Gerenciar o Trident Protect

Trident

NetApp
September 26, 2025

Índice

Gerenciar o Trident Protect

Gerenciar a autorização e o controle de acesso do Trident Protect

O Trident Protect usa o modelo Kubernetes de controle de acesso baseado em funções (RBAC). Por padrão, o Trident Protect fornece um único namespace de sistema e sua conta de serviço padrão associada. Se você tiver uma organização com muitos usuários ou necessidades de segurança específicas, use os recursos RBAC do Trident Protect para obter controle mais granular sobre o acesso a recursos e espaços de nomes.

O administrador do cluster sempre tem acesso a recursos no namespace padrão `trident-protect` e também pode acessar recursos em todos os outros namespaces. Para controlar o acesso a recursos e aplicações, é necessário criar espaços de nomes adicionais e adicionar recursos e aplicações a esses espaços de nomes.

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

> Somente os administradores devem ter acesso a objetos de recursos personalizados privilegiados do Trident Protect, que incluem:
>
> - **AppVault**: Requer dados de credenciais de bucket
> - **AutoSupportBundle**: Coleta métricas, logs e outros dados confidenciais do Trident Protect
> - **AutoSupportBundleSchedule**: Gerencia os horários de coleta de Registros
>
> Como prática recomendada, use o RBAC para restringir o acesso a objetos privilegiados aos 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: Gerencie o acesso para dois grupos de usuários

Por exemplo, uma organização tem um administrador de cluster, um grupo de usuários de engenharia e um grupo de usuários de marketing. O administrador do cluster concluiria as seguintes tarefas para criar um ambiente onde o grupo de engenharia e o grupo de marketing tenham acesso apenas aos recursos atribuídos aos respetivos namespaces.

Etapa 1: Crie um namespace para conter recursos para 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 recursos em cada namespace

Cada novo namespace que você criar vem com uma conta de serviço padrão, mas você deve criar uma conta de serviço para cada grupo de usuários para que você possa dividir ainda mais Privileges 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
```

Passo 3: Crie um segredo para cada nova conta de serviço

Um segredo de conta de serviço é usado para autenticar com a conta de serviço e pode ser facilmente excluído e recriado se comprometido.

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
```

Passo 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 o 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 do 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
```

Passo 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 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 secreta e AppVault que concede a um usuário acesso a um AppVault. Por exemplo, o CR a seguir 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 um CR de função para permitir que os administradores de 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. Criar e aplicar um 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. Tente recuperar informações de objeto AppVault para todos os namespaces:

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

   Você deve ver saída semelhante ao 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 ver se o usuário pode obter as informações do AppVault que ele agora tem permissão para acessar:

```
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 deve ver saída semelhante ao seguinte:

```
yes
```

Resultado

Os usuários aos quais você concedeu permissões AppVault devem poder usar objetos AppVault autorizados para operações de gerenciamento de dados de aplicativos e não devem poder acessar recursos fora dos namespaces atribuídos ou criar novos recursos aos quais eles não têm acesso.

Gerar um pacote de suporte Trident Protect

O Trident Protect permite que os administradores gerem pacotes que incluem informações úteis ao suporte da NetApp, incluindo logs, métricas e informações de topologia sobre os clusters e aplicativos em gerenciamento. Se você estiver conetado à Internet, poderá fazer upload de 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 nomeie-o (por exemplo, `trident-protect-support-bundle.yaml`).
2. Configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.triggerType**: (*required*) determina se o pacote de suporte é gerado imediatamente ou programado. A geração de pacotes programados acontece às 12AM UTC. Valores possíveis:
     - Programado
     - Manual
   - **Spec.uploadEnabled**: (*Optional*) controla se o pacote de suporte deve ser carregado para o site de suporte da NetApp depois que ele é gerado. Se não for especificado, o padrão é `false`. Valores possíveis:
     - verdadeiro
     - falso (padrão)
   - **Spec.dataWindowStart**: (*Optional*) Uma cadeia de carateres de data no formato RFC 3339 que especifica a data e a hora em que a janela de dados incluídos no pacote de suporte deve começar. Se não for especificado, o padrão é 24 horas atrás. A data da janela mais antiga que você pode especificar é de 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. Depois de preencher o `astra-support-bundle.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie um pacote de suporte usando a CLI

Passos

1. Crie o pacote de suporte, substituindo valores entre parênteses por informações do seu ambiente. O `trigger-type` determina se o pacote é criado imediatamente ou se o tempo de criação é ditado

pelo agendamento e pode ser `Manual` ou `Scheduled`. A predefinição é `Manual`.

Por exemplo:

```
tridentctl-protect create autosupportbundle <my_bundle_name>
--trigger-type <trigger_type>
```

Atualizar o Trident Protect

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

Para atualizar o Trident Protect, execute as etapas a seguir.

Passos

1. Atualize o repositório Helm do Trident:

```
helm repo update
```

2. Atualize os CRDs do Trident Protect:

```
helm upgrade trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1  --namespace trident-protect
```

3. Atualize o Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2410.1 --namespace trident-protect
```

Informações sobre direitos autorais

Informações sobre marcas comerciais