Gérer et surveiller Trident

Trident

NetApp
July 01, 2026

Sommaire

Gérer et surveiller Trident

Mettre à niveau Trident

Mettre à niveau Trident

À compter de la version 24.02, Trident adopte un cycle de publication de quatre mois, avec trois mises à jour majeures chaque année civile. Chaque nouvelle version s'appuie sur les précédentes et propose de nouvelles fonctionnalités, des améliorations de performances, des corrections de bogues et des améliorations. Nous vous encourageons à effectuer une mise à jour au moins une fois par an afin de profiter des nouvelles fonctionnalités de Trident.

Considérations avant la mise à niveau

Lors de la mise à niveau vers la dernière version de Trident, tenez compte des points suivants :

- Une seule instance de Trident doit être installée dans tous les espaces de noms d'un cluster Kubernetes donné.
- Trident 23.07 et versions ultérieures nécessitent des instantanés de volume v1 et ne prennent plus en charge les instantanés alpha ou beta.
- Lors de la mise à niveau, il est important de fournir `parameter.fsType` dans `StorageClasses` utilisé par Trident. Vous pouvez supprimer et recréer `StorageClasses` sans interrompre les volumes existants.
  - Il s'agit d'une **exigence** pour appliquer "contextes de sécurité" aux volumes SAN.
  - Le répertoire sample input contient des exemples, tels que `storage-class-basic.yaml.templ` et `storage-class-bronze-default.yaml`.
  - Pour plus d'informations, consultez "Problèmes connus".

Étape 1 : Sélectionnez une version

Les versions de Trident suivent une `YY.MM` convention de nommage basée sur la date, où « YY » sont les deux derniers chiffres de l'année et « MM » le mois. Les versions dot suivent une `YY.MM.X` convention, où « X » est le niveau de correctif. Vous sélectionnerez la version vers laquelle effectuer la mise à niveau en fonction de la version à partir de laquelle vous effectuez la mise à niveau.

- Vous pouvez effectuer une mise à niveau directe vers n'importe quelle version cible située dans un intervalle de quatre versions maximum par rapport à votre version installée. Par exemple, vous pouvez effectuer une mise à niveau directe de 24.06 (ou de toute version dérivée de 24.06) vers 25.06.
- Si vous effectuez une mise à niveau depuis une version en dehors de la fenêtre de quatre versions, procédez à une mise à niveau en plusieurs étapes. Utilisez les instructions de mise à niveau pour la "version antérieure" version à partir de laquelle vous effectuez la mise à niveau afin de passer à la version la plus récente qui correspond à la fenêtre de quatre versions. Par exemple, si vous utilisez la version 23.07 et souhaitez passer à la version 25.06 :
  a. Première mise à niveau de 23.07 à 24.06.
  b. Ensuite, passez de 24.06 à 25.06.

Lors de la mise à niveau à l'aide de l'opérateur Trident sur OpenShift Container Platform, vous devez passer à Trident 21.01.1 ou une version ultérieure. L'opérateur Trident publié avec la version 21.01.0 contient un problème connu qui a été corrigé dans la version 21.01.1. Pour plus de détails, consultez le "détails du problème sur GitHub".

Étape 2 : Déterminez la méthode d'installation d'origine

Pour déterminer la version que vous avez utilisée pour installer Trident :

1. Utilisez `kubectl get pods -n trident` pour examiner les pods.
   - S'il n'y a pas de module opérateur, Trident a été installé à l'aide de `tridentctl`.
   - S'il existe un pod opérateur, Trident a été installé à l'aide de l'opérateur Trident, soit manuellement, soit à l'aide de Helm.
2. S'il existe un pod opérateur, utilisez `kubectl describe torc` pour déterminer si Trident a été installé à l'aide de Helm.
   - S'il y a une étiquette Helm, Trident a été installé avec Helm.
   - S'il n'y a pas d'étiquette Helm, Trident a été installé manuellement à l'aide de l'opérateur Trident.

Étape 3 : Sélectionnez une méthode de mise à niveau

En règle générale, vous devez effectuer la mise à niveau en utilisant la même méthode que celle employée lors de l'installation initiale, cependant vous pouvez "passer d'une méthode d'installation à une autre". Il existe deux options pour mettre à niveau Trident.

- "Mise à niveau à l'aide de l'opérateur Trident"

Nous vous suggérons de vérifier "Comprendre le workflow de mise à niveau de l'opérateur" avant de procéder à la mise à niveau avec l'opérateur.

*

Mise à niveau avec l'opérateur

Comprendre le workflow de mise à niveau de l'opérateur

Avant d'utiliser l'opérateur Trident pour mettre à niveau Trident, vous devez comprendre les processus en arrière-plan qui se produisent lors de la mise à niveau. Cela inclut les modifications apportées au contrôleur Trident, au pod du contrôleur, aux pods du nœud et au DaemonSet du nœud qui permettent les mises à jour progressives.

Gestion de la mise à niveau de l'opérateur Trident

L'un des nombreux "avantages de l'utilisation de l'opérateur Trident" moyens d'installer et de mettre à niveau Trident est la gestion automatique des objets Trident et Kubernetes sans perturber les volumes montés existants. De cette façon, Trident peut prendre en charge les mises à niveau sans interruption, ou "*mises à jour progressives*". En particulier, l'opérateur Trident communique avec le cluster Kubernetes pour :

- Supprimez et recréez le déploiement du contrôleur Trident et le DaemonSet du nœud.

- Remplacez le Trident Controller Pod et les Trident Node Pods par de nouvelles versions.
  - Si un nœud n'est pas mis à jour, cela n'empêche pas les autres nœuds d'être mis à jour.
  - Seuls les nœuds avec un Trident Node Pod en cours d'exécution peuvent monter des volumes.

Pour plus d'informations sur l'architecture Trident sur le cluster Kubernetes, veuillez vous référer à "Architecture Trident".

Flux de travail de mise à niveau de l'opérateur

Lorsque vous lancez une mise à niveau à l'aide de l'opérateur Trident :

1. L'opérateur **Trident** :
   a. Détecte la version actuellement installée de Trident (version *n*).
   b. Met à jour tous les objets Kubernetes, y compris les CRDs, RBAC et Trident SVC.
   c. Supprime le déploiement du contrôleur Trident pour la version *n*.
   d. Crée le déploiement du contrôleur Trident pour la version *n+1*.
2. **Kubernetes** crée le pod Trident Controller pour *n+1*.
3. L'opérateur **Trident** :
   a. Supprime le DaemonSet Trident Node pour *n*. L'opérateur n'attend pas la fin du Pod Node.
   b. Crée le Daemonset de nœud Trident pour *n+1*.
4. **Kubernetes** crée des pods de nœud Trident sur les nœuds qui n'exécutent pas le pod de nœud Trident *n*. Cela garantit qu'il n'y a jamais plus d'un pod de nœud Trident, de n'importe quelle version, sur un nœud.

Mettez à niveau une installation Trident à l'aide de l'opérateur Trident ou de Helm

Vous pouvez mettre à niveau Trident à l'aide de l'opérateur Trident, soit manuellement, soit à l'aide de Helm. Vous pouvez mettre à niveau une installation de l'opérateur Trident vers une autre installation de l'opérateur Trident ou effectuer une mise à niveau d'une installation `tridentctl` vers une version opérateur Trident. Examinez "Sélectionnez une méthode de mise à niveau" avant de mettre à niveau une installation de l'opérateur Trident.

Mettre à niveau une installation manuelle

Vous pouvez effectuer une mise à niveau d'une installation d'opérateur Trident à l'échelle d'un cluster vers une autre installation d'opérateur Trident à l'échelle d'un cluster. Toutes les versions de Trident utilisent un opérateur à l'échelle d'un cluster.

Pour mettre à niveau Trident installé à l'aide de l'opérateur à portée d'espace de noms (versions 20.07 à 20.10), utilisez les instructions de mise à niveau "votre version installée" de Trident.

À propos de cette tâche

Trident fournit un fichier bundle que vous pouvez utiliser pour installer l'opérateur et créer les objets associés pour votre version de Kubernetes.

- Pour les clusters exécutant Kubernetes 1.25 ou une version ultérieure, utilisez "bundle_post_1_25.yaml".

Avant de commencer

Assurez-vous d'utiliser un cluster Kubernetes exécutant "une version Kubernetes prise en charge".

Étapes

1. Vérifiez votre version de Trident :

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

2. Mettez à jour le `operator.yaml`, `tridentorchestrator_cr.yaml`, et `post_1_25_bundle.yaml` avec le registre et les chemins d'image pour la version vers laquelle vous effectuez la mise à niveau (par exemple 25.06), ainsi que le secret correct.
3. Supprimez l'opérateur Trident qui a été utilisé pour installer l'instance Trident actuelle. Par exemple, si vous effectuez une mise à niveau depuis la version 25.02, exécutez la commande suivante :

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Si vous avez personnalisé votre installation initiale à l'aide des `TridentOrchestrator` attributs, vous pouvez modifier l'objet `TridentOrchestrator` pour modifier les paramètres d'installation. Cela peut inclure des modifications apportées pour spécifier des registres d'images Trident et CSI en miroir pour le mode hors ligne, activer les journaux de débogage ou spécifier des secrets d'extraction d'images.
5. Installez Trident en utilisant le fichier YAML de bundle approprié pour votre environnement, où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon votre version de Kubernetes. Par exemple, si vous installez Trident 25.06.0, exécutez la commande suivante :

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Modifiez le trident torc pour inclure l'image 25.06.0.

Mettre à niveau une installation Helm

Vous pouvez mettre à niveau une installation Trident Helm.

> Lors de la mise à niveau d'un cluster Kubernetes de la version 1.24 à la version 1.25 ou ultérieure sur lequel Trident est installé, vous devez mettre à jour values.yaml pour définir `excludePodSecurityPolicy` sur `true` ou ajouter `--set excludePodSecurityPolicy=true` à la commande `helm upgrade` avant de pouvoir mettre à niveau le cluster.

Si vous avez déjà mis à niveau votre cluster Kubernetes de la version 1.24 à la version 1.25 sans mettre à niveau le helm Trident, la mise à niveau du helm échouera. Pour que la mise à niveau du helm réussisse, effectuez les étapes suivantes comme prérequis :

1. Installez le plugin helm-mapkubeapis depuis https://github.com/helm/helm-mapkubeapis.

2. Effectuez un test à blanc pour la version Trident dans l'espace de noms où Trident est installé. Cela liste les ressources qui seront nettoyées.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Effectuez une exécution complète avec helm pour effectuer le nettoyage.

```
helm mapkubeapis trident  --namespace trident
```

Étapes

1. Si vous "installé Trident à l'aide de Helm" , vous pouvez utiliser `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0` pour effectuer la mise à niveau en une seule étape. Si vous n'avez pas ajouté le dépôt Helm ou si vous ne pouvez pas l'utiliser pour effectuer la mise à niveau :
   a. Téléchargez la dernière version de Trident depuis "la section *Assets* sur GitHub".
   b. Utilisez la commande `helm upgrade` où `trident-operator-26.02.0.tgz` correspond à la version vers laquelle vous souhaitez effectuer la mise à niveau.

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

> Si vous définissez des options personnalisées lors de l'installation initiale (comme la spécification de registres privés et en miroir pour Trident et les images CSI), ajoutez la `helm upgrade` commande en utilisant `--set` pour garantir que ces options sont incluses dans la commande de mise à niveau, sinon les valeurs seront réinitialisées à la valeur par défaut.

2. Exécutez `helm list` pour vérifier que le graphique et la version de l'application ont tous deux été mis à niveau. Exécutez `tridentctl logs` pour examiner les messages de débogage.

Mise à niveau d'une `tridentctl`installation vers l'opérateur Trident

Vous pouvez mettre à niveau vers la dernière version de l'opérateur Trident à partir d'une installation `tridentctl` existante. Les backends et les PVC existants seront automatiquement disponibles.

> Avant de changer de méthode d'installation, consultez "Passer d'une méthode d'installation à une autre".

Étapes

1. Téléchargez la dernière version de Trident.

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

2. Créez la `tridentorchestrator` CRD à partir du manifeste.

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

3. Déployez l'opérateur à portée de cluster dans le même espace de noms.

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

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Créer une `TridentOrchestrator` CR pour installer Trident.

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Confirmez que Trident a été mis à niveau vers la version prévue.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

Mise à niveau avec tridentctl

Vous pouvez facilement mettre à niveau une installation Trident existante en utilisant `tridentctl`.

À propos de cette tâche

La désinstallation et la réinstallation de Trident constituent une mise à niveau. Lorsque vous désinstallez Trident, la Persistent Volume Claim (PVC) et le Persistent Volume (PV) utilisés par le déploiement Trident ne sont pas supprimés. Les PV déjà provisionnés restent disponibles pendant que Trident est hors ligne, et Trident provisionnera des volumes pour toutes les PVC créées entre-temps après son retour en ligne.

Avant de commencer

Vérifiez "Sélectionnez une méthode de mise à niveau" avant la mise à niveau en utilisant `tridentctl`.

Étapes

1. Exécutez la commande de désinstallation dans `tridentctl` pour supprimer toutes les ressources associées à Trident à l'exception des CRD et des objets associés.

```
./tridentctl uninstall -n <namespace>
```

2. Réinstallez Trident. Consultez "Installez Trident à l'aide de tridentctl".

> N'interrompez pas le processus de mise à niveau. Assurez-vous que l'installateur s'exécute jusqu'à la fin.

Gérez Trident à l'aide de tridentctl

Le "Trident bundle d'installation" inclut l' `tridentctl`utilitaire en ligne de commande pour fournir un accès simple à Trident. Les utilisateurs Kubernetes disposant de privilèges suffisants peuvent l'utiliser pour installer Trident ou gérer l'espace de noms qui contient le pod Trident.

Commandes et drapeaux globaux

Vous pouvez exécuter `tridentctl help` pour obtenir une liste des commandes disponibles pour `tridentctl` ou ajouter le drapeau `--help` à n'importe quelle commande pour obtenir une liste des options et des drapeaux pour cette commande spécifique.

```
tridentctl [command] [--optional-flag]
```

L'utilitaire Trident `tridentctl` prend en charge les commandes et indicateurs globaux suivants.

Commandes

`create`
Ajouter une ressource à Trident.

`delete`
Supprimez une ou plusieurs ressources de Trident.

`get`
Obtenez une ou plusieurs ressources de Trident.

`help`
Aide sur n'importe quelle commande.

`images`
Imprimez un tableau des images de conteneurs nécessaires à Trident.

`import`
Importer une ressource existante dans Trident.

`install`
Installez Trident.

`logs`
Imprimez les journaux de Trident.

`send`
Envoyez une ressource depuis Trident.

`uninstall`
Désinstallez Trident.

`update`
Modifier une ressource dans Trident.

`update backend state`
Suspendre temporairement les opérations en arrière-plan.

`upgrade`
Mettre à niveau une ressource dans Trident.

`version`
Imprimez la version de Trident.

Drapeaux globaux

-d, --debug
Sortie de débogage.

-h, --help
Aide pour `tridentctl`.

-k, --kubeconfig string
Spécifiez le `KUBECONFIG` chemin pour exécuter des commandes localement ou d'un cluster Kubernetes à un autre.

> Vous pouvez également exporter la `KUBECONFIG` variable pour qu'elle pointe vers un cluster Kubernetes spécifique et émettre `tridentctl` commandes à ce cluster.

-n, --namespace string
Espace de noms du déploiement Trident.

-o, --output string
Format de sortie. L'un de json|yaml|name|wide|ps (par défaut).

-s, --server string
Adresse/port de l'interface REST de Trident.

> L'interface REST de Trident peut être configurée pour écouter et servir uniquement à 127.0.0.1 (pour IPv4) ou [::1] (pour IPv6).

Options de commande et drapeaux

créer

Utilisez la `create` commande pour ajouter une ressource à Trident.

```
tridentctl create [option]
```

Options

`backend`: Ajouter un backend à Trident.

supprimer

Utilisez la commande `delete` pour supprimer une ou plusieurs ressources de Trident.

```
tridentctl delete [option]
```

Options

`backend`: Supprimer un ou plusieurs backends de stockage de Trident.
`snapshot`: Supprimer un ou plusieurs snapshots de volume de Trident.
`storageclass`: Supprimer une ou plusieurs classes de stockage de Trident.

`volume`: Supprimer un ou plusieurs volumes de stockage de Trident.

obtenir

Utilisez la `get` commande pour obtenir une ou plusieurs ressources de Trident.

```
tridentctl get [option]
```

Options

`backend`: Obtenir un ou plusieurs backends de stockage depuis Trident.
`snapshot`: Obtenir un ou plusieurs snapshots depuis Trident.
`storageclass`: Obtenir une ou plusieurs classes de stockage depuis Trident.
`volume`: Obtenir un ou plusieurs volumes depuis Trident.

Drapeaux

`-h`, `--help` : Aide pour les volumes.
`--parentOfSubordinate string` : Limiter la requête au volume source subordonné.
`--subordinateOf string` : Limiter la requête aux subordonnés du volume.

images

Utilisez `images` les indicateurs pour afficher un tableau des images de conteneur dont Trident a besoin.

```
tridentctl images [flags]
```

Drapeaux

`-h`, `--help` : Aide pour les images.
`-v`, `--k8s-version string` : Version sémantique du cluster Kubernetes.

importer un volume

Utilisez la `import volume` commande pour importer un volume existant dans Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Alias

`volume`, `v`

Drapeaux

`-f`, `--filename string` : Chemin vers le fichier PVC YAML ou JSON.
`-h`, `--help` : Aide pour le volume.
`--no-manage` : Créer uniquement PV/PVC. Ne pas supposer la gestion du cycle de vie du volume.

installer

Utilisez les `install` flags pour installer Trident.

```
tridentctl install [flags]
```

Drapeaux

`--autosupport-image string` : L'image de conteneur pour Autosupport Telemetry (par défaut « netapp/trident autosupport:<current-version> »).
`--autosupport-proxy string` : L'adresse/port d'un proxy pour l'envoi de la télémétrie Autosupport.
`--enable-node-prep` : Tenter d'installer les paquets requis sur les nœuds.
`--generate-custom-yaml` : Générer des fichiers YAML sans rien installer.
`-h, --help` : Aide pour l'installation.
`--http-request-timeout` : Remplacer le délai d'expiration des requêtes HTTP pour l'API REST du contrôleur Trident (par défaut 1m30s).
`--image-registry string` : L'adresse/port d'un registre d'images interne.
`--k8s-timeout duration` : Le délai d'expiration pour toutes les opérations Kubernetes (par défaut 3m0s).
`--kubelet-dir string` : L'emplacement hôte de l'état interne de kubelet (par défaut « /var/lib/kubelet »).
`--log-format string` : Le format de journalisation de Trident (text, json) (par défaut « text »).
`--node-prep` : Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge. À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.**
`--pv string` : Le nom du PV hérité utilisé par Trident, assurez-vous qu'il n'existe pas (par défaut « trident »).
`--pvc string` : Le nom du PVC hérité utilisé par Trident, assurez-vous qu'il n'existe pas (par défaut « trident »).
`--silence-autosupport` : Ne pas envoyer automatiquement les bundles autosupport à NetApp (par défaut true).
`--silent` : Désactiver la plupart des sorties pendant l'installation.
`--trident-image string` : L'image Trident à installer.
`--k8s-api-qps` : La limite de requêtes par seconde (QPS) pour les requêtes API Kubernetes (par défaut 100 ; optionnel).
`--use-custom-yaml` : Utiliser tout fichier YAML existant dans le répertoire de configuration.
`--use-ipv6` : Utiliser IPv6 pour la communication de Trident.

journaux

Utilisez `logs` les indicateurs pour imprimer les journaux de Trident.

```
tridentctl logs [flags]
```

Drapeaux

`-a, --archive` : Créer une archive de support contenant tous les journaux, sauf indication contraire.
`-h, --help` : Aide pour les journaux.
`-l, --log string` : Journal Trident à afficher. Un de trident|auto|trident-operator|all (par défaut « auto »).
`--node string` : Le nom du nœud Kubernetes à partir duquel collecter les journaux des pods du nœud.
`-p, --previous` : Récupérer les journaux de l'instance précédente du conteneur si elle existe.
`--sidecars` : Récupérer les journaux des conteneurs sidecar.

envoyer

Utilisez la `send` commande pour envoyer une ressource depuis Trident.

```
tridentctl send [option]
```

Options

`autosupport`: Envoyer une archive Autosupport à NetApp.

désinstaller

Utilisez `uninstall` les options pour désinstaller Trident.

```
tridentctl uninstall [flags]
```

Drapeaux

`-h, --help`: Aide à la désinstallation.
`--silent`: Désactive la plupart des messages pendant la désinstallation.

mise à jour

Utilisez la `update` commande pour modifier une ressource dans Trident.

```
tridentctl update [option]
```

Options

`backend`: Mettre à jour un backend dans Trident.

mettre à jour l'état du backend

Utilisez la commande `update backend state` pour suspendre ou reprendre les opérations en arrière-plan.

```
tridentctl update backend state <backend-name> [flag]
```

Points à considérer

- Si un backend est créé à l'aide d'un TridentBackendConfig (tbc), le backend ne peut pas être mis à jour à l'aide d'un `backend.json` fichier.
- Si le `userState` a été défini dans un tbc, il ne peut pas être modifié à l'aide de la commande `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Pour retrouver la possibilité de définir le `userState` via tridentctl après qu'il a été défini via tbc, le champ `userState` doit être supprimé du tbc. Cela peut être fait en utilisant la commande `kubectl edit tbc`. Après la suppression du champ `userState`, vous pouvez utiliser la commande `tridentctl update backend state` pour modifier le `userState` d'un backend.
- Utilisez le `tridentctl update backend state` pour modifier le `userState`. Vous pouvez également mettre à jour le `userState` en utilisant `TridentBackendConfig` ou `backend.json` fichier ; cela déclenche une réinitialisation complète du backend et peut prendre du temps.

  #### Drapeaux

  `-h, --help`: Aide pour l'état du backend.
  `--user-state`: Définir sur `suspended` pour mettre en pause les opérations du backend. Définir sur `normal` pour reprendre les opérations du backend. Lorsque défini sur `suspended` :

- `AddVolume` and `Import Volume` sont en pause.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`,

`ReconcileNodeAccess` restent disponibles.

Vous pouvez également mettre à jour l'état du backend à l'aide du champ `userState` dans le fichier de configuration du backend `TridentBackendConfig` ou `backend.json`. Pour plus d'informations, consultez "Options pour la gestion des backends" et "Effectuez la gestion du backend avec kubectl".

Exemple :

JSON

Suivez ces étapes pour mettre à jour le `userState` en utilisant le fichier `backend.json` :

1. Modifiez le `backend.json` fichier pour inclure le `userState` champ avec sa valeur définie sur 'suspended'.
2. Mettez à jour le backend en utilisant la `tridentctl update backend` commande et le chemin vers le fichier `backend.json` mis à jour.

   Exemple: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

Vous pouvez modifier le tbc après son application à l'aide de la `kubectl edit <tbc-name> -n <namespace>` commande. L'exemple suivant met à jour l'état du backend pour le suspendre à l'aide de l'option `userState: suspended` :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

version

Utilisez `version` les options pour afficher la version de `tridentctl` et le service Trident en cours d'exécution.

```
tridentctl version [flags]
```

Drapeaux

`--client`: Version client uniquement (aucun serveur requis).
`-h, --help`: Aide pour version.

Prise en charge des plugins

Tridentctl prend en charge les plugins similaires à kubectl. Tridentctl détecte un plugin si le nom du fichier binaire du plugin suit le schéma « tridentctl-<plugin> », et que le binaire se trouve dans un dossier répertorié dans la variable d'environnement PATH. Tous les plugins détectés sont listés dans la section plugin de l'aide de tridentctl. Vous pouvez également limiter la recherche en spécifiant un dossier de plugin dans la variable d'environnement TRIDENTCTL_PLUGIN_PATH (Example: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Si la variable est utilisée, tridentctl recherche uniquement dans le dossier spécifié.

Surveiller Trident

Trident fournit un ensemble de points de terminaison de métriques Prometheus que vous pouvez utiliser pour surveiller les performances de Trident.

Aperçu

Les indicateurs fournis par Trident vous permettent de faire ce qui suit :

- Gardez un œil sur la santé et la configuration de Trident. Vous pouvez examiner le succès des opérations et vérifier si Trident peut communiquer avec les backends comme prévu.
- Examinez les informations d'utilisation du backend et comprenez combien de volumes sont provisionnés sur un backend et la quantité d'espace consommée, et ainsi de suite.
- Maintenir une cartographie de la quantité de volumes provisionnés sur les backends disponibles.
- Suivez les performances. Vous pouvez voir combien de temps il faut à Trident pour communiquer avec les backends et effectuer des opérations.

> Par défaut, les métriques de Trident sont exposées sur le port cible `8001` à l'endpoint `/metrics`. Ces métriques sont **activées par défaut** lors de l'installation de Trident. Vous pouvez également configurer la consommation des métriques Trident via HTTPS sur le port `8444`.

Ce dont vous aurez besoin

- Un cluster Kubernetes avec Trident installé.
- Une instance Prometheus. Il peut s'agir d'un "déploiement Prometheus conteneurisé" ou vous pouvez choisir d'exécuter Prometheus en tant que "application native".

Étape 1 : Définir une cible Prometheus

Vous devez définir une cible Prometheus pour collecter les métriques et obtenir des informations sur les backends gérés par Trident, les volumes qu'il crée, etc. Voir "Documentation de l'opérateur Prometheus".

Étape 2 : Créer un ServiceMonitor Prometheus

Pour consommer les métriques Trident, vous devez créer un Prometheus ServiceMonitor qui surveille le `trident-csi` service et écoute sur le `metrics` port. Un exemple de ServiceMonitor ressemble à ceci :

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Cette définition de ServiceMonitor récupère les métriques renvoyées par le service `trident-csi` et recherche spécifiquement le point de terminaison `metrics` du service. Par conséquent, Prometheus est désormais configuré pour comprendre les métriques de Trident.

Outre les métriques directement accessibles depuis Trident, kubelet expose de nombreuses `kubelet_volume_*` métriques via son propre point de terminaison de métriques. Kubelet peut fournir des informations sur les volumes attachés, les pods et d'autres opérations internes qu'il gère. Consultez "ici".

Consommer les métriques Trident via HTTPS

Pour consommer les métriques Trident via HTTPS (port 8444), vous devez modifier la définition ServiceMonitor afin d'inclure la configuration TLS. Vous devez également copier le `trident-csi` secret de l' `trident` espace de noms vers l'espace de noms où Prometheus est en cours d'exécution. Vous pouvez le faire à l'aide de la commande suivante :

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Un exemple de ServiceMonitor pour les métriques HTTPS ressemble à ceci :

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident prend en charge les métriques HTTPS dans toutes les méthodes d'installation : tridentctl, Helm chart et Operator :

- Si vous utilisez la commande `tridentctl install`, vous pouvez passer le paramètre `--https -metrics` pour activer les métriques HTTPS.
- Si vous utilisez le Helm chart, vous pouvez définir le paramètre `httpsMetrics` pour activer les métriques HTTPS.
- Si vous utilisez des fichiers YAML, vous pouvez ajouter le `--https_metrics` flag au `trident-main` container dans le `trident-deployment.yaml` fichier.

Étape 3 : Interroger les métriques Trident avec PromQL

PromQL est idéal pour créer des expressions qui renvoient des données de séries temporelles ou tabulaires.

Voici quelques requêtes PromQL que vous pouvez utiliser :

Obtenez des informations sur la santé de Trident

- **Pourcentage de réponses HTTP 2XX provenant de Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Pourcentage de réponses REST de Trident via code d'état**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Durée moyenne en ms des opérations effectuées par Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtenez des informations sur l'utilisation de Trident

- **Taille moyenne du volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espace total de volume alloué par chaque backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtenez l'utilisation individuelle du volume

Ceci n'est activé que si les métriques kubelet sont également collectées.

- **Pourcentage d'espace utilisé pour chaque volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Découvrez la télémétrie AutoSupport de Trident

Par défaut, Trident envoie quotidiennement des métriques Prometheus et des informations de base sur le backend à NetApp.

- Pour empêcher Trident d'envoyer des métriques Prometheus et des informations de base sur le backend à NetApp, transmettez le `--silence-autosupport` indicateur lors de l'installation de Trident.
- Trident peut également envoyer les journaux de conteneurs à l'assistance NetApp à la demande via `tridentctl send autosupport`. Vous devrez déclencher Trident pour qu'il télécharge ses journaux. Avant de soumettre les journaux, vous devez accepter les "politique de confidentialité" de NetApp.
- Sauf indication contraire, Trident récupère les journaux des dernières 24 heures.
- Vous pouvez spécifier la durée de conservation des journaux avec le `--since` indicateur. Par exemple : `tridentctl send autosupport --since=1h`. Ces informations sont collectées et envoyées via un `trident-autosupport` conteneur installé avec Trident. Vous pouvez obtenir l'image du conteneur à "Trident AutoSupport".
- Trident AutoSupport ne collecte ni ne transmet d'informations personnelles identifiables (PII) ni d'informations personnelles. Il est fourni avec une "CLUF" qui ne s'applique pas à l'image de conteneur Trident elle-même. Vous pouvez en apprendre davantage sur l'engagement de NetApp en matière de sécurité des données et de confiance "ici".

Voici un exemple de charge utile envoyée par Trident :

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- Les messages AutoSupport sont envoyés au point de terminaison AutoSupport de NetApp. Si vous utilisez un registre privé pour stocker les images de conteneur, vous pouvez utiliser l'indicateur `--image-registry`.
- Vous pouvez également configurer les URL de proxy en générant les fichiers YAML d'installation. Cela

peut être fait en utilisant `tridentctl install --generate-custom-yaml` pour créer les fichiers YAML et en ajoutant l'argument `--proxy-url` pour le conteneur `trident-autosupport` dans `trident-deployment.yaml`.

Désactiver les métriques Trident

Pour **désactiver** le signalement des métriques, vous devez générer des fichiers YAML personnalisés (à l'aide de l' `--generate-custom-yaml`flag) et les modifier pour supprimer le `--metrics`flag lors de l'appel pour le `trident-main`conteneur.

Désinstaller Trident

Vous devez utiliser la même méthode pour désinstaller Trident que celle que vous avez utilisée pour l'installer.

À propos de cette tâche

- Si vous avez besoin d'un correctif pour des bugs observés après une mise à niveau, des problèmes de dépendances ou une mise à niveau incomplète ou ayant échoué, vous devez désinstaller Trident et réinstaller la version précédente en utilisant les instructions spécifiques pour cela "version". Il s'agit de la seule méthode recommandée pour *downgrader* vers une version antérieure.
- Pour faciliter la mise à niveau et la réinstallation, la désinstallation de Trident ne supprime pas les CRD ni les objets associés créés par Trident. Si vous devez supprimer complètement Trident et toutes ses données, consultez "Supprimez complètement Trident et les CRD".

Avant de commencer

Si vous mettez hors service des clusters Kubernetes, vous devez supprimer toutes les applications qui utilisent des volumes créés par Trident avant de désinstaller. Cela garantit que les PVC sont dépubliés sur les nœuds Kubernetes avant leur suppression.

Déterminez la méthode d'installation d'origine

Vous devez utiliser la même méthode pour désinstaller Trident que celle que vous avez utilisée pour l'installer. Avant de désinstaller, vérifiez quelle version vous avez utilisée pour installer Trident à l'origine.

1. Utilisez `kubectl get pods -n trident` pour examiner les pods.
   - S'il n'y a pas de module opérateur, Trident a été installé à l'aide de `tridentctl`.
   - S'il existe un pod opérateur, Trident a été installé à l'aide de l'opérateur Trident, soit manuellement, soit à l'aide de Helm.
2. S'il existe un pod opérateur, utilisez `kubectl describe tproc trident` pour déterminer si Trident a été installé à l'aide de Helm.
   - S'il y a une étiquette Helm, Trident a été installé avec Helm.
   - S'il n'y a pas d'étiquette Helm, Trident a été installé manuellement à l'aide de l'opérateur Trident.

Désinstaller une installation de l'opérateur Trident

Vous pouvez désinstaller une installation de l'opérateur Trident manuellement ou à l'aide de Helm.

Désinstaller l'installation manuelle

Si vous avez installé Trident à l'aide de l'opérateur, vous pouvez le désinstaller en procédant de l'une des manières suivantes :

1. **Modifier `TridentOrchestrator` CR et définir l'indicateur de désinstallation**:

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

   Lorsque le `uninstall` flag est défini sur `true`, l'opérateur Trident désinstalle Trident, mais ne supprime pas le TridentOrchestrator lui-même. Vous devez nettoyer le TridentOrchestrator et en créer un nouveau si vous souhaitez installer à nouveau Trident.

2. **Supprimer `TridentOrchestrator`** : En supprimant le `TridentOrchestrator` CR qui a été utilisé pour déployer Trident, vous demandez à l'opérateur de désinstaller Trident. L'opérateur traite la suppression de `TridentOrchestrator` et procède à la suppression du déploiement Trident et du daemonset, supprimant les pods Trident qu'il avait créés lors de l'installation.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Désinstaller l'installation Helm

Si vous avez installé Trident à l'aide de Helm, vous pouvez le désinstaller en utilisant `helm uninstall`.

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Désinstaller une `tridentctl` installation

Utilisez la commande `uninstall` dans `tridentctl` pour supprimer toutes les ressources associées à Trident à l'exception des CRDs et des objets associés :

```
./tridentctl uninstall -n <namespace>
```