開始使用
Astra Trident

NetApp
April 16, 2024

目錄

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"。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

請先檢閱支援的前端、後端及主機組態、開始著手。

> 若要瞭解Astra Trident使用的連接埠、請參閱 "請按這裡"。

Kubernetes 1.25的重要資訊

Astra Trident 22.10支援Kubernetes 1.25。升級至Kubernetes 1.25之前、您必須先升級至Astra Trident 22.10。

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 1.19 - 1.24
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.8、4.9、4.10、4.11

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 1.19 - 1.24
- OpenShift 4.8、4.9、4.10、4.11

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。

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9.3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	1.19 - 1.24	否
Volume Snapshot	1.19 - 1.24	否
來自Volume Snapshot的PVc	1.19 - 1.24	否
iSCSI PV調整大小	1.19 - 1.24	否
資訊雙向CHAP ONTAP	1.19 - 1.24	否
動態匯出原則	1.19 - 1.24	否
Trident運算子	1.19 - 1.24	否
自動工作者節點準備（試用版）	1.19 - 1.24	否
csi拓撲	1.19 - 1.24	否

已測試的主機作業系統

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。

儘管Astra Trident並未正式「支援」特定作業系統、但下列Linux套裝作業系統仍可正常運作：

- 受OpenShift Container Platform支援的RedHat CoreOS（RMCOS）版本

- RHEL或CentOS 7.
- Ubuntu 18.04或更新版本（最新版本22.04）

。 `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

視使用中的後端而定、應該在叢集中的所有員工上安裝NFS和/或iSCSI公用程式。請參閱 "請按這裡" 以取得更多資訊。

儲存系統組態

Astra Trident可能需要對儲存系統進行一些變更、才能使用後端組態。請參閱 "請按這裡" 以取得詳細資料。

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
v1.19.0版	• NetApp/Trident：22.07.0 • NetApp/Trident自動支援：22.07 • k8s.gcr.io/sig-storage / csi佈建程式：v2.2.2 • k8s.gcr.io/sig-storage / csi附加程式：v3.5.0 • k8s.gcr.io/sig-storage / csi大小調整：v1.5.0 • k8s.gcr.io/sig-storage / csi快照記錄：v3.0.3 • k8s.gcr.io/sig-storage / csi節點驅動程式登錄程式：v2.5 • NetApp/Trident營運者：22.07.0（選用）
v1.20.0	• NetApp/Trident：22.07.0 • NetApp/Trident自動支援：22.07 • k8s.gcr.io/sig-storage / csi佈建程式：v3.2.1 • k8s.gcr.io/sig-storage / csi附加程式：v3.5.0 • k8s.gcr.io/sig-storage / csi大小調整：v1.5.0 • k8s.gcr.io/sig-storage / cscs-snapshotter：v6.0.1 • k8s.gcr.io/sig-storage / csi節點驅動程式登錄程式：v2.5 • NetApp/Trident營運者：22.07.0（選用）

Kubernetes版本	Container映像
1.21.0版	• NetApp/Trident：22.07.0 • NetApp/Trident自動支援：22.07 • k8s.gcr.io/sig-storage / csi佈建程式：v3.2.1 • k8s.gcr.io/sig-storage / csi附加程式：v3.5.0 • k8s.gcr.io/sig-storage / csi大小調整：v1.5.0 • k8s.gcr.io/sig-storage / cscs-snapshotter：v6.0.1 • k8s.gcr.io/sig-storage / csi節點驅動程式登錄程式：v2.5 • NetApp/Trident營運者：22.07.0（選用）
1.22.0版	• NetApp/Trident：22.07.0 • NetApp/Trident自動支援：22.07 • k8s.gcr.io/sig-storage / csi佈建程式：v3.2.1 • k8s.gcr.io/sig-storage / csi附加程式：v3.5.0 • k8s.gcr.io/sig-storage / csi大小調整：v1.5.0 • k8s.gcr.io/sig-storage / cscs-snapshotter：v6.0.1 • k8s.gcr.io/sig-storage / csi節點驅動程式登錄程式：v2.5 • NetApp/Trident營運者：22.07.0（選用）
1.23.0版	• NetApp/Trident：22.07.0 • NetApp/Trident自動支援：22.07 • k8s.gcr.io/sig-storage / csi佈建程式：v3.2.1 • k8s.gcr.io/sig-storage / csi附加程式：v3.5.0 • k8s.gcr.io/sig-storage / csi大小調整：v1.5.0 • k8s.gcr.io/sig-storage / cscs-snapshotter：v6.0.1 • k8s.gcr.io/sig-storage / csi節點驅動程式登錄程式：v2.5 • NetApp/Trident營運者：22.07.0（選用）

Kubernetes版本	Container映像
1.24.0版	• NetApp/Trident：22.07.0 • NetApp/Trident自動支援：22.07 • k8s.gcr.io/sig-storage / csi佈建程式：v3.2.1 • k8s.gcr.io/sig-storage / csi附加程式：v3.5.0 • k8s.gcr.io/sig-storage / csi大小調整：v1.5.0 • k8s.gcr.io/sig-storage / cscs-snapshotter：v6.0.1 • k8s.gcr.io/sig-storage / csi節點驅動程式登錄程式：v2.5 • NetApp/Trident營運者：22.07.0（選用）

在Kubernetes版本1.20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。

部署總覽

您可以使用Trident運算子或搭配部署Astra Trident `tridentctl`。

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。

Kubernetes 1.25的重要資訊

Astra Trident 22.10支援Kubernetes 1.25。升級至Kubernetes 1.25之前、您必須先升級至Astra Trident 22.10。

選擇部署方法

若要判斷要使用哪種部署方法、請考慮下列事項：

為什麼我應該使用Trident運算子？

。 "Trident運算子" 是動態管理Astra Trident資源及自動化設定階段的絕佳方法。必須滿足一些先決條件。請參閱 "需求"。

Trident營運者提供以下幾項優點：

自我修復功能

您可以監控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。當操作員設定為部署時 `trident-operator-<generated-id>` Pod已建立。此Pod會建立關聯 `TridentOrchestrator`

含Astra Trident安裝的CR可確保只有一個作用中 `TridentOrchestrator`。換句話說、操作人員可確保叢集中只有一個Astra Trident執行個體、並控制其設定、確保安裝具有冪等特性。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。

輕鬆更新現有安裝

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。

自動處理**Kubernetes**升級

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**Cloud Manager**管理**Kubernetes**叢集

與 "使用Cloud Manager的Astra Trident"、您可以升級至最新版的Astra Trident、新增及管理儲存類別、並將其連線至工作環境、以及使用Cloud Backup Service NetApp備份持續的Volume。Cloud Manager可手動或使用Helm、使用Trident營運者來支援Astra Trident部署。

為什麼要使用**Helm**？

如果您有其他使用Helm管理的應用程式、從Astra Trident 21.01開始、您也可以使用Helm來管理部署。

我應該何時使用 `tridenctl`？

如果您現有的部署必須升級至、或是想要高度自訂部署、請參閱使用 "試用"。這是部署Astra Trident的傳統方法。

在部署方法之間移動的考量

不難想像需要在部署方法之間移動的情況。在嘗試從移出之前、您應該考慮下列事項 `tridentctl` 部署至以營運者為基礎的部署、反之亦然：

- 請務必使用相同的方法來解除安裝Astra Trident。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。
- 如果您想要移除及使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。

瞭解部署模式

有三種方法可以部署Astra Trident。

標準部署

在Kubernetes叢集上部署Trident會導致Astra Trident安裝程式執行兩項作業：

- 透過網際網路擷取Container映像
- 建立部署和/或節點取消設定、在Kubernetes叢集中的所有合格節點上執行Astra Trident Pod。

這類的標準部署可透過兩種不同方式執行：

- 使用 `tridentctl install`
- 使用Trident運算子。您可以手動或使用Helm來部署Trident運算子。

這種安裝模式是安裝Astra Trident的最簡單方法、適用於大多數不受網路限制的環境。

離線部署

若要執行無線部署、您可以使用 `--image-registry` 叫用時顯示旗標 `tridentctl install` 指向私有映像登錄。如果使用Trident運算子進行部署、您也可以指定 `spec.imageRegistry` 在您的中 `TridentOrchestrator`。此登錄應包含 "Trident影像"、 "Trident AutoSupport 的圖片"以及Kubernetes版本所需的csi sidecar映像。

若要自訂部署、您可以使用 `tridentctl` 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。

如需自訂部署的詳細資訊、請參閱下列連結：

- "自訂您的營運者型部署"
  *

> 如果您使用的是私有映像儲存庫、則應該新增 `/sig-storage` 到私有登錄URL的結尾。使用的私有登錄時 `tridentctl` 部署、您應該使用 `--trident-image` 和 `--autosupport-image` 與搭配使用 `--image-registry`。如果您使用Trident運算子來部署Astra Trident、請確定Orchestrator CR包含在內 `tridentImage` 和 `autosupportImage` 安裝參數。

遠端部署

以下是遠端部署程序的高階概觀：

- 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。

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：

與Trident營運者一起部署

您可以使用Trident運算子來部署Astra Trident。您可以使用下列兩種方式之一來部署Trident運算子：

- 使用Trident "掌舵表"：Helm圖表可部署Trident運算子、並在單一步驟中安裝Trident。
- 手動：Trident提供 "bunder.yaml" 可用於安裝運算子及建立關聯物件的檔案。

> 如果您尚未熟悉 "基本概念"現在正是這麼做的好時機。

您需要的產品

若要部署Astra Trident、必須符合下列先決條件：

- 您擁有執行Kubernetes 1.19 - 1.24之受支援Kubernetes叢集的完整權限。
- 您可以存取支援的NetApp儲存系統。
- 您可以從所有Kubernetes工作節點掛載磁碟區。
- 您有一部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。
- 您已設定 `KUBECONFIG` 指向Kubernetes叢集組態的環境變數。
- 您已啟用 "具備Astra Trident所需的閘道"。
- 如果您使用Kubernetes搭配Docker Enterprise、 "請依照他們的步驟啟用CLI存取"。

您知道嗎？太棒了！讓我們開始吧。

部署Trident操作員、並使用Helm安裝Astra Trident

使用Helm執行列出的步驟來部署Trident運算子。

您需要的產品

除了上述先決條件之外、若要使用Helm部署Trident運算子、您還需要下列項目：

- Kubernetes 1.19 - 1.24

- Helm版本3

步驟

1. 新增Trident的Helm儲存庫：

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

2. 使用 `helm install` 命令並指定部署名稱。請參閱下列範例：

```
helm install <release-name> netapp-trident/trident-operator --version
22.4.0 --create-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。

安裝期間有兩種傳遞組態資料的方法：

- `--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`：在命令行中指定覆蓋。

例如、變更的預設值 `debug`請執行下列步驟 `--set` 命令：

```
helm install <name> netapp-trident/trident-operator --version 22.7.0 --set
tridentDebug=true
```

。 `values.yaml` 檔案是Helm圖表的一部分、提供金鑰清單及其預設值。

`helm list` 顯示安裝的詳細資料、例如名稱、命名空間、圖表、狀態、 應用程式版本、修訂編號等。

手動部署Trident運算子

執行所列步驟、手動部署Trident運算子。

步驟1：判斷Kubernetes叢集的資格

首先您需要登入Linux主機、然後確認它正在管理_運作_、 "支援的Kubernetes叢集" 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`。

若要驗證Kubernetes版本、請執行下列命令：

```
kubectl version
```

若要查看您是否具有Kubernetes叢集管理員權限、請執行下列命令：

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

若要驗證是否可以從Docker Hub啟動使用映像的Pod、並透過Pod網路連線至儲存系統、請執行下列命令：

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

步驟**2**：下載並設定營運者

> 從21.01開始、Trident運算子就是叢集範圍。若要使用Trident運算子來安裝Trident、必須建立 `TridentOrchestrator` 自訂資源定義（CRD）和定義其他資源。在安裝Astra Trident之前、您應該先執行這些步驟來設定操作員。

1. 請從下載並擷取最新版本的Trident安裝程式套件 "GitHub的_Assets區段"。

   ```
   wget
   https://github.com/NetApp/trident/releases/download/v22.04.0/trident-
   installer-22.04.0.tar.gz
   tar -xf trident-installer-22.04.0.tar.gz
   cd trident-installer
   ```

2. 使用適當的CRD資訊清單來建立 `TridentOrchestrator` 客戶需求日然後建立 `TridentOrchestrator` 稍後再自訂資源以產生操作者的安裝。

   執行下列命令：

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

3. 之後 `TridentOrchestrator` 建立客戶需求日之後、請建立下列操作員部署所需的資源：

   - 營運者的服務帳戶
   - 叢集角色和叢集角色繫結至服務帳戶
   - 專屬的PodSecurity原則
   - 營運者本身

   Trident安裝程式包含定義這些資源的資訊清單。根據預設、操作員會部署在中 `trident` 命名空間。如果是 `trident` 命名空間不存在、請使用下列資訊清單來建立命名空間。

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

4. 可在非預設名稱空間中部署運算子 `trident` 命名空間、您應該更新 `serviceaccount.yaml`、`clusterrolebinding.yaml` 和 `operator.yaml` 提供清單並產生您的 `bundle.yaml`。

   執行下列命令以更新Yaml清單並產生您的 `bundle.yaml` 使用 `kustomization.yaml`：

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

   執行下列命令以建立資源並部署營運者：

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

5. 若要在部署後驗證操作員的狀態、請執行下列步驟：

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

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

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

NAME                                READY   STATUS             RESTARTS
AGE
trident-operator-54cb664d-lnjxh     1/1     Running            0
3m
```

營運者部署成功建立一個在叢集中其中一個工作節點上執行的Pod。

> 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3：建立 `TridentOrchestrator` 並安裝Trident

您現在可以使用運算子來安裝Astra Trident！這需要建立 `TridentOrchestrator`。Trident安裝程式隨附建立的範例定義 `TridentOrchestrator`。這會啟動中的安裝 `trident` 命名空間。

```
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:21.04
    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:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Trident運算子可讓您使用中的屬性、自訂Astra Trident的安裝方式 `TridentOrchestrator` 規格請參閱 "自訂您的Trident部署"。

的狀態 `TridentOrchestrator` 指出安裝是否成功、並顯示安裝的Trident版本。

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。

安裝期間的狀態 `TridentOrchestrator` 變更來源 `Installing` 至 `Installed`。如果您觀察到 `Failed` 狀態且操作員無法自行恢復、您應該檢查操作員的記錄。請參閱 "疑難排解" 區段。

您可以查看已建立的Pod、確認Astra Trident安裝是否已完成：

```
kubectl get pod -n trident

NAME                                  READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw          5/5     Running   0           1m
trident-csi-mr6zc                     2/2     Running   0           1m
trident-csi-xrp7w                     2/2     Running   0           1m
trident-csi-zh2jt                     2/2     Running   0           1m
trident-operator-766f7b8658-ldzsv     1/1     Running   0           3m
```

您也可以使用 `tridentctl` 檢查安裝的Astra Trident版本。

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

現在您可以繼續建立後端。請參閱 "部署後工作"。

如需部署期間的疑難排解問題、請參閱 "疑難排解" 區段。

自訂Trident營運者部署

Trident運算子可讓您使用中的屬性、自訂Astra Trident的安裝方式 `TridentOrchestrator` 規格

請參閱下表以取得屬性清單：

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
`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：21.04.0」
`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	
`controllerPluginNodeSelector`	執行Trident控制器csi外掛程式之Pod的其他節點選取器。格式與pod.spec.nodeSelector相同。	無預設值；選用
`controllerPluginTolerations`	覆寫執行Trident控制器csi外掛程式之Pod的容錯能力。格式與po.spec.TBolerations相同。	無預設值；選用
`nodePluginNodeSelector`	執行Trident Node SCSI外掛程式之Pod的其他節點選取器。格式與pod.spec.nodeSelector相同。	無預設值；選用

參數	說明	預設
`nodePluginTolerations`	覆寫執行Trident Node SCSI外掛程式之Pod的容錯能力。格式與po.spec.TBolerations相同。	無預設值；選用

`spec.namespace` 在中指定 `TridentOrchestrator` 以表示安裝在哪個命名空間Astra Trident。此參數*無法在安裝Astra Trident之後更新*。嘗試這麼做會導致的狀態 `TridentOrchestrator` 變更為 `Failed`。Astra Trident不打算跨命名空間移轉。

如需格式化Pod參數的詳細資訊、請參閱 "將Pod指派給節點"。

您可以在定義時使用上述屬性 `TridentOrchestrator` 以自訂安裝。範例如下：

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

以下是另一個範例、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
```

如果您想要自訂安裝內容以外的內容 `TridentOrchestrator` 參數允許、您應該考慮使用 `tridentctl` 產生可視需要修改的自訂Y反 洗錢清單。

使用tridentctl部署

您可以使用部署Astra Trident `tridentctl`。

> 如果您尚未熟悉 "基本概念"現在正是這麼做的好時機。

> 若要自訂部署、請參閱 "請按這裡"。

您需要的產品

若要部署Astra Trident、必須符合下列先決條件：

- 您擁有支援的Kubernetes叢集的完整權限。
- 您可以存取支援的NetApp儲存系統。
- 您可以從所有Kubernetes工作節點掛載磁碟區。
- 您有一部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。
- 您已設定 `KUBECONFIG` 指向Kubernetes叢集組態的環境變數。
- 您已啟用 "具備Astra Trident所需的閘道"。
- 如果您使用Kubernetes搭配Docker Enterprise、 "請依照他們的步驟啟用CLI存取"。

您知道嗎？太棒了！讓我們開始吧。

> 如需自訂部署的相關資訊、請參閱 "請按這裡"。

步驟1：判斷Kubernetes叢集的資格

首先您需要登入Linux主機、然後驗證它是否正在管理_工作_、 "支援的Kubernetes叢集" 您擁有必要的權限。

> 使用OpenShift、您就能使用 `oc` 而非 `kubectl` 在後續的所有範例中、您應該先執行*系統：admin*登入 `oc login -u system:admin` 或 `oc login -u kube-admin`。

若要檢查Kubernetes版本、請執行下列命令：

```
kubectl version
```

若要查看您是否具有Kubernetes叢集管理員權限、請執行下列命令：

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

若要驗證是否可以從Docker Hub啟動使用映像的Pod、並透過Pod網路連線至儲存系統、請執行下列命令：

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

識別Kubernetes伺服器版本。安裝Astra Trident時會使用此功能。

步驟2：下載並解壓縮安裝程式

> Trident安裝程式會建立Trident pod、設定用來維護其狀態的CRD物件、並初始化執行動作的csi sidecars、例如資源配置和將磁碟區附加至叢集主機。

您可以從下載並擷取最新版本的Trident安裝程式套件 "GitHub的_Assets區段"。

例如、如果最新版本為21.07.1：

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

步驟3：安裝Astra Trident

執行、在所需的命名空間中安裝Astra Trident `tridentctl install` 命令。

```
./tridentctl install -n trident
....
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-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

安裝程式完成後、看起來會是這樣。根據Kubernetes叢集中的節點數量、您可能會看到更多的Pod：

```
kubectl get pod -n trident
NAME                              READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx      4/4     Running   0          5m29s
trident-csi-vgc8n                 2/2     Running   0          5m29s

./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

如果您看到類似上述範例的輸出、表示您已完成此步驟、但尚未完整設定Astra Trident。請繼續進行下一步。請參閱 "部署後工作"。

不過、如果安裝程式未成功完成、或您看不到*執行中* `trident-csi-<generated id>`、平台尚未安裝。

> 如需部署期間的疑難排解問題、請參閱 "疑難排解" 區段。

自訂試用部署

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？

部署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。"