



## **Cisco HyperFlex Systems リリース 4.5 アドミニストレーション ガイド (Kubernetes 用)**

初版：2021 年 1 月 22 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/>

【注意】 シスコ製品をご使用になる前に、安全上の注意（[www.cisco.com/jp/go/safety\\_warning/](http://www.cisco.com/jp/go/safety_warning/)）をご確認ください。本書は、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。また、契約等の記述については、弊社販売パートナー、または、弊社担当者にご確認ください。

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com go trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2021–2023 Cisco Systems, Inc. All rights reserved.



## 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

ヘルムユーティリティを使用した展開	16
Cisco HyperFlex CSI コンポーネントを使用した HXCSI の展開	17
HXCSI サンプルポッド	18
Cisco HyperFlex CSI ストレージクラスの作成	19
Verifying Cisco HyperFlex CSI ストレージクラスの作成	21

---

第 5 章	<b>Cisco HyperFlex CSI によるステートフルアプリケーションの展開</b>	23
	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	23
	管理者ホスト	23
	ステートフルアプリケーションの展開	24
	永続的なボリューム要求の作成	24
	ステートフル Kubernetes ワークロードの展開	25

---

第 6 章	<b>トラブルシューティング</b>	29
	トラブルシューティング	29
	HXCSI ポッドの展開時の ImagePullBackOff ステータスエラー	29
	ボリューム削除の失敗	30
	ノード削除中の ContainerCreating 状態のアプリケーションポッド	30
	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	31



## 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ル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

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

次の表に、『Cisco HyperFlex Systems リリース 4.5(x) アドミニストレーションガイド (Kubernetes 用)』の新機能と変更された機能、およびそれらの説明がどこに記載されているかをまとめます。

機能	説明	日付/追加されたリリース	参照先
Cisco HyperFlex Systems リリース 4.5(x) アドミニストレーションガイド (Kubernetes 用)	次のガイドが導入されました。	HX 4.5	該当なし







## 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、Cisco HyperFlex が Cisco HyperFlex 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：

HyperFlex Data Platform バージョン	CSI 仕様バージョン	Kubernetes のバージョン	シスコ認定 CCP バージョン	シスコ認定 Anthos バージョン	iSCSI を開く
4.0(1a)	1.0	1.14	5.0、5.1	1.1、1.2、1.3	
4.0(2a)	1.0	1.15	5.1、5.2	1.3	
4.0(2b)	1.0	1.16	6.0、7.0	1.4.1	
4.0(2c)	1.0	1.17	6.0、7.0	1.5.1	
HXDP 4.5(1a)	1.2	1.18.2	-	-	Open iSCSI 2.0.874-5ubuntu2.10
HXDP 4.5(1a) HXCSI 1.2(1a)	1.2	1.18.2、1.19.8	-	-	Open iSCSI 2.0.874-5ubuntu2.10
HXDP 4.5(2b) HXDP 4.5(2d) HXCSI 1.2(1b)	1.2	1.18.2、1.19.8	-	-	Open iSCSI 2.0.874-5ubuntu2.10



(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 \(6 ページ\)](#)

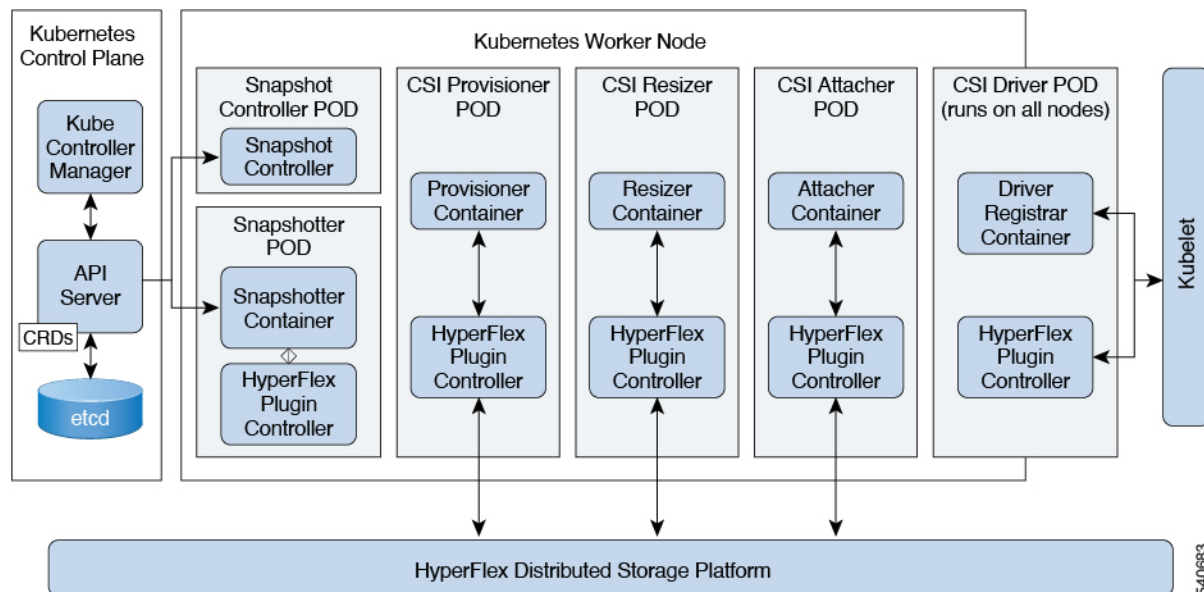
## Cisco HyperFlex Kubernetes CSI について

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 1 つ。
- 目的: CSI が必要ですが、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 1 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 1 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 \(21 ページ\)](#)

## 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 4.5(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 Administration Guide, Release 4.5](#)』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 (iscsid)
```

```
Loaded: loaded (/lib/systemd/system/iscsid.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。

### 手順

- 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-1.2.1-601.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。

## 手順

---

`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-1.2.0-601.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。

## 手順

**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-1.2.1-601.tar k8s-worker1:/tmp
```

```
administrator-host:hxcsi$ scp ./images/hxcsi-1.2.1-601.tar k8s-worker2:/tmp
```

```
administrator-host:hxcsi$ scp ./images/hxcsi-1.2.1-601.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。

**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-1.2.2a-626.tar
```

```
Loaded image: hxcsi:hxcsi-1.2.1-601
```

```
k8s-worker2:/tmp# docker load -input ./hxcsi-1.2.1-601.tar Loaded image:
```

```
hxcsi:hxcsi-1.2.1-601
```

```
k8s-workerN:/tmp# docker load -input ./hxcsi-1.2.1-601.tar Loaded image:
```

```
hxcsi:hxcsi-1.2.1-601
```

**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-3.0.2-ciscol.tar、csi-node-driver-registrar-2.0.1-ciscol.tarcsi-resizer-1.0.1-ciscol.tarcsi-provisioner-2.0.4-ciscol.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。 <a href="#">1</a>
-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、 <a href="#">前提条件 (9 ページ)</a> 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	クラスター管理 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 クラスター 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)

<sup>1</sup> 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。

### 手順

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 :

次の例は、各ノードにアップロードされた Cisco HyperFlex CSI コンテナイメージの展開を示しています。イメージ名は `hxcsi`、タグ名は `hxcsi-1.2.1-601` です。

```
administrator-host:hxcsi$ ./setup/hxcsi-setup -cluster-name demo-hxcsi
-hx-csi-image hxcsi-1.2.1-601
-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

## ヘルムユーティリティを使用した展開

ヘルムチャートパッケージを使用して Cisco HyperFlex CSI インテグレーションを展開するには、helm ユーティリティ (バージョン v3.5.2 以降) を実行する必要があります。次の表に、helm コマンドで指定できるパラメータを示します。

表 2: HELM コマンドパラメータ

パラメータ名	必須またはオプション	説明
Hx.clientId string	オプション	テナントの ClientID。 [表 1 の注を参照]
Hx.iscsiUrl 文字列	Required	HyperFlex クラスターの eth-iscsi1:0 インターフェイスの HyperFlex iSCSI クラスター IP アドレス。前提条件の詳細については、「 <a href="#">前提条件</a> 」を参照してください。
Hx.token string	Required	サービス認証トークン。helm インストールを呼び出す前に、帯域外でトークンを作成する必要があります。
Hx.url 文字列	Required	HyperFlex クラスター管理 IP アドレス。この IP はボリュームのプロビジョニングに使用されます。
hx.dockerRegistryName 文字列	Required	Docker レジストリ名 (例: mydockerhub.com/hx-docker)



(注) トークンを作成する方法の 1 つは、前述の hxcsi-setup ユーティリティを使用することです。トークンは、生成された「hxcsi-deploy/hxcsi-config.yaml」ファイルにあります。このトークンは、helm ユーティリティに渡す前に base64 でデコードする必要があります。



(注) 次の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。

### 手順

次の例は、各 Kubernetes ノードからアクセス可能な docker レジストリにアップロードされた Cisco HyperFlex CSI コンテナイメージの展開を示しています。

例 :

```

administrator-host:hxcsi$ helm install hxcsi hxcsi-1.2.1.tgz --set hx.url=10.2.17.13 \
--set hx.iscsiUrl=10.2.17.18 --set hx.clientId=myClientId \
--set hx.dockerRegistryName=mydockerhub.com/hx-docker --set hx.token=myToken123456
administrator-host:hxcsi$
NAME: hxcsi
LAST DEPLOYED: Wed May 26 15:01:59 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
administrator-host:hxcsi$ kubectl get pods
NAME READY STATUS RESTARTS AGE
csi-attacher-hxcsi-0 2/2 Running 0 9s
csi-nodeplugin-hxcsi-rm4h4 2/2 Running 0 9s
csi-nodeplugin-hxcsi-w5fxt 2/2 Running 0 9s
csi-provisioner-hxcsi-0 2/2 Running 0 9s
csi-resizer-hxcsi-0 2/2 Running 0 9s
administrator-host:hxcsi$
administrator-host:hxcsi$ helm status hxcsi
NAME: hxcsi
LAST DEPLOYED: Wed May 26 15:01:59 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1

```

## 次のタスク

Cisco HyperFlex CSI ストレージクラスを作成します。

# 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-1.2.1-601.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。

## 手順

- 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 1 つのインスタンス、`csi-provisioner-hxcsi` ポッドの 1 つのインスタンス、`csi-resizer-hxcsi` ポッドの 1 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-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-resizer-hxcsi-0                  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 3: 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、「hxpvcclaim-default」という名前のPVCを作成します。



#	ディレクトリ名	説明
2	sample-hxcsi-csi-clone	「sample-hxcsi」からクローンを作成するポッド。 「dataSource」は「hxpvcclaim-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
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	hxpvcclaim-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

## 手順

**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: "2000000000000000"
```

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:~# 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:"200000000000000"
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。

### 手順

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 :

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

```
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](#) (23 ページ)
- [管理者ホスト](#) (23 ページ)
- [ステートフルアプリケーションの展開](#) (24 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 4.5(2a) 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 Administration Guide, Release 4.5](#)』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。

### 手順

**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。

手順

**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、「/sqlldb」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: demovolumel
      persistentVolumeClaim:
        claimName: message-board-pvc
  containers:
    - name: message-board
      image: michzimm/message_board:version1
      ports:
        - containerPort: 5000
      volumeMounts:
        - mountPath: "/sqldb"
          name: demovolumel
---
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-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-resizer-hxcsi-5b444c8478-6qxws	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

# トラブルシューティング

- [トラブルシューティング \(29 ページ\)](#)
- [HXCSI ポッドの展開時の ImagePullBackOff ステータスエラー \(29 ページ\)](#)
- [ボリューム削除の失敗 \(30 ページ\)](#)
- [ノード削除中の ContainerCreating 状態のアプリケーション ポッド \(30 ページ\)](#)
- [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 \(31 ページ\)](#)

## 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-3.0.2-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`

## 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。

```
kubect1 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。

```
kubect1 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が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。

```
kubect1 drain <node-name>
kubect1 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 \(How to gracefully remove a node from Kubernetes?\)](#)」を参照してください。

## 削除されたポッドが同じノードに戻るよう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>
```

削除されたポッドが同じノードに戻るようスケジュールされている

## 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。