Administra Trident Protect

Trident

NetApp
July 01, 2026

Tabla de contenidos

Administra Trident Protect

Administra la autorización y el control de acceso de Trident Protect

Trident Protect utiliza el modelo de control de acceso basado en roles (RBAC) de Kubernetes. Por defecto, Trident Protect proporciona un único espacio de nombres del sistema y su cuenta de servicio predeterminada asociada. Si tienes una organización con muchos usuarios o necesidades de seguridad específicas, puedes usar las funciones de RBAC de Trident Protect para tener un control más granular sobre el acceso a los recursos y espacios de nombres.

El administrador del clúster siempre tiene acceso a los recursos en el espacio de nombres predeterminado `trident-protect` y también puede acceder a los recursos en todos los demás espacios de nombres. Para controlar el acceso a los recursos y aplicaciones, necesitas crear espacios de nombres adicionales y agregar recursos y aplicaciones a esos espacios de nombres.

Ten en cuenta que ningún usuario puede crear CR de gestión de datos de aplicaciones en el espacio de nombres predeterminado `trident-protect`. Necesitas crear CR de gestión de datos de aplicaciones en un espacio de nombres de aplicaciones (como mejor práctica, crea los CR de gestión de datos de aplicaciones en el mismo espacio de nombres que su aplicación asociada).

> Solo los administradores deben tener acceso a los objetos de recursos personalizados privilegiados de Trident Protect, que incluyen:
>
> - **AppVault**: requiere datos de credenciales de cubo
> - **AutoSupportBundle**: recoge métricas, registros y otros datos confidenciales de Trident Protect
> - **AutoSupportBundleSchedule**: gestiona los calendarios de recogida de registros
>
> Como mejor práctica, usa RBAC para restringir el acceso a objetos privilegiados a los administradores.

Para obtener más información sobre cómo RBAC regula el acceso a los recursos y espacios de nombres, consulta la "Documentación de Kubernetes RBAC".

Para más información sobre las cuentas de servicio, consulta el "Documentación de cuentas de servicio de Kubernetes".

Ejemplo: gestionar el acceso de dos grupos de usuarios

Por ejemplo, una organización tiene un administrador de clúster, un grupo de usuarios de ingeniería y un grupo de usuarios de marketing. El administrador de clúster realizaría las siguientes tareas para crear un entorno donde el grupo de ingeniería y el grupo de marketing tengan acceso solo a los recursos asignados a sus respectivos espacios de nombres.

Paso 1: crea un espacio de nombres para contener los recursos de cada grupo

Crear un espacio de nombres te permite separar lógicamente los recursos y controlar mejor quién tiene acceso a esos recursos.

Pasos

1. Crea un espacio de nombres para el grupo de ingeniería:

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

2. Crea un espacio de nombres para el grupo de marketing:

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

Paso 2: crea nuevas cuentas de servicio para interactuar con recursos en cada espacio de nombres

Cada nuevo espacio de nombres que crees viene con una cuenta de servicio por defecto, pero deberías crear una cuenta de servicio para cada grupo de usuarios para que puedas dividir aún más los privilegios entre grupos en el futuro si es necesario.

Pasos

1. Crea una cuenta de servicio para el grupo de ingeniería:

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

2. Crea una cuenta de servicio para el grupo de marketing:

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

Paso 3: crea un secreto para cada nueva cuenta de servicio

Un secreto de cuenta de servicio se usa para autenticarse con la cuenta de servicio y se puede borrar y recrear fácilmente si se ve comprometido.

Pasos

1. Crea un secreto para la cuenta de servicio de ingeniería:

```
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. Crea un secreto para la cuenta del servicio de 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
```

Paso 4: crea un objeto RoleBinding para vincular el objeto ClusterRole a cada nueva cuenta de servicio

Se crea un objeto ClusterRole por defecto cuando instalas Trident Protect. Puedes vincular este ClusterRole a la cuenta de servicio creando y aplicando un objeto RoleBinding.

Pasos

1. Vincula el ClusterRole a la cuenta de servicio de ingeniería:

```
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. Vincula el ClusterRole a la cuenta de servicio de 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
```

Paso 5: probar los permisos

Comprueba que los permisos son correctos.

Pasos

1. Confirma que los usuarios de ingeniería pueden acceder a los recursos de ingeniería:

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

2. Confirma que los usuarios de ingeniería no pueden acceder a los recursos de marketing:

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

Paso 6: concede acceso a los objetos de AppVault

Para realizar tareas de gestión de datos como copias de seguridad e instantáneas, el administrador del clúster debe conceder acceso a AppVault objetos a usuarios individuales.

Pasos

1. Crea y aplica un archivo YAML de combinación de AppVault y secreto que le da a un usuario acceso a un AppVault. Por ejemplo, el siguiente CR da acceso a un AppVault al usuario `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. Crea y aplica un Role CR para permitir que los administradores de clúster otorguen acceso a recursos específicos en un namespace. Por ejemplo:

```
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. Crea y aplica una CR RoleBinding para vincular los permisos al usuario eng-user. Por ejemplo:

```
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. Verifica que los permisos sean correctos.

   a. Intenta recuperar la información del objeto AppVault para todos los espacios de nombres:

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

   Deberías ver una salida similar a la siguiente:

```
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. Prueba para ver si el usuario puede obtener la información de AppVault que ahora tiene permiso para acceder:

```
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
```

Deberías ver una salida similar a la siguiente:

```
yes
```

Resultado

Los usuarios a los que has concedido permisos de AppVault deberían poder usar los objetos autorizados de AppVault para operaciones de gestión de datos de la aplicación, y no deberían poder acceder a ningún recurso fuera de los espacios de nombres asignados ni crear nuevos recursos a los que no tengan acceso.

Supervisa los recursos de Trident Protect

Puedes usar las herramientas de código abierto kube-state-metrics, Prometheus y Alertmanager para monitorear la salud de los recursos protegidos por Trident Protect.

El servicio kube-state-metrics genera métricas a partir de la comunicación de la API de Kubernetes. Si lo usas con Trident Protect, te muestra información útil sobre el estado de los recursos en tu entorno.

Prometheus es un conjunto de herramientas que puede ingerir los datos generados por kube-state-metrics y presentarlos como información fácilmente legible sobre estos objetos. Juntos, kube-state-metrics y Prometheus te ofrecen una forma de monitorear la salud y el estado de los recursos que estás gestionando con Trident Protect.

Alertmanager es un servicio que ingiere las alertas enviadas por herramientas como Prometheus y las dirige a los destinos que tú configures.

> Las configuraciones y orientaciones incluidas en estos pasos son solo ejemplos; necesitas personalizarlas para que coincidan con tu entorno. Consulta la siguiente documentación oficial para instrucciones y soporte específicos:
>
> - "documentación de kube-state-metrics"
> - "Documentación de Prometheus"
> - "Documentación de Alertmanager"

Paso 1: instala las herramientas de supervisión

Para habilitar la supervisión de recursos en Trident Protect, necesitas instalar y configurar kube-state-metrics, Promethus y Alertmanager.

Instala kube-state-metrics

Puedes instalar kube-state-metrics usando Helm.

Pasos

1. Agrega el Helm chart de kube-state-metrics. Por ejemplo:

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

2. Aplica el CRD de Prometheus ServiceMonitor al clúster:

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

3. Crea un archivo de configuración para el Helm chart (por ejemplo, `metrics-config.yaml`). Puedes personalizar el siguiente ejemplo de configuración para que se ajuste a tu entorno:

metrics-config.yaml: configuración del Helm chart de 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. Instala kube-state-metrics desplegando el chart de Helm. Por ejemplo:

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

5. Configura kube-state-metrics para generar métricas para los recursos personalizados usados por Trident Protect siguiendo las instrucciones en el "documentación de recursos personalizados de kube-state-metrics".

Instala Prometheus

Puedes instalar Prometheus siguiendo las instrucciones en el "Documentación de Prometheus".

Instala Alertmanager

Puedes instalar Alertmanager siguiendo las instrucciones en "Documentación de Alertmanager".

Paso 2: configura las herramientas de supervisión para que funcionen juntas

Después de instalar las herramientas de supervisión, necesitas configurarlas para que funcionen juntas.

Pasos

1. Integra kube-state-metrics con Prometheus. Edita el archivo de configuración de Prometheus (`prometheus.yaml` y añade la información del servicio kube-state-metrics. Por ejemplo:

   prometheus.yaml: integración del servicio 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. Configura Prometheus para enrutar las alertas a Alertmanager. Edita el archivo de configuración de Prometheus (`prometheus.yaml` y agrega la siguiente sección:

prometheus.yaml: enviar alertas a Alertmanager

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

Resultado

Prometheus puede ahora recopilar métricas de kube-state-metrics y puede enviar alertas a Alertmanager. Ahora estás listo para configurar qué condiciones activan una alerta y a dónde se deben enviar las alertas.

Paso 3: Configura las alertas y los destinos de alertas

Después de configurar las herramientas para que funcionen juntas, necesitas configurar qué tipo de información activa las alertas y a dónde se deben enviar.

Ejemplo de alerta: fallo de backup

El siguiente ejemplo define una alerta crítica que se activa cuando el estado del recurso personalizado de copia de seguridad se establece en `Error` durante 5 segundos o más. Puedes personalizar este ejemplo para adaptarlo a tu entorno e incluir este fragmento YAML en tu `prometheus.yaml` archivo de configuración:

rules.yaml: define una alerta Prometheus para copias de seguridad fallidas

```
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."
```

Configura Alertmanager para enviar alertas a otros canales

Puedes configurar Alertmanager para enviar notificaciones a otros canales, como correo electrónico, PagerDuty, Microsoft Teams u otros servicios de notificación especificando la configuración respectiva en el archivo `alertmanager.yaml`.

El siguiente ejemplo configura Alertmanager para enviar notificaciones a un canal de Slack. Para adaptar este ejemplo a tu entorno, reemplaza el valor de la clave `api_url` con la URL del webhook de Slack que usas en tu entorno:

alertmanager.yaml: enviar alertas a un canal de 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 paquete de soporte Trident Protect

Trident Protect permite a los administradores generar paquetes que incluyen información útil para NetApp Support, incluyendo registros, métricas e información de topología sobre los clústeres y las apps bajo gestión. Si estás conectado a internet, puedes subir paquetes de soporte al NetApp Support Site (NSS) usando un archivo custom resource (CR).

Crea un paquete de soporte usando un CR

Pasos

1. Crea el archivo de recurso personalizado (CR) y ponle un nombre (por ejemplo, `trident-protect-support-bundle.yaml`).
2. Configura los siguientes atributos:
   - **metadata.name**: (*Required*) El nombre de este recurso personalizado; elige un nombre único y sensato para tu entorno.
   - **spec.triggerType**: (*Obligatorio*) determina si el paquete de soporte se genera de inmediato o se programa. La generación programada del paquete ocurre a las 12AM UTC. Valores posibles:
     - Programado
     - Manual
   - **spec.uploadEnabled**: (*Opcional*) Controla si el paquete de soporte debe cargarse en el NetApp Support Site después de generarse. Si no se especifica, el valor predeterminado es `false`. Valores posibles:
     - verdadero
     - false (predeterminado)
   - **spec.dataWindowStart**: (*Opcional*) Una cadena de fecha en formato RFC 3339 que especifica la fecha y hora en que debe comenzar la ventana de datos incluidos en el soporte bundle. Si no se especifica, el valor predeterminado es hace 24 horas. La fecha de ventana más temprana que puedes especificar es hace 7 días.

     Ejemplo de 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. Después de rellenar el archivo `trident-protect-support-bundle.yaml` con los valores correctos, aplica la CR:

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

Crea un paquete de soporte usando la CLI

Pasos

1. Crea el paquete de soporte, reemplazando los valores entre corchetes por información de tu entorno.

El `trigger-type` determina si el paquete se crea inmediatamente o si la hora de creación la dicta la programación, y puede ser `Manual` o `Scheduled`. La configuración predeterminada es `Manual`.

Por ejemplo:

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

Supervisa y recupera el paquete de soporte

Después de crear un paquete de soporte usando cualquiera de los dos métodos, puedes supervisar el progreso de su generación y recuperarlo en tu sistema local.

Pasos

1. Espera a que `status.generationState` llegue al estado `Completed`. Puedes monitorear el progreso de la generación con el siguiente comando:

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

2. Recupera el paquete de soporte en tu sistema local. Obtén el comando de copia del paquete completo AutoSupport:

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

   Busca el comando `kubectl cp` en la salida y ejecútalo, reemplazando el argumento de destino por tu directorio local preferido.

Actualizar Trident Protect

Puedes actualizar Trident Protect a la última versión para aprovechar las nuevas funciones o correcciones de errores.

- Al actualizar desde la versión 24.10, las instantáneas que se estén ejecutando durante la actualización pueden fallar. Este fallo no impide que se creen futuras instantáneas, ya sean manuales o programadas. Si una instantánea falla durante la actualización, puedes crear manualmente una nueva instantánea para asegurarte de que tu aplicación esté protegida.

  Para evitar posibles fallos, puedes desactivar todas las programaciones de instantáneas antes de la actualización y volver a activarlas después. Sin embargo, esto provoca que se pierdan las instantáneas programadas durante el periodo de actualización.

- Para instalaciones de registro privado, asegúrate de que el Helm chart requerido y las imágenes para la versión de destino estén disponibles en tu registro privado, y verifica que tus valores personalizados de Helm sean compatibles con la nueva versión del chart. Para obtener más información, consulta "Instala Trident Protect desde un registro privado".

Paso 1: selecciona una versión

Las versiones de Trident Protect siguen una convención de nomenclatura `YY.MM` basada en fechas, donde "YY" son los dos últimos dígitos del año y "MM" es el mes. Las versiones de punto siguen una `YY.MM.X` convención, donde "X" es el nivel de parche. Vas a seleccionar la versión a la que quieres actualizar según la versión desde la que estás actualizando.

- Puedes hacer una actualización directa a cualquier versión de destino que esté dentro de una ventana de cuatro versiones respecto a la versión que tienes instalada. Por ejemplo, puedes actualizar directamente de 24.10 (o cualquier versión puntual de 24.10) a 25.10.
- Si estás actualizando desde una versión fuera de la ventana de cuatro versiones, realiza una actualización en varios pasos. Usa las instrucciones de actualización para la "versión anterior" de la que estás actualizando para pasar a la versión más reciente que entre en el periodo de cuatro versiones. Por ejemplo, si estás ejecutando 24.10 y quieres actualizar a 26.02:
  a. Primero actualiza de 24.10 a 25.02.
  b. Luego actualiza de 25.02 a 26.02.

Paso 2: actualizar Trident Protect

Para actualizar Trident Protect, realiza los siguientes pasos.

Pasos

1. Actualiza el repositorio de Trident Helm:

```
helm repo update
```

2. Actualiza los CRD de Trident Protect:

   Este paso es necesario si estás actualizando desde una versión anterior a la 25.06, ya que los CRDs ahora están incluidos en el Helm chart de Trident Protect.

   a. Ejecuta este comando para cambiar la gestión de los CRD de `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. Ejecuta este comando para borrar el secreto de Helm del chart `trident-protect-crds`:

> No desinstales el chart `trident-protect-crds` usando Helm, ya que esto podría eliminar tus CRDs y cualquier dato relacionado.

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

3. Actualiza Trident Protect:

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

> Puedes configurar el nivel de registro durante la actualización añadiendo `--set logLevel=debug` al comando de actualización. El nivel de registro predeterminado es `warn`. Se recomienda el registro de depuración para la solución de problemas, ya que ayuda al soporte de NetApp a diagnosticar problemas sin necesidad de cambiar el nivel de registro ni reproducir el problema.