



Trident Protectの管理

Trident

NetApp
January 14, 2026

目次

| | |
|--------------------------------------|----|
| Trident Protectの管理 | 1 |
| Trident Protectの認証とアクセス制御を管理する | 1 |
| 例：2つのユーザグループのアクセスを管理する | 1 |
| Trident Protectリソースを監視する | 7 |
| 手順1：監視ツールをインストールする | 8 |
| ステップ2：監視ツールが連携するように設定する | 10 |
| 手順3：アラートとアラートの送信先を設定する | 11 |
| Trident Protect サポートバンドルを生成する | 12 |
| Tridentプロテクトのアップグレード | 14 |

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、常にデフォルトのネームスペース内のリソースにアクセスできます `trident-protect`。また、他のすべてのネームスペース内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。

デフォルトの名前空間にアプリケーションデータ管理CRSを作成することはできないことに注意して `'trident-protect'` ください。アプリケーションデータ管理CRSは、アプリケーションネームスペース内に作成する必要があります（ベストプラクティスとして、アプリケーションデータ管理CRSは、関連付けられているアプリケーションと同じネームスペースに作成します）。

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 ***：バケット資格情報データが必要です。
- *** 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。

RBACを使用して、権限付きオブジェクトへのアクセスを管理者に制限することを推奨します。

RBACでリソースおよびネームスペースへのアクセスを制御する方法の詳細については、を参照して ["Kubernetes RBACのドキュメント"](#) ください。

サービスアカウントの詳細については、を参照して ["Kubernetesサービスアカウントのドキュメント"](#) ください。

例：2つのユーザグループのアクセスを管理する

たとえば、ある組織に、クラスタ管理者、エンジニアリングユーザのグループ、およびマーケティングユーザのグループがあるとします。クラスタ管理者は次のタスクを実行して、engineeringグループとmarketingグループがそれぞれのネームスペース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します。

手順1：各グループのリソースを含むネームスペースを作成する

ネームスペースを作成すると、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をより細かく制御できます。

手順

1. engineeringグループの名前空間を作成します。

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

2. marketing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：各ネームスペース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ていますが、将来必要に応じてPrivilegesをグループ間でさらに分割できるように、ユーザーの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. engineering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、侵害された場合は簡単に削除および再作成できます。

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

```

手順4：RoleBindingオブジェクトを作成して、**ClusterRole**オブジェクトを新しい各サービスアカウントにバインドする

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
```

手順5：権限のテスト

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
```

手順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. AppVaultへのユーザー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、AppVaultへのアクセスをユーザーに許可し`eng-user`ます。

```

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. 役割CRを作成して適用し、クラスタ管理者がネームスペース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eng-userというユーザにバインドします。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ます。

a. すべての名前空間のAppVaultオブジェクト情報の取得を試みます。

```
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. ユーザがAppVault情報を取得できるかどうかをテストして、アクセス許可を得ているかどうかを確認します。

```
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に承認されたAppVaultオブジェクトを使用できる必要があります。また、割り当てられた名前空間以外のリソースにアクセスしたり、アクセスでき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などのツールから送信されたアラートを取り込み、設定した送信先にルーティングするサービスです。

これらの手順に記載されている構成とガイダンスは一例にすぎません。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 ノートブックメント"](#)
- ["AlertManagerのドキュメント"](#)

手順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、Prometheus、および Alertmanager をインストールして構成する必要があります。

インストール **kube-state-metrics**

kube-state-metricsはHelmを使用して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. Helmチャートの構成ファイルを作成します（例：metrics-config.yaml）。次の設定例は、環境に合わせてカスタマイズできます。

metrics-config.yaml : kube-state-metrics Helmチャート構成

```
---
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
```

3. Helmチャートを展開して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
```

4. 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カスタムリソースドキュメント](#)"。

Prometheus をインストールする

の手順に従ってPrometheusをインストールできます。"[Prometheusノドキュメント](#)"

AlertManagerのインストール

の手順に従って、AlertManagerをインストールできます "[AlertManagerのドキュメント](#)"。

ステップ2：監視ツールが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ら、それらが連携するように設定する必要があります。

手順

1. kube-state-metricsとPrometheusを統合Prometheus構成ファイル(`prometheus.yaml`を編集)、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を設定します。Prometheus構成ファイル(`prometheus.yaml`を編集)、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では、kube-state-metricsから指標を収集し、アラートをAlertmanagerに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設定する準備ができました。

手順3：アラートとアラートの送信先を設定する

ツールが連携して動作するように設定したら、アラートをトリガーする情報の種類とアラートの送信先を設定する必要があります。

アラートの例：バックアップの失敗

次の例は、バックアップカスタムリソースのステータスが5秒以上に設定された場合にトリガーされるCriticalアラートを定義します Error。この例を環境に合わせてカスタマイズし、このYAMLスニペットを構成ファイルに含めることができます prometheus.yaml。

rules.yaml: 失敗したバックアップに関する 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."
```

アラートを他のチャンネルに送信するようにAlertManagerを設定する

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ンネルに通知を送信するようにAlertManagerを設定するには、ファイルでそれぞれの設定を指定し `alertmanager.yaml` ます。

次の例では、Slackチャンネルに通知を送信するようにAlertManagerを設定します。この例を環境に合わせてカスタマイズするには、キーの値を環境で使用されているSlack Webhook URLに置き換え `api_url` 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を使用したサポートバンドルの作成

手順

1. カスタムリソース (CR) ファイルを作成し、という名前を付けます (例: `trident-protect-support-bundle.yaml`)。
2. 次の属性を設定します。
 - `* metadata.name*:` (*required*) このカスタムリソースの名前。環境に適した一意の適切な名前を選択します。
 - `* spec.triggerType *:` (*required*) サポートバンドルをすぐに生成するかスケジュールするかを指定します。スケジュールされたバンドル生成は12AM UTCに行われます。有効な値:
 - スケジュール済み
 - 手動
 - `* spec.uploadEnabled *:` (*_Optional_*) サポートバンドルの生成後に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、デフォルトはになります `false`。有効な値:
 - 正しい
 - `false` (デフォルト)
 - `spec.dataWindowStart:` (*Optional*) サポートバンドルに含まれるデータのウィンドウを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期間の日付は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. ファイルに正しい値を入力したら `astra-support-bundle.yaml`、CRを適用します。

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

CLIを使用したサポートバンドルの作成

手順

1. サポートバンドルを作成し、角カッコ内の値を環境からの情報に置き換えます。は `trigger-type`、バンドルをすぐに作成するか、スケジュールによって作成時間が指定されているかを決定

し、または `Scheduled` を指定できます `Manual`。デフォルト設定は `Manual`。

例えば：

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

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。

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。

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

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 100.2502.0 --namespace trident-protect
```


著作権に関する情報

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および / または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータ ソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。