管理 Trident Protect

Trident

NetApp
July 01, 2026

目錄

管理 Trident Protect

管理 Trident Protect 授權和存取控制

Trident Protect 使用 Kubernetes 的角色型存取控制（RBAC）模型。預設情況下，Trident Protect 提供單一系統命名空間及其相關的預設服務帳戶。如果您的組織擁有眾多使用者或有特定的安全需求，則可以使用 Trident Protect 的 RBAC 功能來更精細地控制對資源和命名空間的存取。

叢集管理員始終擁有對預設 `trident-protect` 命名空間中資源的存取權限，並且還可以存取所有其他命名空間中的資源。要控制對資源和應用程式的存取，您需要建立其他命名空間，並將資源和應用程式新增至這些命名空間。

請注意，任何使用者都無法在預設 `trident-protect`命名空間中建立應用程式資料管理 CR。您需要在應用程式命名空間中建立應用程式資料管理 CR（最佳實踐是將應用程式資料管理 CR 建立在與其關聯應用程式相同的命名空間中）。

> 只有管理員才能存取具有特權的 Trident Protect 自訂資源物件，其中包括：
>
> - **AppVault**：需要儲存桶憑證資料
> - **AutoSupportBundle**：收集指標、日誌和其他敏感的 Trident Protect 數據
> - **AutoSupportBundleSchedule**：管理日誌收集排程
>
> 最佳實踐是使用 RBAC 將對特權物件的存取限制在管理員範圍內。

如需有關 RBAC 如何管理對資源和命名空間的存取的詳細資訊，請參閱 "Kubernetes RBAC 文件"。

如需服務帳戶的相關資訊，請參閱 "Kubernetes 服務帳戶文件"。

範例：管理兩組使用者的存取權限

例如，一個組織有一名 cluster 管理員、一組工程用戶和一組行銷用戶。cluster 管理員需要完成以下任務，以建立一個環境，使工程使用者群組和行銷使用者群組各自只能存取分配給其各自 namespace 的資源。

步驟 **1**：建立命名空間以包含每個群組的資源

建立命名空間可讓您以邏輯方式分隔資源，並更有效地控制誰可以存取這些資源。

步驟

1. 為工程群組建立命名空間：

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

2. 為行銷組建立命名空間：

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

步驟 **2**：建立新的服務帳戶、以便與每個命名空間中的資源互動

您建立的每個新命名空間都附帶一個預設服務帳戶，但您應該為每個使用者群組建立一個服務帳戶，以便將來必要時可以進一步在群組之間劃分權限。

步驟

1. 為工程群組建立服務帳戶：

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

2. 為行銷群組建立服務帳戶：

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

步驟 **3**：為每個新服務帳戶建立一個密碼

服務帳戶密碼用於對服務帳戶進行身分驗證，如果遭到洩露，可以輕鬆刪除並重新建立。

步驟

1. 為工程服務帳戶建立密鑰：

```
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. 為行銷服務帳戶建立密鑰：

```
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
```

步驟 **4**：建立一個 **RoleBinding** 對象，並將該 **ClusterRole** 對象綁定到每個新的服務帳戶

當您安裝 Trident Protect 時，會自動建立一個預設的 ClusterRole 物件。您可以透過建立並套用 RoleBinding 物件，將此 ClusterRole 綁定至服務帳戶。

步驟

1. 將 ClusterRole 綁定到工程服務帳戶：

```
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. 將 ClusterRole 綁定到行銷服務帳戶：

```
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
```

步驟 **5**：測試權限

測試權限是否正確。

步驟

1. 確認工程使用者可以存取工程資源：

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

2. 確認工程使用者無法存取行銷資源：

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

步驟 **6**：授予 **AppVault** 物件存取權限

要執行備份和快照等資料管理任務，叢集管理員需要授予個別使用者對 AppVault 物件的存取權限。

步驟

1. 建立並套用 AppVault 和密碼組合 YAML 檔案，授予使用者存取 AppVault 的權限。例如，以下 CR 授予使用者存取 AppVault 的權限 `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. 建立並套用 Role CR，以使叢集管理員能夠授予對命名空間中特定資源的存取權。例如：

```
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. 建立並套用 RoleBinding CR，將權限綁定到使用者 eng-user。例如：

```
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. 確認權限是否正確。

   a. 嘗試檢索所有命名空間的 AppVault 物件資訊：

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

   您應該會看到類似以下內容的輸出：

```
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. 測試用戶是否可以獲得他們現在有權訪問的 AppVault 資訊：

```
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
```

您應該會看到類似以下內容的輸出：

```
yes
```

結果

您授予 AppVault 權限的使用者應該能夠使用授權 AppVault 物件進行應用程式資料管理操作，並且不應該能夠存取指派的命名空間以外的任何資源，或建立他們無權存取的新資源。

監控 Trident Protect 資源

您可以使用 kube-state-metrics、Prometheus 和 Alertmanager 這些開源工具來監控由 Trident Protect 保護的資源健康狀況。

kube-state-metrics 服務會從 Kubernetes API 通訊產生指標。將其與 Trident Protect 結合使用，可公開有關環境中資源狀態的實用資訊。

Prometheus 是一個工具包，可以接收由 kube-state-metrics 產生的數據，並將其呈現為易於閱讀的這些物件資訊。kube-state-metrics 和 Prometheus 結合使用，為你提供一種方式來監控你使用 Trident Protect 管理的資源的健康狀態和狀態。

Alertmanager 是一項服務，它可以接收 Prometheus 等工具發送的警報，並將它們路由到您配置的目標位置。

> 這些步驟中包含的組態和指南僅為範例；您需要根據自己的環境進行自訂。有關具體說明和支援，請參閱以下官方文件：
>
> - "kube-state-metrics 說明文件"
> - "Prometheus 說明文件"
> - "Alertmanager 文件"

步驟 1：安裝監控工具

要在 Trident Protect 中啟用資源監控，您需要安裝和設定 kube-state-metrics、Prometheus 和 Alertmanager。

安裝 **kube-state-metrics**

您可以使用 Helm 安裝 kube-state-metrics。

步驟

1. 新增 kube-state-metrics Helm chart。例如：

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

2. 將 Prometheus ServiceMonitor CRD 應用於叢集：

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

3. 為 Helm chart 建立一個設定檔（例如， `metrics-config.yaml`）。您可以自訂以下範例設定以符合您的環境：

metrics-config.yaml：kube-state-metrics Helm chart 配置

```
---
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. 透過部署 Helm chart 來安裝 kube-state-metrics。例如：

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

5. 請依照 "kube-state-metrics 自訂資源文件"中的說明配置 kube-state-metrics，以產生 Trident Protect 使用的自訂資源的指標。

安裝 **Prometheus**

您可以按照 "Prometheus 說明文件"中的說明安裝 Prometheus。

安裝 **Alertmanager**

您可以按照 "Alertmanager 文件"中的說明安裝 Alertmanager。

步驟 **2**：設定監控工具以協同運作

安裝監控工具後，您需要設定它們以使其協同工作。

步驟

1. 將 kube-state-metrics 與 Prometheus 整合。編輯 Prometheus 配置文件 (`prometheus.yaml`）並新增 kube-state-metrics 服務資訊。例如：

   prometheus.yaml：**kube-state-metrics** 服務與 **Prometheus** 的整合

```
---
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. 設定 Prometheus 將警示路由至 Alertmanager。編輯 Prometheus 組態檔 (`prometheus.yaml`，並新增以下區段：

prometheus.yaml：將警報傳送至 **Alertmanager**

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

結果

Prometheus 現在可以從 kube-state-metrics 收集指標，並向 Alertmanager 發送警報。現在您可以設定觸發警報的條件以及警報的發送位置。

步驟 3：設定警示和警示目的地

配置好工具協同工作後，您需要配置哪些類型的資訊會觸發警示，以及警示應該發送到哪裡。

警示範例：備份失敗

以下範例定義了一個關鍵警報，當備份自訂資源的狀態設定為 `Error`且持續時間達到 5 秒或更長時，將觸發該警報。您可以根據自己的環境自訂此範例，並將此 YAML 程式碼片段新增至您的 `prometheus.yaml`組態檔：

rules.yaml：定義備份失敗的 **Prometheus** 警報

```
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."
```

配置 **Alertmanager** 以將警示傳送至其他管道

您可以設定 Alertmanager，使其會向其他管道（例如電子郵件、PagerDuty、Microsoft Teams 或其他通知服務）發送通知，只需在 `alertmanager.yaml` 文件中指定相應的配置即可。

以下範例配置 Alertmanager 向 Slack 頻道發送通知。若要根據您的環境自訂此範例，請將 `api_url` 鍵值替換為您環境中使用的 Slack webhook URL：

alertmanager.yaml ：向 **Slack** 頻道發送警報

```
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
```

產生 Trident Protect 支援套件

Trident Protect 讓管理員產生包含對 NetApp 支援團隊有用的資訊的軟體包，例如受管理叢集和應用程式的日誌、指標和拓撲資訊。如果您已連接到互聯網，則可以使用自訂資源（CR）檔案將支援軟體包上傳至 NetApp 支援網站（NSS）。

使用 **CR** 建立支援套件

步驟

1. 建立自訂資源（CR）檔案並將其命名為（例如、 `trident-protect-support-bundle.yaml`）。
2. 設定下列屬性：
   - **metadata.name**：（必填）此自訂資源的名稱；請為您的環境選擇一個唯一且有意義的名稱。
   - **spec.triggerType**：（必填）確定支援包是立即產生還是定時產生。定時產生會在 UTC 時間凌晨 12 點進行。可選值：
     - 已排程
     - 手動
   - **spec.uploadEnabled**：（選用）控制產生支援服務包後是否應將其上傳至 NetApp 支援網站。如果未指定、則預設為 `false`。可選值：
     - true
     - false （預設）
   - **spec.dataWindowStart**：（選用）RFC 3339 格式的日期字串，用於指定支援套件中包含的資料視窗的起始日期和時間。如果未指定，則預設為 24 小時前。您可以指定的最早視窗日期為 7 天前。

     YAML 範例：

```
---
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. 在 `trident-protect-support-bundle.yaml` 檔案中填入正確的值後，套用 CR：

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

使用 **CLI** 建立支援套件

步驟

1. 建立支援包，將括號中的值替換為您環境中的資訊。 `trigger-type`決定是立即建立支援包還是根據排程建立，其值可以是 `Manual`或 `Scheduled`。預設設定為 `Manual`。

   例如：

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

監控並擷取支援服務包

使用任一方法建立支援套件後，您可以監控其產生進度並將其擷取到本機系統。

步驟

1. 等待 `status.generationState` 達到 `Completed` 狀態。您可以使用以下命令監控產生進度：

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

2. 將支援套件組合擷取至本機系統。從已完成的 AutoSupport 套件組合取得複製命令：

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

從輸出中找到 `kubectl cp`命令並運行它，將目標參數替換為您首選的本機目錄。

升級 Trident Protect

您可以將 Trident Protect 升級至最新版本，以利用新功能或錯誤修正。

- 從 24.10 版本升級時，升級期間執行的快照可能會失敗。此失敗不會阻止未來的快照（無論是手動還是排程）被建立。如果快照在升級期間失敗，您可以手動建立新快照以確保應用程式受到保護。

  為避免潛在故障，您可以在升級前停用所有快照排程，並在升級後重新啟用。但是，這樣做會導致升級期間所有已排程的快照遺失。

- 對於私有鏡像倉庫安裝，請確保目標版本所需的 Helm chart 和鏡像已存在於您的私有鏡像倉庫中，並驗證您的自訂 Helm 值與新 chart 版本相容。如需更多資訊，請參閱 "從私人註冊表安裝 Trident Protect"。

步驟 1：選擇版本

Trident Protect 版本遵循基於日期的 `YY.MM`命名規則，其中「YY」代表年份的後兩位數字，「MM」代表月份。小版本更新遵循 `YY.MM.X`規則，其中「X」代表補丁級別。您將根據要升級的版本選擇要升級到的版本。

- 您可以將目前版本直接升級到與其相差不超過四個版本號的目標版本。例如，您可以直接從 24.10（或任何 24.10 的小版本）升級到 25.10。

- 如果您要從超出四版本視窗期的版本升級，請執行多步驟升級。使用您要升級的 "早期版本" 版本對應的升級說明，升級到符合四版本視窗期的最新版本。例如，如果您目前運行的是 24.10 版本，並且想要升級到 26.02 版本：
  a. 首次從 24.10 升級到 25.02。
  b. 然後從 25.02 升級到 26.02 。

步驟 2：升級 Trident Protect

若要升級 Trident Protect、請執行下列步驟。

步驟

1. 更新 Trident Helm 儲存庫：

```
helm repo update
```

2. 升級 Trident Protect CRD：

> 如果您是從 25.06 之前的版本升級，則需要執行此步驟，因為 CRD 現在已包含在 Trident Protect Helm 圖表中。

a. 執行此命令將 CRD 的管理權限從 `trident-protect-crds`切換到 `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. 執行以下指令刪除 `trident-protect-crds` chart 的 Helm secret：

> 請勿使用 Helm 卸載 `trident-protect-crds`圖表，因為這可能會刪除您的 CRD 和任何相關資料。

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

3. 升級 Trident Protect：

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

> 您可以透過在升級命令中新增 `--set logLevel=debug`來配置升級期間的日誌等級。預設日誌等級為 `warn`。建議啟用偵錯日誌以進行故障排除，因為它有助於 NetApp 支援人員診斷問題，而無需更改日誌等級或重現問題。