



升级 Astra Trident

Astra Trident

NetApp
April 04, 2024

目录

升级 Astra Trident	1
升级 Astra Trident	1
使用操作员升级	3
使用 tridentctl 进行升级	10

升级 Astra Trident

升级 Astra Trident

Astra Trident 遵循季度发布节奏，每个日历年提供四个主要版本。每个新版本都是在先前版本的基础上构建的，可提供新功能和性能增强以及错误修复和改进功能。我们建议您每年至少升级一次、以利用Astra Trident中的新功能。

升级前的注意事项

升级到最新版本的 Astra Trident 时，请考虑以下事项：

- 在给定Kubernetes集群中的所有命名空间中只应安装一个Astra Trident实例。
- 从 Trident 20.01 开始，仅限测试版 "卷快照" 受支持。Kubernetes 管理员应注意安全地备份或将 alpha Snapshot 对象转换为测试版，以保留原有的 alpha Snapshot 。
 - 从 Kubernetes 1.20 开始，CSI 卷快照现在是 GA 功能。升级之前、您应使用删除Alpha Snapshot CRD `tridentctl obliviate alpha-snapshot-crd` 删除字母快照规范的CRD。
 - 测试版的卷快照引入了一组经过修改的CRD和一个快照控制器、这两个组件都应在升级Astra Trident之前进行设置。
 - 有关详细信息，请参见 ["升级Kubernetes集群前需要了解的事项"](#)。
- 从19.04及更早版本升级的所有版本都需要自行迁移Astra Trident数据 etcd 到CRD对象。确保选中 ["您的Astra Trident版本的专用文档"](#) 以了解升级的工作原理。
- 升级时、请务必提供 `parameter.fsType` 在中 `StorageClasses` 由Astra Trident使用。您可以删除并重新创建 `StorageClasses` 而不会中断已有卷。
 - 这是执行的一项 * 要求 * ["安全上下文"](#) SAN 卷。
 - [sample输入目录](#)包含<https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-basic.yaml.template>等示例[`storage-class-basic.yaml.template`]和链接：<https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-bronze-default.yaml.template>。有关详细信息，请参见 ["已知问题"](#)。

第1步：选择版本

Astra Trident版本采用基于日期的版本 `YY.MM` 命名约定、其中"YY"是一年中的最后两位数字、"MM"是月份。DOT版本会遵循`YY.MM.X` 约定、其中"X"是修补程序级别。您将根据要从中升级的版本选择要升级到的版本。

- 您可以直接升级到已安装版本的四个版本窗口中的任何目标版本。例如、您可以直接从22.04升级到23.04 (包括任何DOT版本、例如22.04.1)。
- 如果您使用的是早期版本、则应使用相应版本的文档执行多步升级、以获取具体说明。这要求您首先升级到适合您的四个版本窗口的最新版本。例如、如果您运行的是18.07、并且希望升级到20.07版本、请按照下面提供的多步升级过程进行操作：
 - a. 首次从 18.07 升级到 19.07 。
 - b. 然后从 19.07 升级到 20.07 。



在OpenShift容器平台上使用TRIDent操作程序进行升级时、应升级到TRIDent 21.01.1或更高版本。21.01.0 版发布的 Trident 运算符包含一个已知的问题描述，该 已在 21.01.1 中修复。有关详细信息，请参见 ["GitHub 上的问题描述详细信息"](#)。

第2步：确定原始安装方法

通常、您应使用与初始安装相同的方法进行升级、但您可以这样做 ["在安装方法之间切换"](#)。

若要确定最初安装Astra Trdent时使用的版本、请执行以下操作：

1. 使用 `... kubectl get pods - trident` 检查Pod。
 - 如果没有操作员POD、则使用安装了Asta三端盘 `tridentctl`。
 - 如果有操作员模块、则使用三端操作员手动或使用Helm安装了A作用 曲三端。
2. 如果有操作员控制盒、请使用 `kubectl describe tproc trident` 确定是否使用Helm安装了Astra Trdent。
 - 如果有Helm标签、则表示Asta Trdent是使用Helm安装的。
 - 如果没有Helm标签、则使用Trident操作人员手动安装A作用 于Trident。

第3步：选择升级方法

有两种方法可升级Asta三端到端。

何时使用操作员进行升级

您可以 ["使用三端修复操作符升级"](#) 条件：

- 您最初使用操作员或安装了Astra Trdent `tridentctl`。
- 您卸载了CSI三元数据、但安装中的元数据仍然存在。
- 您已安装基于CSI的Astra三端技术。自2007年19月7日起的所有版本均基于CSI。您可以检查三端存储命名空间中的Pod以验证您的版本。
 - 23.01之前版本中的POD命名使用： `trident-csi-*`
 - 23.01及更高版本中的Pod命名使用：
 - `trident-controller-<generated id>` 控制器Pod
 - `trident-node-<operating system>-<generated id>` 节点Pod
 - `trident-operator-<generated id>` 对于操作舱



如果您使用的是、请勿使用运算符升级Trident `etcd` 基于Trident的版本(19.04或更早版本)。

何时使用升级 `tridentctl`

您可以 如果您最初是使用`tridentctl`安装Astra三端安装的。

`tridentctl` 是安装Asta Trident的传统方法、为需要复杂自定义的用户提供了大多数选项。有关详细信息，请参见 ["选择安装方法"](#)。

对运算符进行了更改

21.01版的Asta三端设计引入了对运营商的架构进行的更改：

- 操作符现在为 * 集群范围 *。以前的 Trident 运算符实例（版本 20.04 到 20.10）为 * 命名空间范围 *。集群范围内的运算符具有优势，原因如下：
 - 资源责任：操作员现在可以在集群级别管理与 Astra Trident 安装相关的资源。在安装Astra Trident过程中、操作员使用创建和维护多个资源 ownerReferences。维护 ownerReferences 在集群范围的资源上、某些Kubernetes分销商可能会引发错误、例如OpenShift。使用集群范围的运算符可缓解此问题。对于 Trident 资源的自动修复和修补，这是一项基本要求。
 - 卸载期间清理：要完全删除 Astra Trident，需要删除所有关联的资源。命名空间范围的运算符可能会在删除集群范围的资源（例如 clusterRole， ClusterRoleBinding-and PodSecurityPolicy）时遇到问题，并导致清理不完整。集群范围的运算符可消除此问题描述。用户可以完全卸载 Astra Trident 并在需要时重新安装。
- TridentProvisioner 现已替换为 TridentOrchestrator 作为用于安装和管理Astra Trident的自定义资源。此外、还会在中引入一个新字段 TridentOrchestrator 规格用户可以指定必须使用安装/升级命名空间Trident spec.namespace 字段。您可以查看一个示例 ["此处"](#)。

使用操作员升级

您可以使用操作员手动或Helm轻松升级现有的Asta三端安装。

使用Trident操作符进行升级

通常、您应使用最初安装Asta三端存储时所用的方法来升级Asta三端存储。请查看 ["选择升级方法"](#) 在尝试使用三端修复操作符进行升级之前。

从使用命名空间范围的运算符(版本20.07到20.10)安装的A作用于Trident实例升级时、该运算符会自动：



- 迁移 tridentProvisioner 到A tridentOrchestrator 具有相同名称的对象、
- 删除 TridentProvisioner 对象和 tridentprovisioner CRD
- 将A作用 域操作符升级到所使用的集群范围操作符版本
- 将A作用 于最初安装的命名空间安装在相同的命名空间中

升级集群范围的Trident操作员安装

您可以升级集群范围的三端操作员安装。所有Astra Trident 21.01及更高版本均使用集群范围的运算符。

开始之前

确保您使用的是正在运行的Kubbernetes集群 ["支持的Kubernetes版本"](#)。

步骤

1. 验证Astra Trident版本：

```
./tridentctl -n trident version
```

2. 删除用于安装当前 Astra Trident 实例的 Trident 运算符。例如、如果要从22.01升级、请运行以下命令：

```
kubectl delete -f 22.01/trident-installer/deploy/bundle.yaml -n trident
```

3. 如果您使用自定义了初始安装 `TridentOrchestrator` 属性、您可以编辑 `TridentOrchestrator` 用于修改安装参数的对象。其中可能包括为脱机模式指定镜像Trident和CSI映像注册表、启用调试日志或指定映像提取密钥所做的更改。
4. 使用适用于您的环境和Astra Trident版本的正确捆绑包YAML文件安装Astra Trident。例如、如果要为Kubernetes 1.27安装Astra Trident 23.04、请运行以下命令：

```
kubectl create -f 23.04.0/trident-installer/deploy/bundle_post_1_25.yaml  
-n trident
```

Trident提供了一个捆绑包文件、可用于安装操作员并为Kubernetes版本创建关联对象。



- 对于运行Kubornetes 1.24或更早版本的集群、请使用 "`bundle_pre_1_25.yaml`"。
- 对于运行Kubernetes 1.25或更高版本的集群、请使用 "`捆绑包_后_1_25.yaml`"。

结果

Trident操作员将确定现有的Astra Trident安装并将其升级到与操作员相同的版本。

升级命名空间范围的操作员安装

您可以从使用命名空间范围的操作符(版本20.07到20.10)安装的Astra Dent实例升级到集群范围的操作符安装。

开始之前

您需要使用捆绑包YAML文件从中部署命名空间范围的运算符

<https://github.com/NetApp/trident/tree/stable/vXX.XX/deploy/BUNDLE.YAML> 其中：
`vXX.XX` 是版本号和 `BUNDLE.YAML` 是捆绑包YAML文件名。

步骤

1. 验证 `TridentProvisioner` 现有的 `{\f270通过} {\f151。} Installed`。

```
kubectl describe tprov trident -n trident | grep Message: -A 3  
  
Message:  Trident installed  
Status:   Installed  
Version:  v20.10.1
```



如果状态显示 Updating、请确保先解决此问题、然后再继续。有关可能的状态值列表，请参见 ["此处"](#)。

2. 创建 TridentOrchestrator 使用Trident安装程序随附的清单创建CRD。

```
# Download the release required [23.04.0]
mkdir 23.04.0
cd 23.04.0
wget
https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 使用其清单删除命名空间范围的运算符。

a. 确保您位于正确的目录中。

```
pwd
/root/20.10.1/trident-installer
```

b. 删除命名空间范围的运算符。

```
kubectl delete -f deploy/<BUNDLE.YAML> -n trident

serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator"
deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted
```

c. 确认已删除三端修复操作符。

```
kubectl get all -n trident
```

NAME	READY	STATUS	RESTARTS	AGE
pod/trident-csi-68d979fb85-dsrmn	6/6	Running	12	99d
pod/trident-csi-8jfhf	2/2	Running	6	105d
pod/trident-csi-jtnjz	2/2	Running	6	105d
pod/trident-csi-lcxvh	2/2	Running	8	105d

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
service/trident-csi	ClusterIP	10.108.174.125	<none>
34571/TCP,9220/TCP	105d		

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AGE
daemonset.apps/trident-csi	3	3	3	3	
3					kubernetes.io/arch=amd64,kubernetes.io/os=linux 105d

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/trident-csi	1/1	1	1	105d

NAME	DESIRED	CURRENT	READY
replicaset.apps/trident-csi-68d979fb85	1	1	1
105d			

4. (可选)如果需要修改安装参数、请更新 `TridentProvisioner` 规格这可能包括更改、例如更改：`tridentImage`, `autosupportImage`、私有映像存储库和提供 `imagePullSecrets`)。有关可更新的完整参数列表、请参见 ["配置选项"](#)。

```
kubectl patch tprov <trident-provisioner-name> -n <trident-namespace>
--type=merge -p '{"spec":{"debug":true}}'
```

5. 安装在集群范围内的TRident操作符。
- 确保您位于正确的目录中。

```
pwd
/root/23.04.0/trident-installer
```

- 在同一命名空间中安装集群范围的运算符。

Trident提供了一个捆绑包文件、可用于安装操作员并为Kubernetes版本创建关联对象。



- 对于运行Kubornetes 1.24或更早版本的集群、请使用 "bundle_pre_1_25.yaml"。
- 对于运行Kubernetes 1.25或更高版本的集群、请使用 "捆绑包_后_1_25.yaml"。

```
kubect1 create -f deploy/<BUNDLE.YAML>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#All tridentProvisioners will be removed, including the CRD itself
kubect1 get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the
requested resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
kubect1 get torc
NAME          AGE
trident      13s
```

c. 检查命名空间中的三端Pod。。 trident-controller 和POD名称反映了23.01中引入的命名约定。

```
kubect1 get pods -n trident

NAME                                                    READY   STATUS    RESTARTS
AGE
trident-controller-79df798bdc-m79dc                    6/6     Running   0
1m41s
trident-node-linux-xrst8                               2/2     Running   0
1m41s
trident-operator-5574dbbc68-nthjv                      1/1     Running   0
1m52s
```

d. 确认已将三端到端更新到预期版本。

```
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.04.0
```

升级基于 Helm 的操作员安装

要升级基于 Helm 的操作员安装，请执行以下步骤。



将安装了Astra Trident的Kubernetes集群从1.24升级到1.25或更高版本时、必须将values.yaml更新为set excludePodSecurityPolicy to true 或添加 --set excludePodSecurityPolicy=true 到 helm upgrade 命令。

步骤

1. 下载最新的 Astra Trident 版本。
2. 使用 helm upgrade 命令位置 trident-operator-23.04.0.tgz 反映了要升级到的版本。

```
helm upgrade <name> trident-operator-23.04.0.tgz
```

如果在初始安装期间设置了任何非默认选项(例如为Trident和CSI映像指定专用的镜像注册表)、请使用 --set 为了确保这些选项包含在upgrade命令中、否则这些值将重置为默认值。



例如、要更改的默认值 `tridentDebug` 下，运行以下命令：

```
helm upgrade <name> trident-operator-23.04.0-custom.tgz --set
tridentDebug=true
```

3. 运行 helm list 验证图表和应用程序版本均已升级。运行 tridentctl logs 查看任何调试消息。

结果

Trident操作员将确定现有的Astra Trident安装并将其升级到与操作员相同的版本。

从非操作员安装升级

您可以从升级到最新版本的Trident操作员 tridentctl 安装。

步骤

1. 下载最新的 Astra Trident 版本。

```
# Download the release required [23.04.0]
mkdir 23.04.0
cd 23.04.0
wget
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

2. 创建 tridentorchestrator 清单中的CRD。

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. 将集群范围的运算符部署在同一命名空间中。

```
kubectl create -f deploy/<BUNDLE.YAML>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc 6/6     Running   0           150d
trident-node-linux-xrst8            2/2     Running   0           150d
trident-operator-5574dbbc68-nthjv   1/1     Running   0           1m30s
```

4. 创建 TridentOrchestrator 安装Astra Trident的CR。

```

cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0           1m
trident-csi-xrst8                    2/2     Running   0           1m
trident-operator-5574dbbc68-nthjv    1/1     Running   0           5m41s

```

5. 确认已将三项功能升级到预期版本。

```

kubectl describe torc trident | grep Message -A 3

Message:          Trident installed
Namespace:        trident
Status:           Installed
Version:          v23.04.0

```

结果

现有后端和 PVC 会自动可用。

使用 `tridentctl` 进行升级

您可以使用轻松升级现有的Astra Trident安装 `tridentctl`。

使用升级Astra三端到功能 `tridentctl`

卸载并重新安装 Astra Trident 可作为升级。卸载 Trident 时，不会删除 Astra Trident 部署所使用的永久性卷声明（PVC）和永久性卷（PV）。在 Astra Trident 脱机期间，已配置的 PV 仍可用，而 Astra Trident 将在恢复联机后为在此期间创建的任何 PVC 配置卷。

开始之前

请查看 ["选择升级方法"](#) 升级之前 `tridentctl`。

步骤

1. 在中运行卸载命令 `tridentctl` 删除与Asta Trident关联的所有资源(CRD和相关对象除外)。

```
./tridentctl uninstall -n <namespace>
```

2. 重新安装Asta Trident。请参见 ["使用 tridentctl 安装 Astra Trident"](#)。



请勿中断升级过程。确保安装程序运行完毕。

使用升级卷 tridentctl

升级后、您可以利用较新的三元数据版本中提供的一组丰富功能(例如按需卷快照)、也可以使用升级卷 `tridentctl upgrade` 命令:

如果存在原有卷、则应将其从NFS或iSCSI类型升级到CSI类型、以使用Asta三元数据中的一整套新功能。Trident 配置的原有 PV 支持传统功能集。

开始之前

在决定将卷升级到CSI类型之前、请考虑以下事项:

- 您可能不需要升级所有卷。以前创建的卷将继续可访问并正常运行。
- 升级时, PV 可以作为部署 / 状态集的一部分挂载。不需要关闭部署 / 状态集。
- 升级时, 您 * 无法 * 将 PV 连接到独立 POD。在升级卷之前, 您应关闭 POD。
- 您只能升级绑定到 PVC 的卷。升级前, 应删除和导入未绑定到 PVC 的卷。

步骤

1. 运行 `kubectl get pv` 列出PV。

```
kubectl get pv
NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                                STORAGECLASS   REASON   AGE
default-pvc-1-a8475                    1073741824   RWO           Delete
Bound   default/pvc-1                        standard              19h
default-pvc-2-a8486                    1073741824   RWO           Delete
Bound   default/pvc-2                        standard              19h
default-pvc-3-a849e                    1073741824   RWO           Delete
Bound   default/pvc-3                        standard              19h
default-pvc-4-a84de                    1073741824   RWO           Delete
Bound   default/pvc-4                        standard              19h
trident                                2Gi         RWO           Retain
Bound   trident/trident                      standard              19h
```

目前、Trident 20.07使用创建了四个PV `netapp.io/trident` 配置程序。

2. 运行 `kubectl describe pv` 以获取PV的详细信息。

```
kubectl describe pv default-pvc-2-a8486

Name:          default-pvc-2-a8486
Labels:        <none>
Annotations:   pv.kubernetes.io/provisioned-by: netapp.io/trident
               volume.beta.kubernetes.io/storage-class: standard
Finalizers:    [kubernetes.io/pv-protection]
StorageClass:  standard
Status:        Bound
Claim:         default/pvc-2
Reclaim Policy: Delete
Access Modes:  RWO
VolumeMode:    Filesystem
Capacity:      1073741824
Node Affinity: <none>
Message:
Source:
  Type:        NFS (an NFS mount that lasts the lifetime of a pod)
  Server:      10.xx.xx.xx
  Path:        /trid_1907_alpha_default_pvc_2_a8486
  ReadOnly:    false
```

PV是使用创建的 `netapp.io/trident` 配置程序和类型为NFS。要支持 Astra Trident 提供的所有新功能，应将此 PV 升级到 CSI 类型。

3. 运行 `tridentctl upgrade volume <name-of-trident-volume>` 用于将原有Astra Trident卷升级到CSI规范的命令。

```

./tridentctl get volumes -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS | PROTOCOL |
BACKEND UUID            | STATE  | MANAGED      |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS | PROTOCOL |
BACKEND UUID            | STATE  | MANAGED      |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true         |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

4. 运行 `kubectl describe pv` 验证此卷是否为CSI卷。

```
kubectl describe pv default-pvc-2-a8486
Name:                default-pvc-2-a8486
Labels:              <none>
Annotations:         pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                    volume.beta.kubernetes.io/storage-class: standard
Finalizers:          [kubernetes.io/pv-protection]
StorageClass:        standard
Status:              Bound
Claim:               default/pvc-2
Reclaim Policy:      Delete
Access Modes:        RWO
VolumeMode:          Filesystem
Capacity:            1073741824
Node Affinity:       <none>
Message:
Source:
  Type:               CSI (a Container Storage Interface (CSI) volume
source)
  Driver:              csi.trident.netapp.io
  VolumeHandle:        default-pvc-2-a8486
  ReadOnly:            false
  VolumeAttributes:    backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                    name=default-pvc-2-a8486
                    protocol=file
Events:               <none>
```


版权信息

版权所有 © 2024 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。