Trident Protect 관리

Trident

NetApp
July 01, 2026

목차

Trident Protect 관리

Trident Protect 권한 부여 및 액세스 제어 관리

Trident Protect는 Kubernetes 모델의 역할 기반 액세스 제어(RBAC)를 사용합니다. 기본적으로 Trident Protect는 단일 시스템 네임스페이스와 해당 기본 서비스 계정을 제공합니다. 사용자 수가 많거나 특정 보안 요구 사항이 있는 조직의 경우 Trident Protect의 RBAC 기능을 활용하여 리소스 및 네임스페이스에 대한 액세스 제어를 더욱 세밀하게 제어할 수 있습니다.

클러스터 관리자는 항상 기본 `trident-protect` 네임스페이스의 리소스에 액세스할 수 있으며 다른 모든 네임스페이스의 리소스에도 액세스할 수 있습니다. 리소스 및 애플리케이션에 대한 액세스 제어를 위해 추가 네임스페이스를 생성하고 해당 네임스페이스에 리소스와 애플리케이션을 추가해야 합니다.

참고로, 기본 `trident-protect` 네임스페이스에서는 어떤 사용자도 애플리케이션 데이터 관리 CR을 생성할 수 없습니다. 애플리케이션 데이터 관리 CR은 애플리케이션 네임스페이스에서 생성해야 합니다(권장되는 방법은 해당 애플리케이션과 동일한 네임스페이스에 애플리케이션 데이터 관리 CR을 생성하는 것입니다).

> 관리자만 권한 있는 Trident Protect 사용자 지정 리소스 개체에 액세스할 수 있어야 합니다. 이러한 개체에는 다음이 포함됩니다.
>
> - **AppVault**: 버킷 자격 증명 데이터가 필요합니다
> - **AutoSupportBundle**: Trident Protect의 메트릭, 로그 및 기타 민감한 데이터를 수집합니다
> - **AutoSupportBundleSchedule**: 로그 수집 일정을 관리합니다
>
> 모범 사례로, RBAC를 사용하여 권한이 있는 개체에 대한 액세스를 클러스터 관리자로 제한하십시오.

RBAC가 리소스 및 네임스페이스에 대한 액세스를 규제하는 방법에 대한 자세한 내용은 "Kubernetes RBAC 문서"을 참조하십시오.

서비스 계정에 대한 자세한 내용은 "Kubernetes 서비스 계정 문서"을 참조하십시오.

예: 두 사용자 그룹에 대한 액세스 관리

예를 들어, 한 조직에 클러스터 관리자, 엔지니어링 사용자 그룹, 마케팅 사용자 그룹이 있다고 가정해 보겠습니다. 클러스터 관리자는 엔지니어링 그룹과 마케팅 그룹이 각각 자신의 네임스페이스에 할당된 리소스에만 액세스할 수 있는 환경을 구축하기 위해 다음과 같은 작업을 수행합니다.

1단계**:** 각 그룹의 리소스를 담을 네임스페이스를 생성합니다

네임스페이스를 생성하면 리소스를 논리적으로 분리하고 해당 리소스에 대한 액세스 권한을 더 효과적으로 제어할 수 있습니다.

단계

1. 엔지니어링 그룹을 위한 네임스페이스를 생성합니다.

```
kubectl create ns engineering-ns
```

2. 마케팅 그룹을 위한 네임스페이스를 생성하세요:

```
kubectl create ns marketing-ns
```

2단계**:** 각 네임스페이스의 리소스와 상호 작용할 새 서비스 계정 생성

새 네임스페이스를 생성할 때마다 기본 서비스 계정이 제공되지만, 향후 필요에 따라 그룹 간에 권한을 더욱 세분화할 수 있도록 각 사용자 그룹별로 서비스 계정을 생성해야 합니다.

단계

1. 엔지니어링 그룹을 위한 서비스 계정을 생성하세요:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. 마케팅 그룹에 대한 서비스 계정 생성:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

3단계**:** 각 새 서비스 계정에 대한 시크릿 생성

서비스 계정 암호는 서비스 계정 인증에 사용되며, 유출된 경우 쉽게 삭제하고 다시 생성할 수 있습니다.

단계

1. 엔지니어링 서비스 계정에 대한 암호를 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. 마케팅 서비스 계정에 대한 암호를 생성합니다.

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

4단계: **RoleBinding** 객체를 생성하여 **ClusterRole** 객체를 각 새 서비스 계정에 바인딩합니다

Trident Protect를 설치하면 기본 ClusterRole 오브젝트가 생성됩니다. 이 ClusterRole을 서비스 계정에 바인딩하려면 RoleBinding 오브젝트를 생성하고 적용할 수 있습니다.

단계

1. ClusterRole을 엔지니어링 서비스 계정에 바인딩합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. ClusterRole을 마케팅 서비스 계정에 바인딩합니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

5단계**:** 권한 테스트

권한이 올바른지 테스트합니다.

단계

1. 엔지니어링 사용자가 엔지니어링 리소스에 액세스할 수 있는지 확인하십시오.

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. 엔지니어링 사용자가 마케팅 리소스에 액세스할 수 없는지 확인합니다.

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

6단계**: AppVault** 개체에 대한 액세스 권한 부여

백업 및 스냅샷과 같은 데이터 관리 작업을 수행하려면 클러스터 관리자가 개별 사용자에게 AppVault 객체에 대한 액세스 권한을 부여해야 합니다.

단계

1. AppVault와 시크릿 조합 YAML 파일을 생성하여 사용자에게 AppVault에 대한 액세스 권한을 부여합니다. 예를 들어, 다음 CR은 사용자에게 AppVault에 대한 액세스 권한을 부여합니다 `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. 클러스터 관리자가 네임스페이스 내 특정 리소스에 대한 액세스 권한을 부여할 수 있도록 역할 CR을 생성하고 적용합니다. 예를 들면 다음과 같습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. RoleBinding CR을 생성하고 적용하여 eng-user 사용자에게 권한을 바인딩합니다. 예를 들면 다음과 같습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. 권한이 올바른지 확인합니다.

   a. 모든 네임스페이스에 대한 AppVault 객체 정보를 검색하려고 시도합니다.

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   다음과 유사한 출력이 표시됩니다.

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. 사용자가 이제 액세스 권한이 있는 AppVault 정보를 가져올 수 있는지 테스트합니다.

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

다음과 유사한 출력이 표시됩니다.

```
yes
```

결과

AppVault 권한을 부여받은 사용자는 승인된 AppVault 개체를 사용하여 애플리케이션 데이터 관리 작업을 수행할 수 있어야 하며, 할당된 네임스페이스 외부의 리소스에 액세스하거나 액세스 권한이 없는 새 리소스를 생성할 수 없어야 합니다.

Trident Protect 리소스를 모니터링하세요

kube-state-metrics, Prometheus 및 Alertmanager 오픈 소스 도구를 사용하여 Trident Protect로 보호되는 리소스의 상태를 모니터링할 수 있습니다.

kube-state-metrics 서비스는 Kubernetes API 통신에서 메트릭을 생성합니다. 이 서비스를 Trident Protect와 함께 사용하면 환경 내 리소스 상태에 대한 유용한 정보를 확인할 수 있습니다.

Prometheus는 kube-state-metrics에서 생성된 데이터를 수집하여 해당 객체에 대한 읽기 쉬운 정보로 제공하는 툴킷입니다. kube-state-metrics와 Prometheus를 함께 사용하면 Trident Protect로 관리하는 리소스의 상태를 모니터링할 수 있습니다.

Alertmanager는 Prometheus와 같은 도구에서 전송된 알림을 수집하여 구성한 대상으로 라우팅하는 서비스입니다.

> 이 단계에 포함된 구성 및 지침은 예시일 뿐이며, 사용자의 환경에 맞게 수정해야 합니다. 구체적인 지침 및 지원은 다음 공식 문서를 참조하십시오:
>
> - "kube-state-metrics 문서"
> - "Prometheus 문서"
> - "Alertmanager 문서"

1단계: 모니터링 도구 설치

Trident Protect에서 리소스 모니터링을 활성화하려면 kube-state-metrics, Prometheus 및 Alertmanager를 설치하고 구성해야 합니다.

kube-state-metrics 설치

Helm을 사용하여 kube-state-metrics를 설치할 수 있습니다.

단계

1. kube-state-metrics Helm 차트를 추가합니다. 예를 들면 다음과 같습니다.

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. 클러스터에 Prometheus ServiceMonitor CRD를 적용합니다.

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Helm 차트용 구성 파일을 생성합니다(예: `metrics-config.yaml`). 다음 예시 구성을 환경에 맞게 사용자 지정할 수 있습니다.

metrics-config.yaml: kube-state-metrics Helm 차트 구성

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Helm 차트를 배포하여 kube-state-metrics를 설치하세요. 예를 들면 다음과 같습니다.

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. "kube-state-metrics 사용자 정의 리소스 문서"의 지침에 따라 Trident Protect에서 사용하는 사용자 지정 리소스에 대한 메트릭을 생성하도록 kube-state-metrics를 구성합니다.

Prometheus 설치

https://prometheus.io/docs/prometheus/latest/installation/["Prometheus 문서"^]의 지침에 따라 Prometheus를 설치할 수 있습니다.

Alertmanager 설치

https://github.com/prometheus/alertmanager?tab=readme-ov-file#install["Alertmanager 문서"^]의 지침에 따라 Alertmanager를 설치할 수 있습니다.

2단계: 모니터링 도구가 함께 작동하도록 구성합니다

모니터링 도구를 설치한 후에는 함께 작동하도록 구성해야 합니다.

단계

1. kube-state-metrics를 Prometheus와 통합합니다. Prometheus 구성 파일(`prometheus.yaml`)을 편집하고 kube-state-metrics 서비스 정보를 추가합니다. 예를 들면 다음과 같습니다.

prometheus.yaml: Prometheus와 **kube-state-metrics** 서비스 통합

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Prometheus가 Alertmanager로 알림을 라우팅하도록 구성합니다. Prometheus 구성 파일(`prometheus.yaml`)을

편집하고 다음 섹션을 추가합니다.

prometheus.yaml: Alertmanager로 알림 전송

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

결과

이제 Prometheus는 kube-state-metrics에서 메트릭을 수집하고 Alertmanager로 알림을 보낼 수 있습니다. 이제 알림을 트리거하는 조건과 알림을 보낼 위치를 구성할 준비가 되었습니다.

3단계: 알림 및 알림 대상 구성

도구들을 연동하도록 구성한 후에는 어떤 유형의 정보가 알림을 발생시키는지, 그리고 알림을 어디로 보내야 하는지를 구성해야 합니다.

알림 예: 백업 실패

다음 예제는 백업 사용자 지정 리소스의 상태가 `Error`5초 이상 설정될 때 트리거되는 중요 경고를 정의합니다. 이 예제를 환경에 맞게 사용자 지정하고 이 YAML 코드 조각을 `prometheus.yaml` 구성 파일에 포함할 수 있습니다.

rules.yaml: 백업 실패에 대한 **Prometheus** 알림 정의

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Alertmanager가 다른 채널로 알림을 보내도록 구성합니다

Alertmanager를 구성하여 이메일, PagerDuty, Microsoft Teams 또는 기타 알림 서비스와 같은 다른 채널로 알림을 보낼 수 있습니다. 이를 위해서는 `alertmanager.yaml` 파일에 해당 구성을 지정하면 됩니다.

다음 예제는 Alertmanager가 Slack 채널로 알림을 보내도록 구성합니다. 이 예제를 사용자 환경에 맞게 사용자 지정하려면 `api_url` 키의 값을 사용자 환경에서 사용하는 Slack 웹훅 URL로 바꾸십시오.

alertmanager.yaml: Slack 채널로 알림 전송

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Trident Protect 지원 번들을 생성합니다

Trident Protect를 사용하면 관리자가 관리 중인 클러스터 및 애플리케이션에 대한 로그, 메트릭 및 토폴로지 정보를 포함하여 NetApp 지원에 유용한 정보가 담긴 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 사용자 지정 리소스(CR) 파일을 사용하여 지원 번들을 NetApp 지원 사이트(NSS)에 업로드할 수 있습니다.

CR을 사용하여 지원 번들을 생성합니다

단계

1. 사용자 정의 리소스(CR) 파일을 생성하고 이름을 지정합니다(예: `trident-protect-support-bundle.yaml`).
2. 다음 속성을 구성하십시오.
   - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 환경에 맞는 고유하고 적절한 이름을 선택하세요.
   - **spec.triggerType**: (필수) 지원 번들이 즉시 생성될지, 아니면 예약될지를 결정합니다. 예약된 번들 생성은 UTC 기준 오전 12시에 발생합니다. 가능한 값:
     - 예약됨
     - 수동
   - **spec.uploadEnabled**: (선택 사항) 지원 번들을 생성한 후 NetApp Support Site에 업로드할지 여부를 제어합니다. 지정하지 않으면 기본값은 `false`입니다. 가능한 값:
     - true
     - false(기본값)
   - **spec.dataWindowStart**: (선택 사항) 지원 번들에 포함된 데이터 기간이 시작되는 날짜 및 시간을 지정하는 RFC 3339 형식의 날짜 문자열입니다. 지정하지 않으면 기본값은 24시간 전입니다. 지정할 수 있는 가장 빠른 기간 시작 날짜는 7일 전입니다.

     YAML 예:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. `trident-protect-support-bundle.yaml` 파일에 올바른 값을 입력한 후 CR을 적용하십시오.

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

CLI를 사용하여 지원 번들 생성

단계

1. 지원 번들을 생성하려면 괄호 안의 값을 사용자 환경 정보로 바꾸십시오. `trigger-type`은 번들이 즉시 생성되는지 아니면 예약된 시간에 생성되는지를 결정하며, `Manual` 또는 `Scheduled`일 수 있습니다.

기본 설정은 `Manual`입니다.

예를 들면 다음과 같습니다.

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

지원 번들을 모니터링하고 검색합니다

두 가지 방법 중 하나를 사용하여 지원 번들을 생성한 후에는 생성 진행 상황을 모니터링하고 로컬 시스템으로 가져올 수 있습니다.

단계

1. `status.generationState`이(가) `Completed` 상태에 도달할 때까지 기다립니다. 다음 명령을 사용하여 생성 진행 상황을 모니터링할 수 있습니다.

```
kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
```

2. 지원 번들을 로컬 시스템으로 가져옵니다. 완료된 AutoSupport 번들에서 복사 명령을 가져옵니다.

```
kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
```

출력에서 `kubectl cp` 명령을 찾아 실행하되, 대상 인수를 원하는 로컬 디렉터리로 바꾸십시오.

Trident Protect 업그레이드

Trident Protect를 최신 버전으로 업그레이드하면 새로운 기능이나 버그 수정 사항을 활용할 수 있습니다.

- 버전 24.10에서 업그레이드할 때 업그레이드 중에 실행되는 스냅샷이 실패할 수 있습니다. 이러한 실패는 수동 또는 예약된 향후 스냅샷 생성을 방해하지 않습니다. 업그레이드 중에 스냅샷이 실패하면 수동으로 새 스냅샷을 생성하여 애플리케이션을 보호할 수 있습니다.

  잠재적인 오류를 방지하려면 업그레이드 전에 모든 스냅샷 일정을 비활성화한 다음 업그레이드 후에 다시 활성화할 수 있습니다. 그러나 이렇게 하면 업그레이드 기간 동안 예약된 스냅샷이 누락될 수 있습니다.

- 프라이빗 레지스트리 설치의 경우, 타겟 버전에 필요한 Helm 차트 및 이미지가 프라이빗 레지스트리에 있는지 확인하고, 사용자 지정 Helm 값이 새 차트 버전과 호환되는지 확인하십시오. 자세한 내용은 "개인 레지스트리에서 Trident Protect를 설치하세요"을 참조하십시오.

1단계: 버전 선택

Trident Protect 버전은 날짜 기반 `YY.MM` 명명 규칙을 따르며, 여기서 "YY"는 연도의 마지막 두 자리 숫자이고 "MM"은 월을 나타냅니다. 도트 릴리스는 `YY.MM.X` 규칙을 따르며, 여기서 "X"는 패치 레벨을 나타냅니다. 업그레이드할 버전은 업그레이드하려는 현재 버전을 기준으로 선택합니다.

- 설치된 버전에서 4개 릴리스 이내의 대상 릴리스로 직접 업그레이드할 수 있습니다. 예를 들어 24.10(또는 24.10의 모든 마이너 버전)에서 25.10으로 직접 업그레이드할 수 있습니다.
- 4개 릴리스 기간 외의 버전에서 업그레이드하는 경우 여러 단계를 거쳐 업그레이드하십시오. "이전 버전"에서 업그레이드하는 경우 해당 버전의 업그레이드 지침을 사용하여 4개 릴리스 기간에 맞는 최신 릴리스로 업그레이드하십시오. 예를 들어 24.10 버전을 실행 중이고 26.02으로 업그레이드하려는 경우:
  a. 먼저 24.10에서 25.02으로 업그레이드하십시오.
  b. 그다음 25.02에서 26.02으로 업그레이드하세요.

2단계: Trident Protect 업그레이드

Trident Protect를 업그레이드하려면 다음 단계를 수행하십시오.

단계

1. Trident Helm 리포지토리를 업데이트합니다.

```
helm repo update
```

2. Trident Protect CRD를 업그레이드합니다.

> 25.06 이전 버전에서 업그레이드하는 경우 이 단계가 필요합니다. CRD가 이제 Trident Protect Helm 차트에 포함되어 있기 때문입니다.

   a. `trident-protect-crds`에서 `trident-protect`로 CRD 관리를 전환하려면 다음 명령을 실행합니다.

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

   b. `trident-protect-crds` 차트의 Helm 시크릿을 삭제하려면 다음 명령을 실행하십시오.

> Helm을 사용하여 `trident-protect-crds` 차트를 제거하지 마십시오. 이렇게 하면 CRD 및 관련 데이터가 모두 삭제될 수 있습니다.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Trident Protect 업그레이드:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2602.0 --namespace trident-protect
```

`--set logLevel=debug`을 업그레이드 명령에 추가하여 업그레이드 중 로깅 수준을 구성할 수 있습니다. 기본 로깅 수준은 `warn`입니다. 디버그 로깅은 NetApp 지원팀이 로그 수준 변경이나 문제 재현 없이 문제를 진단하는 데 도움이 되므로 문제 해결을 위해 권장됩니다.