



安裝 Trident Trident

NetApp
October 30, 2026

目錄

安裝 Trident	1
了解 Trident 安裝	1
關於 Trident 26.06 的關鍵資訊	1
開始之前	1
選擇您的安裝方式	1
選擇您的安裝模式	3
根據您的方法和模式選擇流程	3
在安裝方法之間移動	4
其他已知組態選項	4
使用 Trident 操作員進行安裝	4
手動部署 Trident 操作員（標準模式）	4
手動部署 Trident 操作員（離線模式）	9
使用 Helm 部署 Trident Operator（標準模式）	14
使用 Helm 部署 Trident Operator（離線模式）	21
自訂 Trident 操作員安裝	28
使用 tridentctl 安裝	38
使用 tridentctl 安裝 Trident	38
自訂 Trident tridentctl 安裝	41
使用 OpenShift 認證的操作人員進行安裝	42
使用 OpenShift OperatorHub 安裝 Trident	42
切換到 OpenShift 認證的 Trident 操作員	46

安裝 Trident

了解 Trident 安裝

為了確保 Trident 能夠安裝在各種不同的環境和組織中，NetApp 提供了多種安裝選項。您可以使用 Trident Operator（手動或使用 Helm）或透過 `tridentctl` 安裝 Trident。本主題提供了選擇適合您的安裝流程的重要資訊。

關於 Trident 26.06 的關鍵資訊

您必須閱讀以下關於 **Trident** 的重要資訊。

▼ 關於 Trident 的重要資訊

- Trident 現在支援 Kubernetes。請先升級 Trident，再升級 Kubernetes。
 - Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 `multipath.conf` 檔案中設定 `find_multipaths: no` 值。
- 使用非多路徑配置或在 `multipath.conf` 檔案中使用 `find_multipaths: yes` 或 `find_multipaths: smart` 值會導致掛載失敗。Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

開始之前

無論採用何種安裝路徑，您都必須具備：

- 對執行受支援 Kubernetes 版本且已啟用相關功能需求的受支援 Kubernetes 叢集擁有完整權限。詳情請查看 ["要求"](#)。
- 可存取受支援的 NetApp 儲存系統。
- 能夠從所有 Kubernetes 工作節點掛載磁碟區。
- 一台安裝了 `kubectl`（或 `oc`，如果您正在使用 OpenShift）並配置用於管理您要使用的 Kubernetes 叢集的 Linux 主機。
- `KUBECONFIG` 環境變數設定為指向您的 Kubernetes 叢集配置。
- 如果您正在將 Kubernetes 與 Docker Enterprise 一起使用，["請依照其步驟啟用 CLI 存取"](#)
- 叢集必須支援特權工作負載。



如果您還沒有熟悉 ["基本概念"](#)，現在正是了解的好時機。

選擇您的安裝方式

選擇適合您的安裝方式。在做出決定之前，您還應該考慮 ["方法之間的轉換"](#)。

使用 Trident 運算子

無論手動部署或使用 Helm，Trident operator 都是簡化安裝和動態管理 Trident 資源的絕佳方式。您甚至可以[自訂您的 Trident 操作員部署](#)使用 `TridentOrchestrator` 自訂資源（CR）中的屬性。

使用 Trident 運算子的優點包括：

▼ **Trident 物件建立**

Trident 運算子會自動為您的 Kubernetes 版本建立下列物件。

- 操作員的 ServiceAccount
- ClusterRole 和 ClusterRoleBinding 至 ServiceAccount
- 專用 PodSecurityPolicy（適用於 Kubernetes 1.25 及更早版本）
- 運算子本身

▼ **資源問責制**

叢集範圍的 Trident 運算子可管理與叢集層級 Trident 安裝相關的資源。這可減少使用命名空間範圍的運算子維護叢集範圍資源時可能造成的錯誤。這對於自我修復和修補至關重要。

▼ **自我修復功能**

操作員會監控 Trident 安裝，並主動採取措施來解決問題，例如刪除部署或意外修改部署時。系統會建立 `trident-operator-<generated-id>` Pod，將 `TridentOrchestrator` CR 與 Trident 安裝建立關聯。這可確保叢集中只有一個 Trident 執行個體，並控制其設定，確保安裝具有等冪性。當安裝發生變更時（例如刪除部署或節點 daemonset），操作員會識別這些變更並逐一修正。

▼ **輕鬆更新現有安裝**

您可以使用 operator 輕鬆更新現有部署。您只需編輯 `TridentOrchestrator` CR 即可對安裝進行更新。

例如，假設您需要啟用 Trident 產生偵錯日誌。為此，請修改您的 `TridentOrchestrator` 以進行設定 ``spec.debug`` 為 ``true``：

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

更新 `TridentOrchestrator` 後，操作員會處理更新並修補現有安裝。這可能會觸發建立新的 Pod 以相應地修改安裝。

▼ **全新重新安裝**

叢集範圍的 Trident 運算子可實現叢集範圍資源的完全移除。使用者可以完全解除安裝 Trident 並輕鬆重新安裝。

▼ **自動 Kubernetes 升級處理**

當叢集的 Kubernetes 版本升級到支援的版本時，操作員會自動更新現有的 Trident 安裝，並對其進行更改，以確保其符合 Kubernetes 版本的要求。



如果叢集升級到不支援的版本，操作符會阻止安裝 Trident。如果 Trident 已透過操作符安裝，則會顯示警告，指出 Trident 安裝在不支援的 Kubernetes 版本上。

使用 `tridentctl`

如果您有需要升級的現有部署，或者您希望對部署進行高度定制，則應考慮使用。這是部署 Trident 的傳統方法。

您可以產生 Trident 資源的清單。這包括部署、daemonset、服務帳戶以及 Trident 在安裝過程中所建立的叢集角色。



從 22.04 版本開始，每次安裝 Trident 時將不再重新產生 AES 金鑰。在此版本中，Trident 將安裝一個新的秘密物件，該物件在後續安裝中保持不變。這意味著，`tridentctl` 在 22.04 版本中可以解除安裝先前版本的 Trident，但先前版本無法解除安裝 22.04 版本的安裝。請選擇合適的安裝方法。

選擇您的安裝模式

根據貴組織所需的 *installation mode*（Standard、Offline 或 Remote）確定部署流程。

標準安裝

這是安裝 Trident 最簡單的方法，適用於大多數沒有網路限制的環境。標準安裝模式使用預設登錄來儲存所需的 Trident (docker.io 和 CSI (registry.k8s.io 映像。

使用標準模式時，Trident 安裝程式：

- 從網際網路擷取容器映像
- 建立一個部署或節點守護程式集，該程式集會在 Kubernetes 叢集中所有符合條件的節點上啟動 Trident Pod

離線安裝

在實體隔離或安全環境中，可能需要離線安裝模式。在這種情況下，您可以建立單一私有鏡像註冊表，或建立兩個鏡像註冊表來儲存所需的 Trident 和 CSI 映像。



無論您的登錄配置如何，CSI 映像都必須位於同一個登錄檔中。

遠端安裝

以下是遠端安裝過程的概要：

- 在要部署 Trident 的遠端機器上部署適當版本的 `kubectl`。
- 從 Kubernetes 叢集複製設定檔，並在遠端機器上設定 `KUBECONFIG` 環境變數。
- 發起 `kubectl get nodes` 命令以驗證是否可以連接到所需的 Kubernetes 叢集。
- 使用標準安裝步驟，從遠端電腦完成部署。

根據您的方法和模式選擇流程

做出決定後，請選擇適當的流程。

方法	安裝模式
Trident 操作員（手動）	"標準安裝" "離線安裝"
Trident 操作員（Helm）	"標準安裝" "離線安裝"
tridentctl	"標準安裝或離線安裝"

在安裝方法之間移動

您可以決定更改安裝方式。在執行此操作之前，請考慮以下事項：

- 安裝和解除安裝 Trident 時，請務必使用相同的方法。如果您使用 `tridentctl` 進行部署，則應使用對應版本的 ``tridentctl`` 二進位檔案來解除安裝 Trident。同樣，如果您使用 `operator` 進行部署，則應編輯 ``TridentOrchestrator`` CR 並設定 ``spec.uninstall=true`` 以解除安裝 Trident。
- 如果您有一個基於 Operator 的部署，想要將其移除並改用 ``tridentctl`` 來部署 Trident，則應先編輯 ``TridentOrchestrator`` 並設定 ``spec.uninstall=true`` 以卸載 Trident。然後刪除 ``TridentOrchestrator`` 和 Operator 部署。之後，您可以使用 ``tridentctl`` 進行安裝。
- 如果您使用的是手動操作符部署，並且想要使用基於 Helm 的 Trident 操作符部署，則應先手動卸載操作符，然後再執行 Helm 安裝。這樣 Helm 才能使用所需的標籤和註解來部署 Trident 操作符。否則，基於 Helm 的 Trident 操作符部署將會失敗，並出現標籤驗證錯誤和註解驗證錯誤。
- 如果您有 ``tridentctl`` 型部署，則可以在不卸載 Trident 的情況下執行基於 Helm 或基於 Operator 的部署。

其他已知組態選項

在 VMware Tanzu Portfolio 產品上安裝 Trident 時：

- 此 `--kubelet-dir`` 標誌應設定為 `kubelet` 目錄的位置。預設情況下，此值為 ``/var/vcap/data/kubelet``。

已知使用 ``--kubelet-dir`` 指定 `kubelet` 位置的方法對 Trident Operator、Helm 和 ``tridentctl`` 部署有效。

使用 Trident 操作員進行安裝

手動部署 Trident 操作員（標準模式）

您可以手動部署 Trident Operator 來安裝 Trident。此程序適用於 Trident 所需的容器映像未儲存在私人登錄中的安裝。如果您確實擁有私有映像登錄，請使用 ["離線部署流程"](#)。

關於 **Trident 26.06** 的關鍵資訊

您必須閱讀以下關於 **Trident** 的重要資訊。

▼ **關於 Trident 的重要資訊**

- Trident 現在支援 Kubernetes 1.37。請先升級 Trident，再升級 Kubernetes。
- Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 multipath.conf 檔案中設定 `find_multipaths: no` 值。

使用非多路徑配置或在 multipath.conf 檔案中使用 `find_multipaths: yes` 或 `find_multipaths: smart` 值會導致掛載失敗。Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

手動部署 Trident 操作員並安裝 Trident

檢查 ["安裝概述"](#) 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

在開始安裝之前，請登入 Linux 主機並驗證它是否正在管理運作中的 ["支援的 Kubernetes 叢集"](#)，以及您是否擁有必要的權限。



使用 OpenShift 時，請在以下所有範例中使用 `oc` 而非 `kubectl`，並先執行 `oc login -u system:admin` 或 `oc login -u kube-admin` 以 **system:admin** 身分登入。

▼ Details

1. 驗證您的 Kubernetes 版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. 驗證您是否可以啟動使用 Docker Hub 映像的 pod，並透過 pod 網路存取您的儲存系統：

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

步驟 1：下載 Trident 安裝程式套件

Trident 安裝套件包含部署 Trident Operator 和安裝 Trident 所需的一切。從 ["GitHub 上的_資產_部分"](#) 下載並解壓縮最新版本的 Trident 安裝程式。

```
wget https://github.com/NetApp/trident/releases/download/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

步驟 2：建立 TridentOrchestrator CRD

建立 TridentOrchestrator Custom Resource Definition (CRD)。稍後您將建立 TridentOrchestrator Custom Resource。請使用 `deploy/crds` 中對應的 CRD YAML 版本來建立 TridentOrchestrator CRD。

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

步驟 3：部署 Trident 操作人員

Trident 安裝程式提供了一個捆綁文件，可用來安裝 Operator 並建立關聯物件。該捆綁檔案提供了一種簡單的方法，可以使用預設配置部署 Operator 並安裝 Trident。

- 對於運行 Kubernetes 1.24 的叢集，請使用 `bundle_pre_1_25.yaml`。
- 對於運行 Kubernetes 1.25 或更高版本的集群，請使用 `bundle_post_1_25.yaml`。

開始之前

- 根據預設，Trident 安裝程式會將操作員部署在 `trident` 命名空間中。如果 `trident` 命名空間不存在，請使用以下命令建立：

```
kubectl apply -f deploy/namespace.yaml
```

- 若要將運算子部署到 `trident` 命名空間以外的其他命名空間，請更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml`，並使用 `kustomization.yaml` 產生您的捆綁包檔案。
 - 使用以下命令建立 `kustomization.yaml`，其中 `<bundle.yaml>` 是 `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml`，具體取決於您的 Kubernetes 版本。

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

- 使用以下命令編譯捆綁包，其中 `<bundle.yaml>` 是 `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml`，取決於您的 Kubernetes 版本。

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

步驟

1. 建立資源並部署 operator：

```
kubectl create -f deploy/<bundle.yaml>
```

2. 確認已建立 operator、deployment 和 replicaset。

```
kubectl get all -n <operator-namespace>
```



Kubernetes 叢集中只能存在 一個 Operator 實例。請勿建立多個 Trident Operator 部署。

步驟 4：建立 **TridentOrchestrator** 並安裝 **Trident**

現在您可以建立 **TridentOrchestrator** 並安裝 **Trident**。或者，您可以["自訂您的 Trident 安裝"](#)使用 **TridentOrchestrator** 規範中的屬性。

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:      true
  Namespace:  trident
  nodePrep:
    - iscsi
Status:
  Current Installation Params:
    IPv6:          false
    Autosupport Hostname:
    Autosupport Image:      netapp/trident-autosupport:26.06
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:          true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:      30
    Kubelet Dir:      /var/lib/kubelet
    Log Format:       text
    Silence Autosupport:  false
    Trident Image:    netapp/trident:26.06.0
  Message:          Trident installed Namespace:
trident
  Status:           Installed
```

```
Version:                                v26.06.0
Events:
  Type Reason Age From Message ---- -
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed
```

驗證安裝

有幾種方法可以驗證您的安裝。

使用 **TridentOrchestrator** 狀態

``TridentOrchestrator`` 的狀態指示安裝是否成功，並顯示已安裝的 Trident 版本。安裝過程中，``TridentOrchestrator`` 的狀態會從 ``Installing`` 變更為 ``Installed``。如果您觀察到 ``Failed`` 狀態，且操作員無法自行恢復，`link:../troubleshooting.html["檢查日誌"]`。

狀態	說明
安裝	操作員正在使用此 <code>TridentOrchestrator</code> CR 安裝 Trident。
已安裝	Trident 已成功安裝。
解除安裝	操作員正在卸載 Trident，因為 <code>spec.uninstall=true</code> 。
已解除安裝	Trident 已卸載。
失敗	操作員無法安裝、修補、更新或解除安裝 Trident；操作員將自動嘗試從此狀態復原。如果此狀態持續存在，則需要進行疑難排解。
正在更新	操作員正在更新現有安裝。
錯誤	該 <code>`TridentOrchestrator`</code> 未使用。已存在另一個。

使用 **pod** 建立狀態

您可以透過檢視已建立的 pod 的狀態來確認 Trident 安裝是否已完成：

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw 6/6     Running   0
1m
trident-node-linux-mr6zc             2/2     Running   0
1m
trident-node-linux-xrp7w             2/2     Running   0
```

```

1m
trident-node-linux-zh2jt          2/2      Running    0
1m
trident-operator-766f7b8658-ldzsv 1/1      Running    0
3m

```

使用 **tridentctl**

您可以使用 `tridentctl` 檢查已安裝的 Trident 版本。

```

./tridentctl -n trident version

+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+
| 26.06.0       | 26.06.0       |
+-----+

```

手動部署 Trident 操作員（離線模式）

您可以手動部署 Trident Operator 來安裝 Trident。此程序適用於 Trident 所需的容器映像儲存在私人登錄中的安裝。如果您沒有私有映像登錄，請使用 ["標準部署流程"](#)。

關於 Trident 的重要資訊

您必須閱讀以下關於 Trident 的重要資訊。

▼ 關於 Trident 的重要資訊

- Trident 現在支援 Kubernetes 1.37。請先升級 Trident，再升級 Kubernetes。
 - Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 `multipath.conf` 檔案中設定 `find_multipaths: no` 值。
- 使用非多路徑配置或在 `multipath.conf` 檔案中使用 `find_multipaths: yes` 或 `find_multipaths: smart` 值會導致掛載失敗。Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

手動部署 Trident 操作員並安裝 Trident

檢查 ["安裝概述"](#) 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

登入 Linux 主機，並驗證它是否正在管理一個正常運作的 ["支援的 Kubernetes 叢集"](#)，以及您是否擁有必要的權限。



使用 OpenShift 時，請在以下所有範例中使用 `oc` 而非 `kubectl`，並先執行 `oc login -u system:admin` 或 `oc login -u kube-admin` 以 **system:admin** 身分登入。

▼ Details

1. 驗證您的 Kubernetes 版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. 驗證您是否可以啟動使用 Docker Hub 映像的 pod，並透過 pod 網路存取您的儲存系統：

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

步驟 1：下載 Trident 安裝程式套件

Trident 安裝套件包含部署 Trident Operator 和安裝 Trident 所需的一切。從["GitHub 上的_資產_部分"](#)下載並解壓縮最新版本的 Trident 安裝程式。

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

步驟 2：建立 TridentOrchestrator CRD

建立 TridentOrchestrator Custom Resource Definition (CRD)。稍後您將建立 TridentOrchestrator Custom Resources。請使用 `deploy/crds` 中對應的 CRD YAML 版本建立 `TridentOrchestrator CRD`：

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

步驟 3：更新 operator 中的登錄位置

在 `deploy/operator.yaml` 中，更新 `image: docker.io/netapp/trident-operator:26.06.0` 以反映您的映像登錄的位置。您的 ["Trident 與 CSI 影像"](#) 可以位於一個登錄中，也可以位於不同的登錄中，但所有 CSI 映像必須位於同一個登錄中。例如：

- `image: <your-registry>/trident-operator:26.06.0` 如果您的所有映像都位於同一個登錄中。

- image: <your-registry>/netapp/trident-operator:26.06.0 如果您的 Trident 映像與 CSI 映像位於不同的登錄檔。

步驟 4：部署 Trident 操作人員

Trident 安裝程式提供了一個捆綁文件，可用來安裝 Operator 並建立關聯物件。該捆綁檔案提供了一種簡單的方法，可以使用預設配置部署 Operator 並安裝 Trident。

- 對於運行 Kubernetes 1.24 的叢集，請使用 bundle_pre_1_25.yaml。
- 對於運行 Kubernetes 1.25 或更高版本的集群，請使用 bundle_post_1_25.yaml。

開始之前

- 根據預設，Trident 安裝程式會將操作員部署在 trident 命名空間中。如果 trident 命名空間不存在，請使用以下命令建立：

```
kubectl apply -f deploy/namespace.yaml
```

- 若要將運算子部署到 trident 命名空間以外的其他命名空間，請更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml`，並使用 `kustomization.yaml` 產生您的捆綁包檔案。
 - a. 使用以下命令建立 kustomization.yaml，其中 <bundle.yaml> 是 bundle_pre_1_25.yaml 或 bundle_post_1_25.yaml，具體取決於您的 Kubernetes 版本。

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

- b. 使用以下命令編譯捆綁包，其中 <bundle.yaml> 是 bundle_pre_1_25.yaml 或 bundle_post_1_25.yaml，取決於您的 Kubernetes 版本。

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

步驟

1. 建立資源並部署 operator：

```
kubectl create -f deploy/<bundle.yaml>
```

2. 確認已建立 operator、deployment 和 replicaset。

```
kubectl get all -n <operator-namespace>
```



Kubernetes 叢集中只能存在一個 Operator 實例。請勿建立多個 Trident Operator 部署。

步驟 5：更新 **TridentOrchestrator** 中的映像登錄位置

您的 **"Trident 與 CSI 影像"** 可以位於一個登錄中，也可以位於不同的登錄中，但所有 CSI 映像必須位於同一個登錄中。更新 `deploy/crds/tridentorchestrator_cr.yaml` 以根據您的登錄配置新增其他位置規格。

一個登錄中的映像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

不同登錄中的映像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

步驟 6：建立 **TridentOrchestrator** 並安裝 **Trident**

現在您可以建立 **TridentOrchestrator** 並安裝 **Trident**。您也可以選擇進一步**"自訂您的 Trident 安裝"**，方法是使用 `TridentOrchestrator` 規格中的屬性。以下範例顯示 **Trident** 和 **CSI** 映像檔位於不同註冊中心的安裝情況。

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/trident-autosupport:26.06
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.06.0
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
```

```

Autosupport Image:      <your-registry>/trident-autosupport:26.06
Autosupport Proxy:
Autosupport Serial Number:
Debug:                  true
Http Request Timeout:   90s
Image Pull Secrets:
Image Registry:         <your-registry>
k8sTimeout:             30
Kubelet Dir:            /var/lib/kubelet
Log Format:              text
Probe Port:              17546
Silence Autosupport:    false
Trident Image:          <your-registry>/trident:26.06.0
Message:                 Trident installed
Namespace:               trident
Status:                  Installed
Version:                 v26.06.0
Events:
  Type Reason Age From Message ---- -
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

驗證安裝

有幾種方法可以驗證您的安裝。

使用 **TridentOrchestrator** 狀態

`TridentOrchestrator` 的狀態指示安裝是否成功，並顯示已安裝的 Trident 版本。安裝過程中，`TridentOrchestrator` 的狀態會從 `Installing` 變更為 `Installed`。如果您觀察到 `Failed` 狀態，且操作員無法自行恢復，[link:../troubleshooting.html\["檢查日誌"\]](#)。

狀態	說明
安裝	操作員正在使用此 TridentOrchestrator CR 安裝 Trident。
已安裝	Trident 已成功安裝。
解除安裝	操作員正在卸載 Trident，因為 <code>spec.uninstall=true</code> 。
已解除安裝	Trident 已卸載。
失敗	操作員無法安裝、修補、更新或解除安裝 Trident；操作員將自動嘗試從此狀態復原。如果此狀態持續存在，則需要進行疑難排解。

狀態	說明
正在更新	操作員正在更新現有安裝。
錯誤	該 `TridentOrchestrator` 未使用。已存在另一個。

使用 **pod** 建立狀態

您可以透過檢視已建立的 pod 的狀態來確認 Trident 安裝是否已完成：

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS
AGE			
trident-controller-7d466bf5c7-v4cpw	6/6	Running	0
1m			
trident-node-linux-mr6zc	2/2	Running	0
1m			
trident-node-linux-xrp7w	2/2	Running	0
1m			
trident-node-linux-zh2jt	2/2	Running	0
1m			
trident-operator-766f7b8658-ldzsv	1/1	Running	0
3m			

使用 **tridentctl**

您可以使用 `tridentctl` 檢查已安裝的 Trident 版本。

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

使用 Helm 部署 Trident Operator（標準模式）

您可以使用 Helm 部署 Trident Operator 並安裝 Trident。此程序適用於 Trident 所需的容器映像未儲存在私人登錄中的安裝。如果您確實擁有私有映像登錄，請使用 "[離線部署流程](#)"。

關於 **Trident 26.06** 的關鍵資訊

您必須閱讀以下關於 **Trident** 的重要資訊。

▼ 關於 Trident 的重要資訊

- Trident 現在支援 Kubernetes 1.37。請先升級 Trident，再升級 Kubernetes。
- Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 multipath.conf 檔案中設定 `find_multipaths: no` 值。

使用非多路徑配置或在 multipath.conf 檔案中使用 `find_multipaths: yes` 或 `find_multipaths: smart` 值會導致掛載失敗。Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

使用 Helm 部署 Trident Operator 並安裝 Trident

使用 Trident "[Helm Chart](#)"，您可以在一個步驟中部署 Trident 操作員並安裝 Trident。

檢查 "[安裝概述](#)" 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

除了 "[部署先決條件](#)" 之外，您還需要 "[Helm 版本 3](#)"。

步驟

1. 新增 Trident Helm 儲存庫：

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 使用 `helm install` 並為您的部署指定一個名稱，如下例所示，其中 100.2606.0 是您要安裝的 Trident 版本。

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0  
--create-namespace --namespace <trident-namespace>
```



如果您已經為 Trident 建立命名空間，則 `--create-namespace` 參數不會建立額外的命名空間。

您可以使用 `helm list` 來檢閱安裝詳細資料，例如名稱、命名空間、圖表、狀態、應用程式版本和修訂編號。

安裝期間傳遞組態資料

安裝過程中有兩種方法可以傳遞組態資料：

選項	說明
<code>--values</code> (或 <code>-f</code>)	指定一個包含覆蓋設定的 YAML 檔案。可以多次指定，最右側的檔案優先順序最高。
<code>--set</code>	在命令列中指定覆寫設定。

例如，若要變更 debug 的預設值，請執行下列命令，其中 100.2606.0 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--create-namespace --namespace trident --set tridentDebug=true
```

組態選項

該表和 values.yaml 檔案（屬於 Helm chart 的一部分）提供了鍵及其預設值的列表。

選項	說明	預設
nodeSelector	用於 Pod 分配的節點標籤	
podAnnotations	Pod 註解	
deploymentAnnotations	部署註解	
tolerations	Pod 指派的容許	
affinity	對 pod 分配的親和性	<pre>affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux</pre> <div> 請勿從 values.yaml 檔案中移除預設親和性設定。如果您需要自訂親和性，請擴展預設親和性設定。</div>
tridentControllerPluginNodeSelector	Pod 的其他節點選擇器。如需詳細資訊，請參閱 了解控制器 Pod 和節點 Pod 。	

選項	說明	預設
tridentControllerPluginTolerations	覆寫 Kubernetes 對 Pod 的容忍度。如需詳細資訊，請參閱 了解控制器 Pod 和節點 Pod 。	
tridentNodePluginNodeSelector	Pod 的其他節點選擇器。如需詳細資訊，請參閱 了解控制器 Pod 和節點 Pod 。	
tridentNodePluginTolerations	覆寫 Kubernetes 對 Pod 的容忍度。如需詳細資訊，請參閱 了解控制器 Pod 和節點 Pod 。	
imageRegistry	識別 trident-operator、trident 和其他映像的登錄。留空以接受預設值。重要：在私有儲存庫中安裝 Trident 時，如果您使用 `imageRegistry` 開關指定儲存庫位置，請勿在儲存庫路徑中使用 `/netapp/`。	""
imagePullPolicy	設定 trident-operator 的鏡像拉取策略。	IfNotPresent
imagePullSecrets	設定 trident-operator、trident 及其他映像的映像提取密鑰。	
kubeletDir	允許覆蓋 kubelet 內部狀態的主機位置。	"/var/lib/kubelet"
operatorLogLevel	允許將 Trident 運算子的日誌等級設為：trace、debug、info、warn、error 或 fatal。	"info"
operatorDebug	允許將 Trident 操作符的日誌等級設定為 debug。	true
operatorImage	允許完全覆蓋 trident-operator 的影像。	""
operatorImageTag	允許覆蓋 trident-operator 影像的標籤。	""
tridentIPv6	允許啟用 Trident 以在 IPv6 叢集中運作。	false
tridentK8sTimeout	覆寫大多數 Kubernetes API 操作的預設 30 秒逾時（如果非零、則以秒為單位）。	0
tridentHttpRequestTimeout	覆蓋 HTTP 請求的預設 90 秒逾時時間，0s 逾時時間可以設定為無限長。不允許使用負值。	"90s"
tridentSilenceAutosupport	允許禁用 Trident 定期 AutoSupport 報告。	false

選項	說明	預設
tridentAutoSupportImageTag	允許覆蓋 Trident AutoSupport 容器的映像標籤。	<version>
tridentAutoSupportProxy	使 Trident AutoSupport 容器能夠透過 HTTP Proxy 連接到伺服器。	""
tridentLogFormat	設定 Trident 日誌格式 ((text 或 json))。	"text"
tridentDisableAuditLog	停用 Trident 稽核記錄器。	true
tridentLogLevel	允許將 Trident 的日誌等級設為：trace、debug、info、warn、error 或 fatal。	"info"
tridentDebug	允許將 Trident 的日誌等級設為 debug。您可以透過與節點健康狀況檢查 (NHC) 操作符整合來自動執行強制分離程序。如需相關資訊，請參閱 "使用 Trident 自動化工具狀態應用程式的容錯移轉" 。	false
tridentLogWorkflows	允許為追蹤記錄或記錄抑制啟用特定的 Trident 工作流程。	""
tridentLogLayers	允許為追蹤記錄或記錄抑制啟用特定的 Trident 層。	""
tridentImage	允許完全覆蓋 Trident 的映像。	""
tridentImageTag	允許覆蓋 Trident 的映像標籤。	""
tridentProbePort	允許覆蓋 Kubernetes 存活 / 就緒探測使用的預設連接埠。	""
windows	允許在 Windows 工作節點上安裝 Trident。	false
enableForceDetach	允許啟用強制分離功能。	false
excludePodSecurityPolicy	不建立操作員 Pod 安全性原則。	false
cloudProvider	在 AKS 叢集上使用託管身分或雲端身分時，請設定為 "Azure"。在 EKS 叢集上使用雲端身分時，請設定為「AWS」。	""

選項	說明	預設
cloudIdentity	在 AKS 叢集上使用雲端身分時，設定為工作負載身分 ("azure.workload.identity/client-id: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXX")。在 EKS 叢集上使用雲端身分時，設定為 AWS IAM 角色 ("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role")。	""
iscsiSelfHealingInterval	呼叫 iSCSI 自我修復的時間間隔。	5m0s
iscsiSelfHealingWaitTime	iSCSI 自癒功能在此持續時間後會嘗試透過執行登出和後續登入來解決過時的工作階段。	7m0s
nodePrep	啟用 Trident 以準備 Kubernetes 叢集的節點，使用指定的資料儲存協定來管理磁碟區。*目前，iscsi 是唯一支援的值。*注意：從 OpenShift 4.19 開始，此功能支援的最低 Trident 版本為 25.06.1。	
enableConcurrency	啟用並行 Trident 控制器作業，以提高處理量。如需支援驅動程式的詳細資訊，請參閱連結： 控制器擴充性文件 。	錯誤
k8sAPIQPS	控制器與 Kubernetes API 伺服器通訊時所使用的每秒查詢次數（QPS）限制。突發值會根據 QPS 值自動設定。	100；選用

選項	說明	預設
resources	<p>設定 Trident 控制器、節點和 operator Pod 的 Kubernetes 資源限制和請求。您可以為每個容器和 sidecar 配置 CPU 和記憶體，以管理 Kubernetes 中的資源分配。</p> <p>有關配置資源請求和限制的更多資訊，請參閱 "Pod 和容器的資源管理"。</p> <div>  <ul style="list-style-type: none"> 請勿變更任何容器或欄位的名稱。 請勿變更縮排 - YAML 縮排對於正確剖析至關重要。 </div> <div>  <ul style="list-style-type: none"> 預設不套用任何限制 - 只有請求才有預設值，如果未指定則會自動套用。 容器名稱依其在 Pod 規格中的顯示方式列出。 Sidecar 列在每個主容器下方。 查看 TORC 的 <code>`status.CurrentInstallationParams`</code> 欄位以查看目前應用的值。 </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node:</pre>

選項	說明	預設
httpsMetrics	為 Prometheus 指標端點啟用 HTTPS。	錯誤
hostNetwork	為 Trident 控制器啟用主機網路功能。當您想要在多宿主網路中分離前端和後端流量時，這非常有用。	錯誤

了解控制器 Pod 和節點 Pod

Trident 以單一控制器 Pod 和叢集中每個工作節點上的一個節點 Pod 的形式運作。節點 Pod 必須運行在您希望掛載 Trident 磁碟區的任何主機上。

Kubernetes ["節點選取器"](#) 和 ["容忍度與污點"](#) 用於限制 Pod 在特定或首選節點上運行。使用 ControllerPlugin 和 NodePlugin，您可以指定約束和覆蓋。

- 控制器外掛程式負責處理磁碟區配置和管理，例如快照和調整大小。
- 節點外掛程式負責將儲存設備連接到節點。

使用 Helm 部署 Trident Operator（離線模式）

您可以使用 Helm 部署 Trident Operator 並安裝 Trident。此程序適用於 Trident 所需的容器映像儲存在私人登錄中的安裝。如果您沒有私有映像登錄，請使用 ["標準部署流程"](#)。

關於 Trident 26.06 的關鍵資訊

您必須閱讀以下關於 Trident 的重要資訊。

▼ 關於 Trident 的重要資訊

- Trident 現在支援 Kubernetes 1.37。請先升級 Trident，再升級 Kubernetes。
- Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 multipath.conf 檔案中設定 `find_multipaths: no` 值。

使用非多路徑配置或在 multipath.conf 檔案中使用 `find_multipaths: yes` 或 `find_multipaths: smart` 值會導致掛載失敗。Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

使用 Helm 部署 Trident Operator 並安裝 Trident

使用 Trident ["Helm Chart"](#)，您可以在一個步驟中部署 Trident 操作員並安裝 Trident。

檢查 ["安裝概述"](#) 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

除了 ["部署先決條件"](#) 之外，您還需要 ["Helm 版本 3"](#)。



在私人儲存庫中安裝 Trident 時，如果您使用 `imageRegistry` 開關指定儲存庫位置，請勿在儲

存庫路徑中使用 `/netapp/`。

步驟

1. 新增 Trident Helm 儲存庫：

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. 使用 `helm install` 並為您的部署和映像登錄位置指定名稱。您的 "Trident 與 CSI 影像" 可以位於一個登錄中，也可以位於不同的登錄中，但所有 CSI 映像必須位於同一個登錄中。在範例中，`100.2606.0` 是您要安裝的 Trident 版本。

一個登錄中的映像

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0 --set imageRegistry=<your-registry> --create-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

不同登錄中的映像

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0 --set imageRegistry=<your-registry> --set operatorImage=<your-registry>/trident-operator:26.06.0 --set tridentAutosupportImage=<your-registry>/trident-autosupport:26.06 --set tridentImage=<your-registry>/trident:26.06.0 --create-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```



如果您已經為 Trident 建立命名空間，則 `--create-namespace` 參數不會建立額外的命名空間。

您可以使用 `helm list` 來檢閱安裝詳細資料，例如名稱、命名空間、圖表、狀態、應用程式版本和修訂編號。

安裝期間傳遞組態資料

安裝過程中有兩種方法可以傳遞組態資料：

選項	說明
<code>--values</code> (或 <code>-f</code>)	指定一個包含覆蓋設定的 YAML 檔案。可以多次指定，最右側的檔案優先順序最高。
<code>--set</code>	在命令列中指定覆寫設定。

例如，若要變更 `debug` 的預設值，請執行下列命令，其中 `100.2606.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
```

```
--create-namespace --namespace trident --set tridentDebug=true
```

若要新增 nodePrep 值，請執行以下命令：

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0  
--create-namespace --namespace trident --set nodePrep={iscsi}
```

組態選項

該表和 values.yaml 檔案（屬於 Helm chart 的一部分）提供了鍵及其預設值的列表。



請勿從 values.yaml 檔案中移除預設親和性設定。如果您需要自訂親和性，請擴展預設親和性設定。

選項	說明	預設
nodeSelector	用於 Pod 分配的節點標籤	
podAnnotations	Pod 註解	
deploymentAnnotations	部署註解	
tolerations	Pod 指派的容許	

選項	說明	預設
affinity	對 pod 分配的親和性	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div>  <p>請勿從 values.yaml 檔案中移除預設親和性設定。如果您需要自訂親和性，請擴展預設親和性設定。</p> </div>
tridentControllerPluginNodeSelector	Pod 的其他節點選擇器。如需詳細資訊，請參閱 "了解控制器 Pod 和節點 Pod" 。	
tridentControllerPluginTolerations	覆寫 Kubernetes 對 Pod 的容忍度。如需詳細資訊，請參閱 "了解控制器 Pod 和節點 Pod" 。	
tridentNodePluginNodeSelector	Pod 的其他節點選擇器。如需詳細資訊，請參閱 "了解控制器 Pod 和節點 Pod" 。	
tridentNodePluginTolerations	覆寫 Kubernetes 對 Pod 的容忍度。如需詳細資訊，請參閱 "了解控制器 Pod 和節點 Pod" 。	

選項	說明	預設
imageRegistry	識別 trident-operator、trident 和其他映像的登錄。留空以接受預設值。重要：在私有儲存庫中安裝 Trident 時，如果您使用 `imageRegistry` 開關指定儲存庫位置，請勿在儲存庫路徑中使用 `/netapp/`。	""
imagePullPolicy	設定 trident-operator 的鏡像拉取策略。	IfNotPresent
imagePullSecrets	設定 trident-operator、trident 及其他映像的映像提取密鑰。	
kubeletDir	允許覆蓋 kubelet 內部狀態的主機位置。	"/var/lib/kubelet"
operatorLogLevel	允許將 Trident 運算子的日誌等級設為：trace、debug、info、warn、error 或 fatal。	"info"
operatorDebug	允許將 Trident 操作符的日誌等級設定為 debug。	true
operatorImage	允許完全覆蓋 trident-operator 的影像。	""
operatorImageTag	允許覆蓋 trident-operator 影像的標籤。	""
tridentIPv6	允許啟用 Trident 以在 IPv6 叢集中運作。	false
tridentK8sTimeout	<p>覆寫大多數 Kubernetes API 操作的預設 180 秒逾時（如果非零、則以秒為單位）。</p> <div>  <p>`tridentK8sTimeout` 參數僅適用於 Trident 安裝。</p> </div>	180
tridentHttpRequestTimeout	覆蓋 HTTP 請求的預設 90 秒逾時時間，0s 逾時時間可以設定為無限長。不允許使用負值。	"90s"
tridentSilenceAutosupport	允許禁用 Trident 定期 AutoSupport 報告。	false
tridentAutosupportImageTag	允許覆蓋 Trident AutoSupport 容器的映像標籤。	<version>
tridentAutosupportProxy	使 Trident AutoSupport 容器能夠透過 HTTP Proxy 連接到伺服器。	""
tridentLogFormat	設定 Trident 日誌格式 ((text 或 json))。	"text"

選項	說明	預設
tridentDisableAuditLog	停用 Trident 稽核記錄器。	true
tridentLogLevel	允許將 Trident 的日誌等級設為：trace、debug、info、warn、error 或 fatal。	"info"
tridentDebug	允許將 Trident 的日誌等級設為 debug。	false
tridentLogWorkflows	允許為追蹤記錄或記錄抑制啟用特定的 Trident 工作流程。	""
tridentLogLayers	允許為追蹤記錄或記錄抑制啟用特定的 Trident 層。	""
tridentImage	允許完全覆蓋 Trident 的映像。	""
tridentImageTag	允許覆蓋 Trident 的映像標籤。	""
tridentProbePort	允許覆蓋 Kubernetes 存活 / 就緒探測使用的預設連接埠。	""
windows	允許在 Windows 工作節點上安裝 Trident。	false
enableForceDetach	允許啟用強制分離功能。您可以透過與節點健康狀況檢查 (NHC) 操作符整合來自動執行強制分離程序。如需相關資訊，請參閱 "使用 Trident 自動化工具狀態應用程式的容錯移轉" 。	false
excludePodSecurityPolicy	不建立操作員 Pod 安全性原則。	false
nodePrep	<p>啟用 Trident 以準備 Kubernetes 叢集的節點，使用指定的資料儲存協定來管理磁碟區。目前，iscsi 是唯一支援的值。</p> <div>  <p>從 OpenShift 4.19 版本開始，此功能支援的最低 Trident 版本為 25.06.1。</p> </div>	

選項	說明	預設
resources	<p>設定 Trident 控制器、節點和 operator Pod 的 Kubernetes 資源限制和請求。您可以為每個容器和 sidecar 配置 CPU 和記憶體，以管理 Kubernetes 中的資源分配。</p> <p>有關配置資源請求和限制的更多資訊，請參閱 "Pod 和容器的資源管理"。</p> <div>  <ul style="list-style-type: none"> 請勿變更任何容器或欄位的名稱。 請勿變更縮排 - YAML 縮排對於正確剖析至關重要。 </div> <div>  <ul style="list-style-type: none"> 預設不套用任何限制 - 只有請求才有預設值。 容器名稱依其在 Pod 規格中的顯示方式列出。 Sidecar 列在每個主容器下方。 查看 TORC 的 <code>`status.CurrentInstallationParams`</code> 欄位以查看目前應用的值。 </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory:</pre>

自訂 Trident 操作員安裝

Trident 運算子可讓您使用 `TridentOrchestrator spec` 中的屬性自訂 Trident 安裝。如果您想要自訂的安裝超出 `TridentOrchestrator` 引數允許的範圍，請考慮使用 `tridentctl` 產生自訂 YAML 資訊清單並根據需要進行修改。

了解控制器 Pod 和節點 Pod

Trident 以單一控制器 Pod 和叢集中每個工作節點上的一個節點 Pod 的形式運作。節點 Pod 必須運行在您希望掛載 Trident 磁碟區的任何主機上。

Kubernetes ["節點選取器"](#) 和 ["容忍度與污點"](#) 用於限制 Pod 在特定或首選節點上運行。使用 `ControllerPlugin` 和 `NodePlugin`，您可以指定約束和覆蓋。

- 控制器外掛程式負責處理磁碟區配置和管理，例如快照和調整大小。
- 節點外掛程式負責將儲存設備連接到節點。

組態選項



`spec.namespace` 在 `TridentOrchestrator` 中指定，用於表示安裝 Trident 的命名空間。此參數*無法在 Trident 安裝後更新*。嘗試更新會導致 `TridentOrchestrator` 狀態變更為 `Failed`。Trident 不適用於跨命名空間遷移。

此表詳細列出了 `TridentOrchestrator` 屬性。

參數	說明	預設
<code>namespace</code>	用於安裝 Trident 的命名空間	<code>"default"</code>
<code>debug</code>	啟用 Trident 的偵錯功能	<code>false</code>
<code>enableForceDetach</code>	<code>ontap-san</code> 、 <code>ontap-san-economy</code> 、 <code>ontap-nas</code> 和 <code>ontap-nas-economy</code> 僅支援。與 Kubernetes 非優雅節點關閉 (NGNS) 配合使用，使叢集管理員能夠在節點發生故障時安全地將掛載磁碟區的工作負載遷移到新節點。如需相關資訊，請參閱 "使用 Trident 自動化具狀態應用程式的容錯移轉" 。	<code>false</code>
<code>windows</code>	設定為 <code>true</code> 可在 Windows 工作節點上進行安裝。	<code>false</code>
<code>cloudProvider</code>	在 AKS 叢集上使用託管身分或雲端身分時，請設定為 <code>"Azure"</code> 。在 EKS 叢集上使用雲端身分時，設定為 <code>"AWS"</code> 。在 GKE 叢集上使用雲端身分時，設定為 <code>"GCP"</code> 。	<code>" "</code>

參數	說明	預設
cloudIdentity	在 AKS 叢集上使用雲端身分時，設定為工作負載身分 ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx")。在 EKS 叢集上使用雲端身分時，設定為 AWS IAM 角色 ("eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role")。在 GKE 叢集上使用雲端身分時，設定為雲端身分 ("iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com")。	""
IPv6	透過 IPv6 安裝 Trident	錯誤
k8sTimeout	Kubernetes 操作逾時。  `k8sTimeout` 參數僅適用於 Trident 安裝。	180sec
silenceAutosupport	不要自動將 autosupport 套件組合傳送至 NetApp	false
autosupportImage	Autosupport Telemetry 的容器映像	"netapp/trident-autosupport10"
autosupportProxy	用於發送 Autosupport 遙測資料的 Proxy 位址 / 連接埠	"http://proxy.example.com:8888"
uninstall	用於卸載 Trident 的旗標	false
logFormat	要使用的 Trident 日誌記錄格式 [text,json]	"text"
tridentImage	要安裝的 Trident 映像	"netapp/trident:26.06"
imageRegistry	內部登錄路徑，格式為 <registry FQDN>[:port][/subpath]	"registry.k8s.io"
kubeletDir	主機上 kubelet 目錄的路徑	"/var/lib/kubelet"
wipeout	若要徹底移除 Trident 所需刪除的資源清單	
imagePullSecrets	從內部登錄檔提取映像的密碼	
imagePullPolicy	設定 Trident 運算子的映像拉取原則。有效值為： Always 一律拉取映像。 IfNotPresent 僅在節點上尚未存在映像時才拉取映像。 Never 永不拉取映像。	IfNotPresent
controllerPluginNodeSelector	Pod 的其他節點選擇器。格式與 `pod.spec.nodeSelector` 相同。	無預設值；選用
controllerPluginTolerations	覆寫 Kubernetes 對 Pod 的容忍度。遵循與 `pod.spec.Tolerations` 相同的格式。	無預設值；選用
nodePluginNodeSelector	Pod 的其他節點選擇器。格式與 `pod.spec.nodeSelector` 相同。	無預設值；選用
nodePluginTolerations	覆寫 Kubernetes 對 Pod 的容忍度。遵循與 `pod.spec.Tolerations` 相同的格式。	無預設值；選用

參數	說明	預設
nodePrep	<p>啟用 Trident 以準備 Kubernetes 叢集的節點，使用指定的資料儲存協定來管理磁碟區。目前，iscsi 是唯一支援的值。</p> <div>  <p>從 OpenShift 4.19 版本開始，此功能支援的最低 Trident 版本為 25.06.1。</p> </div>	
k8sAPIQPS	控制器與 Kubernetes API 伺服器通訊時所使用的每秒查詢次數（QPS）限制。突發值會根據 QPS 值自動設定。	100；選用
enableConcurrency	啟用並行 Trident 控制器作業，以提高處理量。如需支援驅動程式的詳細資訊，請參閱連結： 控制器擴充性文件 。	resources

參數	說明	預設
<p>設定 Trident 控制器和節點 Pod 的 Kubernetes 資源限制和請求。您可以為每個容器和 sidecar 配置 CPU 和記憶體，以管理 Kubernetes 中的資源分配。</p> <p>有關配置資源請求和限制的更多資訊，請參閱 "Pod 和容器的資源管理"。</p> <div> <div>⚠</div> <ul style="list-style-type: none"> 請勿變更任何容器或欄位的名稱。 請勿變更縮排 - YAML 縮排對於正確剖析至關重要。 </div> <div> <div>i</div> <ul style="list-style-type: none"> 預設不套用任何限制 - 只有請求才有預設值，如果未指定則會自動套用。 容器名稱依其在 Pod 規格中的顯示方式列出。 Sidecar 列在每個主容器下方。 <p>查看 TORC</p> </div>	<pre> resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node: </pre>	<p>httpsMetrics</p>

參數	說明	預設
為 Prometheus 指標端點啟用 HTTPS。	錯誤	hostNetwork



有關格式化 pod 參數的更多資訊，請參閱 ["將 Pod 指派給節點"](#)。

範例組態

您可以在定義[\[組態選項\]](#)時使用 `TridentOrchestrator` 中的屬性，以自訂您的安裝。

▼ 基本自訂組態

該範例是在運行 `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` 命令後創建的，代表一個基本的自訂安裝：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

▼ 節點選擇器

此範例使用節點選擇器安裝 Trident。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

▼ Windows 工作節點

執行 `cat deploy/crds/tridentorchestrator_cr.yaml` 指令後建立的此範例會在 Windows 工作節點上安裝 Trident。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

▼ AKS 叢集上的託管身分

本範例安裝 Trident 以在 AKS 叢集上啟用託管身分識別。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

▼ AKS 叢集上的雲端身分

本範例在 AKS 叢集上安裝 Trident 以用於雲端身分。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxxx'
```

▼ EKS 叢集上的雲端身分

本範例在 AKS 叢集上安裝 Trident 以用於雲端身分。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
```

```
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

▼ GKE 的雲端身分

本範例在 GKE 叢集上安裝 Trident 以用於雲端身分。

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

▼ Trident 控制器和 Trident Linux 節點 Pod 的 Kubernetes 資源請求和限制配置

本範例為 Trident 控制器和 Trident Linux 節點 Pod 設定 Kubernetes 資源請求和限制。



免責聲明：本範例中提供的請求和限制值僅用於示範目的。請根據您的環境和工作負載需求調整這些值。

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
```

```
    limits:
      cpu: 200m
      memory: 256Mi
# sidecars
csi-provisioner:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-attacher:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-resizer:
  requests:
    cpu: 3m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-snapshotter:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
trident-autosupport:
  requests:
    cpu: 1m
    memory: 30Mi
  limits:
    cpu: 50m
    memory: 128Mi
node:
  linux:
    trident-main:
      requests:
        cpu: 10m
        memory: 60Mi
      limits:
```

```

        cpu: 200m
        memory: 256Mi
# sidecars
node-driver-registrar:
  requests:
    cpu: 1m
    memory: 10Mi
  limits:
    cpu: 50m
    memory: 32Mi

```

▼ **Kubernetes** 資源請求和限制配置，適用於 **Trident** 控制器以及 **Trident Windows** 和 **Linux** 節點 **Pod**

本範例為 Trident 控制器和 Trident Windows 和 Linux 節點 Pod 設定 Kubernetes 資源請求和限制。



免責聲明：本範例中提供的請求和限制值僅用於示範目的。請根據您的環境和工作負載需求調整這些值。

```

apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
    csi-provisioner:
      requests:
        cpu: 2m
        memory: 20Mi
      limits:
        cpu: 100m
        memory: 64Mi
    csi-attacher:

```

```
    requests:
      cpu: 2m
      memory: 20Mi
    limits:
      cpu: 100m
      memory: 64Mi
csi-resizer:
  requests:
    cpu: 3m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-snapshotter:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
trident-autosupport:
  requests:
    cpu: 1m
    memory: 30Mi
  limits:
    cpu: 50m
    memory: 128Mi
node:
  linux:
    trident-main:
      requests:
        cpu: 10m
        memory: 60Mi
      limits:
        cpu: 200m
        memory: 256Mi
    # sidecars
    node-driver-registrar:
      requests:
        cpu: 1m
        memory: 10Mi
      limits:
        cpu: 50m
        memory: 32Mi
  windows:
    trident-main:
```

```

requests:
  cpu: 6m
  memory: 40Mi
limits:
  cpu: 200m
  memory: 128Mi
# sidecars
node-driver-registrar:
  requests:
    cpu: 6m
    memory: 40Mi
  limits:
    cpu: 100m
    memory: 128Mi
liveness-probe:
  requests:
    cpu: 2m
    memory: 40Mi
  limits:
    cpu: 50m
    memory: 64Mi

```

使用 tridentctl 安裝

使用 tridentctl 安裝 Trident

您可以使用 `tridentctl` 安裝 Trident。此程序適用於 Trident 所需的容器映像儲存在私有登錄中或未儲存的安裝。若要自訂 `tridentctl` 部署，請參閱["自訂 tridentctl 部署"](#)。

關於 **Trident10** 的關鍵資訊

您必須閱讀以下關於 **Trident** 的重要資訊。

▼ 關於 Trident 的重要資訊

- Trident 現在支援 Kubernetes 1.27。請先升級 Trident，再升級 Kubernetes。
- Trident 嚴格強制要求在 SAN 環境中使用多路徑配置，建議在 multipath.conf 檔案中設定 `find_multipaths: no` 值。

使用非多路徑配置或在 multipath.conf 檔案中使用 `find_multipaths: yes` 或 `find_multipaths: smart` 值會導致掛載失敗。Trident 自 21.07 版本起就建議使用 `find_multipaths: no`。

使用以下方式安裝 Trident **tridentctl**

檢查 ["安裝概述"](#) 以確保滿足安裝先決條件並為您的環境選擇了正確的安裝選項。

開始之前

在開始安裝之前，請登入 Linux 主機並驗證它是否正在管理運作中的 "支援的 Kubernetes 叢集"，以及您是否擁有必要的權限。



使用 OpenShift 時，請在以下所有範例中使用 `oc` 而非 `kubectl`，並先執行 `oc login -u system:admin` 或 `oc login -u kube-admin` 以 **system:admin** 身分登入。

▼ Details

1. 驗證您的 Kubernetes 版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. 驗證您是否可以啟動使用 Docker Hub 映像的 pod，並透過 pod 網路存取您的儲存系統：

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
ping <management IP>
```

步驟 1：下載 Trident 安裝程式套件

Trident 安裝程式套件會建立一個 Trident pod，配置用於維護其狀態的 CRD 對象，並初始化 CSI sidecar 以執行諸如配置磁碟區和將磁碟區附加到叢集主機等操作。從["GitHub 上的_資產_部分"](#)下載並解壓縮最新版本的 Trident 安裝程式。將範例中的 `<trident-installer-XX.XX.X.tar.gz>` 更新為您選擇的 Trident 版本。

```
wget https://github.com/NetApp/trident/releases/download/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

步驟 2：安裝 Trident

透過執行 `tridentctl install` 指令，在所需的命名空間中安裝 Trident。您可以新增其他參數來指定映像登錄位置。

標準模式

```
./tridentctl install -n trident
```

一個登錄中的映像

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.06 --trident
-image <your-registry>/trident:26.06.0
```

不同登錄中的映像

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.06 --trident
-image <your-registry>/trident:26.06.0
```

您的安裝狀態應該類似這樣。

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=26.10.0
INFO Trident installation succeeded.
....
```

驗證安裝

您可以使用 `pod` 建立狀態或 `tridentctl` 來驗證您的安裝。

使用 `pod` 建立狀態

您可以透過檢視已建立的 `pod` 的狀態來確認 Trident 安裝是否已完成：

```
kubectl get pods -n trident
```

NAME	READY	STATUS	RESTARTS	AGE
trident-controller-679648bd45-cv2mx	6/6	Running	0	5m29s



如果安裝程式未成功完成或 `trident-controller-<generated id>` (`trident-csi-<generated id>` 在 23.01 之前的版本中未顯示 **Running** 狀態，則表示平台未安裝。請使用 `-d` 來[開啟偵錯模式](#)並排查問題。

使用 `tridentctl`

您可以使用 `tridentctl` 檢查已安裝的 Trident 版本。

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

範例組態

以下範例提供使用 `tridentctl` 安裝 Trident 的範例組態。

▼ Windows 節點

要讓 Trident 能夠在 Windows 節點上運作：

```
tridentctl install --windows -n trident
```

▼ 強制分離

如需相關資訊，請參閱 ["使用 Trident 自動化工具狀態應用程式的容錯移轉"](#)。

```
tridentctl install --enable-force-detach=true -n trident
```

▼ 啟用並發的 Trident 控制器操作

若要啟用並行 Trident 控制器作業以提高處理量，請在安裝期間新增 `--enable-concurrency` 選項，如本範例所示。如需支援驅動程式的詳細資訊，請參閱連結：[控制器擴充性文件](#)。

```
tridentctl install --enable-concurrency -n trident
```

自訂 Trident `tridentctl` 安裝

您可以使用 Trident 安裝程式來自訂安裝。

了解安裝程式

Trident 安裝程式可讓您自訂屬性。例如，如果您已將 Trident 映像複製到私人儲存庫，則可以使用 `--trident-image` 指定映像名稱。如果您已將 Trident 映像以及所需的 CSI sidecar 映像複製到私人儲存庫，則最好使用 `--image-registry` 開關指定該儲存庫的位置，其格式為 `<registry FQDN>[:port]`。



在私人儲存庫中安裝 Trident 時，如果您使用 `--image-registry` 開關指定儲存庫位置，請勿在儲存庫路徑中使用 `/netapp/`。例如：`./tridentctl install --image-registry <image-registry> -n <namespace>`

如果您使用的是 Kubernetes 發行版，其中 `kubelet` 將其資料保存在與通常路徑 `/var/lib/kubelet` 不同的路徑上，則可以使用 `--kubelet-dir` 指定備用路徑。

如果您需要對安裝進行超出安裝程式參數允許範圍的自訂設置，您也可以自訂部署檔案。使用 `--generate-custom-yaml` 參數會在安裝程式的 `setup` 目錄中建立以下 YAML 檔案：

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml` *

產生這些檔案後，您可以根據需要修改它們，然後使用 `--use-custom-yaml` 來安裝您的自訂部署。

```
./tridentctl install -n trident --use-custom-yaml
```

使用 OpenShift 認證的操作人員進行安裝

使用 OpenShift OperatorHub 安裝 Trident

如果您使用 Red Hat OpenShift，則可以使用 Red Hat 認證的操作員來安裝 NetApp Trident。請依照以下步驟從 Red Hat OpenShift Container Platform 安裝 Trident。

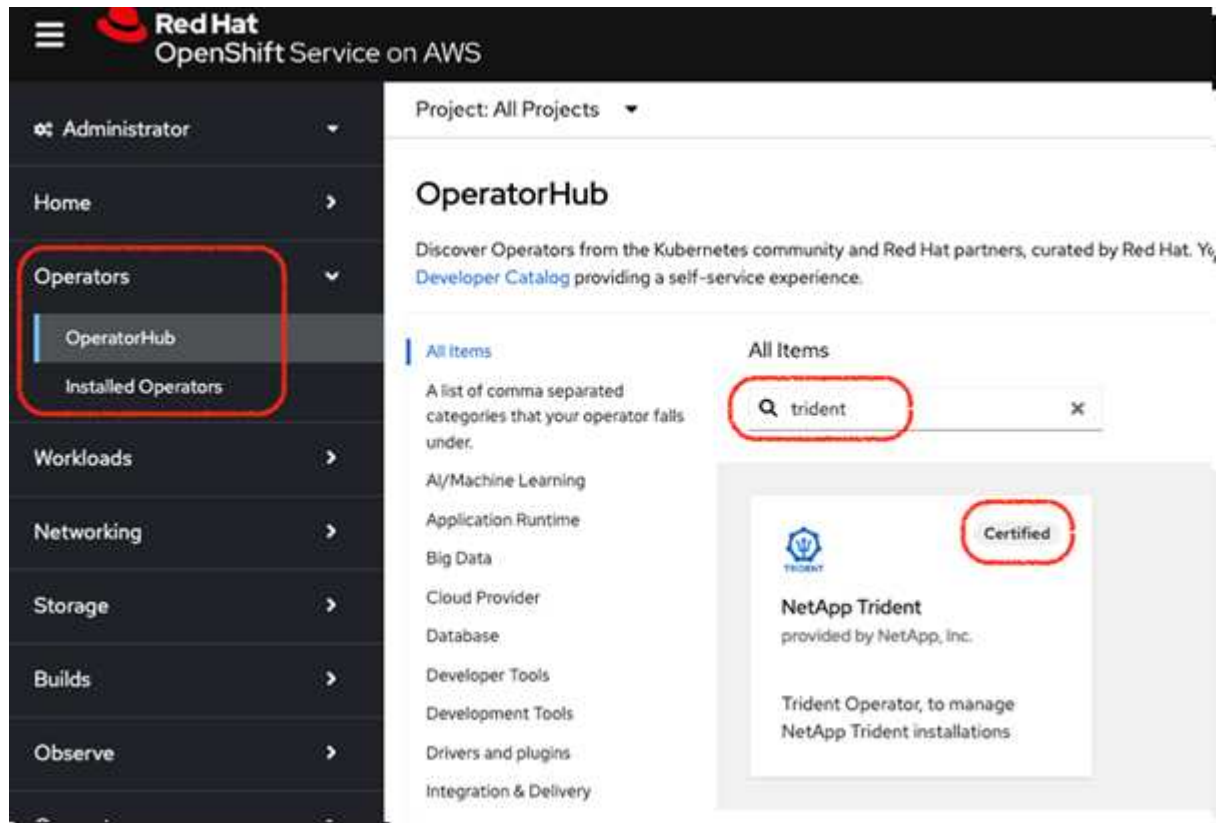
開始之前

在開始安裝之前，["為 Trident 安裝準備環境"](#)。

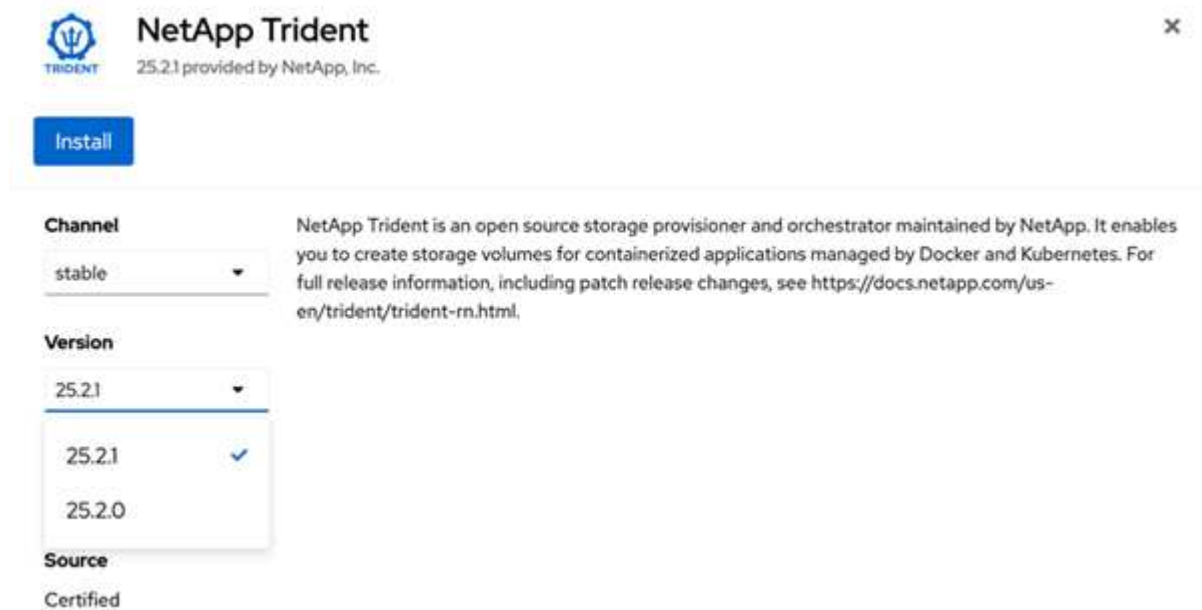
找到並安裝 Trident 操作員

步驟

1. 導航至 OpenShift OperatorHub 並搜尋 NetApp Trident。



2. 點選 **NetApp Trident** 開啟安裝設定。
3. 選擇所需選項，然後按一下 **Install** 以開啟 Operator 配置。



請確保選擇最新版本的 Operator。

4. 保留所有參數不變，然後點選 **Install**。

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel * ⓘ

stable

Version *

25.2.1

Installation mode *

- ☒ All namespaces on the cluster (default)
Operator will be available in all Namespaces.
- ☐ A specific namespace on the cluster
This mode is not supported by this Operator

Installed Namespace *

PR openshift-operators

Update approval * ⓘ

- ☒ Automatic
- ☐ Manual

 **NetApp Trident**
provided by NetApp, Inc.

Provided APIs

 **Trident Orchestrator**
Used to deploy NetApp Trident.

 **Trident Configurator**
Automates AWS FSxN backend configuration

5. 點擊 **View Operator** 查看 Operator 的詳細資料。

 **NetApp Trident**
25.2.1 provided by NetApp, Inc.

Details YAML Subscription Events All instances Trident Orchestrator Trident Configurator

Provided APIs

 **Trident Orchestrator**
Used to deploy NetApp Trident.

 [Create instance](#)

 **Trident Configurator**
Automates AWS FSxN backend configuration

 [Create instance](#)

6. 點擊 **YAML view** 並將以下內容貼到表單中：

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
```

```
debug: false
nodePrep:
- iscsi
imageRegistry: ''
k8sTimeout: 180s
namespace: trident
silenceAutosupport: false
```



UI 提供預設範例。您可以直接編輯，而無需複製完整組態。

選用：啟用並行

若要啟用並行性，請在規格中新增下列欄位：

```
enableConcurrency: true
```



- Red Hat Enterprise Linux CoreOS (RHCOS) 未啟用和設定 iSCSI。
- 您可以新增 `nodePrep` 參數，以在所有 OpenShift 工作節點上設定和啟用 iSCSI 和多路徑服務。
- 從 OpenShift 4.19 版本開始，此功能支援的最低 Trident 版本為 25.06.1。

1. 點選 **Create**；Trident Orchestrator 將會完全安裝。

Installed Operators > Operator details

NetApp Trident
25.21 provided by NetApp, Inc.

Actions

Details YAML Subscription Events All instances **Trident Orchestrator** Trident Configurator

TridentOrchestrators [Create TridentOrchestrator](#)

Name Search by name...

Name	Kind	Status	Labels	Last updated
trident	TridentOrchestrator	Status: Installed	No labels	Apr 6, 2025, 8:02 PM

卸載 Trident 操作員

步驟

1. 從已安裝的運算子清單中選擇 Trident 運算子。
2. 選擇是否要從運算子中刪除所有操作數實例。



如果您沒有選取「從該運算子中刪除所有運算元實例」複選框，則不會卸載 Trident。

3. 按一下 **Uninstall**。

切換到 OpenShift 認證的 Trident 操作員

您可以從社群版 Operator、基於 Helm 的安裝或手動部署的 Operator 切換到 Red Hat OpenShift 認證的 Trident Operator。每種方法的過程都包括卸載現有 Operator，然後使用 OperatorHub 安裝認證的 Operator。

開始之前

在開始安裝之前，"[為 Trident 安裝準備環境](#)"。

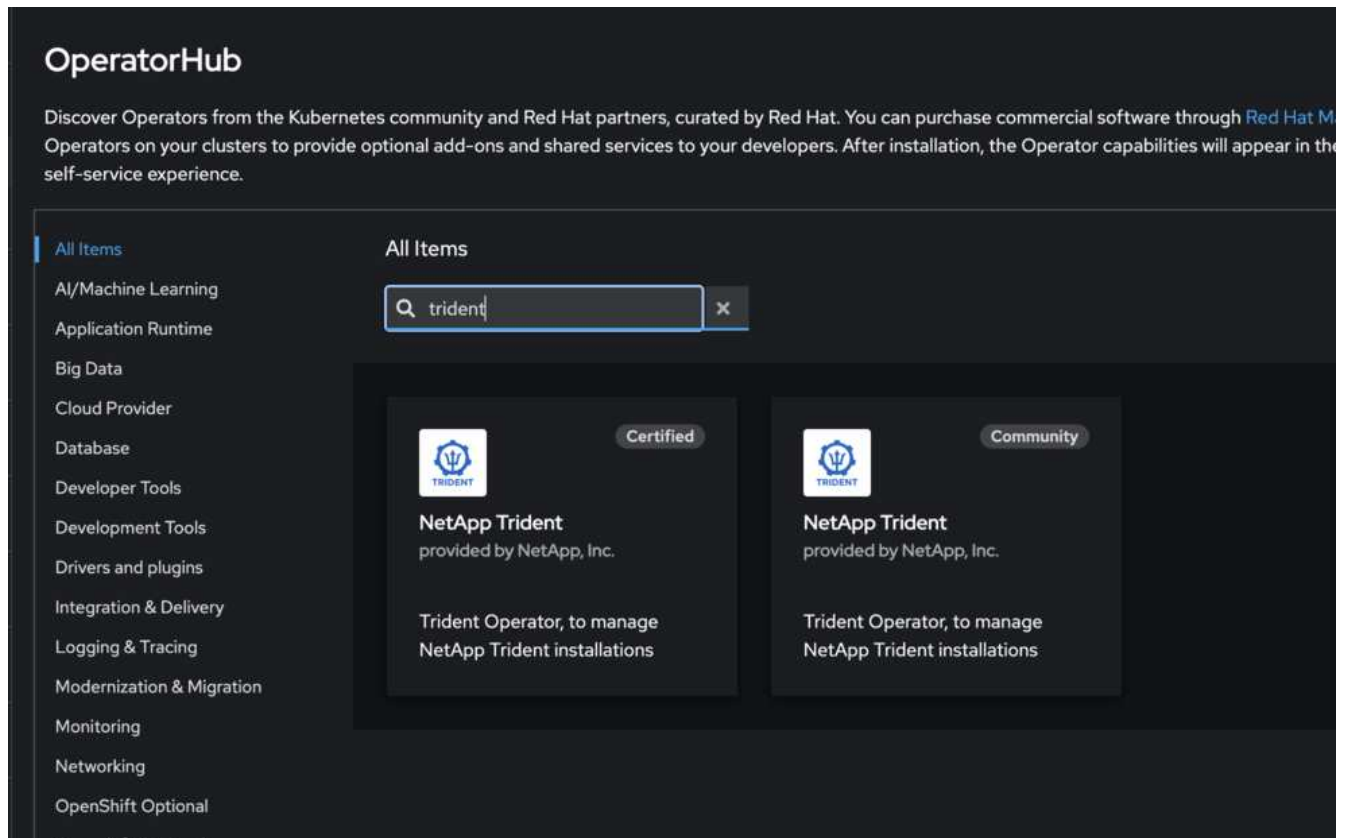


在卸載過程中請勿刪除 `TridentOrchestrator` 自訂資源 (CR)。
`TridentOrchestrator` CR 會保留您的後端和儲存類別配置，這在安裝認證操作員後是必要的。

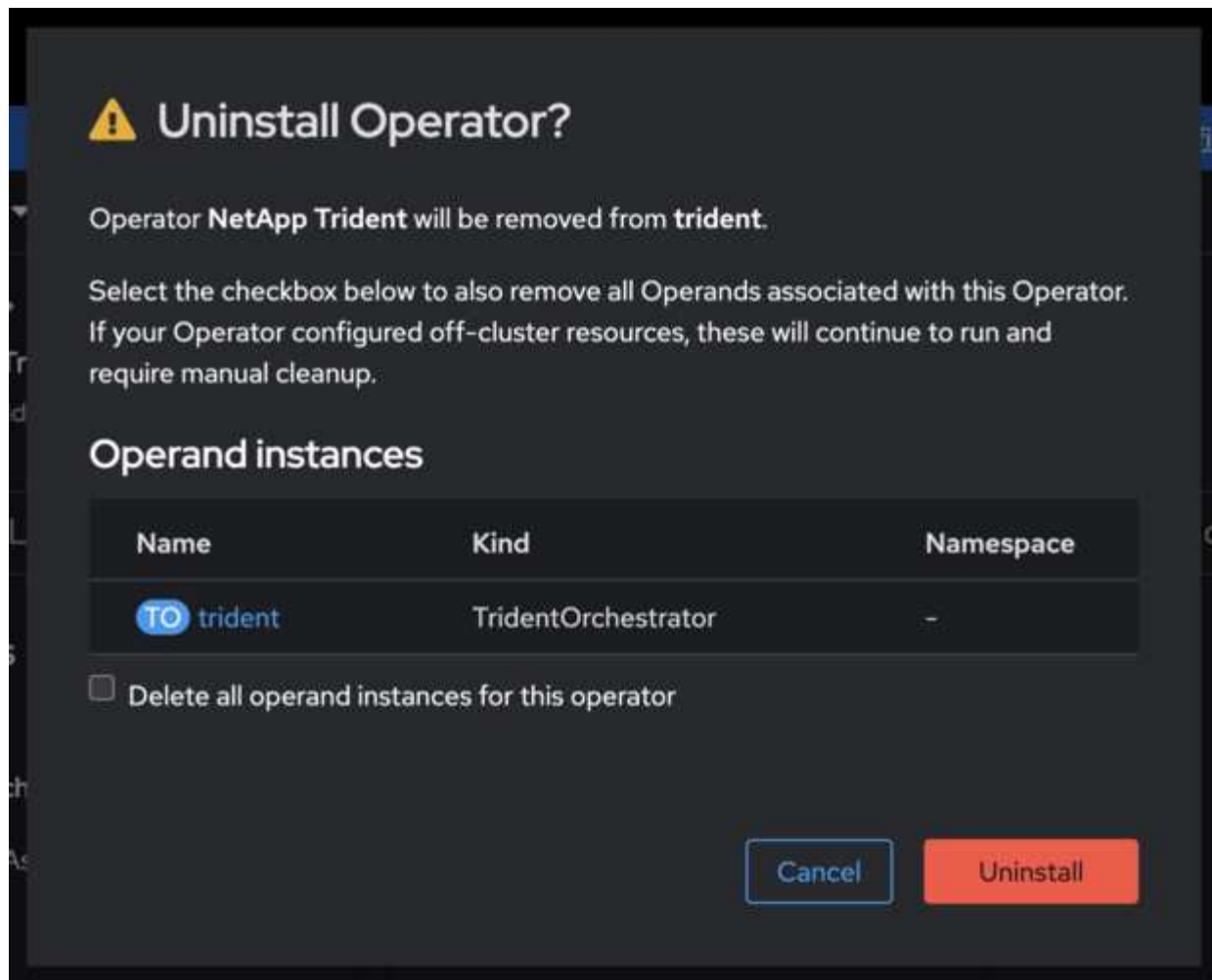
從社群營運商切換

步驟

1. 使用 OpenShift 控制台導航到 OperatorHub。



2. 找到 NetApp Trident 社群操作員。



不要選擇 **Delete all operand instances from this operator**。

3. 按一下 **Uninstall**。
4. 卸載完成後，繼續執行 [安裝 OpenShift 認證操作員](#)。

從基於 **Helm** 的操作員安裝切換

步驟

1. 列出您的 Trident 安裝所使用的 Helm 版本：

```
helm ls -n trident
```

2. 解除安裝 Helm 版本：

```
helm uninstall <release-name> -n trident
```

3. 卸載完成後，繼續執行 [安裝 OpenShift 認證操作員](#)。

從手動部署的 **operator** 切換

如果您是透過 `bundle.yaml` 安裝程式套件中的檔案手動部署操作員來安裝 Trident，請透過刪除相同的資訊清單來移除。

步驟

1. 使用套件組合資訊清單刪除操作員部署：

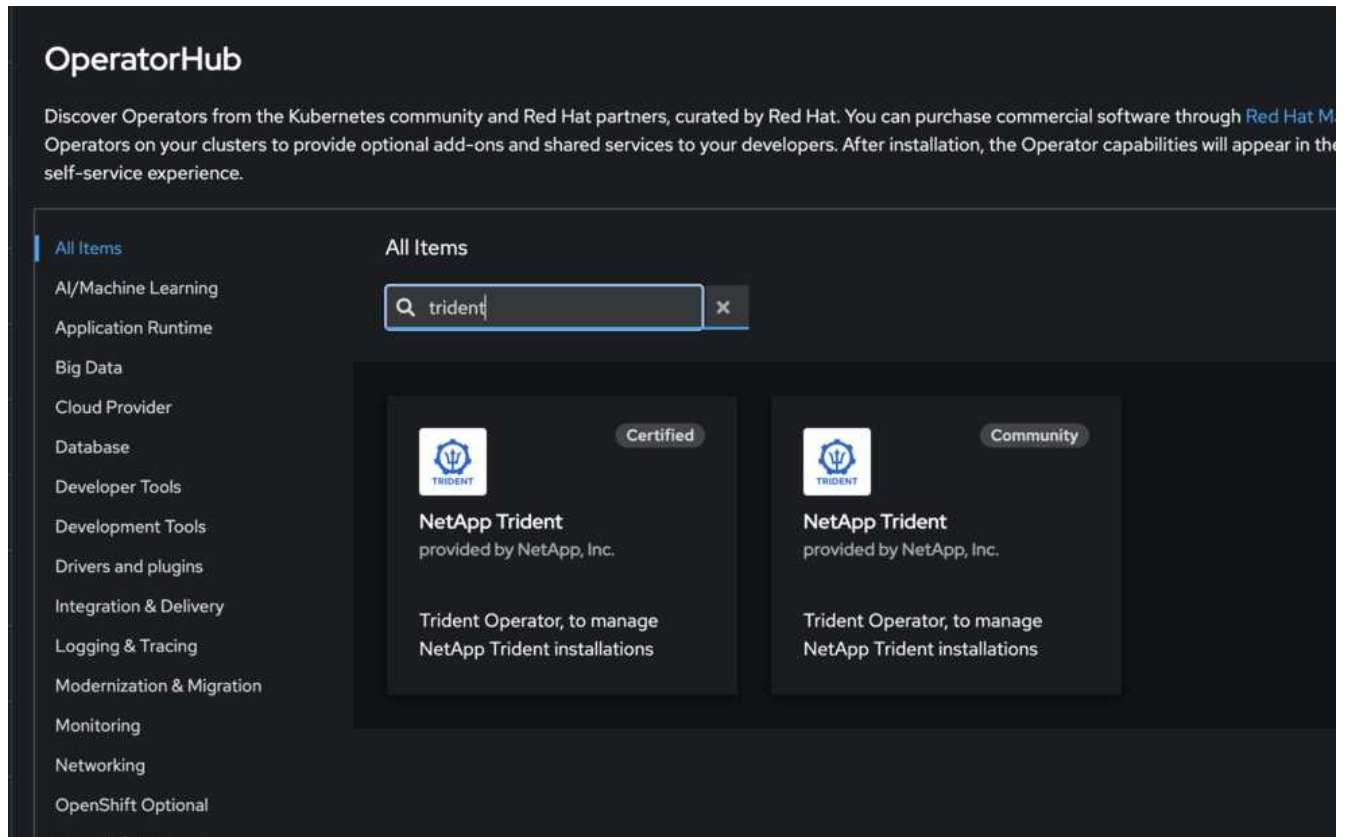
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. 卸載完成後，繼續執行 [安裝 OpenShift 認證操作員](#)。

安裝 **OpenShift** 認證操作員

步驟

1. 導航至 Red Hat OperatorHub。
2. 搜尋並選擇 NetApp Trident Operator。



3. 請依照螢幕上的指示安裝 operator。

驗證

- 檢查控制台中的 OperatorHub，確保新認證的操作員已成功安裝。

版權資訊

Copyright © 2026 NetApp, Inc. 版權所有。台灣印製。非經版權所有人事先書面同意，不得將本受版權保護文件的任何部分以任何形式或任何方法（圖形、電子或機械）重製，包括影印、錄影、錄音或儲存至電子檢索系統中。

由 NetApp 版權資料衍伸之軟體必須遵守下列授權和免責聲明：

此軟體以 NETAPP「原樣」提供，不含任何明示或暗示的擔保，包括但不限於有關適售性或特定目的適用性之擔保，特此聲明。於任何情況下，就任何已造成或基於任何理論上責任之直接性、間接性、附隨性、特殊性、懲罰性或衍生性損害（包括但不限於替代商品或服務之採購；使用、資料或利潤上的損失；或企業營運中斷），無論是在使用此軟體時以任何方式所產生的契約、嚴格責任或侵權行為（包括疏忽或其他）等方面，NetApp 概不負責，即使已被告知有前述損害存在之可能性亦然。

NetApp 保留隨時變本文所述之任何產品的權利，恕不另行通知。NetApp 不承擔因使用本文所述之產品而產生的責任或義務，除非明確經過 NetApp 書面同意。使用或購買此產品並不會在依據任何專利權、商標權或任何其他 NetApp 智慧財產權的情況下轉讓授權。

本手冊所述之產品受到一項（含）以上的美國專利、國外專利或申請中專利所保障。

有限權利說明：政府機關的使用、複製或公開揭露須受 DFARS 252.227-7013（2014 年 2 月）和 FAR 52.227-19（2007 年 12 月）中的「技術資料權利 - 非商業項目」條款 (b)(3) 小段所述之限制。

此處所含屬於商業產品和 / 或商業服務（如 FAR 2.101 所定義）的資料均為 NetApp, Inc. 所有。根據本協議提供的所有 NetApp 技術資料和電腦軟體皆屬於商業性質，並且完全由私人出資開發。美國政府對於該資料具有非專屬、非轉讓、非轉授權、全球性、有限且不可撤銷的使用權限，僅限於美國政府為傳輸此資料所訂合約所允許之範圍，並基於履行該合約之目的方可使用。除非本文另有規定，否則未經 NetApp Inc. 事前書面許可，不得逕行使用、揭露、重製、修改、履行或展示該資料。美國政府授予國防部之許可權利，僅適用於 DFARS 條款 252.227-7015(b)（2014 年 2 月）所述權利。

商標資訊

NETAPP、NETAPP 標誌及 <http://www.netapp.com/TM> 所列之標章均為 NetApp, Inc. 的商標。文中所涉及的所有其他公司或產品名稱，均為其各自所有者的商標，不得侵犯。