ベストプラクティスと推奨事項
Trident

NetApp
March 02, 2026

目次

ベストプラクティスと推奨事項

導入

Tridentを導入する際には、ここに記載されている推奨事項に従ってください。

専用のネームスペースに導入します

"ネームスペース"異なるアプリケーション間で管理を分離し、リソース共有の障壁となります。たとえば、あるネームスペースの PVC を別のネームスペースから使用することはできません。Tridentは、Kubernetesクラスタ内のすべてのネームスペースにPVリソースを提供するため、Privilegesを昇格させたサービスアカウントを利用します。

また、 Trident ポッドにアクセスすると、ユーザがストレージシステムのクレデンシャルやその他の機密情報にアクセスできるようになります。アプリケーションユーザ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します

Kubernetes には、 2 つの機能があります。これらの機能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" 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使用制限をネームスペース単位で実装できます。さらに、を使用します "範囲制限" 要求がプロビジョニングツールに転送される前に、 PVC 要求が最小値と最大値の両方の範囲内にあることを確認します。

これらの値はネームスペース単位で定義されます。つまり、各ネームスペースに、リソースの要件に応じた値を定義する必要があります。の詳細については、こちらを参照してください "クォータの活用方法"。

ストレージ構成

ネットアップポートフォリオの各ストレージプラットフォームには、コンテナ化されたアプリケーションやそうでない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 や Element と連携1 つのプラットフォームが他のプラットフォームよりもすべてのアプリケーションとシナリオに適しているわけではありませんが、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。

使用するプロトコルに対応した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。必要に応じて、アプリケーションのベストプラクティスを適用する際に、バックエンド、ストレージクラス、 PVC の設定を利用して、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もできます。

ONTAP と Cloud Volumes ONTAP のベストプラクティス

Trident 向けに ONTAP と Cloud Volumes ONTAP を設定するためのベストプラクティスをご確認ください。

次に示す推奨事項は、 Trident によって動的にプロビジョニングされたボリュームを消費するコンテナ化されたワークロード用に ONTAP を設定する際のガイドラインです。それぞれの要件を考慮し、環境内で適切かどうかを評価する必要があります。

Trident 専用の **SVM** を使用

Storage Virtual Machine （ SVM ）を使用すると、 ONTAP システムのテナントを分離し、管理者が分離できます。SVM をアプリケーション専用にしておくと、権限の委譲が可能になり、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ます。

SVM の管理には、いくつかのオプションを使用できます。

- バックエンド構成でクラスタ管理インターフェイスを適切なクレデンシャルとともに指定し、 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。
- NFS データインターフェイスで管理ロールを共有します。

いずれの場合も、インターフェイスは DNS にあり、 Trident の設定時には 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 を保持しなくても SVM-DR などの一部の DR シナリオが簡単になります。

専用の管理 LIF または共有の管理 LIF を SVM に使用する方法は推奨されませんが、ネットワークセキュリティポリシーを選択した方法と一致させる必要があります。最大の柔軟性を確保するには、どのような場合でも DNS 経由で管理 LIF にアクセスできるようにします "SVM-DR" Trident と組み合わせて使用できます。

最大ボリューム数を制限します

ONTAP ストレージシステムの最大ボリューム数は、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。を参照してください "NetApp Hardware Universe の略" 具体的な制限については、使用しているプラットフォームと ONTAP のバージョンに対応しています。ボリューム数を使い果たした場合、 Trident のプロビジョニング処理だけでなく、すべてのストレージ要求に対してプロビジョニング処理が失敗します。

Trident の「 ONTAP - NAS 」と「 ONTAP - SAN 」ドライバは、作成された Kubernetes 永続ボリューム（ PV ）ごとに FlexVol をプロビジョニングします。「 ONTAP-NAS-エコノミー 」ドライバは、 PVS 200 個につき約 1 つの FlexVol を作成します（ 50 ～ 300 の範囲で構成可能）。「 ONTAP-SAN-エコノミー 」ドライバは、 PVS 100 個につき約 1 つの FlexVol を作成します（ 50 ～ 200 の範囲で設定可能）。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条件によって異なります。

- ONTAP クラスタ内の既存のボリュームの数
- 他のアプリケーション用に Trident 外部でプロビジョニングするボリュームの数
- Kubernetes アプリケーションで消費されると予想される永続ボリュームの数

「 mAX-volumes 」の値は、 ONTAP クラスタ内のすべてのノードでプロビジョニングされたボリュームの合計数であり、個々の ONTAP ノードではプロビジョニングされません。その結果、 ONTAP クラスタノードの

Trident でプロビジョニングされたボリュームの数が、別のノードよりもはるかに多い、または少ない場合があります。

たとえば、2ノードONTAPクラスタでは、最大2、000個のFlexVolボリュームをホストできます。最大ボリューム数を 1250 に設定していると、非常に妥当な結果が得られます。ただし、SVMに1つのノードからしか割り当てられていない場合や、一方のノードから割り当てられたアグリゲートを（容量などの理由で）プロビジョニングできない場合は "アグリゲート"、Tridentでプロビジョニングされるすべてのボリュームのターゲットにもう一方のノードがなります。つまり、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その他のボリューム処理の両方に影響が及ぶ可能性があり`max-volumes`ます。* クラスタ内の各ノードのアグリゲートを、 Trident が使用する SVM に同じ番号で確実に割り当てることで、この状況を回避できます。 *

ボリュームのクローンを作成します

NetApp Tridentは、 `ontap-nas` 、 `ontap-san` 、 そして `solidfire-san`ストレージ ドライバー。使用する際は `ontap-nas-flexgroup`または `ontap-nas-economy`ドライバーの場合、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。

> 異なるストレージクラスに関連付けられたPVCのクローン作成は避けてください。互換性を確保し、予期しない動作を防ぐため、クローン作成操作は同じストレージクラス内で実行してください。

Trident で作成できるボリュームの最大サイズを制限

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、「 backend.json 」の定義で「 limitVolumeSize 」パラメータを使用します。

ストレージアレイでボリュームサイズを制御するだけでなく、 Kubernetes の機能も利用する必要があります。

Tridentで作成される**FlexVol**の最大サイズを制限する

ONTAPドライバ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FlexVolの最大サイズを設定するには、 `limitVolumePoolSize` `backend.json`定義でパラメータを使用します。

双方向 **CHAP** を使用するように **Trident** を設定します

バックエンド定義で CHAP イニシエータとターゲットのユーザ名とパスワードを指定し、 Trident を使用して SVM で CHAP を有効にすることができます。を使用する `useCHAP` バックエンド構成のパラメータである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します

SVM に適用された ONTAP QoS ポリシーを使用すると、 Trident でプロビジョニングされたボリュームが使用できる IOPS の数が制限されます。これはに役立ちます "Bully を防止します" Trident SVM 外のワークロードに影響を及ぼす、制御不能なコンテナ。

SVM の QoS ポリシーはいくつかの手順で作成します。正確な情報については、ご使用の ONTAP バージョンのマニュアルを参照してください。次の例は、 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>
```

また、使用しているバージョンの ONTAP でサポートされている場合は、最小 QoS を使用してコンテナ化されたワークロードへのスループットを保証することもできます。アダプティブ QoS は SVM レベルのポリシーには対応していません。

コンテナ化されたワークロード専用の IOPS は、さまざまな要素によって異なります。その中には、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 環境とは関係なく、ストレージリソースを利用するほかのワークロードがある場合は、それらのワークロードが誤って影響を受けないように注意する必要があります。
- 想定されるワークロードはコンテナで実行されます。IOPS 要件が高いワークロードをコンテナで実行する場合は、 QoS ポリシーの値が低いとエクスペリエンスが低下します。

SVM レベルで割り当てた QoS ポリシーを使用すると、 SVM にプロビジョニングされたすべてのボリュームで同じ IOPS プールが共有されることに注意してください。コンテナ化されたアプリケーションの 1 つまたは少数のに高い IOPS が必要な場合、コンテナ化された他のワークロードに対する Bully になる可能性があります。その場合は、外部の自動化を使用したボリュームごとの QoS ポリシーの割り当てを検討してください。

ONTAP バージョン 9.8 より前の場合は、 QoS ポリシーグループを SVM * only * に割り当ててください。

Trident の **QoS** ポリシーグループを作成

Quality of Service （ QoS ；サービス品質）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ようにします。ONTAP の QoS ポリシーグループには、ボリュームに対する QoS オプションが用意されており、ユーザは 1 つ以上のワークロードに対するスループットの上限を定義できます。QoSの詳細については、を参照してください。 "QoS によるスループットの保証"。
QoS ポリシーグループはバックエンドまたはストレージプールに指定でき、そのプールまたはバックエンドに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、最大スループット（以降のバージョンでは最小スループット）がフラットに表示されます。アダプティブ QoS では、ワークロードのサイズの変更に合わせてスループットが自動的に調整され、 TB または GB あたりの IOPS が一定に維持されます。これにより、何百何千という数のワークロードを管理する大規模な環境では大きなメリットが得られます。

QoS ポリシーグループを作成するときは、次の点に注意してください。

- バックエンド構成の「金庫」ブロックに「 QOSPolicy 」キーを設定する必要があります。次のバックエンド設定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
```

- ボリュームごとにポリシーグループを適用して、各ボリュームがポリシーグループの指定に従ってスループット全体を取得するようにします。共有ポリシーグループはサポートされません。

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

ストレージリソースへのアクセスを **Kubernetes** クラスタメンバーに制限する

Tridentで作成されたNFSボリューム、iSCSI LUN、およびFC LUNへのアクセスを制限することは、Kubernetes環境のセキュリティ体制にとって重要な要素です。これにより、 Kubernetes クラスタに属していないホストがボリュームにアクセスしたり、データが予期せず変更されたりすることを防止できます。

ネームスペースは Kubernetes 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。ただし、同じネームスペース内のリソースは共有可能であることが前提です。重要なのは、ネームスペース間に機能がないことです。つまり、 PVS 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、同じネームスペース内のポッドからのみアクセス可能です。* 適切な場合は、名前空間を使用して分離することが重要です。 *

Kubernetes 環境でデータセキュリティを使用する場合、ほとんどの組織で最も懸念されるのは、コンテナ内のプロセスがホストにマウントされたストレージにアクセスできることですが、コンテナ用ではないためです。 "ネームスペース" この種の妥協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。

権限付きコンテナは、通常よりもホストレベルの権限で実行されるコンテナです。デフォルトでは拒否されないため、を使用してこの機能を無効にしてください "ポッドセキュリティポリシー"。

Kubernetes と外部ホストの両方からアクセスが必要なボリュームでは、 Trident ではなく管理者が導入した PV で、ストレージを従来の方法で管理する必要があります。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ていない場合にのみ、ストレージボリュームが破棄されます。また、カスタムエクスポートポリシーを適用して、 Kubernetes クラスタノードおよび Kubernetes クラスタの外部にある

ターゲットサーバからのアクセスを可能にすることもできます。

専用のインフラノード（OpenShiftなど）や、ユーザアプリケーションをスケジュールできない他のノードを導入する場合は、ストレージリソースへのアクセスをさらに制限するために別 々 のエクスポートポリシーを使用する必要があります。これには、これらのインフラノードに導入されているサービス（ OpenShift Metrics サービスや Logging サービスなど）のエクスポートポリシーの作成と、非インフラノードに導入されている標準アプリケーション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します

Kubernetes クラスタ内の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各バックエンドに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、管理できます。これにより、 Trident はプロビジョニング対象のボリュームへのアクセスを Kubernetes クラスタ内のノードに制限し、ノードの追加や削除を簡易化します。

また、エクスポートポリシーを手動で作成し、各ノードの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。
- 「 vserver export-policy rule create 」 ONTAP CLI コマンドを使用して、エクスポートポリシーにルールを追加します。

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。

無効にします `showmount` アプリケーション**SVM**用

この `showmount`機能を使用すると、`NFS`クライアントが`SVM`に照会して使用可能な`NFS`エクスポートのリストを確認できます。`Kubernetes`クラスタに導入されたポッドは、に対してコマンドを実行して、使用可能なマウント（ポッドがアクセスできないマウントを含む）のリストを受け取ることができます `showmount -e`。これだけではセキュリティ上の妥協ではありませんが、権限のないユーザが NFS エクスポートに接続するのを阻止する可能性のある不要な情報が提供されます。

SVM レベルの ONTAP CLI コマンドを使用して、 SVM の howmount を無効にする必要があります。

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

SolidFire のベストプラクティス

Trident に SolidFire ストレージを設定するためのベストプラクティスをご確認ください。

SolidFire アカウントを作成します

各 SolidFire アカウントは固有のボリューム所有者で、 Challenge Handshake Authentication Protocol （ CHAP ；チャレンジハンドシェイク認証プロトコル）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、アカウント名とその CHAP クレデンシャルを使用してアクセスするか、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。アカウントには最大 2 、 000 個のボリュームを関連付けることができますが、 1 つのボリュームが属することのできるアカウントは 1 つだけです。

QoS ポリシーを作成する

標準的なサービス品質設定を作成して保存し、複数のボリュームに適用する場合は、 SolidFire のサービス品質（ QoS ）ポリシーを使用します。

QoS パラメータはボリューム単位で設定できます。QoS を定義する 3 つの設定可能なパラメータである Min IOPS 、 Max IOPS 、 Burst IOPS を設定することで、各ボリュームのパフォーマンスが保証されます。

4KB のブロックサイズの最小 IOPS 、最大 IOPS 、バースト IOPS の値を次に示します。

IOPS パラメータ	定義（ **Definition** ）	最小価値	デフォルト値	最大値（ **4KB** ）
最小 IOPS	ボリュームに対して保証されたレベルのパフォーマンス。	50	50	15000
最大 IOPS	パフォーマンスはこの制限を超えません。	50	15000	200,000
バースト IOPS	短時間のバースト時に許容される最大 IOPS 。	50	15000	200,000

Max IOPS と Burst IOPS は最大 200 、 000 に設定できますが、実際のボリュームの最大パフォーマンスは、クラスタの使用量とノードごとのパフォーマンスによって制限されます。

ブロックサイズと帯域幅は、 IOPS に直接影響します。ブロックサイズが大きくなると、システムはその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やします。帯域幅が増えると、システムが処理可能な IOPS は減少します。を参照してください "SolidFire のサービス品質" QoS およびパフォーマンスの詳細については、を参照してください。

SolidFire 認証

Element では、認証方法として CHAP とボリュームアクセスグループ（ VAG ）の 2 つがサポートされています。CHAP は CHAP プロトコルを使用して、バックエンドへのホストの認証を行います。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。CHAP はシンプルで拡張性に制限がないため、認証に使用することを推奨します。

Trident と強化された CSI プロビジョニングツールは、 CHAP 認証の使用をサポートします。VAG は、従来の CSI 以外の動作モードでのみ使用する必要があります。

CHAP 認証（イニシエータが対象のボリュームユーザであることの確認）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 を使用している場合は、単方向 CHAP と双方向 CHAP の 2 つのオプションがあります。単方向 CHAP は、 SolidFire アカウント名とイニシエータ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。双方向の CHAP オプションを使用すると、ボリュームがアカウント名とイニシエータシークレットを使用してホストを認証し、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、ボリュームを最も安全に認証できます。

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、ホストのイニシエータ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

関係なく、グループ内の各ボリュームにアクセスできます。iSCSI イニシエータが CHAP 認証を使用するように設定されている場合は、アカウントベースのアクセス制御が使用されます。iSCSI イニシエータが CHAP 認証を使用するように設定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。

詳細情報の入手方法

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。を検索します "NetApp ライブラリ" 最新バージョンの場合。

- ONTAP *
- "『 NFS Best Practice and Implementation Guide 』を参照してください"
- "SAN の管理"（iSCSIの場合）
- "RHEL 向けの iSCSI のクイック構成"
- Element ソフトウェア *
- "SolidFire for Linux を設定しています"
- 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 での MySQL に関するベストプラクティスです"
- "SolidFire での MySQL に関するベストプラクティスです"
- "NetApp SolidFire および 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です"

すべてのアプリケーションに具体的なガイドラインがあるわけではありません。そのためには、ネットアップのチームと協力し、を使用することが重要です "NetApp ライブラリ" 最新のドキュメントを検索できます。

Tridentの統合

Tridentを統合するには、ドライバの選択と導入、ストレージクラスの設計、仮想プールの設計、永続的ボリューム要求（PVC）によるストレージプロビジョニングへの影響、ボリューム処理、Tridentを使用したOpenShiftサービスの導入など、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。

ドライバの選択と展開

ストレージシステム用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、使用されるプロトコルと、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異なります。そのため、どのドライバを展開するかを決定する際には、慎重に検討する必要があります。

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 ベースのドライバがデフォルトで選択されますが、ブロックベースの iSCSI ドライバ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、ストレージチームとインフラチームの快適さレベルに基づいてプロトコルを選択してください。一般的に、ほとんどのアプリケーションでは両者の違いはほとんどないため、共有ストレージ（複数のポッドで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判断することがよくあります。

使用可能なONTAP バックエンドドライバは次のとおりです。

- [ONTAP-NAS] ：プロビジョニングされた各 PV は、フル ONTAP FlexVol です。
- 「 ONTAP-NAS-エコノミー 」：プロビジョニングされた各 PV は qtree で、 FlexVol あたりの qtree 数を設定できます（デフォルトは 200 ）。
- 「 ONTAP-NAS-flexgroup 」：フル ONTAP FlexGroup としてプロビジョニングされた各 PV と、 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、固有の FlexVol 内の LUN です。
- 「 ONTAP-SAN-エコノミー 」：各 PV がプロビジョニングされた LUN で、 FlexVol あたりの LUN 数を設定できます（デフォルトは 100 ）。

3 つの NAS ドライバの間で選択すると、アプリケーションで使用できる機能にいくつかの影響があります。

次の表では、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、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。上付き文字の脚注は、機能やドライバごとに機能を区別します。

ONTAP NAS ドライバ	**Snapshot**	クローン	動的な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。	○脚注： 5[]	はい。	○脚注： 1[]	はい。	○脚注： 1[]
「 ONTAP - NAS - エコノミー」	注：3[]	注：3[]	○脚注： 5[]	はい。	注：3[]	はい。	注：3[]
「 ONTAP-NAS-flexgroup 」	○脚注： 1[]	いいえ	○脚注： 5[]	はい。	○脚注： 1[]	はい。	○脚注： 1[]

Tridentでは、ONTAP向けに2つのSANドライバを提供しています。その機能は次のとおりです。

ONTAP SAN ドライバ	**Snapshot**	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。	○脚注： 4[]	はい。	○脚注： 1[]	はい。	○脚注： 1[]
「 ONTAP - SAN - エコノミー」	はい。	はい。	○脚注： 4[]	はい。	注：3[]	はい。	注：3[]

> 上記の表の脚注: Yes [1]: Tridentで管理されないYes [2]: Tridentで管理されるが、PVでは管理されないNO [3]: TridentとPVで管理されないYes [4]: raw-blockボリュームでサポートYes [5]: Tridentでサポート

PV に細分化されていない機能は FlexVol 全体に適用され、 PVS （共有 FlexVol 内の qtree または LUN ）にはすべて共通のスケジュールが適用されます。

上の表に示すように 'ONTAP-NAS' と「 ONTAP-NAS-エコノミー 」の機能の多くは同じですしかし 'ONTAP-NAS-エコノミー のドライバは ' スケジュールを PV 単位で制御する機能を制限するため ' これは特に災害復旧やバックアップ計画に影響を与える可能性がありますONTAP ストレージ上で PVC クローン機能を活用したい開発チームの場合 ' これは 'ONTAP-NAS' ONTAP -SAN' または ONTAP -SAN' のいずれかのドライバを使用する場合にのみ可能です

> 「 olidfire -san 」ドライバは PVC のクローン作成にも対応しています。

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ストレージなど、さまざまなユースケースでデータ制御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 の互換性のあるドライバは、「 ONTAP-NAS' 」、「 ONTAP-NAS-エコノミー 」、「 ONTAP-SAN' 」、「 ONTAP-SAN-エコノミー 」です。Cloud Volume ONTAP for Azure と Cloud Volume ONTAP for GCP に該当します。

ONTAP バックエンドドライバ用の**Amazon FSX**

Amazon FSx for NetApp ONTAPを使用すると、AWSにデータを格納する際のシンプルさ、即応性、セキュリティ、拡張性を活用しながら、使い慣れたNetAppの機能、パフォーマンス、管理機能を活用できます。FSx for ONTAPは、多くの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 の互換性のあるドライバはです `ontap-nas`、 `ontap-nas-economy`、 `ontap-nas-flexgroup`、 `ontap-san` および `ontap-san-economy`。

NetApp HCI / SolidFireバックエンドドライバ

NetApp HCI / SolidFire プラットフォームで使用される「 olidfire -SAN」 ドライバは、管理者が QoS 制限に基づいて Trident の Element バックエンドを構成するのに役立ちます。Trident によってプロビジョニングされるボリュームに特定の QoS 制限を設定するため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レプリケーションなどの Element ストレージ機能は、 'olidfire-san' ドライバではサポートされていません。これらの処理は、 Element ソフトウェアの Web UI から手動で実行する必要があります。

SolidFire ドライバ	**Snapshot**	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
「 olidfire -san 」	はい。	はい。	○脚注： 2[]	はい。	はい。	はい。	○脚注： 1[]

> 脚注:はい脚注: 1[]: Tridentで管理されていません脚注: 2[]: raw-block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はドライバを使用して `azure-netapp-files`サービスを管理し"Azure NetApp Files の特長"ます。

このドライバとその設定方法の詳細については、を参照してください"Azure NetApp Files 向けの Trident バックエンド構成"。

Azure NetApp Files ドライバ	**Snapshot**	クローン	マルチアタッチ	**QoS**	を展開します	レプリケーション
「 azure-NetApp-files 」と入力します	はい。	はい。	はい。	はい。	はい。	○脚注： 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、アプリケーション用のストレージクラスの設計方法について説明します。

特定のバックエンド使用率

フィルタリングは、特定のストレージクラスオブジェクト内で使用でき、そのストレージクラスで使用するストレージプールまたはプールのセットを決定します。ストレージクラスでは
'`toragePools’additionalStoragePools’excludeStoragePools" の 3 セットのフィルタを設定できます

パラメータを使用 `storagePools`すると、指定した属性に一致するプールだけにストレージを制限できます。パラメータは、 `additionalStoragePools`属性とパラメータで選択された一連のプールとともに、Tridentがプロビジョニングに使用する一連のプールを拡張するために使用し `storagePools`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、両方を使用して、適切なストレージプールセットが選択されていることを確認することもできます。

excludeStoragePools' パラメータを使用して ' 属性に一致するプールの一覧を除外します

QoSポリシーをエミュレートします

ストレージクラスを設計して Quality of Service ポリシーをエミュレートする場合は ' 「メディア」属性を「 hdd 」または「 sd 」として ' ストレージクラスを作成しますストレージクラスで言及されている「メディア」属性に基づいて、 Trident は「 hdd 」アグリゲートまたは「 sd 」アグリゲートにメディア属性と一致させる適切なバックエンドを選択し、ボリュームのプロビジョニングを特定のアグリゲートに誘導します。したがって、「メディア」属性が「 SD 」に設定されているストレージクラス Premium を作成して、プレミアム QoS ポリシーに分類できます。メディア属性を「 hdd 」に設定し、標準の QoS ポリシーとして分類できる、別のストレージクラス標準を作成できます。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とシックプロビジョニング、 Snapshot 、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を直接プロビジョニング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、必要な機能を有効にしてバックエンドに適した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バックエンドで使用できます。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で参照可能なバックエンド上に抽象化レベルを作成して、バックエンドにボリュームを柔軟かつ効率的に配置できます。同じサービスクラスを使用して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。ストレージクラスに特定のラベルを持つセレクタが設定されている場合、Tridentはボリュームを配置するすべてのセレクタ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の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つをボリュームのプロビジョニング元として選択します。

仮想プールの設計

バックエンドを作成する際には、一般的に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認証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、Kubernetesストレージクラスを介してセレクタとし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を使用して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、 Azure NetApp Filesが含まれます。

> ⓘ 仮想プールを定義する場合は、バックエンド定義で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。また、既存の仮想プールの属性を編集または変更したり、新しい仮想プールを定義したりしないことを推奨します。

さまざまなサービスレベル/**QoS**のエミュレート

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できます。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セットアップする方法を見ていきましょう。Azure NetApp Filesバックエンドには、異なるパフォーマンスレベルを表す複数のラベルを設定します。設定 `servicelevel` 適切なパフォーマンスレベルを考慮し、各ラベルの下にその他の必要な側面を追加します。次に、異なる仮想プールにマッピングするさまざまなKubernetesストレージクラスを作成します。を使用する `parameters.selector` 各StorageClassは、ボリュームのホストに使用できる仮想プールを呼び出します。

特定の一連の側面を割り当てます

特定の側面を持つ複数の仮想プールは、単一のストレージバックエンドから設計できます。そのためには、バックエンドに複数のラベルを設定し、各ラベルに必要な側面を設定します。を使用して、さまざまなKubernetesストレージクラスを作成します `parameters.selector` 異なる仮想プールにマッピングされるフィールド。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に定義された設定が適用されます。

ストレージプロビジョニングに影響する **PVC** 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は、PVCの作成時にTridentプロビジョニングの決定プロセスに影響す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 つがアクセスモードです。必要なモードは、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を与える可能性があります。

Trident は、以下のマトリックスに記載されているアクセス方法で使用されているストレージプロトコルと一致するかどうかを試みます。これは、基盤となるストレージプラットフォームに依存しません。

	ReadWriteOnce コマンドを使用します	**ReadOnlyMany**	**ReadWriteMany**
iSCSI	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 環境に送信された ReadWriteMany PVC が要求された場合、ボリュームはプロビジョニングされません。このため、リクエスタは、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

永続ボリュームの変更

永続ボリュームとは、 Kubernetes で変更不可のオブジェクトを 2 つだけ除いてです。再利用ポリシーとサイズは、いったん作成されると変更できます。ただし、これにより、ボリュームの一部の要素がKubernetes以外で変更されることが防止されるわけではありません。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、誤って容量が消費されないようにしたり、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便利です。

> Kubernetesのツリー内プロビジョニングツールは、現時点ではNFS、iSCSI、またはFC PVSのボリュームサイズ変更処理をサポートしていません。Tridentでは、NFS、iSCSI、FCの両方のボリュームの拡張がサポートされています。

作成後に PV の接続の詳細を変更することはできません。

オンデマンドのボリューム **Snapshot** を作成

Trident では、 CSI フレームワークを使用して、ボリュームスナップショットのオンデマンド作成とスナップショットからの PVC の作成がサポートされます。Snapshot は、データのポイントインタイムコピーを管理し、 Kubernetes のソース PV とは無関係にライフサイクルを管理する便利な方法です。これらの Snapshot を使用して、 PVC をクローニングできます。

Snapshot からボリュームを作成します

Trident では、ボリューム Snapshot から PersistentVolumes を作成することもできます。そのためには、PersistentVolumeClaimを作成し、ボリュームの作成元となるSnapshotとしてを指定します `datasource`。Trident は、 Snapshot にデータが存在するボリュームを作成することで、この PVC を処理します。この機能を使用すると、複数のリージョン間でデータを複製したり、テスト環境を作成したり、破損した本番ボリューム全体を交換したり、特定のファイルとディレクトリを取得して別の接続ボリュームに転送したりできます。

クラスタ内でボリュームを移動します

ストレージ管理者は、 ONTAP クラスタ内のアグリゲート間およびコントローラ間で、ストレージ利用者への無停止でボリュームを移動できます。この処理は、Tridentが使用しているSVMからアクセスできるデスティネーションアグリゲートであるかぎり、TridentまたはKubernetesクラスタには影響しません。重要なことは、アグリゲートがSVMに新しく追加されている場合は、バックエンドをTridentに再追加してリフレッシュする必要があることです。これにより、 Trident が SVM のインベントリを再設定し、新しいアグリゲートが認識されます。

ただし、バックエンド間でのボリュームの移動は Trident では自動で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、または別のストレージプラットフォームへの SVM の間も含まれます（ Trident に接続されているストレージシステムの場合も含む）。

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。

ボリュームを展開します

Trident は、 NFS、iSCSI、および FC PV のサイズ変更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のサイズを直接変更できるようになります。ボリューム拡張は、ONTAP、 SolidFire/ NetApp HCIバックエンドを含むすべての主要なNetAppストレージ プラットフォームで可能です。後で拡張できるように設定するには `allowVolumeExpansion`に `true`ボリュームに関連付けられた StorageClass 内。永続ボリュームのサイズを変更する必要がある場合は、 `spec.resources.requests.storage`永続ボリューム要求の注釈を必要なボリューム サイズに設定します。Trident は、ストレージ クラスター上のボリュームのサイズ変更を自動的に処理します。

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` 、 そして `azure-netapp-files`ドライバー。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、災害復旧シナリオ時に役立ちます。

ONTAPドライバとドライバを使用する場合 `solidfire-san`は、コマンドを使用し `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml`て、Tridentで管理するKubernetesに既存のボリュームをインポートします。import volume コマンドで使用した PVC YAML または JSON ファイルは、 Trident をプロビジョニングツールとして識別するストレージクラスを指定します。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にクローニングします。

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

上記のコマンドが実行されると、 Trident はバックエンド上のボリュームを検出してサイズを確認します。設定されたPVCのボリュームサイズを自動的に追加（および必要に応じて上書き）します。Trident が新しい PV を作成し、 Kubernetes が PVC を PV にバインド

特定のインポートされた PVC を必要とするようにコンテナを導入した場合、ボリュームインポートプロセスによって PVC/PV ペアが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ると、他に問題がなければコンテナが起動します。

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、に記載されています "netapp.io のコマンドです" を参照してください "ブログ"。

ロギングサービス

他の OpenShift サービスと同様に、ログ記録サービスは、 Ansible と、インベントリファイル（別名）で提供される構成パラメータを使用して導入されますホスト。プレイブックに含まれています。ここでは、 OpenShift の初期インストール時にロギングを導入し、 OpenShift のインストール後にロギングを導入するという、 2 つのインストール方法について説明します。

> Red Hat OpenShift バージョン 3.9 以降、データ破損に関する懸念があるため、記録サービスに NFS を使用しないことを公式のドキュメントで推奨しています。これは、 Red Hat 製品のテストに基づいています。ONTAP NFSサーバにはこのような問題がないため、ロギング環境を簡単にバックアップできます。ロギングサービスには最終的にどちらかのプロトコルを選択する必要がありますが、両方のプロトコルがネットアッププラットフォームを使用する場合に適していることと、 NFS を使用する理由がないことを確認してください。

ロギング・サービスで NFS を使用する場合は、インストーラが失敗しないように、 Ansible 変数「 OpenShift 」の「 OpenShift 」 enable_unsupported _configurations 」を「 true 」に設定する必要があります。

はじめに

ロギングサービスは、必要に応じて、両方のアプリケーションに導入することも、 OpenShift クラスタ自体のコア動作に導入することもできます。オペレーション・ログを配置する場合 ' 変数 OpenShift の logging_use_ops を true として指定すると ' サービスの 2 つのインスタンスが作成されます操作のロギングインスタンスを制御する変数には「 ops 」が含まれ、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で正しいストレージが使用されるようにするには、導入方法に応じてAnsible変数を設定することが重要です。それぞれの導入方法のオプションを見てみましょう。

> 次の表には、ロギングサービスに関連するストレージ構成に関連する変数のみを示します。展開に応じて、レビュー、設定、および使用する必要がある他のオプションを見つけることができます"Red Hat OpenShiftのロギングに関するドキュメント"。

次の表の変数では、入力した詳細を使用してロギングサービスの PV と PVC を作成する Ansible プレイブックが作成されます。この方法は、 OpenShift 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はるかに柔軟性に劣るが、既存のボリュームがある場合はオプションとなります。

変数（ **Variable** ）	詳細
「 OpenShift 」ロギング・ストレージ・タイプ	インストーラがログサービス用の NFS PV を作成するように 'NFS' に設定します
「 OpenShift 」ロギング・ストレージ・ホスト	NFS ホストのホスト名または IP アドレス。この値は、仮想マシンのdataLIFに設定する必要があります。
「 OpenShift 」ロギング・ストレージ・ NFS_DIRECT'	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 」としてジャンクションされている場合、この変数にそのパスを使用します。

変数（ **Variable** ）	詳細
「 OpenShift 」ロギング・ストレージ・ボリューム名	作成する PV の名前（「 pv_ose_logs 」など）。
「 OpenShift 」ロギング・ストレージ・ボリューム・サイズ	NFS エクスポートのサイズ（例： 100Gi ）

OpenShift クラスタがすでに実行中で、そのため Trident を導入して設定した場合、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（ **Variable** ）	詳細
'OpenShift の logging_es_vpc_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場合は true に設定します。
「 OpenShift logging _es_vpc_storage_class_name 」	PVC で使用されるストレージクラスの名前。
「 OpenShift logging _es_vpc_size 」を参照してください	PVC で要求されたボリュームのサイズ。
「 OpenShift logging _es_vpc_prefix 」を参照してください	ロギングサービスで使用される PVC のプレフィックス。
'OpenShift の logging_es_ops_pvc_dynamic	動的にプロビジョニングされたボリュームを ops ロギングインスタンスに使用するには、「 true 」に設定します。
「 OpenShift logging _es_ops_pvc_storage_class_name 」を参照してください	処理ロギングインスタンスのストレージクラスの名前。
'OpenShift logging _es_ops_pvc_size	処理インスタンスのボリューム要求のサイズ。
「 OpenShift logging _es_ops_pvc_prefix 」を参照してください	ops インスタンス PVC のプレフィックス。

ロギングスタックを導入します

初期の OpenShift インストールプロセスの一部としてロギングを導入する場合、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 オブジェクトを構成および導入して、 Ansible が完了したらすぐにサービスを利用できるようにします。

ただし、最初のインストール後に導入する場合は、コンポーネントプレイブックを Ansible で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変更される場合がありますので、お使いのバージョンに合わせてお読みください"Red Hat OpenShift Container Platform 3.11のドキュメント"。

指標サービス

この指標サービスは、 OpenShift クラスタのステータス、リソース利用率、可用性に関する重要な情報を管理者に提供します。ポッドの自動拡張機能にも必要であり、多くの組織では、チャージバックやショーバックのアプリケーションに指標サービスのデータを使用しています。

ロギングサービスや OpenShift 全体と同様に、 Ansible を使用して指標サービスを導入します。また、ロギン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中、またはコンポーネントのインストール方法を使用して運用後に導入できます。次の表に、指標サービスに永続的ストレージを設定する際に重要となる変数を示します。

以下の表には、指標サービスに関連するストレージ構成に関連する変数のみが含まれています。このドキュメントには、他にも導入環境に応じて確認、設定、使用できるオプションが多数あります。

変数（ **Variable** ）	詳細
「 OpenShift _ metrics _ storage _kind 」	インストーラがログサービス用の NFS PV を作成するように 'NFS' に設定します
「 OpenShift _ metrics _storage_host 」というようになります	NFS ホストのホスト名または IP アドレス。この値は、SVMのdataLIFに設定する必要があります。
「 OpenShift _ metrics _storage_nfs_directory 」というエラーが表示されます	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 メトリック」として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 」という形式で指定します	作成する PV の名前（「 pv_ose_metrics 」など）。
「 OpenShift _ metrics _storage_volume_size 」というようになります	NFS エクスポートのサイズ（例： 100Gi ）

OpenShift クラスタがすでに実行中で、そのため Trident を導入して設定した場合、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（ **Variable** ）	詳細
「 OpenShift _ metrics _ cassandra_vpc_prefix 」という形式で指定します	メトリック PVC に使用するプレフィックス。
「 OpenShift _ metrics _ cassandra_vp_size' 」のようになります	要求するボリュームのサイズ。
「 OpenShift _ metrics _ cassandra_storage_type 」のようになります	指標に使用するストレージのタイプ。適切なストレージクラスを使用して PVC を作成するには、 Ansible に対してこれを dynamic に設定する必要があります。
「 OpenShift _ metrics _cassanda_pvc_storage_class_name 」という形式で指定します	使用するストレージクラスの名前。

指標サービスを導入する

ホスト / インベントリファイルに適切な Ansible 変数を定義して、 Ansible でサービスを導入します。OpenShift インストール時に導入する場合は、 PV が自動的に作成されて使用されます。コンポーネントプレイブックを使用して導入する場合は、OpenShiftのインストール後に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らサービスが導入されます。

上記の変数と導入プロセスは、 OpenShift の各バージョンで変更される可能性があります。使用しているバージョンを確認し、環境に合わせて構成されるようにして"Red Hat OpenShift導入ガイド"ください。

データ保護とディザスタリカバリ

TridentとTridentを使用して作成された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。永続性に関する要件があるアプリケーションごとに、データ保護とリカバリの戦略を用意しておく必要があります。

Tridentのレプリケーションとリカバリ

災害発生時にTridentをリストアする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格納および管理し、Kubernetesクラスタetcdを使用してメタデータを格納します。

手順

1. を使用してKubernetesクラスタetcdをバックアップします "Kubernetes：etcdクラスタのバックアップ"。
2. FlexVol volumeへのバックアップアーティファクトの配置

> NetAppでは、FlexVolが配置されているSVMを別のSVMとのSnapMirror関係で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して、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が格納されているボリュームを、マスターノードとしてセットアップす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のにコピーします `/etc/kubernetes/pki` およびの下のetcdメンバーファイル `/var/lib/etcd`。
3. を使用して、etcdバックアップからKubernetesクラスタをリストアします "Kubernetes：etcdクラスタをリストアします"。
4. を実行します `kubectl get crd` Tridentのカスタムリソースがすべて稼働していることを確認し、Tridentオブジェクトを読み出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では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を使用してSVMをレプリケートできます "ONTAP SnapMirrorの略"。

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。システムがリストアされたら、プライマリに戻すことができます。

このタスクについて

SnapMirror SVMレプリケーション機能を使用する場合は、次の点を考慮してください。

- SVM-DRを有効にした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。
- SVM-DRをサポートするバックエンドにレプリケーション不要のボリュームをプロビジョニングしないように、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設定します。
- アプリケーション管理者は、レプリケーションに伴う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プランを慎重に検討する必要があります。

SVMレプリケーション

を使用できます "ONTAP：SnapMirrorレプリケーション" をクリックしてSVMレプリケーション関係を作成します。

SnapMirrorでは、レプリケートする対象を制御するオプションを設定できます。プリフォーム時に選択したオプションを知っておく必要がTridentを使用したSVMのリカバリあります。

- "-identity-preserve trueを指定します" SVMの設定全体をレプリケートします。
- "-discard-configs network" LIFと関連ネットワークの設定を除外します。
- "-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**のリカバリ

Tridentでは、SVMの障害は自動的に検出されません。災害が発生した場合、管理者は新しいSVMへのTridentフェイルオーバーを手動で開始できます。

手順

1. スケジュールされた実行中の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. を指定した場合 `-identity-preserve false` または `-discard-config network` SVMレプリケーションを設定する場合は、を更新します `managementLIF` および `dataLIF` をTridentバックエンド定義ファイルに追加します。
3. 確認します `storagePrefix` は、Tridentバックエンド定義ファイルに含まれています。このパラメータは変更できません。省略しています `storagePrefix` バックエンドの更新が失敗するように原因します。
4. 次のコマンドを使用して、必要なすべてのバックエンドを更新して新しいデスティネーションSVM名を反映します。

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

5. を指定した場合 `-identity-preserve false` または `discard-config network`、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。

> を指定する `-identity-preserve true`と、デスティネーションSVMがアクティブ化されたときに、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からデータの提供が開始されます。

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"ONTAP SnapMirrorのレプリケーションとリカバリ"。

その後、を使用して、リカバリしたボリュームをTridentにインポートできます"tridentctlボリュームインポート"。

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

Snapshotデータの保護

次のコマンドを使用してデータを保護およびリストアできます。

- 永続ボリューム（PV）のKubernetesボリュームSnapshotを作成するための外部のSnapshotコントローラとCRD。

  "ボリューム Snapshot"

- ONTAP Snapshot：ボリュームの内容全体のリストア、または個 々 のファイルまたはLUNのリカバリに使用します。

  "ONTAPスナップショット"

Tridentによるステートフル 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 クラスター内の異常なノードからボリュームを自動的にデタッチして、データの破損を防ぎ、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。

フォースデタッチの詳細

強制デタッチは、 `ontap-san` 、 `ontap-san-economy` 、 `ontap-nas` 、 そして `ontap-nas-economy`のみ。強制デタッチを有効にする前に、Kubernetes クラスターで非正常なノード シャットダウン (NGNS) を有効にする必要があります。Kubernetes 1.28 以降では、NGNS はデフォルトで有効になっています。詳細については、 "Kubernetes：正常なノードシャットダウンではありません" 。

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

TridentはKubernetes NGNに依存しているため、許容できないすべてのワークロードのスケジュールを再設定するまで、正常でないノードからテイントを削除しないで `out-of-service`ください。汚染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taintをノードに適用し、 `enableForceDetach`をに設定する `true`と `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute`、Tridentはノードのステータスを確認し、次の処理を行います。

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（新しいパブリケーションに対しては安全ではない）としてマークします `dirty`。

Tridentコントローラは、Tridentノードポッドによって（とマークされた後で）ノードが再修飾されるまで、新しいパブリッシュボリューム要求を拒否し `dirty`ます。マウントされたPVCを使用して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が完了したあとも）は、Tridentがそのノードを検証できるようになるまで受け入れられません `clean`（新しいパブリケーションに対して安全）。

ノードの健常性が回復してtaintが削除されると、Tridentは次の処理を実行します。

1. ノード上の古い公開パスを特定してクリーンアップします。
2. ノードが状態（アウトオブサービス状態が削除され、ノードが `Ready`状態）で、古い公開パスがすべてクリーンである場合、 `cleanable`Tridentはノードをとして再登録し、新しい公開ボリュームをそのノードに許可します `clean`。

自動フェイルオーバーの詳細

統合により強制デタッチプロセスを自動化できます。"ノードヘルスチェック（NHC）オペレーター" 。ノード障害が発生すると、NHC は、障害が発生したノードを定義する Trident の名前空間に TridentNodeRemediation CR を作成して、 Tridentノード修復 (TNR) をトリガーし、自動的に強制的にデタッチします。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。

ノードポッドの削除プロセスが失敗しました

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 エコノミー ボリューム。

参照[フォースデタッチの詳細]。

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がない限り、障害が発生したノードから削除されません。 `trident.netapp.io/podRemediationPolicy: delete`が設定されています。

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]` 。これらの注釈は、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注釈が消えないように、Kubernetes デプロイメント/レプリカセット ポッド仕様に注釈を適用します。

- `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ポッドは削除されません。
- `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ポッドが削除されます。

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、データ破損やマルチアタッチの問題が発生するリスクがあります。

トライデントノード修復 **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 (新規発行には安全ではない) としてマークされています。
- 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

- ノード回復保留中:
  - コントローラー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。
- 失敗した：
  - 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、[フォースデタッチの詳細] 。
- Kubernetes クラスターにノード ヘルス チェック (NHC) をインストールします。
  - "オペレーター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、クラスターに Operator Lifecycle Manager (OLM) をインストールします。 `operator-sdk olm install` 。
  - ノードヘルスチェックオペレーターをインストールします。 `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml` 。

> ノード障害を検出するには、以下の指定に従って別の方法を使用することもできます。[Integrating Custom Node Health Check Solutions]以下のセクションをご覧ください。

見る"ノードヘルスチェックオペレーター"詳細についてはこちらをご覧ください。

手順

1. クラスター内のワーカーノードを監視するには、 Trident名前空間に NodeHealthCheck (NHC) CR を作成します。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. ノードヘルスチェックCRを `trident`名前空間。

```
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。

その `unhealthyConditions`CR 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" 。

追加のセットアップ情報

強制デタッチを有効にしてTrident を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 (TridentNodeRemediationTemplate (TNRT) と ClusterRole) が自動的に作成されます。

TridentNodeRemediationTemplate (TNRT):

TNRT は NHC コントローラーのテンプレートとして機能し、NHC コントローラーは TNRT を使用して必要に応じて 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: {}
```

クラスターロール:

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、Trident名前空間内の TNR に NHC 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 クラスター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、メンテナンスが完了した後、仕様から pauseRequests を削除します。

制限

- 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には、カスタム ソリューションで次のことを行う必要があります。

- 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 が成功状態になったら、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ます。

独自のネームスペースで**Trident**を実行

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するためには、アプリケーション、アプリケーション管理者、ユーザ、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。

他のアプリケーションとユーザをTridentから分離するには、必ずTridentを独自のKubernetesネームスペースにインストールし(`trident`ます）。Tridentを独自のネームスペースに配置すると、Kubernetes管理者のみがTridentポッドと、名前空間CRDオブジェクトに格納されているアーティファクト（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。Tridentネームスペースへのアクセスを管理者のみに許可し、アプリケーションへのアクセスを許可する必要があり `tridentctl`ます。

ONTAP SAN バックエンドで **CHAP** 認証を使用します

Tridentでは、ONTAP SANワークロードに対してCHAPベースの認証がサポートされます（ドライバと `ontap-san-economy`ドライバを使用 `ontap-san`）。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、でCHAPユーザ名とシークレットを管理できます `tridentctl`。TridentがONTAPバックエンドでCHAPを構成する方法については、を参照してください"バックエンドにONTAP SANドライバを設定する準備をします"。

NetApp HCI および SolidFire バックエンドで CHAP 認証を使用します

ホストと NetApp HCI バックエンドと SolidFire バックエンドの間の認証を確保するために、双方向の 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をインストールすると、CHAPシークレットが管理され、それぞれのPVのCRオブジェクトに格納され `tridentvolume`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を介してNetApp HCIおよびSolidFireシステムと通信します。

Tridentで作成されるボリュームは、どのボリュームアクセスグループにも関連付けられません。

NVEおよびNAEでのTridentの使用

NetApp ONTAP は、保管データの暗号化を提供し、ディスクが盗難、返却、転用された場合に機密データを保護します。詳細については、を参照してください "NetApp Volume Encryption の設定の概要"。

- バックエンドでNAEが有効になっている場合、TridentでプロビジョニングされたすべてのボリュームでNAEが有効になります。
  - 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 `""`。
- バックエンドでNAEが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（デフォルト値）に設定されていないかぎり、Tridentでプロビジョニングされたボリュームは`false`NVE対応になり `false`ます。

NAE対応バックエンドのTridentで作成されたボリュームは、NVEまたはNAEで暗号化する必要があります。

- 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暗号化をオーバーライドし、ボリューム単位で特定の暗号化キーを使用できます。
- NAE対応バックエンドでNVE暗号化フラグをに設定する `false`と、`NAE`対応ボリュームが作成されます。`NVE`暗号化フラグをに設定して`NAE`暗号化を無効にすることはできません `false`。

- 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に設定し `true`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 のSAN構成オプション"
- "ONTAP NASの構成オプション"

Linux Unified Key Setup（LUKS；統合キーセットアップ）

Linuxユニファイドキーセットアップ（LUKS）を有効にして、Trident上のONTAP SANおよびONTAP SANエコノミー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のローテーションとボリューム拡張をサポートしています。

Tridentでは、LUKSで暗号化されたボリュームでAES-XTS-plain64暗号化およびモードが使用されます（の推奨）"NIST"。

LUKS暗号化はASA r2システムではサポートされていません。ASAr2システムの詳細については、以下を参照してください。 "ASA R2ストレージシステムの詳細" 。

作業を開始する前に

- ワーカーノードにはcryptsetup 2.1以上（3.0よりも下位）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、を参照してください "Gitlab: cryptsetup"。
- 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ています。AES-NI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、お使いのプロセッサはAES-NIをサポートしていません。AES-NIの詳細については、以下を参照してください。 "Intel：Advanced Encryption Standard Instructions（AES-NI）"。

LUKS暗号化を有効にします

ONTAP SANおよびONTAP SANエコノミーボリュームでは、Linux Unified Key Setup（LUKS；Linux統合キーセットアップ）を使用して、ボリューム単位のホスト側暗号化を有効にできます。

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のバックエンド構成オプションの詳細については、を参照してください "ONTAP のSAN構成オプション"。

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

2. 使用 `parameters.selector` LUKS暗号化を使用してストレージプールを定義する方法。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

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、バックエンドでをに(`true`設定する必要があります `luksEncryption`。このオプションを指定する `luksEncryption`と、(`false`次の例に示すように、ボリュームがLUKS準拠である(`true`かどうかがTridentに通知されます。

```
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**設定

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

```
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します

LUKSのパスフレーズをローテーションしてローテーションを確認できます。

> パスフレーズは、ボリューム、Snapshot、シークレットで参照されなくなることを確認するまで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ず、データが暗号化されたままアクセスできなくなることがあります。

このタスクについて

LUKSパスフレーズのローテーションは、ボリュームをマウントするポッドが、新しいLUKSパスフレーズの指定後に作成されたときに行われます。新しいPOD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。

- ボリュームのパスフレーズがシークレットでアクティブなパスフレーズと一致しない場合、ローテーションが実行されます。
- ボリュームのパスフレーズがシークレットのアクティブなパスフレーズと一致する場合は、を参照してください `previous-luks-passphrase` パラメータは無視されます。

手順

1. を追加します `node-publish-secret-name` および `node-publish-secret-namespace` StorageClassパラメータ。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Snapshotの既存のパスフレーズを特定します。

   ボリューム

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

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

   スナップショット

```
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前のパスフレーズを指定します。確認します `previous-luke-passphrase-name` および `previous-luks-passphrase` 前のパスフレーズと同じにします。

```
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。

ボリューム

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

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

スナップショット

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

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

結果

パスフレーズは、ボリュームとSnapshotに新しいパスフレーズのみが返されたときに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などです `luksPassphraseNames: ["B", "A"]`回転が不完全です。回転を完了するために、新しいポッドをトリガできます。

ボリュームの拡張を有効にします

LUKS暗号化ボリューム上でボリューム拡張を有効にできます。

手順

1. を有効にします `CSINodeExpandSecret` 機能ゲート（ベータ1.25+）。を参照してください "Kubernetes 1.25：CSIボリュームのノードベースの拡張にシークレットを使用します" を参照してください。
2. を追加します `node-expand-secret-name` および `node-expand-secret-namespace` StorageClass パラメータ。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

ストレージのオンライン拡張を開始すると、ドライバに適切なクレデンシャルが渡されます。

Kerberos転送中暗号化

Kerberos転送中暗号化を使用すると、管理対象クラスタとストレージバックエンドの間のトラフィックの暗号化を有効にすることで、データアクセスセキュリティを強化できます。

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およびアップストリームのKubernetesクラスタからオンプレミスのONTAPボリュームへのNFSv3 / NFSv4接続でKerberos暗号化をサポートしています。

作成、削除、サイズ変更、スナップショット、クローン、 読み取り専用のクローンを作成し、NFS暗号化を使用するボリュームをインポートします。

オンプレミスの**ONTAP**ボリュームでの転送中**Kerberos**暗号化の設定

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

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

作業を開始する前に

- ユーティリティにアクセスできることを確認し `tridentctl` ます。
- 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ます。
- 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確認しておきます。
- NFSボリュームのKerberos暗号化をサポートするようにONTAP Storage VMを準備しておく必要があります。手順については、を参照してください "データLIFでKerberosを有効にする" 。

- Kerberos暗号化で使用するNFSv4ボリュームが正しく設定されていることを確認します。のNetApp NFSv4ドメインの設定セクション（13ページ）を参照してください "『NetApp NFSv4 Enhancements and Best Practices Guide』"。

ONTAPエクスポートポリシーを追加または変更する

既存のONTAPエクスポートポリシーにルールを追加するか、ONTAP Storage VMのルートボリュームおよびアップストリームのKubernetesクラスタと共有するONTAPボリュームに対してKerberos暗号化をサポートする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新規に作成するエクスポートポリシーでは、次のアクセスプロトコルとアクセス権限がサポートされている必要があります。

アクセスプロトコル

NFS、NFSv3、およびNFSv4の各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設定します。

詳細を確認

ボリュームのニーズに応じて、次の3つのバージョンのいずれかを設定できます。

- * Kerberos 5 *-（認証と暗号化）
- * Kerberos 5i *-（ID保護による認証と暗号化）
- * Kerberos 5p *-（IDおよびプライバシー保護による認証および暗号化）

適切なアクセス権限を指定してONTAPエクスポートポリシールールを設定します。たとえば、Kerberos 5i暗号化とKerberos 5p暗号化が混在しているNFSボリュームをクラスタにマウントする場合は、次のアクセス設定を使用します。

を入力します	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および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を作成する"
- "エクスポートポリシーにルールを追加する"

ストレージバックエンドの作成

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。

このタスクについて

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場合は、パラメータを使用して次の3つのバージョンのKerberos暗号化のいずれかを指定でき `spec.nfsMountOptions` ます。

- `spec.nfsMountOptions: sec=krb5` （認証と暗号化）
- `spec.nfsMountOptions: sec=krb5i` （ID保護による認証と暗号化）

- `spec.nfsMountOptions: sec=krb5p` （IDおよびプライバシー保護による認証および暗号化）

Kerberosレベルを1つだけ指定してください。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は、最初のオプションのみが使用されます。

手順

1. 管理対象クラスタで、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<>の値は、環境の情報で置き換えます。

```
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、バックエンドの設定に何か問題があります。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
```

構成ファイルで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。

ストレージクラスを作成して、Kerberos暗号化を使用してボリュームをプロビジョニングできます。

このタスクについて

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

- `mountOptions: sec=krb5` （認証と暗号化）
- `mountOptions: sec=krb5i` （ID保護による認証と暗号化）
- `mountOptions: sec=krb5p` （IDおよびプライバシー保護による認証および暗号化）

Kerberosレベルを1つだけ指定してください。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は、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した。手順については、を参照してください "ボリュームをプロビジョニングする"。

Azure NetApp Filesボリュームでの転送中Kerberos暗号化の設定

管理対象クラスタと単一の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の間のストレージトラフィックに対してKerberos暗号化を有効にすることができます。

作業を開始する前に

- 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ます。
- ユーティリティにアクセスできることを確認し `tridentctl` ます。
- 要件を確認し、の手順に従って、Kerberos暗号化用のAzure NetApp Filesストレージバックエンドの準備が完了していることを確認します。 "Azure NetApp Files のドキュメント"
- Kerberos暗号化で使用するNFSv4ボリュームが正しく設定されていることを確認します。のNetApp NFSv4ドメインの設定セクション（13ページ）を参照してください "『NetApp NFSv4 Enhancements and Best Practices Guide』"。

ストレージバックエンドの作成

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。

このタスクについて

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場合は、次の2つのレベルのいずれかで適用するように定義できます。

- フィールドを使用した* storage backend level * `spec.kerberos`
- フィールドを使用した*仮想プールレベル* `spec.storage.kerberos`

仮想プールレベルで構成を定義する場合、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、次の3つのバージョンのKerberos暗号化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保護による認証と暗号化）
- `kerberos: sec=krb5p` （IDおよびプライバシー保護による認証および暗号化）

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<>の値は、環境の情報で置き換えます。

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、バックエンドの設定に何か問題があります。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
```

構成ファイルで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。

ストレージクラスを作成して、Kerberos暗号化を使用してボリュームをプロビジョニングできます。

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した。手順については、を参照してください "ボリュームをプロビジョニングする"。