Gerenciar Trident Protect

Trident

NetApp
July 01, 2026

Índice

Gerenciar Trident Protect

Gerencie a autorização e o controle de acesso do Trident

Trident Protect utiliza o modelo Kubernetes de controle de acesso baseado em funções (RBAC). Por padrão, Trident Protect fornece um único namespace de sistema e sua respectiva conta de serviço padrão. Se você tem uma organização com muitos usuários ou necessidades de segurança específicas, pode usar os recursos de RBAC do Trident Protect para obter um controle mais granular sobre o acesso a recursos e namespaces.

O administrador do cluster sempre tem acesso aos recursos no namespace padrão `trident-protect` e também pode acessar recursos em todos os outros namespaces. Para controlar o acesso a recursos e aplicativos, você precisa criar namespaces adicionais e adicionar recursos e aplicativos a esses namespaces.

Observe que nenhum usuário pode criar CRs de gerenciamento de dados de aplicativos no namespace `trident-protect`padrão. Você precisa criar CRs de gerenciamento de dados de aplicativos em um namespace de aplicativo (como prática recomendada, crie CRs de gerenciamento de dados de aplicativos no mesmo namespace do aplicativo associado).

> Somente os administradores devem ter acesso aos objetos de recursos personalizados privilegiados do Trident Protect, que incluem:
>
> - **AppVault**: requer dados de credencial do bucket
> - **AutoSupportBundle**: coleta métricas, registros e outros dados confidenciais do Trident Protect
> - **AutoSupportBundleSchedule**: gerencia os agendamentos de coleta de logs
>
> Como prática recomendada, use RBAC para restringir o acesso a objetos privilegiados a administradores.

Para obter mais informações sobre como o RBAC regula o acesso a recursos e namespaces, consulte o "Documentação do Kubernetes RBAC".

Para obter informações sobre contas de serviço, consulte o "Documentação da conta de serviço do Kubernetes".

Exemplo: gerenciar o acesso para dois grupos de usuários

Por exemplo, uma organização possui um administrador de cluster, um grupo de usuários de engenharia e um grupo de usuários de marketing. O administrador de cluster executaria as seguintes tarefas para criar um ambiente onde o grupo de engenharia e o grupo de marketing tenham acesso apenas aos recursos atribuídos aos seus respectivos namespaces.

Etapa 1: crie um espaço de nomes para conter os recursos de cada grupo

Criar um namespace permite separar recursos logicamente e controlar melhor quem tem acesso a esses recursos.

Passos

1. Crie um namespace para o grupo de engenharia:

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

2. Crie um namespace para o grupo de marketing:

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

Etapa 2: crie novas contas de serviço para interagir com os recursos em cada namespace

Cada novo namespace que você cria vem com uma conta de serviço padrão, mas você deve criar uma conta de serviço para cada grupo de usuários para que possa dividir ainda mais os privilégios entre grupos no futuro, se necessário.

Passos

1. Crie uma conta de serviço para o grupo de engenharia:

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

2. Crie uma conta de serviço para o grupo de marketing:

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

Etapa 3: crie um segredo para cada nova conta de serviço

Uma chave secreta da conta de serviço é usada para autenticar com a conta de serviço e pode ser facilmente excluída e recriada se for comprometida.

Passos

1. Crie um segredo para a conta de serviço de engenharia:

```
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. Crie um segredo para a conta do serviço de marketing:

```
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
```

Etapa 4: Crie um objeto RoleBinding para vincular o objeto ClusterRole a cada nova conta de serviço

Um objeto ClusterRole padrão é criado quando você instala Trident Protect. Você pode vincular esse ClusterRole à conta de serviço criando e aplicando um objeto RoleBinding.

Passos

1. Vincule o ClusterRole à conta de serviço de engenharia:

```
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. Vincule o ClusterRole à conta de serviço de marketing:

```
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
```

Etapa 5: testar permissões

Teste se as permissões estão corretas.

Passos

1. Confirme se os usuários de engenharia podem acessar os recursos de engenharia:

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

2. Confirme que os usuários de engenharia não podem acessar os recursos de marketing:

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

Etapa 6: conceder acesso a objetos AppVault

Para executar tarefas de gerenciamento de dados, como backups e snapshots, o administrador do cluster precisa conceder acesso a objetos AppVault a usuários individuais.

Passos

1. Crie e aplique um arquivo YAML de combinação de AppVault e segredo que conceda a um usuário acesso a um AppVault. Por exemplo, o seguinte CR concede acesso a um AppVault ao usuário `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. Crie e aplique uma Role CR para permitir que os administradores do cluster concedam acesso a recursos específicos em um namespace. Por exemplo:

```
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. Crie e aplique uma RoleBinding CR para vincular as permissões ao usuário eng-user. Por exemplo:

```
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. Verifique se as permissões estão corretas.

   a. Tentativa de recuperar informações do objeto AppVault para todos os namespaces:

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

   Você deverá ver uma saída semelhante à seguinte:

```
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. Teste para verificar se o usuário consegue obter as AppVault informações às quais agora tem permissão de acesso:

```
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
```

Você deverá ver uma saída semelhante à seguinte:

```
yes
```

Resultado

Os usuários aos quais você concedeu permissões do AppVault devem poder usar objetos AppVault autorizados para operações de gerenciamento de dados do aplicativo e não devem poder acessar recursos fora dos namespaces atribuídos nem criar novos recursos aos quais não tenham acesso.

Monitorar recursos do Trident Protect

Você pode usar as ferramentas de código aberto kube-state-metrics, Prometheus e Alertmanager para monitorar a integridade dos recursos protegidos pelo Trident Protect.

O serviço kube-state-metrics gera métricas a partir da comunicação da API do Kubernetes. Usá-lo com Trident Protect expõe informações úteis sobre o estado dos recursos em seu ambiente.

Prometheus é um conjunto de ferramentas que pode ingerir os dados gerados pelo kube-state-metrics e apresentá-los como informações facilmente legíveis sobre esses objetos. Juntos, kube-state-metrics e Prometheus fornecem uma maneira de monitorar a integridade e o status dos recursos que você gerencia com Trident Protect.

Alertmanager é um serviço que recebe os alertas enviados por ferramentas como Prometheus e os encaminha para destinos que você configurar.

> As configurações e orientações incluídas nestas etapas são apenas exemplos; você precisa personalizá-las para corresponder ao seu ambiente. Consulte a seguinte documentação oficial para instruções específicas e suporte:
>
> - "documentação do kube-state-metrics"
> - "Documentação do Prometheus"
> - "Documentação do Alertmanager"

Passo 1: instale as ferramentas de monitoramento

Para habilitar o monitoramento de recursos no Trident Protect, você precisa instalar e configurar kube-state-metrics, Prometheus e Alertmanager.

Instalar kube-state-metrics

Você pode instalar kube-state-metrics usando Helm.

Passos

1. Adicione o gráfico Helm kube-state-metrics. Por exemplo:

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

2. Aplique o CRD ServiceMonitor do Prometheus ao cluster:

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

3. Crie um arquivo de configuração para o Helm chart (por exemplo, `metrics-config.yaml`). Você pode personalizar a seguinte configuração de exemplo para corresponder ao seu ambiente:

metrics-config.yaml: configuração do Helm chart kube-state-metrics

```
---
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. Instale o kube-state-metrics implantando o gráfico Helm. Por exemplo:

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

5. Configure o kube-state-metrics para gerar métricas para os recursos personalizados usados pelo Trident Protect seguindo as instruções em "Documentação do recurso personalizado kube-state-metrics".

Instale o Prometheus

Você pode instalar o Prometheus seguindo as instruções no "Documentação do Prometheus".

Instale o Alertmanager

Você pode instalar o Alertmanager seguindo as instruções no "Documentação do Alertmanager".

Etapa 2: configure as ferramentas de monitoramento para funcionarem em conjunto

Após instalar as ferramentas de monitoramento, você precisa configurá-las para que funcionem juntas.

Passos

1. Integre o kube-state-metrics com o Prometheus. Edite o arquivo de configuração do Prometheus (`prometheus.yaml` e adicione as informações do serviço kube-state-metrics. Por exemplo:

   prometheus.yaml: integração do serviço kube-state-metrics com Prometheus

```
---
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. Configure o Prometheus para encaminhar alertas para o Alertmanager. Edite o arquivo de configuração do Prometheus (`prometheus.yaml` e adicione a seguinte seção:

prometheus.yaml: enviar alertas para o Alertmanager

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

Resultado

Prometheus agora pode coletar métricas do kube-state-metrics e enviar alertas para Alertmanager. Agora você está pronto para configurar quais condições acionam um alerta e para onde os alertas devem ser enviados.

Etapa 3: configurar alertas e destinos de alerta

Depois de configurar as ferramentas para funcionarem em conjunto, você precisa configurar que tipo de informação aciona alertas e para onde os alertas devem ser enviados.

Exemplo de alerta: falha no backup

O exemplo a seguir define um alerta crítico que é acionado quando o status do recurso personalizado de backup é definido como `Error` por 5 segundos ou mais. Você pode personalizar este exemplo para corresponder ao seu ambiente e incluir este trecho de YAML em seu arquivo de configuração `prometheus.yaml`:

rules.yaml: defina um alerta do Prometheus para backups com falha

```
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."
```

Configure o Alertmanager para enviar alertas para outros canais

Você pode configurar o Alertmanager para enviar notificações para outros canais, como e-mail, PagerDuty, Microsoft Teams ou outros serviços de notificação, especificando a respectiva configuração no arquivo `alertmanager.yaml`.

O exemplo a seguir configura o Alertmanager para enviar notificações a um canal do Slack. Para personalizar este exemplo para o seu ambiente, substitua o valor da `api_url` chave pela URL do webhook do Slack

usada em seu ambiente:

alertmanager.yaml: enviar alertas para um canal do Slack

```
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
```

Gerar um pacote de suporte Trident Protect

Trident Protect permite que os administradores gerem pacotes que incluem informações úteis para o NetApp Support, incluindo logs, métricas e informações de topologia sobre os clusters e aplicativos sob gerenciamento. Se você estiver conectado à internet, poderá enviar pacotes de suporte para o site de suporte da NetApp (NSS) usando um arquivo de recurso personalizado (CR).

Crie um pacote de suporte usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-support-bundle.yaml`).
2. Configurar os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.triggerType**: (*Obrigatório*) Determina se o pacote de suporte é gerado imediatamente ou agendado. A geração agendada do pacote ocorre às 00h UTC. Valores possíveis:
     - Agendado
     - Manual
   - **spec.uploadEnabled**: (*Opcional*) Controla se o pacote de suporte deve ser carregado no site de suporte da NetApp após ser gerado. Se não for especificado, o padrão é `false`. Valores possíveis:
     - verdadeiro
     - falso (default)
   - **spec.dataWindowStart**: (*Opcional*) Uma string de data no formato RFC 3339 que especifica a data e hora em que a janela de dados incluída no pacote de suporte deve começar. Se não for especificado, o padrão é 24 horas atrás. A data mais antiga da janela que você pode especificar é 7 dias atrás.

     Exemplo 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. Após preencher o arquivo `trident-protect-support-bundle.yaml` com os valores corretos, aplique a CR:

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

Crie um pacote de suporte usando a CLI

Passos

1. Crie o pacote de suporte, substituindo os valores entre colchetes pelas informações do seu ambiente.

O `trigger-type` determina se o pacote será criado imediatamente ou se o horário de criação será definido pelo agendamento, e pode ser `Manual` ou `Scheduled`. A configuração padrão é `Manual`.

Por exemplo:

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

Monitore e recupere o pacote de suporte

Após criar um pacote de suporte usando qualquer um dos métodos, você pode monitorar o progresso da geração e recuperá-lo para seu sistema local.

Passos

1. Aguarde até que o `status.generationState` atinja o estado `Completed`. Você pode monitorar o progresso da geração com o seguinte comando:

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

2. Recupere o pacote de suporte para o seu sistema local. Obtenha o comando de cópia do pacote AutoSupport concluído:

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

   Localize o comando `kubectl cp` na saída e execute-o, substituindo o argumento de destino pelo diretório local de sua preferência.

Atualizar Trident Protect

Você pode atualizar Trident Protect para a versão mais recente para aproveitar novos recursos ou correções de bugs.

- Ao atualizar da versão 24.10, snapshots em execução durante a atualização podem falhar. Essa falha não impede que snapshots futuros, sejam manuais ou agendados, sejam criados. Se um snapshot falhar durante a atualização, você pode criar manualmente um novo snapshot para garantir que seu aplicativo esteja protegido.

  Para evitar possíveis falhas, você pode desativar todos os agendamentos de snapshots antes da atualização e reativá-los posteriormente. No entanto, isso resulta na perda de quaisquer snapshots agendados durante o período de atualização.

- Para instalações em registros privados, certifique-se de que o Helm chart e as imagens necessárias para a versão de destino estejam disponíveis em seu registro privado e verifique se seus valores Helm personalizados são compatíveis com a nova versão do chart. Para obter mais informações, consulte "Instale Trident Protect a partir de um registro privado".

Passo 1: selecione uma versão

As versões do Trident Protect seguem uma convenção de nomenclatura baseada em datas `YY.MM`, onde "YY" são os dois últimos dígitos do ano e "MM" é o mês. As versões com ponto seguem uma `YY.MM.X` convenção, onde "X" é o nível de patch. Você selecionará a versão para atualizar com base na versão da qual está atualizando.

- Você pode realizar uma atualização direta para qualquer versão de destino que esteja dentro de um intervalo de quatro versões da sua versão instalada. Por exemplo, você pode atualizar diretamente de 24.10 (ou qualquer versão secundária de 24.10) para 25.10.
- Se você estiver atualizando de uma versão fora do período de quatro versões, execute uma atualização em várias etapas. Use as instruções de atualização para a "versão anterior" da qual você está atualizando para atualizar para a versão mais recente que se encaixe no período de quatro versões. Por exemplo, se você estiver executando a versão 24.10 e quiser atualizar para a versão 26.02:
  a. Primeira atualização da 24.10 para a 25.02.
  b. Em seguida, atualize de 25.02 para 26.02.

Etapa 2: atualize Trident Protect

Para atualizar Trident Protect, execute as seguintes etapas.

Passos

1. Atualize o repositório do Trident:

```
helm repo update
```

2. Atualize os CRDs do Trident Protect:

   Esta etapa é necessária se você estiver atualizando de uma versão anterior à 25.06, pois os CRDs agora estão incluídos no Helm chart do Trident Protect.

   a. Execute este comando para transferir o gerenciamento de CRDs de `trident-protect-crds` para `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. Execute este comando para excluir o segredo do Helm para o `trident-protect-crds` chart:

> Não desinstale o `trident-protect-crds` chart usando o Helm, pois isso pode remover seus CRDs e quaisquer dados relacionados.

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

3. Atualizar Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2602.0 --namespace trident-protect
```

> Você pode configurar o nível de registro durante a atualização adicionando `--set logLevel=debug` ao comando de atualização. O nível de registro padrão é `warn`. O registro de depuração é recomendado para solução de problemas, pois ajuda a equipe de suporte da NetApp a diagnosticar problemas sem exigir alterações no nível de registro ou reprodução do problema.

Informações sobre direitos autorais

Informações sobre marcas comerciais