Gestisci Trident Protect

Trident

NetApp
February 02, 2026

Sommario

Gestisci Trident Protect

Gestisci l'autorizzazione e il controllo degli accessi Trident Protect

Trident Protect utilizza il modello Kubernetes di controllo degli accessi basato sui ruoli (RBAC). Per impostazione predefinita, Trident Protect fornisce un singolo namespace di sistema e il relativo account di servizio predefinito. Se la tua organizzazione ha molti utenti o esigenze di sicurezza specifiche, puoi utilizzare le funzionalità RBAC di Trident Protect per ottenere un controllo più granulare sull'accesso alle risorse e agli spazi dei nomi.

L'amministratore del cluster ha sempre accesso alle risorse nello spazio dei nomi predefinito `trident-protect` e può anche accedere alle risorse in tutti gli altri namespace. Per controllare l'accesso a risorse e applicazioni, è necessario creare spazi dei nomi aggiuntivi e aggiungere risorse e applicazioni a tali spazi dei nomi.

Si noti che nessun utente può creare CRS per la gestione dei dati delle applicazioni nello spazio dei nomi predefinito `trident-protect`. È necessario creare CRS per la gestione dei dati delle applicazioni in uno spazio dei nomi delle applicazioni (come Best practice, creare CRS per la gestione dei dati delle applicazioni nello stesso spazio dei nomi dell'applicazione associata).

> Solo gli amministratori dovrebbero avere accesso agli oggetti di risorse personalizzate Trident Protect privilegiati, tra cui:
>
> - **AppVault**: Richiede i dati delle credenziali del bucket
> - **AutoSupportBundle**: raccoglie metriche, registri e altri dati sensibili Trident Protect
> - **AutoSupportBundleSchedule**: Gestisce i programmi di raccolta dei log
>
> Come Best practice, utilizzare RBAC per limitare l'accesso agli oggetti con privilegi agli amministratori.

Per ulteriori informazioni su come RBAC regola l'accesso alle risorse e agli spazi dei nomi, fare riferimento alla "Documentazione RBAC di Kubernetes" .

Per informazioni sugli account di servizio, fare riferimento alla "Documentazione dell'account del servizio Kubernetes" .

Esempio: Gestire l'accesso per due gruppi di utenti

Ad esempio, un'organizzazione dispone di un amministratore cluster, di un gruppo di utenti di progettazione e di un gruppo di utenti di marketing. L'amministratore del cluster dovrebbe completare le seguenti attività per creare un ambiente in cui il gruppo di progettazione e il gruppo di marketing hanno ciascuno accesso solo alle risorse assegnate ai rispettivi namespace.

Passaggio 1: Creare uno spazio dei nomi che contenga risorse per ciascun gruppo

La creazione di uno spazio dei nomi consente di separare logicamente le risorse e di controllare meglio chi ha accesso a tali risorse.

Fasi

1. Creare uno spazio dei nomi per il gruppo tecnico:

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

2. Creare uno spazio dei nomi per il gruppo di marketing:

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

Passaggio 2: Creare nuovi account di servizio per interagire con le risorse in ogni spazio dei nomi

Ogni nuovo spazio dei nomi creato viene fornito con un account di servizio predefinito, ma è necessario creare un account di servizio per ogni gruppo di utenti in modo da poter dividere ulteriormente Privileges tra i gruppi in futuro, se necessario.

Fasi

1. Creare un account di servizio per il gruppo tecnico:

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

2. Creare un account di servizio per il gruppo di marketing:

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

Passaggio 3: Creare un segreto per ogni nuovo account di servizio

Un segreto dell'account di servizio viene utilizzato per l'autenticazione con l'account di servizio e può essere facilmente eliminato e ricreato se compromesso.

Fasi

1. Creare un segreto per l'account del servizio tecnico:

```
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. Creare un segreto per l'account del servizio di marketing:

```
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
```

Passaggio 4: Creare un oggetto RoleBinding per associare l'oggetto ClusterRole a ogni nuovo account di servizio

Quando si installa Trident Protect, viene creato un oggetto ClusterRole predefinito. È possibile associare questo ClusterRole all'account di servizio creando e applicando un oggetto RoleBinding.

Fasi

1. Associare ClusterRole all'account del servizio tecnico:

```
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. Associare ClusterRole all'account del servizio di marketing:

```
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
```

Passaggio 5: Verifica delle autorizzazioni

Verificare che le autorizzazioni siano corrette.

Fasi

1. Verificare che gli utenti tecnici possano accedere alle risorse di progettazione:

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

2. Verificare che gli utenti tecnici non possano accedere alle risorse di marketing:

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

Passaggio 6: Concedere l'accesso agli oggetti AppVault

Per eseguire attività di gestione dei dati come backup e snapshot, l'amministratore del cluster deve garantire l'accesso agli oggetti AppVault ai singoli utenti.

Fasi

1. Creare e applicare un file YAML di combinazione di AppVault e segreto che consenta a un utente di accedere a un AppVault. Ad esempio, la seguente CR concede l'accesso ad AppVault all'utente `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. Creare e applicare un ruolo CR per consentire agli amministratori del cluster di concedere l'accesso a risorse specifiche in uno spazio dei nomi. Ad esempio:

```
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. Creare e applicare un RoleBinding CR per associare le autorizzazioni all'utente eng-user. Ad esempio:

```
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. Verificare che le autorizzazioni siano corrette.

   a. Tentativo di recuperare le informazioni sull'oggetto AppVault per tutti gli spazi dei nomi:

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

   L'output dovrebbe essere simile a quanto segue:

```
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. Verificare se l'utente può ottenere le informazioni AppVault a cui ora dispone dell'autorizzazione per accedere:

```
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
```

L'output dovrebbe essere simile a quanto segue:

```
yes
```

Risultato

Gli utenti a cui sono state concesse le autorizzazioni AppVault dovrebbero essere in grado di utilizzare gli oggetti AppVault autorizzati per le operazioni di gestione dei dati delle applicazioni e non dovrebbero essere in grado di accedere a risorse esterne agli spazi dei nomi assegnati o creare nuove risorse a cui non hanno accesso.

Monitorare le risorse Trident Protect

È possibile utilizzare gli strumenti open source kube-state-metrics, Prometheus e Alertmanager per monitorare lo stato di integrità delle risorse protette da Trident Protect.

Il servizio kube-state-metrics genera metriche dalla comunicazione API di Kubernetes. Utilizzandolo con Trident Protect, puoi ottenere informazioni utili sullo stato delle risorse nel tuo ambiente.

Prometheus è un toolkit in grado di acquisire i dati generati da kube-state-metrics e presentarli come informazioni facilmente leggibili su questi oggetti. Insieme, kube-state-metrics e Prometheus ti consentono di monitorare lo stato e l'integrità delle risorse che gestisci con Trident Protect.

Alertmanager è un servizio che acquisisce gli avvisi inviati da strumenti come Prometheus e li indirizza alle destinazioni configurate dall'utente.

Le configurazioni e le istruzioni incluse in questa procedura sono solo esempi; è necessario personalizzarle in base all'ambiente in uso. Per istruzioni specifiche e assistenza, consultare la seguente documentazione ufficiale:

- "documentazione kube-state-metrics"
- "Documentazione Prometheus"
- "Documentazione di Alertmanager"

Fase 1: Installare gli strumenti di monitoraggio

Per abilitare il monitoraggio delle risorse in Trident Protect, è necessario installare e configurare kube-state-metrics, Promethus e Alertmanager.

Installa metriche-stato-kube

È possibile installare parametri kube-state-metrics utilizzando Helm.

Fasi

1. Aggiungere il grafico Helm kube-state-metrics. Ad esempio:

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

2. Applicare il CRD di Prometheus ServiceMonitor al cluster:

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

3. Creare un file di configurazione per il grafico Helm (ad esempio, `metrics-config.yaml`). È possibile personalizzare la seguente configurazione di esempio in base all'ambiente in uso:

Metrics-config.yaml: Configurazione del grafico Helm kube-state-metrics

```
---
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. Installare le metriche di stato kube distribuendo il grafico Helm. Ad esempio:

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

5. Configurare kube-state-metrics per generare metriche per le risorse personalizzate utilizzate da Trident Protect seguendo le istruzioni in "documentazione sulle risorse personalizzate kube-state-metrics" .

Installare Prometheus

È possibile installare Prometheus seguendo le istruzioni riportate nella "Documentazione Prometheus" .

Installare Alertmanager

È possibile installare Alertmanager seguendo le istruzioni riportate nella "Documentazione di Alertmanager" .

Fase 2: Configurare gli strumenti di monitoraggio per lavorare insieme

Dopo aver installato gli strumenti di monitoraggio, è necessario configurarli per lavorare insieme.

Fasi

1. Integra metriche-stato-kube con Prometheus. Modificare il file di configurazione di Prometheus (`prometheus.yaml`) e aggiungere le informazioni del servizio kube-state-metrics. Ad esempio:

   prometheus.yaml: integrazione del servizio kube-state-metrics con 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. Configurare Prometheus per instradare gli avvisi ad Alertmanager. Modificare il file di configurazione di Prometheus (`prometheus.yaml`) e aggiungere la seguente sezione:

prometheus.yaml: Invia avvisi ad Alertmanager

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

Risultato

Prometheus può ora raccogliere le metriche dalle metriche dello stato del kube e inviare avvisi ad Alertmanager. Ora si è pronti a configurare quali condizioni attivano un avviso e dove inviare gli avvisi.

Passaggio 3: Configurare le destinazioni degli avvisi e degli avvisi

Dopo aver configurato gli strumenti per lavorare insieme, è necessario configurare il tipo di informazioni che attivano gli avvisi e la posizione in cui devono essere inviati.

Esempio di avviso: Errore di backup

Nell'esempio seguente viene definito un avviso critico che viene attivato quando lo stato della risorsa personalizzata di backup è impostato su `Error` per 5 secondi o più. È possibile personalizzare questo esempio in base all'ambiente in uso e includere questo frammento YAML nel `prometheus.yaml` file di configurazione:

rules.yaml: Definisci un avviso Prometheus per i backup non riusciti

```
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."
```

Configurare Alertmanager per inviare avvisi ad altri canali

È possibile configurare Alertmanager in modo che invii notifiche ad altri canali, quali e-mail, PagerDuty, Microsoft Teams o altri servizi di notifica specificando la rispettiva configurazione nel `alertmanager.yaml` file.

Nell'esempio seguente, Alertmanager configura l'invio di notifiche a un canale Slack. Per personalizzare questo esempio in base all'ambiente in uso, sostituire il valore della `api_url` chiave con l'URL slack webhook utilizzato nell'ambiente in uso:

alertmanager.yaml: invia avvisi a un canale Slack

```
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
```

Genera un pacchetto di supporto Trident Protect

Trident Protect consente agli amministratori di generare bundle che includono informazioni utili al supporto NetApp , tra cui registri, metriche e informazioni sulla topologia dei cluster e delle app in gestione. Se sei connesso a Internet, puoi caricare i bundle di supporto sul sito di supporto NetApp (NSS) utilizzando un file di risorse personalizzato (CR).

Creare un pacchetto di supporto utilizzando una CR

Fasi

1. Creare il file di risorsa personalizzata (CR) e assegnargli un nome (ad esempio, `trident-protect-support-bundle.yaml`).
2. Configurare i seguenti attributi:
   - **metadata.name**: (*required*) il nome di questa risorsa personalizzata; scegliere un nome univoco e sensibile per il proprio ambiente.
   - **Spec.triggerType**: (*required*) determina se il bundle di supporto viene generato immediatamente o pianificato. La generazione pianificata del pacchetto avviene alle 12am:00 UTC. Valori possibili:
     - Pianificato
     - Manuale
   - **Spec.uploadEnabled**: (*Optional*) Controlla se il bundle di supporto deve essere caricato nel sito di supporto NetApp dopo che è stato generato. Se non specificato, il valore predefinito è `false` . Valori possibili:
     - vero
     - false (impostazione predefinita)
   - **Spec.dataWindowStart**: (*Optional*) stringa di data in formato RFC 3339 che specifica la data e l'ora di inizio della finestra dei dati inclusi nel pacchetto di supporto. Se non specificato, il valore predefinito è 24 ore fa. La prima data della finestra che è possibile specificare è 7 giorni fa.

     Esempio 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. Dopo aver popolato il `trident-protect-support-bundle.yaml` file con i valori corretti, applicare CR:

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

Creare un bundle di supporto utilizzando la CLI

Fasi

1. Creare il pacchetto di supporto, sostituendo i valori tra parentesi con le informazioni dell'ambiente. `trigger-type`Determina se il bundle viene creato immediatamente o se l'ora

di creazione è dettata dalla pianificazione e può essere `Manual` o `Scheduled`. L'impostazione predefinita è `Manual`.

Ad esempio:

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

Monitorare e recuperare il pacchetto di supporto

Dopo aver creato un pacchetto di supporto utilizzando uno dei due metodi, puoi monitorarne l'avanzamento della generazione e recuperarlo nel tuo sistema locale.

Fasi

1. Aspetta il `status.generationState` raggiungere `Completed` stato. È possibile monitorare l'avanzamento della generazione con il seguente comando:

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

2. Recupera il pacchetto di supporto sul tuo sistema locale. Ottieni il comando di copia dal bundle AutoSupport completato:

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

   Trova il `kubectl cp` comando dall'output ed eseguilo, sostituendo l'argomento di destinazione con la directory locale preferita.

Aggiorna Trident Protect

Puoi aggiornare Trident Protect all'ultima versione per sfruttare le nuove funzionalità o le correzioni di bug.

- Quando si esegue l'aggiornamento dalla versione 24.10, gli snapshot in esecuzione durante l'aggiornamento potrebbero non funzionare. Questo problema non impedisce la creazione di snapshot futuri, manuali o pianificati. Se uno snapshot non funziona durante l'aggiornamento, è possibile crearne manualmente uno nuovo per garantire la protezione dell'applicazione.

  Per evitare potenziali errori, è possibile disabilitare tutte le pianificazioni degli snapshot prima dell'aggiornamento e riabilitarle in seguito. Tuttavia, ciò comporterà la perdita di tutti gli snapshot pianificati durante il periodo di aggiornamento.

- Per le installazioni di registri privati, assicurati che il grafico Helm e le immagini richiesti per la versione di destinazione siano disponibili nel tuo registro privato e verifica che i tuoi valori Helm personalizzati siano compatibili con la nuova versione del grafico. Per maggiori informazioni, fare riferimento a"Installa Trident Protect da un registro privato" .

Per aggiornare Trident Protect, procedere come segue.

Fasi

1. Aggiornare il repository di Trident Helm:

```
helm repo update
```

2. Aggiorna i CRD Trident Protect:

   Questo passaggio è necessario se si esegue l'aggiornamento da una versione precedente alla 25.06, poiché i CRD sono ora inclusi nella tabella Trident Protect Helm.

   a. Eseguire questo comando per spostare la gestione dei CRD da `trident-protect-crds` A `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. Esegui questo comando per eliminare il segreto Helm per `trident-protect-crds` grafico:

   Non disinstallare il `trident-protect-crds` grafico utilizzando Helm, poiché ciò potrebbe rimuovere i CRD e tutti i dati correlati.

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

3. Aggiorna Trident Protect:

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

È possibile configurare il livello di registrazione durante l'aggiornamento aggiungendo `--set logLevel=debug` al comando di aggiornamento. Il livello di registrazione predefinito è `warn` . La registrazione del debug è consigliata per la risoluzione dei problemi, poiché aiuta il supporto NetApp a diagnosticare i problemi senza richiedere modifiche al livello di registro o la riproduzione del problema.

Informazioni sul copyright

Informazioni sul marchio commerciale