管理 Trident Protect

Trident

NetApp
March 12, 2025

目錄

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：

```
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：

```
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：

```
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`：

```
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：

```
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：

```
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：

   ```
   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. 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

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: astra.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: ["backups.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
```

3. 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
```

4. 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.yml`）並添加 kube 狀態指標服務信息。例如：

Prometheus.yml ：與 **Prometheus** 整合的 **Kube-state** 度量服務

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: astra-connector
data:
  prometheus.yml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.astra-connector.svc:8080']
```

2. 設定 Prometheus 將警示路由至 AlertManager 。編輯 Prometheus 配置文件(`prometheus.yml`）並添加以下部分：

Prometheus.yml ：傳送警示給 **Alertmanager**

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.astra-connector.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.yml`：

rules.yml ：定義失敗備份的 **Prometheus** 警示

```
rules.yml: |
  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.yml`。

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.yml ：傳送警示至 **Slack** 頻道

```
data:
  alertmanager.yml: |
    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 可讓系統管理員產生套件組合，其中包含 NetApp 支援所需的資訊，包括所管理叢集和應用程式的記錄，度量和拓撲資訊。如果您已連線至網際網路，則可以使用自訂資源（ CR ）檔案，將支援套件上傳至 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：

```
---
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. 在您以正確的值填入檔案之後 `astra-support-bundle.yaml` 、請套用 CR ：

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

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>
```

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。

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：

```
helm upgrade trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2502.0  --namespace trident-protect
```

3. 升級 Trident Protect ：

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2502.0 --namespace trident-protect
```