Administrar Trident Protect

Trident

NetApp
January 15, 2026

Tabla de contenidos

Administrar Trident Protect

Administrar la autorización y el control de acceso de Trident Protect

Trident Protect utiliza el modelo Kubernetes de control de acceso basado en roles (RBAC). De forma predeterminada, Trident Protect proporciona un único espacio de nombres de sistema y su cuenta de servicio predeterminada asociada. Si tiene una organización con muchos usuarios o necesidades de seguridad específicas, puede utilizar las funciones RBAC de Trident Protect para obtener 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 la configuración predeterminada. `trident-protect` espacio de nombres, y también puede acceder a recursos en todos los demás espacios de nombres. Para controlar el acceso a los recursos y las aplicaciones, debe crear espacios de nombres adicionales y agregar los recursos y las aplicaciones a esos espacios de nombres.

Tenga en cuenta que ningún usuario puede crear solicitudes de cambio (CR) de administración de datos de aplicaciones en la configuración predeterminada. `trident-protect` espacio de nombres. Debe crear los CR de administración de datos de la aplicación en un espacio de nombres de la aplicación (como práctica recomendada, cree los CR de administración de datos de la aplicación 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 del bucket
> - **AutoSupportBundle**: Recopila métricas, registros y otros datos confidenciales de Trident Protect
> - **AutoSupportBundleSchedule**: Gestiona los horarios de recopilación de registros.
>
> Como práctica recomendada, utilice 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, consulte la documentación. "Documentación de RBAC de Kubernetes" .

Para obtener información sobre las cuentas de servicio, consulte la "Documentación de la cuenta de servicio de Kubernetes" .

Ejemplo: Gestionar el acceso para 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 del clúster completaría las siguientes tareas para crear un entorno donde el grupo de ingeniería y el grupo de marketing tengan acceso únicamente a los recursos asignados a sus respectivos espacios de nombres.

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

La creación de un espacio de nombres le permite separar lógicamente los recursos y controlar mejor quién tiene acceso a ellos.

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: Cree nuevas cuentas de servicio para interactuar con los recursos en cada espacio de nombres.

Cada nuevo espacio de nombres que cree viene con una cuenta de servicio predeterminada, pero debería crear una cuenta de servicio para cada grupo de usuarios para que pueda dividir aún más los privilegios entre grupos en el futuro si fuera necesario.

Pasos

1. Cree 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: Cree un secreto para cada nueva cuenta de servicio

Se utiliza una clave secreta de cuenta de servicio para autenticarse con la cuenta de servicio, y se puede eliminar y volver a crear fácilmente si se ve comprometida.

Pasos

1. Cree un secreto para la cuenta del 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 una clave secreta 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: Cree un objeto RoleBinding para vincular el objeto ClusterRole a cada nueva cuenta de servicio.

Se crea un objeto ClusterRole predeterminado cuando instala Trident Protect. Puede vincular este ClusterRole a la cuenta de servicio creando y aplicando un objeto RoleBinding.

Pasos

1. Vincule 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 del 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 permisos

Comprueba que los permisos sean correctos.

Pasos

1. Confirme 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: Conceder acceso a los objetos de AppVault

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

Pasos

1. Cree y aplique un archivo YAML de combinación de AppVault y secreto que otorgue a un usuario acceso a un AppVault. Por ejemplo, la siguiente solicitud CR otorga 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. Cree y aplique una solicitud de cambio de rol para permitir que los administradores del clúster otorguen acceso a recursos específicos en un espacio de nombres. 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. Cree y aplique un 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. Verifique que los permisos sean correctos.

   a. Intento de recuperar la información de objetos de AppVault para todos los espacios de nombres:

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

   Debería ver un resultado similar al 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 comprobar si el usuario puede obtener la información de AppVault a la que ahora tiene permiso de acceso:

```
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
```

Debería ver un resultado similar al siguiente:

```
yes
```

Resultado

Los usuarios a los que les haya concedido permisos de AppVault deberían poder utilizar los objetos autorizados de AppVault para las 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.

Supervisar los recursos de Trident Protect

Puede utilizar las herramientas de código abierto kube-state-metrics, Prometheus y Alertmanager para supervisar el estado de los recursos protegidos por Trident Protect.

El servicio kube-state-metrics genera métricas a partir de la comunicación con la API de Kubernetes. Usarlo con Trident Protect expone información útil sobre el estado de los recursos en su 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 le brindan una manera de monitorear la salud y el estado de los recursos que administra con Trident Protect.

Alertmanager es un servicio que recibe las alertas enviadas por herramientas como Prometheus y las dirige a los destinos que usted configure.

> Las configuraciones y la orientación incluidas en estos pasos son solo ejemplos; debe personalizarlas para que se ajusten a su entorno. Consulte la siguiente documentación oficial para obtener instrucciones y asistencia específicas:
>
> - "Documentación de kube-state-metrics"
> - "Documentación de Prometeo"
> - "Documentación de Alertmanager"

Paso 1: Instalar las herramientas de monitorización

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

Instalar kube-state-metrics

Puedes instalar kube-state-metrics usando Helm.

Pasos

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

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

2. Aplique 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 gráfico de Helm (por ejemplo, `metrics-config.yaml`). Puede personalizar la siguiente configuración de ejemplo para que se ajuste a su entorno:

metrics-config.yaml: Configuración del gráfico Helm 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. Instale kube-state-metrics implementando el gráfico Helm. Por ejemplo:

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

5. Configure kube-state-metrics para generar métricas para los recursos personalizados utilizados por Trident Protect siguiendo las instrucciones en "Documentación del recurso personalizado kube-state-metrics" .

Instalar Prometheus

Puedes instalar Prometheus siguiendo las instrucciones del manual. "Documentación de Prometeo" .

Instalar Alertmanager

Puede instalar Alertmanager siguiendo las instrucciones del siguiente enlace: "Documentación de Alertmanager" .

Paso 2: Configurar las herramientas de monitorización para que funcionen conjuntamente.

Después de instalar las herramientas de monitorización, deberá configurarlas para que funcionen conjuntamente.

Pasos

1. Integra kube-state-metrics con Prometheus. Edita el archivo de configuración de Prometheus(`prometheus.yaml`) y agregue 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 que dirija las alertas a Alertmanager. Edita el archivo de configuración de Prometheus(`prometheus.yaml`) y agregue la siguiente sección:

prometheus.yaml: Enviar alertas a Alertmanager

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

Resultado

Prometheus ahora puede recopilar métricas de kube-state-metrics y enviar alertas a Alertmanager. Ahora ya puedes configurar qué condiciones activan una alerta y dónde deben enviarse las alertas.

Paso 3: Configurar alertas y destinos de alertas

Una vez configuradas las herramientas para que funcionen conjuntamente, es necesario configurar qué tipo de información activa las alertas y dónde deben enviarse dichas alertas.

Ejemplo de alerta: fallo de copia de seguridad

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 que se ajuste a tu entorno e incluir este fragmento YAML en tu `prometheus.yaml` archivo de configuración:

rules.yaml: Define una alerta de 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.

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

El siguiente ejemplo configura Alertmanager para enviar notificaciones a un canal de Slack. Para personalizar este ejemplo a su entorno, reemplace el valor de `api_url` clave con la URL del webhook de Slack utilizada en su 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
```

Generar un paquete de soporte de Trident Protect

Trident Protect permite a los administradores generar paquetes que incluyen información útil para el soporte de NetApp , incluidos registros, métricas e información de topología sobre los clústeres y las aplicaciones bajo administración. Si está conectado a Internet, puede cargar paquetes de soporte en el sitio de soporte de NetApp (NSS) mediante un archivo de recursos personalizados (CR).

Cree un paquete de soporte utilizando un CR.

Pasos

1. Cree el archivo de recursos personalizados (CR) y asígnele un nombre (por ejemplo, `trident-protect-support-bundle.yaml`).
2. Configure los siguientes atributos:
   - **metadata.name**: (*Obligatorio*) El nombre de este recurso personalizado; elija un nombre único y adecuado para su entorno.
   - **spec.triggerType**: (*Obligatorio*) Determina si el paquete de soporte se genera inmediatamente o se programa. La generación programada de paquetes se produce a las 12 AM UTC. Valores posibles:
     - Programado
     - Manual
   - **spec.uploadEnabled**: (*Opcional*) Controla si el paquete de soporte debe cargarse en el sitio de soporte de NetApp después de generarse. Si no se especifica, el valor predeterminado es `false` . Valores posibles:
     - verdadero
     - falso (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 paquete de soporte. Si no se especifica, se tomará como valor predeterminado las últimas 24 horas. La fecha más antigua 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 `trident-protect-support-bundle.yaml` Archivo con los valores correctos, aplicar el CR:

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

Crea un paquete de soporte usando la CLI.

Pasos

1. Cree el paquete de soporte, reemplazando los valores entre corchetes con información de su entorno. El `trigger-type` Determina si el paquete se crea inmediatamente o si el tiempo de creación viene dictado por 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
   ```

Supervisar y recuperar el paquete de soporte

Después de crear un paquete de soporte utilizando cualquiera de los métodos, puede supervisar el progreso de su generación y recuperarlo en su sistema local.

Pasos

1. Espera a que `status.generationState` para alcanzar `Completed` estado. Puede monitorear el progreso de la generación con el siguiente comando:

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

2. Recupere el paquete de soporte para su sistema local. Obtenga el comando de copia del paquete AutoSupport completado:

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

   Encuentra el `kubectl cp` Copie el comando de la salida y ejecútelo, reemplazando el argumento de destino con el directorio local que prefiera.

Actualizar Trident Protect

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

> Al actualizar desde la versión 24.10, es posible que fallen las instantáneas que se ejecutan durante la actualización. Este fallo no impide que se creen instantáneas futuras, ya sean manuales o programadas. Si una instantánea falla durante la actualización, puede crear manualmente una nueva instantánea para garantizar la protección de su aplicación.
>
> Para evitar posibles fallos, puede deshabilitar todas las programaciones de instantáneas antes de la actualización y volver a habilitarlas después. Sin embargo, esto provoca que se pierdan las instantáneas programadas durante el período de actualización.

Para actualizar Trident Protect, realice los siguientes pasos.

Pasos

1. Actualizar el repositorio de Trident Helm:

```
helm repo update
```

2. Actualice los CRD Trident Protect:

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

a. Ejecute este comando para transferir la administración de CRD desde `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 eliminar el secreto de Helm para el `trident-protect-crds` cuadro:

> No desinstale el `trident-protect-crds` No utilice Helm para crear gráficos, ya que esto podría eliminar sus CRD y cualquier dato relacionado.

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

3. Mejora Trident Protect:

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