使用Trident

Trident

NetApp
January 15, 2026

目录

使用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您为 Pod 配置的卷。要准备工作节点，您必须根据所选驱动程序安装 NFS、iSCSI、NVMe/TCP 或 FC 工具。

选择合适的工具

如果您使用的是多种驱动程序，则应安装所有驱动程序所需的工具。最新版本的 Red Hat Enterprise Linux CoreOS (RHCOS) 默认安装了这些工具。

NFS 工具

"安装 NFS 工具"如果您正在使用： `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， `azure-netapp-files` ， `gcp-cvs` 。

iSCSI 工具

"安装 iSCSI 工具"如果您正在使用： `ontap-san` ， `ontap-san-economy` ， `solidfire-san` 。

NVMe 工具

"安装 NVMe 工具"如果你正在使用 `ontap-san`用于基于 TCP 的非易失性存储器高速接口 (NVMe) (NVMe/TCP) 协议。

NetApp建议 NVMe/TCP 使用ONTAP 9.12 或更高版本。

通过光纤通道 **(FC)** 进行 **SCSI** 的工具

请参阅"配置 FC 和 FC-NVMe SAN 主机的方法"有关配置 FC 和 FC-NVMe SAN 主机的更多信息。

"安装 FC 工具"如果你正在使用 `ontap-san`使用 `sanType` `fcp` （通过光纤通道进行 SCSI 通信）。

需要考虑的要点：* OpenShift 和 KubeVirt 环境支持通过 FC 进行 SCSI 通信。 * Docker 不支持通过 FC 传输 SCSI。 * iSCSI 自愈功能不适用于基于 FC 的 SCSI。

节点服务发现

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

节点服务发现功能可以识别已发现的服务，但不能保证服务配置正确。反之，未发现服务并不保证卷挂载一定会失败。

回顾事件

Trident会为节点创建事件，以识别已发现的服务。要查看这些事件，请运行：

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

查看已发现的服务

Trident识别Trident节点 CR 上每个节点启用的服务。要查看已发现的服务，请运行：

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

NFS卷

使用适用于您操作系统的命令安装 NFS 工具。确保NFS服务在启动时启动。

RHEL 8+

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

Ubuntu

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

安装 NFS 工具后重启工作节点，以防止将卷附加到容器时发生故障。

iSCSI 卷

Trident可以自动建立 iSCSI 会话、扫描 LUN、发现多路径设备、格式化设备并将其挂载到 pod 中。

iSCSI自愈能力

对于ONTAP系统， Trident每五分钟运行一次 iSCSI 自愈程序，以：

1. *确定*所需的 iSCSI 会话状态和当前的 iSCSI 会话状态。
2. 将理想状态与当前状态进行比较，以确定需要进行的维修。 Trident确定维修优先级以及何时进行抢修。
3. 执行必要的修复，使当前的 iSCSI 会话状态恢复到所需的 iSCSI 会话状态。

自愈活动的日志位于…… `trident-main`相应 Daemonset pod 上的容器。要查看日志，您必须已设置 `debug`在Trident安装过程中设置为“true”。

Trident iSCSI 的自愈功能可以帮助预防：

- 网络连接问题后可能出现过期或不健康的 iSCSI 会话。如果会话已过期， Trident会等待七分钟，然后注销并重新与门户建立连接。

例如，如果存储控制器上轮换了 CHAP 密钥，并且网络失去连接，则旧的（过时的）CHAP 密钥可能会保留下来。自愈功能可以识别这一点，并自动重新建立会话以应用更新后的 CHAP 密钥。

- 缺少 iSCSI 会话
- 缺少 LUN

升级**Trident**之前需要考虑的要点

- 如果仅使用每个节点的 igroup（在 23.04+ 中引入），则 iSCSI 自愈功能将启动 SCSI 总线上所有设备的 SCSI 重新扫描。
- 如果仅使用后端范围的 igroup（自 23.04 版本起已弃用），则 iSCSI 自愈功能将启动 SCSI 重新扫描，以查找 SCSI 总线上的确切 LUN ID。
- 如果同时使用节点级 igroup 和后端级 igroup，iSCSI 自愈功能将启动 SCSI 重新扫描，以查找 SCSI 总线上的精确 LUN ID。

安装 **iSCSI** 工具

使用适用于您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有一个唯一的 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 发行版，则需要执行以下操作：`solidfire-san`对于驱动程序和Element OS 12.5 或更早版本，请确保将 CHAP 身份验证算法设置为 MD5。 `/etc/iscsi/iscsid.conf` Element 12.7 提供符合 FIPS 标准的 CHAP 安全算法 SHA1 、SHA-256 和 SHA3-256。

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

- 当使用运行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且带有 iSCSI PV 的工作节点时，请指定 `discard`StorageClass 中的 mountOption 用于执行内联空间回收。参考 "红帽文档"。
- 请确保您已升级到最新版本。 `multipath-tools` 。

RHEL 8+

1. 安装以下系统软件包：

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

2. 请检查 iscsi-initiator-utils 版本是否为 6.2.0.874-2.el7 或更高版本：

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

3. 将扫描方式设置为手动：

```
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`在下面 `defaults`。

5. 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. 请检查 open-iscsi 版本是否为 2.0.874-5ubuntu2.10 或更高版本（适用于 bionic）或 2.0.874-7.1ubuntu6.1 或更高版本（适用于 focal）：

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

3. 将扫描方式设置为手动：

```
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`在下面 `defaults。

5. 确保 `open-iscsi`和 `multipath-tools`已启用并正在运行：

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

> 对于 Ubuntu 18.04，您必须使用以下命令发现目标端口： `iscsiadm`开始之前 `open-iscsi`启动 iSCSI 守护进程。您也可以修改 `iscsi`服务启动 `iscsid`自动地。

配置或禁用 **iSCSI** 自愈

您可以配置以下Trident iSCSI 自愈设置来修复过期的会话：

- **iSCSI** 自愈间隔：确定 iSCSI 自愈的调用频率（默认值：5 分钟）。你可以通过设置较小的数字来增加运行频率，或者通过设置较大的数字来降低运行频率。

> 将 iSCSI 自愈间隔设置为 0 将完全停止 iSCSI 自愈功能。我们不建议禁用 iSCSI 自愈功能；只有在 iSCSI 自愈功能无法按预期工作或出于调试目的时，才应禁用该功能。

- **iSCSI** 自愈等待时间：确定 iSCSI 自愈在注销不健康的会话并尝试再次登录之前等待的时间（默认值：7 分

钟）。您可以将其配置为更大的数字，以便将识别为不健康的会话在注销之前等待更长时间，然后再尝试重新登录；或者配置为更小的数字，以便更快地注销和登录。

舵

要配置或更改 iSCSI 自愈设置，请传递以下参数： `iscsiSelfHealingInterval`和 `iscsiSelfHealingWaitTime`Helm 安装或 Helm 更新期间的参数。

以下示例将 iSCSI 自愈间隔设置为 3 分钟，自愈等待时间设置为 6 分钟：

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

三叉戟

要配置或更改 iSCSI 自愈设置，请传递以下参数： `iscsi-self-healing-interval`和 `iscsi-self-healing-wait-time`tridentctl 安装或更新期间的参数。

以下示例将 iSCSI 自愈间隔设置为 3 分钟，自愈等待时间设置为 6 分钟：

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe/TCP 卷

使用适用于您操作系统的命令安装 NVMe 工具。

- NVMe 需要 RHEL 9 或更高版本。
- 如果您的 Kubernetes 节点的内核版本太旧，或者您的内核版本没有 NVMe 软件包，则您可能需要将节点的内核版本更新为包含 NVMe 软件包的版本。

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

验证安装

安装完成后，使用以下命令验证 Kubernetes 集群中的每个节点是否都具有唯一的 NQN：

```
cat /etc/nvme/hostnqn
```

Trident修改了 `ctrl_device_tmo`确保 NVMe 在路径中断时不会放弃路径的值。请勿更改此设置。

通过 FC 卷进行 SCSI 连接

现在您可以使用光纤通道 (FC) 协议和Trident在ONTAP系统上配置和管理存储资源。

前提条件

配置 FC 所需的网络和节点设置。

网络设置

1. 获取目标接口的 WWPN。请参阅 "network interface show"了解更多信息。
2. 获取发起方（主机）上接口的 WWPN。

   请参考相应的主机操作系统实用程序。

3. 使用主机和目标的 WWPN 在 FC 交换机上配置区域。

   有关信息，请参阅相应交换机供应商的文档。

   详情请参阅以下ONTAP文档：

   - "光纤通道和FCoE分区概述"
   - "配置 FC 和 FC-NVMe SAN 主机的方法"

安装 **FC** 工具

使用适用于您操作系统的命令安装 FC 工具。

- 当使用运行 RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) 且带有 FC PV 的工作节点时，请指定 `discard`StorageClass 中的 mountOption 用于执行内联空间回收。参考 "红帽文档"。

RHEL 8+

1. 安装以下系统软件包：

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. 启用多路径：

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

> 确保 /etc/multipath.conf`包含 `find_multipaths no`在下面 `defaults。

3. 确保 `multipathd`正在运行：

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

Ubuntu

1. 安装以下系统软件包：

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

2. 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`在下面 `defaults。

3. 确保 `multipath-tools`已启用并正在运行：

```
sudo systemctl status multipath-tools
```

配置和管理后端

配置后端

后端定义了Trident与存储系统之间的关系。它告诉Trident如何与该存储系统通信，以及Trident应该如何从中配置卷。

Trident会自动提供符合存储类定义要求的后端存储池。了解如何配置存储系统的后端。

- "配置Azure NetApp Files后端"
- "配置Google Cloud NetApp Volumes后端"
- "为 Google Cloud Platform 后端配置Cloud Volumes Service"
- "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Trident与Amazon FSx for NetApp ONTAP"

Azure NetApp Files

配置**Azure NetApp Files**后端

您可以将Azure NetApp Files配置为Trident的后端。您可以使用Azure NetApp Files后端附加 NFS 和 SMB 卷。 Trident还支持使用托管标识对 Azure Kubernetes 服务 (AKS) 集群进行凭据管理。

Azure NetApp Files驱动程序详细信息

Trident提供以下Azure NetApp Files存储驱动程序，以便与集群通信。支持的访问模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`azure-netapp-files`	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	`nfs`， `smb`

注意事项

- Azure NetApp Files服务不支持小于 50 GiB 的卷。如果请求的卷较小， Trident会自动创建 50GiB 的卷。
- Trident仅支持挂载到运行在 Windows 节点上的 pod 的 SMB 卷。

为 **AKS** 管理身份

Trident支持"托管身份"适用于 Azure Kubernetes 服务集群。要利用托管身份提供的简化凭证管理功能，您必须具备以下条件：

- 使用 AKS 部署的 Kubernetes 集群
- 在 AKS Kubernetes 集群上配置的托管身份
- 已安装的Trident包括 `cloudProvider`指定 `"Azure"`。

Trident操作员

要使用Trident操作员安装Trident，请编辑 `tridentorchestrator_cr.yaml`设置 `cloudProvider`到 `"Azure"`。例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

舵

以下示例安装Trident套装 `cloudProvider`使用环境变量连接到 `Azure` `$CP`：

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

以下示例安装Trident并进行设置 `cloudProvider`标记 `Azure`：

```
tridentctl install --cloud-provider="Azure" -n trident
```

AKS 的云身份

云身份使 Kubernetes pod 能够通过作为工作负载身份进行身份验证来访问 Azure 资源，而无需提供显式的 Azure 凭据。

要在 Azure 中利用云身份功能，您必须具备以下条件：

- 使用 AKS 部署的 Kubernetes 集群
- 在 AKS Kubernetes 集群上配置工作负载身份和 oidc-issuer
- 已安装的Trident包括 `cloudProvider`指定 `"Azure"`和 `cloudIdentity`指定工作负载标识

Trident操作员

要使用Trident操作员安装Trident ，请编辑 `tridentorchestrator_cr.yaml`设置 `cloudProvider`到 `"Azure"`并设置 `cloudIdentity`到 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`。

例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

舵

使用以下环境变量设置 **cloud-provider (CP)** 和 **cloud-identity (CI)** 标志的值：

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx'"
```

以下示例安装Trident并进行设置 `cloudProvider`使用环境变量连接到 `Azure` `$CP`并设置 `cloudIdentity`使用环境变量 `$CI`：

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

请使用以下环境变量设置 云提供商 和 云身份 标志的值：

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxx"
```

以下示例安装Trident并进行设置 `cloud-provider`标记 `$CP`， 和 `cloud-identity`到 `$CI`：

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

在配置Azure NetApp Files后端之前，需要确保满足以下要求。

NFS 和 SMB 卷的先决条件

如果您是第一次使用Azure NetApp Files或在新的位置使用，则需要进行一些初始配置来设置 Azure NetApp文件并创建 NFS 卷。参考 "Azure：设置Azure NetApp Files并创建 NFS 卷"。

配置和使用 "Azure NetApp Files"后端需要以下组件：

- `subscriptionID`， `tenantID`， `clientID`， `location`， 和 `clientSecret`在 AKS 集群上使用托管身份时，这些是可选的。
- `tenantID`， `clientID`， 和 `clientSecret`在 AKS 集群上使用云身份时，这些是可选的。

- 容量池。参考"微软：为Azure NetApp Files创建容量池"。
- 委派给Azure NetApp Files 的子网。参考"Microsoft：将子网委派给Azure NetApp Files"。
- `subscriptionID`来自已启用Azure NetApp Files功能的 Azure 订阅。
- `tenantID`， `clientID`， 和 `clientSecret`来自"应用程序注册"在 Azure Active Directory 中拥有对Azure NetApp Files服务的足够权限。应用程序注册应使用以下任一方式：
  - 所有者或贡献者角色"Azure 预定义"。
  - 一个"自定义贡献者角色"订阅级别(`assignableScopes`）但权限仅限于Trident所需的权限。创建自定义角色后，"使用 Azure 门户分配角色" 。

自定义贡献者角色

```
{
  "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/
write",

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

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/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": []
      }
    ]
  }
}
```

- 蔚蓝 `location`包含至少一个 "委托子网"。截至Trident 22.01版本， `location`参数是后端配置文件顶层的一个必填字段。虚拟池中指定的位置值将被忽略。
- 使用 `Cloud Identity`得到 `client ID`从一个 "用户分配的托管身份"并指定该 `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` ID 。

中小企业卷的附加要求

要创建 SMB 卷，您必须具备以下条件：

- Active Directory 已配置并连接到Azure NetApp Files。参考"Microsoft：创建和管理Azure NetApp Files管理器的 Active Directory 连接"。
- 一个 Kubernetes 集群，包含一个 Linux 控制器节点和至少一个运行 Windows Server 2022 的 Windows 工作节点。 Trident仅支持挂载到运行在 Windows 节点上的 pod 的 SMB 卷。
- 至少需要一个包含 Active Directory 凭据的Trident密钥，以便Azure NetApp Files可以向 Active Directory 进行身份验证。生成秘密 `smbcreds`：

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

- 配置为 Windows 服务的 CSI 代理。要配置 `csi-proxy`，请参阅"GitHub：CSI代理"或者"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Azure NetApp Files的 NFS 和 SMB 后端配置选项，并查看配置示例。

Trident使用您的后端配置（子网、虚拟网络、服务级别和位置），在请求的位置中可用的容量池上创建Azure NetApp Files卷，并与请求的服务级别和子网相匹配。

* 从NetApp Trident 25.06 版本开始，手动 QoS 容量池作为技术预览版受到支持。*

Azure NetApp Files后端提供以下配置选项。

参数	描述	默认
`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	司机姓名 + "_" + 随机字符
`subscriptionID`	Azure 订阅的订阅 ID（在 AKS 群集上启用托管标识时为可选）。	
`tenantID`	在 AKS 集群上使用托管身份或云身份时，应用程序注册中的租户 ID 是可选的。	
`clientID`	在 AKS 集群上使用托管身份或云身份时，应用程序注册中的客户端 ID 是可选的。	
`clientSecret`	在 AKS 集群上使用托管身份或云身份时，应用程序注册中的客户端密钥是可选的。	
`serviceLevel`	之一 `Standard`， `Premium` ， 或者 `Ultra`	““ （随机的）
`location`	将在 Azure 中创建新卷的位置名称（在 AKS 群集上启用托管标识时为可选）。	
`resourceGroups`	用于筛选已发现资源的资源组列表	（无滤镜）
`netappAccounts`	用于筛选已发现资源的NetApp帐户列表	（无滤镜）
`capacityPools`	用于筛选已发现资源的容量池列表	（无过滤，随机）
`virtualNetwork`	具有委派子网的虚拟网络的名称	""
`subnet`	委派给的子网名称 `Microsoft.Netapp/volumes`	""
`networkFeatures`	卷的 VNet 功能集，可能是 `Basic`` 或者 `Standard`。网络功能并非在所有地区都可用，可能需要通过订阅才能启用。指定 `networkFeatures`未启用该功能会导致卷配置失败。	""

参数	描述	默认
nfsMountOptions	对 NFS 挂载选项进行精细控制。对于 SMB 卷，此设置将被忽略。要使用 NFS 版本 4.1 挂载卷，请包含以下内容 `nfsvers=4`在以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	如果请求的卷大小大于此值，则配置失败。	（默认情况下不强制执行）
debugTraceFlags	故障排除时要使用的调试标志。例子， \{"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。	无效的
nasType	配置 NFS 或 SMB 卷的创建。选项有 nfs， `smb`或空值。设置为 null 则默认使用 NFS 卷。	nfs
supportedTopologies	表示此后端支持的区域和区域列表。更多信息，请参阅"使用 CSI 拓扑"。	
qosType	表示 QoS 类型：自动或手动。* Trident 25.06 技术预览版*	自动
maxThroughput	设置允许的最大吞吐量，单位为 MiB/秒。仅支持手动 QoS 容量池。* Trident 25.06 技术预览版*	4 MiB/sec

有关网络功能的更多信息，请参阅"为Azure NetApp Files卷配置网络功能"。

所需权限和资源

如果在创建 PVC 时收到“未找到容量池”错误，则可能是您的应用程序注册没有关联的必要权限和资源（子网、虚拟网络、容量池）。如果启用调试功能， Trident将记录在创建后端时发现的 Azure 资源。请确认是否使用了合适的角色。

值 resourceGroups， netappAccounts ， capacityPools ， virtualNetwork ， 和 `subnet`可以使用简称或完全限定名称来指定。大多数情况下建议使用完全限定名称，因为短名称可能会匹配多个同名资源。

这 resourceGroups， netappAccounts ， 和 `capacityPools`值是过滤器，用于将发现的资源集限制为该存储后端可用的资源，并且可以以任意组合指定。完全限定名称遵循以下格式：

类型	格式
资源组	<资源组>
NetApp账户	<资源组>/<NetApp 帐户>
容量池	<资源组>/<NetApp 帐户>/<容量池>

类型	格式
虚拟网络	<资源组>/<虚拟网络>
子网	<资源组>/<虚拟网络>/<子网>

卷配置

您可以通过在配置文件的特定部分中指定以下选项来控制默认卷配置。参考 [示例配置] 了解详情。

参数	描述	默认
`exportRule`	新卷的出口规则。 `exportRule`必须是以逗号分隔的 IPv4 地址或 IPv4 子网的任意组合列表，采用 CIDR 表示法。对于 SMB 卷，此设置将被忽略。	“0.0.0.0/0”
`snapshotDir`	控制 .snapshot 目录的可见性	NFSv4 为“true”，NFSv3 为“false”。
`size`	新卷的默认大小	100G
`unixPermissions`	新卷的 Unix 权限（4 位八进制数字）。对于 SMB 卷，此设置将被忽略。	（预览功能，需订阅并加入白名单）

示例配置

以下示例展示了基本配置，其中大多数参数都保留默认值。这是定义后端最简单的方法。

最小配置

这是最基本的后端配置。通过此配置， Trident会发现配置位置中所有委派给Azure NetApp Files存储的NetApp帐户、容量池和子网，并将新卷随机放置在其中一个池和子网上。因为 `nasType`省略了 `nfs`默认设置生效，后端将为 NFS 卷进行配置。

如果您刚开始使用Azure NetApp Files并进行尝试，此配置是理想的选择，但在实践中，您需要为预配的卷提供额外的范围。

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  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
```

为 **AKS** 管理身份

此后端配置省略了 `subscriptionID`， `tenantID`， `clientID`， 和 `clientSecret`在使用托管身份时，这些是可选的。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

AKS 的云身份

此后端配置省略了 `tenantID`， `clientID` ， 和 `clientSecret`在使用云身份时，这些是可选的。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

具有容量池过滤器的特定服务级别配置

此后端配置将卷放置在 Azure 中 `eastus`位置 `Ultra`容量池。 Trident会自动发现该位置中委派给Azure NetApp Files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
```

具有手动 **QoS** 容量池的后端示例

此后端配置将卷放置在 Azure 中 `eastus`具有手动 QoS 容量池的位置。* NetApp Trident 25.06 中的技术预览*。

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

高级配置

此后端配置进一步缩小了卷放置范围到单个子网，并且还修改了一些卷配置默认值。

```
---
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"
```

虚拟池配置

此后端配置在单个文件中定义了多个存储池。当您有多个容量池支持不同的服务级别，并且想要在 Kubernetes 中创建代表这些级别的存储类时，这将非常有用。虚拟池标签用于根据以下因素区分池子：`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
```

支持的拓扑配置

Trident可根据区域和可用区为工作负载提供卷。这 `supportedTopologies`此后端配置中的块用于提供每个后端的区域和区域列表。此处指定的区域和区域值必须与每个 Kubernetes 集群节点上的标签中的区域和区域值相匹配。这些区域和分区代表存储类中可以提供的允许值的列表。对于包含后端提供的区域和可用区子集的存储类， Trident会在所述区域和可用区中创建卷。更多信息，请参阅"使用 CSI 拓扑"。

```
---
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
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

存储类定义

下列 `StorageClass`上述定义指的是存储池。

使用示例定义 **`parameter.selector`**场地

使用 `parameter.selector`你可以为每个对象指定。 `StorageClass`用于托管卷的虚拟池。该卷将具有所选池中定义的方面。

```
---
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。

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"
```

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}
```

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`筛选支持 SMB 卷的存储池。 `nasType: nfs`或者 `nasType: null`NFS池过滤器。

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，则后端配置存在问题。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs
```

在您发现并纠正配置文件中的问题后，您可以再次运行创建命令。

Google Cloud NetApp Volumes

配置Google Cloud NetApp Volumes后端

现在您可以将Google Cloud NetApp Volumes配置为Trident的后端。您可以使用Google Cloud NetApp Volumes后端来附加 NFS 和 SMB 卷。

Google Cloud NetApp Volumes驱动程序详情

Trident提供 `google-cloud-netapp-volumes`驱动程序与集群通信。支持的访问模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`google-cloud-netapp-volumes`	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	`nfs`， `smb`

GKE 的云身份

云身份使 Kubernetes pod 能够通过作为工作负载身份进行身份验证来访问 Google Cloud 资源，而无需提供显式的 Google Cloud 凭据。

要在 Google Cloud 中利用云身份功能，您必须具备以下条件：

- 使用 GKE 部署的 Kubernetes 集群。
- 在 GKE 集群上配置工作负载标识，并在节点池上配置 GKE 元数据服务器。
- 具有Google Cloud NetApp Volumes管理员 (roles/netapp.admin) 角色或自定义角色的 GCP 服务帐户。
- Trident已安装，其中包括 cloudProvider（指定“GCP”）和 cloudIdentity（指定新的 GCP 服务帐户）。下面给出一个例子。

Trident操作员

要使用Trident操作员安装Trident，请编辑 `tridentorchestrator_cr.yaml`设置 `cloudProvider`到 `"GCP"`并设置 `cloudIdentity`到 `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`。

例如：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com'
```

舵

使用以下环境变量设置 **cloud-provider (CP)** 和 **cloud-identity (CI)** 标志的值：

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com'"
```

以下示例安装Trident并进行设置 `cloudProvider`使用环境变量连接到 `GCP` `$CP`并设置 `cloudIdentity`使用环境变量 `$ANNOTATION`：

```
helm install trident trident-operator-100.6.0.tgz --set cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code>tridentctl</code>

请使用以下环境变量设置 云提供商 和 云身份 标志的值：

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com'"
```

以下示例安装Trident并进行设置 `cloud-provider`标记 `$CP`，和 `cloud-identity`到 `$ANNOTATION`：

```
tridentctl install --cloud-provider=$CP --cloud-identity="$ANNOTATION" -n trident
```

准备配置Google Cloud NetApp Volumes后端

在配置Google Cloud NetApp Volumes后端之前，您需要确保满足以下要求。

NFS 卷的先决条件

如果您是第一次使用Google Cloud NetApp Volumes或在新的位置使用，则需要进行一些初始配置来设置Google Cloud NetApp Volumes并创建 NFS 卷。参考"开始之前"。

配置Google Cloud NetApp Volumes后端之前，请确保您已具备以下条件：

- 已配置Google Cloud NetApp Volumes服务的 Google Cloud 帐户。参考"Google Cloud NetApp Volumes"。
- 您的 Google Cloud 帐户项目编号。参考"确定项目"。
- 拥有NetApp Volumes 管理员权限的 Google Cloud 服务帐户(`roles/netapp.admin`） 角色。参考"身份和访问管理角色和权限"。
- GCNV账户的API密钥文件。请参阅"创建服务帐户密钥"
- 储水池。参考"存储池概览"。

有关如何设置对Google Cloud NetApp Volumes 的访问权限的更多信息，请参阅："设置对Google Cloud NetApp Volumes 的访问权限" 。

Google Cloud NetApp Volumes后端配置选项和示例

了解Google Cloud NetApp Volumes的后端配置选项并查看配置示例。

后端配置选项

每个后端都在单个 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	描述	默认
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	价值 `storageDriverName` 必须指定为“google-cloud-netapp-volumes”。
`backendName`	（可选）存储后端自定义名称	驱动程序名称 + "_" + API 密钥的一部分
`storagePools`	用于指定卷创建存储池的可选参数。	
`projectNumber`	Google Cloud 帐户项目编号。该值可在 Google Cloud 门户网站首页找到。	
`location`	Trident创建 GCNV 卷的 Google Cloud 位置。创建跨区域 Kubernetes 集群时，在以下位置创建的卷：`location`可用于跨多个 Google Cloud 区域的节点上调度的工作负载。跨区域运输会产生额外费用。	

参数	描述	默认
apiKey	用于 Google Cloud 服务帐户的 API 密钥 netapp.admin`角色。它包含 Google Cloud 服务帐户私钥文件的 JSON 格式内容（原封不动地复制到后端配置文件中）。这 `apiKey`必须包含以下键的键值对： `type ， project_id ， client_email ， client_id ， auth_uri ， token_uri ， auth_provider_x509_cert_url ， 和 client_x509_cert_url。	
nfsMountOptions	对 NFS 挂载选项进行精细控制。	"nfsvers=3"
limitVolumeSize	如果请求的卷大小大于此值，则配置失败。	（默认情况下不强制执行）
serviceLevel	存储池的服务级别及其容量。这些值是 flex， standard ， premium ， 或者 extreme。	
labels	要应用于卷的任意 JSON 格式标签集	""
network	Google Cloud 网络用于 GCNV 卷。	
debugTraceFlags	故障排除时要使用的调试标志。例子， {"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。	无效的
nasType	配置 NFS 或 SMB 卷的创建。选项有 nfs， `smb`或空值。设置为 null 则默认使用 NFS 卷。	nfs
supportedTopologies	表示此后端支持的区域和区域列表。更多信息，请参阅"使用 CSI 拓扑"。例如： supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a	

卷配置选项

您可以控制默认卷配置 `defaults`配置文件部分。

参数	描述	默认
exportRule	新卷的出口规则。必须是以逗号分隔的 IPv4 地址列表，地址可以任意组合。	“0.0.0.0/0”
snapshotDir	访问 `.snapshot`目录	NFSv4 为“true”，NFSv3 为“false”。
snapshotReserve	快照预留的卷百分比	（接受默认值 0）
unixPermissions	新卷的 Unix 权限（4 位八进制数字）。	""

以下示例展示了基本配置，其中大多数参数都保留默认值。这是定义后端最简单的方法。

最小配置

这是最基本的后端配置。通过这种配置， Trident会发现您已委派给配置位置中Google Cloud NetApp Volumes的所有存储池，并随机将新卷放置在其中一个存储池上。因为 `nasType`省略了 `nfs`默认设置生效，后端将为 NFS 卷进行配置。

如果您刚开始使用Google Cloud NetApp Volumes并进行尝试，这种配置是理想的，但在实践中，您很可能需要为配置的卷提供额外的范围。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----END PRIVATE KEY-----\n

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

SMB卷配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

使用存储池过滤器进行配置

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

虚拟池配置

此后端配置在单个文件中定义了多个虚拟池。虚拟池在以下位置定义： `storage`部分。当您有多个支持不同服务级别的存储池，并且想要在 Kubernetes 中创建代表这些存储级别的存储类时，它们非常有用。虚拟池标签用于区分不同的池子。例如，在下面的例子中 `performance`标签和 `serviceLevel`类型用于区分虚拟池。

您还可以设置一些适用于所有虚拟池的默认值，并覆盖各个虚拟池的默认值。在下面的例子中，`snapshotReserve`和 `exportRule`作为所有虚拟池的默认值。

更多信息，请参阅"虚拟池"。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

GKE 的云身份

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

支持的拓扑配置

Trident可根据区域和可用区为工作负载提供卷。这 `supportedTopologies`此后端配置中的块用于提供每个后端的区域和区域列表。此处指定的区域和区域值必须与每个 Kubernetes 集群节点上的标签中的区域和区域值相匹配。这些区域和分区代表存储类中可以提供的允许值的列表。对于包含后端提供的区域和可用区子集的存储类， Trident会在所述区域和可用区中创建卷。更多信息，请参阅"使用 CSI 拓扑"。

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

下一步是什么?

创建后端配置文件后，运行以下命令：

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

要验证后端是否创建成功，请运行以下命令：

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

如果后端创建失败，则后端配置存在问题。您可以使用以下方式描述后端： `kubectl get tridentbackendconfig <backend-name>`运行以下命令查看日志以确定原因：

```
tridentctl logs
```

在您发现并纠正配置文件中的问题后，您可以删除后端并再次运行创建命令。

存储类定义

以下是一个基本内容 `StorageClass`上述后端所指的定义。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

使用以下示例定义 `**parameter.selector**`场地：

使用 `parameter.selector`你可以为每个对象指定。 `StorageClass`这"虚拟池"用于托管卷。该卷将具有所选池中定义的方面。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

有关存储类的更多详细信息，请参阅"创建存储类"。

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。任何 Active Directory 用户/密码，无论拥有任何权限或没有权限，都可以用作节点阶段密钥。

默认命名空间上的基本配置

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

每个命名空间使用不同的密钥

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

每个卷使用不同的密钥

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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`筛选支持 SMB 卷的存储池。 `nasType: nfs`或者 `nasType: null`NFS池过滤器。

PVC定义示例

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

要验证 PVC 是否已绑定，请运行以下命令：

```
kubectl get pvc gcnv-nfs-pvc

NAME           STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc   Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213   100Gi
RWX            gcnv-nfs-sc  1m
```

为 Google Cloud 后端配置Cloud Volumes Service

了解如何使用提供的示例配置，将NetApp Cloud Volumes Service for Google Cloud 配置为Trident安装的后端。

Google Cloud 驱动程序详情

Trident提供 `gcp-cvs`驱动程序与集群通信。支持的访问模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`gcp-cvs`	NFS	Filesystem	RWO、ROX、RWX、RWOP	`nfs`

了解Trident对 Google Cloud Cloud Volumes Service 的支持

Trident可以在以下两种格式之一中创建Cloud Volumes Service卷："服务类型"：

- **CVS-Performance**： Trident 的默认服务类型。这种性能优化型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，支持最小 100 GiB 大小的卷。您可以选择其中之一"三个服务级别"：
  - `standard`

- premium
- extreme

- **CVS**：CVS 服务类型提供较高的区域可用性，但性能水平有限或中等。 CVS 服务类型是一种软件选项，它使用存储池来支持小至 1 GiB 的卷。存储池最多可包含 50 个卷，所有卷共享池的容量和性能。您可以选择其中之一"两种服务级别"：
  - standardsw
  - zoneredundantstandardsw

你需要什么

配置和使用 "适用于 Google Cloud 的Cloud Volumes Service"后端需要以下组件：

- 已配置NetApp Cloud Volumes Service的 Google Cloud 帐户
- 您的 Google Cloud 帐户的项目编号
- 拥有 Google Cloud 服务帐户 `netappcloudvolumes.admin`角色
- 您的Cloud Volumes Service帐户的 API 密钥文件

后端配置选项

每个后端都在单个 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	描述	默认
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， Trident 会假定为 CVS-Performance 服务类型（`hardware）。	
storagePools	仅限CVS服务类型。用于指定卷创建存储池的可选参数。	
projectNumber	Google Cloud 帐户项目编号。该值可在 Google Cloud 门户网站首页找到。	
hostProjectNumber	如果使用共享 VPC 网络，则必须执行此操作。在这种情况下， `projectNumber`这是一个服务项目，而且 `hostProjectNumber`是宿主项目。	
apiRegion	Trident创建Cloud Volumes Service卷的 Google Cloud 区域。创建跨区域 Kubernetes 集群时，在以下位置创建的卷： `apiRegion`可用于跨多个 Google Cloud 区域的节点上调度的工作负载。跨区域运输会产生额外费用。	

参数	描述	默认
apiKey	用于 Google Cloud 服务帐户的 API 密钥 `netappcloudvolumes.admin`角色。它包含 Google Cloud 服务帐户私钥文件的 JSON 格式内容（原封不动地复制到后端配置文件中）。	
proxyURL	如果需要代理服务器才能连接到 CVS 帐户，请提供代理 URL。代理服务器可以是HTTP代理，也可以是HTTPS代理。对于 HTTPS 代理，会跳过证书验证，以允许在代理服务器中使用自签名证书。不支持启用身份验证的代理服务器。	
nfsMountOptions	对 NFS 挂载选项进行精细控制。	"nfsvers=3"
limitVolumeSize	如果请求的卷大小大于此值，则配置失败。	（默认情况下不强制执行）
serviceLevel	CVS-Performance 或 CVS 服务级别（适用于新卷）。CVS-Performance 值是 standard， premium ， 或者 extreme。 CVS 值是 standardsw`或者 `zoneredundantstandardsw。	CVS-Performance 默认值为“标准”。 CVS 默认值为“standardsw”。
network	Google Cloud 网络用于Cloud Volumes Service卷。	“默认”
debugTraceFlags	故障排除时要使用的调试标志。例子，\{"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。	无效的
allowedTopologies	要启用跨区域访问，您的 StorageClass 定义如下：allowedTopologies`必须包含所有地区。例如：`- key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

卷配置选项

您可以控制默认卷配置 `defaults`配置文件部分。

参数	描述	默认
exportRule	新卷的出口规则。必须是以逗号分隔的 IPv4 地址或 IPv4 子网的列表，采用 CIDR 表示法。	“0.0.0.0/0”
snapshotDir	访问 `.snapshot`目录	"false"
snapshotReserve	快照预留的卷百分比	（接受 CVS 默认值 0）
size	新卷的规模。 CVS-Performance 最低要求为 100 GiB。 CVS 最小容量为 1 GiB。	CVS-Performance 服务类型默认为“100GiB”。 CVS 服务类型不设置默认值，但要求至少 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：最小配置

这是使用默认 CVS-Performance 服务类型和默认“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-----
    <key_value>
    -----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

此示例展示了后端配置选项，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-----
    <key_value>
    -----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`指的是他们。请参阅[存储类定义]查看存储类的定义方式。

这里为所有虚拟池设置了特定的默认值，这些默认值决定了： `snapshotReserve` 5%和 `exportRule` 至 `0.0.0.0/0`。虚拟池在以下位置定义： `storage`部分。每个虚拟池都定义了自己的规则。 `serviceLevel`并且有些池会覆盖默认值。虚拟池标签用于根据以下因素区分池子： `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-----
    <key_value>
    -----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
```

存储类定义

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。该卷将具有所选池中定义的方面。

存储类示例

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

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

- 第一个存储类(`cvs-extreme-extra-protection`) 映射到第一个虚拟池。这是唯一一个提供极致性能且快照储备为 10% 的存储池。
- 最后一个存储类(`cvs-extra-protection`) 调用任何提供 10% 快照保留的存储池。 Trident决定选择哪个虚拟池，并确保满足快照储备要求。

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**：最小配置

这是使用最简后端配置 `storageClass`指定 CVS 服务类型和默认值 `standardsw`服务水平。

```
---
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-----
    <key_value>
    -----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-----
    <key_value>
    -----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，则后端配置存在问题。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs
```

在您发现并纠正配置文件中的问题后，您可以再次运行创建命令。

配置NetApp HCI或SolidFire后端

了解如何在Trident安装中创建和使用 Element 后端。

元素驱动程序详情

Trident提供 `solidfire-san`用于与集群通信的存储驱动程序。支持的访问模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

这 `solidfire-san`存储驱动程序支持_文件_和_块_卷模式。对于 `Filesystem`volumeMode， Trident创建一个卷并创建一个文件系统。文件系统类型由 StorageClass 指定。

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`solidfire-san`	iSCSI	块	RWO、ROX、RWX、RWOP	没有文件系统。原始块设备。
`solidfire-san`	iSCSI	Filesystem	RWO，RWOP	`xfs`， `ext3`，`ext4`

开始之前

创建 Element 后端之前，您需要以下内容。

- 一个支持运行 Element 软件的存储系统。
- 拥有NetApp HCI/ SolidFire集群管理员或租户用户权限，可以管理卷。
- 所有 Kubernetes 工作节点都应该安装相应的 iSCSI 工具。参考"工作节点准备信息"。

后端配置选项

请参阅下表了解后端配置选项：

参数	描述	默认
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	始终是“solidfire-san”
`backendName`	自定义名称或存储后端	"solidfire_" + 存储 (iSCSI) IP 地址
`Endpoint`	针对SolidFire集群的 MVIP，包含租户凭证	
`SVIP`	存储（iSCSI）IP 地址和端口	

参数	描述	默认
`labels`	要应用于卷的任意 JSON 格式标签集。	""
`TenantName`	要使用的租户名称（如果找不到则创建）	
`InitiatorIFace`	将 iSCSI 流量限制到特定主机接口	“默认”
`UseCHAP`	使用 CHAP 对 iSCSI 进行身份验证。 Trident使用 CHAP。	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。	（默认情况下不强制执行）
`debugTraceFlags`	故障排除时要使用的调试标志。例如，{"api":false, "method":true}	无效的

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

示例 **1**：后端配置 **`solidfire-san`**具有三种音量类型的驱动器

本示例展示了一个使用 CHAP 认证的后端文件，并对三种具有特定 QoS 保证的卷类型进行了建模。最有可能的情况是，您会定义存储类来使用它们中的每一个。 `IOPS`存储类参数。

```
---
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：后端和存储类配置 **`solidfire-san`**带虚拟池的驱动程序

此示例显示了配置了虚拟池的后端定义文件以及引用这些虚拟池的存储类。

Trident在配置时将存储池中存在的标签复制到后端存储 LUN。为了方便起见，存储管理员可以为每个虚拟池定义标签，并按标签对卷进行分组。

在下方所示的示例后端定义文件中，所有存储池都设置了特定的默认值，这些默认值决定了： `type`银级。虚拟池在以下位置定义： `storage`部分。在这个例子中，一些存储池设置了自己的类型，而一些存储池则覆盖了上面设置的默认值。

```
---
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 定义与上述虚拟池相关。使用 `parameters.selector`字段中，每个 StorageClass 都会指定哪些虚拟池可用于托管卷。卷将具有所选虚拟池中定义的各个方面。

第一个存储类(`solidfire-gold-four`将映射到第一个虚拟池。这是唯一一家提供黄金级性能的泳池。

`Volume Type QoS`黄金。最后一个存储类(`solidfire-silver`）指出任何提供银级性能的存储池。Trident将决定选择哪个虚拟池，并确保满足存储需求。

```
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
```

查找更多信息

- "卷访问组"

ONTAP SAN 驱动程序

ONTAP SAN 驱动程序概述

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

ONTAP SAN 驱动程序详情

Trident提供以下 SAN 存储驱动程序，用于与ONTAP集群通信。支持的访问模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`ontap-san`	iSCSI 通过光纤通道提供 SCSI 服务	块	RWO、ROX、RWX、RWOP	无文件系统；原始块设备
`ontap-san`	iSCSI 通过光纤通道提供 SCSI 服务	Filesystem	RWO，RWOP ROX 和 RWX 在文件系统卷模式下不可用。	`xfs`， `ext3` ， `ext4`
`ontap-san`	NVMe/TCP 参考NVMe/TCP 的其他注意事项。	块	RWO、ROX、RWX、RWOP	无文件系统；原始块设备
`ontap-san`	NVMe/TCP 参考NVMe/TCP 的其他注意事项。	Filesystem	RWO，RWOP ROX 和 RWX 在文件系统卷模式下不可用。	`xfs`， `ext3` ， `ext4`

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`ontap-san-economy`	iSCSI	块	RWO、ROX、RWX、RWOP	无文件系统；原始块设备
`ontap-san-economy`	iSCSI	Filesystem	RWO，RWOP ROX 和 RWX 在文件系统卷模式下不可用。	`xfs`， `ext3` ， `ext4`

- 使用 `ontap-san-economy`仅当预计持续卷使用量高于……时"支持的ONTAP容量限制"。
- 使用 `ontap-nas-economy`仅当预计持续卷使用量高于……时"支持的ONTAP容量限制"以及 `ontap-san-economy`驱动程序无法使用。
- 请勿使用 `ontap-nas-economy`如果您预计需要数据保护、灾难恢复或移动办公。
- NetApp不建议在所有ONTAP驱动程序中使用 Flexvol 自动增长，ontap-san 除外。作为一种变通方法， Trident支持使用快照储备，并相应地调整 Flexvol 容量。

用户权限

Trident预期以ONTAP或 SVM 管理员身份运行，通常使用以下方式： `admin`集群用户或 `vsadmin`SVM 用户，或者具有相同角色但名称不同的用户。对于Amazon FSx for NetApp ONTAP部署， Trident需要以ONTAP或 SVM 管理员身份运行，并使用集群。 `fsxadmin`用户或 `vsadmin`SVM 用户，或者具有相同角色但名称不同的用户。这 `fsxadmin`用户是集群管理员用户的有限替代品。

如果你使用 `limitAggregateUsage`需要参数和集群管理员权限。当使用Amazon FSx for NetApp ONTAP和Trident时， `limitAggregateUsage`参数将无法与 `vsadmin`和 `fsxadmin`用户账户。如果指定此参数，配置操作将失败。

虽然可以在ONTAP中创建一个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。 Trident的大多数新版本都会调用额外的 API，这些 API 必须加以考虑，这使得升级变得困难且容易出错。

NVMe/TCP 的其他注意事项

Trident支持使用非易失性存储器高速接口 (NVMe) 协议 `ontap-san`驱动程序包括：

- IPv6
- NVMe卷的快照和克隆
- 调整 NVMe 卷的大小
- 导入在Trident外部创建的 NVMe 卷，以便Trident可以管理其生命周期。
- NVMe原生多路径
- K8s节点的优雅关闭或非优雅关闭（24.06）

Trident不支持：

- NVMe 原生支持的 DH-HMAC-CHAP
- 设备映射器 (DM) 多路径

- LUKS 加密

NVMe 仅支持ONTAP REST API，不支持 ONTAPI (ZAPI)。

准备配置后端**ONTAP SAN** 驱动程序

了解配置ONTAP后端和ONTAP SAN 驱动程序的要求和身份验证选项。

要求

对于所有ONTAP后端， Trident要求至少将一个聚合分配给 SVM。

"ASA r2 系统"与其他ONTAP系统（ASA、 AFF和FAS）在存储层的实现上有所不同。在ASA r2 系统中，使用存储可用区而不是聚合。请参阅"这"知识库文章，介绍如何在ASA r2 系统中将聚合分配给 SVM。

请记住，您还可以运行多个驱动程序，并创建指向其中一个或另一个驱动程序的存储类。例如，您可以配置一个 `san-dev`使用类 `ontap-san`司机和 `san-default`使用类 `ontap-san-economy`一。

所有 Kubernetes 工作节点都必须安装相应的 iSCSI 工具。参考 "准备工作节点" 了解详情。

对**ONTAP**后端进行身份验证

Trident提供两种ONTAP后端身份验证方式。

- 基于凭证：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`或者 `vsadmin`确保与ONTAP版本最大程度兼容。
- 基于证书： Trident还可以使用安装在后端的证书与ONTAP集群通信。此处，后端定义必须包含客户端证书、密钥和受信任 CA 证书（如果使用，建议使用）的 Base64 编码值。

您可以更新现有后端，以在基于凭据的方法和基于证书的方法之间进行切换。但是，一次只能支持一种身份验证方法。要切换到不同的身份验证方法，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，并出现错误，提示配置文件中提供了多个身份验证方法。

启用基于凭据的身份验证

Trident需要 SVM 范围/集群范围管理员的凭据才能与ONTAP后端通信。建议使用标准的、预定义的角色，例如： `admin`或者 `vsadmin`。这样可以确保与未来ONTAP版本向前兼容，这些版本可能会公开一些功能 API，供未来的Trident版本使用。虽然可以创建自定义安全登录角色并将其与Trident一起使用，但不建议这样做。

后端定义示例如下所示：

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

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

请注意，后端定义是唯一以纯文本形式存储凭据的地方。后端创建完成后，用户名/密码将使用 Base64 进行编码，并存储为 Kubernetes 密钥。只有创建或更新后端时才需要了解凭据。因此，这是一项仅限管理员执行的操作，由 Kubernetes/存储管理员执行。

启用基于证书的身份验证

新的和现有的后端都可以使用证书与ONTAP后端通信。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。
- trustedCACertificate：受信任 CA 证书的 Base64 编码值。如果使用受信任的 CA，则必须提供此参数。如果没有使用受信任的证书颁发机构，则可以忽略此步骤。

典型的工作流程包括以下步骤。

步骤

1. 生成客户端证书和密钥。生成时，将通用名称 (CN) 设置为要进行身份验证的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. 向ONTAP集群添加受信任的 CA 证书。这可能已经由存储管理员处理了。如果没有使用受信任的证书颁发机构，则忽略此操作。

```
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（来自步骤 1）。

```
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. 使用生成的证书进行身份验证。请将 < ONTAP管理 LIF> 和 <vserver 名称> 替换为管理 LIF IP 地址和 SVM 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、密钥和受信任的 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，以使用不同的身份验证方法或轮换其凭据。这种方法是双向的：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，您必须删除现有的身份验证方法并添加新的身份验证方法。然后使用包含所需参数的更新后的 backend.json 文件来执行 `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，存储管理员必须首先更新ONTAP上用户的密码。接下来将进行后端更新。轮换证书时，可以为用户添加多个证书。然后更新后端以使用新证书，之后即可从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，也不会影响之后建立的卷连接。后端更新成功表明Trident可以与ONTAP后端通信并处理未来的卷操作。

为**Trident**创建自定义**ONTAP**角色

您可以创建一个具有最低权限的ONTAP集群角色，这样您就不必使用ONTAP管理员角色在Trident中执行操作。在Trident后端配置中包含用户名时， Trident将使用您创建的ONTAP集群角色来执行操作。

请参阅"Trident自定义角色生成器"有关创建Trident自定义角色的更多信息。

使用**ONTAP CLI**

1. 使用以下命令创建新角色：

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. 为Trident用户创建用户名：

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 将角色映射到用户：

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

使用系统管理器

在ONTAP系统管理器中执行以下步骤：

1. 创建自定义角色：
   a. 要在集群级别创建自定义角色，请选择“集群 > 设置”。

      （或者）要在 SVM 级别创建自定义角色，请选择“存储”>“存储虚拟机”> `required SVM` > 设置 > 用户和角色*。

   b. 选择“用户和角色”旁边的箭头图标（→）。
   c. 在“角色”下选择“+添加”。
   d. 定义角色规则，然后点击“保存”。
2. 将角色映射到Trident用户： + 在“用户和角色”页面上执行以下步骤：
   a. 在“用户”下方选择“添加”图标 **+**。
   b. 选择所需的用户名，然后在“角色”下拉菜单中选择角色。
   c. 单击“保存”。

更多信息请参阅以下页面：

- "用于管理ONTAP的自定义角色"或者"定义自定义角色"
- "与角色和用户协作"

使用双向 **CHAP** 验证连接

Trident可以使用双向 CHAP 对 iSCSI 会话进行身份验证。 `ontap-san`和 `ontap-san-economy`司机。这需要启用 `useCHAP`在后端定义中添加选项。设置为 `true`Trident将 SVM 的默认发起程序安全配置为双向 CHAP，并从后端文件中设置用户名和密钥。 NetApp建议使用双向 CHAP 协议对连接进行身份验证。请参见以下示例配置：

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

这 `useCHAP`该参数是一个布尔选项，只能配置一次。默认值为 false。一旦将其设置为 true，就无法再将其设置为 false。

此外 `useCHAP=true`， 这 `chapInitiatorSecret`， `chapTargetInitiatorSecret` ， `chapTargetUsername` ， 和 `chapUsername`字段必须包含在后端定义中。创建后端后，可以通过运行以下命令来更改密钥： `tridentctl update`。

工作原理

通过设置 `useCHAP`如果设置为 true，则存储管理员指示Trident在存储后端配置 CHAP。其中包括以下内容：

- 在SVM上设置CHAP：
  - 如果 SVM 的默认启动器安全类型为“无”（默认设置）*并且*卷中不存在任何预先存在的 LUN，Trident会将默认安全类型设置为“无”。 `CHAP`然后继续配置 CHAP 发起程序和目标用户名及密钥。
  - 如果 SVM 包含 LUN， Trident将不会在 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 发起程序和目标用户名和密钥；这些选项必须在后端配置中指定（如上所示）。

后端创建完成后， Trident会创建一个相应的实例。 `tridentbackend` CRD 并将 CHAP 密钥和用户名存储为 Kubernetes 密钥。 Trident在此后端创建的所有 PV 都将通过 CHAP 进行安装和连接。

轮换凭证并更新后端

您可以通过更新 CHAP 参数来更新 CHAP 凭据。 `backend.json`文件。这将需要更新 CHAP 密钥并使用 `tridentctl update`命令反映这些更改。

更新后端 CHAP 密钥时，必须使用 `tridentctl`更新后端。请勿使用ONTAP CLI 或ONTAP系统管理器更新存储集群上的凭据，因为Trident将无法检测到这些更改。

```
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",
    "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Trident在 SVM 上更新凭据，则这些连接将继续保持活动状态。新连接使用更新后的凭据，现有连接将继续保持活动状态。断开并重新连接旧的PV将使它们使用更新后的凭据。

ONTAP SAN 配置选项和示例

了解如何在Trident安装中创建和使用ONTAP SAN 驱动程序。本节提供后端配置示例以及将后端映射到 StorageClasses 的详细信息。

"ASA r2 系统"与其他ONTAP系统（ASA、 AFF和FAS）在存储层的实现上有所不同。这些变化会影响某些参数的使用，如注释中所述。"了解更多关于ASA r2 系统与其他ONTAP系统之间的区别"。

只有 `ontap-san`ASA r2 系统支持驱动程序（支持 iSCSI 和 NVMe/TCP 协议）。

在Trident后端配置中，无需指定您的系统是ASA r2。当您选择 `ontap-san`作为 `storageDriverName`Trident可自动检测ASA r2 或传统的ONTAP系统。如下表所示，某些后端配置参数不适用于ASA r2 系统。

请参阅下表了解后端配置选项：

参数	描述	默认
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	`ontap-san`或者 `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。 可以指定一个完全限定域名（FQDN）。 如果Trident安装时使用了 IPv6 标志，则可以设置为使用 IPv6 地址。 IPv6 地址必须用方括号定义，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 为了实现MetroCluster 的无缝切换，请参阅MetroCluster示例。 ⓘ 如果您使用的是“vsadmin”凭据，`managementLIF`必须是SVM的凭据；如果使用“admin”凭据，`managementLIF`必须是集群的那个。	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	LIF协议的IP地址。如果Trident安装时使用了 IPv6 标志，则可以设置为使用 IPv6 地址。 IPv6 地址必须用方括号定义，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。 *请勿指定使用 iSCSI。*Trident的使用"ONTAP选择性 LUN 地图"发现建立多路径会话所需的 iSCSI LIF。如果出现以下情况，则会生成警告： `dataLIF`已明确定义。 *Metrocluster 除外。*查看MetroCluster示例。	由支持向量机导出
`svm`	要使用的存储虚拟机 *Metrocluster 除外。*查看MetroCluster示例。	如果是 SVM 则推导而来`managementLIF`已指定
`useCHAP`	使用 CHAP 对ONTAP SAN 驱动程序的 iSCSI 进行身份验证 [布尔值]。设置为 `true`让Trident配置并使用双向 CHAP 作为后端给定 SVM 的默认身份验证。参考 "准备配置后端ONTAP SAN 驱动程序" 了解详情。 不支持**FCP**或**NVMe/TCP**协议。	`false`
`chapInitiatorSecret`	CHAP 发起者密钥。如果是必填项 `useCHAP=true`	""
`labels`	要应用于卷的任意 JSON 格式标签集	""

参数	描述	默认
`chapTargetInitiatorSecret`	CHAP 目标发起者密钥。如果是必填项 `useCHAP=true`	""
`chapUsername`	入站用户名。如果是必填项 `useCHAP=true`	""
`chapTargetUsername`	目标用户名。如果是必填项 `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`	客户端私钥的 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。用于基于凭证的身份验证。有关 Active Directory 身份验证，请参阅 "使用 Active Directory 凭据向后端 SVM 验证Trident 的身份"。	""
`password`	与ONTAP集群通信所需的密码。用于基于凭证的身份验证。有关 Active Directory 身份验证，请参阅 "使用 Active Directory 凭据向后端 SVM 验证Trident 的身份"。	""
`svm`	使用的存储虚拟机	如果是 SVM 则推导而来 `managementLIF`已指定
`storagePrefix`	在 SVM 中配置新卷时使用的前缀。之后无法修改。要更新此参数，您需要创建一个新的后端。	`trident`
`aggregate`	用于配置的聚合（可选；如果设置，则必须分配给 SVM）。对于 `ontap-nas-flexgroup`驱动程序，此选项将被忽略。如果未分配，则可以使用任何可用的聚合来配置FlexGroup卷。 当 SVM 中的聚合数据更新时， Trident 会自动轮询 SVM 进行更新，而无需重启Trident控制器。当您在Trident中配置特定聚合以配置卷时，如果该聚合被重命名或移出 SVM，则在轮询 SVM 聚合时， Trident中的后端将变为失败状态。您必须将聚合更改为 SVM 上存在的聚合，或者将其完全删除，才能使后端恢复联机。 请勿指定用于**ASA r2** 系统。	""

参数	描述	默认
limitAggregateUsage	如果使用率超过此百分比，则配置失败。如果您使用的是Amazon FSx for NetApp ONTAP后端，请勿指定limitAggregateUsage。提供的 `fsxadmin`和 `vsadmin`不包含检索汇总使用情况和使用Trident限制它所需的权限。 请勿指定用于**ASA r2** 系统。	（默认情况下不强制执行）
limitVolumeSize	如果请求的卷大小大于此值，则配置失败。同时限制其管理的 LUN 卷的最大大小。	（默认情况下不强制执行）
lunsPerFlexvol	每个 Flexvol 的最大 LUN 数量必须在 [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} 除非您正在进行故障排除并且需要详细的日志转储，否则请勿使用此方法。	null
useREST	使用ONTAP REST API 的布尔参数。 `useREST`设置为 `true`Trident使用ONTAP REST API与后端通信；当设置为`false`Trident使用 ONTAPI (ZAPI)调用与后端通信。此功能需要ONTAP 9.11.1及更高版本。此外，所使用的ONTAP登录角色必须具有访问权限。`ontapi`应用。预定义项满足了这一点。`vsadmin`和 `cluster-admin`角色。从Trident 24.06 版本和ONTAP 9.15.1 或更高版本开始，`useREST`设置为 `true`默认；更改`useREST`到 `false`使用 ONTAPI (ZAPI) 调用。 `useREST`完全符合 NVMe/TCP 标准。 NVMe 仅支持ONTAP REST API，不支持 ONTAPI (ZAPI)。 如果指定，则始终设置为 **`true`**适用于**ASA r2** 系统。	`true`适用于ONTAP 9.15.1 或更高版本，否则 `false`。
sanType	用于选择 `iscsi`对于 iSCSI， `nvme`适用于 NVMe/TCP 或 `fcp`用于光纤通道 (FC) 上的 SCSI。	`iscsi`如果为空

参数	描述	默认
formatOptions	使用 `formatOptions`为以下情况指定命令行参数 `mkfs`该命令将在每次格式化卷时应用。这样您就可以根据自己的喜好格式化音量。请确保指定与 mkfs 命令选项类似的 formatOptions，但不包括设备路径。例如：“-E nodiscard” 支持 `**ontap-san**`和 `**ontap-san-economy**`支持 **iSCSI** 协议的驱动程序。 此外，在使用 **iSCSI** 和 **NVMe/TCP** 协议时， **ASA r2** 系统也受支持。	
limitVolumePoolSize	在 ontap-san-economy 后端中使用 LUN 时可请求的最大FlexVol大小。	（默认情况下不强制执行）
denyNewVolumePools	限制 `ontap-san-economy`后端创建新的FlexVol卷来包含它们的 LUN。只有预先存在的 Flexvol 才能用于配置新的 PV。	

使用 **formatOptions** 的建议

Trident建议采用以下选项来加快格式化过程：

-E nodiscard:

- 保留，不要在执行 mkfs 时尝试丢弃块（最初丢弃块对固态设备和稀疏/精简配置存储很有用）。这取代了已弃用的选项“-K”，并且适用于所有文件系统（xfs、ext3 和 ext4）。

使用 **Active Directory** 凭据向后端 **SVM** 验证**Trident** 的身份

您可以配置Trident以使用 Active Directory (AD) 凭据对后端 SVM 进行身份验证。在 AD 帐户可以访问 SVM 之前，您必须配置 AD 域控制器对集群或 SVM 的访问权限。对于使用 AD 帐户进行集群管理，您必须创建域隧道。参考 "在ONTAP中配置 Active Directory 域控制器访问" 了解详情。

步骤

1. 为后端 SVM 配置域名系统 (DNS) 设置：

   ```
   vserver services dns create -vserver <svm_name> -dns-servers
   <dns_server_ip1>,<dns_server_ip2>
   ```

2. 运行以下命令在 Active Directory 中为 SVM 创建计算机帐户：

   ```
   vserver active-directory create -vserver DataSVM -account-name ADSERVER1
   -domain demo.netapp.com
   ```

3. 使用此命令创建 AD 用户或组来管理集群或 SVM

   ```
   security login create -vserver <svm_name> -user-or-group-name
   <ad_user_or_group> -application <application> -authentication-method domain
   -role vsadmin
   ```

4. 在Trident后端配置文件中，设置 `username` 和 `password` 参数分别为 AD 用户或组名称和密码。

您可以使用以下选项控制默认配置。 `defaults`配置部分。例如，请参见下面的配置示例。

参数	描述	默认
`spaceAllocation`	LUN 的空间分配	“true” 如果指定，则设置为 **`true`**适用于**ASA r2** 系统。
`spaceReserve`	空间预留模式；“无”（细）或“大量”（粗）。 设置为 **`none`**适用于**ASA r2** 系统。	“没有任何”
`snapshotPolicy`	要使用的快照策略。 设置为 **`none`**适用于**ASA r2** 系统。	“没有任何”
`qosPolicy`	要为创建的卷分配的 QoS 策略组。每个存储池/后端选择 qosPolicy 或 adaptiveQosPolicy 之一。将 QoS 策略组与Trident结合使用需要ONTAP 9.8 或更高版本。您应该使用非共享的 QoS 策略组，并确保该策略组单独应用于每个成员。共享的 QoS 策略组强制规定所有工作负载的总吞吐量上限。	""
`adaptiveQosPolicy`	要为创建的卷分配的自适应 QoS 策略组。每个存储池/后端选择 qosPolicy 或 adaptiveQosPolicy 之一	""
`snapshotReserve`	为快照预留的卷百分比。 请勿指定用于**ASA r2** 系统。	如果为“0”， `snapshotPolicy`为“无”，否则为“
`splitOnClone`	创建时将克隆体从其母体中分离出来	"false"
`encryption`	在新卷上启用NetApp卷加密 (NVE)；默认设置为 `false`。要使用此选项，必须在集群上获得 NVE 许可并启用 NVE。如果后端启用了 NAE，则在Trident中配置的任何卷都将启用 NAE。更多信息，请参阅：“Trident如何与 NVE 和 NAE 协同工作” 。	“false” 如果指定，则设置为 **`true`**适用于**ASA r2** 系统。
`luksEncryption`	启用LUKS加密。参考“使用 Linux 统一密钥设置 (LUKS)”。	设置为 **`false`**适用于**ASA r2** 系统。
`tieringPolicy`	分层策略使用“无” 请勿为**ASA r2** 系统指定 。	
`nameTemplate`	用于创建自定义卷名称的模板。	""

卷配置示例

以下是一个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
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

对于使用以下方式创建的所有卷 `ontap-san`驱动程序Trident为FlexVol增加了 10% 的额外容量，以容纳 LUN 元数据。 LUN 将按照用户在 PVC 中请求的确切大小进行配置。 Trident使FlexVol增加 10%（在ONTAP中显示为可用尺寸）。用户现在将获得他们所申请的可用容量。此项更改还可以防止 LUN 在可用空间未完全利用之前变为只读。这不适用于 ontap-san-economy。

对于定义后端 `snapshotReserve`Trident计算体积大小的方法如下：

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

1.1 是Trident为容纳 LUN 元数据而额外添加到FlexVol 的10%。为了 `snapshotReserve`= 5%，PVC 请求 = 5 GiB，总体积大小为 5.79 GiB，可用大小为 5.5 GiB。这 `volume show`该命令应显示与此示例类似的结果：

目前，调整大小是将新计算方法应用于现有体积的唯一途径。

最小配置示例

以下示例展示了基本配置，其中大多数参数都保留默认值。这是定义后端最简单的方法。

> 如果您在NetApp ONTAP上使用Amazon FSx和Trident， NetApp建议您为 LIF 指定 DNS 名称而不是 IP 地址。

ONTAP SAN 示例

这是使用以下方法的基本配置： `ontap-san`司机。

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

MetroCluster示例

您可以配置后端，以避免在切换和切换回后端后手动更新后端定义。"SVM复制和恢复" 。

为了实现无缝切换和切换回，请指定 SVM `managementLIF`并省略 `svm`参数。例如：

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

ONTAP SAN 经济示例

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

基于证书的身份验证示例

在这个基本配置示例中 `clientCertificate`， `clientPrivateKey`， 和 `trustedCACertificate`（如果使用受信任的 CA，则为可选） `backend.json`分别取客户端证书、私钥和受信任 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
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

双向 **CHAP** 示例

这些示例创建了一个后端。 `useCHAP`设置为 `true`。

ONTAP SAN CHAP 示例

```
---
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
username: vsadmin
password: <password>
```

ONTAP SAN 经济 **CHAP** 示例

```
---
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
username: vsadmin
password: <password>
```

NVMe/TCP 示例

您的ONTAP后端必须配置有使用 NVMe 的 SVM。这是 NVMe/TCP 的基本后端配置。

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

SCSI over FC (FCP) 示例

您的ONTAP后端必须配置有 FC 的 SVM。这是 FC 的基本后端配置。

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

使用 **nameTemplate** 的后端配置示例

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

ontap-san-economy 驱动程序的 **formatOptions** 示例

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

具有虚拟池的后端示例

在这些示例后端定义文件中，所有存储池都设置了特定的默认值，例如： `spaceReserve`没有，`spaceAllocation`为假，并且 `encryption`为假。虚拟池在存储部分中定义。

Trident在“备注”字段中设置配置标签。在FlexVol volume上设置注释。Trident 在配置时Trident虚拟池上存在的所有标签复制到存储卷。为了方便起见，存储管理员可以为每个虚拟池定义标签，并按标签对卷进行分组。

在这些示例中，一些存储池会设置自己的参数。 `spaceReserve` ， `spaceAllocation` ， 和 `encryption`有

些值会覆盖默认值，有些池会覆盖默认值。

ONTAP SAN 示例

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
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"
```

ONTAP SAN 经济示例

```
---
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
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"
  - labels:
      department: legal
      creditpoints: "5000"
```

```
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

NVMe/TCP 示例

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

将后端映射到存储类

以下 StorageClass 定义指的是[具有虚拟池的后端示例]。使用 `parameters.selector`字段中，每个 StorageClass 都会指出哪些虚拟池可用于托管卷。卷将具有所选虚拟池中定义的各个方面。

- 这 `protection-gold`StorageClass 将映射到第一个虚拟池。 `ontap-san`后端。这是唯一提供黄金级保护的泳池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 这 `protection-not-gold`StorageClass 将映射到第二个和第三个虚拟池。 `ontap-san`后端。除了黄金级别之外，只有这些金池提供其他级别的保护。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 这 `app-mysqldb`StorageClass 将映射到第三个虚拟池 `ontap-san-economy`后端。这是唯一一个为mysqldb类型应用程序提供存储池配置的存储池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 这 `protection-silver-creditpoints-20k`StorageClass 将映射到第二个虚拟池 `ontap-san`后端。这是唯一提供银级保护和 20000 积分的彩池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 这 `creditpoints-5k`StorageClass 将映射到第三个虚拟池 `ontap-san`后端和第四个虚拟池 `ontap-san-economy`后端。这是唯一提供 5000 积分的彩池产品。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- 这 `my-test-app-sc`StorageClass 将映射到 `testAPP`虚拟池 `ontap-san`司机 `sanType: nvme。这是唯一一家提供泳池的公司 testApp。

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident将决定选择哪个虚拟池，并确保满足存储需求。

ONTAP NAS 驱动程序

ONTAP NAS 驱动程序概述

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

ONTAP NAS 驱动程序详情

Trident提供以下 NAS 存储驱动程序，用于与ONTAP集群通信。支持的访问模式有：*ReadWriteOnce* (RWO)、*ReadOnlyMany* (ROX)、*ReadWriteMany* (RWX)、*ReadWriteOncePod* (RWOP)。

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`ontap-nas`	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	", `nfs`, `smb`

驱动程序	协议	音量模式	支持的访问模式	支持的文件系统
`ontap-nas-flexgroup`	NFS SMB	Filesystem	RWO、ROX、RWX、RWOP	", `nfs`, `smb`

- 使用 `ontap-san-economy`仅当预计持续卷使用量高于……时"支持的ONTAP容量限制"。
- 使用 `ontap-nas-economy`仅当预计持续卷使用量高于……时"支持的ONTAP容量限制"以及 `ontap-san-economy`驱动程序无法使用。
- 请勿使用 `ontap-nas-economy`如果您预计需要数据保护、灾难恢复或移动办公。
- NetApp不建议在所有ONTAP驱动程序中使用 Flexvol 自动增长，ontap-san 除外。作为一种变通方法， Trident支持使用快照储备，并相应地调整 Flexvol 容量。

用户权限

Trident预期以ONTAP或 SVM 管理员身份运行，通常使用以下方式： `admin`集群用户或 `vsadmin`SVM 用户，或者具有相同角色但名称不同的用户。

对于Amazon FSx for NetApp ONTAP部署， Trident需要以ONTAP或 SVM 管理员身份运行，并使用集群。`fsxadmin`用户或 `vsadmin`SVM 用户，或者具有相同角色但名称不同的用户。这 `fsxadmin`用户是集群管理员用户的有限替代品。

如果你使用 `limitAggregateUsage`需要参数和集群管理员权限。当使用Amazon FSx for NetApp ONTAP和Trident时， `limitAggregateUsage`参数将无法与 `vsadmin`和 `fsxadmin`用户账户。如果指定此参数，配置操作将失败。

虽然可以在ONTAP中创建一个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。 Trident的大多数新版本都会调用额外的 API，这些 API 必须加以考虑，这使得升级变得困难且容易出错。

准备配置带有**ONTAP NAS** 驱动程序的后端

了解配置带有ONTAP NAS 驱动程序的ONTAP后端的要求、身份验证选项和导出策略。

要求

- 对于所有ONTAP后端， Trident要求至少将一个聚合分配给 SVM。
- 您可以运行多个驱动程序，并创建指向其中一个或另一个驱动程序的存储类。例如，您可以配置一个使用以下方式的 Gold 类： `ontap-nas`驾驶员和使用青铜级的驾驶员 `ontap-nas-economy`一。
- 所有 Kubernetes 工作节点都必须安装相应的 NFS 工具。请参阅"此处"更多详情请见下文。
- Trident仅支持挂载到运行在 Windows 节点上的 pod 的 SMB 卷。参考 准备配置SMB卷 了解详情。

对**ONTAP**后端进行身份验证

Trident提供两种ONTAP后端身份验证方式。

- 基于凭证：此模式需要对ONTAP后端拥有足够的权限。建议使用与预定义的安全登录角色关联的帐户，例如： `admin`或者 `vsadmin`确保与ONTAP版本最大程度兼容。
- 基于证书：此模式要求后端安装证书， Trident才能与ONTAP集群通信。此处，后端定义必须包含客户端证

书、密钥和受信任 CA 证书（如果使用，建议使用）的 Base64 编码值。

您可以更新现有后端，以在基于凭据的方法和基于证书的方法之间进行切换。但是，一次只能支持一种身份验证方法。要切换到不同的身份验证方法，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，并出现错误，提示配置文件中提供了多个身份验证方法。

启用基于凭据的身份验证

Trident需要 SVM 范围/集群范围管理员的凭据才能与ONTAP后端通信。建议使用标准的、预定义的角色，例如： `admin`或者 `vsadmin`。这样可以确保与未来ONTAP版本向前兼容，这些版本可能会公开一些功能 API，供未来的Trident版本使用。虽然可以创建自定义安全登录角色并将其与Trident一起使用，但不建议这样做。

后端定义示例如下所示：

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

请注意，后端定义是唯一以纯文本形式存储凭据的地方。后端创建完成后，用户名/密码将使用 Base64 进行编码，并存储为 Kubernetes 密钥。后端创建/更新是唯一需要了解凭据的步骤。因此，这是一项仅限管理员执行的操作，由 Kubernetes/存储管理员执行。

启用基于证书的身份验证

新的和现有的后端都可以使用证书与ONTAP后端通信。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。
- trustedCACertificate：受信任 CA 证书的 Base64 编码值。如果使用受信任的 CA，则必须提供此参数。如果没有使用受信任的证书颁发机构，则可以忽略此步骤。

典型的工作流程包括以下步骤。

步骤

1. 生成客户端证书和密钥。生成时，将通用名称 (CN) 设置为要进行身份验证的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. 向ONTAP集群添加受信任的 CA 证书。这可能已经由存储管理员处理了。如果没有使用受信任的证书颁发机构，则忽略此操作。

```
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（来自步骤 1）。

```
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. 使用生成的证书进行身份验证。请将 < ONTAP管理 LIF> 和 <vserver 名称> 替换为管理 LIF IP 地址和 SVM 名称。您必须确保 LIF 的服务策略已设置为 `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、密钥和受信任的 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，以使用不同的身份验证方法或轮换其凭据。这种方法是双向的：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，您必须删除现有的身份验证方法并添加新的身份验证方法。然后使用包含所需参数的更新后的 backend.json 文件来执行 `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，存储管理员必须首先更新ONTAP上用户的密码。接下来将进行后端更新。轮换证书时，可以为用户添加多个证书。然后更新后端以使用新证书，之后即可从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，也不会影响之后建立的卷连接。后端更新成功表明Trident可以与ONTAP后端通信并处理未来的卷操作。

为Trident创建自定义ONTAP角色

您可以创建一个具有最低权限的ONTAP集群角色，这样您就不必使用ONTAP管理员角色在Trident中执行操作。在Trident后端配置中包含用户名时， Trident将使用您创建的ONTAP集群角色来执行操作。

请参阅"Trident自定义角色生成器"有关创建Trident自定义角色的更多信息。

使用ONTAP CLI

1. 使用以下命令创建新角色：

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. 为Trident用户创建用户名：

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. 将角色映射到用户：

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

使用系统管理器

在ONTAP系统管理器中执行以下步骤：

1. 创建自定义角色：
   a. 要在集群级别创建自定义角色，请选择“集群 > 设置”。

      （或者）要在 SVM 级别创建自定义角色，请选择“存储”>“存储虚拟机”> `required SVM` > 设置 > 用户和角色*。

   b. 选择“用户和角色”旁边的箭头图标（→）。
   c. 在“角色”下选择“+添加”。
   d. 定义角色规则，然后点击“保存”。
2. 将角色映射到Trident用户： + 在“用户和角色”页面上执行以下步骤：
   a. 在“用户”下方选择“添加”图标 **+**。
   b. 选择所需的用户名，然后在“角色”下拉菜单中选择角色。
   c. 单击“保存”。

更多信息请参阅以下页面：

- "用于管理ONTAP的自定义角色"或者"定义自定义角色"

- "与角色和用户协作"

管理 **NFS** 导出策略

Trident使用 NFS 导出策略来控制对其所配置卷的访问。

Trident在处理出口策略时提供两种选择：

- Trident可以动态管理导出策略本身；在这种操作模式下，存储管理员指定一个 CIDR 块列表，这些 CIDR 块代表可接受的 IP 地址。 Trident会在发布时自动将属于这些范围内的适用节点 IP 添加到导出策略中。或者，如果没有指定 CIDR，则会将发布卷的节点上找到的所有全局范围的单播 IP 添加到导出策略中。
- 存储管理员可以创建导出策略并手动添加规则。除非在配置中指定了不同的导出策略名称，否则Trident将使用默认导出策略。

动态管理出口策略

Trident提供了动态管理ONTAP后端导出策略的功能。这样，存储管理员就可以指定工作节点 IP 的允许地址空间，而无需手动定义明确的规则。它大大简化了导出策略管理；对导出策略的修改不再需要对存储集群进行人工干预。此外，这有助于将对存储集群的访问限制在仅允许挂载卷且 IP 地址在指定范围内的节点上，从而支持细粒度和自动化管理。

> 使用动态导出策略时，请勿使用网络地址转换（NAT）。使用 NAT 时，存储控制器看到的是前端 NAT 地址而不是实际的 IP 主机地址，因此当在导出规则中找不到匹配项时，访问将被拒绝。

示例

必须使用两种配置选项。以下是一个后端定义示例：

```
---
version: 1
storageDriverName: ontap-nas-economy
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，必须确保 SVM 中的根连接点具有先前创建的导出策略，该策略的导出规则允许节点 CIDR 块（例如默认导出策略）。始终遵循NetApp推荐的最佳实践，为Trident专用一个 SVM。

下面以上述示例为例，解释此功能的工作原理：

- `autoExportPolicy`设置为 `true`。这表明Trident会为使用此后端配置的每个卷创建一个导出策略。`svm1`使用 SVM 处理规则的添加和删除 `autoexportCIDRs`地址块。在卷连接到节点之前，该卷使用空的

导出策略，没有任何规则来防止对该卷的未经授权的访问。当卷发布到节点时， Trident会创建一个导出策略，该策略的名称与包含指定 CIDR 块内节点 IP 的底层 qtree 的名称相同。这些 IP 地址也将添加到父FlexVol volume使用的导出策略中。

- 例如：
  - 后端 UUID 403b5326-8482-40db-96d0-d83fb3f4daec
  - `autoExportPolicy`设置为 `true`
  - 存储前缀 `trident`
  - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
  - 名为 trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c 的 qtree 为名为FlexVol的 FlexVol 创建了一个导出策略。 `trident-403b5326-8482-40db96d0-d83fb3f4daec` ，名为 qtree 的导出策略 `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`以及一个名为“空的导出策略”的 `trident_empty`在支持向量机上。 FlexVol导出策略的规则将是 qtree 导出策略中包含的任何规则的超集。任何未附加的卷都将重用空导出策略。

- `autoExportCIDRs`包含地址块列表。此字段为可选字段，默认值为 ["0.0.0.0/0", "::/0"]。如果未定义，Trident会添加在工作节点上找到的所有具有发布的全局作用域的单播地址。

在这个例子中， `192.168.0.0/24`已提供地址空间。这意味着，位于此地址范围内且已发布 Kubernetes 节点 IP 的节点将被添加到Trident创建的导出策略中。当Trident注册它运行所在的节点时，它会检索该节点的 IP 地址，并将其与提供的地址块进行比对。 `autoExportCIDRs`在发布时， Trident在过滤 IP 地址后，会为它要发布到的节点的客户端 IP 地址创建导出策略规则。

您可以更新 `autoExportPolicy`和 `autoExportCIDRs`创建后端之后，需要进行以下操作。您可以为自动管理的后端添加新的 CIDR，或者删除现有的 CIDR。删除 CIDR 时要格外小心，确保现有连接不会断开。您也可以选择禁用 `autoExportPolicy`对于后端，如果出现问题，则回退到手动创建的导出策略。这将需要进行设置 `exportPolicy`后端配置中的参数。

Trident创建或更新后端后，您可以使用以下命令检查后端： `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
```

当一个节点被移除时， Trident会检查所有导出策略，以移除与该节点对应的访问规则。通过从受管后端导出策略中移除此节点 IP， Trident可以防止恶意挂载，除非集群中的新节点重新使用此 IP。

对于先前存在的后端，使用以下方式更新后端： `tridentctl update backend`确保Trident自动管理出口策略。这样，当需要时，就会创建两个以后端 UUID 和 qtree 名称命名的新导出策略。后端存在的卷在卸载并重新挂载后，将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。如果后端被重新创建，它将被视为一个新的后端，并将导致创建一个新的导出策略。

如果运行中的节点的 IP 地址更新，则必须重启该节点上的Trident pod。 Trident随后将更新其管理的后端的导出策略，以反映此 IP 变更。

准备配置SMB卷

稍作准备，您就可以使用以下方式配置 SMB 卷 `ontap-nas`司机。

> 您必须在 SVM 上同时配置 NFS 和 SMB/CIFS 协议才能创建 `ontap-nas-economy`适用于ONTAP本地集群的 SMB 卷。如果未能配置这些协议中的任何一个，都会导致 SMB 卷创建失败。

> `autoExportPolicy`不支持 SMB 卷。

开始之前

在配置 SMB 卷之前，您必须具备以下条件。

- 一个 Kubernetes 集群，包含一个 Linux 控制器节点和至少一个运行 Windows Server 2022 的 Windows 工作节点。 Trident仅支持挂载到运行在 Windows 节点上的 pod 的 SMB 卷。
- 至少有一个包含您的 Active Directory 凭据的Trident密钥。生成秘密 `smbcreds`：

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

- 配置为 Windows 服务的 CSI 代理。要配置 `csi-proxy`，请参阅"GitHub：CSI代理"或者"GitHub：适用于 Windows 的 CSI 代理"适用于在 Windows 上运行的 Kubernetes 节点。

步骤

1. 对于本地部署的ONTAP，您可以选择创建 SMB 共享，或者Trident可以为您创建一个。

   Amazon FSx for ONTAP需要 SMB 共享。

   您可以通过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 CLI。使用ONTAP CLI 创建 SMB 共享：

   a. 如有必要，请创建共享的目录路径结构。

      这 `vserver cifs share create`该命令检查在创建共享时 -path 选项中指定的路径。如果指定的路径不存在，则命令执行失败。

   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. 确认共享已创建：

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

   请参阅"创建 SMB 共享"了解详细信息。

2. 创建后端时，必须配置以下内容以指定 SMB 卷。有关所有 FSx for ONTAP后端配置选项，请参阅"FSx for ONTAP配置选项和示例"。

参数	描述	示例
`smbShare`	您可以指定以下一项：使用 Microsoft 管理控制台或ONTAP CLI 创建的 SMB 共享的名称；允许Trident创建 SMB 共享的名称；或者您可以将参数留空以防止对卷的公共共享访问。对于本地部署的ONTAP，此参数是可选的。此参数是Amazon FSx for ONTAP后端所必需的，不能为空。	`smb-share`
`nasType`	*必须设置为 `smb`.*如果为空，则默认为空。 `nfs` 。	`smb`
`securityStyle`	新卷的安全样式。 必须设置为 **`ntfs`**或者 **`mixed`**适用于 **SMB** 卷。	`ntfs`或者 `mixed`适用于 SMB 卷
`unixPermissions`	新卷模式。 对于 **SMB** 卷，此项必须留空。	""

启用安全的 **SMB**

从 25.06 版本开始， NetApp Trident支持使用以下方式安全配置创建的 SMB 卷： `ontap-nas`和 `ontap-nas-economy`后端。启用安全 SMB 后，您可以使用访问控制列表 (ACL) 为 Active Directory (AD) 用户和用户组提供对 SMB 共享的受控访问。

需要记住的要点

- 输入 `ontap-nas-economy`不支持卷。
- 仅支持只读克隆 `ontap-nas-economy`卷。
- 如果启用了安全 SMB， Trident将忽略后端提到的 SMB 共享。
- 更新 PVC 注解、存储类注解和后端字段不会更新 SMB 共享 ACL。
- 克隆 PVC 注释中指定的 SMB 共享 ACL 将优先于源 PVC 中的 ACL。
- 启用安全 SMB 时，请确保提供有效的 AD 用户。无效用户将不会被添加到访问控制列表 (ACL) 中。
- 如果在后端、存储类和 PVC 中为同一个 AD 用户提供不同的权限，则权限优先级为：PVC、存储类，然后是后端。
- 支持安全 SMB `ontap-nas`仅适用于托管卷导入，不适用于非托管卷导入。

步骤

1. 在 TridentBackendConfig 中指定 adAdminUser，如下例所示：

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. 在存储类中添加注解。

   添加 `trident.netapp.io/smbShareAdUser`对存储类进行注解，以启用安全可靠的 SMB 连接。用户为注释指定的值 `trident.netapp.io/smbShareAdUser`应该与指定的用户名相同 `smbcreds`秘密。您可以从以下选项中选择一项 `smbShareAdUserPermission: full_control , change , 或者 read。默认权限是 full_control。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. 创建 PVC。

   以下示例创建PVC：

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

ONTAP NAS 配置选项和示例

学习如何在Trident系统中创建和使用ONTAP NAS 驱动程序。本节提供后端配置示例以及将后端映射到 StorageClasses 的详细信息。

后端配置选项

请参阅下表了解后端配置选项：

参数	描述	默认
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	`ontap-nas`， `ontap-nas-economy` ， 或者 `ontap-nas-flexgroup`
`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 地址可以指定完全限定域名 (FQDN)。如果Trident安装时使用了 IPv6 标志，则可以设置为使用 IPv6 地址。 IPv6 地址必须用方括号定义，例如：`[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` 。为了实现MetroCluster 的无缝切换，请参阅MetroCluster示例。	"10.0.0.1", "[2001:1234:abcd::fefe]"

参数	描述	默认
dataLIF	LIF协议的IP地址。 NetApp建议指定 dataLIF。如果未提供， Trident将从 SVM 获取 dataLIF。您可以指定一个完全限定域名 (FQDN) 用于 NFS 挂载操作，从而创建轮询 DNS 以在多个 dataLIF 之间进行负载均衡。初始设置后可以更改。参考。如果Trident安装时使用了 IPv6 标志，则可以设置为使用 IPv6 地址。 IPv6 地址必须用方括号定义，例如： [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555] 。 *Metrocluster 除外。*查看MetroCluster示例。	指定地址或从 SVM 派生，如果未指定（不推荐）
svm	要使用的存储虚拟机 *Metrocluster 除外。*查看MetroCluster示例。	如果是 SVM 则推导而来 `managementLIF`已指定
autoExportPolicy	启用自动导出策略创建和更新 [布尔值]。使用 `autoExportPolicy`和 `autoExportCIDRs`可选功能包括： Trident可以自动管理出口策略。	false
autoExportCIDRs	用于过滤 Kubernetes 节点 IP 的 CIDR 列表 `autoExportPolicy`已启用。使用 `autoExportPolicy`和 `autoExportCIDRs`可选功能包括： Trident可以自动管理出口策略。	["0.0.0.0/0", "::/0"]`
labels	要应用于卷的任意 JSON 格式标签集	""
clientCertificate	客户端证书的 Base64 编码值。用于基于证书的身份验证	""
clientPrivateKey	客户端私钥的 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	用于连接到集群/SVM 的用户名。用于基于凭据的身份验证。有关 Active Directory 身份验证，请参阅 "使用 Active Directory 凭据向后端 SVM 验证Trident 的身份"。	
password	连接到集群/SVM 的密码。用于基于凭据的身份验证。有关 Active Directory 身份验证，请参阅 "使用 Active Directory 凭据向后端 SVM 验证Trident 的身份"。	
storagePrefix	在 SVM 中配置新卷时使用的前缀。设置后无法更新 当使用 ontap-nas-economy 和 24 个或更多字符的 storagePrefix 时，qtree 将不会嵌入存储前缀，尽管它会包含在卷名称中。	“三叉戟”

参数	描述	默认
`aggregate`	用于配置的聚合（可选；如果设置，则必须分配给SVM）。对于 `ontap-nas-flexgroup`驱动程序，此选项将被忽略。如果未分配，则可以使用任何可用的聚合来配置FlexGroup卷。 当 SVM 中的聚合数据更新时， Trident会自动轮询 SVM 进行更新，而无需重启Trident控制器。当您在Trident中配置特定聚合以配置卷时，如果该聚合被重命名或移出 SVM，则在轮询 SVM 聚合时， Trident中的后端将变为失败状态。您必须将聚合更改为 SVM 上存在的聚合，或者将其完全删除，才能使后端恢复联机。	""
`limitAggregateUsage`	如果使用率超过此百分比，则配置失败。 不适用于**Amazon FSx for ONTAP**。	（默认情况下不强制执行）
flexgroupAggregateList	用于配置的聚合列表（可选；如果设置，则必须分配给 SVM）。分配给 SVM 的所有聚合都用于配置FlexGroup卷。支持 **ontap-nas-flexgroup** 存储驱动程序。 当 SVM 中的聚合列表更新时， Trident会通过轮询 SVM 自动更新该列表，而无需重新启动Trident控制器。当您在Trident中配置了特定的聚合列表来配置卷时，如果聚合列表被重命名或移出SVM，则在轮询 SVM 聚合时， Trident中的后端将进入失败状态。您必须将聚合列表更改为 SVM 上存在的列表，或者将其完全删除，才能使后端恢复联机。	""
`limitVolumeSize`	如果请求的卷大小大于此值，则配置失败。此外，它还限制了 qtree 管理的卷的最大大小，以及 `qtreesPerFlexvol`此选项允许自定义每个FlexVol volume的最大 qtree 数量。	（默认情况下不强制执行）
`debugTraceFlags`	故障排除时要使用的调试标志。例如，{"api":false, "method":true} 请勿使用 `debugTraceFlags`除非您正在进行故障排除并且需要详细的日志转储。	无效的
`nasType`	配置 NFS 或 SMB 卷的创建。选项有 `nfs`， `smb`或空值。设置为 null 则默认使用 NFS 卷。	`nfs`
`nfsMountOptions`	以逗号分隔的 NFS 挂载选项列表。 Kubernetes 持久卷的挂载选项通常在存储类中指定，但如果存储类中没有指定挂载选项， 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任何挂载选项， Trident将不会在关联的持久卷上设置任何挂载选项。	""

参数	描述	默认
qtreesPerFlexvol	每个FlexVol 的最大 Qtree 数量必须在 [50, 300] 范围内	“200”
smbShare	您可以指定以下一项：使用 Microsoft 管理控制台或ONTAP CLI 创建的 SMB 共享的名称；允许Trident 创建 SMB 共享的名称；或者您可以将参数留空以防止对卷的公共共享访问。对于本地部署的ONTAP，此参数是可选的。此参数是Amazon FSx for ONTAP后端所必需的，不能为空。	smb-share
useREST	使用ONTAP REST API 的布尔参数。 `useREST`设置为 `true`Trident使用ONTAP REST API 与后端通信；当设置为 `false`Trident使用 ONTAPI (ZAPI) 调用与后端通信。此功能需要ONTAP 9.11.1 及更高版本。此外，所使用的ONTAP登录角色必须具有访问权限。 `ontapi`应用。预定义项满足了这一点。 `vsadmin`和 `cluster-admin`角色。从Trident 24.06 版本和ONTAP 9.15.1 或更高版本开始， `useREST`设置为 `true`默认值；更改 `useREST`到 `false`使用 ONTAPI (ZAPI) 调用。	true`适用于ONTAP 9.15.1 或更高版本，否则 `false。
limitVolumePoolSize	在 ontap-nas-economy 后端使用 Qtrees 时可请求的最大FlexVol大小。	（默认情况下不强制执行）
denyNewVolumePools	限制 `ontap-nas-economy`后端创建新的FlexVol卷来包含它们的 Qtree。只有预先存在的 Flexvol 才能用于配置新的 PV。	
adAdminUser	具有对 SMB 共享完全访问权限的 Active Directory 管理员用户或用户组。使用此参数可为 SMB 共享提供管理员权限，并赋予其完全控制权限。	

卷配置的后端配置选项

您可以使用以下选项控制默认配置。 `defaults`配置部分。例如，请参见下面的配置示例。

参数	描述	默认
spaceAllocation	Q树的空间分配	“真的”
spaceReserve	空间预留模式；“无”（细）或“大量”（粗）	“没有任何”
snapshotPolicy	要使用的快照策略	“没有任何”
qosPolicy	要为创建的卷分配的 QoS 策略组。每个存储池/后端选择 qosPolicy 或 adaptiveQosPolicy 之一	""
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。每个存储池/后端选择 qosPolicy 或 adaptiveQosPolicy 之一。ontap-nas-economy 不支持此功能。	""
snapshotReserve	快照预留的卷百分比	如果为“0”， `snapshotPolicy`为“无”，否则为“

参数	描述	默认
splitOnClone	创建时将克隆体从其母体中分离出来	"false"
encryption	在新卷上启用NetApp卷加密 (NVE)；默认设置为 false。要使用此选项，必须在集群上获得 NVE 许可并启用 NVE。如果后端启用了 NAE，则在Trident中配置的任何卷都将启用 NAE。更多信息，请参阅："Trident如何与 NVE 和 NAE 协同工作" 。	"false"
tieringPolicy	分层策略使用“无”	
unixPermissions	新卷模式	NFS 卷的端口号为“777”；SMB 卷的端口号为空（不适用）。
snapshotDir	控制对以下方面的访问 `.snapshot`目录	NFSv4 为“true”，NFSv3 为“false”。
exportPolicy	出口政策的使用	“默认”
securityStyle	新卷的安全样式。 NFS 支持 `mixed`和 `unix`安全措施。中小企业支持 `mixed`和 `ntfs`安全措施。	NFS 默认值为 unix。 SMB 默认值为 ntfs。
nameTemplate	用于创建自定义卷名称的模板。	""

将 QoS 策略组与Trident结合使用需要ONTAP 9.8 或更高版本。您应该使用非共享的 QoS 策略组，并确保该策略组单独应用于每个成员。共享的 QoS 策略组强制规定所有工作负载的总吞吐量上限。

卷配置示例

以下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"
```

为了 ontap-nas`和 `ontap-nas-flexgroups`Trident现在使用新的计算方法，以确保FlexVol 的尺寸与 snapshotReserve 百分比和 PVC 正确匹配。当用户请求 PVC 时， Trident会使用新的计算方法创建具有更多空间的原始FlexVol 。此计算方法可确保用户在 PVC 中获得其请求的可写空间，而不是少于其请求的空间。在 v21.07 之前，当用户请求 PVC（例如 5 GiB）时，如果快照预留百分比为 50%，则只能获得 2.5 GiB 的可写空间。这是因为用户请求的是整个卷。 `snapshotReserve`是其中的百分比。在Trident 21.07 中，用户请求的是可写空间，而Trident定义了该空间。 `snapshotReserve`将数值表示为占总体积的百分比。这不适用于 `ontap-nas-economy。请参阅以下示例了解其工作原理：

计算方法如下：

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

对于快照预留 = 50% 且 PVC 请求 = 5 GiB 的情况，总卷大小为 5/.5 = 10 GiB，可用大小为 5 GiB，这正是用户在 PVC 请求中请求的大小这 `volume show`该命令应显示与此示例类似的结果：

升级Trident时，先前安装的现有后端将按上述方式配置卷。对于升级前创建的卷，您应该调整其大小才能观察到更改。例如，一个 2 GiB 的 PVC `snapshotReserve=50`之前的结果是一个提供 1 GiB 可写空间的卷。例如，将卷大小调整为 3 GiB，将在 6 GiB 的卷上为应用程序提供 3 GiB 的可写空间。

最小配置示例

以下示例展示了基本配置，其中大多数参数都保留默认值。这是定义后端最简单的方法。

> 如果您在NetApp ONTAP上使用Amazon FSx和Trident，建议为 LIF 指定 DNS 名称而不是 IP 地址。

ONTAP NAS 经济示例

```
---
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 Flexgroup 示例

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

MetroCluster示例

您可以配置后端，以避免在切换和切换回后端后手动更新后端定义。"SVM复制和恢复" 。

为了实现无缝切换和切换回，请指定 SVM `managementLIF`并省略 `dataLIF`和 `svm`参数。例如：

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

SMB 卷示例

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

基于证书的身份验证示例

这是一个最小后端配置示例。 `clientCertificate` ， `clientPrivateKey` ， 和 `trustedCACertificate`（如果使用受信任的 CA，则为可选） `backend.json`分别取客户端证书、私钥和受信任 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_
```

自动出口政策示例

本示例向您展示如何指示Trident使用动态导出策略来自动创建和管理导出策略。这对于以下情况也适用：`ontap-nas-economy`和 `ontap-nas-flexgroup`司机。

```
---
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
```

IPv6 地址示例

这个例子表明 `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
```

使用 **SMB** 卷的**Amazon FSx for ONTAP**示例

这 `smbShare`使用 SMB 卷的 FSx for ONTAP需要此参数。

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

使用 **nameTemplate** 的后端配置示例

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

具有虚拟池的后端示例

在下方所示的示例后端定义文件中，所有存储池都设置了特定的默认值，例如： `spaceReserve`没有，`spaceAllocation`为假，并且 `encryption`错误。虚拟池在存储部分中定义。

Trident在“备注”字段中设置配置标签。 FlexVol上已设置评论 `ontap-nas`或`FlexGroup `ontap-nas-flexgroup`。 Trident在配置时会将虚拟池上的所有标签复制到存储卷。为了方便起见，存储管理员可以为每个虚拟池定义标签，并按标签对卷进行分组。

在这些示例中，一些存储池会设置自己的参数。 `spaceReserve`， `spaceAllocation`， 和 `encryption`有些值会覆盖默认值，有些池会覆盖默认值。

ONTAP NAS 示例

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
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:
      department: legal
      creditpoints: "5000"
    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"
```

ONTAP NAS FlexGroup示例

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
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"
```

ONTAP NAS 经济示例

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
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:
      protection: bronze
      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"
```

将后端映射到存储类

以下 StorageClass 定义指的是[具有虚拟池的后端示例]。使用 `parameters.selector`字段中，每个 StorageClass 都会指定哪些虚拟池可用于托管卷。卷将具有所选虚拟池中定义的各个方面。

- 这 `protection-gold`StorageClass 将映射到第一个和第二个虚拟池。 `ontap-nas-flexgroup`后端。只有这些泳池提供黄金级保护。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- 这 `protection-not-gold`StorageClass 将映射到第三个和第四个虚拟池。 `ontap-nas-flexgroup`后端。除了黄金之外，只有这些金池提供其他级别的保护。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- 这 `app-mysqldb`StorageClass 将映射到第四个虚拟池。 `ontap-nas`后端。这是唯一一个为mysqldb类型应用程序提供存储池配置的存储池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- 该 `protection-silver-creditpoints-20k`StorageClass 将映射到第三个虚拟池。 `ontap-nas-flexgroup`后端。这是唯一提供银级保护和 20000 积分的彩池。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- 这 `creditpoints-5k`StorageClass 将映射到第三个虚拟池。 `ontap-nas`后端和第二个虚拟池 `ontap-nas-economy`后端。这是唯一提供 5000 积分的彩池产品。

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident将决定选择哪个虚拟池，并确保满足存储需求。

更新 `dataLIF`初始配置后

初始配置完成后，您可以通过运行以下命令来更改 dataLIF，从而为新的后端 JSON 文件提供更新后的 dataLIF。

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

如果 PVC 连接到一个或多个舱体，则必须将所有相应的舱体放下，然后再将它们重新升起，以便新的 dataLIF 生效。

安全的中小企业示例

使用 **ontap-nas** 驱动程序进行后端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

使用 **ontap-nas-economy** 驱动程序进行后端配置

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

后端配置及存储池

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

使用 **ontap-nas** 驱动程序的存储类示例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

请确保添加 `annotations`实现安全的SMB。如果没有注释，无论后端或 PVC 中设置了什么配置，安全 SMB 都无法工作。

使用 **ontap-nas-economy** 驱动程序的存储类示例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

包含单个 **AD** 用户的 **PVC** 示例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

包含多个 **AD** 用户的 **PVC** 示例

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

将Trident与Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP"是一项完全托管的 AWS 服务，使客户能够启动和运行由NetApp ONTAP存储操作系统支持的文件系统。 FSx for ONTAP使您能够利用您熟悉的NetApp功能、性能和管理能力，同时享受在 AWS 上存储数据的简单性、敏捷性、安全性和可扩展性。 FSx for ONTAP支持ONTAP文件系统功能和管理 API。

您可以将Amazon FSx for NetApp ONTAP文件系统与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支持的块和文件持久卷。

文件系统是Amazon FSx中的主要资源，类似于本地的ONTAP集群。在每个 SVM 中，您可以创建一个或多个卷，这些卷是用于存储文件系统中的文件和文件夹的数据容器。 Amazon FSx for NetApp ONTAP将作为云端托管文件系统提供。新的文件系统类型称为 * NetApp ONTAP*。

通过将Trident与Amazon FSx for NetApp ONTAP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支持的块和文件持久卷。

要求

此外"Trident的要求"要将 FSx for ONTAP与Trident集成，您需要：

- 现有的 Amazon EKS 集群或自管理 Kubernetes 集群 `kubectl`已安装。
- 集群工作节点可访问的现有Amazon FSx for NetApp ONTAP文件系统和存储虚拟机 (SVM)。
- 已准备好的工作节点"NFS 或 iSCSI"。

> 请务必按照 Amazon Linux 和 Ubuntu 所需的节点准备步骤进行操作。 "亚马逊机器图像"（AMI）取决于您的 EKS AMI 类型。

注意事项

- SMB 卷：
  - 使用以下方式支持 SMB 卷 `ontap-nas`仅限司机。
  - Trident EKS 插件不支持 SMB 卷。
  - Trident仅支持挂载到运行在 Windows 节点上的 pod 的 SMB 卷。参考 "准备配置SMB卷" 了解详情。
- 在Trident 24.02 之前，在Amazon FSx文件系统上创建的、启用了自动备份的卷无法被Trident删除。为防止在Trident 24.02 或更高版本中出现此问题，请指定 `fsxFilesystemID`，AWS `apiRegion` ，AWS `apikey`以及 AWS `secretKey`在 AWS FSx for ONTAP的后端配置文件中。

> 如果您要为Trident指定 IAM 角色，则可以省略指定以下内容： `apiRegion` ， `apiKey` ， 和 `secretKey`明确地将字段传递给Trident 。更多信息，请参阅"FSx for ONTAP配置选项和示例"。

同时使用**Trident SAN/iSCSI** 和 **EBS-CSI** 驱动程序

如果您计划将 ontap-san 驱动程序（例如 iSCSI）与 AWS（EKS、ROSA、EC2 或任何其他实例）一起使用，则节点上所需的多路径配置可能会与 Amazon Elastic Block Store (EBS) CSI 驱动程序冲突。为了确保多路径功能不会干扰同一节点上的 EBS 磁盘，您需要在多路径设置中排除 EBS。这个例子展示了 `multipath.conf`包含所需Trident设置但排除 EBS 磁盘进行多路径配置的文件：

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

身份验证

Trident提供两种身份验证方式。

- 基于凭证（推荐）：将凭证安全地存储在 AWS Secrets Manager 中。您可以使用 `fsxadmin`文件系统的用户或 `vsadmin`用户已配置到您的 SVM。

> Trident预计将以……的形式运营 `vsadmin`SVM 用户，或者使用具有相同角色但名称不同的用户。 Amazon FSx for NetApp ONTAP具有 `fsxadmin`该用户是ONTAP的有限替代品 `admin`集群用户。我们强烈建议使用 `vsadmin`用Trident。

- 基于证书： Trident将使用安装在 SVM 上的证书与 FSx 文件系统上的 SVM 进行通信。

有关启用身份验证的详细信息，请参阅您的驱动程序类型的身份验证说明：

- "ONTAP NAS 认证"
- "ONTAP SAN 身份验证"

已测试的亚马逊机器映像 **(AMI)**

EKS 集群支持各种操作系统，但 AWS 针对容器和 EKS 优化了某些 Amazon Machine Image (AMI)。以下 AMI 已使用NetApp Trident 25.02 进行测试。

急性心肌梗死	NAS	NAS经济	iSCSI	iSCSI经济型
AL2023_x86_64_STANDARD	是	是	是	是
AL2_x86_64	是	是	是的*	是的*
BOTTLEROCKET_x86_64	是的**	是	不适用	不适用
AL2023_ARM_64_STANDARD	是	是	是	是
AL2_ARM_64	是	是	是的*	是的*
BOTTLEROCKET_ARM_64	是的**	是	不适用	不适用

- * 如果不重启节点，则无法删除 PV
- ** 不适用于Trident版本 25.02 的 NFSv3。

> 如果您所需的 AMI 未在此处列出，并不意味着它不受支持；这仅仅意味着它尚未经过测试。此列表可作为 AMI 已知可用系统的指南。

使用以下工具进行测试：

- EKS版本：1.32
- 安装方法：Helm 25.06 和 AWS 附加组件 25.06
- 对于 NAS，NFSv3 和 NFSv4.1 都进行了测试。
- SAN 仅测试了 iSCSI，未测试 NVMe-oF。

已执行测试：

- 创建：存储类、PVC、Pod
- 删除：pod、pvc（常规、qtree/lun – 经济型、带 AWS 备份的 NAS）

查找更多信息

- "Amazon FSx for NetApp ONTAP文档"
- "关于Amazon FSx for NetApp ONTAP的博客文章"

创建 IAM 角色和 AWS Secret

您可以配置 Kubernetes pod 以通过 AWS IAM 角色进行身份验证来访问 AWS 资源，而不是提供显式的 AWS 凭证。

要使用 AWS IAM 角色进行身份验证，您必须拥有一个使用 EKS 部署的 Kubernetes 集群。

创建 AWS Secrets Manager 密钥

由于Trident将向 FSx 虚拟服务器发出 API 来为您管理存储，因此它需要凭据才能执行此操作。传递这些凭证的安全方法是通过 AWS Secrets Manager 密钥。因此，如果您还没有 AWS Secrets Manager 密钥，则需要创建一个包含 vsadmin 帐户凭证的密钥。

此示例创建一个 AWS Secrets Manager 密钥来存储Trident CSI 凭证：

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

创建 IAM 策略

Trident也需要 AWS 权限才能正常运行。因此，您需要创建一个策略，赋予Trident所需的权限。

以下示例使用 AWS CLI 创建 IAM 策略：

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

策略 JSON 示例：

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

创建 **Pod** 身份或 **IAM** 角色以关联服务帐户 **(IRSA)**

您可以配置 Kubernetes 服务账户，使其承担 AWS Identity and Access Management (IAM) 角色（使用 EKS Pod Identity 或 IAM 角色进行服务账户关联）(IRSA)。任何配置为使用该服务账户的 Pod 都可以访问该角色有权访问的任何 AWS 服务。

Pod 身份

Amazon EKS Pod 身份关联允许您管理应用程序的凭证，类似于 Amazon EC2 实例配置文件向 Amazon EC2 实例提供凭证的方式。

在 **EKS** 集群上安装 **Pod Identity**：

您可以通过 AWS 控制台创建 Pod 标识，也可以使用以下 AWS CLI 命令：

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

更多信息请参阅"设置 Amazon EKS Pod 身份代理"。

创建 **trust-relationship.json** 文件：

创建 trust-relationship.json 文件，使 EKS 服务主体能够承担 Pod 身份的此角色。然后使用以下信任策略创建角色：

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json 文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

将角色策略附加到 **IAM** 角色：

将上一步创建的角色策略附加到已创建的 IAM 角色：

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

创建 **Pod** 身份关联：

在 IAM 角色和Trident服务帐户 (trident-controller) 之间创建 Pod 身份关联

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

服务帐户关联 **(IRSA)** 的 **IAM** 角色

使用 **AWS CLI**：

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

trust-relationship.json 文件：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

更新以下值 `trust-relationship.json`文件：

- **<account_id>** - 您的 AWS 账户 ID
- **<oidc_provider>** - 您的 EKS 集群的 OIDC。您可以通过运行以下命令获取oidc_provider：

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

将 **IAM** 角色与 **IAM** 策略关联：

角色创建完成后，使用以下命令将策略（在上一步中创建的策略）附加到该角色：

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

请核实 **OICD** 提供商是否已关联：

请确认您的 OIDC 提供程序已与您的集群关联。您可以使用以下命令进行验证：

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

如果输出为空，请使用以下命令将 IAM OIDC 关联到您的集群：

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

如果您使用的是 **eksctl**，请使用以下示例在 EKS 中为服务帐户创建 IAM 角色：

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

安装**Trident**

Trident简化了 Kubernetes 中Amazon FSx for NetApp ONTAP存储管理，使您的开发人员和管理员能够专注于应用程序部署。

您可以使用以下方法之一安装Trident：

- 舵
- EKS 附加组件

如果要使用快照功能，请安装 CSI 快照控制器插件。请参阅"为CSI卷启用快照功能"了解更多信息。

通过 **Helm** 安装**Trident**。

Pod 身份

1. 添加Trident Helm 仓库：

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

2. 请按照以下示例安装Trident：

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

您可以使用 `helm list`查看安装详细信息的命令，例如名称、命名空间、图表、状态、应用程序版本和修订号。

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

服务账户协会（**IRSA**）

1. 添加Trident Helm 仓库：

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

2. 设置*云提供商*和*云身份*的值：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

您可以使用 `helm list`查看安装详细信息的命令，例如名称、命名空间、图表、状态、应用程序版本和修订号。

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2506.0    25.06.0
```

如果您计划使用 iSCSI，请确保您的客户端计算机上已启用 iSCSI。如果您使用的是 AL2023 工作节点操作系统，可以通过在 Helm 安装中添加节点准备参数来自动安装 iSCSI 客户端：

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

通过 **EKS** 插件安装**Trident**

Trident EKS 插件包含最新的安全补丁和错误修复，并经过 AWS 验证，可与 Amazon EKS 配合使用。 EKS 插件使您能够持续确保您的 Amazon EKS 集群安全稳定，并减少安装、配置和更新插件所需的工作量。

前提条件

在为 AWS EKS 配置Trident插件之前，请确保您已具备以下条件：

- 带有附加订阅的 Amazon EKS 集群帐户
- AWS 对 AWS Marketplace 的权限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 类型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 Arm (AL2_ARM_64)
- 节点类型：AMD 或ARM
- 现有的Amazon FSx for NetApp ONTAP文件系统

启用适用于 **AWS** 的**Trident**插件

管理控制台

1. 打开 Amazon EKS 控制台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，选择“集群”。
3. 选择要为其配置NetApp Trident CSI 插件的集群名称。
4. 选择“附加组件”，然后选择“获取更多附加组件”。
5. 请按照以下步骤选择插件：
   a. 向下滚动到“AWS Marketplace 附加组件”部分，然后在搜索框中输入“Trident”。
   b. 选中“Trident by NetApp”框右上角的复选框。
   c. 选择“下一步”。
6. 在“配置所选插件”设置页面上，执行以下操作：

   如果您使用 **Pod Identity** 关联，请跳过这些步骤。

   a. 请选择您要使用的*版本*。
   b. 如果您使用IRSA身份验证，请确保设置可选配置设置中提供的配置值：
      - 请选择您要使用的*版本*。
      - 按照*附加组件配置方案*进行操作，并将*配置值*部分中的*configurationValues*参数设置为您在上一步创建的角色 ARN（值应采用以下格式）：

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
如果选择“覆盖”作为冲突解决方法，则现有插件的一个或多个设置可能会被 Amazon EKS 插件设置覆盖。如果您不启用此选项，并且与您现有的设置存在冲突，则操作将失败。您可以使用生成的错误信息来排查冲突问题。在选择此选项之前，请确保 Amazon EKS 插件不会管理您需要自行管理的设置。

7. 选择“下一步”。
8. 在“审核和添加”页面上，选择“创建”。

   插件安装完成后，您将看到已安装的插件。

AWS CLI

1.创建 **`add-on.json`**文件：

对于 **Pod** 标识，请使用以下格式：

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

对于**IRSA**认证，请使用以下格式：

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

代替 `<role ARN>`使用上一步创建的角色的 ARN。

2.安装**Trident EKS** 插件。

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

以下示例命令安装Trident EKS 插件：

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

更新**Trident EKS** 插件

管理控制台

1. 打开 Amazon EKS 控制台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，选择“集群”。
3. 选择要为其更新NetApp Trident CSI 插件的集群名称。
4. 选择“附加组件”选项卡。
5. 选择“ NetApp Trident ”，然后选择“编辑”。
6. 在“配置NetApp Trident ”页面上，执行以下操作：
   a. 请选择您要使用的*版本*。
   b. 展开“可选配置设置”，并根据需要进行修改。
   c. 选择“保存更改”。

AWS CLI

以下示例更新 EKS 插件：

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- 请检查您的 FSxN Trident CSI 插件的当前版本。代替 `my-cluster`使用您的集群名称。

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  示例输出：

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 将插件更新到上一步输出中“UPDATE AVAILABLE”下返回的版本。

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

如果你移除 `--force`如果选项和任何 Amazon EKS 插件设置与您现有的设置冲突，则更新 Amazon EKS 插件将失败；您将收到一条错误消息，以帮助您解决冲突。在指定此选项之前，请确保 Amazon EKS 插件不会管理您需要管理的设置，因为此选项会覆盖这些设置。有关此设置的其他选项的更多信息，请参阅"插件"。有关 Amazon EKS Kubernetes 字段管理的更多信息，请参阅"Kubernetes 字段管理"。

卸载/移除**Trident EKS** 插件

您可以通过两种方式移除 Amazon EKS 插件：

- 保留集群上的附加软件 – 此选项移除 Amazon EKS 对所有设置的管理。它还取消了 Amazon EKS 通知您更新以及在您启动更新后自动更新 Amazon EKS 插件的功能。但是，它可以保留集群上的附加软件。此选项使插件成为自管理安装，而不是 Amazon EKS 插件。选择此选项，插件不会出现停机时间。保留 `--preserve`命令中的选项用于保留插件。
- 从集群中完全移除附加软件 – NetApp建议，仅当集群中没有任何资源依赖于 Amazon EKS 附加组件时，才从集群中移除该附加组件。移除 `--preserve`选项 `delete`移除插件的命令。

如果插件关联了 IAM 帐户，则不会删除该 IAM 帐户。

管理控制台

1. 打开 Amazon EKS 控制台 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，选择“集群”。
3. 选择要从中移除NetApp Trident CSI 插件的集群名称。
4. 选择“附加组件”选项卡，然后选择“ NetApp Trident ”。
5. 选择*删除*。
6. 在“移除 netapp_trident-operator 确认”对话框中，执行以下操作：
   a. 如果您希望 Amazon EKS 停止管理插件的设置，请选择“在集群上保留”。如果您希望在集群上保留附加软件，以便您可以自行管理附加软件的所有设置，请执行此操作。
   b. 输入 **netapp_trident-operator**。
   c. 选择*删除*。

AWS CLI

代替 `my-cluster`输入集群名称，然后运行以下命令。

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

以下命令卸载Trident EKS 插件：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

ONTAP SAN 和 NAS 驱动程序集成

要创建存储后端，您需要创建 JSON 或 YAML 格式的配置文件。该文件需要指定您想要的存储类型（NAS 或 SAN）、文件系统、要从中获取存储的 SVM 以及如何对其进行身份验证。以下示例展示了如何定义基于 NAS 的存储，以及如何使用 AWS Secret 来存储要使用的 SVM 的凭证：

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

运行以下命令以创建和验证Trident后端配置 (TBC)：

- 从 yaml 文件创建 Trident 后端配置 (TBC)，并运行以下命令：

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- 验证 Trident 后端配置 (TBC) 是否已成功创建：

```
Kubectl get tbc -n trident
```

```
NAME                           BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas          tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

FSx 适用于ONTAP驱动程序的详细信息

您可以使用以下驱动程序将Trident与Amazon FSx for NetApp ONTAP集成：

- `ontap-san`每个已配置的 PV 都是其自身Amazon FSx for NetApp ONTAP卷中的一个 LUN。推荐用于块存储。
- `ontap-nas`每个已配置的 PV 都是一个完整的Amazon FSx for NetApp ONTAP卷。推荐用于NFS和SMB。
- `ontap-san-economy`每个已配置的 PV 都是一个 LUN，每个Amazon FSx for NetApp ONTAP卷可配置 LUN 的数量。
- `ontap-nas-economy`每个已配置的 PV 都是一个 qtree，每个Amazon FSx for NetApp ONTAP卷可以配置 qtree 的数量。
- `ontap-nas-flexgroup`每个已配置的 PV 都是一个完整的Amazon FSx for NetApp ONTAP FlexGroup卷。

有关司机详情，请参阅"NAS驱动程序"和"SAN驱动程序"。

配置文件创建完成后，运行以下命令将其创建在 EKS 中：

```
kubectl create -f configuration_file
```

要验证状态，请运行以下命令：

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

```
NAME                      BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

后端高级配置和示例

请参阅下表了解后端配置选项：

参数	描述	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`
`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 地址可以指定完全限定域名 (FQDN)。如果Trident安装时使用了 IPv6 标志，则可以设置为使用 IPv6 地址。 IPv6 地址必须用方括号定义，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。如果您提供 `fsxFilesystemID`在 `aws`字段中，您无需提供 `managementLIF`因为Trident可以检索 SVM `managementLIF`来自 AWS 的信息。因此，您必须提供 SVM 下用户的凭据（例如：vsadmin），并且该用户必须拥有以下权限： `vsadmin`角色。	"10.0.0.1", "[2001:1234:abcd::fefe]"

参数	描述	示例
dataLIF	LIF协议的IP地址。 * ONTAP NAS驱动程序*： NetApp建议指定dataLIF。如果未提供， Trident将从SVM 获取 dataLIF。您可以指定一个完全限定域名 (FQDN) 用于 NFS 挂载操作，从而创建轮询 DNS 以在多个 dataLIF 之间进行负载均衡。初始设置后可以更改。参考。 * ONTAP SAN 驱动程序*：不要为 iSCSI 指定。 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。如果显式定义了dataLIF，则会生成警告。如果Trident安装时使用了 IPv6 标志，则可以设置为使用 IPv6 地址。 IPv6 地址必须用方括号定义，例如 [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]。	
autoExportPolicy	启用自动导出策略创建和更新 [布尔值]。使用 `autoExportPolicy`和 `autoExportCIDRs`可选功能包括：Trident可以自动管理出口策略。	false
autoExportCIDRs	用于过滤 Kubernetes 节点 IP 的 CIDR 列表 `autoExportPolicy`已启用。使用 `autoExportPolicy`和 `autoExportCIDRs`可选功能包括：Trident可以自动管理出口策略。	"["0.0.0.0/0", "::/0"]"
labels	要应用于卷的任意 JSON 格式标签集	""
clientCertificate	客户端证书的 Base64 编码值。用于基于证书的身份验证	""
clientPrivateKey	客户端私钥的 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	连接到集群或 SVM 的用户名。用于基于凭证的身份验证。例如，vsadmin。	
password	连接集群或SVM的密码。用于基于凭证的身份验证。	
svm	使用的存储虚拟机	如果指定了 SVM 管理 LIF，则派生出该 LIF。
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

参数	描述	示例
`limitAggregateUsage`	*请勿指定用于Amazon FSx for NetApp ONTAP。*提供的 `fsxadmin`和 `vsadmin`不包含检索汇总使用情况和使用Trident限制它所需的权限。	请勿使用。
`limitVolumeSize`	如果请求的卷大小大于此值，则配置失败。此外，它还限制了其管理的 qtree 和 LUN 卷的最大大小，以及 `qtreesPerFlexvol`此选项允许自定义每个FlexVol volume的最大 qtree 数量。	（默认情况下不强制执行）
`lunsPerFlexvol`	每个 Flexvol 卷的最大 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} 请勿使用 `debugTraceFlags`除非您正在进行故障排除并且需要详细的日志转储。	无效的
`nfsMountOptions`	以逗号分隔的 NFS 挂载选项列表。Kubernetes 持久卷的挂载选项通常在存储类中指定，但如果存储类中没有指定挂载选项， 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任何挂载选项， Trident将不会在关联的持久卷上设置任何挂载选项。	""
`nasType`	配置 NFS 或 SMB 卷的创建。选项有 `nfs`， `smb`或空值。 *必须设置为 `smb`适用于 SMB 卷。*设置为 null 则默认使用 NFS 卷。	`nfs`
`qtreesPerFlexvol`	每个FlexVol volume的最大 Qtree 数量必须在 [50, 300] 范围内	`"200"`
`smbShare`	您可以指定以下名称之一：使用 Microsoft 管理控制台或ONTAP CLI 创建的 SMB 共享的名称，或者允许Trident创建 SMB 共享的名称。此参数是Amazon FSx for ONTAP后端所必需的。	`smb-share`
`useREST`	使用ONTAP REST API 的布尔参数。设置为 `true`Trident将使用ONTAP REST API 与后端进行通信。此功能需要ONTAP 9.11.1 及更高版本。此外，所使用的ONTAP登录角色必须具有访问权限。 `ontap`应用。预定义项满足了这一点。 `vsadmin`和 `cluster-admin`角色。	`false`

参数	描述	示例
aws	您可以在 AWS FSx for ONTAP的配置文件中指定以下内容： - fsxFilesystemID：指定 AWS FSx 文件系统的 ID。 - apiRegion AWS API 区域名称。 - apikey AWS API 密钥。 - secretKey AWS 密钥。	"" "" ""
credentials	指定要存储在 AWS Secrets Manager 中的 FSx SVM 凭证。 - name：包含 SVM 凭证的密钥的 Amazon 资源名称 (ARN)。 - type`设置为 `awsarn。请参阅"创建 AWS Secrets Manager 密钥"了解更多信息。	

卷配置的后端配置选项

您可以使用以下选项控制默认配置。 `defaults`配置部分。例如，请参见下面的配置示例。

参数	描述	默认
spaceAllocation	LUN 的空间分配	true
spaceReserve	空间预留模式；“无”（细）或“大量”（粗）	none
snapshotPolicy	要使用的快照策略	none
qosPolicy	要为创建的卷分配的 QoS 策略组。每个存储池或后端选择 qosPolicy 或 adaptiveQosPolicy 之一。将 QoS 策略组与Trident结合使用需要ONTAP 9.8 或更高版本。您应该使用非共享的 QoS 策略组，并确保该策略组单独应用于每个成员。共享的 QoS 策略组强制规定所有工作负载的总吞吐量上限。	""
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。每个存储池或后端选择 qosPolicy 或 adaptiveQosPolicy 之一。 ontap-nas-economy 不支持此功能。	""
snapshotReserve	为快照“0”预留的卷百分比	如果 snapshotPolicy`是 `none，else “”
splitOnClone	创建时将克隆体从其母体中分离出来	false

参数	描述	默认
`encryption`	在新卷上启用NetApp卷加密 (NVE)；默认设置为 `false`。要使用此选项，必须在集群上获得 NVE 许可并启用 NVE。如果后端启用了 NAE，则在Trident中配置的任何卷都将启用 NAE。更多信息，请参阅："Trident如何与 NVE 和 NAE 协同工作" 。	`false`
`luksEncryption`	启用LUKS加密。参考"使用 Linux 统一密钥设置 (LUKS)"。仅限 SAN。	""
`tieringPolicy`	分层策略的使用 `none`	
`unixPermissions`	新卷模式。 **SMB**卷请留空。	""
`securityStyle`	新卷的安全样式。 NFS 支持 `mixed`和 `unix`安全措施。中小企业支持 `mixed`和 `ntfs`安全措施。	NFS 默认值为 `unix`。 SMB 默认值为 `ntfs`。

准备配置**SMB**卷

您可以使用以下方式配置 SMB 卷： `ontap-nas`司机。在你完成之前ONTAP SAN 和 NAS 驱动程序集成请完成以下步骤。

开始之前

在使用以下方式配置 SMB 卷之前： `ontap-nas`驾驶员，您必须具备以下条件。

- 一个 Kubernetes 集群，包含一个 Linux 控制器节点和至少一个运行 Windows Server 2019 的 Windows 工作节点。 Trident仅支持挂载到运行在 Windows 节点上的 pod 的 SMB 卷。
- 至少有一个包含您的 Active Directory 凭据的Trident密钥。生成秘密 `smbcreds`：

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

- 配置为 Windows 服务的 CSI 代理。要配置 `csi-proxy`，请参阅"GitHub：CSI代理"或者"GitHub：适用于 Windows 的 CSI 代理"适用于在 Windows 上运行的 Kubernetes 节点。

步骤

1. 创建SMB共享。您可以通过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 CLI。使用ONTAP CLI 创建 SMB 共享：
   a. 如有必要，请创建共享的目录路径结构。

      这 `vserver cifs share create`该命令检查在创建共享时 -path 选项中指定的路径。如果指定的路径不存在，则命令执行失败。

   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. 确认共享已创建：

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

请参阅"创建 SMB 共享"了解详细信息。

2. 创建后端时，必须配置以下内容以指定 SMB 卷。有关所有 FSx for ONTAP后端配置选项，请参阅"FSx for ONTAP配置选项和示例"。

参数	描述	示例
`smbShare`	您可以指定以下名称之一：使用 Microsoft 管理控制台或ONTAP CLI 创建的 SMB 共享的名称，或者允许Trident创建 SMB 共享的名称。此参数是Amazon FSx for ONTAP后端所必需的。	`smb-share`
`nasType`	*必须设置为 `smb`.*如果为空，则默认为空。 `nfs` 。	`smb`
`securityStyle`	新卷的安全样式。 必须设置为 **`ntfs`**或者 **`mixed`**适用于 **SMB** 卷。	`ntfs`或者 `mixed`适用于 SMB 卷
`unixPermissions`	新卷模式。 对于 **SMB** 卷，此项必须留空。	""

配置存储等级和**PVC**

配置 Kubernetes StorageClass 对象并创建存储类，以指示Trident如何配置卷。创建一个使用已配置的 Kubernetes StorageClass 的 PersistentVolumeClaim (PVC) 来请求访问 PV。然后您可以将光伏组件安装到支架上。

创建存储类

配置 **Kubernetes StorageClass** 对象

这 "Kubernetes StorageClass 对象"该对象将Trident标识为该类使用的配置器，并指示Trident如何配置卷。使用此示例为使用 NFS 的卷设置 Storageclass（有关属性的完整列表，请参阅下面的Trident属性部分）：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

使用此示例为使用 iSCSI 的卷设置 Storageclass：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

要在 AWS Bottlerocket 上配置 NFSv3 卷，请添加所需的 `mountOptions`到存储类：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

请参阅"Kubernetes 和Trident对象"有关存储类如何与……交互的详细信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的参数。

创建存储类

步骤

1. 这是一个 Kubernetes 对象，所以请使用 `kubectl`在 Kubernetes 中创建它。

```
kubectl create -f storage-class-ontapnas.yaml
```

2. 现在您应该在 Kubernetes 和Trident中都看到 **basic-csi** 存储类，并且Trident应该已经发现了后端上的存储池。

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

制作**PVC**管

一个 "*PersistentVolumeClaim*"（PVC）是对集群上持久卷的访问请求。

PVC 可以配置为请求存储特定尺寸或访问模式。使用关联的 StorageClass，集群管理员可以控制的不仅仅是 PersistentVolume 的大小和访问模式，还可以控制性能或服务级别。

制作好 PVC 后，就可以将容积安装到舱体中。

样品清单

PersistentVolumeClaim 样本清单

这些示例展示了PVC的基本配置选项。

带**RWX**接口的**PVC**

此示例展示了一个具有 RWX 访问权限的基本 PVC，它与一个名为 StorageClass 的 StorageClass 相关联。 `basic-csi` 。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

使用 **iSCSI** 示例的 **PVC**

此示例展示了一个与名为 StorageClass 的存储类关联的、具有 RWO 访问权限的 iSCSI 基本 PVC。 `protection-gold` 。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 1Gi
storageClassName: protection-gold
```

创建 **PVC**

步骤

1. 创建 PVC。

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

2. 核实PVC状态。

```
kubectl get pvc
```

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

请参阅"Kubernetes 和Trident对象"有关存储类如何与……交互的详细信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的参数。

Trident属性

这些参数决定了应使用哪些 Trident 管理的存储池来配置给定类型的卷。

属性	类型	价值观	提供	要求	由……支持
媒体[1]	string	机械硬盘、混合硬盘、固态硬盘	Pool 包含此类媒体；混合型媒体是指两者兼具。	指定的媒体类型	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san
供应类型	string	薄的，厚的	池支持这种配置方法	指定的配置方法	厚：全部 ontap ；薄：全部 ontap 和 solidfire-san
后端类型	string	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、gcp-cvs 、azure-netapp-files、ontap-san-economy	池属于这种类型的后端	指定的后端	所有司机
snapshots	布尔值	真，假	存储池支持带快照的卷	已启用快照的卷	ontap-nas 、ontap-san 、solidfire-san 、gcp-cvs
个克隆	布尔值	真，假	存储池支持卷克隆	已启用克隆的卷	ontap-nas 、ontap-san 、solidfire-san 、gcp-cvs

属性	类型	价值观	提供	要求	由.....支持
加密	布尔值	真，假	存储池支持加密卷	已启用加密的卷	ontap-nas、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
IOPS	整数	正整数	Pool 能够保证在此范围内的 IOPS	容量保证了这些IOPS	solidfire-san

[1]： ONTAP Select系统不支持此系统

部署示例应用程序

创建存储等级和 PVC 后，即可将光伏组件安装到舱体上。本节列出了将 PV 连接到 pod 的示例命令和配置。

步骤

1. 将音量旋钮安装在一个音控器中。

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

以下示例展示了将PVC连接到舱体的基本配置：基本配置：

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

您可以使用以下方式监控进度 `kubectl get pod --watch`。

2. 确认卷已挂载到 `/my/mount/path`。

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

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

现在您可以删除该 Pod 了。 Pod 应用将不复存在，但卷将保留。

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

在 **EKS** 集群上配置**Trident EKS** 插件

NetApp Trident简化了 Kubernetes 中Amazon FSx for NetApp ONTAP存储管理，使您的开发人员和管理员能够专注于应用程序部署。 NetApp Trident EKS 插件包含最新的安全补丁和错误修复，并经过 AWS 验证，可与 Amazon EKS 配合使用。 EKS 插件使您能够持续确保您的 Amazon EKS 集群安全稳定，并减少安装、配置和更新插件所需的工作量。

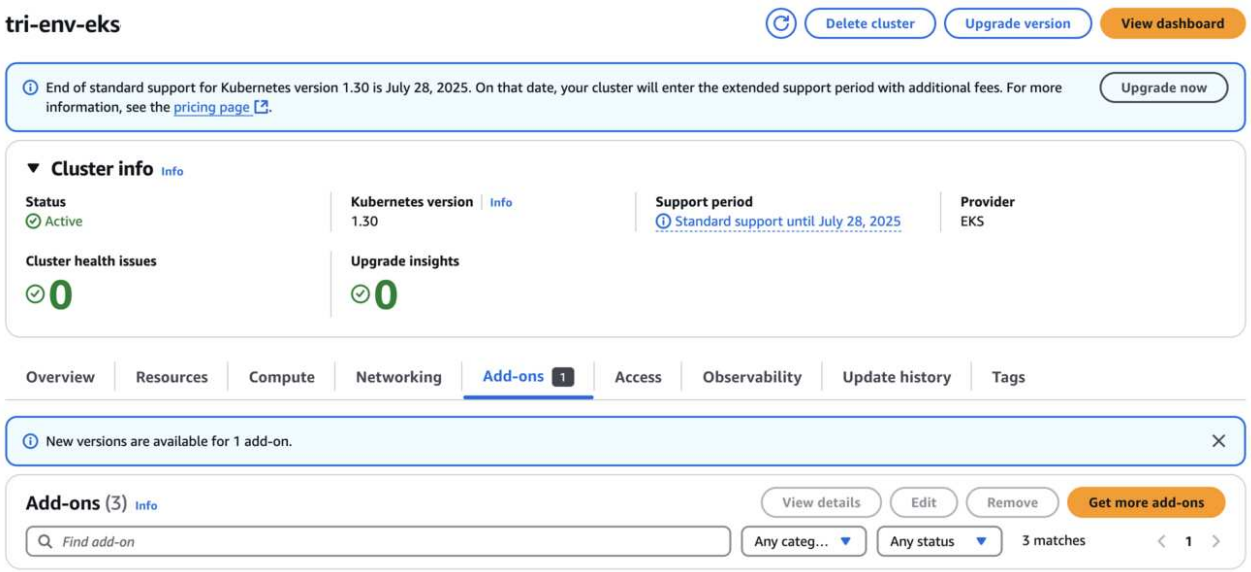
前提条件

在为 AWS EKS 配置Trident插件之前，请确保您已具备以下条件：

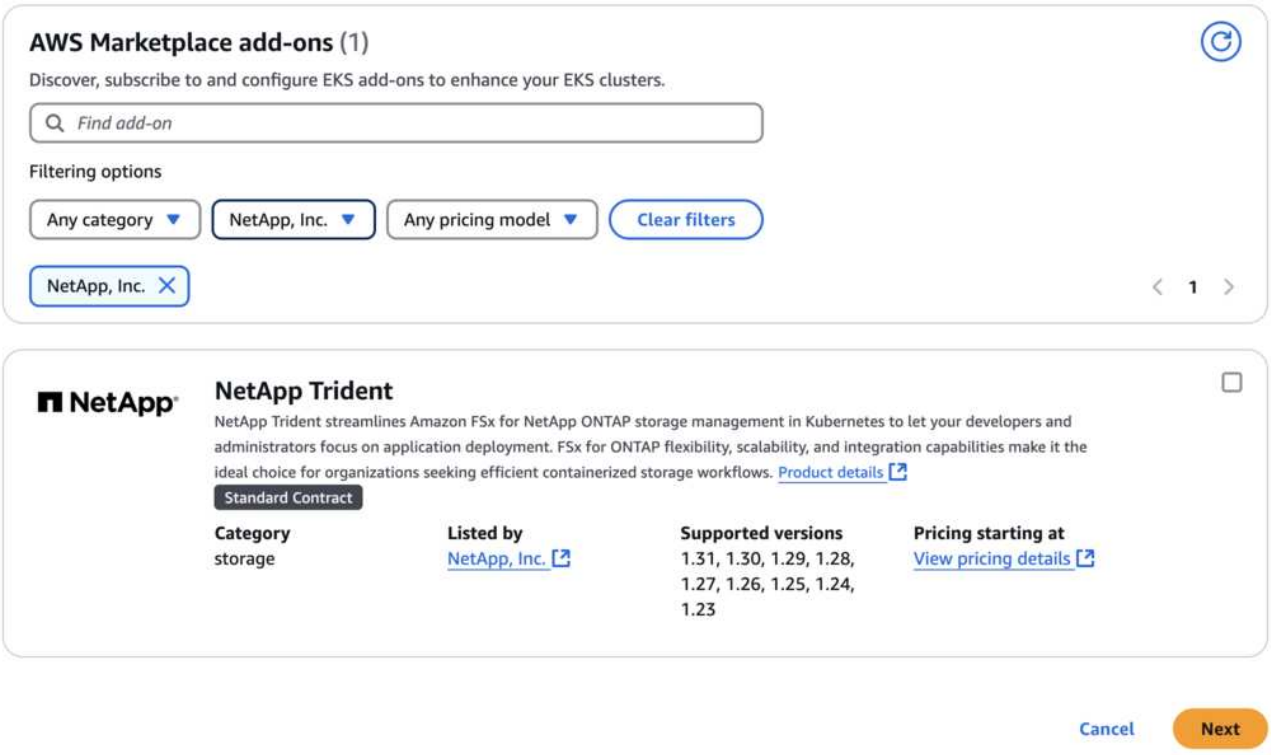
- 具有使用插件权限的 Amazon EKS 集群账户。参考"Amazon EKS 附加组件"。
- AWS 对 AWS Marketplace 的权限：
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI 类型：Amazon Linux 2 (AL2_x86_64) 或 Amazon Linux 2 Arm (AL2_ARM_64)
- 节点类型：AMD 或ARM
- 现有的Amazon FSx for NetApp ONTAP文件系统

步骤

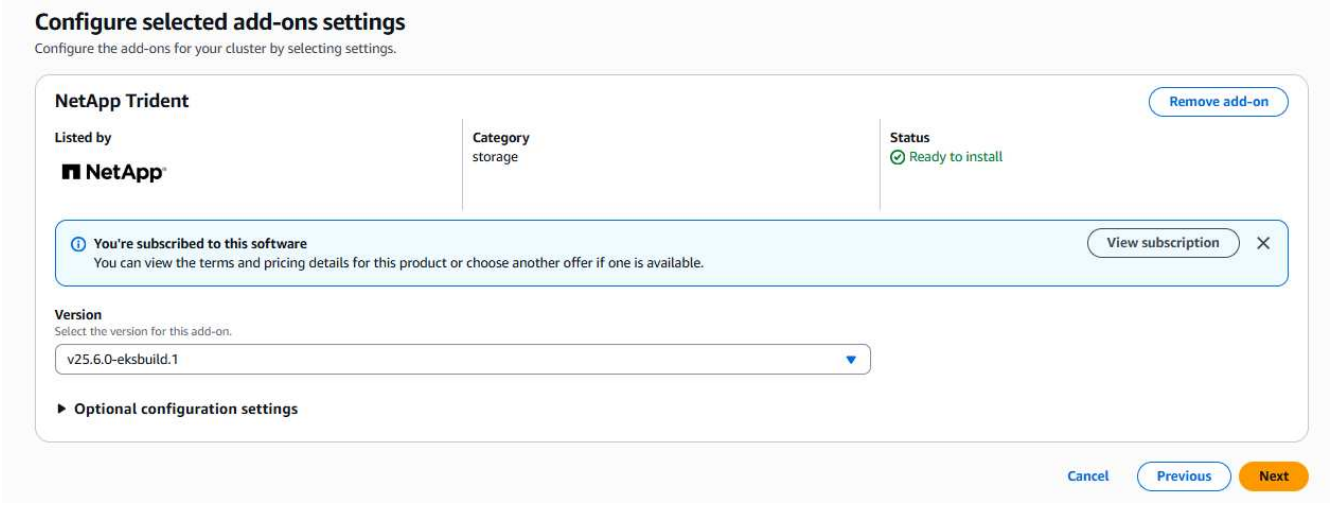
1. 请务必创建 IAM 角色和 AWS 密钥，以使 EKS pod 能够访问 AWS 资源。有关说明，请参阅"创建 IAM 角色和 AWS Secret"。
2. 在 EKS Kubernetes 集群上，导航到“附加组件”选项卡。



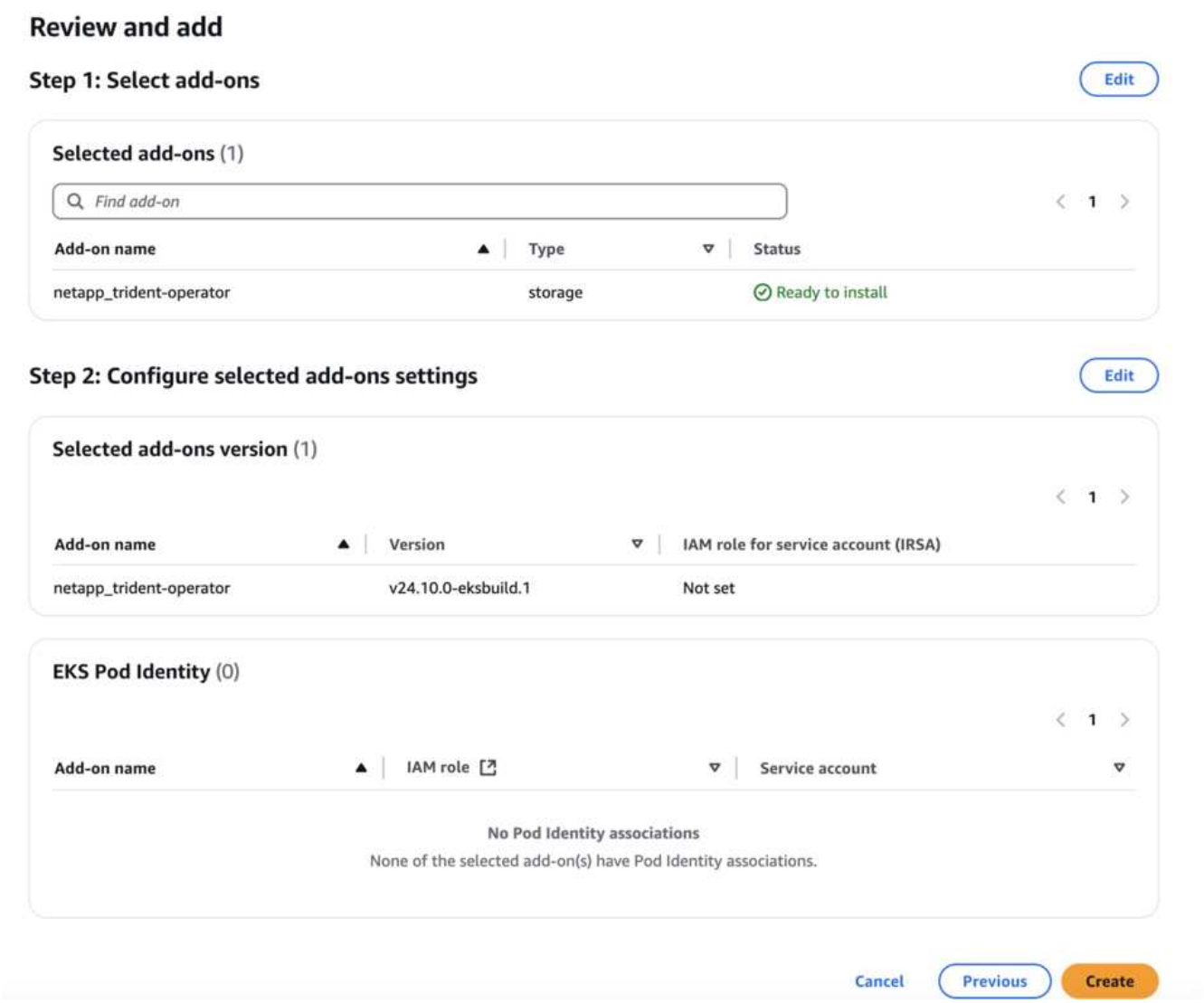
3. 前往 **AWS Marketplace** 插件，然后选择 *storage* 类别。



4. 找到 * NetApp Trident*，选中Trident插件的复选框，然后单击 下一步。
5. 选择所需的插件版本。

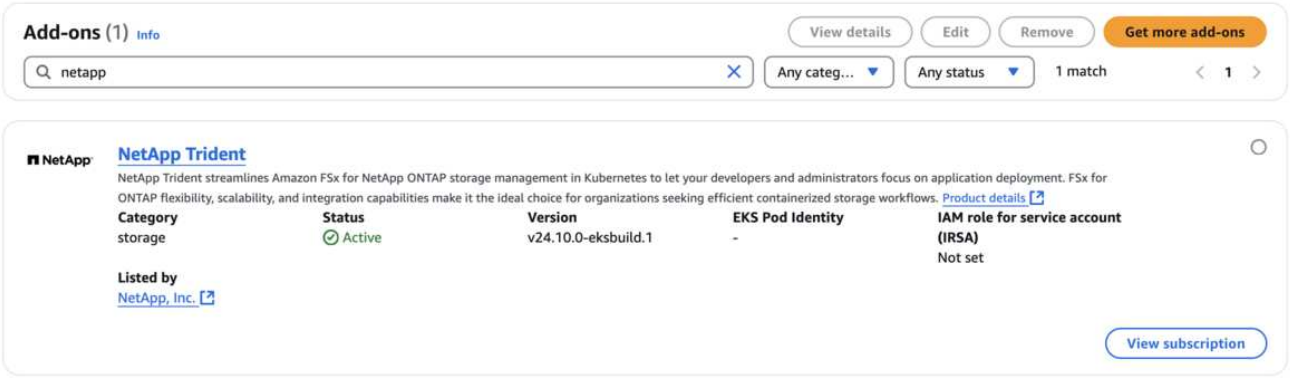


6. 配置所需的附加组件设置。



7. 如果您使用的是 IRSA（服务帐户的 IAM 角色），请参阅其他配置步骤。"此处" 。
8. 选择“创建”。

9. 确认插件状态为_Active_。



10. 运行以下命令以验证Trident是否已正确安装在集群上：

```
kubectl get pods -n trident
```

11. 继续进行设置并配置存储后端。有关信息，请参阅"配置存储后端"。

使用 **CLI** 安装/卸载**Trident EKS** 插件

使用 **CLI** 安装**NetApp Trident EKS** 插件：

以下示例命令安装Trident EKS 插件：

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v25.6.0-eksbuild.1 （另有专用版本）
```

使用 **CLI** 卸载**NetApp Trident EKS** 插件：

以下命令卸载Trident EKS 插件：

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

使用 **kubectl** 创建后端

后端定义了Trident与存储系统之间的关系。它告诉Trident如何与该存储系统通信，以及Trident应该如何从中配置卷。 Trident安装完成后，下一步是创建后端。这 `TridentBackendConfig`自定义资源定义 (CRD) 使您能够通过 Kubernetes 界面直接创建和管理Trident后端。您可以使用 `kubectl`或者适用于您的 Kubernetes 发行版的等效 CLI 工具。

`TridentBackendConfig`

`TridentBackendConfig`(`tbc`， `tbconfig`， `tbackendconfig`)是一个前端、命名空间 CRD，使您能够使用 来管理Trident后端 `kubectl`。现在，Kubernetes 和存储管理员可以直接通过 Kubernetes CLI 创建和管理后端，而无需使用专门的命令行实用程序。(`tridentctl`）。

在创建之后 `TridentBackendConfig`对于该对象，会发生以下情况：

- Trident会根据您提供的配置自动创建后端。这在内部表示为 `TridentBackend`（tbe， tridentbackend ）CR。
- 这 `TridentBackendConfig`与……有着独特的联系 `TridentBackend`这是由Trident生产的。

每个 `TridentBackendConfig`与……保持一对一映射关系 `TridentBackend`前者是提供给用户设计和配置后端的界面；后者是Trident表示实际后端对象的方式。

> `TridentBackend`CR 由Trident自动创建。你*不应该*修改它们。如果您想对后端进行更新，请通过修改以下文件来实现： `TridentBackendConfig`目的。

以下示例展示了格式： `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
```

您还可以查看以下示例： "trident-installer"包含所需存储平台/服务的示例配置的目录。

这 `spec`接受后端特定的配置参数。在这个例子中，后端使用了 `ontap-san`存储驱动程序，并使用此处表格中列出的配置参数。有关所需存储驱动程序的配置选项列表，请参阅"存储驱动程序的后端配置信息"。

这 `spec`本节还包括 `credentials`和 `deletionPolicy`这些字段是新引入的 `TridentBackendConfig`CR：

- `credentials`此参数为必填字段，包含用于向存储系统/服务进行身份验证的凭据。这设置为用户创建的 Kubernetes Secret。凭据不能以明文形式传递，否则将导致错误。
- `deletionPolicy`此字段定义了当……时应该发生的情况 `TridentBackendConfig`被删除。它可以取以下两个值之一：
  - `delete`这会导致两者被删除。 `TridentBackendConfig` CR及其相关后端。这是默认值。
  - `retain`当 `TridentBackendConfig`CR 被删除后，后端定义仍然存在，并且可以通过以下方式进行管理： `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），并保留已创建的后端。该字段的值可以在之后更新 `TridentBackendConfig`已创建。

> 后端名称是通过以下方式设置的： `spec.backendName` 。如果未指定，则后端名称设置为……的名称 `TridentBackendConfig`对象（`metadata.name`）。建议使用显式方式设置后端名称。 `spec.backendName` 。

用以下方式创建的后端 `tridentctl`没有关联的 `TridentBackendConfig`目的。您可以选择使用以下方式管理此类后端 `kubectl`通过创建一个 `TridentBackendConfig`CR。必须注意指定完全相同的配置参数（例如： `spec.backendName`， `spec.storagePrefix`， `spec.storageDriverName`， 等等）。 Trident将自动绑定新创建的`TridentBackendConfig`利用现有的后端。

步骤概述

使用以下方式创建新的后端 `kubectl`你应该这样做：

1. 创建一个 "Kubernetes Secret"该密钥包含Trident与存储集群/服务通信所需的凭据。
2. 创建一个 `TridentBackendConfig`目的。这包含有关存储集群/服务的具体信息，并引用上一步中创建的密钥。

创建后端后，您可以使用以下方式观察其状态 `kubectl get tbc <tbc-name> -n <trident-namespace>`并收集更多细节信息。

步骤 **1**：创建 **Kubernetes Secret**

创建一个包含后端访问凭据的密钥。这是每个存储服务/平台独有的。以下是一个例子：

```
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: password
```

下表总结了每个存储平台密钥中必须包含的字段：

存储平台密钥字段描述	秘密	字段描述
Azure NetApp Files	客户端ID	应用注册中的客户端 ID
Cloud Volumes Service for GCP	私钥 ID	私钥的ID。具有 CVS 管理员角色的 GCP 服务帐户的部分 API 密钥
Cloud Volumes Service for GCP	私钥	私钥。具有 CVS 管理员角色的 GCP 服务帐户的部分 API 密钥

存储平台密钥字段描述	秘密	字段描述
元素（NetApp HCI/ SolidFire）	端点	针对SolidFire集群的 MVIP，包含租户凭证
ONTAP	用户名	用于连接到集群/SVM 的用户名。用于基于凭证的身份验证
ONTAP	password	连接到集群/SVM 的密码。用于基于凭证的身份验证
ONTAP	客户端私钥	客户端私钥的 Base64 编码值。用于基于证书的身份验证
ONTAP	chap用户名	入站用户名。如果 useCHAP=true，则为必填项。为了 `ontap-san`和 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 发起者密钥。如果 useCHAP=true，则此项为必填项。为了 `ontap-san`和 `ontap-san-economy`
ONTAP	chapTargetUsername	目标用户名。如果 useCHAP=true，则此项为必填项。为了 `ontap-san`和 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 目标发起者密钥。如果 useCHAP=true，则此项为必填项。为了 `ontap-san`和 `ontap-san-economy`

此步骤中创建的秘密将在以下位置引用： `spec.credentials`领域的 `TridentBackendConfig`在下一步创建的对象。

步骤 **2**：创建 **`TridentBackendConfig`CR**

您现在可以开始创建您的 `TridentBackendConfig`CR。在这个例子中，后端使用了 `ontap-san`驱动程序是通过使用以下方式创建的： `TridentBackendConfig`下图所示物体：

```
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
```

步骤 3：验证状态 `TridentBackendConfig`CR

现在你已经创建了 `TridentBackendConfig`CR，您可以核实状态。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
```

后端已成功创建并绑定到 `TridentBackendConfig`CR。

相位可以取以下值之一：

- `Bound`： 这 `TridentBackendConfig`CR 与后端关联，该后端包含 `configRef`设置为 `TridentBackendConfig`CR 的 uid。
- `Unbound`：用以下方式表示 ""。这 `TridentBackendConfig`该对象未绑定到后端。所有新创建 `TridentBackendConfig`CR 默认处于此阶段。阶段变化后，它无法再恢复到未绑定状态。
- `Deleting`： 这 `TridentBackendConfig`CR的 `deletionPolicy`已设置为删除。当 `TridentBackendConfig`CR 被删除后，状态变为正在删除。
  - 如果后端不存在持久卷声明 (PVC)，则删除 `TridentBackendConfig`这将导致Trident删除后端以及 `TridentBackendConfig`CR。
  - 如果后端存在一个或多个 PVC，则进入删除状态。这 `TridentBackendConfig`CR随后也进入删除阶段。后端和 `TridentBackendConfig`只有在所有 PVC 都被删除后才会删除。
- `Lost`与后端相关的 `TridentBackendConfig`CR 被意外或故意删除，并且 `TridentBackendConfig`CR 仍然保留着对已删除后端的引用。这 `TridentBackendConfig`无论如何，CR 仍然可以被删除。 `deletionPolicy`价值。
- `Unknown`Trident无法确定与以下系统关联的后端的状态或是否存在： `TridentBackendConfig` CR。例如，如果 API 服务器没有响应，或者如果 `tridentbackends.trident.netapp.io`CRD 缺失。这可能需要干预。

至此，后端已成功创建！此外，还可以处理以下几种操作："后端更新和后端删除"。

（可选）步骤 **4**：了解更多详情

您可以运行以下命令来获取有关后端的更多信息：

```
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 YAML/JSON 转储文件。`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`以及 `backendUUID`后端是根据以下情况创建的：`TridentBackendConfig CR。这 lastOperationStatus`该字段表示上次操作的状态`TridentBackendConfig`CR（变更请求）可以由用户触发（例如，用户更改了某些内容）。 `spec）或由Trident触发（例如，在Trident重启期间）。结果要么是成功，要么是失败。 phase`代表了以下关系的状态：`TridentBackendConfig CR 和后端。在上面的例子中， `phase`具有 Bound 值，这意味着`TridentBackendConfig`CR与后端相关。

您可以运行 `kubectl -n trident describe tbc <tbc-cr-name>`获取事件日志详细信息的命令。

> 您无法更新或删除包含关联后端的后端。 TridentBackendConfig`使用对象`tridentctl。要了解在以下两者之间切换所涉及的步骤： tridentctl`和`TridentBackendConfig，"参见此处"。

管理后端

使用 **kubectl** 执行后端管理

了解如何使用以下方式执行后端管理操作 kubectl。

删除后端

通过删除一个 TridentBackendConfig`您指示Trident删除/保留后端（基于 `deletionPolicy）。要删除后端，请确保 deletionPolicy`已设置为删除。仅删除 `TridentBackendConfig`确保 `deletionPolicy`设置为保留。这样可以确保后端仍然存在，并且可以通过以下方式进行管理：`tridentctl 。

运行以下命令：

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

Trident不会删除正在使用的 Kubernetes Secrets。 TridentBackendConfig 。 Kubernetes 用户负责清理密钥。删除机密信息时务必谨慎。只有当后端不再使用密钥时，才应该删除密钥。

查看现有后端

运行以下命令：

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

你也可以运行 tridentctl get backend -n trident`或者 `tridentctl get backend -o yaml -n trident`获取所有现有后端的列表。此列表还将包括使用以下方式创建的后端： `tridentctl 。

更新后端

更新后端的原因可能有很多：

- 存储系统的凭证已更改。要更新凭据，需要更新 Kubernetes Secret，该 Secret 用于：`TridentBackendConfig`对象必须更新。 Trident会自动使用提供的最新凭据更新后端。运行以下命令更新 Kubernetes Secret：

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

- 需要更新参数（例如正在使用的ONTAP SVM 的名称）。
  - 您可以更新 `TridentBackendConfig`使用以下命令直接通过 Kubernetes 访问对象：

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

  - 或者，您可以对现有内容进行更改。 `TridentBackendConfig`使用以下命令进行回车：

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

- 如果后端更新失败，后端将继续保持其最后一次已知的配置。您可以通过运行以下命令查看日志以确定原因。 `kubectl get tbc <tbc-name> -o yaml -n trident`或者 `kubectl describe tbc <tbc-name> -n trident`。
- 在您发现并纠正配置文件中的问题后，您可以重新运行更新命令。

使用 **tridentctl** 执行后端管理

了解如何使用以下方式执行后端管理操作 `tridentctl`。

创建后端

创建之后"后端配置文件"运行以下命令：

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

如果后端创建失败，则说明后端配置存在问题。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs -n trident
```

在您发现并纠正配置文件中的问题后，您只需运行以下命令即可。 `create`再次发出命令。

删除后端

要从Trident中删除后端，请执行以下操作：

1. 获取后端名称：

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

2. 删除后端：

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

如果Trident已从此后端配置了仍然存在的卷和快照，则删除后端将阻止从该后端配置新卷。后端将继续处于“删除”状态。

查看现有后端

要查看Trident已知的后端，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，则说明后端配置有问题，或者您尝试了无效的更新。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs -n trident
```

在您发现并纠正配置文件中的问题后，您只需运行以下命令即可。 `update`再次发出命令。

确定使用后端存储的存储类

这是一个您可以使用 JSON 回答的问题示例： `tridentctl`后端对象的输出。这使用 `jq`您需要安装该实用程序。

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

这也适用于使用以下方式创建的后端： `TridentBackendConfig` 。

在后端管理选项之间切换

了解Trident中管理后端的不同方法。

后端管理选项

随着 `TridentBackendConfig`现在，管理员有两种独特的后端管理方式。这就引出了以下问题：

- 可以使用以下方式创建后端 `tridentctl`**以……进行管理** `TridentBackendConfig`?
- 可以使用以下方式创建后端 `TridentBackendConfig`**可通过以下方式进行管理** `tridentctl`?

管理 tridentctl`使用后端 `TridentBackendConfig

本节介绍管理使用以下方式创建的后端所需的步骤： `tridentctl`直接通过 Kubernetes 接口创建 `TridentBackendConfig`物体。

这适用于以下情况：

- 预先存在的后端，没有 TridentBackendConfig`因为它们是用……创造的 `tridentctl。
- 使用以下方式创建的新后端 `tridentctl`而其他 `TridentBackendConfig`物体是存在的。

在这两种情况下，后端都将继续存在， Trident将调度卷并对其进行操作。管理员此时有两种选择：

- 继续使用 `tridentctl`管理使用它创建的后端。
- 使用以下方式创建的绑定后端 tridentctl`到一个新的 `TridentBackendConfig`目的。这样做意味着后端将使用以下方式进行管理： `kubectl`而不是 `tridentctl。

使用以下方式管理预先存在的后端 `kubectl`您需要创建一个 `TridentBackendConfig`它与现有后端绑定。以下是其工作原理概述：

1. 创建 Kubernetes Secret。该密钥包含Trident与存储集群/服务通信所需的凭据。
2. 创建一个 TridentBackendConfig`目的。这包含有关存储集群/服务的具体信息，并引用上一步中创建的密钥。必须注意指定完全相同的配置参数（例如： `spec.backendName， spec.storagePrefix， spec.storageDriverName， 等等）。 `spec.backendName`必须设置为现有后端的名称。

步骤 0：确定后端

创建一个 `TridentBackendConfig`如果要绑定到现有后端，则需要获取后端配置。在这个例子中，我们假设使用以下 JSON 定义创建了一个后端：

```
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"
      }
    }
  ]
}
```

步骤 1：创建 **Kubernetes Secret**

创建一个包含后端凭据的 Secret，如下例所示：

```
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
```

步骤 2：创建 **`TridentBackendConfig`CR**

下一步是创建一个 `TridentBackendConfig`CR` 将自动绑定到预先存在的 `ontap-nas-backend（如本例所示）。请确保满足以下要求：

- 后端名称在以下位置定义： `spec.backendName` 。
- 配置参数与原后端相同。
- 虚拟池（如果存在）必须保持与原始后端相同的顺序。
- 凭证通过 Kubernetes Secret 提供，而不是以明文形式提供。

在这种情况下， `TridentBackendConfig`将会像这样：

```
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
```

步骤 3：验证状态 `TridentBackendConfig`CR

之后 `TridentBackendConfig`已经创建，它的阶段必须是 `Bound`。它还应该反映与现有后端相同的后端名称和 UUID。

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

后端现在将完全使用以下方式进行管理： `tbc-ontap-nas-backend` `TridentBackendConfig`目的。

管理 `TridentBackendConfig`使用后端 `tridentctl`

`tridentctl`可用于列出使用以下方式创建的后端： `TridentBackendConfig` 。此外，管理员还可以选择通过以下方式完全管理此类后端： `tridentctl`通过删除 `TridentBackendConfig`并确保 `spec.deletionPolicy`设置为 `retain`。

步骤 0：确定后端

例如，假设我们使用以下方式创建了以下后端 `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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

从输出结果可以看出： `TridentBackendConfig`已成功创建并绑定到后端[观察后端的 UUID]。

步骤 **1**：确认 `deletionPolicy`设置为 `retain`

让我们来看看它的价值 `deletionPolicy`。需要将其设置为 `retain`。这确保了当 `TridentBackendConfig`CR 被删除后，后端定义仍然存在，并且可以通过以下方式进行管理： `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
```

除非另有说明，否则请勿进行下一步。 `deletionPolicy`设置为 `retain。

步骤二：删除 `TridentBackendConfig`CR

最后一步是删除 `TridentBackendConfig`CR。确认后 `deletionPolicy`设置为 `retain`您可以继续删除：

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

删除后 `TridentBackendConfig`Trident只是移除该对象，而不会实际删除后端本身。

创建和管理存储类

创建存储类

配置 Kubernetes StorageClass 对象并创建存储类，以指示Trident如何配置卷。

配置 **Kubernetes StorageClass** 对象

这 "Kubernetes StorageClass 对象"确定Trident是该类使用的配置器，并指示Trident如何配置卷。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

请参阅"Kubernetes 和Trident对象"有关存储类如何与……交互的详细信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的参数。

创建存储类

创建 StorageClass 对象后，即可创建存储类。[存储类示例]提供一些您可以使用或修改的基本示例。

步骤

1. 这是一个 Kubernetes 对象，所以请使用 `kubectl`在 Kubernetes 中创建它。

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. 现在您应该在 Kubernetes 和Trident中都看到 **basic-csi** 存储类，并且Trident应该已经发现了后端上的存储池。

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

存储类示例

Trident提供 "针对特定后端的简单存储类定义"。

或者，您可以编辑 `sample-input/storage-class-csi.yaml.templ`安装程序附带的文件并替换 `*BACKEND_TYPE*` 使用存储驱动程序名称。

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

管理存储类

您可以查看现有存储类、设置默认存储类、识别存储类后端以及删除存储类。

查看现有存储类

- 要查看现有的 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 Trident 的同步存储类，请运行以下命令：

```
tridentctl get storageclass
```

- 要查看 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。如果用户未在持久卷声明 (PVC) 中指定持久卷，则将使用此存储类来配置持久卷。

- 通过设置注解来定义默认存储类 `storageclass.kubernetes.io/is-default-class`在存储类定义中设置为 true。根据规范，任何其他值或缺少注释均被解释为错误。
- 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，您可以使用以下命令移除默认的存储类注解：

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

Trident安装程序包中也有一些包含此注释的示例。

集群中一次只能存在一个默认存储类。从技术上讲，Kubernetes 并不阻止你拥有多个默认存储类，但它的行为就好像根本没有默认存储类一样。

确定存储类的后端

这是一个您可以使用 JSON 回答的问题示例： `tridentctl` Trident后端对象的输出。这使用 `jq`您可能需要先安装该实用程序。

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

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>`应该替换成你的存储类。

通过此存储类别创建的任何持久卷都将保持不变， Trident将继续管理它们。

Trident强制执行空白 `fsType`因为它创造了大量的销量。对于 iSCSI 后端，建议强制执行 `parameters.fsType`在存储类中。您应该删除现有的 StorageClasses 并重新创建它们。 `parameters.fsType`指定的。

配置和管理卷

提供一定量

创建一个使用已配置的 Kubernetes StorageClass 的 PersistentVolumeClaim (PVC) 来请求访问 PV。然后您可以将光伏组件安装到支架上。

概述

一个 "*PersistentVolumeClaim*"（PVC）是对集群上持久卷的访问请求。

PVC 可以配置为请求存储特定尺寸或访问模式。使用关联的 StorageClass，集群管理员可以控制的不仅仅是 PersistentVolume 的大小和访问模式，还可以控制性能或服务级别。

制作好PVC管后，就可以将管体安装到管座中。

制作**PVC**管

步骤

1. 创建 PVC。

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

2. 核实PVC状态。

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage Bound    pv-name    1Gi        RWO                    5m
```

1. 将音量旋钮安装在一个音控器中。

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

> 您可以使用以下方式监控进度 `kubectl get pod --watch`。

2. 确认卷已挂载到 `/my/mount/path`。

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

3. 现在您可以删除该 Pod 了。 Pod 应用将不复存在，但卷将保留。

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

样品清单

PersistentVolumeClaim 样本清单

这些示例展示了PVC的基本配置选项。

PVC管材，带RWO通道

此示例展示了一个具有 RWO 访问权限的基本 PVC，它与一个名为 StorageClass 的 StorageClass 相关联。 `basic-csi` 。

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

PVC 和 NVMe/TCP

此示例展示了一个与名为 StorageClass 的 StorageClass 关联的、具有 RWO 访问权限的 NVMe/TCP 基本 PVC。 `protection-gold` 。

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

Pod清单示例

这些示例展示了将 PVC 连接到舱体的基本配置。

基本配置

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

基本 **NVMe/TCP** 配置

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

请参阅"Kubernetes 和Trident对象"有关存储类如何与……交互的详细信息 `PersistentVolumeClaim`以及控制Trident如何分配容量的参数。

扩大规模

Trident为 Kubernetes 用户提供了在创建卷后扩展卷的能力。查找有关扩展 iSCSI、NFS、SMB、NVMe/TCP 和 FC 卷所需的配置的信息。

扩展 iSCSI 卷

您可以使用 CSI 配置程序扩展 iSCSI 持久卷 (PV)。

> iSCSI 卷扩展受以下方式支持： `ontap-san` ， `ontap-san-economy` ， `solidfire-san`驱动程序需要 Kubernetes 1.16 及更高版本。

步骤 1：配置 StorageClass 以支持卷扩展

编辑 StorageClass 定义以进行设置 `allowVolumeExpansion`田野 `true`。

```
cat storageclass-ontapsan.yaml
```

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

对于已存在的 StorageClass，对其进行编辑以包含以下内容： `allowVolumeExpansion`范围。

步骤 2：使用您创建的 StorageClass 创建 PVC。

编辑PVC定义并更新 `spec.resources.requests.storage`为了反映新的所需尺寸，该尺寸必须大于原始尺寸。

```
cat pvc-ontapsan.yaml
```

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

Trident创建一个持久销量 (PV) 并将其与此持久销量声明 (PVC) 关联起来。

```
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
```

步骤 **3**：定义一个连接 **PVC** 的舱体

将光伏组件连接到舱体上，以便调整其尺寸。调整 iSCSI PV 大小时有两种情况：

- 如果 PV 连接到 pod， Trident会扩展存储后端上的卷，重新扫描设备，并调整文件系统的大小。
- 尝试调整未连接的 PV 的大小时， Trident会在存储后端扩展卷。 PVC 与 pod 绑定后， Trident会重新扫描设备并调整文件系统的大小。 Kubernetes 会在扩展操作成功完成后更新 PVC 大小。

在这个例子中，创建了一个使用以下功能的 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
```

步骤 **4**：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
 # ...
```

步骤 **5**：验证扩展

您可以通过检查PVC、PV和Trident的体积来验证扩容是否正确：

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

扩大 **FC** 体积

您可以使用 CSI 配置程序扩展 FC 持久卷 (PV)。

FC体积扩张得到了以下方面的支持： `ontap-san`驱动程序需要 Kubernetes 1.16 及更高版本。

步骤 **1**：配置 **StorageClass** 以支持卷扩展

编辑 StorageClass 定义以进行设置 `allowVolumeExpansion`田野 `true`。

```
cat storageclass-ontapsan.yaml
```

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

对于已存在的 StorageClass，对其进行编辑以包含以下内容： `allowVolumeExpansion`范围。

步骤 2：使用您创建的 **StorageClass** 创建 **PVC**。

编辑PVC定义并更新 `spec.resources.requests.storage`为了反映新的所需尺寸，该尺寸必须大于原始尺寸。

```
cat pvc-ontapsan.yaml
```

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

Trident创建一个持久销量 (PV) 并将其与此持久销量声明 (PVC) 关联起来。

```
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
```

步骤 3：定义一个连接 **PVC** 的舱体

将光伏组件连接到舱体上，以便调整其尺寸。调整燃料电池光伏组件容量时有两种情况：

- 如果 PV 连接到 pod， Trident会扩展存储后端上的卷，重新扫描设备，并调整文件系统的大小。
- 尝试调整未连接的 PV 的大小时， Trident会在存储后端扩展卷。 PVC 与 pod 绑定后， Trident会重新扫描设备并调整文件系统的大小。 Kubernetes 会在扩展操作成功完成后更新 PVC 大小。

在这个例子中，创建了一个使用以下功能的 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
```

步骤 **4**：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
 # ...
```

步骤 **5**：验证扩展

您可以通过检查PVC、PV和Trident的体积来验证扩容是否正确：

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

扩展 NFS 卷

Trident支持为已配置的 NFS PV 扩展容量。 `ontap-nas` ， `ontap-nas-economy` ， `ontap-nas-flexgroup` ， `gcp-cvs` ， 和 `azure-netapp-files`后端。

步骤 1：配置 StorageClass 以支持卷扩展

要调整 NFS PV 的大小，管理员首先需要配置存储类以允许卷扩展，方法是设置以下参数：`allowVolumeExpansion`田野 `true`：

```
cat storageclass-ontapnas.yaml
```

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

如果您已经创建了一个没有此选项的存储类，则可以通过使用以下命令直接编辑现有存储类： `kubectl edit storageclass`允许体积膨胀。

步骤 2：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
```

Trident应该为该 PVC 创建一个 20 MiB 的 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
```

步骤 3：展开 **PV**

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

```
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
# ...
```

步骤 **4**：验证扩展

您可以通过检查 PVC、PV 和Trident体积的大小来验证调整大小是否正确：

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

进口量

您可以使用以下方式将现有存储卷导入为 Kubernetes PV： `tridentctl import` 。

概述和注意事项

您可以将卷导入Trident以执行以下操作：

- 将应用程序容器化并重用其现有数据集
- 为临时应用程序使用数据集的克隆版本
- 重建失败的 Kubernetes 集群
- 在灾难恢复期间迁移应用程序数据

注意事项

导入卷之前，请考虑以下事项。

- Trident只能导入 RW（读写）类型的ONTAP卷。 DP（数据保护）类型卷是SnapMirror目标卷。在将卷导入Trident之前，应该先断开镜像关系。

- 我们建议导入没有活动连接的卷。要导入正在使用的卷，请克隆该卷，然后执行导入操作。

这对于块卷来说尤其重要，因为 Kubernetes 不知道之前的连接，很容易将活动卷附加到 pod 上。这可能导致数据损坏。

- 尽管 `StorageClass`必须在 PVC 上指定， Trident在导入过程中不使用此参数。在创建卷时，会使用存储类根据存储特性从可用存储池中进行选择。由于卷已存在，因此导入时无需选择存储池。因此，即使卷存在于与 PVC 中指定的存储类不匹配的后端或池中，导入也不会失败。
- 现有容积尺寸在PVC中确定和设定。卷被存储驱动程序导入后，PV 会创建，并带有指向 PVC 的 ClaimRef。
  - 回收策略初始设置为 `retain`在PV中。 Kubernetes 成功绑定 PVC 和 PV 后，回收策略会更新为与存储类的回收策略相匹配。
  - 如果存储类的回收策略是 `delete`当 PV 被删除时，存储卷也会被删除。
- 默认情况下， Trident管理 PVC，并在后端重命名FlexVol volume和 LUN。你可以通过 `--no-manage`导入非托管卷的标志。如果你使用 `--no-manage`在对象的生命周期内， Trident不会对 PVC 或 PV 执行任何额外的操作。删除 PV 时，存储卷不会被删除，其他操作（如卷克隆和卷调整大小）也会被忽略。

如果您想使用 Kubernetes 来管理容器化工作负载，但又想在 Kubernetes 之外管理存储卷的生命周期，则此选项非常有用。

- 在 PVC 和 PV 中添加注释，其作用有两个：一是指示卷已导入，二是指示 PVC 和 PV 是否已管理。此注释不应修改或删除。

导入卷

您可以使用 `tridentctl import`导入卷。

步骤

1. 创建持久卷声明 (PVC) 文件（例如， `pvc.yaml` ）将用于制造PVC。 PVC文件应包含 `name`， `namespace` ， `accessModes` ， 和 `storageClassName`。 （可选）您可以指定 `unixPermissions`在你的PVC定义中。

   以下是一个最低规格示例：

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

不要包含其他参数，例如 PV 名称或体积大小。这可能会导致导入命令失败。

2. 使用 `tridentctl import`用于指定包含卷的Trident后端名称以及唯一标识存储上卷的名称的命令（例如：ONTAP FlexVol、Element Volume、 Cloud Volumes Service路径）。这 `-f`需要提供参数来指定PVC文件的路径。

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

示例

请查看以下卷导入示例，了解支持的驱动程序。

ONTAP NAS 和ONTAP NAS FlexGroup

Trident支持使用以下方式导入卷： `ontap-nas`和 `ontap-nas-flexgroup`司机。

- Trident不支持使用 `ontap-nas-economy` 司机。
- 这 `ontap-nas`和 `ontap-nas-flexgroup`驱动程序不允许重复的卷名称。

每卷都是用以下方式创建的 `ontap-nas`driver 是ONTAP集群上的FlexVol volume。使用以下方式导入FlexVol卷 `ontap-nas`驱动程序的工作原理相同。已存在于ONTAP集群上的FlexVol卷可以作为卷导入。 `ontap-nas` PVC。同样， FlexGroup卷也可以导入为 `ontap-nas-flexgroup`PVC。

ONTAP NAS 示例

下面展示了托管卷和非托管卷导入的示例。

托管卷

以下示例导入一个名为“ `managed_volume`在名为 `ontap_nas`：

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

未托管卷

使用时 `--no-manage`理由是， Trident不会重命名卷。

以下示例导入 `unmanaged_volume`在 `ontap_nas`后端：

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

ONTAP SAN

Trident支持使用卷导入 `ontap-san` （iSCSI、NVMe/TCP 和 FC）和 `ontap-san-economy` 司机。

Trident可以导入包含单个 LUN 的ONTAP SAN FlexVol卷。这与 `ontap-san` 驱动程序，它为每个 PVC 创建一个FlexVol volume，并在FlexVol volume内创建一个 LUN。Trident导入FlexVol volume并将其与 PVC 定义关联。Trident可以导入 `ontap-san-economy` 包含多个 LUN 的卷。

ONTAP SAN 示例

下面展示了托管卷和非托管卷导入的示例。

托管卷

对于托管卷， Trident会将FlexVol volume重命名为 `pvc-<uuid>`FlexVol volume中的 LUN 格式 `lun0。

以下示例导入了 `ontap-san-managed`FlexVol volume存在于 `ontap_san_default`后端：

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

未托管卷

以下示例导入 `unmanaged_example_volume`在 `ontap_san`后端：

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

如果将 LUN 映射到与 Kubernetes 节点 IQN 共享同一 IQN 的 igroup，如下例所示，则会收到以下错误： `LUN already mapped to initiator(s) in this group`。您需要移除启动器或取消映射 LUN 才能导入卷。

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

Element

Trident支持使用NetApp Element软件和NetApp HCI卷导入 `solidfire-san`司机。

> Element驱动程序支持重复的卷名称。但是，如果卷名称重复， Trident会返回错误。作为一种变通方法，克隆卷，提供一个唯一的卷名称，然后导入克隆的卷。

元素示例

以下示例导入一个 `element-managed`后端容量 `element_default。

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

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

Google Cloud Platform

Trident支持使用以下方式导入卷： `gcp-cvs`司机。

> 要将由NetApp Cloud Volumes Service支持的卷导入 Google Cloud Platform，请通过卷路径识别该卷。卷路径是卷导出路径中位于以下位置之后的部分： `:/` 。例如，如果导出路径是 `10.0.0.1:/adroit-jolly-swift`体积路径为 `adroit-jolly-swift。

Google Cloud Platform 示例

以下示例导入一个 `gcp-cvs`后端容量 `gcpcvs_YEppr`具有体积路径 `adroit-jolly-swift。

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

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

Azure NetApp Files

Trident支持使用以下方式导入卷： `azure-netapp-files`司机。

> 要导入Azure NetApp Files卷，请通过卷路径识别该卷。卷路径是卷导出路径中位于以下位置之后的部分： `:/` 。例如，如果挂载路径是 `10.0.0.2:/importvol1`体积路径为 `importvol1`。

Azure NetApp Files示例

以下示例导入一个 `azure-netapp-files`后端容量 `azurenetappfiles_40517`体积路径 `importvol1`。

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

Google Cloud NetApp Volumes

Trident支持使用以下方式导入卷： `google-cloud-netapp-volumes`司机。

Google Cloud NetApp Volumes示例

以下示例导入一个 google-cloud-netapp-volumes`后端容量 `backend-tbc-gcnv1`音量 `testvoleasiaeast1。

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

以下示例导入一个 `google-cloud-netapp-volumes`当两个体积存在于同一区域时，体积为：

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+-----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

自定义卷名和标签

使用Trident，您可以为创建的卷分配有意义的名称和标签。这可以帮助您识别卷并将其轻松映射到各自的 Kubernetes 资源（PVC）。您还可以在后端级别定义模板，以创建自定义卷名称和自定义标签；您创建、导入或克隆的任何卷都将遵循这些模板。

开始之前

支持自定义卷名称和标签：

1. 卷创建、导入和克隆操作。
2. 对于 ontap-nas-economy 驱动程序，只有 Qtree 卷的名称符合名称模板。
3. 对于 ontap-san-economy 驱动程序，只有 LUN 名称符合名称模板。

限制

1. 可自定义卷名称仅与ONTAP本地驱动程序兼容。
2. 可自定义的卷名称不适用于现有卷。

可自定义卷名称的关键行为

1. 如果由于名称模板中的语法无效而导致失败，则后端创建将失败。但是，如果模板应用失败，则卷将按照现有的命名约定命名。
2. 当使用后端配置中的名称模板命名卷时，存储前缀不适用。可以直接在模板中添加任何所需的前缀值。

后端配置示例，包含名称模板和标签

可以在根级别和/或池级别定义自定义名称模板。

根级别示例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

池级示例

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

名称模板示例

示例 **1**：

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

示例 **2**：

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

需要考虑的几点

1. 对于卷导入，只有当现有卷具有特定格式的标签时，才会更新标签。例如：`{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`。
2. 对于托管卷导入，卷名称遵循后端定义根级别定义的名称模板。
3. Trident不支持将切片运算符与存储前缀一起使用。
4. 如果模板无法生成唯一的卷名称， Trident将添加一些随机字符来创建唯一的卷名称。
5. 如果 NAS 经济型卷的自定义名称长度超过 64 个字符， Trident将按照现有的命名约定命名这些卷。对于所有其他ONTAP驱动程序，如果卷名称超过名称限制，则卷创建过程将失败。

跨命名空间共享 **NFS** 卷

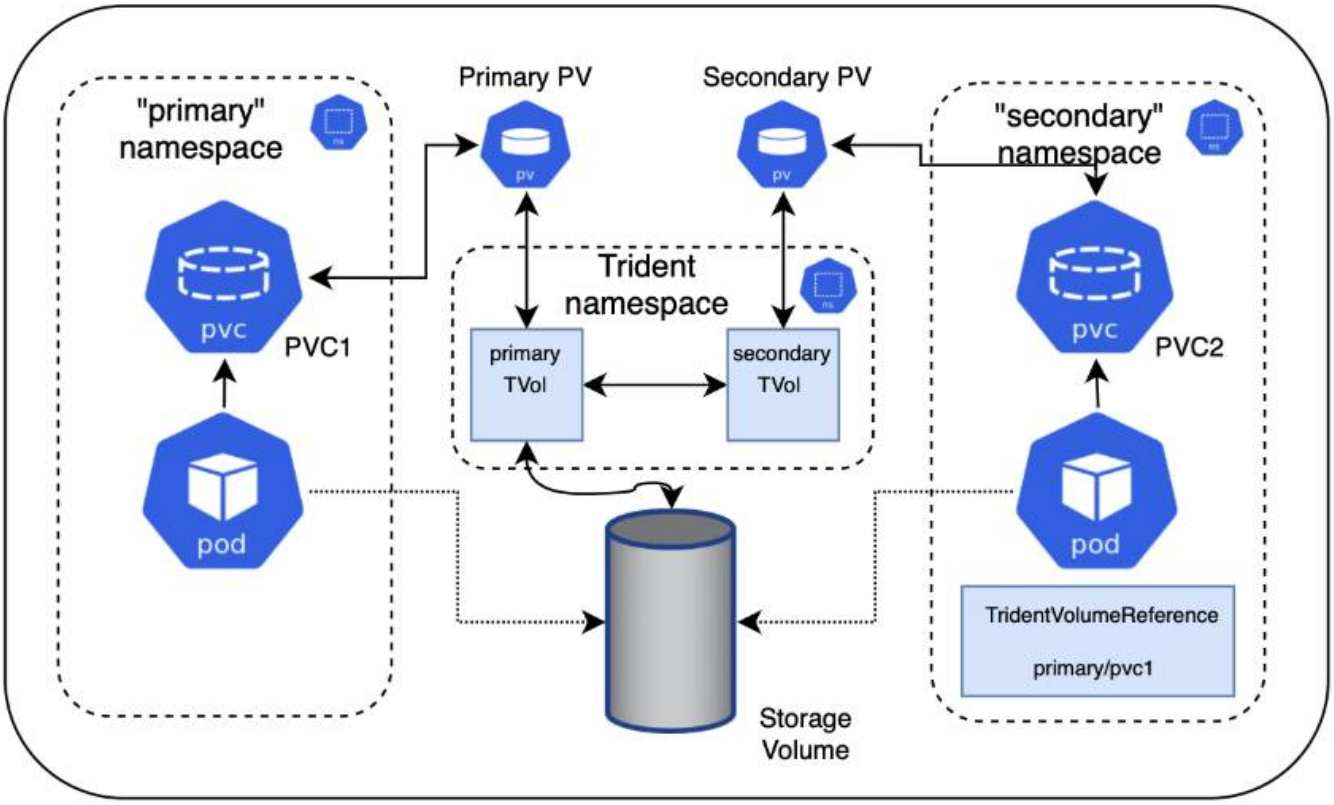
使用Trident，您可以在主命名空间中创建卷，并在一个或多个辅助命名空间中共享该卷。

功能

TridentVolumeReference CR 允许您在一个或多个 Kubernetes 命名空间之间安全地共享 ReadWriteMany (RWX) NFS 卷。这种 Kubernetes 原生解决方案具有以下优点：

- 多级访问控制以确保安全性
- 适用于所有Trident NFS 卷驱动程序
- 不依赖 tridentctl 或任何其他非原生 Kubernetes 功能

此图展示了跨两个 Kubernetes 命名空间的 NFS 卷共享。



快速启动

只需几个步骤即可设置NFS卷共享。

1 配置源**PVC**以共享体积

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

集群管理员授予目标命名空间的所有者创建 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。

配置源命名空间和目标命名空间

为确保安全，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行动。用户角色在每个步骤中都会被指定。

步骤

1. 源命名空间所有者： 创建PVC(`pvc1` ）在源命名空间中，授予与目标命名空间共享的权限(`namespace2`) 使用 `shareToNamespace`注解。

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

Trident创建 PV 及其后端 NFS 存储卷。

- 您可以使用逗号分隔的列表将 PVC 共享给多个命名空间。例如，`trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4` 。
- 您可以使用以下方式共享到所有命名空间 `*`。例如，`trident.netapp.io/shareToNamespace: *`
- 您可以更新PVC以包含 `shareToNamespace`随时可以添加注释。

2. *集群管理员：*确保已建立适当的 RBAC，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一个指向源命名空间的 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` ）在目标命名空间(`namespace2`) 使用 `shareFromPVC`注释以指定来源 PVC。

```
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管的尺寸必须小于或等于源PVC管的尺寸。

结果

Trident读取 `shareFromPVC`在目标 PVC 上添加注释，并将目标 PV 创建为没有自身存储资源的从属卷，该卷指向源 PV 并共享源 PV 存储资源。目的地 PVC 和 PV 看起来已正常绑定。

删除共享卷

您可以删除跨多个命名空间共享的卷。 Trident将移除对源命名空间中该卷的访问权限，并保留对共享该卷的其他命名空间的访问权限。当所有引用该卷的命名空间都被删除后， Trident会删除该卷。

使用 **`tridentctl get`**查询子卷

使用[`tridentctl`实用程序，您可以运行 `get`获取子卷的命令。更多信息，请参阅链接：../trident-reference/tridentctl.html[`tridentctl`命令和选项]。

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

标志：

- `-h, --help`：有关卷的帮助。
- `--parentOfSubordinate string`：将查询限制在子源卷上。
- `--subordinateOf string`：将查询限制在卷的下级。

限制

- Trident无法阻止目标命名空间写入共享卷。您应该使用文件锁定或其他方法来防止覆盖共享卷数据。
- 您无法通过移除源PVC来撤销对源PVC的访问权限。 `shareToNamespace`或者 `shareFromNamespace`注

释或删除 `TridentVolumeReference`CR。要撤销访问权限，必须删除从属 PVC。

- 从属卷无法进行快照、克隆和镜像操作。

了解更多信息

要了解有关跨命名空间卷访问的更多信息：

- 访问"在命名空间之间共享卷：迎接跨命名空间卷访问"。
- 观看演示"NetAppTV"。

跨命名空间克隆卷

使用Trident，您可以利用同一 Kubernetes 集群中不同命名空间内的现有卷或卷快照创建新卷。

前提条件

克隆卷之前，请确保源后端和目标后端是同一类型，并且具有相同的存储类别。

跨命名空间克隆仅支持以下情况： `ontap-san`和 `ontap-nas`存储驱动程序。不支持只读克隆。

快速启动

只需几个步骤即可设置卷克隆。

1 配置源 **PVC** 以克隆卷

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

集群管理员授予目标命名空间的所有者创建 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创建 TridentVolumeReference CR 以引用源 PVC。

4 在目标命名空间中创建克隆 **PVC**

目标命名空间的所有者创建 PVC 以克隆源命名空间中的 PVC。

配置源命名空间和目标命名空间

为确保安全，跨命名空间克隆卷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。用户角色在每个步骤中都会被指定。

步骤

1. 源命名空间所有者： 创建PVC(`pvc1`）在源命名空间中(`namespace1`)授予与目标命名空间共享的权限(`namespace2`)使用 `cloneToNamespace`注解。

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

Trident创建 PV 及其后端存储卷。

- 您可以使用逗号分隔的列表将 PVC 共享给多个命名空间。例如，`trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4` 。
- 您可以使用以下方式共享到所有命名空间 `*`。例如，`trident.netapp.io/cloneToNamespace: *`
- 您可以更新PVC以包含 `cloneToNamespace`随时可以添加注释。

2. 集群管理员： 确保已配置正确的基于角色的访问控制 (RBAC)，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。(`namespace2` ）。
3. 目标命名空间所有者： 在目标命名空间中创建一个指向源命名空间的 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` ）在目标命名空间(`namespace2`) 使用 `cloneFromPVC`或者 `cloneFromSnapshot`， 和 `cloneFromNamespace`用于指定源PVC的注释。

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

限制

- 对于使用 ontap-nas-economy 驱动程序配置的 PVC，不支持只读克隆。

使用SnapMirror复制卷

Trident支持一个集群上的源卷与对等集群上的目标卷之间的镜像关系，用于复制数据以实现灾难恢复。 您可以使用名为Trident镜像关系 (TMR) 的命名空间自定义资源定义 (CRD) 来执行以下操作：

- 在体积（PVC）之间建立镜像关系
- 移除卷之间的镜像关系
- 打破镜像关系
- 在灾难情况下（故障转移）提升备用卷的可用性
- 在计划内故障转移或迁移期间，实现应用程序在集群间的无损迁移。

复制的前提条件

开始之前，请确保满足以下先决条件：

ONTAP集群

- * Trident *：使用ONTAP作为后端的源 Kubernetes 集群和目标 Kubernetes 集群上必须存在Trident版本 22.10 或更高版本。
- 许可证：使用数据保护包的ONTAP SnapMirror异步许可证必须在源 ONTAP 集群和目标ONTAP集群上启用。请参阅 "ONTAP中的SnapMirror许可概述"了解更多信息。

  从ONTAP 9.10.1 开始，所有许可证均以NetApp许可证文件 (NLF) 的形式交付，这是一个可以启用多种功能的单个文件。请参阅"ONTAP One 附带的许可证"了解更多信息。

仅支持SnapMirror异步保护。

对等

- 集群和 **SVM**： ONTAP存储后端必须相互连接。请参阅 "集群和SVM对等连接概述"了解更多信息。

  确保两个ONTAP集群之间复制关系中使用的 SVM 名称是唯一的。

- * Trident和 SVM*：对等远程 SVM 必须可供目标集群上的Trident使用。

支持的驱动程序

NetApp Trident支持使用NetApp SnapMirror技术进行卷复制，该技术使用以下驱动程序支持的存储类：**`ontap-nas`**：**NFS** `ontap-san`：iSCSI **`ontap-san`**：**FC** `ontap-san`：NVMe/TCP（最低要求ONTAP版本 9.15.1）

ASA r2 系统不支持使用SnapMirror进行卷复制。有关ASA r2 系统的信息，请参阅"了解ASA r2 存储系统"。

制作镜面**PVC**

按照这些步骤，并使用 CRD 示例，在主卷和辅助卷之间创建镜像关系。

步骤

1. 在主 Kubernetes 集群上执行以下步骤：
   a. 使用以下方式创建一个 StorageClass 对象 `trident.netapp.io/replication: true`范围。

      示例

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

   b. 使用先前创建的 StorageClass 创建 PVC。

示例

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

c. 使用本地信息创建镜像关系变更请求。

示例

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

Trident获取卷的内部信息和卷的当前数据保护 (DP) 状态，然后填充 MirrorRelationship 的状态字段。

d. 获取 TridentMirrorRelationship CR 以获取 PVC 的内部名称和 SVM。

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

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

2. 在辅助 Kubernetes 集群上执行以下步骤：

   a. 创建一个 StorageClass，并设置 trident.netapp.io/replication: true 参数。

   示例

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

   b. 创建包含目标和源信息的镜像关系 CR。

   示例

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

Trident将使用配置的关系策略名称（或ONTAP的默认值）创建SnapMirror关系并对其进行初始化。

c. 创建一个 PVC，使用先前创建的 StorageClass 作为辅助存储类（SnapMirror目标）。

示例

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

Trident将检查 TridentMirrorRelationship CRD，如果该关系不存在，则创建卷失败。如果存在此关系，Trident将确保将新的FlexVol volume放置在与 MirrorRelationship 中定义的远程 SVM 对等的 SVM 上。

卷复制状态

Trident镜像关系 (TMR) 是一种 CRD，它表示 PVC 之间复制关系的一端。目标 TMR 具有一个状态，该状态告诉Trident期望的状态是什么。目的地 TMR 具有以下状态：

- 已建立：本地 PVC 是镜像关系的目标量，这是一个新的关系。
- 已推广：本地 PVC 可读写且可安装，目前没有镜像关系。
- 重新建立：本地 PVC 是镜像关系的目标量，并且之前也处于该镜像关系中。
  - 如果目标卷曾经与源卷存在关联，则必须使用重新建立的状态，因为它会覆盖目标卷的内容。
  - 如果卷之前未与源建立关系，则重新建立状态将失败。

在计划外故障切换期间促进辅助**PVC**的运行

在辅助 Kubernetes 集群上执行以下步骤：

- 将 TridentMirrorRelationship 的 *spec.state* 字段更新为 `promoted`。

在计划故障切换期间推广备用**PVC**

在计划故障转移（迁移）期间，执行以下步骤以提升辅助 PVC：

步骤

1. 在主 Kubernetes 集群上，创建 PVC 的快照，并等待快照创建完成。

2. 在主 Kubernetes 集群上，创建 SnapshotInfo CR 以获取内部详细信息。

   示例

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

3. 在辅助 Kubernetes 集群上，将 *TridentMirrorRelationship* CR 的 *spec.state* 字段更新为 *promoted*，并将 *spec.promotedSnapshotHandle* 更新为快照的内部名称。
4. 在辅助 Kubernetes 集群上，确认 TridentMirrorRelationship 的状态（status.state 字段）是否已提升。

故障转移后恢复镜像关系

在恢复镜像关系之前，选择你想作为新主要方的那一方。

步骤

1. 在辅助 Kubernetes 集群上，确保 TridentMirrorRelationship 上的 *spec.remoteVolumeHandle* 字段的值已更新。
2. 在辅助 Kubernetes 集群上，将 TridentMirrorRelationship 的 *spec.mirror* 字段更新为 `reestablished`。

附加手术

Trident支持对主卷和辅助卷执行以下操作：

将主**PVC**复制到新的辅助**PVC**

请确保您已备有主PVC管和备用PVC管。

步骤

1. 从已建立的辅助（目标）集群中删除 PersistentVolumeClaim 和 TridentMirrorRelationship CRD。
2. 从主（源）集群中删除 TridentMirrorRelationship CRD。
3. 在主（源）集群上为要建立的新辅助（目标）PVC 创建一个新的 TridentMirrorRelationship CRD。

调整镜像、主或次级**PVC**的尺寸

PVC 可以像往常一样调整大小，如果数据量超过当前大小， ONTAP将自动扩展任何目标 flevxols。

从 **PVC** 中移除复制

要移除复制，请对当前辅助卷执行以下操作之一：

- 删除辅助 PVC 上的镜像关系。这会破坏复制关系。
- 或者，将 spec.state 字段更新为 *promoted*。

删除一个**PVC**（之前已镜像）

Trident会检查是否存在复制的 PVC，并在尝试删除卷之前释放复制关系。

删除 **TMR**

删除镜像关系一侧的 TMR 会导致剩余的 TMR 在Trident完成删除之前转换为 *promoted* 状态。如果选择删除的 TMR 已处于 *promoted* 状态，则不存在镜像关系，TMR 将被删除， Trident会将本地 PVC 提升为 *ReadWrite*。此删除操作会释放ONTAP中本地卷的SnapMirror元数据。如果将来要将此卷用于镜像关系，则在创建新的镜像关系时，必须使用具有 *established* 卷复制状态的新 TMR。

ONTAP在线时，更新镜像关系

镜像关系建立后可以随时更新。您可以使用 `state: promoted`或者 `state: reestablished`用于更新关系的字段。将目标卷提升为常规读写卷时，可以使用 *promotedSnapshotHandle* 指定要将当前卷还原到的特定快照。

ONTAP离线时更新镜像关系

您可以使用 CRD 执行SnapMirror更新，而无需Trident与ONTAP集群直接连接。请参考以下 TridentActionMirrorUpdate 的示例格式：

示例

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

`status.state`反映 TridentActionMirrorUpdate CRD 的状态。它可以取值 成功、进行中_或_失败。

使用 **CSI** 拓扑

Trident可以利用以下方式选择性地创建卷并将其附加到 Kubernetes 集群中的节点： "CSI 拓扑功能" 。

概述

使用 CSI 拓扑功能，可以根据区域和可用区域将对卷的访问限制到节点子集。如今，云服务提供商允许 Kubernetes 管理员创建基于区域的节点。节点可以位于一个区域内的不同可用区，也可以跨多个区域。为了方便在多区域架构中为工作负载配置卷， Trident使用了 CSI 拓扑。

了解更多关于 CSI 拓扑功能的信息 "此处"。

Kubernetes 提供了两种独特的卷绑定模式：

- 和 `VolumeBindingMode`设置为 `Immediate`Trident创建卷时没有任何拓扑感知。卷绑定和动态配置在创建

PVC 时处理。这是默认设置。 `VolumeBindingMode`适用于不强制执行拓扑约束的集群。持久卷的创建不依赖于请求 pod 的调度要求。

- 和 `VolumeBindingMode`设置为 `WaitForFirstConsumer`PVC 的持久卷的创建和绑定将被延迟，直到使用该 PVC 的 pod 被调度和创建。这样，就可以创建卷来满足拓扑要求所强制执行的调度约束。

> 这 `WaitForFirstConsumer`绑定模式不需要拓扑标签。这可以独立于 CSI 拓扑功能使用。

你需要什么

要使用 CSI 拓扑结构，您需要以下组件：

- 运行中的 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"}
```

- 集群中的节点应带有标签，以体现拓扑感知能力。(`topology.kubernetes.io/region`和`topology.kubernetes.io/zone`）。在安装Trident之前，集群中的节点上*应该存在这些标签*，以便Trident能够感知拓扑结构。

```
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"}]
```

步骤 **1**：创建拓扑感知后端

Trident存储后端可以设计为根据可用区选择性地配置卷。每个后端都可以携带一个可选组件 `supportedTopologies`表示所支持的区域和地区列表的块。对于使用此类后端的 StorageClasses，只有在受支持的区域/区域中调度的应用程序请求时才会创建卷。

以下是一个后端定义示例：

YAML

```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

```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`用于提供每个后端区域和分区的列表。这些区域和分区代表了StorageClass 中可以提供的允许值的列表。对于包含后端提供的区域和可用区子集的存储类，Trident会在后端创建一个卷。

你可以定义 `supportedTopologies`每个存储池也是如此。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
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

在这个例子中， `region`和 `zone`标签代表存储池的位置。 `topology.kubernetes.io/region`和 `topology.kubernetes.io/zone`决定存储池可以从何处使用。

步骤 **2**：定义拓扑感知的存储类。

根据提供给集群中节点的拓扑标签，可以定义 StorageClasses 来包含拓扑信息。这将决定哪些存储池可作为 PVC 请求的候选对象，以及哪些节点子集可以使用Trident提供的卷。

请参见以下示例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata: null
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions: null
  - 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`。使用此 StorageClass 请求的 PVC 只有在 pod 中被引用后才会执行。和， `allowedTopologies`提供要使用的区域和范围。这 `netapp-san-us-east1`StorageClass 在存储上创建 PVC。 `san-backend-us-east1`后端定义如上所述。

步骤 3：制作并使用 PVC

创建 StorageClass 并将其映射到后端后，现在可以创建 PVC。

请参阅示例 `spec`以下：

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

使用此清单创建 PVC 将产生以下结果：

```
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
```

为了让Trident形成体积并将其与 PVC 结合，请使用 PVC 管。请参见以下示例：

```yaml
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 将 pod 调度到存在于以下位置的节点上： `us-east1`在该区域中，选择任何存在的节点。 `us-east1-a`或者 `us-east1-b`区域。

请查看以下输出：

```
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
```

更新后端以包含 `supportedTopologies`

可以更新现有的后端，使其包含以下列表： `supportedTopologies`使用 `tridentctl backend update`。这不会影响已经分配的容量，只会用于后续的PVC。

查找更多信息

- "管理容器资源"
- "节点选择器"
- "亲和力和反亲和力"
- "污点和容忍度"

使用快照

Kubernetes 持久卷 (PV) 的卷快照可以创建卷的某个时间点副本。您可以创建使用Trident创建的卷的快照，导入在Trident之外创建的快照，从现有快照创建新卷，以及从快照恢复卷数据。

概述

卷快照受支持 `ontap-nas`， `ontap-nas-flexgroup` ， `ontap-san` ， `ontap-san-economy` ， `solidfire-san` ， `gcp-cvs` ， `azure-netapp-files` ， 和 `google-cloud-netapp-volumes`司机。

开始之前

要使用快照，您必须拥有外部快照控制器和自定义资源定义 (CRD)。这是 Kubernetes 编排器（例如：Kubeadm、GKE、OpenShift）的职责。

如果您的 Kubernetes 发行版不包含快照控制器和 CRD，请参阅[部署卷快照控制器]。

> 如果在 GKE 环境中创建按需卷快照，则不要创建快照控制器。 GKE 使用内置的隐藏快照控制器。

创建卷快照

步骤

1. 创建一个 `VolumeSnapshotClass`更多信息，请参阅……"卷快照类" 。
   - 这 `driver`指向Trident CSI 驱动程序。
   - `deletionPolicy`可以 `Delete`或者 `Retain`。设置为 `Retain`即使发生以下情况，存储集群上的底层物理快照仍会被保留： `VolumeSnapshot`对象已被删除。

     示例

     ```
     cat snap-sc.yaml
     ```

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

2. 创建现有PVC的快照。

   示例

   - 此示例创建现有 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
     ```

   - 此示例为名为“ `pvc1`快照名称设置为 `pvc1-snap`。 VolumeSnapshot 类似于 PVC，并且与某个 PVC 相关联。 `VolumeSnapshotContent`代表实际快照的对象。

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

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

- 你可以识别出 `VolumeSnapshotContent`对象为 `pvc1-snap`通过描述来获取卷快照。这 `Snapshot Content Name`标识提供此快照的 VolumeSnapshotContent 对象。这 `Ready To Use`该参数表明快照可用于创建新的 PVC。

```
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
...
```

从卷快照创建**PVC**

您可以使用 `dataSource`使用名为 VolumeSnapshot 的 VolumeSnapshot 创建 PVC `<pvc-name>`作为数据来源。 PVC管制作完成后，可以将其连接到舱体上，并像其他PVC管一样使用。

> PVC 将在与源卷相同的后端创建。参考"知识库：无法在备用后端从Trident PVC 快照创建 PVC。"。

以下示例使用以下方法创建 PVC： `pvc1-snap`作为数据源。

```
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
```

导入卷快照

Trident支持"Kubernetes 预配置快照流程"使集群管理员能够创建 `VolumeSnapshotContent`在Trident之外创建的对象和导入快照。

开始之前

Trident肯定创建或导入了快照的父卷。

步骤

1. 集群管理员： 创建一个 `VolumeSnapshotContent`引用后端快照的对象。这将启动Trident中的快照工作流程。
   - 指定后端快照的名称 `annotations`作为 `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`。
   - 指定 `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>`在 `snapshotHandle`这是外部快照程序提供给Trident的唯一信息。 `ListSnapshots`称呼。

     这 `<volumeSnapshotContentName>`由于 CR 命名限制，有时无法与后端快照名称完全匹配。

     示例

     以下示例创建了一个 `VolumeSnapshotContent`引用后端快照的对象 `snap-01`。

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

2. 集群管理员： 创建 `VolumeSnapshot`引用 CR `VolumeSnapshotContent`目的。这是对使用权限的请求 `VolumeSnapshot`在给定的命名空间中。

   示例

   以下示例创建了一个 `VolumeSnapshot`CR命名 `import-snap`指的是 `VolumeSnapshotContent` 命名 `import-snap-content。

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

3. 内部处理（无需操作）： 外部快照程序识别新创建的内容 `VolumeSnapshotContent`并运行 `ListSnapshots`称呼。`Trident`创造了 `TridentSnapshot。

   - 外部快照程序设置 `VolumeSnapshotContent`到 `readyToUse`以及 `VolumeSnapshot`到 `true。
   - Trident回归 `readyToUse=true。

4. 任何用户： 创建一个 `PersistentVolumeClaim`参考新的 `VolumeSnapshot，其中 `spec.dataSource`（或者 `spec.dataSourceRef`名称是 `VolumeSnapshot`姓名。

   示例

以下示例创建一个引用 PVC 的 `VolumeSnapshot`命名 `import-snap`。

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

使用快照恢复卷数据

默认情况下，快照目录是隐藏的，以确保使用以下方式配置的卷的最大兼容性： `ontap-nas`和 `ontap-nas-economy`司机。启用 `.snapshot`直接从快照恢复数据的目录。

使用卷快照恢复ONTAP CLI 将卷恢复到先前快照中记录的状态。

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

恢复快照副本时，现有卷配置将被覆盖。创建快照副本后对卷数据所做的更改将会丢失。

从快照进行原地体积恢复

Trident利用快照提供快速、原位体积恢复功能 `TridentActionSnapshotRestore`（TASR）CR。此 CR 作为一项强制性 Kubernetes 操作，在操作完成后不会持久保存。

Trident支持快照恢复 `ontap-san`， `ontap-san-economy` ， `ontap-nas` ， `ontap-nas-flexgroup` ， `azure-netapp-files` ， `gcp-cvs` ， `google-cloud-netapp-volumes` ， 和 `solidfire-san`司机。

开始之前

您必须拥有已绑定的PVC和可用的卷快照。

- 确认PVC状态是否已绑定。

```
kubectl get pvc
```

- 确认卷快照已准备就绪。

```
kubectl get vs
```

步骤

1. 创建 TASR CR。此示例为PVC创建CR。 `pvc1`和卷快照 `pvc1-snapshot`。

   TASR CR 必须位于 PVC 和 VS 存在的命名空间中。

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. 应用 CR 从快照恢复。此示例从快照恢复 `pvc1`。

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

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

结果

Trident从快照中恢复数据。您可以验证快照恢复状态：

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- 大多数情况下， Trident不会在操作失败后自动重试。您需要再次执行此操作。
- 没有管理员权限的 Kubernetes 用户可能需要管理员授予权限才能在其应用程序命名空间中创建 TASR CR。

删除包含关联快照的 **PV**

删除具有关联快照的持久卷时，相应的Trident卷将更新为“正在删除”状态。删除卷快照以删除Trident卷。

部署卷快照控制器

如果您的 Kubernetes 发行版不包含快照控制器和 CRD，您可以按如下方式部署它们。

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. 创建快照控制器。

```
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
```

如有必要，打开 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`并更新 `namespace`添加到您的命名空间。

相关链接

- "卷快照"
- "卷快照类"

使用卷组快照

NetApp Trident提供了创建多个卷（一组卷快照）快照的功能，可用于创建持久卷 (PV) 的 Kubernetes 卷组快照。此卷组快照表示在同一时间点从多个卷中获取的副本。

VolumeGroupSnapshot 是 Kubernetes 中的一项测试版功能，其 API 也处于测试版阶段。VolumeGroupSnapshot 所需的最低 Kubernetes 版本为 1.32。

创建卷组快照

卷组快照受支持 `ontap-san`该驱动程序仅适用于 iSCSI 协议，尚不支持光纤通道 (FCP) 或 NVMe/TCP。开始之

前

- 请确保您的 Kubernetes 版本为 K8s 1.32 或更高版本。
- 要使用快照，您必须拥有外部快照控制器和自定义资源定义 (CRD)。这是 Kubernetes 编排器（例如：Kubeadm、GKE、OpenShift）的职责。

  如果您的 Kubernetes 发行版不包含外部快照控制器和 CRD，请参阅[部署卷快照控制器]。

  > 如果在 GKE 环境中创建按需卷组快照，则不要创建快照控制器。 GKE 使用内置的隐藏快照控制器。

- 在快照控制器 YAML 中，设置 `CSIVolumeGroupSnapshot`将功能门设置为“true”，以确保启用卷组快照。
- 在创建卷组快照之前，请先创建所需的卷组快照类。
- 确保所有 PVC/卷都在同一 SVM 上，以便能够创建 VolumeGroupSnapshot。

步骤

- 在创建 VolumeGroupSnapshot 之前，请先创建 VolumeGroupSnapshotClass。更多信息，请参阅"卷组快照类"。

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- 使用现有的存储类别创建带有所需标签的 PVC，或者将这些标签添加到现有的 PVC。

  以下示例使用以下方法创建 PVC： `pvc1-group-snap`作为数据源和标签 `consistentGroupSnapshot: groupA`。根据您的需求定义标签的键和值。

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

- 创建具有相同标签的卷组快照(`consistentGroupSnapshot: groupA`）在PVC中规定。

  此示例创建卷组快照：

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

使用组快照恢复卷数据

您可以使用作为卷组快照一部分创建的各个快照来恢复各个持久卷。您无法将卷组快照作为一个整体恢复。

使用卷快照恢复ONTAP CLI 将卷恢复到先前快照中记录的状态。

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

恢复快照副本时，现有卷配置将被覆盖。创建快照副本后对卷数据所做的更改将会丢失。

从快照进行原地体积恢复

Trident利用快照提供快速、原位体积恢复功能 `TridentActionSnapshotRestore`（TASR）CR。此 CR 作为一项强制性 Kubernetes 操作，在操作完成后不会持久保存。

有关详细信息，请参阅 "从快照进行原地体积恢复"。

删除包含关联组快照的 **PV**

删除组卷快照时：

- 您可以删除整个 VolumeGroupSnapshots，而不是删除组中的单个快照。
- 如果在持久卷存在快照的情况下删除该持久卷， Trident会将该卷移至“正在删除”状态，因为必须先删除快照才能安全地删除该卷。
- 如果使用分组快照创建了克隆，然后要删除该组，则会开始在克隆时进行拆分操作，并且在拆分完成之前无法删除该组。

部署卷快照控制器

如果您的 Kubernetes 发行版不包含快照控制器和 CRD，您可以按如下方式部署它们。

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-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. 创建快照控制器。

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```

如有必要，打开 `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml`并更新 `namespace`添加到您的命名空间。

相关链接

- "卷组快照类"
- "卷快照"