安装Trident Protect

Trident

NetApp
January 15, 2026

目录

安装Trident Protect

Trident保护要求

首先，请验证您的运行环境、应用程序集群、应用程序和许可证是否准备就绪。确保您的环境满足部署和运行Trident Protect 的这些要求。

Trident Protect Kubernetes 集群兼容性

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

- 亚马逊弹性 Kubernetes 服务 (EKS)
- 谷歌 Kubernetes 引擎 (GKE)
- Microsoft Azure Kubernetes 服务 (AKS)
- 红帽 OpenShift
- SUSE Rancher
- VMware Tanzu 产品组合
- 上游 Kubernetes

- Trident Protect备份仅支持Linux计算节点。Windows 计算节点不支持备份操作。
- 确保安装Trident Protect 的集群已配置运行中的快照控制器和相关的 CRD。要安装快照控制器，请参阅 "这些说明"。

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

请确保您的存储后端满足以下要求：

- 确保连接到集群的NetApp存储使用的是Trident 24.02 或更高版本（建议使用Trident 24.10）。
- 请确保您拥有NetApp ONTAP存储后端。
- 请确保您已配置用于存储备份的对象存储桶。
- 创建您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；如果您在自定义资源中指定了不存在的命名空间，则操作将失败。

nas-economy 容量要求

Trident Protect 支持对 nas-economy 卷进行备份和恢复操作。目前不支持将快照、克隆和SnapMirror复制到 nas-economy 卷。您需要为计划与Trident Protect 一起使用的每个 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，您需要通过在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
```

您可以通过对每个 nas-economy 卷运行以下命令来启用快照目录，并将命令替换为 `<volume-UUID>`使用要更改的卷的 UUID：

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

您可以通过设置Trident后端配置选项，为新卷默认启用快照目录。 `snapshotDir`到 `true`。现有卷数不受影响。

使用 KubeVirt 虚拟机保护数据

Trident Protect 24.10 和 24.10.1 及更高版本在保护运行在 KubeVirt VM 上的应用程序时，行为有所不同。对于这两个版本，您都可以在数据保护操作期间启用或禁用文件系统冻结和解冻结。

在恢复操作期间，任何 `VirtualMachineSnapshots`为虚拟机 (VM) 创建的配置文件无法恢复。

Trident Protect 24.10

Trident Protect 24.10 在数据保护操作期间不会自动确保 KubeVirt VM 文件系统的一致性状态。如果您想使用Trident Protect 24.10 保护您的 KubeVirt VM 数据，则需要在执行数据保护操作之前手动启用文件系统的冻结/解冻功能。这样可以确保文件系统处于一致状态。

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

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

Trident Protect 24.10.1 及更高版本

从Trident Protect 24.10.1 开始， Trident Protect 会在数据保护操作期间自动冻结和解冻 KubeVirt 文件系统。您也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
```

SnapMirror复制的要求

NetApp SnapMirror复制功能可与Trident Protect 配合使用，适用于以下ONTAP解决方案：

- 本地部署的NetApp FAS、 AFF和ASA集群
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP集群对SnapMirror复制的要求

如果您计划使用SnapMirror复制功能，请确保您的ONTAP集群满足以下要求：

- * NetApp Trident *：使用ONTAP作为后端服务的源 Kubernetes 集群和目标 Kubernetes 集群上都必须存在NetApp Trident 。Trident Protect 支持使用NetApp SnapMirror技术进行复制，该技术使用以下驱动程序支持的存储类：
  - `ontap-nas`：极品飞车
  - `ontap-san`：iSCSI
  - `ontap-san`：FC
  - `ontap-san`：NVMe/TCP（最低要求ONTAP版本 9.15.1）
- 许可证：使用数据保护包的ONTAP SnapMirror异步许可证必须在源 ONTAP 集群和目标ONTAP集群上启用。请参阅 "ONTAP中的SnapMirror许可概述"了解更多信息。

  从ONTAP 9.10.1 开始，所有许可证均以NetApp许可证文件 (NLF) 的形式交付，这是一个可以启用多种功能的单个文件。请参阅"ONTAP One 附带的许可证"了解更多信息。

  仅支持SnapMirror异步保护。

SnapMirror复制的对等连接注意事项

如果您计划使用存储后端对等互连，请确保您的环境满足以下要求：

- 集群和 **SVM**： ONTAP存储后端必须相互连接。请参阅 "集群和SVM对等连接概述"了解更多信息。

  确保两个ONTAP集群之间复制关系中使用的 SVM 名称是唯一的。

- * NetApp Trident和 SVM*：对等远程 SVM 必须可供目标集群上的NetApp Trident使用。
- 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 要求您至少配置一个支持源集群和目标集群复制的存储后端。如果源集群和目标集群相同，为了获得最佳弹性，目标应用程序应该使用与源应用程序不同的存储后端。

SnapMirror复制的 **Kubernetes** 集群要求

请确保您的 Kubernetes 集群满足以下要求：

- **AppVault** 可访问性：源集群和目标集群都必须具有网络访问权限，才能从 AppVault 读取数据和写入数据，以实现应用程序对象复制。
- 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 之间通过 WAN 进行通信。

许多企业环境在广域网连接中实施严格的防火墙策略。在配置复制之前，请与您的基础架构团队确认这些网络要求。

安装并配置Trident Protect

如果您的环境满足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，或者从您自己的私有注册表中安装它。如果您的集群无法访问互联网，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 仓库：

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

2. 使用 Helm 安装Trident Protect。代替 `<name-of-cluster>`集群名称将分配给集群，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 100.2506.0 --create
-namespace --namespace trident-protect
```

从私有注册表安装**Trident Protect**

如果您的 Kubernetes 集群无法访问互联网，您可以从私有镜像仓库安装Trident Protect。在这些示例中，请将括号中的值替换为您环境中的信息：

步骤

1. 将以下镜像拉取到本地计算机，更新标签，然后将其推送到您的私有镜像仓库：

```
netapp/controller:25.06.0
netapp/restic:25.06.0
netapp/kopia:25.06.0
netapp/trident-autosupport:25.06.0
netapp/exechook:25.06.0
netapp/resourcebackup:25.06.0
netapp/resourcerestore:25.06.0
netapp/resourcedelete:25.06.0
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

例如：

```
docker pull netapp/controller:25.06.0
```

```
docker tag netapp/controller:25.06.0 <private-registry-
url>/controller:25.06.0
```

```
docker push <private-registry-url>/controller:25.06.0
```

2. 创建Trident Protect 系统命名空间：

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

3. 登录注册表：

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

4. 创建用于私有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 仓库：

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

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

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. 使用 Helm 安装Trident Protect。代替 `<name_of_cluster>`集群名称将分配给集群，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 100.2506.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

安装Trident Protect CLI 插件

您可以使用Trident Protect 命令行插件，它是Trident的一个扩展。 `tridentctl`用于创建和与Trident Protect 自定义资源 (CR) 交互的实用程序。

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集群的计算机上。根据您的机器使用的是 x64 还是ARM CPU，请按照以下步骤操作。

下载适用于 **Linux 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/25.06.0/tridentctl-protect-linux-amd64
```

下载适用于 **Linux 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/25.06.0/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/25.06.0/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/25.06.0/tridentctl-protect-macos-arm64
```

1. 启用插件二进制文件的执行权限：

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

2. 将插件二进制文件复制到 PATH 环境变量中定义的位置。例如， `/usr/bin`或者 `/usr/local/bin`（您可能需要更高的权限）：

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

3. 您也可以选择将插件二进制文件复制到您主目录中的某个位置。在这种情况下，建议确保该位置已添加到您的 PATH 环境变量中：

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

将插件复制到 PATH 环境变量中的某个位置，即可通过键入以下命令来使用该插件。 `tridentctl-protect`或者 `tridentctl protect`从任何地点。

查看Trident CLI 插件帮助

您可以使用插件内置的帮助功能来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
```

启用命令自动补全

安装Trident Protect CLI 插件后，您可以为某些命令启用自动补全功能。

为 **Bash shell** 启用自动补全功能

步骤

1. 下载完成脚本：

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/25.06.0/tridentctl-completion.bash
```

2. 在您的用户目录下创建一个新目录，用于存放脚本：

```
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 `~/.bashrc`您主目录中的文件：

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

为 **Z shell** 启用自动补全

步骤

1. 下载完成脚本：

```
curl -L -O https://github.com/NetApp/tridentctl-
protect/releases/download/25.06.0/tridentctl-completion.zsh
```

2. 在您的用户目录下创建一个新目录，用于存放脚本：

```
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 `~/.zprofile`您主目录中的文件：

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

结果

下次登录 shell 时，您可以使用 tridentctl-protect 插件进行命令自动补全。

定制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来指定Trident Protect 容器的资源限制。设置资源限制可以控制Trident Protect 操作消耗集群资源的程度。

步骤

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

   以下示例配置文件显示了可用设置，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. 应用以下值 `resourceLimits.yaml`文件：

```
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来修改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. 创建一个名为的文件 `sccconfig.yaml`。
2. 在文件中添加 SCC 选项，并根据您的环境需要修改参数。

   以下示例显示了 SCC 选项的参数默认值：

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

下表描述了SCC选项的参数：

参数	描述	默认
create	确定是否可以创建 SCC 资源。仅当 `scc.create`设置为 `true`Helm 安装过程会识别 OpenShift 环境。如果不是在 OpenShift 上运行，或者如果 `scc.create`设置为 `false`不会创建 SCC 资源。	true
name	指定SCC的名称。	三叉戟保护工作
优先事项	确定 SCC 的优先级。优先级较高的SCC会优先于优先级较低的SCC进行评估。	1

3. 应用以下值 `sccconfig.yaml`文件：

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

这将用指定的值替换默认值。 `sccconfig.yaml`文件。

配置其他Trident Protect 舵图设置

您可以自定义AutoSupport设置和命名空间过滤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	描述
自动支持代理	string	为NetApp AutoSupport连接配置代理 URL。使用此功能通过代理服务器路由支持包上传。例子：`http://my.proxy.url` 。
自动支持.不安全	布尔值	设置为“跳过AutoSupport代理连接的 TLS 验证” `true` 。仅用于不安全的代理连接。 （默认： `false` ）
自动支持已启用	布尔值	启用或禁用每日Trident Protect AutoSupport捆绑包上传。设置为 `false`每日定时上传功能已禁用，但您仍然可以手动生成支持包。 （默认： `true` ）

参数	类型	描述
恢复跳过命名空间注释	string	要从备份和恢复操作中排除的命名空间注释的逗号分隔列表。允许您根据注释过滤命名空间。
restoreSkipNamespaceLabels	string	要从备份和恢复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. 创建一个配置文件并将其命名为 `values.yaml`。
2. 在您创建的文件中，添加您想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. 填写完之后 `values.yaml`将包含正确值的文件应用配置文件：

```
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. 使用以下命令： `--set`用于指定各个参数的标志：

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

将Trident Protect Pod 限制在特定节点上

您可以使用 Kubernetes nodeSelector 节点选择约束，根据节点标签来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条件。

步骤

1. 创建一个名为的文件 `nodeSelectorConfig.yaml`。
2. 在文件中添加 nodeSelector 选项，并修改文件以添加或更改节点标签，从而根据您的环境需要进行限制。例如，以下文件包含默认的操作系统限制，但同时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. 应用以下值 `nodeSelectorConfig.yaml`文件：

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

这将默认限制替换为您在设置中指定的限制。 `nodeSelectorConfig.yaml`文件。