Cisco HyperFlex Systems リリース 1.2(x) 管理者ガイド（Kubernetes 用）

初版：2021 年 11 月 10 日

最終更新：2023 年 6 月 5 日

シスコシステムズ合同会社
〒107-6227 東京都港区赤坂9-7-1 ミッドタウン・タワー
http://www.cisco.com/jp
お問い合わせ先：シスコ コンタクトセンター
0120-092-255 （フリーコール、携帯・PHS含む）
電話受付時間：平日 10:00～12:00、13:00～17:00
http://www.cisco.com/jp/go/contactcenter/

目 次

Full Cisco Trademarks with Software License ?

はじめに：

通信、サービス、偏向のない言語、およびその他の情報 v

第 1 章

新機能および変更された機能 1

新機能および変更された機能に関する情報 1

第 2 章

Cisco HyperFlex Kubernetes サポート 3

サポートの概要 3

Cisco HyperFlex CSI 相互運用性メトリックス 4

第 3 章

Cisco HyperFlex コンテナ ストレージ インターフェイス (CSI)（Kubernetes 用） 5

Cisco HyperFlex Kubernetes CSI について 5

Cisco HyperFlex CSI コンポーネント 6

第 4 章

Cisco HyperFlex CSI インターフェイス（Kubernetes 用）の構成 9

前提条件 9

管理者ホスト 10

Cisco HyperFlex CSI Integration for Kubernetes のインストール 10

Cisco HyperFlex CSI バンドルのダウンロード 10

Cisco HyperFlex CSI バンドルのオープンと抽出 11

Cisco HyperFlex CSI コンテナ イメージのアップロード 12

各 Kubernetes ワーカー ノードに直接 Cisco HyperFlex CSI コンテナ イメージを手動でインポート 12

hxcsi-setup ユーティリティを使用した HXCSI の展開 13

Cisco HyperFlex CSI コンポーネントを使用した HXCSI の展開 16
HXCSI サンプルポッド 17
Cisco HyperFlex CSI ストレージ クラスの作成 18
Verifying Cisco HyperFlex CSI ストレージ クラスの作成 20

第 5 章 **Cisco HyperFlex CSI によるステートフル アプリケーションの展開** 21
Cisco HyperFlex CSI によるステートフル アプリケーションの展開の前提条件 21
管理者ホスト 21
ステートフル アプリケーションの展開 22
永続的なボリューム要求の作成 22
ステートフル Kubernetes ワークロードの展開 23

第 6 章 **ソフトウェア暗号化ボリューム、スナップショットの作成、および CHAP の使用** 25
ソフトウェア暗号化ボリュームの作成 25
ソフトウェア暗号化ボリュームを作成するための前提条件 25
ソフトウェア暗号化ボリュームの作成 25
Creating Volume Snapshots 26
ボリューム スナップショットを作成するための前提条件 26
ボリュームから HXCSI スナップショットを作成 27
ボリュームの CHAP 保護の使用 29
ボリュームに CHAP を使用するための前提条件 30
ボリュームでの CHAP 保護の有効化 30

第 7 章 **トラブルシューティング** 33
トラブルシューティング 33
HXCSI ポッドの展開時の ImagePullBackOff ステータスエラー 33
ボリューム削除の失敗 34
ノード削除中の ContainerCreating 状態のアプリケーション ポッド 34
終了中または ContainerCreating 状態のアプリケーション ポッド スタッド 35
削除されたポッドが同じノードに戻るようにスケジュールされている 36

通信、サービス、偏向のない言語、およびその他の情報

- シスコからタイムリーな関連情報を受け取るには、Cisco Profile Manager でサインアップしてください。
- 重要な技術によりビジネスに必要な影響を与えるには、Cisco Services にアクセスしてください。
- サービス リクエストを送信するには、Cisco Support にアクセスしてください。
- 安全で検証済みのエンタープライズクラスのアプリケーション、製品、ソリューション、およびサービスを探して参照するには、Cisco Marketplace にアクセスしてください。
- 一般的なネットワーキング、トレーニング、認定関連の出版物を入手するには、Cisco Press にアクセスしてください。
- 特定の製品または製品ファミリの保証情報を探すには、Cisco Warranty Finder にアクセスしてください。

マニュアルに関するフィードバック

シスコのテクニカルドキュメントに関するフィードバックを提供するには、それぞれのオンラインドキュメントの右側のペインにあるフィードバックフォームを使用してください。

Cisco バグ検索ツール

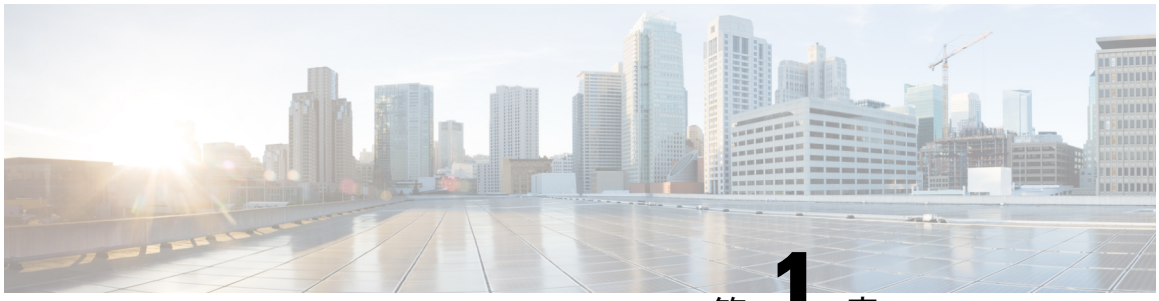
Cisco Bug Search Tool（BST）は、シスコ製品とソフトウェアの障害と脆弱性の包括的なリストを管理する Cisco バグ追跡システムへのゲートウェイとして機能する、Web ベースのツールです。BST は、製品とソフトウェアに関する詳細な障害情報を提供します。

偏向のない言語

この製品のマニュアルセットは、偏向のない言語を使用するように配慮されています。このドキュメントセットでの偏向のない言語とは、年齢、障害、性別、人種的アイデンティティ、民族的アイデンティティ、性的指向、社会経済的地位、およびインターセクショナリティに基づく差別を意味しない言語として定義されています。製品ソフトウェアのユーザーインターフェ

通信、サービス、偏向のない言語、およびその他の情報

イスにハードコードされている言語、基準ドキュメントに基づいて使用されている言語、または参照されているサードパーティ製品で使用されている言語によりドキュメントに例外が存在する場合があります。



第 1 章

新機能および変更された機能

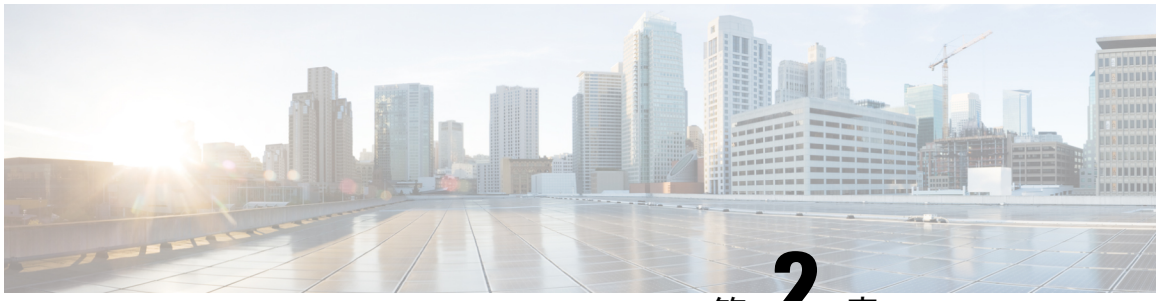
• 新機能および変更された機能に関する情報 （1 ページ）

新機能および変更された機能に関する情報

この表には、『Cisco HyperFlex システム管理者ガイド、リリース 1.2(x)』の新機能と変更された機能、およびその説明がどこに出ているかがまとめられています。

特長	説明	リリース/追加日	参照先
Cisco HyperFlex Systems、HXCSI リリース 1.2(3a) アドミニストレーションガイド（Kubernetes 用）	HXCSI 1.2(3a) の新機能を有効にして構成する方法の説明、前提条件、および手順を追加しました。	HXCSI リリース 1.2(3a)	• Cisco HyperFlex CSI コンポーネント （6 ページ） • ソフトウェア暗号化ボリュームの作成 （25 ページ） • Creating Volume Snapshots （26 ページ） • ボリュームの CHAP 保護の使用 （29 ページ）
HyperFlex CSI 相互運用性メトリックスにアップデートします	HXDP 5.0(1b) のバージョン アップデート	2022 年 1 月 27 日	Cisco HyperFlex CSI 相互運用性メトリックス （4 ページ）

特長	説明	リリース/追加日	参照先
Cisco HyperFlex Systems、HXCSI リリース 1.2(2a) アドミニストレーションガイド（Kubernetes 用）	Cisco HyperFlex Systems、HXCSI リリース 1.2(2a) アドミニストレーションガイド（Kubernetes 用）の最初のリリース	HXCSI リリース 1.2(2a)	本書です。



第 2 章

Cisco HyperFlex Kubernetes サポート

- サポートの概要 （3 ページ）
- Cisco HyperFlex CSI 相互運用性メトリックス （4 ページ）

サポートの概要

Cisco HyperFlex での Kubernetes のバージョンまたはディストリビューションのサポートを決定する際に考慮する必要がある 2 つの主要なコンポーネントがあります。

- Cisco HyperFlex での Kubernetes バージョンまたはディストリビューションのサポート。
- 特定の Kubernetes バージョンまたはディストリビューションとの Cisco HyperFlex Container Storage Interface（CSI）ストレージ統合のサポート。

（注） Kubernetes ストレージ特別利益団体（K8 SIG コミュニティ）は、ストレッチ クラスタではサポートされていません。

一般に、Cisco HyperFlex は Kubernetes のすべてのバージョンまたはディストリビューションをサポートしますが、Kubernetes の Cisco HyperFlex CSI ストレージ統合でテストされ、推奨されるバージョンとディストリビューションの特定のサブセットがあります。また、HyperFlex CSI ストレージ統合を使用せずに Cisco HyperFlex で Kubernetes およびコンテナベースのワークロードを実行することもできますが、永続ストレージを必要とするステートフルな Kubernetes ベースのアプリケーションおよびサービスを実行する場合は、ネイティブ機能を活用することを強く推奨します。

Cisco HyperFlex Storage Integration for Kubernetes により、CiscoHyperFlex が CiscoHyperFlex で実行されているステートフルな Kubernetes ワークロードに永続ストレージを動的に提供することを許可します。この統合により、Cisco HyperFlex が永続ボリュームのオブジェクト ライフサイクル全体のオーケストレーションをオフロードおよび管理できるようになる一方で、最終的には開発者やユーザーがそれを、標準の Kubernetes 永続ボリューム クレーム オブジェクトを通じて駆動（開始）できるようになります。開発者とユーザーは、開発者やユーザーの観点か

らは追加の管理オーバーヘッドを発生させずに、Kubernetes の永続ストレージのニーズに Cisco HyperFlex を活用するという利点を得ることができます。

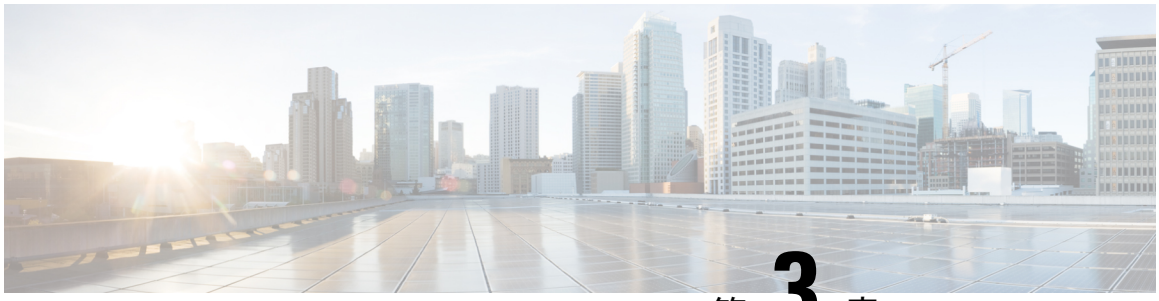
Cisco HyperFlex CSI 相互運用性メトリックス

Cisco HyperFlex CSI および Kubernetes プラットフォームのバージョンとディストリビューションの相互運用性：

HXDP バージョン HXCSI バージョン[1]	Kubernetes のバージョン	シスコ認定 Anthos バージョン[2]	シスコ認定 OpenShift バージョン[3]
HXDP 4.5(1a)	1.18.2	1.7.2	-
HXDP 4.5(1a) HXCSI 1.2(1a)	1.18.2、1.19.8	-	-
HXDP 5.0(1a) HXCSI 1.2(2a)	1.19.8	1.9.2	4.8、4.9
HXDP 5.0(1b) HXCSI 1.2(2a)	1.19.16	-	-
HXDP 5.0(2a) HXCSI 1.2(3a)	1.22.7	-	-

[1] この表は、特定のバージョンで完了した検証と、以前のコンポーネントと互換性のある HXDP リリースを示しています。
[2] シスコは Anthos Ready プラットフォーム パートナーです。インストールされている Kubernetes のバージョンについては、Anthos のドキュメントを参照してください。
[3] シスコは認定 RedHat パートナーです。インストールされている Kubernetes のバージョンについては、OpenShift のドキュメントを参照してください。

（注） HXCSI は、Ubuntu 18.04 上の open-iscsi バージョン 2.0.874-5ubuntu2.10 で認定されています。



第 3 章

Cisco HyperFlex コンテナ ストレージ インターフェイス (CSI)（Kubernetes 用）

- Cisco HyperFlex Kubernetes CSI について （5 ページ）
- Cisco HyperFlex CSI コンポーネント （6 ページ）

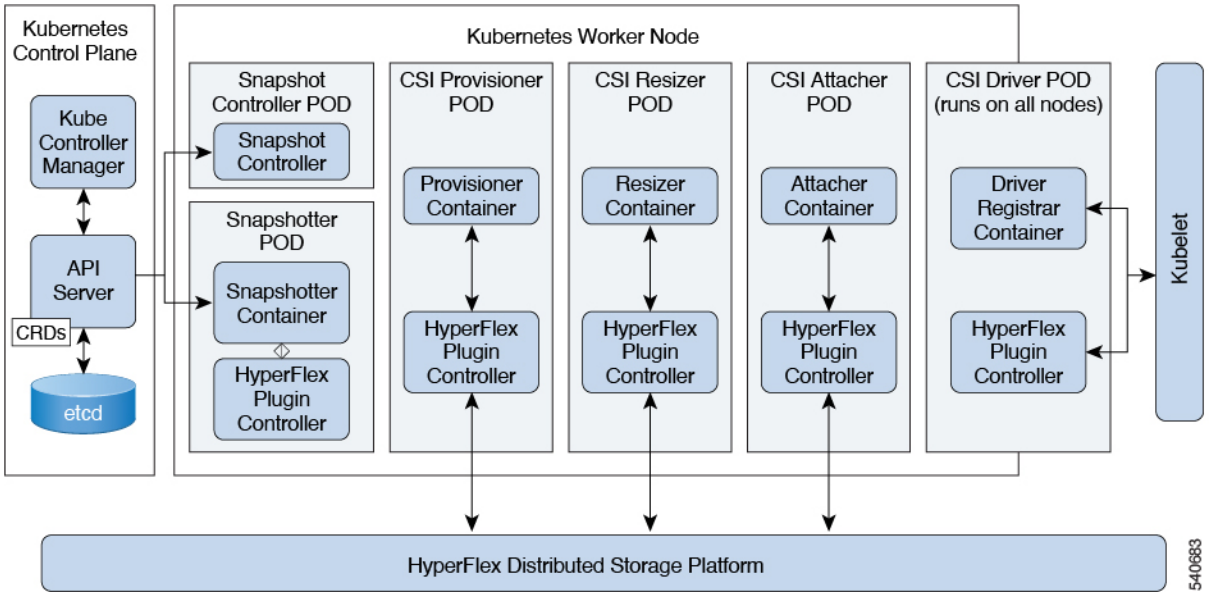
Cisco HyperFlex Kubernetes CSI について

Cisco HyperFlex Container Storage Interface（CSI）は、永続的なボリューム要求やストレージクラスなどの標準的なKubernetesの基本要素を介して展開および使用される、アウトオブツリーコンテナ ベースの Kubernetes ストレージ統合です。Cisco HyperFlex CSI は次の機能をサポートしています。

- ボリュームの動的な作成と削除
- ダイナミック ボリュームのアタッチとデタッチ
- アクセス サポートのブロック
- ボリュームの複製（送信元ボリュームが同じデータストアからの場合）
- さまざまなファイル システム（Ext4、Ext3、XFS）での PV サポート
- CSI 仕様ごとのボリューム領域統計レポート
- ブロック モードのみのマルチライタ サポート（ReadWriteMany）。
- Kubernetes 1.14、1.19 サポート
- 専用イニシエータ グループを使用した Kubernetes クラスタ マルチテナンシー ターゲット マスキング
- CSI 1.2 仕様 API のサポート
- ブロック モードボリューム、ext3、ext4、および xfs ファイル システム ボリュームのボリューム サイズ変更サポート。（拡張）
- ヘルム チャートによる CSI プラグインのインストールとアップグレード

Cisco HyperFlex CSI コンポーネント

Cisco HyperFlex CSI インテグレーションは、ターゲット Kubernetes クラスタの最上位にコンテナとして展開されます。次の図は、Cisco HyperFlex CSI 展開のさまざまなコンポーネントと、それらが相互にどのように相互作用するかを示しています。



展開には、次のポッドが含まれます。

csi-attacher-hxcsi

- **タイプ:** StatefulSet
- **インスタンス数**：Kubernetes クラスタごとに 2 つ。
- **目的**：CSI で必要ですが、シスコの導入では現在使用されていません。

csi-provisioner-hxcsi

- **タイプ:** StatefulSet
- **インスタンス数**：Kubernetes クラスタごとに 2 つ。
- **目的**：Kubernetes Persistent Volume Claim オブジェクトを監視し、Kubernetes CSI仕様の一部として CreateVolume および DeleteVolume 操作をトリガーします。

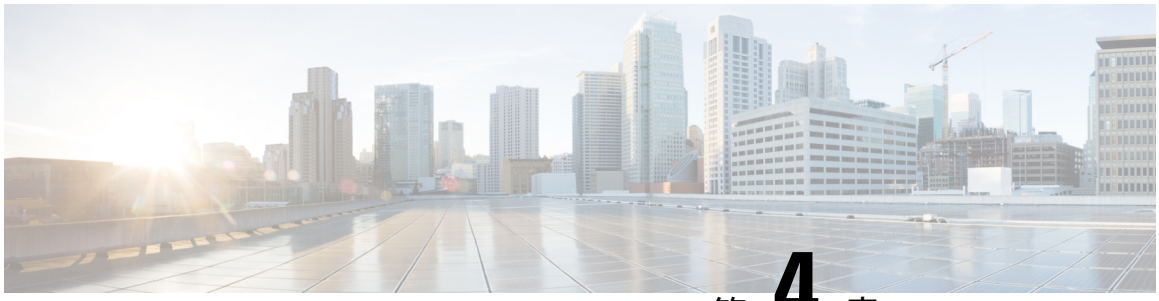
csi-nodeplugin-hxcsi

- **タイプ**：DaemonSet
- **インスタンス数**：Kubernetes ワーカー ノードごとに 1 つ

- **目的**：Kubernetes ワーカー ノードでプロビジョニングされた HyperFlex iSCSI LUN の検出とフォーマット。NodePublish/NodeUnpublish Volume API を Kubernetes CSI 仕様の一部として実装します。

csi-resizer-hxcsi

- **タイプ:** StatefulSet
- **インスタンス数**：Kubernetes クラスタごとに 2 つ。
- **目的**：Kubernetes Persistent Volume Claim オブジェクトを監視し、Kubernetes CSI 仕様の一部として ControllerExpandVolume および NodeExpandVolume 操作をトリガーします。



第 4 章

Cisco HyperFlex CSI インターフェイス（Kubernetes 用）の構成

- 前提条件 （9 ページ）
- 管理者ホスト （10 ページ）
- Cisco HyperFlex CSI Integration for Kubernetes のインストール （10 ページ）
- Verifying Cisco HyperFlex CSI ストレージ クラスの作成 （20 ページ）

前提条件

次の前提条件は、Cisco HyperFlex CSI インテグレーションの構成の前に満たす必要があります。

HyperFlexクラスタ上：

- Cisco HyperFlex クラスタがインストールされ、HX 5.0(1a) 以降を実行している。
- インストールする前に、HX ConnectでiSCSIネットワークを設定します。iSCSIネットワークの設定の詳細については、『』『Cisco HyperFlex Data Platform Administration Guide、Release 5.0』を参照してください。

Kubernetesクラスタ上：

- HXCSI で進める前に、すべての Kubernetes ノードに `2.0.874-5ubuntu2.10` 以降のバージョンの `open-iscsi` パッケージがインストールされていることを確認します。これを行うには、`$iscsid-version` コマンドを実行します。

  `open-iscsi バージョン 2.0.874-5ubuntu2.10` をインストールするには、`apt-get install -y open-iscsi=2.0.874-5ubuntu2.10` コマンドを実行します。

- 各 Kubernetesノードに HX iSCSI ネットワーク上の専用インターフェイスがあるか、またはHX iSCSI ネットワークへのルーティング可能なアクセスがあることを確認します。
- `iscsid` がシステムのリブート時に開始されるようにするには、次のコマンドを実行します。

  ```
  sudo systemctl enable iscsid
  ```

`iscsid` ステータスが表示されます（例）。

```
$ sudo systemctl status iscsid

iscsid.service：iSCSIイニシエータ デーモン（iscid）

Loaded: loaded (/lib/systemd/system/iscid.service; enabled; vendor preset: enabled
```

- 各 Kubernetes プライマリ（「マスター」とも呼ばれる）ホストシステムに「`/etc/kubernetes/manifests/kube-controller-manager.yaml`」ファイル（`--disable-attach-detach-reconcile-sync=true` を含む）が含まれていることを確認します。
- ファイルの `-command` セクションに次のテキストを追加します。
  `--disable-attach-detach-reconcile-sync=true`

管理者ホスト

このガイドでは、管理者ホストは kubectl コマンドなどを Kubernetes クラスタに対して実行するための Linux ベースのシステムのことです。これは通常、Kubernetes クラスタの一部ではない別のシステム（VM）ですが、別のシステム（VM）をインストール/管理する必要がない場合は、管理者ホストとして Kubernetes ノードの 1 つを使用できます。

Cisco HyperFlex CSI Integration for Kubernetes のインストール

Cisco HyperFlex CSI Integration をインストールするには、次の手順を記載されている順序で実行します。

Cisco HyperFlex CSI バンドルのダウンロード

Cisco HyperFlex CSI バンドル（ファイル）をダウンロードするには、次の手順を実行します。

ステップ 1 https://software.cisco.com に移動する

ステップ 2 Cisco ID のクレデンシャルを使用してログインします。

ステップ 3 **[ダウンロードとアップグレード（Download and Upgrade）]** セクションで、**[ソフトウェアのダウンロード（Software Download）]** を選択します。

ステップ 4 **[製品の選択（Select a Product）]** 検索フィールドに、**`HyperFlex HX Data Platform`** と入力し、**Enter** をクリックします。

ステップ 5 左側の [リリース（Release）] ナビゲーション ウィンドウを使用して、クラスタで実行されている HyperFlex データ プラットフォーム ソフトウェアのバージョンを選択します。

Cisco HyperFlex Data Platform リリース 4.5(x) 以降では、Cisco HyperFlex CSI インテグレーションが必要です。

ステップ 6 メインナビゲーション ペインで、「Cisco HyperFlex Kubernetes Container Storage Interface（HX-CSI）bundle（tar.gz）」ファイルをローカル マシンにダウンロードします。

以降、Cisco HyperFlex Kubernetes Container Storage Interface（HX-CSI）バンドル（tar.gz）ファイルを「Cisco HyperFlex CSI バンドル」と呼びます。

ステップ 7 管理者ホストで、`hxcsi` という名前の新しいディレクトリを作成します。

例：

```
administrator-host:~$ mkdir hxcsi
```

ステップ 8 セキュア コピー（scp）またはその他の優先ファイル転送方式を使用して、ダウンロードしたCisco HyperFlex CSI バンドルをローカル マシンから管理者ホストの「hxcsi」ディレクトリに転送（移動またはコピー）します。結果は次のようになります。

例：

```
administrator-host:hxcsi$ ls

hxcsi-1.2.2a-626.tar.gz
```

次のタスク

Cisco HyperFlex CSI バンドルのオープンと抽出

Cisco HyperFlex CSI バンドルのオープンと抽出

Cisco HyperFlex CSI バンドルを開くには、次の手順を実行します。

始める前に

Cisco HyperFlex CSI バンドルをダウンロードします。

`tar` コマンドを使用して、HyperFlex CSI バンドル（.tar.gz ファイル）をアーカイブ解除します。

例：

```
administrator-host:hxcsi$ tar -xf hxcsi-1.2.2a-626.tar.gz
```

完了すると、次のディレクトリ構造が存在します。

- **サンプル（ディレクトリ）**：HXCSIインテグレーションを使用するためのサンプルYAMLファイルが含まれています。

- **イメージ（ディレクトリ）**：HXCSI インテグレーション用の HXCSI docker コンテナ イメージが含まれます。これには、Provisioner、Attacher、Node-driver、および Resizer の基本 CSI イメージも含まれます。
- **setup（ディレクトリ）**：HXCSI 統合を展開するためのセットアップ スクリプトが含まれています。
- **support（ディレクトリ）**：デバッグに役立つログを収集するためのスクリプトが含まれています。
- **hxcsi-1.2.1.tgz（ファイル）**：これは、このリリースの HXCSI の HELM チャートパッケージ です。

例

```
administrator-host:hxcsi$ ls -l
total 133196
-rw-r--r--  1 ubuntu ubuntu      6791 May 10 11:23 hxcsi-1.2.1.tgz
drwxr-xr-x  2 ubuntu ubuntu      4096 May 10 11:23 support
drwxr-xr-x  2 ubuntu ubuntu      4096 May 10 11:23 setup
drwxr-xr-x  2 ubuntu ubuntu      4096 May 10 11:23 images
drwxr-xr-x 11 ubuntu ubuntu      4096 May 10 11:23 examples
```

次のタスク

Cisco HyperFlex CSI コンテナ イメージのアップロード

Cisco HyperFlex CSI コンテナ イメージのアップロード

Cisco HyperFlex CSI インテグレーション コンポーネントは、Cisco HyperFlex CSI バンドルの「images」ディレクトリで提供される単一のコンテナ イメージから展開されます。hxcsi コンテナ イメージは、同じディレクトリ内の他の 4 つのベース CSI イメージを活用します。コンテナ イメージを展開する前に、Kubernetes クラスタ ワーカー ノードで実行されている Docker にアクセス可能な場所にコンテナ イメージを移動します。

各 Kubernetes ワーカー ノードに直接 Cisco HyperFlex CSI コンテナ イメージを手動でインポート

Cisco HyperFlex CSI コンテナ イメージを各 Kubernetes ワーカー ノードに直接追加するには、次の手順を実行します。

始める前に

Cisco HyperFlex CSI バンドルを開きます。

ステップ 1 管理者ホストで、「images」ディレクトリにある Cisco HyperFlex CSI コンテナ イメージ（.tar）ファイルを各 Kubernetes ワーカー ノードの /tmp ディレクトリにコピーします。

例：

```
administrator-host:hxcsi$ scp ./images/hxcsi-1.2.3a-659.tar k8s-worker1:/tmp

administrator-host:hxcsi$ scp ./images/hxcsi-1.2.3a-659.tar k8s-worker2:/tmp

administrator-host:hxcsi$ scp ./images/hxcsi-1.2.3a-659.tar k8s-workerN:/tmp
```

ステップ 2 他の基本 CSI コンテナイメージファイルを各 Kubernetes ノードにコピーします。

`csi-attacher-3.0.2-cisco1.tar`、`csi-node-driver-registrar-2.0.1-cisco1.tar`、`csi-resizer-1.0.1-cisco1.tar`、`csi-provisioner-2.0.4-cisco1.tar`

ステップ 3 各 Kubernetes ワーカーノードで、`docker load --input` コマンドを使用して Cisco HyperFlex CSI コンテナイメージをロードします。

例：

```
k8s-worker1:/tmp# docker load –input ./hxcsi-1.2.3a-659.tar
Loaded image: hxcsi:hxcsi-1.2.3a-659

k8s-worker2:/tmp# docker load –input ./hxcsi-1.2.3a-659.tar Loaded image: hxcsi:hxcsi-1.2.2a-626

k8s-workerN:/tmp# docker load –input ./hxcsi-1.2.3a-659.tar Loaded image: hxcsi:hxcsi-1.2.2a-626
```

ステップ 4 Docker によって他の基本 CSI コンテナイメージファイルが各 Kubernetes ノードにロードされます。

`csi-attacher-3.2.1-cisco1.tar`、`csi-node-driver-registrar-2.2.0-cisco1.tar`、`csi-resizer-1.2.0-cisco1.tar`、`csi-provisioner-2.2.1-cisco1.tar`

次のタスク

Cisco HyperFlex CSI をインストールします。

hxcsi-setup ユーティリティを使用した HXCSI の展開

Cisco HyperFlex CSI インテグレーションを展開するには、`hxcsi-setup` スクリプトを実行する必要があります。`hxcsi-setup` スクリプトは「setup」ディレクトリにあり、必要な YAML ファイルまたはヘルム チャートを自動的に生成して、Kubernetes クラスタに適用（送信）して、Cisco HyperFlex CSI コンポーネントを展開します。

次の表に、`hxcsi-setup` コマンドで指定できるパラメータを示します。

表 *1 : hxcsi-setup* のパラメータ

パラメータ	必須またはオプション	説明
`client-id`	オプション	テナントのクライアント ID。 （注） 複数の Kubernetes クラスタを作成して、同じ HX クラスタからストレージを要求できます。「clientId」パラメータは、これらのクライアント/テナントそれぞれのストレージ割り当ての分離に役立ちます。
`-cluster-name`	Required	この特定の Kubernetes クラスタを一意に識別する名前を指定します。
`-helm-chart`	オプション	ヘルムインストールのヘルム チャートを生成します（デフォルトは YAML ファイルを生成します）
`-hx-csi-image string`	Required	Cisco HyperFlex CSI コンテナ イメージの名前と場所。これにより、Cisco HyperFlex CSI コンテナ イメージを取得する場所が Kubernetes に通知されます。 4
`-iscsi-url string`	Required	HyperFlex クラスタの eth-iscsi1:0 インターフェイスの HyperFlex iSCSI クラスタ IP アドレス。詳細については、前提条件 （9 ページ）を参照してください。
`-output-dir string`	オプション	出力ディレクトリ（デフォルトは「./hxcsi-deploy/」）
`-password string`	Required （最初に入力しない場合は、プロンプトが表示されます）	HX クラスタ API へのパスワード
`-token string`	オプション	サービス認証トークン。hxcsi-setup を呼び出す前に、アウトオブバンドでトークンを作成できます。

パラメータ	必須またはオプション	説明
`-url string`	Required	クラスタ管理IPアドレス。このIPはボリューム プロビジョニングとして使用されます。
`-username string`	Required	HX クラスタ API のユーザ名（つまり、「admin」）
`-docker-registry`	オプション	Docker レジストリ名（例: mydockerhub.com/hx-docker）

[4] Cisco HyperFlex CSI コンテナイメージが各 Kubernetes ワーカーノードの Docker に直接インポートされた場合、このパラメータの形式は **<repository_name>:<tag>** のように入力する必要があります。

始める前に

Cisco HyperFlex CSI コンテナイメージとベース CSI コンテナイメージをアップロードします。

管理者ホストで、「setup」ディレクトリで `hxcsi-setup` コマンドを使用して、必要な Cisco HyperFlex CSI 展開ファイルを作成します。

（注） **hxcsi-setup** コマンドを実行する前に、url および iscsi-url パラメータで指定された IP が Kubernetes ノードから到達可能であることを確認します。

例：

すべてのイメージが dockerhub にアップロードされた場合：次の例は、すべてのイメージが各 Kubernetes ノードからアクセス可能な dockerhub にアップロードされた場合を示しています。イメージ名は `hxcsi`、タグ名は `hxcsi-1.2.3a-659` です。

```
administrator-host:hxcsi$ ./setup/hxcsi-setup -cluster-name demo-hxcsi -clientId
 demo-client1 -hx-csi-image  hxcsi-1.2.3a-659 -iscsi-url 10.2.17.18 -url 10.2.17.13
-username admin -docker-registry dockerhub.cisco.com/hx-dev-docker

password for [admin] at [10.2.17.13]: *******
wrote config to hxcsi-deploy/hxcsi-config.yaml
wrote config to hxcsi-deploy/csi-attacher-hxcsi.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-hxcsi.yaml
wrote config to hxcsi-deploy/csi-provisioner-hxcsi.yaml
wrote config to hxcsi-deploy/csi-attacher-rbac.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-rbac.yaml
wrote config to hxcsi-deploy/csi-provisioner-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-hxcsi.yaml
```

例：

すべてのイメージが各ノードにローカルにドッカーアップロードされている場合：次に、各ノードにアップロードされている Cisco HyperFlex CSI コンテナイメージの展開例を示します。イメージ名は `hxcsi`、タグ名は `hxcsi-1.2.3a-659` です。この使用例は、中央の場所からイメージを取得するために使用される dockerhub がない場合に適用されます。

```
administrator-host:hxcsi$ ./setup/hxcsi-setup -cluster-name demo-hxcsi -clientId demo-client1
-hx-csi-image hxcsi:hxcsi-1.2.3a-659  -iscsi-url 10.2.17.18 -url 10.2.17.13 -username admin

password for [admin] at [10.2.17.13]: *******
wrote config to hxcsi-deploy/hxcsi-config.yaml
wrote config to hxcsi-deploy/csi-attacher-hxcsi.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-hxcsi.yaml
wrote config to hxcsi-deploy/csi-provisioner-hxcsi.yaml
wrote config to hxcsi-deploy/csi-attacher-rbac.yaml
wrote config to hxcsi-deploy/csi-nodeplugin-rbac.yaml
wrote config to hxcsi-deploy/csi-provisioner-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-rbac.yaml
wrote config to hxcsi-deploy/csi-resizer-hxcsi.yaml
```

次のタスク

Cisco HyperFlex CSI コンポーネントの展開

Cisco HyperFlex CSI コンポーネントを使用した HXCSI の展開

hxcsi-setup スクリプトを実行し、Cisco HyperFlex CSI 展開ファイルを生成すると、新しい「hxcsi-deploy」ディレクトリが管理者ホストに作成されます。

```
root@administrator-host:hxcsi$ ls
examples  hxcsi-1.2.2a-626.tar.gz  hxcsi-1.2.1.tgz  hxcsi-deploy  images  setup  support
```

始める前に

Cisco HyperFlex CSI 展開ファイルを作成します。

ステップ 1 管理者ホストで `kubectl create -f` コマンドを使用して、Cisco HyperFlex CSI コンポーネントを展開します。

例：

```
administrator-host:hxcsi$ kubectl create -f ./hxcsi-deploy/
service/csi-attacher-hxcsi created
statefulset.apps/csi-attacher-hxcsi created
serviceaccount/csi-attacher created
clusterrole.rbac.authorization.k8s.io/external-attacher-runner created
clusterrolebinding.rbac.authorization.k8s.io/csi-attacher-role created
daemonset.apps/csi-nodeplugin-hxcsi created
serviceaccount/csi-nodeplugin created
clusterrole.rbac.authorization.k8s.io/csi-nodeplugin created
clusterrolebinding.rbac.authorization.k8s.io/csi-nodeplugin created
service/csi-provisioner-hxcsi created
statefulset.apps/csi-provisioner-hxcsi created
serviceaccount/csi-provisioner created
clusterrole.rbac.authorization.k8s.io/external-provisioner-runner created
clusterrolebinding.rbac.authorization.k8s.io/csi-provisioner-role created
deployment.apps/csi-resizer-hxcsi created
serviceaccount/csi-resizer created
clusterrole.rbac.authorization.k8s.io/external-resizer-runner created
clusterrolebinding.rbac.authorization.k8s.io/csi-resizer-role created
role.rbac.authorization.k8s.io/external-resizer-cfg created
```

```
rolebinding.rbac.authorization.k8s.io/csi-resizer-role-cfg created
secret/hxcsitoken created
configmap/hxcsi-config created
```

ステップ 2 管理者ホストで `kubectl get pods` コマンドを使用して、HXCSI コンポーネントが展開され、ステータスが `[実行中（Running）]` であることを確認します。

例：

（注） 各 Kubernetes ワーカー ノードに対して、`csi-attacher-hxcsi` ポッドの 2 つのインスタンス、`csi-provisioner-hxcsi` ポッドの2つのインスタンス、`csi-resizer-hxcsi` ポッドの 2 つのインスタンス、および `csi-nodeplugin-hxcsi` ポッドの 1 つのインスタンスが必要です。したがって、合計2つのKubernetesワーカーノードがある場合は、次の例に示すように、`csi-nodeplugin-hxcsi` ポッドの 2 つのインスタンスが表示されます。

```
administrator-host:hxcsi$ kubectl get pods
NAME                                READY    STATUS    RESTARTS     AGE
csi-attacher-hxcsi-0                 2/2     Running       0         37h
csi-attacher-hxcsi-1                 2/2     Running       0         37h
csi-nodeplugin-hxcsi-2nsfq           2/2     Running       2         37h
csi-nodeplugin-hxcsi-qjh9n           2/2     Running       2         37h
csi-provisioner-hxcsi-0              2/2     Running       0         37h
csi-provisioner-hxcsi-1              2/2     Running       0         37h
csi-resizer-hxcsi-0                  2/2     Running       0         37h
csi-resizer-hxcsi-1                  2/2     Running       0         37h
```

次のタスク

Cisco HyperFlex CSI ストレージ クラスを作成します。

HXCSI サンプルポッド

HXCSI パッケージには、ポッドを作成するためのいくつかの例が含まれています。

表 *2 : HXCSI* パッケージのサンプル

#	ディレクトリ名	説明
1	`sample-hxcsi`	`nginx` を実行しているポッド。基本的な例では、「`hxpvclaim-default`」という名前のPVCを作成します。
2	`sample-hxcsi-csi-clone`	「`sample-hxcsi`」からクローンを作成するポッド。「`dataSource`」は「`hxpvclaim-default`」として表示されます。
3	`sample-hxcsi-ds`	名前付きデータストア「`test-ds`」を使用するポッド
4	`sample-hxcsi-fs`	デフォルトのデータストアおよびファイル システム型「`xfs`」を使用するポッド

#	ディレクトリ名	説明
5	sample-hxcsi-no-ds	デフォルトのデータストアとデフォルトのファイル システムを使用するポッド。
6	sample-hxcsi-no-ds-clone	データストア名が指定されたサンプル「sample-hxcsi-no-ds」から複製します。
7	sample-resize-block	ブロックボリュームのサイズを変更し、属性として allowVolumeExpansion: true を使用します。
8	sample-resize-fs	デフォルト ファイルシステム「ext4」をサイズ変更します。 allowVolumeExpansion：true
9	sample-resize-clone	hxpvclaim-default-resize 「sample-resize-fs」サンプル ポッドからのクローン。

（注） ボリュームのサイズを変更するには、次の手順を実行します。

1. PVC の yaml ファイルのボリュームのサイズを変更します。
2. kubectl apply -f <pvc.yaml>コマンドを実行して、新しいサイズ設定を適用します。

Cisco HyperFlex CSI ストレージ クラスの作成

コンポーネントが稼働したら、Cisco HyperFlex CSI インテグレーションを通じて開発者がストレージを使用できるストレージ クラスを作成する必要があります。

始める前に

Cisco HyperFlex CSI コンポーネントの展開

ステップ 1 管理者ホストで、「hxcsi-storage-class.yaml」という名前のファイルを次の内容で作成します。

例：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default
provisioner: csi-hxcsi
parameters:
    datastore: default-ds
    datastoreSize:"20000000000000”
```

上記のように、パラメータ セクションでデータストアの名前とサイズを指定できます。オプションで、これをデフォルトのストレージクラスにすることもできます。つまり、使用する他のストレージクラスを指定しない永続ボリューム クレームに対しては、Cisco HyperFlex CSI ストレージ インテグレーションがデフォルトで使用されます。Cisco HyperFlex CSI ストレージ クラスをデフォルトのストレージ クラスにする場合は、「hxcsi-storage-class.yaml」ファイルに次の内容が含まれている必要があります。

例：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default
    annotations:
        storageclass.kubernetes.io/is-default-class: "true"
provisioner: csi-hxcsi
parameters:
```

（注） データストアがまだ存在しない場合は、新しいデータストアが作成されます。データストア名を指定しない場合、「iscsiDs」という名前のデフォルトのデータストアが作成されます。

（注） 作成するボリュームよりも大きいデータストアを常に作成します。

ステップ 2 管理者ホストで kubectl create -f コマンドを使用して、Cisco HyperFlex CSI ストレージ クラスを作成します。

例：

```
root@administrator-host:hxcsi$ kubectl create -f ./hxcsi-storage-class.yaml

storageclass.storage.k8s.io/csi-hxcsi-default created
```

ボリューム サイズ変更のストレージ クラスの例

例：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default-resize
provisioner: csi-hxcsi
parameters:
          datastore: default-ds
          datastoreSize:"20000000000000"
allowVolumeExpansion: true
```

ボリュームのサイズ変更では、このストレージクラスに対してプロビジョニングされたボリュームのみがサイズ変更をサポートすることに注意してください。ボリュームの実際のサイズを変更するには、PVC 仕様を編集して新しいサイズに変更する必要があります。たとえば、PVC YAMLファイルを編集し、kubectl apply -f<pvc-yaml> を実行します。

ファイル システムのサンプル ストレージ クラス

例：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-default-fs
provisioner: csi-hxcsi
parameters:
    fsType: xfs
```

注：デフォルトのファイル システムは「ext4」です。

次のタスク

Cisco HyperFlex CSI ストレージ クラスの作成を確認します。

Verifying Cisco HyperFlex CSI ストレージ クラスの作成

ストレージ クラスの作成を確認するには、次の手順を実行します。

（注） Cisco HyperFlex CSIストレージクラスをデフォルトとして設定する場合は、ストレージ クラス名の横に「（default）」が表示されていることを確認します。

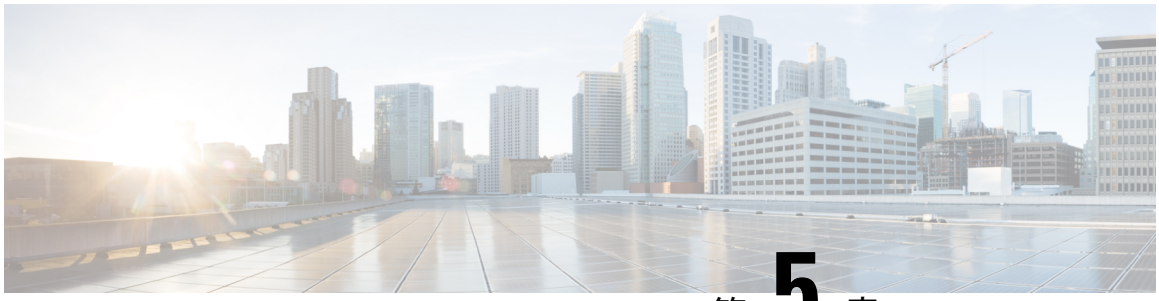
始める前に

Cisco HyperFlex CSI ストレージ クラスを作成します。

管理者ホストで kubectl get sc コマンドを使用して、Cisco HyperFlex CSI ストレージ クラスが作成されたことを確認します。

例：

```
root@administrator-host:hxcsi$ kubectl get sc
NAME    PROVISIONER  AGE
csi-hxcsi (default) csi-hxcsi  67s
```



第 5 章

Cisco HyperFlex CSI によるステートフル アプリケーションの展開

- Cisco HyperFlex CSI によるステートフル アプリケーションの展開の前提条件 （21 ページ）
- 管理者ホスト （21 ページ）
- ステートフル アプリケーションの展開 （22 ページ）

Cisco HyperFlex CSI によるステートフル アプリケーションの展開の前提条件

HyperFlex CSI ストレージ インテグレーションを使用してステートフル アプリケーションを展開する前に、次の前提条件を満たしている必要があります。

- Cisco HyperFlex クラスタがインストールされ、HX 5.0(x) 以降を実行している。
- Cisco HyperFlex CSI インテグレーションが展開されました。
- HX Connect の [iSCSI] タブから、iSCSI ネットワークを最初に作成する必要があります。詳細については、『』『Cisco HyperFlex Data Platform Administration Guide、Release 5.0』を参照してください。

管理者ホスト

このガイドでは、管理者ホストは kubectl コマンドなどを Kubernetes クラスタに対して実行するための Linux ベースのシステムのことです。これは通常、Kubernetes クラスタの一部ではない別のシステム（VM）ですが、別のシステム（VM）をインストール/管理する必要がない場合は、管理者ホストとして Kubernetes ノードの 1 つを使用できます。

ステートフル アプリケーションの展開

ステートフル アプリケーションを展開するには、次の手順を実行します。

永続的なボリューム要求の作成

永続ボリューム要求は、単にユーザによるストレージの要求です。ユーザは、ストレージ要件、必要なストレージのサイズまたは容量、およびその他のオプションを指定します。関連付けられたストレージクラスに応じて、ストレージ要件は、要求されたストレージをプロビジョニングし、Kubernetes で使用できるようにする適切なプロビジョニング担当者にルーティングされます。

（注） 最大 PVC サイズは 64Ti です。サポートされている最小 PVC サイズは 1 Gi です。

（注） CHAP で保護されたボリュームを作成できます。ターゲットごとに 1 つのストレージクラスで作成できるボリューム（永続的なボリューム要求）は最大 255 です。

ステップ 1 管理者ホストで、次の内容の「message-board-pvc.yaml」という名前のファイルを作成します。

例：

```
administrator-host:hxcsi$ cat ./message-board-pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: message-board-pvc
spec:
  storageClassName: csi-hxcsi-default
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

ステップ 2 管理者ホストで、`kubectl create -f` コマンドを使用して永続ボリューム クレームを作成します。

例：

```
administrator-host:hxcsi$ kubectl create -f ./message-board-pvc.yaml

persistentvolumeclaim/message-board-pvc created
```

ステップ 3 管理者ホストで `kubectl get pvc` コマンドを使用して、永続ボリューム クレームが作成され、永続ボリュームに正常にバインドされていることを確認します。

例：

```
administrator-host:hxcsi$ kubectl get pvc
```

```
NAME    STATUS  VOLUME  CAPACITY  ACCESS  MODES  STORAGECLASS  AGE
message-board-pvc  BOUND  pvc-8069462e-662c-11e9-a163-005056a086d9  10Gi  RWO  csi-hxcsi-default
20s
```

ステートフル Kubernetes ワークロードの展開

Kubernetes ワークロードは、Kubernetes ワークロードのタイプに関係なく、ポッドや展開などのさまざまな形式で提供され、それぞれが Cisco HyperFlex CSI インテグレーションと永続ボリューム クレームを使用して永続ストレージを活用できます。次に、Cisco HyperFlex CSI インテグレーションのテストに使用できるCisco Message Boardと呼ばれるサンプル オープン ソース アプリケーションの導入を示します。同じ方法と手順に従って、独自のアプリケーションでテストすることもできます。

ステップ 1 管理者ホストで、展開するワークロードを定義する YAML ファイルを作成します。

例：

以下は、Kubernetes Deployment と NodePort を介して展開された Cisco Message Board アプリケーションへの接続を可能にする Kubernetes Service の両方を作成するサンプルの Cisco Message Board アプリケーションの YAML ファイルを示しています。

（注） Kubernetes Deployment 定義の「ボリューム」セクションで永続ボリューム クレーム名を参照していること。この例では、「message-board-pvc」永続ボリューム クレームにバインドされた永続ボリュームが、「/sqldb」の場所（パス）にある「message_board:version1」コンテナ内にマウントされます。

```
administrator-host:hxcsi$ cat ./message-board-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: message-board
    labels:
        app: message-board
spec:
    replicas: 1
    selector:
        matchLabels:
            app: message-board
    template:
        metadata:
            labels:
                app: message-board
                name: message-board
        spec:
            volumes:
                - name: demovolume1
                  persistentVolumeClaim:
                    claimName: message-board-pvc
            containers:
            - name: message-board
              image: michzimm/message_board:version1
              ports:
              - containerPort: 5000
```

```
            volumeMounts:
                - mountPath: "/sqldb"
                  name: demovolume1
---
apiVersion: v1
kind: Service
metadata:
  name: message-board
  labels:
    name: message-board
  namespace: default
spec:
  type: NodePort
  ports:
  - port: 5000
    nodePort: 30002
  selector:
    name: message-board
```

ステップ 2 管理者ホストで `kubectl create -f` コマンドを使用して、展開とサービスを作成します。

例：

```
administrator-host:hxcsi$ kubectl create -f ./message-board-deployment.yaml
deployment.apps/message-board created
service/message-board created
```

ステップ 3 管理者ホストで `kubectl get pods` コマンドを使用して、展開されたポッドのステータスを確認します。

例：

```
administrator-host:hxcsi$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
csi-attacher-hxcsi-0                    2/2     Running   0          3h51m
csi-attacher-hxcsi-1                    2/2     Running   0          3h51m
csi-nodeplugin-hxcsi-9fgsf              2/2     Running   0          3h51m
csi-nodeplugin-hxcsi-qqvwj              2/2     Running   0          3h51m
csi-provisioner-hxcsi-0                 2/2     Running   0          3h51m
csi-provisioner-hxcsi-1                 2/2     Running   0          3h51m
csi-resizer-hxcsi-0                     2/2     Running   0          3h51m
csi-resizer-hxcsi-1                     2/2     Running   0          3h51m
message-board-6df65d6b59-49xhq          1/1     Running   0          95s
```

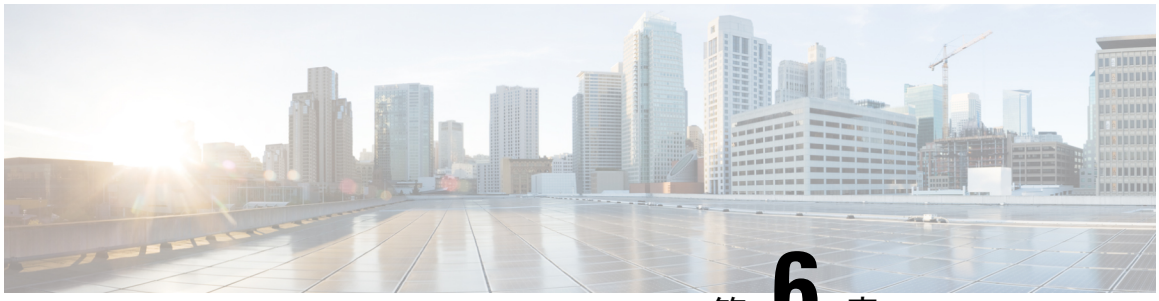
例：

ステップ 4 管理者ホストで、`kubectl get services` コマンドを使用して、展開されたサービスのステータスを確認します。

例：

```
root@administrator-host:hxcsi$ kubectl get services
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
csi-attacher-hxcsi      ClusterIP   10.98.79.159     <none>        12346/TCP        3h53m
csi-provisioner-hxcsi   ClusterIP   10.99.73.185     <none>        12345/TCP        3h53m
kubernetes              ClusterIP   10.96.0.1        <none>        443/TCP          4h24m
message-board           NodePort    10.107.227.152   <none>        5000:30002/TCP   2m59s
```

サンプルの Cisco Message Board アプリケーションでは、サービスは「NodePort」とポート「30002」を使用して設定されます。これは、Web ブラウザで任意の Kubernetes ノードの IP アドレスとポート「30002」を指定することにより、アプリケーションが稼働中であることを意味します。例：http://<k8s-worker1>:30002



第 6 章

ソフトウェア暗号化ボリューム、スナップショットの作成、および CHAP の使用

- ソフトウェア暗号化ボリュームの作成 （25 ページ）
- Creating Volume Snapshots （26 ページ）
- ボリュームの CHAP 保護の使用 （29 ページ）

ソフトウェア暗号化ボリュームの作成

HXCSI 1.2（3a）以降、ソフトウェア暗号化ボリュームを作成できます。

ソフトウェア暗号化ボリュームを作成するための前提条件

HyperFlex CSI ストレージの統合を使用してソフトウェア暗号化されたボリュームを作成する前に、次の前提条件を満たしている必要があります。

- Cisco HyperFlex クラスタがインストールされ、5.0(2a) 以降を実行している。詳細については、『VMware ESXi 用 Cisco HyperFlex システム リリース 5.0 インストール ガイド』を参照してください。
- クラスター上で HyperFlex ソフトウェア暗号化を有効にする詳細については、『Cisco HyperFlex Data Platform Administration Guide、Release 5.0』の中の『[HyperFlex ソフトウェア暗号化（HyperFlex Software Encryption）]』を参照してください。

ソフトウェア暗号化ボリュームの作成

ソフトウェア暗号化ボリュームを作成するには、ソフトウェア暗号化を有効にして Kubernetes ストレージ クラスを作成します。

始める前に

このクラスタのデータストア（DS）を暗号化する前に、HX クラスタがソフトウェア暗号化をサポートしている必要があります。

ステップ 1 ソフトウェア暗号化を有効にして Kubernetes ストレージ クラスを作成します。これを行うには、ストレージ クラス ファイルでデータストア名（「ds-se」など）を指定し、datastoreEnryption 属性を「true」に設定します。

たとえば、「 hxcsi-storage-class.yaml 」というストレージ クラス ファイル上

例：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: hxcsi-software-encryption
provisioner: csi-hxcsi
parameters:
    datastore: ds-se
    datastoreSize: "1000000000000"
    datastoreEncryption: "true"
```

（注） HXCSI の「 datastoreEncryption 」属性はデフォルトで「 false 」に設定されています。つまり、この属性が StorageClass に含まれていない場合、データストアは暗号化されません。

ステップ 2 Create a persistent volume claim which refers to the storage class file using the *storageClassName* field.

たとえば、永続ボリューム クレームでは、ソフトウェアで暗号化されたストレージ クラスを参照できます。

例：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
    name: hxpvclaim-se
spec:
    accessModes:
    - ReadWriteOnce
    resources:
        requests:
            storage: 5Gi
    storageClassName: hxcsi-software-encryption
```

Creating Volume Snapshots

HXCSI 1.2(3a) 以降では、ボリューム スナップショットを作成できます。

ボリューム スナップショットを作成するための前提条件

HyperFlex CSI ストレージの統合を使用してボリューム スナップショットを作成する前に、次の前提条件を満たしている必要があります。

- Cisco HyperFlex クラスタがインストールされ、5.0(2a) 以降を実行している。詳細については、『VMware ESXi 用 Cisco HyperFlex システム リリース 5.0 インストール ガイド』を参照してください。

ボリュームから HXCSI スナップショットを作成

ボリューム スナップショットの作成を有効にするには、次の手順を実行します。

- スナップショット クラスの構成を作成します。
- ボリュームのスナップショットの構成を作成します。
- スナップショットからボリュームの構成を作成します。
- 新しいボリュームを使用するポッドを展開します。
- コンフィギュレーション ファイルを作成します。
- 新しく作成された技術情報を表示します。

ステップ 1 スナップショット クラスの構成を作成します。管理者ホストで、「hxcsi-snapshot-class.yaml」という名前のファイルを作成します。

たとえば、「sample-hxcsi-snapshot」フォルダのスナップショット クラスには、次のものが含まれています。

例：

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
    name: csi-hxcsi-default-snapshot
driver: csi-hxcsi
deletionPolicy: Delete
```

deletePolicy が `Delete` に設定されている場合、基になるストレージ スナップショットは VolumeSnapshotContent オブジェクトとともに削除されます。deletePolicy が `Retain` に設定されている場合、基礎となるスナップショットと VolumeSnapshotContent の両方が残ります。

（注） `Retain` が使用されている場合、ユーザーは、基礎となるスナップショットとボリューム napshotcontent を削除する必要があります。

ステップ 2 ボリュームのスナップショットの構成を作成します。管理者ホストで、「hxcsi-snapshot.yaml」という名前のファイルを作成します。

例：

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
    name: hxpvclaim-default-snapshot
spec:
    volumeSnapshotClassName: csi-hxcsi-default-snapshot
    source:
        persistentVolumeClaimName: hxpvclaim-default
```

`persistentVolumeClaimName` は、スナップショットの PersistentVolumeClaim データ 送信元の名前であり、スナップショットを取得する前に存在している必要があります。このフィールドは、スナップショットをダイナミックにプロビジョニングするために必要です。ボリューム スナップショットは、属性

volumeSnapshotClassName を使用して VolumeSnapshotClass の名前を指定することにより、特定のストレージクラスを要求できます。何も設定されていない場合、使用可能な場合はデフォルトのクラスが使用されます。

ステップ 3 スナップショットからボリュームの構成を作成します。管理者ホストで、「hxcsi-pvc-from-snapshot.yaml」という名前のファイルを作成します。

例：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
    name: hxpvclaim-default-from-snapshot
spec:
    storageClassName: csi-hxcsi-default
    accessModes:
    - ReadWriteOnce
    resources:
        requests:
            storage: 5Gi
    dataSource:
        kind: VolumeSnapshot
        name: hxpvclaim-default-snapshot
        apiGroup: "snapshot.storage.k8s.io
```

「dataSource」属性は、新しいボリュームの送信元を記述します。この場合、以前に作成したスナップショットの名前を持つ VolumeSnapshot です。これにより、スナップショットの格納ファイルで新しいボリュームが作成されます。「ストレージ」属性は、新しいボリュームのサイズを指します。Cisco HXCSI を使用してスナップショットからボリュームを作成する場合、ボリューム サイズの拡張はサポートされていません。

ステップ 4 ポッド（つまり、ngnix）を展開して、新しいボリュームを使用します。管理者ホストで、「hxcsi-nginx-from-snapshot.yaml」という名前のファイルを作成します。

例：

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: test-from-snapshot
    labels:
        app: test-from-snapshot
spec:
    replicas: 1
    selector:
        matchLabels:
            app: test-from-snapshot
    template:
        metadata:
            labels:
                app: test-from-snapshot
        spec:
            volumes:
                - name: test-snapshot-volume
                  persistentVolumeClaim:
                    claimName: hxpvclaim-default-from-snapshot
            containers:
            - name: nginx
              image: nginx:1.7.9
              ports:
              - containerPort: 80
```

```
volumeMounts:
    - mountPath: "/usr/mnt/test"
      name: test-snapshot-volume
```

ステップ 5 すべての構成ファイルを作成したら、作成された順序で構成ごとに「**kubectl create**」コマンドを実行します。

例：

```
kubectl create -f <<filename.yaml>>
```

技術情報をクリーンアップするには、次のコマンドを使用します。

例：

```
kubectl delete -f <<filename.yaml>>
```

ステップ 6 新しく作成された技術情報を表示します。

ボリューム スナップショット クラスを表示するには、次のコマンドを実行します。

例：

```
kubectl get volumesnapshotclass
```

ボリュームのスナップショットを表示するには、次のコマンドを実行します。

例：

```
kubectl get volumesnapshot
```

ボリューム スナップショットの内容を表示するには、次のコマンドを実行します。

例：

```
kubectl get volumesnapshotcontent
```

ボリュームの CHAP 保護の使用

HXCSI 1.2(3a) 以降、ボリュームに CHAP 保護を使用できます。CHAP（チャレンジ ハンドシェイク認証プロトコル）は、サーバーでリモート ユーザーまたはシステムの ID を検証するために使用されるチャレンジアンドレスポンス認証方式です。ボリュームは、HyperFlex iSCSI LUN によってバッキングされます。CHAP 保護は、ターゲット レベルのストレージ オブジェクトに対してサポートされています。イニシエータが CHAP が有効なターゲットとのセッションを確立しようとすると、CHAP が適用されます。iSCSI セッションの確立中に、提供されたユーザー名やパスワードが構成されたログイン情報と一致しない場合、セッションの確立は機能不全になります。

CHAP ログイン情報の構成は、ストレージ クラスを介して行われます。CHAP 保護を有効にするには、ストレージ クラスの新しいフィールドを使用して、*[*ターゲット名（*target name*）*]*や *[CHAP* ログイン情報（*CHAP credentials*）*]*などの追加情報を提供します。CHAP 保護が必要なボリュームは、CHAP 対応のストレージクラスに属する永続ボリュームを作成する必要があります。

（注） iSCSI では、ターゲットに最大 255 の LUN を作成できます。このため、特定の CHAP 対応ストレージ クラスに属する最大 255 のボリュームを作成できます。同じ認証情報で保護された 255 を超えるボリュームを作成するには、同じシークレットを使用して、異なるターゲットで複数のストレージボリュームを作成する必要があります。2つのストレージクラスが同じターゲットを参照している場合は、同じシークレットを参照するか、同じユーザー名とパスワードを持つ異なるシークレットを参照していることを確認してください。

次の制限事項に注意してください：

- CHAP 対応のストレージ クラスを変更して、非 CHAP 対応のストレージ クラスにすることはできません。また、CHAP 非対応のストレージクラスを変更して、CHAP 対応のストレージ クラスにすることもできません。
- CHAP 対応ボリュームを変更して非 CHAP 対応ボリュームにすることも、非 CHAP 対応ボリュームを変更して CHAP 永続ボリュームにすることもできません。
- また、CHAP が有効でないストレージ クラスで CHAP が有効なターゲットを使用することもできません。また、CHAP が有効なストレージクラスでは、CHAP が有効になっていないターゲットを使用することもできません。

ボリュームに CHAP を使用するための前提条件

HyperFlex CSI ストレージの統合を使用してCHAP 保護をボリュームに使用する前に、次の前提条件を満たしている必要があります。

- Cisco HyperFlex クラスタがインストールされ、5.0(2a) 以降を実行している。詳細については、『VMware ESXi 用 Cisco HyperFlex システム リリース 5.0 インストール ガイド』を参照してください。

ボリュームでの CHAP 保護の有効化

ボリュームで CHAP 保護を有効にするには、次の手順を実行します。

- Kubernetes シークレットを作成します。
- CHAP 対応の Kubernetes ストレージ クラスを作成します。
- *storageClassName* フィールドを使用して、ストレージ クラス ファイルを参照する永続ボリューム クレームを作成します。
- *[*ボリューム（*volumes*）*]* セクションの *persistentVolumeClaim* フィールドを使用して作成された永続ボリューム クレームを参照する展開またはポッドを作成します。

ステップ 1 Kubernetes シークレットを作成します。これを行うには、シークレット yaml ファイルで *kubectl create secret* コマンドを使用します。*node_sesion_auth.username* および *node_session.auth_password* フィールドを使用して、ユーザー名とパスワードが base64 でエンコードされた文字列であることを確認します。

たとえば、秘密の yaml ファイルで次のようにします。

例：

```
apiVersion: v1
kind: Secret
metadata:
    name: hxcsi-chap-secret
    namespace: default
type: "kubernetes.io/iscsi-chap"
data:
    node.session.auth.username: YWRtaW4=
    node.session.auth.password: Q2lzY28xMjM=
```

ステップ 2 CHAP 対応の Kubernetes ストレージ クラスを作成します。ストレージ クラス ファイルには、Kubernetes シークレットへの参照と、*targetForChap* フィールドを使用してボリュームが作成されるターゲットへの参照が含まれている必要があります。

たとえば、CHAP を使用するストレージ クラス ファイルでは、次のようにします。

例：

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
    name: csi-hxcsi-sc-chap
provisioner: csi-hxcsi
parameters:
    csi.storage.k8s.io/node-publish-secret-name: hxcsi-chap-secret
    csi.storage.k8s.io/node-publish-secret-namespace: default
    csi.storage.k8s.io/provisioner-secret-name: hxcsi-chap-secret
    csi.storage.k8s.io/provisioner-secret-namespace: default
    # Uncomment below controller-expand parameters to support CHAP for volume resize as well
    # csi.storage.k8s.io/controller-expand-secret-name: hxcsi-chap-secret
    # csi.storage.k8s.io/controller-expand-secret-namespace: default
    targetForChap: testTargetChap
```

ステップ 3 *storageClassName* フィールドを使用して、ストレージ クラス ファイルを参照する永続ボリューム クレームを作成します。

たとえば、CHAP が有効なストレージ クラスを参照する CHAP が有効な永続ボリューム クレームでは、次のようになります。

例：

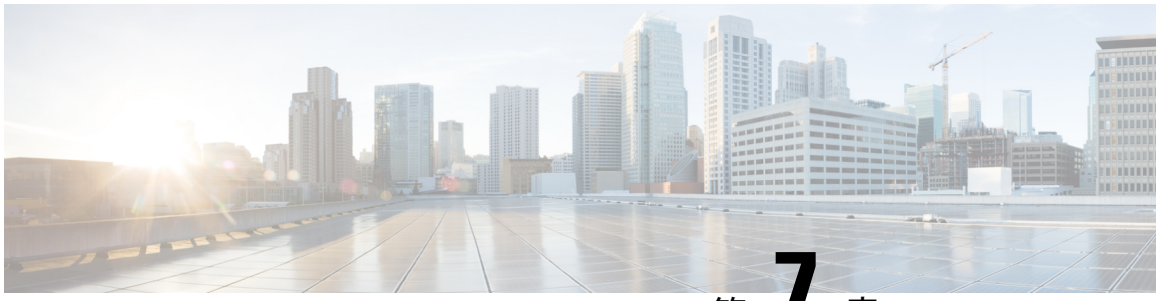
```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
    name: hxpvclaim-default-chap
spec:
    accessModes:
    - ReadWriteOnce
    resources:
        requests:
            storage: 5Gi
    storageClassName: csi-hxcsi-sc-chap
```

ステップ 4 *[*ボリューム（*volumes*）*]* セクションの *persistentVolumeClaim* フィールドを使用して作成された永続ボリューム クレームを参照する展開またはポッドを作成します。

たとえば、CHAP 対応ボリュームを使用する展開では、次のようにします。

例：

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: test-chap
    labels:
        app: test-chap
spec:
    replicas: 1
    selector:
        matchLabels:
            app: test-chap
    template:
        metadata:
            labels:
                app: test-chap
        spec:
            volumes:
                - name: test-volume-chap
                  persistentVolumeClaim:
                    claimName: hxpvclaim-default-chap
            containers:
            - name: nginx
              image: nginx:1.7.9
              ports:
              - containerPort: 80
              volumeMounts:
                  - mountPath: "/usr/mnt/test-chap"
                    name: test-volume-chap
```



第 7 章

トラブルシューティング

- トラブルシューティング （33 ページ）
- HXCSI ポッドの展開時の ImagePullBackOff ステータスエラー （33 ページ）
- ボリューム削除の失敗 （34 ページ）
- ノード削除中の ContainerCreating 状態のアプリケーション ポッド （34 ページ）
- 終了中または ContainerCreating 状態のアプリケーション ポッド スタッド （35 ページ）
- 削除されたポッドが同じノードに戻るようにスケジュールされている （36 ページ）

トラブルシューティング

次のセクションでは、HyperFlex CSI 統合をインストールして使用するときに見られる一般的な問題について説明します。提供される情報には、問題の診断に役立つ症状と、問題を解決するための解決策が含まれています。

HXCSI ポッドの展開時の ImagePullBackOff ステータスエラー

- **症状 1**：コマンド「kubectl get pods [-n <namespace>]」を実行すると、HXCSI ポッドのステータスが「ImagePullBackOff」であると表示されます。
- **症状 2**：コマンド「kubectl description pod」の実行<csi-pod_name>」には、「Error：ErrImagePull」および「Back-off pulling image…」というエラーを含むメッセージが表示されます。

解決方法：

- **解決策 1**：hxcsi-setup スクリプトに指定された HXCSI コンテナイメージ名が正しいことを確認します。
- **解決策 2**：HXCSI コンテナイメージが、各 Kubernetes ワーカーノードの Docker 内に直接存在するか、またはローカル コンテナイメージ レジストリに存在することを確認します。

- **解決策 3**：hxcsi-setup スクリプトによって生成される次の YAML ファイルの「imagePullPolicy」行が「IfNotPresent」に設定されていることを確認します。`csi-attacher-hxcsi.yaml`、`csi-nodeplugin-hxcsi.yaml`、`csi-provisioner-hxcsi .yaml`
- **解決策 4**：次のイメージが各 Kubernetes ノードのローカルコンテナ イメージ レジストリにロードされていることを確認します。`csi-attacher-3.2.1-cisco1.tar`、`csi-node-driver-registrar-2.2.0-cisco1.tar`、`csi -resizer-1.2.0-cisco1.tar`、`csi-provisioner-2.2.1-cisco1`

ボリューム削除の失敗

NodeUnpublish が成功し、ボリュームがマウント解除された後でも、古いボリューム接続が存在するため、ボリュームの削除は失敗します。これは、etcd リーダーの選択中に delete volumeattachment kubernetes api が失われた場合に発生します。nodeUnpublish が完了し、ボリュームがノードから正常にアンマウントされた後でも、ボリュームの削除は失敗します。

external-provisioner のログは次のように表示されます。

```
ボリュームの削除に失敗しました：persistentvolume <pv-name> はまだノード <node-name> に接続されています。
```

external-attacher のログは次のように表示されます。

```
<Volume-attachment> はすでに接続されています。
```

解決方法：

次のコマンドを使用して、古いボリューム接続を削除します。

```
kubectl delete volumeattachments <VA-name>
```

プロビジョニング担当者が再試行すると、数秒後に pv が削除されます。

または、次のコマンドを使用して手動で削除することもできます。

```
kubectl delete pv <pv-name>
```

ノード削除中の ContainerCreating 状態のアプリケーション ポッド

ノード削除中の ContainerCreating 状態のアプリケーション ポッドまたはまたはマルチ アタッチ エラー状態でスタックし、ボリュームをマウントできません。これは、クラスタから k8s ワーカー ノードを削除または削除するときに発生することがあり、ポッドは新しいワーカー ノードに移行します。

K8 ワーカー ノードを削除する推奨方法は、次のコマンドを使用することです。

```
kubectl drain <node-name>
kubectl delete  node   <node-name>
```

詳細については、「Kubernetes からノードを正常に削除するには（How to graceful remove a node from Kubernetes?）」を参照してください。

終了中または ContainerCreating 状態のアプリケーション ポッド スタッド

アプリケーション ポッドは、Terminating または ContainerCreating 状態で表示される場合があります。一般的な理由には、VM の再起動が完了せず、ハング状態になっていることが含まれます。これは、次の例に示すように、systemd に禁止ロックを設定するプロセスがあり、そのプロセスがシャットダウン前のタスクを完了せず、systemd がプロセスの終了に失敗した場合に発生します。

```
ubuntu@m5-k8-3:~$ systemd-inhibit --list
Who: Unattended Upgrades Shutdown (UID 0/root, PID 778/unattended-upgr)
What: shutdown
Why: Stop ongoing upgrades or perform upgrades before shutdown
Mode: delay

1 inhibitors listed.
ubuntu@m5-k8-3:~$ ps aux | grep 778
root 778 0.0 0.1 185948 20028 ? Ssl May13 0:00 /usr/bin/python3
 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
ubuntu@m5-k8-3:~$
```

iscsid が正常にシャットダウンできなかった場合、VM がハング状態のままになることがあります。シャットダウン プロセスは iscsid を強制終了する前に最大時間待機する場合がありますが、VM がリブートしない場合があります。次の例に示すように、報告された iSCSI 接続エラーを確認します。

```
Jun 09 19:12:46 m5-k19-3 iscsid[967]: Kernel reported iSCSI connection 2:0 error
 (1010 - ISCSI_ERR_BAD_ITT: Received invalid initiator task tag from target) state (3)
Jun 09 19:12:48 m5-k19-3 iscsid[967]: connection2:0 is operational after recovery (1
attempts)
Jun 11 15:17:27 m5-k19-3 iscsid[967]: Kernel reported iSCSI connection 2:0 error (1010
-
ISCSI_ERR_BAD_ITT: Received invalid initiator task tag from target) state (3)
Jun 11 15:17:29 m5-k19-3 iscsid[967]: connection2:0 is operational after recovery (1
attempts)
```

ソリューション

VM コンソールにログインし、VM が正しくリブートし、Kubelet が応答していることを確認します。

- ワーカー ノードは Ready 状態である必要があります。VM が完全にシャットダウンしていない場合は、再起動します。
- VM の再起動中に iscsid が正常にシャットダウンしない場合は、HXDP ターゲットへの iSCSI データ パス接続（接続エラー、MTU、再試行など）を確認します。

VM の再起動またはリブートが実行された後、アプリケーションポッドが Terminating または ContainerCreating 状態で表示されることがあります。根本的な原因としては、VM がハングし、

シャットダウン プロセスが完了せず、Kubelet が応答しない状態になっていることが考えられます。デフォルトでは、API コントローラ マネージャは kubernetes ノードが起動するまで 5 分間待機し、その後アプリケーション ポッドを別の場所で削除または再作成することを決定します。

この状態から回復するには、VM を再起動またはリセットし、VM が起動することを確認します。

削除されたポッドが同じノードに戻るようにスケジュールされている

実行中（Running）状態のポッドが `kubectl delete pod` コマンドを使用して削除された後に再作成され、名前空間を削除すると、**終了中（Terminating）**状態でスタックしました。

実行中のポッドで `kubectl delete pod` コマンドを使用する代わりに、次のベスト プラクティスの方法が推奨されます。

1. 削除するポッドが実行されているノード名をメモします。

   ```
   kubectl get pods -o wide --all-namespaces
   ```

2. ポッドが実行されているノードのコードンをオフにします。

   ```
   kubectl cordon <node-name>
   ```

3. ポッドを削除します。

   ```
   kubectl delete pod <pod-name>
   ```

4. 削除されたポッドが別のノードでスケジュールされていることを次を使用して確認します。

   ```
   kubectl get pods -o wide --all-namespaces
   ```

5. ノードのコードンを外します。

   ```
   kubectl uncordon <node-name>
   ```

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。