開始使用
Astra Trident

NetApp
November 14, 2025

This PDF was generated from https://docs.netapp.com/zh-tw/trident-2301/trident-get-started/quickstart.html on November 14, 2025. Always check docs.netapp.com for the latest.

目錄

開始使用 1
歡迎試用 1
深入瞭解測試磁碟機 1
需求 1
Astra Trident的重要資訊23.01 1
支援的前端（協調器） 1
支援的後端（儲存） 2
功能需求 2
已測試的主機作業系統 3
主機組態 3
儲存系統組態 3
Astra Trident連接埠 3
Container映像和對應的Kubernetes版本 3
安裝Astra Trident 6
瞭解Astra Trident安裝 6
使用Trident操作員安裝 9
使用tridentctl安裝 31
接下來呢？ 35
步驟1：建立後端 35
步驟2：建立儲存類別 36
步驟3：配置第一個Volume 38
步驟4：在Pod中掛載磁碟區 39

開始使用

歡迎試用

NetApp提供立即可用的實驗室映像、您可以透過申請 "NetApp試用"。

深入瞭解測試磁碟機

測試磁碟機提供您一個沙箱環境、其中安裝並設定了三節點Kubernetes叢集和Astra Trident。這是熟悉Astra Trident並探索其特色的絕佳方式。

另一個選項是查看 "Kubeadm安裝指南" 由Kubernetes提供。

> 您不應該在正式作業中使用這些指令所建置的Kubernetes叢集。請使用經銷商提供的正式作業部署指南、建立正式作業就緒的叢集。

如果這是您第一次使用Kubernetes、請熟悉這些概念和工具 "請按這裡"。

需求

在安裝Astra Trident之前、您應該先檢閱這些一般系統需求。特定後端可能有其他需求。

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。

支援的前端（協調器）

Astra Trident支援多個容器引擎和協調器、包括：

- Anthos on -Prem（VMware）和Anthos on裸機1.9、1.10、1.11
- Kubernetes 1.21 - 1.26
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.9 - 4.12

這些版本支援Trident運算子：

- Anthos on -Prem（VMware）和Anthos on裸機1.9、1.10、1.11
- Kubernetes 1.21 - 1.26
- OpenShift 4.9 - 4.12

Astra Trident也能與其他全管理且自我管理的Kubernetes產品搭配使用、包括Google Kubernetes Engine（GKE）、Amazon Elastic Kubernetes Services（EKS）、Azure Kubernetes Service（KS）、Rancher及VMware Tanzu Portfolio。

在將Kubernetes叢集從1.24升級至1.25或更新版本、且已安裝Astra Trident之前、請參閱 "升級Helm型的營運者安裝"。

支援的後端（儲存）

若要使用Astra Trident、您需要下列一或多個支援的後端：

- Amazon FSX for NetApp ONTAP 產品
- Azure NetApp Files
- Cloud Volumes ONTAP
- 適用於 GCP Cloud Volumes Service
- FAS/AFF/選取9.5或更新版本
- NetApp All SAN Array ASA （ESAN）
- NetApp HCI / Element軟體11或更新版本

功能需求

下表摘要說明此Astra Trident版本的可用功能及其支援的Kubernetes版本。

功能	**Kubernetes**版本	需要功能閘道？
csi Trident	1.21 - 1.26	否
Volume Snapshot	1.21 - 1.26	否
來自Volume Snapshot的PVc	1.21 - 1.26	否
iSCSI PV調整大小	1.21 - 1.26	否
資訊雙向CHAP ONTAP	1.21 - 1.26	否
動態匯出原則	1.21 - 1.26	否
Trident運算子	1.21 - 1.26	否
csi拓撲	1.21 - 1.26	否

已測試的主機作業系統

儘管Astra Trident並未正式支援特定作業系統、但已知下列項目仍能正常運作：

- 受OpenShift Container Platform支援的RedHat CoreOS（RMCOS）版本
- RHEL 8以上
- Ubuntu 22.04或更新版本
- Windows Server 2019

依預設、Astra Trident會在容器中執行、因此會在任何Linux工作者上執行。不過、這些員工必須能夠使用標準NFS用戶端或iSCSI啟動器來掛載Astra Trident提供的磁碟區、視您使用的後端而定。

。 `tridentctl` 公用程式也可在這些Linux版本中的任何一種上執行。

主機組態

Kubernetes叢集中的所有工作節點都必須能夠掛載您已為Pod配置的磁碟區。若要準備工作節點、您必須根據所選的驅動程式來安裝NFS或iSCSI工具。

"準備工作節點"

儲存系統組態

Astra Trident可能需要變更儲存系統、才能使用後端組態。

"設定後端"

Astra Trident連接埠

Astra Trident需要存取特定連接埠才能進行通訊。

"Astra Trident連接埠"

Container映像和對應的Kubernetes版本

對於空拍安裝、下列清單是安裝Astra Trident所需的容器映像參考資料。使用 `tridentctl images` 用於驗證所需容器映像清單的命令。

Kubernetes版本	Container映像
1.21.0版	• Docker 。 IO/NetApp/Trident ： 23.01.1 • Docker 。 IO/NetApp/trident 自動支援： 23.01 • registry ・ k8s.io/SIG-storage / csi 置備程序： v3.4.0 • 登錄 .k8s.io/SIG-storage / csi 附加程式： v4.1.0 • 登錄 .k8s.io/SIG-storage / csi 大小調整： v1.7.0 • 登錄 .k8s.IO/SIG-storage / csi 快照機： v6.2.1 • 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器： v2.2.0 • Docker 。 IO/NetApp/Trident 營運商： 23.01.1 （選用）
1.22.0版	• Docker 。 IO/NetApp/Trident ： 23.01.1 • Docker 。 IO/NetApp/trident 自動支援： 23.01 • registry ・ k8s.io/SIG-storage / csi 置備程序： v3.4.0 • 登錄 .k8s.io/SIG-storage / csi 附加程式： v4.1.0 • 登錄 .k8s.io/SIG-storage / csi 大小調整： v1.7.0 • 登錄 .k8s.IO/SIG-storage / csi 快照機： v6.2.1 • 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器： v2.2.0 • Docker 。 IO/NetApp/Trident 營運商： 23.01.1 （選用）
1.23.0版	• Docker 。 IO/NetApp/Trident ： 23.01.1 • Docker 。 IO/NetApp/trident 自動支援： 23.01 • registry ・ k8s.io/SIG-storage / csi 置備程序： v3.4.0 • 登錄 .k8s.io/SIG-storage / csi 附加程式： v4.1.0 • 登錄 .k8s.io/SIG-storage / csi 大小調整： v1.7.0 • 登錄 .k8s.IO/SIG-storage / csi 快照機： v6.2.1 • 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器： v2.2.0 • Docker 。 IO/NetApp/Trident 營運商： 23.01.1 （選用）

Kubernetes版本	Container映像
1.24.0版	• Docker 。 IO/NetApp/Trident ： 23.01.1 • Docker 。 IO/NetApp/trident 自動支援： 23.01 • registry ・ k8s.io/SIG-storage / csi 置備程序： v3.4.0 • 登錄 .k8s.io/SIG-storage / csi 附加程式： v4.1.0 • 登錄 .k8s.io/SIG-storage / csi 大小調整： v1.7.0 • 登錄 .k8s.IO/SIG-storage / csi 快照機： v6.2.1 • 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器： v2.2.0 • Docker 。 IO/NetApp/Trident 營運商： 23.01.1 （選用）
v1.25.0	• Docker 。 IO/NetApp/Trident ： 23.01.1 • Docker 。 IO/NetApp/trident 自動支援： 23.01 • registry ・ k8s.io/SIG-storage / csi 置備程序： v3.4.0 • 登錄 .k8s.io/SIG-storage / csi 附加程式： v4.1.0 • 登錄 .k8s.io/SIG-storage / csi 大小調整： v1.7.0 • 登錄 .k8s.IO/SIG-storage / csi 快照機： v6.2.1 • 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器： v2.2.0 • Docker 。 IO/NetApp/Trident 營運商： 23.01.1 （選用）
1.26.0版	• Docker 。 IO/NetApp/Trident ： 23.01.1 • Docker 。 IO/NetApp/trident 自動支援： 23.01 • registry ・ k8s.io/SIG-storage / csi 置備程序： v3.4.0 • 登錄 .k8s.io/SIG-storage / csi 附加程式： v4.1.0 • 登錄 .k8s.io/SIG-storage / csi 大小調整： v1.7.0 • 登錄 .k8s.IO/SIG-storage / csi 快照機： v6.2.1 • 登錄 .k8s.io/SIG-storage / csi 節點驅動程式登錄器： v2.2.0 • Docker 。 IO/NetApp/Trident 營運商： 23.01.1 （選用）

在Kubernetes 1.21版及更新版本上、請使用已驗證的 `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` 僅在以下情況下顯示映像 `v1` 版本正在提供 `volumesnapshots.snapshot.storage.k8s.gcr.io` 客戶需求日如果是 `v1beta1` 版本為CRD提供/不提供 `v1` 版本、請使用已驗證的 `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` 映像。

安裝Astra Trident

瞭解Astra Trident安裝

為了確保Astra Trident可安裝在各種環境和組織中、NetApp提供多種安裝選項。您可以使用Trident運算子（手動或使用Helm）或搭配安裝Astra Trident `tridentctl`。本主題提供重要資訊、協助您選擇正確的安裝程序。

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。

開始之前

無論安裝路徑為何、您都必須具備：

- 對執行支援版本Kubernetes及啟用功能需求的Kubernetes叢集擁有完整權限。檢閱 "需求" 以取得詳細資料。
- 存取支援的NetApp儲存系統。
- 能夠從所有Kubernetes工作節點掛載磁碟區。
- Linux主機 `kubectl` （或 `oc`（如果您使用OpenShift）已安裝並設定為管理您要使用的Kubernetes叢集。
- 。 `KUBECONFIG` 環境變數設定為指向Kubernetes叢集組態。
- 如果您使用Kubernetes搭配Docker Enterprise、 "請依照他們的步驟啟用CLI存取"。

如果您尚未熟悉 "基本概念"現在正是這麼做的好時機。

選擇您的安裝方法

選取最適合您的安裝方法。您也應該檢閱的考量事項 "在方法之間移動" 做出決定之前。

使用**Trident**運算子

無論是手動部署或使用Helm、Trident營運者都是簡化安裝及動態管理Astra Trident資源的絕佳方式。您甚至可以"自訂您的Trident營運者部署" 使用中的屬性 `TridentOrchestrator` 自訂資源（CR）。

使用Trident營運者的好處包括：

不只是個**Trident**物件、更是個可說是個可說是個地方** **

Trident運算子會自動為Kubernetes版本建立下列物件。

- 營運者服務帳戶
- 叢集角色和叢集角色繫結至服務帳戶
- 專屬的PodSecurity原則（適用於Kubernetes 1.25及更早版本）
- 營運者本身

還原功能的功能** **

營運者會監控Astra Trident安裝、並主動採取措施來處理問題、例如刪除部署或意外修改部署。答 `trident-operator-<generated-id>` 建立Pod以建立關聯 `TridentOrchestrator` 含Astra Trident安裝的CR。如此可確保叢集中只有一個Astra Trident執行個體、並控制其設定、確保安裝功能強大。當對安裝進行變更（例如刪除部署或節點取消設定）時、操作員會分別識別並修正這些變更。

更新至現有安裝的更新功能** **

您可以輕鬆地與營運者一起更新現有的部署。您只需要編輯 `TridentOrchestrator` 以更新安裝。

例如、假設您需要啟用Astra Trident來產生偵錯記錄的案例。若要這麼做、請修補您的 `TridentOrchestrator` 以設定 `spec.debug` 至 `true`：

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

之後 `TridentOrchestrator` 更新後、營運者會處理更新並修補現有安裝。這可能會觸發建立新的Pod、以據此修改安裝。

支援升級功能的不一樣之處** **

當叢集的Kubernetes版本升級至支援的版本時、營運者會自動更新現有的Astra Trident安裝、並加以變更、以確保其符合Kubernetes版本的要求。

> 如果叢集升級至不受支援的版本、則操作員將無法安裝Astra Trident。如果操作員已安裝Astra Trident、則會顯示警告、指出Astra Trident安裝在不受支援的Kubernetes版本上。

使用NetApp Console管理 Kubernetes 叢集

透過使用NetApp Console 的Astra Trident ，您可以升級至最新版本的Astra Trident，新增和管理儲存類別並將其連接到工作環境，以及使用Cloud Backup Service備份持久性磁碟區。控制台支援使用Trident操作員手動或使用 Helm 部署Astra Trident 。

使用 `tridentctl`

如果您現有的部署必須升級、或是想要高度自訂部署、您應該考慮 。這是部署Astra Trident的傳統方法。

您可以 以產生Trident資源的資訊清單。這包括部署、取消程式集、服務帳戶、以及Astra Trident在安裝過程中所建立的叢集角色。

從22.04版開始、每次安裝Astra Trident時、AES金鑰就不會再重新產生。在這個版本中、Astra Trident會安裝一個新的秘密物件、並在安裝過程中持續存在。這表示、 `tridentctl` 在22.04中可以解除安裝舊版Trident、但舊版無法解除安裝22.04安裝。選取適當的安裝方法_。

選擇安裝模式

根據組織所需的安裝模式_（標準、離線或遠端）來判斷您的部署程序。

標準安裝

這是安裝Astra Trident最簡單的方法、適用於大多數不受網路限制的環境。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`）影像。

使用標準模式時、Astra Trident安裝程式會：

- 透過網際網路擷取容器映像
- 在Kubernetes叢集中的所有合格節點上建立部署或節點取消設定、以執行Astra Trident Pod

離線安裝

在無線或安全的位置可能需要離線安裝模式。在此案例中、您可以建立單一私有、鏡射的登錄或兩個鏡射登錄、以儲存所需的Trident和csi映像。

無論您的登錄組態為何、都必須將csi映像存放在單一登錄中。

遠端安裝

以下是遠端安裝程序的高階概觀：

- 部署適當版本的 `kubectl` 在您要部署Astra Trident的遠端機器上。
- 從Kubernetes叢集複製組態檔案、然後設定 `KUBECONFIG` 遠端機器上的環境變數。
- 啟動 `kubectl get nodes` 命令來驗證您是否可以連線至所需的Kubernetes叢集。
- 使用標準安裝步驟、從遠端機器完成部署。

根據您的方法和模式選取程序

做出決策後、請選擇適當的程序。

方法	安裝模式
Trident運算子（手動）	"標準安裝" "離線安裝"
Trident運算子（Helm）	"標準安裝" "離線安裝"
`tridentctl`	"標準或離線安裝"

在安裝方法之間移動

您可以決定變更安裝方法。在執行此操作之前、請先考慮下列事項：

- 安裝及解除安裝Astra Trident時、請務必使用相同的方法。如果您已部署 `tridentctl`、您應該使用適當版本的 `tridentctl` 二進位以解除安裝Astra Trident。同樣地、如果您是與操作員一起部署、則應該編輯 `TridentOrchestrator` 並設定 `spec.uninstall=true` 解除安裝Astra Trident。
- 如果您想要移除並改用以營運者為基礎的部署 `tridentctl` 若要部署Astra Trident、您應該先編輯 `TridentOrchestrator` 並設定 `spec.uninstall=true` 解除安裝Astra Trident。然後刪除 `TridentOrchestrator` 以及營運者部署。然後您可以使用安裝 `tridentctl`。
- 如果您有手動的操作員型部署、而且想要使用以Helm為基礎的Trident操作員部署、您應該先手動解除安裝操作員、然後再執行Helm安裝。如此一來、Helm就能部署具有所需標籤和註釋的Trident運算子。如果您不這麼做、則Helm型Trident營運者部署將會失敗、並顯示標籤驗證錯誤和註釋驗證錯誤。如果您有 `tridentctl` 根據部署、您可以使用以Helm為基礎的部署、而不會發生問題。

其他已知組態選項

在VMware Tanzu產品組合產品上安裝Astra Trident時：

- 叢集必須支援特殊權限的工作負載。
- 。 `--kubelet-dir` 旗標應設定為kubelet目錄的位置。依預設、這是 `/var/vcap/data/kubelet`。

  使用指定kubelet位置 `--kubelet-dir` 已知適用於Trident運算子、Helm和 `tridentctl` 部署：

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 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

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 ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

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 ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

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 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。

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
`--values` （或 `-f`）	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。
`--set`	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。	

選項	說明	預設
`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>`
`tridentAutosupportProxy`	允許 Astra Trident AutoSupport 容器透過 HTTP Proxy 撥打電話回家。	"

選項	說明	預設
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 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。

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
`--values` （或 `-f`）	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。
`--set`	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
```

使用tridentctl安裝

使用tridentctl安裝

您可以使用安裝Astra Trident `tridentctl`。此程序適用於將Astra Trident所需的容器映像儲存在私有登錄中或不儲存在私有登錄中的安裝。以自訂您的 `tridentctl` 部署、請參閱"自訂試用部署"。

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。

使用安裝Astra Trident `tridentctl`

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安裝程式套件會建立Trident pod、設定用來維持其狀態的CRD物件、並初始化csi sidecar以執行資源配置和將磁碟區附加至叢集主機等動作。從下載並擷取Trident安裝程式的最新版本 "GitHub的_Assets區段"。請使用<trident-installer-XX.XX.X.tar.gz> 您所選的Astra Trident版本來更新範例中的_SUR__。

```
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**：安裝**Astra Trident**

執行、在所需的命名空間中安裝Astra Trident `tridentctl install` 命令。您可以新增其他引數來指定映像登錄位置。

若要在Windows節點上執行Astra Trident、請新增 `--windows` 安裝命令的旗標： `$ ./tridentctl install --windows -n trident`。

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:23.01 --trident
-image <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。

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.01 --trident-image <your-
registry>/netapp/trident:23.01.1
```

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=23.01.1
INFO Trident installation succeeded.
....
```

驗證安裝

您可以使用Pod建立狀態或來驗證安裝 `tridentctl`。

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-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     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）沒有*執行中*的狀態、表示平台尚未安裝。使用 `-d` 至 "開啟偵錯模式" 並疑難排解問題。

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"。

自訂**tridentctl**安裝

您可以使用Astra Trident安裝程式來自訂安裝。

深入瞭解安裝程式

Astra Trident安裝程式可讓您自訂屬性。例如、如果您已將Trident映像複製到私有儲存庫、則可以使用來指定映像名稱 `--trident-image`。如果您已將Trident映像及所需的csi sidecar映像複製到私有儲存庫、最好使用指定該儲存庫的位置 `--image-registry` 交換器、採用格式 `<registry FQDN>[:port]`。

如果您使用Kubernetes的發佈版本、其中 `kubelet` 將資料保留在一般路徑以外的路徑上 `/var/lib/kubelet`、您可以使用來指定替代路徑 `--kubelet-dir`。

如果您需要自訂安裝、而不需要安裝程式的引數允許、也可以自訂部署檔案。使用 `--generate-custom-yaml` 參數會在安裝程式中建立下列Yaml檔案 `setup` 目錄：

- `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`

產生這些檔案之後、您可以根據自己的需求加以修改、然後使用 `--use-custom-yaml` 以安裝自訂部署。

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

接下來呢？

安裝Astra Trident之後、您可以繼續建立後端、建立儲存類別、配置磁碟區、以及將磁碟區掛載至Pod。

步驟1：建立後端

您現在可以繼續建立後端、由Astra Trident用來配置磁碟區。若要這麼做、請建立 `backend.json` 包含必要參數的檔案。不同後端類型的組態檔範例可在中找到 `sample-input` 目錄。

請參閱 "請按這裡" 如需如何為後端類型設定檔案的詳細資訊、請參閱。

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

如果建立失敗、表示後端組態有問題。您可以執行下列命令來檢視記錄、以判斷原因：

```
./tridentctl -n trident logs
```

解決問題之後、只要返回此步驟的開頭、然後再試一次即可。如需更多疑難排解秘訣、請參閱 "疑難排解" 區段。

步驟2：建立儲存類別

Kubernetes使用者使用指定的持續磁碟區宣告（PVCS）來配置磁碟區 "儲存類別" 依名稱。使用者會隱藏詳細資料、但儲存類別會識別該類別所使用的資源配置程式（本例中為Trident）、以及該類別對資源配置程式的意義。

建立儲存類別Kubernetes使用者要指定何時需要磁碟區。類別的組態需要建構您在上一個步驟中建立的後端、以便Astra Trident使用它來配置新的磁碟區。

最簡單的儲存類別是以為基礎的 `sample-input/storage-class-csi.yaml.templ` 安裝程式隨附的檔案、取代 *`BACKEND_TYPE`* 儲存驅動程式名稱。

```
./tridentctl -n trident get backend
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

這是Kubernetes物件、所以您可以使用 `kubectl` 在Kubernetes中建立。

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

現在您應該會看到Kubernetes和Astra Trident中的* basic、csi *儲存類別、而Astra Trident應該已經在後端探索集區。

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

步驟3：配置第一個Volume

現在您可以動態配置第一個Volume了。這是透過建立Kubernetes來完成 "持續磁碟區宣告" （PVc）物件。

為使用您剛建立之儲存類別的磁碟區建立一個永久虛擬磁碟。

請參閱 `sample-input/pvc-basic-csi.yaml` 例如：請確定儲存類別名稱符合您所建立的名稱。

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

步驟4：在Pod中掛載磁碟區

現在讓我們掛載磁碟區。我們將推出可安裝PV的Ngin像Pod `/usr/share/nginx/html`。

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

此時、Pod（應用程式）不再存在、但磁碟區仍然存在。如果需要、您可以從其他Pod使用。

若要刪除磁碟區、請刪除請款：

```
kubectl delete pvc basic
```

您現在可以執行其他工作、例如：

- "設定其他後端。"
- "建立其他儲存類別。"

版權資訊

Copyright © 2025 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 保留隨時變更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。