参照
Trident

NetApp
February 20, 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、こちらを参照してください。

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	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。

活性/レディネスプローブポートは、を使用して設置するときに変更できます `--probe-port` フラグ。このポートがワーカーノード上の別のプロセスで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"Trident REST APIを操作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を使用する状況

REST APIは、Kubernetes以外の環境で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役立ちます。

セキュリティを強化するため、ポッド内で実行する場合、Tridentは `REST API`デフォルトでlocalhostに制限されています。この動作を変更するには、ポッド構成でTridentの引数を設定する必要があり `-address`ます。

REST APIを使用する

これらのAPIの呼び出し方法の例については、debug)(`-d`フラグを渡します。詳細については、を参照してください "tridentctlを使用したTridentの管理"。

API は次のように機能します。

取得

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>

指定したオブジェクトの詳細を取得します。

投稿（ **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、を参照してください"tridentctlを使用したTridentの管理"。
- オブジェクトがすでに存在する場合、動作は一定ではありません。バックエンドが既存のオブジェクトを更新しますが、それ以外のすべてのオブジェクトタイプで処理が失敗します。

削除

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

指定した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ているボリュームは削除されず、削除されません。詳細については、を参照してください "tridentctlを使用したTridentの管理"。

コマンドラインオプション

Tridentでは、Tridentオーケストレーションツールのコマンドラインオプションがいくつか公開されています。これらのオプションを使用して、導入環境を変更できます。

ロギング

-debug

デバッグ出力を有効にします。

-loglevel <level>

ロギングレベル（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` をクリックしてKubernetesのサポートを有効にしこれを設定すると、 Trident はポッドの 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、サービスアカウントが有効になっている Kubernetes クラスタで Trident がポッドとして

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を使用し `-k8s_pod`て、Kubernetesのサポートを有効にします。Trident を指定すると、セキュアでないアドレスとポートを使用して Kubernetes API サーバに接続されます。これにより、Tridentをポッドの外部に導入できますが、サポートされるのはAPIサーバへの安全でない接続のみです。安全に接続するには、オプションを使用してポッドにTridentを導入し `-k8s_pod`ます。

Docker です

-volume_driver <name>

Dockerプラグインの登録時に使用するドライバ名。デフォルトはです `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サーバがリスンするアドレスを指定します。デフォルトは localhost です。localhost で聞いて Kubernetes ポッド内で実行しているときに、 REST インターフェイスにポッド外から直接アクセスすることはできません。使用 `-address ""` RESTインターフェイスにポッドのIPアドレスからアクセスできるようにするため。

> 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）のみをリスンして処理するように設定できます。

-port <port-number>

TridentのRESTサーバがリ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 オブジェクトと Trident オブジェクト

リソースオブジェクトの読み取りと書き込みを行うことで、 REST API を使用して Kubernetes や Trident を操作できます。Kubernetes と Trident 、 Trident とストレージ、 Kubernetes とストレージの関係を決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中には Kubernetes で管理されるものと Trident で管理されるものがあります。

オブジェクトは相互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、操作方法を理解する最も簡単な方法は、 Kubernetes ユーザからのスト

レージ要求を 1 回だけ処理することです。

1. ユーザーは、「 PersistentVolumeClaim 」を作成して、特定のサイズの新しい「 PersistentVolume 」を、管理者が以前に設定した Kubernetes の「 torageClass 」から要求します。
2. Kubernetes の「 torageClass 」は、 Trident をプロビジョニングツールとして識別し、要求されたクラスのボリュームのプロビジョニング方法を Trident に指示するパラメータを含んでいます。
3. Trident は、対応する「 Backends 」と「 toragePools 」を識別する同じ名前の「 S torageClass 」を参照します。この名前は、このクラスのボリュームのプロビジョニングに使用できます。
4. Trident は、対応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 では、「 PersistentVolume 」とは、ボリュームを検索、マウント、処理する方法を Kubernetes に伝える「 PersistentVolume 」と、「 PersistentVolume 」と実際のストレージの関係を保持する 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PersistentVolume が実行される任意のホストに PersistentVolume をマウントする 'PersistentVolumeClaim を含むポッド。
6. ユーザーは、 Trident を指す「 VolumeSnapshotClass 」を使用して、既存の PVC の「 VolumeSnapshot 」を作成します。
7. Trident が PVC に関連付けられているボリュームを特定し、バックエンドにボリュームの Snapshot を作成します。また ' スナップショットの識別方法を Kubernetes に指示する 'VolumeSnapshotContent' も作成します
8. ユーザーは 'VolumeSnapshot' をソースとして使用して 'PersistentVolumeClaim を作成できます
9. Trident は必要なスナップショットを識別し、「 PersistentVolume 」と「 Volume 」の作成に関連する一連のステップを実行します。

Kubernetes オブジェクトの詳細については、を参照することを強く推奨します "永続ボリューム" Kubernetes のドキュメントのセクション。

Kubernetes `PersistentVolumeClaim` オブジェクト

Kubernetes の「 PersistentVolumeClaim 」オブジェクトは、 Kubernetes クラスタユーザが作成したストレージの要求です。

Trident では、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設定を上書きする場合に、ボリューム固有の次のアノテーションを指定できます。

アノテーション	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	ファイルシステム	ONTAP-SAN、solidfire-san-エコノミー 構成、solidfire-san-SAN間にあるSolidFireを実現します
trident.netapp.io/cloneFromPVC	cloneSourceVolume の実行中です	ontap-nas、ontap-san、solidfire-san、azure-netapp-files、ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ONTAP - NAS 、 ONTAP - SAN
trident.netapp.io/protocol	プロトコル	任意
trident.netapp.io/exportPolicy	エクスポートポリシー	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup

アノテーション	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/snapshotPolicy	Snapshot ポリシー	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup 、 ONTAP-SAN
trident.netapp.io/snapshotReserve	Snapshot リザーブ	ontap-nas、ontap-nas-flexgroup 、ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory の略	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup
trident.netapp.io/blockSize	ブロックサイズ	solidfire - SAN
trident.netapp.io/skipリカバリキュー	リカバリキューをスキップ	ontap-nas、ontap-nas-economy 、ontap-nas-flexgroup、ontap-san、ontap-san-economy

作成された PV に「削除」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 を削除したときに）、 PV と元のボリュームの両方を削除します。削除操作が失敗した場合、 Trident は PV をマークします。そのような状態で操作が成功するか、 PV が手動で削除されるまで、定期的に再試行します。PV が「 + Retain + 」ポリシーを使用している場合、 Trident はそのポリシーを無視し、管理者が Kubernetes とバックエンドからクリーンアップすると想定します。これにより、ボリュームを削除する前にバックアップまたは検査を行うことができます。PV を削除しても、原因 Trident で元のボリュームが削除されないことに注意してください。REST API (tridentctl') を使用して削除してください。

Trident では CSI 仕様を使用したボリュームスナップショットの作成がサポートされています。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 のクローンを作成できます。これにより、 PVS のポイントインタイムコピーを Kubernetes にスナップショットの形で公開できます。作成した Snapshot を使用して新しい PVS を作成できます。「 +On-Demand Volume Snapshots + 」を見て、これがどのように機能するかを確認してください。

Tridentが提供するのも `cloneFromPVC` および `splitOnClone` クローンを作成するためのアノテーションこれらの注釈を使用して、CSI実装を使用せずにPVCのクローンを作成できます。

例 : ユーザがすでに「 m ysql 」という PVC を持っている場合、ユーザは「 trident.netapp.io/cloneFromPVC: mysql 」などの注釈を使用して「 m ysqlclone 」という新しい PVC を作成できます。このアノテーションセットを使用すると、 Trident はボリュームをゼロからプロビジョニングするのではなく、 MySQL PVC に対応するボリュームのクローンを作成します。

次の点を考慮してください。

- NetAppでは、アイドル状態のボリュームをクローニングすることを推奨
- PVC とそのクローンは、同じ Kubernetes ネームスペースに存在し、同じストレージクラスを持つ必要があります。
- また 'ONTAP-NAS' および 'ONTAP-SAN' ドライバを使用すると 'pvc 注釈 trident.netapp.io/splitOnClone` 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 は 'trident.netapp.io/splitOnClone` を true に設定した場合 ' クローン・ボリュームを親ボリュームからスプリットするため ' 一部のストレージ効率を失うことなく ' クローン・ボリュームのライフサイクルを親ボリュームから完全に分離しますtrident.netapp.io/splitOnClone` を設定したり 'false に設定したりしないと ' 親ボリュームとクローンボリューム間の依存関係を作成する代わりに ' バックエンドでのスペース消費が削減されますこれにより ' クローンを最初に削除しない限り ' 親ボリュームを削除できなくなりますクロー

ンをスプリットするシナリオでは、空のデータベースボリュームをクローニングする方法が効果的です。このシナリオでは、ボリュームとそのクローンで使用するデータベースボリュームのサイズが大きく異なっており、 ONTAP ではストレージ効率化のメリットはありません。

。 `sample-input` Directoryには、Tridentで使用するPVC定義の例が含まれています。を参照してください をクリックして、Tridentボリュームに関連付けられているパラメータと設定の完全な概要を確認します。

Kubernetes `PersistentVolume` オブジェクト

Kubernetes の 'PersistentVolume' オブジェクトは 'Kubernetes クラスタで利用できるようになったストレージの一部ですポッドに依存しないライフサイクルがあります。

Trident は 'PersistentVolume' オブジェクトを作成し ' プロビジョニングするボリュームに基づいて自動的に Kubernetes クラスタに登録します自分で管理することは想定されていません。

Trident をベースとする「 torageClass 」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、そのボリュームに新しい 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 の構成では、 Trident は次のルールに従います。

- Trident は、 Kubernetes に PV 名を生成し、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、名前がスコープ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できるだけ近いサイズに一致しますが、プラットフォームによっては、最も近い割り当て可能な数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 の「 torageClass 」オブジェクトは、「 PersistentVolumeClaims 」内の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が、使用するプロビジョニングツールを特定し、プロビジョニングツールが理解できる一連のプロパティを定義します。

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うちの 1 つです。もう 1 つは Trident バックエンドオブジェクトです。

Trident を使用する Kubernetes の「 torageClass 」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 固有で、クラスのボリュームのプロビジョニング方法を 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	を入力します	必須	説明
属性（ Attributes ）	[string] 文字列をマップします	いいえ	後述の「属性」セクションを参照してください
ストレージプール	[string] StringList をマップします	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	[string] StringList をマップします	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	[string] StringList をマップします	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有効な値は、ストレージプールの選択属性と Kubernetes 属性に分類できます。

ストレージプールの選択の属性

これらのパラメータは、特定のタイプのボリュームのプロビジョニングに使用する Trident で管理されているストレージプールを決定します。

属性	を入力します	値	提供	リクエスト	でサポートされます
メディア ^1	文字列	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ています。ハイブリッドは両方を意味します	メディアタイプが指定されました	ONTAPNAS 、 ONTAPNAS エコノミー、 ONTAP-NAS-flexgroup 、 ONTAPSAN 、 solidfire-san-SAN 、 solidfire-san-SAN のいずれかに対応しています
プロビジョニングタイプ	文字列	シン、シック	プールはこのプロビジョニング方法をサポートします	プロビジョニング方法が指定されました	シック：All ONTAP ；thin ：All ONTAP & solidfire-san-SAN

属性	を入力します	値	提供	リクエスト	でサポートされます
backendType	文字列	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ています	バックエンドが指定されて	すべてのドライバ
Snapshot	ブール値	true false	プールは、Snapshot を含むボリュームをサポートします	Snapshot が有効なボリューム	ontap-nas 、ontapさん、solidfireさん
クローン	ブール値	true false	プールはボリュームのクローニングをサポートします	クローンが有効なボリューム	ontap-nas 、ontapさん、solidfireさん
暗号化	ブール値	true false	プールでは暗号化されたボリュームをサポート	暗号化が有効なボリューム	ONTAP-NAS 、ONTAP-NAS-エコノミー 、ONTAP-NAS-FlexArray グループ、 ONTAP-SAN
IOPS	整数	正の整数	プールは、この範囲内で IOPS を保証する機能を備えています	ボリュームで IOPS が保証されました	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した場合、シックプロビジョニングボリュームが使用されます。ただし、 Element ストレージプールでは、提供されている IOPS の最小値と最大値を使用して、要求された値ではなく 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のみに使用されます。

理想的には ' 属性だけを使用して ' 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特性をモデル化できますTrident は ' 指定した属性の *ALL* 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切なプールを自動的に選択できない場合は、「 toragePools 」および「 additionalStoragePools 」パラメータを使用してプールをさらに改良したり、特定のプールセットを選択したりできます。

'toragePools' パラメータを使用すると ' 指定した属性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 'attributes' パラメータと 'toragePools' パラメータで指定されたプールの交点をプロビジョニングに使用しますどちらか一方のパラメータを単独で使用することも、両方を同時に使用することも

「 additionalStoragePools 」パラメータを使用すると、「 attributes 」パラメータと「 toragePools 」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。

'toragePools' パラメータと 'additionalStoragePools' パラメータでは ' 各エントリは '<backend>:<storagePoolList>' の形式で指定し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。これらのリストでは、バックエンド値とリスト値の両方に正規表現値を使用できます。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、動的プロビジョニングの際に Trident が選択するストレージプール / バックエンドには影響しません。代わりに、 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を必要とする場合があります。

属性	を入力します	値	説明	関連するドライバ	**Kubernetes** のバージョン
FSstype （英語）	文字列	ext4、ext3、xfs	ブロックボリュームのファイルシステムのタイプ	solidfire-san-group、ontap/nas、ontap -nas-エコノミー、ontap -nas-flexgroup 、ontap -san 、ONTAP - SAN -経済性	すべて
allowVolumeExpansion の略	ブール値	true false	PVC サイズの拡張のサポートをイネーブルまたはディセーブルにします	ontap-nas 、ontap-nas-economy、ontap -nas-flexgroup 、ontap-san 、ontap-san-economy、solidfire-san、azure-netapp-files	1.11 以上
volumeBindingMode のようになりました	文字列	即時、WaitForFirstConsumer	ボリュームバインドと動的プロビジョニングを実行するタイミングを選択します	すべて	1.19～1.26

- 。 `fsType` パラメータは、SAN LUNに必要なファイルシステムタイプを制御する場合に使用します。また、Kubernetesでは、の機能も使用されます `fsType` ファイルシステムが存在することを示すために、ストレージクラスに格納します。ボリューム所有権は、を使用して制御できます `fsGroup` ポッドのセキュリティコンテキスト（使用する場合のみ） `fsType` が設定されます。を参照してください "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します" を使用したボリューム所有権の設定の概要については、を参照してください `fsGroup` コンテキスト（Context）。Kubernetesでが適用されます `fsGroup` 次の場合のみ値を指定します

  - 「 fsType 」はストレージクラスで設定されます。
  - PVC アクセスモードは RWO です。

  NFS ストレージドライバの場合、 NFS エクスポートには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この値は 'NFS' に設定することも ' ヌル以外の任意の値に設定することもできます

- を参照してください "ボリュームを展開します" ボリューム拡張の詳細については、を参照してください。
- Trident インストーラバンドルには、「 `sample -input/storageclass-*.yaml 」で Trident で使用するストレージクラス定義の例がいくつか用意されています。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 'S torageClasses' に似ていますこの Snapshot コピーは、複数のストレージクラスの定義に役立ちます。また、ボリューム Snapshot によって参照され、Snapshot を必要な Snapshot クラスに関連付けます。各ボリューム Snapshot は、単一のボリューム Snapshot クラスに関連付けられます。

スナップショットを作成するには 'VolumeSnapshotClass' を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スナップショットの要求が Trident によって処理される Kubernetes を指定します。「要素ポリシー」は、スナップショットを削除する必要がある場合に実行されるアクションを指定します。「削除ポリシー」が「削除」に設定されている場合、 Snapshot を削除すると、ボリューム Snapshot オブジェクトおよびストレージクラスタ上の基盤となる Snapshot は削除されます。または、「 Retain 」に設定すると、「 VolumeSnapshotContent 」と物理スナップショットが保持されます。

Kubernetes `VolumeSnapshot` オブジェクト

Kubernetes の VolumeSnapshot オブジェクトは ' 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からの要求を表すのと同様に、ボリュームスナップショットは、ユーザが既存の PVC のスナップショットを作成する要求です。

ボリュームスナップショット要求が受信されると、 Trident はバックエンドでのボリュームのスナップショット作成を自動的に管理し、ユニークな「 VolumeSnapshotContent 」オブジェクトを作成することによっ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。つまり、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 を削除すると、 Trident はその PVC のバッキングボリュームを **Deleting** 状態でマークしますが、完全には削除しません。関連付けられている Snapshot がすべて削除されると、ボリュームは削除されます。

Kubernetes `VolumeSnapshotContent` オブジェクト

Kubernetes の「 VolumeSnapshotContent 」オブジェクトは、すでにプロビジョニングされている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 」と似ており、ストレージ・クラスタ上でプロビジョニングされた Snapshot を表します。「 PersistentVolumeClaim 」および「 PersistentVolume 」オブジェクトと同様に、スナップショットが作成されると、「 VolumeContent Snapshot」 オブジェクトは「 VolumeSnapshot 」オブジェクトへの 1 対 1 のマッピングを保持します。これは、スナップショットの作成を要求しました。

「 VolumeSnapshotContent 」オブジェクトには、スナップショットを一意に識別する詳細（「 napshotHandle 」など）が含まれています。この「 napshotHandle 」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ものです。

Trident では、スナップショット要求を受信すると、バックエンドにスナップショットが作成されます。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、そのスナップショットを Kubernetes API に公開します。

> 通常、オブジェクトを管理する必要はありませ `VolumeSnapshotContent`ん。ただし、Trident の外部でを作成する場合は例外です"ボリュームSnapshotのインポート"。

`VolumeGroupSnapshotClass`Kubernetesオブジェクト

Kubernetes `VolumeGroupSnapshotClass`オブジェクトはに似てい `VolumeSnapshotClass`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スナップショットによって参照され、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。各ボリュームグループスナップショットは、単一のボリュームグループスナップショットクラスに関連付けられます。

あ `VolumeGroupSnapshotClass`スナップショットのグループを作成するには、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、以下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
```

はTridentによって処理されます。に実行するアクションを指定します。 `deletionPolicy`設定されている `Delete`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 オブジェクトと、ストレージ クラスター上の基礎となるスナップショットが削除されます。または、に設定する `Retain`と、 `VolumeGroupSnapshotContent`物理Snapshotが保持されます。

`VolumeGroupSnapshot`Kubernetes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ための要求です。PVCがユーザーによるボリュームへ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、ユーザーによる既存のPVCのスナップショットを作成するための要求です。

ボリュームグループ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、一意のスナップショットを作成してスナップショットを公開します。 `VolumeGroupSnapshotContent`物体。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 とは独立しています。つまり、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 を削除すると、 Trident はその PVC のバッキングボリュームを **Deleting** 状態でマークしますが、完全には削除しません。ボリュームグループスナップショットは、関連するすべてのスナップショットが削除されると削除されます。

`VolumeGroupSnapshotContent`Kubernetes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 スナップショットを表します。これは、に似て `PersistentVolume`おり、ストレージクラスタにプロビジョニングされたSnapshotを示します。オブジェクトと `PersistentVolume`オブジェクトと同様に、 `PersistentVolumeClaim`Snapshotが作成されると、 `VolumeSnapshotContent`オブジェクトはSnapshotの作成を要求したオブジェクトへの1対1のマッピングを保持し `VolumeSnapshot`ます。

その `VolumeGroupSnapshotContent`オブジェクトには、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。 `volumeGroupSnapshotHandle`およびストレージ システム上に存在する個別の volumeSnapshotHandles。

スナップショット要求が届くと、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`オブジェクトを作成し、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、管理者が定義した Kubernetes API 内のエンドポイントであり、類似するオブジェクトのグループ化に使用されます。Kubernetes では、オブジェクトのコレクションを格納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を取得するには 'kubectl get CRDs' を実行します

カスタムリソース定義（ CRD ）と関連する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格納されます。これにより、 Trident の独立したストアが不要になります。

Tridentは、オブジェクトを使用し `CustomResourceDefinition`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 ID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。また、 CSI のボリュームスナップショットフレームワークには、ボリュームスナップショットの定義に必要ないくつかの SSD が導入されています。

CRD は Kubernetes の構成要素です。上記で定義し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 tridentBackendsCRD オブジェクトが Kubernetes によって消費されるように作成されます

Trident の CRD については、次の点に注意してください。

- Trident をインストールすると、一連の CRD 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。
- Tridentをアンインストールするには、を使用します `tridentctl uninstall` コマンドであるTridentポッドが削除されましたが、作成されたSSDはクリーンアップされません。を参照してください "Trident をアンインストールします" Trident を完全に削除して再構成する方法を理解する。

Trident `StorageClass`オブジェクト

TridentではKubernetesに対応するストレージクラスが作成されます `StorageClass` を指定するオブジェクト `csi.trident.netapp.io` プロビジョニング担当者のフィールドに入力します。ストレージクラス名がKubernetesの名前と一致していること `StorageClass` 表すオブジェクト。

> Kubernetes では、 Trident をプロビジョニングツールとして使用する Kubernetes 「 torageClass 」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、ボリュームの一連の要件で構成されます。Trident は、これらの要件と各ストレージプール内の属性を照合し、一致する場合は、そのストレージプールが、そのストレージクラスを使用するボリュームのプロビジョニング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設定を作成できます。ただし、 Kubernetes の導入では、新しい Kubernetes の「 torageClass 」オブジェクトを登録するときに、これらのオブジェクトが作成されることを期待しています。

Trident バックエンドオブジェクト

バックエンドとは、 Trident がボリュームをプロビジョニングする際にストレージプロバイダを表します。 1 つの Trident インスタンスであらゆる数のバックエンドを管理できます。

> これは、自分で作成および管理する 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 の「 torageClass 」オブジェクトです。

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

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に使用できる個別の場所を表します。ONTAPの場合、これらは SVM のアグリゲートに対応します。NetApp HCI/ SolidFireの場合、これらは管理者が指定した QoS バンドに対応します。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。

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て管理されます。

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 LUN、FC LUNなどのバックエンドエンドエンドポイントで構成されます。Kubernetesでは、これらはに直接対応し `PersistentVolumes`ます。ボリュームを作成するときは、そのボリュームにストレージクラスが含まれていることを確認します。このクラスによって、ボリュームをプロビジョニングできる場所とサイズが決まります。

- Kubernetes では、これらのオブジェクトが自動的に管理されます。Trident がプロビジョニングしたものを表示できます。
- 関連付けられた Snapshot がある PV を削除すると、対応する Trident ボリュームが * Deleting * 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 Snapshot 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	を入力します	必須	説明
バージョン	文字列	いいえ	Trident API のバージョン（「 1 」）
名前	文字列	はい。	作成するボリュームの名前
ストレージクラス	文字列	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	文字列	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	文字列	いいえ	使用するプロトコルの種類：「 file 」または「 block 」
インターン名	文字列	いいえ	Trident が生成した、ストレージシステム上のオブジェクトの名前
cloneSourceVolume の実行中です	文字列	いいえ	ONTAP （ NAS 、 SAN ） & SolidFire - * ：クローン元のボリュームの名前

属性	を入力します	必須	説明
splitOnClone	文字列	いいえ	ONTAP （ NAS 、 SAN ）：クローンを親からスプリットします
Snapshot ポリシー	文字列	いいえ	ONTAP - * ：使用する Snapshot ポリシー
Snapshot リザーブ	文字列	いいえ	ONTAP - * ： Snapshot 用にリザーブされているボリュームの割合
エクスポートポリシー	文字列	いいえ	ONTAP-NAS* ：使用するエクスポートポリシー
snapshotDirectory の略	ブール値	いいえ	ONTAP-NAS* ： Snapshot ディレクトリが表示されているかどうか
unixPermissions	文字列	いいえ	ONTAP-NAS* ：最初の UNIX 権限
ブロックサイズ	文字列	いいえ	SolidFire - * ：ブロック / セクターサイズ
ファイルシステム	文字列	いいえ	ファイルシステムのタイプ
リカバリキューをスキップ	文字列	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 ' ボリュームの作成時に internalName を生成しますこの構成は 2 つのステップで構成されます。最初に、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構成のプレフィックス）をボリューム名の前に付加し、「 <prefix> - <volume-name> 」という形式の名前を付けます。その後、名前の完全消去が行われ、バックエンドで許可されていない文字が置き換えられます。ONTAP バックエンドでは、ハイフンをアンダースコアで置き換えます（つまり、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。

ボリューム設定を使用して、 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 環境では、ほとんどのユーザが標準の Kubernetes の「 PersistentVolumeClaim 」メソッドを使用することを想定しています。Trident は、プロビジョニングプロセスの一環として、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

Snapshot はボリュームのポイントインタイムコピーで、新しいボリュームのプロビジョニングやリストア状態に使用できます。Kubernetes では ' これらは 'VolumeSnapshotContent' オブジェクトに直接対応します各 Snapshot には、 Snapshot のデータのソースであるボリュームが関連付けられます。

個々の「スナップショット」オブジェクトには、以下のプロパティが含まれています。

属性	を入力します	必須	説明
バージョン	文字列	はい。	Trident API のバージョン（「 1 」）
名前	文字列	はい。	Trident Snapshot オブジェクトの名前
インターン名	文字列	はい。	ストレージシステム上の Trident Snapshot オブジェクトの名前
ボリューム名	文字列	はい。	Snapshot を作成する永続的ボリュームの名前
ボリュームの内部名	文字列	はい。	ストレージシステムに関連付けられている 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が自動的に管理されます。Trident がプロビジョニングしたものを表示できます。

Kubernetes の「 VolumeSnapshot 」オブジェクト要求が作成されると、 Trident は元の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によって動作します。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「 VolumeSnapshot 」オブジェクトの「 UID 」を組み合わせることによって生成されます（例：「 snapshot-e8d8d8a0ca-9826-11e9-9807-525400f3f660 」）。「 volumeName 」と「 volumeInternalName 」には、バッキングボリュームの詳細を取得して値を設定します。

Trident `ResourceQuota`オブジェクト

Tridentデーモンセットは、Kubernetesで利用可能な最高の優先度クラスであるプライオリティクラスを使用して `system-node-critical`、ノードの正常なシャットダウン時にTridentがボリュームを識別してクリーンアップできるようにし、リソースへの負荷が高いクラスタでは、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るようにします。

これを実現するために、Tridentはオブジェクトを使用し `ResourceQuota`て、`Trident`デーモンセットの「`system-node-critical`」優先クラスを確実に満たします。デプロイメントおよびデーモンセットの作成の前に、`Trident`はオブジェクトを検索し `ResourceQuota`、検出されていない場合は適用します。

デフォルトのリソース割り当ておよび優先クラスをより詳細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オブジェクトを構成します

次に示すのは'ResourceQuota'オブジェクトがTridentのデマ作用を優先する例です

```
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、を参照してください。 "Kubernetes：リソースクォータ"。

クリーンアップ `ResourceQuota` インストールが失敗した場合

まれに'ResourceQuotaオブジェクトが作成された後にインストールが失敗する場合は'最初に実行します "アンインストール中です" を再インストールします。

それでも解決しない場合は'ResourceQuotaオブジェクトを手動で削除します

取り外します `ResourceQuota`

独自のリソース割り当てを制御する場合は、次のコマンドを使用してTridentオブジェクトを削除できます `ResourceQuota`。

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

PODセキュリティ標準（PSS）およびセキュリティコンテキストの制約（SCC）

Kubernetesポッドのセキュリティ標準（PSS）とポッドのセキュリティポリシー（PSP）によって、権限レベルが定義され、ポッドの動作が制限されます。また、OpenShift Security Context Constraints（SCC）でも、OpenShift Kubernetes Engine固有のポッド制限を定義します。このカスタマイズを行うために、Tridentはインストール時に特定の権限を有効にします。次のセクションでは、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セキュリティポリシー（PSP）に代わるものです。PSPはKubernetes v1.21で廃止され、v1.25で削除されます。詳細については、を参照してください。 "Kubernetes：セキュリティ"。

必須のKubernetes Security Contextと関連フィールド

アクセス権	説明
権限があります	CSIでは、マウントポイントが双方向である必要があります。つまり、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。詳細については、を参照してください "Kubernetes：マウントの伝播"。
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'IPC（プロセス間通信）を使用して'nfsdと通信します
ホストPID	NFSで開始する必要があり `rpc-statd`ます。Tridentは、NFSボリュームをマウントする前にが実行されているかどうかをホストプロセスに照会して判断し `rpc-statd`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は特権コンテナにこれらの機能を設定します。「CapPrm：0000003fffffffffffff`Capff：0000003fffffffff'」
Seccom	Seccompプロファイルは特権コンテナでは常に「制限されていない」ため、Tridentでは有効にできません。
SELinux	OpenShiftでは、特権コンテナは（「Super Privileged Container」）ドメインで実行され、非特権コンテナはドメインで実行さ `spc_t`れ `container_t`ます。では `containerd`、がインストールされていると `container-selinux`、すべてのコンテナがドメイン内で実行され `spc_t`、SELinuxが実質的に無効になります。そのため、Tridentはコンテナに追加しません `seLinuxOptions`。
DAC	特権コンテナは、ルートとして実行する必要があります。CSIに必要なUNIXソケットにアクセスするために、非特権コンテナはrootとして実行されます。

PODセキュリティ標準（PSS）

ラベル	説明	デフォルト
'pod -security.esvus.io/enforce `po-security.Kubernetes io/enforce-version	Tridentコントローラとノードをインストールネームスペースに登録できるようにします。ネームスペースラベルは変更しないでください。	「enforce：特権」「enforce-version：」は、現在のクラスタのバージョンまたはテストされたPSSの最新バージョンです

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、「Error creating：…」または「Warning：trident-csi-…」が表示される場合があります。この場合は'Privilegedの名前空間ラベルが変更されていないかどうかを確認してくださいその場合は、Tridentを再インストールします。

PoDセキュリティポリシー（PSP）

フィールド	説明	デフォルト
'allowPrivilegeEscalation'	特権コンテナは、特権昇格を許可する必要があります。	「真」
allowedCSIDrivers	TridentはインラインCSIエフェメラルボリュームを使用しません。	空です
「allowedCapabilities」を参照してください	権限のないTridentコンテナにはデフォルトよりも多くの機能が必要ないため、特権コンテナには可能なすべての機能が付与されます。	空です
「allowedFlexVolumes」を参照してください	Tridentはを利用しません "FlexVolドライバ"そのため、これらのボリュームは許可されるボリュームのリストに含まれていません。	空です
「allowedHostPaths」を参照してください	Tridentノードポッドでノードのルートファイルシステムがマウントされるため、このリストを設定してもメリットはありません。	空です
allowedProcMountTypes	Tridentでは'ProcMountTypes'は使用しません	空です
"allowedUnsafeSysctls"	Tridentには安全でないsysctls'は必要ありません。	空です
defaultAddCapabilities	特権コンテナに追加する機能は必要ありません。	空です
defaultAllowPrivilegeEscalation`	権限の昇格は、各Tridentポッドで処理されます。	「偽」
「forbiddenSysctls」と入力します	sysctlsは許可されません	空です
「fsGroup」と入力します	Tridentコンテナはrootとして実行されます。	「RunAsAny」
「hostIPC」	NFSボリュームをマウントするには'nfsdと通信するためにホストIPCが必要です	「真」
「ホストネットワーク」	iscsiadmには、iSCSIデーモンと通信するためのホストネットワークが必要です。	「真」
「hostPID」	ノードでRPC-statdが実行されているかどうかを確認するには'ホストPIDが必要です	「真」

フィールド	説明	デフォルト
「hostPorts」	Tridentはホストポートを使用しません。	空です
「特権」	Tridentノードのポッドでは、ボリュームをマウントするために特権コンテナを実行する必要があります。	「真」
「readOnlyRootFilesystem」	Tridentノードのポッドは、ノードのファイルシステムに書き込む必要があります。	「偽」
DDropCapabilitiesが必要です	Tridentノードのポッドは特権コンテナを実行するため、機能をドロップ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	空です
SELinux	Tridentは'seLinuxOptions'を設定していませんこれは'コンテナランタイムとKubernetesディストリビューションがSELinuxを処理する方法には現在のところ違いがあるためです	空です
「supplementalGroups」を参照してください	Tridentコンテナはrootとして実行されます。	「RunAsAny」
「ボリューム」	Tridentポッドには、このボリュームプラグインが必要です。	「hostPath」、「projected」、「emptyDir」

セキュリティコンテキストの制約（**SCC**）

ラベル	説明	デフォルト
allowHostDirVolumePlugin	Tridentノードのポッドは、ノードのルートファイルシステムをマウントします。	「真」
"allowHostIPC"	NFSボリュームをマウントするには'nfsdと通信するためにホストIPCが必要です	「真」
「allowHostNetwork」を参照してください	iscsiadmには、iSCSIデーモンと通信するためのホストネットワークが必要です。	「真」
「allowHostPID」を参照してください	ノードでRPC-statdが実行されているかどうかを確認するには'ホストPIDが必要です	「真」

ラベル	説明	デフォルト
"allowHostPorts`	Tridentはホストポートを使用しません。	「偽」
'allowPrivilegeEscalation'	特権コンテナは、特権昇格を許可する必要があります。	「真」
allowPrivilegeContainer]を参照してください	Tridentノードのポッドでは、ボリュームをマウントするために特権コンテナを実行する必要があります。	「真」
"allowedUnsafeSysctls"	Tridentには安全でないsysctls'は必要ありません。	「 NONE 」
「allowedCapabilities」を参照してください	権限のないTridentコンテナにはデフォルトよりも多くの機能が必要ないため、特権コンテナには可能なすべての機能が付与されます。	空です
defaultAddCapabilities	特権コンテナに追加する機能は必要ありません。	空です
「fsGroup」と入力します	Tridentコンテナはrootとして実行されます。	「RunAsAny」
「グループ」	このSCCはTridentに固有で、ユーザにバインドされています。	空です
「readOnlyRootFilesystem」	Tridentノードのポッドは、ノードのファイルシステムに書き込む必要があります。	「偽」
DDropCapabilitiesが必要です	Tridentノードのポッドは特権コンテナを実行するため、機能をドロップ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'を設定していませんこれは'コンテナランタイムとKubernetesディストリビューションがSELinuxを処理する方法には現在のところ違いがあるためです	空です
「seccompProfiles」	特権のあるコンテナは常に「閉鎖的」な状態で実行されます。	空です
「supplementalGroups」を参照してください	Tridentコンテナはrootとして実行されます。	「RunAsAny」
「ユーザー」	このSCCをTridentネームスペースのTridentユーザにバインドするエントリが1つあります。	該当なし
「ボリューム」	Tridentポッドには、こ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