使用Trident操作員安裝

Astra Trident

NetApp
June 28, 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） . . . 15
自訂Trident操作員安裝 . . . 19

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 24.02 的重要資訊

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 1.27 現在支援 Trident 。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/v24.02.0/trident-
installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.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 或更早版本的叢集、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`。

開始之前

- 根據預設、 Trident 安裝程式會在中部署運算子 `trident` 命名空間。如果是 `trident` 命名空間不存在、請使用以下方式建立：

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

  a. 建立 `kustomization.yaml` 使用下列命令、其中包含 <bundle.yaml> `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

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

  b. 使用以下命令編譯套件（其中的 <bundle.yaml> 是） `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

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

步驟

1. 建立資源並部署營運者：

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

2. 確認已建立運算子、部署和複本集。

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

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:24.02
    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:24.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v24.02.0
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

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	操作員無法安裝、修補、更新或解除安裝 Astra Trident ；操作人員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。另一個已有的存在。

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 |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

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 24.02 的重要資訊

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 1.27 現在支援 Trident 。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/v24.02.0/trident-installer-24.02.0.tar.gz
tar -xf trident-installer-24.02.0.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:24.02.0` 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:24.02.0` 如果您的映像都位於單一登錄中。
- `image: <your-registry>/netapp/trident-operator:24.02.0` 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

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 或更早版本的叢集、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`。

開始之前

- 根據預設、 Trident 安裝程式會在中部署運算子 `trident` 命名空間。如果是 `trident` 命名空間不存在、請使用以下方式建立：

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

  a. 建立 `kustomization.yaml` 使用下列命令、其中包含 <bundle.yaml> `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

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

  b. 使用以下命令編譯套件（其中的 <bundle.yaml> 是） `bundle_pre_1_25.yaml` 或 `bundle_post_1_25.yaml` 以 Kubernetes 版本為基礎。

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

步驟

1. 建立資源並部署營運者：

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

2. 確認已建立運算子、部署和複本集。

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

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:24.02"
tridentImage: "<your-registry>/trident:24.02.0"
```

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:24.02"
tridentImage: "<your-registry>/netapp/trident:24.02.0"
```

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:24.02
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:24.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:24.02
    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:24.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v24.02.0
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

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	操作員無法安裝、修補、更新或解除安裝 Astra Trident ；操作人員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。另一個已有的存在。

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 |
+----------------+----------------+
| 24.02.0        | 24.02.0        |
+----------------+----------------+
```

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 24.02 的重要資訊

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 1.27 現在支援 Trident 。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 `100.2402.0` 是您要安裝的Astra Trident版本。

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--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 `100.2402.0` 您要安裝的Astra Trident版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --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	

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>
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。	""

選項	說明	預設
`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`
`cloudProvider`	設定為 `"Azure"` 在 AKS 叢集上使用託管身分識別或雲端身分識別時。在 EKS 叢集上使用雲端身分識別時、請設定為「 AWS 」。	`""`
`cloudIdentity`	在 AKS 叢集上使用雲端身分識別時、請設定為工作負載身分識別（「 azure.Workload .idental/client-id ： XXXXXXXX-xxxx-xxxx-xxxx-xxxx-xxxxxxx 」）。在 EKS 叢集上使用雲端身分識別時、請設定為 AWS IAM 角色（「 eks.amazonaws.com/role-arn: arn:AWS:iam::123456 ：角色 / 身分識別角色」）。	`""`
`iscsiSelfHealingInterval`	啟動 iSCSI 自我修復的時間間隔。	`5m0s`
`iscsiSelfHealingWaitTime`	iSCSI 自我修復透過執行登出和後續登入來嘗試解決過時工作階段的持續時間。	`7m0s`

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。

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 24.02 的重要資訊

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 1.27 現在支援 Trident 。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、 `100.2402.0` 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
100.2402.0 --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
100.2402.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:24.02.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:24.02 --set tridentImage=<your-
registry>/netapp/trident:24.02.0 --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 `100.2402.0` 您要安裝的Astra Trident版本：

```
helm install <name> netapp-trident/trident-operator --version 100.2402.0
--create-namespace --namespace trident --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

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	`"default"`
`debug`	啟用Astra Trident的偵錯功能	`false`
`enableForceDetach`	`ontap-san` 和 `ontap-san-economy` 僅限。 與 Kubernetes Non-Graceful Node Shutdown （ NGNS ）一起運作、讓叢集管理員能夠在節點發生問題時、將已掛載磁碟區的工作負載安全移轉至新節點。	`false`
`windows`	設定為 `true` 可在Windows工作節點上安裝。	`false`
`cloudProvider`	設定為 `"Azure"` 在 AKS 叢集上使用託管身分識別或雲端身分識別時。在 EKS 叢集上使用雲端身分識別時、請設定為「 AWS 」。	`""`
`cloudIdentity`	在 AKS 叢集上使用雲端身分識別時、請設定為工作負載身分識別（「 azure.Workload .idental/client-id ： XXXXXXXX-xxxx-xxxx-xxxx-xxxx-xxxxxxx 」）。在 EKS 叢集上使用雲端身分識別時、請設定為 AWS IAM 角色（「 eks.amazonaws.com/role-arn: arn:AWS:iam::123456 ：角色 / 身分識別角色」）。	`""`
`IPv6`	透過IPv6安裝Astra Trident	錯
`k8sTimeout`	Kubernetes作業逾時	`30sec`
`silenceAutosupport`	請勿將 AutoSupport 套裝組合傳送至 NetApp 自動	`false`
`autosupportImage`	遙測的容器影像AutoSupport	`"netapp/trident-autosupport:24.02"`
`autosupportProxy`	用於傳送 AutoSupport 的 Proxy 位址 / 連接埠遙測	`"http://proxy.example.com:8888"`

參數	說明	預設
`uninstall`	用來解除安裝Astra Trident的旗標	`false`
`logFormat`	要使用的Astra Trident記錄格式[text、json]	`"text"`
`tridentImage`	要安裝的Astra Trident映像	`"netapp/trident:24.02"`
`imageRegistry`	內部登錄的路徑、格式 `<registry FQDN>[:port][/subpath]`	`"k8s.gcr.io/sig-storage"` （Kubernetes 1.19+） 或 `"quay.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的資源清單 Astra Trident	
`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。 格式與相同 `pod.spec.nodeSelector`。	無預設值；選用
`controllerPluginTolerations`	覆寫 Pod 的 Kubernetes 公差。遵循與相同的格式 `pod.spec.Tolerations`。	無預設值；選用
`nodePluginNodeSelector`	用於 Pod 的其他節點選取器。格式與相同 `pod.spec.nodeSelector`。	無預設值；選用
`nodePluginTolerations`	覆寫 Pod 的 Kubernetes 公差。遵循與相同的格式 `pod.spec.Tolerations`。	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"。

強制分離的詳細資料

可以使用強制分離 `ontap-san` 和 `ontap-san-economy` 僅限。啟用強制分離之前、必須先在 Kubernetes 叢集上啟用非正常節點關機（ NGNS ）。如需詳細資訊、請參閱 "Kubernetes ：非正常節點關機"。

由於 Astra Trident 仰賴 Kubernetes NGNS 、因此請勿移除 `out-of-service` 從不正常的節點污染、直到重新排程所有不可容忍的工作負載為止。如果不考慮套用或移除污染、可能會危及後端資料保護。

當 Kubernetes 叢集管理員套用時 `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` 污染到節點和 `enableForceDetach` 設為 `true`、 Astra Trident 將決定節點狀態、並：

1. 停止掛載到該節點之磁碟區的後端 I/O 存取。
2. 將 Astra Trident 節點物件標記為 `dirty` （新出版品不安全）。

> Trident 控制器會拒絕新的發佈 Volume 要求、直到節點重新符合資格為止（在標記為之後） `dirty` ）。在 Astra Trident 能夠驗證節點之前、任何以掛載的 PVC 排程的工作負載（即使在叢集節點健全且準備就緒之後）都不會被接受 `clean` （新出版品的安全性）。

當節點健全狀況恢復且刪除污染物時、 Astra Trident 將：

1. 識別並清除節點上過時的已發佈路徑。
2. 如果節點位於 `cleanable` 狀態（服務外污點已移除、節點位於中 `Ready` 狀態）且所有過時的已發佈路徑都是乾淨的、 Astra Trident 將重新將節點重新接收為 `clean` 並允許新發行的磁碟區到節點。

組態範例

您可以在中使用屬性 [組態選項] 定義時 `TridentOrchestrator` 以自訂安裝。

基本自訂組態

這是基本自訂安裝的範例。

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

節點選取器

此範例會安裝 Astra Trident 搭配節點選取器。

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

Windows 工作者節點

此範例會在 Windows 工作者節點上安裝 Astra Trident 。

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

在 **AKS** 叢集上的託管身分識別

此範例安裝 Astra Trident 、可在 Aaks 叢集上啟用受管理的身分識別。

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

在 **AKS** 叢集上的雲端身分識別

本範例安裝 Astra Trident 、以搭配 Astra 叢集上的雲端身分識別使用。

```
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 叢集上的雲端身分識別

本範例安裝 Astra Trident 、以搭配 Astra 叢集上的雲端身分識別使用。

```
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/astratrident-role'"
```

版權資訊

商標資訊