Trident Protect를 설치합니다

Trident

NetApp
November 14, 2025

목차

Trident Protect를 설치합니다

Trident 보호 요구 사항

운영 환경, 애플리케이션 클러스터, 애플리케이션 및 라이센스의 준비 상태를 확인하는 것으로 시작하십시오. Trident Protect를 배포 및 운영하기 위해 사용자 환경이 이러한 요구 사항을 충족하는지 확인합니다.

Trident는 Kubernetes 클러스터 호환성을 보호합니다

Trident Protect는 다음과 같은 완전관리형 및 자가 관리형 Kubernetes 오퍼링과 호환됩니다.

- Amazon Elastic Kubernetes Service(EKS)
- Google Kubernetes Engine(GKE)
- Microsoft Azure Kubernetes Service(AKS)
- Red Hat OpenShift
- 수세 목장
- VMware Tanzu 포트폴리오
- 업스트림 Kubernetes

- Trident Protect 백업은 Linux 컴퓨트 노드에서만 지원됩니다. Windows 컴퓨팅 노드에서는 백업 작업이 지원되지 않습니다.
- Trident Protect를 설치하는 클러스터가 실행 중인 스냅샷 컨트롤러와 관련 CRD로 구성되어 있는지 확인합니다. 스냅샷 컨트롤러를 설치하려면 을 "참조하십시오"참조하십시오.

Trident는 스토리지 백엔드 호환성을 보호합니다

Trident Protect는 다음 스토리지 백엔드를 지원합니다.

- NetApp ONTAP용 Amazon FSx
- Cloud Volumes ONTAP
- ONTAP 스토리지 어레이
- Google Cloud NetApp 볼륨
- Azure NetApp Files

스토리지 백엔드가 다음 요구 사항을 충족하는지 확인합니다.

- 클러스터에 연결된 NetApp 스토리지가 Trident 24.02 이상(Trident 24.10 권장)을 사용하는지 확인하세요.
- NetApp ONTAP 스토리지 백엔드가 있는지 확인합니다.
- 백업을 저장할 오브젝트 스토리지 버킷을 구성했는지 확인합니다.
- 응용 프로그램 또는 응용 프로그램 데이터 관리 작업에 사용할 응용 프로그램 네임스페이스를 만듭니다. Trident Protect는 이러한 네임스페이스를 자동으로 만들지 않습니다. 사용자 지정 리소스에 존재하지 않는 네임스페이스를

지정하면 작업이 실패합니다.

NAS 경제 볼륨에 대한 요구 사항

Trident Protect는 NAS 경제 볼륨에 대한 백업 및 복원 작업을 지원합니다. 현재 NAS 경제 볼륨으로의 스냅샷, 클론, SnapMirror 복제는 지원되지 않습니다. Trident Protect와 함께 사용할 각 NAS 경제 볼륨에 대해 스냅샷 디렉토리를 활성화해야 합니다.

일부 애플리케이션은 스냅샷 디렉토리를 사용하는 볼륨과 호환되지 않습니다. 이러한 애플리케이션의 경우 ONTAP 스토리지 시스템에서 다음 명령을 실행하여 스냅샷 디렉토리를 숨겨야 합니다.

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

각 NAS 경제 볼륨에 대해 다음 명령을 실행하여 스냅샷 디렉토리를 설정할 수 있으며, 변경할 볼륨의 UUID로 바꿀 수 `<volume-UUID>` 있습니다.

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

Trident 백엔드 구성 옵션을 로 `true` 설정하면 기본적으로 새 볼륨에 대해 스냅샷 디렉토리를 설정할 수 `snapshotDir` 있습니다. 기존 볼륨은 영향을 받지 않습니다.

KubeVirt VM으로 데이터 보호

Trident Protect는 데이터 보호 작업 중에 KubeVirt 가상 머신에 파일 시스템 동결 및 동결 해제 기능을 제공하여 데이터 일관성을 보장합니다. VM 동결 작업에 대한 구성 방법과 기본 동작은 Trident Protect 버전마다 다르며, 최신 릴리스에서는 Helm 차트 매개변수를 통해 간소화된 구성을 제공합니다.

복원 작업 중에 `VirtualMachineSnapshots` 가상 머신(VM)에 대해 생성된 항목은 복원되지 않습니다.

Trident Protect 25.10 이상

Trident Protect는 일관성을 보장하기 위해 데이터 보호 작업 중에 KubeVirt 파일 시스템을 자동으로 동결 및 동결 해제합니다. Trident protect 25.10부터 다음을 사용하여 이 동작을 비활성화할 수 있습니다. `vm.freeze` Helm 차트 설치 중 매개변수. 해당 매개변수는 기본적으로 활성화되어 있습니다.

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

Trident 프로텍트 24.10.1~25.06

Trident Protect 24.10.1부터 Trident Protect는 데이터 보호 작업 중에 KubeVirt 파일 시스템이 자동으로 작동 중지되고 작동 중지되지 않습니다. 다음 명령을 사용하여 이 자동 동작을 비활성화할 수도 있습니다.

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

Trident Protect 24.10

Trident Protect 24.10은 데이터 보호 작업 중에 KubeVirt VM 파일 시스템의 일관된 상태를 자동으로 보장하지 않습니다. Trident Protect 24.10을 사용하여 KubeVirt VM 데이터를 보호하려면 데이터 보호 작업 전에 파일 시스템에 대해 고정/고정 해제 기능을 수동으로 활성화해야 합니다. 이렇게 하면 파일 시스템이 정합성 보장 상태가 됩니다.

를 사용하여 데이터 보호 작업 중에 VM 파일 시스템의 고정 및 고정 해제를 관리하도록 Trident Protect 24.10을 구성한 후 다음 명령을 사용하여 관리할 수 "가상화 구성"있습니다.

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

SnapMirror 복제에 대한 요구 사항

NetApp SnapMirror 복제는 다음 ONTAP 솔루션의 Trident Protect와 함께 사용할 수 있습니다.

- 온프레미스 NetApp FAS, AFF, ASA 클러스터
- NetApp ONTAP Select를 참조하십시오
- NetApp Cloud Volumes ONTAP를 참조하십시오
- NetApp ONTAP용 Amazon FSx

SnapMirror 복제를 위한 **ONTAP** 클러스터 요구 사항

SnapMirror 복제를 사용하려는 경우 ONTAP 클러스터가 다음 요구 사항을 충족하는지 확인하십시오.

- **NetApp Trident**: NetApp Trident는 백엔드로 ONTAP을 활용하는 소스 및 대상 Kubernetes 클러스터 모두에 존재해야 합니다. Trident Protect는 다음 드라이버를 통해 지원되는 스토리지 클래스를 사용하여 NetApp SnapMirror 기술을 통한 복제를 지원합니다.
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` : FC
  - `ontap-san` : NVMe/TCP(최소 ONTAP 버전 9.15.1 필요)
- * 라이센스 *: 소스 및 대상 ONTAP 클러스터 모두에서 데이터 보호 번들을 사용하는 ONTAP SnapMirror 비동기

라이센스를 활성화해야 합니다. 자세한 내용은 을 "ONTAP의 SnapMirror 라이센스 개요" 참조하십시오.

ONTAP 9.10.1부터 모든 라이센스는 여러 기능을 사용할 수 있는 단일 파일인 NetApp 라이센스 파일(NLF)로 제공됩니다. 자세한 내용은 을 "ONTAP One에 포함된 라이선스" 참조하십시오.

SnapMirror 비동기 보호만 지원됩니다.

SnapMirror 복제에 대한 피어링 고려 사항

스토리지 백엔드 피어링을 사용하려는 경우 환경이 다음 요구 사항을 충족하는지 확인하십시오.

- * 클러스터 및 SVM *: ONTAP 스토리지 백엔드를 피어링해야 합니다. 자세한 내용은 을 "클러스터 및 SVM 피어링 개요" 참조하십시오.

  두 ONTAP 클러스터 간의 복제 관계에 사용되는 SVM 이름이 고유한지 확인합니다.

- **NetApp Trident** 및 **SVM**: 피어링된 원격 SVM은 대상 클러스터의 NetApp Trident에서 사용할 수 있어야 합니다.
- * 관리되는 백엔드 *: 복제 관계를 만들려면 Trident Protect에서 ONTAP 스토리지 백엔드를 추가 및 관리해야 합니다.

SnapMirror 복제를 위한 **Trident/ONTAP** 구성

Trident Protect를 사용하려면 소스 및 대상 클러스터 모두에 대해 복제를 지원하는 스토리지 백엔드를 하나 이상 구성해야 합니다. 소스 및 대상 클러스터가 동일한 경우 대상 애플리케이션은 최상의 복원력을 위해 소스 애플리케이션과 다른 스토리지 백엔드를 사용해야 합니다.

SnapMirror 복제를 위한 **Kubernetes** 클러스터 요구 사항

Kubernetes 클러스터가 다음 요구 사항을 충족하는지 확인하세요.

- **AppVault** 접근성: 소스 클러스터와 대상 클러스터 모두 애플리케이션 개체 복제를 위해 AppVault에서 읽고 쓸 수 있는 네트워크 액세스 권한이 있어야 합니다.
- 네트워크 연결: WAN을 통해 클러스터와 AppVault 간의 통신을 활성화하기 위해 방화벽 규칙, 버킷 권한 및 IP 허용 목록을 구성합니다.

많은 기업 환경에서는 WAN 연결 전반에 걸쳐 엄격한 방화벽 정책을 구현합니다. 복제를 구성하기 전에 인프라 팀과 함께 이러한 네트워크 요구 사항을 확인하세요.

Trident Protect를 설치하고 구성합니다

사용 중인 환경이 Trident Protect 요구 사항을 충족하는 경우 다음 단계에 따라 클러스터에 Trident Protect를 설치할 수 있습니다. NetApp에서 Trident Protect를 얻거나 개인 레지스트리에서 설치할 수 있습니다. 클러스터가 인터넷에 액세스할 수 없는 경우 개인 레지스트리에서 설치하는 것이 유용합니다.

Trident Protect를 설치합니다

NetApp에서 **Trident Protect**를 설치합니다

단계

1. Trident Helm 저장소 추가:

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

2. Helm을 사용하여 Trident Protect를 설치합니다. 클러스터에 할당되고 클러스터의 백업 및 스냅샷을 식별하는 데 사용되는 클러스터 이름으로 바꿉니다 `<name-of-cluster>`.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

3. 선택적으로, 디버그 로깅을 활성화하려면(문제 해결에 권장됨) 다음을 사용합니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

디버그 로깅은 NetApp 이 로그 수준을 변경하거나 문제를 재현하지 않고도 문제를 해결하는 데 도움이 됩니다.

개인 레지스트리로부터 **Trident Protect**를 설치합니다

Kubernetes 클러스터가 인터넷에 액세스할 수 없는 경우 개인 이미지 레지스트리에서 Trident Protect를 설치할 수 있습니다. 이 예제에서는 대괄호 안의 값을 사용자 환경의 정보로 바꿉니다.

단계

1. 다음 이미지를 로컬 컴퓨터로 가져와서 태그를 업데이트한 다음 개인 레지스트리에 푸시합니다.

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

예를 들면 다음과 같습니다.

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

> Helm 차트를 얻으려면 먼저 인터넷 접속이 가능한 컴퓨터에서 Helm 차트를 다운로드하세요. `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`그런 다음 결과를 복사합니다. `trident-protect-100.2510.0.tgz` 오프라인 환경에 파일을 업로드하고 다음을 사용하여 설치하세요. `helm install trident-protect ./trident-protect-100.2510.0.tgz` 마지막 단계에서 저장소 참조 대신.

2. Trident Protect 시스템 네임스페이스를 생성합니다.

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

3. 레지스트리에 로그인합니다.

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

4. 개인 레지스트리 인증에 사용할 풀 암호를 만듭니다.

```
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 저장소 추가:

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

6. 라는 이름의 파일을 `protectValues.yaml` 만듭니다. Trident Protect에 다음 설정이 포함되어 있는지 확인합니다.

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

그만큼 `imageRegistry` 그리고 `imagePullSecrets` 값은 다음을 포함한 모든 구성 요소 이미지에 적용됩니다. `resourcebackup` 그리고 `resourcerestore` . 레지스트리 내의 특정 저장소 경로에 이미지를 푸시하는 경우(예: `example.com:443/my-repo`), 레지스트리 필드에 전체 경로를 포함합니다. 이렇게 하면 모든 이미지가 다음에서 가져오게 됩니다. `<private-registry-url>/<image-name>:<tag>`.

7. Helm을 사용하여 Trident Protect를 설치합니다. 클러스터에 할당되고 클러스터의 백업 및 스냅샷을 식별하는 데 사용되는 클러스터 이름으로 바꿉니다 `<name_of_cluster>`.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. 선택적으로, 디버그 로깅을 활성화하려면(문제 해결에 권장됨) 다음을 사용합니다.

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

디버그 로깅은 NetApp 이 로그 수준을 변경하거나 문제를 재현하지 않고도 문제를 해결하는 데 도움이 됩니다.

AutoSupport 설정 및 네임스페이스 필터링을 포함한 추가 Helm 차트 구성 옵션은 다음을 참조하세요. "Trident Protect 설치를 사용자 지정합니다".

Trident Protect CLI 플러그인을 설치합니다

Trident 유틸리티의 확장인 Trident Protect 명령줄 플러그인을 사용하여 Trident CRS(사용자 지정 리소스)를 만들고 상호 작용할 수 있습니다 `tridentctl`.

Trident Protect CLI 플러그인을 설치합니다

명령줄 유틸리티를 사용하기 전에 클러스터에 액세스하는 데 사용하는 시스템에 설치해야 합니다. 컴퓨터에서 x64 또는 ARM CPU를 사용하는지 여부에 따라 다음 단계를 수행합니다.

Linux AMD64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인 다운로드:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

Linux ARM64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인 다운로드:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

Mac AMD64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인 다운로드:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

Mac ARM64 CPU용 플러그인을 다운로드합니다

단계

1. Trident Protect CLI 플러그인 다운로드:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. 플러그인 바이너리에 대한 실행 권한 활성화:

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

2. PATH 변수에 정의된 위치에 플러그인 바이너리를 복사합니다. 예를 `/usr/bin` 들어, 또는 `/usr/local/bin` (Elevated Privileges가 필요할 수 있음):

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

3. 선택적으로 플러그인 바이너리를 홈 디렉토리의 위치로 복사할 수 있습니다. 이 경우 위치가 PATH 변수의 일부인지 확인하는 것이 좋습니다.

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

PATH 변수의 위치에 플러그인을 복사하면 또는 `tridentctl protect` 임의의 위치에서 플러그인을 사용할 수 `tridentctl-protect` 있습니다.

Trident CLI 플러그인 도움말을 봅니다

내장 플러그인 도움말 기능을 사용하여 플러그인 기능에 대한 자세한 도움말을 얻을 수 있습니다.

단계

1. 도움말 기능을 사용하여 사용 지침을 봅니다.

```
tridentctl-protect help
```

명령 자동 완성 활성화

Trident Protect CLI 플러그인을 설치한 후 특정 명령에 대해 자동 완성 기능을 활성화할 수 있습니다.

Bash 셸에 대해 자동 완성 기능을 활성화합니다

단계

1. 완료 스크립트를 만듭니다.

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

2. 스크립트를 포함할 홈 디렉토리에 새 디렉토리를 만듭니다.

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

3. 다운로드한 스크립트를 디렉터리로 이동합니다 `~/.bash/completions`.

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

4. 홈 디렉토리의 파일에 다음 줄을 추가합니다 `~/.bashrc`.

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

Z 쉘에 대한 자동 완성 기능을 활성화합니다

단계

1. 완료 스크립트를 만듭니다.

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

2. 스크립트를 포함할 홈 디렉토리에 새 디렉토리를 만듭니다.

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

3. 다운로드한 스크립트를 디렉터리로 이동합니다 `~/.zsh/completions`.

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

4. 홈 디렉토리의 파일에 다음 줄을 추가합니다 `~/.zprofile`.

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

결과

다음 쉘 로그인 시 tridentctl-protect 플러그인으로 명령 자동 완성 기능을 사용할 수 있습니다.

Trident Protect 설치를 사용자 지정합니다

Trident Protect의 기본 구성은 사용자 환경의 특정 요구 사항에 맞게 사용자 지정할 수 있습니다.

Trident Protect 컨테이너 리소스 제한을 지정합니다

Trident Protect를 설치한 후 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 리소스 제한을 지정할 수 있습니다. 리소스 제한을 설정하면 Trident Protect 작업에서 사용되는 클러스터 리소스의 양을 제어할 수 있습니다.

단계

1. 라는 이름의 파일을 `resourceLimits.yaml` 만듭니다.
2. 환경 요구 사항에 따라 Trident Protect 컨테이너에 대한 리소스 제한 옵션을 파일에 채웁니다.

   다음 예제 구성 파일은 사용 가능한 설정을 보여 주며 각 리소스 제한에 대한 기본값을 포함합니다.

```
---
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. 파일의 값을 적용합니다 `resourceLimits.yaml`.

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

보안 컨텍스트 제약 조건을 사용자 지정합니다

Trident Protect를 설치한 후 구성 파일을 사용하여 Trident Protect 컨테이너에 대한 OpenShift SCC(Security Context Constraint)를 수정할 수 있습니다. 이러한 제약은 Red Hat OpenShift 클러스터의 Pod에 대한 보안 제한을 정의합니다.

단계

1. 라는 이름의 파일을 `sccconfig.yaml` 만듭니다.
2. 파일에 SCC 옵션을 추가하고 사용자 환경의 필요에 따라 매개 변수를 수정합니다.

   다음 예는 SCC 옵션에 대한 매개 변수의 기본값을 보여줍니다.

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

다음 표에서는 SCC 옵션에 대한 매개 변수를 설명합니다.

매개 변수	설명	기본값
생성	SCC 자원을 생성할 수 있는지 여부를 결정한다. SCC 리소스는 가로 설정되어 `true` 있고 Helm 설치 프로세스에서 OpenShift 환경을 식별하는 경우에만 `scc.create` 생성됩니다. OpenShift에서 작동하지 않거나 이 로 설정된 `false` 경우 `scc.create` SCC 리소스가 생성되지 않습니다.	참
이름	SCC의 이름을 지정합니다.	Trident-protect-job을 선택합니다
우선 순위	SCC의 우선 순위를 정의합니다. 우선 순위가 높은 SCC는 값이 낮은 SCC보다 먼저 평가됩니다.	1

3. 파일의 값을 적용합니다 `sccconfig.yaml`.

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

그러면 기본값이 파일에 지정된 값으로 `sccconfig.yaml` 바뀝니다.

추가 **Trident** 보호 헬름 차트 설정 구성

사용자의 특정 요구 사항에 맞게 AutoSupport 설정과 네임스페이스 필터링을 사용자 정의할 수 있습니다. 다음 표에서는 사용 가능한 구성 매개변수를 설명합니다.

매개 변수	유형	설명
autoSupport.proxy	문자열	NetApp AutoSupport 연결을 위한 프록시 URL을 구성합니다. 이를 사용하면 프록시 서버를 통해 지원 번들 업로드를 라우팅할 수 있습니다. 예: `http://my.proxy.url` .
autoSupport.안전하지 않음	부울	AutoSupport 프록시 연결에 대한 TLS 검증을 건너뜁니다. `true` . 안전하지 않은 프록시 연결에만 사용하세요. (기본: `false`)
autoSupport.활성화됨	부울	일일 Trident Protect AutoSupport 번들 업로드를 활성화하거나 비활성화합니다. 설정 시 `false` , 예약된 일일 업로드는 비활성화되지만, 여전히 수동으로 지원 번들을 생성할 수 있습니다. (기본: `true`)

매개 변수	유형	설명
restoreSkipNamespaceAnnotations	문자열	백업 및 복원 작업에서 제외할 네임스페이스 주석의 쉼표로 구분된 목록입니다. 주석을 기준으로 네임스페이스를 필터링할 수 있습니다.
restoreSkipNamespaceLabels	문자열	백업 및 복원 작업에서 제외할 네임스페이스 레이블의 쉼표로 구분된 목록입니다. 라벨을 기준으로 네임스페이스를 필터링할 수 있습니다.

YAML 구성 파일이나 명령줄 플래그를 사용하여 이러한 옵션을 구성할 수 있습니다.

YAML 파일 사용

단계

1. 구성 파일을 만들고 이름을 지정하세요. `values.yaml` .
2. 생성한 파일에 사용자 정의하려는 구성 옵션을 추가합니다.

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

3. 다음을 채운 후 `values.yaml` 올바른 값을 가진 파일을 만들려면 구성 파일을 적용하세요:

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

CLI 플래그 사용

단계

1. 다음 명령을 사용하십시오. `--set` 개별 매개변수를 지정하는 플래그:

```
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
```

Trident Protect Pod를 특정 노드로 제한합니다

Kubernetes nodeSelector 노드 선택 제약 조건을 사용하여 노드 레이블을 기준으로 Trident Protect Pod를 실행할 자격이 있는 노드를 제어할 수 있습니다. 기본적으로 Trident Protect는 Linux를 실행하는 노드로 제한됩니다. 필요에 따라 이러한 제약 조건을 추가로 사용자 지정할 수 있습니다.

단계

1. 라는 이름의 파일을 `nodeSelectorConfig.yaml` 만듭니다.
2. 파일에 nodeSelector 옵션을 추가하고 해당 파일을 수정하여 노드 레이블을 추가하거나 변경하여 환경 요구 사항에 따라 제한합니다. 예를 들어, 다음 파일에는 기본 OS 제한이 포함되어 있지만 특정 지역 및 앱 이름도 대상으로 합니다.

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

3. 파일의 값을 적용합니다 `nodeSelectorConfig.yaml`.

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

이렇게 하면 기본 제한 사항이 파일에 지정한 제한 사항으로 `nodeSelectorConfig.yaml` 바뀝니다.