ベストプラクティスと推奨事項
Trident

NetApp
July 01, 2026

目次

ベストプラクティスと推奨事項

導入

Tridentを導入する際は、ここに記載されている推奨事項を使用してください。

専用の名前空間にデプロイする

"ネームスペース"異なるアプリケーション間の管理上の分離を提供し、リソース共有の障壁となります。たとえば、ある名前空間のPVCを別の名前空間で使用することはできません。TridentはKubernetesクラスター内のすべての名前空間にPVリソースを提供し、その結果、昇格された権限を持つサービスアカウントを活用します。

さらに、Tridentポッドへのアクセスにより、ユーザーがストレージシステムの資格情報やその他の機密情報にアクセスできるようになる可能性があります。アプリケーションユーザーと管理アプリケーションがTridentオブジェクト定義またはポッド自体にアクセスできないようにすることが重要です。

クォータと範囲制限を使用してストレージ消費を制御

Kubernetesには2つの機能があり、これらを組み合わせることで、アプリケーションによるリソース消費を制限する強力なメカニズムが提供されます。 "ストレージクォータメカニズム"により、管理者は、グローバルおよびストレージクラス固有の容量とオブジェクト数の消費制限をネームスペースごとに実装できます。さらに、 "範囲制限"を使用することで、リクエストがプロビジョナーに転送される前に、PVCリクエストが最小値と最大値の範囲内であることを確認します。

これらの値は名前空間ごとに定義されます。つまり、各名前空間には、そのリソース要件に適合する値が定義されている必要があります。詳細については、こちらをご覧ください "割り当てを活用する方法"。

ストレージ構成

NetApp ポートフォリオの各ストレージプラットフォームには、コンテナ化されているかどうかに関係なく、アプリケーションに役立つ独自の機能があります。

プラットフォームの概要

TridentはONTAPおよびElementと連携します。すべてのアプリケーションやシナリオに他のプラットフォームよりも適したプラットフォームは存在しませんが、プラットフォームを選択する際には、アプリケーションのニーズとデバイスを管理するチームを考慮する必要があります。

利用しているプロトコルに応じて、ホスト オペレーティング システムのベースライン ベスト プラクティスに従う必要があります。オプションとして、利用可能な場合は、バックエンド、ストレージ クラス、および PVC 設定にアプリケーションのベスト プラクティスを組み込んで、特定のアプリケーションのストレージを最適化することを検討することもできます。

ONTAPおよびCloud Volumes ONTAPのベストプラクティス

Trident 向けに ONTAP および Cloud Volumes ONTAP を設定する際のベストプラクティスを学習します。

以下の推奨事項は、Tridentによって動的にプロビジョニングされるボリュームを消費するコンテナ化されたワークロード向けにONTAPを設定するためのガイドラインです。それぞれの推奨事項について、お客様の環境における適切性を考慮して評価する必要があります。

Trident専用の**SVM**を使用する

Storage Virtual Machine（SVM）は、ONTAPシステム上のテナント間の分離と管理の分離を実現します。SVMをアプリケーション専用にすることで、権限の委任が可能になり、リソース消費を制限するためのベストプラクティスを適用できるようになります。

SVM の管理にはいくつかのオプションがあります：

- バックエンド構成でクラスタ管理インターフェイスを指定し、適切なクレデンシャルとともに SVM 名を指定します。
- ONTAP System Manager または CLI を使用して、 SVM 専用の管理インターフェイスを作成します。
- 管理ロールを NFS データインターフェイスと共有します。

いずれの場合も、インターフェイスはDNSに登録され、Tridentの設定時にDNS名を使用する必要があります。これにより、ネットワークIDの保持を使用しないSVM-DRなど、一部のDRシナリオが容易になります。

SVM に専用の管理 LIF を使用するか、共有の管理 LIF を使用するかという優先順位はありませんが、ネットワーク セキュリティ ポリシーが選択したアプローチと一致していることを確認する必要があります。いずれにせよ、管理 LIF は DNS 経由でアクセスできるようにして、 "SVM-DR"Trident と組み合わせて使用する場合に最大限の柔軟性を実現する必要があります。

最大ボリューム数を制限する

ONTAP ストレージシステムには最大ボリューム数があり、ソフトウェアのバージョンとハードウェア プラットフォームによって異なります。特定のプラットフォームと ONTAP バージョンの正確な制限を確認するには、 "NetApp Hardware Universe"を参照してください。ボリューム数が上限に達すると、Trident だけでなく、すべてのストレージ要求に対してプロビジョニング操作が失敗します。

Tridentの `ontap-nas`および `ontap-san`ドライバーは、作成される各Kubernetes永続ボリューム（PV）に対してFlexVolumeをプロビジョニングします。 `ontap-nas-economy`ドライバーは、200PVごとに約1つのFlexVolumeを作成します（50～300の間で設定可能）。 `ontap-san-economy`ドライバーは、100PVごとに約1つのFlexVolumeを作成します（50～200の間で設定可能）。Tridentがストレージシステム上の使用可能なボリュームをすべて消費しないようにするには、SVMに制限を設定する必要があります。コマンドラインから次のように実行できます：

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

`max-volumes`の値は、環境に固有のいくつかの基準によって異なります：

- ONTAP クラスタ内の既存のボリューム数
- Trident 以外のアプリケーション用にプロビジョニングする予定のボリュームの数
- Kubernetes アプリケーションで使用される永続的ボリュームの数

この `max-volumes`値は、ONTAP クラスタ内のすべてのノードにプロビジョニングされたボリュームの合計

であり、個々の ONTAP ノードのものではありません。その結果、ONTAP クラスタノードによっては、Trident でプロビジョニングされたボリュームが他のノードよりもはるかに多い場合や少ない場合があります。

たとえば、2ノードのONTAPクラスタには、最大2000個のFlexVolボリュームをホストする機能があります。最大ボリューム数を1250に設定することは非常に妥当に思えます。ただし、1つのノードからの "アグリゲート"のみがSVMに割り当てられている場合、または1つのノードから割り当てられたアグリゲートをプロビジョニングできない場合（容量不足など）、もう一方のノードがすべてのTridentプロビジョニングボリュームのターゲットになります。つまり、 `max-volumes`の値に達する前に、そのノードのボリューム制限に達する可能性があり、その結果、Tridentとそのノードを使用する他のボリューム操作の両方に影響を与えます。この状況を回避するには、クラスタ内の各ノードからのアグリゲートが、**Trident**で使用される**SVM**に同数割り当てられるようにします。

ボリュームをクローニングする

NetApp Tridentは、 `ontap-nas`、 `ontap-san`、および `solidfire-san`ストレージドライバーを使用する場合、ボリュームのクローン作成をサポートします。 `ontap-nas-flexgroup`または `ontap-nas-economy`ドライバーを使用する場合、クローン作成はサポートされません。既存のボリュームから新しいボリュームを作成すると、新しいスナップショットが作成されます。

異なるStorageClassに関連付けられたPVCの複製は避けてください。互換性を確保し、予期しない動作を防ぐために、同じStorageClass内でクローン操作を実行してください。

Tridentで作成されるボリュームの最大サイズを制限する

Tridentで作成できるボリュームの最大サイズを設定するには、 `limitVolumeSize`パラメータを `backend.json`定義で使用します。

ストレージ アレイでのボリューム サイズの制御に加えて、Kubernetes の機能も活用する必要があります。

Tridentによって作成される**FlexVols**の最大サイズを制限する

ontap-san-economyドライバおよびontap-nas-economyドライバのプールとして使用されるFlexVolsの最大サイズを設定するには、 `limitVolumePoolSize`パラメータを `backend.json`定義で使用してください。

双方向**CHAP**を使用するように**Trident**を設定

バックエンド定義で CHAP イニシエーターとターゲットのユーザー名とパスワードを指定し、Trident で SVM の CHAP を有効にすることができます。バックエンド構成で `useCHAP`パラメータを使用すると、Trident は CHAP を使用して ONTAP バックエンドの iSCSI 接続を認証します。

SVM QoSポリシーを作成して使用する

SVMに適用されるONTAP QoSポリシーを活用することで、Tridentでプロビジョニングされたボリュームが消費できるIOPSの数を制限します。これにより、 "いじめを防ぐ"または制御不能なコンテナがTrident SVM外部のワークロードに影響を与えるのを防ぎます。

SVM の QoS ポリシーは数ステップで作成できます。最も正確な情報については、お使いのバージョンの ONTAP のドキュメントを参照してください。以下の例では、SVM で使用可能な合計 IOPS を 5000 に制限する QoS ポリシーを作成します。

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

さらに、ONTAP のバージョンでサポートされている場合は、コンテナ化されたワークロードへのスループット量を保証するために、QoS 最小値の使用を検討できます。アダプティブ QoS は SVM レベルのポリシーと互換性がありません。

コンテナ化されたワークロード専用の IOPS の数は、さまざまな側面によって異なります。とりわけ、次のようなものがあります：

- ストレージ アレイを使用するその他のワークロード。Kubernetes デプロイメントに関連しない他のワークロードがストレージ リソースを利用している場合は、それらのワークロードが誤って悪影響を受けないように注意する必要があります。
- コンテナ内で実行されると予想されるワークロード。高い IOPS 要件を持つワークロードがコンテナ内で実行される場合、低い QoS ポリシーではエクスペリエンスが悪くなります。

SVM レベルで割り当てられた QoS ポリシーにより、SVM にプロビジョニングされたすべてのボリュームが同じ IOPS プールを共有することになるということを覚えておくことが重要です。コンテナ化されたアプリケーションの 1 つまたは少数に高い IOPS 要件がある場合、他のコンテナ化されたワークロードに対して脅威となる可能性があります。このような場合は、外部自動化を使用してボリュームごとの QoS ポリシーを割り当てることを検討してください。

QoSポリシーグループをSVMに割り当てる必要があるのは、ONTAPバージョンが9.8より前の場合*のみ*です。

Trident用の**QoS**ポリシーグループを作成する

サービス品質（QoS）は、競合するワークロードによって重要なワークロードのパフォーマンスが低下しないことを保証します。ONTAP QoS ポリシーグループはボリュームの QoS オプションを提供し、ユーザーが 1 つ以上のワークロードのスループット上限を定義できるようにします。QoS の詳細については、 "QoSによるスループットの保証"を参照してください。バックエンドまたはストレージプールで QoS ポリシーグループを指定すると、そのプールまたはバックエンドで作成された各ボリュームに適用されます。

ONTAP には、従来型とアダプティブ型の 2 種類の QoS ポリシー グループがあります。従来のポリシー グループは、IOPS で一定した最大（または後のバージョンでは最小）スループットを提供します。アダプティブ QoS は、ワークロードのサイズの変化に応じて IOPS と TB|GB の比率を維持しながら、スループットをワークロードのサイズに合わせて自動的にスケーリングします。これは、大規模な展開で数百または数千のワークロードを管理する場合に大きな利点となります。

QoS ポリシー グループを作成するときは、次の点を考慮してください：

- `qosPolicy`キーは、バックエンド構成の `defaults`ブロックに設定する必要があります。次のバックエンド構成の例を参照してください：

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
  - labels:
      performance: extreme
    defaults:
      adaptiveQosPolicy: extremely-adaptive-pg
  - labels:
      performance: premium
    defaults:
      qosPolicy: premium-pg
```

- 各ボリュームがポリシー グループで指定されたスループット全体を取得できるように、ボリュームごとにポリシー グループを適用する必要があります。共有ポリシー グループはサポートされていません。

QoSポリシーグループの詳細については、 "ONTAPコマンド リファレンス"を参照してください。

Kubernetes クラスタメンバーへのストレージリソースアクセスを制限する

Tridentによって作成されたNFSボリューム、iSCSI LUN、FC LUNへのアクセスを制限することは、Kubernetesデプロイメントのセキュリティ体制にとって重要なコンポーネントです。こうすることで、Kubernetesクラスターの一部ではないホストがボリュームにアクセスして予期せずデータを変更する可能性を防ぐことができます。

名前空間は Kubernetes 内のリソースの論理的な境界であることを理解することが重要です。同じ名前空間内のリソースは共有できると想定されていますが、重要なのは、名前空間間の機能がないことです。つまり、PV はグローバル オブジェクトですが、PVC にバインドされている場合は、同じ名前空間にあるポッドからのみアクセスできます。適切な場合には、名前空間を使用して分離を行うことが重要です。

Kubernetes コンテキストでのデータセキュリティに関してほとんどの組織が主に懸念するのは、コンテナ内のプロセスが、ホストにマウントされているがコンテナ用ではないストレージにアクセスできることです。 "ネームスペース"は、この種の侵害を防ぐように設計されています。ただし、特権コンテナという例外が 1 つあります。

特権コンテナとは、通常よりも大幅に高いホストレベルの権限で実行されるコンテナです。これらはデフォルトでは拒否されないため、 "Pod セキュリティポリシー"を使用してこの機能を無効にしてください。

Kubernetes と外部ホストの両方からアクセスする必要があるボリュームの場合、ストレージは従来の方法で管理する必要があります。PV は管理者によって導入され、Trident では管理されません。これにより、Kubernetes と外部ホストの両方が切断され、ボリュームを使用しなくなった場合にのみ、ストレージ ボリュームが破棄されるようになります。さらに、カスタム エクスポート ポリシーを適用して、Kubernetes ク

ラスター ノードおよび Kubernetes クラスター外部の対象サーバーからのアクセスを有効にすることもできます。

専用のインフラストラクチャノードを持つデプロイメント（例：OpenShift）またはユーザーアプリケーションをスケジュールできないその他のノードの場合は、別のエクスポートポリシーを使用して、ストレージリソースへのアクセスをさらに制限する必要があります。これには、これらのインフラストラクチャノードに導入されるサービス（例：OpenShift MetricsおよびLoggingサービス）、およびインフラストラクチャ以外のノードに導入される標準アプリケーションのエクスポートポリシーの作成が含まれます。

専用のエクスポートポリシーを使用する

各バックエンドに対して、Kubernetes クラスター内に存在するノードへのアクセスのみを許可するエクスポート ポリシーが存在することを確認する必要があります。Trident はエクスポート ポリシーを自動的に作成および管理できます。これにより、Trident は Kubernetes クラスター内のノードにプロビジョニングされたボリュームへのアクセスを制限し、ノードの追加/削除を簡素化します。

あるいは、エクスポート ポリシーを手動で作成し、各ノード アクセス要求を処理する 1 つ以上のエクスポート ルールを設定することもできます：

- `vserver export-policy create` ONTAP CLI コマンドを使用して、エクスポート ポリシーを作成します。
- エクスポートポリシーにルールを追加するには、 `vserver export-policy rule create` ONTAP CLI コマンドを使用します。

これらのコマンドを実行すると、データにアクセスできる Kubernetes ノードを制限できます。

アプリケーション **SVM** の **`showmount`**を無効にします

`showmount`機能により、NFSクライアントはSVMに対して利用可能なNFSエクスポートのリストを照会できます。Kubernetesクラスタにデプロイされたポッドは、 `showmount -e`コマンドを実行し、アクセスできないマウントも含め、使用可能なマウントのリストを受け取ることができます。これ自体はセキュリティ侵害ではありませんが、権限のないユーザーがNFSエクスポートに接続する際に役立つ可能性のある不要な情報を提供することになります。

`showmount`を無効にするには、SVMレベルのONTAP CLIコマンドを使用します：

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire ベストプラクティス

Trident 用の SolidFire ストレージを設定する際のベストプラクティスを学びます。

SolidFireアカウントを作成する

各SolidFireアカウントは一意のボリューム所有者を表し、独自のチャレンジハンドシェイク認証プロトコル（CHAP）クレデンシャルのセットを受け取ります。アカウントに割り当てられたボリュームには、アカウント名と関連するCHAPクレデンシャルを使用するか、ボリュームアクセスグループを通じてアクセスできま

す。1つのアカウントには最大2,000個のボリュームを割り当てることができますが、1つのボリュームは1つのアカウントにのみ属することができます。

QoS ポリシーを作成する

SolidFire のサービス品質（QoS）ポリシーは、多くのボリュームに適用できる標準化されたサービス品質設定を作成して保存する場合に使用します。

ボリュームごとに QoS パラメータを設定できます。QoS を定義する 3 つの構成可能なパラメータ（最小 IOPS、最大 IOPS、バースト IOPS）を設定することで、各ボリュームのパフォーマンスを保証できます。

4KB ブロック サイズで可能な最小、最大、およびバースト IOPS 値は次のとおりです。

IOPSパラメータ	定義	最小値	デフォルト値	最大値（**4Kb**）
最小 IOPS	ボリュームの保証されたパフォーマンス レベル。	50	50	15000
最大 IOPS	パフォーマンスはこの制限を超えることはありません。	50	15000	200,000
バースト IOPS	短いバーストシナリオで許可される最大 IOPS。	50	15000	200,000

> 最大 IOPS とバースト IOPS は最大 200,000 まで設定できますが、ボリュームの実際の最大パフォーマンスは、クラスターの使用状況とノードごとのパフォーマンス レベルによって制限されます。

ブロック サイズと帯域幅は IOPS の数に直接影響します。ブロック サイズが大きくなるにつれて、システムはより大きなブロック サイズを処理するために必要なレベルまで帯域幅を増加させます。帯域幅が増加すると、システムが達成できる IOPS の数は減少します。QoS とパフォーマンスの詳細については、 "SolidFire のサービス品質"を参照してください。

SolidFire 認証

Element は、CHAP とボリューム アクセス グループ（VAG）の 2 つの認証方法をサポートしています。CHAP は、CHAP プロトコルを使用して、バックエンドに対してホストを認証します。ボリューム アクセス グループは、プロビジョニングするボリュームへのアクセスを制御します。NetApp は、認証には、よりシンプルでスケーリングの制限がない CHAP を使用することをお勧めします。

> 拡張 CSI プロビジョナーを使用する Trident では、 CHAP 認証の使用がサポートされます。VAG は、従来の非 CSI 動作モードでのみ使用してください。

CHAP 認証（イニシエーターが意図したボリューム ユーザーであることの検証）は、アカウント ベースのアクセス制御でのみサポートされます。認証に CHAP を使用する場合は、単方向 CHAP と双方向 CHAP の 2 つのオプションが利用できます。単方向 CHAP は、SolidFire アカウント名とイニシエーター シークレットを使用してボリューム アクセスを認証します。双方向 CHAP オプションは、ボリュームがアカウント名とイニ

シエーター シークレットを使用してホストを認証し、次にホストがアカウント名とターゲット シークレットを使用してボリュームを認証するため、ボリュームを認証する最も安全な方法を提供します。

ただし、CHAP を有効にできず、VAG が必要な場合は、アクセス グループを作成し、ホスト イニシエーターとボリュームをアクセス グループに追加します。アクセス グループに追加した各 IQN は、CHAP 認証の有無にかかわらず、グループ内の各ボリュームにアクセスできます。iSCSI イニシエーターが CHAP 認証を使用するように構成されている場合は、アカウントベースのアクセス制御が使用されます。iSCSI イニシエーターが CHAP 認証を使用するように構成されていない場合は、ボリューム アクセス グループのアクセス制御が使用されます。

さらに詳しい情報はどこで見つかりますか？

ベスト プラクティスのドキュメントの一部を以下に示します。 "NetApp ライブラリ"で最新バージョンを検索してください。

ONTAP

- "NFS のベストプラクティスと実装ガイド"
- "SAN の管理"（iSCSI の場合）
- "RHEL の iSCSI Express 構成"

Element ソフトウェア

- "Linux用SolidFireの設定"

NetApp HCI

- "NetApp HCI 導入の前提条件"
- "NetApp Deployment Engine にアクセスします"

アプリケーションのベストプラクティス情報

- "ONTAP上のMySQLのベストプラクティス"
- "SolidFire での MySQL のベストプラクティス"
- "NetApp SolidFire と Cassandra"
- "SolidFire での Oracle のベストプラクティス"
- "SolidFire での PostgreSQL のベストプラクティス"

すべてのアプリケーションに特定のガイドラインがあるわけではないため、NetApp チームと協力し、"NetApp ライブラリ"を使用して最新のドキュメントを見つけることが重要です。

Tridentを統合

Tridentを統合するには、次のような設計とアーキテクチャの要素を統合する必要があります：ドライバーの選択と展開、ストレージクラスの設計、仮想プールの設計、永続ボリュームクレーム（PVC）がストレージ プロビジョニングに与える影響、ボリューム操作、およびTridentを使用したOpenShiftサービスの展開。

ドライバの選択と導入

ストレージ システムのバックエンド ドライバを選択して導入します。

ONTAP バックエンドドライバ

ONTAP バックエンド ドライバは、使用するプロトコルとストレージ システムでのボリュームのプロビジョニング方法によって区別されます。そのため、導入するドライバを決定する際は慎重に検討してください。

より高いレベルでは、アプリケーションに共有ストレージを必要とするコンポーネント（同じ PVC にアクセスする複数のポッド）がある場合、NAS ベースのドライバーがデフォルトの選択肢になりますが、ブロックベースの iSCSI ドライバーは非共有ストレージのニーズを満たします。アプリケーションの要件と、ストレージおよびインフラストラクチャ チームの習熟度に基づいてプロトコルを選択します。一般的に言えば、ほとんどのアプリケーションではそれらの間にほとんど違いがないため、共有ストレージ（複数のポッドが同時にアクセスする必要がある場合）が必要かどうかに基づいて決定することがよくあります。

利用可能な ONTAP バックエンドドライバーは次のとおりです：

- `ontap-nas`：プロビジョニングされた各PVは完全なONTAP FlexVolumeです。
- `ontap-nas-economy`：プロビジョニングされた各PVはqtreeで、FlexVolumeあたりのqtree数は設定可能です（デフォルトは200）。
- `ontap-nas-flexgroup`：各PVは完全なONTAP FlexGroupとしてプロビジョニングされ、SVMに割り当てられたすべてのアグリゲートが使用されます。
- `ontap-san`：プロビジョニングされた各PVは、それ自身のFlexVolume内のLUNです。
- `ontap-san-economy`：プロビジョニングされた各PVはLUNであり、FlexVolumeあたりのLUN数は設定可能です（デフォルトは100）。

3 つの NAS ドライバーのいずれかを選択すると、アプリケーションで利用できる機能にいくつかの影響が生じます。

以下の表では、すべての機能がTridentを通じて公開されているわけではないことに注意してください。一部の機能は、その機能が必要な場合、プロビジョニング後にストレージ管理者が適用する必要があります。上付き脚注は、機能およびドライバーごとに機能を区別します。

ONTAP NASドライバー	**Snapshot**数	クローン	動的エクスポートポリシー	マルチアタッチ	**QoS**	サイズ変更	レプリケーション
`ontap-nas`	はい	はい	はい [5]	はい	はい [1]	はい	はい [1]
`ontap-nas-economy`	NOfootnote：3 []	NOfootnote：3 []	はい [5]	はい	NOfootnote：3 []	はい	NOfootnote：3 []
`ontap-nas-flexgroup`	はい [1]	いいえ	はい [5]	はい	はい [1]	はい	はい [1]

Tridentは、ONTAP用に2つのSANドライバを提供しており、その機能は以下のとおりです。

ONTAP SANドライバー	**Snapshot**数	クローン	マルチアタッチ	双方向**CHAP**	**QoS**	サイズ変更	レプリケーション
`ontap-san`	はい	はい	はい [4]	はい	はい [1]	はい	はい [1]

ONTAP SANドライバー	Snapshot数	クローン	マルチアタッチ	双方向CHAP	QoS	サイズ変更	レプリケーション
ontap-san-economy	はい	はい	はい [4]	はい	NOfootnote：3 []	はい	NOfootnote：3 []

> 上記の表の脚注：Yes [1]：Tridentで管理されていません Yes [2]：Tridentで管理されていますが、PV単位ではありません NO [3]：Tridentで管理されておらず、PV単位でもありません Yes [4]：rawブロックボリュームでサポートされています Yes [5]：Tridentでサポートされています

PV粒度ではない機能は、FlexVolume全体に適用され、すべてのPV（つまり、共有FlexVols内のqtreeまたはLUN）は共通のスケジュールを共有します。

上の表からわかるように、 ontap-nas`と `ontap-nas-economy`の機能の多くは同じです。ただし、`ontap-nas-economy`ドライバーはPV単位の粒度でスケジュールを制御する機能を制限するため、特にディザスタリカバリとバックアップ計画に影響を与える可能性があります。ONTAPストレージでPVCクローン機能を活用したい開発チームにとって、これは `ontap-nas、 `ontap-san`または `ontap-san-economy`ドライバーを使用する場合にのみ可能です。

`solidfire-san`ドライバは PVC のクローニングも可能です。

Cloud Volumes ONTAP バックエンドドライバ

Cloud Volumes ONTAPは、ファイル共有や、NASおよびSANプロトコル（NFS、SMB/CIFS、iSCSI）を提供するブロックレベルのストレージなど、さまざまなユースケースに対応するデータ管理とエンタープライズクラスのストレージ機能を提供します。Cloud Volume ONTAPと互換性のあるドライバーは ontap-nas、ontap-nas-economy、 ontap-san、 `ontap-san-economy`です。これらはCloud Volume ONTAP for AzureおよびCloud Volume ONTAP for GCPに適用されます。

Amazon FSx for ONTAP バックエンドドライバ

Amazon FSx for NetApp ONTAP では、使い慣れたNetAppの機能、パフォーマンス、管理機能を活用しながら、AWS にデータを保存することのシンプルさ、俊敏性、セキュリティ、スケーラビリティのメリットを享受できます。FSx for ONTAP は、多数の ONTAP ファイルシステム機能と管理 API をサポートしています。Cloud Volume ONTAP と互換性のあるドライバーは、 ontap-nas、 ontap-nas-economy、 ontap-nas-flexgroup、 ontap-san、 `ontap-san-economy`です。

NetApp HCI/SolidFire バックエンドドライバ

`solidfire-san`ドライバは、NetApp HCI/SolidFireプラットフォームで使用され、管理者がQoS制限に基づいてTrident用のElementバックエンドを設定するのに役立ちます。Tridentによってプロビジョニングされたボリュームに特定のQoS制限を設定するようにバックエンドを設計する場合は、バックエンドファイルの`type`パラメータを使用します。管理者は、`limitVolumeSize`パラメータを使用して、ストレージ上に作成できるボリュームサイズを制限することもできます。現在、ボリュームのサイズ変更やボリュームの複製などのElementストレージ機能は、 `solidfire-san`ドライバではサポートされていません。これらの操作は、Element Software Web UIを通じて手動で実行する必要があります。

SolidFireドライバ	Snapshot数	クローン	マルチアタッチ	CHAP	QoS	サイズ変更	レプリケーション
solidfire-san	はい	はい	はい [2]	はい	はい	はい	はい [1]

> 脚注：はい脚注：1[]：Tridentで管理されていません はい脚注：2[]：rawブロックボリュームでサポートされています

Azure NetApp Files バックエンドドライバ

Tridentは`azure-netapp-files`ドライバーを使用して"Azure NetApp Files"サービスを管理します。

このドライバの詳細と設定方法については、"Azure NetApp Files の Trident バックエンド構成"を参照してください。

Azure NetApp Filesドライバ	Snapshot数	クローン	マルチアタッチ	QoS	拡張	レプリケーション
azure-netapp-files	はい	はい	はい	はい	はい	はい [1]

> 脚注：はい脚注：1[]：Tridentで管理されていません

ストレージクラスの設計

Kubernetes ストレージクラスオブジェクトを作成するには、個々のストレージクラスを設定して適用する必要があります。このセクションでは、アプリケーションのストレージクラスを設計する方法について説明します。

特定のバックエンドの利用

特定のストレージ クラス オブジェクト内でフィルタリングを使用すると、その特定のストレージ クラスで使用されるストレージ プールまたはプール セットを決定できます。ストレージ クラスでは、次の 3 セットのフィルターを設定できます： `storagePools`、 `additionalStoragePools`、および/または `excludeStoragePools`。

`storagePools`パラメータは、指定された属性に一致するプールのセットにストレージを制限するのに役立ちます。 `additionalStoragePools`パラメータは、属性および`storagePools`パラメータによって選択されたプールのセットとともに、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するために使用されます。いずれかのパラメータを単独で使用することも、両方を組み合わせて使用することもでき、適切なストレージプールのセットが選択されるようにすることができます。

`excludeStoragePools`パラメータは、属性に一致するリストされたプールのセットを具体的に除外するために使用されます。

QoS ポリシーをエミュレートする

ストレージクラスを設計してサービス品質ポリシーをエミュレートする場合は、 `media`属性を `hdd`または`ssd`として設定したストレージクラスを作成します。ストレージクラスに記載されている `media`属性に基づいて、Tridentはメディア属性に一致する `hdd`または `ssd`アグリゲートを提供する適切なバックエンドを選択し、特定のアグリゲートへのボリュームのプロビジョニングを指示します。したがって、 `media`属性を`ssd`に設定したストレージクラスPREMIUMを作成でき、これはPREMIUM QoSポリシーとして分類できます。メディア属性を`hdd'に設定した別のストレージクラスSTANDARDを作成でき、これはSTANDARD QoSポリシーとして分類できます。また、ストレージクラスの``IOPS"属性を使用して、QoSポリシーとして定義できるElementアプライアンスにプロビジョニングをリダイレクトすることもできます。

特定の機能に基づいてバックエンドを利用する

ストレージ クラスは、シン プロビジョニング、シック プロビジョニング、スナップショット、クローン、暗号化などの機能が有効になっている特定のバックエンドでボリューム プロビジョニングを指示するように設計できます。使用するストレージを指定するには、必要な機能が有効になっている適切なバックエンドを指定するストレージ クラスを作成します。

仮想プール

仮想プールはすべての Trident バックエンドで利用できます。Trident が提供する任意のドライバを使用して、任意のバックエンドに仮想プールを定義できます。

仮想プールを使用すると、管理者はストレージ クラスを通じて参照できるバックエンドの抽象化レベルを作成できるため、バックエンドでのボリュームの配置の柔軟性と効率性が向上します。同じサービス クラスで異なるバックエンドを定義できます。さらに、同じバックエンド上に、異なる特性を持つ複数のストレージ プールを作成することもできます。ストレージ クラスが特定のラベルを持つセレクタで構成されている場合、Trident はボリュームを配置するために、すべてのセレクター ラベルに一致するバックエンドを選択します。ストレージ クラス セレクタ ラベルが複数のストレージ プールに一致する場合、Trident はそのうちの 1 つを選択してボリュームをプロビジョニングします。

仮想プールの設計

バックエンドを作成するときに、通常は一連のパラメータを指定できます。管理者が同じストレージ資格情報と異なるパラメータセットを持つ別のバックエンドを作成することは不可能でした。仮想プールの導入により、この問題は軽減されました。仮想プールは、バックエンドと Kubernetes ストレージ クラスの間に導入されたレベルの抽象化であり、管理者は、バックエンドに依存しない方法で、セレクターとして Kubernetes ストレージ クラスを通じて参照できるラベルとともにパラメータを定義できます。仮想プールは、Trident でサポートされているすべての NetApp バックエンドに対して定義できます。そのリストには、SolidFire/NetApp

HCI、ONTAP、および Azure NetApp Files が含まれます。

> 仮想プールを定義するときは、バックエンド定義内の既存の仮想プールの順序を変更しないことをお勧めします。既存の仮想プールの属性を編集/変更せず、代わりに新しい仮想プールを定義することもお勧めします。

異なるサービスレベル／**QoS**のエミュレーション

サービスクラスをエミュレートするための仮想プールを設計することが可能です。Cloud Volume Service for Azure NetApp Filesの仮想プール実装を使用して、さまざまなサービスクラスを設定する方法を見てみましょう。Azure NetApp Filesバックエンドを、異なるパフォーマンスレベルを表す複数のラベルで構成します。`servicelevel`アスペクトを適切なパフォーマンスレベルに設定し、各ラベルの下に他の必要なアスペクトを追加します。次に、異なる仮想プールにマッピングされる異なるKubernetesストレージクラスを作成します。`parameters.selector`フィールドを使用して、各StorageClassは、ボリュームをホストするために使用できる仮想プールを指定します。

特定のアスペクトセットの割り当て

単一のストレージ バックエンドから、特定の側面を持つ複数の仮想プールを設計できます。そのためには、バックエンドを複数のラベルで構成し、各ラベルの下に必要な側面を設定します。次に、`parameters.selector`フィールドを使用して、異なる仮想プールにマップされる異なるKubernetesストレージクラスを作成します。バックエンドでプロビジョニングされるボリュームには、選択した仮想プールで定義された側面が設定されます。

ストレージ プロビジョニングに影響する**PVC**の特性

要求されたストレージクラスを超える一部のパラメータが、PVC を作成する際の Trident プロビジョニング決定プロセスに影響を与える可能性があります。

アクセス モード

PVC 経由でストレージを要求する場合、必須フィールドの 1 つはアクセスモードです。希望するモードは、ストレージ要求をホストするために選択されたバックエンドに影響する可能性があります。

Tridentは、次のマトリックスに従って指定されたアクセス方法で使用されるストレージ プロトコルとの一致を試みます。これは、基盤となるストレージ プラットフォームとは独立しています。

	ReadWriteOnce	**ReadOnlyMany**	**ReadWriteMany**
iSCSI	はい	はい	はい（Rawブロック）
NFS	はい	はい	はい

NFS バックエンドが設定されていない Trident 環境に ReadWriteMany PVC の要求を送信すると、ボリュームはプロビジョニングされません。このため、要求者はアプリケーションに適したアクセス モードを使用する必要があります。

ボリューム操作

永続ボリュームを変更する

永続ボリュームは、2つの例外を除き、Kubernetes内の不変オブジェクトです。作成したら、再利用ポリシー

とサイズを変更できます。ただし、これによってボリュームの一部の側面がKubernetesの外部で変更されるのを防ぐことはできません。これは、特定のアプリケーション用にボリュームをカスタマイズしたり、容量が誤って消費されないようにしたり、あるいは何らかの理由でボリュームを別のストレージコントローラに移動したりする場合に望ましいことがあります。

> Kubernetes のツリー内プロビジョナーは、現時点では NFS、iSCSI、または FC PV のボリューム サイズ変更操作をサポートしていません。Trident は NFS、iSCSI、FC ボリュームの拡張をサポートします。

PV の接続詳細は作成後に変更できません。

オンデマンドボリューム**Snapshot**を作成

Tridentは、CSIフレームワークを使用したオンデマンドボリュームスナップショットの作成と、スナップショットからのPVCの作成をサポートしています。スナップショットは、ポイントインタイムデータのコピーを維持する便利な方法であり、KubernetesのソースPVから独立したライフサイクルを持ちます。これらのスナップショットを使用してPVCをクローニングできます。

スナップショットからボリュームを作成する

Tridentは、ボリュームスナップショットからのPersistentVolumesの作成もサポートしています。これを実現するには、PersistentVolumeClaimを作成し、 `datasource`をボリュームの作成元として必要なスナップショットとして指定します。Tridentは、スナップショットに存在するデータを含むボリュームを作成することで、このPVCを処理します。この機能により、リージョン間でのデータの複製、テスト環境の作成、破損または壊れた本番ボリューム全体の交換、特定のファイルやディレクトリの取得と別の接続されたボリュームへの転送が可能になります。

クラスタ内のボリュームを移動する

ストレージ管理者は、ストレージコンシューマに影響を与えることなく、ONTAPクラスタ内のアグリゲートとコントローラ間でボリュームを無停止で移動できます。この処理は、TridentまたはKubernetesクラスタに影響しません。ただし、デスティネーションアグリゲートが、Tridentで使用しているSVMがアクセスできるものである必要があります。重要な点として、アグリゲートがSVMに新しく追加された場合は、Tridentに再度追加してバックエンドを更新する必要があります。これにより、Tridentが新しいアグリゲートを認識できるようにSVMの再インベントリが実行されます。

ただし、バックエンド間でのボリュームの移動は、Tridentでは自動的にサポートされていません。これには、同じクラスタ内のSVM間、クラスタ間、または異なるストレージプラットフォームへの移動が含まれます（そのストレージシステムがTridentに接続されている場合でも）。

ボリュームが別の場所にコピーされた場合、ボリュームインポート機能を使用して現在のボリュームをTridentにインポートできます。

ボリュームを拡張する

Tridentは、NFS、iSCSI、FCのPVのリサイズをサポートしています。これにより、ユーザーはKubernetesレイヤーを通じてボリュームを直接リサイズできます。ボリューム拡張は、ONTAPを含むすべての主要なNetAppストレージプラットフォームおよびSolidFire/NetApp HCIバックエンドで可能です。後で拡張できるようにするには、ボリュームに関連付けられたStorageClassで `allowVolumeExpansion`を `true`に設定してください。永続ボリュームのリサイズが必要な場合は、Persistent Volume Claimの `spec.resources.requests.storage`アノテーションを必要なボリュームサイズに編集します。Tridentはストレージクラスタ上で自動的にボリュームのリサイズを行います。

既存のボリュームを**Kubernetes**にインポートする

ボリュームインポートでは、既存のストレージボリュームをKubernetes環境にインポートできます。これは現在、 `ontap-nas`、 `ontap-nas-flexgroup`、 `solidfire-san`、および `azure-netapp-files`ドライバでサポートされています。この機能は、既存のアプリケーションをKubernetesに移植する場合や、ディザスタリカバリシナリオで役立ちます。

ONTAPおよび `solidfire-san`ドライバーを使用する場合は、コマンド `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml`を使用して、既存のボリュームをKubernetesにインポートし、Tridentで管理します。インポートボリュームコマンドで使用されるPVC YAMLまたはJSONファイルは、Tridentをプロビジョナーとして識別するストレージクラスを指します。NetApp HCI/SolidFireバックエンドを使用する場合は、ボリューム名が一意であることを確認してください。ボリューム名が重複している場合は、ボリュームインポート機能で区別できるように、ボリュームを一意の名前でクローニングします。

> `azure-netapp-files`ドライバを使用する場合は、コマンド `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml`
> を使用して、ボリュームを`Kubernetes`にインポートし、`Trident`で管理します。これにより、一意のボリューム参照が保証されます。

上記のコマンドを実行すると、Tridentはバックエンドでボリュームを見つけて、そのサイズを読み取ります。設定されたPVCのボリュームサイズが自動的に追加され（必要に応じて上書きされます）。Tridentは新しいPVを作成し、KubernetesはPVCをPVにバインドします。

特定のインポートされた PVC を必要とするコンテナがデプロイされた場合、PVC/PV ペアがボリューム インポート プロセスによってバインドされるまで、コンテナは保留状態のままになります。PVC/PV ペアがバインドされた後、他の問題がなければコンテナが起動するはずです。

レジストリサービス

レジストリのストレージの導入と管理については、"netapp.io"の"ブログ"に記載されています。

ログサービス

他のOpenShiftサービスと同様に、ログサービスは、プレイブックに提供されるインベントリファイル（ホスト）によって指定される構成パラメータを使用して、Ansibleを使用してデプロイされます。ここでは、2つのインストール方法について説明します：OpenShiftの初期インストール時にロギングをデプロイする方法と、OpenShiftのインストール後にロギングをデプロイする方法です。

> Red Hat OpenShiftバージョン3.9以降、公式ドキュメントでは、データ破損に関する懸念から、ロギングサービスにNFSを使用しないことを推奨しています。これはRed Hatによる自社製品のテストに基づいています。ONTAP NFSサーバにはこれらの問題はなく、ロギング環境を簡単にバックアップできます。最終的には、ロギングサービスのプロトコルの選択はお客様次第ですが、NetAppプラットフォームを使用する場合、どちらも問題なく動作することを知っておいてください。NFSがお好みであれば、NFSを避ける理由はありません。

ログサービスでNFSを使用する場合は、Ansible変数 `openshift_enable_unsupported_configurations`を `true`に設定して、インストーラーが失敗するのを防ぐ必要があります。

ログサービスは、オプションで、アプリケーションとOpenShiftクラスタ自体のコア運用の両方に導入できます。変数 `openshift_logging_use_ops`を `true`として指定して運用ログを導入することを選択した場合、サービスのインスタンスが2つ作成されます。運用のログインスタンスを制御する変数には「ops」が含まれますが、アプリケーションのインスタンスには含まれません。

基盤となるサービスによって正しいストレージが確実に利用されるようにするには、デプロイメント方法に応じて Ansible 変数を設定することが重要です。それぞれのデプロイメント方法のオプションを見てみましょう。

以下の表には、ログ サービスに関連するストレージ構成に関係する変数のみが含まれています。その他のオプションについては"Red Hat OpenShiftログ記録ドキュメント"を参照してください。これらは、導入に応じて確認、設定、および使用する必要があります。

以下の表の変数により、Ansible プレイブックは提供された詳細を使用して、ログ サービス用の PV と PVC を作成します。この方法は、OpenShift のインストール後にコンポーネントインストールプレイブックを使用するよりも柔軟性が大幅に低下しますが、既存のボリュームが利用可能な場合は、オプションになります。

変数	詳細
`openshift_logging_storage_kind`	`nfs`に設定すると、インストーラーでログ サービス用のNFS PVを作成します。
`openshift_logging_storage_host`	NFSホストのホスト名またはIPアドレス。これは仮想マシンのdataLIFに設定する必要があります。
`openshift_logging_storage_nfs_directory`	NFSエクスポートのマウントパス。たとえば、ボリュームが `/openshift_logging`のようにジャンクションされている場合は、この変数にそのパスを使用します。
`openshift_logging_storage_volume_name`	作成するPVの名前（例： `pv_ose_logs`）。
`openshift_logging_storage_volume_size`	NFSエクスポートのサイズ（例： `100Gi`）。

OpenShiftクラスタがすでに実行されており、Tridentが導入および設定されている場合、インストーラは動的プロビジョニングを使用してボリュームを作成できます。次の変数を設定する必要があります。

変数	詳細
`openshift_logging_es_pvc_dynamic`	動的にプロビジョニングされたボリュームを使用するには、trueに設定します。
`openshift_logging_es_pvc_storage_class_name`	PVC で使用されるストレージ クラスの名前。
`openshift_logging_es_pvc_size`	PVC で要求されたボリュームのサイズ。
`openshift_logging_es_pvc_prefix`	ロギング サービスで使用される PVC のプレフィックス。
`openshift_logging_es_ops_pvc_dynamic`	`true`に設定すると、オペレーションログインスタンスに動的にプロビジョニングされたボリュームを使用します。
`openshift_logging_es_ops_pvc_storage_class_name`	オペレーションログインスタンスのストレージクラスの名前。

変数	詳細
`openshift_logging_es_ops_pvc_size`	ops インスタンスのボリューム要求のサイズ。
`openshift_logging_es_ops_pvc_prefix`	ops インスタンス PVC のプレフィックス。

ログスタックをデプロイする

初期OpenShiftインストール プロセスの一部としてログ記録を導入する場合は、標準の導入プロセスに従うだけで済みます。Ansibleは必要なサービスとOpenShiftオブジェクトを設定して導入するため、Ansibleが完了するとすぐにサービスが利用できるようになります。

ただし、初期インストール後にデプロイする場合は、Ansible でコンポーネント プレイブックを使用する必要があります。このプロセスは、OpenShift のバージョンによって若干異なる場合があるため、お使いのバージョンの"Red Hat OpenShift Container Platform 3.11ドキュメント"を必ずお読みになり、従ってください。

メトリクスサービス

メトリクスサービスは、管理者にOpenShiftクラスタのステータス、リソース使用率、可用性に関する貴重な情報を提供します。また、ポッドの自動スケール機能にも必要であり、多くの組織はメトリクスサービスからのデータをチャージバックやショーバックアプリケーションに使用しています。

ログサービスと同様に、OpenShift全体として、Ansibleはメトリクスサービスのデプロイに使用されます。また、ログサービスと同様に、メトリクスサービスは、クラスタの初期セットアップ時、またはコンポーネントインストール方法を使用して運用開始後にデプロイできます。次の表には、メトリクスサービスの永続ストレージを構成するときに重要な変数が含まれています。

以下の表には、メトリック サービスに関連するストレージ構成に関係する変数のみが含まれています。ドキュメントには他にも多くのオプションが記載されており、導入に応じて確認、設定、使用する必要があります。

変数	詳細
`openshift_metrics_storage_kind`	`nfs`に設定すると、インストーラーでログ サービス用のNFS PVを作成します。
`openshift_metrics_storage_host`	NFSホストのホスト名またはIPアドレス。これは、SVMのdataLIFに設定する必要があります。
`openshift_metrics_storage_nfs_directory`	NFSエクスポートのマウントパス。たとえば、ボリュームが `/openshift_metrics`のようにジャンクションされている場合は、この変数にそのパスを使用します。
`openshift_metrics_storage_volume_name`	作成するPVの名前（例： `pv_ose_metrics`）。
`openshift_metrics_storage_volume_size`	NFSエクスポートのサイズ（例： `100Gi`）。

OpenShiftクラスタがすでに実行されており、Tridentが導入および設定されている場合、インストーラは動的プロビジョニングを使用してボリュームを作成できます。次の変数を設定する必要があります。

変数	詳細
`openshift_metrics_cassandra_pvc_prefix`	メトリック PVC に使用するプレフィックス。
`openshift_metrics_cassandra_pvc_size`	要求するボリュームのサイズ。

変数	詳細
`openshift_metrics_cassandra_storage_type`	メトリックに使用するストレージのタイプ。Ansibleが適切なストレージクラスを持つPVCを作成するには、これをdynamicに設定する必要があります。
`openshift_metrics_cassanda_pvc_storage_class_name`	使用するストレージ クラスの名前。

メトリクスサービスを導入する

hosts/inventory ファイルに適切な Ansible 変数を定義して、Ansible を使用してサービスをデプロイします。OpenShift のインストール時にデプロイする場合は、PV が自動的に作成されて使用されます。OpenShift のインストール後にコンポーネントプレイブックを使用してデプロイする場合は、Ansible によって必要な PVC が作成され、Trident によってストレージがプロビジョニングされた後、サービスがデプロイされます。

上記の変数と導入プロセスは、OpenShiftのバージョンごとに変更される可能性があります。お使いの環境に合わせて構成されるように、バージョンに応じた"Red HatのOpenShift導入ガイド"を必ず確認して従ってください。

データ保護とディザスタ リカバリ

Tridentを使用して作成されたTridentおよびボリュームの保護とリカバリのオプションについて説明します。永続性を必要とするアプリケーションごとに、データ保護およびリカバリ戦略を用意する必要があります。

Tridentのレプリケーションとリカバリ

災害が発生した場合に備えて、Tridentを復元するためのバックアップを作成できます。

Tridentレプリケーション

Trident は Kubernetes CRD を使用して独自の状態を保存および管理し、Kubernetes クラスター etcd を使用してメタデータを保存します。

手順

1. "Kubernetes：etcdクラスタのバックアップ"を使用してKubernetesクラスタetcdをバックアップします。
2. バックアップアーティファクトをFlexVolボリュームに配置する

> NetAppでは、FlexVolが存在するSVMをSnapMirror関係で別のSVMに保護することを推奨しています。

Tridentリカバリ

Kubernetes CRDとKubernetesクラスタetcdスナップショットを使用すると、Tridentをリカバリできます。

手順

1. デスティネーション SVM から、Kubernetes etcd データファイルと証明書を含むボリュームを、マスターノードとしてセットアップされるホストにマウントします。

2. Kubernetesクラスタに関連する必要な証明書をすべて `/etc/kubernetes/pki`にコピーし、etcdメンバーファイルを `/var/lib/etcd`にコピーします。
3. "Kubernetes：etcdクラスタのリストア"を使用して、etcdバックアップからKubernetesクラスタを復元します。
4. 実行 `kubectl get crd`すべてのTridentカスタムリソースが起動したことを確認し、Tridentオブジェクトを取得してすべてのデータが利用可能であることを確認します。

SVMのレプリケーションとリカバリ

Tridentはレプリケーション関係を構成することはできませんが、ストレージ管理者は "ONTAP SnapMirror"を使用してSVMをレプリケートできます。

災害が発生した場合は、SnapMirrorデスティネーションSVMをアクティブ化してデータの提供を開始できます。システムが復元されたら、プライマリに切り替えることができます。

タスク概要

SnapMirror SVM レプリケーション機能を使用する際は、以下の点にご留意ください：

- SVM-DR が有効になっている各 SVM ごとに個別のバックエンドを作成する必要があります。
- レプリケーションを必要としないボリュームがSVM-DRをサポートするバックエンドにプロビジョニングされることを回避するために、必要な場合にのみレプリケートされたバックエンドを選択するようにストレージクラスを構成します。
- アプリケーション管理者は、レプリケーションに関連する追加コストと複雑さを理解し、このプロセスを開始する前にリカバリ計画を慎重に検討する必要があります。

SVMレプリケーション

"ONTAP：SnapMirror SVMレプリケーション"を使用してSVMレプリケーション関係を作成できます。

SnapMirrorでは、複製する内容を制御するオプションを設定できます。Tridentを使用したSVMリカバリを実行する際に選択したオプションを把握しておく必要があります。

- "-identity-preserve true"SVM設定全体をレプリケートします。
- "-discard-configs network"LIFおよび関連するネットワーク設定は除外されます。
- "-identity-preserve false"ボリュームとセキュリティ設定のみをレプリケートします。

Tridentを使用したSVMリカバリ

TridentはSVMの障害を自動的に検出しません。災害が発生した場合、管理者は手動でTridentフェイルオーバーを新しいSVMに開始できます。

手順

1. スケジュール済みおよび実行中のSnapMirror転送をキャンセルし、レプリケーション関係を解除し、ソースSVMを停止してから、SnapMirrorデスティネーションSVMをアクティブ化します。
2. SVMレプリケーションの設定時に `-identity-preserve false`または `-discard-config network`を指定した場合は、Tridentバックエンド定義ファイル内の `managementLIF`および `dataLIF`を更新します。
3. `storagePrefix`がTridentバックエンド定義ファイルに存在することを確認します。このパラメータは変更

できません。 `storagePrefix`を省略すると、バックエンドの更新が失敗します。

4. 次のコマンドを使用して、新しいデスティネーションSVM名を反映するように必要なすべてのバックエンドを更新します：

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. `-identity-preserve false`または `discard-config network`を指定した場合は、すべてのアプリケーションポッドをバウンスする必要があります。

> `-identity-preserve true`を指定した場合、Tridentによってプロビジョニングされたすべてのボリュームは、デスティネーションSVMがアクティブ化されたときにデータの提供を開始します。

ボリュームのレプリケーションとリカバリ

TridentではSnapMirrorレプリケーション関係を設定できませんが、ストレージ管理者は"ONTAP SnapMirrorレプリケーションとリカバリ"を使用してTridentで作成されたボリュームをレプリケートできます。

その後、復元したボリュームを"tridentctl volume import"を使用してTridentにインポートできます。

> `ontap-nas-economy`、 `ontap-san-economy`、または `ontap-flexgroup-economy`ドライバではインポートはサポートされていません。

Snapshotデータ保護

以下を使用してデータを保護および復元できます：

- 永続ボリューム（PV）のKubernetesボリュームスナップショットを作成するための外部スナップショットコントローラーとCRD。

  "ボリュームスナップショット"

- ONTAPスナップショットを使用して、ボリュームの内容全体を復元したり、個々のファイルまたはLUNをリカバリしたりできます。

  "ONTAPスナップショット"

Tridentを使用したステートフルアプリケーションのフェイルオーバーの自動化

Tridentの強制デタッチ機能を使用すると、Kubernetesクラスター内の正常でないノードからボリュームを自動的にデタッチして、データ破損を防ぎ、アプリケーションの可用性を確保できます。この機能は、ノードが応答しなくなったり、メンテナンスのためにオフラインになったりするシナリオで特に役立ちます。

強制デタッチの詳細

強制デタッチは `ontap-san`、 `ontap-san-economy`、 `ontap-nas`、 `ontap-nas-economy`でのみ使用できます。強制デタッチを有効にする前に、Kubernetesクラスタで非正常なノードシャットダウン（NGNS）を有効にする必要があります。NGNSはKubernetes 1.28以降ではデフォルトで有効になっています。詳細については、"Kubernetes：非正常なノードシャットダウン"を参照してください。

> `ontap-nas`または `ontap-nas-economy`ドライバを使用する場合は、バックエンド構成の `autoExportPolicy`パラメータを `true`に設定して、Tridentが管理対象のエクスポートポリシーを使用してテイントが適用されたKubernetesノードからのアクセスを制限できるようにする必要があります。

> Tridentは Kubernetes NGNS に依存しているため、すべての許容できないワークロードが再スケジュールされるまで、異常なノードから `out-of-service`テイントを削除しないでください。テイントを無謀に適用または削除すると、バックエンドのデータ保護が危険にさらされる可能性があります。

Kubernetesクラスタ管理者がノードに `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute`テイントを適用し、 `enableForceDetach`が `true`に設定されている場合、Tridentはノードのステータスを判断し、次の処理を実行します：

1. そのノードにマウントされたボリュームのバックエンドI/Oアクセスを停止します。
2. Tridentノードオブジェクトを `dirty`（新規パブリケーションには安全ではない）としてマークします。

> Tridentコントローラは、Tridentノードポッドによってノードが再認定されるまで（ `dirty`とマークされた後）、新しいボリューム公開要求を拒否します。マウントされたPVCでスケジュールされたワークロード（クラスタノードが正常で準備完了した後でも）は、Tridentがノード `clean`（新しい公開に対して安全）を検証できるまで受け入れられません。

ノードの健全性が回復し、汚染が除去されると、Tridentは次の処理を実行します：

1. ノード上の古くなった公開パスを識別してクリーンアップします。
2. ノードが `cleanable`状態（サービス停止の汚染が除去され、ノードが `Ready`状態）で、すべての古い公開済みパスがクリーンである場合、Tridentはノードを `clean`として再度承認し、新しい公開ボリュームをノードに許可します。

自動フェイルオーバーの詳細

"node health check（NHC）operator"との統合により、強制デタッチプロセスを自動化できます。ノード障害が発生すると、NHCはTridentの名前空間で障害が発生したノードを定義するTridentNodeRemediation CRを作成することで、Tridentノード修復（TNR）をトリガーし、強制デタッチを自動的に実行します。TNRはノード障害時にのみ作成され、ノードがオンラインに戻るかノードが削除されるとNHCによって削除されます。

障害ノードのポッド削除プロセス

自動フェイルオーバーは、障害が発生したノードから削除するワークロードを選択します。TNRが作成されると、TNRコントローラーはノードをダーティとしてマークし、新しいボリュームの公開を防止し、強制デタッチがサポートされているポッドとそのボリュームアタッチメントの削除を開始します。

強制デタッチでサポートされているすべてのボリューム/PVCは、自動フェイルオーバーでもサポートされます：

- 自動エクスポートポリシーを使用するNASおよびNASエコノミーボリューム（SMBはまだサポートされていません）。
- SAN および SAN-economy ボリューム。

[強制デタッチの詳細]を参照してください。

デフォルトの動作：

- 強制デタッチがサポートされているボリュームを使用しているポッドは、障害が発生したノードから削除されます。Kubernetesはこれらを正常なノードに再スケジュールします。
- 強制デタッチでサポートされていないボリューム（Trident以外のボリュームを含む）を使用するポッドは、障害が発生したノードから削除されません。
- ステートレスポッド（PVCではない）は、ポッドアノテーション `trident.netapp.io/podRemediationPolicy: delete`が設定されていない限り、障害が発生したノードから削除されません。

ポッド削除動作のオーバーライド：

ポッドの削除動作は、ポッド アノテーションを使用してカスタマイズできます `trident.netapp.io/podRemediationPolicy[retain, delete]`。これらのアノテーションは、フェイルオーバーが発生したときに検査され、使用されます。フェイルオーバー後にアノテーションが消えないように、Kubernetesデプロイメント/レプリカセット ポッド仕様にアノテーションを適用します：

- `retain` - 自動フェイルオーバー中に、障害が発生したノードからPodは削除されません。
- `delete` - 自動フェイルオーバー中に、障害が発生したノードからPodが削除されます。

これらの注釈はどのポッドにも適用できます。

- 強制デタッチをサポートするボリュームの場合、障害が発生したノードでのみI/O操作がブロックされます。
- 強制デタッチをサポートしていないボリュームの場合、データ破損およびマルチアタッチの問題が発生するリスクがあります。

TridentNodeRemediation CR

TridentNodeRemediation（TNR）CRは障害が発生したノードを定義します。TNRの名前は、障害が発生したノードの名前です。

TNRの例：

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR の状態：TNR のステータスを表示するには、次のコマンドを使用します。
`kubectl get tnr <name> -n <trident-namespace>`

TNR は次のいずれかの状態になります：

- 修復中：
  - そのノードにマウントされた、強制デタッチでサポートされているボリュームへのバックエンドI/Oアクセスを停止します。
  - Tridentノードオブジェクトはダーティとしてマークされています（新しいパブリケーションには安全ではありません）。
  - ノードからポッドとボリュームアタッチメントを削除します
- *NodeRecoveryPending*：
  - コントローラはノードがオンラインに戻るのを待機しています。
  - ノードがオンラインになると、パブリッシュ強制により、ノードがクリーンであり、新しいボリュームのパブリケーションの準備ができていることが確認されます。
- ノードが K8s から削除されると、TNR コントローラーは TNR を削除し、調整を停止します。
- 成功：
  - すべての修復およびノード回復手順が正常に完了しました。ノードはクリーンであり、新しいボリュームの公開の準備ができています。
- *Failed*：
  - 回復不能なエラーです。エラー理由は、CRのstatus.messageフィールドに設定されます。

自動フェイルオーバーの有効化

前提条件：

- 自動フェイルオーバーを有効にする前に、強制デタッチが有効になっていることを確認してください。詳細については、[強制デタッチの詳細]を参照してください。
- Kubernetesクラスタにノードヘルスチェック（NHC）をインストールします。
  - "operator-sdkをインストールする"。
  - まだインストールされていない場合は、クラスタに Operator Lifecycle Manager（OLM）をインストールします `operator-sdk olm install`
  - Node Health check Operatorをインストールします `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`。

> 以下の[Integrating Custom Node Health Check Solutions]セクションで指定されている別の方法を使用して、ノード障害を検出することもできます。

詳細については、"ノードヘルスチェックオペレーター"を参照してください。

手順

1. Trident ネームスペース内に NodeHealthCheck（NHC）CR を作成して、クラスター内のワーカーノードを監視します。例：

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. `trident`ネームスペースにノードヘルスチェックCRを適用します。

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

上記の CR は、K8s ワーカーノードのノード状態 Ready：false および Unknown を監視するように構成されています。自動フェイルオーバーは、ノードが Ready：false または Ready：Unknown 状態になるとトリガーされます。

CR の `unhealthyConditions`では 0 秒の猶予期間が使用されます。これにより、K8s がノードからのハートビートを失った後に設定されるノード条件 Ready：false を K8s が設定すると、自動フェイルオーバーが直ちにトリガーされます。K8s では、最後のハートビートの後に Ready：false を設定するまで、デフォルトで 40 秒間待機します。この猶予期間は、K8s デプロイメント オプションでカスタマイズできます。

追加の設定オプションについては、"Node-Healthcheck-Operator ドキュメント"を参照してください。

追加のセットアップ情報

Tridentを強制デタッチを有効にしてインストールすると、NHCとの統合を容易にするために、Trident名前空間に2つの追加リソースが自動的に作成されます：TridentNodeRemediationTemplate（TNRT）とClusterRole。

TridentNodeRemediationTemplate（TNRT）：

TNRT は NHC コントローラのテンプレートとして機能し、必要に応じて TNR リソースを生成します。

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole：

強制デタッチが有効になっている場合、インストール中にクラスター ロールも追加されます。これにより、NHCにTrident名前空間のTNRに対する権限が付与されます。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

K8s クラスタのアップグレードとメンテナンス

フェイルオーバーを防ぐには、ノードがダウンしたり再起動することが予想される K8s メンテナンスまたはアップグレード中は、自動フェイルオーバーを一時停止します。NHC CR（上記で説明）を一時停止するには、その CR にパッチを適用します。

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

これにより、自動フェイルオーバーが一時停止されます。自動フェイルオーバーを再度有効にするには、メンテナンス完了後に仕様からpauseRequestsを削除します。

制限事項

- 強制デタッチでサポートされているボリュームの場合、障害が発生したノード上でのみ I/O 操作が防止されます。強制デタッチでサポートされているボリューム/PVC を使用しているポッドのみが自動的に削除されます。
- 自動フェイルオーバーと強制デタッチは、trident-controller ポッド内で実行されます。trident-controller をホストしているノードに障害が発生した場合、K8s がポッドを正常なノードに移動するまで、自動フェイルオーバーは遅延されます。

カスタムノードヘルスチェックソリューションの統合

自動フェイルオーバーをトリガーするために、Node Healthcheck Operator を代替のノード障害検出ツールに置き換えることができます。自動フェイルオーバーメカニズムとの互換性を確保するために、カスタムソリューションは次の要件を満たす必要があります：

- ノード障害が検出されると、障害が発生したノードの名前を TNR CR 名として使用して TNR を作成します。
- ノードが回復し、TNR が Succeeded 状態になったら、TNR を削除します。

セキュリティ

セキュリティ

ここに記載されている推奨事項を使用して、Tridentのインストールが安全であることを確認してください。

Trident を独自の名前空間で実行する

アプリケーション、アプリケーション管理者、ユーザー、および管理アプリケーションがTridentオブジェクト定義やポッドにアクセスできないようにすることが重要です。これにより、信頼性の高いストレージを確保し、潜在的な悪意のあるアクティビティをブロックできます。

他のアプリケーションやユーザーを Trident から分離するには、常に Trident を独自の Kubernetes 名前空間にインストールしてください(`trident`）。Trident を独自の名前空間に配置することで、Kubernetes 管理担当者のみが Trident ポッドおよび名前空間 CRD オブジェクトに保存されている成果物（該当する場合はバックエンドや CHAP シークレットなど）にアクセスできるようになります。管理者のみが Trident 名前空間にアクセスでき、したがって `tridentctl`アプリケーションにアクセスできるようにする必要があります。

ONTAP SANバックエンドで**CHAP**認証を使用する

Tridentは、ONTAP SANワークロード（ `ontap-san`および `ontap-san-economy`ドライバーを使用）のCHAPベースの認証をサポートしています。NetAppでは、ホストとストレージバックエンド間の認証にTridentで双方向CHAPを使用することを推奨しています。

SAN ストレージドライバーを使用する ONTAP バックエンドの場合、 Trident は双方向 CHAP を設定し、 CHAP ユーザー名とシークレットを `tridentctl`で管理できます。"ONTAP SANドライバを使用してバックエンドを設定する準備をします"を参照して、 Trident が ONTAP バックエンドで CHAP を設定する方法を理解し

てください。

NetApp HCI および **SolidFire** バックエンドで **CHAP** 認証を使用する

NetAppは、ホストとNetApp HCIおよびSolidFireバックエンド間の認証を確実にするために、双方向CHAPを導入することを推奨します。Tridentは、テナントごとに2つのCHAPパスワードを含むシークレットオブジェクトを使用します。Tridentがインストールされると、CHAPシークレットを管理し、それぞれのPVに対応する `tridentvolume`CRオブジェクトに保存します。PVを作成すると、TridentはCHAPシークレットを使用してiSCSIセッションを開始し、CHAP経由でNetApp HCIおよびSolidFireシステムと通信します。

Trident によって作成されたボリュームは、どのボリューム アクセス グループにも関連付けられていません。

Tridentと**NVE**および**NAE**を使用

NetApp ONTAP は、ディスクが盗難、返却、または再利用された場合に機密データを保護するために、保存データの暗号化を提供します。詳細については、"NetApp Volume Encryptionの設定 - 概要"を参照してください。

- NAEがバックエンドで有効になっている場合、Tridentでプロビジョニングされたボリュームは、NAE対応になります。
  - NVE 暗号化フラグを `""`に設定すると、NAE 対応ボリュームを作成できます。
- NAEがバックエンドで有効になっていない場合、バックエンド構成でNVE暗号化フラグが `false`（デフォルト値）に設定されていない限り、TridentでプロビジョニングされたボリュームはすべてNVEが有効になります。

NAE 対応のバックエンドで Trident に作成されたボリュームは、NVE または NAE で暗号化されている必要があります。

- Trident バックエンド構成で NVE 暗号化フラグを `true`に設定すると、NAE 暗号化を上書きして、ボリュームごとに特定の暗号化キーを使用できます。
- NAE対応のバックエンドでNVE暗号化フラグを `false`に設定すると、NAE対応のボリュームが作成されます。NVE暗号化フラグを `false`に設定してもNAE暗号化を無効にすることはできません。

- Trident で NVE ボリュームを手動で作成するには、NVE 暗号化フラグを明示的に `true`に設定します。

バックエンド構成オプションの詳細については、以下を参照してください：

- "ONTAP SAN 構成オプション"
- "ONTAP NAS 構成オプション"

Linux Unified Key Setup（LUKS）

Linux Unified Key Setup（LUKS）を有効にして、Trident上のONTAP SANおよびONTAP SAN ECONOMYボリュームを暗号化できます。Tridentは、LUKSで暗号化されたボリュームのパスフレーズローテーションとボリューム拡張をサポートします。

Tridentでは、LUKSで暗号化されたボリュームは、"NIST"で推奨されているaes-xts-plain64暗号とモードを使

用します。

LUKS 暗号化は ASA r2 システムではサポートされていません。ASA r2システムの詳細については、"ASA r2ストレージシステムの詳細"を参照してください。

開始する前に

- ワーカー ノードには、cryptsetup 2.1 以上（3.0 未満）がインストールされている必要があります。詳細については、"Gitlab：cryptsetup"を参照してください。
- パフォーマンス上の理由から、NetAppではワーカーノードで Advanced Encryption Standard New Instructions（AES-NI）をサポートすることを推奨します。AES-NI のサポートを確認するには、次のコマンドを実行します：

```
grep "aes" /proc/cpuinfo
```

何も返されない場合、プロセッサはAES-NIをサポートしていません。AES-NIの詳細については、以下をご覧ください："Intel：Advanced Encryption Standard Instructions（AES-NI）"。

LUKS暗号化を有効にする

Linux Unified Key Setup（LUKS）を使用して、ONTAP SANおよびONTAP SAN ECONOMYボリュームのボリュームごとにホスト側の暗号化を有効にすることができます。

手順

1. バックエンド構成で LUKS 暗号化属性を定義します。ONTAP SAN のバックエンド設定オプションの詳細については、"ONTAP SAN 構成オプション"を参照してください。

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. `parameters.selector`を使用して、LUKS暗号化を使用するストレージプールを定義します。例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. LUKS パスフレーズを含むシークレットを作成します。例：

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

制限事項

LUKSで暗号化されたボリュームは、ONTAPの重複排除と圧縮を利用できません。

LUKSボリュームをインポートするためのバックエンド構成

LUKSボリュームをインポートするには、バックエンドで `luksEncryption`を（`true`に設定する必要があります。 `luksEncryption`オプションは、次の例に示すように、ボリュームがLUKS準拠（（`true`）かLUKS非準拠（(false）かをTridentに通知します。

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

LUKSボリュームをインポートするための**PVC**構成

LUKS ボリュームを動的にインポートするには、アノテーション `trident.netapp.io/luksEncryption`を `true`に設定し、この例に示すように、PVC に LUKS 対応ストレージクラスを含めます。

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

LUKSパスフレーズをローテーションする

LUKS パスフレーズをローテーションし、ローテーションを確認できます。

> パスフレーズがボリューム、スナップショット、またはシークレットによって参照されなくなったことを確認するまで、パスフレーズを忘れないでください。参照されているパスフレーズが失われた場合、ボリュームをマウントできなくなり、データは暗号化されたままアクセスできなくなる可能性があります。

タスク概要

新しい LUKS パスフレーズが指定された後にボリュームをマウントするポッドが作成されると、LUKS パスフレーズのローテーションが発生します。新しいポッドが作成されると、Trident はボリューム上の LUKS パスフレーズをシークレット内のアクティブなパスフレーズと比較します。

- ボリューム上のパスフレーズがシークレット内のアクティブなパスフレーズと一致しない場合は、ローテーションが発生します。
- ボリューム上のパスフレーズがシークレット内のアクティブなパスフレーズと一致する場合、 `previous-luks-passphrase` パラメータは無視されます。

手順

1. `node-publish-secret-name`と `node-publish-secret-namespace` StorageClassパラメータを追加します。例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. ボリュームまたはスナップショット上の既存のパスフレーズを特定します。

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. ボリュームの LUKS シークレットを更新して、新しいパスフレーズと以前のパスフレーズを指定します。`previous-luke-passphrase-name`と `previous-luks-passphrase`が以前のパスフレーズと一致することを確認します。

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. ボリュームをマウントする新しいポッドを作成します。これはローテーションを開始するために必要です。

5. パスフレーズがローテーションされたことを確認します。

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

結果

ボリュームとスナップショットで新しいパスフレーズのみが返されたときに、パスフレーズがローテーションされました。

> たとえば、2つのパスフレーズが返された場合 `luksPassphraseNames: ["B", "A"]`、ローテーションは不完全です。新しいポッドをトリガーして、ローテーションを完了させることができます。

ボリューム拡張を有効にする

LUKS で暗号化されたボリュームでストレージ拡張を有効にすることができます。

手順

1. `CSINodeExpandSecret`機能ゲートを有効にします（ベータ版 1.25 以降）。詳細については、"Kubernetes 1.25：CSI ボリュームのノード駆動型拡張に Secret を使用する"を参照してください。
2. `node-expand-secret-name`と `node-expand-secret-namespace` StorageClassパラメータを追加します。例：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

結果

オンラインストレージ拡張を開始すると、kubelet は適切な資格情報をドライバーに渡します。

Kerberos のインフライト暗号化

Kerberos インフライト暗号化を使用すると、マネージドクラスタとストレージバックエンド間のトラフィックの暗号化を有効にして、データアクセスのセキュリティを向上させることができます。

Trident は、ストレージバックエンドとして ONTAP の Kerberos 暗号化をサポートしています：

- オンプレミス **ONTAP** - Trident は、Red Hat OpenShift および上流の Kubernetes クラスタからオンプレミス ONTAP ボリュームへの NFSv3 および NFSv4 接続での Kerberos 暗号化をサポートします。

NFS 暗号化を使用するボリュームを作成、削除、サイズ変更、スナップショット、クローン、読み取り専用クローン、およびインポートできます。

オンプレミス **ONTAP** ボリュームでインフライト **Kerberos** 暗号化を設定する

管理対象クラスタとオンプレミス ONTAP ストレージバックエンド間のストレージトラフィックで Kerberos 暗号化を有効にすることができます。

> オンプレミス ONTAP ストレージバックエンドを使用した NFS トラフィックの Kerberos 暗号化は、 `ontap-nas`ストレージドライバーを使用する場合にのみサポートされます。

開始する前に

- `tridentctl`ユーティリティにアクセスできることを確認してください。
- ONTAP ストレージバックエンドへの管理者アクセス権があることを確認してください。
- ONTAP ストレージバックエンドから共有するボリュームの名前を必ず確認してください。
- NFS ボリュームの Kerberos 暗号化をサポートするように ONTAP ストレージ VM を準備しておいてください。手順については "データLIFでKerberosを有効にする"を参照してください。

- Kerberos 暗号化で使用する NFSv4 ボリュームが正しく構成されていることを確認します。NetApp NFSv4 ドメイン設定セクション（13 ページ） "NetApp NFSv4 の機能強化とベストプラクティスガイド" を参照してください。

ONTAP エクスポートポリシーの追加または変更

ONTAP ストレージ VM ルートボリュームおよびアップストリーム Kubernetes クラスタと共有される ONTAP ボリュームに対して、Kerberos 暗号化をサポートする既存の ONTAP エクスポートポリシーにルールを追加するか、新しいエクスポートポリシーを作成する必要があります。追加するエクスポートポリシールール、または作成する新しいエクスポートポリシーは、次のアクセスプロトコルとアクセス権限をサポートする必要があります（

アクセス プロトコル

NFS、NFSv3、および NFSv4 アクセスプロトコルを使用してエクスポートポリシーを構成します。

アクセスの詳細

ボリュームのニーズに応じて、3 つの異なるバージョンの Kerberos 暗号化のいずれかを設定できます：

- **Kerberos 5** -（認証と暗号化）
- **Kerberos 5i** -（ID保護を備えた認証と暗号化）
- **Kerberos 5p** -（IDとプライバシー保護を使用した認証と暗号化）

適切なアクセス権限を持つ ONTAP エクスポート ポリシー ルールを設定します。たとえば、クラスタが Kerberos 5i と Kerberos 5p 暗号化を組み合わせて NFS ボリュームをマウントする場合は、次のアクセス設定を使用します：

タイプ	読み取り専用アクセス	読み取り/書き込みアクセス	スーパーユーザーアクセス
UNIX	有効	有効	有効
Kerberos 5i	有効	有効	有効
Kerberos 5p	有効	有効	有効

ONTAP エクスポートポリシーとエクスポートポリシールールの作成方法については、次のドキュメントを参照してください：

- "エクスポート ポリシーの作成"
- "エクスポート ポリシーへのルールの追加"

ストレージバックエンドを作成する

Kerberos 暗号化機能を含む Trident ストレージバックエンド構成を作成できます。

タスク概要

Kerberos暗号化を設定するストレージバックエンド構成ファイルを作成するときに、 `spec.nfsMountOptions` パラメータを使用して3つの異なるバージョンのKerberos暗号化のいずれかを指定できます：

- `spec.nfsMountOptions: sec=krb5`（認証と暗号化）
- `spec.nfsMountOptions: sec=krb5i`（ID 保護を備えた認証と暗号化）

- `spec.nfsMountOptions: sec=krb5p`（IDとプライバシー保護を備えた認証と暗号化）

Kerberos レベルを 1 つだけ指定します。パラメータリストで複数の Kerberos 暗号化レベルを指定した場合、最初のオプションのみが使用されます。

手順

1. マネージド クラスターで、次の例を使用してストレージ バックエンド構成ファイルを作成します。括弧内の値<>を環境からの情報に置き換えます：

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. 前の手順で作成した構成ファイルを使用して、バックエンドを作成します：

```
tridentctl create backend -f <backend-configuration-file>
```

バックエンドの作成に失敗した場合は、バックエンドの構成に問題があります。次のコマンドを実行すると、ログを表示して原因を特定できます：

```
tridentctl logs
```

構成ファイルの問題を特定して修正したら、create コマンドを再度実行できます。

ストレージクラスを作成する

Kerberos 暗号化を使用してボリュームをプロビジョニングするためのストレージクラスを作成できます。

タスク概要

ストレージクラスオブジェクトを作成するときに、 `mountOptions`パラメータを使用してKerberos暗号化の3つの異なるバージョンのいずれかを指定できます：

- `mountOptions: sec=krb5`（認証と暗号化）
- `mountOptions: sec=krb5i`（ID 保護を備えた認証と暗号化）
- `mountOptions: sec=krb5p`（IDとプライバシー保護を備えた認証と暗号化）

Kerberos レベルを 1 つだけ指定します。パラメータリストで複数の Kerberos 暗号化レベルを指定した場合、最初のオプションのみが使用されます。ストレージバックエンド構成で指定した暗号化レベルがストレージクラスオブジェクトで指定したレベルと異なる場合は、ストレージクラスオブジェクトが優先されます。

手順

1. 次の例を使用して、StorageClass Kubernetes オブジェクトを作成します：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. ストレージクラスを作成します：

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. ストレージクラスが作成されていることを確認します：

```
kubectl get sc ontap-nas-sc
```

次のような出力が表示されます。

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

ボリュームのプロビジョニング

ストレージバックエンドとストレージクラスを作成したら、ボリュームをプロビジョニングできるようになります。手順については、 "ボリュームをプロビジョニングする"を参照してください。

Azure NetApp Files ボリュームでインフライト Kerberos 暗号化を設定する

マネージドクラスタと単一のAzure NetApp Filesストレージバックエンド、またはAzure NetApp Filesストレージバックエンドの仮想プール間のストレージトラフィックでKerberos暗号化を有効にすることができます。

開始する前に

- 管理対象の Red Hat OpenShift クラスタで Trident が有効になっていることを確認してください。
- `tridentctl`ユーティリティにアクセスできることを確認してください。
- Kerberos 暗号化用の Azure NetApp Files ストレージバックエンドを準備済みであることを確認してください。要件を確認し、 "Azure NetApp Files のドキュメント"の手順に従ってください。
- Kerberos 暗号化で使用する NFSv4 ボリュームが正しく構成されていることを確認します。NetApp NFSv4 ドメイン設定セクション（13 ページ） "NetApp NFSv4 の機能強化とベストプラクティスガイド"を参照してください。

ストレージバックエンドを作成する

Kerberos 暗号化機能を含む Azure NetApp Files ストレージバックエンド構成を作成できます。

タスク概要

Kerberos 暗号化を構成するストレージバックエンド構成ファイルを作成するときに、次の 2 つのレベルのいずれかで適用されるように定義できます：

- `spec.kerberos`フィールドを使用した*ストレージバックエンドレベル*
- `spec.storage.kerberos`フィールドを使用した*仮想プールレベル*

仮想プールレベルで設定を定義する場合、ストレージクラスのラベルを使用してプールが選択されます。

どちらのレベルでも、 Kerberos 暗号化の 3 つの異なるバージョンのいずれかを指定できます：

- `kerberos: sec=krb5`（認証と暗号化）
- `kerberos: sec=krb5i`（ID 保護を備えた認証と暗号化）
- `kerberos: sec=krb5p`（IDとプライバシー保護を備えた認証と暗号化）

手順

1. 管理対象クラスタで、ストレージバックエンドを定義する必要がある場所（ストレージバックエンドレベルまたは仮想プールレベル）に応じて、次のいずれかの例を使用してストレージバックエンド構成ファイルを作成します。括弧内の値<>を環境からの情報に置き換えます：

ストレージバックエンドレベルの例

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

仮想プールレベルの例

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. 前の手順で作成した構成ファイルを使用して、バックエンドを作成します：

```
tridentctl create backend -f <backend-configuration-file>
```

バックエンドの作成に失敗した場合は、バックエンドの構成に問題があります。次のコマンドを実行すると、ログを表示して原因を特定できます：

```
tridentctl logs
```

構成ファイルの問題を特定して修正したら、create コマンドを再度実行できます。

ストレージクラスを作成する

Kerberos 暗号化を使用してボリュームをプロビジョニングするためのストレージクラスを作成できます。

手順

1. 次の例を使用して、StorageClass Kubernetes オブジェクトを作成します：

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. ストレージクラスを作成します：

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. ストレージクラスが作成されていることを確認します：

```
kubectl get sc -sc-nfs
```

次のような出力が表示されます。

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

ボリュームのプロビジョニング

ストレージバックエンドとストレージクラスを作成したら、ボリュームをプロビジョニングできるようになります。手順については、 "ボリュームをプロビジョニングする"を参照してください。