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

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
```

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。