



管理Trident Protect Trident

NetApp
January 15, 2026

目錄

管理Trident Protect	1
管理Trident Protect 授權和存取控制	1
範例：管理兩組使用者的存取權限	1
監控Trident保護資源	7
步驟 1：安裝監控工具	7
步驟 2：配置監控工具以協同工作	10
步驟 3：設定警報和警報目標	11
產生Trident Protect 支援包	12
監視和檢索支援包	14
升級Trident保護	14

管理Trident Protect

管理Trident Protect 授權和存取控制

Trident Protect 使用 Kubernetes 的角色為基礎的存取控制 (RBAC) 模型。預設情況下，Trident Protect 提供一個系統命名空間及其關聯的預設服務帳戶。如果您的組織擁有眾多使用者或特定的安全需求，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資源的存取權限。`trident-protect`命名空間，並且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訪問，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，預設情況下，任何使用者都無法建立應用程式資料管理變更請求 (CR)。`trident-protect`命名空間。您需要在應用程式命名空間中建立應用程式資料管理 CR（最佳實踐是在與其關聯的應用程式相同的命名空間中建立應用程式資料管理 CR）。

只有管理員才能存取具有特權的Trident Protect 自訂資源對象，其中包括：



- **AppVault**：需要儲存桶憑證數據
- **AutoSupportBundle**：收集指標、日誌和其他敏感的Trident Protect數據
- **AutoSupportBundleSchedule**：管理日誌收集計劃

最佳實踐是使用基於角色的存取控制 (RBAC) 將對特權物件的存取限制在管理員範圍內。

有關基於角色的存取控制 (RBAC) 如何管理對資源和命名空間的存取的更多信息，請參閱... ["Kubernetes RBAC 文檔"](#)。

有關服務帳戶的信息，請參閱 ["Kubernetes 服務帳戶文檔"](#)。

範例：管理兩組使用者的存取權限

例如，一個組織有集群管理員、一組工程用戶和一組行銷用戶。叢集管理員將完成以下任務，以建立一個環境，其中工程組和行銷組各自只能存取分配給其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

創建命名空間可以让你從邏輯上分離資源，並更好地控制誰可以存取這些資源。

步驟

1. 為工程組建立一個命名空間：

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

2. 為行銷組創造命名空間：

```
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. 為工程團隊建立一個服務帳戶：

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

2. 為行銷團隊建立一個服務帳戶：

```
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金鑰

服務帳戶金鑰用於對服務帳戶進行身份驗證，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一個金鑰：

```
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一個金鑰：

```
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 **RoleBinding** 對象，將 **ClusterRole** 物件綁定到每個新的服務帳戶。

安裝Trident Protect 時會建立一個預設的 ClusterRole 物件。您可以透過建立和套用 RoleBinding 物件將此 ClusterRole 綁定到服務帳戶。

步驟

1. 將叢集角色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. 將群集角色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. 確認工程用戶無法存取行銷資源：

```

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** 物件的存取權限

若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 和金鑰組合的 YAML 文件，以授予使用者對 AppVault 的存取權限。例如，以下 CR 授予使用者對 AppVault 的存取權限 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. 建立並套用角色 CR，使叢集管理員能夠授予對命名空間中特定資源的存取權限。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. 建立並套用角色綁定 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. 請確認權限是否正確。

- 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的 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之外的任何資源，或建立他們無權存取的新資源。

監控Trident保護資源

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

kube-state-metrics 服務從 Kubernetes API 通訊產生指標。將其與Trident Protect 結合使用，可以顯示有關環境中資源狀態的有用資訊。

Prometheus 是一個工具包，它可以接收 kube-state-metrics 產生的數據，並將其呈現為關於這些物件的易於閱讀的資訊。kube-state-metrics 和 Prometheus 共同提供了一種方法，讓您可以監控使用Trident Protect 管理的資源的健康狀況和狀態。

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。

這些步驟中包含的配置和指導僅供參考；您需要根據自己的環境進行自訂。請參閱以下官方文件以取得具體說明和支援：



- ["kube-state-metrics 文檔"](#)
- ["普羅米修斯文檔"](#)
- ["Alertmanager 文檔"](#)

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 chart 建立一個設定檔（例如，metrics-config.yaml）。您可以根據自身環境自訂以下範例配置：

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. 請依照下列說明配置 kube-state-metrics，以產生 Trident Protect 使用的自訂資源的指標：["kube-state-metrics 自訂資源文檔"](#)。

安裝 Prometheus

您可以按照以下說明安裝 Prometheus：["普羅米修斯文檔"](#)。

安裝 Alertmanager

您可以按照以下說明安裝 Alertmanager：["Alertmanager 文檔"](#)。

步驟 2：配置監控工具以協同工作

安裝監控工具後，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. 配置 Prometheus 將警報路由到 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

Prometheus 現在可以從 kube-state-metrics 收集指標，並且可以向 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：設定警報和警報目標

配置好工具協同工作後，還需要配置哪些類型的信息會觸發警報，以及警報應該發送到哪裡。

警報範例：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「是」時，警報將會被觸發。`Error`持續5秒或更長時間。您可以自訂此範例以符合您的環境，並將此 YAML 程式碼片段包含在您的專案中。`prometheus.yaml`設定檔：

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 向其他管道發送警報

您可以設定 Alertmanager，使其將通知傳送到其他管道，例如電子郵件、PagerDuty、Microsoft Teams 或其他通知服務，只需在設定檔中指定相應的配置即可。`alertmanager.yaml`文件。

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，請取代以下值：`api\_url`金鑰包含您環境中使用的 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 支援包

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) 檔案將支援包上傳至NetApp支援網站 (NSS)。

使用 CR 建立支援包

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. 配置以下屬性：
 - **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:** (*Required*) 確定支援包是立即產生還是按計劃產生。計劃的資料包產生時間為世界協調時凌晨 12 點。可能的值：
 - 已安排
 - 手動的
 - **spec.uploadEnabled:** (可選) 控制產生支援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`。可能的值：
 - 真的
 - `false` (預設值)
 - **spec.dataWindowStart:** (可選) RFC 3339 格式的日期字串，指定支援包中包含的資料視窗應開始的日期和時間。如果未指定，則預設為 24 小時前。您最早可以指定的日期範圍是 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. 填寫完後 `trident-protect-support-bundle.yaml` 將檔案的值正確後，套用 CR：

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

使用 CLI 建立支援包

步驟

1. 建立支援包，將括號中的值替換為您環境中的資訊。這 `trigger-type` 決定捆綁包是立即建立還是由計劃安排決定創建時間，並且可以是 `Manual` 或者 `Scheduled`。預設是 `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` 從輸出中讀取命令並運行它，將目標參數替換為您首選的本機目錄。

升級Trident保護

您可以將Trident Protect 升級到最新版本，以享受新功能或修復錯誤。



從 24.10 版本升級時，升級期間執行的快照可能會失敗。此故障不會阻止將來建立快照，無論是手動建立還是計劃建立。如果在升級過程中快照失敗，您可以手動建立新的快照，以確保您的應用程式受到保護。

為避免潛在的故障，您可以在升級前停用所有快照計劃，並在升級後重新啟用它們。但是，這會導致在升級期間錯過任何計劃的快照。

若要升級Trident Protect，請執行下列步驟。

步驟

1. 更新Trident Helm 倉庫：

```
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. 執行此命令以將 CRD 的管理權從 `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. 執行此命令以刪除 Helm 金鑰 `trident-protect-crds` 圖表：



不要卸載 `trident-protect-crds` 使用 Helm 建立圖表可能會刪除您的 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 保護：

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

版權資訊

Copyright © 2026 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。