Trident Protectの管理

Trident

NetApp
January 15, 2026

目次

Trident Protectの管理

Trident Protectの認証とアクセス制御を管理する

Trident Protect は、ロールベースのアクセス制御 (RBAC) の Kubernetes モデルを使用します。デフォルトでは、 Trident Protect は単一のシステム名前空間とそれに関連付けられたデフォルトのサービス アカウントを提供します。組織内に多数のユーザーや特定のセキュリティ ニーズがある場合は、 Trident Protect の RBAC 機能を使用して、リソースや名前空間へのアクセスをより細かく制御できます。

クラスタ管理者は常にデフォルトのリソースにアクセスできます。 `trident-protect`名前空間だけでなく、他のすべての名前空間のリソースにもアクセスできます。リソースとアプリケーションへのアクセスを制御するには、追加の名前空間を作成し、それらの名前空間にリソースとアプリケーションを追加する必要があります。

デフォルトでは、どのユーザーもアプリケーションデータ管理CRを作成できないことに注意してください。`trident-protect`名前空間。アプリケーション名前空間にアプリケーション データ管理 CR を作成する必要があります (ベスト プラクティスとして、関連付けられているアプリケーションと同じ名前空間にアプリケーション データ管理 CR を作成します)。

> 特権のあるTrident Protect カスタム リソース オブジェクトには、管理者のみがアクセスできる必要があります。これには次のものが含まれます。
>
> - **AppVault**: バケット認証情報データが必要
> - **AutoSupportBundle**: メトリック、ログ、その他の機密性の高いTrident Protect データを収集します
> - **AutoSupportBundleSchedule**: ログ収集スケジュールを管理します
>
> ベスト プラクティスとして、RBAC を使用して、特権オブジェクトへのアクセスを管理者に制限します。

RBACがリソースと名前空間へのアクセスをどのように制御するかの詳細については、 "Kubernetes RBAC ドキュメント" 。

サービスアカウントの詳細については、 "Kubernetes サービス アカウントのドキュメント" 。

例: 2つのユーザーグループのアクセスを管理する

たとえば、組織にはクラスター管理者、エンジニアリング ユーザーのグループ、マーケティング ユーザーのグループがあります。クラスター管理者は、エンジニアリング グループとマーケティング グループがそれぞれの名前空間に割り当てられたリソースのみにアクセスできる環境を作成するために、次のタスクを実行します。

ステップ**1:** 各グループのリソースを格納する名前空間を作成する

名前空間を作成すると、リソースを論理的に分離し、それらのリソースにアクセスできるユーザーをより適切に制御できるようになります。

手順

1. エンジニアリング グループの名前空間を作成します。

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

2. マーケティング グループの名前空間を作成します。

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

ステップ**2:** 各名前空間内のリソースとやり取りするための新しいサービス アカウントを作成する

作成する新しい名前空間にはそれぞれデフォルトのサービス アカウントが付属しますが、将来必要に応じてグループ間で権限をさらに分割できるように、ユーザー グループごとにサービス アカウントを作成する必要があります。

手順

1. エンジニアリング グループのサービス アカウントを作成します。

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

2. マーケティング グループのサービス アカウントを作成します。

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

ステップ**3:** 新しいサービスアカウントごとにシークレットを作成する

サービス アカウント シークレットは、サービス アカウントでの認証に使用され、侵害された場合には簡単に削除して再作成できます。

手順

1. エンジニアリング サービス アカウントのシークレットを作成します。

```
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. マーケティング サービス アカウントのシークレットを作成します。

```
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
```

ステップ**4: ClusterRole**オブジェクトを各新しいサービスアカウントにバインドするための**RoleBinding**オブジェクトを作成する

Trident Protect をインストールすると、デフォルトの ClusterRole オブジェクトが作成されます。RoleBinding オブジェクトを作成して適用することで、この ClusterRole をサービス アカウントにバインドできます。

手順

1. ClusterRole をエンジニアリング サービス アカウントにバインドします。

```
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. ClusterRole をマーケティング サービス アカウントにバインドします。

```
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
```

ステップ**5:** 権限をテストする

権限が正しいことをテストします。

手順

1. エンジニアリング ユーザーがエンジニアリング リソースにアクセスできることを確認します。

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

2. エンジニアリング ユーザーがマーケティング リソースにアクセスできないことを確認します。

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

ステップ**6: AppVault**オブジェクトへのアクセスを許可する

バックアップやスナップショットなどのデータ管理タスクを実行するには、クラスター管理者が個々のユーザーに AppVault オブジェクトへのアクセスを許可する必要があります。

手順

1. ユーザーに AppVault へのアクセスを許可する AppVault とシークレットの組み合わせ YAML ファイルを作成して適用します。たとえば、次のCRはユーザーにAppVaultへのアクセスを許可します。 `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. ロール CR を作成して適用し、クラスター管理者が名前空間内の特定のリソースへのアクセスを許可できるようにします。例えば：

```
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. RoleBinding CR を作成して適用し、権限をユーザー eng-user にバインドします。例えば：

```
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. 権限が正しいことを確認してください。

   a. すべての名前空間の AppVault オブジェクト情報を取得しようとします。

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

   次のような出力が表示されます。

```
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. ユーザーが現在アクセス権限を持っている AppVault 情報を取得できるかどうかをテストします。

```
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
```

次のような出力が表示されます。

```
yes
```

結果

AppVault 権限を付与したユーザーは、アプリケーション データ管理操作のために承認された AppVault オブジェクトを使用でき、割り当てられた名前空間外のリソースにアクセスしたり、アクセス権のない新しいリソースを作成したりすることはできません。

Trident Protectリソースを監視する

kube-state-metrics、Prometheus、および Alertmanager オープンソース ツールを使用して、 Trident Protect によって保護されているリソースの健全性を監視できます。

kube-state-metrics サービスは、Kubernetes API 通信からメトリックを生成します。Trident Protect と併用すると、環境内のリソースの状態に関する有用な情報が公開されます。

Prometheus は、kube-state-metrics によって生成されたデータを取り込み、これらのオブジェクトに関する読みやすい情報として提示できるツールキットです。kube-state-metrics と Prometheus を組み合わせることで、 Trident Protect で管理しているリソースの健全性とステータスを監視する方法が提供されます。

Alertmanager は、Prometheus などのツールによって送信されたアラートを取り込み、設定した宛先にルーティングするサービスです。

> これらの手順に含まれる構成とガイダンスは単なる例であり、環境に合わせてカスタマイズする必要があります。具体的な手順とサポートについては、次の公式ドキュメントを参照してください。
>
> - "kube-state-metrics ドキュメント"
> - "Prometheusのドキュメント"
> - "Alertmanager ドキュメント"

ステップ**1:** 監視ツールをインストールする

Trident Protect でリソース監視を有効にするには、kube-state-metrics、Promethus、および Alertmanager をインストールして構成する必要があります。

kube-state-metricsをインストールする

Helm を使用して kube-state-metrics をインストールできます。

手順

1. kube-state-metrics Helm チャートを追加します。例えば：

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

2. Prometheus ServiceMonitor CRD をクラスターに適用します。

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

3. Helmチャートの設定ファイルを作成します（例： `metrics-config.yaml` ）。次の例の構成を、環境に合わせてカスタマイズできます。

metrics-config.yaml: kube-state-metrics Helm チャート設定

```
---
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. Helm チャートをデプロイして kube-state-metrics をインストールします。例えば：

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

5. kube-state-metricsを設定して、 Trident Protectが使用するカスタムリソースのメトリックを生成するには、以下の手順に従ってください。 "kube-state-metrics カスタムリソースのドキュメント" 。

Prometheusをインストールする

Prometheusは、以下の手順に従ってインストールできます。 "Prometheusのドキュメント" 。

Alertmanagerをインストールする

Alertmanagerは、以下の手順に従ってインストールできます。 "Alertmanager ドキュメント" 。

ステップ**2:** 監視ツールを連携するように設定する

監視ツールをインストールした後、それらが連携するように構成する必要があります。

手順

1. kube-state-metrics を Prometheus と統合します。 Prometheus設定ファイルを編集する(`prometheus.yaml`) をクリックし、kube-state-metrics サービス情報を追加します。例えば：

prometheus.yaml: kube-state-metrics サービスと **Prometheus** の統合

```
---
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. アラートを Alertmanager にルーティングするように Prometheus を構成します。 Prometheus設定ファイルを編集する(`prometheus.yaml`) に次のセクションを追加します。

prometheus.yaml: Alertmanagerにアラートを送信する

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

結果

Prometheus は kube-state-metrics からメトリクスを収集し、Alertmanager にアラートを送信できるようになりました。これで、アラートをトリガーする条件とアラートの送信先を構成する準備が整いました。

ステップ3: アラートとアラートの送信先を設定する

ツールが連携するように構成したら、アラートをトリガーする情報の種類と、アラートを送信する場所を構成する必要があります。

アラートの例: バックアップ失敗

次の例では、バックアップカスタムリソースのステータスがに設定されたときにトリガーされる重要なアラートを定義します。 `Error` 5秒以上。この例を環境に合わせてカスタマイズし、このYAMLスニペットを `prometheus.yaml`設定ファイル:

rules.yaml: 失敗したバックアップに対する **Prometheus** アラートを定義する

```
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."
```

Alertmanager を設定して他のチャネルにアラートを送信する

Alertmanagerは、電子メール、PagerDuty、Microsoft Teams、その他の通知サービスなどの他のチャネルに通知を送信するように設定できます。 `alertmanager.yaml`ファイル。

次の例では、Slack チャネルに通知を送信するように Alertmanager を構成します。この例を自分の環境に合わせてカスタマイズするには、 `api_url`お使いの環境で使用されている Slack Webhook URL にキーを設定します。

alertmanager.yaml: Slackチャンネルにアラートを送信する

```
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
```

Trident Protect サポートバンドルを生成する

Trident Protect を使用すると、管理者は、管理対象のクラスタとアプリケーションに関するログ、メトリック、トポロジ情報など、 NetAppサポートに役立つ情報を含むバンドルを生成できます。インターネットに接続している場合は、カスタム リソース (CR) ファイルを使用して、サポート バンドルをNetAppサポート サイト (NSS) にアップロードできます。

CRを使用してサポートバンドルを作成する

手順

1. カスタムリソース（CR）ファイルを作成し、名前を付けます（例： `trident-protect-support-bundle.yaml` ）。
2. 次の属性を構成します。
   - **metadata.name**: (必須) このカスタム リソースの名前。環境に合わせて一意かつ適切な名前を選択してください。
   - **spec.triggerType**: (必須) サポート バンドルをすぐに生成するか、スケジュールに従って生成するかを決定します。スケジュールされたバンドル生成は、UTC の午前 12 時に行われます。有効な値は次のとおりです。
     - スケジュール済み
     - 手動
   - **spec.uploadEnabled**: (オプション) サポート バンドルを生成後にNetAppサポート サイトにアップロードするかどうかを制御します。指定しない場合は、デフォルトは `false`。有効な値は次のとおりです。
     - true
     - false（デフォルト）
   - **spec.dataWindowStart**: (オプション) サポート バンドルに含まれるデータのウィンドウが開始する日時を指定する RFC 3339 形式の日付文字列。指定しない場合は、デフォルトで 24 時間前になります。指定できる最も早いウィンドウ日付は 7 日前です。

     YAMLの例:

```
---
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. 入力したら `trident-protect-support-bundle.yaml`正しい値を持つファイルには、CR を適用します。

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

CLIを使用してサポートバンドルを作成する

手順

1. 括弧内の値を環境の情報に置き換えて、サポート バンドルを作成します。その `trigger-type`バンドルがすぐに作成されるか、作成時間がスケジュールによって決定されるかを決定します。`Manual`または `Scheduled`。デフォルト設定は `Manual`。

   例えば：

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

サポートバンドルを監視して取得する

いずれかの方法を使用してサポート バンドルを作成した後、その生成の進行状況を監視し、ローカル システムに取得することができます。

手順

1. 待つ `status.generationState`到達する `Completed`州。次のコマンドで生成の進行状況を監視できます。

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

2. サポート バンドルをローカル システムに取得します。完了したAutoSupportバンドルからコピー コマンドを取得します。

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

   見つける `kubectl cp`出力からコマンドを取得して実行し、宛先引数を希望のローカル ディレクトリに置き換えます。

Tridentプロテクトのアップグレード

新しい機能やバグ修正を利用するには、 Trident Protect を最新バージョンにアップグレードできます。

> バージョン 24.10 からアップグレードすると、アップグレード中に実行されるスナップショットが失敗する可能性があります。この障害により、手動またはスケジュールされた将来のスナップショットの作成が妨げられることはありません。アップグレード中にスナップショットが失敗した場合は、アプリケーションが保護されるように手動で新しいスナップショットを作成できます。
>
> 潜在的な障害を回避するために、アップグレード前にすべてのスナップショット スケジュールを無効にして、アップグレード後に再度有効にすることができます。ただし、これにより、アップグレード期間中にスケジュールされたスナップショットが失われることになります。

Trident Protect をアップグレードするには、次の手順を実行します。

手順

1. Trident Helm リポジトリを更新します。

```
helm repo update
```

2. Trident Protect CRD をアップグレードします。

> 25.06 より前のバージョンからアップグレードする場合は、CRD がTrident Protect Helm チャートに含まれるようになったため、この手順は必須です。

a. このコマンドを実行すると、CRDの管理を `trident-protect-crds`に `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. このコマンドを実行してHelmシークレットを削除します `trident-protect-crds`チャート：

> アンインストールしないでください `trident-protect-crds`Helm を使用してチャートを作成しないでください。CRD と関連データが削除される可能性があります。

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

3. Tridentプロテクトのアップグレード:

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

著作権に関する情報

商標に関する情報