Trident Protect verwalten

Trident

NetApp
January 15, 2026

Inhalt

Trident Protect verwalten

Trident Protect-Autorisierung und Zugriffskontrolle verwalten

Trident Protect nutzt das Kubernetes-Modell der rollenbasierten Zugriffskontrolle (RBAC). Standardmäßig stellt Trident Protect einen einzelnen System-Namespace und das zugehörige Standard-Dienstkonto bereit. Wenn Sie eine Organisation mit vielen Benutzern oder besonderen Sicherheitsanforderungen haben, können Sie die RBAC-Funktionen von Trident Protect nutzen, um eine detailliertere Kontrolle über den Zugriff auf Ressourcen und Namensräume zu erhalten.

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

Beachten Sie, dass im Standardmodus keine Benutzer Änderungsanforderungen für die Anwendungsdatenverwaltung erstellen können. `trident-protect` Namespace. Sie müssen Anwendungsdatenverwaltungs-CRs in einem Anwendungs-Namespace erstellen (als Best Practice sollten Anwendungsdatenverwaltungs-CRs im selben Namespace wie die zugehörige Anwendung erstellt werden).

Nur Administratoren sollten Zugriff auf privilegierte benutzerdefinierte Ressourcenobjekte von Trident Protect haben, darunter:

- **AppVault**: Erfordert Bucket-Anmeldeinformationen
- **AutoSupportBundle**: Erfasst Metriken, Protokolle und andere sensible Trident Protect-Daten
- **AutoSupportBundleSchedule**: Verwaltet die Zeitpläne für die Protokollerfassung

Als bewährte Methode empfiehlt es sich, RBAC zu verwenden, um den Zugriff auf privilegierte Objekte auf Administratoren zu beschränken.

Weitere Informationen darüber, wie RBAC den Zugriff auf Ressourcen und Namensräume regelt, finden Sie in der "Kubernetes RBAC-Dokumentation" .

Weitere Informationen zu Servicekonten finden Sie unter "Dokumentation zum Kubernetes-Dienstkonto" .

Beispiel: Zugriffsverwaltung für zwei Benutzergruppen

Eine Organisation verfügt beispielsweise über einen Cluster-Administrator, eine Gruppe von Entwicklern und eine Gruppe von Marketing-Anwendern. Der Cluster-Administrator würde die folgenden Aufgaben erledigen, um eine Umgebung zu schaffen, in der die Entwicklungsabteilung und die Marketingabteilung jeweils nur Zugriff auf die Ressourcen haben, die ihren jeweiligen Namensräumen zugewiesen sind.

Schritt 1: Erstellen Sie einen Namensraum, der die Ressourcen für jede Gruppe enthält.

Durch die Erstellung eines Namensraums können Sie Ressourcen logisch trennen und besser kontrollieren, wer Zugriff auf diese Ressourcen hat.

Schritte

1. Erstellen Sie einen Namensraum für die Entwicklungsgruppe:

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

2. Erstellen Sie einen Namensraum für die Marketinggruppe:

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

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

Jeder neu erstellte Namespace verfügt über ein Standarddienstkonto. Sie sollten jedoch für jede Benutzergruppe ein Dienstkonto erstellen, um die Berechtigungen bei Bedarf später weiter zwischen den Gruppen aufteilen zu können.

Schritte

1. Erstellen Sie ein Dienstkonto für die Entwicklungsgruppe:

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

2. Erstellen Sie ein Servicekonto für die Marketinggruppe:

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

Schritt 3: Erstellen Sie ein Geheimnis für jedes neue Dienstkonto.

Ein Dienstkontogeheimnis dient zur Authentifizierung beim Dienstkonto und kann im Falle einer Kompromittierung leicht gelöscht und neu erstellt werden.

Schritte

1. Erstellen Sie ein Geheimnis für das Engineering-Service-Konto:

```
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 Marketing-Service-Konto:

```
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 Dienstkonto zu binden.

Bei der Installation von Trident Protect wird ein Standard-ClusterRole-Objekt 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 die ClusterRole an das Marketing-Servicekonto:

```
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: Berechtigungen testen

Prüfen Sie, ob die Berechtigungen korrekt sind.

Schritte

1. Stellen Sie sicher, dass die technischen Benutzer auf die technischen 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 die technischen Benutzer keinen Zugriff auf Marketingressourcen haben:

```
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 Datenverwaltungsaufgaben wie Backups und Snapshots durchzuführen, muss der Clusteradministrator einzelnen Benutzern Zugriff auf AppVault-Objekte gewähren.

Schritte

1. Erstellen und Anwenden einer YAML-Datei mit einer Kombination aus AppVault und Geheimnis, die einem Benutzer Zugriff auf einen AppVault gewährt. Beispielsweise gewährt die folgende CR dem Benutzer 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 wenden Sie eine Rollen-CR an, um Cluster-Administratoren die Möglichkeit zu geben, Zugriff auf bestimmte Ressourcen in einem Namespace zu gewähren. 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 eine 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. Versuch, AppVault-Objektinformationen für alle Namespaces abzurufen:

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

   Die Ausgabe sollte in etwa wie folgt aussehen:

```
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 abrufen kann, für die er nun Zugriffsberechtigung 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
```

Die Ausgabe sollte in etwa wie folgt aussehen:

```
yes
```

Ergebnis

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

Überwachen Sie die Ressourcen von Trident Protect.

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 Dienst kube-state-metrics generiert Metriken aus der Kubernetes-API-Kommunikation. Die Verwendung mit Trident Protect liefert nützliche Informationen über den Zustand der Ressourcen in Ihrer Umgebung.

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

Alertmanager ist ein Dienst, der die von Tools wie Prometheus gesendeten Warnmeldungen aufnimmt und sie an von Ihnen konfigurierte 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-metrics mit Helm installieren.

Schritte

1. Füge das Helm-Chart 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-Chart (zum Beispiel, `metrics-config.yaml`). Sie können die folgende Beispielkonfiguration an Ihre Umgebung anpassen:

metrics-config.yaml: kube-state-metrics Helm-Chart-Konfiguration

```
---
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 durch Bereitstellung des Helm-Charts. Beispiel:

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

5. Konfigurieren Sie kube-state-metrics so, dass Metriken für die von Trident Protect verwendeten benutzerdefinierten Ressourcen generiert werden, indem Sie den Anweisungen in der "Dokumentation der benutzerdefinierten Ressource kube-state-metrics" Die

Prometheus installieren

Sie können Prometheus installieren, indem Sie den Anweisungen in der "Prometheus-Dokumentation" .

Installieren Sie Alertmanager

Sie können Alertmanager installieren, indem Sie den Anweisungen in der "Alertmanager-Dokumentation" .

Schritt 2: Konfigurieren Sie die Überwachungstools für die Zusammenarbeit

Nach der Installation der Überwachungstools müssen Sie diese so konfigurieren, dass sie zusammenarbeiten.

Schritte

1. kube-state-metrics mit Prometheus integrieren. Bearbeiten Sie die Prometheus-Konfigurationsdatei(`prometheus.yaml`) und fügen Sie die Informationen zum kube-state-metrics-Dienst 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 so, dass Warnmeldungen an Alertmanager weitergeleitet werden. Bearbeiten Sie die Prometheus-Konfigurationsdatei(`prometheus.yaml`) und fügen Sie den folgenden Abschnitt hinzu:

prometheus.yaml: Benachrichtigungen an Alertmanager senden

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

Ergebnis

Prometheus kann nun Metriken von kube-state-metrics erfassen und Warnungen an Alertmanager senden. Sie können nun konfigurieren, unter welchen Bedingungen eine Warnung ausgelöst wird und wohin die Warnungen gesendet werden sollen.

Schritt 3: Benachrichtigungen und Benachrichtigungsziele konfigurieren

Nachdem Sie die Tools so konfiguriert haben, dass sie zusammenarbeiten, müssen Sie festlegen, welche Art von Informationen Warnmeldungen auslösen und wohin die Warnmeldungen gesendet werden sollen.

Beispiel für eine Warnung: Backup-Fehler

Das folgende Beispiel definiert eine kritische Warnung, die ausgelöst wird, wenn der Status der benutzerdefinierten Sicherungsressource auf „“ gesetzt wird. `Error` für 5 Sekunden oder länger. Sie können dieses Beispiel an Ihre Umgebung anpassen und diesen YAML-Ausschnitt in Ihre Datei einfügen. `prometheus.yaml` Konfigurationsdatei:

rules.yaml: Definiere eine Prometheus-Warnung 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 Benachrichtigungen 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 `alertmanager.yaml` Datei.

Das folgende Beispiel konfiguriert Alertmanager so, dass Benachrichtigungen an einen Slack-Kanal gesendet werden. Um dieses Beispiel an Ihre Umgebung anzupassen, ersetzen Sie den Wert von `api_url` Schlüssel

mit der in Ihrer Umgebung verwendeten Slack-Webhook-URL:

alertmanager.yaml: Benachrichtigungen an einen Slack-Kanal senden

```
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

Trident Protect ermöglicht es Administratoren, Pakete zu generieren, die für den NetApp -Support nützliche Informationen enthalten, darunter Protokolle, Metriken und Topologieinformationen über die verwalteten Cluster und Anwendungen. 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 ein Support-Bundle mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR-Datei) und benennen Sie sie (zum Beispiel). `trident-protect-support-bundle.yaml`).
2. Konfigurieren Sie die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und aussagekräftigen Namen für Ihre Umgebung.
   - **spec.triggerType**: (*Erforderlich*) Legt fest, ob das Support-Bundle sofort oder geplant generiert wird. Die geplante Bündelgenerierung erfolgt um 0:00 Uhr UTC. Mögliche Werte:
     - Geplant
     - Handbuch
   - **spec.uploadEnabled**: (*Optional*) Steuert, ob das Support-Bundle nach seiner Generierung auf die NetApp Support-Website hochgeladen werden soll. Falls nicht anders angegeben, wird standardmäßig der Wert verwendet. `false` . Mögliche Werte:
     - true
     - falsch (Standardeinstellung)
   - **spec.dataWindowStart**: (*Optional*) Eine Datumszeichenfolge im RFC 3339-Format, die Datum und Uhrzeit angibt, zu der das Fenster der im Support-Bundle enthaltenen Daten beginnen soll. Falls nicht anders angegeben, wird standardmäßig der Wert von vor 24 Stunden verwendet. Das früheste Datum, das Sie angeben können, liegt 7 Tage zurück.

     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 korrekten Werten, CR anwenden:

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

Erstellen Sie ein Support-Bundle mithilfe der CLI.

Schritte

1. Erstellen Sie das Support-Bundle und ersetzen Sie die Werte in Klammern durch Informationen aus

Ihrer Umgebung. Der `trigger-type` bestimmt, ob das Paket sofort erstellt wird oder ob die Erstellungszeit durch den Zeitplan vorgegeben ist, und kann sein `Manual` oder `Scheduled` . 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` Führen Sie den Befehl aus der Ausgabe aus und ersetzen Sie dabei das Zielargument durch Ihr bevorzugtes lokales Verzeichnis.

Upgrade Trident Protect

Sie können Trident Protect auf die neueste Version aktualisieren, um von neuen Funktionen oder Fehlerbehebungen zu profitieren.

Beim Upgrade von Version 24.10 kann es vorkommen, dass während des Upgrades ausgeführte Snapshots fehlschlagen. Dieser Fehler verhindert nicht, dass künftige Snapshots, ob manuell oder geplant, erstellt werden. Falls ein Snapshot während des Upgrades fehlschlägt, können Sie manuell einen neuen Snapshot erstellen, um sicherzustellen, dass Ihre Anwendung geschützt ist.

Um mögliche Fehler zu vermeiden, können Sie vor dem Upgrade alle Snapshot-Zeitpläne deaktivieren und anschließend wieder aktivieren. Dies hat jedoch zur Folge, dass während des Upgrade-Zeitraums geplante Snapshots verpasst werden.

Um Trident Protect zu aktualisieren, führen Sie die folgenden Schritte aus.

Schritte

1. Aktualisieren Sie das Trident Helm-Repository:

```
helm repo update
```

2. Rüsten Sie die Trident Protect CRDs auf:

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

   a. Führen Sie diesen Befehl aus, um die Verwaltung von CRDs zu ändern 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-Secret für die `trident-protect-crds` Diagramm:

> Deinstallieren Sie die `trident-protect-crds` Verwenden Sie Helm nicht für Charts, da dies Ihre CRDs und alle zugehörigen Daten entfernen könnte.

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

3. Upgrade Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2506.0 --namespace trident-protect
```