管理Trident Protect

Trident

NetApp
February 02, 2026

This PDF was generated from https://docs.netapp.com/zh-cn/trident/trident-protect/manage-authorization-access-control.html on February 02, 2026. Always check docs.netapp.com for the latest.

目录

管理Trident Protect 1
管理Trident Protect 授权和访问控制 1
示例：管理两组用户的访问权限 1
监控Trident保护资源 7
第1步：安装监控工具 7
第2步：配置监控工具以协同工作 10
第3步：配置警报和警报目标 11
生成Trident Protect 支持包 12
监视和检索支持包 14
升级Trident保护 14

管理Trident Protect

管理Trident Protect 授权和访问控制

Trident Protect 使用 Kubernetes 的基于角色的访问控制 (RBAC) 模型。默认情况下，Trident Protect 提供一个系统命名空间及其关联的默认服务帐户。如果您的组织拥有众多用户或特定的安全需求，则可以使用Trident Protect 的 RBAC 功能来更精细地控制对资源和命名空间的访问。

集群管理员始终可以访问默认命名空间中的资源 `trident-protect`、也可以访问所有其他命名空间中的资源。要控制对资源和应用程序的访问、您需要创建更多的名分并将资源和应用程序添加到这些名分。

请注意、任何用户都不能在默认命名空间中创建应用程序数据管理CRS `trident-protect`。您需要在应用程序命名空间中创建应用程序数据管理CRS (最佳做法是、在与其关联的应用程序相同的命名空间中创建应用程序数据管理CRS)。

> 只有管理员才能访问具有特权的Trident Protect 自定义资源对象，其中包括：
>
> - **AppVault**：需要存储分段凭据数据
> - **AutoSupportBundle**：收集指标、日志和其他敏感的Trident Protect数据
> - *AutoSupportBundleSchedule：管理日志收集计划
>
> 作为最佳实践、请使用RBAC限制管理员对有权限的对象的访问。

有关RBAC如何控制对资源和称表的访问的详细信息，请参阅 "Kubbernetes RBAC文档"。

有关服务帐户的信息，请参见 "Kubbernetes服务帐户文档"。

示例：管理两组用户的访问权限

例如、一个组织有一个集群管理员、一组工程用户和一组营销用户。集群管理员应完成以下任务、以创建一个环境、在此环境中、工程组和营销组各自只能访问分配给各自命名区域的资源。

第**1**步：创建一个命名空间以包含每个组的资源

通过创建命名空间、您可以从逻辑上分离资源、并更好地控制谁有权访问这些资源。

步骤

1. 为工程组创建命名空间：

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

2. 为营销组创建命名空间：

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

第**2**步：创建新的服务帐户、以便与每个命名空间中的资源进行交互

您创建的每个新命名空间都会附带一个默认服务帐户、但您应为每组用户创建一个服务帐户、以便将来根据需要在各个组之间进一步划分Privileges。

步骤

1. 为工程组创建服务帐户：

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

2. 为营销组创建服务帐户：

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

第**3**步：为每个新服务帐户创建一个密钥

服务帐户密钥用于向服务帐户进行身份验证、如果泄露、可以轻松删除和重新创建。

步骤

1. 为工程服务帐户创建一个密钥：

```yaml
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. 为营销服务帐户创建密钥：

```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
```

第**4**步：创建**RoleBinding**对象以将**ClusterRole**对象绑定到每个新服务帐户

安装Trident Protect 时会创建一个默认的 ClusterRole 对象。您可以通过创建和应用 RoleBinding 对象将此 ClusterRole 绑定到服务帐户。

步骤

1. 将ClusterRole绑定到工程服务帐户：

```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. 将ClusterRole绑定到营销服务帐户：

```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
```

第**5**步：测试权限

测试权限是否正确。

步骤

1. 确认工程用户可以访问工程资源：

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

2. 确认工程用户无法访问营销资源：

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

第**6**步：授予对**AppVault**对象的访问权限

要执行备份和快照等数据管理任务、集群管理员需要向各个用户授予对AppVault对象的访问权限。

步骤

1. 创建并应用AppVault和机密组合YAML文件、以授予用户对AppVault的访问权限。例如、以下CR将授予用户对AppVault的访问权限 `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. 创建并应用角色CR、使集群管理员能够授予对命名空间中特定资源的访问权限。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. 创建并应用RoleBinding CR以将权限绑定到用户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. 验证权限是否正确。

   a. 尝试检索所有名称库的AppVault对象信息：

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

   您应看到类似于以下内容的输出：

```
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. 测试用户是否可以获取他们现在有权访问的AppVault信息：

```
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
```

您应看到类似于以下内容的输出：

```
yes
```

结果

您为其授予了AppVault权限的用户应该能够使用授权的AppVault对象执行应用程序数据管理操作、并且不能访问分配的命名区之外的任何资源、也不能创建他们无权访问的新资源。

监控Trident保护资源

您可以使用 kube-state-metrics、Prometheus 和 Alertmanager 开源工具来监控Trident Protect 保护的资源的健康状况。

kube-state-metrics 服务从 Kubernetes API 通信生成指标。将其与Trident Protect 结合使用，可以显示有关环境中资源状态的有用信息。

Prometheus 是一个工具包，它可以接收 kube-state-metrics 生成的数据，并将其呈现为关于这些对象的易于阅读的信息。kube-state-metrics 和 Prometheus 共同提供了一种方法，让您可以监控使用Trident Protect 管理的资源的健康状况和状态。

警报管理器是一项服务、可接收Prometheus等工具发送的警报、并将其路由到您配置的目标。

> 这些步骤中包含的配置和指导仅为示例；您需要对其进行自定义以符合您的环境。有关具体说明和支持、请参见以下官方文档：
>
> - "Kube-state-metrics 文档"
> - "Prometheus文档"
> - "记录"

第1步：安装监控工具

要在Trident Protect 中启用资源监控，您需要安装和配置 kube-state-metrics、Promethus 和 Alertmanager。

安装Kube-state-metrics

您可以使用Helm安装Kube-state-metrics。

步骤

1. 添加Kube-state-metrics Helm图表。例如：

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

2. 将 Prometheus ServiceMonitor CRD 应用到集群：

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

3. 为Helm图表创建配置文件(例如 `metrics-config.yaml`)。您可以根据您的环境自定义以下示例配置：

metrics-config.yaml：Kube-state-metrics Helm图表配置

```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. 通过部署Helm图表来安装Kube-state-metrics。例如：

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

5. 按照以下说明配置 kube-state-metrics，以生成Trident Protect 使用的自定义资源的指标： "Kube-state-metrics自定义资源文档" 。

安装 **Prometheus**

您可以按照中的说明安装Prometheus "Prometheus文档"。

安装活动管理器

您可以按照中的说明安装提示管理器 "记录"。

第2步：配置监控工具以协同工作

安装监控工具后、您需要将其配置为协同工作。

步骤

1. 将Kube-state-metrics与Prometheus集成。编辑Prometheus配置文件(`prometheus.yaml`)并添加Kube-state-metrics服务信息。例如：

   prometheus.yaml：kube-state-metrics 服务与 **Prometheus** 的集成

```yaml
---
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以将警报路由到警报管理器。编辑Prometheus配置文件(`prometheus.yaml`)并添加以下部分：

prometheus.yaml：向 **Alertmanager** 发送警报

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

结果

Prometheus现在可以从Kube-state-metrics收集指标、并可向警报管理器发送警报。现在、您可以配置触发警报的条件以及警报的发送位置。

第3步：配置警报和警报目标

将这些工具配置为协同工作后、您需要配置触发警报的信息类型以及警报的发送位置。

警报示例：备份失败

以下示例定义了备份自定义资源的状态设置为5秒或更长时间时触发的严重警报 `Error`。您可以自定义此示例以匹配您的环境、并将此YAML段包含在您的配置文件中 `prometheus.yaml`：

rules.yaml：定义失败备份的 **Prometheus** 警报

```
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."
```

将警报管理器配置为向其他通道发送警报

通过在文件中指定相应的配置、您可以将警报管理器配置为向其他通道发送通知、例如电子邮件、PagerDty、Microsoft团队或其他通知服务 `alertmanager.yaml`。

以下示例将配置警报管理器、以便向SI延 时信道发送通知。要根据您的环境自定义此示例、请将此密钥的值替换 `api_url`为您的环境中使用的Slackwebhook URL：

alertmanager.yaml：向 Slack 频道发送警报

```
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
```

生成Trident Protect 支持包

Trident Protect 使管理员能够生成包含对NetApp支持有用的信息的捆绑包，包括有关受管理集群和应用程序的日志、指标和拓扑信息。如果您已连接到互联网，则可以使用自定义资源 (CR) 文件将支持包上传到NetApp支持站点 (NSS)。

使用**CR**创建支持包

步骤

1. 创建自定义资源(CR)文件并将其命名(例如 `trident-protect-support-bundle.yaml`)。
2. 配置以下属性：
   - 。**metadata.name**:(*required_*)此自定义资源的名称；请为您的环境选择一个唯一且合理的名称。
   - **spec.t**触发 器**Type**：(*required*)用于确定是立即生成支持包、还是按计划生成支持包。计划在UTC时间中午12点生成捆绑包。可能值：
     - 已计划
     - 手动
   - **spec.u**倍 加载已启用：(可 选)控制是否应在生成支持包后将其上传到NetApp支持站点。如果未指定，则默认为 `false`。可能值：
     - true
     - false (默认)
   - **spec.dataWindowStart**：(可 选) RFC 3339格式的日期字符串，指定支持包中包含的数据窗口应开始的日期和时间。如果未指定、则默认为24小时前。您可以指定的最早窗口日期是7天前。

     YAML示例：

```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. 填充后 `trident-protect-support-bundle.yaml`具有正确值的文件，应用 CR：

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

使用命令行界面创建支持包

步骤

1. 创建支持包、将括号中的值替换为您环境中的信息。 `trigger-type`确定是否立即创建分发包，或者是否由计划决定了创建时间，可以是 `Manual`或 `Scheduled`。默认设置为 `Manual`。

   例如：

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

监视和检索支持包

使用任一方法创建支持包后，您可以监视其生成进度并将其检索到本地系统。

步骤

1. 等待 `status.generationState`到达 `Completed`状态。您可以使用以下命令监控生成进度：

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

2. 将支持包检索到您的本地系统。从已完成的AutoSupport包中获取复制命令：

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

找到 `kubectl cp`从输出中执行命令并运行它，用您喜欢的本地目录替换目标参数。

升级Trident保护

您可以将Trident Protect 升级到最新版本，以享受新功能或修复错误。

- 从版本 24.10 升级时，升级期间运行的快照可能会失败。此失败不会阻止将来创建快照（无论是手动快照还是计划快照）。如果升级期间快照失败，您可以手动创建新快照以确保应用程序受到保护。

  为避免潜在的故障，您可以在升级前禁用所有快照计划，然后在升级后重新启用。但是，这会导致升级期间丢失所有计划的快照。

- 对于私有镜像仓库安装，请确保目标版本所需的 Helm Chart 和镜像在您的私有镜像仓库中可用，并验证您的自定义 Helm 值与新 Chart 版本兼容。更多信息，请参阅"从私有注册表安装Trident Protect"。

要升级Trident Protect，请执行以下步骤。

步骤

1. 更新Trident Helm存储库：

```
helm repo update
```

2. 升级Trident Protect CRD：

> 如果您是从 25.06 之前的版本升级，则需要执行此步骤，因为 CRD 现在已包含在Trident Protect Helm 图表中。

a. 运行此命令将 CRD 的管理从 `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. 运行此命令删除 `trident-protect-crds`图表：

> 不要卸载 `trident-protect-crds`图表使用 Helm，因为这可能会删除您的 CRD 和任何相关数据。

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

3. 升级Trident保护：

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

> 您可以通过添加以下内容来配置升级期间的日志级别。 `--set logLevel=debug`升级命令。默认日志级别为 `warn`。建议启用调试日志记录进行故障排除，因为它可以帮助NetApp支持人员诊断问题，而无需更改日志级别或重现问题。

版权信息

版权所有 © 2026 NetApp, Inc.。保留所有权利。中国印刷。未经版权所有者事先书面许可，本文档中受版权保护的任何部分不得以任何形式或通过任何手段（图片、电子或机械方式，包括影印、录音、录像或存储在电子检索系统中）进行复制。

从受版权保护的 NetApp 资料派生的软件受以下许可和免责声明的约束：

本软件由 NetApp 按“原样”提供，不含任何明示或暗示担保，包括但不限于适销性以及针对特定用途的适用性的隐含担保，特此声明不承担任何责任。在任何情况下，对于因使用本软件而以任何方式造成的任何直接性、间接性、偶然性、特殊性、惩罚性或后果性损失（包括但不限于购买替代商品或服务；使用、数据或利润方面的损失；或者业务中断），无论原因如何以及基于何种责任理论，无论出于合同、严格责任或侵权行为（包括疏忽或其他行为），NetApp 均不承担责任，即使已被告知存在上述损失的可能性。

NetApp 保留在不另行通知的情况下随时对本文档所述的任何产品进行更改的权利。除非 NetApp 以书面形式明确同意，否则 NetApp 不承担因使用本文档所述产品而产生的任何责任或义务。使用或购买本产品不表示获得 NetApp 的任何专利权、商标权或任何其他知识产权许可。

本手册中描述的产品可能受一项或多项美国专利、外国专利或正在申请的专利的保护。

有限权利说明：政府使用、复制或公开本文档受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中“技术数据权利 — 非商用”条款第 (b)(3) 条规定的限制条件的约束。

本文档中所含数据与商业产品和/或商业服务（定义见 FAR 2.101）相关，属于 NetApp, Inc. 的专有信息。根据本协议提供的所有 NetApp 技术数据和计算机软件具有商业性质，并完全由私人出资开发。 美国政府对这些数据的使用权具有非排他性、全球性、受限且不可撤销的许可，该许可既不可转让，也不可再许可，但仅限在与交付数据所依据的美国政府合同有关且受合同支持的情况下使用。除本文档规定的情形外，未经 NetApp, Inc. 事先书面批准，不得使用、披露、复制、修改、操作或显示这些数据。美国政府对国防部的授权仅限于 DFARS 的第 252.227-7015(b)（2014 年 2 月）条款中明确的权利。

商标信息

NetApp、NetApp 标识和 http://www.netapp.com/TM 上所列的商标是 NetApp, Inc. 的商标。其他公司和产品名称可能是其各自所有者的商标。