最佳实践和建议
Trident

NetApp
July 01, 2026

目录

最佳实践和建议

部署

部署 Trident 时，请使用此处列出的建议。

部署到专用命名空间

"命名空间" 提供不同应用程序之间的管理隔离，并且是资源共享的障碍。例如，来自一个命名空间的 PVC 不能从另一个命名空间使用。Trident 为 Kubernetes 集群中的所有命名空间提供 PV 资源，从而利用具有提升权限的服务帐户。

此外，访问 Trident pod 可能会使用户能够访问存储系统凭据和其他敏感信息。确保应用程序用户和管理应用程序无法访问 Trident 对象定义或 pod 本身非常重要。

使用配额和范围限制来控制存储消耗

Kubernetes 有两个特性，当它们结合在一起时，提供了一个强大的机制来限制应用程序的资源消耗。 "存储配额机制"使管理员能够在每个命名空间的基础上实现全局和存储类特定的容量和对象计数消耗限制。此外，使用 "范围限制"确保在将请求转发给供应者之前，PVC 请求处于最小值和最大值之间。

这些值是在每个命名空间的基础上定义的，这意味着每个命名空间都应该定义符合其资源要求的值。有关 "如何利用配额"的信息，请参阅此处。

存储配置

NetApp 产品组合中的每个存储平台都具有独特的功能，使应用程序（无论是否是容器化的）受益。

平台概述

Trident 与 ONTAP 和 Element 配合使用。没有一个平台比另一个平台更适合所有应用程序和场景，但是，在选择平台时应考虑应用程序和管理设备的团队的需求。

您应该遵循主机操作系统的基本最佳实践，以及您正在利用的协议。或者，您可能需要考虑将应用程序最佳实践（如果可用）与后端、存储类和 PVC 设置相结合，以优化特定应用程序的存储。

ONTAP 和 Cloud Volumes ONTAP 最佳实践

了解为 Trident 配置 ONTAP 和 Cloud Volumes ONTAP 的最佳实践。

以下建议是为容器化工作负载配置 ONTAP 的指导原则，这些工作负载使用由 Trident 动态配置的卷。应考虑并评估每种方法是否适合您的环境。

使用专用于 **Trident** 的 **SVM**

Storage Virtual Machine （ SVM ）在 ONTAP 系统上的租户之间提供隔离和管理隔离。将 SVM 专用于应用程序可以实现权限委派，并能够应用限制资源消耗的最佳实践。

有多种选择可用于管理 SVM：

- 在后端配置中提供集群管理界面，以及相应的凭据，并指定 SVM 名称。
- 使用 ONTAP System Manager 或 CLI 为 SVM 创建专用管理界面。
- 通过 NFS 数据接口共享管理角色。

在每种情况下，接口都应在 DNS 中，并且在配置 Trident 时应使用 DNS 名称。这有助于促进某些灾难恢复场景，例如，不使用网络身份保留的 SVM-DR。

对于 SVM，没有专用或共享管理 LIF 的偏好，但是，您应该确保您的网络安全策略与您选择的方法保持一致。无论如何，管理 LIF 应可通过 DNS 访问，以实现最大的灵活性，如果 "SVM-DR"与 Trident 结合使用。

限制最大卷计数

ONTAP 存储系统具有最大卷计数，该计数因软件版本和硬件平台而异。请参阅 "NetApp Hardware Universe"以确定您的特定平台和 ONTAP 版本的确切限制。当卷计数耗尽时，不仅 Trident 的配置操作会失败，所有存储请求的配置操作都会失败。

Trident 的 `ontap-nas`和 `ontap-san`驱动程序为创建的每个 Kubernetes 持久卷 (PV) 配置一个 FlexVolume。`ontap-nas-economy`驱动程序大约每 200 个 PV 创建一个 FlexVolume（可配置在 50 到 300 之间）。 `ontap-san-economy`驱动程序大约每 100 个 PV 创建一个 FlexVolume（可配置在 50 到 200 之间）。要防止 Trident 消耗存储系统上的所有可用卷，您应该在 SVM 上设置限制。您可以从命令行执行此操作：

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

`max-volumes` 的值根据您环境的几个特定标准而有所不同：

- ONTAP 集群中的现有卷数
- 您希望在 Trident 之外为其他应用程序配置的卷数量
- Kubernetes 应用程序预计消耗的持久卷数

该 `max-volumes`值是在 ONTAP 集群中的所有节点上配置的总卷，而不是在单个 ONTAP 节点上配置的总卷。因此，您可能会遇到某些情况，其中 ONTAP 集群节点可能比其他节点具有更多或更少 Trident 配置卷。

例如，双节点 ONTAP 集群最多可以托管 2000 个 FlexVol 卷。将最大卷计数设置为 1250 似乎非常合理。但是，如果仅 "聚合"从一个节点分配给 SVM，或者无法对从一个节点分配的聚合进行配置（例如，由于容量），则另一个节点将成为所有 Trident 配置卷的目标。这意味着在达到 `max-volumes`值之前，可能会达到该节点的卷限制，从而影响 Trident 和使用该节点的其他卷操作。您可以通过确保将集群中每个节点的聚合以相等数量分配给 **Trident** 使用的 **SVM** 来避免这种情况。

克隆卷

NetApp Trident 在使用 `ontap-nas`、 `ontap-san` 和 `solidfire-san` 存储驱动程序时支持克隆卷。使用 `ontap-nas-flexgroup` 或 `ontap-nas-economy` 驱动程序时，不支持克隆。从现有卷创建新卷将导致创建新的快照。

避免克隆与不同 StorageClass 关联的 PVC。在相同的 StorageClass 内执行克隆操作，以确保兼容性并防止意外行为。

限制 **Trident** 创建的卷的最大大小

要配置 Trident 可创建的卷的最大大小，请在您的 `limitVolumeSize`定义中使用 `backend.json`参数。

除了控制存储阵列的卷大小之外，还应利用 Kubernetes 功能。

限制 **Trident** 创建的 **FlexVols** 的最大大小

要为 ontap-san-economy 和 ontap-nas-economy 驱动程序用作池的 FlexVols 配置最大大小，请在 `limitVolumePoolSize` 参数中，在 `backend.json` 定义中使用。

配置 **Trident** 使用双向 **CHAP**

您可以在后端定义中指定 CHAP 启动程序和目标用户名和密码，并让 Trident 在 SVM 上启用 CHAP。使用后端配置中的 `useCHAP`参数，Trident 使用 CHAP 对 ONTAP 后端的 iSCSI 连接进行身份验证。

创建和使用 **SVM QoS** 策略

利用应用于 SVM 的 ONTAP QoS 策略，可限制 Trident 配置卷可消耗的 IOPS 数量。这有助于 "防止霸凌"或失控容器影响 Trident SVM 外部的工作负载。

您可以通过几个步骤为 SVM 创建 QoS 策略。有关最准确的信息，请参阅您的 ONTAP 版本文档。以下示例创建了一个 QoS 策略，将 SVM 可用的总 IOPS 限制为 5000。

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

此外，如果您的 ONTAP 版本支持它，则可以考虑使用 QoS 最小值来保证容器化工作负载的吞吐量。自适应 QoS 与 SVM 级别策略不兼容。

专用于容器化工作负载的 IOPS 数量取决于许多方面。其中包括：

- 使用存储阵列的其他工作负载。如果有其他与 Kubernetes 部署无关的工作负载利用存储资源，则应注意确保这些工作负载不会受到意外不利影响。
- 在容器中运行的预期工作负载。如果具有高 IOPS 要求的工作负载将在容器中运行，则低 QoS 策略会导致不良体验。

重要的是要记住，在 SVM 级别分配的 QoS 策略会导致调配到 SVM 的所有卷共享相同的 IOPS 池。如果一个或少数容器化应用程序具有较高的 IOPS 要求，它可能会成为其他容器化工作负载的欺凌者。如果是这种情况，您可能需要考虑使用外部自动化来分配每个卷的 QoS 策略。

仅当您的 ONTAP 版本早于 9.8 时，才应将 QoS 策略组分配给 SVM。

为 **Trident** 创建 **QoS** 策略组

服务质量 (QoS) 保证关键工作负载的性能不会因竞争工作负载而降低。ONTAP QoS 策略组为卷提供 QoS 选项，并使用户能够为一个或多个工作负载定义吞吐量上限。有关 QoS 的详细信息，请参阅 "通过 QoS 保证吞吐量"。您可以在后端或存储池中指定 QoS 策略组，并将其应用于在该池或后端中创建的每个卷。

ONTAP 有两种 QoS 策略组：传统和自适应。传统策略组在 IOPS 中提供固定的最大（或更高版本中的最小值）吞吐量。自适应 QoS 自动将吞吐量扩展到工作负载大小，随着工作负载大小的变化而保持 IOPS 与 TBs|GBs 的比例。当您在大型部署中管理数百或数千个工作负载时，这提供了显著的优势。

创建 QoS 策略组时，请考虑以下事项：

- 您应在后端配置的 `defaults` 块中设置 `qosPolicy` 键。请参见以下后端配置示例：

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
  - labels:
      performance: extreme
    defaults:
      adaptiveQosPolicy: extremely-adaptive-pg
  - labels:
      performance: premium
    defaults:
      qosPolicy: premium-pg
```

- 您应该按卷应用策略组，以便每个卷获得策略组指定的整个吞吐量。不支持共享策略组。

有关 QoS 策略组的详细信息，请参阅 "ONTAP 命令参考"。

限制存储资源访问 **Kubernetes** 集群成员

限制对 Trident 创建的 NFS 卷、iSCSI LUN 和 FC LUN 的访问是 Kubernetes 部署安全态势的关键组成部分。这样做可以防止不属于 Kubernetes 集群的主机访问卷，并可能意外修改数据。

请务必了解，命名空间是 Kubernetes 中资源的逻辑边界。假设相同命名空间中的资源可以共享，但重要的是，没有跨命名空间功能。这意味着，即使 PV 是全局对象，当绑定到 PVC 时，它们只能由位于同一命名空间中的 pod 访问。确保在适当的时候使用命名空间来提供分隔至关重要。

大多数组织在 Kubernetes 环境中对数据安全性的主要关注点是，容器中的进程可以访问挂载到主机的存储，但这些存储并非为该容器准备的。 "命名空间" 旨在防止这种类型的妥协。但是，有一个例外：特权容器。

特权容器是使用比正常情况下更多的主机级权限运行的容器。默认情况下，这些功能不会被拒绝，因此请确保使用 "pod 安全策略" 禁用此功能。

对于需要从 Kubernetes 和外部主机进行访问的卷，应以传统方式管理存储，PV 由管理员引入，而不是由 Trident 管理。这确保了只有当 Kubernetes 和外部主机都断开连接并且不再使用卷时，存储卷才会被销毁。此外，还可以应用自定义导出策略，允许从 Kubernetes 集群节点和 Kubernetes 集群外部的目标服务器进行访问。

对于具有专用基础架构节点（例如 OpenShift）的部署或其他无法安排用户应用程序的节点，应使用单独的导出策略来进一步限制对存储资源的访问。这包括为部署到这些基础设施节点的服务（例如，OpenShift Metrics 和 Logging 服务）以及部署到非基础设施节点的标准应用程序创建导出策略。

使用专用导出策略

您应该确保每个后端都存在一个导出策略，该策略仅允许访问 Kubernetes 集群中存在的节点。Trident 可以自动创建和管理导出策略。通过这种方式，Trident 将其配置的卷的访问限制为 Kubernetes 集群中的节点，并简化了节点的添加/删除。

或者，您也可以手动创建导出策略，并为其填充一个或多个处理每个节点访问请求的导出规则：

- 使用 `vserver export-policy create` ONTAP CLI 命令创建导出策略。
- 使用 `vserver export-policy rule create` ONTAP CLI 命令将规则添加到导出策略。

运行这些命令可以限制哪些 Kubernetes 节点可以访问这些数据。

为应用程序 **SVM** 禁用 `showmount`

该 `showmount`功能使 NFS 客户端能够查询 SVM 以获取可用 NFS 导出的列表。部署到 Kubernetes 集群的 pod 可以对发出 `showmount -e`命令，并接收可用挂载的列表，包括它无权访问的挂载。虽然这本身并不是安全威胁，但它确实提供了不必要的信息，可能会帮助未经授权的用户连接到 NFS 导出。

您应该使用 SVM 级 ONTAP CLI 命令禁用 `showmount`：

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire 最佳实践

了解为 Trident 配置 SolidFire 存储的最佳实践。

创建 **SolidFire** 账户

每个 SolidFire 帐户代表一个唯一的卷所有者，并接收其自己的一组 Challenge-Handshake Authentication Protocol (CHAP) 凭据。您可以使用帐户名和相关 CHAP 凭据或通过卷访问组访问分配给帐户的卷。一个帐户最多可以分配两千个卷，但一个卷只能属于一个帐户。

创建 **QoS** 策略

如果要创建和保存可应用于多个卷的标准化服务质量设置，请使用 SolidFire 服务质量 (QoS) 策略。

您可以按卷设置 QoS 参数。可以通过设置定义 QoS 的三个可配置参数来确保每个卷的性能：最小 IOPS、最大 IOPS 和突发 IOPS。

下面是 4Kb 块大小可能的最小、最大和突发 IOPS 值。

IOPS 参数	定义	最小值	默认值	最大值 **(4Kb)**
最小 IOPS	卷的保证性能级别。	50	50	15000
最大 IOPS	性能不会超过此限制。	50	15000	200,000
突发 IOPS	短时间突发场景中允许的最大 IOPS。	50	15000	200,000

虽然最大 IOPS 和突发 IOPS 可以设置为高达 200,000，但卷的实际最大性能受到群集使用率和每个节点性能的限制。

块大小和带宽对 IOPS 数量有直接影响。随着块大小的增加，系统将带宽增加到处理较大块大小所需的级别。随着带宽的增加，系统能够实现的 IOPS 数量会减少。有关 QoS 和性能的更多信息，请参阅 "SolidFire 服务质量"。

SolidFire 身份验证

Element 支持两种身份验证方法：CHAP 和卷访问组 (VAG)。CHAP 使用 CHAP 协议向后端验证主机。卷访问组控制对其设置的卷的访问。NetApp 建议使用 CHAP 进行身份验证，因为它更简单且没有扩展限制。

具有增强 CSI 配置程序的 Trident 支持使用 CHAP 身份验证。VAG 只能在传统的非 CSI 操作模式中使用。

只有基于帐户的访问控制才支持 CHAP 身份验证（验证启动器是预期的卷用户）。如果使用 CHAP 进行身份验证，则有两个选项可用：单向 CHAP 和双向 CHAP。单向 CHAP 使用 SolidFire 帐户名和启动器密码对卷访问进行身份验证。双向 CHAP 选项提供了最安全的卷身份验证方式，因为卷通过帐户名和启动器密码对主机进行身份验证，然后主机通过帐户名和目标密码对卷进行身份验证。

但是，如果无法启用 CHAP 并且需要 VAG，请创建访问组并将主机启动程序和卷添加到访问组。添加到访问组中的每个 IQN 都可以使用 CHAP 身份验证或不使用 CHAP 身份验证访问组中的每个卷。如果将 iSCSI 启动程序配置为使用 CHAP 身份验证，则使用基于帐户的访问控制。如果未将 iSCSI 启动程序配置为使用 CHAP 身份验证，则使用卷访问组访问控制。

在哪里可以找到更多信息？

下面列出了一些最佳实践文档。搜索 "NetApp 库"以查找最新版本。

ONTAP

- "NFS 最佳实践和实施指南"
- "SAN 管理" (用于 iSCSI)
- "适用于 RHEL 的 iSCSI Express 配置"

Element 软件

- "为 Linux 配置 SolidFire"

NetApp HCI

- "NetApp HCI 部署先决条件"
- "访问 NetApp 部署引擎"

应用程序最佳实践信息

- "ONTAP 上 MySQL 的最佳实践"
- "MySQL 在 SolidFire 上的最佳实践"
- "NetApp SolidFire 和 Cassandra"
- "Oracle 在 SolidFire 上的最佳实践"
- "PostgreSQL 在 SolidFire 上的最佳实践"

并非所有应用程序都有特定的指导方针，与您的 NetApp 团队合作并使用 "NetApp 库"查找最新文档非常重要。

整合 Trident

要集成 Trident，需要集成以下设计和架构元素：驱动程序选择和部署、存储类设计、虚拟池设计、持久卷声明 (PVC) 对存储配置的影响、卷操作以及使用 Trident 的 OpenShift 服务部署。

驱动程序选择和部署

为您的存储系统选择并部署后端驱动程序。

ONTAP 后端驱动程序

ONTAP 后端驱动程序根据使用的协议以及如何在存储系统上调配卷而有所不同。因此，在决定部署哪个驱动程序时，请仔细考虑。

在更高的级别上，如果您的应用程序具有需要共享存储的组件（多个 pod 访问相同的 PVC），则基于 NAS 的驱动程序将成为默认选择，而基于块的 iSCSI 驱动程序将满足非共享存储的需求。根据应用程序的要求以及存储和基础设施团队的舒适度来选择协议。一般来说，对于大多数应用程序，它们之间几乎没有区别，因此决定通常取决于是否需要共享存储（其中需要同时访问多个 pod）。

可用的 ONTAP 后端驱动程序包括：

- `ontap-nas`: 配置的每个 PV 都是完整的 ONTAP FlexVolume。

- `ontap-nas-economy`：配置的每个 PV 都是一个 qtree，每个 FlexVolume 的 qtree 数量可配置（默认值为 200）。
- `ontap-nas-flexgroup`：每个 PV 配置为完整的 ONTAP FlexGroup，并使用分配给 SVM 的所有聚合。
- `ontap-san`：配置的每个 PV 都是其自身 FlexVolume 内的 LUN。
- `ontap-san-economy`：调配的每个 PV 都是一个 LUN，每个 FlexVolume 具有可配置的 LUN 数（默认值为 100）。

在三个 NAS 驱动程序之间进行选择会对应用程序提供的功能产生一些影响。

请注意，在下表中，并非所有功能都通过 Trident 公开。如果需要该功能，则存储管理员必须在配置后应用某些功能。上标脚注区分了每个功能和驱动程序的功能。

ONTAP NAS 驱动程序	**Snapshot**	克隆	动态导出策略	多重连接	**QoS**	调整大小	复制
`ontap-nas`	是	是	是脚注:5[]	是	是脚注:1[]	是	是脚注:1[]
`ontap-nas-economy`	NO [3]	NO [3]	是脚注:5[]	是	NO [3]	是	NO [3]
`ontap-nas-flexgroup`	是脚注:1[]	否	是脚注:5[]	是	是脚注:1[]	是	是脚注:1[]

Trident 为 ONTAP 提供 2 个 SAN 驱动程序，其功能如下所示。

ONTAP SAN 驱动程序	**Snapshot**	克隆	多重连接	双向 **CHAP**	**QoS**	调整大小	复制
`ontap-san`	是	是	是脚注:4[]	是	是脚注:1[]	是	是脚注:1[]
`ontap-san-economy`	是	是	是脚注:4[]	是	NO [3]	是	NO [3]

> 以上表格的脚注：是脚注：1[]：不由 Trident 管理 是脚注：2[]：由 Trident 管理，但不是 PV 粒度 否脚注：3[]：不由 Trident 管理且不是 PV 粒度 是脚注：4[]：支持原始块卷 是脚注：5[]：由 Trident 支持

非 PV 粒度的功能将应用于整个 FlexVolume，并且所有 PV（即共享 FlexVols 中的 qtree 或 LUN）都将共享一个共同的时间表。

如上表所示， `ontap-nas` 和 `ontap-nas-economy` 之间的大部分功能是相同的。但是，由于 `ontap-nas-economy` 驱动程序限制了按 PV 粒度控制计划的能力，这尤其会影响您的灾难恢复和备份计划。对于希望在 ONTAP 存储上利用 PVC 克隆功能的开发团队，只有在使用 `ontap-nas`、 `ontap-san` 或 `ontap-san-economy` 驱动程序时才有可能。

> `solidfire-san` 驱动程序还能够克隆 PVC。

Cloud Volumes ONTAP 后端驱动程序

Cloud Volumes ONTAP 为各种用例提供数据控制以及企业级存储功能，包括为 NAS 和 SAN 协议（NFS 、SMB/CIFS 和 iSCSI）提供服务的文件共享和块级存储。Cloud Volume ONTAP 的兼容驱动程序为 `ontap-nas`、 `ontap-nas-economy`、 `ontap-san`和 `ontap-san-economy`。这些适用于 Azure 的 Cloud Volume ONTAP、GCP 的 Cloud Volume ONTAP。

Amazon FSx for ONTAP 后端驱动程序

Amazon FSx for NetApp ONTAP 可让您利用熟悉的 NetApp 功能、性能和管理功能，同时利用在 AWS 上存储数据的简单性、敏捷性、安全性和可扩展性。FSx for ONTAP 支持许多 ONTAP 文件系统功能和管理 API 。Cloud Volume ONTAP 的兼容驱动程序是 `ontap-nas`、 `ontap-nas-economy`、 `ontap-nas-flexgroup`、 `ontap-san` 和 `ontap-san-economy`。

NetApp HCI/SolidFire 后端驱动程序

与 NetApp HCI/SolidFire 平台一起使用的 `solidfire-san`驱动程序可帮助管理员根据 QoS 限制为 Trident 配置 Element 后端。如果您想设计后端以在 Trident 配置的卷上设置特定的 QoS 限制，请使用后端文件中的 `type`参数。管理员还可以使用 `limitVolumeSize`参数限制可以在存储上创建的卷大小。目前， `solidfire-san`驱动程序不支持调整卷大小和卷复制等 Element 存储功能。这些操作应通过 Element Software Web UI 手动完成。

SolidFire 驱动程序	Snapshot	克隆	多重连接	CHAP	QoS	调整大小	复制
`solidfire-san`	是	是	是脚注:2[]	是	是	是	是脚注:1[]

> 脚注：是脚注：1[]：不由 Trident 管理 是脚注：2[]：支持原始块卷

Azure NetApp Files 后端驱动程序

Trident 使用 `azure-netapp-files` 驱动程序来管理 "Azure NetApp Files" 服务。

有关此驱动程序及其配置方法的详细信息，请参见 "Azure NetApp Files 的 Trident 后端配置"。

Azure NetApp Files 驱动程序	Snapshot	克隆	多重连接	QoS	展开	复制
`azure-netapp-files`	是	是	是	是	是	是脚注:1[]

> 脚注：是脚注：1[]：不由 Trident 管理

存储类设计

需要配置和应用单个存储类来创建 Kubernetes Storage Class 对象。本节讨论如何为应用程序设计存储类。

特定后端利用率

筛选可以在特定存储类对象内使用，以确定要与该特定存储类一起使用的存储池或池集。可以在存储类中设置三组过滤器： `storagePools`、 `additionalStoragePools` 和/或 `excludeStoragePools`。

该 `storagePools` 参数有助于将存储限制为与任何指定属性匹配的一组池。该 `additionalStoragePools` 参数用于扩展 Trident 用于配置的池集以及由属性和 `storagePools` 参数选择的池集。您可以单独使用参数或同时使用两者，以确保选择了相应的存储池集。

`excludeStoragePools` 参数用于专门排除与属性匹配的已列出池集。

模拟 **QoS** 策略

如果要设计模拟服务质量策略的存储类，请创建一个具有 `media` 属性为 `hdd` 或 `ssd` 的存储类。根据存储类中提到的 `media` 属性，Trident 将选择适当的后端来提供 `hdd` 或 `ssd` 聚合以匹配介质属性，然后将卷的配置引导到特定的聚合。因此，我们可以创建一个存储类 PREMIUM，其 `media` 属性设置为 `ssd`，可以将其归类为 PREMIUM QoS 策略。我们可以创建另一个存储类 STANDARD，将介质属性设置为 `hdd`，可以将其归类为 STANDARD QoS 策略。我们还可以使用存储类中的 `IOPS` 属性将配置重定向到可以定义为 QoS 策略的 Element 设备。

根据特定功能使用后端

存储类可以设计为在启用了精简和厚配置、快照、克隆和加密等功能的特定后端上指导卷配置。若要指定要使用的存储，请创建指定适当后端并启用所需功能的存储类。

虚拟池

虚拟池可用于所有 Trident 后端。您可以使用 Trident 提供的任何驱动程序为任何后端定义虚拟池。

虚拟池允许管理员在后端创建可通过存储类引用的抽象级别，以提高后端卷的灵活性和高效放置。可以使用相同的服务类别定义不同的后端。此外，可以在同一后端创建多个存储池，但具有不同的特征。当存储类配置有带有特定标签的选择器时，Trident 会选择与所有选择器标签匹配的后端来放置卷。如果存储类选择器标签与多个存储池匹配，Trident 将选择其中一个来配置卷。

虚拟池设计

创建后端时，通常可以指定一组参数。管理员无法使用相同的存储凭据和不同的参数集创建另一个后端。随着虚拟池的引入，这一问题得到了缓解。虚拟池是在后端和 Kubernetes Storage Class 之间引入的一个级别抽象，以便管理员可以以与后端无关的方式定义参数以及可以通过 Kubernetes Storage Classes 作为选择器引用的标签。可以为所有支持的 NetApp 后端使用 Trident 定义虚拟池。该列表包括 SolidFire/NetApp HCI、ONTAP 以及 Azure NetApp Files。

> 定义虚拟池时，建议不要尝试在后端定义中重新排列现有虚拟池的顺序。还建议不要编辑/修改现有虚拟池的属性，而是定义新的虚拟池。

模拟不同的服务级别/**QoS**

可以为模拟服务类设计虚拟池。使用 Cloud Volume Service for Azure NetApp Files 的虚拟池实现，让我们检查如何设置不同的服务类。使用多个标签配置 Azure NetApp Files 后端，代表不同的性能级别。将 `servicelevel` 方面设置为适当的性能水平，并在每个标签下添加其他必需的方面。现在创建映射到不同虚拟池的不同 Kubernetes Storage Classes。使用 `parameters.selector`字段，每个 StorageClass 调用哪些虚拟池可用于托管卷。

分配特定的一组方面

可以从单个存储后端设计具有特定方面的多个虚拟池。为此，请使用多个标签配置后端，并在每个标签下设置所需的方面。现在，使用映射到不同虚拟池的 `parameters.selector` 字段创建不同的 Kubernetes 存储类。在后端调配的卷将具有所选虚拟池中定义的方面。

影响存储配置的 **PVC** 特性

在创建 PVC 时，超出请求存储类的某些参数可能会影响 Trident 配置决策过程。

访问模式

通过 PVC 请求存储时，必填字段之一是访问模式。所需的模式可能会影响所选的托管存储请求的后端。

Trident 将尝试根据以下矩阵匹配与指定访问方法一起使用的存储协议。这与底层存储平台无关。

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	是	是	是（Raw 块）
NFS	是	是	是

在未配置 NFS 后端的情况下，提交给 Trident 部署的 ReadWriteMany PVC 请求将导致未配置任何卷。为此，请求者应使用适合其应用程序的访问模式。

卷操作

修改持久卷

持久卷是 Kubernetes 中的不可变对象，但有两个例外。创建后，可以修改回收策略和大小。但是，这并不能阻止在 Kubernetes 之外修改卷的某些方面。这可能是理想的，以便为特定应用定制卷，确保容量不会意外消耗，或者只是出于任何原因将卷移动到不同的存储控制器。

目前，Kubernetes 树内配置程序不支持 NFS、iSCSI 或 FC PV 的卷大小调整操作。Trident 支持扩展 NFS、iSCSI 和 FC 卷。

创建后，便无法修改 PV 的连接详细信息。

按需创建卷快照

Trident 支持使用 CSI 框架创建按需卷快照和从快照创建 PVC。快照为维护数据的时间点副本提供了一种方便的方法，并且在 Kubernetes 中具有独立于源 PV 的生命周期。这些快照可用于克隆 PVC。

从快照创建卷

Trident 还支持从卷快照创建 PersistentVolumes。要完成此操作，只需创建一个 PersistentVolumeClaim 并提及 `datasource`作为需要从中创建卷的必需快照。Trident 将通过使用快照上存在的数据创建卷来处理此 PVC。使用此功能，可以跨区域复制数据、创建测试环境、完全替换损坏或损坏的生产卷，或检索特定文件和目录并将其传输到另一个附加卷。

移动集群中的卷

存储管理员能够以不中断存储使用者的方式将卷在 ONTAP 集群中的聚合和控制器之间移动。此操作不会影响 Trident 或 Kubernetes 集群，只要目标聚合是 Trident 正在使用的 SVM 有权访问的聚合。重要的是，如果聚合已新添加到 SVM，则需要通过将其重新添加到 Trident 来刷新后端。这将触发 Trident 重新清点 SVM，以便识别新的聚合。

但是，Trident 不会自动支持跨后端移动卷。这包括同一集群中的 SVM 之间、集群之间或不同存储平台上的 SVM（即使该存储系统连接到 Trident）。

如果将卷复制到其他位置，则可以使用卷导入功能将当前卷导入 Trident。

扩展卷

Trident 支持调整 NFS、iSCSI 和 FC PV 的大小。这使用户能够通过 Kubernetes 层直接调整卷的大小。所有主要 NetApp 存储平台（包括 ONTAP 和 SolidFire/NetApp HCI 后端）都可以进行卷扩展。要允许稍后进行可能的扩展，请在与卷关联的 StorageClass 中将 `allowVolumeExpansion`设置为 `true`。每当永久卷需要调整大小时，将永久卷声明中的 `spec.resources.requests.storage`注释编辑为所需的卷大小。Trident 将自动调整存储集群上的卷大小。

将现有卷导入 **Kubernetes**

卷导入提供了将现有存储卷导入 Kubernetes 环境的功能。目前 `ontap-nas`、 `ontap-nas-flexgroup`、`solidfire-san`和 `azure-netapp-files`驱动程序支持此功能。此功能在将现有应用程序移植到 Kubernetes 或灾难恢复场景期间非常有用。

使用 ONTAP 和 `solidfire-san`驱动程序时，使用命令 `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml`将现有卷导入 Kubernetes 以由 Trident 管理。import volume 命令中使用的 PVC YAML 或 JSON 文件指向将 Trident 标识为配置程序的存储类。使用 NetApp HCI/SolidFire 后端时，请确保卷名称是唯一的。如果卷名重复，请将该卷克隆为唯一的名称，以便卷导入功能可以区分它们。

如果使用 `azure-netapp-files` 驱动程序，请使用命令 `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` 将卷导入 Kubernetes 以由 Trident 管理。这确保了唯一的卷引用。

执行上述命令时，Trident 将在后端找到卷并读取其大小。它将自动添加（并在必要时覆盖）配置的 PVC 的卷大小。然后，Trident 创建新的 PV，Kubernetes 将 PVC 绑定到 PV。

如果容器的部署需要特定的导入 PVC，则它将保持挂起状态，直到通过卷导入过程绑定 PVC/PV 对。在 PVC/PV 对绑定后，如果没有其他问题，容器应该启动。

Registry 服务

已将部署和管理注册表的存储记录在 "netapp.io" 的 "博客" 中。

日志记录服务

与其他 OpenShift 服务一样，日志记录服务使用 Ansible 部署，配置参数由提供给剧本的清单文件（又名主机）提供。将涵盖两种安装方法：在初始 OpenShift 安装期间部署日志记录和在 OpenShift 安装后部署日志记录。

> 从 Red Hat OpenShift 版本 3.9 开始，由于担心数据损坏，官方文档建议不要将 NFS 用于日志记录服务。这是基于 Red Hat 对其产品的测试。ONTAP NFS 服务器没有这些问题，并且可以轻松支持日志记录部署。最终，日志记录服务的协议选择取决于您，只需知道在使用 NetApp 平台时两者都可以很好地工作，如果这是您的偏好，则没有理由避免使用 NFS。

如果选择将 NFS 与日志记录服务一起使用，则需要将 Ansible 变量 `openshift_enable_unsupported_configurations`设置为 `true`以防止安装程序失败。

开始使用

日志记录服务可以可选地为应用程序以及 OpenShift 集群本身的核心操作部署。如果选择部署操作日志记录，则通过将变量 `openshift_logging_use_ops`指定为 `true`，将创建服务的两个实例。控制操作的日志记录实例的变量包含"ops"，而应用程序的实例不包含"ops"。

根据部署方法配置 Ansible 变量对于确保底层服务使用正确的存储非常重要。让我们来看看每个部署方法的选

项。

下表仅包含与日志记录服务相关的存储配置变量。您可以在 "Red Hat OpenShift 日志记录文档" 中找到其他选项，应根据您的部署情况进行查看、配置和使用。

下表中的变量将导致 Ansible 剧本使用提供的详细信息为日志记录服务创建 PV 和 PVC。此方法的灵活性远不如在 OpenShift 安装后使用组件安装剧本，但是，如果您有可用的现有卷，这也是一个选择。

变量	详细信息
`openshift_logging_storage_kind`	设置为 `nfs`让安装程序为日志记录服务创建 NFS PV。
`openshift_logging_storage_host`	NFS 主机的主机名或 IP 地址。这应设置为虚拟机的 dataLIF。
`openshift_logging_storage_nfs_directory`	NFS 导出的挂载路径。例如，如果卷连接为 `/openshift_logging`，则您将使用该路径作为此变量。
`openshift_logging_storage_volume_name`	要创建的 PV 的名称，例如 `pv_ose_logs`。
`openshift_logging_storage_volume_size`	NFS 导出的大小，例如 `100Gi`。

如果您的 OpenShift 集群已在运行，因此 Trident 已部署和配置，安装程序可以使用动态配置来创建卷。需要配置以下变量。

变量	详细信息
`openshift_logging_es_pvc_dynamic`	设置为 true 以使用动态配置的卷。
`openshift_logging_es_pvc_storage_class_name`	将在 PVC 中使用的存储类的名称。
`openshift_logging_es_pvc_size`	PVC 中请求的卷的大小。
`openshift_logging_es_pvc_prefix`	日志记录服务使用的 PVC 的前缀。
`openshift_logging_es_ops_pvc_dynamic`	设置为 `true`以将动态配置的卷用于 ops 日志记录实例。
`openshift_logging_es_ops_pvc_storage_class_name`	ops 日志记录实例的存储类的名称。
`openshift_logging_es_ops_pvc_size`	ops 实例的卷请求的大小。
`openshift_logging_es_ops_pvc_prefix`	ops 实例 PVC 的前缀。

部署日志记录堆栈

如果将日志记录部署为初始 OpenShift 安装过程的一部分，则只需遵循标准部署过程即可。Ansible 将配置和部署所需的服务和 OpenShift 对象，以便在 Ansible 完成后立即提供服务。

但是，如果您在初始安装后进行部署，则 Ansible 需要使用组件攻略。此过程可能会因 OpenShift 的不同版本而略有不同，因此请务必阅读并遵循"Red Hat OpenShift Container Platform 3.11 文档"您的版本。

指标服务

指标服务为管理员提供有关 OpenShift 集群状态、资源利用率和可用性的宝贵信息。这对于 pod 自动缩放功能也是必要的，许多组织将指标服务中的数据用于其退款和/或显示应用程序。

与日志记录服务一样，OpenShift 作为一个整体，Ansible 用于部署指标服务。此外，与日志记录服务一样，可以在集群的初始设置期间或使用组件安装方法运行后部署指标服务。下表包含为指标服务配置持久存储时重要的变量。

> 下表仅包含与指标服务相关的存储配置变量。文档中还有许多其他选项，应根据您的部署进行审查、配置和使用。

变量	详细信息
`openshift_metrics_storage_kind`	设置为 `nfs`让安装程序为日志记录服务创建 NFS PV。
`openshift_metrics_storage_host`	NFS 主机的主机名或 IP 地址。这应该设置为 SVM 的 dataLIF。
`openshift_metrics_storage_nfs_directory`	NFS 导出的挂载路径。例如，如果卷连接为 `/openshift_metrics`，则您将使用该路径作为此变量。
`openshift_metrics_storage_volume_name`	要创建的 PV 的名称，例如 `pv_ose_metrics`。
`openshift_metrics_storage_volume_size`	NFS 导出的大小，例如 `100Gi`。

如果您的 OpenShift 集群已在运行，因此 Trident 已部署和配置，安装程序可以使用动态配置来创建卷。需要配置以下变量。

变量	详细信息
`openshift_metrics_cassandra_pvc_prefix`	用于度量 PVC 的前缀。
`openshift_metrics_cassandra_pvc_size`	要请求的卷的大小。
`openshift_metrics_cassandra_storage_type`	要用于指标的存储类型，必须将其设置为 dynamic，以便 Ansible 使用适当的存储类创建 PVC。
`openshift_metrics_cassanda_pvc_storage_class_name`	要使用的存储类的名称。

部署指标服务

使用主机/清单文件中定义的相应 Ansible 变量，使用 Ansible 部署服务。如果您在 OpenShift 安装时部署，则 PV 将自动创建和使用。如果使用组件剧本进行部署，在 OpenShift 安装后，Ansible 会创建所需的任何 PVC，并在 Trident 为它们配置存储之后，部署服务。

上述变量以及部署过程可能会随着 OpenShift 的每个版本而变化。确保查看并遵循 "Red Hat 的 OpenShift 部署指南" 您的版本，以便为您的环境配置。

数据保护和灾难恢复

了解 Trident 和使用 Trident 创建的卷的保护和恢复选项。您应该为每个具有持久性要求的

应用程序制定数据保护和恢复策略。

Trident 复制和恢复

您可以创建备份以在发生灾难时还原 Trident。

Trident 复制

Trident 使用 Kubernetes CRD 来存储和管理自己的状态，并使用 Kubernetes 集群 etcd 来存储其元数据。

步骤

1. 使用 "Kubernetes：备份 etcd 集群" 备份 Kubernetes 集群 etcd。
2. 将备份项目放置在 FlexVol 卷上

> NetApp 建议您通过将 FlexVol 所在的 SVM 与另一个 SVM 建立 SnapMirror 关系来保护该 SVM。

Trident 恢复

使用 Kubernetes CRD 和 Kubernetes 集群 etcd snapshot，可以恢复 Trident。

步骤

1. 从目标 SVM 中，将包含 Kubernetes etcd 数据文件和证书的卷挂载到将设置为主节点的主机上。
2. 复制与 Kubernetes 集群相关的所有必需证书，位于 `/etc/kubernetes/pki` 下，以及 etcd 成员文件，位于 `/var/lib/etcd` 下。
3. 使用 "Kubernetes：恢复 etcd 集群" 从 etcd 备份还原 Kubernetes 集群。
4. 运行 `kubectl get crd`以验证所有 Trident 自定义资源已出现并检索 Trident 对象以验证所有数据可用。

SVM 复制和恢复

Trident 无法配置复制关系，但是，存储管理员可以使用 "ONTAP SnapMirror"来复制 SVM。

在发生灾难时，您可以激活 SnapMirror 目标 SVM 以开始提供数据。系统还原后，您可以切换回主系统。

关于此任务

使用 SnapMirror SVM 复制功能时，请考虑以下事项：

- 您应该为每个启用了 SVM-DR 的 SVM 创建一个独特的后端。
- 配置存储类，仅在需要时选择复制的后端，以避免将不需要复制的卷调配到支持 SVM-DR 的后端。
- 应用程序管理员应了解与复制相关的额外成本和复杂性，并在开始此过程之前仔细考虑其恢复计划。

SVM 复制

您可以使用 "ONTAP：SnapMirror SVM 复制" 创建 SVM 复制关系。

SnapMirror 允许您设置选项以控制要复制的内容。您需要知道在执行 使用 Trident 进行 SVM 恢复 时选择了哪

些选项。

- "-identity-preserve true" 复制整个 SVM 配置。
- "-discard-configs network" 不包括 LIF 和相关网络设置。
- "-identity-preserve false" 仅复制卷和安全配置。

使用 **Trident** 进行 **SVM** 恢复

Trident 不会自动检测 SVM 故障。在发生灾难时，管理员可以手动启动 Trident 故障转移到新的 SVM。

步骤

1. 取消计划和正在进行的 SnapMirror 传输，断开复制关系，停止源 SVM，然后激活 SnapMirror 目标 SVM。
2. 如果在配置 SVM 复制时指定了 `-identity-preserve false`或 `-discard-config network`，请更新 Trident 后端定义文件中的 `managementLIF`和 `dataLIF`。
3. 确认 `storagePrefix` 存在于 Trident 后端定义文件中。此参数无法更改。省略 `storagePrefix` 将导致后端更新失败。
4. 使用以下命令更新所有必需的后端，以反映新的目标 SVM 名称：

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. 如果指定了 `-identity-preserve false` 或 `discard-config network`，则必须重启所有应用程序 Pod。

> 如果指定 `-identity-preserve true`，则在激活目标 SVM 时，由 Trident 配置的所有卷都开始提供数据。

卷复制和恢复

Trident 无法配置 SnapMirror 复制关系，但存储管理员可以使用 "ONTAP SnapMirror 复制和恢复" 来复制由 Trident 创建的卷。

然后，您可以使用 "tridentctl volume import" 将恢复的卷导入 Trident。

> `ontap-nas-economy`、 `ontap-san-economy` 或 `ontap-flexgroup-economy` 驱动程序不支持导入。

Snapshot 数据保护

您可以使用以下方式保护和恢复数据：

- 用于创建持久卷 (PV) 的 Kubernetes 卷快照的外部快照控制器和 CRD。

  "卷快照"

- ONTAP Snapshots 可还原卷的整个内容或恢复单个文件或 LUN。

  "ONTAP 快照"

使用 Trident 自动化有状态应用程序的故障转移

Trident 的强制分离功能允许您自动从 Kubernetes 集群中的不正常节点分离卷，从而防止数据损坏并确保应用程序可用性。此功能在节点无响应或因维护而离线的情况下特别有用。

关于强制分离的详细信息

强制分离仅适用于 `ontap-san`、 `ontap-san-economy`、 `ontap-nas`和 `ontap-nas-economy`。在启用强制分离之前，必须在 Kubernetes 集群上启用非优雅节点关闭（NGNS）。默认情况下，Kubernetes 1.28 及更高版本已启用 NGNS。有关详细信息，请参阅 "Kubernetes：非 Graceful 节点关闭"。

> 使用 `ontap-nas` 或 `ontap-nas-economy` 驱动程序时，需要在后端配置中将 `autoExportPolicy` 参数设置为 `true`，以便 Trident 可以使用托管导出策略限制来自应用了污点的 Kubernetes 节点的访问。

> 由于 Trident 依赖于 Kubernetes NGNS，因此在重新安排所有无法忍受的工作负载之前，请勿从不正常的节点中删除 `out-of-service`污点。不计后果地应用或删除污点可能会危及后端数据保护。

当 Kubernetes 集群管理员将 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` taint 应用到节点并将 `enableForceDetach`设置为 `true`时，Trident 将确定节点状态并：

1. 停止对装载到该节点的卷的后端 I/O 访问。
2. 将 Trident 节点对象标记为 `dirty`（对新发布不安全）。

> Trident 控制器将拒绝新的发布卷请求，直到节点被 Trident 节点 Pod 重新限定（在被标记为 `dirty`之后）。在 `Trident` 能够验证节点 `clean`（对新发布安全）之前，将不接受使用已挂载 PVC 调度的任何工作负载（即使在集群节点健康且就绪之后）。

当节点运行状况恢复并清除污点时，Trident 将：

1. 识别并清除节点上过时的已发布路径。
2. 如果该节点处于 `cleanable`状态（已清除停用污点，并且该节点处于 `Ready`状态），并且所有过时、已发布的路径都是干净的，则 Trident 将重新接收该节点为 `clean`并允许新发布的卷进入该节点。

有关自动故障转移的详细信息

您可以通过与 "节点运行状况检查 (NHC) 操作器" 集成来自动执行强制分离过程。当发生节点故障时，NHC 会触发 Trident 节点修复（TNR），并通过在 Trident 命名空间中创建 TridentNodeRemediation CR 来定义故障节点，从而自动进行强制分离。TNR 仅在节点故障时创建，并在节点恢复联机或节点被删除后由 NHC 删除。

节点 **pod** 移除过程失败

自动故障转移选择要从故障节点中删除的工作负载。创建 TNR 后，TNR 控制器将节点标记为脏节点，防止任何新的卷发布，并开始删除强制拆卸支持的 Pod 及其卷附件。

自动故障转移支持强制拆卸支持的所有卷/PVC：

- 使用自动导出策略的 NAS 和 NAS-economy 卷（尚不支持 SMB）。
- SAN 和 SAN-economy 卷。

请参见 [关于强制分离的详细信息]。

默认行为：

- 使用强制拆卸支持的卷的 Pod 将从故障节点中删除。Kubernetes 会将这些重新调度到健康的节点上。
- 使用强制拆卸不支持的卷（包括非 Trident 卷）的 Pod 不会从故障节点中删除。
- 除非设置了 Pod 注释 `trident.netapp.io/podRemediationPolicy: delete`，否则不会从故障节点中删除无状态 Pod（非 PVC）。

覆盖 **Pod** 移除行为：

可以使用 Pod 注释自定义 Pod 移除行为： `trident.netapp.io/podRemediationPolicy[retain, delete]`。当发生故障转移时，会检查并使用这些注释。将注释应用于 Kubernetes deployment/replicaset pod 规范，以防止故障转移后注释消失：

- `retain` - 在自动故障转移期间，不会从故障节点中删除 Pod。
- `delete` - 在自动故障转移期间，Pod 将从故障节点中删除。

这些注释可以应用于任何 pod。

- 对于支持强制分离的卷，I/O 操作仅在故障节点上被阻止。
- 对于不支持强制分离的卷，存在数据损坏和多重连接问题的风险。

TridentNodeRemediation CR

TridentNodeRemediation (TNR) CR 定义失败的节点。TNR 的名称是失败节点的名称。

TNR 示例：

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR 状态：使用以下命令查看 TNR 的状态：
`kubectl get tnr <name> -n <trident-namespace>`

TNR 可能处于以下状态之一：

- 修复中：
  - 停止对装载到该节点的强制拆卸所支持的卷的后端 I/O 访问。
  - Trident 节点对象被标记为脏（对于新出版物不安全）。
  - 从节点中删除 Pod 和卷附件
- *NodeRecoveryPending*：
  - 控制器正在等待节点恢复联机。
  - 节点联机后，发布强制将确保节点干净，可供新卷发布。
- 如果从 K8s 中删除节点，TNR 控制器将删除 TNR 并停止对账。
- 成功：
  - 已成功完成所有修正和节点恢复步骤。节点已干净，可供新卷发布。
- 失败：
  - 不可恢复的错误。错误原因在 CR 的 status.message 字段中设置。

启用自动故障转移

前提条件：

- 在启用 automated-failover 之前，请确保已启用强制分离。有关详细信息，请参阅 [关于强制分离的详细信息]。
- 在 Kubernetes 集群中安装节点运行状况检查 (NHC)。
  - "安装 operator-sdk".
  - 在集群中安装 Operator Lifecycle Manager (OLM)（如果尚未安装）： `operator-sdk olm install`。
  - 安装 Node Health check Operator： `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`。

> 您还可以使用其他方法来检测节点故障，如下面的 [Integrating Custom Node Health Check Solutions] 部分所述。

有关详细信息，请参见 "节点健康检查运算符"。

步骤

1. 在 Trident 命名空间中创建 NodeHealthCheck (NHC) CR 以监视集群中的工作节点。示例：

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. 在 `trident` 命名空间中应用节点运行状况检查 CR。

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

上述 CR 配置为监视 K8s worker 节点的节点条件 Ready: false 和 Unknown。节点进入 Ready: false 或 Ready: Unknown 状态时将触发 Automated-Failover。

CR 中的 `unhealthyConditions`使用 0 秒宽限期。这导致自动故障转移在 K8s 设置节点条件 Ready: false 时立即触发，这是在 K8s 失去节点的心跳后设置的。K8s 的默认值为最后一次心跳后等待 40 秒，然后设置 Ready: false。可以在 K8s 部署选项中自定义此宽限期。

要查看其他配置选项，请参阅 "Node-Healthcheck-Operator 文档"。

其他设置信息

安装 Trident 并启用强制拆卸功能时，系统将在 Trident 命名空间中自动创建两个额外资源，以促进与 NHC 的集成：TridentNodeRemediationTemplate (TNRT) 和 ClusterRole。

TridentNodeRemediationTemplate (TNRT)：

TNRT 充当 NHC 控制器的模板，该控制器根据需要使用 TNRT 生成 TNR 资源。

```yaml
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole：

启用强制拆卸后，在安装过程中也会添加群集角色。这赋予了 NHC 对 Trident 命名空间中 TNR 的权限。

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

K8s 集群升级和维护

为了防止任何故障转移，请在 K8s 维护或升级期间暂停自动故障转移，其中节点预计会停机或重新启动。您可以通过修补 CR 来暂停 NHC CR（如上所述）：

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

这会暂停自动故障转移。要重新启用自动故障转移，请在维护完成后从规范中删除 pauseRequests。

限制

- 仅在强制分离支持的卷的故障节点上阻止 I/O 操作。只有使用强制分离支持的卷/PVC 的 Pod 才会自动移除。
- 自动故障转移和强制拆卸在 trident-controller pod 内运行。如果托管 trident-controller 的节点出现故障，则自动故障转移将被延迟，直到 K8s 将 pod 移动到健康节点。

集成自定义节点运行状况检查解决方案

您可以将 Node Healthcheck Operator 替换为备用节点故障检测工具以触发自动故障转移。为了确保与自动故障转移机制的兼容性，您的自定义解决方案应：

- 检测到节点故障时创建 TNR，使用故障节点的名称作为 TNR CR 名称。
- 当节点已恢复且 TNR 处于 Succeeded 状态时，请删除 TNR。

安全性

安全性

请使用此处列出的建议，以确保 Trident 的安装安全。

在自己的命名空间中运行 **Trident**

必须防止应用程序、应用程序管理员、用户和管理应用程序访问 Trident 对象定义或 Pod，以确保可靠的存储并阻止潜在的恶意活动。

要将其他应用程序和用户与 Trident 分离，请始终在自己的 Kubernetes 命名空间中安装 Trident(`trident`）。将 Trident 放在自己的命名空间中可确保只有 Kubernetes 管理人员才能访问 Trident Pod 和存储在命名空间 CRD 对象中的工件（如后端和 CHAP 机密（如果适用））。您应该确保只允许管理员访问 Trident 命名空间，从而访问 `tridentctl`应用程序。

对 **ONTAP SAN** 后端使用 **CHAP** 身份验证

Trident 支持基于 CHAP 的 ONTAP SAN 工作负载身份验证（使用 `ontap-san`和 `ontap-san-economy`驱动程序）。NetApp 建议使用双向 CHAP 与 Trident 进行主机和存储后端之间的身份验证。

对于使用 SAN 存储驱动程序的 ONTAP 后端，Trident 可以设置双向 CHAP 并通过 `tridentctl`管理 CHAP 用户名和机密。请参阅"准备使用 ONTAP SAN 驱动程序配置后端"以了解 Trident 如何在 ONTAP 后端上配置 CHAP。

将 **CHAP** 身份验证与 **NetApp HCI** 和 **SolidFire** 后端一起使用

NetApp 建议部署双向 CHAP，以确保主机与 NetApp HCI 和 SolidFire 后端之间的身份验证。Trident 使用一个秘密对象，其中每个租户包含两个 CHAP 密码。安装 Trident 后，它会管理 CHAP 机密并将其存储在相应 PV 的 `tridentvolume` CR 对象中。创建 PV 时，Trident 使用 CHAP 机密来启动 iSCSI 会话，并通过 CHAP 与 NetApp HCI 和 SolidFire 系统进行通信。

由 Trident 创建的卷不与任何卷访问组关联。

将 **Trident** 与 **NVE** 和 **NAE** 配合使用

NetApp ONTAP 提供静态数据加密，以便在磁盘被盗、退回或重新使用时保护敏感数据。有关详细信息，请参阅 "配置 NetApp 卷加密概述"。

- 如果在后端启用了 NAE，则在 Trident 中配置的任何卷都将启用 NAE。
  - 您可以将 NVE 加密标志设置为 `""`以创建启用 NAE 的卷。
- 如果未在后端启用 NAE，则在 Trident 中配置的任何卷都将启用 NVE，除非在后端配置中将 NVE 加密标志设置为 `false`（默认值）。

> 在启用 NAE 的后端上在 Trident 中创建的卷必须进行 NVE 或 NAE 加密。
>
> - 您可以在 Trident 后端配置中将 NVE 加密标志设置为 `true`，以覆盖 NAE 加密，并在每个卷的基础上使用特定的加密密钥。
> - 在启用 NAE 的后端上将 NVE 加密标志设置为 `false`可创建启用 NAE 的卷。无法通过将 NVE 加密标志设置为 `false`来禁用 NAE 加密。

- 您可以通过显式设置 NVE 加密标志为 `true`在 Trident 中手动创建 NVE 卷。

有关后端配置选项的更多信息，请参阅：

- "ONTAP SAN 配置选项"
- "ONTAP NAS 配置选项"

Linux 统一密钥设置 (LUKS)

您可以启用 Linux Unified Key Setup (LUKS) 来加密 Trident 上的 ONTAP SAN 和 ONTAP SAN ECONOMY 卷。Trident 支持 LUKS 加密卷的密码短语轮换和卷扩展。

在 Trident 中， LUKS 加密的卷使用 aes-xts-plain64 密码和模式，如 "NIST" 所推荐。

> ASA r2 系统不支持 LUKS 加密。有关 ASA r2 系统的信息，请参见 "了解 ASA r2 存储系统"。

开始之前

- 工作节点必须安装 cryptsetup 2.1 或更高版本（但低于 3.0）。有关更多信息，请访问 "Gitlab：cryptsetup"。
- 出于性能原因，NetApp 建议工作节点支持高级加密标准新指令 (AES-NI)。要验证 AES-NI 支持，请运行以下命令：

```
grep "aes" /proc/cpuinfo
```

如果没有返回任何内容，则表示您的处理器不支持 AES-NI。有关 AES-NI 的更多信息，请访问："Intel：高级加密标准指令 (AES-NI)"。

启用 LUKS 加密

您可以使用 Linux Unified Key Setup (LUKS) 为 ONTAP SAN 和 ONTAP SAN ECONOMY 卷启用每个卷的主机端加密。

步骤

1. 在后端配置中定义 LUKS 加密属性。有关 ONTAP SAN 后端配置选项的更多信息，请参阅 "ONTAP SAN 配置选项"。

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. 使用 `parameters.selector` 来定义使用 LUKS 加密的存储池。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. 创建一个包含 LUKS 密码短语的密钥。例如：

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

限制

LUKS 加密的卷无法利用 ONTAP 重复数据删除和压缩。

用于导入 **LUKS** 卷的后端配置

要导入 LUKS 卷，您必须在后端将 `luksEncryption`设置为(`true。 luksEncryption`选项告诉 Trident 卷是否符合 LUKS 标准(`true`）或不符合 LUKS 标准(`false`），如以下示例所示。

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

用于导入 **LUKS** 卷的 **PVC** 配置

要动态导入 LUKS 卷，请将注释 `trident.netapp.io/luksEncryption` 设置为 `true` 并在 PVC 中包含启用 LUKS 的存储类，如本示例所示。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

轮换 **LUKS** 密码短语

您可以轮换 LUKS 密码短语并确认轮换。

> 在验证密码短语不再被任何卷、快照或密码引用之前，请不要忘记该密码短语。如果引用的密码短语丢失，您可能无法挂载卷，并且数据将保持加密且无法访问。

关于此任务

当在指定新的 LUKS 密码短语后创建挂载卷的 Pod 时，会发生 LUKS 密码短语轮换。创建新 Pod 时，Trident 会将卷上的 LUKS 密码短语与密码中的活动密码短语进行比较。

- 如果卷上的密码与 secret 中的活动密码不匹配，则会发生旋转。
- 如果卷上的密码与 secret 中的活动密码匹配，则此 `previous-luks-passphrase` 参数将被忽略。

步骤

1. 添加 `node-publish-secret-name` 和 `node-publish-secret-namespace` StorageClass 参数。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. 识别卷或快照上的现有密码短语。

   卷

   ```
   tridentctl -d get volume luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

   ...luksPassphraseNames:["A"]
   ```

   Snapshot

   ```
   tridentctl -d get snapshot luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

   ...luksPassphraseNames:["A"]
   ```

3. 更新卷的 LUKS 密钥以指定新的和以前的密码。请确保 `previous-luke-passphrase-name` 和 `previous-luks-passphrase` 匹配之前的密码短语。

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: luks-pvc1
   stringData:
     luks-passphrase-name: B
     luks-passphrase: secretB
     previous-luks-passphrase-name: A
     previous-luks-passphrase: secretA
   ```

4. 创建挂载卷的新 pod。这是启动轮换所必需的。
5. 验证密码短语已轮换。

   卷

   ```
   tridentctl -d get volume luks-pvc1
   GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

   ...luksPassphraseNames:["B"]
   ```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

结果

当仅在卷和快照上返回新密码短语时，密码短语已轮换。

> 如果返回两个密码短语，例如 `luksPassphraseNames: ["B", "A"]`，则旋转是不完整的。您可以触发一个新的 pod 来尝试完成旋转。

启用卷扩展

您可以在 LUKS 加密的卷上启用卷扩展。

步骤

1. 启用 `CSINodeExpandSecret`功能门（beta 1.25+）。有关详细信息，请参见 "Kubernetes 1.25：使用 Secrets 进行节点驱动的 CSI 卷扩展"。
2. 添加 `node-expand-secret-name` 和 `node-expand-secret-namespace` StorageClass 参数。例如：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

结果

启动在线存储扩展时， kubelet 会将相应的凭据传递给驱动程序。

Kerberos 传输中加密

通过使用 Kerberos 在线加密，您可以为托管群集和存储后端之间的流量启用加密，从而提高数据访问安全性。

Trident 支持将 Kerberos 加密用于 ONTAP 作为存储后端：

- 本地 **ONTAP** - Trident 支持通过 NFSv3 和 NFSv4 连接从 Red Hat OpenShift 和上游 Kubernetes 集群到本地 ONTAP 卷的 Kerberos 加密。

可以创建、删除、调整大小、快照、克隆、只读克隆以及导入使用 NFS 加密的卷。

使用本地 **ONTAP** 卷配置运行中的 **Kerberos** 加密

您可以在托管集群与本地 ONTAP 存储后端之间的存储流量上启用 Kerberos 加密。

> 仅使用 `ontap-nas` 存储驱动程序支持针对具有本地 ONTAP 存储后端的 NFS 流量的 Kerberos 加密。

开始之前

- 请确保您拥有该 `tridentctl` 实用程序的访问权限。
- 确保您拥有对 ONTAP 存储后端的管理员访问权限。
- 确保您知道要从 ONTAP 存储后端共享的卷的名称。
- 确保已准备好 ONTAP 存储虚拟机，以支持 NFS 卷的 Kerberos 加密。有关说明，请参阅 "在 dataLIF 上启用 Kerberos"。
- 请确保您使用 Kerberos 加密的任何 NFSv4 卷配置正确。请参阅 NetApp NFSv4 域配置部分（第 13 页）"NetApp NFSv4 增强功能和最佳实践指南"。

添加或修改 **ONTAP** 导出策略

您需要向现有的 ONTAP 导出策略添加规则，或创建支持 Kerberos 加密的新导出策略，用于 ONTAP 存储 VM 根卷以及与上游 Kubernetes 集群共享的任何 ONTAP 卷。您添加的导出策略规则或创建的新导出策略需要支持以下访问协议和访问权限：

访问协议

使用 NFS、NFSv3 和 NFSv4 访问协议配置导出策略。

访问详细信息

根据您对卷的需求，您可以配置三种不同版本的 Kerberos 加密之一：

- **Kerberos 5** -（身份验证和加密）
- **Kerberos 5i** -（具有身份保护的身份验证和加密）
- **Kerberos 5p** -（具有身份和隐私保护的身份验证和加密）

使用适当的访问权限配置 ONTAP 导出策略规则。例如，如果群集将使用 Kerberos 5i 和 Kerberos 5p 加密的混合方式装载 NFS 卷，请使用以下访问设置：

类型	只读访问	读/写访问权限	超级用户访问
UNIX	已启用	已启用	已启用
Kerberos 5i	已启用	已启用	已启用
Kerberos 5p	已启用	已启用	已启用

有关如何创建 ONTAP 导出策略和导出策略规则，请参阅以下文档：

- "创建导出策略"
- "将规则添加到导出策略"

创建存储后端

您可以创建包含 Kerberos 加密功能的 Trident 存储后端配置。

关于此任务

在创建配置 Kerberos 加密的存储后端配置文件时，可以使用 `spec.nfsMountOptions` 参数指定三个不同版本的 Kerberos 加密之一：

- `spec.nfsMountOptions: sec=krb5` （身份验证和加密）
- `spec.nfsMountOptions: sec=krb5i`（具有身份保护的身份验证和加密）
- `spec.nfsMountOptions: sec=krb5p` （具有身份和隐私保护的身份验证和加密）

仅指定一个 Kerberos 级别。如果在参数列表中指定多个 Kerberos 加密级别，则仅使用第一个选项。

步骤

1. 在托管群集上，使用以下示例创建存储后端配置文件。使用环境中的信息替换括号 <> 中的值：

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. 使用上一步中创建的配置文件创建后端：

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

如果后端创建失败，则后端配置有问题。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs
```

在识别并更正配置文件的问题后，您可以再次运行 create 命令。

创建存储类

您可以创建存储类，以使用 Kerberos 加密配置卷。

关于此任务

在创建存储类对象时，可以使用 `mountOptions` 参数指定 Kerberos 加密的三个不同版本之一：

- `mountOptions: sec=krb5` （身份验证和加密）
- `mountOptions: sec=krb5i`（具有身份保护的身份验证和加密）
- `mountOptions: sec=krb5p` （具有身份和隐私保护的身份验证和加密）

仅指定一个 Kerberos 级别。如果在参数列表中指定多个 Kerberos 加密级别，则仅使用第一个选项。如果在存储后端配置中指定的加密级别与在存储类对象中指定的级别不同，则存储类对象优先。

步骤

1. 使用以下示例创建 StorageClass Kubernetes 对象：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. 创建存储类：

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. 确保已创建存储类：

```
kubectl get sc ontap-nas-sc
```

此时将显示与以下内容类似的输出：

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

配置卷

创建存储后端和存储类后，现在可以配置卷。有关说明，请参阅 "预配卷"。

使用 **Azure NetApp Files** 卷配置运行中的 **Kerberos** 加密

您可以在托管集群与单个 Azure NetApp Files 存储后端或 Azure NetApp Files 存储后端虚拟池之间的存储流量上启用 Kerberos 加密。

开始之前

- 请确保已在托管红帽 OpenShift 群集上启用 Trident。
- 请确保您拥有该 `tridentctl` 实用程序的访问权限。
- 通过注意要求并按照 "Azure NetApp Files 文档"中的说明，确保已为 Kerberos 加密准备 Azure NetApp Files 存储后端。
- 请确保您使用 Kerberos 加密的任何 NFSv4 卷配置正确。请参阅 NetApp NFSv4 域配置部分（第 13 页）"NetApp NFSv4 增强功能和最佳实践指南"。

创建存储后端

您可以创建包含 Kerberos 加密功能的 Azure NetApp Files 存储后端配置。

关于此任务

当您创建配置 Kerberos 加密的存储后端配置文件时，您可以将其定义为应在以下两种可能的级别之一应用：

- 使用 `spec.kerberos` 字段的*存储后端级别*
- 使用 `spec.storage.kerberos` 字段的 虚拟池级别

在虚拟池级别定义配置时，将使用存储类中的标签选择池。

在任一级别，您都可以指定 Kerberos 加密的三个不同版本之一：

- `kerberos: sec=krb5` （身份验证和加密）
- `kerberos: sec=krb5i`（具有身份保护的身份验证和加密）
- `kerberos: sec=krb5p` （具有身份和隐私保护的身份验证和加密）

步骤

1. 在托管群集上，使用以下示例之一创建存储后端配置文件，具体取决于需要定义存储后端的位置（存储后端级别或虚拟池级别）。使用环境中的信息替换括号 <> 中的值：

存储后端级别示例

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

虚拟池级别示例

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. 使用上一步中创建的配置文件创建后端：

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

如果后端创建失败，则后端配置有问题。您可以通过运行以下命令查看日志以确定原因：

```
tridentctl logs
```

在识别并更正配置文件的问题后，您可以再次运行 create 命令。

创建存储类

您可以创建存储类，以使用 Kerberos 加密配置卷。

步骤

1. 使用以下示例创建 StorageClass Kubernetes 对象：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. 创建存储类：

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. 确保已创建存储类：

```
kubectl get sc -sc-nfs
```

此时将显示与以下内容类似的输出：

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

配置卷

创建存储后端和存储类后，现在可以配置卷。有关说明，请参阅 "预配卷"。