Virtual Machine	如果是SVM、则派生 managementLIF 已指定
igroupName	要使用的SAN卷的igroup的名称。请参见 有关详细信息 …	"trident — < 后端 UUID >"
storagePrefix	在 SVM 中配置新卷时使用的前缀。无法稍后修改。要更新此参数、您需要创建一个新的后端。	Trident
limitAggregateUsage	如果使用量超过此百分比，则配置失败。如果您使用适用于NetApp ONTAP 后端的Amazon FSX、请勿指定 limitAggregateUsage。提供的 fsxadmin 和 vsadmin 请勿包含检索聚合使用情况所需的权限、并使用Astra Trident对其进行限制。	"（默认情况下不强制实施）
limitVolumeSize	如果请求的卷大小超过此值、则配置失败。此外、还会限制它为qtree和LUN管理的卷的最大大小。	"（默认情况下不强制实施）
lunsPerFlexvol	每个 FlexVol 的最大 LUN 数，必须在 50 ， 200 范围内	100
debugTraceFlags	故障排除时要使用的调试标志。例如、除非您正在进行故障排除并需要详细的日志转储、否则｛"api"：false、"method "：true｝请勿使用。	空
useREST	用于使用 ONTAP REST API 的布尔参数。* 技术预览 * useREST 作为一个*技术预览版提供、建议用于测试环境、而不是生产工作负载。设置为时 true 、Astra Trident将使用ONTAP REST API与后端进行通信。此功能需要使用ONTAP 9.11.1及更高版本。此外、使用的ONTAP 登录角色必须有权访问 ontap 应用程序。这一点可通过预定义来满足 vsadmin 和 cluster-admin 角色。useREST MetroCluster 不支持。	false

详细信息 igroupName

igroupName 可以设置为已在ONTAP 集群上创建的igroup。如果未指定、则Astra Trident会自动创建名为的igroup trident-<backend-UUID>。

如果要在环境之间共享SVM、则如果要提供预定义的igroupName、建议为每个Kubernetes集群使用一个

igroup。这对于Astra Trident自动维护IQN添加和删除是必需的。

- `igroupName` 可以更新为指向在Astra Trident之外的SVM上创建和管理的新igroup。
- `igroupName` 可以省略。在这种情况下、Astra Trident将创建并管理名为的igroup `trident-<backend-UUID>` 自动。

在这两种情况下，仍可访问卷附件。未来的卷附件将使用更新后的 igroup 。此更新不会中断对后端卷的访问。

用于配置卷的后端配置选项

您可以在中使用这些选项控制默认配置 `defaults` 配置部分。有关示例，请参见以下配置示例。

参数	Description	Default
`spaceAllocation`	LUN 的空间分配	true
`spaceReserve`	空间预留模式； " 无 " （精简）或 " 卷 " （厚）	无
`snapshotPolicy`	要使用的 Snapshot 策略	无
`qosPolicy`	要为创建的卷分配的 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一。在 Astra Trident 中使用 QoS 策略组需要 ONTAP 9.8 或更高版本。建议使用非共享QoS策略组、并确保策略组分别应用于每个成分卷。共享 QoS 策略组将对所有工作负载的总吞吐量实施上限。	"
`adaptiveQosPolicy`	要为创建的卷分配的自适应 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一	"
`snapshotReserve`	为快照预留的卷百分比为 "0"	条件 `snapshotPolicy` 为"无"、否则为""
`splitOnClone`	创建克隆时，从其父级拆分该克隆	false
`encryption`	在新卷上启用NetApp卷加密(NVE)；默认为 `false`。要使用此选项，必须在集群上获得 NVE 的许可并启用 NVE 。如果在后端启用了NAE、则在Astra Trident中配置的任何卷都将启用NAE。有关详细信息、请参见： "Astra Trident如何与NVE和NAE配合使用"。	false
`luksEncryption`	启用LUKS加密。请参见 "使用Linux统一密钥设置(LUKS)"。	""
`securityStyle`	新卷的安全模式	`unix`
`tieringPolicy`	使用 " 无 " 的分层策略	适用于 ONTAP 9.5 SVM-DR 之前的配置的 " 仅快照 "

卷配置示例

下面是定义了默认值的示例：

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: password
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
igroupName: custom
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

用于使用创建的所有卷 `ontap-san` 驱动程序、Astra Trident会向FlexVol 额外添加10%的容量、以容纳LUN元数据。LUN 将使用用户在 PVC 中请求的确切大小进行配置。Astra Trident 将 FlexVol 增加 10% （在 ONTAP 中显示为可用大小）。用户现在将获得所请求的可用容量。此更改还可防止 LUN 变为只读状态，除非已充分利用可用空间。这不适用于 ontap-san-economy.

用于定义的后端 `snapshotReserve`、Astra Trident将按如下所示计算卷大小：

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

1.1 是 Astra Trident 向 FlexVol 额外添加 10% 以容纳 LUN 元数据。适用于 `snapshotReserve` = 5%、PVC请求= 5GiB、卷总大小为5.79GiB、可用大小为5.5GiB。。 `volume show` 命令应显示与以下示例类似的结果：

目前，调整大小是对现有卷使用新计算的唯一方法。

最低配置示例

以下示例显示了将大多数参数保留为默认值的基本配置。这是定义后端的最简单方法。

> 如果您正在将 NetApp ONTAP 上的 Amazon FSx 与 Astra Trident 结合使用，建议为 LIF 指定 DNS 名称，而不是 IP 地址。

`ontap-san` 具有基于证书的身份验证的驱动程序

这是一个最低后端配置示例。 `clientCertificate`， `clientPrivateKey`，和 `trustedCACertificate` (如果使用可信CA、则可选)将填充 `backend.json` 和分别采用客户端证书、专用密钥和可信CA证书的base64编码值。

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

`ontap-san` 具有双向**CHAP**的驱动程序

这是一个最低后端配置示例。此基本配置将创建 `ontap-san` 后端 `useCHAP` 设置为 `true`。

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
```

`ontap-san-economy` 驱动程序

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
```

虚拟池后端示例

在下面所示的示例后端定义文件中、会为所有存储池设置特定的默认值、例如 `spaceReserve` 无、`spaceAllocation` 为false、和 `encryption` 为false。虚拟池在存储部分中进行定义。

Astra Trident会在"Comments"字段中设置配置标签。注释在FlexVol 上设置。在配置时、Astra Trident会将虚拟池上的所有标签复制到存储卷。为了方便起见、存储管理员可以按标签为每个虚拟池和组卷定义标签。

在此示例中、某些存储池会设置自己的存储池 `spaceReserve`， `spaceAllocation`，和 `encryption` 值、而某些池会覆盖上述设置的默认值。

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    protection: gold
    creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
    protection: silver
    creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
    protection: bronze
    creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

以下是的iSCSI示例 `ontap-san-economy` 驱动程序：

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
    app: oracledb
    cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
    app: postgresdb
    cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
    app: mysqldb
    cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

将后端映射到 StorageClasses

以下StorageClass定义引用了上述虚拟池。使用 `parameters.selector` 字段中、每个StorageClass都会调用可用于托管卷的虚拟池。卷将在选定虚拟池中定义各个方面。

- 第一个StorageClass (`protection-gold`)将映射到中的第一个、第二个虚拟池 `ontap-nas-flexgroup` 中的后端和第一个虚拟池 `ontap-san` 后端。这是唯一一个提供黄金级保护的池。
- 第二个StorageClass (`protection-not-gold`)将映射到中的第三个、第四个虚拟池 `ontap-nas-flexgroup` 中的后端和第二个、第三个虚拟池 `ontap-san` 后端。这些池是唯一提供黄金级以外保护级别的池。
- 第三个StorageClass (`app-mysqldb`)将映射到中的第四个虚拟池 `ontap-nas` 中的后端和第三个虚拟池 `ontap-san-economy` 后端。这些池是唯一为 mysqldb 类型的应用程序提供存储池配置的池。
- 第四个StorageClass (`protection-silver-creditpoints-20k`)将映射到中的第三个虚拟池 `ontap-nas-flexgroup` 中的后端和第二个虚拟池 `ontap-san` 后端。这些池是唯一以 20000 个信用点提供黄金级保护的池。
- 第五个StorageClass (`creditpoints-5k`)将映射到中的第二个虚拟池 `ontap-nas-economy` 中的后端和第三个虚拟池 `ontap-san` 后端。这些是唯一一款具有 5000 个信用点的池产品。

Astra Trident将决定选择哪个虚拟池、并确保满足存储要求。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

配置ONTAP NAS后端

了解如何使用 ONTAP 和 Cloud Volumes ONTAP NAS 驱动程序配置 ONTAP 后端。

- "准备"
- "配置和示例"

> Astra Control可为使用创建的卷提供无缝保护、灾难恢复和移动性(在Kubernetes集群之间移动卷)`ontap-nas`， `ontap-nas-flexgroup`，和 `ontap-san` 驱动程序。请参见 "Astra Control复制前提条件" 了解详细信息。
>
> - 您必须使用 `ontap-nas` 适用于需要数据保护、灾难恢复和移动性的生产工作负载。
> - 使用 … `ontap-san-economy` 预期的卷使用量应远远高于ONTAP 支持的容量。
> - 使用 … `ontap-nas-economy` 仅当预期的卷使用量应远高于ONTAP 支持的容量时、以及 `ontap-san-economy` 无法使用驱动程序。
> - 请勿使用 `ontap-nas-economy` 预测数据保护、灾难恢复或移动性的需求。

用户权限

Astra Trident应以ONTAP 或SVM管理员身份运行、通常使用 `admin` 集群用户或 `vsadmin` SVM用户或具有相同角色的其他名称的用户。对于适用于NetApp ONTAP 的Amazon FSX部署、Astra Trident应使用集群以ONTAP 或SVM管理员身份运行 `fsxadmin` 用户或 `vsadmin` SVM用户或具有相同角色的其他名称的用户。。 `fsxadmin` 用户是集群管理员用户的有限替代用户。

> 如果您使用 `limitAggregateUsage` 参数、需要集群管理员权限。在将适用于NetApp ONTAP 的Amazon FSx与Astra Trident结合使用时、会显示 `limitAggregateUsage` 参数不适用于 `vsadmin` 和 `fsxadmin` 用户帐户。如果指定此参数，配置操作将失败。

虽然可以在 ONTAP 中创建 Trident 驱动程序可以使用的限制性更强的角色，但我们不建议这样做。大多数新版本的 Trident 都会调用需要考虑的其他 API ，从而使升级变得困难且容易出错。

准备使用ONTAP NAS驱动程序配置后端

了解如何准备使用 ONTAP NAS 驱动程序配置 ONTAP 后端。对于所有 ONTAP 后端， Astra Trident 需要至少为 SVM 分配一个聚合。

对于所有 ONTAP 后端， Astra Trident 需要至少为 SVM 分配一个聚合。

请记住，您还可以运行多个驱动程序，并创建指向其中一个驱动程序的存储类。例如、您可以配置一个使用的黄金类 `ontap-nas` 驱动程序和使用的铜牌类 `ontap-nas-economy` 一个。

所有Kubernetes工作节点都必须安装适当的NFS工具。请参见 "此处" 有关详细信息：

身份验证

Astra Trident 提供了两种对 ONTAP 后端进行身份验证的模式。

- Credential Based ： 具有所需权限的 ONTAP 用户的用户名和密码。建议使用预定义的安全登录角色、例如 `admin` 或 `vsadmin` 以确保与ONTAP 版本的最大兼容性。

- 基于证书： Astra Trident 还可以使用后端安装的证书与 ONTAP 集群进行通信。此处，后端定义必须包含客户端证书，密钥和可信 CA 证书的 Base64 编码值（如果使用）（建议）。

您可以更新现有后端、以便在基于凭据的方法和基于证书的方法之间移动。但是、一次仅支持一种身份验证方法。要切换到其他身份验证方法、必须从后端配置中删除现有方法。

如果您尝试同时提供*凭据和证书*、则后端创建将失败、并显示一条错误、指出配置文件中提供了多种身份验证方法。

启用基于凭据的身份验证

Astra Trident 需要 SVM 范围 / 集群范围的管理员的凭据才能与 ONTAP 后端进行通信。建议使用标准的预定义角色、例如 `admin` 或 `vsadmin`。这样可以确保与未来的 ONTAP 版本向前兼容，这些版本可能会使功能 API 公开供未来的 Astra Trident 版本使用。可以创建自定义安全登录角色并将其用于 Astra Trident ，但不建议使用。

后端定义示例如下所示：

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

请注意，后端定义是凭据以纯文本格式存储的唯一位置。创建后端后，用户名 / 密码将使用 Base64 进行编码并存储为 Kubernetes 密钥。创建 / 更新后端是唯一需要了解凭据的步骤。因此，这是一项仅由管理员执行的操作，由 Kubernetes 或存储管理员执行。

启用基于证书的身份验证

新的和现有的后端可以使用证书并与 ONTAP 后端进行通信。后端定义需要三个参数。

- clientCertificate： 客户端证书的 Base64 编码值。
- clientPrivateKey： 关联私钥的 Base64 编码值。
- trustedCACertifate： 受信任 CA 证书的 Base64 编码值。如果使用可信 CA ，则必须提供此参数。如果不使用可信 CA ，则可以忽略此设置。

典型的工作流包括以下步骤。

步骤

1. 生成客户端证书和密钥。生成时，将公用名（ Common Name ， CN ）设置为要作为身份验证的 ONTAP 用户。

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. 将可信 CA 证书添加到 ONTAP 集群。此问题可能已由存储管理员处理。如果未使用可信 CA ，则忽略。

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. 在 ONTAP 集群上安装客户端证书和密钥（从步骤 1 开始）。

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. 确认ONTAP 安全登录角色支持 `cert` 身份验证方法。

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. 使用生成的证书测试身份验证。将 <SVM 管理 LIF> 和 <SVM 名称 > 替换为管理 LIF IP 和 ONTAP 名称。您必须确保LIF的服务策略设置为 `default-data-management`。

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. 使用 Base64 对证书，密钥和可信 CA 证书进行编码。

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. 使用从上一步获得的值创建后端。

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

更新身份验证方法或轮换凭据

您可以更新现有后端以使用其他身份验证方法或轮换其凭据。这两种方式都适用：使用用户名 / 密码的后端可以更新为使用证书；使用证书的后端可以更新为基于用户名 / 密码的后端。为此、您必须删除现有身份验证方法并添加新的身份验证方法。然后、使用更新后的backend.json文件、该文件包含要执行的所需参数 `tridentctl update backend`。

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

> 轮换密码时，存储管理员必须先在 ONTAP 上更新用户的密码。然后进行后端更新。轮换证书时，可以向用户添加多个证书。之后，后端将更新以使用新证书，然后可以从 ONTAP 集群中删除旧证书。

更新后端不会中断对已创建卷的访问，也不会影响在之后建立的卷连接。成功的后端更新表明， Astra Trident 可以与 ONTAP 后端进行通信并处理未来的卷操作。

管理 **NFS** 导出策略

Astra Trident 使用 NFS 导出策略来控制对其配置的卷的访问。

使用导出策略时， Astra Trident 提供了两个选项：

- Astra Trident 可以动态管理导出策略本身；在此操作模式下，存储管理员会指定一个表示可接受 IP 地址的 CIDR 块列表。Astra Trident 会自动将属于这些范围的节点 IP 添加到导出策略中。或者，如果未指定任何 CIDR ，则在节点上找到的任何全局范围的单播 IP 都将添加到导出策略中。
- 存储管理员可以手动创建导出策略和添加规则。除非在配置中指定了不同的导出策略名称，否则 Astra Trident 将使用默认导出策略。

动态管理导出策略

CSI Trident 20.04 版可以动态管理 ONTAP 后端的导出策略。这样，存储管理员就可以为工作节点 IP 指定允许的地址空间，而不是手动定义显式规则。它大大简化了导出策略管理；修改导出策略不再需要手动干预存储集群。此外、这有助于将对存储集群的访问限制为仅允许IP位于指定范围内的工作节点访问、从而支持精细的自动化管理。

只有 CSI Trident 才支持动态管理导出策略。请务必确保工作节点未被 NAT 处理。

示例

必须使用两个配置选项。下面是一个后端定义示例：

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
- 192.168.0.0/24
autoExportPolicy: true
```

使用此功能时、您必须确保SVM中的根接合具有先前创建的导出策略、并具有允许节点CIDR块的导出规则(例如默认导出策略)。请始终遵循 NetApp 建议的最佳实践，为 Astra Trident 专用 SVM 。

以下是使用上述示例对此功能的工作原理进行的说明：

- `autoExportPolicy` 设置为 `true`。这表示Astra Trident将为创建导出策略 `svm1` SVM并使用处理规则的添加和删除 `autoExportCIDRs` 地址块。例如、UUID为403b5326-8482-40db-96d0-d83fb3f4daec和的后端 `autoExportPolicy` 设置为 `true` 创建名为的导出策略 `trident-403b5326-8482-40db-96d0-d83fb3f4daec` 在 SVM 上。
- `autoExportCIDRs` 包含地址块列表。此字段为可选字段，默认为 "0.0.0.0/0 ， "：： /0" 。如果未定义，则 Astra Trident 会添加在工作节点上找到的所有全局范围的单播地址。

在此示例中、将显示 `192.168.0.0/24` 提供了地址空间。这表示此地址范围内的 Kubernetes 节点 IP 将添加到 Astra Trident 创建的导出策略中。当Astra Trident注册其运行所在的节点时、它会检索该节点的IP地址并根据中提供的地址块对其进行检查 `autoExportCIDRs`。筛选 IP 后， Astra Trident 会为其发现的客户端 IP 创建导

出策略规则，并为其标识的每个节点创建一个规则。

您可以更新 `autoExportPolicy` 和 `autoExportCIDRs` 用于后端。您可以为自动管理的后端附加新的 CIDR ，也可以删除现有的 CIDR 。删除 CIDR 时请务必小心，以确保现有连接不会断开。您也可以选择禁用 `autoExportPolicy` 用于后端、并回退到手动创建的导出策略。这需要设置 `exportPolicy` 参数。

在Astra Trident创建或更新后端之后、您可以使用检查后端 `tridentctl` 或相应的 `tridentbackend` CRD：

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

当节点添加到Kubernetes集群并注册到Astra Trident控制器时、现有后端的导出策略会进行更新(前提是它们位于中指定的地址范围内 `autoExportCIDRs` 后端)。

删除节点后， Astra Trident 会检查所有联机后端，以删除该节点的访问规则。通过从受管后端的导出策略中删除此节点 IP ， Astra Trident 可防止恶意挂载，除非此 IP 可由集群中的新节点重复使用。

对于以前存在的后端、请使用更新后端 `tridentctl update backend` 将确保Astra Trident自动管理导出策略。这将创建一个以后端 UUID 命名的新导出策略，后端上存在的卷将在重新挂载时使用新创建的导出策略。

> 删除具有自动管理导出策略的后端将删除动态创建的导出策略。如果重新创建后端，则会将其视为新的后端，并会创建新的导出策略。

如果更新了活动节点的 IP 地址，则必须在此节点上重新启动 Astra Trident Pod 。然后， Astra Trident 将更新其管理的后端的导出策略，以反映此 IP 更改。

ONTAP NAS配置选项和示例

了解如何在您的 Astra Trident 安装中创建和使用 ONTAP NAS 驱动程序。本节提供了后端配置示例以及有关如何将后端映射到 StorageClasses 的详细信息。

有关后端配置选项，请参见下表：

参数	Description	Default
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	"ontap-nas"， "ontap-nas-economy-"， "ontap-nas-flexgroup"， "ontap-san "，"ontap-san-economy-"
`backendName`	自定义名称或存储后端	驱动程序名称 + "_" + dataLIF
`managementLIF`	集群或SVM管理LIF的IP地址要进行无缝MetroCluster 切换、必须指定SVM管理LIF。可以指定完全限定域名(FQDN)。如果使用安装了Astra Trident、则可以将设置为使用IPv6地址 `--use-ipv6` 标志。IPv6地址必须用方括号定义、例如：[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]。	"10.0.0.1 "， "2001： 1234：abcd：：： fefe] "
`dataLIF`	协议 LIF 的 IP 地址。建议指定 `dataLIF`。如果未提供此参数、则Astra Trident会从SVM提取数据LIF。您可以指定用于NFS挂载操作的完全限定域名(FQDN)、从而可以创建循环DNS、以便在多个数据LIF之间实现负载平衡。可以在初始设置后更改。请参见 。如果使用安装了Astra Trident、则可以将设置为使用IPv6地址 `--use-ipv6` 标志。IPv6地址必须用方括号定义、例如：[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]。	指定的地址或派生自SVM (如果未指定)(不建议)
`autoExportPolicy`	启用自动创建和更新导出策略[布尔值]。使用 `autoExportPolicy` 和 `autoExportCIDRs` 选项、Astra Trident可以自动管理导出策略。	false
`autoExportCIDRs`	用于筛选Kubernetes节点IP的CIDR列表 `autoExportPolicy` 已启用。使用 `autoExportPolicy` 和 `autoExportCIDRs` 选项、Astra Trident可以自动管理导出策略。	["0.0.0.0/0 "， "：： /0 "]`
`labels`	要应用于卷的一组任意 JSON 格式的标签	"
`clientCertificate`	客户端证书的 Base64 编码值。用于基于证书的身份验证	"
`clientPrivateKey`	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证	"

参数	Description	Default
`trustedCACertificate`	受信任 CA 证书的 Base64 编码值。可选。用于基于证书的身份验证	"
`username`	用于连接到集群 /SVM 的用户名。用于基于凭据的身份验证	
`password`	连接到集群 /SVM 的密码。用于基于凭据的身份验证	
`svm`	要使用的 Storage Virtual Machine	如果是SVM、则派生 `managementLIF` 已指定
`storagePrefix`	在 SVM 中配置新卷时使用的前缀。设置后无法更新	Trident
`limitAggregateUsage`	如果使用量超过此百分比，则配置失败。* 不适用于适用于 ONTAP 的 Amazon FSx *	"（默认情况下不强制实施）
`limitVolumeSize`	如果请求的卷大小超过此值、则配置失败。	"（默认情况下不强制实施）
`limitVolumeSize`	如果请求的卷大小超过此值、则配置失败。此外、还会限制它为qtree和LUN以及管理的卷的最大大小 `qtreesPerFlexvol` 选项用于自定义每个FlexVol 的最大qtree数。	"（默认情况下不强制实施）
`lunsPerFlexvol`	每个 FlexVol 的最大 LUN 数，必须在 50 ， 200 范围内	100
`debugTraceFlags`	故障排除时要使用的调试标志。例如、 {"api"：false、"method "：true｝不使用 `debugTraceFlags` 除非您正在进行故障排除并需要详细的日志转储。	空
`nfsMountOptions`	NFS挂载选项的逗号分隔列表。Kubernetes持久卷的挂载选项通常在存储类中指定、但如果在存储类中未指定挂载选项、则Astra Trident将回退到使用存储后端配置文件中指定的挂载选项。如果在存储类或配置文件中未指定挂载选项、则Astra Trident不会在关联的永久性卷上设置任何挂载选项。	"
`qtreesPerFlexvol`	每个 FlexVol 的最大 qtree 数，必须在 50 ， 300 范围内	200

参数	Description	Default
useREST	用于使用 ONTAP REST API 的布尔参数。* 技术预览 * useREST 作为一个*技术预览版提供、建议用于测试环境、而不是生产工作负载。设置为时 true 、Astra Trident将使用ONTAP REST API与后端进行通信。此功能需要使用ONTAP 9.11.1及更高版本。此外、使用的ONTAP 登录角色必须有权访问 ontap 应用程序。这一点可通过预定义来满足 vsadmin 和 cluster-admin 角色。useREST MetroCluster 不支持。	false

用于配置卷的后端配置选项

您可以在中使用这些选项控制默认配置 defaults 配置部分。有关示例，请参见以下配置示例。

参数	Description	Default
spaceAllocation	LUN 的空间分配	true
spaceReserve	空间预留模式； " 无 " （精简）或 " 卷 " （厚）	无
snapshotPolicy	要使用的 Snapshot 策略	无
qosPolicy	要为创建的卷分配的 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一	"
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一。不受 ontap-nas-economy.	"
snapshotReserve	为快照预留的卷百分比为 "0"	条件 snapshotPolicy 为"无"、否则为""
splitOnClone	创建克隆时，从其父级拆分该克隆	false
encryption	在新卷上启用NetApp卷加密(NVE)；默认为 false。要使用此选项，必须在集群上获得 NVE 的许可并启用 NVE 。如果在后端启用了NAE、则在Astra Trident中配置的任何卷都将启用NAE。有关详细信息、请参见： "Astra Trident如何与NVE和NAE配合使用"。	false
tieringPolicy	使用 " 无 " 的分层策略	适用于 ONTAP 9.5 SVM-DR 之前的配置的 " 仅快照 "
unixPermissions	新卷的模式	对于NFS卷为"777"；对于SMB卷为空(不适用)

参数	Description	Default
snapshotDir	控制的可见性 .snapshot 目录	false
exportPolicy	要使用的导出策略	default
securityStyle	新卷的安全模式。NFS支持 mixed 和 unix 安全模式。SMB支持 mixed 和 ntfs 安全模式。	NFS默认值为 unix。SMB默认值为 ntfs。

在 Astra Trident 中使用 QoS 策略组需要 ONTAP 9.8 或更高版本。建议使用非共享 QoS 策略组，并确保策略组分别应用于每个成分卷。共享 QoS 策略组将对所有工作负载的总吞吐量实施上限。

卷配置示例

下面是一个定义了默认值的示例：

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: password
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: '10'
```

适用于 ontap-nas 和 ontap-nas-flexgroups`现在、Astra Trident会使用新的计算方法来确保FlexVol 的大小与snapshotReserve百分比和PVC相同。当用户请求 PVC 时， Astra Trident 会使用新计算创建具有更多空间的原始 FlexVol 。此计算可确保用户在 PVC 中收到所请求的可写空间，而不是小于所请求的空间。在 v21.07 之前，如果用户请求 PVC （例如， 5GiB ），并且 snapshotReserve 为 50% ，则只会获得 2.5 GiB 的可写空间。这是因为用户请求的是整个卷和 `snapshotReserve 是其中

的一个百分比。在Trident 21.07中、用户请求的是可写空间、Astra Trident定义了 `snapshotReserve` 数字表示整个卷的百分比。这不适用于 `ontap-nas-economy`。请参见以下示例以了解其工作原理：

计算方法如下：

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

对于 snapshotReserve = 50% ， PVC 请求 = 5GiB ，卷总大小为 2/.5 = 10GiB ，可用大小为 5GiB ，这是用户在 PVC 请求中请求的大小。。 `volume show` 命令应显示与以下示例类似的结果：

在升级 Astra Trident 时，先前安装的现有后端将按照上述说明配置卷。对于在升级之前创建的卷，您应调整其卷的大小，以便观察到所做的更改。例如、具有的2 GiB PVC `snapshotReserve=50` 之前的结果是、卷可提供1 GiB的可写空间。例如，将卷大小调整为 3GiB 可为应用程序在一个 6 GiB 卷上提供 3GiB 的可写空间。

示例

最低配置示例

以下示例显示了将大多数参数保留为默认值的基本配置。这是定义后端的最简单方法。

> 如果在采用 Trident 的 NetApp ONTAP 上使用 Amazon FSx ，建议为 LIF 指定 DNS 名称，而不是 IP 地址。

上的默认选项 **<code>ontap-nas-economy</code>**

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

这是一个最低后端配置示例。 `clientCertificate`， `clientPrivateKey`，和 `trustedCACertificate` (如果使用可信CA、则可选)将填充 `backend.json` 和分别采用客户端证书、专用密钥和可信CA证书的base64编码值。

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

以下示例显示了如何指示Astra Trident使用动态导出策略自动创建和管理导出策略。此操作对于也是如此 `ontap-nas-economy` 和 `ontap-nas-flexgroup` 驱动程序。

ontap-NAS 驱动程序

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

<code>ontap-nas-flexgroup</code> 驱动程序

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: test-cluster-east-1b
  backend: test1-ontap-cluster
svm: svm_nfs
username: vsadmin
password: password
```

使用**IPv6**地址

此示例显示了 `managementLIF` 使用IPv6地址。

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

`ontap-nas-economy` 驱动程序

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

`ontap-nas` 适用于使用**SMB**卷的**Amazon FSX for ONTAP** 的驱动程序

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

在下面所示的示例后端定义文件中、会为所有存储池设置特定的默认值、例如 `spaceReserve` 无、`spaceAllocation` 为false、和 `encryption` 为false。虚拟池在存储部分中进行定义。

Astra Trident会在"Comments"字段中设置配置标签。注释在FlexVol 上为设置 `ontap-nas` 或FlexGroup `ontap-nas-flexgroup`。在配置时、Astra Trident会将虚拟池上的所有标签复制到存储卷。为了方便起见、存储管理员可以按标签为每个虚拟池和组卷定义标签。

在此示例中、某些存储池会设置自己的存储池 `spaceReserve`， `spaceAllocation`，和 `encryption` 值、而某些池会覆盖上述设置的默认值。

<code>ontap-nas</code> 驱动程序

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: admin
password: password
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: 'false'
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    app: msoffice
    cost: '100'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
    adaptiveQosPolicy: adaptive-premium
- labels:
    app: slack
    cost: '75'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    app: wordpress
    cost: '50'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
```

```
- labels:
    app: mysqldb
    cost: '25'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

<code>ontap-nas-flexgroup</code> 驱动程序

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    protection: gold
    creditpoints: '50000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: gold
    creditpoints: '30000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: silver
    creditpoints: '20000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    protection: bronze
    creditpoints: '10000'
```

```
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

<code>ontap-nas-economy</code> 驱动程序

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: nas_economy_store
region: us_east_1
storage:
- labels:
    department: finance
    creditpoints: '6000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    department: legal
    creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    department: engineering
    creditpoints: '3000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    department: humanresource
    creditpoints: '2000'
  zone: us_east_1d
```

```
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

更新 `dataLIF` 初始配置后

您可以在初始配置后更改数据LIF、方法是运行以下命令、为新的后端JSON文件提供更新的数据LIF。

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```

> 如果PVC连接到一个或多个Pod、则必须关闭所有对应Pod、然后将其恢复到、新数据LIF才能生效。

将后端映射到 **StorageClasses**

以下StorageClass定义引用了上述虚拟池。使用 `parameters.selector` 字段中、每个StorageClass都会调用可用于托管卷的虚拟池。卷将在选定虚拟池中定义各个方面。

- 第一个StorageClass (`protection-gold`)将映射到中的第一个、第二个虚拟池 `ontap-nas-flexgroup` 中的后端和第一个虚拟池 `ontap-san` 后端。这是唯一一个提供黄金级保护的池。
- 第二个StorageClass (`protection-not-gold`)将映射到中的第三个、第四个虚拟池 `ontap-nas-flexgroup` 中的后端和第二个、第三个虚拟池 `ontap-san` 后端。这些池是唯一提供黄金级以外保护级别的池。
- 第三个StorageClass (`app-mysqldb`)将映射到中的第四个虚拟池 `ontap-nas` 中的后端和第三个虚拟池 `ontap-san-economy` 后端。这些池是唯一为 mysqldb 类型的应用程序提供存储池配置的池。
- 第四个StorageClass (`protection-silver-creditpoints-20k`)将映射到中的第三个虚拟池 `ontap-nas-flexgroup` 中的后端和第二个虚拟池 `ontap-san` 后端。这些池是唯一以 20000 个信用点提供黄金级保护的池。
- 第五个StorageClass (`creditpoints-5k`)将映射到中的第二个虚拟池 `ontap-nas-economy` 中的后端和第三个虚拟池 `ontap-san` 后端。这些是唯一一款具有 5000 个信用点的池产品。

Astra Trident将决定选择哪个虚拟池、并确保满足存储要求。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

适用于 **NetApp ONTAP** 的 **Amazon FSX**

将 **Astra Trident** 与适用于 **NetApp ONTAP** 的 **Amazon FSX** 结合使用

"适用于 NetApp ONTAP 的 Amazon FSX" 是一种完全托管的AWS服务、可使客户启动和运行由NetApp ONTAP 存储操作系统提供支持的文件系统。借助适用于ONTAP 的FSx、您可以利用您熟悉的NetApp功能、性能和管理功能、同时利用在AWS上存储数据的简便性、灵活性、安全性和可扩展性。FSX for ONTAP 支持ONTAP 文件系统功能和管理API。

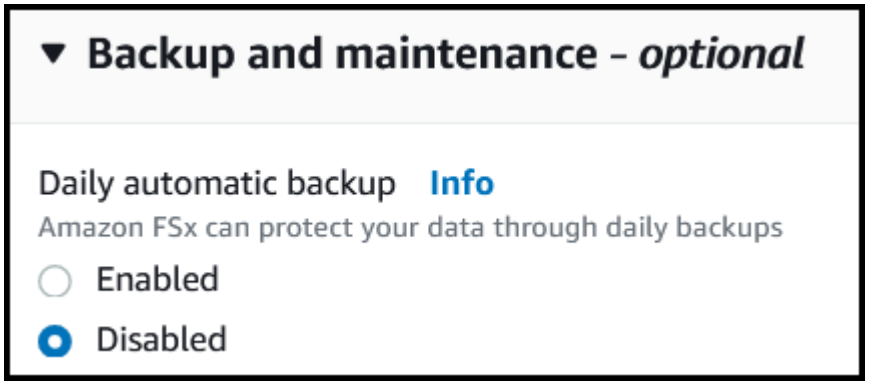
文件系统是 Amazon FSX 中的主要资源，类似于内部部署的 ONTAP 集群。在每个 SVM 中，您可以创建一个或多个卷，这些卷是将文件和文件夹存储在文件系统中的数据容器。借助适用于 NetApp ONTAP 的 Amazon FSX ， Data ONTAP 将作为云中的托管文件系统提供。新的文件系统类型称为 * NetApp ONTAP * 。

通过将 Astra Trident 与适用于 NetApp ONTAP 的 Amazon FSx 结合使用，您可以确保在 Amazon Elastic Kubernetes Service （ EKS ）中运行的 Kubernetes 集群可以配置由 ONTAP 备份的块和文件永久性卷。

适用于 NetApp ONTAP 的 Amazon FSX 使用 "FabricPool" 以管理存储层。通过它，您可以根据数据是否经常访问来将数据存储在层中。

注意事项

- SMB卷：
  - SMB卷支持使用 `ontap-nas` 仅限驱动程序。
  - Astra Trident仅支持将SMB卷挂载到Windows节点上运行的Pod。
  - Astra Trident不支持Windows ARM 架构。
- Trident无法删除在启用了自动备份的Amazon FSX文件系统上创建的卷。要删除 PVC ，您需要手动删除 PV 和 ONTAP 的 FSX 卷。要防止此问题描述 ，请执行以下操作：
  - 请勿使用 "* 快速创建 "* 来创建适用于 ONTAP 的 FSX 文件系统。快速创建工作流可启用自动备份，但不提供选择退出选项。
  - 使用 "** 标准创建 " 时，禁用自动备份。禁用自动备份可以使 Trident 成功删除卷，而无需进一步手动干预。



驱动程序

您可以使用以下驱动程序将Astra Trident与适用于NetApp ONTAP 的Amazon FSx集成：

- `ontap-san`：配置的每个PV都是其自己的Amazon FSX for NetApp ONTAP 卷中的一个LUN。
- `ontap-san-economy`：配置的每个PV都是一个LUN、对于NetApp ONTAP 卷、每个Amazon FSX的LUN数量是可配置的。
- `ontap-nas`：配置的每个PV都是一个适用于NetApp ONTAP 的完整Amazon FSX卷。
- `ontap-nas-economy`：配置的每个PV都是一个qtree、对于NetApp ONTAP 卷、每个Amazon FSX的qtree数量是可配置的。
- `ontap-nas-flexgroup`：配置的每个PV都是一个适用于NetApp ONTAP FlexGroup 的完整Amazon FSX卷。

有关驱动程序详细信息、请参见 "ONTAP 驱动程序"。

身份验证

Astra Trident提供两种身份验证模式。

- 基于证书： Astra Trident 将使用 SVM 上安装的证书与 FSX 文件系统上的 SVM 进行通信。
- 基于凭据：您可以使用 `fsxadmin` 文件系统或的用户 `vsadmin` 为SVM配置的用户。

> Astra Trident应作为运行 `vsadmin` SVM用户或具有相同角色的其他名称的用户。适用于NetApp ONTAP 的Amazon FSX具有 `fsxadmin` 有限更换ONTAP 的用户 `admin` 集群用户。我们强烈建议使用 `vsadmin` 使用Astra Trident。

您可以更新后端以在基于凭据的方法和基于证书的方法之间移动。但是、如果您尝试提供*凭据和证书*、则后端创建将失败。要切换到其他身份验证方法、必须从后端配置中删除现有方法。

有关启用身份验证的详细信息、请参阅适用于您的驱动程序类型的身份验证：

- "ONTAP NAS身份验证"
- "ONTAP SAN身份验证"

了解更多信息

- "Amazon FSX for NetApp ONTAP 文档"
- "有关适用于 NetApp ONTAP 的 Amazon FSX 的博客文章"

集成适用于**NetApp ONTAP** 的**Amazon FSX**

您可以将适用于NetApp ONTAP 的Amazon FSX文件系统与Astra Trident集成、以确保在Amazon Elastic Kubernetes Service (EKS)中运行的Kubernetes集群可以配置由ONTAP提供支持的块和文件永久性卷。

开始之前

此外 "Astra Trident 要求"要将适用于ONTAP 的FSx与Astra Trident集成、您需要：

- 具有的现有Amazon EKS集群或自管理Kubernetes集群 `kubectl` 已安装。
- 可从集群的工作节点访问的适用于 NetApp ONTAP 文件系统和 Storage Virtual Machine （ SVM ）的现有

Amazon FSX 。

- 为准备工作的工作节点 "NFS或iSCSI"。

> 确保按照Amazon Linux和Ubuntu所需的节点准备步骤进行操作 "Amazon Machine 映像" （ AMIS ），具体取决于您的 EKS AMI 类型。

SMB卷的其他要求

- 一个Kubernetes集群、其中包含一个Linux控制器节点以及至少一个运行Windows Server 2019的Windows工作节点。Astra Trident仅支持将SMB卷挂载到Windows节点上运行的Pod。
- 至少一个包含Active Directory凭据的Astra Trident密钥。以生成密钥 `smbcreds`：

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- 配置为Windows服务的CSI代理。配置 `csi-proxy`、请参见 "GitHub：CSI代理" 或 "GitHub：适用于Windows的CSI代理" 适用于在Windows上运行的Kubernetes节点。

ONTAP SAN和NAS驱动程序集成

> 如果要为SMB卷配置、则必须读取 准备配置SMB卷 创建后端之前。

步骤

1. 使用其中一种部署Astra Trident "部署方法"。
2. 收集SVM管理LIF DNS名称。例如、使用AWS命令行界面查找 `DNSName` 下的条目 `Endpoints` → `Management` 运行以下命令后：

```
aws fsx describe-storage-virtual-machines --region <file system region>
```

3. 创建和安装证书 "NAS后端身份验证" 或 "SAN后端身份验证"。

> 您可以从可以访问文件系统的任何位置使用 SSH 登录到文件系统（例如，安装证书）。使用 `fsxadmin` 用户、创建文件系统时配置的密码以及中的管理DNS名称 `aws fsx describe-file-systems`。

4. 使用您的证书和管理 LIF 的 DNS 名称创建后端文件，如以下示例所示：

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: customBackendName
managementLIF: svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal
svm: svm01
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXX.fs-
XXXXXXXXXXXXXXXXX.fsx.us-east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
 }
```

有关创建后端的信息，请参见以下链接：

- "使用 ONTAP NAS 驱动程序配置后端"
- "使用 ONTAP SAN 驱动程序配置后端"

结果

部署完成后、您可以创建 "存储类，配置卷以及将卷挂载到 Pod 中"。

准备配置SMB卷

您可以使用配置SMB卷 `ontap-nas` 驱动程序。完成前 ONTAP SAN和NAS驱动程序集成 完成以下步骤。

步骤

1. 创建SMB共享。您可以使用以下两种方式之一创建SMB管理共享 "Microsoft管理控制台" 共享文件夹管理单元或使用ONTAP 命令行界面。要使用ONTAP 命令行界面创建SMB共享、请执行以下操作：
   a. 如有必要，为共享创建目录路径结构。

。 `vserver cifs share create` 命令会在创建共享期间检查-path选项中指定的路径。如果指定路径不存在，则命令将失败。

b. 创建与指定SVM关联的SMB共享：

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. 验证是否已创建共享：

```
vserver cifs share show -share-name share_name
```

请参见 "创建 SMB 共享" 了解完整详细信息。

2. 创建后端时、必须配置以下内容以指定SMB卷。有关适用于ONTAP 后端的所有FSX配置选项、请参见 "适用于ONTAP 的FSX配置选项和示例"。

参数	Description	示例
`smbShare`	使用共享文件夹Microsoft管理控制台创建的SMB共享的名称。例如"smb-share"。对于**SMB**卷为必需项。	`smb-share`
`nasType`	*必须设置为 `smb`如果为空、则默认为 `nfs`。	`smb`
`securityStyle`	新卷的安全模式。必须设置为 **`ntfs`** 或 **`mixed`** 用于**SMB**卷。	`ntfs` 或 `mixed` 对于SMB卷
`unixPermissions`	新卷的模式。对于SMB卷、必须留空。	""

适用于**ONTAP** 的**FSX**配置选项和示例

了解适用于ONTAP 的Amazon FSX的后端配置选项。本节提供了后端配置示例。

后端配置选项

有关后端配置选项，请参见下表：

参数	Description	示例
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	"ontap-nas"， "ontap-nas-economy-"， "ontap-nas-flexgroup"， "ontap-san "，"ontap-san-economy-"

参数	Description	示例
backendName	自定义名称或存储后端	驱动程序名称 + "_" + dataLIF
managementLIF	集群或SVM管理LIF的IP地址要进行无缝MetroCluster 切换、必须指定SVM管理LIF。可以指定完全限定域名(FQDN)。如果使用安装了Astra Trident、则可以将设置为使用IPv6地址 --use-ipv6 标志。IPv6地址必须用方括号定义、例如：[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]。	"10.0.0.1 "， "2001： 1234：abcd：：： fefe] "
dataLIF	协议 LIF 的 IP 地址。* ONTAP NAS驱动程序*：建议指定dataLIF。如果未提供此参数、则Astra Trident会从SVM提取数据LIF。您可以指定用于NFS挂载操作的完全限定域名(FQDN)、从而可以创建循环DNS、以便在多个数据LIF之间实现负载平衡。可以在初始设置后更改。请参见 。* ONTAP SAN驱动程序*：不为iSCSI指定。Astra Trident使用ONTAP 选择性LUN映射来发现建立多路径会话所需的iSCI LIF。如果明确定义了dataLIF、则会生成警告。如果使用安装了Astra Trident、则可以将设置为使用IPv6地址 --use-ipv6 标志。IPv6地址必须用方括号定义、例如：[28e8：d9fb：a825：b7bf：69a8：d02f：9e7b：3555]。	
autoExportPolicy	启用自动创建和更新导出策略[布尔值]。使用 autoExportPolicy 和 autoExportCIDRs 选项、Astra Trident可以自动管理导出策略。	false
autoExportCIDRs	用于筛选Kubernetes节点IP的CIDR列表 autoExportPolicy 已启用。使用 autoExportPolicy 和 autoExportCIDRs 选项、Astra Trident可以自动管理导出策略。	"["0.0.0.0/0 "、"：： /0 "]"
labels	要应用于卷的一组任意 JSON 格式的标签	""
clientCertificate	客户端证书的 Base64 编码值。用于基于证书的身份验证	""
clientPrivateKey	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证	""
trustedCACertificate	受信任 CA 证书的 Base64 编码值。可选。用于基于证书的身份验证。	""

参数	Description	示例
username	用于连接到集群或SVM的用户名。用于基于凭据的身份验证。例如、vsadmin。	
password	用于连接到集群或SVM的密码。用于基于凭据的身份验证。	
svm	要使用的 Storage Virtual Machine	如果指定SVM管理LIF则派生。
igroupName	要使用的SAN卷的igroup的名称。请参见 。	"trident — < 后端 UUID >"
storagePrefix	在 SVM 中配置新卷时使用的前缀。创建后无法修改。要更新此参数、您需要创建一个新的后端。	Trident
limitAggregateUsage	*请勿为适用于NetApp ONTAP的Amazon FSX指定。*提供的 fsxadmin 和 vsadmin 请勿包含检索聚合使用情况所需的权限、并使用Astra Trident对其进行限制。	请勿使用。
limitVolumeSize	如果请求的卷大小超过此值、则配置失败。此外、还会限制它为qtree和LUN以及管理的卷的最大大小 qtreesPerFlexvol 选项用于自定义每个FlexVol 的最大qtree数。	"（默认情况下不强制实施）
lunsPerFlexvol	每个FlexVol 的最大LUN数必须在50、200范围内。仅限SAN。	"100"
debugTraceFlags	故障排除时要使用的调试标志。例如、 {"api"：false、"method "：true｝不使用 debugTraceFlags 除非您正在进行故障排除并需要详细的日志转储。	空
nfsMountOptions	NFS挂载选项的逗号分隔列表。Kubernetes持久卷的挂载选项通常在存储类中指定、但如果在存储类中未指定挂载选项、则Astra Trident将回退到使用存储后端配置文件中指定的挂载选项。如果在存储类或配置文件中未指定挂载选项、则Astra Trident不会在关联的永久性卷上设置任何挂载选项。	""
nasType	配置NFS或SMB卷创建。选项包括 nfs， smb`或为空。*必须设置为 `smb 对于SMB卷。*如果设置为空、则默认为NFS卷。	"NFs"
qtreesPerFlexvol	每个 FlexVol 的最大 qtree 数，必须在 50 ， 300 范围内	"200"

参数	Description	示例
smbShare	使用共享文件夹Microsoft管理控制台创建的SMB共享的名称。对于**SMB**卷为必需项。	"smb-share"
useREST	用于使用 ONTAP REST API 的布尔参数。* 技术预览 * useREST 作为一个*技术预览版提供、建议用于测试环境、而不是生产工作负载。设置为时 true 、Astra Trident将使用ONTAP REST API与后端进行通信。此功能需要使用ONTAP 9.11.1及更高版本。此外、使用的ONTAP 登录角色必须有权访问 ontap 应用程序。这一点可通过预定义来满足 vsadmin 和 cluster-admin 角色。	false

详细信息 igroupName

igroupName 可以设置为已在ONTAP 集群上创建的igroup。如果未指定、则Astra Trident会自动创建名为的igroup trident-<backend-UUID>。

如果要在环境之间共享SVM、则如果要提供预定义的igroupName、建议为每个Kubernetes集群使用一个igroup。这对于Astra Trident自动维护IQN添加和删除是必需的。

- igroupName 可以更新为指向在Astra Trident之外的SVM上创建和管理的新igroup。
- igroupName 可以省略。在这种情况下、Astra Trident将创建并管理名为的igroup trident-<backend-UUID> 自动。

在这两种情况下，仍可访问卷附件。未来的卷附件将使用更新后的 igroup 。此更新不会中断对后端卷的访问。

更新 dataLIF 初始配置后

您可以在初始配置后更改数据LIF、方法是运行以下命令、为新的后端JSON文件提供更新的数据LIF。

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```

如果PVC连接到一个或多个Pod、则必须关闭所有对应Pod、然后将其恢复到、新数据LIF才能生效。

用于配置卷的后端配置选项

您可以在中使用这些选项控制默认配置 defaults 配置部分。有关示例，请参见以下配置示例。

参数	Description	Default
spaceAllocation	LUN 的空间分配	true

参数	Description	Default
spaceReserve	空间预留模式；" 无 "（精简）或 " 卷 "（厚）	无
snapshotPolicy	要使用的 Snapshot 策略	无
qosPolicy	要为创建的卷分配的 QoS 策略组。选择每个存储池或后端的qosPolicy或adaptiveQosPolicy之一。在 Astra Trident 中使用 QoS 策略组需要 ONTAP 9.8 或更高版本。建议使用非共享QoS策略组、并确保策略组分别应用于每个成分卷。共享 QoS 策略组将对所有工作负载的总吞吐量实施上限。	"
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。选择每个存储池或后端的qosPolicy或adaptiveQosPolicy之一。不受 ontap-nas-economy.	"
snapshotReserve	为快照预留的卷百分比为 "0"	条件 snapshotPolicy 为"无"、否则为""
splitOnClone	创建克隆时，从其父级拆分该克隆	false
encryption	在新卷上启用NetApp卷加密(NVE)；默认为 false。要使用此选项，必须在集群上获得 NVE 的许可并启用 NVE 。如果在后端启用了NAE、则在Astra Trident中配置的任何卷都将启用NAE。有关详细信息、请参见： "Astra Trident如何与NVE和NAE配合使用"。	false
luksEncryption	启用LUKS加密。请参见 "使用Linux统一密钥设置(LUKS)"。仅限SAN。	""
tieringPolicy	使用 " 无 " 的分层策略	适用于 ONTAP 9.5 SVM-DR 之前的配置的 " 仅快照 "
unixPermissions	新卷的模式。对于**SMB**卷保留为空。	""
securityStyle	新卷的安全模式。NFS支持 mixed 和 unix 安全模式。SMB支持 mixed 和 ntfs 安全模式。	NFS默认值为 unix。SMB默认值为 ntfs。

示例

使用 nasType， node-stage-secret-name，和 node-stage-secret-namespace、您可以指定SMB卷并提供所需的Active Directory凭据。SMB卷支持使用 ontap-nas 仅限驱动程序。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

使用 kubectl 创建后端

后端定义了 Astra Trident 与存储系统之间的关系。它告诉 Astra Trident 如何与该存储系统进行通信，以及 Astra Trident 如何从该存储系统配置卷。安装 Astra Trident 后，下一步是创建后端。。 `TridentBackendConfig` 通过自定义资源定义(CRD)、您可以直接通过Kubernetes界面创建和管理Trident后端。您可以使用执行此操作 `kubectl` 或与Kubernetes分发版等效的CLI工具。

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`， `tbconfig`， `tbackendconfig`)是一个命名为节奏的前端CRD、可用于管理Astra Trident后端 `kubectl`。现在、Kubernetes和存储管理员可以直接通过Kubernetes命令行界面创建和管理后端、而无需专用命令行实用程序 (`tridentctl`）。

创建时 `TridentBackendConfig` 对象、将发生以下情况：

- Astra Trident 会根据您提供的配置自动创建后端。此值在内部表示为 `TridentBackend` (`tbe`， `tridentbackend`) CR。
- 。 `TridentBackendConfig` 唯一绑定到 `TridentBackend` 这是由Astra Trident创建的。

每个 `TridentBackendConfig` 使用维护一对一映射 `TridentBackend`。前者是为用户提供的用于设计和配置后端的接口；后者是 Trident 表示实际后端对象的方式。

> `TridentBackend` CRS由Astra Trident自动创建。您 * 不应 * 修改它们。如果要更新后端、请通过修改来执行此操作 `TridentBackendConfig` 对象。

有关的格式、请参见以下示例 `TridentBackendConfig` CR：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

您还可以查看中的示例 "Trident 安装程序" 所需存储平台 / 服务的示例配置目录。

。 `spec` 获取后端特定的配置参数。在此示例中、后端使用 `ontap-san` 存储驱动程序、并使用此处所示的配置参数。有关所需存储驱动程序的配置选项列表，请参见 "存储驱动程序的后端配置信息"。

。 `spec` 第节还包括 `credentials` 和 `deletionPolicy` 字段、这些字段是在中新增的 `TridentBackendConfig` CR：

- `credentials`：此参数是必填字段、包含用于向存储系统/服务进行身份验证的凭据。此密码设置为用户创建的 Kubernetes Secret 。凭据不能以纯文本形式传递，因此会导致错误。
- `deletionPolicy`：此字段定义了在使用时应执行的操作 `TridentBackendConfig` 已删除。它可以采用以下两种可能值之一：
  - `delete`：这将删除这两者 `TridentBackendConfig` CR以及关联的后端。这是默认值。
  - `retain`：当出现时 `TridentBackendConfig` CR已删除、后端定义仍存在、可使用进行管理 `tridentctl`。将删除策略设置为 `retain` 允许用户降级到早期版本(21.04之前的版本)并保留创建的后端。此字段的值可以在之后更新 `TridentBackendConfig` 已创建。

后端的名称使用进行设置 `spec.backendName`。如果未指定、则后端的名称将设置为的名称 `TridentBackendConfig` 对象(metadata.name)。建议使用显式设置后端名称 `spec.backendName`。

使用创建的后端 `tridentctl` 没有关联的 `TridentBackendConfig` 对象。您可以选择使用管理此类后端 `kubectl` 通过创建 `TridentBackendConfig` CR.必须小心指定相同的配置参数(例如 `spec.backendName`， `spec.storagePrefix`， `spec.storageDriverName`等）。Astra Trident将自动绑定新创建的 `TridentBackendConfig` 使用预先存在的后端。

步骤概述

以使用创建新后端 `kubectl`、您应执行以下操作：

1. 创建 "Kubernetes 机密"。此密钥包含 Astra Trident 与存储集群 / 服务通信所需的凭据。

2. 创建 `TridentBackendConfig` 对象。其中包含有关存储集群 / 服务的详细信息，并引用了上一步中创建的密钥。

创建后端后、您可以使用观察其状态 `kubectl get tbc <tbc-name> -n <trident-namespace>` 并收集其他详细信息。

第 1 步：创建 Kubernetes 机密

创建一个机密，其中包含后端的访问凭据。这是每个存储服务 / 平台所特有的。以下是一个示例：

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

下表汇总了每个存储平台的机密中必须包含的字段：

存储平台机密字段问题描述	机密	字段问题描述
Azure NetApp Files	clientId	应用程序注册中的客户端 ID
适用于 GCP 的 Cloud Volumes Service	private_key_id	专用密钥的 ID 。具有 CVS 管理员角色的 GCP 服务帐户的 API 密钥的一部分
适用于 GCP 的 Cloud Volumes Service	private_key	专用密钥。具有 CVS 管理员角色的 GCP 服务帐户的 API 密钥的一部分
Element （ NetApp HCI/SolidFire ）	端点	使用租户凭据的 SolidFire 集群的 MVIP
ONTAP	username	用于连接到集群 /SVM 的用户名。用于基于凭据的身份验证
ONTAP	password	连接到集群 /SVM 的密码。用于基于凭据的身份验证
ONTAP	客户端权限密钥	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证

存储平台机密字段问题描述	机密	字段问题描述
ONTAP	用户名	入站用户名。如果 useCHAP=true ，则为必需项。适用于 `ontap-san` 和 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 启动程序密钥。如果 useCHAP=true ，则为必需项。适用于 `ontap-san` 和 `ontap-san-economy`
ONTAP	chapTargetUsername	目标用户名。如果 useCHAP=true ，则为必需项。适用于 `ontap-san` 和 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 目标启动程序密钥。如果 useCHAP=true ，则为必需项。适用于 `ontap-san` 和 `ontap-san-economy`

将在中引用此步骤中创建的机密 `spec.credentials` 字段 `TridentBackendConfig` 在下一步中创建的对象。

第2步：创建 `TridentBackendConfig` CR

现在、您可以创建了 `TridentBackendConfig` CR.在此示例中、是使用的后端 `ontap-san` 驱动程序是使用创建的 `TridentBackendConfig` 对象如下所示：

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

第3步：验证的状态 TridentBackendConfig CR

现在、您创建了 TridentBackendConfig cr、您可以验证状态。请参见以下示例：

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                       BACKEND NAME           BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

已成功创建后端并将其绑定到 TridentBackendConfig CR.

阶段可以采用以下值之一：

- Bound： TridentBackendConfig CR与后端关联、后端包含 configRef 设置为 TridentBackendConfig CR的UID。
- Unbound：表示使用 ""。。 TridentBackendConfig 对象未绑定到后端。所有新创建的 TridentBackendConfig 默认情况下、CRS处于此阶段。此阶段发生更改后，它将无法再次还原为 "Unbound （已取消绑定） "。
- Deleting： TridentBackendConfig CR deletionPolicy 已设置为delete。当 TridentBackendConfig CR将被删除、它将过渡到Deleting状态。
  - 如果后端不存在永久性卷请求(PVC)、请删除 TridentBackendConfig 将导致Astra Trident删除后端以及 TridentBackendConfig CR.
  - 如果后端存在一个或多个 PVC ，则会进入删除状态。。 TridentBackendConfig CR随后也进入删除阶段。后端和 TridentBackendConfig 只有在删除所有PVC后才会删除。
- Lost：与关联的后端 TridentBackendConfig 意外或故意删除了CR和 TridentBackendConfig CR仍引用已删除的后端。。 TridentBackendConfig 无论使用什么、仍可删除CR deletionPolicy 价值。
- Unknown：Astra Trident无法确定与关联的后端的状态或是否存在 TridentBackendConfig CR.例如、如果API服务器未响应或 tridentbackends.trident.netapp.io 缺少CRD。这可能需要用户干预。

在此阶段，已成功创建后端！此外，还可以处理多个操作，例如 "后端更新和后端删除"。

（可选）第 4 步：获取更多详细信息

您可以运行以下命令来获取有关后端的详细信息：

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

此外、您还可以获取的YAML/JSON转储 `TridentBackendConfig`。

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` 包含 `backendName` 和 `backendUUID` 为响应创建的后端的 `TridentBackendConfig` CR.。`lastOperationStatus` 字段表示上次操作的状态 `TridentBackendConfig` cr、可以由用户触发(例如、用户在中更改了某些内容 `spec`)或由Astra Trident触发(例如、在Astra Trident重新启动期间)。可以是成功，也可以是失败。 `phase` 表示之间关系的状态 `TridentBackendConfig` CR和后端。在上面的示例中、 `phase` 已绑定值、这意味着 `TridentBackendConfig` CR与后端关联。

您可以运行 `kubectl -n trident describe tbc <tbc-cr-name>` 命令以获取事件日志的详细信息。

> 您不能更新或删除包含关联的后端 `TridentBackendConfig` 对象使用 `tridentctl`。了解切换所涉及的步骤 `tridentctl` 和 `TridentBackendConfig`， "请参见此处"。

使用 kubectl 执行后端管理

了解如何使用执行后端管理操作 `kubectl`。

删除后端

删除 `TridentBackendConfig`、您可以指示Astra Trident删除/保留后端(基于 `deletionPolicy`）。要删除后端、请确保 `deletionPolicy` 设置为delete。仅删除 `TridentBackendConfig`、请确保 `deletionPolicy` 设置为保留。这样可以确保后端仍然存在、并可使用进行管理 `tridentctl`。

运行以下命令：

```
kubectl delete tbc <tbc-name> -n trident
```

Astra Trident不会删除正在使用的Kubernetes机密 `TridentBackendConfig`。Kubernetes 用户负责清理密钥。删除机密时必须小心。只有在后端未使用机密时，才应将其删除。

查看现有后端

运行以下命令：

```
kubectl get tbc -n trident
```

您也可以运行 `tridentctl get backend -n trident` 或 `tridentctl get backend -o yaml -n trident` 获取所有后端的列表。此列表还将包括使用创建的后端 `tridentctl`。

更新后端

更新后端可能有多种原因：

- 存储系统的凭据已更改。要更新凭据、请使用中使用的Kubernetes Secret `TridentBackendConfig` 必须更新对象。Astra Trident 将使用提供的最新凭据自动更新后端。运行以下命令以更新 Kubernetes Secret：

```
kubectl apply -f <updated-secret-file.yaml> -n trident
```

- 需要更新参数（例如所使用的 ONTAP SVM 的名称）。在这种情况下、 `TridentBackendConfig` 可以直接通过Kubernetes更新对象。

```
kubectl apply -f <updated-backend-file.yaml>
```

或者、也可以对现有进行更改 `TridentBackendConfig` cr运行以下命令：

```
kubectl edit tbc <tbc-name> -n trident
```

如果后端更新失败，则后端仍会保持在其上次已知配置中。您可以通过运行来查看日志以确定发生原因 `kubectl get tbc <tbc-name> -o yaml -n trident` 或 `kubectl describe tbc <tbc-name> -n trident`。

确定并更正配置文件中的问题后，您可以重新运行 update 命令。

使用 tridentctl 执行后端管理

了解如何使用执行后端管理操作 `tridentctl`。

创建后端

创建后 "后端配置文件"下，运行以下命令：

```
tridentctl create backend -f <backend-file> -n trident
```

如果后端创建失败，则后端配置出现问题。您可以运行以下命令来查看日志以确定发生原因：

```
tridentctl logs -n trident
```

确定并更正配置文件中的问题后、您只需运行即可 `create` 命令。

删除后端

要从 Astra Trident 中删除后端，请执行以下操作：

1. 检索后端名称：

   ```
   tridentctl get backend -n trident
   ```

2. 删除后端：

   ```
   tridentctl delete backend <backend-name> -n trident
   ```

如果 Astra Trident 从此后端配置了仍存在的卷和快照，则删除后端将阻止其配置新卷。后端将继续处于 " 删除 " 状态，而 Trident 将继续管理这些卷和快照，直到将其删除为止。

查看现有后端

要查看 Trident 了解的后端，请执行以下操作：

- 要获取摘要，请运行以下命令：

```
tridentctl get backend -n trident
```

- 要获取所有详细信息，请运行以下命令：

```
tridentctl get backend -o json -n trident
```

更新后端

创建新的后端配置文件后，运行以下命令：

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

如果后端更新失败，则后端配置出现问题或您尝试的更新无效。您可以运行以下命令来查看日志以确定发生原因：

```
tridentctl logs -n trident
```

确定并更正配置文件中的问题后、您只需运行即可 `update` 命令。

确定使用后端的存储类

这是一个示例、说明您可以通过问题解答 与JSON一起提出的问题 `tridentctl` 后端对象的输出。此操作将使用 `jq` 实用程序、您需要安装该实用程序。

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

这也适用于使用创建的后端 `TridentBackendConfig`。

在后端管理选项之间移动

了解如何在 Astra Trident 中管理后端。随附 `TridentBackendConfig`现在、管理员可以通过两种独特的方式管理

后端。这会提出以下问题：

- 可以使用创建后端 `tridentctl` 通过进行管理 `TridentBackendConfig`?
- 可以使用创建后端 `TridentBackendConfig` 可使用进行管理 `tridentctl`?

管理 `tridentctl` 后端使用 `TridentBackendConfig`

本节介绍管理使用创建的后端所需的步骤 `tridentctl` 通过创建直接通过Kubernetes界面 `TridentBackendConfig` 对象。

这适用于以下情形：

- 预先存在的后端、没有 `TridentBackendConfig` 因为它们是使用创建的 `tridentctl`。
- 使用创建的新后端 `tridentctl`而其他 `TridentBackendConfig` 对象存在。

在这两种情况下，后端仍会存在，其中 Astra Trident 会计划卷并对其进行操作。管理员可以选择以下两种方式之一：

- 继续使用 `tridentctl` 以管理使用它创建的后端。
- 使用创建的绑定后端 `tridentctl` 到新的 `TridentBackendConfig` 对象。这样做意味着后端将使用进行管理 `kubectl` 而不是 `tridentctl`。

使用管理已有后端 `kubectl`、您需要创建 `TridentBackendConfig` 绑定到现有后端。下面简要介绍了它的工作原理：

1. 创建 Kubernetes 机密。此密钥包含 Astra Trident 与存储集群 / 服务通信所需的凭据。
2. 创建 `TridentBackendConfig` 对象。其中包含有关存储集群 / 服务的详细信息，并引用了上一步中创建的密钥。必须小心指定相同的配置参数(例如 `spec.backendName`， `spec.storagePrefix`，`spec.storageDriverName`等)。 `spec.backendName` 必须设置为现有后端的名称。

第 0 步：确定后端

以创建 `TridentBackendConfig` 如果绑定到现有后端、则需要获取后端配置。在此示例中，假设已使用以下 JSON 定义创建了后端：

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+---------+----------+
|           NAME        | STORAGE DRIVER |                  UUID
| STATE   | VOLUMES |
+---------------------+----------------
+--------------------------------------+---------+----------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+---------+----------+
```

```
cat ontap-nas-backend.json

{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.10.10.1",
    "dataLIF": "10.10.10.2",
    "backendName": "ontap-nas-backend",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "admin-password",

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

第 1 步：创建 Kubernetes 机密

创建一个包含后端凭据的机密，如以下示例所示：

```
cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password

kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

第2步：创建 `TridentBackendConfig` CR

下一步是创建 `TridentBackendConfig` 将自动绑定到已有的的CR `ontap-nas-backend` (如本示例所示)。确保满足以下要求：

- 中定义了相同的后端名称 `spec.backendName`。
- 配置参数与原始后端相同。
- 虚拟池(如果存在)必须与原始后端的顺序相同。
- 凭据通过 Kubernetes Secret 提供，而不是以纯文本形式提供。

在这种情况下、将显示 `TridentBackendConfig` 将如下所示：

```
cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

第3步：验证的状态 `TridentBackendConfig` CR

在之后 `TridentBackendConfig` 已创建、其阶段必须为 `Bound`。它还应反映与现有后端相同的后端名称和 UUID 。

```
kubectl -n trident get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

现在、后端将使用进行完全管理 `tbc-ontap-nas-backend` `TridentBackendConfig` 对象。

管理 `TridentBackendConfig` 后端使用 `tridentctl`

`tridentctl` 可用于列出使用创建的后端 `TridentBackendConfig`。此外、管理员还可以选择通过完全管理此类后端 `tridentctl` 删除 `TridentBackendConfig` 并确保 `spec.deletionPolicy` 设置为 `retain`。

第 0 步：确定后端

例如、假设以下后端是使用创建的 `TridentBackendConfig`：

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

从输出中可以看出这一点 `TridentBackendConfig` 已成功创建并绑定到后端[观察后端的UUUUID]。

第1步：确认 `deletionPolicy` 设置为 `retain`

让我们来了解一下的价值 `deletionPolicy`。此值需要设置为 `retain`。这样可以确保在出现时 `TridentBackendConfig` CR已删除、后端定义仍存在、可使用进行管理 `tridentctl`。

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

请勿继续执行下一步、除非 `deletionPolicy` 设置为 `retain`。

第2步：删除 `TridentBackendConfig` CR

最后一步是删除 `TridentBackendConfig` CR.确认后 `deletionPolicy` 设置为 `retain`、您可以继续执行删除操作：

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

删除时 `TridentBackendConfig` 对象、Astra Trident只需将其删除、而不实际删除后端本身。

管理存储类

查找有关创建存储类，删除存储类以及查看现有存储类的信息。

设计存储类

请参见 "此处" 有关什么是存储类以及如何配置这些类的详细信息，请参见。

创建存储类。

创建存储类文件后，运行以下命令：

```
kubectl create -f <storage-class-file>
```

`<storage-class-file>` 应替换为存储类文件名。

删除存储类

要从 Kubernetes 中删除存储类，请运行以下命令：

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` 应替换为您的存储类。

通过此存储类创建的任何永久性卷将保持不变， Astra Trident 将继续对其进行管理。

> Astra Trident强制使用空 `fsType` 创建的卷。对于iSCSI后端、建议强制实施 `parameters.fsType` 在StorageClass中。您应删除现有StorageClasses并使用重新创建它们 `parameters.fsType` 已指定。

查看现有存储类

- 要查看现有 Kubernetes 存储类，请运行以下命令：

```
kubectl get storageclass
```

- 要查看 Kubernetes 存储类详细信息，请运行以下命令：

```
kubectl get storageclass <storage-class> -o json
```

- 要查看 Astra Trident 的同步存储类，请运行以下命令：

```
tridentctl get storageclass
```

- 要查看 Astra Trident 的同步存储类详细信息，请运行以下命令：

```
tridentctl get storageclass <storage-class> -o json
```

设置默认存储类

Kubernetes 1.6 增加了设置默认存储类的功能。如果用户未在永久性卷声明（ PVC ）中指定永久性卷，则此存储类将用于配置永久性卷。

- 通过设置标注来定义默认存储类 `storageclass.kubernetes.io/is-default-class` 在存储类定义中为true。根据规范，任何其他值或标注不存在均视为 false 。
- 您可以使用以下命令将现有存储类配置为默认存储类：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 同样，您也可以使用以下命令删除默认存储类标注：

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Trident 安装程序包中也有包含此标注的示例。

> 在任何给定时间，集群中只能有一个默认存储类。Kubernetes 在技术上不会阻止您拥有多个存储类，但其行为就像根本没有默认存储类一样。

确定存储类的后端

这是一个示例、说明您可以通过问题解答 与JSON一起提出的问题 `tridentctl` Astra Trident后端对象的输出。此操作将使用 `jq` 实用程序、您可能需要先安装此实用程序。

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

执行卷操作

了解 Astra Trident 为管理卷提供的功能。

- "使用 CSI 拓扑"
- "使用快照"
- "展开卷"
- "导入卷"

使用 **CSI** 拓扑

Astra Trident 可以通过使用有选择地创建卷并将其附加到 Kubernetes 集群中的节点 "CSI 拓扑功能"。使用 CSI 拓扑功能，可以根据区域和可用性区域将对卷的访问限制为一小部分节点。如今，借助云提供商， Kubernetes 管理员可以生成基于分区的节点。节点可以位于一个区域内的不同可用性区域中，也可以位于不同区域之间。为了便于在多区域架构中为工作负载配置卷， Astra Trident 使用了 CSI 拓扑。

> 了解有关 CSI 拓扑功能的更多信息 "此处"。

Kubernetes 提供了两种唯一的卷绑定模式：

- 使用 `VolumeBindingMode` 设置为 `Immediate`、Astra Trident将创建卷、而不会感知任何拓扑。创建 PVC 时会处理卷绑定和动态配置。这是默认值 `VolumeBindingMode` 和适用于不强制实施拓扑限制的集群。创建永久性卷时，不会依赖于请求的 Pod 的计划要求。
- 使用 `VolumeBindingMode` 设置为 `WaitForFirstConsumer`、在计划和创建使用PVC的Pod之前、将延迟为PVC创建和绑定永久性卷。这样，卷就会根据拓扑要求强制实施的计划限制来创建。

> 。 `WaitForFirstConsumer` 绑定模式不需要拓扑标签。此功能可独立于 CSI 拓扑功能使用。

您需要的内容

要使用 CSI 拓扑，您需要满足以下条件：

- 运行的Kubernetes集群 "支持的Kubernetes版本"

```
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"}
```

- 集群中的节点应具有可引入拓扑感知的标签 (`topology.kubernetes.io/region` 和 `topology.kubernetes.io/zone`）。在安装 Astra Trident 之前，集群中的节点上应存在这些标签 * ，以使 Astra Trident 能够识别拓扑。

```
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"}]
```

第 1 步：创建可感知拓扑的后端

可以设计 Astra Trident 存储后端，以便根据可用性区域有选择地配置卷。每个后端都可以具有一个可选的 `supportedTopologies` 表示必须支持的分区和区域列表的块。对于使用此后端的 StorageClasses ，只有在受支持区域 / 区域中计划的应用程序请求时，才会创建卷。

下面是一个后端定义示例：

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` 用于提供每个后端的区域和分区列表。这些区域和分区表示可在 StorageClass 中提供的允许值列表。对于包含后端提供的部分区域和分区的 StorageClasses ，Astra Trident 将在后端创建卷。

您可以定义 `supportedTopologies` 也是每个存储池的一个。请参见以下示例：

```
---
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
```

在此示例中、将显示 `region` 和 `zone` 标签表示存储池的位置。 `topology.kubernetes.io/region` 和 `topology.kubernetes.io/zone` 指定存储池的使用位置。

第 2 步：定义可识别拓扑的 StorageClasses

根据为集群中的节点提供的拓扑标签，可以将 StorageClasses 定义为包含拓扑信息。这将确定用作 PVC 请求候选对象的存储池，以及可使用 Trident 配置的卷的节点子集。

请参见以下示例：

```
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"
```

在上述StorageClass定义中、 `volumeBindingMode` 设置为 `WaitForFirstConsumer`。在此存储类中请求的 PVC 在 Pod 中引用之前不会执行操作。和、 `allowedTopologies` 提供要使用的分区和区域。。 `netapp-san-us-east1` StorageClass将在上创建PVC `san-backend-us-east1` 上述定义的后端。

第 **3** 步：创建和使用 **PVC**

创建 StorageClass 并将其映射到后端后，您现在可以创建 PVC 。

请参见示例 `spec` 以下：

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

使用此清单创建 PVC 将导致以下结果：

```
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
```

要使 Trident 创建卷并将其绑定到 PVC ，请在 Pod 中使用 PVC 。请参见以下示例：

```
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
```

此podSpec指示Kubernetes在中的节点上计划Pod `us-east1` 区域、然后从中的任何节点中进行选择 `us-east1-a` 或 `us-east1-b` 分区。

请参见以下输出：

```
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
```

更新后端以包括 `supportedTopologies`

可以更新已有后端以包括列表 `supportedTopologies` 使用 `tridentctl backend update`。这不会影响已配置的卷，并且仅用于后续的 PVC 。

了解更多信息

- "管理容器的资源"
- "节点选择器"
- "关联性和反关联性"
- "损害和公差"

使用快照

您可以创建永久性卷(PV)的Kubernetes VolumeSnapshot (卷快照)、以维护Astra Trident卷的时间点副本。此外、您还可以从现有卷快照创建一个新卷、也称为_clone_。支持卷快照 `ontap-nas`， `ontap-nas-flexgroup`， `ontap-san`， `ontap-san-economy`， `solidfire-san`， `gcp-cvs`，和 `azure-netapp-files` 驱动程序。

开始之前

您必须具有外部快照控制器和自定义资源定义(CRD)。这是Kubernetes流程编排程序(例如：Kubeadm、GKE、OpenShift)的职责。

如果您的Kubernetes分发版不包含快照控制器和CRD、请参见 [部署卷快照控制器]。

> 如果在GKE-环境中创建按需卷快照、请勿创建快照控制器。GKE-使用内置的隐藏快照控制器。

第**1**步：创建 `VolumeSnapshotClass`

此示例将创建一个卷快照类。

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

。 `driver` 指向Astra Trident的CSI驱动程序。 `deletionPolicy` 可以是 `Delete` 或 `Retain`。设置为时 `Retain`、存储集群上的底层物理快照会保留、即使在使用时也是如此 `VolumeSnapshot` 对象已删除。

有关详细信息、请参见链接：./trident引用/objects.html#Kubernetes -volumesnapshotclass-objects[`VolumeSnapshotClass`]。

第 **2** 步：创建现有 **PVC** 的快照

此示例将创建现有PVC的快照。

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

在此示例中、为名为的PVC创建快照 `pvc1` 快照的名称设置为 `pvc1-snap`。

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

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

这就创建了 `VolumeSnapshot` 对象。VolumeSnapshot类似于PVC、并与关联 `VolumeSnapshotContent` 表示实际快照的对象。

可以标识 `VolumeSnapshotContent` 的对象 `pvc1-snap` VolumeSnapshot的说明。

```
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
.
.
```

。 `Snapshot Content Name` 标识提供此快照的VolumeSnapshotContent对象。。 `Ready To Use` 参数表示可使用Snapshot创建新的PVC。

第 3 步：从 VolumeSnapshots 创建 PVC

以下示例将使用快照创建PVC：

```
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` 显示必须使用名为的VolumeSnapshot创建PVC `pvc1-snap` 作为数据源。此操作将指示 Astra Trident 从快照创建 PVC 。创建 PVC 后，可以将其附加到 Pod 上，并像使用任何其他 PVC 一样使用。

> 删除具有关联快照的永久性卷时，相应的 Trident 卷将更新为 " 正在删除 " 状态。要删除 Astra Trident 卷，应删除该卷的快照。

部署卷快照控制器

如果您的Kubernetes分发版不包含快照控制器和CRD、则可以按如下所示进行部署。

步骤

1. 创建卷快照CRD。

```
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. 在所需命名空间中创建Snapshot控制器。编辑以下 YAML 清单以修改命名空间。

```
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
```

相关链接

- "卷快照"
- "VolumeSnapshotClass"

展开卷

通过 Astra Trident ， Kubernetes 用户可以在创建卷后对其进行扩展。查找有关扩展 iSCSI 和 NFS 卷所需配置的信息。

展开 **iSCSI** 卷

您可以使用 CSI 配置程序扩展 iSCSI 永久性卷（ PV ）。

> 支持iSCSI卷扩展 `ontap-san`， `ontap-san-economy`， `solidfire-san` 驱动程序并需要Kubernetes 1.16及更高版本。

概述

扩展 iSCSI PV 包括以下步骤：

- 编辑StorageClass定义以设置 `allowVolumeExpansion` 字段设置为 `true`。
- 编辑PVC定义并更新 `spec.resources.requests.storage` 以反映新需要的大小、该大小必须大于原始大小。
- 要调整 PV 大小，必须将 PV 连接到 Pod 。调整 iSCSI PV 大小时，有两种情况：
  - 如果 PV 连接到 Pod ，则 Astra Trident 会扩展存储后端的卷，重新扫描设备并调整文件系统大小。
  - 尝试调整未连接 PV 的大小时， Astra Trident 会扩展存储后端的卷。将 PVC 绑定到 Pod 后， Trident 会重新扫描设备并调整文件系统大小。然后， Kubernetes 会在扩展操作成功完成后更新 PVC 大小。

以下示例显示了扩展 iSCSI PV 的工作原理。

第 **1** 步：配置 **StorageClass** 以支持卷扩展

```
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
```

对于已存在的StorageClass、请对其进行编辑以包括 `allowVolumeExpansion` 参数。

第 **2** 步：使用您创建的 **StorageClass** 创建 **PVC**

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

Astra Trident 会创建一个永久性卷（ PV ）并将其与此永久性卷声明（ PVC ）关联。

```
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
```

第 3 步：定义连接 PVC 的 POD

在此示例中、创建了一个使用的POD `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
```

第 4 步：展开 PV

要将已创建的PV从1Gi调整为2Gi、请编辑PVC定义并更新 `spec.resources.requests.storage` 至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
 ...
```

第 **5** 步：验证扩展

您可以通过检查 PVC ， PV 和 Astra Trident 卷的大小来验证扩展是否正常运行：

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

展开 **NFS** 卷

Astra Trident支持对上配置的NFS PV进行卷扩展 `ontap-nas`， `ontap-nas-economy`， `ontap-nas-flexgroup`， `gcp-cvs`，和 `azure-netapp-files` 后端。

第 **1** 步：配置 **StorageClass** 以支持卷扩展

要调整NFS PV的大小、管理员首先需要通过设置来配置存储类以允许卷扩展 `allowVolumeExpansion` 字段设置为 `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
```

如果您已创建没有此选项的存储类、则只需使用编辑现有存储类即可 `kubectl edit storageclass` 以允许卷扩展。

第 2 步：使用您创建的 StorageClass 创建 PVC

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

Astra Trident 应为此 PVC 创建一个 20 MiB NFS PV：

```
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
```

第 3 步：展开 PV

要将新创建的20MiB PV调整为1GiB、请编辑PVC并进行设置 `spec.resources.requests.storage` 到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
...
```

第 4 步：验证扩展

您可以通过检查 PVC ， PV 和 Astra Trident 卷的大小来验证调整大小是否正常工作：

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

导入卷

您可以使用将现有存储卷作为Kubernetes PV导入 `tridentctl import`。

支持卷导入的驱动程序

下表介绍了支持导入卷的驱动程序及其引入的版本。

驱动程序	版本。
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04
`gcp-cvs`	19.04

驱动程序	版本。
ontap-san	19.04

为什么应导入卷?

将卷导入到 Trident 的使用情形有多种：

- 对应用程序进行容器化并重复使用其现有数据集
- 为临时应用程序使用数据集的克隆
- 重建发生故障的 Kubernetes 集群
- 在灾难恢复期间迁移应用程序数据

导入的工作原理是什么?

卷导入过程使用永久性卷声明（ PVC ）文件创建 PVC 。PVC 文件应至少包含 name ， namespace ， accessModes 和 storageClassName 字段，如以下示例所示。

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

。 `tridentctl` 客户端用于导入现有存储卷。Trident 通过保留卷元数据并创建 PVC 和 PV 来导入卷。

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

要导入存储卷，请指定包含该卷的 Astra Trident 后端的名称以及用于唯一标识存储上的卷的名称（例如：ONTAP FlexVol ， Element Volume ， CVS 卷路径）。存储卷必须允许读 / 写访问，并可由指定的 Astra Trident 后端访问。。 `-f` 字符串参数为必填项、用于指定YAML或JSON PVC文件的路径。

当 Astra Trident 收到导入卷请求时，系统会在 PVC 中确定并设置现有卷大小。存储驱动程序导入卷后，系统将创建 PV ，并为其创建一个 Claims Ref 。回收策略最初设置为 `retain` 在PV中。Kubernetes 成功绑定 PVC 和 PV 后，将更新回收策略以匹配存储类的回收策略。存储类的回收策略为时 `delete`、删除PV时、存储卷将被删除。

使用导入卷时 `--no-manage` 参数中、Trident不会在对象的生命周期内对PVC或PV执行任何其他操作。因为Trident会忽略的PV和PVC事件 `--no-manage` 对象、删除PV时不会删除存储卷。卷克隆和卷大小调整等其他操作也会被忽略。如果要对容器化工作负载使用 Kubernetes ，但希望在 Kubernetes 外部管理存储卷的生命周期，则此选项非常有用。

PVC 和 PV 中会添加一个标注，用于指示卷已导入以及 PVC 和 PV 是否已管理。不应修改或删除此标注。

Trident 19.07 及更高版本可处理 PV 的连接，并在导入卷时挂载该卷。对于使用早期版本的 Astra Trident 进行的导入，数据路径中不会执行任何操作，卷导入将不会验证是否可以挂载卷。如果卷导入出错(例如、StorageClass不正确)、您可以通过将PV上的回收策略更改为来恢复 `retain`、删除PVC和PV、然后重试volume import命令。

`ontap-nas` 和 `ontap-nas-flexgroup` 导入

使用创建的每个卷 `ontap-nas` 驱动程序是ONTAP 集群上的FlexVol。使用导入FlexVol `ontap-nas` 驱动程序的工作原理相同。ONTAP 集群上已存在的FlexVol 可以作为导入 `ontap-nas` PVC。同样、FlexGroup vols也可以作为导入 `ontap-nas-flexgroup` PVC。

> 要由 Trident 导入 ONTAP 卷，必须为 rw 类型。如果卷的类型为 DP ，则为 SnapMirror 目标卷；应先中断镜像关系，然后再将卷导入到 Trident 中。

> 。 `ontap-nas` 驱动程序无法导入和管理qtree。。 `ontap-nas` 和 `ontap-nas-flexgroup` 驱动程序不允许使用重复的卷名称。

例如、导入名为的卷 `managed_volume` 位于名为的后端 `ontap_nas`、请使用以下命令：

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

导入名为的卷 `unmanaged_volume` (在上 `ontap_nas backend`)、而Trident不会管理此项、请使用以下命令：

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

使用时 `--no-manage` 参数、Trident不会重命名卷或验证卷是否已挂载。如果卷未手动挂载，则卷导入操作将失败。

> 先前存在的使用自定义 UnixPermissions 导入卷的错误已得到修复。您可以在 PVC 定义或后端配置中指定 unixPermissions ，并指示 Astra Trident 相应地导入卷。

`ontap-san` 导入

Astra Trident 还可以导入包含单个 LUN 的 ONTAP SAN FlexVol 。这与一致 `ontap-san` 驱动程序、用于为FlexVol 中的每个PVC和LUN创建FlexVol。您可以使用 `tridentctl import` 命令的方式与其他情况相同：

- 包括的名称 `ontap-san` 后端。
- 提供需要导入的 FlexVol 的名称。请记住，此 FlexVol 仅包含一个必须导入的 LUN 。
- 提供必须与结合使用的PVC定义路径 `-f` 标志。
- 可以选择对 PVC 进行管理，也可以选择不对其进行管理。默认情况下， Trident 将管理 PVC 并重命名后端的 FlexVol 和 LUN 。要作为非受管卷导入、请传递 `--no-manage` 标志。

> 导入非受管时 `ontap-san` 卷中的LUN、您应确保FlexVol 中的LUN名为 `lun0` 和映射到具有所需启动程序的igroup。Astra Trident 会自动为受管导入处理此问题。

然后， Astra Trident 将导入 FlexVol 并将其与 PVC 定义关联。Astra Trident还会将FlexVol 重命名为 `pvc-<uuid>` 将FlexVol 中的LUN格式化为 `lun0`。

> 建议导入没有活动连接的卷。如果要导入当前使用的卷，请先克隆该卷，然后再执行导入。

示例

以导入 `ontap-san-managed` 上存在的FlexVol `ontap_san_default` 后端、运行 `tridentctl import` 命令为：

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

ONTAP 卷的类型必须为 RW ，才能由 Astra Trident 导入。如果卷的类型为 DP ，则为 SnapMirror 目标卷；在将卷导入到 Astra Trident 之前，应中断镜像关系。

`element` 导入

您可以使用 Trident 将 NetApp Element 软件 /NetApp HCI 卷导入到 Kubernetes 集群中。您需要提供Astra Trident后端的名称、以及作为参数的卷和PVC文件的唯一名称 `tridentctl import` 命令：

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

Element 驱动程序支持重复的卷名称。如果卷名称重复，则 Trident 的卷导入过程将返回错误。作为临时解决策，克隆卷并提供唯一的卷名称。然后导入克隆的卷。

`gcp-cvs` 导入

要在 GCP 中导入由 NetApp Cloud Volumes Service 支持的卷，请按卷路径而非名称来标识该卷。

导入 gcp-cvs 后端上的卷称为 gcpcvs_YEppr 卷路径 adroit-jolly-swift、请使用以下命令：

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

卷路径是卷导出路径中： / 之后的部分。例如、如果导出路径为 10.0.0.1:/adroit-jolly-swift、卷路径为 adroit-jolly-swift。

azure-netapp-files 导入

导入 azure-netapp-files 后端上的卷称为 azurenetappfiles_40517 卷路径 `importvol1`下，运行以下命令：

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

ANF 卷的卷路径位于： / 之后的挂载路径中。例如、如果挂载路径为 10.0.0.2:/importvol1、卷路径为 importvol1。

在命名空间之间共享NFS卷

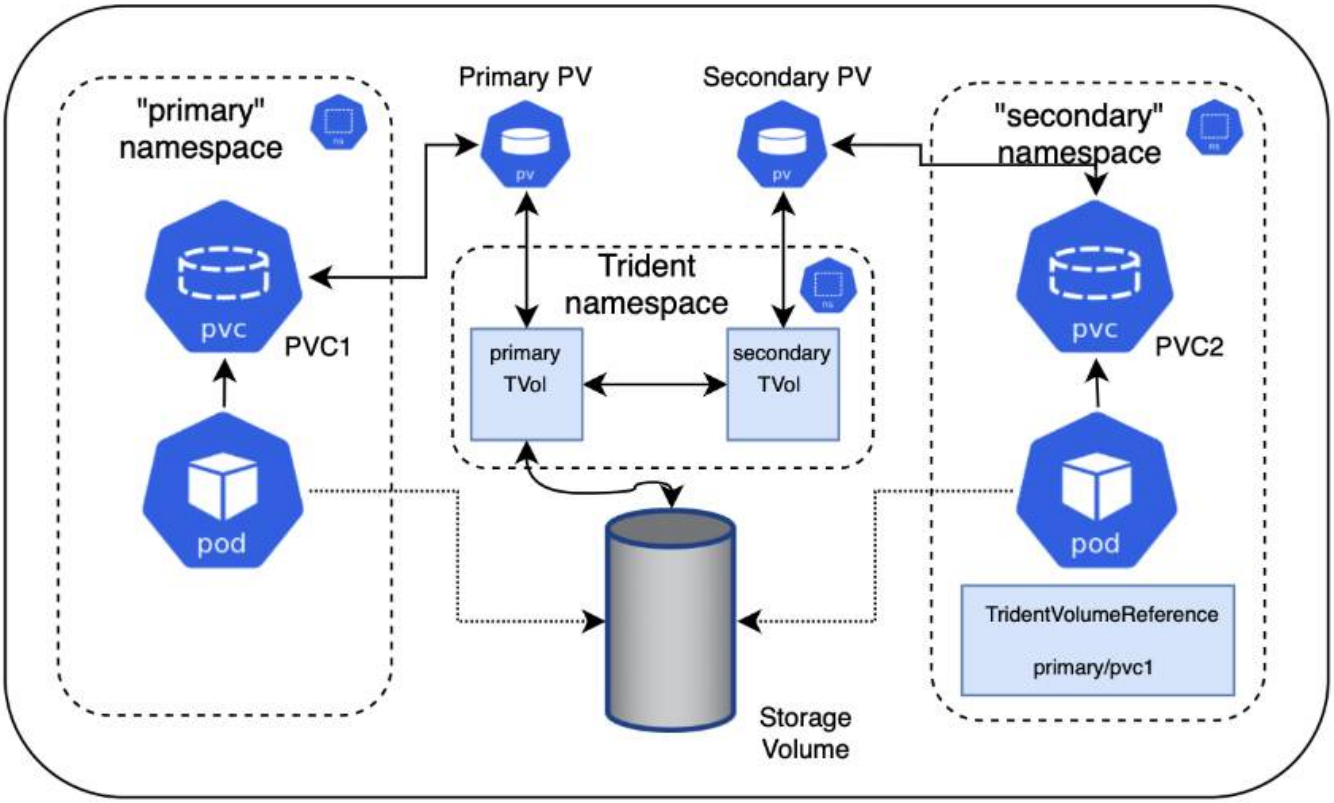
使用Astra Trident、您可以在主命名空间中创建卷、并在一个或多个二级命名空间中共享该卷。

功能

使用Astra TridentVolumeReference CR、您可以在一个或多个Kubernetes命名空间之间安全地共享ReadWriteMany (rwx) NFS卷。此Kubernetes本机解决方案 具有以下优势：

- 可通过多个级别的访问控制来确保安全性
- 适用于所有Trident NFS卷驱动程序
- 不依赖于tridentctl或任何其他非本机Kubernetes功能

此图显示了两个Kubernetes命名空间之间的NFS卷共享。



快速入门

只需几个步骤即可设置NFS卷共享。

1. 配置源**PVC**以共享卷

源命名空间所有者授予访问源PVC中数据的权限。

2 授予在目标命名空间中创建**CR**的权限

集群管理员向目标命名空间的所有者授予创建TridentVolumeReference CR的权限。

3 在目标命名空间中创建**TridentVolumeReference**

目标命名空间的所有者将创建TridentVolumeReference CR以引用源PVC。

4 在目标命名空间中创建从属**PVC**

目标命名空间的所有者创建从属PVC以使用源PVC中的数据源。

配置源和目标命名空间

为了确保安全性、跨命名空间共享需要源命名空间所有者、集群管理员和目标命名空间所有者的协作和操作。每个步骤都会指定用户角色。

步骤

1. *源命名空间所有者：*创建PVC (`pvc1`)、以授予与目标命名空间共享的权限 (`namespace2`) `shareToNamespace` 标注。

```
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会创建PV及其后端NFS存储卷。

- 您可以使用逗号分隔列表将PVC共享给多个命名空间。例如：`trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`。
- 您可以使用共享到所有命名空间 `*`。例如：`trident.netapp.io/shareToNamespace: *`
- 您可以更新PVC以包括 `shareToNamespace` 随时添加标注。

2. *集群管理员：*创建自定义角色并执行kubeconfig、以授予目标命名空间所有者在目标命名空间中创建TridentVolumeReference CR的权限。
3. *目标命名空间所有者：*在目标命名空间中创建引用源命名空间的TridentVolumeReference CR `pvc1`。

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

4. *目标命名空间所有者：*创建PVC (`pvc2`) (`namespace2`) `shareFromPVC` 用于指定源PVC的标注。

```
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
```

目标PVC的大小必须小于或等于源PVC。

结果

Astra Trident读取 `shareFromPVC` 在目标PVC上添加标注、并将目标PV创建为一个从属卷、而其自身没有指向源PV的存储资源、并共享源PV存储资源。目标PVC和PV显示为正常绑定。

删除共享卷

您可以删除跨多个命名空间共享的卷。Astra Trident将删除对源命名空间上卷的访问、并保持对共享该卷的其他命名空间的访问。删除引用卷的所有命名空间后、Astra Trident将删除该卷。

使用 ... `tridentctl get` 查询从属卷

使用[`tridentctl` 实用程序中、您可以运行 `get` 用于获取从属卷的命令。有关详细信息、请参见链接：./trident reference/tridentctl.html[`tridentctl` 命令和选项]。

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

flags

- `-h, --help`：卷帮助。
- `--parentOfSubordinate string`：将查询限制为从源卷。
- `--subordinateOf string`：将查询限制为卷的下属。

限制

- Astra Trident无法阻止目标命名空间写入共享卷。您应使用文件锁定或其他进程来防止覆盖共享卷数据。
- 您不能通过删除来撤消对源PVC的访问 `shareToNamespace` 或 `shareFromNamespace` 标注或删除 `TridentVolumeReference` CR.要撤消访问、必须删除从属PVC。
- 无法在从属卷上执行快照、克隆和镜像。

有关详细信息 ...

要了解有关跨命名空间卷访问的详细信息、请执行以下操作：

- 请访问 "在命名空间之间共享卷：对跨命名空间卷访问说Hello"。
- 观看演示 "NetAppTV"。

监控 Astra Trident

Astra Trident 提供了一组 Prometheus 指标端点，您可以使用这些端点监控 Astra Trident 的性能。

通过 Astra Trident 提供的指标，您可以执行以下操作：

- 保留有关 Astra Trident 运行状况和配置的选项卡。您可以检查操作的成功程度以及它是否能够按预期与后端进行通信。
- 检查后端使用情况信息，并了解在后端配置的卷数量以及占用的空间量等。
- 维护可用后端配置的卷数量的映射关系。
- 跟踪性能。您可以了解 Astra Trident 与后端通信并执行操作所需的时间。

> 默认情况下、Trident的指标会显示在目标端口上 `8001` 在上 `/metrics` 端点。安装 Trident 时，这些指标默认为 * 已启用 * 。

您需要的内容

- 安装了 Astra Trident 的 Kubernetes 集群。
- 一个 Prometheus 实例。可以是 "容器化 Prometheus 部署" 或者，您也可以选择将 Prometheus 作为运行 "原生应用程序"。

第 1 步：定义 Prometheus 目标

您应定义一个 Prometheus 目标以收集指标并获取有关后端 Astra Trident 管理的信息，它创建的卷等。这 "博客" 介绍如何将 Prometheus 和 Grafana 与 Astra Trident 结合使用来检索指标。博客介绍了如何在 Kubernetes 集群中以操作员身份运行 Prometheus ，以及如何创建 ServiceMonitor 来获取 Astra Trident 的指标。

第 2 步：创建 Prometheus ServiceMonitor

要使用Trident指标、您应创建一个监控的Prometheus ServiceMonitor `trident-csi` 服务并侦听 `metrics` 端口。示例 ServiceMonitor 如下所示：

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
      release: prom-operator
  spec:
    jobLabel: trident
    selector:
      matchLabels:
        app: controller.csi.trident.netapp.io
    namespaceSelector:
      matchNames:
      - trident
    endpoints:
    - port: metrics
      interval: 15s
```

此ServiceMonitor定义将检索返回的指标 `trident-csi` 服务、并专门查找 `metrics` 服务的端点。因此，Prometheus 现在已配置为了解 Astra Trident 的指标。

除了直接从Astra Trident获得的指标之外、kubelet还公开了许多指标 `kubelet_volume_*` 通过自己的指标端点查看指标。Kubelet 可以提供有关已连接的卷， Pod 及其处理的其他内部操作的信息。请参见 "此处"。

第 3 步：使用 PromQL 查询 Trident 指标

PromQL 非常适合创建返回时间序列或表格数据的表达式。

您可以使用以下 PromQL 查询：

获取 **Trident** 运行状况信息

- 来自 Astra Trident 的 HTTP 2XX 响应的百分比

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- 通过状态代码来自 Astra Trident 的 REST 响应的百分比

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- * 由 Astra Trident 执行的操作的平均持续时间（毫秒） *

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

获取 **Astra Trident** 使用信息

- 卷大小 * 平均值 *

```
trident_volume_allocated_bytes/trident_volume_count
```

- * 每个后端配置的卷总空间 *

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

获取单个卷的使用情况

只有在同时收集 kubelet 指标时，才会启用此功能。

- * 每个卷的已用空间百分比 *

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

了解有关 **Astra Trident AutoSupport** 遥测的信息

默认情况下， Astra Trident 会按每日节奏向 NetApp 发送 Prometheus 指标和基本后端信息。

- 要阻止Astra Trident向NetApp发送Prometheus指标和基本后端信息、请传递 `--silence-autosupport` 在Astra Trident安装期间标记。
- Astra Trident还可以根据需要将容器日志发送到NetApp支持部门 `tridentctl send autosupport`。您需要触发 Astra Trident 以上传其日志。在提交日志之前，您应接受 NetApp 的https://www.netapp.com/company/legal/privacy-policy/["隐私政策"^]。
- 除非另有说明，否则 Astra Trident 会从过去 24 小时提取日志。
- 您可以使用指定日志保留时间范围 `--since` 标志。例如： `tridentctl send autosupport --since=1h`。此信息通过收集和发送 `trident-autosupport` 与Astra Trident一起安装的容器。您可以从获取容器映像 "Trident AutoSupport"。
- Trident AutoSupport 不会收集或传输个人身份信息（ PiII ）或个人信息。它附带了 "EULA" 这不适用于 Trident 容器映像本身。您可以详细了解 NetApp 对数据安全和信任的承诺 "此处"。

Astra Trident 发送的有效负载示例如下：

```
---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- AutoSupport 消息将发送到 NetApp 的 AutoSupport 端点。如果您使用私有注册表存储容器映像、则可以使用 `--image-registry` 标志。
- 您也可以通过生成安装 YAML 文件来配置代理 URL 。可以使用完成此操作 `tridentctl install --generate-custom-yaml` 创建YAML文件并添加 `--proxy-url` 的参数 `trident-autosupport` 容器 `trident-deployment.yaml`。

禁用 **Astra Trident** 指标

要*禁止报告指标、应使用生成自定义YAML `--generate-custom-yaml` 标志)并对其进行编辑以删除 `--metrics` 用于调用的标志 `trident-main`容器。