配置和管理后端
Trident

NetApp
September 26, 2025

目录

配置和管理后端

配置后端

后端用于定义Trident与存储系统之间的关系。它会告诉Trident如何与该存储系统通信、以及Trident如何从该存储系统配置卷。

Trident会自动从满足存储类定义的要求的后端提供存储池。了解如何为存储系统配置后端。

- "配置 Azure NetApp Files 后端"
- "配置Google Cloud NetApp卷后端"
- "配置适用于 Google 云平台的 Cloud Volumes Service 后端"
- "配置 NetApp HCI 或 SolidFire 后端"
- "使用ONTAP或Cloud Volumes ONTAP NAS驱动程序配置后端"
- "使用ONTAP或Cloud Volumes ONTAP SAN驱动程序配置后端"
- "将Trident与Amazon FSx for NetApp ONTAP结合使用"

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后端连接NFS和SMB卷。Trident还支持使用托管身份对Azure Kubnetes Services (AKS)集群进行凭据管理。

Azure NetApp Files驱动程序详细信息

Trident提供了以下Azure NetApp Files存储驱动程序来与集群进行通信。支持的访问模式包括：*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(rwx)、*ReadWriteOncePod*(RWOP)。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`azure-netapp-files`	NFS SMB	文件系统	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50 GiB卷。
- Trident仅支持挂载到Windows节点上运行的Pod的SMB卷。

AKS的受管身份

Trident支持"受管身份"Azure Kubnetes服务集群。要利用受管身份提供的简化凭据管理、您必须：

- 使用AKS部署的Kubenetes集群
- 在AKS Kubbernetes集群上配置的受管身份
- 安装了Trident，其中包括要指定 `"Azure"`的 `cloudProvider。

Trident 运算符

要使用Trident运算符安装Trident，请编辑 `tridentorchestrator_cr.yaml`以将设置 `cloudProvider`为 `"Azure"。例如：

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

掌舵

以下示例使用环境变量将Trident集安装 `cloudProvider`到Azure `$CP`：

```
helm install trident trident-operator-100.2410.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code></code>

以下示例将安装Trident并将标志设置 `cloudProvider`为 `Azure`：

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

适用于**AKS**的云身份

通过云身份、Kubnetes Pod可以通过作为工作负载身份进行身份验证来访问Azure资源、而不是提供明确的Azure凭据。

要在Azure中利用云身份、您必须：

- 使用AKS部署的Kubenetes集群
- 在AKS Kubelnetes集群上配置的工作负载身份和oidc-Issuer
- 已安装Trident、其中包括 `cloudProvider`用于指定 `"Azure"`和 `cloudIdentity`指定工作负载标识的

Trident 运算符

要使用Trident运算符安装Trident，请编辑 `tridentorchestrator_cr.yaml`以将设置为，并将 `cloudIdentity`设置 `cloudProvider`为 `"Azure" 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'*
```

掌舵

使用以下环境变量设置*云提供程序(CP)*和*云身份(CI)*标志的值：

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

以下示例将安装Trident并使用环境变量将设置 `cloudProvider`为`Azure `$CP`、并使用环境变量 `$CI`设置 `cloudIdentity`：

```
helm install trident trident-operator-100.2410.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code></code>

使用以下环境变量设置*云提供程序*和*云身份*标志的值：

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

以下示例将安装Trident并将标志设置 `cloud-provider`为 `$CP`、和 `cloud-identity` `$CI`：

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

准备配置Azure NetApp Files 后端

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

NFS和SMB卷的前提条件

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

要配置和使用 "Azure NetApp Files"后端、您需要满足以下条件：

- `clientID` `location`在AKS集群上使用受管标识时，`subscriptionID`、、 `tenantID`和 `clientSecret`是可选的。
- `tenantID` `clientID`在AKS集群上使用云标识时、和 `clientSecret`是可选的。

- 一个容量池。请参阅 "Microsoft：为Azure NetApp Files 创建容量池"。
- 委派给Azure NetApp Files 的子网。请参阅 "Microsoft：将子网委派给Azure NetApp Files"。
- `subscriptionID`通过启用了Azure NetApp Files的Azure订阅。
- `tenantID` `clientID`和 `clientSecret` "应用程序注册"中具有足够权限的Azure NetApp Files服务。应用程序注册应使用以下任一项：
  - 所有者或贡献者角色"由Azure预定义"。
  - "自定义贡献者角色"订阅级别(`assignableScopes`的)具有以下权限，这些权限仅限于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": []
            }
        ]
    }
}
```

- 至少包含一个 "委派子网"的Azure `location`。自Trident 22.01起、 `location`参数是后端配置文件顶层的必填字段。在虚拟池中指定的位置值将被忽略。
- 要使用 `Cloud Identity`，请从 "用户分配的托管身份"获取 `client ID`并在中指定该`ID` `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`。

SMB卷的其他要求

要创建SMB卷、您必须具有：

- 已配置Active Directory并连接到Azure NetApp Files。请参阅 "Microsoft：创建和管理Azure NetApp Files的Active Directory连接"。
- 一个Kubernetes集群、其中包含一个Linux控制器节点以及至少一个运行Windows Server 2022的Windows工作节点。Trident仅支持挂载到Windows节点上运行的Pod的SMB卷。
- 至少一个包含Active Directory凭据的Trident密钥、以便Azure NetApp Files可以向Active Directory进行身份验证。生成密钥 `smbcreds`：

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

- 配置为Windows服务的CSI代理。要配置 `csi-proxy`，请参阅"GitHub：CSI代理"或了解在Windows上运行的Kubornetes"GitHub：适用于Windows的CSI代理"节点。

Azure NetApp Files 后端配置选项和示例

了解适用于Azure NetApp Files的NFS和SMB后端配置选项并查看配置示例。

后端配置选项

Trident可使用您的后端配置(子网、虚拟网络、服务级别和位置)在请求位置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匹配。

Trident不支持手动QoS容量池。

Azure NetApp Files后端提供了这些配置选项。

参数	说明	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	驱动程序名称 + "_" + 随机字符
`subscriptionID`	在AKS集群上启用托管标识时、Azure订阅中的订阅ID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为可选。	
`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为可选。	
`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可选。	
`serviceLevel`	`Premium`或 `Ultra`之一 `Standard`	"" （随机）
`location`	要在其中创建新卷的Azure位置的名称在AKS集群上启用受管标识时为可选。	
`resourceGroups`	用于筛选已发现资源的资源组列表	"[]" （无筛选器）
`netappAccounts`	用于筛选已发现资源的 NetApp 帐户列表	"[]" （无筛选器）
`capacityPools`	用于筛选已发现资源的容量池列表	"[]" （无筛选器，随机）
`virtualNetwork`	具有委派子网的虚拟网络的名称	""
`subnet`	委派给的子网的名称 `Microsoft.Netapp/volumes`	""

参数	说明	默认
networkFeatures	一个卷的一组vNet功能，可以是 Basic`或 `Standard。网络功能并非在所有地区都可用、可能需要在订阅中启用。如果指定 `networkFeatures`未启用此功能的时间、则会导致卷配置失败。	""
nfsMountOptions	精细控制 NFS 挂载选项。SMB卷已忽略。要使用NFS 4.1挂载卷、请在逗号分隔挂载选项列表中包含 `nfsvers=4`以选择NFS v4.1。存储类定义中设置的挂载选项会覆盖后端配置中设置的挂载选项。	"nfsvers=3"
limitVolumeSize	如果请求的卷大小超过此值，则配置失败	"" （默认情况下不强制实施）
debugTraceFlags	故障排除时要使用的调试标志。例如， \{"api": false, "method": true, "discovery": true}。除非您正在进行故障排除并需要详细的日志转储，否则请勿使用此功能。	空
nasType	配置NFS或SMB卷创建。选项为 nfs、 `smb`或null。默认情况下、将设置为空会将NFS卷设置为空。	nfs
supportedTopologies	表示此后端支持的区域和区域的列表。有关详细信息，请参阅 "使用 CSI 拓扑"。	

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

所需权限和资源

如果在创建PVC时收到"No Capacity Pools"(未找到容量池)错误、则您的应用程序注册可能没有关联的所需权限和资源(子网、虚拟网络、容量池)。如果启用了调试、Trident将记录在创建后端时发现的Azure资源。验证是否正在使用适当的角色。

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

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

键入	格式
Resource group	< 资源组 >
NetApp 帐户	< 资源组 >/< NetApp 帐户 >
容量池	< 资源组 >/< NetApp 帐户 >/< 容量池 >
虚拟网络	< 资源组 >/< 虚拟网络 >
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殊部分中指定以下选项来控制默认卷配置。有关详细信息、请参见 [示例配置] 。

参数	说明	默认
`exportRule`	新卷的导出规则。`exportRule`必须是IPv4地址或IPv4子网的任意组合的逗号分隔列表(采用CIDR表示法)。SMB卷已忽略。	"0.0.0.0/0
`snapshotDir`	控制 .snapshot 目录的可见性	对于NFSv4、为"TRUE"；对于NFSv3、为"false"
`size`	新卷的默认大小	"100 克 "
`unixPermissions`	新卷的UNIX权限(4个八进制数字)。SMB卷已忽略。	"" （预览功能，需要在订阅中列入白名单）

示例配置

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

最低配置

这是绝对的最低后端配置。使用此配置时、Trident会发现在所配置位置委派给Azure NetApp Files的所有NetApp帐户、容量池和子网、并随机将新卷放置在其中一个池和子网上。由于 `nasType` 省略了、因此会 `nfs` 应用默认设置、后端将为NFS卷配置。

当您刚刚开始使用Azure NetApp Files并尝试某些操作时、此配置是理想的选择、但实际上、您需要为所配置的卷提供额外的范围界定。

```
---
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
```

AKS的受管身份

此后端配置会省略 `subscriptionID`、 `tenantID`、 `clientID`和 `clientSecret`，它们在使用受管身份时是可选的。

```
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
```

适用于**AKS**的云身份

此后端配置会省略 `tenantID`、 `clientID`和 `clientSecret`，它们在使用云标识时是可选的。

```
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
```

使用容量池筛选器的特定服务级别配置

此后端配置会将卷放置在Azure的 `eastus`容量池中 `Ultra`。Trident会自动发现该位置委派给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
```

高级配置

此后端配置进一步将卷放置范围缩小为一个子网，并修改了某些卷配置默认值。

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

虚拟池配置

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

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

支持的拓扑配置

Trident可以根据区域和可用性区域为工作负载配置卷。 `supportedTopologies`此后端配置中的块用于提供每个后端的区域和分区列表。此处指定的区域和分区值必须与每个Kubnetes集群节点上标签中的区域和分区值匹配。这些区域和分区表示可在存储类中提供的允许值列表。对于包含后端提供的部分区域和区域的存储类、Trident会在上述区域和区域中创建卷。有关详细信息，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

以下 `StorageClass`定义涉及上述存储池。

使用字段的示例定义 `parameter.selector`

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

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

SMB卷的示例定义

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

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

创建后端

创建后端配置文件后，运行以下命令：

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

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

```
tridentctl logs
```

确定并更正配置文件中的问题后，您可以再次运行 create 命令。

Google Cloud NetApp卷

配置Google Cloud NetApp卷后端

现在、您可以将Google Cloud NetApp卷配置为Trident的后端。您可以使用Google Cloud NetApp卷后端连接NFS卷。

Google Cloud NetApp卷驱动程序详细信息

Trident提供了 `google-cloud-netapp-volumes`用于与集群通信的驱动程序。支持的访问模式包括：*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(rwx)、*ReadWriteOncePod*(RWOP)。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`google-cloud-netapp-volumes`	NFS	文件系统	Rwo、ROX、rwx、RWOP	`nfs`

适用于GKE的云身份

通过云身份、Kubnetes Pod可以通过作为工作负载身份进行身份验证来访问Google Cloud资源、而不是提供明确的Google Cloud凭据。

要在Google Cloud中利用云身份、您必须：

- 使用GKE部署的Kubbernetes集群。
- 在GKE集群上配置的工作负载标识以及在节点池上配置的GKE元数据服务器。
- 具有Google Cloud NetApp卷管理员(角色/GCP .admin)角色或自定义角色的NetApp服务帐户。
- 已安装Trident、其中包括用于指定"gcp"的云提供程序和用于指定新GCP服务帐户的云标识。下面给出了一个示例。

Trident 运算符

要使用Trident运算符安装Trident，请编辑 `tridentorchestrator_cr.yaml`以将设置为，并将 `cloudIdentity`设置 `cloudProvider`为 `"GCP" 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'
```

掌舵

使用以下环境变量设置*云提供程序(CP)*和*云身份(CI)*标志的值：

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

以下示例将安装Trident并使用环境变量将设置 `cloudProvider`为`GCP `$CP`，并使用环境变量 `$ANNOTATION`设置 `cloudIdentity`：

```
helm install trident trident-operator-100.2406.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code></code>

使用以下环境变量设置*云提供程序*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"
```

以下示例将安装Trident并将标志设置 `cloud-provider`为 `$CP、和 `cloud-identity `$ANNOTATION：

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

准备配置Google Cloud NetApp卷后端

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

NFS卷的前提条件

如果您是首次使用Google Cloud NetApp卷或在新位置使用、则需要进行一些初始配置才能设置Google Cloud NetApp卷和创建NFS卷。请参阅 "开始之前"。

在配置Google Cloud NetApp卷后端之前、请确保您满足以下条件：

- 配置有Google Cloud NetApp卷服务的Google Cloud帐户。请参阅 "Google Cloud NetApp卷"。
- 您的Google Cloud帐户的项目编号。请参阅 "确定项目"。
- 具有NetApp卷管理员角色的Google Cloud服务帐户 (`roles/netapp.admin`。请参阅 "身份和访问管理角色和权限"。
- 您的GCNV帐户的API密钥文件。请参见 "创建服务帐户密钥"
- 存储池。请参阅 "存储池概述"。

有关如何设置对Google Cloud NetApp卷的访问权限的详细信息，请参阅 "设置对Google Cloud NetApp卷的访问权限"。

Google Cloud NetApp Volumes后端配置选项和示例

了解适用于Google Cloud NetApp卷的NFS后端配置选项并查看配置示例。

后端配置选项

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

参数	说明	默认
`version`		始终为 1
`storageDriverName`	存储驱动程序的名称	的值 `storageDriverName` 必须指定为"gosle-Cloud NetApp-volumes"。
`backendName`	(可选)存储后端的自定义名称	驱动程序名称 + "_" + API密钥的一部分
`storagePools`	用于指定用于创建卷的存储池的可选参数。	
`projectNumber`	Google Cloud 帐户项目编号。此值可在Google Cloud门户主页上找到。	
`location`	Trident创建GCNV卷的Google Cloud位置。创建跨区域Kubnetes集群时、在中创建的卷 `location` 可用于在多个Google Cloud区域的节点上计划的工作负载。跨区域流量会产生额外成本。	

参数	说明	默认
apiKey	具有角色的Google Cloud服务帐户的API密钥 netapp.admin 。它包括 Google Cloud 服务帐户专用密钥文件的 JSON 格式的内容（逐字复制到后端配置文件）。 apiKey`必须包括以下键的键值对：`type、 project_id、 client_email、、client_id auth_uri token_uri auth_provider_x509_cert_url`和`client_x509_cert_url。	
nfsMountOptions	精细控制 NFS 挂载选项。	"nfsvers=3"
limitVolumeSize	如果请求的卷大小超过此值、则配置失败。	"" （默认情况下不强制实施）
serviceLevel	存储池及其卷的服务级别。值为 flex、、 standard premium`或 `extreme。	
network	用于GCNV卷的Google Cloud网络。	
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需要详细的日志转储，否则请勿使用此功能。	空
supportedTopologies	表示此后端支持的区域和区域的列表。有关详细信息，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

您可以在配置文件的部分中控制默认卷配置 defaults。

参数	说明	默认
exportRule	新卷的导出规则。必须是IPv4地址任意组合的逗号分隔列表。	"0.0.0.0/0
snapshotDir	对目录的访问权限 .snapshot	对于NFSv4、为"TRUE"；对于NFSv3、为"false"
snapshotReserve	为快照预留的卷百分比	""(接受默认值0)
unixPermissions	新卷的UNIX权限(4个八进制数字)。	""

示例配置

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

最低配置

这是绝对的最低后端配置。使用此配置时、Trident会发现在配置位置委派给Google Cloud NetApp卷的所有存储池、并随机将新卷放置在其中一个池上。由于 `nasType` 省略了、因此会 `nfs` 应用默认设置、后端将为NFS卷配置。

当您刚刚开始使用Google Cloud NetApp卷并尝试操作时、此配置是理想的选择、但实际上、您很可能需要为您配置的卷提供额外的范围界定。

```
---

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
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\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
```

使用**StoragePools**筛选器的配置

```
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
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---

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

此后端配置可在一个文件中定义多个虚拟池。虚拟池在一节中进行了定义 `storage` 。如果您有多个存储池支持不同的服务级别、而您希望在Kubbernetes中创建表示这些服务级别的存储类、则这些选项板非常有用。虚拟池标签用于区分池。例如、在以下示例中 `performance` 、标签和 `serviceLevel` 类型用于区分虚拟池。

此外、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。在以下示例中、`snapshotReserve` 和 `exportRule` 用作所有虚拟池的默认值。

有关详细信息，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
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
```

```
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---

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
```

适用于**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可以根据区域和可用性区域为工作负载配置卷。 `supportedTopologies`此后端配置中的块用于提供每个后端的区域和分区列表。此处指定的区域和分区值必须与每个Kubnetes集群节点上标签中的区域和分区值匹配。这些区域和分区表示可在存储类中提供的允许值列表。对于包含后端提供的部分区域和区域的存储类、Trident会在上述区域和区域中创建卷。有关详细信息，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>
```

要验证是否已成功创建后端、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，则后端配置出现问题。您可以使用命令说明后端 `kubectl get tridentbackendconfig <backend-name>` 、或者运行以下命令查看日志以确定原因：

```
tridentctl logs
```

确定并更正配置文件的问题后、您可以删除后端并再次运行create命令。

更多示例

存储类定义示例

下面是有关上述后端的基本 `StorageClass` 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"
```

使用字段的示例定义 **`parameter.selector`** ：

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

```
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"
```

有关存储类的详细信息，请参见 "创建存储类。"。

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适用于Google Cloud的NetApp Cloud Volumes Service配置为Trident安装的后端。

Google Cloud驱动程序详细信息

Trident提供了 `gcp-cvs`用于与集群通信的驱动程序。支持的访问模式包括：*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(rwx)、*ReadWriteOncePod*(RWOP)。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`gcp-cvs`	NFS	文件系统	Rwo、ROX、rwx、RWOP	`nfs`

了解Trident对适用于Google Cloud的Cloud Volumes Service的支持

Trident可以使用"服务类型"以下两种方法之一创建Cloud Volumes Service卷：

- **CVS**性能：默认的Trident服务类型。这种性能优化的服务类型最适合重视性能的生产工作负载。CVS-Performance服务类型是一种硬件选项、支持的卷大小至少为100 GiB。您可以选择"三个服务级别"以下选项之一：
  - `standard`
  - `premium`
  - `extreme`
- * CVS*：CVS服务类型提供高区域可用性、性能级别限制为中等。CVS服务类型是一个软件选项、可使用存储池支持小至1 GiB的卷。存储池最多可包含50个卷、其中所有卷都共享池的容量和性能。您可以选择"两个服务级别"以下选项之一：
  - `standardsw`
  - `zoneredundantstandardsw`

您需要的内容

要配置和使用 "适用于 Google Cloud 的 Cloud Volumes Service"后端、您需要满足以下条件：

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

- Cloud Volumes Service 帐户的API密钥文件

后端配置选项

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

参数	说明	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。 `software`用于选择CVS服务类型。否则，Trident会采用CVS性能服务类型(`hardware)。	
`storagePools`	仅限CVS服务类型。用于指定用于创建卷的存储池的可选参数。	
`projectNumber`	Google Cloud 帐户项目编号。此值可在Google Cloud 门户主页上找到。	
`hostProjectNumber`	如果使用共享VPC网络、则为必需项。在此方案中、`projectNumber`是服务项目、是主项目`hostProjectNumber。	
`apiRegion`	Trident创建Cloud Volumes Service卷的Google Cloud区域。创建跨区域Kubnetes集群时、在中创建的卷`apiRegion`可用于在多个Google Cloud区域的节点上计划的工作负载。跨区域流量会产生额外成本。	
`apiKey`	具有角色的Google Cloud服务帐户的API密钥 `netappcloudvolumes.admin` 。它包括 Google Cloud 服务帐户专用密钥文件的 JSON 格式的内容（逐字复制到后端配置文件）。	
`proxyURL`	代理服务器需要连接到CVS帐户时的代理URL。代理服务器可以是 HTTP 代理，也可以是 HTTPS 代理。对于 HTTPS 代理，将跳过证书验证，以允许在代理服务器中使用自签名证书。不支持启用了身份验证的代理服务器。	
`nfsMountOptions`	精细控制 NFS 挂载选项。	"nfsvers=3"
`limitVolumeSize`	如果请求的卷大小超过此值、则配置失败。	"" （默认情况下不强制实施）
`serviceLevel`	新卷的CVS-Performance或CVS服务级别。CVS性能值为 `standard`、 `premium`或 `extreme`。CVS值为 `standardsw`或`zoneredundantstandardsw`。	CVS-Performance默认值为"standard"。CVS默认值为"standardsw"。
`network`	用于Cloud Volumes Service 卷的Google云网络。	default

参数	说明	默认
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需要详细的日志转储，否则请勿使用此功能。	空
allowedTopologies	要启用跨区域访问、的存储类定义allowedTopologies`必须包括所有区域。例如：`- key: topology.kubernetes.io/region values: - us-east1 - europe-west1	

卷配置选项

您可以在配置文件的部分中控制默认卷配置 defaults。

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

CVS-Performance服务类型示例

以下示例提供了CVS-Performance服务类型的示例配置。

示例 **1**： 最低配置

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

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

此示例说明了后端配置选项、包括服务级别和卷默认值。

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <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`配置虚拟池以及引用这些虚拟池的 `StorageClasses。请参见[存储类定义]以了解存储类的定义方式。

此处为所有虚拟池设置了特定默认值、将设置为5%、将设置 snapshotReserve`为 `exportRule`0.0.0.0/0。虚拟池在一节中进行了定义 `storage。每个虚拟池都定义自己的 serviceLevel，而某些池会覆盖默认值。虚拟池标签用于根据和 protection`区分池 `performance。

```
---
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。卷将在选定池中定义各个方面。

存储类示例

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: 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StorageClass(`cvs-extreme-extra-protection`)映射到第一个虚拟池。这是唯一一个可提供极高性能且 Snapshot 预留为 10% 的池。
- 最后一个StorageClass(`cvs-extra-protection`)调用提供10%快照预留的任何存储池。Trident决定选择哪个虚拟池、并确保满足快照预留要求。

CVS服务类型示例

以下示例提供了CVS服务类型的示例配置。

示例**1**：最低配置

这是用于指定CVS服务类型和默认 `standardsw`服务级别的最低后端配置 `storageClass`。

```
---
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，则后端配置出现问题。您可以运行以下命令来查看日志以确定发生原因：

```
tridentctl logs
```

确定并更正配置文件中的问题后，您可以再次运行 create 命令。

配置 NetApp HCI 或 SolidFire 后端

了解如何在Trident安装中创建和使用Element后端。

Element驱动程序详细信息

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

`solidfire-san`存储驱动程序支持_file_和_block_卷模式。对于 `Filesystem` 卷模式、Trident会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	卷模式	支持的访问模式	支持的文件系统
solidfire-san	iSCSI	块	Rwo、ROX、rwx、RWOP	无文件系统。原始块设备。
solidfire-san	iSCSI	文件系统	Rwo、RWO1.	xfs、 ext3、 ext4

开始之前

在创建Element后端之前、您需要满足以下要求。

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

后端配置选项

有关后端配置选项，请参见下表：

参数	说明	默认
version		始终为 1
storageDriverName	存储驱动程序的名称	始终为 "solidfire-san"
backendName	自定义名称或存储后端	SolidFire + 存储（ iSCSI ） IP 地址
Endpoint	使用租户凭据的 SolidFire 集群的 MVIP	

参数	说明	默认
SVIP	存储（ iSCSI ） IP 地址和端口	
labels	要应用于卷的一组任意 JSON 格式的标签。	"
TenantName	要使用的租户名称（如果未找到，则创建）	
InitiatorIFace	将 iSCSI 流量限制为特定主机接口	default
UseCHAP	使用CHAP对iSCSI进行身份验证。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	" （默认情况下不强制实施）
debugTraceFlags	故障排除时要使用的调试标志。示例 ｛ "api"： false ， "method "： true ｝	空

除非正在进行故障排除并需要详细的日志转储、否则请勿使用 debugTraceFlags。

示例1：具有三种卷类型的驱动程序的后端配置 solidfire-san

此示例显示了一个后端文件，该文件使用 CHAP 身份验证并使用特定 QoS 保证对三种卷类型进行建模。然后、您很可能会使用storage class参数定义要使用其中每个存储类的存储类 IOPS。

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

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

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

在下面显示的示例后端定义文件中、为所有存储池设置了特定默认值、并将设置 `type`为银牌。虚拟池在一节中进行了定义 `storage`。在此示例中、某些存储池会设置自己的类型、而某些存储池会覆盖上面设置的默认值。

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

第一个StorageClass(`solidfire-gold-four`)将映射到第一个虚拟池。这是唯一一个提供金牌性能的池

`Volume Type QoS`。最后一个StorageClass(`solidfire-silver`)会调用任何提供银牌性能的存储池。Trident将决定选择哪个虚拟池、并确保满足存储要求。

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

了解更多信息

- "卷访问组"

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驱动程序详细信息

Trident提供了以下SAN存储驱动程序来与ONTAP集群进行通信。支持的访问模式包括：*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(rwx)、*ReadWriteOncePod*(RWOP)。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`ontap-san`	基于FC的iSCSI SCSI (Trident 24.10中的技术预览)	块	Rwo、ROX、rwx、RWOP	无文件系统；原始块设备
`ontap-san`	基于FC的iSCSI SCSI (Trident 24.10中的技术预览)	文件系统	Rwo、RWO1. Rox和rwx在文件系统卷模式下不可用。	`xfs`、 `ext3`、 `ext4`
`ontap-san`	NVMe/TCP 请参阅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 请参阅NVMe/TCP的其他注意事项。	文件系统	Rwo、RWO1. Rox和rwx在文件系统卷模式下不可用。	`xfs`、 `ext3`、 `ext4`
`ontap-san-economy`	iSCSI	块	Rwo、ROX、rwx、RWOP	无文件系统；原始块设备

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`ontap-san-economy`	iSCSI	文件系统	Rwo、RWO1. Rox和rwx在文件系统卷模式下不可用。	`xfs、 ext3、 ext4`

- 只有当永久性卷使用量计数预期大于时才使用 `ontap-san-economy`"支持的ONTAP卷限制"。
- `ontap-nas-economy`仅当永久性卷使用量计数预计高于且 `ontap-san-economy`无法使用驱动程序时才"支持的ONTAP卷限制"使用。
- 如果您预计需要数据保护、灾难恢复或移动性、请勿使用 `ontap-nas-economy`。

用户权限

Trident应以ONTAP或SVM管理员身份运行、通常使用集群用户 `vsadmin`或SVM用户、或者使用 `admin`具有相同角色的其他名称的用户。对于Amazon FSx for NetApp ONTAP部署、Trident应使用集群用户 `vsadmin`或SVM用户以ONTAP或SVM管理员身份运行、或者使用具有相同角色的其他名称的用户运行 `fsxadmin`。此 `fsxadmin`用户只能有限地替代集群管理员用户。

如果使用 `limitAggregateUsage`参数、则需要集群管理员权限。将Amazon FSx for NetApp ONTAP与Trident结合使用时、 `limitAggregateUsage`参数不适用于 `vsadmin`和 `fsxadmin`用户帐户。如果指定此参数，配置操作将失败。

虽然可以在ONTAP中创建一个可以由三端驱动程序使用的限制性更强的角色、但我们不建议这样做。大多数新版本的 Trident 都会调用需要考虑的其他 API ，从而使升级变得困难且容易出错。

NVMe/TCP的其他注意事项

Trident使用以下驱动程序支持非易失性内存快速(NVMe)协议 `ontap-san`：

- IPv6
- NVMe卷的快照和克隆
- 调整NVMe卷大小
- 导入在Trident外部创建的NVMe卷、以便Trident可以管理其生命周期
- NVMe本机多路径
- 正常或非正常关闭K8s节点(24.06)

Trident不支持：

- DH-HMAC-CHAP、由NVMe本机提供支持
- 设备映射程序(Device maper、DM)多路径
- 进行了加密

准备使用ONTAP SAN驱动程序配置后端

了解使用ONTAP SAN驱动程序配置ONTAP后端的要求和身份验证选项。

要求

对于所有ONTAP后端、Trident要求至少为SVM分配一个聚合。

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

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

对ONTAP后端进行身份验证

Trident提供了两种对ONTAP后端进行身份验证的模式。

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

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

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

启用基于凭据的身份验证

Trident需要SVM范围/集群范围的管理员的凭据才能与ONTAP后端进行通信。建议使用标准的预定义角色，如`admin`或 `vsadmin`。这样可以确保与未来ONTAP版本的正向兼容性、这些版本可能会公开未来Trident版本要使用的功能API。可以创建自定义安全登录角色并将其用于Trident、但不建议这样做。

后端定义示例如下所示：

YAML

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

JSON

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

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

启用基于证书的身份验证

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

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

典型的工作流包括以下步骤。

步骤

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

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

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

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

3. 在 ONTAP 集群上安装客户端证书和密钥（从步骤 1 开始）。

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

4. 确认ONTAP安全登录角色支持 `cert`身份验证方法。

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

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

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

6. 使用 Base64 对证书，密钥和可信 CA 证书进行编码。

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

7. 使用从上一步获得的值创建后端。

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

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

更新身份验证方法或轮换凭据

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

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

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

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

更新后端不会中断对已创建卷的访问，也不会影响在之后建立的卷连接。后端更新成功表示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Privileges最低的ONTAP集群角色、这样就不必使用ONTAP管理员角色在Trident中执行操作。如果在Trident后端配置中包含用户名、则Trident将使用您创建的ONTAP集群角色来执行操作。

有关创建Trident自定义角色的详细信息、请参见"Trident自定义角色生成器"。

使用**ONTAP**命令行界面

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 **System Manager**

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*Cluster > Settings*。

      (或)要在SVM级别创建自定义角色、请选择*存储> Storage VM `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. 定义角色的规则，然后单击*Save*。
2. 将角色映射到**Trident user**：+在*Users and Roles*页面上执行以下步骤：
   a. 在*用户*下选择添加图标*+*。
   b. 选择所需的用户名，然后在下拉菜单中为*rouser*选择一个角色。
   c. 单击 * 保存 * 。

有关详细信息、请参见以下页面：

- "用于管理ONTAP的自定义角色"或"定义自定义角色"
- "使用角色和用户"

使用双向 **CHAP** 对连接进行身份验证

Trident可以使用和 `ontap-san-economy`驱动程序的双向`CHAP`对`iSCSI`会话进行身份验证 `ontap-san`。这需要在后端定义中启用此 `useCHAP`选项。设置为时 `true`，Trident会将SVM的默认启动程序安全性配置为双向CHAP，并设置后端文件中的用户名和密钥。NetApp 建议使用双向 CHAP 对连接进行身份验证。请参见以

下配置示例：

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

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

除了之外， `chapInitiatorSecret` `chapTargetUsername`后端定义中还 `useCHAP=true`必须包括、、 `chapTargetInitiatorSecret`和 `chapUsername`字段。在创建后端后，可以通过运行来更改这些密钥 `tridentctl update`。

工作原理

如果将设置 `useCHAP`为true、则存储管理员将指示Trident在存储后端配置CHAP。其中包括：

- 在 SVM 上设置 CHAP ：
  - 如果SVM的默认启动程序安全类型为none (默认设置)*和*卷中已没有已有的LUN、则Trident会将默认安全类型设置为 `CHAP`、并继续配置CHAP启动程序以及目标用户名和密码。
  - 如果SVM包含LUN、则Trident不会在此SVM上启用CHAP。这样可确保对SVM上已存在的LUN的访问不受限制。
- 配置 CHAP 启动程序以及目标用户名和密码；必须在后端配置中指定这些选项（如上所示）。

创建后端后、Trident会创建相应的 `tridentbackend`CRD并将CHAP密码和用户名存储为Kubbernetes密码。Trident在此后端创建的所有PV,都将通过CHAP进行挂载和连接。

轮换凭据并更新后端

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

更新后端的CHAP密码时、必须使用 `tridentctl`更新后端。请勿通过CLI/RAID ONTAP UI更新存储集群上的凭据、因为Trident将无法接受这些更改。

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

现有连接不会受到影响；如果Trident在SVM上更新凭据、这些连接将继续保持活动状态。新连接将使用更新后的凭据、现有连接将继续保持活动状态。断开并重新连接旧的 PV 将导致它们使用更新后的凭据。

ONTAP SAN配置选项和示例

了解如何在Trident安装中创建和使用ONTAP SAN驱动程序。本节提供了将后端映射到StorageClasses的后端配置示例和详细信息。

后端配置选项

有关后端配置选项，请参见下表：

参数	说明	默认
`version`		始终为 1

参数	说明	默认
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是使用IPv6标志安装的、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]。有关无缝MetroCluster切换的信息，请参见[mcc-best]。	"10.0.0.1 "， "2001： 1234：abcd：：： fefe] "
dataLIF	协议 LIF 的 IP 地址。如果Trident是使用IPv6标志安装的、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]。*不指定iSCSI。*Trident使用"ONTAP 选择性LUN映射"发现建立多路径会话所需的iCI LUN。如果明确定义、则会生成警告 dataLIF。*省略MetroCluster。*请参见[mcc-best]。	由SVM派生
svm	要使用的Storage Virtual Machine *省略for MetroCluster。*请参见[mcc-best]。	如果指定了SVM、则派生此参数 managementLIF
useCHAP	使用CHAP对iSCSI的ONTAP SAN驱动程序进行身份验证[布尔值]。将设置为 true、以便Trident配置双向CHAP并将其用作后端中给定SVM的默认身份验证。有关详细信息、请参见 "准备使用ONTAP SAN驱动程序配置后端" 。	false
chapInitiatorSecret	CHAP 启动程序密钥。如果需要、则为必需项 useCHAP=true	""
labels	要应用于卷的一组任意 JSON 格式的标签	""
chapTargetInitiatorSecret	CHAP 目标启动程序密钥。如果需要、则为必需项 useCHAP=true	""
chapUsername	入站用户名。如果需要、则为必需项 useCHAP=true	""
chapTargetUsername	目标用户名。如果需要、则为必需项 useCHAP=true	""
clientCertificate	客户端证书的 Base64 编码值。用于基于证书的身份验证	""
clientPrivateKey	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证	""
trustedCACertificate	受信任 CA 证书的 Base64 编码值。可选。用于基于证书的身份验证。	""

参数	说明	默认
username	与ONTAP 集群通信所需的用户名。用于基于凭据的身份验证。	""
password	与ONTAP 集群通信所需的密码。用于基于凭据的身份验证。	""
svm	要使用的 Storage Virtual Machine	如果指定了SVM、则派生此参数 managementLIF
storagePrefix	在 SVM 中配置新卷时使用的前缀。无法稍后修改。要更新此参数、您需要创建一个新的后端。	trident
aggregate	要配置的聚合（可选；如果设置了聚合，则必须将其分配给 SVM ）。对于 `ontap-nas-flexgroup`驱动程序、此选项将被忽略。如果未分配、则 可以使用任何可用聚合来配置FlexGroup卷。 在SVM中更新聚合后、该聚合将在Trident中自动更新、方法是轮询SVM、而无需重新启动Trident控制器。在Trident中配置了特定聚合以配置卷后、如果将该聚合重命名或移出SVM、则在轮询SVM聚合时、后端将在Trident中变为故障状态。您必须将聚合更改为SVM上的聚合、或者将其全部删除、以使后端恢复联机。	""
limitAggregateUsage	如果使用量超过此百分比，则配置失败。如果您使用的是Amazon FSx for NetApp ONTAP后端，请勿指定 limitAggregateUsage。提供的和 `vsadmin`不包含使用Trident检索聚合使用情况并对其进行限制所需的 `fsxadmin`权限。	"" （默认情况下不强制实施）
limitVolumeSize	如果请求的卷大小超过此值、则配置失败。此外、还会限制它为LUN管理的卷的大小上限。	"" （默认情况下不强制实施）
lunsPerFlexvol	每个 FlexVol 的最大 LUN 数，必须在 50 ， 200 范围内	100
debugTraceFlags	故障排除时要使用的调试标志。例如、除非您正在进行故障排除并需要详细的日志转储、否则不会使用 {"api"：false、"METHO"：true} 。	null

参数	说明	默认
useREST	用于使用 ONTAP REST API 的布尔参数。useREST`设置为时 `true，Trident使用ONTAP REST API与后端通信；设置为时 false，Trident使用ONTAP ZAPI调用与后端通信。此功能需要使用ONTAP 9.11.1及更高版本。此外、使用的ONTAP登录角色必须有权访问 ontap 应用程序。预定义的和角色可以满足这一 vsadmin 要求 cluster-admin 。从Trident 24.06版和ZAPI.151或更高版本开始、默认情况下会设置为 true；更useREST`改为 `false`以使用ONTAP 9 `useREST ONTAP调用。useREST 完全符合NVMe/TCP要求。	true 对于ONTAP 9.151或更高版本，否则 false。
sanType	用于为iSCSI、 nvme`NVMe/TCP或基于光纤通道的 `fcp`SCSI (FC)选择 `iscsi。**"FCP"(基于FC的SCSI)是Trident 24.10**版本中的一项技术预览功能。	`iscsi`如果为空
formatOptions	`formatOptions`用于指定命令的命令行参数、每当对卷进行格式化时、都会应用这些参数`mkfs`。这样、您可以根据偏好格式化卷。请确保指定与mkfs命令选项类似的格式选项，但不包括设备路径。示例："-E nobdiscard" • ontap-san `ontap-san-economy`仅支持和驱动程序。*	
limitVolumePoolSize	在LUS-SAN-Economy后端使用ONTAP时可要求的最大FlexVol大小。	"" （默认情况下不强制实施）
denyNewVolumePools	限制 `ontap-san-economy`后端创建新的FlexVol卷以包含其LUN。仅会使用已有的FlexVol配置新的PV.	

有关使用**formatOptions**的建议

Trident建议使用以下选项来加快格式化过程：

-E NODiscard：

- 保留、不要尝试在mkfs时间丢弃块(丢弃块最初在固态设备和稀疏/精简配置存储上很有用)。此选项将取代已弃用的选项"-K"、并适用于所有文件系统(xfs、ext3和ext4)。

用于配置卷的后端配置选项

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

参数	说明	默认
spaceAllocation	LUN 的空间分配	"正确"
spaceReserve	空间预留模式；"无"(精简)或"卷"(厚)	"无"
snapshotPolicy	要使用的 Snapshot 策略	"无"
qosPolicy	要为创建的卷分配的 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一。将QoS策略组与Trident结合使用需要使用ONTAP 9™8或更高版本。您应使用非共享QoS策略组、并确保此策略组分别应用于每个成分卷。共享QoS策略组会对所有工作负载的总吞吐量实施上限。	""
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一	""
snapshotReserve	为快照预留的卷百分比	如果为"none"、则为"0" snapshotPolicy、否则为""
splitOnClone	创建克隆时，从其父级拆分该克隆	"错误"
encryption	在新卷上启用NetApp卷加密(NVE)；默认为 false。要使用此选项，必须在集群上获得 NVE 的许可并启用 NVE 。如果在后端启用了NAE、则在Trident中配置的任何卷都将启用NAE。有关详细信息，请参阅："Trident如何与NVE和NAE配合使用"。	"错误"
luksEncryption	启用LUKS加密。请参阅 "使用Linux统一密钥设置(LUKS)"。NVMe/TCP不支持使用此类数据加密。	""
securityStyle	新卷的安全模式	unix
tieringPolicy	使用"无"的层策略	对于ONTAP 9.5 SVM-DR之前的配置、为"仅快照"
nameTemplate	用于创建自定义卷名称的模板。	""

卷配置示例

下面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'
```

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

对于定义的后端 `snapshotReserve`，Trident将按如下所示计算卷的大小：

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

1.1是Trident为容纳LUN元数据而向FlexVol额外增加的10%。对于 `snapshotReserve`= 5%、PVC请求= 5 GiB、则卷总大小为5.79 GiB、可用大小为5.5 GiB。此 `volume show`命令应显示类似于以下示例的结果：

目前，调整大小是对现有卷使用新计算的唯一方法。

最低配置示例

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

> 如果您将Amazon FSx on NetApp ONTAP与结合使用、建议您为Trident指定DNS名称、而不是IP地址。

ONTAP SAN示例

这是使用驱动程序的基本配置 `ontap-san`。

```
---
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>
```

ONTAP SAN经济性示例

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

1. 示例

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

要进行无缝切换和切回、请使用指定SVM `managementLIF`、并省略 `dataLIF`和 `svm`参数。例如：

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

基于证书的身份验证示例

在此基本配置示例中 `clientCertificate`， `clientPrivateKey`、和 `trustedCACertificate`(如果使用受信任CA，则为可选)分别填充 `backend.json`并采用base64编码的客户端证书值、专用密钥值和受信任CA证书值。

```
---
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

以下示例将创建一个后端，并 `useCHAP`将设置为 `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经济性**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

您必须在ONTAP后端为SVM配置NVMe。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
```

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}}_{{.volume.R
equestName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
"{{.volume.Namespace}}_{{.volume.RequestName}}"}
```

formatOptions <code> ONTAP－san－</code>驱动程序示例

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

虚拟池后端示例

在这些示例后端定义文件中、会为所有存储池设置特定默认值、例如 `spaceReserve`、在none、`spaceAllocation`在false和 `encryption`在false。虚拟池在存储部分中进行定义。

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

在这些示例中、某些存储池会设置自己的、 `spaceAllocation`和 `encryption`值、而某些存储 `spaceReserve`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经济性示例

```
---
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 **StorageClasses**

以下StorageClass定义参见[虚拟池后端示例]。通过 `parameters.selector`字段、每个StorageClass都会调用可用于托管卷的虚拟池。卷将在选定虚拟池中定义各个方面。

- `protection-gold`StorageClass将映射到后端的第一个虚拟池 `ontap-san`。这是唯一提供金牌保护的池。

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

- `protection-not-gold`StorageClass将映射到后端的第二个和第三个虚拟池 `ontap-san。只有这些池提供的保护级别不是gold。

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

- `app-mysqldb`StorageClass将映射到后端的第三个虚拟池 `ontap-san-economy。这是为mysqldb类型的应用程序提供存储池配置的唯一池。

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

- `protection-silver-creditpoints-20k`StorageClass将映射到后端的第二个虚拟池 `ontap-san。这是唯一提供银牌保护和20000个信用点的池。

```
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"
```

- `creditpoints-5k`StorageClass将映射到后端的第三个虚拟池 `ontap-san`和后端的第四个虚拟池 `ontap-san-economy`。这是唯一一款信用点数为5000的池产品。

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

- `my-test-app-sc`StorageClass将映射到 `testAPP`驱动程序 `sanType: nvme`中的虚拟池 `ontap-san`。这是唯一的池选项 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、并确保满足存储要求。

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驱动程序详细信息

Trident提供了以下NAS存储驱动程序来与ONTAP集群进行通信。支持的访问模式包括：*ReadWriteOnce*(RWO)、*ReadOnlyMany*(ROX)、*ReadWriteMany*(rwx)、*ReadWriteOncePod*(RWOP)。

驱动程序	协议	卷模式	支持的访问模式	支持的文件系统
`ontap-nas`	NFS SMB	文件系统	Rwo、ROX、rwx、RWOP	""、 `nfs`、 `smb`
`ontap-nas-economy`	NFS SMB	文件系统	Rwo、ROX、rwx、RWOP	""、 `nfs`、 `smb`
`ontap-nas-flexgroup`	NFS SMB	文件系统	Rwo、ROX、rwx、RWOP	""、 `nfs`、 `smb`

- 只有当永久性卷使用量计数预期大于时才使用 `ontap-san-economy`"支持的ONTAP卷限制"。
- `ontap-nas-economy`仅当永久性卷使用量计数预计高于且 `ontap-san-economy`无法使用驱动程序时才"支持的ONTAP卷限制"使用。
- 如果您预计需要数据保护、灾难恢复或移动性、请勿使用 `ontap-nas-economy`。

用户权限

Trident应以ONTAP或SVM管理员身份运行、通常使用集群用户 `vsadmin`或SVM用户、或者使用 `admin`具有相同角色的其他名称的用户。

对于Amazon FSx for NetApp ONTAP部署、Trident应使用集群用户 `vsadmin`或SVM用户以ONTAP或SVM管理员身份运行、或者使用具有相同角色的其他名称的用户运行 `fsxadmin`。此 `fsxadmin`用户只能有限地替代集群管理员用户。

如果使用 `limitAggregateUsage`参数、则需要集群管理员权限。将Amazon FSx for NetApp ONTAP与Trident结合使用时、 `limitAggregateUsage`参数不适用于 `vsadmin`和 `fsxadmin`用户帐户。如果指定此参数，配置操作将失败。

虽然可以在ONTAP中创建一个可以由三端驱动程序使用的限制性更强的角色、但我们不建议这样做。大多数新版本的 Trident 都会调用需要考虑的其他 API ，从而使升级变得困难且容易出错。

准备使用ONTAP NAS驱动程序配置后端

了解使用ONTAP NAS驱动程序配置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要求至少为SVM分配一个聚合。
- 您可以运行多个驱动程序，并创建指向其中一个驱动程序的存储类。例如，您可以配置一个使用驱动程序的Gold类和一个使用驱动程序的Bronze `ontap-nas-economy`类 `ontap-nas`。
- 所有Kubernetes工作节点都必须安装适当的NFS工具。"此处"有关详细信息、请参见。

- Trident仅支持挂载到Windows节点上运行的Pod的SMB卷。有关详细信息、请参见 准备配置SMB卷 。

对**ONTAP**后端进行身份验证

Trident提供了两种对ONTAP后端进行身份验证的模式。

- 基于凭据：此模式需要对ONTAP后端具有足够的权限。建议使用与预定义的安全登录角色关联的帐户、例如 `admin`或、 `vsadmin`以确保与ONTAP版本最大程度地兼容。
- 基于证书：此模式需要在后端安装证书、Trident才能与ONTAP集群进行通信。此处，后端定义必须包含客户端证书，密钥和可信 CA 证书的 Base64 编码值（如果使用）（建议）。

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

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

启用基于凭据的身份验证

Trident需要SVM范围/集群范围的管理员的凭据才能与ONTAP后端进行通信。建议使用标准的预定义角色，如 `admin`或 `vsadmin`。这样可以确保与未来ONTAP版本的正向兼容性、这些版本可能会公开未来Trident版本要使用的功能API。可以创建自定义安全登录角色并将其用于Trident、但不建议这样做。

后端定义示例如下所示：

YAML

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

JSON

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

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

启用基于证书的身份验证

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

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

典型的工作流包括以下步骤。

步骤

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

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

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

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

3. 在 ONTAP 集群上安装客户端证书和密钥（从步骤 1 开始）。

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

4. 确认ONTAP安全登录角色支持 `cert`身份验证方法。

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

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

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

6. 使用 Base64 对证书，密钥和可信 CA 证书进行编码。

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

7. 使用从上一步获得的值创建后端。

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

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

更新身份验证方法或轮换凭据

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

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

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

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

更新后端不会中断对已创建卷的访问，也不会影响在之后建立的卷连接。后端更新成功表示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Privileges最低的ONTAP集群角色、这样就不必使用ONTAP管理员角色在Trident中执行操作。如果在Trident后端配置中包含用户名、则Trident将使用您创建的ONTAP集群角色来执行操作。

有关创建Trident自定义角色的详细信息、请参见"Trident自定义角色生成器"。

使用**ONTAP**命令行界面

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 **System Manager**

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*Cluster > Settings*。

      (或)要在SVM级别创建自定义角色、请选择*存储> Storage VM `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. 定义角色的规则，然后单击*Save*。
2. 将角色映射到**Trident user**：+在*Users and Roles*页面上执行以下步骤：
   a. 在*用户*下选择添加图标*+*。
   b. 选择所需的用户名，然后在下拉菜单中为*rouser*选择一个角色。
   c. 单击 * 保存 * 。

有关详细信息、请参见以下页面：

- "用于管理ONTAP的自定义角色"或"定义自定义角色"
- "使用角色和用户"

管理 **NFS** 导出策略

Trident使用NFS导出策略控制对其配置的卷的访问。

使用导出策略时、Trident提供了两个选项：

- Trident可以动态管理导出策略本身；在此操作模式下、存储管理员可以指定一个表示可接受IP地址的CIDR块列表。Trident会在发布时自动将这些范围内的适用节点IP添加到导出策略中。或者、如果未指定CIDR、则在要发布卷的节点上找到的所有全局范围单播IP都将添加到导出策略中。
- 存储管理员可以手动创建导出策略和添加规则。除非在配置中指定了其他导出策略名称、否则Trident将使用默认导出策略。

动态管理导出策略

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

使用动态导出策略时、请勿使用网络地址转换(Network Address Translation、NAT)。使用NAT时、存储控制器会看到前端NAT地址、而不是实际IP主机地址、因此、如果在导出规则中找不到匹配项、则会拒绝访问。

在Trident 24.10中、 `ontap-nas`存储驱动程序将继续与早期版本一样工作；ONTAP NAS驱动程序未进行任何更改。在Trident 24.10中、只有 `ontap-nas-economy`存储驱动程序具有基于卷的粒度访问控制。

示例

必须使用两个配置选项。下面是一个后端定义示例：

```
---
version: 1
storageDriverName: ontap-nas-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、您必须确保SVM中的根接合具有先前创建的导出策略、并具有允许节点CIDR块的导出规则(例如默认导出策略)。始终遵循NetApp建议的最佳实践、将SVM专用于Trident。

以下是使用上述示例对此功能的工作原理进行的说明：

- `autoExportPolicy`设置为 `true`。这表示Trident会为SVM的使用此后端配置的每个卷创建一个导出策略 `svm1`、并使用地址块处理规则的添加和删除 `autoexportCIDRs`。在将卷连接到节点之前、此卷会使用一个空导出策略、此策略不带任何规则来防止对该卷进行不必要的访问。将卷发布到节点后、Trident会创建一个与指定CIDR块中包含节点IP的底层qtree同名的导出策略。这些IP也会添加到父FlexVol使用的导出策略中。

- 例如：
  - 后端UUID 403b5326/8482-40db-96d0-d83fb3f4daec
  - `autoExportPolicy`将设置为 `true`
  - 存储前缀 `trident`
  - pvc UUID a79bcf5f-7b6d-4a40-9876- e2551f159c1c
  - 名为svm_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c的qtree会为名为的FlexVol创建一个导出策略、为名为的qtree创建一个导出策略、`trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`并在Trident上创建`trident-403b5326-8482-40db96d0-d83fb3f4daec`一个名为的空导出策略`trident_empty`。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。空导出策略将由所有未附加的卷重复使用。

- `autoExportCIDRs`包含地址块列表。此字段为可选字段，默认为 "0.0.0.0/0 ， "：： /0" 。如果未定义、则Trident会添加在具有出版物的工作节点上找到的所有全局范围单播地址。

在此示例中、 `192.168.0.0/24`提供了地址空间。这表示属于此地址范围且发布内容的Kub联网 节点IP将添加到Trident创建的导出策略中。当Trident注册运行该功能的节点时，它将检索该节点的IP地址，并根据中提供的地址块对其进行检查 `autoExportCIDRs`。发布时，在筛选IP之后，Trident将为要发布到的节点的客户端IP创建导出策略规则。

您可以在创建后端后为后端更新 `autoExportPolicy`和 `autoExportCIDRs`。您可以为自动管理的后端附加新的 CIDR ，也可以删除现有的 CIDR 。删除 CIDR 时请务必小心，以确保现有连接不会断开。您也可以选择对后端禁用 `autoExportPolicy`、并回退到手动创建的导出策略。这需要在后端配置中设置 `exportPolicy`参数。

在Trident创建或更新后端后、您可以使用或相应的 `tridentbackend`CRD检查后端 `tridentctl`：

```
./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
```

删除节点后、Trident会检查所有导出策略以删除与该节点对应的访问规则。通过从受管后端的导出策略中删除此节点IP、Trident可防止恶意挂载、除非集群中的新节点重复使用此IP。

对于以前存在的后端、使用更新后端 `tridentctl update backend`可确保Trident自动管理导出策略。这样会根据需要创建两个新的导出策略、并以后端的UUID和qtree名称命名。后端上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。如果重新创建后端，则会将其视为新的后端，并会创建新的导出策略。

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

准备配置**SMB**卷

只需稍作准备、即可使用驱动程序配置SMB卷 `ontap-nas`。

> 您必须在SVM上同时配置NFS和SMB/CCIFS协议、才能为内部ONTAP创建 `ontap-nas-economy`SMB卷。如果未能配置其中任一协议、则发生原因 SMB卷创建将失败。

> `autoExportPolicy`SMB卷不支持。

开始之前

在配置SMB卷之前、您必须满足以下条件。

- 一个Kubernetes集群、其中包含一个Linux控制器节点以及至少一个运行Windows Server 2022的Windows工作节点。Trident仅支持挂载到Windows节点上运行的Pod的SMB卷。
- 至少一个包含Active Directory凭据的Trident密钥。生成密钥 `smbcreds`：

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

- 配置为Windows服务的CSI代理。要配置 `csi-proxy`，请参阅"GitHub：CSI代理"或了解在Windows上运行的Kubornetes"GitHub：适用于Windows的CSI代理"节点。

步骤

1. 对于内部ONTAP、您可以选择创建SMB共享、也可以选择Trident为您创建一个共享。

   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管理员共享：使用"Microsoft管理控制台"共享文件夹管理单元或使用ONTAP命令行界面。要使用ONTAP 命令行界面创建SMB共享、请执行以下操作：

   a. 如有必要，为共享创建目录路径结构。

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

   b. 创建与指定SVM关联的SMB共享：

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

   c. 验证是否已创建共享：

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

      有关完整详细信息、请参见"创建 SMB 共享"。

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后端配置选项的信息，请参阅"适用于ONTAP 的FSX配置选项和示例"。

参数	说明	示例
smbShare	您可以指定以下选项之一：使用Microsoft管理控制台或ONTAP命令行界面创建的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进行通用共享访问。对于内部ONTAP、此参数是可选的。此参数对于Amazon FSx for ONTAP后端为必填项、不能为空。	smb-share
nasType	*必须设置为 smb.*如果为空，则默认为 nfs。	smb
securityStyle	新卷的安全模式。对于**SMB**卷，必须设置为 **ntfs**` 或 `**mixed**。	`ntfs`或 `mixed`SMB卷
unixPermissions	新卷的模式。对于SMB卷、必须留空。	""

ONTAP NAS配置选项和示例

了解如何在Trident安装中创建和使用ONTAP NAS驱动程序。本节提供了将后端映射到StorageClasses的后端配置示例和详细信息。

后端配置选项

有关后端配置选项，请参见下表：

参数	说明	默认
version		始终为 1
storageDriverName	存储驱动程序的名称	"ONTAP NAS "、"ONTAP NAS经济"、"ONTAP NAS灵活组"、"ONTAP SAN "、"ONTAP SAN经济"
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是使用IPv6标志安装的、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]。有关无缝MetroCluster切换的信息，请参见[mcc-best]。	"10.0.0.1 "， "2001： 1234：abcd：：： fefe] "
dataLIF	协议 LIF 的 IP 地址。建议指定 dataLIF。如果不提供此参数、则Trident将从SVM提取数据LUN。您可以指定用于NFS挂载操作的完全限定域名(FQDN)、从而可以创建循环DNS、以便在多个数据LIF之间实现负载平衡。可以在初始设置后更改。请参阅 。如果Trident是使用IPv6标志安装的、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]。*省略MetroCluster。*请参见[mcc-best]。	指定的地址或派生自SVM (如果未指定)(不建议)

参数	说明	默认
`svm`	要使用的Storage Virtual Machine *省略for MetroCluster。*请参见[mcc-best]。	如果指定了SVM、则派生此参数 `managementLIF`
`autoExportPolicy`	启用自动创建和更新导出策略[布尔值]。使用 `autoExportPolicy`和 `autoExportCIDRs`选项、Trident可以自动管理导出策略。	false
`autoExportCIDRs`	用于筛选KubeNet节点IP的CIDR列表(启用时)。 `autoExportPolicy`使用 `autoExportPolicy`和 `autoExportCIDRs`选项、Trident可以自动管理导出策略。	["0.0.0.0/0、"：：：/0"]`
`labels`	要应用于卷的一组任意 JSON 格式的标签	""
`clientCertificate`	客户端证书的 Base64 编码值。用于基于证书的身份验证	""
`clientPrivateKey`	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证	""
`trustedCACertificate`	受信任 CA 证书的 Base64 编码值。可选。用于基于证书的身份验证	""
`username`	用于连接到集群 /SVM 的用户名。用于基于凭据的身份验证	
`password`	连接到集群 /SVM 的密码。用于基于凭据的身份验证	
`storagePrefix`	在 SVM 中配置新卷时使用的前缀。设置后无法更新 如果使用的ONTAP是包含24个或更多字符的storagePrefix、则qtrees不会嵌入存储前缀、但会显示在卷名称中。	"三级联"
`aggregate`	要配置的聚合（可选；如果设置了聚合，则必须将其分配给 SVM ）。对于 `ontap-nas-flexgroup`驱动程序、此选项将被忽略。如果未分配、则 可以使用任何可用聚合来配置FlexGroup卷。 在SVM中更新聚合后、该聚合将在Trident中自动更新、方法是轮询SVM、而无需重新启动Trident控制器。在Trident中配置了特定聚合以配置卷后、如果将该聚合重命名或移出SVM、则在轮询SVM聚合时、后端将在Trident中变为故障状态。您必须将聚合更改为SVM上的聚合、或者将其全部删除、以使后端恢复联机。	""
`limitAggregateUsage`	如果使用量超过此百分比，则配置失败。* 不适用于适用于 ONTAP 的 Amazon FSx *	"" （默认情况下不强制实施）

参数	说明	默认
FlexgroupGroup GroupRegateList	要配置的聚合列表(可选；如果已设置、则必须将其分配给SVM)。分配给SVM的所有聚合均用于配置FlexGroup卷。支持* ONTAP－NAS－FlexGroup－Storage驱动程序。 在SVM中更新聚合列表后、此列表将在Trident中自动更新、方法是轮询SVM、而无需重新启动Trident控制器。在Trident中配置特定聚合列表以配置卷后、如果聚合列表重命名或移出SVM、则在轮询SVM聚合时、后端将在Trident中变为故障状态。您必须将聚合列表更改为SVM上的聚合列表、或者将其全部删除、以使后端恢复联机。	""
`limitVolumeSize`	如果请求的卷大小超过此值、则配置失败。此外、还会限制它为qtrees管理的卷的大小上限、并且此`qtreesPerFlexvol`选项允许自定义每个FlexVol的qtrees的最大数量。	"" （默认情况下不强制实施）
`debugTraceFlags`	故障排除时要使用的调试标志。例如、除非您正在进行故障排除并需要详细的日志转储、否则不会使用 {"api"：false、"METHO"：true} `debugTraceFlags`。	空
`nasType`	配置NFS或SMB卷创建。选项为 `nfs`、 `smb`或null。默认情况下、将设置为空会将NFS卷设置为空。	`nfs`
`nfsMountOptions`	NFS挂载选项的逗号分隔列表。通常会在存储类中为Kubnetes-永久性 卷指定挂载选项、但如果在存储类中未指定挂载选项、则Trident将回退到使用存储后端配置文件中指定的挂载选项。如果在存储类或配置文件中未指定挂载选项、则Trident不会在关联的永久性卷上设置任何挂载选项。	""
`qtreesPerFlexvol`	每个 FlexVol 的最大 qtree 数，必须在 50 ， 300 范围内	"200"
`smbShare`	您可以指定以下选项之一：使用Microsoft管理控制台或ONTAP命令行界面创建的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进行通用共享访问。对于内部ONTAP、此参数是可选的。此参数对于Amazon FSx for ONTAP后端为必填项、不能为空。	`smb-share`

参数	说明	默认
useREST	用于使用 ONTAP REST API 的布尔参数。 useREST` 设置为时 `true，Trident使用ONTAP REST API与后端通信；设置为时 false，Trident使用ONTAP ZAPI调用与后端通信。此功能需要使用ONTAP 9.11.1及更高版本。此外、使用的ONTAP登录角色必须有权访问 ontap 应用程序。预定义的和角色可以满足这一 vsadmin 要求 cluster-admin 。从Trident 24.06版和ZAPI.151或更高版本开始、默认情况下会设置为 true；更 useREST`改为 `false`以使用ONTAP 9 `useREST ONTAP调用。	true 对于ONTAP 9.151或更高版本，否则 false。
limitVolumePoolSize	在qtree-NAS ONTAP经济型后端使用qtrees时可请求的最大FlexVol大小。	""（默认情况下不强制实施）
denyNewVolumePools	限制 `ontap-nas-economy`后端创建新的FlexVol卷以包含其qtrees。仅会使用已有的FlexVol配置新的PV.	

用于配置卷的后端配置选项

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

参数	说明	默认
spaceAllocation	qtrees的空间分配	"正确"
spaceReserve	空间预留模式；"无"(精简)或"卷"(厚)	"无"
snapshotPolicy	要使用的 Snapshot 策略	"无"
qosPolicy	要为创建的卷分配的 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一	""
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。选择每个存储池 / 后端的 qosPolicy 或 adaptiveQosPolicy 之一。不受 ontap-nas-economy.	""
snapshotReserve	为快照预留的卷百分比	如果为"none"、则为"0" snapshotPolicy、否则为""
splitOnClone	创建克隆时，从其父级拆分该克隆	"错误"
encryption	在新卷上启用NetApp卷加密(NVE)；默认为 false。要使用此选项，必须在集群上获得 NVE 的许可并启用 NVE 。如果在后端启用了NAE、则在Trident中配置的任何卷都将启用NAE。有关详细信息，请参阅："Trident如何与NVE和NAE配合使用"。	"错误"
tieringPolicy	使用"无"的层策略	对于ONTAP 9.5 SVM-DR之前的配置、为"仅快照"
unixPermissions	新卷的模式	"777"表示NFS卷；空(不适用)表示SMB卷

参数	说明	默认
snapshotDir	控制对目录的访问 .snapshot	对于NFSv4、为"TRUE"；对于NFSv3、为"false"
exportPolicy	要使用的导出策略	default
securityStyle	新卷的安全模式。NFS支持 `mixed`和 `unix`安全模式。SMB支持 `mixed`和 `ntfs`安全模式。	NFS默认值为 unix。SMB默认值为 ntfs。
nameTemplate	用于创建自定义卷名称的模板。	""

将QoS策略组与Trident结合使用需要使用ONTAP 9™8或更高版本。您应使用非共享QoS策略组、并确保此策略组分别应用于每个成分卷。共享QoS策略组会对所有工作负载的总吞吐量实施上限。

卷配置示例

下面是一个定义了默认值的示例：

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

对于 ontap-nas`和 `ontap-nas-flexgroups，Trident现在使用新的计算方法来确保使用snapshotReserve百分比和pvc正确调整FlexVol的大小。当用户请求PVC时、Trident会使用新计算方法创建具有更多空间的原始FlexVol。此计算可确保用户在 PVC 中收到所请求的可写空间，而不是小于所请求的空间。在 v21.07 之前，如果用户请求 PVC （例如， 5GiB ），并且 snapshotReserve 为 50% ，则只会获得 2.5 GiB 的

可写空间。这是因为用户请求的是整个卷、并且 `snapshotReserve`是其中的一个百分比。在Trident 21.07中、用户请求的是可写空间、Trident将该数字定义 `snapshotReserve`为整个卷的百分比。这不适用于 `ontap-nas-economy`。请参见以下示例以了解其工作原理：

计算方法如下：

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

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

先前安装的现有后端将在升级Trident时按照上文所述配置卷。对于在升级之前创建的卷，您应调整其卷的大小，以便观察到所做的更改。例如、使用较早版本的2GiB PVC `snapshotReserve=50`会导致卷提供1GiB的可写空间。例如，将卷大小调整为 3GiB 可为应用程序在一个 6 GiB 卷上提供 3GiB 的可写空间。

最低配置示例

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

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

ONTAP NAS经济性示例

```
---
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，以避免在期间切换和切回后手动更新后端定义"SVM复制和恢复"。

要进行无缝切换和切回、请使用指定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

这是一个最小的后端配置示例。 clientCertificate clientPrivateKey`和 `trustedCACertificate(如果使用受信任CA，则为可选)将分别填充 `backend.json`并采用base64编码的客户端证书值、私钥值和受信任CA证书值。

```
---
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_
```

自动导出策略示例

此示例介绍如何指示Trident使用动态导出策略自动创建和管理导出策略。这对于和 ontap-nas-flexgroup`驱动程序是相同的 `ontap-nas-economy。

```
---
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

此示例显示了 `managementLIF`如何使用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
```

Amazon FSx for ONTAP使用SMB卷示例

`smbShare`对于使用SMB卷的FSx for ONTAP、参数是必需的。

```
---
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}}_{{.volume.R
equestName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
"{{.volume.Namespace}}_{{.volume.RequestName}}"}
```

虚拟池后端示例

在下面显示的示例后端定义文件中、为所有存储池设置了特定默认值、例如 `spaceReserve`、在none、`spaceAllocation`在false和 `encryption`在false。虚拟池在存储部分中进行定义。

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

在这些示例中、某些存储池会设置自己的、 `spaceAllocation`和 `encryption`值、而某些存储 `spaceReserve`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经济性示例

```
---
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 **StorageClasses**

以下StorageClass定义请参见[虚拟池后端示例]。通过 `parameters.selector`字段、每个StorageClass都会调用可用于托管卷的虚拟池。卷将在选定虚拟池中定义各个方面。

- `protection-gold`StorageClass将映射到后端的第一个和第二个虚拟池 `ontap-nas-flexgroup`。这些池是唯一提供金牌保护的池。

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

- `protection-not-gold`StorageClass将映射到后端的第三个和第四个虚拟池 `ontap-nas-flexgroup`。这些池是唯一提供黄金级以外保护级别的池。

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

- `app-mysqldb`StorageClass将映射到后端的第四个虚拟池 `ontap-nas`。这是为mysqldb类型的应用程序提供存储池配置的唯一池。

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

- protection-silver-creditpoints-20k`StorageClass将映射到后端的第三个虚拟池 `ontap-nas-flexgroup。这是唯一提供银牌保护和20000个信用点的池。

```
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"
```

- creditpoints-5k`StorageClass将映射到后端的第三个虚拟池 `ontap-nas`和后端的第二个虚拟池 `ontap-nas-economy。这是唯一一款信用点数为5000的池产品。

```
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、并确保满足存储要求。

在初始配置后更新 dataLIF

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

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

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

适用于 NetApp ONTAP 的 Amazon FSX

将Trident与Amazon FSx for NetApp ONTAP结合使用

"适用于 NetApp ONTAP 的 Amazon FSX"是一项完全托管的AWS服务、支持客户启动和运行由NetApp ONTAP存储操作系统提供支持的文件系统。借助适用于ONTAP 的FSx、您可以利用您熟悉的NetApp功能、性能和管理功能、同时利用在AWS上存储数据的简便性、灵活性、安全性和可扩展性。FSX for ONTAP 支持ONTAP 文件系统功能和管理API。

您可以将Amazon FSx for NetApp ONTAP文件系统与Trident进行集成、以确保在Amazon Elic Kubelnetes Service (EKS)中运行的Kubelnetes集群可以配置ONTAP支持的块和文件永久性卷。

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

通过将Trident与Amazon FSx for NetApp ONTAP结合使用、您可以确保在Amazon Elic Kubelnetes Service (EKS)中运行的Kubelnetes集群可以配置ONTAP支持的块和文件永久性卷。

要求

除了"Trident要求"，要将FSx for ONTAP与Trident集成，您还需要：

- 已安装的现有Amazon EKS集群或自行管理的Kubornetes集群 `kubectl`。
- 可从集群的工作节点访问的现有Amazon FSx for NetApp ONTAP文件系统和Storage Virtual Machine (SVM)。
- 准备用于的工作节点"NFS或iSCSI"。

根据您的EKS AMI类型、确保按照Amazon Linux和Ubuntu (AMI)所需的节点准备步骤进行操作 "Amazon Machine 映像"。

注意事项

- SMB卷：
  - 仅使用驱动程序支持SMB卷 `ontap-nas`。
  - Trident EKS加载项不支持SMB卷。
  - Trident仅支持挂载到Windows节点上运行的Pod的SMB卷。有关详细信息、请参见 "准备配置SMB卷" 。
- 在Trident 24.02之前的版本中、Trident无法删除在已启用自动备份的Amazon FSx文件系统上创建的卷。要在Trident 24.02或更高版本中防止此问题，请在AWS FSx for ONTAP的后端配置文件中指定 `fsxFilesystemID`、AWS、 `apikey`AWS `apiRegion`和AWS `secretKey`。

如果要为Trident指定IAM角色，则可以省略为Trident明确指定 `apiRegion`、 `apiKey`和 `secretKey`字段。有关详细信息，请参阅 "适用于ONTAP 的FSX配置选项和示例"。

身份验证

Trident提供两种身份验证模式。

- 基于凭据(建议)：将凭据安全地存储在AWS机密管理器中。您可以使用文件系统的用户、也可以使用 `fsxadmin` `vsadmin` 为SVM配置的用户。

Trident应以SVM用户身份运行、或者以具有相同角色的其他名称的用户身份运行 `vsadmin` 。Amazon FSx for NetApp ONTAP的某个 `fsxadmin`用户只能有限地替代`ONTAP` `admin`集群用户。强烈建议将与`Trident`结合使用 `vsadmin`。

- 基于证书：Trident将使用SVM上安装的证书与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、请参阅适用于您的驱动程序类型的身份验证：

- "ONTAP NAS身份验证"
- "ONTAP SAN身份验证"

测试过的**Amazon**计算机映像**(AMI)**

EKS集群支持各种操作系统、但AWS已针对容器和EKS优化了某些Amazon计算机映像(AMI)。以下AMI已通过Trident 24.10的测试。

AMI	NAS	NAS经济型	SAN	SAN经济型
AL2023_x86_64_STANDARD	是	是	是	是
AL2_x86_64	是	是	是**	是**
BOTTLEROCKET_x86_64	是 *	是	不适用	不适用
AL2023_ARM_64_STANDARD	是	是	是	是
AL2_ARM_64	是	是	是**	是**
BOTTLEROCKET_ARM_64	是 *	是	不适用	不适用

- *必须在挂载选项中使用"nolock"。
- **在不重新启动节点的情况下，无法删除PV

如果此处未列出您所需的AMI、并不表示它不受支持、而只是表示它尚未经过测试。此列表可作为已知有效的AMI的指南。

使用以下项执行的测试：

- EKS版本：1.30
- 安装方法：Helm和作为AWS插件
- 对于NAS、我们同时测试了NFSv3和NFSv4.1。
- 对于SAN、测试的是仅iSCSI、而不是NVMe-oF。

执行的测试：

- 创建：存储类、PVC、POD
- 删除：POD、PVC (常规、qtree/LUN—经济型、NAS与AWS备份)

了解更多信息

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

创建IAM角色和AWS机密

您可以通过作为AWS IAM角色进行身份验证(而不是提供显式AWS凭据)来配置Kubbernetes Pod以访问AWS资源。

要使用AWS IAM角色进行身份验证、您必须使用EKS部署Kubenetes集群。

创建AWS机密管理器密钥

以下示例将创建一个AWS机密管理器密钥、用于存储Trident CSI凭据：

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

创建IAM策略

以下示例将使用AWS命令行界面创建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
Secret manager"
```

策略JSON文件：

```
policy.json:
{
    "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"
}
```

为服务帐户创建**IAM**角色

AWS命令行界面

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

信任关系**.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"
            }
        }
    }
   ]
}
```

更新文件中的以下值 `trust-relationship.json`：

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

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

将**IAM**角色附加到**IAM**策略：

创建角色后、使用以下命令将在上述步骤中创建的策略附加到此角色：

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

验证**OCD**提供程序是否关联：

验证OIDC提供程序是否已与集群关联。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
```

使用以下命令将IAM OIDC与集群关联：

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

eksc

以下示例将在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Kubelnetes中适用于NetApp ONTAP的Amazon FSx存储管理、使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：

- 掌舵
- EKS附加项

如果要使用快照功能、请安装CSI快照控制器加载项。有关详细信息、请参见 "为CSI卷启用快照功能" 。

通过舵安装**Trident**

1. 下载Trident安装程序包

   Trident安装程序包包含部署Trident Operator和安装Trident所需的一切。从GitHub上的"Assets"部分下载并提取最新版本的Trident安装程序。

   ```
   wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
   installer-24.10.0.tar.gz
   tar -xf trident-installer-24.10.0.tar.gz
   cd trident-installer/helm
   ```

2. 使用以下环境变量设置*云提供程序*和*云身份*标志的值：

以下示例将安装Trident并将标志设置 `cloud-provider`为 `$CP`、和 `cloud-identity` `$CI`：

```
helm install trident trident-operator-100.2410.0.tgz --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`命令查看安装详细信息、例如名称、命名空间、图表、状态、应用程序版本和修订版号。

```
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-100.2410.0    24.10.0
```

通过**EKS**插件安装**Trident**

Trident EKS加载项包括最新的安全修补程序和错误修复、并已通过AWS验证、可与Amazon EKS配合使用。通过EKS加载项、您可以始终确保Amazon EKS集群安全稳定、并减少安装、配置和更新加载项所需的工作量。

前提条件

在配置适用于AWS EKS的Trident加载项之前、请确保满足以下条件：

- 具有附加订阅的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 ARM 2 (AL2_x86_64)或Amazon Linux 2 ARM (AL2_AMAZON_64)
- 节点类型：AMD或ARM
- 现有Amazon FSx for NetApp ONTAP文件系统

启用适用于**AWS**的**Trident**加载项

eksc

以下示例命令用于安装Trident EKS加载项：

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> \
    --service-account-role-arn
arn:aws:iam::<account_id>:role/<role_name> --force
```

管理控制台

1. 打开Amazon EKS控制台，网址为 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，单击*群集*。
3. 单击要为其配置NetApp Trident CSI加载项的集群的名称。
4. 单击*Add-ones*，然后单击*Get more add-ones*。
5. 在*Select add-ons*页上，执行以下操作：
   a. 在AWS Marketplace EKS-addons部分中、选中* Trident by NetApp *复选框。
   b. 单击 * 下一步 * 。
6. 在“*配置选定的附加项*设置”页面上，执行以下操作：
   a. 选择要使用的*版本*。
   b. 对于*Select IAM Role*，保留为*not set*。
   c. 展开*可选配置设置*，遵循*附加配置架构*，并将*配置值*部分中的configurationvalues*参数设置为您在上一步中创建的roole-arn (值格式应为： `eks.amazonaws.com/role-arn: arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`)。如果您为冲突解决方法选择覆盖、则可以使用Amazon EKS附加设置覆盖现有附加项的一个或多个设置。如果未启用此选项、并且与现有设置存在冲突、则操作将失败。您可以使用生成的错误消息来解决冲突。在选择此选项之前、请确保Amazon EKS附加组件未管理您需要自行管理的设置。
7. 选择“下一步”。
8. 在*Review and add*页上，选择*Cree*。

   加载项安装完成后、您将看到已安装的加载项。

AWS命令行界面

1. 创建 `add-on.json` 文件：

```
add-on.json
{

   "clusterName": "<eks-cluster>",
   "addonName": "netapp_trident-operator",
   "addonVersion": "v24.10.0-eksbuild.1",
   "serviceAccountRoleArn": "<arn:aws:iam::123456:role/astratrident-
role>",
   "configurationValues": "{"cloudIdentity":
"'eks.amazonaws.com/role-arn:
<arn:aws:iam::123456:role/astratrident-role>'",
   "cloudProvider": "AWS"}"
}
```

2. 安装Trident EKS附加软件"

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

更新Trident EKS加载项

eksc

- 检查FSxN Trident CSI加载项的当前版本。请替换 `my-cluster` 为您的集群名称。
  `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    v24.10.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- 将此加载项更新到上一步输出中的update下返回的版本。
  `eksctl update addon --name netapp_trident-operator --version v24.10.0-eksbuild.1 --cluster my-cluster --force`

  如果您删除了该 `--force` 选项、并且任何Amazon EKS附加设置与您的现有设置冲突、则更新Amazon EKS附加设置将失败；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附加组件不会管理您需要管理的设置、因为这些设置会被此选项覆盖。有关此设置的其他选项的详细信息，请参见 "插件"。有关Amazon EKS Kubenetes字段管理的详细信息，请参阅 "Kubbernetes现场管理"。

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，单击*群集*。
3. 单击要更新的NetApp Trident CSI加载项的集群的名称。
4. 单击*Add-ones*选项卡。
5. 单击Trident by NetApp，然后单击*Edit*。
6. 在“*按NetApp配置Trident”页上，执行以下操作：
   a. 选择要使用的*版本*。
   b. 展开*可选配置设置*并根据需要进行修改。
   c. 单击 * 保存更改 * 。

AWS命令行界面

以下示例将更新EKS加载项：

```
aws eks update-addon --cluster-name my-cluster netapp_trident-operator
vpc-cni --addon-version v24.6.1-eksbuild.1 \
    --service-account-role-arn arn:aws:iam::111122223333:role/role-name
--configuration-values '{}' --resolve-conflicts --preserve
```

您可以通过两种方式删除Amazon EKS附加项：

- 保留集群上的附加软件–此选项将删除Amazon EKS对任何设置的管理。此外、它还会使Amazon EKS无法通知您更新、并在您启动更新后自动更新Amazon EKS附加项。但是、它会保留集群上的附加软件。此选项可使附加组件成为自管理安装、而不是Amazon EKS附加组件。通过此选项、此附加组件不会出现停机。保留命令中的 `--preserve` 选项以保留此附加项。
- 从您的集群中完全删除附加软件–我们建议您仅在集群中没有依赖于此附加软件的资源时、才从集群中删除此附加软件。从命令中删除 `--preserve` 此选项 `delete` 以删除此加载项。

如果此附加项具有关联的IAM帐户、则不会删除此IAM帐户。

eksc

以下命令将卸载Trident EKS加载项：

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

管理控制台

1. 打开Amazon EKS控制台，网址为 https://console.aws.amazon.com/eks/home#/clusters。
2. 在左侧导航窗格中，单击*群集*。
3. 单击要删除的NetApp Trident CSI加载项的集群的名称。
4. 单击*Add-ons*选项卡，然后单击Trident by NetApp．*
5. 单击 * 删除 * 。
6. 在*Remove NetApp_trdent-operator con確 認*对话框中，执行以下操作：
   a. 如果您希望Amazon EKS停止管理此附加组件的设置、请选择*保留集群*。如果要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_trdent-operator*。
   c. 单击 * 删除 * 。

AWS命令行界面

请使用集群的名称进行替换 `my-cluster` 、然后运行以下命令。

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

配置存储后端

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。该文件需要指定所需的存储类型(NAS或SAN)、文件系统和用于获取该文件的SVM以及如何向其进行身份验证。以下示例显示了如何定义基于NAS的存储以及如何使用AWS密钥将凭据存储到要使用的SVM：

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
```

- 验证是否已成功创建Trident后端配置(TBC)：

```
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 for ONTAP驱动程序详细信息

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

- `ontap-san`：配置的每个PV都是其自身Amazon FSx for NetApp ONTAP卷中的一个LUN。建议用于块存储。
- `ontap-nas`：配置的每个PV都是一个完整的Amazon FSx for NetApp ONTAP卷。建议用于NFS和SMB。
- `ontap-san-economy`：配置的每个PV都是一个LUN，每个Amazon FSx for NetApp ONTAP卷具有可配置数量的LUN。
- `ontap-nas-economy`：配置的每个PV都是一个qtree、每个Amazon FSx for NetApp ONTAP卷具有一个可配置数量的qtree。
- `ontap-nas-flexgroup`：配置的每个PV都是一个完整的Amazon FSx for NetApp ONTAP FlexGroup卷。

有关驱动程序的详细信息，请参阅"NAS驱动程序"和"SAN驱动程序"。

创建配置文件后、运行此命令在EKS中创建该文件：

```
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

有关后端配置选项，请参见下表：

参数	说明	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是使用IPv6标志安装的、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]。如果在字段下 `aws`提供 `fsxFilesystemID`、则无需提供、 `managementLIF`因为`Trident`会从`AWS`检索`SVM` `managementLIF`信息。因此、您必须提供`SVM`下某个用户的凭据(例如`vsadmin`)、并且该用户必须具有此 `vsadmin` 角色。	"10.0.0.1 "， "2001： 1234：abcd：：： fefe] "

参数	说明	示例
dataLIF	协议 LIF 的 IP 地址。* ONTAP NAS驱动程序*：建议指定dataLIF。如果不提供此参数、则Trident将从SVM提取数据LUN。您可以指定用于NFS挂载操作的完全限定域名(FQDN)、从而可以创建循环DNS、以便在多个数据LIF之间实现负载平衡。可以在初始设置后更改。请参阅 。* ONTAP SAN驱动程序*：不为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 LI。如果明确定义了dataLIF、则会生成警告。如果Trident是使用IPv6标志安装的、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	启用自动创建和更新导出策略[布尔值]。使用 `autoExportPolicy`和 `autoExportCIDRs`选项、Trident可以自动管理导出策略。	false
autoExportCIDRs	用于筛选KubeNet节点IP的CIDR列表(启用时)。 `autoExportPolicy`使用 `autoExportPolicy`和 `autoExportCIDRs`选项、Trident可以自动管理导出策略。	"["0.0.0.0/0 "、"：：/0 "]"
labels	要应用于卷的一组任意 JSON 格式的标签	""
clientCertificate	客户端证书的 Base64 编码值。用于基于证书的身份验证	""
clientPrivateKey	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证	""
trustedCACertificate	受信任 CA 证书的 Base64 编码值。可选。用于基于证书的身份验证。	""
username	用于连接到集群或SVM的用户名。用于基于凭据的身份验证。例如、vsadmin。	
password	用于连接到集群或SVM的密码。用于基于凭据的身份验证。	
svm	要使用的 Storage Virtual Machine	如果指定SVM管理LIF则派生。
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	说明	示例
limitAggregateUsage	*请勿指定Amazon FSx for NetApp ONTAP。*提供的和 `vsadmin`不包含使用Trident检索聚合使用情况并对其进行限制所需的 `fsxadmin`权限。	请勿使用。
limitVolumeSize	如果请求的卷大小超过此值、则配置失败。此外、还会限制它为qtrees和FlexVol管理的卷的大小上限、并且此选项允许自定义每个LUN `qtreesPerFlexvol`的qtrees的最大数量。	"（默认情况下不强制实施）
lunsPerFlexvol	每个FlexVol 的最大LUN数必须在50、200范围内。仅SAN。	“100”
debugTraceFlags	故障排除时要使用的调试标志。例如、除非您正在进行故障排除并需要详细的日志转储、否则不会使用 {"ap1"：false、"METHOU"：true } debugTraceFlags。	空
nfsMountOptions	NFS挂载选项的逗号分隔列表。通常会在存储类中为Kubnetes-永久性卷指定挂载选项、但如果在存储类中未指定挂载选项、则Trident将回退到使用存储后端配置文件中指定的挂载选项。如果在存储类或配置文件中未指定挂载选项、则Trident不会在关联的永久性卷上设置任何挂载选项。	""
nasType	配置NFS或SMB卷创建。选项包括 nfs、 smb`或null。*对于SMB卷，必须设置为 `smb。*默认情况下、将设置为空会将NFS卷设置为空。	nfs
qtreesPerFlexvol	每个 FlexVol 的最大 qtree 数，必须在 50 ， 300 范围内	"200"
smbShare	您可以指定以下选项之一：使用Microsoft管理控制台或ONTAP命令行界面创建的SMB共享的名称、或者允许Trident创建SMB共享的名称。对于Amazon FSx for ONTAP后端、此参数是必需的。	smb-share

参数	说明	示例
useREST	用于使用 ONTAP REST API 的布尔参数。技术预览 useREST 以技术预览的形式提供，建议用于测试环境，而不用于生产工作负载。如果设置为 true，则Trident将使用ONTAP REST API与后端进行通信。此功能需要使用ONTAP 9.11.1及更高版本。此外、使用的ONTAP登录角色必须有权访问 ontap 应用程序。预定义的和角色可以满足这一 vsadmin 要求 cluster-admin 。	false
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机密管理器中的FSx SVM凭据。- name：密钥的Amazon资源名称(ARN)、其中包含SVM的凭据。- type：设置为 awsarn。有关详细信息、请参见 "创建AWS机密管理器密钥" 。	

用于配置卷的后端配置选项

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

参数	说明	默认
spaceAllocation	LUN 的空间分配	true
spaceReserve	空间预留模式；" 无 "（精简）或 " 卷 "（厚）	none
snapshotPolicy	要使用的 Snapshot 策略	none
qosPolicy	要为创建的卷分配的 QoS 策略组。选择每个存储池或后端的qosPolicy或adaptiveQosPolicy之一。将QoS策略组与Trident结合使用需要使用ONTAP 9™8或更高版本。您应使用非共享QoS策略组、并确保此策略组分别应用于每个成分卷。共享QoS策略组会对所有工作负载的总吞吐量实施上限。	"

参数	说明	默认
adaptiveQosPolicy	要为创建的卷分配的自适应 QoS 策略组。选择每个存储池或后端的qosPolicy或adaptiveQosPolicy之一。不受 ontap-nas-economy.	"
snapshotReserve	为快照预留的卷百分比为 "0"	如果 snapshotPolicy`为 `none、 `else`""
splitOnClone	创建克隆时，从其父级拆分该克隆	false
encryption	在新卷上启用NetApp卷加密(NVE)；默认为 false。要使用此选项，必须在集群上获得 NVE 的许可并启用 NVE 。如果在后端启用了NAE、则在Trident中配置的任何卷都将启用NAE。有关详细信息，请参阅：Trident如何与NVE和NAE配合使用"。	false
luksEncryption	启用LUKS加密。请参阅 "使用Linux统一密钥设置(LUKS)"。仅SAN。	""
tieringPolicy	要使用的层策略 none	`snapshot-only`对于ONTAP 9 5之前的SVM-DR配置
unixPermissions	新卷的模式。对于**SMB**卷保留为空。	""
securityStyle	新卷的安全模式。NFS支持 `mixed`和 `unix`安全模式。SMB支持 `mixed`和 `ntfs`安全模式。	NFS默认值为 unix。SMB默认值为 ntfs。

准备配置**SMB**卷

您可以使用驱动程序配置SMB卷 ontap-nas。完成以下步骤之前。ONTAP SAN和NAS驱动程序集成

开始之前

在使用驱动程序配置SMB卷之前、 `ontap-nas`您必须满足以下条件。

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

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

- 配置为Windows服务的CSI代理。要配置 csi-proxy，请参阅"GitHub：CSI代理"或了解在Windows上运行的Kubornetes"GitHub：适用于Windows的CSI代理"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管理员共享：使用"Microsoft管理控制台"共享文件夹

管理单元或使用ONTAP命令行界面。要使用ONTAP 命令行界面创建SMB共享、请执行以下操作：

a. 如有必要，为共享创建目录路径结构。

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

b. 创建与指定SVM关联的SMB共享：

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

c. 验证是否已创建共享：

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

有关完整详细信息、请参见"创建 SMB 共享"。

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后端配置选项的信息，请参阅"适用于ONTAP 的FSX配置选项和示例"。

参数	说明	示例
smbShare	您可以指定以下选项之一：使用Microsoft管理控制台或ONTAP命令行界面创建的SMB共享的名称、或者允许Trident创建SMB共享的名称。对于Amazon FSx for ONTAP后端、此参数是必需的。	smb-share
nasType	*必须设置为 smb.*如果为空，则默认为 nfs。	smb
securityStyle	新卷的安全模式。对于**SMB**卷，必须设置为 **ntfs**`或 `**mixed**。	`ntfs`或 `mixed`SMB卷
unixPermissions	新卷的模式。对于SMB卷、必须留空。	""

配置存储类和**PVC**

配置Kubnetes 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来请求对PV的访问的永久性卷(PV)和永久性卷克莱姆(PVC)。然后、您可以将PV挂载到POD。

创建存储类。

配置**Kubnetes StorageClass**对象

```
https://kubernetes.io/docs/concepts/storage/storage-classes/["Kubnetes StorageClass对象"^]将Trident标识为用于该类的配置程序、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
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
```

有关存储类如何与和参数交互以控制Trident如何配置卷的详细信息 `PersistentVolumeClaim`、请参见"Kubernetes 和 Trident 对象"。

创建存储类。

步骤

1. 这是一个Kubbernetes对象、因此、请使用 `kubectl`在Kubbernetes中创建它。

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

2. 现在，Kubernetes和Trident中都应显示一个*BASIC－Csi*存储类，并且Trident应已发现后端的池。

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

创建**PV**和**PVC**

A "*PersigentVolume*" (PV)是由集群管理员在Kubbernetes集群上配置的物理存储资源。 "*PersigentVolumeClaim*"(PVC)是指请求访问集群上的永久卷。

可以将PVC配置为请求特定大小的存储或访问模式。通过使用关联的StorageClass，集群管理员可以控制不限于持续卷大小和访问模式(例如性能或服务级别)。

创建PV和PVC后、您可以将卷挂载到Pod中。

PerfsentVolume示例清单

此示例清单文件显示了与StorageClass关联的10gi的基本PV `basic-csi`。

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

PersentVolumeClaim示例清单

这些示例显示了基本的PVC配置选项。

PVC、可接入**rwx**

此示例显示了一个具有rwx访问权限的基本PVC，该PVC与名为的StorageClass关联 `basic-csi`。

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

采用**NVMe/TCP**的**PVC**

此示例显示了与名为的StorageClass关联的具有读取权限的NVMe/TCP的基本PVC `protection-gold`。

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

创建**PV**和**PVC**

步骤

1. 创建PV。

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

2. 验证PV状态。

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

3. 创建PVC。

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

4. 验证PVC状态。

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

有关存储类如何与和参数交互以控制Trident如何配置卷的详细信息 `PersistentVolumeClaim`、请参见"Kubernetes 和 Trident 对象"。

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	键入	值	优惠	请求	支持
介质[1]	string	HDD ，混合，SSD	Pool 包含此类型的介质；混合表示两者	指定的介质类型	ontap-nas ，ontap-nas-economy. ontap-nas-flexgroup ，ontap-san ，solidfire-san
配置类型	string	精简，厚	Pool 支持此配置方法	指定的配置方法	Thick：All ONTAP ；Thin ：All ONTAP & solidfire-san

属性	键入	值	优惠	请求	支持
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.	池属于此类型的后端	指定后端	所有驱动程序
snapshots	池	true false	Pool 支持具有快照的卷	启用了快照的卷	ontap-nas ， ontap-san ， solidfire-san ， gcp-cvs
克隆	池	true false	Pool 支持克隆卷	启用了克隆的卷	ontap-nas ， ontap-san ， solidfire-san ， gcp-cvs
加密	池	true false	池支持加密卷	已启用加密的卷	ontap-nas ， ontap-nas-economy-、 ontap-nas-flexgroups ， ontap-san
IOPS	内部	正整数	Pool 能够保证此范围内的 IOPS	卷保证这些 IOPS	solidfire-san

[1] ： ONTAP Select 系统不支持

部署示例应用程序

部署示例应用程序。

步骤

1. 将卷挂载到Pod中。

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

以下示例显示了将PVC连接到POD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
```

您可以使用监控进度 `kubectl get pod --watch`。

2. 验证卷是否已挂载在上 `/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Pod。Pod应用程序将不再存在、但卷将保留。

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

在EKS集群上配置Trident EKS加载项

NetApp Trident简化了Kubelnetes中适用于NetApp ONTAP的Amazon FSx存储管理、使开发人员和管理员能够专注于应用程序部署。NetApp Trident EKS加载项包括最新的安全修补程序和错误修复、并已通过AWS验证、可与Amazon EKS配合使用。通过EKS加载项、您可以始终确保Amazon EKS集群安全稳定、并减少安装、配置和更新加载项所需的工作量。

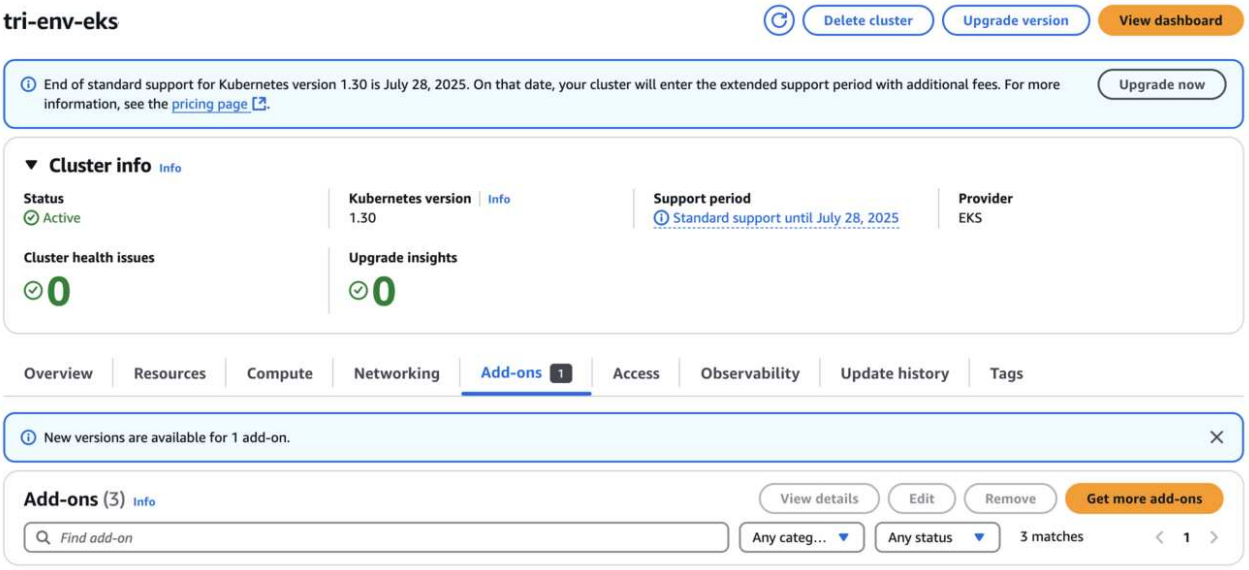
前提条件

在配置适用于AWS EKS的Trident加载项之前、请确保满足以下条件：

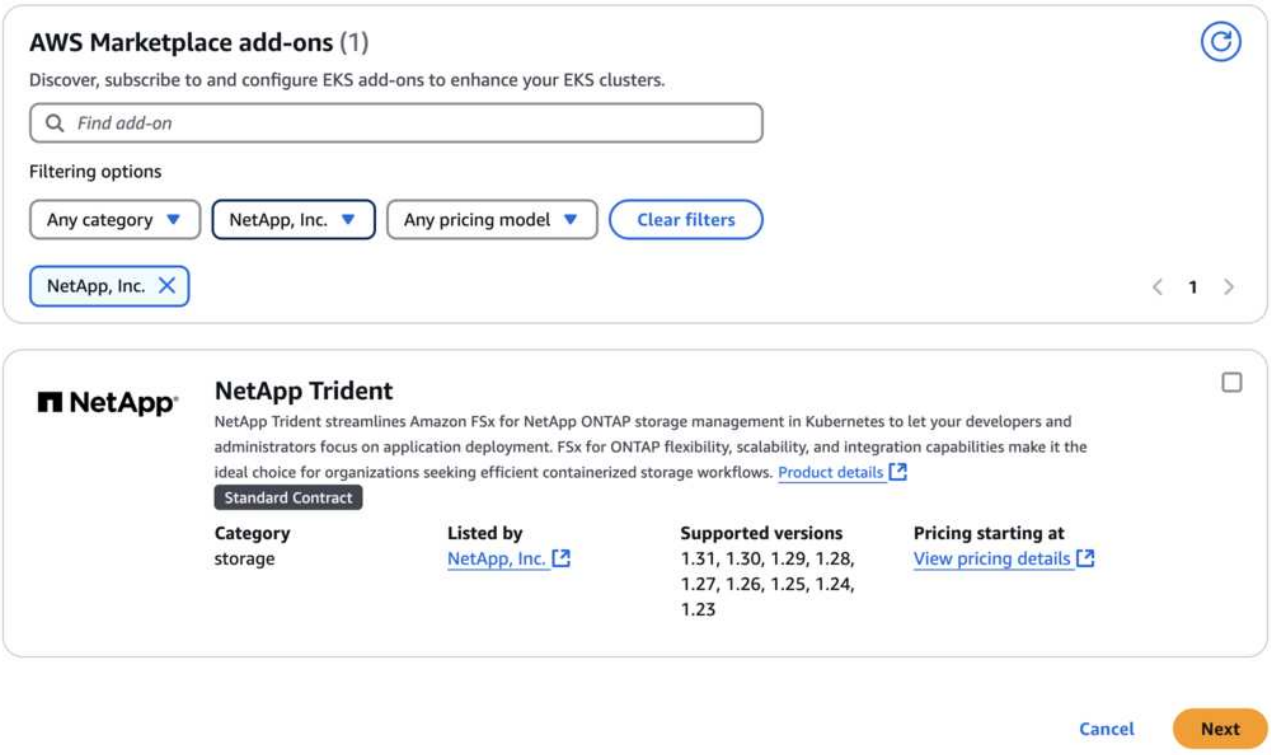
- 具有使用加载项的权限的Amazon EKS集群帐户。请参阅 "Amazon EKS附加项"。
- AWS对AWS Marketplace的权限：
  ```
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
  ```
- AMI类型：Amazon ARM 2 (AL2_x86_64)或Amazon Linux 2 ARM (AL2_AMAZON_64)
- 节点类型：AMD或ARM
- 现有Amazon FSx for NetApp ONTAP文件系统

步骤

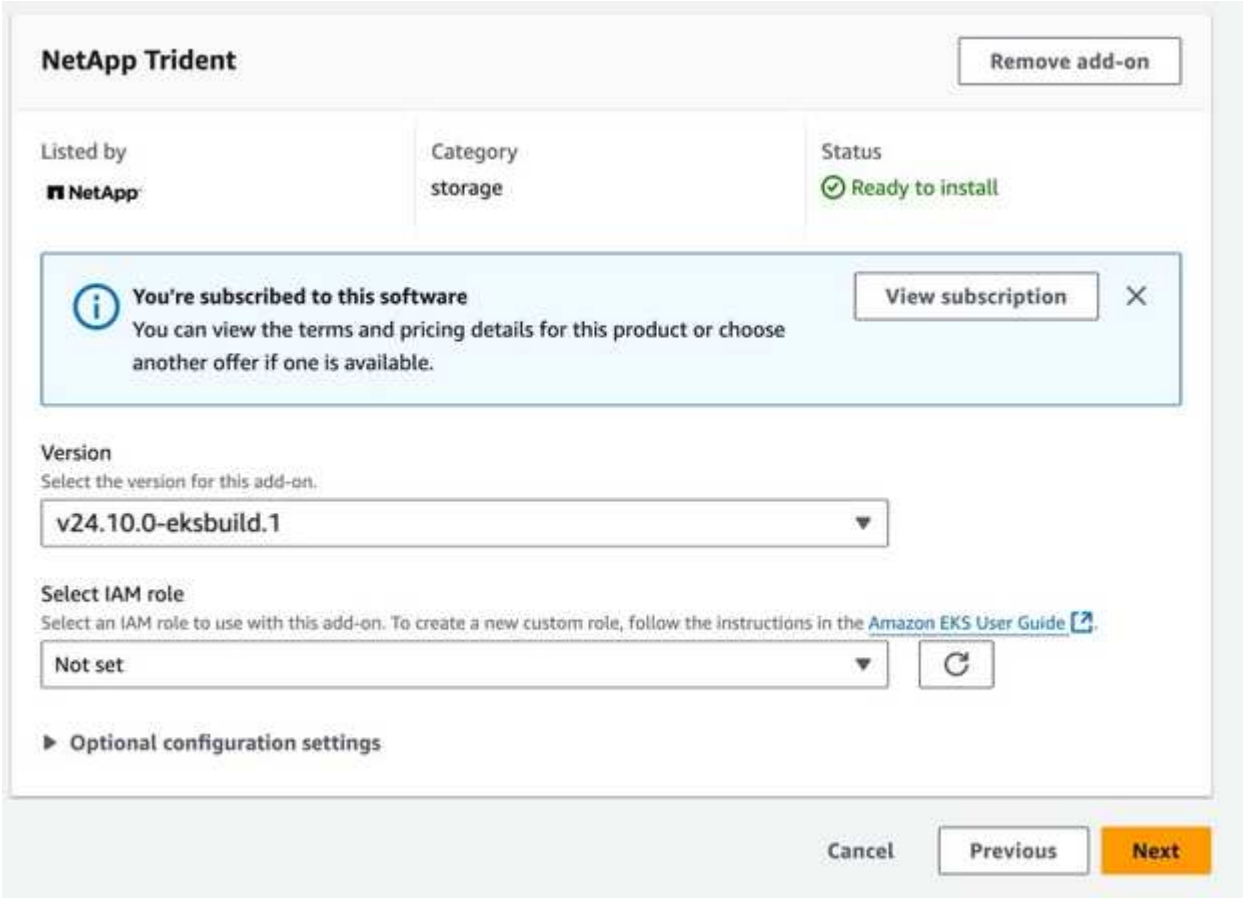
1. 请务必创建IAM角色和AWS密钥、以使EKS Pod能够访问AWS资源。有关说明，请参阅"创建IAM角色和AWS机密"。
2. 在EKS Kubernetes集群上、导航到*加载项*选项卡。



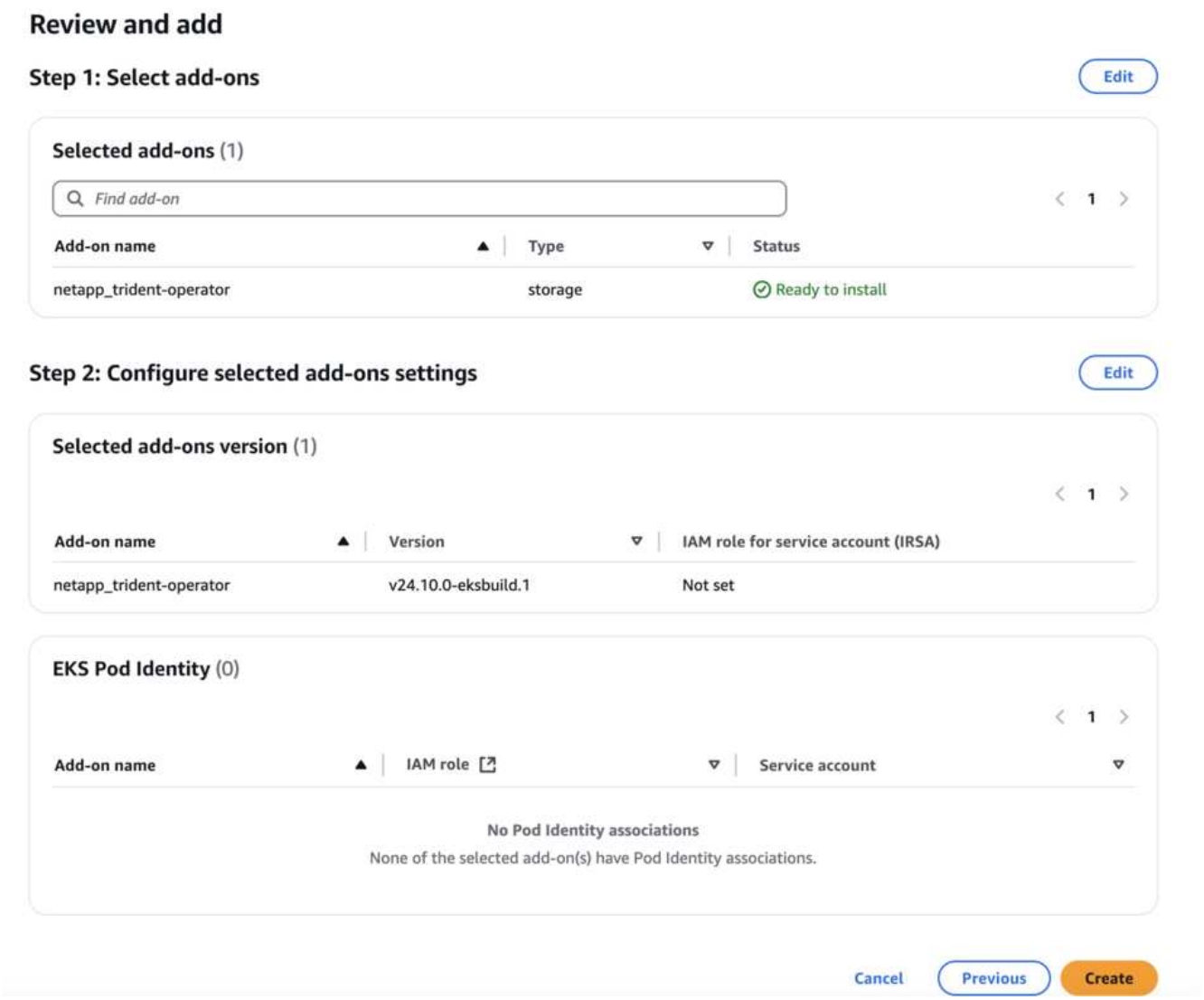
3. 转到*AWS Marketplace附加项*并选择_storage_类别。



4. 找到*Next* NetApp Trident并选中Trident插件的复选框，然后单击*Next*。
5. 选择所需的附加软件版本。



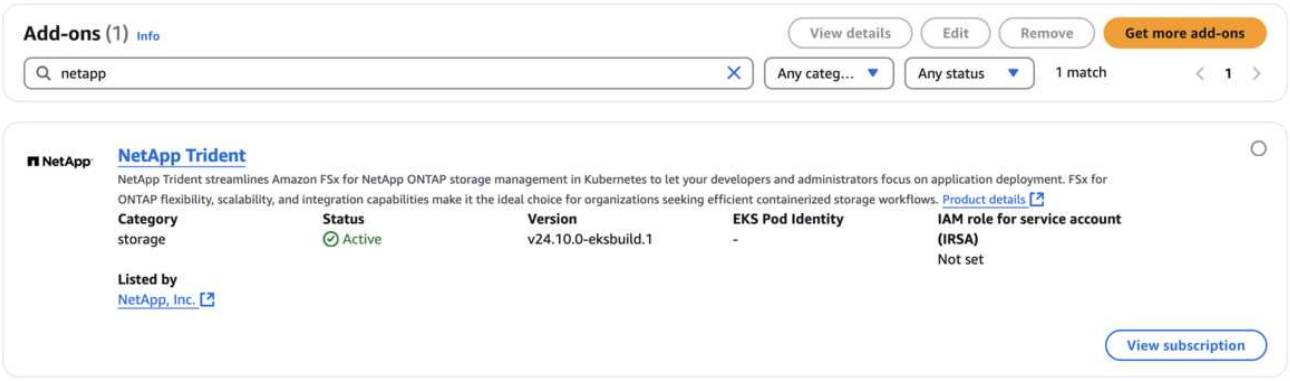
6. 选择要从节点继承的IAM角色选项。



7. 根据需要配置任何可选配置设置，然后选择*Next*。

遵循*附加配置模式*，并将*配置值*部分中的配置值参数设置为您在上一步(步骤1)中创建的ro-arn (值格式应为：)。 `eks.amazonaws.com/role-arn: arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`注意：如果您为冲突解决方法选择覆盖、则现有加载项的一个或多个设置可能会被Amazon EKS加载项设置覆盖。如果未启用此选项、并且与现有设置存在冲突、则操作将失败。您可以使用生成的错误消息来解决冲突。在选择此选项之前、请确保Amazon EKS附加组件未管理您需要自行管理的设置。

8. 选择 * 创建 * 。
9. 验证此加载项的状态是否为_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. 继续设置并配置存储后端。有关信息，请参见 "配置存储后端"。

使用命令行界面安装/卸载**Trident EKS**加载项

使用命令行界面安装**NetApp Trident EKS**加载项：
以下示例命令将安装Trident EKS加载项：

```
eksctl create addon --name aws-ebs-csi-driver --cluster <cluster_name>
--service-account-role-arn arn:aws:iam::<account_id>:role/<role_name>
--force
```

使用命令行界面卸载**NetApp Trident EKS**加载项：

以下命令将卸载Trident EKS加载项：

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

使用 kubectl 创建后端

后端用于定义Trident与存储系统之间的关系。它会告诉Trident如何与该存储系统通信、以及Trident如何从该存储系统配置卷。安装Trident后、下一步是创建后端。`TridentBackendConfig`通过自定义资源定义(CRD)、您可以直接通过Kubednetes界面创建和管理Trident后端。您可以使用或对Kubornetes分发等效的命令行界面工具来执行此操作 `kubectl。

TridentBackendConfig

`TridentBackendConfig`(tbc `tbconfig`、、 `tbackendconfig`)是一个前端，具有名称节奏的CRD，使您可以使用管理Trident后端 `kubectl`。现在，Kubbernetes和存储管理员可以直接通过Kubbernetes CLI创建和管理后端，而无需专用的命令行实用程序(`tridentctl`)。

创建对象时 `TridentBackendConfig`、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。
- `TridentBackendConfig`唯一绑定到由Trident创建的 `TridentBackend。

每个都 `TridentBackendConfig`与保持一对一映射 `TridentBackend。前者是为用户提供的用于设计和配置后端的界面；后者是Trident如何表示实际后端对象。

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

请参见以下CR格式示例 `TridentBackendConfig`：

```
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
```

您还可以查看目录中的示例、了解所需存储平台/服务的示例 "Trident 安装程序"配置。

`spec`采用特定于后端的配置参数。在此示例中、后端使用 `ontap-san`存储驱动程序、并使用此处表中列出的配置参数。有关所需存储驱动程序的配置选项列表，请参阅link:backends.html["存储驱动程序的后端配置信息"^]。

本 spec`节还包括 `credentials`和 `deletionPolicy`字段、这些字段是CR中新推出的 `TridentBackendConfig`：

- 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。此密码设置为用户创建的 Kubernetes Secret 。凭据不能以纯文本形式传递，因此会导致错误。
- deletionPolicy：此字段定义删除时应执行的操作 TridentBackendConfig。它可以采用以下两种可能值之一：
  - delete：这会导致删除 `TridentBackendConfig`CR和关联的后端。这是默认值。
  - retain： TridentBackendConfig`删除CR后，后端定义仍然存在，可以使用进行管理 `tridentctl。将删除策略设置为 retain`允许用户降级到早期版本(21.04之前的版本)并保留创建的后端。此字段的值可在创建后更新 `TridentBackendConfig。

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

使用创建的后端 tridentctl`没有关联 `TridentBackendConfig`对象。您可以通过创建CR来 `TridentBackendConfig`选择使用管理此类后端 `kubectl。必须注意指定相同的配置参数(如 spec.backendName、、 spec.storagePrefix spec.storageDriverName` 等)。Trident将自动将新创建的与已有的后端绑定 `TridentBackendConfig。

步骤概述

要使用创建新的后端 `kubectl`，应执行以下操作：

1. 创建 "Kubernetes 机密"。此密钥包含Trident与存储集群/服务通信所需的凭据。
2. 创建 `TridentBackendConfig`对象。其中包含有关存储集群 / 服务的详细信息，并引用了上一步中创建的密钥。

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

第 1 步：创建 Kubernetes 机密

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

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

下表汇总了每个存储平台的机密中必须包含的字段：

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

存储平台机密字段问题描述	机密	字段问题描述
ONTAP	客户端权限密钥	客户端专用密钥的 Base64 编码值。用于基于证书的身份验证
ONTAP	用户名	入站用户名。如果 useCHAP=true ，则为必需项。对于 `ontap-san` 和 `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP 启动程序密钥。如果 useCHAP=true ，则为必需项。对于 `ontap-san` 和 `ontap-san-economy`
ONTAP	chapTargetUsername	目标用户名。如果 useCHAP=true ，则为必需项。对于 `ontap-san` 和 `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP 目标启动程序密钥。如果 useCHAP=true ，则为必需项。对于 `ontap-san` 和 `ontap-san-economy`

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

第2步：创建 `TridentBackendConfig`CR

现在、您可以创建 `TridentBackendConfig`CR了。在此示例中、使用驱动程序的后端 `ontap-san`是使用以下对象创建的 `TridentBackendConfig`：

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

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

第3步：验证CR的状态 TridentBackendConfig

创建CR后 TridentBackendConfig、您可以验证状态。请参见以下示例：

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

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

阶段可以采用以下值之一：

- Bound： `TridentBackendConfig`CR与一个后端关联，该后端包含 `configRef`设置为 `TridentBackendConfig`CR的uid。
- Unbound：使用表示 ""。 `TridentBackendConfig`对象不会绑定到后端。默认情况下、所有新创建的 `TridentBackendConfig`CRS都处于此阶段。此阶段发生更改后，它将无法再次还原为 "Unbound （已取消绑定） "。
- Deleting： TridentBackendConfig`CR `deletionPolicy`已设置为删除。删除CR后 `TridentBackendConfig、它将过渡到Deleting状态。
  - 如果后端不存在永久性卷请求(PVC)、则删除 TridentBackendConfig`将导致Trident删除后端以及CR。 `TridentBackendConfig
  - 如果后端存在一个或多个 PVC ，则会进入删除状态。 TridentBackendConfig`CR随后也进入删除阶段。只有在删除所有PVC后、才会删除后端和 `TridentBackendConfig。
- Lost：与CR关联的后端 TridentBackendConfig`被意外或故意删除，而 `TridentBackendConfig`CR仍有对已删除后端的引用。 `TridentBackendConfig`无论值如何、均可删除CR `deletionPolicy。
- Unknown：Trident无法确定与CR关联的后端的状态或是否存在 TridentBackendConfig。例如、如

果API服务器未响应或 `tridentbackends.trident.netapp.io`缺少CRD。这可能需要干预。

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

（可选）第 4 步：获取更多详细信息

您可以运行以下命令来获取有关后端的详细信息：

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

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

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

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

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

`backendInfo`包含 `backendName`响应CR而创建的后端的 `TridentBackendConfig`和 `backendUUID。 lastOperationStatus`字段表示CR的上次操作状态，可以是用户触发的操作（例如，用户在中更改了某些内容），也可以是Trident触发的操作 `TridentBackendConfig(例如， spec` 在Trident重新启动期间）。可以是成功、也可以是失败。 `phase`表示CR和后端之间关系的状态 `TridentBackendConfig。在上面的示例中、 `phase`具有绑定值、这意味着 `TridentBackendConfig`CR与后端关联。

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

> 您不能使用更新或删除包含关联对象 `tridentctl`的后端 `TridentBackendConfig。要了解在和之间切换所涉及的 `TridentBackendConfig`步骤 `tridentctl，"请参见此处"。

管理后端

使用 **kubectl** 执行后端管理

了解如何使用执行后端管理操作 `kubectl`。

删除后端

通过删除 `TridentBackendConfig`，您可以指示Trident删除/保留后端(基于 `deletionPolicy`)。要删除后端、请确保 `deletionPolicy`将设置为`delete`。要仅删除 `TridentBackendConfig`，请确保 `deletionPolicy`将设置为保留。这样可以确保后端仍然存在，并且可以使用进行管理 `tridentctl`。

运行以下命令：

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

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

查看现有后端

运行以下命令：

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

您还可以运行 `tridentctl get backend -n trident`或 `tridentctl get backend -o yaml -n trident`以获取所有已存在后端的列表。此列表还将包括使用创建的后端 `tridentctl`。

更新后端

更新后端可能有多种原因：

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

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

- 需要更新参数（例如所使用的 ONTAP SVM 的名称）。
  - 您可以使用以下命令直接通过Kubbernetes更新 `TridentBackendConfig`对象：

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

  - 或者、您也可以使用以下命令更改现有 `TridentBackendConfig`CR：

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

- 如果后端更新失败，则后端仍会保持在其上次已知配置中。您可以通过运行或 `kubectl describe tbc <tbc-name> -n trident`来查看日志以确定原因 `kubectl get tbc <tbc-name> -o yaml -n trident`。
- 确定并更正配置文件中的问题后，您可以重新运行 update 命令。

使用 tridentctl 执行后端管理

了解如何使用执行后端管理操作 tridentctl。

创建后端

创建后"后端配置文件"，运行以下命令：

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

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

```
tridentctl logs -n trident
```

确定并更正配置文件的问题后、只需再次运行命令即可 `create`。

删除后端

要从Trident中删除后端、请执行以下操作：

1. 检索后端名称：

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

2. 删除后端：

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

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

查看现有后端

要查看 Trident 了解的后端，请执行以下操作：

- 要获取摘要，请运行以下命令：

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

- 要获取所有详细信息，请运行以下命令：

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

更新后端

创建新的后端配置文件后，运行以下命令：

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

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

```
tridentctl logs -n trident
```

确定并更正配置文件的问题后、只需再次运行命令即可 `update`。

确定使用后端的存储类

以下是您可以使用为后端对象输出的JSON回答的问题示例 `tridentctl`。这将使用 `jq`您需要安装的实用程序。

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

这也适用于使用创建的后端 `TridentBackendConfig`。

在后端管理选项之间移动

了解在Trident中管理后端的不同方法。

用于管理后端的选项

随着的推出 `TridentBackendConfig`，管理员现在可以通过两种独特的方式管理后端。这会提出以下问题：

- 使用创建的后端是否 `tridentctl`可通过进行管理 `TridentBackendConfig`?
- 是否可以使用管理 `tridentctl`使用创建的后端 `TridentBackendConfig`?

使用管理 `tridentctl`后端 `TridentBackendConfig`

本节介绍通过创建对象直接通过Kubenetes界面创建后端所需的管理步骤 `tridentctl`。`TridentBackendConfig`

这适用于以下情形：

- 已有的后端，因为它们是使用创建的，所以 `tridentctl`没有 `TridentBackendConfig`。
- 使用创建的新后端 `tridentctl`，而存在其他 `TridentBackendConfig`对象。

在这两种情况下、都将继续存在后端、Trident会为这些后端计划卷并在其上运行。管理员可以选择以下两种方式之一：

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

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

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

第 0 步：确定后端

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

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

cat ontap-nas-backend.json
```

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

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

第 1 步：创建 Kubernetes 机密

创建一个包含后端凭据的机密，如以下示例所示：

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

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

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

第2步：创建 `TridentBackendConfig`CR

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

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

在这种情况下、 `TridentBackendConfig`将如下所示：

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

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

第3步：验证CR的状态 `TridentBackendConfig`

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

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

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

现在、可以使用对象完全管理后端 `tbc-ontap-nas-backend` `TridentBackendConfig`。

使用管理 `TridentBackendConfig`后端 `tridentctl`

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

第 0 步：确定后端

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

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

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

从输出中可以看出、 `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`

让我们来看看的价值 `deletionPolicy`。需要将其设置为 `retain`。这样可以确保在删除CR时 `TridentBackendConfig`，后端定义仍然存在，并且可以使用进行管理 `tridentctl`。

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

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

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

除非将设置为， `retain`否则请勿继续下一步 `deletionPolicy`。

第2步：删除 `TridentBackendConfig`CR

最后一步是删除 `TridentBackendConfig`CR。确认已设置为 `retain`后 `deletionPolicy`，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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

删除对象后 `TridentBackendConfig`、Trident会直接将其删除、而不会实际删除后端本身。