Trident Protect를 관리합니다

Trident

NetApp
September 26, 2025

목차

Trident Protect를 관리합니다

Trident 보호 인증 및 액세스 제어를 관리합니다

Trident Protect는 역할 기반 액세스 제어(RBAC)의 Kubernetes 모델을 사용합니다. 기본적으로 Trident Protect는 단일 시스템 네임스페이스 및 연결된 기본 서비스 계정을 제공합니다. 조직에 많은 사용자 또는 특정 보안 요구 사항이 있는 경우 Trident Protect의 RBAC 기능을 사용하여 리소스 및 네임스페이스에 대한 액세스를 보다 세부적으로 제어할 수 있습니다.

클러스터 관리자는 항상 기본 네임스페이스의 리소스에 액세스할 수 `trident-protect` 있으며 다른 모든 네임스페이스의 리소스에 액세스할 수도 있습니다. 리소스 및 응용 프로그램에 대한 액세스를 제어하려면 추가 네임스페이스를 만들고 해당 네임스페이스에 리소스 및 응용 프로그램을 추가해야 합니다.

사용자는 기본 네임스페이스에 응용 프로그램 데이터 관리 CRS를 만들 수 `trident-protect` 없습니다. 응용 프로그램 네임스페이스에 응용 프로그램 데이터 관리 CRS를 만들어야 합니다(모범 사례로서 연결된 응용 프로그램과 동일한 네임스페이스에 응용 프로그램 데이터 관리 CRS를 만듭니다).

> 관리자만 권한이 있는 Trident Protect 사용자 지정 리소스 객체에 액세스할 수 있어야 합니다. 여기에는 다음이 포함됩니다.
>
> - * AppVault *: 버킷 자격 증명 데이터가 필요합니다
> - * AutoSupport 번들 *: 메트릭, 로그 및 기타 민감한 Trident 보호 데이터를 수집합니다
> - * AutoSupport기술에 따라 로그 수집 일정을 관리할 수 있습니다
>
> 가장 좋은 방법은 RBAC를 사용하여 권한이 있는 객체에 대한 액세스를 관리자에게 제한하는 것입니다.

RBAC가 리소스 및 네임스페이스에 대한 액세스를 규제하는 방법에 대한 자세한 내용은 를 "Kubernetes RBAC 설명서"참조하십시오.

서비스 계정에 대한 자세한 내용은 을 "Kubernetes 서비스 계정 설명서"참조하십시오.

예: 두 사용자 그룹에 대한 액세스를 관리합니다

예를 들어, 조직에는 클러스터 관리자, 엔지니어링 사용자 그룹 및 마케팅 사용자 그룹이 있습니다. 클러스터 관리자는 엔지니어링 그룹과 마케팅 그룹이 각각 해당 네임스페이스에 할당된 리소스에만 액세스할 수 있는 환경을 만들기 위해 다음 작업을 완료합니다.

1단계**:** 각 그룹의 리소스를 포함할 네임스페이스를 만듭니다

네임스페이스를 만들면 리소스를 논리적으로 분리하고 해당 리소스에 액세스할 수 있는 사용자를 보다 효율적으로 제어할 수 있습니다.

단계

1. 엔지니어링 그룹의 네임스페이스를 만듭니다.

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

2. 마케팅 그룹의 네임스페이스를 만듭니다.

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

2단계**:** 각 네임스페이스의 리소스와 상호 작용할 새 서비스 계정을 만듭니다

새로 만드는 각 네임스페이스에는 기본 서비스 계정이 함께 제공되지만, 나중에 필요한 경우 그룹 간에 Privileges를 추가로 나눌 수 있도록 각 사용자 그룹에 대한 서비스 계정을 만들어야 합니다.

단계

1. 엔지니어링 그룹에 대한 서비스 계정을 생성합니다.

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

2. 마케팅 그룹의 서비스 계정 만들기:

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

3단계**:** 각 새 서비스 계정에 대한 암호를 만듭니다

서비스 계정 암호는 서비스 계정을 인증하는 데 사용되며, 손상된 경우 쉽게 삭제하고 다시 만들 수 있습니다.

단계

1. 엔지니어링 서비스 계정에 대한 암호를 생성합니다.

```
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. 마케팅 서비스 계정에 대한 암호 만들기:

```
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
```

4단계**: RoleBinding** 개체를 만들어 **ClusterRole** 개체를 각 새 서비스 계정에 바인딩합니다

기본 ClusterRole 개체는 Trident Protect를 설치할 때 만들어집니다. RoleBinding 개체를 만들고 적용하여 이 ClusterRole 을 서비스 계정에 바인딩할 수 있습니다.

단계

1. ClusterRole을 엔지니어링 서비스 계정에 바인딩합니다.

```
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. ClusterRole을 마케팅 서비스 계정에 연결:

```
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
```

5단계**:** 권한을 테스트합니다

권한이 올바른지 테스트합니다.

단계

1. 엔지니어링 사용자가 엔지니어링 리소스에 액세스할 수 있는지 확인합니다.

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

2. 엔지니어링 사용자가 마케팅 리소스에 액세스할 수 없는지 확인합니다.

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

6단계**: AppVault** 객체에 대한 액세스 권한 부여

백업 및 스냅샷과 같은 데이터 관리 작업을 수행하려면 클러스터 관리자가 개별 사용자에게 AppVault 개체에 대한 액세스 권한을 부여해야 합니다.

단계

1. AppVault에 대한 사용자 액세스 권한을 부여하는 AppVault 및 암호 조합 YAML 파일을 만들고 적용합니다. 예를 들어, 다음 CR은 사용자에게 AppVault에 대한 액세스 권한을 `eng-user` 부여합니다.

```
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. 클러스터 관리자가 네임스페이스의 특정 리소스에 대한 액세스 권한을 부여할 수 있도록 역할 CR을 생성하고 적용합니다. 예를 들면 다음과 같습니다.

```
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 CR을 만들고 적용하여 권한을 사용자 eng-user에 바인딩합니다. 예를 들면 다음과 같습니다.

```
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. 권한이 올바른지 확인합니다.

   a. 모든 네임스페이스에 대한 AppVault 개체 정보를 검색하려고 합니다.

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

   다음과 유사한 출력이 표시됩니다.

```
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. 사용자가 이제 액세스 권한이 있는 AppVault 정보를 얻을 수 있는지 테스트해 봅니다.

```
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
```

다음과 유사한 출력이 표시됩니다.

```
yes
```

결과

AppVault 권한을 부여한 사용자는 응용 프로그램 데이터 관리 작업에 승인된 AppVault 개체를 사용할 수 있어야 하며, 할당된 네임스페이스 외부의 리소스에 액세스하거나 액세스 권한이 없는 새 리소스를 생성할 수 없습니다.

Trident Protect 지원 번들을 생성합니다

Trident Protect를 사용하면 관리자가 관리 중인 클러스터 및 앱에 대한 로그, 메트릭, 토폴로지 정보 등 NetApp 지원에 유용한 정보를 포함하는 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 CR(사용자 지정 리소스) 파일을 사용하여 NSS(NetApp 지원 사이트)에 지원 번들을 업로드할 수 있습니다.

CR을 사용하여 지원 번들을 만듭니다

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-support-bundle.yaml`).
2. 다음 특성을 구성합니다.
   - **metadata.name**: (*required*) 이 사용자 정의 리소스의 이름입니다. 사용자 환경에 맞는 고유하고 합리적인 이름을 선택하십시오.
   - * spec.triggerType *:(*required*) 지원 번들이 즉시 생성되는지 또는 예약되는지 여부를 결정합니다. 예약된 번들 생성은 UTC 오전 12시에 수행됩니다. 가능한 값:
     - 예약됨
     - 수동
   - **spec.uploadEnabled**: (*Optional*) 지원 번들이 생성된 후 NetApp 지원 사이트에 업로드되어야 하는지 여부를 제어합니다. 지정하지 않으면 가 기본값으로 `false` 설정됩니다. 가능한 값:
     - 참
     - FALSE(기본값)
   - **spec.dataWindowStart**: (*Optional*) 지원 번들에 포함된 데이터의 창이 시작되는 날짜와 시간을 지정하는 RFC 3339 형식의 날짜 문자열입니다. 지정하지 않을 경우 기본값은 24시간 이전입니다. 지정할 수 있는 가장 빠른 기간 날짜는 7일 이전입니다.

     YAML 예:

```
---
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. 파일을 올바른 값으로 채운 후 `astra-support-bundle.yaml` CR:

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

CLI를 사용하여 지원 번들을 생성합니다

단계

1. 괄호 안의 값을 사용자 환경의 정보로 대체하여 지원 번들을 만듭니다. 은 `trigger-type` 번들이 즉시 생성되는지 아니면 생성 시간이 스케줄에 따라 결정되는지, 또는 `Scheduled` 이 될 수 있는지 결정합니다 `Manual`. 기본 설정은 `Manual`입니다.

예를 들면 다음과 같습니다.

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

Trident Protect를 업그레이드합니다

Trident Protect를 최신 버전으로 업그레이드하여 새로운 기능 또는 버그 수정을 활용할 수 있습니다.

Trident Protect를 업그레이드하려면 다음 단계를 수행하십시오.

단계

1. Trident Helm 리포지토리를 업데이트합니다.

```
helm repo update
```

2. Trident Protect CRD 업그레이드:

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

3. Trident Protect 업그레이드:

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

저작권 정보

상표 정보