



# **Trident Protect 관리**

## Trident

NetApp  
January 15, 2026

# 목차

Trident Protect 관리	1
Trident Protect 권한 및 액세스 제어 관리	1
예: 두 그룹의 사용자에 대한 액세스 관리	1
Trident Protect 리소스 모니터링	7
1단계: 모니터링 도구 설치	8
2단계: 모니터링 도구를 함께 작동하도록 구성	10
3단계: 알림 및 알림 대상 구성	11
Trident Protect 지원 번들 생성	12
지원 번들을 모니터링하고 검색합니다.	14
Trident 프로텍트 업그레이드	14

# Trident Protect 관리

## Trident Protect 권한 및 액세스 제어 관리

Trident Protect는 역할 기반 액세스 제어(RBAC)의 Kubernetes 모델을 사용합니다. 기본적으로 Trident Protect는 단일 시스템 네임스페이스와 연관된 기본 서비스 계정을 제공합니다. 사용자 수가 많거나 특정 보안 요구 사항이 있는 조직인 경우 Trident Protect의 RBAC 기능을 사용하여 리소스와 네임스페이스에 대한 액세스를 보다 세부적으로 제어할 수 있습니다.

클러스터 관리자는 항상 기본 리소스에 액세스할 수 있습니다. `trident-protect` 네임스페이스에 액세스할 수 있으며 다른 모든 네임스페이스의 리소스에도 액세스할 수 있습니다. 리소스와 애플리케이션에 대한 액세스를 제어하려면 추가 네임스페이스를 만들고 해당 네임스페이스에 리소스와 애플리케이션을 추가해야 합니다.

기본적으로 사용자는 애플리케이션 데이터 관리 CR을 생성할 수 없습니다. `trident-protect` 네임스페이스. 애플리케이션 네임스페이스에 애플리케이션 데이터 관리 CR을 만들어야 합니다(가장 좋은 방법은 연관된 애플리케이션과 동일한 네임스페이스에 애플리케이션 데이터 관리 CR을 만드는 것입니다).

다음을 포함하는 권한이 있는 Trident Protect 사용자 정의 리소스 개체에 대한 액세스 권한은 관리자에게만 부여됩니다.

- **AppVault**: 버킷 자격 증명 데이터가 필요합니다.
- **AutoSupportBundle**: 메트릭, 로그 및 기타 중요한 Trident Protect 데이터를 수집합니다.
- **AutoSupportBundleSchedule**: 로그 수집 일정을 관리합니다.

가장 좋은 방법은 RBAC를 사용하여 권한이 있는 개체에 대한 액세스를 관리자로 제한하는 것입니다.

RBAC가 리소스 및 네임스페이스에 대한 액세스를 규제하는 방법에 대한 자세한 내용은 다음을 참조하세요. ["Kubernetes RBAC 문서"](#) .

서비스 계정에 대한 자세한 내용은 다음을 참조하세요. ["Kubernetes 서비스 계정 문서"](#) .

### 예: 두 그룹의 사용자에 대한 액세스 관리

예를 들어, 어떤 조직에 클러스터 관리자, 엔지니어링 사용자 그룹, 마케팅 사용자 그룹이 있다고 가정해 보겠습니다. 클러스터 관리자는 엔지니어링 그룹과 마케팅 그룹이 각자의 네임스페이스에 할당된 리소스에만 액세스할 수 있는 환경을 만들기 위해 다음 작업을 완료합니다.

#### 1단계: 각 그룹의 리소스를 포함할 네임스페이스 만들기

네임스페이스를 만들면 리소스를 논리적으로 분리하고 해당 리소스에 누가 액세스할 수 있는지 더 효과적으로 제어할 수 있습니다.

#### 단계

1. 엔지니어링 그룹에 대한 네임스페이스를 만듭니다.

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

## 2. 마케팅 그룹을 위한 네임스페이스를 만듭니다.

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

### 2단계: 각 네임스페이스의 리소스와 상호 작용하기 위한 새 서비스 계정 만들기

새로 만든 네임스페이스마다 기본 서비스 계정이 제공되지만, 나중에 필요한 경우 그룹 간에 권한을 더욱 세분화할 수 있도록 각 사용자 그룹에 대한 서비스 계정을 만들어야 합니다.

#### 단계

1. 엔지니어링 그룹에 대한 서비스 계정을 만듭니다.

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

## 2. 마케팅 그룹을 위한 서비스 계정을 만듭니다.

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

### 3단계: 각 새 서비스 계정에 대한 비밀 만들기

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

#### 단계

1. 엔지니어링 서비스 계정에 대한 비밀을 만듭니다.

```
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 비밀을 만드세요.

```
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: ClusterRole 객체를 각각의 새 서비스 계정에 바인딩하기 위해 RoleBinding 객체를 만듭니다.

Trident Protect를 설치하면 기본 ClusterRole 객체가 생성됩니다. RoleBinding 객체를 생성하고 적용하여 이 ClusterRole을 서비스 계정에 바인딩할 수 있습니다.

### 단계

1. 엔지니어링 서비스 계정에 ClusterRole을 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 바인딩합니다.

```
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: 권한 테스트

권한이 올바른지 테스트합니다.

### 단계

- 엔지니어링 사용자가 엔지니어링 리소스에 액세스할 수 있는지 확인하세요.

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

- 엔지니어링 사용자가 마케팅 리소스에 액세스할 수 없는지 확인하세요.

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

## 6단계: AppVault 개체에 대한 액세스 권한 부여

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

### 단계

- AppVault 및 비밀번호 조합 YAML 파일을 만들고 적용하여 사용자에게 AppVault에 대한 액세스 권한을 부여합니다. 예를 들어, 다음 CR은 사용자에게 AppVault에 대한 액세스 권한을 부여합니다. 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. 클러스터 관리자가 네임스페이스의 특정 리소스에 대한 액세스 권한을 부여할 수 있도록 역할 CR을 만들고 적용합니다. 예를 들어:

```

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 생성하고 적용하여 사용자 eng-user에게 권한을 바인딩합니다. 예를 들어:

```

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 확인하세요.

a. 모든 네임스페이스에 대한 AppVault 개체 정보를 검색해 보세요.

```

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

```

다음과 비슷한 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 테스트합니다.

```
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 비슷한 출력이 표시됩니다.

```
yes
```

결과

AppVault 권한을 부여한 사용자는 애플리케이션 데이터 관리 작업에 대해 권한이 있는 AppVault 개체를 사용할 수 있어야 하며, 할당된 네임스페이스 외부의 리소스에 액세스하거나 액세스 권한이 없는 새 리소스를 만들 수 없습니다.

## Trident Protect 리소스 모니터링

`kube-state-metrics`, `Prometheus`, `Alertmanager` 오픈 소스 도구를 사용하여 Trident Protect로 보호되는 리소스의 상태를 모니터링할 수 있습니다.

`kube-state-metrics` 서비스는 Kubernetes API 통신에서 메트릭을 생성합니다. Trident Protect와 함께 사용하면 사용자 환경의 리소스 상태에 대한 유용한 정보가 제공됩니다.

`Prometheus`는 `kube-state-metrics`에서 생성된 데이터를 수집하고 이러한 객체에 대한 쉽게 읽을 수 있는 정보로 제공할 수 있는 툴킷입니다. `kube-state-metrics`와 `Prometheus`를 함께 사용하면 Trident Protect로 관리하는 리소스의 상태와 상태를 모니터링할 수 있는 방법을 제공합니다.

`Alertmanager`은 `Prometheus`와 같은 도구에서 보낸 알림을 수집하여 사용자가 구성한 대상으로 라우팅하는 서비스입니다.

이 단계에 포함된 구성과 지침은 단지 예시일 뿐입니다. 사용자 환경에 맞게 사용자 정의해야 합니다. 구체적인 지침과 지원에 대해서는 다음 공식 문서를 참조하세요.



- "["kube-state-metrics 문서"](#)
- "["프로메테우스 문서"](#)
- "["Alertmanager 문서"](#)

## 1단계: 모니터링 도구 설치

Trident Protect에서 리소스 모니터링을 활성화하려면 kube-state-metrics, Prometheus, Alertmanager를 설치하고 구성해야 합니다.

### **kube-state-metrics 설치**

Helm을 사용하여 kube-state-metrics를 설치할 수 있습니다.

#### 단계

1. kube-state-metrics Helm 차트를 추가합니다. 예를 들어:

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

2. 클러스터에 Prometheus ServiceMonitor CRD를 적용합니다.

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

3. Helm 차트에 대한 구성 파일을 만듭니다(예: metrics-config.yaml). 다음 예제 구성은 사용자 환경에 맞게 사용자 정의할 수 있습니다.

## metrics-config.yaml: kube-state-metrics Helm 차트 구성

```
---
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. Helm 차트를 배포하여 kube-state-metrics를 설치합니다. 예를 들어:

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

5. Trident Protect에서 사용하는 사용자 정의 리소스에 대한 메트릭을 생성하도록 kube-state-metrics를 구성하려면 다음 지침을 따르세요. ["kube-state-metrics 사용자 정의 리소스 설명서"](#).

#### 프로메테우스 설치

다음 지침에 따라 Prometheus를 설치할 수 있습니다. ["프로메테우스 문서"](#).

#### Alertmanager 설치

다음 지침에 따라 Alertmanager를 설치할 수 있습니다. ["Alertmanager 문서"](#).

## 2단계: 모니터링 도구를 함께 작동하도록 구성

모니터링 도구를 설치한 후에는 도구가 함께 작동하도록 구성해야 합니다.

#### 단계

1. kube-state-metrics를 Prometheus와 통합합니다. Prometheus 구성 파일 편집(prometheus.yaml) 및 kube-state-metrics 서비스 정보를 추가합니다. 예를 들어:

#### **prometheus.yaml: Prometheus와 kube-state-metrics 서비스 통합**

```
---  
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. Prometheus를 구성하여 알림을 Alertmanager로 라우팅합니다. Prometheus 구성 파일 편집(prometheus.yaml) 다음 섹션을 추가합니다.

## **prometheus.yaml: Alertmanager에 알림 보내기**

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

### 결과

이제 Prometheus는 kube-state-metrics에서 메트릭을 수집하고 Alertmanager에 알림을 보낼 수 있습니다. 이제 알림을 트리거하는 조건과 알림을 보낼 위치를 구성할 준비가 되었습니다.

## **3단계: 알림 및 알림 대상 구성**

도구가 함께 작동하도록 구성한 후에는 어떤 유형의 정보가 알림을 트리거하는지, 알림을 어디로 보낼지 구성해야 합니다.

### 경고 예: 백업 실패

다음 예제에서는 백업 사용자 정의 리소스의 상태가 다음과 같이 설정될 때 트리거되는 중요 경고를 정의합니다. Error 5초 이상. 이 예제를 사용자 환경에 맞게 사용자 정의하고 이 YAML 스니펫을 포함할 수 있습니다.

**prometheus.yaml** 구성 파일:

### **rules.yaml: 실패한 백업에 대한 Prometheus 알림을 정의합니다.**

```
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."
```

다른 채널에 알림을 보내도록 **Alertmanager**를 구성합니다.

이메일, PagerDuty, Microsoft Teams 또는 기타 알림 서비스와 같은 다른 채널에 알림을 보내도록 Alertmanager를 구성하려면 해당 구성을 지정해야 합니다. **alertmanager.yaml** 파일.

다음 예제에서는 Alertmanager가 Slack 채널에 알림을 보내도록 구성합니다. 이 예제를 사용자 환경에 맞게 사용자 지정하려면 다음 값을 바꾸십시오. **api\_url** 사용자 환경에서 사용되는 Slack 웹훅 URL이 포함된 키:

#### alertmanager.yaml: Slack 채널에 알림 보내기

```
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 지원 번들 생성

Trident Protect를 사용하면 관리자는 관리 중인 클러스터와 앱에 대한 로그, 메트릭, 토플로지 정보 등 NetApp 지원에 유용한 정보가 포함된 번들을 생성할 수 있습니다. 인터넷에 연결되어 있는 경우 사용자 정의 리소스(CR) 파일을 사용하여 NetApp 지원 사이트(NSS)에 지원 번들을 업로드할 수 있습니다.

**CR**을 사용하여 지원 번들을 만듭니다.

단계

1. 사용자 정의 리소스(CR) 파일을 만들고 이름을 지정합니다(예: `trident-protect-support-bundle.yaml` ).
2. 다음 속성을 구성합니다.
  - **metadata.name**: (필수) 이 사용자 지정 리소스의 이름입니다. 사용자 환경에 맞게 고유하고 적절한 이름을 선택하세요.
  - **spec.triggerType**: (필수) 지원 번들이 즉시 생성되는지, 아니면 예약되는지를 결정합니다. 예약된 번들 생성은 UTC 기준 오전 12시에 이루어집니다. 가능한 값:
    - 예정됨
    - 수동
  - **spec.uploadEnabled**: (선택 사항) 지원 번들이 생성된 후 NetApp 지원 사이트에 업로드할지 여부를 제어합니다. 지정하지 않으면 기본값으로 설정됩니다. `false`. 가능한 값:
    - `true`
    - `false` (기본값)
  - **spec.dataWindowStart**: (선택 사항) 지원 번들에 포함된 데이터 창이 시작되어야 하는 날짜와 시간을 지정하는 RFC 3339 형식의 날짜 문자열입니다. 지정하지 않으면 기본적으로 24시간 전으로 설정됩니다. 지정할 수 있는 가장 빠른 날짜는 7일 전입니다.

YAML 예시:

```
---
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. 다음을 채운 후 `trident-protect-support-bundle.yaml` 을바른 값으로 파일을 만들고 CR을 적용합니다.

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

**CLI**를 사용하여 지원 번들을 만듭니다.

단계

1. 팔호 안의 값을 사용자 환경의 정보로 바꿔서 지원 번들을 만듭니다. 그만큼 trigger-type 번들이 즉시

생성되는지 또는 생성 시간이 일정에 따라 결정되는지 여부를 결정하고 다음을 수행할 수 있습니다. Manual 또는 Scheduled. 기본 설정은 다음과 같습니다. Manual.

예를 들어:

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

지원 번들을 모니터링하고 검색합니다.

두 가지 방법 중 하나를 사용하여 지원 번들을 만든 후에는 생성 진행 상황을 모니터링하고 로컬 시스템으로 검색할 수 있습니다.

단계

1. 기다리다 `status.generationState` 도달하다 `Completed` 상태. 다음 명령을 사용하여 생성 진행 상황을 모니터링할 수 있습니다.

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

2. 지원 번들을 로컬 시스템으로 가져옵니다. 완성된 AutoSupport 번들에서 복사 명령을 가져옵니다.

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

찾아라 `kubectl cp` 출력에서 명령을 선택하고 대상 인수를 원하는 로컬 디렉토리로 바꿔서 실행합니다.

## Trident 프로텍트 업그레이드

Trident Protect를 최신 버전으로 업그레이드하면 새로운 기능이나 버그 수정을 활용할 수 있습니다.

버전 24.10에서 업그레이드하는 경우 업그레이드 중에 실행 중인 스냅샷이 실패할 수 있습니다. 이 오류로 인해 향후 스냅샷(수동 또는 예약)이 생성되는 것은 방해받지 않습니다. 업그레이드 중에 스냅샷이 실패하면 수동으로 새 스냅샷을 만들어서 애플리케이션을 보호할 수 있습니다.

 잠재적인 실패를 방지하려면 업그레이드 전에 모든 스냅샷 일정을 비활성화한 다음 나중에 다시 활성화할 수 있습니다. 하지만 이로 인해 업그레이드 기간 동안 예약된 스냅샷이 누락되는 문제가 발생합니다.

Trident Protect를 업그레이드하려면 다음 단계를 수행하세요.

단계

### 1. Trident Helm 저장소를 업데이트하세요:

```
helm repo update
```

### 2. Trident Protect CRD 업그레이드:



25.06 이전 버전에서 업그레이드하는 경우 CRD가 이제 Trident Protect Helm 차트에 포함되었으므로 이 단계가 필요합니다.

a. CRD 관리를 전환하려면 이 명령을 실행하세요. `trident-protect-crds`에게 `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. Helm 비밀을 삭제하려면 이 명령을 실행하세요. `trident-protect-crds` 차트:



제거하지 마십시오 `trident-protect-crds` Helm을 사용하여 차트를 만들면 CRD와 관련 데이터가 제거될 수 있습니다.

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

### 3. Trident 프로젝트 업그레이드:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect  
--version 100.2506.0 --namespace trident-protect
```

## 저작권 정보

Copyright © 2026 NetApp, Inc. All Rights Reserved. 미국에서 인쇄됨 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그레픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

## 상표 정보

NETAPP, NETAPP 로고 및 <http://www.netapp.com/TM>에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.