参照
Trident

NetApp
July 01, 2026

目次

参照

Tridentポート

Tridentが通信に使用するポートの詳細については、こちらをご覧ください。

概要

Tridentは、Kubernetesクラスタ内およびストレージバックエンドとの通信にさまざまなポートを使用します。以下は、主要なポート、その目的、およびセキュリティに関する考慮事項の概要です。

- アウトバウンドフォーカス：Kubernetesノード（コントローラとワーカー）は主にストレージLIF/IPへのトラフィックを開始するため、iptablesルールではこれらのポート上のノードIPから特定のストレージIPへのアウトバウンドを許可する必要があります。広範な「any-to-any」ルールは避けてください。
- 受信制限：内部Tridentポートをクラスタ内部トラフィックに制限します（たとえば、CalicoなどのCNIを使用）。ホストファイアウォール上で不要な受信露出はありません。
- プロトコルセキュリティ：
  - 可能な場合は TCP を使用します（信頼性が高くなります）。
  - 機密の場合はiSCSIにCHAP/IPsecを有効にし、管理にはTLS/HTTPSを有効にします（ポート443/8443）。
  - NFSv4（Tridentのデフォルト）の場合、必要ない場合はUDP/古いNFSv3ポート（例：4045～4049）を削除します。
  - 信頼できるサブネットに制限し、Prometheus（オプションのポート8001）などのツールを使用して監視します。

コントローラノードのポート

これらのポートは主にTridentオペレータ（バックエンド管理）用です。すべての内部ポートはポッドレベルです。ホストファイアウォールがCNIに干渉する場合にのみノードで許可します。

ポート / プロトコル	送受信方向	目的	ドライバ/プロトコル	セキュリティに関する注意事項
TCP 8000	受信/送信（クラスタ内部）	Trident RESTサーバー（オペレーターとコントローラー間の通信）	すべて	ポッドCIDRに制限、外部への公開はありません。
TCP 8443	受信/送信（クラスタ内部）	バックチャネル HTTPS（安全な内部 API）	すべて	TLS暗号化。使用する場合はKubernetesサービスメッシュに制限されます。
TCP 8001	受信（クラスタ内部、オプション）	Prometheusメトリクス	すべて	監視ツール（RBACの使用など）にのみ公開し、使用しない場合は無効にします。
TCP 443	アウトバウンド	HTTPSからONTAP SVM/クラスタ管理LIF	ONTAP（すべて）、ANF	TLS証明書の検証が必要です。管理LIF IPのみに制限します。

ポート / プロトコル	送受信方向	目的	ドライバ/プロトコル	セキュリティに関する注意事項
TCP 8443	アウトバウンド	HTTPSからEシリーズWeb Services Proxy	Eシリーズ（iSCSI）	デフォルトREST API；証明書を使用します。バックエンドYAMLで構成可能です。

ワーカーノードのポート

これらのポートは、CSI ノードデーモンセットとポッドマウント用です。データポートはストレージデータ LIF へのアウトバウンドです。NFSv3 を使用する場合は NFSv3 エクストラを含めます（NFSv4 の場合はオプション）。

ポート / プロトコル	送受信方向	目的	ドライバ/プロトコル	セキュリティに関する注意事項
TCP 17546	受信（ポッドへのローカル）	CSI ノードの liveness / readiness プローブ	すべて	設定可能（--probe-port）；ホストの競合がないことを確認する；ローカルのみ。
TCP 8000	受信/送信（クラスタ内部）	Trident REST サーバー	すべて	上記と同様、pod-internal。
TCP 8443	受信/送信（クラスタ内部）	バックチャネル HTTPS	すべて	上記の通りです。
TCP 8001	受信（クラスタ内部、オプション）	Prometheusメトリクス	すべて	上記の通りです。
TCP 443	アウトバウンド	HTTPSからONTAP SVM/クラスタ管理LIF	ONTAP（すべて）、ANF	上記と同様、検出に使用されます。
TCP 8443	アウトバウンド	HTTPSからEシリーズWeb Services Proxy	Eシリーズ（iSCSI）	上記の通りです。
TCP/UDP 111	アウトバウンド	RPCBIND／portmapper	ONTAP-NAS（NFSv3/v4）、ANF（NFS）	v3では必須、v4（ファイアウォールオフロード）ではオプション、NFSv4のみを使用する場合は制限されます。
TCP/UDP 2049	アウトバウンド	NFSデーモン	ONTAP-NAS（NFSv3/v4）、ANF（NFS）	コアデータ、よく知られている、信頼性のためにTCPを使用。
TCP/UDP 635	アウトバウンド	マウントデーモン	ONTAP-NAS（NFSv3/v4）、ANF（NFS）	マウント。双方向コールバックが可能です（必要に応じて着信の一時コールを許可します）。
UDP 4045	アウトバウンド	NFSロックマネージャー（nlockmgr）	ONTAP-NAS（NFSv3）	ファイルロック。v4（pNFSハンドル）の場合はスキップ。UDP のみ。

ポート / プロトコル	送受信方向	目的	ドライバ/プロトコル	セキュリティに関する注意事項
UDP 4046	アウトバウンド	NFSステータスモニター（statd）	ONTAP-NAS（NFSv3）	通知。コールバックには受信一時ポート（1024～65535）が必要になる場合があります。
UDP 4049	アウトバウンド	NFSクォータデーモン（rquotad）	ONTAP-NAS（NFSv3）	クォータ。v4の場合はスキップします。
TCP 3260	アウトバウンド	iSCSIターゲット（検出/データ/CHAP）	ONTAP-SAN（iSCSI）、Eシリーズ（iSCSI）	既知のポート。このポートでCHAP 認証を実行。セキュリティのために相互 CHAP を有効にします。
TCP 445	アウトバウンド	SMB / CIFS	ONTAP-NAS（SMB）、ANF（SMB）	よく知られている、暗号化されたSMB3を使用する（Tridentアノテーション netapp.io/smb-encryption=true）。
TCP/UDP 88（オプション）	アウトバウンド	Kerberos認証	ONTAP（NFS/SMB/iSCSI と Kerb）	Kerberos を使用する場合（デフォルトではない）、ストレージではなく AD サーバーに接続します。
TCP/UDP 389（オプション）	アウトバウンド	LDAP	ONTAP（LDAP を使用した NFS/SMB）	同様に、名前解決/認証の場合、ADに制限します。

ライブネス/レディネスプローブポートは、インストール中に `--probe-port`フラグを使用して変更できます。このポートがワーカーノード上の別のプロセスによって使用されていないことを確認することが重要です。

Trident REST API

"tridentctl コマンドとオプション"は Trident REST API と対話する最も簡単な方法ですが、必要に応じて REST エンドポイントを直接使用することもできます。

REST APIを使用する場合

REST API は、Kubernetes 以外のデプロイメントで Trident をスタンドアロンバイナリとして使用する高度なインストールに便利です。

セキュリティ強化のため、Trident `REST API`はポッド内で実行する場合、デフォルトでは localhost に制限されます。この動作を変更するには、Tridentの `-address`引数をポッド構成内で設定する必要があります。

REST APIの使用

これらの API の呼び出し例については、デバッグ(`-d`フラグを渡してください。詳細については、"tridentctl を使用してTridentを管理する"を参照してください。

API は次のように動作します：

GET

`GET <trident-address>/trident/v1/<object-type>`

そのタイプのすべてのオブジェクトを一覧表示します。

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

名前付きオブジェクトの詳細を取得します。

POST

`POST <trident-address>/trident/v1/<object-type>`

指定したタイプのオブジェクトを作成します。

- オブジェクトを作成するには JSON 構成が必要です。各オブジェクトタイプの仕様については、"tridentctlを使用してTridentを管理する"を参照してください。
- オブジェクトがすでに存在する場合、動作は異なります。バックエンドは既存のオブジェクトを更新しますが、他のすべてのオブジェクトタイプは操作に失敗します。

DELETE

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

名前付きリソースを削除します。

バックエンドまたはストレージクラスに関連付けられたボリュームは引き続き存在するため、これらは個別に削除する必要があります。詳細については、"tridentctlを使用してTridentを管理する"を参照してください。

コマンドラインオプション

Tridentは、Tridentオーケストレーター用のいくつかのコマンドラインオプションを公開します。これらのオプションを使用して、デプロイメントを変更できます。

ロギング

`-debug`

デバッグ出力を有効にします。

`-loglevel <level>`

ログレベル（debug、info、warn、error、fatal）を設定します。デフォルトはinfoです。

Kubernetes

`-k8s_pod`

このオプションまたは `-k8s_api_server`を使用して、Kubernetesサポートを有効にします。これを設定すると、Tridentは、含まれているポッドのKubernetesサービスアカウントのクレデンシャルを使用してAPI

サーバに接続します。これは、Tridentがサービスアカウントが有効になっているKubernetesクラスタ内のポッドとして実行されている場合にのみ機能します。

-k8s_api_server <insecure-address:insecure-port>

このオプションまたは `-k8s_pod`を使用して、Kubernetesサポートを有効にします。指定すると、Tridentは、提供された安全でないアドレスとポートを使用してKubernetes APIサーバーに接続します。これにより、Tridentをポッドの外部に導入できますが、APIサーバーへの安全でない接続のみがサポートされます。安全に接続するには、 `-k8s_pod`オプションを使用してポッド内にTridentを導入します。

Docker

-volume_driver <name>

Dockerプラグインを登録する際に使用されるドライバ名。デフォルトは `netapp`。

-driver_port <port-number>

UNIX ドメインソケットではなく、このポートでリッスンします。

-config <file>

必須。バックエンド構成ファイルへのこのパスを指定する必要があります。

REST

-address <ip-or-host>

Trident の REST サーバーがリッスンするアドレスを指定します。デフォルトは localhost です。localhost でリッスンし、Kubernetes ポッド内で実行している場合、ポッドの外部から REST インターフェイスに直接アクセスすることはできません。 `-address ""`を使用して、ポッド IP アドレスから REST インターフェイスにアクセスできるようにします。

Trident REST インターフェイスは、127.0.0.1（IPv4 の場合）または [::1]（IPv6 の場合）でのみリッスンおよびサービスを提供するように構成できます。

-port <port-number>

Trident の REST サーバーがリッスンするポートを指定します。デフォルトは 8000 です。

-rest

REST インターフェイスを有効にします。デフォルトは true です。

KubernetesとTridentオブジェクト

Kubernetes と Trident を使用して REST API でリソース オブジェクトを読み書きすることで対話できます。Kubernetes と Trident 、 Trident とストレージ、および Kubernetes とストレージの関係を規定するリソース オブジェクトがいくつかあります。これらのオブジェクトの一部は Kubernetes を通じて管理され、その他は Trident を通じて管理されます。

オブジェクトは互いにどのように相互作用しますか？

おそらく、オブジェクト、その目的、相互作用の仕方を理解する最も簡単な方法は、Kubernetes ユーザーからの単一のストレージ要求を追跡することです：

1. ユーザーは、管理者によって事前に設定されたKubernetes `StorageClass`から、特定のサイズの新しい`PersistentVolume`を要求する `PersistentVolumeClaim`を作成します。
2. Kubernetes `StorageClass`は、Trident をプロビジョナとして識別し、要求されたクラスのボリュームを Trident がプロビジョニングする方法を指示するパラメータを含みます。
3. Tridentは、同じ名前を持つ独自の `StorageClass`を参照し、クラスのボリュームのプロビジョニングに使用できる、一致する `Backends`と `StoragePools`を識別します。
4. Tridentは、一致するバックエンドにストレージをプロビジョニングし、2つのオブジェクトを作成します。Kubernetesでボリュームの検索、マウント、および処理方法をKubernetesに指示する`PersistentVolume`と、 `PersistentVolume`と実際のストレージ間の関係を保持するTrident内のボリュームです。
5. Kubernetesは `PersistentVolumeClaim`を新しい `PersistentVolume`にバインドします。`PersistentVolumeClaim`を含むポッドは、実行されるどのホストでもそのPersistentVolumeをマウントします。
6. ユーザーは、既存のPVCの `VolumeSnapshot`を作成し、Tridentを指す `VolumeSnapshotClass`を使用します。
7. Trident は PVC に関連付けられているボリュームを識別し、そのバックエンドにボリュームのスナップショットを作成します。また、 `VolumeSnapshotContent`を作成し、 Kubernetes にスナップショットを識別する方法を指示します。
8. ユーザーは、 `PersistentVolumeClaim`を作成し、 `VolumeSnapshot`をソースとして使用できます。
9. Tridentは必要なスナップショットを識別し、 `PersistentVolume`と `Volume`の作成に関連する同じ一連の手順を実行します。

Kubernetes オブジェクトの詳細については、Kubernetes ドキュメントの "永続ボリューム"セクションをお読みになることを強くお勧めします。

Kubernetes `PersistentVolumeClaim`オブジェクト

Kubernetes `PersistentVolumeClaim` オブジェクトは、Kubernetes クラスタ ユーザによるストレージの要求です。

標準仕様に加えて、バックエンド構成で設定したデフォルトを上書きする場合、Tridentではユーザーは次のボリューム固有の注釈を指定できます：

注釈	ボリューム オプション	サポートされているドライバ
trident.netapp.io/fileSystem	fileSystem	ontap-san、solidfire-san、ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas、ontap-san、solidfire-san、azure-netapp-files、ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas、ontap-san

注釈	ボリューム オプション	サポートされているドライバ
trident.netapp.io/protocol	プロトコル	any
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas、ontap-nas-flexgroup、ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy

作成されたPVに `Delete`再利用ポリシーがある場合、TridentはPVが解放されると（つまり、ユーザーがPVCを削除すると）、PVとバッキングボリュームの両方を削除します。削除アクションが失敗した場合、TridentはPVをそのようにマークし、操作が成功するかPVが手動で削除されるまで定期的に操作を再試行します。PVが `Retain`ポリシーを使用している場合、Tridentはこれを無視し、管理者がKubernetesとバックエンドからこれをクリーンアップし、ボリュームを削除する前にバックアップまたは検査できるようにすることを想定します。PVを削除しても、Tridentはバッキングボリュームを削除しないことに注意してください。REST API(`tridentctl`を使用して削除する必要があります。

TridentはCSI仕様を使用したボリュームスナップショットの作成をサポートしています。ボリュームスナップショットを作成し、それをデータソースとして使用して既存のPVCを複製できます。このようにして、PVのポイントインタイムコピーをスナップショットの形式でKubernetesに公開できます。スナップショットを使用して新しいPVを作成できます。 `On-Demand Volume Snapshots`を参照して、これがどのように機能するかを確認してください。

Tridentは、クローンを作成するための `cloneFromPVC`および `splitOnClone`アノテーションも提供します。これらのアノテーションを使用すると、CSI実装を使用せずにPVCをクローニングできます。

例：ユーザーがすでにPVC `mysql`を持っている場合、アノテーション `trident.netapp.io/cloneFromPVC: mysql`を使用して、新しいPVC `mysqlclone`を作成できます。このアノテーションが設定されていると、Tridentはmysql PVCに対応するボリュームをクローンし、新規にボリュームをプロビジョニングするのではなく複製します。

次の点を考慮してください：

- NetApp では、アイドルボリュームのクローンを作成することを推奨しています。
- PVC とそのクローンは同じ Kubernetes ネームスペースにあり、同じストレージクラスを持つ必要があります。
- `ontap-nas`および `ontap-san`ドライバーを使用する場合、 `trident.netapp.io/splitOnClone`PVCアノテーションを `trident.netapp.io/cloneFromPVC`と組み合わせて設定することが望ましい場合があります。 `trident.netapp.io/splitOnClone`が `true`に設定されている場合、Tridentはクローンボリュームを親ボリュームから分離し、その結果、クローンボリュームのライフサイクルが親ボリュームから完全に切り離され

ますが、一部のストレージ効率が失われます。 `trident.netapp.io/splitOnClone`を設定しない、または`false`に設定すると、親ボリュームとクローンボリューム間に依存関係が生じ、クローンが先に削除されない限り親ボリュームを削除できなくなる代わりに、バックエンドでのスペース消費が削減されます。クローンの分離が有効なシナリオとしては、空のデータベースボリュームをクローンし、そのボリュームとクローンが大きく分岐し、ONTAPが提供するストレージ効率の恩恵を受けないことが予想される場合が挙げられます。

`sample-input`ディレクトリには、Tridentで使用するPVC
定義の例が含まれています。Tridentボリュームに関連するパラメータと設定の詳細については
、を参照してください。

Kubernetes `PersistentVolume`オブジェクト

Kubernetes `PersistentVolume`オブジェクトは、Kubernetesクラスタで使用できるストレージの一部を表します。これには、それを使用するポッドから独立したライフサイクルがあります。

Tridentは、プロビジョニングしたボリュームに基づいて `PersistentVolume`オブジェクトを作成し、Kubernetesクラスタに自動的に登録します。自分で管理する必要はありません。

Tridentベースの `StorageClass`を参照するPVCを作成すると、Tridentは対応するストレージクラスを使用して新しいボリュームをプロビジョニングし、そのボリュームの新しいPVを登録します。プロビジョニングされたボリュームと対応するPVを構成する際、Tridentは以下のルールに従います：

- Tridentは、Kubernetesの PV 名と、ストレージのプロビジョニングに使用する内部名を生成します。どちらの場合も、名前がその範囲内で一意であることが保証されます。
- ボリュームのサイズは、PVC で要求されたサイズに可能な限り一致しますが、プラットフォームによっては、最も近い割り当て可能な量に切り上げられる場合があります。

Kubernetes `StorageClass`オブジェクト

Kubernetes `StorageClass`オブジェクトは `PersistentVolumeClaims`で名前によって指定され、一連のプロパティを持つストレージをプロビジョニングします。ストレージクラス自体は、使用するプロビジョナーを識別し、プロビジョナーが理解できる用語でそのプロパティセットを定義します。

これは、管理者が作成および管理する必要がある 2 つの基本オブジェクトの 1 つです。もう一つは Trident バックエンドオブジェクトです。

Tridentを使用するKubernetes `StorageClass`オブジェクトは次のようになります：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

これらのパラメータはTrident固有のもので、クラスのボリュームをプロビジョニングする方法をTridentに指示します。

ストレージクラスのパラメータは次のとおりです：

属性	タイプ	必須	概要
attributes	map[string]string	いいえ	下記の属性セクションを参照してください
storagePools	map[string]StringList	いいえ	内のストレージプールのリストへのバックエンド名のマッピング
additionalStoragePools	map[string]StringList	いいえ	内のストレージプールのリストへのバックエンド名のマッピング
excludeStoragePools	map[string]StringList	いいえ	内のストレージプールのリストへのバックエンド名のマッピング

ストレージ属性とその可能な値は、ストレージプール選択属性と Kubernetes 属性に分類できます。

ストレージプールの選択属性

これらのパラメータは、特定のタイプのボリュームをプロビジョニングするためにどのTrident管理ストレージプールを使用するかを決定します。

属性	タイプ	値	オファー	要求	サポート対象
メディア[1]	string	HDD、ハイブリッド、SSD	プールにはこのタイプのメディアが含まれます。ハイブリッドとは両方を意味します	指定されたメディアタイプ	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、solidfire-san

属性	タイプ	値	オファー	要求	サポート対象
provisioningType	string	薄い、厚い	プールはこのプロビジョニング方法をサポートしています	プロビジョニング方法が指定されました	thick：すべての ONTAP、thin：すべての ONTAP および solidfire-san
backendType	string	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroup 、ontap-san 、solidfire-san 、azure-netapp-files、ontap-san-economy	プールはこのタイプのバックエンドに属します	バックエンドが指定されました	すべてのドライバー
Snapshot	ブール値	true、false	プールは Snapshot 付きのボリュームをサポートします	スナップショットが有効になっているボリューム	ontap-nas 、ontap-san 、solidfire-san
クローン	ブール値	true、false	プールはボリュームのクローン作成をサポート	クローンが有効なボリューム	ontap-nas 、ontap-san 、solidfire-san
暗号化	ブール値	true、false	プールは暗号化されたボリュームをサポートします	暗号化が有効になっているボリューム	ontap-nas 、ontap-nas-economy、ontap-nas-flexgroups、ontap-san
IOPS	int	正の整数	プールはこの範囲の IOPS を保証できる	ボリュームで保証されるIOPS	solidfire-san

[1]: ONTAP Selectシステムではサポートされていません

ほとんどの場合、要求された値はプロビジョニングに直接影響します。たとえば、シックプロビジョニングを要求すると、シックプロビジョニングされたボリュームが生成されます。ただし、Elementストレージプールは、要求された値ではなく、提供されたIOPSの最小値と最大値を使用してQoS値を設定します。この場合、要求された値はストレージプールの選択にのみ使用されます。

理想的には、 `attributes`だけを使用して、特定のクラスのニーズを満たすために必要なストレージの品質をモデル化できます。Tridentは、指定した `attributes`の_すべて_に一致するストレージプールを自動的に検出して選択します。

> `attributes`を使用してクラスに適したプールを自動的に選択できない場合は、 `storagePools`および `additionalStoragePools`パラメータを使用して、プールをさらに絞り込んだり、特定のプールのセットを選択したりできます。

`storagePools`パラメータを使用して、指定された`attributes`に一致するプールのセットをさらに制限できます。つまり、Tridentは、プロビジョニングのために `attributes`パラメータと`storagePools`パラメータによって識別されるプールの共通部分を使用します。いずれかのパラメータを単独で使用することも、両方を一緒に使用することもできます。

`additionalStoragePools`パラメータを使用して、 `attributes`および`storagePools`パラメータで選択されたプールに関係なく、Tridentがプロビジョニングに使用するプールのセットを拡張できます。

`excludeStoragePools`パラメータを使用して、Tridentがプロビジョニングに使用するプールのセットをフィルタリングできます。このパラメータを使用すると、一致するプールがすべて削除されます。

`storagePools`および `additionalStoragePools`パラメータでは、各エントリは`<backend>:<storagePoolList>`の形式をとります。ここで、`<storagePoolList>`は指定されたバックエンドのストレージプールのカンマ区切りリストです。たとえば、 `additionalStoragePools`の値は`ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`のようになります。これらのリストは、バックエンドとリスト値の両方に対して正規表現値を受け入れます。 `tridentctl get backend`を使用して、バックエンドとそのプールのリストを取得できます。

Kubernetesの属性

これらの属性は、Trident による動的プロビジョニング中のストレージプール/バックエンドの選択には影響しません。代わりに、これらの属性は Kubernetes Persistent Volume でサポートされるパラメータを提供するだけです。ワーカーノードはファイルシステムの作成操作を担当し、xfsprogs などのファイルシステムユーティリティが必要になる場合があります。

属性	タイプ	値	概要	関連するドライバー	**Kubernetes**バージョン
fsType	string	ext4、ext3、xfs	ブロックボリュームのファイルシステムタイプ	solidfire-san、ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy	すべて

属性	タイプ	値	概要	関連するドライバー	**Kubernetes**バージョン
allowVolumeExpansion	ブーリアン	true、false	PVC サイズの拡張のサポートを有効または無効にする	ontap-nas、ontap-nas-economy、ontap-nas-flexgroup、ontap-san、ontap-san-economy、solidfire-san、azure-netapp-files	1.11+
volumeBindingMode	string	即座、WaitForFirstConsumer	ボリュームバインディングと動的プロビジョニングを実行するタイミングを選択する	すべて	1.19 - 1.26

- `fsType` パラメータは、SAN LUN用の希望するファイルシステムタイプを制御するために使用されます。さらに、Kubernetesはストレージクラス内に `fsType` が存在することでファイルシステムが存在することを示します。ボリュームの所有権は、 `fsGroup` セキュリティコンテキストをポッドで使用し、 `fsType` が設定されている場合のみ制御できます。"Kubernetes：ポッドまたはコンテナのセキュリティコンテキストを設定する" を参照して、 `fsGroup` コンテキストを使用したボリューム所有権の設定概要をご覧ください。Kubernetesは、次の場合にのみ `fsGroup` の値を適用します：

  - `fsType`がストレージクラスに設定されます。
  - PVC アクセスモードは RWO です。

  NFS ストレージドライバーの場合、NFS エクスポートの一部としてファイルシステムがすでに存在します。 `fsGroup`を使用するには、ストレージクラスで `fsType`を指定する必要があります。 `nfs`または null 以外の値に設定できます。

- ボリューム拡張の詳細については、"ボリュームを拡張する"を参照してください。
- Tridentインストーラバンドルは、`sample-input/storage-class-*.yaml`でTridentと併用するためのストレージクラス定義例をいくつか提供しています。Kubernetesストレージクラスを削除すると、対応するTridentストレージクラスも削除されます。

Kubernetes `VolumeSnapshotClass`オブジェクト

Kubernetes `VolumeSnapshotClass`オブジェクトは `StorageClasses`に類似しています。これらは複数のストレージ クラスを定義するのに役立ち、ボリューム スナップショットによって参照されて、スナップショットを必要なスナップショット クラスに関連付けます。各ボリューム スナップショットは、単一のボリューム スナップショット クラスに関連付けられます。

`VolumeSnapshotClass`は、`Snapshot`を作成するために管理者が定義する必要があります。ボリューム`Snapshot`クラスは、次の定義で作成されます：

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

`driver`は、 `csi-snapclass`クラスのボリュームSnapshotのリクエストが Tridentによって処理されることをKubernetesに指定します。 `deletionPolicy` は、Snapshotを削除する必要があるときに実行するアクションを指定します。 `deletionPolicy`が `Delete`に設定されている場合、Snapshot が削除されると、ボリュームSnapshotオブジェクトとストレージクラスター上の基礎となるSnapshotが削除されます。または、 `Retain`に設定すると、 `VolumeSnapshotContent` と物理Snapshotが保持されます。

Kubernetes `VolumeSnapshot`オブジェクト

Kubernetes `VolumeSnapshot`オブジェクトは、ボリュームのSnapshotを作成する要求です。PVCがボリュームに対するユーザーからの要求を表すのと同様に、ボリュームSnapshotは既存のPVCのSnapshotを作成するためのユーザーからの要求です。

ボリュームスナップショットのリクエストが届くと、Tridentはバックエンドのボリュームのスナップショットの作成を自動的に管理し、一意の `VolumeSnapshotContent`オブジェクトを作成してスナップショットを公開します。既存のPVCからスナップショットを作成し、新しいPVCを作成するときにそのスナップショットをDataSourceとして使用できます。

VolumeSnapshotのライフサイクルは、ソースPVCから独立しています。ソースPVCが削除された後もスナップショットは保持されます。スナップショットが関連付けられているPVCを削除する場合、TridentはこのPVCのバッキングボリュームを*削除中*状態にマークしますが、完全には削除しません。関連付けられているすべてのスナップショットが削除されると、ボリュームは削除されます。

Kubernetes `VolumeSnapshotContent`オブジェクト

Kubernetes `VolumeSnapshotContent`オブジェクトは、すでにプロビジョニングされたボリュームから取得されたスナップショットを表します。これは `PersistentVolume`に類似しており、ストレージ クラスター上にプロビジョニングされたSnapshotを示します。 `PersistentVolumeClaim`および `PersistentVolume`オブジェクトと同様に、Snapshotが作成されると、 `VolumeSnapshotContent`オブジェクトは、Snapshotの作成を要求した `VolumeSnapshot`オブジェクトとの1対1のマッピングを維持します。

`VolumeSnapshotContent`オブジェクトには、 `snapshotHandle`などの Snapshotを一意に識別する詳細が含まれています。 `snapshotHandle`は、PVの名前と `VolumeSnapshotContent`オブジェクトの名前を組み合わせた一意の値です。

スナップショットのリクエストが来ると、Tridentはバックエンドにスナップショットを作成します。スナップ

ショットが作成されると、Tridentは `VolumeSnapshotContent`オブジェクトを構成し、スナップショットをKubernetes APIに公開します。

通常、 `VolumeSnapshotContent`オブジェクトを管理する必要はありません。例外は、Tridentの外部で作成された"ボリューム Snapshot をインポートする"を使用する場合です。

Kubernetes `VolumeGroupSnapshotClass`オブジェクト

Kubernetes `VolumeGroupSnapshotClass`オブジェクトは `VolumeSnapshotClass`に類似しています。これらは複数のストレージ クラスを定義するのに役立ち、ボリューム グループ Snapshotによって参照されて、Snapshotを必要なSnapshotクラスに関連付けます。各ボリューム グループ Snapshotは、単一のボリューム グループ Snapshotクラスに関連付けられます。

`VolumeGroupSnapshotClass`は、スナップショットのグループを作成するために管理者が定義する必要があります。ボリューム グループ スナップショット
クラスは、次の定義で作成されます：

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

`driver`は、 `csi-group-snap-class`クラスのボリュームグループSnapshot
の要求がTridentによって処理されることをKubernetesに指定します。
`deletionPolicy`は、グループSnapshotを削除する必要があるときに実行するアクションを指定します。 `deletionPolicy`が `Delete`に設定されている場合、
Snapshotが削除されると、ボリュームグループSnapshotオブジェクトとストレージクラスタ上の基盤となるSnapshotが削除されます。または、 `Retain`に設定すると、
`VolumeGroupSnapshotContent`と物理Snapshotが保持されます。

Kubernetes `VolumeGroupSnapshot`オブジェクト

Kubernetes `VolumeGroupSnapshot`オブジェクトは、複数のボリュームのスナップショットを作成する要求です。PVCがボリュームに対するユーザーの要求を表すのと同様に、ボリューム グループ スナップショットは、既存のPVCのスナップショットを作成するためのユーザーの要求です。

ボリュームグループのスナップショット要求が来ると、Tridentはバックエンドのボリュームのグループスナップショットの作成を自動的に管理し、一意の `VolumeGroupSnapshotContent`オブジェクトを作成してスナップショットを公開します。既存のPVCからスナップショットを作成し、新しいPVCを作成するときにそのスナップショットをDataSourceとして使用できます。

VolumeGroupSnapshotのライフサイクルは、ソースPVCから独立しています。ソースPVCが削除された後もスナップショットは保持されます。スナップショットが関連付けられているPVCを削除する場合、TridentはこのPVCのバッキングボリュームを*削除中*状態にマークしますが、完全には削除しません。関連するすべてのスナップショットが削除されると、ボリュームグループのスナップショットも削除されます。

Kubernetes `VolumeGroupSnapshotContent`オブジェクト

Kubernetes `VolumeGroupSnapshotContent`オブジェクトは、すでにプロビジョニングされたボリュームから取得されたグループSnapshotを表します。これは `PersistentVolume`に類似しており、ストレージ クラスター上にプロビジョニングされたSnapshotを示します。 `PersistentVolumeClaim`および `PersistentVolume`オブジェクトと同様に、Snapshotが作成されると、 `VolumeSnapshotContent`オブジェクトは、Snapshotの作成を要求した `VolumeSnapshot`オブジェクトとの1対1のマッピングを維持します。

`VolumeGroupSnapshotContent`オブジェクトには、`volumeGroupSnapshotHandle`やストレージシステムに存在する個々の`volumeSnapshotHandles`など、スナップショットグループを識別する詳細が含まれます。

スナップショットのリクエストが来ると、Tridentはバックエンドにボリューム グループのスナップショットを作成します。ボリューム グループのスナップショットが作成されると、Tridentは `VolumeGroupSnapshotContent`オブジェクトを構成し、スナップショットをKubernetes APIに公開します。

Kubernetes `CustomResourceDefinition`オブジェクト

Kubernetes カスタム リソースは、管理者によって定義され、類似のオブジェクトをグループ化するために使用される Kubernetes API のエンドポイントです。Kubernetes は、オブジェクトのコレクションを保存するためのカスタム リソースの作成をサポートしています。これらのリソース定義は、以下を実行することで取得できます `kubectl get crds`。

カスタム リソース定義（CRD）とそれに関連付けられたオブジェクト メタデータは、Kubernetesによってメタデータ ストアに保存されます。これにより、Trident用の別のストアが不要になります。

Tridentは、 `CustomResourceDefinition`オブジェクトを使用して、Tridentバックエンド、Tridentストレージクラス、Tridentボリュームなどの Tridentオブジェクトのアイデンティティを保持します。これらのオブジェクトはTridentによって管理されます。さらに、CSIボリュームSnapshotフレームワークでは、ボリュームSnapshotを定義するために必要ないくつかのCRDが導入されています。

CRDはKubernetesの構造です。上記で定義されたリソースのオブジェクトはTridentによって作成されます。簡単な例として、 `tridentctl`を使用してバックエンドを作成すると、対応する `tridentbackends` CRDオブジェクトがKubernetesで使用するために作成されます。

以下に、TridentのCRDについて留意すべき点をいくつか示します：

- Tridentがインストールされると、CRDのセットが作成され、他のリソース タイプと同様に使用できるようになります。
- `tridentctl uninstall`コマンドを使用してTridentをアンインストールすると、Tridentポッドは削除されますが、作成されたCRDはクリーンアップされません。"Tridentのアンインストール"を参照して、Tridentを完全に削除して最初から再構成する方法を理解してください。

Trident `StorageClass`オブジェクト

Tridentは、プロビジョナー フィールドで `csi.trident.netapp.io`を指定するKubernetes `StorageClass`オブジェクト用に対応するストレージクラスを作成します。ストレージクラス名は、それが表すKubernetes `StorageClass`オブジェクトの名前と一致します。

> Kubernetesでは、Tridentをプロビジョナーとして使用するKubernetes `StorageClass`が登録されると、これらのオブジェクトが自動的に作成されます。

ストレージ クラスは、ボリュームの要件のセットから構成されます。Tridentはこれらの要件を各ストレージ プールに存在する属性と照合します。一致する場合、そのストレージ プールはそのストレージ クラスを使用してボリュームをプロビジョニングするための有効なターゲットになります。

REST APIを使用してストレージクラスを直接定義するストレージクラス構成を作成できます。ただし、Kubernetesのデプロイメントの場合、新しいKubernetes `StorageClass`オブジェクトを登録するときに作成されることを想定しています。

Tridentバックエンドオブジェクト

バックエンドは、Tridentがボリュームをプロビジョニングするストレージプロバイダーを表します。単一のTridentインスタンスは任意の数のバックエンドを管理できます。

> これは、自分で作成して管理する 2 つのオブジェクト タイプのうちの 1 つです。もう 1 つは Kubernetes `StorageClass` オブジェクトです。

これらのオブジェクトの構築方法の詳細については、"バックエンドの設定"を参照してください。

Trident `StoragePool`オブジェクト

ストレージ プールは、各バックエンドでプロビジョニングできる個別の場所を表します。ONTAP の場合、これらは SVM のアグリゲートに相当します。NetApp HCI/SolidFire の場合、これらは管理者が指定した QoS 帯域に対応します。各ストレージ プールには、パフォーマンス特性とデータ保護特性を定義する一連の個別のストレージ属性があります。

ここでの他のオブジェクトとは異なり、ストレージプールの候補は常に自動的に検出され、管理されます。

Trident `Volume`オブジェクト

ボリュームはプロビジョニングの基本単位であり、NFS 共有、iSCSI、FC LUN などのバックエンド エンドポイントで構成されます。Kubernetes では、これらは `PersistentVolumes`に直接対応します。ボリュームを作成するときは、ボリュームをプロビジョニングできる場所とサイズを決定するストレージ クラスがあることを確認します。

> - Kubernetes では、これらのオブジェクトは自動的に管理されます。それらを表示して、Trident がプロビジョニングした内容を確認できます。
> - 関連するスナップショットを持つPVを削除する場合、対応するTridentボリュームは*削除中*状態に更新されます。Tridentボリュームを削除するには、ボリュームのスナップショットを削除する必要があります。

ボリューム構成は、プロビジョニングされたボリュームに必要なプロパティを定義します。

属性	タイプ	必須	概要
version	string	いいえ	Trident APIのバージョン（「1」）
名前	string	はい	作成するボリュームの名前
storageClass	string	はい	ボリュームのプロビジョニング時に使用するストレージクラス
サイズ	string	はい	プロビジョニングするボリュームのサイズ（バイト単位）
プロトコル	string	いいえ	使用するプロトコルタイプ：「file」または「block」
internalName	string	いいえ	ストレージシステム上のオブジェクトの名前。Tridentによって生成されます
cloneSourceVolume	string	いいえ	ontap (nas、san) & solidfire-*：クローン元のボリュームの名前
splitOnClone	string	いいえ	ONTAP (NAS、SAN): クローンを親から分離する
snapshotPolicy	string	いいえ	ONTAP-*：使用するSnapshotポリシー
snapshotReserve	string	いいえ	ontap-*：Snapshot用に予約されているボリュームの割合
exportPolicy	string	いいえ	ontap-nas*：使用するエクスポートポリシー
snapshotDirectory	ブール値	いいえ	ontap-nas*：Snapshotディレクトリが表示されるかどうか
unixPermissions	string	いいえ	ontap-nas*：初期UNIX権限
blockSize	string	いいえ	SolidFire-*：ブロック/セクターサイズ
fileSystem	string	いいえ	ファイルシステムの種類
skipRecoveryQueue	string	いいえ	ボリュームの削除中は、ストレージ内のリカバリキューをバイパスし、ボリュームを直ちに削除します。

Tridentは、ボリュームを作成するときに `internalName`を生成します。これは2つのステップで構成されま

す。まず、ストレージプレフィックス（デフォルトの `trident`またはバックエンド設定のプレフィックス）をボリューム名の前に付加して、 `<prefix>-<volume-name>`という形式の名前にします。次に、バックエンドで許可されていない文字を置き換えて、名前をサニタイズします。ONTAPバックエンドの場合、ハイフンをアンダースコアに置き換えます（したがって、内部名は `<prefix>_<volume-name>`になります）。Elementバックエンドの場合、アンダースコアはハイフンに置き換えられます。

ボリューム構成を使用すると、REST APIを使用してボリュームを直接プロビジョニングできますが、Kubernetesの導入では、ほとんどのユーザーが標準のKubernetes `PersistentVolumeClaim`方法を使用することが想定されています。Tridentは、プロビジョニング プロセスの一部として、このボリューム オブジェクトを自動的に作成します。

Trident `Snapshot`オブジェクト

スナップショットはボリュームのポイントインタイム コピーであり、新しいボリュームのプロビジョニングや状態の復元に使用できます。Kubernetesでは、これらは `VolumeSnapshotContent`オブジェクトに直接対応します。各スナップショットは、スナップショットのデータのソースであるボリュームに関連付けられています。

各 `Snapshot`オブジェクトには、以下のプロパティが含まれます：

属性	タイプ	必須	概要
version	文字列	はい	Trident APIのバージョン（「1」）
名前	文字列	はい	Tridentスナップショットオブジェクトの名前
internalName	文字列	はい	ストレージシステム上のTridentスナップショットオブジェクトの名前
volumeName	文字列	はい	Snapshotが作成される永続ボリュームの名前
volumeInternalName	文字列	はい	ストレージシステム上の関連するTridentボリュームオブジェクトの名前

Kubernetes では、これらのオブジェクトは自動的に管理されます。それらを表示して、Tridentがプロビジョニングした内容を確認できます。

Kubernetes `VolumeSnapshot`オブジェクトリクエストが作成されると、`Trident`はバックエンドストレージシステム上にスナップショットオブジェクトを作成することで動作します。このスナップショットオブジェクトの `internalName`は、プレフィックス `snapshot-`と `UID`、`VolumeSnapshot`オブジェクトの`snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`を組み合わせて生成されます。 `volumeName`および`volumeInternalName`は、バックエンドボリュームの詳細を取得することで設定されます。

Trident `ResourceQuota`オブジェクト

Tridentデーモンセットは `system-node-critical`優先度クラス（Kubernetesで利用可能な最高の優先度クラス）を使用します。これにより、Tridentはノードの正常なシャットダウン中にボリュームを識別してクリーンアップでき、リソースの負荷が高いクラスターでは、Tridentデーモンセットポッドが優先度の低いワークロードをプリエンプトできます。

これを実現するために、Tridentは `ResourceQuota`オブジェクトを使用して、Trident daemonsetで「system-node-critical」優先度クラスが満たされるようにします。導入とdaemonsetの作成前に、Tridentは `ResourceQuota`オブジェクトを検索し、検出されない場合は適用します。

デフォルトのResource QuotaとPriority Classをさらに制御する必要がある場合は、 `custom.yaml`を生成するか、Helmチャートを使用して `ResourceQuota`オブジェクトを設定できます。

以下は、Tridentデーモンセットを優先する`ResourceQuota`オブジェクトの例です。

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

リソースクォータの詳細については、"Kubernetes：リソースクォータ"を参照してください。

インストールに失敗した場合はクリーンアップします `ResourceQuota`

`ResourceQuota`オブジェクトの作成後にインストールが失敗するまれなケースでは、まずlink:../trident-managing-k8s/uninstall-trident.html["アンインストール"]を試してから再インストールしてください。

それでも解決しない場合は、手動で `ResourceQuota`オブジェクトを削除してください。

削除 `ResourceQuota`

自分でリソースの割り当てを制御したい場合は、次のコマンドを使用してTrident `ResourceQuota`オブジェクトを削除できます：

```
kubectl delete quota trident-csi -n trident
```

Pod Security Standards（PSS）とSecurity Context Constraints（SCC）

Kubernetes ポッド セキュリティ標準（PSS）とポッド セキュリティ ポリシー（PSP）

は、権限レベルを定義し、ポッドの動作を制限します。OpenShift セキュリティ コンテキスト制約（SCC）も同様に、OpenShift Kubernetes エンジンに固有のポッド制限を定義します。このカスタマイズを提供するために、Trident はインストール中に特定の権限を有効にします。以下のセクションでは、Trident によって設定される権限について詳しく説明します。

PSSはPod Security Policies（PSP）に代わるものです。PSPはKubernetes v1.21で非推奨となり、v1.25で削除されます。詳細については、"Kubernetes：セキュリティ"を参照してください。

必要な **Kubernetes** セキュリティコンテキストと関連フィールド

権限	概要
特権	CSI ではマウントポイントが双方向であることが求められており、Trident ノード ポッドは特権コンテナを実行する必要があります。詳細については、"Kubernetes：マウント伝播"を参照してください。
ホストネットワーク	iSCSI デーモンに必要です。 `iscsiadm`は iSCSI マウントを管理し、ホストネットワークを使用して iSCSI デーモンと通信します。
ホスト IPC	NFS は、プロセス間通信（IPC）を使用して NFSD と通信します。
ホスト PID	NFS の場合は `rpc-statd`を開始する必要があります。Trident は、NFS ボリュームをマウントする前に `rpc-statd`が実行されているかどうかを判断するためにホストプロセスを照会します。

機能	`SYS_ADMIN`機能は、特権コンテナのデフォルト機能の一部として提供されます。たとえば、`Docker` は特権コンテナに対して次の機能を設定します：
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	Seccompプロファイルは特権コンテナでは常に「Unconfined」であるため、Tridentでは有効にできません。
SELinux	OpenShift では、特権コンテナは `spc_t`（「Super Privileged Container」）ドメインで実行され、非特権コンテナは `container_t`ドメインで実行されます。 `containerd`では、 `container-selinux`がインストールされている場合、すべてのコンテナは `spc_t`ドメインで実行され、SELinux が事実上無効になります。したがって、Trident はコンテナに `seLinuxOptions`を追加しません。
DAC	特権コンテナは root として実行する必要があります。非特権コンテナは、CSI に必要な Unix ソケットにアクセスするために root として実行されます。

Pod Security Standards（PSS）

ラベル	概要	デフォルト
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Trident コントローラとノードがインストール名前空間に許可されるようにします。名前空間ラベルを変更しないでください。	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

名前空間ラベルを変更すると、ポッドがスケジュールされず、「作成エラー：...」または「警告：trident-csi-...」が表示される場合があります。このような場合は、 `privileged`の名前空間ラベルが変更されたかどうかを確認してください。変更されている場合は、Tridentを再インストールしてください。

Pod Security Policies（PSP）

フィールド	概要	デフォルト
`allowPrivilegeEscalation`	特権コンテナは特権の昇格を許可する必要があります。	`true`
`allowedCSIDrivers`	Tridentはインライン CSI 一時ボリュームを使用しません。	空
`allowedCapabilities`	非特権Tridentコンテナはデフォルトセット以上の機能を必要とせず、特権コンテナには可能なすべての機能が付与されます。	空
`allowedFlexVolumes`	Tridentは"FlexVolumeドライバ"を使用しないため、許可されるボリュームのリストには含まれません。	空
`allowedHostPaths`	Tridentノード ポッドはノードのルート ファイルシステムをマウントするため、このリストを設定する利点はありません。	空
`allowedProcMountTypes`	Tridentは使用しません `ProcMountTypes`	空
`allowedUnsafeSysctls`	Tridentでは、安全でないものは必要ありません `sysctls`。	空
`defaultAddCapabilities`	特権コンテナに機能を追加する必要はありません。	空
`defaultAllowPrivilegeEscalation`	権限昇格の許可はそれぞれのTridentポッドで処理されます。	`false`
`forbiddenSysctls`	`sysctls`は許可されていません。	空
`fsGroup`	Trident コンテナは root として実行されます。	`RunAsAny`

フィールド	概要	デフォルト
hostIPC	NFSボリュームをマウントするには、ホストIPCとの通信が必要です nfsd	true
hostNetwork	iscsiadm では、iSCSI デーモンと通信するためにホストネットワークが必要です。	true
hostPID	ホストPIDは、 `rpc-statd`がノード上で実行されているかどうかを確認するために必要です。	true
hostPorts	Trident はホストポートを使用しません。	空
privileged	ボリュームをマウントするには、Tridentノードポッドで特権コンテナを実行する必要があります。	true
readOnlyRootFilesystem	Trident ノード ポッドはノード ファイル システムに書き込む必要があります。	false
requiredDropCapabilities	Tridentノードポッドは特権コンテナを実行するため、機能を削除することはできません。	none
runAsGroup	Trident コンテナは root として実行されます。	RunAsAny
runAsUser	Trident コンテナは root として実行されます。	runAsAny
runtimeClass	Tridentは `RuntimeClasses`を使用しません。	空
seLinux	Tridentは `seLinuxOptions`を設定しません。これは、コンテナランタイムとKubernetesディストリビューションがSELinuxを処理する方法に現在違いがあるためです。	空
supplementalGroups	Trident コンテナは root として実行されます。	RunAsAny
volumes	Tridentポッドにはこれらのボリュームプラグインが必要です。	hostPath, projected, emptyDir

Security Context Constraints（SCC）

ラベル	概要	デフォルト
allowHostDirVolumePlugin	Trident ノードポッドは、ノードのルートファイルシステムをマウントします。	true

ラベル	概要	デフォルト
allowHostIPC	NFSボリュームをマウントするには、ホストIPCと通信する必要があります nfsd。	true
allowHostNetwork	iscsiadm では、iSCSI デーモンと通信するためにホストネットワークが必要です。	true
allowHostPID	ホストPIDは、 `rpc-statd`がノード上で実行されているかどうかを確認するために必要です。	true
allowHostPorts	Trident はホストポートを使用しません。	false
allowPrivilegeEscalation	特権コンテナは特権の昇格を許可する必要があります。	true
allowPrivilegedContainer	ボリュームをマウントするには、Tridentノードポッドで特権コンテナを実行する必要があります。	true
allowedUnsafeSysctls	Tridentでは、安全でないものは必要ありません sysctls。	none
allowedCapabilities	非特権Tridentコンテナはデフォルトセット以上の機能を必要とせず、特権コンテナには可能なすべての機能が付与されます。	空
defaultAddCapabilities	特権コンテナに機能を追加する必要はありません。	空
fsGroup	Trident コンテナは root として実行されます。	RunAsAny
groups	このSCCはTrident専用であり、そのユーザーにバインドされています。	空
readOnlyRootFilesystem	Trident ノード ポッドはノード ファイル システムに書き込む必要があります。	false
requiredDropCapabilities	Tridentノードポッドは特権コンテナを実行するため、機能を削除することはできません。	none
runAsUser	Trident コンテナは root として実行されます。	RunAsAny
seLinuxContext	Tridentは `seLinuxOptions`を設定しません。これは、コンテナランタイムとKubernetesディストリビューションがSELinuxを処理する方法に現在違いがあるためです。	空

ラベル	概要	デフォルト
`seccompProfiles`	特権コンテナは常に"Unconfined"で実行されます。	空
`supplementalGroups`	Trident コンテナは root として実行されます。	`RunAsAny`
`users`	このSCCをTrident名前空間のTridentユーザーにバインドするためのエントリが1つ提供されています。	N/A
`volumes`	Tridentポッドにはこれらのボリュームプラグインが必要です。	`hostPath, downwardAPI, projected, emptyDir`

著作権に関する情報

商標に関する情報