使用 **Trident** 操作員進行安裝

Trident

NetApp
July 01, 2026

目錄

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 26.02 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 1.35。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。

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/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.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。

  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 **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:26.02
    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:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    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 `spec.uninstall=true`。
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 `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 |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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 1.35。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。

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-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.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:26.02.0`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:26.02.0` 如果您的所有映像都位於同一個登錄中。
- `image: <your-registry>/netapp/trident-operator:26.02.0` 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:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

不同登錄中的映像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

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:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    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:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    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 `spec.uninstall=true`。
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 `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 |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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 25.10 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 1.35。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..0` 是您要安裝的 Trident 版本。

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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
`--values`（或 `-f`）	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。
`--set`	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 `100.2602.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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	

選項	說明	預設

`affinity`	對 pod 分配的親和性	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		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。

`tridentControllerPluginNodeSelector`	Pod 的其他節點選擇器。詳情請參閱 了解控制器 Pod 和節點 Pod。	
`tridentControllerPluginTolerations`	覆寫 Kubernetes 對 Pod 的容忍度。詳情請參閱 了解控制器 Pod 和節點 Pod。	
`tridentNodePluginNodeSelector`	Pod 的其他節點選擇器。詳情請參閱 了解控制器 Pod 和節點 Pod。	
`tridentNodePluginTolerations`	覆寫 Kubernetes 對 Pod 的容忍度。詳情請參閱 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/`。	""

選項	說明	預設
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
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"

選項	說明	預設
`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」。	`""`
`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-xxxxxxxxxxx"）。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`

選項	說明	預設
`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`	支援並發 Trident 控制器操作，以提高處理量。 技術預覽：此功能為實驗性功能，目前支援使用 ONTAP-NAS（僅限 NFS）和 ONTAP-SAN（統一 ONTAP 9 中的 NVMe）驅動程式的有限平行工作流程，此外還有 ONTAP-SAN 驅動程式（統一 ONTAP 9 中的 iSCSI 和 FCP 協定）的現有技術預覽版。	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`	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。
	
	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"。
	
	• 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。
	
	• 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 `status.CurrentInstallationParams `欄位以查看目前應用的值。
		<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:
		    linux:
		      trident-main:</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 26.02** 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 1.35。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， `100.2602.0`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
100.2602.0 --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
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --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
`--values`（或 `-f`）	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。

選項	說明
`--set`	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 `100.2602.0` 是您要安裝的 Trident 版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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 100.2602.0
--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	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		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。

`tridentControllerPluginNodeSelector`	Pod 的其他節點選擇器。詳情請參閱 "了解控制器 Pod 和節點 Pod"。	
`tridentControllerPluginTolerations`	覆寫 Kubernetes 對 Pod 的容忍度。詳情請參閱 "了解控制器 Pod 和節點 Pod"。	
`tridentNodePluginNodeSelector`	Pod 的其他節點選擇器。詳情請參閱 "了解控制器 Pod 和節點 Pod"。	
`tridentNodePluginTolerations`	覆寫 Kubernetes 對 Pod 的容忍度。詳情請參閱 "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	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）。 `tridentK8sTimeout` 參數僅適用於 Trident 安裝。	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。	""

選項	說明	預設
`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`。	`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`	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。 從 OpenShift 4.19 版本開始，此功能支援的最低 Trident 版本為 25.06.1。	

選項	說明	預設

`resources`	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。
	
	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"。
	
	 
	• 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。
	
	 
	• 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 `status.CurrentInstallationParams `欄位以查看目前應用的值。
		`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:`
		`  linux:`

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
`namespace`	用於安裝 Trident 的命名空間	`"default"`
`debug`	啟用 Trident 的偵錯功能	`false`
`enableForceDetach`	`ontap-san`、`ontap-san-economy`、`ontap-nas` 和 `ontap-nas-economy` 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"。	`false`
`windows`	設定為 `true`可在 Windows 工作節點上進行安裝。	`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，設定為 `"AWS"`。在 GKE 叢集上使用雲端身分時，設定為 `"GCP"`。	`""`


```
        trident-main:
          requests:
            cpu: 10m
            memory:
60Mi
          limits:
            cpu:
            memory:
        node-driver-
registrar:
          requests:
            cpu: 1m
            memory:
10Mi
          limits:
            cpu:
            memory:
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory:
40Mi
          limits:
            cpu:
            memory:
    operator:
      requests:
        cpu: 10m
        memory: 40Mi
      limits:
```

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-xxxxxxxxxxx"）。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:26.02"`
`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

參數	說明	預設
`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
`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。 從 OpenShift 4.19 版本開始，此功能支援的最低 Trident 版本為 25.06.1。	
`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`	支援並發 Trident 控制器操作，以提高處理量。 技術預覽：此功能為實驗性功能，目前支援使用 ONTAP-NAS（僅限 NFS）和 ONTAP-SAN（統一 ONTAP 9 中的 NVMe）驅動程式的有限平行工作流程，此外還有 ONTAP-SAN 驅動程式（統一 ONTAP 9 中的 iSCSI 和 FCP 協定）的現有技術預覽版。	錯誤

參數	說明	預設

`resources`	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。
	
	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"。
	
	• 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。
	
	• 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 `status.CurrentInstallationParams` 欄位以查看目前應用的值。
		`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:`

參數	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

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
```

cpu:

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-xxxxxxxxxxx'
```

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
```