



使用**Trident**操作員安裝 Astra Trident

NetApp
April 16, 2024

目錄

使用Trident操作員安裝	1
手動部署Trident運算子（標準模式）	1
手動部署Trident運算子（離線模式）	6
使用Helm部署Trident運算子（標準模式）	12
使用Helm部署Trident運算子（離線模式）	16
自訂Trident操作員安裝	20

使用Trident操作員安裝

手動部署Trident運算子（標準模式）

您可以手動部署Trident運算子來安裝Astra Trident。此程序適用於未將Astra Trident所需的容器映像儲存在私有登錄中的安裝。如果您有私有映像登錄、請使用 ["離線部署程序"](#)。

Astra Trident的重要資訊23.01

您必須閱讀下列有關Astra Trident的重要資訊。

關於Astra Trident-功能 的資訊

- Kubernetes 1.26現在已在Trident中受到支援。升級Kubernetes之前先升級Trident。
- Astra Trident在SAN環境中嚴格執行多重路徑組態的使用、建議的值為 `find_multipaths: no` 在多重路徑.conf檔案中。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

手動部署Trident運算子並安裝Trident

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

在開始安裝之前、請先登入Linux主機、然後確認它正在管理正常運作的 ["支援的Kubernetes叢集"](#) 而且您擁有必要的權限。



使用OpenShift、使用 `oc` 而非 `kubectl` 在以下所有範例中、請先執行*系統：admin*登入 `oc login -u system:admin` 或 `oc login -u kube-admin`。

1. 驗證Kubernetes版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

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

3. 確認您可以啟動使用Docker Hub映像的Pod、並透過Pod網路連線至儲存系統：

```
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

Astra Trident安裝程式套件包含部署Trident操作員及安裝Astra Trident所需的一切。從下載並擷取Trident安裝程式的最新版本 "[GitHub的_Assets區段](#)"。

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

步驟2：建立 TridentOrchestrator 客戶需求日

建立 TridentOrchestrator 自訂資源定義（CRD）。您將建立 TridentOrchestrator 稍後再自訂資源。請使用中適當的CRD Y反 洗錢版本 deploy/crds 以建立 TridentOrchestrator 客戶需求日

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

步驟3：部署Trident運算子

Astra Trident安裝程式提供一個套件檔案、可用來安裝運算子及建立相關的物件。套裝組合檔案是使用預設組態部署操作員及安裝Astra Trident的簡易方法。

- 對於執行Kubernetes 1.24或更低層級的叢集、請使用 bundle_pre_1_25.yaml。
- 對於執行Kubernetes 1.25或更新版本的叢集、請使用 bundle_post_1_25.yaml。

Trident安裝程式會在中部署操作員 trident 命名空間。如果是 trident 命名空間不存在、請使用 kubectl

apply -f deploy/namespace.yaml 以建立。

步驟

1. 建立資源並部署營運者：

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



可在非的命名空間中部署運算子 trident 命名空間、更新 serviceaccount.yaml、clusterrolebinding.yaml 和 operator.yaml 並使用產生套裝組合檔案 kustomization.yaml：

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

2. 驗證是否已部署操作員。

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

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
trident-operator	1/1	1	1	3m



Kubernetes叢集中只應有*一個運算子執行個體*。請勿建立Trident營運者的多個部署。

步驟4：建立 TridentOrchestrator 並安裝Trident

您現在可以建立 TridentOrchestrator 並安裝Astra Trident。您也可以選擇 ["自訂您的Trident安裝"](#) 使用中的屬性 TridentOrchestrator 規格

```

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
Status:
  Current Installation Params:
    IPv6:          false
    Autosupport Hostname:
    Autosupport Image:      netapp/trident-autosupport:23.01
    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:23.01.1
  Message:          Trident installed  Namespace:
trident
  Status:           Installed
  Version:          v23.01.1
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

驗證安裝的方法有多種。

使用 TridentOrchestrator 狀態

狀態 TridentOrchestrator 指出安裝是否成功、並顯示安裝的Trident版本。安裝期間的狀態 TridentOrchestrator 變更來源 Installing 至 Installed。如果您觀察到 Failed 狀態、而且營運者無法自行恢復、["檢查記錄"](#)。

狀態	說明
安裝	操作員正在使用此工具安裝Astra Trident TridentOrchestrator CR.
已安裝	Astra Trident已成功安裝。
正在解除安裝	因為、操作者正在解除安裝Astra Trident spec.uninstall=true。
已解除安裝	Astra Trident已解除安裝。
失敗	營運者無法安裝、修補、更新或解除安裝Astra Trident；營運者將自動嘗試從此狀態恢復。如果此狀態持續存在、您將需要疑難排解。
正在更新	營運者正在更新現有的安裝。
錯誤	。TridentOrchestrator 未使用。另一個已經存在。

使用Pod建立狀態

您可以檢閱建立的Pod狀態、確認是否已完成Astra Trident安裝：

```
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 檢查安裝的Astra Trident版本。

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 23.01.1        | 23.01.1        |
+-----+-----+
```

下一步

現在您可以了 ["建立後端和儲存類別、配置磁碟區、並將磁碟區掛載到Pod中"](#)。

手動部署Trident運算子（離線模式）

您可以手動部署Trident運算子來安裝Astra Trident。此程序適用於將Astra Trident所需的容器映像儲存在私有登錄中的安裝。如果您沒有私有映像登錄、請使用 ["標準部署程序"](#)。

Astra Trident的重要資訊23.01

您必須閱讀下列有關**Astra Trident**的重要資訊。

關於**Astra Trid-功能** 的資訊

- Kubernetes 1.26現在已在Trident中受到支援。升級Kubernetes之前先升級Trident。
- Astra Trident在SAN環境中嚴格執行多重路徑組態的使用、建議的值為 `find_multipaths: no` 在多重路徑.conf檔案中。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

手動部署Trident運算子並安裝Trident

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

登入Linux主機、驗證其是否正在管理正常運作的和 ["支援的Kubernetes叢集"](#) 而且您擁有必要的權限。



使用OpenShift、使用 `oc` 而非 `kubectl` 在以下所有範例中、請先執行*系統：admin*登入 `oc login -u system:admin` 或 `oc login -u kube-admin`。

1. 驗證Kubernetes版本：

```
kubectl version
```

2. 驗證叢集管理員權限：

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

3. 確認您可以啟動使用Docker Hub映像的Pod、並透過Pod網路連線至儲存系統：

```
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

Astra Trident安裝程式套件包含部署Trident操作員及安裝Astra Trident所需的一切。從下載並擷取Trident安裝程式的最新版本 "[GitHub的_Assets區段](#)"。

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

步驟2：建立 TridentOrchestrator 客戶需求日

建立 TridentOrchestrator 自訂資源定義（CRD）。您將建立 TridentOrchestrator 稍後再自訂資源。請使用中適當的CRD Y反 洗錢版本 deploy/crds 以建立 TridentOrchestrator 客戶需求日：

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

步驟3：更新操作員中的登錄位置

在中 /deploy/operator.yaml、更新 image: docker.io/netapp/trident-operator:23.01.1 以反映映像登錄的位置。您的 "[Trident和csi影像](#)" 可以位於一個登錄或不同的登錄中、但所有的SCSI映像都必須位於同一個登錄中。例如：

- image: <your-registry>/trident-operator:23.01.1 如果您的映像都位於單一登錄中。
- image: <your-registry>/netapp/trident-operator:23.01.1 如果Trident映像與您的csi映像位於不同的登錄中。

步驟4：部署Trident運算子

Trident安裝程式會在中部署操作員 trident 命名空間。如果是 trident 命名空間不存在、請使用 `kubectl apply -f deploy/namespace.yaml` 以建立。

可在非的命名空間中部署運算子 trident 命名空間、更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml` 部署營運者之前。

1. 建立資源並部署營運者：

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



Astra Trident安裝程式提供一個套件檔案、可用來安裝運算子及建立相關的物件。套裝組合檔案是使用預設組態部署操作員及安裝Astra Trident的簡易方法。

- 對於執行Kubernetes 1.24或更低層級的叢集、請使用 `bundle_pre_1_25.yaml`。
- 對於執行Kubernetes 1.25或更新版本的叢集、請使用 `bundle_post_1_25.yaml`。

2. 驗證是否已部署操作員。

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

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
trident-operator	1/1	1	1	3m



Kubernetes叢集中只應有*一個運算子執行個體*。請勿建立Trident營運者的多個部署。

步驟5：更新中的映像登錄位置 TridentOrchestrator

您的 "[Trident和csi影像](#)" 可以位於一個登錄或不同的登錄中、但所有的SCSI映像都必須位於同一個登錄中。更新 `deploy/crds/tridentorchestrator_cr.yaml` 根據登錄組態新增額外的位置規格。

一個登錄中的映像

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

不同登錄中的映像

您必須附加 sig-storage 至 imageRegistry 使用不同的登錄位置。

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

步驟6：建立 TridentOrchestrator 並安裝Trident

您現在可以建立 TridentOrchestrator 並安裝Astra Trident。您也可以選擇進一步 ["自訂您的Trident安裝"](#) 使用中的屬性 TridentOrchestrator 規格下列範例顯示Trident與csi映像位於不同登錄中的安裝。

```

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>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image:  <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:     <your-registry>/sig-storage
    k8sTimeout:          30
    Kubelet Dir:         /var/lib/kubelet
    Log Format:           text
    Probe Port:          17546
    Silence Autosupport: false
    Trident Image:      <your-registry>/netapp/trident:23.01.1
  Message:              Trident installed
  Namespace:            trident
  Status:                Installed
  Version:               v23.01.1
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

驗證安裝的方法有多種。

使用 TridentOrchestrator 狀態

狀態 TridentOrchestrator 指出安裝是否成功、並顯示安裝的Trident版本。安裝期間的狀態 TridentOrchestrator 變更來源 Installing 至 Installed。如果您觀察到 Failed 狀態、而且營運者無法自行恢復、["檢查記錄"](#)。

狀態	說明
安裝	操作員正在使用此工具安裝Astra Trident TridentOrchestrator CR.
已安裝	Astra Trident已成功安裝。
正在解除安裝	因為、操作者正在解除安裝Astra Trident spec.uninstall=true。
已解除安裝	Astra Trident已解除安裝。
失敗	營運者無法安裝、修補、更新或解除安裝Astra Trident；營運者將自動嘗試從此狀態恢復。如果此狀態持續存在、您將需要疑難排解。
正在更新	營運者正在更新現有的安裝。
錯誤	。TridentOrchestrator 未使用。另一個已經存在。

使用Pod建立狀態

您可以檢閱建立的Pod狀態、確認是否已完成Astra Trident安裝：

```
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` 檢查安裝的Astra Trident版本。

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 23.01.1        | 23.01.1        |
+-----+-----+
```

下一步

現在您可以了 ["建立後端和儲存類別、配置磁碟區、並將磁碟區掛載到Pod中"](#)。

使用Helm部署Trident運算子（標準模式）

您可以部署Trident運算子、並使用Helm安裝Astra Trident。此程序適用於未將Astra Trident所需的容器映像儲存在私有登錄中的安裝。如果您有私有映像登錄、請使用 ["離線部署程序"](#)。

Astra Trident的重要資訊23.01

您必須閱讀下列有關Astra Trident的重要資訊。

關於Astra Trident-功能 的資訊

- Kubernetes 1.26現在已在Trident中受到支援。升級Kubernetes之前先升級Trident。
- Astra Trident在SAN環境中嚴格執行多重路徑組態的使用、建議的值為 `find_multipaths: no` 在多重路徑.conf檔案中。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

部署Trident操作員、並使用Helm安裝Astra Trident

使用Trident ["掌舵表"](#) 您可以部署Trident運算子、並在單一步驟中安裝Trident。

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

此外 ["部署先決條件"](#) 您的需求 ["Helm版本3"](#)。

步驟

1. 新增Astra Trident Helm儲存庫：

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

2. 使用 `helm install` 並指定部署名稱、如下例所示 23.01.1 是您要安裝的Astra Trident版本。

```
helm install <name> netapp-trident/trident-operator --version 23.01.1  
--create-namespace --namespace <trident-namespace>
```



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

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

在安裝期間傳遞組態資料

安裝期間有兩種傳遞組態資料的方法：

選項	說明
<code>--values</code> (或 <code>-f</code>)	指定具有覆寫的Yaml檔案。這可以多次指定、最右邊的檔案會優先。
<code>--set</code>	在命令列上指定置換。

例如、變更的預設值 `debug`、請執行下列步驟、`--set` 命令位置 23.01.1 您要安裝的Astra Trident版本：

```
helm install <name> netapp-trident/trident-operator --version 23.01.1  
--create-namespace --namespace --set tridentDebug=true
```

組態選項

此表格和 `values.yaml` 檔案是 Helm 圖表的一部分、提供按鍵清單及其預設值。

選項	說明	預設
<code>nodeSelector</code>	Pod 指派的節點標籤	
<code>podAnnotations</code>	Pod 註釋	
<code>deploymentAnnotations</code>	部署註釋	
<code>tolerations</code>	Pod 指派的容錯功能	
<code>affinity</code>	Pod 指派的關聯性	

選項	說明	預設
tridentControllerPluginNodeSelector	用於 Pod 的其他節點選取器。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentControllerPluginTolerations	覆寫 Pod 的 Kubernetes 公差。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentNodePluginNodeSelector	用於 Pod 的其他節點選取器。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentNodePluginTolerations	覆寫 Pod 的 Kubernetes 公差。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
imageRegistry	識別的登錄 trident-operator、`trident` 和其他影像。保留空白以接受預設值。	"
imagePullPolicy	設定的映像拉出原則 trident-operator。	IfNotPresent
imagePullSecrets	設定的影像拉出秘密 trident-operator、`trident` 和其他影像。	
kubeletDir	允許覆寫 kubelet 內部狀態的主機位置。	'/var/lib/kubelet'
operatorLogLevel	允許 Trident 運算子的記錄層級設定為：trace、debug、info、warn、error 或 fatal。	"info"
operatorDebug	允許將 Trident 運算子的記錄層級設為偵錯。	true
operatorImage	允許完全置換的映像 trident-operator。	"
operatorImageTag	允許覆寫的標記 trident-operator 映像。	"
tridentIPv6	允許 Astra Trident 在 IPv6 叢集中運作。	false
tridentK8sTimeout	覆寫大部分 Kubernetes API 作業的預設 30 秒逾時（如果非零、則以秒為單位）。	0
tridentHttpRequestTimeout	以取代 HTTP 要求的預設 90 秒逾時 0s 是超時的無限持續時間。不允許使用負值。	"90s"
tridentSilenceAutosupport	可停用 Astra Trident 定期 AutoSupport 報告。	false
tridentAutosupportImageTag	可覆寫 Astra Trident AutoSupport 容器的映像標記。	<version>

選項	說明	預設
tridentAutosupportProxy	允許 Astra Trident AutoSupport 容器透過 HTTP Proxy 撥打電話回家。	"
tridentLogFormat	設定 Astra Trident 記錄格式 (text 或 json) 。	"text"
tridentDisableAuditLog	停用 Astra Trident 稽核記錄程式。	true
tridentLogLevel	允許將 Astra Trident 的記錄層級設定為：trace、debug、info、warn、error 或 fatal。	"info"
tridentDebug	允許將 Astra Trident 的記錄層級設定為 debug。	false
tridentLogWorkflows	允許啟用特定的 Astra Trident 工作流程、以進行追蹤記錄或記錄抑制。	"
tridentLogLayers	允許啟用特定的 Astra Trident 圖層、以進行追蹤記錄或記錄抑制。	"
tridentImage	允許完整置換 Astra Trident 的影像。	"
tridentImageTag	可覆寫 Astra Trident 的影像標記。	"
tridentProbePort	允許覆寫 Kubernetes 活性 / 整備性探查所使用的預設連接埠。	"
windows	允許在 Windows 工作節點上安裝 Astra Trident。	false
enableForceDetach	允許啟用強制分離功能。	false
excludePodSecurityPolicy	不建立營運商 Pod 安全性原則。	false

瞭解控制器 Pod 和節點 Pod

Astra Trident 會以單一控制器 Pod 的形式執行、並在叢集中的每個工作節點上提供節點 Pod。節點 Pod 必須在任何想要裝載 Astra Trident Volume 的主機上執行。

Kubernetes ["節點選取器"](#) 和 ["容忍和污染"](#) 用於限制 Pod 在特定或偏好的節點上執行。使用「ControllerPlugin」和 NodePlugin，您可以指定限制和置換。

- 控制器外掛程式可處理磁碟區資源配置與管理、例如快照和調整大小。
- 節點外掛程式會處理將儲存設備附加至節點的問題。

下一步

現在您可以了 ["建立後端和儲存類別、配置磁碟區、並將磁碟區掛載到Pod中"](#)。

使用Helm部署Trident運算子（離線模式）

您可以部署Trident運算子、並使用Helm安裝Astra Trident。此程序適用於將Astra Trident所需的容器映像儲存在私有登錄中的安裝。如果您沒有私有映像登錄、請使用 ["標準部署程序"](#)。

Astra Trident的重要資訊23.01

您必須閱讀下列有關Astra Trident的重要資訊。

關於Astra Trident-功能 的資訊

- Kubernetes 1.26現在已在Trident中受到支援。升級Kubernetes之前先升級Trident。
- Astra Trident在SAN環境中嚴格執行多重路徑組態的使用、建議的值為 `find_multipaths: no` 在多重路徑.conf檔案中。

使用非多重路徑組態或使用 `find_multipaths: yes` 或 `find_multipaths: smart` 多重路徑.conf檔案中的值會導致掛載失敗。Trident建議使用 `find_multipaths: no` 自21.07版本以來。

部署Trident操作員、並使用Helm安裝Astra Trident

使用Trident ["掌舵表"](#) 您可以部署Trident運算子、並在單一步驟中安裝Trident。

檢閱 ["安裝總覽"](#) 為了確保您符合安裝先決條件、並為您的環境選擇正確的安裝選項。

開始之前

此外 ["部署先決條件"](#) 您的需求 ["Helm版本3"](#)。

步驟

1. 新增Astra Trident Helm儲存庫：

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

2. 使用 `helm install` 並指定部署和映像登錄位置的名稱。您的 ["Trident和csi影像"](#) 可以位於一個登錄或不同的登錄中、但所有的SCSI映像都必須位於同一個登錄中。在範例中、23.01.1 是您要安裝的Astra Trident版本。

一個登錄中的映像

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

不同登錄中的映像

您必須附加 sig-storage 至 imageRegistry 使用不同的登錄位置。

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```



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

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

在安裝期間傳遞組態資料

安裝期間有兩種傳遞組態資料的方法：

選項	說明
<code>--values</code> (或 <code>-f</code>)	指定具有覆寫的Yaml檔案。這可以多次指定、最右邊的檔案會優先。
<code>--set</code>	在命令列上指定置換。

例如、變更的預設值 `debug`、請執行下列步驟、`--set` 命令位置 23.01.1 您要安裝的Astra Trident版本：

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

組態選項

此表格和 `values.yaml` 檔案是 Helm 圖表的一部分、提供按鍵清單及其預設值。

選項	說明	預設
nodeSelector	Pod 指派的節點標籤	
podAnnotations	Pod 註釋	
deploymentAnnotations	部署註釋	
tolerations	Pod 指派的容錯功能	
affinity	Pod 指派的關聯性	
tridentControllerPluginNodeSelector	用於 Pod 的其他節點選取器。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentControllerPluginTolerations	覆寫 Pod 的 Kubernetes 公差。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentNodePluginNodeSelector	用於 Pod 的其他節點選取器。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
tridentNodePluginTolerations	覆寫 Pod 的 Kubernetes 公差。請參閱 瞭解控制器 Pod 和節點 Pod 以取得詳細資料。	
imageRegistry	識別的登錄 trident-operator、`trident` 和其他影像。保留空白以接受預設值。	"
imagePullPolicy	設定的映像拉出原則 trident-operator。	IfNotPresent
imagePullSecrets	設定的影像拉出秘密 trident-operator、`trident` 和其他影像。	
kubeletDir	允許覆寫 kubelet 內部狀態的主機位置。	'/var/lib/kubelet'
operatorLogLevel	允許 Trident 運算子的記錄層級設定為：trace、debug、info、warn、error 或 fatal。	"info"
operatorDebug	允許將 Trident 運算子的記錄層級設為偵錯。	true
operatorImage	允許完全置換的映像 trident-operator。	"
operatorImageTag	允許覆寫的標記 trident-operator 映像。	"
tridentIPv6	允許 Astra Trident 在 IPv6 叢集中運作。	false
tridentK8sTimeout	覆寫大部分 Kubernetes API 作業的預設 30 秒逾時（如果非零、則以秒為單位）。	0

選項	說明	預設
tridentHttpRequestTimeout	以取代 HTTP 要求的預設 90 秒逾時 0s 是超時的無限持續時間。不允許使用負值。	"90s"
tridentSilenceAutosupport	可停用 Astra Trident 定期 AutoSupport 報告。	false
tridentAutosupportImageTag	可覆寫 Astra Trident AutoSupport 容器的映像標記。	<version>
tridentAutosupportProxy	允許 Astra Trident AutoSupport 容器透過 HTTP Proxy 撥打電話回家。	"
tridentLogFormat	設定 Astra Trident 記錄格式 (text 或 json) 。	"text"
tridentDisableAuditLog	停用 Astra Trident 稽核記錄程式。	true
tridentLogLevel	允許將 Astra Trident 的記錄層級設定為：trace、debug、info、warn、error 或 fatal。	"info"
tridentDebug	允許將 Astra Trident 的記錄層級設定為 debug。	false
tridentLogWorkflows	允許啟用特定的 Astra Trident 工作流程、以進行追蹤記錄或記錄抑制。	"
tridentLogLayers	允許啟用特定的 Astra Trident 圖層、以進行追蹤記錄或記錄抑制。	"
tridentImage	允許完整置換 Astra Trident 的影像。	"
tridentImageTag	可覆寫 Astra Trident 的影像標記。	"
tridentProbePort	允許覆寫 Kubernetes 活性 / 整備性探查所使用的預設連接埠。	"
windows	允許在 Windows 工作節點上安裝 Astra Trident 。	false
enableForceDetach	允許啟用強制分離功能。	false
excludePodSecurityPolicy	不建立營運商 Pod 安全性原則。	false

瞭解控制器 Pod 和節點 Pod

Astra Trident 會以單一控制器 Pod 的形式執行、並在叢集中的每個工作節點上提供節點 Pod。節點 Pod 必須在任何想要裝載 Astra Trident Volume 的主機上執行。

Kubernetes ["節點選取器"](#) 和 ["容忍和污染"](#) 用於限制 Pod 在特定或偏好的節點上執行。使用「ControllerPlugin」和 NodePlugin，您可以指定限制和置換。

- 控制器外掛程式可處理磁碟區資源配置與管理、例如快照和調整大小。

- 節點外掛程式會處理將儲存設備附加至節點的問題。

下一步

現在您可以了 ["建立後端和儲存類別、配置磁碟區、並將磁碟區掛載到Pod中"](#)。

自訂Trident操作員安裝

Trident運算子可讓您使用中的屬性來自訂Astra Trident安裝 TridentOrchestrator 規格如果您想要自訂安裝內容以外的內容 TridentOrchestrator 引數允許、請考慮使用 tridentctl 產生自訂的 YAML 資訊清單、以視需要進行修改。

瞭解控制器 Pod 和節點 Pod

Astra Trident 會以單一控制器 Pod 的形式執行、並在叢集中的每個工作節點上提供節點 Pod 。節點 Pod 必須在任何想要裝載 Astra Trident Volume 的主機上執行。

Kubernetes ["節點選取器"](#) 和 ["容忍和污染"](#) 用於限制 Pod 在特定或偏好的節點上執行。使用「ControllerPlugin」和 NodePlugin，您可以指定限制和置換。

- 控制器外掛程式可處理磁碟區資源配置與管理、例如快照和調整大小。
- 節點外掛程式會處理將儲存設備附加至節點的問題。

組態選項



spec.namespace 在中指定 TridentOrchestrator 表示 Astra Trident 安裝的命名空間。此參數*無法在安裝Astra Trident之後更新*。嘗試這麼做會導致 TridentOrchestrator 要變更為的狀態 Failed。Astra Trident不打算跨命名空間移轉。

本表詳細說明 TridentOrchestrator 屬性。

參數	說明	預設
namespace	用於安裝Astra Trident的命名空間	"預設"
debug	啟用Astra Trident的偵錯功能	錯
windows	設定為 true 可在Windows工作節點上安裝。	錯
IPv6	透過IPv6安裝Astra Trident	錯
k8sTimeout	Kubernetes作業逾時	30秒
silenceAutosupport	請勿AutoSupport 自動將此套裝組合傳送至NetApp	錯
enableNodePrep	自動管理工作節點相依性 (* BETA *)	錯
autosupportImage	遙測的容器影像AutoSupport	「NetApp/Trident自動支援：23.01」

參數	說明	預設
autosupportProxy	代理伺服器的位址/連接埠、用於傳送AutoSupport「遙測」功能	"http://proxy.example.com:8888"
uninstall	用來解除安裝Astra Trident的旗標	錯
logFormat	要使用的Astra Trident記錄格式 [text、json]	"文字"
tridentImage	要安裝的Astra Trident映像	「NetApp/Trident：21.04」
imageRegistry	內部登錄的路徑、格式 <registry FQDN>[:port] [/subpath]	"k8s.gcr.io/sig-storage (k8s 1.19+) 或kay.io/k8scsi"
kubeletDir	主機上的kubelet目錄路徑	"/var/lib/kubelet"
wipeout	要刪除以執行完整移除Astra Trident的資源清單	
imagePullSecrets	從內部登錄擷取映像的機密	
imagePullPolicy	設定Trident運算子的影像提取原則。有效值包括： Always 永遠拉出映像。 IfNotPresent 僅當節點上尚未存在映像時才提取映像。 Never 永遠不要拉動映像。	IfNotPresent
controllerPluginNodeSelector	用於 Pod 的其他節點選取器。格式與pod.spec.nodeSelector相同。	無預設值；選用
controllerPluginTolerations	覆寫 Pod 的 Kubernetes 公差。格式與po.spec.TBolerations相同。	無預設值；選用
nodePluginNodeSelector	用於 Pod 的其他節點選取器。格式與pod.spec.nodeSelector相同。	無預設值；選用
nodePluginTolerations	覆寫 Pod 的 Kubernetes 公差。格式與po.spec.TBolerations相同。	無預設值；選用



如需格式化Pod參數的詳細資訊、請參閱 ["將Pod指派給節點"](#)。

組態範例

您可以在定義時使用上述屬性 TridentOrchestrator 以自訂安裝。

範例1：基本自訂組態

這是基本自訂組態的範例。

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

範例2：使用節點選取器進行部署

此範例說明如何使用節點選取器來部署Trident：

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


範例3：部署在Windows工作節點上

此範例說明如何在Windows工作節點上進行部署。

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

版權資訊

Copyright © 2024 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。