管理 Trident Protect

Trident

NetApp
November 14, 2025

This PDF was generated from https://docs.netapp.com/zh-tw/trident/trident-protect/manage-authorization-access-control.html on November 14, 2025. Always check docs.netapp.com for the latest.

目錄

管理 Trident Protect . . . 1
管理 Trident 保護授權與存取控制 . . . 1
範例：管理兩組使用者的存取權 . . . 1
監控 Trident Protect 資源 . . . 7
步驟 1 ：安裝監控工具 . . . 7
步驟 2 ：設定監控工具以共同作業 . . . 10
步驟 3 ：設定警示和警示目的地 . . . 11
產生 Trident Protect 支援服務組合 . . . 12
監視和檢索支援包 . . . 14
升級 Trident Protect . . . 14

管理 Trident Protect

管理 Trident 保護授權與存取控制

Trident Protect 採用 Kubernetes 角色型存取控制（RBAC）模式。根據預設， Trident Protect 提供單一系統命名空間及其相關的預設服務帳戶。如果組織有許多使用者或特定的安全需求，您可以使用 Trident Protect 的 RBAC 功能，更精細地控制對資源和命名空間的存取。

叢集管理員一律可以存取預設命名空間中的資源 `trident-protect`，也可以存取所有其他命名空間中的資源。若要控制對資源和應用程式的存取，您需要建立額外的命名空間，並將資源和應用程式新增至這些命名空間。

請注意，沒有使用者可以在預設命名空間中建立應用程式資料管理 CRS `trident-protect` 。您需要在應用程式命名空間中建立應用程式資料管理 CRS （最佳做法是在與其相關應用程式相同的命名空間中建立應用程式資料管理 CRS）。

> 只有系統管理員才能存取授權的 Trident 保護自訂資源物件，包括：
>
> - * AppVault* ：需要儲存庫認證資料
> - * AutoSupportBundle * ：收集指標，記錄及其他敏感的 Trident Protect 資料
> - * AutoSupportBundleSchedule* ：管理記錄收集排程
>
> 最佳做法是使用 RBAC 來限制系統管理員存取權限物件。

如需 RBAC 如何規範資源和命名空間存取的詳細資訊，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

例如，組織有叢集管理員，一組工程設計使用者，以及一組行銷使用者。叢集管理員將完成下列工作，以建立一個環境，其中工程群組和行銷群組各自只能存取指派給各自命名空間的資源。

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，並更有效地控制誰有權存取這些資源。

步驟

1. 為工程群組建立命名空間：

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

2. 為行銷群組建立命名空間：

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

步驟 **2** ：建立新的服務帳戶，與每個命名空間中的資源互動

您所建立的每個新命名空間都有預設服務帳戶，但您應該為每個使用者群組建立服務帳戶，以便日後在必要時在群組之間進一步分割 Privileges 。

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** ：為每個新的服務帳戶建立秘密

服務帳戶密碼是用來驗證服務帳戶，如果受到入侵，也可以輕鬆刪除和重新建立。

步驟

1. 為工程服務帳戶建立秘密：

```
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秘密：

```yaml
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 **RoleBinding** 物件，將 **ClusterRole** 物件繫結至每個新的服務帳戶

安裝 Trident Protect 時會建立預設的 ClusterRole 物件。您可以建立並套用角色繫結物件，將此 ClusterRole 繫結至服務帳戶。

步驟

1. 將 ClusterRole 繫結至工程服務帳戶：

```yaml
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 連結至行銷服務帳戶：

```yaml
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** ：授予對 **AppVault** 物件的存取權

若要執行資料管理工作，例如備份和快照，叢集管理員必須將 AppVault 物件的存取權授予個別使用者。

步驟

1. 建立並套用 AppVault 和加密組合 YAML 檔案，以授予使用者存取 AppVault 的權限。例如，下列 CR 將 AppVault 的存取權授予使用者 `eng-user`：

```yaml
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. 建立並套用角色 CR ，讓叢集管理員能夠授與對命名空間中特定資源的存取權。例如：

```yaml
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 ，將權限繫結至使用者 eng-user 。例如：

```yaml
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. 確認權限正確。

   a. 嘗試擷取所有命名空間的 AppVault 物件資訊：

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

   您應該會看到類似下列的輸出：

```
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. 測試以查看使用者是否能取得他們現在有權存取的 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
```

您應該會看到類似下列的輸出：

```
yes
```

結果

您已授予 AppVault 權限的使用者應該能夠使用授權的 AppVault 物件來執行應用程式資料管理作業，而且不應能夠存取指派命名空間以外的任何資源，或建立他們無法存取的新資源。

監控 Trident Protect 資源

您可以使用 kube-state 度量， Prometheus 和 Alertmanager 開放原始碼工具來監控受 Trident Protect 保護的資源健全狀況。

kube-state 度量服務會從 Kubernetes API 通訊產生度量。搭配 Trident Protect 使用可提供環境中資源狀態的實用資訊。

Prometheus 是一個工具組，可擷取由 kube 狀態度量所產生的資料，並將其呈現為這些物件的易讀資訊。kube 狀態指標和 Prometheus 共同提供一種方法，讓您使用 Trident Protect 監控所管理資源的健全狀況和狀態。

AlertManager 是一項服務，可擷取 Prometheus 等工具所傳送的警示，並將其路由至您設定的目的地。

> 這些步驟所包含的組態和指南僅為範例，您需要自訂以符合您的環境。請參閱下列正式文件，以取得特定指示與支援：
>
> - "Kube-state指標文件"
> - "Prometheus 文件"
> - "AlertManager 文件"

步驟 1 ：安裝監控工具

若要在 Trident Protect 中啟用資源監控，您必須安裝及設定 kube-st狀態 度量， Promethus 和 Alertmanager 。

安裝 **kube** 狀態度量

您可以使用 Helm 來安裝 kube 狀態度量。

步驟

1. 新增 kube 狀態指標 Helm 圖表。例如：

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

2. 將 Prometheus ServiceMonitor CRD 應用到叢集：

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

3. 為 Helm 圖表建立組態檔（例如 `metrics-config.yaml`）。您可以自訂下列範例組態，以符合您的環境：

測量數據 **-config.yaml** ： **Kube-state** 度量 **Helm** 圖表組態

```yaml
---
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. 部署 Helm 圖表以安裝 kube 狀態度量。例如：

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

5. 依照中的指示，設定 kube 狀態度量，以產生 Trident Protect 所使用之自訂資源的度量 "Kube-state 度量自訂資源文件"。

安裝Prometheus

您可以依照中的指示來安裝 Prometheus "Prometheus 文件" 。

安裝 AlertManager

您可以依照中的指示安裝 AlertManager "AlertManager 文件" 。

步驟 2 ：設定監控工具以共同作業

安裝監控工具之後，您需要將它們設定為一起運作。

步驟

1. 將 kube 狀態指標與 Prometheus 整合。編輯 Prometheus 配置文件(`prometheus.yaml`）並添加 kube 狀態指標服務信息。例如：

   prometheus.yaml：kube-state-metrics 服務與 Prometheus 的集成

```
---
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 配置文件(`prometheus.yaml`）並添加以下部分：

prometheus.yaml：向 **Alertmanager** 發送警報

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

結果

現在， Prometheus 可以從 kube-state 度量收集度量，並可傳送警示給 Alertmanager 。您現在已準備好設定觸發警示的條件，以及應傳送警示的位置。

步驟 3 ：設定警示和警示目的地

設定工具以共同作業之後，您需要設定觸發警示的資訊類型，以及應傳送警示的位置。

警示範例：備份失敗

以下範例定義當備份自訂資源的狀態設定為 5 秒或更長時間時觸發的關鍵警示 `Error`。您可以自訂此範例以符合您的環境，並將此 YAML 片段包含在組態檔案中 `prometheus.yaml`：

rules.yaml：定義失敗備份的 **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."
```

設定 **AlertManager** 以傳送警示至其他頻道

您可以將 AlertManager 設定為傳送通知給其他通道，例如電子郵件， PagerDuty ， Microsoft 團隊或其他通知服務，方法是在檔案中指定個別的組態 `alertmanager.yaml`。

以下範例將警示管理員設定為傳送通知至 Slack 頻道。若要根據您的環境自訂此範例，請將金鑰的值取代為 `api_url`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 支援服務組合

Trident Protect 使管理員能夠產生包含對NetApp支援有用的信息的捆綁包，包括有關所管理的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) 檔案將支援包上傳至NetApp支援網站 (NSS)。

使用 **CR** 建立支援服務組合

步驟

1. 建立自訂資源（CR）檔案並命名（例如 `trident-protect-support-bundle.yaml`）。
2. 設定下列屬性：
   - * metadata.name*: （_required）此自訂資源的名稱；為您的環境選擇唯一且合理的名稱。
   - **spec.triggerType** ：（_required _）決定是立即產生支援套件，還是排程產生。排定的套件產生時間為上午 12 點， UTC 。可能值：
     - 已排程
     - 手冊
   - **SPEC.uploadEnabled** ：（*Optional*）控制是否應在支援服務組合產生後，將其上傳至 NetApp 支援網站。如果未指定，則默認爲 `false`。可能值：
     - 是的
     - 否（預設）
   - **spec.daWindowStart** ：（*Optional*） RFC 3339 格式的日期字串，指定支援套件中所包含資料的視窗應開始的日期與時間。如果未指定，則預設為 24 小時前。您可以指定的最早時間是 7 天前。

     YAML 範例：

```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. 填充後 `trident-protect-support-bundle.yaml`具有正確值的文件，應用 CR：

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

使用 **CLI** 建立支援服務包

步驟

1. 建立支援服務組合，以環境資訊取代括號中的值。 `trigger-type`決定套件是立即建立，還是建立時間取決於排程，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
```

監視和檢索支援包

使用任一方法建立支援包後，您可以監視其產生進度並將其檢索到本機系統。

步驟

1. 等待 `status.generationState`到達 `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. 將支援包檢索到您的本機系統。從已完成的AutoSupport套件中取得複製指令：

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

找到 `kubectl cp`從輸出執行命令並運行它，用您喜歡的本機目錄取代目標參數。

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。

- 從版本 24.10 升級時，升級期間執行的快照可能會失敗。此失敗不會阻止將來建立快照（無論是手動快照還是計劃快照）。如果升級期間快照失敗，您可以手動建立新快照以確保應用程式受到保護。

  為避免潛在的故障，您可以在升級前停用所有快照計劃，然後在升級後重新啟用。但是，這會導致升級期間遺失所有計劃的快照。

- 對於私人鏡像倉庫安裝，請確保目標版本所需的 Helm Chart 和鏡像在您的私人鏡像倉庫中可用，並驗證您的自訂 Helm 值與新 Chart 版本相容。更多信息，請參閱"從私有登錄安裝 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 客戶需求日：

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

a. 運行此命令將 CRD 的管理從 `trident-protect-crds`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. 運行此命令刪除 `trident-protect-crds`圖表：

> 不要卸載 `trident-protect-crds`圖表使用 Helm，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 100.2510.0 --namespace trident-protect
```

> 您可以透過新增以下內容來配置升級期間的日誌等級。 `--set logLevel=debug`升級命令。預設日誌等級為 `warn`。建議啟用偵錯日誌記錄進行故障排除，因為它可以幫助NetApp支援人員診斷問題，而無需更改日誌等級或重現問題。

版權資訊

Copyright © 2025 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變更本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。 美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 http://www.netapp.com/TM 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。