BeeGFSクラスタの管理

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

目次

BeeGFSクラスタの管理

概要、主要な概念、用語

BeeGFS HAクラスタの導入後の管理方法について説明します。

概要

このセクションは、BeeGFS HAクラスタを導入したあとに管理する必要があるクラスタ管理者を対象としています。Linux HAクラスタに詳しい場合でも、このガイドをよくお読みください。クラスタの管理方法には、特にAnsibleによる再設定に関していくつかの違いがあるためです。

主な概念

これらの概念の一部はメイン"用語と概念"ページで紹介されていますが、BeeGFS HAクラスタのコンテキストで再紹介すると便利です。

クラスタノード: PacemakerサービスとCorosyncサービスを実行しており、HAクラスタに参加しているサーバ。

ファイルノード: 1つ以上のBeeGFS管理'メタデータ'またはストレージサービスを実行するために使用されるクラスタノード

ブロックノード:ファイルノードにブロックストレージを提供するNetApp Eシリーズストレージシステム。これらのノードは独自のスタンドアロンHA機能を提供するため、BeeGFS HAクラスタには参加しません。各ノードは、ブロックレイヤで高可用性を提供する2台のストレージコントローラで構成されます。

- BeeGFSサービス** BeeGFS管理'メタデータ'ストレージ・サービス各ファイルノードは、ブロックノード上のボリュームを使用してデータを格納する1つ以上のサービスを実行します。

ビルディングブロック： BeeGFSファイルノード、Eシリーズブロックノード、およびBeeGFSサービスを標準で導入し、NetApp Verified Architectureに基づくBeeGFS HAクラスタ/ファイルシステムのスケールアウトを簡易化します。カスタムHAクラスタもサポートされますが、多くの場合、拡張を簡易化するビルディングブロック方式が採用されています。

- BeeGFS HA Cluster：**ブロックノードによってサポートされるBeeGFSサービスの実行に使用される拡張可能なファイルノード数。BeeGFSデータを高可用性で保存します。業界で実証済みのオープンソースコンポーネントPacemakerとCorosyncにAnsibleを使用してパッケージと導入を行いました。

クラスタサービスは、クラスタに参加している各ノードでPacemakerサービスおよびCorosyncサービスを実行していることを意味します。注：ノードでBeeGFSサービスを実行せずに、2つのファイルノードしか必要としない場合は「Tiebreaker」ノードとしてクラスタに参加するだけで済みます。

クラスタリソース:クラスタ内で実行されているBeeGFSサービスごとに、BeeGFSモニタリソースと、BeeGFSターゲットのリソース、IPアドレス（フローティングIP）、BeeGFSサービスそのものを含むリソースグループが表示されます。

- Ansible：**ソフトウェアのプロビジョニング、構成管理、アプリケーション導入のためのツールで、コードとしてのインフラストラクチャを実現します。BeeGFS HAクラスタをパッケージ化することで、ネットアップのBeeGFSの導入、再構成、更新を簡易化します。

- PC:**クラスタ内の任意のファイルノードから使用可能なコマンドラインインターフェイス。クラスタ内のノードおよびリソースの状態を照会および制御するために使用します。

一般的な用語

フェールオーバー:各BeeGFSサービスには、そのノードで障害が発生しない限り、実行される優先ファイルノードがあります。BeeGFSサービスが非優先/セカンダリファイルノードで実行されている場合は'フェイルオーバー中と呼ばれます

フェイルバック:優先されないファイルノードから優先ノードにBeeGFSサービスを移動する動作。

- HAペア：**同じブロックノードのセットにアクセスできる2つのファイルノードは、HAペアと呼ばれることもあります。ネットアップ全体で、相互にテイクオーバーできる2台のストレージコントローラまたはノードを指します。

Maintenance Mode：すべてのリソース監視を無効にし、Pacemakerによるクラスタ内のリソースの移動や管理を防止します（の項も参照"メンテナンスモード"）。

- HAクラスタ:**クラスタ内の複数のノード間でフェイルオーバー可能なBeeGFSサービスを実行する1つ以上のファイルノードが'高可用性BeeGFSファイルシステムを作成します多くの場合、ファイルノードはHAペアに構成され、クラスタ内のBeeGFSサービスのサブセットを実行できるようになります。

AnsibleとPCSツールを使用するタイミング

HAクラスタの管理にAnsibleとPCコマンドラインツールを使用する必要があるのはどのような場合ですか？

外部のAnsibleコントロールノードからAnsibleを使用して、クラスタの導入と再設定のすべてのタスクを完了する必要があります。クラスタの状態の一時的な変更（ノードのスタンバイへの切り替えなど）は、通常、クラスタの1つのノード（デグレード状態でないノードやメンテナンスを予定しているノード）にログインし、PCSコマンドラインツールを使用して実行します。

リソース、制約、プロパティ、BeeGFSサービスなどのクラスタ構成を変更するには、必ずAnsibleを使用します。Ansibleのインベントリとプレイブックの最新のコピーを維持すること（変更を追跡するためのソース管理に理想的）は、クラスタの管理の一部です。設定に変更を加える必要がある場合は、インベントリを更新してAnsibleプレイブックを再実行し、BeeGFS HAロールをインポートします。

HAロールでは、クラスタをメンテナンスモードにしてから、BeeGFSまたはクラスタサービスを再起動して新しい設定を適用する前に必要な変更を行います。通常、最初の導入ではノードの完全なリブートは必要ありませんが、Ansibleの再実行は一般に「安全な」手順 とみなされます。ただし、メンテナンス時間中や、BeeGFSサービスの再起動が必要になった場合は、オフ時間帯に行うことを推奨します。このような再起動によって、通常は原因 アプリケーションエラーが発生することはありませんが、パフォーマンスが低下する可能性があります（一部のアプリケーションは他のアプリケーションよりも処理能力が高い）。

Ansibleの再実行も、クラスタ全体を最適な状態に戻したい場合に選択できます。場合によっては、PCを使用する場合よりも簡単にクラスタの状態をリカバリできる可能性があります。特に、何らかの理由でクラスタが停止した緊急時に、すべてのノードが稼働状態に戻ってからAnsibleを再実行すると、PCを使用するよりも迅速かつ確実にクラスタをリカバリできます。

クラスタの状態を確認します

PCを使用してクラスタの状態を表示します。

概要

実行中です `pcs status` いずれかのクラスタノードから、クラスタの全体的な状態と各リソースのステータス（BeeGFSサービスやその依存関係など）を確認する最も簡単な方法があります。このセクションでは、の出力で確認できる内容について説明します `pcs status` コマンドを実行します

からの出力を理解する `pcs status`

を実行します `pcs status` クラスタサービス（PacemakerとCorosync）が開始されているクラスタノードで実行します。出力の一番上にクラスタの概要が表示されます。

```
[root@beegfs_01 ~]# pcs status
Cluster name: hacluster
Cluster Summary:
  * Stack: corosync
  * Current DC: beegfs_01 (version 2.0.5-9.el8_4.3-ba59be7122) - partition
with quorum
  * Last updated: Fri Jul  1 13:37:18 2022
  * Last change:  Fri Jul  1 13:23:34 2022 by root via cibadmin on
beegfs_01
  * 6 nodes configured
  * 235 resource instances configured
```

次のセクションには、クラスタ内のノードが表示されます。

```
Node List:
  * Node beegfs_06: standby
  * Online: [ beegfs_01 beegfs_02 beegfs_04 beegfs_05 ]
  * OFFLINE: [ beegfs_03 ]
```

これは、スタンバイまたはオフラインのノードを示します。スタンバイ状態のノードは引き続きクラスタに参加していますが、リソースを実行する資格がありません。ノードがオフラインの場合は、そのノードでクラスタサービスが実行されていないことを示します。これは、手動で停止されているか、ノードがリブートまたはシャットダウンされたことが原因です。

ノードの初回起動時にクラスタサービスが停止し、リソースが異常なノードに誤ってフェイルバックされないように手動で開始する必要があります。

管理者以外の理由（障害など）によりノードがスタンバイまたはオフラインになっている場合、ノードの状態の横に括弧内に追加のテキストが表示されます。たとえば、フェンシングが無効で、リソースに障害が発生した場合などです `Node <HOSTNAME>: standby (on-fail)`。もう1つの状態はです `Node <HOSTNAME>:

UNCLEAN (offline)`を使用すると、ノードがフェンシングされていると短時間だけ表示されますが、クラスタがノードの状態を確認できないことを示すフェンシングが失敗した場合も維持されます（これにより、リソースが他のノードで開始されない場合があります）。

次のセクションでは、クラスタ内のすべてのリソースとその状態のリストを示します。

```
Full List of Resources:
  * mgmt-monitor    (ocf::eseries:beegfs-monitor):   Started beegfs_01
  * Resource Group: mgmt-group:
    * mgmt-FS1  (ocf::eseries:beegfs-target):    Started beegfs_01
    * mgmt-IP1  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-IP2  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-service  (systemd:beegfs-mgmtd):  Started beegfs_01
[...]
```

ノードと同様に、リソースに問題がある場合は、リソースの状態の横にかっこで囲んだテキストが追加で表示されます。たとえば、Pacemakerがリソースの停止を要求し、割り当てられた時間内に完了できなかった場合、Pacemakerはノードの遮断を試みます。フェンシングが無効になっているか、フェンシング処理が失敗した場合、リソースの状態はになります `FAILED <HOSTNAME> (blocked)` また、Pacemakerは別のノードで起動できません。

BeeGFS HAクラスタでは、最適化されたいくつかのBeeGFSカスタムOCFリソースエージェントを使用することに注意してください。特に、BeeGFSモニタは、特定のノードのBeeGFSリソースを使用できない場合にフェイルオーバーをトリガーします。

HAクラスタとBeeGFSを再設定します

Ansibleを使用してクラスタを再構成します。

概要

通常、BeeGFS HAクラスタの再設定は、Ansibleインベントリを更新して `ansible-playbook`コマンドを再実行することで行う必要があります。これには、アラートの更新、永続的なフェンシング設定の変更、BeeGFSサービス設定の調整などが含まれます。これらは `group_vars/ha_cluster.yml`ファイルを使用して調整され、オプションの完全なリストは"Common File Node Configurationを指定します"セクションにあります。

一部の設定オプションについては、以下を参照してください。これらのオプションを選択した場合、管理者は、クラスタのメンテナンスやサービスを行う際に注意が必要になります。

フェンシングを無効にして有効にする方法

フェンシングは、クラスタのセットアップ時にデフォルトで有効/必要になります。場合によっては、フェンシングを一時的に無効にして、特定のメンテナンス処理（オペレーティングシステムのアップグレードなど）の実行時にノードが誤ってシャットダウンされないようにすることが望ましい場合もあります。この機能は手動で無効にできますが、管理者が注意する必要があるトレードオフがあります。

オプション**1**：**Ansible**によるフェンシングを無効にします（推奨）。

Ansibleを使用してフェンシングを無効にすると、BeeGFSモニタの失敗時のアクションが「フェンス」から「standby」に変更されます。つまり、BeeGFSモニタが障害を検出すると、ノードがスタンバイ状態になり、すべてのBeeGFSサービスをフェイルオーバーしようとします。一般的に、オプション2よりも、外部のアクティブなトラブルシューティング/テストの方が望ましい。短所は、あるリソースが元のノードで停止しないと、そのリソースが他の場所からブロックされる場合です（そのため、通常は本番環境のクラスタでフェンシングが必要となります）。

1. Ansibleのインベントリで `groups_vars/ha_cluster.yml` 次の構成を追加します。

```
beegfs_ha_cluster_crm_config_options:
    stonith-enabled: False
```

2. Ansibleプレイブックを再実行して、クラスタに変更を適用します。

オプション**2**：フェンシングを手動で無効にする

場合によっては、Ansibleなしでフェンシングを一時的に無効にすることもできます。たとえば、クラスタのトラブルシューティングやテストを実施する場合などです。

この設定では、BeeGFSモニタが障害を検出すると、クラスタは対応するリソースグループの停止を試みます。フルフェイルオーバーはトリガーされず、影響を受けるリソースグループを再起動したり別のホストに移動したりすることもありません。リカバリするには、問題を解決してからを実行します `pcs resource cleanup` または、手動でノードをスタンバイにします。

手順

1. フェンシング（stonith）がグローバルに有効になっているか無効になっているかを確認するには、次のコマンドを実行 `pcs property show stonith-enabled`
2. フェンシングをディセーブルにするには、 `pcs property set stonith-enabled=false`
3. フェンシングを有効にするには、次の `pcs property set stonith-enabled=true`

この設定は、次回 Ansible プレイブックを実行するときに上書きされます。

HAクラスタコンポーネントの更新

BeeGFS サービスのアップグレード

Ansible を使用して、HA クラスターで実行されている BeeGFS バージョンを更新します。

概要

BeeGFSには `major.minor.patch`バージョン管理方式が採用されています。`BeeGFS HA Ansible`のロールは、サポートされる `major.minor`バージョンごとに用意されています（ `beegfs_ha_7_2`やなど

`beegfs_ha_7_3`）。各HAロールは、Ansibleコレクションのリリース時に利用可能な最新のBeeGFSパッチバージョンに固定されています。

BeeGFSのすべてのアップグレードにはAnsibleを使用する必要があります。これには、BeeGFSのメジャーバージョン、マイナーバージョン、パッチバージョン間の移行も含まれます。BeeGFSをアップデートするには、まずBeeGFS Ansibleコレクションをアップデートする必要があります。これにより、デプロイメント/管理自動化と基盤となるHAクラスタの最新の修正と機能強化も取得されます。コレクションを最新バージョンにアップデートした後でも、 `-e "beegfs_ha_force_upgrade=true"`を設定して `ansible-playbook`を実行するまでBeeGFSはアップグレードされません。各アップグレードの詳細については、現在のバージョンの"BeeGFSアップグレードのドキュメント"を参照してください。

> BeeGFS v8 にアップグレードする場合は、代わりに"BeeGFS v8にアップグレード"手順を参照してください。

テスト済みのアップグレードパス

次のアップグレード パスがテストおよび検証されています：

元のバージョン	アップグレードバージョン	マルチレール	詳細
7.2.6.	7.3.2の場合	はい。	beegfsコレクションをv3.0.1からv3.1.0にアップグレードすると'マルチレールが追加されました
7.2.6.	7.2.8	いいえ	beegfsコレクションをv3.0.1からv3.1.0にアップグレードしています
7.2.8	7.3.1	はい。	beegfs collection v3.1.0を使用してアップグレードすると、マルチレールが追加されました
7.3.1	7.3.2の場合	はい。	beegfs collection v3.1.0を使用してアップグレードします
7.3.2の場合	7.4.1	はい。	beegfs collection v3.2.0を使用してアップグレードします
7.4.1	7.4.2	はい。	beegfs collection v3.2.0を使用してアップグレードします
7.4.2	7.4.6	はい。	beegfs collection v3.2.0を使用してアップグレードします
7.4.6	8.0	はい。	"BeeGFS v8にアップグレード"手順の指示に従ってアップグレードします。
7.4.6	8.1	はい。	"BeeGFS v8にアップグレード"手順の指示に従ってアップグレードします。
7.4.6	8.2	はい。	"BeeGFS v8にアップグレード"手順の指示に従ってアップグレードします。

BeeGFSのアップグレード手順

次のセクションでは、BeeGFS AnsibleコレクションとBeeGFS自体を更新する手順を示します。BeeGFSのメジャーバージョンまたはマイナーバージョンを更新する際は、特に注意してください。

手順**1**：**BeeGFS**コレクションをアップグレードする

へのアクセスによる収集のアップグレード "Ansible Galaxy"を使用して、次のコマンドを実行します。

```
ansible-galaxy collection install netapp_eseries.beegfs --upgrade
```

オフラインでの収集アップグレードの場合は、から収集をダウンロードします "Ansible Galaxy" 目的のをクリックします `Install Version` 次に `Download tarball`。tarballをAnsibleコントロールノードに転送し、次のコマンドを実行します。

```
ansible-galaxy collection install netapp_eseries-beegfs-<VERSION>.tar.gz
--upgrade
```

を参照してください "コレクションのインストール" を参照してください。

ステップ2：Ansibleインベントリを更新する

クラスタのAnsibleインベントリファイルに必要な更新や希望する更新を行ってください。具体的なアップグレード要件の詳細については、以下の[バージョンのアップグレードに関する注意事項]セクションをご覧ください。BeeGFS HAインベントリの設定に関する一般的な情報については、"Ansibleのインベントリの概要"セクションをご覧ください。

手順3：Ansible Playbookを更新する（メジャーバージョンまたはマイナーバージョンを更新する場合のみ）

メジャーバージョンとマイナーバージョンを切り替える場合は、クラスタの導入とメンテナンスに使用するファイルで、 `playbook.yml` ロールの名前を目的のバージョンに合わせて更新し `beegfs_ha_<VERSION>` ます。たとえば、BeeGFS 7.4を導入する場合は、次のようになり `beegfs_ha_7_4` ます。

```
- hosts: all
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.beegfs
  tasks:
    - name: Ensure BeeGFS HA cluster is setup.
      ansible.builtin.import_role:  # import_role is required for tag
availability.
        name: beegfs_ha_7_4
```

このPlaybookファイルの内容の詳細については、を参照してください"BeeGFS HAクラスタを導入します"。

手順4：BeeGFSのアップグレードを実行する

BeeGFSアップデートを適用するには：

```
ansible-playbook -i inventory.yml beegfs_ha_playbook.yml -e
"beegfs_ha_force_upgrade=true" --tags beegfs_ha
```

BeeGFS HAの役割では、次の処理が行われます。

- 各BeeGFSサービスが優先ノードに配置された状態で、クラスタが最適な状態であることを確認します。
- クラスタをメンテナンスモードにします。
- HAクラスタのコンポーネントを更新します（必要な場合）。
- 各ファイルノードを次のように1つずつアップグレードします。
  - スタンバイにし、サービスをセカンダリノードにフェイルオーバーします。
  - BeeGFSパッケージをアップグレードします。
  - フォールバックサービス。
- クラスタをメンテナンスモードから切り替えます。

バージョンのアップグレードに関する注意事項

BeeGFSバージョン**7.2.6**または**7.3.0**からアップグレードしています

接続ベースの認証に対する変更

BeeGFS バージョン 7.3.2 以降では、接続ベースの認証を設定する必要があります。次のいずれかを行わないと、サービスは起動しません：

- `connAuthFile`の指定、または
- サービスの構成ファイルで `connDisableAuthentication=true`を設定する。

セキュリティのため、接続ベースの認証を有効にすることを強くお勧めします。詳細については、"BeeGFS Connection Based Authenticationの略"をご覧ください。

`beegfs_ha*`ロールは認証ファイルを自動的に生成し、次の場所に配布します：

- クラスター内のすべてのファイルノード
- `<playbook_directory>/files/beegfs/<beegfs_mgmt_ip_address>_connAuthFile`のAnsibleコントロールノード

`beegfs_client`ロールは、このファイルが存在する場合、それを自動的に検出してクライアントに適用します。

`beegfs_client`ロールを使用してクライアントを設定しなかった場合は、各クライアントに認証ファイルを手動で配布し、 `beegfs-client.conf`ファイル内の `connAuthFile`設定を構成する必要があります。接続ベースの認証がないBeeGFSバージョンからアップグレードする場合、 `group_vars/ha_cluster.yml`で `beegfs_ha_conn_auth_enabled: false`を設定してアップグレード中に接続ベースの認証を無効にしない限り、クライアントはアクセスできなくなります（推奨されません）。

追加の詳細と代替構成オプションについては、"Common File Node Configurationを指定します" セクションの接続認証構成手順を参照してください。

BeeGFS v8にアップグレード

BeeGFS HA クラスターをバージョン 7.4.6 から BeeGFS v8 にアップグレードするには、次の手順に従います。

概要

BeeGFS v8では、BeeGFS v7からアップグレードする前に追加の設定が必要となる重要な変更がいくつか導入されています。このドキュメントでは、BeeGFS v8の新しい要件に合わせてクラスターを準備し、BeeGFS v8にアップグレードする方法について説明します。

> ⓘ BeeGFS v8にアップグレードする前に、システムが少なくともBeeGFS 7.4.6を実行していることを確認してください。BeeGFS 7.4.6より前のリリースを実行しているクラスターは、このBeeGFS v8アップグレード手順を進める前に、まず"バージョン7.4.6にアップグレード"を行う必要があります。

BeeGFS v8の主な変更点

BeeGFS v8 では、次の主要な変更が導入されています：

- ライセンスの適用： BeeGFS v8では、ストレージプール、リモートストレージターゲット、BeeONDなどのプレミアム機能を使用するにはライセンスが必要です。アップグレード前に、BeeGFSクラスターの有効なライセンスを取得してください。必要に応じて、一時的なBeeGFS v8評価ライセンスを"BeeGFSライセンスポータル"から取得できます。
- 管理サービス データベースの移行： BeeGFS v8 の新しい TOML ベース形式による構成を有効にするには、BeeGFS v7 管理サービス データベースを更新された BeeGFS v8 形式に手動で移行する必要があります。
- **TLS**暗号化： BeeGFS v8では、サービス間の安全な通信のためにTLSが導入されています。アップグレードの一環として、BeeGFS管理サービスと `beegfs`コマンドラインユーティリティ用のTLS証明書を生成して配布する必要があります。

BeeGFS 8 の詳細と追加の変更については、"BeeGFS v8.0.0 アップグレードガイド"を参照してください。

> ⓘ BeeGFS v8へのアップグレードには、クラスタのダウンタイムが必要です。また、BeeGFS v7クライアントはBeeGFS v8クラスタに接続できません。運用への影響を最小限に抑えるため、クラスタとクライアント間のアップグレードタイミングを慎重に調整してください。

BeeGFS クラスターをアップグレード用に準備する

アップグレードを開始する前に、スムーズな移行を実現し、ダウンタイムを最小限に抑えるために環境を慎重に準備してください。

1. クラスターが正常な状態であり、すべてのBeeGFSサービスがそれぞれの優先ノードで実行されていることを確認してください。BeeGFSサービスを実行しているファイルノードから、すべてのPacemakerリソースがそれぞれの優先ノードで実行されていることを確認します：

```
pcs status
```

2. クラスター構成を記録してバックアップします。
   a. クラスタ構成のバックアップ手順については、"BeeGFS Backupのドキュメント"を参照してください。
   b. 既存の管理データ ディレクトリをバックアップします：

```
cp -r /mnt/mgmt_tgt_mgmt01/data
/mnt/mgmt_tgt_mgmt01/data_beegfs_v7_backup_$(date +%Y%m%d)
```

   c. beegfs クライアントから次のコマンドを実行し、出力を参照用に保存します：

```
beegfs-ctl --getentryinfo --verbose /path/to/beegfs/mountpoint
```

   d. ミラーリングを使用する場合は、詳細な状態情報を収集します：

```
beegfs-ctl --listtargets --longnodes --state --spaceinfo
--mirrorgroups --nodetype=meta
beegfs-ctl --listtargets --longnodes --state --spaceinfo
--mirrorgroups --nodetype=storage
```

3. クライアントのダウンタイムに備えて `beegfs-client`サービスを停止します。各クライアントで以下を実行します：

```
systemctl stop beegfs-client
```

4. 各Pacemakerクラスタで、STONITHを無効にします。これにより、不要なノードの再起動をトリガーすることなく、アップグレード後にクラスタの整合性を検証できます。

```
pcs property set stonith-enabled=false
```

5. BeeGFS 名前空間内のすべての Pacemaker クラスターでは、PCS を使用してクラスターを停止します：

```
pcs cluster stop --all
```

BeeGFSパッケージをアップグレードする

クラスター内のすべてのファイルノードに、Linuxディストリビューション用のBeeGFS v8パッケージリポジトリを追加してください。公式BeeGFSリポジトリの使用方法については、"BeeGFSダウンロードページ"をご覧ください。それ以外の場合は、ローカルのbeegfsミラーリポジトリを適切に設定してください。

以下の手順は、RHEL 9 ファイルノード上の公式 BeeGFS 8.2 リポジトリを使用した手順です。クラスター内のすべてのファイルノードで以下の手順を実行してください：

1. BeeGFS GPG キーをインポートします：

```
rpm --import https://www.beegfs.io/release/beegfs_8.2/gpg/GPG-KEY-beegfs
```

2. BeeGFS リポジトリをインポートします：

```
curl -L -o /etc/yum.repos.d/beegfs-rhel9.repo
https://www.beegfs.io/release/beegfs_8.2/dists/beegfs-rhel9.repo
```

> ⓘ 新しい BeeGFS v8 リポジトリとの競合を避けるため、以前に設定された BeeGFS リポジトリを削除します。

3. パッケージ マネージャーのキャッシュを消去します：

```
dnf clean all
```

4. すべてのファイル ノードで、BeeGFS パッケージを BeeGFS 8.2 に更新します。

```
dnf update beegfs-mgmtd beegfs-storage beegfs-meta libbeegfs-ib
```

> ⓘ 標準クラスターでは、 `beegfs-mgmtd`パッケージは最初の2つのファイル ノードでのみ更新されます。

管理データベースをアップグレードする

BeeGFS 管理サービスを実行しているファイルノードの 1 つで、次の手順を実行して、管理データベースを BeeGFS v7 から v8 に移行します。

1. すべての NVMe デバイスを一覧表示し、管理対象をフィルタリングします：

```
nvme netapp smdevices | grep mgmt_tgt
```

   a. 出力からデバイスパスをメモします。

   b. 管理対象デバイスを既存の管理対象マウント ポイントにマウントします（ `/dev/nvmeXnY`をデバイス パスに置き換えます）：

```
mount /dev/nvmeXnY /mnt/mgmt_tgt_mgmt01/
```

2. 次のコマンドを実行して、BeeGFS 7 管理データを新しいデータベース形式にインポートします：

```
/opt/beegfs/sbin/beegfs-mgmtd --import-from
-v7=/mnt/mgmt_tgt_mgmt01/data/
```

期待される出力：

```
Created new database version 3 at "/var/lib/beegfs/mgmtd.sqlite".
Successfully imported v7 management data from
"/mnt/mgmt_tgt_mgmt01/data/".
```

BeeGFS v8では検証要件が厳格化されているため、自動インポートが必ずしも成功するとは限りません。例えば、ターゲットが存在しないストレージプールに割り当てられている場合、インポートは失敗します。移行に失敗した場合は、アップグレードを続行しないでください。データベース移行問題の解決については、NetAppサポートにお問い合わせください。当面の解決策として、BeeGFS v8パッケージをダウングレードし、問題が解決するまでBeeGFS v7を引き続き実行することができます。

3. 生成された SQLite ファイルを管理サービス マウントに移動します：

```
mv /var/lib/beegfs/mgmtd.sqlite /mnt/mgmt_tgt_mgmt01/data/
```

4. 生成された `beegfs-mgmtd.toml`を管理サービスマウントに移動します：

```
mv /etc/beegfs/beegfs-mgmtd.toml /mnt/mgmt_tgt_mgmt01/mgmt_config/
```

`beegfs-mgmtd.toml`構成ファイルの準備は、次のセクションのライセンスおよび TLS 構成手順を完了した後に行われます。

ライセンスを構成する

1. beegfs管理サービスを実行するすべてのノードにbeegfsライセンスパッケージをインストールします。通常はクラスタの最初の2つのノードです：

```
dnf install libbeegfs-license
```

2. BeeGFS v8 ライセンス ファイルを管理ノードにダウンロードし、次の場所に配置します：

```
/etc/beegfs/license.pem
```

TLS暗号化を構成する

BeeGFS v8では、管理サービスとクライアント間の安全な通信のためにTLS暗号化が必要です。管理サービスとクライアントサービス間のネットワーク通信でTLS暗号化を設定するには、3つのオプションがあります。推奨される最も安全な方法は、信頼できる証明機関によって署名された証明書を使用することです。あるいは、独自のローカルCAを作成して、BeeGFSクラスターの証明書に署名することもできます。暗号化が不要な環境やトラブルシューティングの場合は、TLSを完全に無効にすることもできますが、機密情報がネットワークに公開されるため、これは推奨されません。

続行する前に、"BeeGFS 8 の TLS 暗号化を設定する"ガイドの指示に従って、環境のTLS暗号化を設定してください。

更新管理サービスの構成

BeeGFS v7 構成ファイルから設定を手動で `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml`ファイルに転送して、BeeGFS v8 管理サービス構成ファイルを準備します。

1. 管理ターゲットがマウントされている管理ノードで、 `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.conf`BeeGFS 7の管理サービスファイルを参照し、すべての設定を `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml`ファイルに転送します。基本的な設定では、`beegfs-mgmtd.toml`は以下のようになります：

   ```
   beemsg-port = 8008
   grpc-port = 8010
   log-level = "info"
   node-offline-timeout = "900s"
   quota-enable = false
   auth-disable = false
   auth-file = "/etc/beegfs/<mgmt_service_ip>_connAuthFile"
   db-file = "/mnt/mgmt_tgt_mgmt01/data/mgmtd.sqlite"
   license-disable = false
   license-cert-file = "/etc/beegfs/license.pem"
   tls-disable = false
   tls-cert-file = "/etc/beegfs/mgmtd_tls_cert.pem"
   tls-key-file = "/etc/beegfs/mgmtd_tls_key.pem"
   interfaces = ['i1b:mgmt_1', 'i2b:mgmt_2']
   ```

   必要に応じて、環境と TLS 構成に合わせてすべてのパスを調整します。

2. 管理サービスを実行している各ファイル ノードで、新しい構成ファイルの場所を指すように systemd サービス ファイルを変更します。

   ```
   sudo sed -i 's|ExecStart=.*|ExecStart=nice -n -3
   /opt/beegfs/sbin/beegfs-mgmtd --config-file
   /mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml|'
   /etc/systemd/system/beegfs-mgmtd.service
   ```

a. systemdをリロードします：

```
systemctl daemon-reload
```

3. 管理サービスを実行している各ファイル ノードに対して、管理サービスの gRPC 通信用にポート 8010 を開きます。

a. ポート 8010/tcp を beegfs ゾーンに追加します：

```
sudo firewall-cmd --zone=beegfs --permanent --add-port=8010/tcp
```

b. 変更を適用するには、ファイアウォールをリロードします：

```
sudo firewall-cmd --reload
```

BeeGFSモニタースクリプトを更新する

Pacemaker `beegfs-monitor` OCFスクリプトは、新しいTOML構成形式とsystemdサービス管理をサポートするために更新が必要です。クラスター内の1つのノードでスクリプトを更新し、更新したスクリプトを他のすべてのノードにコピーしてください。

1. 現在のスクリプトのバックアップを作成します：

```
cp /usr/lib/ocf/resource.d/eseries/beegfs-monitor
/usr/lib/ocf/resource.d/eseries/beegfs-monitor.bak.$(date +%F)
```

2. 管理構成ファイルのパスを `.conf`から `.toml`に更新します：

```
sed -i 's|mgmt_config/beegfs-mgmtd\.conf|mgmt_config/beegfs-mgmtd.toml|'
/usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

または、スクリプト内で次のブロックを手動で見つけます：

```
case $type in
  management)
    conf_path="${configuration_mount}/mgmt_config/beegfs-mgmtd.conf"
    ;;
```

これを次のように置き換えます：

```
case $type in
  management)
    conf_path="${configuration_mount}/mgmt_config/beegfs-mgmtd.toml"
    ;;
```

3. `get_interfaces()`および `get_subnet_ips()`関数を更新して、TOML構成をサポートします：
   a. テキスト エディターでスクリプトを開きます：

   ```
   vi /usr/lib/ocf/resource.d/eseries/beegfs-monitor
   ```

   b. 2 つの関数 `get_interfaces()`と `get_subnet_ips()`を見つけます。
   c. `get_interfaces()`から `get_subnet_ips()`の終わりまで、両方の関数全体を削除します。
   d. 次の更新された関数をコピーして、その場所に貼り付けます：

```
# Return network communication interface name(s) from the BeeGFS
resource's connInterfaceFile
get_interfaces() {
  # Determine BeeGFS service network IP interfaces.
  if [ "$type" = "management" ]; then
    interfaces_line=$(grep "^interfaces =" "$conf_path")
    interfaces_list=$(echo "$interfaces_line" | sed "s/.*= \[\(.*
\)\]/\1/")
    interfaces=$(echo "$interfaces_list" | tr -d "'" | tr -d " " | tr
',' '\n')

    for entry in $interfaces; do
      echo "$entry" | cut -d ':' -f 1
    done
  else
    connInterfacesFile_path=$(grep "^connInterfacesFile" "$conf_path"
| tr -d "[:space:]" | cut -f 2 -d "=")

    if [ -f "$connInterfacesFile_path" ]; then
      while read -r entry; do
        echo "$entry" | cut -f 1 -d ':'
      done < "$connInterfacesFile_path"
    fi
  fi
}

# Return list containing all the BeeGFS resource's usable IP
addresses. *Note that these are filtered by the connNetFilterFile
entries.
get_subnet_ips() {
  # Determine all possible BeeGFS service network IP addresses.
  if [ "$type" != "management" ]; then
    connNetFilterFile_path=$(grep "^connNetFilterFile" "$conf_path" |
tr -d "[:space:]" | cut -f 2 -d "=")

    filter_ips=""
    if [ -n "$connNetFilterFile_path" ] && [ -e
$connNetFilterFile_path ]; then
      while read -r filter; do
        filter_ips="$filter_ips $(get_ipv4_subnet_addresses $filter)"
      done < $connNetFilterFile_path
    fi

    echo "$filter_ips"
  fi
}
```

e. テキストエディターを保存して終了します。

f. 続行する前に、次のコマンドを実行してスクリプトの構文エラーを確認してください。出力がない場合は、スクリプトの構文が正しいことを示します。

```
bash -n /usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

4. 更新された `beegfs-monitor` OCF スクリプトをクラスター内の他のすべてのノードにコピーして一貫性を確保します：

```
scp /usr/lib/ocf/resource.d/eseries/beegfs-monitor
user@node:/usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

クラスタをオンラインに戻す

1. これまでのアップグレード手順がすべて完了したら、すべてのノードで BeeGFS サービスを開始して、クラスターをオンラインに戻します。

```
pcs cluster start --all
```

2. `beegfs-mgmtd`サービスが正常に開始されたことを確認します：

```
journalctl -xeu beegfs-mgmtd
```

予想される出力には次のような行が含まれます：

```
Started Cluster Controlled beegfs-mgmtd.
Loaded config file from "/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-
mgmtd.toml"
Successfully initialized certificate verification library.
Successfully loaded license certificate: TMP-113489268
Opened database at "/mnt/mgmt_tgt_mgmt01/data/mgmtd.sqlite"
Listening for BeeGFS connections on [::]:8008
Serving gRPC requests on [::]:8010
```

ジャーナル ログにエラーが表示される場合は、管理構成ファイルのパスを確認し、すべての値が BeeGFS 7 構成ファイルから正しく転送されていることを確認します。

3. 実行 `pcs status`して、クラスターが正常であり、優先ノードでサービスが開始されていることを確認します。

4. クラスターが正常であることが確認されたら、STONITH を再度有効にします：

```
pcs property set stonith-enabled=true
```

5. 次のセクションに進み、クラスター内の BeeGFS クライアントをアップグレードし、BeeGFS クラスターの健全性を確認します。

BeeGFSクライアントのアップグレード

クラスターを BeeGFS v8 に正常にアップグレードした後、すべての BeeGFS クライアントもアップグレードする必要があります。

次の手順では、Ubuntu ベースのシステムで BeeGFS クライアントをアップグレードするプロセスの概要を説明します。

1. まだ行っていない場合は、BeeGFS クライアント サービスを停止します：

```
systemctl stop beegfs-client
```

2. お使いのLinuxディストリビューションにBeeGFS v8パッケージリポジトリを追加してください。公式BeeGFSリポジトリの使い方については、"BeeGFSダウンロードページ"をご覧ください。それ以外の場合は、ローカルのBeeGFSミラーリポジトリを適切に設定してください。

   次の手順では、Ubuntu ベースのシステム上の公式 BeeGFS 8.2 リポジトリを使用します：

3. BeeGFS GPG キーをインポートします：

```
wget https://www.beegfs.io/release/beegfs_8.2/gpg/GPG-KEY-beegfs -O
/etc/apt/trusted.gpg.d/beegfs.asc
```

4. リポジトリ ファイルをダウンロードします：

```
wget https://www.beegfs.io/release/beegfs_8.2/dists/beegfs-noble.list -O
/etc/apt/sources.list.d/beegfs.list
```

   新しい BeeGFS v8 リポジトリとの競合を避けるため、以前に設定された BeeGFS リポジトリを削除します。

5. BeeGFS クライアント パッケージをアップグレードします：

```
apt-get update
apt-get install --only-upgrade beegfs-client
```

6. クライアントのTLSを設定します。BeeGFS CLIを使用するにはTLSが必要です。"BeeGFS 8 の TLS 暗号化を設定する"の手順を参照して、クライアントでTLSを設定してください。

7. BeeGFS クライアント サービスを開始します：

```
systemctl start beegfs-client
```

アップグレードを確認する

BeeGFS v8 へのアップグレードが完了したら、次のコマンドを実行して、アップグレードが成功したことを確認します。

1. ルートinodeが以前と同じメタデータノードによって所有されていることを確認します。管理サービスで`import-from-v7`機能を使用した場合、これは自動的に行われるはずです：

```
beegfs entry info /mnt/beegfs
```

2. すべてのノードとターゲットがオンラインで、良好な状態であることを確認します：

```
beegfs health check
```

「使用可能な容量」チェックでターゲットの空き容量が少ないという警告が表示された場合は、`beegfs-mgmtd.toml`ファイルに定義されている「容量プール」しきい値を調整して、環境に適したものにすることができます。

HAクラスタでのPacemakerおよびCorosyncパッケージのアップグレード

HAクラスタ内のPacemakerおよびCorosyncパッケージをアップグレードする手順は、次のとおりです。

概要

PacemakerとCorosyncをアップグレードすることで、クラスタは新機能、セキュリティパッチ、およびパフォーマンスの向上の恩恵を受けることができます。

アップグレードアプローチ

クラスタのアップグレードには、ローリングアップグレードとクラスタの完全なシャットダウンの2つの方法が推奨されます。各アプローチには独自の利点と欠点があります。アップグレード手順は、Pacemakerのリリースバージョンによって異なる場合があります。使用するアプローチを決定するには、ClusterLabsのドキュメントを参照して"Pacemakerクラスタのアップグレード"ください。アップグレードアプローチに従う前に、次のことを確認してください。

- 新しいPacemakerおよびCorosyncパッケージは、NetApp BeeGFSソリューション内でサポートされています。
- BeeGFSファイルシステムおよびPacemakerクラスタ構成に対して有効なバックアップが存在します。
- クラスタは正常な状態です。

この方法では、各ノードをクラスタから削除してアップグレードし、すべてのノードで新しいバージョンが実行されるまでクラスタに再導入します。この方法ではクラスタの運用を維持できるため、大規模なHAクラスタには最適ですが、処理中にバージョンが混在するリスクがあります。この方法は2ノードクラスタでは使用しないでください。

1. 各BeeGFSサービスが優先ノードで実行され、クラスタが最適な状態であることを確認します。詳細については、を参照してください "クラスタの状態を確認します" 。
2. ノードをアップグレードするには、ノードをスタンバイモードにして、すべてのBeeGFSサービスを停止（または移動）します。

```
pcs node standby <HOSTNAME>
```

3. 次のコマンドを実行して、ノードのサービスが削除されたことを確認します。

```
pcs status
```

スタンバイのノードでサービスがとして報告されていないことを確認します `Started`。

> クラスタのサイズによっては、サービスが姉妹ノードに移動するまでに数秒から数分かかることがあります。姉妹ノードでBeeGFSサービスが開始されない場合は、を参照してください"トラブルシューティングガイド"。

4. ノードのクラスタをシャットダウンします。

```
pcs cluster stop <HOSTNAME>
```

5. ノードのPacemaker、Corosync、およびPCsパッケージをアップグレードします。

> パッケージマネージャのコマンドは、オペレーティングシステムによって異なります。次のコマンドは、RHEL 8以降を実行するシステム用です。

```
dnf update pacemaker-<version>
```

```
dnf update corosync-<version>
```

```
dnf update pcs-<version>
```

6. ノードでPacemakerクラスタサービスを開始します。

```
pcs cluster start <HOSTNAME>
```

7. パッケージが更新された場合は `pcs`、クラスタでノードを再認証します。

```
pcs host auth <HOSTNAME>
```

8. ツールを使用して、ペースメーカーの設定がまだ有効であることを確認します `crm_verify`。

> この検証は、クラスタのアップグレード時に1回だけ実行します。

```
crm_verify -L -V
```

9. ノードのスタンバイを解除します。

```
pcs node unstandby <HOSTNAME>
```

10. すべてのBeeGFSサービスを優先ノードに再配置します。

```
pcs resource relocate run
```

11. クラスタ内の各ノードで上記の手順を繰り返して、すべてのノードで目的のバージョンのペースメーカー、Corosync、およびPCが実行されるようにします。
12. 最後に、を実行し `pcs status`てクラスタが正常であることを確認し、で `Current DC`目的のPacemakerバージョンが報告されます。

> 「mixed-version」と表示される場合 `Current DC`は、クラスタ内のノードが以前のバージョンのPacemakerで実行されているため、アップグレードが必要です。アップグレードしたノードがクラスタに再参加できない場合、またはリソースが起動しない場合は、クラスタログを確認し、アップグレードの既知の問題についてPacemakerのリリースノートまたはユーザガイドを参照してください。

クラスタのシャットダウン後の処理

この方法では、すべてのクラスタノードとリソースをシャットダウンし、ノードをアップグレードしてから、クラスタを再起動します。この方法は、PacemakerバージョンとCorosyncバージョンが混在した構成をサポートしていない場合に必要です。

1. 各BeeGFSサービスが優先ノードで実行され、クラスタが最適な状態であることを確認します。詳細については、を参照してください "クラスタの状態を確認します" 。
2. すべてのノードでクラスタソフトウェア（PacemakerおよびCorosync）をシャットダウンします。

クラスタのサイズによっては、クラスタ全体が停止するまでに数秒から数分かかることがあります。

```
pcs cluster stop --all
```

3. すべてのノードでクラスタサービスがシャットダウンしたら、要件に応じて各ノードのPacemaker、Corosync、およびPCsパッケージをアップグレードします。

パッケージマネージャのコマンドは、オペレーティングシステムによって異なります。次のコマンドは、RHEL 8以降を実行するシステム用です。

```
dnf update pacemaker-<version>
```

```
dnf update corosync-<version>
```

```
dnf update pcs-<version>
```

4. すべてのノードをアップグレードしたら、すべてのノードでクラスタソフトウェアを起動します。

```
pcs cluster start --all
```

5. `pcs`パッケージが更新された場合は、クラスタ内の各ノードを再認証します。

```
pcs host auth <HOSTNAME>
```

6. 最後に、を実行し `pcs status`てクラスタが正常であることを確認し、で `Current DC`正しいPacemakerバージョンが報告されます。

「mixed-version」と表示される場合 `Current DC`は、クラスタ内のノードが以前のバージョンのPacemakerで実行されているため、アップグレードが必要です。

ファイルノードアダプタファームウェアの更新

次の手順に従って、ファイルノードのConnectX-7アダプタを最新のファームウェアに更新します。

概要

新しいMLNX_OFEDドライバのサポート、新機能の有効化、バグの修正には、ConnectX-7アダプタファーム

ウェアの更新が必要になる場合があります。このガイドでは、使いやすさと効率性を考慮して、アダプタのアップデートにNVIDIAのユーティリティを使用し `mlxfwmanager`ます。

アップグレード時の考慮事項

このガイドでは、ConnectX-7アダプタのファームウェアを更新する2つの方法（ローリング更新と2ノードクラスタ更新）について説明します。クラスタのサイズに応じて、適切な更新方法を選択します。ファームウェアの更新を実行する前に、次のことを確認します。

- サポートされているMLNX_OFEDドライバがインストールされている場合は、を参照してください。"テクノロジ要件"
- BeeGFSファイルシステムおよびPacemakerクラスタ構成に対して有効なバックアップが存在します。
- クラスタは正常な状態です。

ファームウェアアップデートの準備

NVIDIAのユーティリティを使用して、NVIDIAのMLNX_OFEDドライバにバンドルされているノードのアダプタファームウェアを更新することを推奨し `mlxfwmanager`ます。アップデートを開始する前に、アダプタのファームウェアイメージをからダウンロードし"NVIDIAのサポートサイト"、各ファイルノードに保存します。

Lenovo ConnectX-7アダプタの場合は、NVIDIAのページにあるツールを"OEM ファームウェア"使用します `mlxfwmanager_LES`。

ローリング更新アプローチ

ノードが3つ以上のHAクラスタでは、この方法が推奨されます。このアプローチでは、一度に1つのファイルノードのアダプタファームウェアを更新して、HAクラスタが要求の処理を継続できるようにしますが、この間はI/Oの処理を行わないことを推奨します。

1. 各BeeGFSサービスが優先ノードで実行され、クラスタが最適な状態であることを確認します。詳細については、を参照してください "クラスタの状態を確認します" 。
2. 更新するファイルノードを選択し、スタンバイモードにします。スタンバイモードでは、そのノードからすべてのBeeGFSサービスが削除（または移動）されます。

```
pcs node standby <HOSTNAME>
```

3. 次のコマンドを実行して、ノードのサービスが削除されたことを確認します。

```
pcs status
```

スタンバイ状態のノードでサービスがとして報告されていないことを確認します `Started`。

クラスタのサイズによっては、BeeGFSサービスが姉妹ノードに移動するまでに数秒から数分かかることがあります。姉妹ノードでBeeGFSサービスが開始されない場合は、を参照してください"トラブルシューティングガイド"。

4. を使用してアダプタファームウェアを更新し `mlxfwmanager`ます。

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

ファームウェアのアップデートを受信している各アダプタのをメモします `PCI Device Name`。

5. ユーティリティを使用して各アダプタをリセットし、 `mlxfwreset`新しいファームウェアを適用します。

> 一部のファームウェアアップデートでは、アップデートを適用するために再起動が必要になる場合があります。詳細については、を参照してください"NVIDIAのmlxfwresetの制限事項"。リブートが必要な場合は、アダプタをリセットする代わりにリブートを実行します。

a. opensmサービスを停止します。

```
systemctl stop opensm
```

b. 前述の各コマンドについて、次のコマンドを実行し `PCI Device Name`ます。

```
mlxfwreset -d <pci_device_name> reset -y
```

c. opensmサービスを開始します。

```
systemctl start opensm
```

d. 再起動する `eseries_nvme_ib.service`。

```
systemctl restart eseries_nvme_ib.service
```

e. E シリーズ ストレージ アレイのボリュームが存在することを確認します。

```
multipath -ll
```

1. を実行し `ibstat`、すべてのアダプタが目的のファームウェアバージョンで実行されていることを確認します。

```
ibstat
```

2. ノードでPacemakerクラスタサービスを開始します。

```
pcs cluster start <HOSTNAME>
```

3. ノードのスタンバイを解除します。

```
pcs node unstandby <HOSTNAME>
```

4. すべてのBeeGFSサービスを優先ノードに再配置します。

```
pcs resource relocate run
```

すべてのアダプタが更新されるまで、クラスタ内のファイルノードごとに上記の手順を繰り返します。

2ノードクラスタ更新アプローチ

この方法は、ノードが2つだけのHAクラスタの場合に推奨されます。このアプローチはローリング更新に似ていますが、1つのノードのクラスタサービスが停止しているときにサービスのダウンタイムを回避するための手順が追加されています。

1. 各BeeGFSサービスが優先ノードで実行され、クラスタが最適な状態であることを確認します。詳細については、を参照してください "クラスタの状態を確認します" 。
2. 更新するファイルノードを選択し、ノードをスタンバイモードにします。スタンバイモードでは、そのノードからすべてのBeeGFSサービスが削除（または移動）されます。

```
pcs node standby <HOSTNAME>
```

3. 次のコマンドを実行して、ノードのリソースが枯渇したことを確認します。

```
pcs status
```

スタンバイ状態のノードでサービスがとして報告されていないことを確認します `Started`。

クラスタのサイズによっては、BeeGFSサービスが姉妹ノードとして報告されるまでに数秒から数分かかることがあります `Started`。BeeGFSサービスが起動しない場合は、を参照してください"トラブルシューティングガイド"。

4. クラスタをメンテナンスモードにします。

```
pcs property set maintenance-mode=true
```

5. を使用してアダプタファームウェアを更新し `mlxfwmanager`ます。

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

ファームウェアのアップデートを受信している各アダプタのをメモします `PCI Device Name`。

6. ユーティリティを使用して各アダプタをリセットし、 `mlxfwreset`新しいファームウェアを適用します。

> 一部のファームウェアアップデートでは、アップデートを適用するために再起動が必要になる場合があります。詳細については、を参照してください"NVIDIAのmlxfwresetの制限事項"。リブートが必要な場合は、アダプタをリセットする代わりにリブートを実行します。

a. opensmサービスを停止します。

```
systemctl stop opensm
```

b. 前述の各コマンドについて、次のコマンドを実行し `PCI Device Name`ます。

```
mlxfwreset -d <pci_device_name> reset -y
```

c. opensmサービスを開始します。

```
systemctl start opensm
```

7. を実行し `ibstat`、すべてのアダプタが目的のファームウェアバージョンで実行されていることを確認します。

```
ibstat
```

8. ノードでPacemakerクラスタサービスを開始します。

```
pcs cluster start <HOSTNAME>
```

9. ノードのスタンバイを解除します。

```
pcs node unstandby <HOSTNAME>
```

10. クラスタのメンテナンスモードを終了します。

```
pcs property set maintenance-mode=false
```

11. すべてのBeeGFSサービスを優先ノードに再配置します。

```
pcs resource relocate run
```

すべてのアダプタが更新されるまで、クラスタ内のファイルノードごとに上記の手順を繰り返します。

E-Seriesストレージアレイのアップグレード

HAクラスタのEシリーズストレージアレイのコンポーネントをアップグレードする手順は、次のとおりです。

概要

HAクラスタのNetApp Eシリーズストレージアレイを最新のファームウェアで最新の状態に保つことで、最適なパフォーマンスとセキュリティが確保されます。ストレージアレイのファームウェア更新は、SANtricity OS、NVSRAM、およびドライブファームウェアファイルを使用して適用されます。

ストレージアレイはHAクラスタをオンラインにしたままアップグレードできますが、すべてのアップグレードでクラスタをメンテナンスモードにすることを推奨します。

ブロックノードのアップグレード手順

次の手順は `Netapp_Eseries.Santricity`、Ansibleコレクションを使用してストレージアレイのファームウェアを更新する方法の概要です。次の手順に進む前に、を参照して"アップグレード時の考慮事項"E-Seriesシステムを更新してください。

SANtricity OS 11.80以降のリリースへのアップグレードは、11.70.5P1以降でのみ可能です。以降のアップグレードを適用する前に、ストレージアレイを11.70.5P1にアップグレードしておく必要があります。

1. Ansibleコントロールノードが最新のSANtricity Ansibleコレクションを使用していることを確認します。
   - へのアクセスによる収集のアップグレード "Ansible Galaxy"を使用して、次のコマンドを実行します。

     ```
     ansible-galaxy collection install netapp_eseries.santricity --upgrade
     ```

   - オフラインアップグレードの場合は、からcollection tarballをダウンロードし"Ansible Galaxy"、制御ノードに転送して、次のコマンドを実行します。

     ```
     ansible-galaxy collection install netapp_eseries-santricity-
     <VERSION>.tar.gz --upgrade
     ```

     を参照してください "コレクションのインストール" を参照してください。
2. ストレージアレイとドライブの最新のファームウェアを入手します。

a. ファームウェアファイルをダウンロードします。

   - * SANtricity OSとNVSRAM：*に移動し、"NetApp Support Site"お使いのストレージアレイモデルに対応した最新リリースのSANtricity OSとNVSRAMをダウンロードします。
   - *ドライブファームウェア：*に移動し"E-Seriesディスクファームウェアサイト"、ストレージアレイの各ドライブモデルに対応する最新のファームウェアをダウンロードします。

b. SANtricity OS、NVSRAM、およびドライブファームウェアのファイルをAnsibleの制御ノードの `<inventory_directory>/packages`ディレクトリに格納します。

3. 必要に応じて、クラスタのAnsibleインベントリファイルを更新して、更新が必要なすべてのストレージアレイ（ブロックノード）を含めます。手順については、を参照してください"Ansibleのインベントリの概要"。
4. 各BeeGFSサービスが優先ノードに配置され、クラスタが最適な状態であることを確認します。詳細については、を参照してください "クラスタの状態を確認します" 。
5. の手順に従って、クラスタをメンテナンスモードにし"クラスタをメンテナンスモードにします"ます。
6. という名前の新しいAnsibleプレイブックを作成し `update_block_node_playbook.yml`ます。Playbookに次の情報を入力し、SANtricity OS、NVSRAM、およびドライブファームウェアのバージョンを目的のアップグレードパスに置き換えます。

```
- hosts: eseries_storage_systems
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.santricity
  vars:
    eseries_firmware_firmware: "packages/<SantricityOS>.dlp"
    eseries_firmware_nvsram: "packages/<NVSRAM>.dlp"
    eseries_drive_firmware_firmware_list:
      - "packages/<drive_firmware>.dlp"
    eseries_drive_firmware_upgrade_drives_online: true

  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

7. 更新を開始するには、Ansibleコントロールノードから次のコマンドを実行します。

```
ansible-playbook -i inventory.yml update_block_node_playbook.yml
```

8. プレイブックが完了したら、各ストレージアレイが最適な状態になっていることを確認します。
9. クラスタをメンテナンスモードから切り替え、各BeeGFSサービスが優先ノードに配置され、クラスタが最適な状態であることを確認します。

サービスとメンテナンス

フェイルオーバーサービスとフェイルバックサービス

クラスタノード間でBeeGFSサービスを移動します。

概要

BeeGFSサービスは、ノード間でフェイルオーバーして、ノードに障害が発生した場合でもクライアントが引き続きファイルシステムにアクセスできるようにすることも、計画的なメンテナンスを実行することもできます。このセクションでは、障害からのリカバリ後、またはノード間でサービスを手動で移動したあとに、クラスタを修復するさまざまな方法について説明します。

手順

フェイルオーバーとフェイルバック

フェイルオーバー（計画的）

通常、保守のために1つのファイルノードをオフラインにする必要がある場合は、そのノードからすべてのBeeGFSサービスを移動（または削除）する必要があります。そのためには、最初にノードをスタンバイにします。

```
pcs node standby <HOSTNAME>
```

使用状況を確認したら `pcs status` すべてのリソースが代替ファイルノードで再起動されました。必要に応じて、ノードをシャットダウンしたり他の変更を加えたりできます。

フェイルバック（計画的フェイルオーバーのあと）

BeeGFSサービスを優先ノードにリストアする準備ができたら、最初に実行します `pcs status` ノードリストで、ステータスがstandbyになっていることを確認します。ノードをリブートした場合、クラスタサービスをオンラインにするまではオフラインと表示されます。

```
pcs cluster start <HOSTNAME>
```

ノードがオンラインになったら、次のコマンドを使用して、ノードをスタンバイ状態から解除します。

```
pcs node unstandby <HOSTNAME>
```

最後に、次のコマンドを使用してすべてのBeeGFSサービスを優先ノードに再配置します。

```
pcs resource relocate run
```

フェイルバック（計画外フェイルオーバー後）

ノードでハードウェアやその他の障害が発生した場合、HAクラスタが正常なノードに自動的に対応してサービスを移動し、管理者が適切に対処できるようにする必要があります。作業を進める前に、"トラブルシューティング"セクションを参照してフェイルオーバーの原因を特定し、未解決の問題を解決してください。ノードの電源がオンになり正常に戻ったら、フェイルバックを続行できます。

計画外（または計画的）リブート後にノードがブートした場合、クラスタサービスは自動的には開始されないため、最初にでノードをオンラインにする必要があります。

```
pcs cluster start <HOSTNAME>
```

次に、リソース障害をクリーンアップし、ノードのフェンシング履歴をリセットします。

```
pcs resource cleanup node=<HOSTNAME>
pcs stonith history cleanup <HOSTNAME>
```

で確認します `pcs status` ノードはオンラインで正常な状態です。デフォルトでは、BeeGFSサービスは、リソースを誤って正常なノードに戻すことを防ぐために、自動的にフェイルバックを行いません。準備ができたら、を指定してクラスタ内のすべてのリソースを優先ノードに戻します。

```
pcs resource relocate run
```

個 々 の**BeeGFS**サービスを代替ファイルノードに移動します

BeeGFSサービスを新しいファイルノードに永続的に移動します

個 々 のBeeGFSサービスの優先ファイルノードを永続的に変更するには、Ansibleのインベントリを調整して優先ノードがリストされるようにしてから、Ansibleプレイブックを再実行します。

たとえば、次のサンプルファイルでは `inventory.yml` 、beegfs_01がBeeGFS管理サービスの実行に優先されるファイルノードです。

```
        mgmt:
          hosts:
            beegfs_01:
            beegfs_02:
```

この順序を逆にすると、beegfs_02で管理サービスが優先されます原因 。

```
        mgmt:
          hosts:
            beegfs_02:
            beegfs_01:
```

BeeGFSサービスを一時的に代替ファイルノードに移動します

通常、ノードのメンテナンス中に[フェイルオーバーとフェイルバックの手順]（フェイルオーバーとフェイルバックの手順）を使用して、すべてのサービスをそのノードから移動します。

何らかの理由で、個 々 のサービスを別のファイルノードに移動する必要がある場合は、次のコマンドを実行します。

```
pcs resource move <SERVICE>-monitor <HOSTNAME>
```

> 個 々 のリソースまたはリソースグループを指定しないでください。再配置するBeeGFSサービスのモニタ名を必ず指定しますたとえば、BeeGFS管理サービスをbeegfs_02に移動するには、次のコマンドを実行し `pcs resource move mgmt-monitor beegfs_02` ます。このプロセスを繰り返して、1つ以上のサービスを優先ノードから移動できます。サービスが新しいノードで再配置/開始されたことを確認します `pcs status 。

BeeGFSサービスを優先ノードに戻すには'最初に一時的なリソースの制約を解除します（複数のサービスの場合はこの手順を繰り返します）

```
pcs resource clear <SERVICE>-monitor
```

次に、サービスを実際に優先ノードに戻す準備ができたら、次のコマンドを実行します。

```
pcs resource relocate run
```

このコマンドは、優先ノードに配置されていない一時的なリソース制約のないサービスを再配置します。

クラスタをメンテナンスモードにします

環境内の意図した変更にHAクラスタが誤って対応しないようにします。

概要

クラスタをメンテナンスモードにすると、リソースの監視がすべて無効になり、ペースメーカーによるクラスタ内のリソースの移動や管理ができなくなります。アクセスを妨げる一時的な障害が発生した場合でも、すべてのリソースは元のノードで実行されたままとなります。次のような場合に推奨/有用です。

- ファイルノードとBeeGFSサービス間の接続を一時的に中断する可能性のあるネットワークメンテナンス。

- ブロックノードのアップグレード。
- ファイルノードのオペレーティングシステム、カーネル、またはその他のパッケージの更新。

通常、クラスタを手動で保守モードにする唯一の理由は、環境内の外部の変更にクラスタが対応できないようにするためです。クラスタ内の個 々 のノードで物理的な修復が必要な場合は、保守モードを使用せずに、そのノードを上記の手順 に従ってスタンバイにします。Ansibleを再実行するとクラスタが自動的に保守モードになり、アップグレードや設定の変更など、ほとんどのソフトウェアメンテナンスが容易になります。

手順

クラスタがメンテナンスモードかどうかを確認するには、次のコマンドを実行します。

```
pcs property config
```

`maintenance-mode`クラスタが正常に動作している場合、プロパティは表示されません。クラスタが現在メンテナンスモードの場合、プロパティはとして報告されます `true`。メンテナンスモードの実行を有効にするには、次の

```
pcs property set maintenance-mode=true
```

PCステータスを実行し、すべてのリソースに「（管理対象外）」と表示されていることを確認することで確認できます。クラスタの保守モードを解除するには、次のコマンドを実行します。

```
pcs property set maintenance-mode=false
```

クラスタを停止して起動します

HAクラスタを正常に停止および起動します。

概要

ここでは、BeeGFSクラスタを正常にシャットダウンして再起動する方法について説明します。これが必要なシナリオの例としては、電気的なメンテナンスやデータセンター間またはラック間の移行などが挙げられます。

手順

何らかの理由でBeeGFSクラスタ全体を停止し、すべてのサービスをシャットダウンする必要がある場合は、次の手順を実行します。

```
pcs cluster stop --all
```

個々のノードでクラスタを停止することもできます（これにより、サービスが別のノードに自動的にフェイルオーバーされます）"フェイルオーバー"。ただし、最初にノードをスタンバイ状態にすることを推奨します（を参照）。

```
pcs cluster stop <HOSTNAME>
```

すべてのノードでクラスタサービスとリソースを開始するには、次のコマンドを実行します。

```
pcs cluster start --all
```

または、次のコマンドを使用して特定のノードでサービスを開始します。

```
pcs cluster start <HOSTNAME>
```

この時点でを実行します `pcs status` すべてのノードでクラスタサービスとBeeGFSサービスが開始され、必要なノードでサービスが実行されていることを確認します。

クラスタのサイズによっては、クラスタ全体が停止するまでに数秒から数分かかる場合やで開始されたと表示される場合があり `pcs status`ます。が5分以上ハングする場合 `pcs cluster <COMMAND>`は、「Ctrl+C」を実行してコマンドをキャンセルする前に、クラスタの各ノードにログインし、を使用して `pcs status`そのノードでクラスタサービス（Corosync / Pacemaker）が実行されているかどうかを確認します。クラスタがまだアクティブになっているノードから、クラスタをブロックしているリソースを確認できます。問題 に手動で対処する必要があります。コマンドを完了するか、再実行して残りのサービスを停止できます。

ファイルノードを交換します

元のサーバに障害がある場合のファイルノードの交換

概要

クラスタ内のファイルノードを交換するために必要な手順の概要を以下に示します。これらの手順は、ハードウェア問題 が原因でファイルノードに障害が発生し、新しい同一のファイルノードに置き換えられたことを前提としています。

手順

1. ファイルノードを物理的に交換し、すべてのケーブルをブロックノードおよびストレージネットワークに接続します。
2. Red Hatサブスクリプションの追加を含め、ファイルノードにオペレーティングシステムを再インストールします。
3. ファイルノードで管理ネットワークとBMCネットワークを設定します。
4. ホスト名、IP、PCIeと論理インターフェイスのマッピング、または新しいファイルノードに関する変更があれば、Ansibleインベントリを更新します。通常この作業は、ノードを同じサーバハードウェアに交換し、元のネットワーク構成を使用している場合は必要ありません。

a. たとえば、ホスト名が変更された場合は、ノードのインベントリファイルを作成（または名前を変更）します (`host_vars/<NEW_NODE>.yml`）をクリックしてから、Ansibleインベントリファイルを使用してください (`inventory.yml`）で、古いノードの名前を新しいノード名に置き換えます。

```
all:
    ...
    children:
    ha_cluster:
        children:
        mgmt:
            hosts:
            node_h1_new:    # Replaced "node_h1" with "node_h1_new"
            node_h2:
```

5. クラスタ内の他のいずれかのノードから古いノードを削除します。 `pcs cluster node remove <HOSTNAME>`。

   この手順を実行する前に、次に進まないでください。

6. Ansibleコントロールノードで：
   a. 次のコマンドを使用して古いSSHキーを削除します。

```
`ssh-keygen -R <HOSTNAME_OR_IP>`
```

   b. 交換用ノードにパスワードなしのSSHを設定します。

```
ssh-copy-id <USER>@<HOSTNAME_OR_IP>
```

7. Ansibleプレイブックを再実行してノードを設定し、クラスタに追加します。

```
ansible-playbook -i <inventory>.yml <playbook>.yml
```

8. この時点で、を実行します `pcs status` 交換したノードが表示され、サービスが実行されていることを確認します。

クラスタを拡張または縮小します

クラスタのビルディングブロックを追加または削除します。

概要

ここでは、BeeGFS HAクラスタのサイズを調整するためのさまざまな考慮事項とオプションについて説明します。通常、クラスタのサイズはビルディングブロックを追加または削除することで調整されます。ビルディ

ングブロックは通常、HAペアとして2つのファイルノードがセットアップされる構成です。必要に応じて、個々 のファイルノード（または他のタイプのクラスタノード）を追加または削除することもできます。

クラスタへのビルディングブロックの追加

考慮事項

ビルディングブロックを追加してクラスタを拡張するプロセスは簡単です。個 々 のHAクラスタ内のクラスタノードの最小数と最大数に関する制限に注意して、既存のHAクラスタにノードを追加するか、新しいHAクラスタを作成するかを決定する必要があります。通常、各ビルディングブロックは2つのファイルノードで構成されますが、クラスタあたりの最小ノード数は3（クォーラムを確立するため）、推奨される最大ノード数は10（テスト済み）です。高度なシナリオでは、2ノードクラスタの導入時にBeeGFSサービスを実行しない「Tiebreaker」ノードを1つ追加できます。このような導入を検討される場合は、ネットアップサポートにお問い合わせください。

クラスタの拡張方法を決定する際には、これらの制限事項と、想定される将来のクラスタの拡張について留意してください。たとえば、6ノードクラスタの場合、ノードを4つ追加する必要があるときは、新しいHAクラスタを開始するだけで十分です。

1つのBeeGFSファイルシステムを複数の独立したHAクラスタで構成できますこれにより、基盤となるHAクラスタコンポーネントの推奨される制限やハード制限をはるかに超えてファイルシステムを継続的に拡張できます。

手順

クラスタにビルディングブロックを追加する場合は `host_vars`、新しいファイルノードおよびブロックノード（E-Seriesアレイ）ごとにファイルを作成する必要があります。これらのホストの名前は、作成する新しいリソースとともにインベントリに追加する必要があります。 `group_vars`新しいリソースごとに対応するファイルを作成する必要があります。"カスタムアーキテクチャの使用"詳細については、を参照してください。

正しいファイルを作成したら、コマンドを使用して自動化を再実行するだけです。

```
ansible-playbook -i <inventory>.yml <playbook>.yml
```

クラスタからのビルディングブロックの削除

ビルディングブロックをリタイアさせる必要がある場合は、いくつかの考慮事項に留意する必要があります。次に例を示します。

- このビルディングブロックで実行されているBeeGFSサービスは何ですか。
- ファイルノードが撤去され、ブロックノードが新しいファイルノードに接続されるだけですか。
- ビルディングブロック全体が廃止される場合、データを新しいビルディングブロックに移動したり、クラスタ内の既存のノードに分散させたり、新しいBeeGFSファイルシステムやその他のストレージシステムに移動したりする必要がありますか。
- これはシステム停止中に発生するか、またはシステムを停止せずに実行する必要がありますか？
- ビルディングブロックがアクティブに使用されているか、またはアクティブではなくなったデータが主に含まれているか。

導入にあたっては、さまざまな出発点や希望する最終状態が考えられるため、ネットアップのサポートにご連

絡ください。ネットアップでは、お客様の環境と要件に基づいて最適な戦略を特定し、導入するお手伝いをいたします。

トラブルシューティングを行う

BeeGFS HAクラスタのトラブルシューティング

概要

このセクションでは、BeeGFS HAクラスタの運用中に発生する可能性のあるさまざまな障害やその他のシナリオを調査してトラブルシューティングする方法を説明します。

トラブルシューティングガイド

予期しないフェイルオーバーを調査してい

ノードが予期せずフェンシングされていてサービスが別のノードに移動された場合は、の下部でクラスタがリソース障害を示していないかどうかを最初に確認する必要があります `pcs status`。通常、フェンシングが正常に完了し、リソースが別のノードで再起動された場合、何も表示されません。

通常、次の手順では、を使用してシステムログを検索します `journalctl` 残りのファイルノードのいずれか（Pacemakerログは、すべてのノードで同期されます）。障害が発生した時刻がわかっている場合は、障害が発生する直前に検索を開始できます（通常は10分前までに実行することをお勧めします）。

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>"
```

以降のセクションでは、ログに含まれているをgrepで検索できる一般的なテキストを示します。これにより、調査をさらに絞り込むことができます。

調査/解決の手順

手順**1**：**BeeGFS**モニタで障害が検出されたかどうかを確認します。

BeeGFSモニタによってフェイルオーバーがトリガーされた場合は、エラーが表示されます（次の手順に進みない場合）。

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i unexpected
[...]
Jul 01 15:51:03 beegfs_01 pacemaker-schedulerd[9246]:  warning: Unexpected
result (error: BeeGFS service is not active!) was recorded for monitor of
meta_08-monitor on beegfs_02 at Jul  1 15:51:03 2022
```

このインスタンスでは、何らかの理由でBeeGFSサービスMETA_08が停止しました。トラブルシューティングを続行するには、beegfs_02を起動し、でサービスのログを確認する必要があります `/var/log/beegfs-meta-meta_08_tgt_0801.log`。たとえば、BeeGFSサービスで内部問題 やノードの問題が原因でアプリケーションエラーが発生した可能性があります。

Pacemakerのログとは異なり、BeeGFSサービスのログはクラスタ内のすべてのノードに分散されるわけではありません。これらのタイプの障害を調査するには、障害が発生した元のノードのログが必要です。

モニタで報告される可能性がある問題は次のとおりです。

- ターゲットにアクセスできません。
  - 概要 ：ブロックボリュームにアクセスできなかったことを示します。
  - トラブルシューティング：
    - 代替ファイルノードでサービスも開始できない場合は、ブロックノードが正常な状態であることを確認してください。
    - このファイルノードからブロックノードにアクセスできなくなる可能性がある物理的な問題がないかどうかを確認します。たとえば、InfiniBandアダプタまたはケーブルに障害がある場合などです。
- ネットワークに到達できません。
  - 概要 ：このBeeGFSサービスへの接続にクライアントが使用するアダプタがオンラインではありませんでした。
  - トラブルシューティング：
    - 複数またはすべてのファイルノードが影響を受けた場合は、BeeGFSクライアントとファイルシステムの接続に使用されるネットワークに障害が発生していないかどうかを確認します。
    - InfiniBandアダプタやケーブルの障害など、このファイルノードからクライアントにアクセスできなくなる物理的な問題がないかどうかを確認してください。
- BeeGFSサービスはアクティブではありません。
  - 概要 ：BeeGFSサービスが予期せず停止しました。
  - トラブルシューティング：
    - エラーが報告されたファイルノードで、影響を受けたBeeGFSサービスのログを調べて、クラッシュが報告されたかどうかを確認します。この場合は、ネットアップサポートに問い合わせてクラッシュを調査できるようにケースをオープンしてください。
    - BeeGFSログにエラーが報告されていない場合は、ジャーナルログを調べて、サービスが停止した理由がシステムに記録されているかどうかを確認します。一部のシナリオでは、プロセスが終了する前（たとえば、誰かが実行された場合）にBeeGFSサービスがメッセージをログに記録できなかった可能性があります `kill -9 <PID>`）。

手順**2**：ノードがクラスタから予期せず離れていないかを確認します

ノードで壊滅的なハードウェア障害が発生した場合（システム基板が故障した場合など）、またはカーネルパニックまたは同様のソフトウェア問題 が発生した場合、BeeGFSモニタはエラーを報告しません。代わりにホスト名を検索し、Pacemakerからのメッセージで、ノードが予期せずに失われたことを示すメッセージが表示されます。

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i <HOSTNAME>
[...]
Jul 01 16:18:01 beegfs_01 pacemaker-attrd[9245]:  notice: Node beegfs_02
state is now lost
Jul 01 16:18:01 beegfs_01 pacemaker-controld[9247]:  warning:
Stonith/shutdown of node beegfs_02 was not expected
```

手順3：**Pacemaker**がノードを遮断できたことを確認します

すべてのシナリオで、ノードが実際にオフラインであることを確認するためにペースメーカーがノードを遮断しようとすると考えられます（正確なメッセージはフェンシングの原因 によって異なる場合があります）。

```
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Cluster
node beegfs_02 will be fenced: peer is no longer part of the cluster
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Node
beegfs_02 is unclean
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Scheduling
Node beegfs_02 for STONITH
```

フェンシング処理が正常に完了すると、次のようなメッセージが表示されます。

```
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
[2214070] (call 27 from pacemaker-controld.9247) for host 'beegfs_02' with
device 'fence_redfish_2' returned: 0 (OK)
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
targeting beegfs_02 on beegfs_01 for pacemaker-
controld.9247@beegfs_01.786df3a1: OK
Jul 01 16:18:14 beegfs_01 pacemaker-controld[9247]:  notice: Peer
beegfs_02 was terminated (off) by beegfs_01 on behalf of pacemaker-
controld.9247: OK
```

何らかの理由でフェンシングアクションが失敗した場合、データ破損のリスクを回避するために別のノードでBeeGFSサービスを再起動できなくなります。たとえば、フェンシングデバイス（PDUまたはBMC）にアクセスできなかったり、誤って設定されていた場合、問題 は個別に調査します。

Address Failed Resource Actions（**PC**ステータスの下部にある）

BeeGFSサービスの実行に必要なリソースに障害が発生すると、BeeGFSモニタによってフェイルオーバーがトリガーされます。この場合は、の下部に[Failed Resource Actions]が表示されない可能性があり `pcs status` ます。[How to]の手順を参照してください。"計画外フェイルオーバー後のフェイルバック"

そうしないと、通常、「Failed Resource Actions」というメッセージが表示されるシナリオは2つだけになります。

シナリオ**1**：フェンシングエージェントで一時的または永続的な問題 が検出され、再起動されたか、別のノードに移動された。

フェンシングエージェントの中には、他のエージェントよりも信頼性の高いものがあり、それぞれがフェンシングデバイスの準備が整ったことを確認する独自の監視方法を実装しています。特に、Redfishフェンシングエージェントは、次のような失敗したリソースアクションを報告するようになりましたが、まだ開始されています。

```
  * fence_redfish_2_monitor_60000 on beegfs_01 'not running' (7):
call=2248, status='complete', exitreason='', last-rc-change='2022-07-26
08:12:59 -05:00', queued=0ms, exec=0ms
```

特定のノードで失敗したリソースアクションを報告するフェンシングエージェントは、そのノードで実行されているBeeGFSサービスのフェールオーバーをトリガーする必要はありません。同じノードまたは別のノードで自動的に再起動されるだけです。

解決手順：

1. フェンシングエージェントが、すべてのノードまたは一部のノードでの実行を常に拒否している場合は、それらのノードがフェンシングエージェントに接続できるかどうかを確認し、フェンシングエージェントがAnsibleインベントリで正しく設定されていることを確認します。
   a. たとえば、Redfish（BMC）フェンシングエージェントがフェンシングを担当するノードで実行されており、OS管理とBMC IPが同じ物理インターフェイス上にある場合、一部のネットワークスイッチ構成では2つのインターフェイス間の通信が許可されないため（ネットワークループが回避されます）、デフォルトでは、HAクラスタは、フェンシングを担当するノードにフェンシングエージェントを配置しないようにしますが、これは一部のシナリオや構成で発生する可能性があります。
2. すべての問題が解決したら（または問題 が一時的なものと思われる場合）、を実行します `pcs resource cleanup` 失敗したリソースアクションをリセットします。

シナリオ**2**：**BeeGFS**モニタが問題 を検出してフェイルオーバーをトリガーしましたが、何らかの理由でセカンダリノードでリソースを起動できませんでした。

フェンシングが有効で、リソースが元のノードで停止しないようにブロックされていない場合（「standby」（on -ffail）のトラブルシューティングのセクションを参照）、最も可能性の高い理由は、次のような理由からセカンダリノードでリソースを起動する際の問題です。

- セカンダリノードはすでにオフラインでした。
- 物理構成または論理構成の問題 によって、セカンダリはBeeGFSターゲットとして使用されるブロックボリュームにアクセスできなくなりました。

解決手順：

1. 失敗したリソースアクションの各エントリについて、次の手順を実行します。
   a. 失敗したリソースアクションが開始操作であることを確認します。
   b. 指定したリソースと、失敗したリソースアクションで指定されたノードに基づきます。
      i. ノードが指定したリソースを起動できないような外部の問題がないかどうかを確認して解決しま

す。たとえば、BeeGFS IP address (floating IP) failed to startの場合は、必要なインターフェイスの少なくとも1つが接続/オンラインであり、適切なネットワークスイッチにケーブル接続されていることを確認します。BeeGFSターゲット（ブロックデバイス/Eシリーズボリューム）に障害が発生した場合は、バックエンドブロックノードへの物理接続が正常に接続されていることを確認し、ブロックノードが正常であることを確認します。

c. 明らかな外部の問題がなく、このインシデントに対するrootの原因 が必要な場合は、以下の手順を進める前にネットアップサポートの調査ケースをオープンして原因 分析（RCA）を実施することを検討することを推奨します。

2. 外部の問題を解決したあと：

a. Ansibleのinventory.ymlファイルから機能しないノードをコメント化し、完全なAnsibleプレイブックを再実行して、すべての論理構成がセカンダリノードで正しくセットアップされていることを確認します。

i. 注：ノードが正常でフェイルバックの準備ができたら、これらのノードのコメントを解除してプレイブックを再実行してください。

b. または、クラスタのリカバリを手動で実行することもできます。

i. 次のコマンドを使用して、オフラインのノードをオンラインに戻します。 `pcs cluster start <HOSTNAME>`

ii. 障害が発生したすべてのリソースアクションをクリアするには、 `pcs resource cleanup`

iii. PCステータスを実行し、すべてのサービスが期待どおりに開始されることを確認します。

iv. 必要に応じてを実行します `pcs resource relocate run` をクリックして、リソースを優先ノードに戻します（使用可能な場合）。

一般的な問題

BeeGFSサービスは、要求されたときにフェイルオーバーやフェイルバックを行いません

可能性の高い問題 ： `pcs resource relocate` 実行コマンドは実行されましたが、正常に終了しませんでした。

*確認方法:*実行 `pcs constraint --full` IDがの場所の制約がないかどうかを確認します `pcs-relocate-<RESOURCE>`。

*解決方法:*実行 `pcs resource relocate clear` 再実行します `pcs constraint --full` 追加の拘束が除去されたことを確認します。

フェンシングが無効な場合、**PC**ステータスの一方のノードに「**standby (on-fail)**」と表示されます

考えられる問題 ： Pacemakerは、障害が発生したノードですべてのリソースが停止していることを正常に確認できませんでした。

解決方法:

1. を実行します `pcs status` および出力の一番下に「started」または「エラーが表示されていないリソースがないかどうかを確認し、問題を解決します。

2. ノードをオンラインに戻すには、次の手順を実行します `pcs resource cleanup --node=<HOSTNAME>`。

想定外のフェイルオーバーが発生すると、フェンシングが有効になっている場合、**PC**のステータスに「**started**（**on-fail**）」と表示されます

*問題 の可能性：*フェールオーバーをトリガーしたが、Pacemakerがノードをフェンシングしていることを確認できなかった問題が発生しました。フェンシングが正しく設定されていないか、フェンシングエージェントを含む問題 が存在することが原因で発生します（例：PDUがネットワークから切断されています）。

解決方法:

1. ノードの電源がオフになっていることを確認します。

> 指定したノードが実際にはオフになっておらず、クラスタのサービスやリソースを実行している場合は、データの破損やクラスタ障害が発生します。

2. フェンシングを手動で確認する場合： `pcs stonith confirm <NODE>`

この時点で、サービスのフェイルオーバーが完了し、別の正常なノードで再開されます。

一般的なトラブルシューティングタスク

BeeGFSサービスを個別に再起動します

通常、BeeGFSサービスを再起動（設定変更を容易にするためなど）する必要がある場合は、Ansibleインベントリを更新してプレイブックを再実行します。一部のシナリオでは、個 々 のサービスを再起動して迅速なトラブルシューティングを実現したい場合があります。たとえば、プレイブック全体の実行を待たずにログレベルを変更する場合などです。

> Ansibleインベントリに手動で変更を追加しないかぎり、次回Ansibleプレイブックが実行されたときに変更が元に戻されます。

オプション**1**：**systemd**で制御された再起動

新しい設定でBeeGFSサービスが適切に再起動しないリスクがある場合は、まずクラスタをメンテナンスモードにして、BeeGFSモニタがサービスを停止して不要なフェイルオーバーをトリガーしないようにします。

```
pcs property set maintenance-mode=true
```

必要に応じて、でサービス設定を変更します `/mnt/<SERVICE_ID>/_config/beegfs-.conf` （例： `/mnt/meta_01_tgt_0101/metadata_config/beegfs-meta.conf`）次にsystemdを使用して再起動します。

```
systemctl restart beegfs-*@<SERVICE_ID>.service
```

例 `systemctl restart beegfs-meta@meta_01_tgt_0101.service`

オプション**2**：ペースメーカーの再起動を制御

新しい設定で原因 サービスが予期せず停止する（ロギングレベルの変更など）か、メンテナンス時間になっ

ていてダウンタイムが気にならない場合は、再起動するサービスのBeeGFSモニタを再起動するだけです。

```
pcs resource restart <SERVICE>-monitor
```

たとえば、BeeGFS管理サービスを再起動するには、次の手順を実行します。 `pcs resource restart mgmt-monitor`

著作権に関する情報

商標に関する情報