Management von Trident Protect

Trident

NetApp
November 14, 2025

Inhalt

Management von Trident Protect

Managen von Trident Schützen Sie die Autorisierung und Zugriffssteuerung

Trident Protect nutzt das Kubernetes-Modell der rollenbasierten Zugriffssteuerung (Role Based Access Control, RBAC). Standardmäßig stellt Trident Protect einen einzelnen System-Namespace und sein dazugehöriges Standarddienstkonto bereit. Wenn Ihr Unternehmen über eine Vielzahl von Benutzern oder spezifische Sicherheitsanforderungen verfügt, können Sie die RBAC-Funktionen von Trident Protect verwenden, um eine granularere Kontrolle über den Zugriff auf Ressourcen und Namespaces zu erlangen.

Der Clusteradministrator hat immer Zugriff auf Ressourcen im Standard- `trident-protect`Namespace und kann auch auf Ressourcen in allen anderen Namespaces zugreifen. Um den Zugriff auf Ressourcen und Anwendungen zu kontrollieren, müssen Sie zusätzliche Namespaces erstellen und diesen Namespaces Ressourcen und Anwendungen hinzufügen.

Beachten Sie, dass keine Benutzer Anwendungsdatenmanagement-CRS im Standard-Namespace erstellen können `trident-protect`. Sie müssen Anwendungsdatenmanagement-CRS in einem Anwendungs-Namespace erstellen (als Best Practice erstellen Sie Anwendungsdatenmanagement-CRS im gleichen Namespace wie ihre zugehörige Anwendung).

> Nur Administratoren sollten Zugriff auf privilegierte Trident haben, die benutzerdefinierte Ressourcenobjekte schützen, darunter:
>
> - **AppVault**: Erfordert Bucket-Zugangsdaten
> - **AutoSupportBundle**: Sammelt Kennzahlen, Protokolle und andere sensible Trident schützen Daten
> - **AutoSupportBundleSchedule**: Verwaltet Zeitpläne für die Protokollsammlung
>
> Verwenden Sie als Best Practice RBAC, um den Zugriff auf privilegierte Objekte auf Administratoren zu beschränken.

Weitere Informationen darüber, wie RBAC den Zugriff auf Ressourcen und Namespaces regelt, finden Sie im "RBAC-Dokumentation für Kubernetes".

Informationen zu Servicekonten finden Sie im "Dokumentation des Kubernetes Service-Kontos".

Beispiel: Zugriff für zwei Benutzergruppen verwalten

Ein Unternehmen verfügt beispielsweise über einen Cluster-Administrator, eine Gruppe von Engineering-Benutzern und eine Gruppe von Marketing-Benutzern. Der Clusteradministrator führt die folgenden Aufgaben aus, um eine Umgebung zu erstellen, in der die Engineering-Gruppe und die Marketing-Gruppe jeweils nur auf die Ressourcen zugreifen können, die ihren jeweiligen Namespaces zugewiesen sind.

Schritt 1: Erstellen Sie einen Namespace, der Ressourcen für jede Gruppe enthält

Durch das Erstellen eines Namespace können Sie Ressourcen logisch trennen und besser kontrollieren, wer

Zugriff auf diese Ressourcen hat.

Schritte

1. Erstellen Sie einen Namespace für die Engineering-Gruppe:

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

2. Erstellen Sie einen Namespace für die Marketinggruppe:

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

Schritt 2: Erstellen Sie neue Dienstkonten, um mit Ressourcen in jedem Namespace zu interagieren

Jeder neue Namespace, den Sie erstellen, verfügt über ein Standard-Dienstkonto. Sie sollten jedoch für jede Benutzergruppe ein Dienstkonto erstellen, damit Sie Privileges bei Bedarf in Zukunft weiter zwischen Gruppen aufteilen können.

Schritte

1. Erstellen Sie ein Servicekonto für die Engineering-Gruppe:

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

2. Erstellen Sie ein Service-Konto für die Marketinggruppe:

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

Schritt 3: Erstellen Sie ein Geheimnis für jedes neue Service-Konto

Ein Dienstkontogeheimnis wird verwendet, um sich beim Dienstkonto zu authentifizieren. Er kann bei einer Kompromittierung einfach gelöscht und neu erstellt werden.

Schritte

1. Einen Schlüssel für das Engineering-Servicekonto erstellen:

```
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. Erstellen Sie ein Geheimnis für das Marketingservicekonto:

```
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
```

Schritt 4: Erstellen Sie ein RoleBinding-Objekt, um das ClusterRole-Objekt an jedes neue Servicekonto zu binden

Bei der Installation von Trident Protect wird ein Standardobjekt für ClusterRole erstellt. Sie können diese ClusterRole an das Dienstkonto binden, indem Sie ein RoleBinding-Objekt erstellen und anwenden.

Schritte

1. Binden Sie die ClusterRole an das Engineering-Servicekonto:

```
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. Binden Sie den ClusterRole an das Marketingservicekonto:

```
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
```

Schritt 5: Testberechtigungen

Überprüfen Sie, ob die Berechtigungen korrekt sind.

Schritte

1. Bestätigung, dass Engineering-Benutzer auf Engineering-Ressourcen zugreifen können:

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

2. Bestätigen Sie, dass Engineering-Benutzer nicht auf Marketing-Ressourcen zugreifen können:

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

Schritt 6: Zugriff auf AppVault-Objekte gewähren

Um Datenmanagementaufgaben wie Backups und Snapshots auszuführen, muss der Clusteradministrator einzelnen Benutzern Zugriff auf AppVault-Objekte gewähren.

Schritte

1. Erstellen und Anwenden einer AppVault- und geheimen YAML-Kombinationsdatei, die einem Benutzer Zugriff auf einen AppVault gewährt. Der folgende CR gewährt dem Benutzer beispielsweise Zugriff auf einen AppVault `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. Erstellen und Anwenden eines Rollen-CR, damit Clusteradministratoren Zugriff auf bestimmte Ressourcen in einem Namespace gewähren können. Beispiel:

```
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. Erstellen und wenden Sie einen RoleBinding CR an, um die Berechtigungen an den Benutzer eng-user zu binden. Beispiel:

```
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. Überprüfen Sie, ob die Berechtigungen korrekt sind.

   a. Es wird versucht, die AppVault-Objektinformationen für alle Namespaces abzurufen:

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

   Sie sollten eine Ausgabe wie die folgende sehen:

```
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. Testen Sie, ob der Benutzer die AppVault-Informationen erhalten kann, auf die er jetzt Zugriff hat:

```
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
```

Sie sollten eine Ausgabe wie die folgende sehen:

```
yes
```

Ergebnis

Die Benutzer, denen Sie AppVault-Berechtigungen erteilt haben, sollten autorisierte AppVault-Objekte für Anwendungsdatenverwaltungsvorgänge verwenden können und nicht in der Lage sein, auf Ressourcen außerhalb der zugewiesenen Namespaces zuzugreifen oder neue Ressourcen zu erstellen, auf die sie keinen Zugriff haben.

Überwachen Sie Trident Protect-Ressourcen

Sie können die Open-Source-Tools kube-State-metrics, Prometheus und Alertmanager verwenden, um den Zustand der durch Trident Protect geschützten Ressourcen zu überwachen.

Der kube-Service für Statusmetriken generiert Kennzahlen aus der Kubernetes-API-Kommunikation. In Kombination mit Trident Protect gibt die Software hilfreiche Informationen über den Zustand der Ressourcen in der Umgebung wieder.

Prometheus ist ein Toolkit, das die von kube-State-metrics generierten Daten aufnehmen und als leicht lesbare Informationen über diese Objekte darstellen kann. Gemeinsam bieten Ihnen kube-State-metrics und Prometheus die Möglichkeit, den Zustand und den Status der Ressourcen zu überwachen, die Sie mit Trident Protect managen.

Alertmanager ist ein Dienst, der die von Tools wie Prometheus gesendeten Warnmeldungen aufnimmt und an die von Ihnen konfigurierten Ziele weiterleitet.

Die in diesen Schritten enthaltenen Konfigurationen und Anleitungen sind nur Beispiele. Sie müssen sie an Ihre Umgebung anpassen. Spezifische Anweisungen und Unterstützung finden Sie in der folgenden offiziellen Dokumentation:

- "kube-State-Metrics-Dokumentation"
- "Prometheus Dokumentation"
- "Alertmanager-Dokumentation"

Schritt 1: Installieren Sie die Überwachungstools

Um die Ressourcenüberwachung in Trident Protect zu aktivieren, müssen Sie kube-State-metrics, Promethus und Alertmanager installieren und konfigurieren.

Installieren Sie kube-State-metrics

Sie können kube-State-Metriken mit Helm installieren.

Schritte

1. Fügen Sie das Helm-Diagramm „kube-State-metrics“ hinzu. Beispiel:

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

2. Wenden Sie den Prometheus ServiceMonitor CRD auf den Cluster an:

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

3. Erstellen Sie eine Konfigurationsdatei für das Helm-Diagramm (z. B. `metrics-config.yaml`). Sie können die folgende Beispielkonfiguration an Ihre Umgebung anpassen:

Metrics-config.yaml: kube-State-metrics Helm Chart Configuration

```
---
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. Installieren Sie kube-State-metrics, indem Sie das Helm-Diagramm bereitstellen. Beispiel:

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

5. Konfigurieren Sie kube-State-metrics, um Metriken für die benutzerdefinierten Ressourcen zu generieren, die von Trident Protect verwendet werden, indem Sie die Anweisungen im befolgen "kube State-metrics Custom Resource Documentation".

Installation Von Prometheus

Sie können Prometheus installieren, indem Sie die Anweisungen im "Prometheus Dokumentation" .

Installieren Sie Alertmanager

Sie können Alertmanager installieren, indem Sie die Anweisungen im "Alertmanager-Dokumentation".

Schritt 2: Konfigurieren Sie die Überwachungstools für die Zusammenarbeit

Nachdem Sie die Überwachungstools installiert haben, müssen Sie sie für die Zusammenarbeit konfigurieren.

Schritte

1. Integrieren Sie kube-State-Metrics mit Prometheus. Bearbeiten Sie die Prometheus(`prometheus.yaml` -Konfigurationsdatei) und fügen Sie die kube-State-metrics-Dienstinformationen hinzu. Beispiel:

   prometheus.yaml: Integration des Kube-State-Metrics-Dienstes mit Prometheus

```
---
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. Konfigurieren Sie Prometheus für die Weiterleitung von Warnmeldungen an Alertmanager. Bearbeiten Sie die Prometheus Konfigurationsdatei (`prometheus.yaml`) und fügen Sie folgenden Abschnitt hinzu:

prometheus.yaml: Senden Sie Warnungen an Alertmanager

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

Ergebnis

Prometheus kann jetzt Kennzahlen von den Zustandsmetriken von kube erfassen und Alarme an Alertmanager senden. Sie können jetzt konfigurieren, welche Bedingungen eine Warnung auslösen und wo die Warnungen gesendet werden sollen.

Schritt 3: Konfigurieren von Warnungen und Warnungszielen

Nachdem Sie die Tools für eine Zusammenarbeit konfiguriert haben, müssen Sie konfigurieren, welche Art von Informationen Warnmeldungen auslöst und an welchen Orten die Meldungen gesendet werden sollen.

Warnbeispiel: Backup-Fehler

Das folgende Beispiel definiert eine kritische Warnung, die ausgelöst wird, wenn der Status der benutzerdefinierten Backup-Ressource auf 5 Sekunden oder länger eingestellt `Error` ist. Sie können dieses Beispiel an Ihre Umgebung anpassen und dieses YAML-Snippet in Ihre Konfigurationsdatei aufnehmen `prometheus.yaml`:

rules.yaml: Definieren Sie einen Prometheus-Alarm für fehlgeschlagene Backups

```
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."
```

Konfigurieren Sie Alertmanager so, dass Warnungen an andere Kanäle gesendet werden

Sie können Alertmanager so konfigurieren, dass Benachrichtigungen an andere Kanäle wie E-Mail, PagerDuty, Microsoft Teams oder andere Benachrichtigungsdienste gesendet werden, indem Sie die entsprechende Konfiguration in der Datei angeben `alertmanager.yaml`.

Im folgenden Beispiel wird Alertmanager so konfiguriert, dass Benachrichtigungen an einen Slack-Kanal gesendet werden. Um dieses Beispiel an Ihre Umgebung anzupassen, ersetzen Sie den Wert des `api_url`

Schlüssels durch die Slack Webhook-URL, die in Ihrer Umgebung verwendet wird:

alertmanager.yaml: Senden Sie Warnungen an einen Slack-Kanal

```
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
```

Generieren Sie ein Trident Protect Supportpaket

Mit Trident Protect können Administratoren Pakete erstellen, die für den NetApp Support nützliche Informationen enthalten, darunter Protokolle, Metriken und Topologieinformationen zu den verwalteten Clustern und Apps. Wenn Sie mit dem Internet verbunden sind, können Sie Support-Pakete mithilfe einer benutzerdefinierten Ressourcendatei (CR) auf die NetApp Support Site (NSS) hochladen.

Erstellen Sie mithilfe eines CR-Systems ein Supportpaket

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (z. B. `trident-protect-support-bundle.yaml`).
2. Konfigurieren Sie die folgenden Attribute:
   - **metadata.name**: (*required*) der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **Spec.triggerType**: (*required*) legt fest, ob das Support-Bundle sofort generiert oder geplant wird. Die geplante Bundle-Generierung findet um 12:00 UHR UTC statt. Mögliche Werte:
     - Geplant
     - Manuell
   - **Spec.UploadEnabled**: (*Optional*) steuert, ob das Supportpaket nach der Generierung auf die NetApp-Support-Website hochgeladen werden soll. Wenn nicht angegeben, wird standardmäßig auf `false`. Mögliche Werte:
     - Richtig
     - False (Standard)
   - **Spec.dataWindowStart**: (*Optional*) Eine Datumstring im RFC 3339-Format, die das Datum und die Uhrzeit angibt, zu der das Fenster der im Support-Bundle enthaltenen Daten beginnen soll. Wenn nicht angegeben, ist die Standardeinstellung vor 24 Stunden. Das früheste Fensterdatum, das Sie angeben können, ist vor 7 Tagen.

     Beispiel YAML:

```
---
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. Nachdem Sie die `trident-protect-support-bundle.yaml` Datei mit den richtigen Werten, wenden Sie den CR an:

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

Erstellen Sie ein Support-Bundle mithilfe der CLI

Schritte

1. Erstellen Sie das Supportpaket, und ersetzen Sie Werte in Klammern durch Informationen aus Ihrer

Umgebung. Der `trigger-type` legt fest, ob das Bündel sofort erstellt wird oder ob die Erstellungszeit vom Zeitplan vorgegeben ist, und kann oder `Scheduled` sein `Manual`. Die Standardeinstellung ist `Manual`.

Beispiel:

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

Überwachen und Abrufen des Support-Pakets

Nachdem Sie mit einer der beiden Methoden ein Support-Paket erstellt haben, können Sie den Generierungsfortschritt überwachen und es auf Ihr lokales System abrufen.

Schritte

1. Warten Sie auf die `status.generationState` erreichen `Completed` Zustand. Sie können den Generierungsfortschritt mit dem folgenden Befehl überwachen:

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

2. Rufen Sie das Support-Paket auf Ihr lokales System ab. Holen Sie sich den Kopierbefehl aus dem vollständigen AutoSupport Paket:

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

   Finden Sie die `kubectl cp` Befehl aus der Ausgabe und führen Sie ihn aus, wobei Sie das Zielargument durch Ihr bevorzugtes lokales Verzeichnis ersetzen.

Upgrade von Trident Protect

Sie können ein Upgrade von Trident Protect auf die neueste Version durchführen, um von neuen Funktionen oder Fehlerkorrekturen zu profitieren.

- Beim Upgrade von Version 24.10 können Snapshots während des Upgrades fehlschlagen. Dieser Fehler verhindert jedoch nicht die Erstellung zukünftiger Snapshots, egal ob manuell oder geplant. Sollte ein Snapshot während des Upgrades fehlschlagen, können Sie manuell einen neuen Snapshot erstellen, um den Schutz Ihrer Anwendung sicherzustellen.

  Um mögliche Fehler zu vermeiden, können Sie alle Snapshot-Zeitpläne vor dem Upgrade deaktivieren und anschließend wieder aktivieren. Dies führt jedoch dazu, dass während des Upgrade-Zeitraums alle geplanten Snapshots fehlen.

- Bei Installationen in privaten Registrys stellen Sie sicher, dass das erforderliche Helm-Chart und die Images für die Zielversion in Ihrer privaten Registry verfügbar sind, und überprüfen Sie, ob Ihre benutzerdefinierten Helm-Werte mit der neuen Chart-Version kompatibel sind. Weitere Informationen finden Sie unter"Installieren Sie Trident Protect aus einer privaten Registrierung" .

Führen Sie zum Upgrade von Trident Protect die folgenden Schritte aus.

Schritte

1. Aktualisieren Sie das Trident Helm-Repository:

```
helm repo update
```

2. Aktualisieren Sie die Trident-Schutz-CRDs:

   Dieser Schritt ist erforderlich, wenn Sie von einer Version vor 25.06 aktualisieren, da die CRDs jetzt im Trident Protect Helm-Diagramm enthalten sind.

   a. Führen Sie diesen Befehl aus, um die Verwaltung von CRDs zu verschieben von `trident-protect-crds` Zu `trident-protect` :

```
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. Führen Sie diesen Befehl aus, um das Helm-Geheimnis für das `trident-protect-crds` Diagramm:

      Deinstallieren Sie nicht die `trident-protect-crds` Diagramm mit Helm, da dadurch Ihre CRDs und alle zugehörigen Daten entfernt werden könnten.

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

3. Upgrade-Trident-Schutz:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

Sie können den Protokollierungsgrad während des Upgrades konfigurieren, indem Sie Folgendes hinzufügen `--set logLevel=debug` zum Upgrade-Befehl. Die Standardprotokollierungsstufe ist `warn` Die Für die Fehlerbehebung wird die Verwendung von Debug-Protokollierung empfohlen, da sie dem NetApp Support hilft, Probleme zu diagnostizieren, ohne dass Änderungen am Protokollierungsgrad oder eine Reproduktion des Problems erforderlich sind.