安装 Trident Protect

Trident

NetApp
July 01, 2026

目录

安装 Trident Protect . . . 1
Trident Protect 要求 . . . 1
Trident Protect Kubernetes 集群兼容性 . . . 1
Trident Protect 存储后端兼容性 . . . 1
nas-economy 卷的要求 . . . 2
使用 KubeVirt VM 保护数据 . . . 2
SnapMirror 复制的要求 . . . 3
安装和配置 Trident Protect . . . 4
安装 Trident Protect . . . 4
安装 Trident Protect CLI 插件 . . . 7
安装 Trident Protect CLI 插件 . . . 7
查看 Trident CLI 插件帮助 . . . 9
启用命令自动完成 . . . 9
自定义 Trident Protect 安装 . . . 11
指定 Trident Protect 容器资源限制 . . . 11
自定义安全上下文约束 . . . 12
配置其他 Trident Protect helm 图表设置 . . . 13
将 Trident Protect Pod 限制到特定节点 . . . 14

安装 Trident Protect

Trident Protect 要求

首先验证操作环境、应用程序群集、应用程序和许可证的准备情况。确保您的环境满足这些要求，以部署和运行 Trident Protect。

Trident Protect Kubernetes 集群兼容性

Trident Protect 与各种完全托管和自我托管的 Kubernetes 产品兼容，包括：

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0（ONTAP iSCSI）
- SUSE Rancher
- VMware Tanzu Portfolio
- 上游 Kubernetes

- 仅 Linux 计算节点支持 Trident Protect 备份。备份操作不支持 Windows 计算节点。
- 确保安装 Trident Protect 的集群配置了正在运行的快照控制器和相关 CRD。要安装快照控制器，请参阅 "这些说明"。
- 确保至少存在一个 VolumeSnapshotClass。有关详细信息，请参阅 "VolumeSnapshotClass"。
- 安装 Trident Protect 需要 Helm 4.x 或更高版本。

Trident Protect 存储后端兼容性

Trident Protect 支持以下存储后端：

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP 存储阵列
- Google Cloud NetApp Volumes
- Azure NetApp Files

请确保存储后端满足以下要求：

- 请确保连接到集群的 NetApp 存储使用 Trident 24.02 或更高版本（建议使用 Trident 24.10）。
- 确保您拥有 NetApp ONTAP 存储后端。
- 确保已配置用于存储备份的对象存储桶。

- 创建计划用于应用程序或应用程序数据管理操作的任何应用程序命名空间。Trident Protect 不会为您创建这些命名空间；如果在自定义资源中指定不存在的命名空间，则操作将失败。

nas-economy 卷的要求

Trident Protect 支持到 nas-economy 卷的备份和还原操作。当前不支持到 nas-economy 卷的快照、克隆和 SnapMirror 复制。您需要为计划用于 Trident Protect 的每个 nas-economy 卷启用快照目录。

某些应用程序与使用快照目录的卷不兼容。对于这些应用程序，需要通过在 ONTAP 存储系统上运行以下命令来隐藏快照目录：

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

您可以通过为每个 nas-economy 卷运行以下命令来启用快照目录，将 `<volume-UUID>` 替换为要更改的卷的 UUID：

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

您可以通过将 Trident 后端配置选项 `snapshotDir`设置为 `true`来默认为新卷启用快照目录。现有卷不受影响。

使用 KubeVirt VM 保护数据

Trident Protect 在数据保护操作期间为 KubeVirt 虚拟机提供文件系统冻结和解冻功能，以确保数据一致性。虚拟机冻结操作的配置方法和默认行为因 Trident Protect 版本而异，较新的版本通过 Helm 图表参数提供简化的配置。

在还原操作期间，不会还原为虚拟机 (VM) 创建的任何 `VirtualMachineSnapshots`。

Trident Protect 25.10 及更高版本

Trident Protect 在数据保护操作期间自动冻结和解冻 KubeVirt 文件系统，以确保一致性。从 Trident Protect 25.10 开始，您可以在 Helm chart 安装期间使用 `vm.freeze` 参数禁用此行为。默认情况下，该参数处于启用状态。

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 至 25.06

从 Trident Protect 24.10.1 开始，Trident Protect 在数据保护操作期间自动冻结和解冻 KubeVirt 文件系统。或者，您可以使用以下命令禁用此自动行为：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 不会在数据保护操作期间自动确保 KubeVirt VM 文件系统的一致状态。如果要使用 Trident Protect 24.10 保护 KubeVirt VM 数据，则需要在数据保护操作之前手动为文件系统启用冻结/解冻功能。这可确保文件系统处于一致状态。

您可以通过 "配置虚拟化" 配置 Trident Protect 24.10 来管理数据保护操作期间虚拟机文件系统的冻结和解冻，然后使用以下命令：

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

SnapMirror 复制的要求

NetApp SnapMirror 复制可用于以下 ONTAP 解决方案的 Trident Protect：

- 本地 NetApp FAS、AFF 和 ASA 系统。目前 ASA r2 系统不支持使用 Trident 保护进行 SnapMirror 复制。
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP 集群对 SnapMirror 复制的要求

如果计划使用 SnapMirror 复制，请确保您的 ONTAP 群集满足以下要求：

- **NetApp Trident**：NetApp Trident 必须存在于使用 ONTAP 作为后端的源和目标 Kubernetes 集群上。Trident Protect 支持使用由以下驱动程序支持的存储类的 NetApp SnapMirror 技术进行复制：
  - `ontap-nas`：NFS
  - `ontap-san`：iSCSI
  - `ontap-san`：FC
  - `ontap-san`: NVMe/TCP（需要最低 ONTAP 版本 9.15.1）
- 许可证：必须在源和目标 ONTAP 集群上启用使用数据保护捆绑包的 ONTAP SnapMirror 异步许可证。请参阅 "ONTAP 中的 SnapMirror 许可概述"以获取更多信息。

从 ONTAP 9.10.1 开始，所有许可证都以 NetApp 许可证文件 (NLF) 的形式交付，该文件是启用多个功能的单个文件。有关详细信息，请参见 "ONTAP One 附带的许可证"。

仅支持 SnapMirror 异步保护。

SnapMirror 复制的对等关系注意事项

如果您计划使用存储后端对等，请确保您的环境满足以下要求：

- 集群和 **SVM**：必须对等 ONTAP 存储后端。请参阅 "集群和 SVM 对等概述"以获取更多信息。

  确保在两个 ONTAP 集群之间的复制关系中使用的 SVM 名称是唯一的。

- **NetApp Trident** 和 **SVM**：对等远程 SVM 必须可用于目标集群上的 NetApp Trident。
- 托管后端：您需要在 Trident Protect 中添加和管理 ONTAP 存储后端，以创建复制关系。

用于 **SnapMirror** 复制的 **Trident / ONTAP** 配置

Trident Protect 要求您配置至少一个同时支持源和目标集群复制的存储后端。如果源和目标集群相同，目标应用程序应使用与源应用程序不同的存储后端，以获得最佳弹性。

Kubernetes 集群 **SnapMirror** 复制需求

确保 Kubernetes 集群满足以下要求：

- **AppVault** 可访问性：源和目标集群必须具有网络访问权限才能从 AppVault 读取和写入以进行应用程序对象复制。
- 网络连接：配置防火墙规则、存储桶权限和 IP 允许列表，以启用两个集群和 AppVault 跨 WAN 之间的通信。

许多企业环境在 WAN 连接上实施严格的防火墙策略。在配置复制之前，请与基础架构团队一起验证这些网络要求。

安装和配置 Trident Protect

如果您的环境符合 Trident Protect 的要求，您可以按照以下步骤在集群上安装 Trident Protect。您可以从 NetApp 获取 Trident Protect，也可以从您自己的私有注册表安装它。如果您的集群无法访问 Internet，从私有注册表安装会很有帮助。

安装 Trident Protect

从 **NetApp** 安装 **Trident Protect**

步骤

1. 添加 Trident Helm 存储库：

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

2. 使用 Helm 安装 Trident Protect。将 `<name-of-cluster>` 替换为集群名称，该名称将分配给集群并用于标识集群的备份和快照：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect
```

3. （可选）要启用调试日志记录（建议用于故障排除），请使用：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect
```

调试日志记录有助于 NetApp 支持疑难解答问题，而无需更改日志级别或重现问题。

从私有注册表安装 **Trident Protect**

如果您的 Kubernetes 集群无法访问互联网，您可以从私有映像注册表安装 Trident Protect。在这些示例中，使用环境中的信息替换括号中的值：

步骤

1. 将以下映像拉到本地计算机，更新标记，然后将其推送到专用注册表：

```
docker.io/netapp/controller:26.02.0
docker.io/netapp/restic:26.02.0
docker.io/netapp/kopia:26.02.0
docker.io/netapp/kopiablockrestore:26.02.0
docker.io/netapp/trident-autosupport:26.02.0
docker.io/netapp/exechook:26.02.0
docker.io/netapp/resourcebackup:26.02.0
docker.io/netapp/resourcerestore:26.02.0
docker.io/netapp/resourcedelete:26.02.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

例如：

```
docker pull docker.io/netapp/controller:26.02.0
```

```
docker tag docker.io/netapp/controller:26.02.0 <private-registry-url>/controller:26.02.0
```

```
docker push <private-registry-url>/controller:26.02.0
```

要获取 Helm 图表，请首先在具有 Internet 访问权限的计算机上使用 `helm pull trident-protect --version 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart` 下载 Helm 图表，然后将生成的 `trident-protect-100.2602.0.tgz`文件复制到离线环境，并在最后步骤中使用 `helm install trident-protect ./trident-protect-100.2602.0.tgz`而不是存储库引用进行安装。

2. 创建 Trident Protect 系统命名空间：

```
kubectl create ns trident-protect
```

3. 登录到注册表：

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. 创建用于专用注册表身份验证的拉取密钥：

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. 添加 Trident Helm 存储库：

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

6. 创建一个名为 `protectValues.yaml` 的文件。确保其中包含以下 Trident Protect 设置：

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

> ⓘ imageRegistry 和 imagePullSecrets 值适用于所有组件图像，包括 resourcebackup 和 resourcerestore。如果将图像推送到注册表中的特定存储库路径（例如 example.com:443/my-repo），请在注册表字段中包含完整路径。这将确保从 <private-registry-url>/<image-name>:<tag> 中提取所有图像。

7. 使用 Helm 安装 Trident Protect。将 <name_of_cluster> 替换为集群名称，该名称将分配给集群并用于标识集群的备份和快照：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. （可选）要启用调试日志记录（建议用于故障排除），请使用：

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

调试日志记录有助于 NetApp 支持疑难解答问题，而无需更改日志级别或重现问题。

> ⓘ 要查看其他 Helm 图表配置选项，包括 AutoSupport 设置和命名空间筛选，请参阅 "自定义 Trident Protect 安装"。

安装 Trident Protect CLI 插件

您可以使用 Trident Protect 命令行插件（Trident tridentctl 实用程序的扩展）来创建 Trident Protect 自定义资源（CR）并与之交互。

安装 Trident Protect CLI 插件

在使用命令行实用程序之前，需要将其安装在用于访问群集的计算机上。根据您的计算机使用的是 x64 还是 ARM CPU，请执行以下步骤。

为 **Linux AMD64 CPU** 下载插件

步骤

1. 下载 Trident Protect CLI 插件：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-amd64
```

为 **Linux ARM64 CPU** 下载插件

步骤

1. 下载 Trident Protect CLI 插件：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-arm64
```

下载适用于 **Mac AMD64 CPU** 的插件

步骤

1. 下载 Trident Protect CLI 插件：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-amd64
```

下载适用于 **Mac ARM64 CPU** 的插件

步骤

1. 下载 Trident Protect CLI 插件：

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-arm64
```

1. 为插件二进制文件启用执行权限：

```
chmod +x tridentctl-protect
```

2. 将插件二进制文件复制到 PATH 变量中定义的位置。例如， `/usr/bin` 或 `/usr/local/bin`（您可能需要提升权限）：

```
cp ./tridentctl-protect /usr/local/bin/
```

3. 或者，您可以将插件二进制文件复制到主目录中的某个位置。在这种情况下，建议确保该位置是 PATH 变量的一部分：

```
cp ./tridentctl-protect ~/bin/
```

将插件复制到 PATH 变量中的某个位置，可以通过键入 `tridentctl-protect`或 `tridentctl protect`从任何位置使用插件。

查看 **Trident CLI** 插件帮助

您可以使用内置插件帮助功能获取有关插件功能的详细帮助：

步骤

1. 使用帮助功能查看使用指南：

```
tridentctl-protect help
```

启用命令自动完成

安装 Trident Protect CLI 插件后，可以启用某些命令的自动完成功能。

启用 **Bash shell** 的自动完成

步骤

1. 创建完成脚本：

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. 在主目录中创建包含脚本的新目录：

```
mkdir -p ~/.bash/completions
```

3. 将下载的脚本移动到 `~/.bash/completions` 目录：

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. 将以下行添加到主目录中的 `~/.bashrc` 文件：

```
source ~/.bash/completions/tridentctl-completion.bash
```

启用 **Z shell** 的自动完成

步骤

1. 创建完成脚本：

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. 在主目录中创建包含脚本的新目录：

```
mkdir -p ~/.zsh/completions
```

3. 将下载的脚本移动到 `~/.zsh/completions` 目录：

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. 将以下行添加到主目录中的 `~/.zprofile` 文件：

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

结果

下次登录 shell 时，您可以使用 tridentctl-protect 插件自动完成命令。

自定义 Trident Protect 安装

您可以自定义 Trident Protect 的默认配置，以满足您环境的特定要求。

指定 Trident Protect 容器资源限制

安装 Trident Protect 后，您可以使用配置文件为 Trident Protect 容器指定资源限制。设置资源限制使您能够控制 Trident Protect 操作占用集群资源的数量。

步骤

1. 创建一个名为 `resourceLimits.yaml` 的文件。
2. 根据您的环境需求，使用 Trident Protect 容器的资源限制选项填充文件。

   以下配置文件示例显示了可用设置，并包含每个资源限制的默认值：

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
```

```
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. 应用 `resourceLimits.yaml` 文件中的值：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

自定义安全上下文约束

安装 Trident Protect 后，您可以使用配置文件修改 Trident Protect 容器的 OpenShift 安全上下文约束 (SCC)。这些约束定义了 Red Hat OpenShift 集群中 Pod 的安全限制。

步骤

1. 创建一个名为 `sccconfig.yaml` 的文件。
2. 将 SCC 选项添加到文件中，并根据环境需要修改参数。

   以下示例显示了 SCC 选项的参数的默认值：

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

   下表描述了 SCC 选项的参数：

参数	说明	默认
create	确定是否可以创建 SCC 资源。仅当 `scc.create` 设置为 `true` 且 Helm 安装过程识别 OpenShift 环境时，才会创建 SCC 资源。如果不在 OpenShift 上操作，或者如果 `scc.create` 设置为 `false`，则不会创建任何 SCC 资源。	true
name	指定 SCC 的名称。	trident-protect-job
优先级	定义 SCC 的优先级。优先级值较高的 SCC 在优先级值较低的 SCC 之前进行评估。	1

3. 应用 `sccconfig.yaml` 文件中的值：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

这将用 `sccconfig.yaml` 文件中指定的值替换默认值。

配置其他 **Trident Protect helm** 图表设置

您可以自定义 AutoSupport 设置和命名空间筛选以满足您的特定要求。下表描述了可用的配置参数：

参数	类型	说明
autoSupport.proxy	string	为 NetApp AutoSupport 连接配置代理 URL。使用此选项可通过代理服务器路由支持包上传。示例：`http://my.proxy.url`。
autoSupport.insecure	布尔值	设置为 `true`时跳过 `AutoSupport` 代理连接的 `TLS` 验证。仅用于不安全的代理连接。（默认值： `false`）
autoSupport.enabled	布尔值	启用或禁用每日 Trident Protect AutoSupport 捆绑包上传。设置为 `false`时，计划的每日上传将被禁用，但您仍然可以手动生成支持捆绑包。（默认值： `true`）
restoreSkipNamespaceAnnotations	string	要从备份和还原操作中排除的命名空间注释的逗号分隔列表。允许您根据注释筛选命名空间。
restoreSkipNamespaceLabels	string	要从备份和还原操作中排除的命名空间标签的逗号分隔列表。允许您根据标签过滤命名空间。

您可以使用 YAML 配置文件或命令行标志配置这些选项：

使用 **YAML** 文件

步骤

1. 创建配置文件并将其命名为 `values.yaml`。
2. 在创建的文件中，添加要自定义的配置选项。

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. 使用正确的值填充 `values.yaml` 文件后，应用配置文件：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

使用 **CLI** 标志

步骤

1. 使用以下带有 `--set` 标志的命令指定单个参数：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

将 **Trident Protect Pod** 限制到特定节点

您可以使用 Kubernetes nodeSelector 节点选择约束来根据节点标签控制哪些节点有资格运行 Trident Protect Pod。默认情况下，Trident Protect 仅限于运行 Linux 的节点。您可以根据需要进一步自定义这些限制。

步骤

1. 创建一个名为 `nodeSelectorConfig.yaml` 的文件。
2. 将 nodeSelector 选项添加到文件中，并根据环境的需要修改文件以添加或更改节点标签以进行限制。例如

，以下文件包含默认操作系统限制，但也针对特定区域和应用程序名称：

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. 应用 `nodeSelectorConfig.yaml` 文件中的值：

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

这会将默认限制替换为您在 `nodeSelectorConfig.yaml` 文件中指定的限制。