Trident Protectのインストール

Trident

NetApp
July 01, 2026

目次

Trident Protectのインストール

Trident Protect の要件

まず、運用環境、アプリケーション クラスタ、アプリケーション、ライセンスの準備状況を確認します。Trident Protectを導入および運用するには、環境がこれらの要件を満たしていることを確認してください。

Trident Protect Kubernetes クラスタの互換性

Trident Protect は、以下を含む幅広いフルマネージドおよびセルフマネージド Kubernetes サービスと互換性があります：

- 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 ポートフォリオ
- アップストリーム Kubernetes

- Trident Protect バックアップは Linux コンピューティング ノードでのみサポートされます。Windows コンピューティング ノードはバックアップ操作ではサポートされていません。
- Trident Protect をインストールするクラスタが、実行中のスナップショット コントローラと関連する CRD で構成されていることを確認してください。スナップショット コントローラをインストールするには、 "これらの手順"を参照してください。
- 少なくとも1つのVolumeSnapshotClassが存在することを確認してください。詳細については、"VolumeSnapshotClass"を参照してください。
- Trident Protect をインストールするには Helm 4.x 以降が必要です。

Trident Protect ストレージバックエンドの互換性

Trident Protect は次のストレージ バックエンドをサポートしています：

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP ストレージアレイ
- Google Cloud NetApp Volumes
- Azure NetApp Files

ストレージ バックエンドが次の要件を満たしていることを確認します：

- クラスタに接続されたNetAppストレージがTrident 24.02以降を使用していることを確認します（Trident 24.10を推奨）。
- NetApp ONTAP ストレージバックエンドがあることを確認してください。
- バックアップを保存するためのオブジェクト ストレージ バケットが設定されていることを確認します。
- アプリケーションまたはアプリケーション データ管理操作に使用する予定のアプリケーション名前空間を作成します。Trident Protect ではこれらの名前空間は作成されません。カスタム リソースに存在しない名前空間を指定すると、操作は失敗します。

nas-economyボリュームの要件

Trident Protect は、nas-economy ボリュームへのバックアップおよびリストア処理をサポートします。スナップショット、クローン、および SnapMirror レプリケーションの nas-economy ボリュームへの実行は、現在サポートされていません。Trident Protect で使用する予定の各 nas-economy ボリュームについて、スナップショット ディレクトリを有効にする必要があります。

一部のアプリケーションは、スナップショット ディレクトリを使用するボリュームと互換性がありません。これらのアプリケーションでは、ONTAP ストレージ システムで次のコマンドを実行してスナップショット ディレクトリを非表示にする必要があります：

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

スナップショットディレクトリを有効にするには、各nas-economyボリュームに対して次のコマンドを実行し、 `<volume-UUID>`を変更したいボリュームのUUIDに置き換えます：

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

新しいボリュームのスナップショットディレクトリをデフォルトで有効にするには、Tridentバックエンド構成オプション `snapshotDir`を `true`に設定します。既存のボリュームは影響を受けません。

KubeVirt VM でデータを保護

Trident Protectは、データ保護操作中にKubeVirt仮想マシンのファイルシステムのフリーズとアンフリーズ機能を提供し、データの整合性を確保します。VMフリーズ操作の設定方法とデフォルトの動作はTrident Protectのバージョンによって異なり、新しいリリースではHelmチャートパラメータを通じて簡素化された設定が提供されます。

復元操作中は、仮想マシン（VM）用に作成された `VirtualMachineSnapshots`は復元されません。

Trident Protect 25.10以降

Trident Protectは、データ保護処理中にKubeVirtファイルシステムを自動的にフリーズおよびアンフリーズして、整合性を確保します。Trident Protect 25.10以降では、Helmチャートのインストール時に`vm.freeze`パラメータを使用してこの動作を無効にできます。このパラメータはデフォルトで有効になっています。

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

Trident Protect 24.10.1～25.06

Trident Protect 24.10.1以降、Trident Protectはデータ保護操作中にKubeVirtファイルシステムを自動的にフリーズおよびアンフリーズします。オプションで、次のコマンドを使用してこの自動動作を無効にすることができます：

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

Trident Protect 24.10

Trident Protect 24.10は、データ保護処理中にKubeVirt VMファイルシステムの整合性のある状態を自動的に保証するものではありません。KubeVirt VMデータをTrident Protect 24.10を使用して保護する場合は、データ保護処理の前に、ファイルシステムのフリーズ/アンフリーズ機能を手動で有効にする必要があります。これにより、ファイルシステムが整合性のある状態になることが保証されます。

Trident Protect 24.10を設定して、"仮想化の設定"を使用し、次のコマンドを実行することで、データ保護操作中にVMファイルシステムの凍結と解凍を管理できます：

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

SnapMirror レプリケーションの要件

NetApp SnapMirrorレプリケーションは、以下のONTAPソリューションでTrident Protectと併用できます：

- オンプレミスのNetApp FAS、AFF、ASAシステム。Trident保護を使用したSnapMirrorレプリケーションは、現在ASA r2システムではサポートされていません。
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

SnapMirror レプリケーションのための ONTAP クラスタ要件

SnapMirror レプリケーションを使用する場合は、ONTAP クラスタが次の要件を満たしていることを確認してください：

- **NetApp Trident**：NetApp Tridentは、ONTAPをバックエンドとして利用するソースKubernetesクラスタとデスティネーションKubernetesクラスタの両方に存在する必要があります。Trident Protectは、次のドライバでサポートされるストレージクラスを使用して、NetApp SnapMirrorテクノロジによるレプリケーションをサポートします：
  - `ontap-nas`：NFS
  - `ontap-san`：iSCSI
  - `ontap-san`：FC
  - `ontap-san`：NVMe/TCP（ONTAP バージョン 9.15.1 以降が必要）
- ライセンス：ONTAP SnapMirror データ保護バンドルを使用する非同期ライセンスは、ソースとデスティネーション クラスタの両方で有効にする必要があります。詳細については、 "ONTAP における SnapMirror ライセンスの概要"を参照してください。

  ONTAP 9.10.1以降では、すべてのライセンスがNetAppライセンス ファイル（NLF）として提供されます。これは、複数の機能を有効にする単一のファイルです。詳細については、"ONTAP Oneに含まれるライセンス" を参照してください。

  SnapMirror非同期保護のみがサポートされています。

SnapMirror レプリケーションのピアリングに関する考慮事項

ストレージ バックエンド ピアリングを使用する予定の場合は、環境が次の要件を満たしていることを確認してください：

- * クラスタと SVM * ： ONTAP ストレージ バックエンドはピアリングする必要があります。詳細については、 "クラスタとSVMのピアリングの概要"を参照してください。

  2つのONTAPクラスタ間のレプリケーション関係で使用されるSVM名が一意であることを確認してください。

- **NetApp Trident**および**SVM**：ピアリングされたリモートSVMは、デスティネーション クラスタ上のNetApp Tridentで使用できる必要があります。
- 管理されたバックエンド：レプリケーション関係を作成するには、Trident ProtectでONTAPストレージバックエンドを追加して管理する必要があります。

SnapMirror レプリケーション用の Trident / ONTAP 設定

Trident Protect では、ソース クラスタとデスティネーション クラスタの両方のレプリケーションをサポートするストレージ バックエンドを少なくとも 1 つ設定する必要があります。ソース クラスタとデスティネーション クラスタが同じ場合、復元力を最大限に高めるには、デスティネーション アプリケーションでソース アプリケーションとは異なるストレージ バックエンドを使用する必要があります。

SnapMirror レプリケーションの **Kubernetes** クラスタの要件

Kubernetes クラスタが次の要件を満たしていることを確認します。

- **AppVault** アクセシビリティ：ソース クラスタとデスティネーション クラスタの両方が、アプリケーション オブジェクトのレプリケーションのために AppVault への読み取りと書き込みを行うためのネットワーク アクセスを持っている必要があります。
- ネットワーク接続：ファイアウォール ルール、バケット権限、IP 許可リストを設定して、両方のクラスタと AppVault 間の WAN 経由の通信を有効にします。

> 多くの企業環境では、WAN 接続全体に厳格なファイアウォール ポリシーが実装されています。レプリケーションを設定する前に、これらのネットワーク要件をインフラストラクチャ チームに確認してください。

Trident Protectのインストールと設定

環境がTrident Protectの要件を満たしている場合は、次の手順に従ってクラスタにTrident Protectをインストールできます。Trident ProtectはNetAppから入手するか、独自のプライベート レジストリからインストールできます。クラスタがインターネットにアクセスできない場合は、プライベート レジストリからインストールすると便利です。

Trident Protectのインストール

NetAppからTrident Protectをインストールする

手順

1. Trident Helm リポジトリを追加します：

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

2. Helmを使用してTrident Protectをインストールします。 `<name-of-cluster>`をクラスタ名に置き換えます。このクラスタ名はクラスタに割り当てられ、クラスタのバックアップとSnapshotを識別するために使用されます：

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

3. オプションで、デバッグ ログを有効にするには（トラブルシューティングに推奨）、次のコマンドを使用します：

```
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
```

デバッグログは、ログ レベルの変更や問題の再現を必要とせずに、NetApp サポートが問題のトラブルシューティングを行うのに役立ちます。

プライベートレジストリから**Trident Protect**をインストールする

Kubernetes クラスタがインターネットにアクセスできない場合は、プライベート イメージ レジストリから Trident Protect をインストールできます。これらの例では、括弧内の値を環境の情報に置き換えます：

手順

1. 次のイメージをローカル マシンにプルし、タグを更新して、プライベート レジストリにプッシュします：

```
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
```

次に例を示します。

```
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
```

Helmチャートを取得するには、まず `helm pull trident-protect --version 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart`を使用してインターネットにアクセスできるマシンにHelmチャートをダウンロードし、その結果の `trident-protect-100.2602.0.tgz`ファイルをオフライン環境にコピーして、最後のステップでリポジトリ参照の代わりに `helm install trident-protect ./trident-protect-100.2602.0.tgz`を使用してインストールします。

2. Trident Protect システムネームスペースを作成します。

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

3. レジストリにログインします：

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

4. プライベート レジストリ認証に使用するプル シークレットを作成します：

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

5. Trident Helm リポジトリを追加します：

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

6. `protectValues.yaml`という名前のファイルを作成します。次のTrident Protect設定が含まれていることを確認してください：

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

> `imageRegistry`および `imagePullSecrets`の値は、 `resourcebackup`および `resourcerestore`を含むすべてのコンポーネントイメージに適用されます。レジストリ内の特定のリポジトリパスにイメージをプッシュする場合（例：`example.com:443/my-repo`）、レジストリフィールドに完全パスを含めてください。これにより、すべてのイメージが`<private-registry-url>/<image-name>:<tag>`から取得されます。

7. Helmを使用してTrident Protectをインストールします。 `<name_of_cluster>`をクラスタ名に置き換えます。このクラスタ名はクラスタに割り当てられ、クラスタのバックアップとSnapshotを識別するために使用されます：

```
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. オプションで、デバッグ ログを有効にするには（トラブルシューティングに推奨）、次のコマンドを使用します：

```
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
```

デバッグログは、ログ レベルの変更や問題の再現を必要とせずに、NetApp サポートが問題のトラブルシューティングを行うのに役立ちます。

AutoSupport設定や名前空間フィルタリングなどの追加のHelmチャート設定オプションについては、"Trident Protectのインストールをカスタマイズする"を参照してください。

Trident Protect CLI プラグインをインストールします

Trident Protect コマンドライン プラグインを使用できます。これは Trident `tridentctl` ユーティリティの拡張機能であり、Trident Protect カスタム リソース（CR）の作成と操作に使用できます。

Trident Protect CLI プラグインをインストールします

コマンドライン ユーティリティを使用する前に、クラスターにアクセスするために使用するマシンにそれをインストールする必要があります。マシンが x64 と ARM のどちらの CPU を使用しているかに応じて、次の手順に従ってください。

Linux AMD64 CPU用プラグインをダウンロード

手順

1. Trident Protect CLI プラグインをダウンロードします：

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

Linux ARM64 CPU用プラグインをダウンロード

手順

1. Trident Protect CLI プラグインをダウンロードします：

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

Mac AMD64 CPU用プラグインをダウンロード

手順

1. Trident Protect CLI プラグインをダウンロードします：

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

Mac ARM64 CPU用プラグインをダウンロード

手順

1. Trident Protect CLI プラグインをダウンロードします：

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

1. プラグイン バイナリの実行権限を有効にします：

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

2. プラグインのバイナリを PATH 変数で定義されている場所にコピーします。例えば、 `/usr/bin`または `/usr/local/bin`（昇格した権限が必要になる場合があります）：

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

3. オプションで、プラグインのバイナリをホームディレクトリ内の場所にコピーすることもできます。この場合、場所が PATH 変数の一部であることを確認することをお勧めします：

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

プラグインを PATH 変数の場所にコピーすると、 `tridentctl-protect`または `tridentctl protect`と入力することで、どこからでもプラグインを使用できるようになります。

Trident CLI プラグインのヘルプを表示

プラグインの機能に関する詳細なヘルプを取得するには、組み込みのプラグイン ヘルプ機能を使用できます：

手順

1. ヘルプ機能を使用して使用方法のガイダンスを表示します：

```
tridentctl-protect help
```

コマンドの自動補完を有効にする

Trident Protect CLI プラグインをインストールしたら、特定のコマンドの自動補完を有効にすることができます。

Bash シェルの自動補完を有効にする

手順

1. 完了スクリプトを作成します：

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

2. スクリプトを格納するための新しいディレクトリをホーム ディレクトリに作成します：

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

3. ダウンロードしたスクリプトを `~/.bash/completions`ディレクトリに移動します：

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

4. ホームディレクトリ内の `~/.bashrc`ファイルに次の行を追加します：

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

Z シェルの自動補完を有効にする

手順

1. 完了スクリプトを作成します：

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

2. スクリプトを格納するための新しいディレクトリをホーム ディレクトリに作成します：

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

3. ダウンロードしたスクリプトを `~/.zsh/completions`ディレクトリに移動します：

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

4. ホームディレクトリ内の `~/.zprofile`ファイルに次の行を追加します：

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

結果

次回のシェルログイン時には、tridentctl-protect プラグインによるコマンドの自動補完を使用できます。

Trident Protectのインストールをカスタマイズする

環境の特定の要件を満たすように Trident Protect のデフォルト設定をカスタマイズできます。

Trident Protectコンテナのリソース制限を指定する

Trident Protectのインストール後、設定ファイルを使用してTrident Protectコンテナのリソース制限を指定できます。リソース制限を設定すると、Trident Protect操作によって消費されるクラスタのリソース量を制御できます。

手順

1. `resourceLimits.yaml`という名前のファイルを作成します。
2. 環境のニーズに応じて、Trident Protectコンテナのリソース制限オプションをファイルに入力します。

   次の構成ファイルの例は、使用可能な設定を示しており、各リソース制限のデフォルト値が含まれています：

```
---
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. `resourceLimits.yaml`ファイルから値を適用します：

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

セキュリティコンテキスト制約をカスタマイズする

Trident Protect のインストール後、設定ファイルを使用して Trident Protect コンテナの OpenShift セキュリティコンテキスト制約（SCC）を変更できます。これらの制約は、Red Hat OpenShift クラスタ内のポッドのセキュリティ制限を定義します。

手順

1. `sccconfig.yaml`という名前のファイルを作成します。
2. ファイルに SCC オプションを追加し、環境のニーズに応じてパラメータを変更します。

   次の例は、SCC オプションのパラメータのデフォルト値を示しています：

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

この表は、SCC オプションのパラメータについて説明しています：

パラメータ	概要	デフォルト
作成する	SCC リソースを作成できるかどうかを決定します。SCC リソースは、 `scc.create`が `true`に設定されていて、Helm インストールプロセスがOpenShift環境を識別した場合にのみ作成されます。OpenShiftで動作していない場合、または `scc.create`が `false`に設定されている場合、SCC リソースは作成されません。	true
名前	SCCの名前を指定します。	trident-protect-job
優先度	SCC の優先度を定義します。優先度の値が高い SCC は、値が低い SCC よりも先に評価されます。	1

3. `sccconfig.yaml`ファイルから値を適用します：

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

これにより、デフォルト値が `sccconfig.yaml`ファイルで指定された値に置き換えられます。

追加のTrident Protectヘルムチャート設定を構成する

AutoSupportの設定と名前空間フィルタリングをカスタマイズして、特定の要件を満たすことができます。次の表に、使用可能な設定パラメータを示します：

パラメータ	タイプ	概要
autoSupport.proxy	string	NetApp AutoSupport接続のプロキシURLを設定します。これを使用して、プロキシ サーバ経由でサポートバンドルのアップロードをルーティングします。例：`http://my.proxy.url`。

パラメータ	タイプ	概要
autoSupport.insecure	ブーリアン	`true`に設定すると、`AutoSupport`プロキシ接続の`TLS`検証をスキップします。安全でないプロキシ接続にのみ使用してください。（デフォルト：`false`）
autoSupport.enabled	ブーリアン	Trident Protect AutoSupportバンドルの日次アップロードを有効または無効にします。 `false`に設定すると、スケジュールされた日次アップロードは無効になりますが、サポート バンドルを手動で生成することはできます。（デフォルト： `true`）
restoreSkipNamespaceAnnotations	string	バックアップおよびリストア処理から除外するネームスペースアノテーションのカンマ区切りリスト。アノテーションに基づいてネームスペースをフィルタリングできます。
restoreSkipNamespaceLabels	string	バックアップおよび復元操作から除外する名前空間ラベルのコンマ区切りリスト。ラベルに基づいて名前空間をフィルタリングできます。

これらのオプションは、YAML 構成ファイルまたはコマンドライン フラグを使用して設定できます：

YAMLファイルを使用する

手順

1. 設定ファイルを作成し、 `values.yaml`という名前を付けます。
2. 作成したファイルに、カスタマイズする設定オプションを追加します。

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

3. `values.yaml`ファイルに正しい値を入力したら、構成ファイルを適用します：

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

CLI フラグを使用する

手順

1. `--set`フラグを指定して次のコマンドを使用し、個々のパラメータを指定します。

```
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
```

特定のノードへの**Trident Protect**ポッドの制限

Kubernetes の nodeSelector ノード選択制約を使用して、ノードラベルに基づいて Trident Protect ポッドを実行できるノードを制御できます。デフォルトでは、Trident Protect は Linux を実行しているノードに制限されます。ニーズに応じてこれらの制約をさらにカスタマイズできます。

手順

1. `nodeSelectorConfig.yaml`という名前のファイルを作成します。
2. nodeSelectorオプションをファイルに追加し、環境のニーズに応じて制限するノード ラベルを追加または変更するようにファイルを修正します。たとえば、次のファイルにはデフォルトの OS 制限が含まれてい

ますが、特定の地域とアプリ名もターゲットにしています：

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

3. `nodeSelectorConfig.yaml`ファイルから値を適用します：

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

これにより、デフォルトの制限が、 `nodeSelectorConfig.yaml`ファイルで指定した制限に置き換えられます。