カスタムアーキテクチャを使用
BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

目次

カスタムアーキテクチャを使用

概要と要件

Ansibleを使用してBeeGFSハイアベイラビリティクラスタを導入する場合は、任意のNetApp E / EFシリーズストレージシステムをBeeGFSブロックノードとして使用し、x86サーバをBeeGFSファイルノードとして使用します。

このセクションで使用される用語の定義については、"用語と概念"ページを参照してください。

はじめに

"NetApp Verified Architectureレポート"事前定義されたリファレンス構成とサイジングガイダンスが提供されますが、一部のお客様やパートナー様は、特定の要件やハードウェアの好みにより適したカスタムアーキテクチャを設計したい場合があります。ネットアップでBeeGFSを選択する主なメリットの1つは、Ansibleを使用してBeeGFS共有ディスクHAクラスタを導入できることです。これにより、クラスタ管理が簡易化され、ネットアップがオーサリングするHAコンポーネントによって信頼性が向上します。ネットアップへのカスタムBeeGFSアーキテクチャの導入はAnsibleでも実行されるため、アプライアンスと同様のアプローチでハードウェアを柔軟に選択することができます。

このセクションでは、ネットアップハードウェアにBeeGFSファイルシステムを導入し、Ansibleを使用してBeeGFSファイルシステムを設定するための一般的な手順を説明します。BeeGFSファイルシステムの設計に関するベストプラクティスと最適化された例の詳細については、"NetApp Verified Architectureレポート"セクションを参照してください。

導入の概要

通常、BeeGFSファイルシステムを導入するには、次の手順を実行します。

- 初期セットアップ：
  - ハードウェアの設置/ケーブル接続
  - ファイルノードとブロックノードをセットアップ
  - Ansibleコントロールノードをセットアップします。
- BeeGFSファイルシステムをAnsibleインベントリとして定義します。
- ファイルノードとブロックノードに対してAnsibleを実行して、BeeGFSを導入します。
  - 必要に応じて、クライアントをセットアップし、BeeGFSをマウントします。

以降のセクションでは、これらの手順について詳しく説明します。

Ansibleは、ソフトウェアのプロビジョニングタスクと設定タスクをすべて処理します。

- ブロックノードでのボリュームの作成/マッピング
- ファイルノードでのボリュームのフォーマットと調整
- ファイルノードへのソフトウェアのインストール/設定
- HAクラスタを確立し、BeeGFSリソースとファイルシステムサービスを設定します。

要件

AnsibleでBeeGFSがサポートされるようになりました "Ansible Galaxy" BeeGFS HAクラスタのエンドツーエンドの導入と管理を自動化するロールとモジュールの集合として。

BeeGFS自体は、<major> .BeeGFSバージョンに準拠してバージョン管理されます。<patch> バージョンのバージョン管理スキームとコレクションでは、サポートされる<major> ごとに役割が維持されます。たとえば、BeeGFS 7.2やBeeGFS 7.3などのBeeGFSバージョンの<minor> <minor> バージョンです。コレクションの更新がリリースされると、各ロールのパッチバージョンが更新され、そのリリースブランチで利用可能な最新のBeeGFSバージョン（例：7.2.8）が示されます。コレクションの各バージョンは、特定の Linux ディストリビューションおよびバージョンでもテストおよびサポートされており、現在、ファイル ノードの場合は Red Hat、クライアントの場合は Red Hat と Ubuntu です。他のディストリビューションを実行することはサポートされていません。また、他のバージョン（特に他のメジャーバージョン）を実行することはお勧めしません。

Ansibleコントロールノード

このノードには、BeeGFSの管理に使用するインベントリとプレイブックが含まれます。次のものが必要です。

- Ansible 6.x（Ansibleコア2.13）
- Python 3.6（またはそれ以降）
- python（pip）パッケージ：ipaddrおよびnetaddr

また、制御ノードからすべてのBeeGFSファイルノードとクライアントにパスワードなしのSSHを設定することを推奨します。

BeeGFSファイルノード

ファイルノードはRed Hat Enterprise Linux (RHEL) 9.4を実行し、必要なパッケージ（pacemaker、corosync、fence-agents-all、resource-agents）を含むHAリポジトリにアクセスできる必要があります。例えば、RHEL 9で適切なリポジトリを有効にするには、以下のコマンドを実行します。

```
subscription-manager repo-override repo=rhel-9-for-x86_64-
highavailability-rpms --add=enabled:1
```

BeeGFSクライアントノード

BeeGFSクライアントのAnsibleロールを使用して、BeeGFSクライアントパッケージをインストールし、BeeGFSマウントを管理できます。このロールは、RHEL 9.4 および Ubuntu 22.04 でテストされていま

す。

Ansibleを使用してBeeGFSクライアントをセットアップしない場合は、BeeGFSをマウントします "BeeGFSはLinuxディストリビューションとカーネルをサポートしています" を使用できます。

初期セットアップ

ハードウェアの設置とケーブル接続

ネットアップでBeeGFSを実行するために使用するハードウェアを設置してケーブルを配線するための手順を説明します。

インストールを計画します

各BeeGFSファイルシステムは、いくつかのブロックノードで提供されるバックエンドストレージを使用して、BeeGFSサービスを実行するいくつかのファイルノードで構成されます。BeeGFSサービスにフォールトトレランスを提供するために、ファイルノードは1つ以上のハイアベイラビリティクラスタに構成されます。各ブロックノードはすでにアクティブ/アクティブHAペアです。各HAクラスタでサポートされるファイルノードの最小数は3で、各クラスタでサポートされるファイルノードの最大数は10です。BeeGFSファイルシステムは、単一のファイルシステムネームスペースを提供するために連携する複数の独立したHAクラスタを導入することで、10ノードを超える規模に拡張できます。

一般に、各HAクラスタは一連の「ビルディングブロック」として導入されます。この場合、一部の数のファイルノード（x86サーバ）がいくつかのブロックノード（通常はEシリーズストレージシステム）に直接接続されます。この設定では、非対称クラスタが作成されます。BeeGFSサービスは、BeeGFSターゲットに使用されるバックエンドブロックストレージにアクセスできる特定のファイルノードでのみ実行できます。各ビルディングブロック内のファイルとブロックのノードと、直接接続に使用されるストレージプロトコルのバランスは、特定のインストール要件によって異なります。

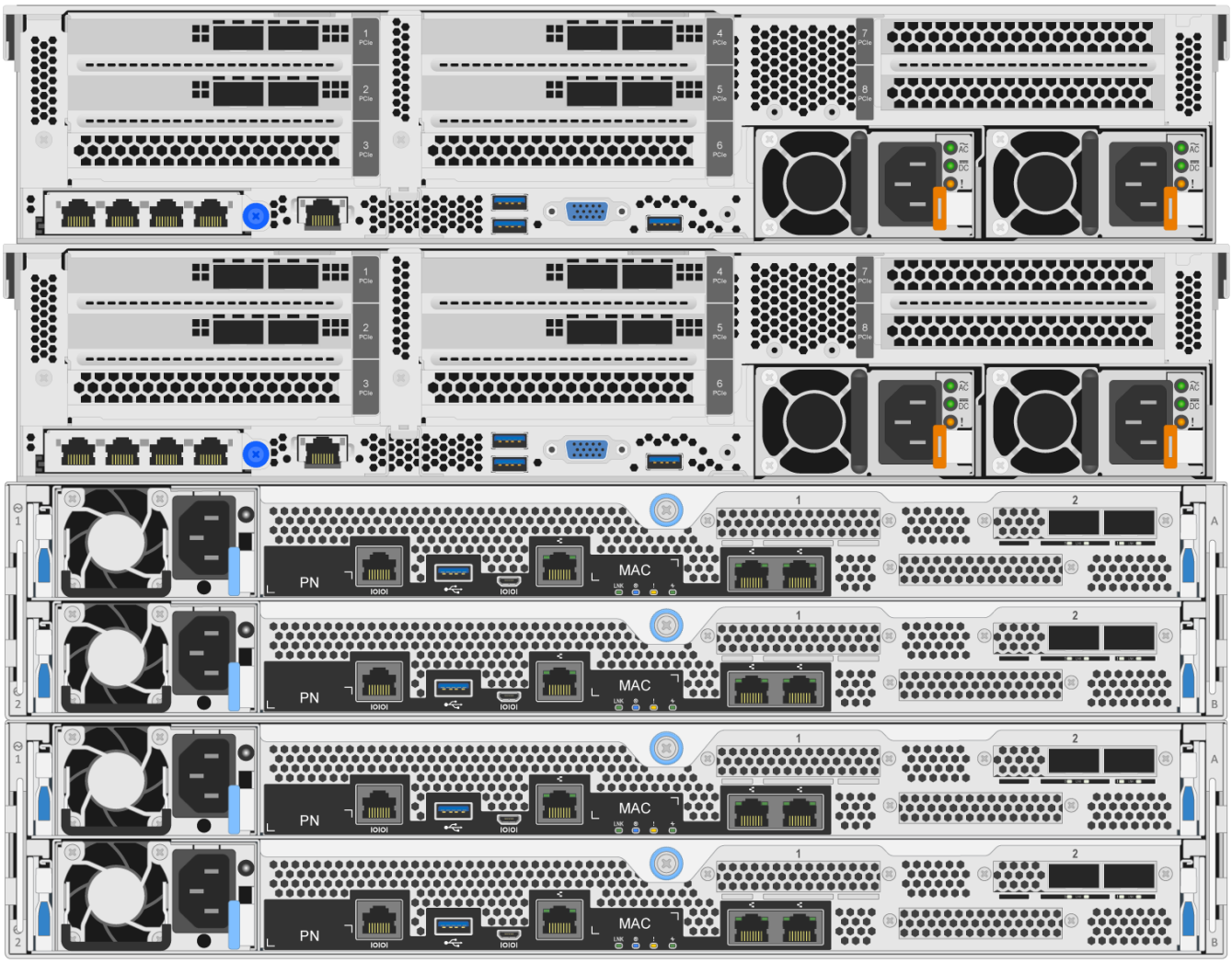
別のHAクラスタアーキテクチャでは、ファイルノードとブロックノードの間にストレージファブリック（ストレージエリアネットワークまたはSANとも呼ばれます）を使用して対称型クラスタを確立します。これにより、BeeGFSサービスを特定のHAクラスタ内の任意のファイルノードで実行できるようになります。一般に対称クラスタは、追加のSANハードウェアによってコスト効率が高くないため、このドキュメントでは、非対称クラスタを1つ以上のビルディングブロックとして配置することを前提としています。

> インストールを開始する前に、特定のBeeGFS導入に必要なファイルシステムアーキテクチャを十分に理解しておく必要があります。

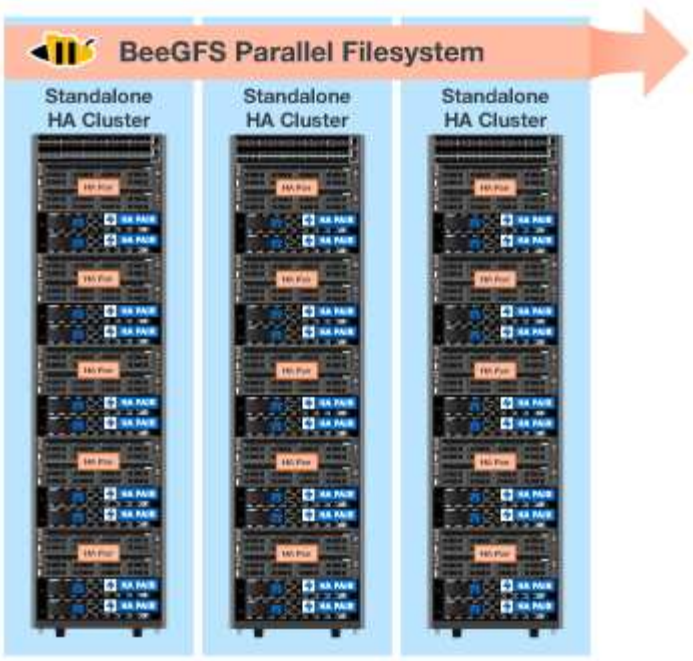
ラックハードウェア

設置を計画する場合、各ビルディングブロック内のすべての機器が隣接するラックユニットにラックに設置されることが重要です。ベストプラクティスとして、各ビルディングブロック内のブロックノードのすぐ上にファイルノードがラックに設置されるようにすることを推奨します。ファイルとのモデルについては、のマニュアルを参照してください "ブロック" ラックにレールとハードウェアを設置するときに使用するノード。

単一のビルディングブロックの例：

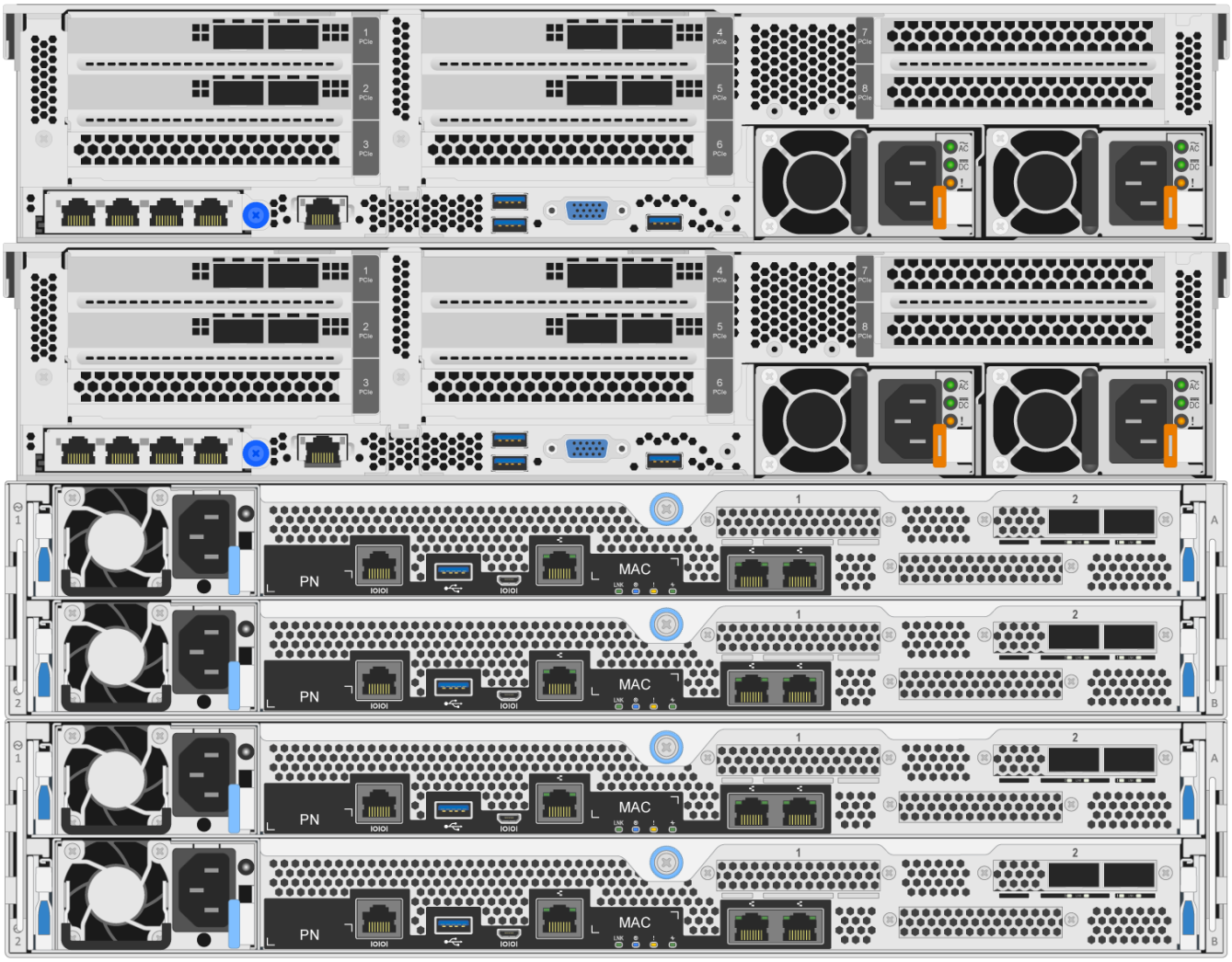


大規模なBeeGFSインストールの例では、各HAクラスタに複数のビルディングブロックがあり、ファイルシステムに複数のHAクラスタがある場合を示します。



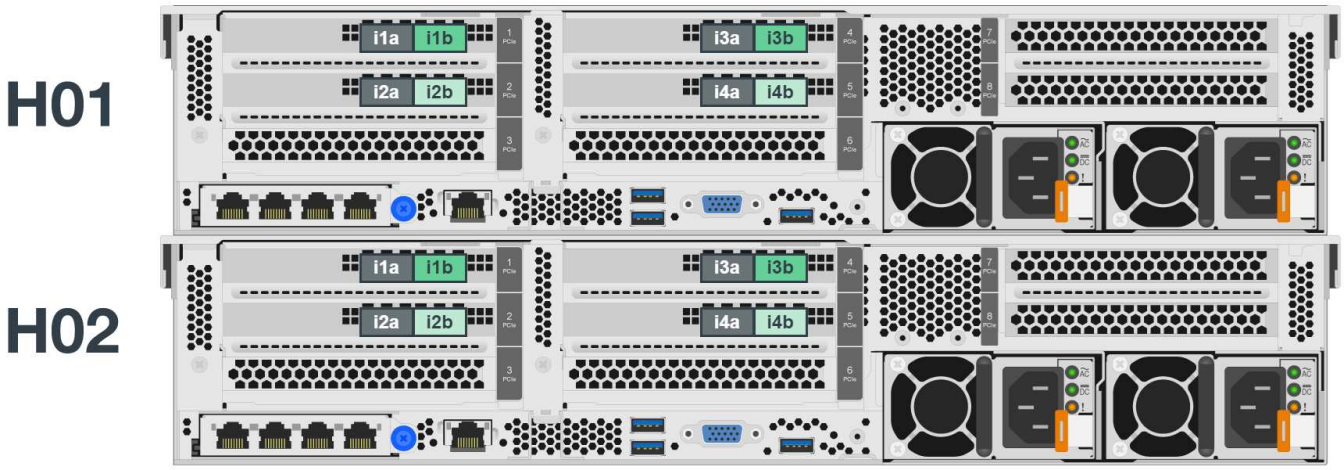
ファイルノードとブロックノードをケーブル接続します

通常、Eシリーズのブロックノードの HIC ポートは、ファイルノードの指定のホストチャネルアダプタ（InfiniBand プロトコルの場合）ポートまたはホストバスアダプタ（ファイバチャネルおよびその他のプロトコルの場合）ポートに直接接続します。これらの接続を確立する正確な方法は、目的のファイルシステムアーキテクチャによって異なります。次に例を示し"NetApp Verified Architecture上の第2世代BeeGFSに基づく"ます。



ファイルノードをクライアントネットワークにケーブル接続します

各ファイルノードには、BeeGFSクライアントトラフィックに指定されたいくつかのInfiniBandポートまたはイーサネットポートがあります。アーキテクチャによっては、各ファイルノードは高パフォーマンスのクライアント/ストレージネットワークに1つ以上の接続を持ち、潜在的に複数のスイッチに接続して冗長性を確保し、帯域幅を増やします。次に、冗長ネットワークスイッチを使用したクライアントケーブル接続の例を示します。濃い緑で強調表示されたポートは別 々 のスイッチに接続されています。



管理ネットワークと電源を接続します

インバンドおよびアウトオブバンドネットワークに必要なネットワーク接続を確立します。

すべての電源装置を接続して、各ファイルノードとブロックノードが複数の配電ユニットに接続して冗長性を確保します（使用可能な場合）。

ファイルノードとブロックノードをセットアップします

Ansibleを実行する前に、ファイルノードとブロックノードを手動でセットアップする必要があります。

ファイルノード

ベースボード管理コントローラ（**BMC**）の設定

ベースボード管理コントローラ（BMC）は、サービスプロセッサとも呼ばれ、オペレーティングシステムがインストールされていない場合やアクセスできない場合でもリモートアクセスを提供できるさまざまなサーバプラットフォームに組み込まれているアウトオブバンド管理機能の一般的な名前です。ベンダーは通常、この機能を独自のブランドで販売しています。たとえば、Lenovo SR665では、BMCはLenovo XClarity Controller（XCC）と呼ばれています。

サーバベンダーのマニュアルに従って、この機能へのアクセスに必要なライセンスを有効にし、BMCがネットワークに接続され、リモートアクセス用に適切に設定されていることを確認してください。

Redfishを使用したBMCベースのフェンシングが必要な場合は、Redfishが有効になっており、BMCインターフェイスにファイルノードにインストールされているOSからアクセスできることを確認します。BMCと動作環境が同じ物理ネットワークインターフェイスを共有している場合は、ネットワークスイッチで特別な設定が必要になることがあります。

システム設定を調整します

セットアップユーティリティ（BIOS/UEFI）インターフェイスを使用して、パフォーマンスを最大化するように設定されていることを確認します。正確な設定と最適な値は、使用しているサーバーモデルによって異なります。ガイダンスはのために提供されて"ファイルノードモデルを確認しました"います。それ以外の場合は、サーバベンダーのドキュメントと、お使いのモデルに基づいたベストプラクティスを参照してください。

オペレーティングシステムをインストールします

リストされているファイルノードの要件に基づいて、サポートされているオペレーティングシステムをインストールし"こちらをご覧ください"ます。Linuxディストリビューションに基づいて、以下の追加手順を参照してください。

Red Hat

、 "RHELシステムを登録および登録する方法"そして "更新を制限する方法" 。

ハイアベイラビリティに必要なパッケージを含むRed Hatリポジトリを有効にします。

```
subscription-manager repo-override --repo=rhel-9-for-x86_64
-highavailability-rpms --add=enabled:1
```

管理ネットワークを設定します

オペレーティングシステムのインバンド管理に必要なネットワークインターフェイスを設定します。具体的な手順は、使用しているLinuxのディストリビューションとバージョンによって異なります。

SSHが有効になっていて、Ansibleコントロールノードからすべての管理インターフェイスにアクセスできることを確認します。

HCAとHBAファームウェアを更新します

すべてのHBAおよびHCAでに記載されているサポート対象のファームウェアバージョンが実行されていることを確認し"NetApp Interoperability Matrix を参照してください"、必要に応じてアップグレードします。NVIDIA ConnectXアダプタに関するその他の推奨事項については"こちらをご覧ください"、こちらを参照してください。

ブロックノード

手順~を実行します "Eシリーズの運用を開始" 各ブロックノードコントローラに管理ポートを設定し、必要に応じて各システムのストレージアレイ名を設定します。

Ansible制御ノードからすべてのブロックノードにアクセスできるようにする以外に、追加の設定は必要ありません。残りのシステム構成はAnsibleで適用/管理されます。

Ansible Control Nodeをセットアップします

ファイルシステムを導入および管理するためのAnsibleコントロールノードをセットアップします。

概要

Ansible制御ノードは、クラスタの管理に使用される物理または仮想Linuxマシンです。次の要件を満たしている必要があります。

- "要件"BeeGFS HAロール（インストールされているAnsible、Python、その他のPythonパッケージなど）

のを確認します。

- 公式情報をご確認ください "Ansibleの制御ノード要件" オペレーティングシステムのバージョンも含まれます。
- すべてのファイルノードとブロックノードに、SSHとHTTPSでアクセスできます。

詳細なインストール手順が"こちらをご覧ください"記載されています。

BeeGFSファイルシステムを定義します

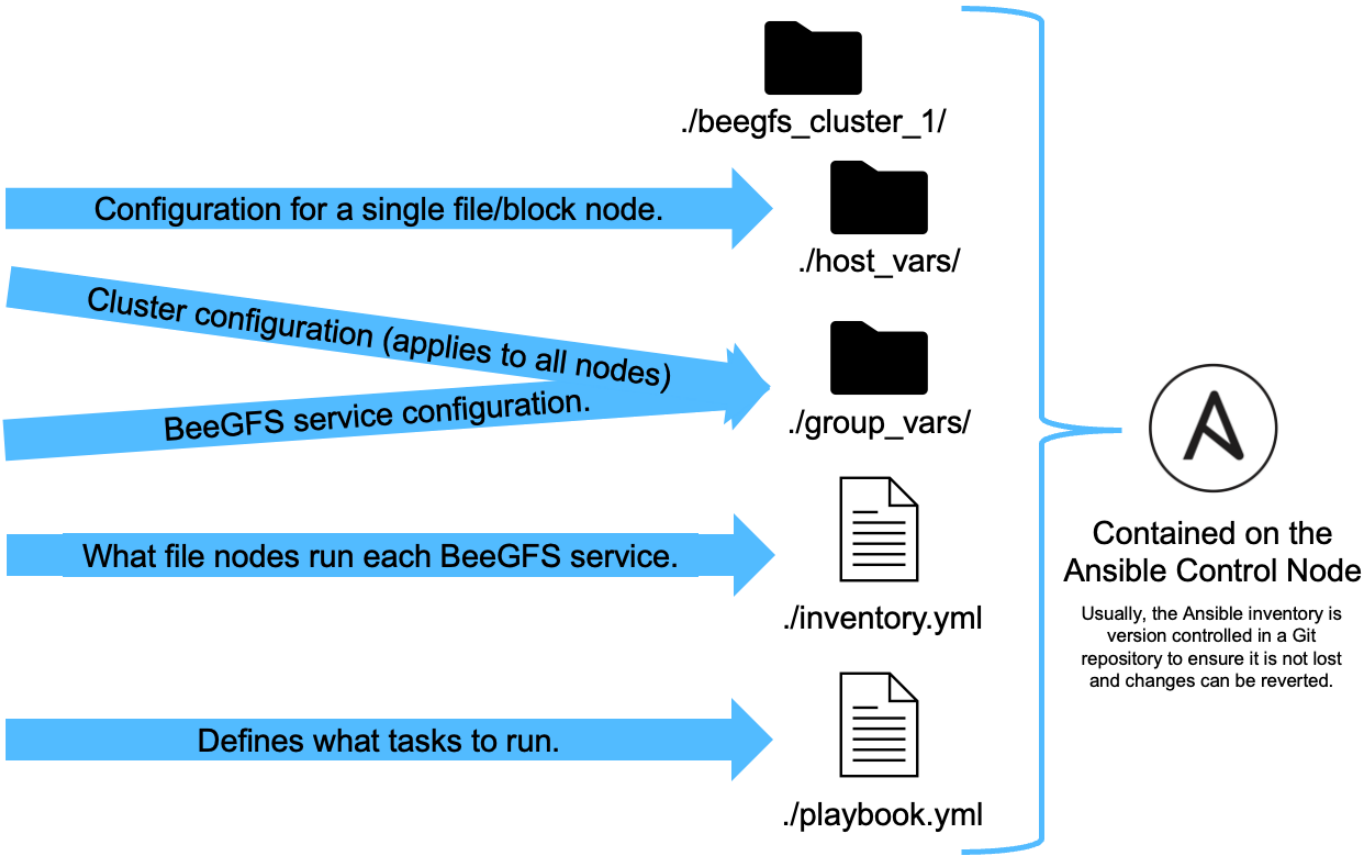
Ansibleのインベントリの概要

Ansibleインベントリは、必要なBeeGFS HAクラスタを定義する一連の構成ファイルです。

概要

の編成については、Ansibleの標準的な手法に従うことを推奨します "在庫"の使用を含む "サブディレクトリ/ファイル" インベントリ全体を1つのファイルに格納する必要はありません。

単一のBeeGFS HAクラスタのAnsibleインベントリは、次のように構成されます。



1つのBeeGFSファイルシステムは複数のHAクラスタにまたがることができるため、大規模なインストールで複数のAnsibleインベントリを使用することが可能です。一般に、問題を回避するために複数のHAクラスタを単一のAnsibleインベントリとして定義することは推奨されません。

手順

1. Ansibleコントロールノードで、導入するBeeGFSクラスタのAnsibleインベントリを含む空のディレクトリを作成します。
   a. ファイルシステムに最終的に複数のHAクラスタが含まれる場合は、まずファイルシステムのディレクトリを作成し、そのあとに各HAクラスタを表すインベントリのサブディレクトリを作成することを推奨します。例：

```
beegfs_file_system_1/
    beegfs_cluster_1/
    beegfs_cluster_2/
    beegfs_cluster_N/
```

2. 導入するHAクラスタのインベントリが格納されているディレクトリに、2つのディレクトリを作成します `group_vars` および `host_vars` 2つのファイルがあります `inventory.yml` および `playbook.yml`。

以降のセクションでは、これらの各ファイルの内容を定義する手順を説明します。

ファイルシステムを計画

Ansibleインベントリを構築する前に、ファイルシステムの導入を計画します。

概要

ファイルシステムを導入する前に、クラスタ内で実行されているすべてのファイルノード、ブロックノード、およびBeeGFSサービスで必要となるIPアドレス、ポート、およびその他の設定を定義する必要があります。具体的な構成はクラスタのアーキテクチャによって異なりますが、ここでは、一般に適用されるベストプラクティスと手順について説明します。

手順

1. IPベースのストレージプロトコル（iSER、iSCSI、NVMe/IB、NVMe/RoCEなど）を使用してファイルノードをブロックノードに接続する場合は、ビルディングブロックごとに次のワークシートに記入します。1つのビルディングブロック内の各直接接続には、一意のサブネットが必要であり、クライアント/サーバ接続に使用されるサブネットと重複しないようにする必要があります。

ファイルノード	IBポート	IP アドレス	ブロックノード	IBポート	物理IP	仮想IP（HDR IBを使用するEF600のみ）
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

各ビルディングブロックのファイルノードとブロックノードが直接接続されている場合、複数のビルディングブロックで同じIP/スキームを再利用することがよくあります。

2. ストレージネットワークにInfiniBandまたはRDMA over Converged Ethernet（RoCE）を使用しているかどうかに関係なく、次のワークシートに記入して、HAクラスタサービス、BeeGFSファイルサービス、およびクライアントが通信するために使用するIP範囲を決定します。

目的	**InfiniBand**ポート	**IP**アドレスまたは**IP**範囲
BeeGFSクラスタIP	<INTERFACE(s)>	<RANGE>
BeeGFS Managementの略	<INTERFACE(s)>	<IP(s)>
BeeGFSメタデータ	<INTERFACE(s)>	<RANGE>
BeeGFS Storage（BeeGFSストレージ）	<INTERFACE(s)>	<RANGE>
BeeGFSクライアント	<INTERFACE(s)>	<RANGE>

   a. 単一のIPサブネットを使用する場合は、ワークシートが1つだけ必要です。それ以外の場合は、2つ目のサブネットのワークシートにも記入してください。

3. 上記に基づいて、クラスタ内の各ビルディングブロックに対して、実行するBeeGFSサービスを定義する次のワークシートに記入します。各サービスについて、優先/セカンダリファイルノード、ネットワークポート、フローティングIP、NUMAゾーン割り当て（必要な場合）、およびターゲットに使用するブロックノードを指定します。ワークシートに記入する際は、次のガイドラインを参照してください。

   a. BeeGFSサービスをいずれかとして指定します `mgmt.yml`、 `meta_<ID>.yml`または `storage_<ID>.yml` ここで'IDは'このファイルシステム内のすべてのBeeGFSサービスの一意の番号を表しますこの規則により、以降のセクションでは、各サービスを設定するためのファイルを作成する際に、このワークシートを簡単に参照できます。

   b. BeeGFSサービスのポートは、特定のビルディングブロック全体で一意である必要があります。ポートの競合を回避するために、同じポート番号のサービスを同じファイルノード上で実行することはできません。

   c. 必要なサービスが複数のブロックノード/ストレージプール（すべてのボリュームを同じコントローラに所有する必要はない）のボリュームを使用できる場合。複数のサービスで同じブロックノードやストレージプール構成を共有することもできます（個 々 のボリュームはこのあとのセクションで定義します）。

BeeGFSサービス（ファイル名）	ファイルノード	ポート	フローティング**IP**	**NUMA**ゾーン	ブロックノード	ストレージプール	所有コントローラ
<SERVICE TYPE>_<ID> .yml	<PREFERRED FILE NODE> <SECONDARY FILE NODE(s)>	<PORT>	<INTERFACE>：<IP/SUBNET> <INTERFACE>：<IP/SUBNET>	<NUMA NODE/ZONE>	<BLOCK NODE>	<STORAGE POOL/VOLUME GROUP>	<A OR B>

標準的な規則、ベストプラクティス、記入例のワークシートの詳細については"ベストプラクティス""BeeGFS

ビルディングブロックを定義します"、『BeeGFS on NetApp Verified Architecture』のおよびのセクションを参照してください。

ファイルノードとブロックノードを定義します

個 々 のファイルノードを設定します

ホスト変数（host_vars）を使用して、個 々 のファイルノードの設定を指定します。

概要

このセクションでは、の入力手順について説明します `host_vars/<FILE_NODE_HOSTNAME>.yml` クラスタ内の各ファイルノードのファイル。これらのファイルには、特定のファイルノードに固有の設定のみを含める必要があります。これには、次のような一般

- AnsibleでIPまたはホスト名を定義して、ノードへの接続に使用する必要があります。
- HAクラスタサービス（PacemakerとCorosync）で他のファイルノードとの通信に使用するインターフェイスおよびクラスタIPを追加で設定しています。デフォルトでは、これらのサービスは管理インターフェイスと同じネットワークを使用しますが、冗長性を確保するために追加のインターフェイスを使用できる必要があります。一般的には、ストレージネットワークに追加のIPを定義して、クラスタまたは管理ネットワークを追加する必要を回避します。
  - クラスタ通信に使用されるネットワークのパフォーマンスは、ファイルシステムのパフォーマンスにとっては重要ではありません。デフォルトのクラスタ構成では、通常、少なくとも1Gb/秒ネットワークを使用すると、ノード状態の同期やクラスタリソース状態の変更の調整など、クラスタ処理に十分なパフォーマンスが提供されます。低速/ビジーなネットワークでは、原因 リソースの状態が通常よりも長くなる可能性があります。また、ノードが妥当な時間内にハートビートを送信できない場合、ノードがクラスタから削除されることがあります。
- 目的のプロトコルを介したブロックノードへの接続に使用するインターフェイスの設定（iSCSI / iSER、NVMe/IB、NVMe/RoCE、FCPなど）

手順

"ファイルシステムを計画"セクションで定義したIPアドレス指定方式を参照して、クラスタ内のファイルノードごとにファイルを作成し `host_vars/<FILE_NODE_HOSTNAME>/yml`、次のように設定します。

1. 上部に、ノードへのSSHとノードの管理にAnsibleで使用するIPまたはホスト名を指定します。

```
ansible_host: "<MANAGEMENT_IP>"
```

2. クラスタトラフィックに使用できる追加のIPを設定します。
   a. ネットワークタイプがの場合 "InfiniBand（IPoIBを使用）"：

```
eseries_ipoib_interfaces:
- name: <INTERFACE>  # Example: ib0 or i1b
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

b. ネットワークタイプがの場合 "RDMA over Converged Ethernet（RoCE）"：

```
eseries_roce_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

c. ネットワークタイプがの場合 "イーサネット（TCPのみ、RDMAなし）"：

```
eseries_ip_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
- name: <INTERFACE>  # Additional interfaces as needed.
  address: <IP/SUBNET>
```

3. 優先IPが高い順に、クラスタトラフィックに使用するIPを指定します。

```
beegfs_ha_cluster_node_ips:
- <MANAGEMENT_IP> # Including the management IP is typically but not
required.
- <IP_ADDRESS>    # Ex: 100.127.100.1
- <IP_ADDRESS>    # Additional IPs as needed.
```

ステップ2で設定したIPSは、に含まれていないかぎり、クラスタIPとして使用されません `beegfs_ha_cluster_node_ips` リストこれにより、必要に応じて他の目的にも使用できるAnsibleを使用して追加のIP /インターフェイスを設定できます。

4. IPベースのプロトコルを使用してノードをブロックするためにファイルノードが通信する必要がある場合は、IPを適切なインターフェイスに設定し、そのプロトコルのインストールまたは設定に必要なパッケージをすべて設定する必要があります。

a. を使用する場合 "iSCSI"：

```
eseries_iscsi_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
```

b. を使用する場合 "iSER"：

```
eseries_ib_iser_interfaces:
- name: <INTERFACE>  # Example: ib0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
  configure: true # If the file node is directly connected to the
block node set to true to setup OpenSM.
```

c. を使用する場合 "NVMe/IB"：

```
eseries_nvme_ib_interfaces:
- name: <INTERFACE>  # Example: ib0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
  configure: true # If the file node is directly connected to the
block node set to true to setup OpenSM.
```

d. を使用する場合 "NVMe/RoCE"：

```
eseries_nvme_roce_interfaces:
- name: <INTERFACE>  # Example: eth0.
  address: <IP/SUBNET> # Example: 100.127.100.1/16
```

e. その他のプロトコル：

i. を使用する場合 "NVMe/FC"個 々 のインターフェイスを設定する必要はありません。BeeGFSクラスタの導入により、プロトコルが自動的に検出され、必要に応じて要件がインストール/設定されます。ファブリックを使用してファイルノードとブロックノードを接続する場合は、ネットアップとスイッチベンダーのベストプラクティスに従ってスイッチを適切にゾーニングしてください。

ii. FCPまたはSASを使用する場合、追加のソフトウェアをインストールまたは設定する必要はありません。FCPを使用する場合は、次に示す手順でスイッチが適切にゾーニングされていることを確認 "ネットアップ" スイッチベンダーのベストプラクティスを確認してください。

iii. 現時点では、IB SRPの使用は推奨されていません。Eシリーズのブロックノードでサポートされているものに応じて、NVMe/IBまたはiSERを使用します。

をクリックします "こちらをご覧ください" たとえば、単一のファイルノードを表す完全なインベントリファイルなどです。

Advanced：イーサネットと**InfiniBand**モードの間で**NVIDIA ConnectX VPI**アダプタを切り替えます

NVIDIA ConnectX-Virtual Protocol Interconnect ®（VPI）アダプタは、InfiniBandとイーサネットの両方をトランスポートレイヤとしてサポートします。モード間の切り替えは自動的にはネゴシエートされないため、に含まれているオープンソースパッケージを使用して設定する必要があります <code>mstconfig</code> <code>mstflint</code> "NVIDIAファームウェアツール(MFT)"。アダプタのモードを変更する必要があるのは一度だけです。これは手動で行うことも、インベントリのセクションを使用して設定されたインターフェイスの一部としてAnsibleインベントリに含めることもでき <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> 、自動的にチェック/適用されます。

たとえば、InfiniBandモードのインターフェイスをイーサネットに変更して、RoCEに使用できるようにするには、次のコマンドを実行します。

1. 設定する各インターフェイスについて、を指定します `mstconfig` を指定するマッピング（またはディクショナリ）として指定します `LINK_TYPE_P<N>` ここで、 `<N>` は、インターフェイスのHCAのポート番号で決まります。。 `<N>` の値はを実行して確認できます `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` PCIスロット名の最後の数字に1を追加し、10進数に変換します。

   a. たとえば、を指定します `PCI_SLOT_NAME=0000:2f:00.2` （2+1 → HCAポート3）→ `LINK_TYPE_P3: eth`：

```
eseries_roce_interfaces:
- name: <INTERFACE>
  address: <IP/SUBNET>
  mstconfig:
    LINK_TYPE_P3: eth
```

詳細については、を参照してください "NetApp Eシリーズホストコレクションのドキュメント" をクリックします。

個 々 のブロックノードを設定します

ホスト変数（host_vars）を使用して個 々 のブロックノードの設定を指定します。

概要

このセクションでは、の入力手順について説明します `host_vars/<BLOCK_NODE_HOSTNAME>.yml` クラスタ内のブロックノードごとにファイルを作成します。これらのファイルに含まれるのは、特定のブロックノードに固有の設定のみである必要があります。これには、次のような一般

- システム名（System Managerに表示）。
- いずれかのコントローラのHTTPS URL（REST APIを使用したシステムの管理に使用）。
- このブロックノードへの接続に使用するストレージプロトコルファイルノード。
- IPアドレスなどのホストインターフェイスカード（HIC）ポートを設定する（必要な場合）。

手順

"ファイルシステムを計画"セクションで定義したIPアドレス指定方式を参照して、クラスタ内のブロックノードごとにファイルを作成し `host_vars/<BLOCK_NODE_HOSTNAME>/yml`、次のように設定します。

1. 上部で、いずれかのコントローラのシステム名とHTTPS URLを指定します。

```
eseries_system_name: <SYSTEM_NAME>
eseries_system_api_url:
https://<MANAGEMENT_HOSTNAME_OR_IP>:8443/devmgr/v2/
```

2. を選択します "プロトコル" ファイルノードはこのブロックノードへの接続に使用します。
   a. サポートされるプロトコル： `auto、 iscsi、 fc、 sas、 ib_srp、 ib_iser、 nvme_ib、 nvme_fc、 nvme_roce。`

```
eseries_initiator_protocol: <PROTOCOL>
```

3. 使用するプロトコルによっては、HICポートの設定を追加する必要があります。HICポートの設定は、必要に応じて各コントローラの設定の一番上のエントリが各コントローラの物理的な左端のポートに対応し、一番下のポートが最も右のポートになるように定義する必要があります。現在使用していないポートでも、すべてのポートで有効な設定が必要です。

> EF600ブロックノードでHDR（200GB）InfiniBandまたは200GBのRoCEを使用している場合は、次のセクションも参照してください。

   a. iSCSIの場合：

```
eseries_controller_iscsi_port:
  controller_a:         # Ordered list of controller A channel
definition.
    - state:            # Whether the port should be enabled.
Choices: enabled, disabled
      config_method:    # Port configuration method Choices: static,
dhcp
      address:          # Port IPv4 address
      gateway:          # Port IPv4 gateway
      subnet_mask:      # Port IPv4 subnet_mask
      mtu:              # Port IPv4 mtu
    - (...)             # Additional ports as needed.
  controller_b:         # Ordered list of controller B channel
definition.
    - (...)             # Same as controller A but for controller B

# Alternatively the following common port configuration can be
defined for all ports and omitted above:
eseries_controller_iscsi_port_state: enabled         # Generally
specifies whether a controller port definition should be applied
Choices: enabled, disabled
eseries_controller_iscsi_port_config_method: dhcp    # General port
configuration method definition for both controllers. Choices:
static, dhcp
eseries_controller_iscsi_port_gateway:               # General port
IPv4 gateway for both controllers.
eseries_controller_iscsi_port_subnet_mask:           # General port
IPv4 subnet mask for both controllers.
eseries_controller_iscsi_port_mtu: 9000              # General port
maximum transfer units (MTU) for both controllers. Any value greater
than 1500 (bytes).
```

b. iSERの場合：

```
eseries_controller_ib_iser_port:
  controller_a:    # Ordered list of controller A channel address
definition.
    -              # Port IPv4 address for channel 1
    - (...)        # So on and so forth
  controller_b:    # Ordered list of controller B channel address
definition.
```

c. NVMe/IB：

```
eseries_controller_nvme_ib_port:
  controller_a:    # Ordered list of controller A channel address
definition.
    -              # Port IPv4 address for channel 1
    - (...)        # So on and so forth
  controller_b:    # Ordered list of controller B channel address
definition.
```

d. NVMe/RoCEの場合：

```
eseries_controller_nvme_roce_port:
  controller_a:         # Ordered list of controller A channel
definition.
    - state:            # Whether the port should be enabled.
      config_method:    # Port configuration method Choices: static,
dhcp
      address:          # Port IPv4 address
      subnet_mask:      # Port IPv4 subnet_mask
      gateway:          # Port IPv4 gateway
      mtu:              # Port IPv4 mtu
      speed:            # Port IPv4 speed
  controller_b:         # Ordered list of controller B channel
definition.
    - (...)             # Same as controller A but for controller B

# Alternatively the following common port configuration can be
defined for all ports and omitted above:
eseries_controller_nvme_roce_port_state: enabled         # Generally
specifies whether a controller port definition should be applied
Choices: enabled, disabled
eseries_controller_nvme_roce_port_config_method: dhcp    # General
port configuration method definition for both controllers. Choices:
static, dhcp
eseries_controller_nvme_roce_port_gateway:               # General
port IPv4 gateway for both controllers.
eseries_controller_nvme_roce_port_subnet_mask:           # General
port IPv4 subnet mask for both controllers.
eseries_controller_nvme_roce_port_mtu: 4200              # General
port maximum transfer units (MTU). Any value greater than 1500
(bytes).
eseries_controller_nvme_roce_port_speed: auto            # General
interface speed. Value must be a supported speed or auto for
automatically negotiating the speed with the port.
```

e. FCプロトコルとSASプロトコルについては、追加の設定は必要ありません。SRPの使用は推奨されません。

iSCSI CHAPの設定など、HICポートとホストプロトコルを設定するその他のオプションについては、を参照してください "ドキュメント" SANtricity コレクションに含まれています。注：BeeGFSを導入する場合は'ストレージ・プール'ボリューム構成'その他のプロビジョニング・ストレージの設定は他の場所で行いますこのファイルでは定義しないでください

をクリックします "こちらをご覧ください" たとえば、1つのブロックノードを表す完全なインベントリファイルなどです。

NetApp EF600ブロックノードで**HDR（200GB）InfiniBand**または**200GB RoCE**を使用：

EF600でHDR（200GB）InfiniBandを使用するには、物理ポートごとに2つ目の「仮想」IPを設定する必要があります。以下は、デュアルポートInfiniBand HDR HICを搭載したEF600の正しい設定方法の例です。

```
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.1.101    # Port 2a (virtual)
    - 192.168.2.101    # Port 2b (virtual)
    - 192.168.1.100    # Port 2a (physical)
    - 192.168.2.100    # Port 2b (physical)
  controller_b:
    - 192.168.3.101    # Port 2a (virtual)
    - 192.168.4.101    # Port 2b (virtual)
    - 192.168.3.100    # Port 2a (physical)
    - 192.168.4.100    # Port 2b (physical)
```

Common File Node Configurationを指定します

グループ変数（group_vars）を使用して共通ファイルノード設定を指定します。

概要

Appleがすべてのファイルノードに適用される構成は、で定義されます `group_vars/ha_cluster.yml`。一般的には次のものが含ま

- 各ファイルノードに接続してログインする方法の詳細。
- 一般的なネットワーク構成。
- 自動リブートが許可されるかどうか。
- ファイアウォールとSELinuxの状態を設定する方法。
- アラートやフェンシングなどのクラスタ構成。
- パフォーマンスの調整。
- Common BeeGFSサービスの設定

このファイルで設定したオプションは、たとえば、異なるハードウェアモデルを混在させる場合や、ノードごとに異なるパスワードを設定する場合など、個 々 のファイルノードに定義することもできます。個 々 のファイルノードの設定は、このファイルの設定よりも優先されます。

手順

ファイルを作成します `group_vars/ha_cluster.yml` 次のように入力します。

1. リモートホストでAnsible Controlノードがどのように認証されるかを指定します。

```
ansible_ssh_user: root
ansible_become_password: <PASSWORD>
```

特に本番環境では、パスワードをプレーンテキストで保存しないでください。代わりにAnsible Vaultを使用します（を参照） "Ansible Vaultを使用したコンテンツの暗号化"）またはをクリックします `--ask-become-pass` プレイブックを実行する際のオプション。状況に応じて `ansible_ssh_user` はすでにrootであるため、必要に応じてを省略できます `ansible_become_password`。

2. イーサネットインターフェイスまたはInfiniBandインターフェイス（クラスタIPなど）に静的IPを設定していて、複数のインターフェイスが同じIPサブネットにある場合（たとえば、ib0が192.168.1.10/24を使用し、ib1が192.168.1.11/24を使用している場合）、マルチホームサポートが正常に機能するためには、追加のIPルーティングテーブルとルールを設定する必要があります。提供されているネットワークインターフェイス設定フックを次のように有効にします。

```
eseries_ip_default_hook_templates:
  - 99-multihoming.j2
```

3. クラスタを導入する際は、ストレージプロトコルによっては、リモートブロックデバイスを検出しやすくするためにノードをリブートしたり（Eシリーズボリューム）、構成の他の要素を適用したりする必要があります。デフォルトでは、ノードはリブート前にプロンプトを表示しますが、次の項目を指定することでノードの自動再起動を許可できます。

```
eseries_common_allow_host_reboot: true
```

   a. リブート後のデフォルトでは、ブロックデバイスやその他のサービスの準備ができていることを確認するために、Ansibleはsystemdまで待機します `default.target` は、導入を続行する前に到達しています。NVMe/IBを使用する場合は、リモートデバイスの初期化、検出、および接続に時間がかかることがあります。その結果、導入の途中で自動化が失敗し続ける可能性があります。NVMe/IBを使用する場合にこの問題を回避するには、以下の条件も定義します。

```
eseries_common_reboot_test_command: "! systemctl status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
```

4. BeeGFSサービスとHAクラスタサービスが通信するためには、多数のファイアウォールポートが必要です。firwallを手動で設定する場合を除き（非推奨）、必要なファイアウォールゾーンを作成し、ポートを自動的に開くように次のように指定します。

```
beegfs_ha_firewall_configure: True
```

5. SELinuxは現時点でサポートされていないため、競合を回避するために（特にRDMAを使用している場合）状態をdisabledに設定することを推奨します。SELinuxが無効になっていることを確認するには、次のように設定

```
eseries_beegfs_ha_disable_selinux: True
eseries_selinux_state: disabled
```

6. ファイルノードが通信できるように認証を設定し、組織のポリシーに基づいてデフォルト設定を必要に応じて調整します。

```
beegfs_ha_cluster_name: hacluster                    # BeeGFS HA cluster
name.
beegfs_ha_cluster_username: hacluster                # BeeGFS HA cluster
username.
beegfs_ha_cluster_password: hapassword               # BeeGFS HA cluster
username's password.
beegfs_ha_cluster_password_sha512_salt: randomSalt # BeeGFS HA cluster
username's password salt.
```

7. "ファイルシステムを計画"セクションに基づいて、このファイルシステムのBeeGFS管理IPを指定します。

```
beegfs_ha_mgmtd_floating_ip: <IP ADDRESS>
```

> 一見冗長に見えても'beegfs_ha_gmtd_floating_ip'は'1つのHAクラスタを超えてBeeGFSファイルシステムを拡張する場合に重要です以降のHAクラスタは、BeeGFS管理サービスを追加せずに導入され、最初のクラスタが提供する管理サービスをポイントします。

8. 必要に応じてEメールアラートを有効にします。

```
beegfs_ha_enable_alerts: True
# E-mail recipient list for notifications when BeeGFS HA resources
change or fail.
beegfs_ha_alert_email_list: ["<EMAIL>"]
# This dictionary is used to configure postfix service
(/etc/postfix/main.cf) which is required to set email alerts.
beegfs_ha_alert_conf_ha_group_options:
      # This parameter specifies the local internet domain name. This is
optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com)
      mydomain: <MY_DOMAIN>
beegfs_ha_alert_verbosity: 3
#  1) high-level node activity
#  3) high-level node activity + fencing action information + resources
(filter on X-monitor)
#  5) high-level node activity + fencing action information + resources
```

9. フェンシングを有効にすることを強く推奨します。そうしないと、プライマリノードで障害が発生したときに、セカンダリノードでサービスが開始されないようにブロックされます。
   a. 次の項目を指定して、フェンシングをグローバルに有効にします

```
beegfs_ha_cluster_crm_config_options:
  stonith-enabled: True
```

      i. メモ必要に応じて、サポートされているものを "クラスタ・プロパティ" ここで指定することもできます。BeeGFS HAロールには十分にテストされた機能が多数付属しているため、これらの調整は通常は必要ありません "デフォルト値です"。

   b. 次に、フェンシングエージェントを選択して構成します。
      i. オプション1：APC Power Distribution Unit（PDU;配電ユニット）を使用してフェンシングをイネーブルにするには、次の手順

```
beegfs_ha_fencing_agents:
  fence_apc:
    - ipaddr: <PDU_IP_ADDRESS>
      login: <PDU_USERNAME>
      passwd: <PDU_PASSWORD>
      pcmk_host_map:
"<HOSTNAME>:<PDU_PORT>,<PDU_PORT>;<HOSTNAME>:<PDU_PORT>,<PDU_PORT>
"
```

      ii. オプション2：Lenovo XCC（および他のBMC）が提供するRedfish APIを使用してフェンシングを有効にするには、次の手順を実行します。

```
redfish: &redfish
  username: <BMC_USERNAME>
  password: <BMC_PASSWORD>
  ssl_insecure: 1 # If a valid SSL certificate is not available
specify "1".

beegfs_ha_fencing_agents:
  fence_redfish:
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
```

iii. 他のフェンシングエージェントの設定の詳細については、を参照してください "Red Hat ドキュメント"。

10. BeeGFS HAロールでは、パフォーマンスをさらに最適化するために、さまざまなチューニングパラメータを適用できます。これには、カーネルメモリ使用率の最適化や、ブロックデバイスI/Oなどのパラメータが含まれます。このロールには、NetApp E-Seriesブロックノードを使用したテストに基づく合理的なセットが付属してい "デフォルト値です" ますが、デフォルトでは次を指定しない限り、これらは適用されません。

```
beegfs_ha_enable_performance_tuning: True
```

a. 必要に応じて、ここでデフォルトのパフォーマンス調整に変更を加えます。詳細については、完全なドキュメントを参照して "パフォーマンス調整パラメータ" ください。

11. BeeGFSサービスに使用されるフローティングIPアドレス（論理インターフェイスとも呼ばれます）がファイルノード間でフェイルオーバーできるようにするには、すべてのネットワークインターフェイスに一貫した名前を付ける必要があります。デフォルトでは、ネットワークインターフェイス名はカーネルによって生成されます。これは、同じPCIeスロットにネットワークアダプタが搭載された同一のサーバモデルであっても、一貫した名前が生成される保証はありません。これは、装置が展開され、生成されたインターフェイス名が認識される前にインベントリを作成する場合にも役立ちます。サーバまたはのブロック図に基づいて、一貫したデバイス名を使用できるようにします `lshw -class network -businfo` 出力で、目的のPCIeアドレスと論理インターフェイスのマッピングを次のように指定します。

a. InfiniBand（IPoIB）ネットワークインターフェイスに対応しています。

```
eseries_ipoib_udev_rules:
  "<PCIe ADDRESS>": <NAME> # Ex: 0000:01:00.0: i1a
```

b. イーサネットネットワークインターフェイスの場合：

```
eseries_ip_udev_rules:
  "<PCIe ADDRESS>": <NAME> # Ex: 0000:01:00.0: e1a
```

インターフェイスの名前を変更したときの競合を回避するには（名前を変更できないようにするため）、eth0、ens9f0、ib0、ibs4f0などの潜在的なデフォルト名は使用しないでください。一般的な命名規則としては、イーサネットまたはInfiniBandには「e」または「i」を使用し、続いてPCIeスロット番号とポートを示す文字を使用します。たとえば、スロット3にInfiniBandアダプタの2番目のポートはi3bとなります。

検証済みファイルノードモデルを使用している場合は、をクリックします "こちらをご覧ください" PCIeアドレスと論理ポートのマッピングの例

12. 必要に応じて、クラスタ内のすべてのBeeGFSサービスに適用する設定を指定します。デフォルトの設定値が表示され "こちらをご覧ください"、サービス単位の設定は他の場所で指定されます。

    a. BeeGFS管理サービス：

    ```
    beegfs_ha_beegfs_mgmtd_conf_ha_group_options:
      <OPTION>: <VALUE>
    ```

    b. BeeGFSメタデータサービス：

    ```
    beegfs_ha_beegfs_meta_conf_ha_group_options:
      <OPTION>: <VALUE>
    ```

    c. BeeGFSストレージサービス：

    ```
    beegfs_ha_beegfs_storage_conf_ha_group_options:
      <OPTION>: <VALUE>
    ```

13. BeeGFS 7.2.7および7.3.1以降 "接続認証" 設定または明示的に無効にする必要があります。Ansibleベースの導入を使用してこれを設定するには、いくつかの方法があります。

    a. デフォルトでは、展開によって自動的に接続認証が設定され、が生成されます `connauthfile` これはすべてのファイルノードに配布され、BeeGFSサービスとともに使用されます。このファイルは、Ansibleの制御ノードにも配置/管理されます `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` このファイルシステムにアクセスする必要のあるクライアントで再利用できるように、安全に保管する必要があります。

        i. 新しいキーを生成するには、をクリックします `-e "beegfs_ha_conn_auth_force_new=True` Ansibleプレイブックを実行している場合。注：これは、がの場合は無視されます `beegfs_ha_conn_auth_secret` が定義されている。

        ii. 詳細オプションについては、に付属のデフォルトの一覧を参照して "BeeGFS HAロール"ください。

    b. カスタムシークレットを使用するには、で以下を定義します `ha_cluster.yml`：

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. 接続認証は完全に無効にできます（非推奨）。

```
beegfs_ha_conn_auth_enabled: false
```

をクリックします "こちらをご覧ください" 一般的なファイルノード設定を表す完全なインベントリファイルの例を次に示します。

NetApp EF600ブロックノードで**HDR（200GB）InfiniBand**を使用：

EF600でHDR（200GB）InfiniBandを使用するには、サブネットマネージャが仮想化をサポートしている必要があります。ファイルノードとブロックノードがスイッチを使用して接続されている場合は、ファブリック全体に対してサブネットマネージャで有効にする必要があります。

ブロックノードとファイルノードがInfiniBandを使用して直接接続されている場合は `opensm`、ブロックノードに直接接続されているインターフェイスごとに、各ファイルノードでのインスタンスを設定する必要があります。そのためには、 `configure: true`whenを指定し"ファイルノードストレージインターフェイスを設定しています"ます。

現在、サポートされているLinuxディストリビューションに同梱されているの受信トレイバージョンで `opensm` は、仮想化はサポートされていません。代わりに、NVIDIA OpenFabrics Enterprise Distribution（OFED）からのバージョンをインストールして設定する必要があります `opensm` 。Ansibleによる導入もサポートされていますが、いくつかの追加手順が必要です。

1. curlまたは任意のツールを使用して、セクションに記載されているOpenSMのバージョンのパッケージをNVIDIAのWebサイトからディレクトリにダウンロードし "テクノロジ要件" `<INVENTORY>/packages/` ます。例：

```
curl -o packages/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.4/x86_64/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
curl -o packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.4/x86_64/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
```

2. の下 `group_vars/ha_cluster.yml` 次の設定を定義します。

```
### OpenSM package and configuration information
eseries_ib_opensm_allow_upgrades: true
eseries_ib_opensm_skip_package_validation: true
eseries_ib_opensm_rhel_packages: []
eseries_ib_opensm_custom_packages:
  install:
    - files:
        add:
          "packages/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
          "packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
    - packages:
        add:
          - /tmp/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
          - /tmp/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
  uninstall:
    - packages:
        remove:
          - opensm
          - opensm-libs
      files:
        remove:
          - /tmp/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
          - /tmp/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm

eseries_ib_opensm_options:
  virt_enabled: "2"
```

Common Block Node Configurationを指定します

グループ変数（group_vars）を使用して共通ブロックノード設定を指定します。

概要

すべてのブロックノードに対してAppleが実施する必要がある設定は、で定義します `group_vars/eseries_storage_systems.yml`。一般的には次のものが含ま

- Ansible制御ノードが、ブロックノードとして使用されるEシリーズストレージシステムに接続する方法の詳細。
- ノードで実行するファームウェア、NVSRAM、およびドライブファームウェアのバージョン。

- キャッシュ設定、ホスト構成、ボリュームのプロビジョニング方法に関する設定を含むグローバル構成。

> このファイルで設定したオプションは、個 々 のブロックノードに定義することもできます。たとえば、異なるハードウェアモデルを混在させる場合や、ノードごとに異なるパスワードを設定する場合などです。個 々 のブロックノードの設定は、このファイルの設定よりも優先されます。

手順

ファイルを作成します `group_vars/eseries_storage_systems.yml` 次のように入力します。

1. Ansibleは、SSHを使用してブロックノードに接続するのではなく、REST APIを使用します。これを実現するには、以下を設定する必要があります

```
ansible_connection: local
```

2. 各ノードを管理するためのユーザ名とパスワードを指定してください。ユーザ名はオプションで省略できます（デフォルトはadmin）。それ以外の場合はadmin権限を持つ任意のアカウントを指定できます。また、SSL証明書を検証するかどうかを指定します。無視するかどうかを指定します。

```
eseries_system_username: admin
eseries_system_password: <PASSWORD>
eseries_validate_certs: false
```

> プレーンテキストでパスワードを一覧表示することは推奨されません。Ansibleバックアップツールを使用するか、を提供します `eseries_system_password` --extra -varsを使用してAnsibleを実行している場合。

3. 必要に応じて、ノードにインストールするコントローラファームウェア、NVSRAM、ドライブファームウェアを指定します。これらのファイルは、にダウンロードする必要があります `packages/` Ansibleを実行する前のディレクトリ。Eシリーズコントローラのファームウェアとnvsramをダウンロードできます "こちらをご覧ください" ドライブファームウェアを定義できます "こちらをご覧ください"：

```
eseries_firmware_firmware: "packages/<FILENAME>.dlp" # Ex.
"packages/RCB_11.80GA_6000_64cc0ee3.dlp"
eseries_firmware_nvsram: "packages/<FILENAME>.dlp" # Ex.
"packages/N6000-880834-D08.dlp"
eseries_drive_firmware_firmware_list:
  - "packages/<FILENAME>.dlp"
  # Additional firmware versions as needed.
eseries_drive_firmware_upgrade_drives_online: true # Recommended unless
BeeGFS hasn't been deployed yet, as it will disrupt host access if set
to "false".
```

この設定を指定すると、コントローラのリブート（必要な場合）を含むすべてのファームウェアがAnsibleで自動的に更新され、追加のプロンプトは表示されません。これはBeeGFS/ホストI/Oに影響しないものと想定されていますが、原因 によってパフォーマンスが一時的に低下する可能性があります。

4. グローバルシステム構成のデフォルトを調整します。ここに示すオプションと値は、ネットアップのBeeGFSには一般的に推奨される設定ですが、必要に応じて調整することもできます。

```
eseries_system_cache_block_size: 32768
eseries_system_cache_flush_threshold: 80
eseries_system_default_host_type: linux dm-mp
eseries_system_autoload_balance: disabled
eseries_system_host_connectivity_reporting: disabled
eseries_system_controller_shelf_id: 99 # Required by default.
```

5. グローバルなボリュームプロビジョニングをデフォルトに設定ここに示すオプションと値は、ネットアップのBeeGFSには一般的に推奨される設定ですが、必要に応じて調整することもできます。

```
eseries_volume_size_unit: pct # Required by default. This allows volume
capacities to be specified as a percentage, simplifying putting together
the inventory.
eseries_volume_read_cache_enable: true
eseries_volume_read_ahead_enable: false
eseries_volume_write_cache_enable: true
eseries_volume_write_cache_mirror_enable: true
eseries_volume_cache_without_batteries: false
```

6. 必要に応じて、次のベストプラクティスに留意しながら、ストレージプールとボリュームグループ用のドライブがAnsibleで選択される順序を調整します。

   a. 管理ボリューム/メタデータボリュームに使用する（小容量の可能性がある）ドライブから先に、ストレージボリュームを最後にリストします。

   b. ディスクシェルフ/ドライブエンクロージャのモデルに基づいて、使用可能なドライブチャネル間でドライブ選択順序を分散してください。たとえば、EF600で拡張が行われていない場合、ドライブ0_{11}はドライブチャネル1に、ドライブ$_{12}$23はドライブチャネルに配置されます。したがって、ドライブ選択のバランスを取るための戦略は、を選択することです `disk shelf:drive` 99：0、99：23、99：1、99：22などエンクロージャが複数ある場合は、1桁目の数字がドライブシェルフIDを表します。

```
# Optimal/recommended order for the EF600 (no expansion):
eseries_storage_pool_usable_drives:
"99:0,99:23,99:1,99:22,99:2,99:21,99:3,99:20,99:4,99:19,99:5,99:18,99
:6,99:17,99:7,99:16,99:8,99:15,99:9,99:14,99:10,99:13,99:11,99:12"
```

をクリックします "こちらをご覧ください" に、一般的なブロックノード構成を表す完全なインベントリファイルの例を示します。

BeeGFSサービスを定義します

BeeGFS管理サービスを定義します

BeeGFSサービスは、グループ変数（group_vars）を使用して設定します。

概要

このセクションでは、BeeGFS管理サービスの定義について説明します。HAクラスタには、このタイプのサービスを特定のファイルシステムに対して1つだけ配置する必要があります。このサービスには、次の定義が含まれます。

- サービスタイプ（管理）。
- このBeeGFSサービスにのみ適用する設定を定義します。
- このサービスに到達できる1つ以上のフローティングIP（論理インターフェイス）の設定。
- このサービス（BeeGFS管理ターゲット）のデータを格納する場所と方法を指定します。

手順

新しいファイルを作成し `group_vars/mgmt.yml`、"ファイルシステムを計画"セクションを参照して次のように入力します。

1. BeeGFS管理サービスの設定を表すファイルを指定します。

```
beegfs_service: management
```

2. このBeeGFSサービスにのみ適用する設定を定義します。通常、からサポートされている設定パラメータを指定してクォータを有効にする必要がないかぎり、この設定は管理サービスには必要ありません `beegfs-mgmtd.conf` 含めることができます。次のパラメータは、自動的に設定されますが、ここでは指定しないでください。 `storeMgmtdDirectory`、 `connAuthFile`、 `connDisableAuthentication`、 `connInterfacesFile`および `connNetFilterFile`。

```
beegfs_ha_beegfs_mgmtd_conf_resource_group_options:
  <beegfs-mgmt.conf:key>:<beegfs-mgmt.conf:value>
```

3. 他のサービスやクライアントがこのサービスへの接続に使用する1つまたは複数のフローティングIPを設定します（これにより、自動的にBeeGFSが設定されます） `connInterfacesFile` オプション）：

```
floating_ips:
  - <INTERFACE>:<IP/SUBNET> # Primary interface. Ex.
i1b:100.127.101.0/16
  - <INTERFACE>:<IP/SUBNET> # Secondary interface(s) as needed.
```

4. 必要に応じて、発信通信に使用できるIPサブネットを1つ以上指定します（これにより、BeeGFSが自動的に設定されます） `connNetFilterFile` オプション）：

```
filter_ip_ranges:
  - <SUBNET>/<MASK> # Ex. 192.168.10.0/24
```

5. 次のガイドラインに従って、このサービスがデータを保存するBeeGFS管理ターゲットを指定します。
   a. 複数のBeeGFSサービス/ターゲットに同じストレージプール名またはボリュームグループ名を使用できます。必ず同じ名前を使用してください `name`、 `raid_level`、 `criteria_*`および `common_*` それぞれの構成（サービスごとに表示されるボリュームは異なるはずです）。
   b. ボリュームサイズは、ストレージプール/ボリュームグループの割合として指定します。また、特定のストレージプール/ボリュームグループを使用するすべてのサービス/ボリュームで、合計サイズが100を超えないようにします。注SSDを使用する場合は、SSDのパフォーマンスと寿命を最大限にするために、ボリュームグループに空きスペースをいくらか残しておくことを推奨します（クリックして"こちらをご覧ください"詳細を確認）。
   c. をクリックします "こちらをご覧ください" で使用可能なすべての設定オプションのリストを表示するには、を参照してください `eseries_storage_pool_configuration`。などのオプションに注意してください `state`、 `host`、 `host_type`、 `workload_name`および `workload_metadata` ボリューム名は自動的に生成されるため、ここでは指定しないでください。

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: netapp_01
    eseries_storage_pool_configuration:
      - name: <NAME> # Ex: beegfs_m1_m2_m5_m6
        raid_level: <LEVEL> # One of: raid1, raid5, raid6, raidDiskPool
        criteria_drive_count: <DRIVE COUNT> # Ex. 4
        common_volume_configuration:
          segment_size_kb: <SEGMENT SIZE> # Ex. 128
        volumes:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
```

をクリックします "こちらをご覧ください" たとえば、BeeGFS管理サービスを表す完全なインベントリファイルの例を示します。

BeeGFSメタデータサービスを定義します

BeeGFSサービスは、グループ変数（group_vars）を使用して設定します。

概要

このセクションでは、BeeGFSメタデータサービスの定義手順を説明します。このタイプのサービスは、特定のファイルシステムのHAクラスタに少なくとも1つ存在する必要があります。このサービスには、次の定義が含まれます。

- サービスのタイプ（メタデータ）。

- このBeeGFSサービスにのみ適用する設定を定義します。
- このサービスに到達できる1つ以上のフローティングIP（論理インターフェイス）の設定。
- このサービス（BeeGFSメタデータターゲット）のデータを格納する場所と方法を指定します。

手順

"ファイルシステムを計画"セクションを参照し、 `group_vars/meta_<ID>.yml`クラスタ内の各メタデータサービスについてファイルをに作成し、次のように設定します。

1. BeeGFSメタデータサービスの設定を表すファイルを指定します。

```
beegfs_service: metadata
```

2. このBeeGFSサービスにのみ適用する設定を定義します。でサポートされる設定パラメータは、最小で目的のTCPポートとUDPポートを指定する必要があります `beegfs-meta.conf` このほか、次のパラメータは、自動的に設定されますが、ここでは指定しないでください。 `sysMgmtdHost`、 `storeMetaDirectory`、 `connAuthFile`、 `connDisableAuthentication`、 `connInterfacesFile`および `connNetFilterFile`。

```
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <TCP PORT>
  connMetaPortUDP: <UDP PORT>
  tuneBindToNumaZone: <NUMA ZONE> # Recommended if using file nodes with
multiple CPU sockets.
```

3. 他のサービスやクライアントがこのサービスへの接続に使用する1つまたは複数のフローティングIPを設定します（これにより、自動的にBeeGFSが設定されます） `connInterfacesFile` オプション）：

```
floating_ips:
  - <INTERFACE>:<IP/SUBNET> # Primary interface. Ex.
i1b:100.127.101.1/16
  - <INTERFACE>:<IP/SUBNET> # Secondary interface(s) as needed.
```

4. 必要に応じて、発信通信に使用できるIPサブネットを1つ以上指定します（これにより、BeeGFSが自動的に設定されます） `connNetFilterFile` オプション）：

```
filter_ip_ranges:
  - <SUBNET>/<MASK> # Ex. 192.168.10.0/24
```

5. 次のガイドラインに従って、このサービスがデータを格納するBeeGFSメタデータターゲットを指定します（これにより、が自動的に設定されます） `storeMetaDirectory` オプション）：

   a. 複数のBeeGFSサービス/ターゲットに同じストレージプール名またはボリュームグループ名を使用できます。必ず同じ名前を使用してください `name`、 `raid_level`、 `criteria_*`および

`common_*` それぞれの構成（サービスごとに表示されるボリュームは異なるはずです）。

b. ボリュームサイズは、ストレージプール/ボリュームグループの割合として指定します。また、特定のストレージプール/ボリュームグループを使用するすべてのサービス/ボリュームで、合計サイズが100を超えないようにします。注SSDを使用する場合は、SSDのパフォーマンスと寿命を最大限にするために、ボリュームグループに空きスペースをいくらか残しておくことを推奨します（クリックして"こちらをご覧ください"詳細を確認）。

c. をクリックします "こちらをご覧ください" で使用可能なすべての設定オプションのリストを表示するには、を参照してください `eseries_storage_pool_configuration`。などのオプションに注意してください `state`、 `host`、 `host_type`、 `workload_name`および `workload_metadata` ボリューム名は自動的に生成されるため、ここでは指定しないでください。

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: netapp_01
    eseries_storage_pool_configuration:
      - name: <NAME> # Ex: beegfs_m1_m2_m5_m6
        raid_level: <LEVEL> # One of: raid1, raid5, raid6, raidDiskPool
        criteria_drive_count: <DRIVE COUNT> # Ex. 4
        common_volume_configuration:
          segment_size_kb: <SEGMENT SIZE> # Ex. 128
        volumes:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
```

をクリックします "こちらをご覧ください" たとえば、BeeGFSメタデータサービスを表す完全なインベントリファイルの例を示します。

BeeGFSストレージサービスを定義します

BeeGFSサービスは、グループ変数（group_vars）を使用して設定します。

概要

このセクションでは、BeeGFSストレージサービスの定義手順を説明します。このタイプのサービスは、特定のファイルシステムのHAクラスタに少なくとも1つ存在する必要があります。このサービスには、次の定義が含まれます。

- サービスのタイプ（storage）。
- このBeeGFSサービスにのみ適用する設定を定義します。
- このサービスに到達できる1つ以上のフローティングIP（論理インターフェイス）の設定。
- このサービス（BeeGFSストレージターゲット）のデータを格納する場所/方法を指定します。

手順

"ファイルシステムを計画"セクションを参照して、 `group_vars/stor_<ID>.yml`クラスタ内の各ストレージサ

ービスのファイルをに作成し、次のように設定します。

1. BeeGFSストレージサービスの設定を表すファイルを指定します。

```
beegfs_service: storage
```

2. このBeeGFSサービスにのみ適用する設定を定義します。でサポートされる設定パラメータは、最小で目的のTCPポートとUDPポートを指定する必要があります `beegfs-storage.conf` このほか、次のパラメータは、自動的に設定されますが、ここでは指定しないでください。 `sysMgmtdHost`、 `storeStorageDirectory`、 `connAuthFile`、 `connDisableAuthentication`、 `connInterfacesFile`および `connNetFilterFile`。

```
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <TCP PORT>
  connStoragePortUDP: <UDP PORT>
  tuneBindToNumaZone: <NUMA ZONE> # Recommended if using file nodes with
multiple CPU sockets.
```

3. 他のサービスやクライアントがこのサービスへの接続に使用する1つまたは複数のフローティングIPを設定します（これにより、自動的にBeeGFSが設定されます） `connInterfacesFile` オプション）：

```
floating_ips:
  - <INTERFACE>:<IP/SUBNET> # Primary interface. Ex.
i1b:100.127.101.1/16
  - <INTERFACE>:<IP/SUBNET> # Secondary interface(s) as needed.
```

4. 必要に応じて、発信通信に使用できるIPサブネットを1つ以上指定します（これにより、BeeGFSが自動的に設定されます） `connNetFilterFile` オプション）：

```
filter_ip_ranges:
  - <SUBNET>/<MASK> # Ex. 192.168.10.0/24
```

5. 次のガイドラインに従って、このサービスがデータを保存するBeeGFSストレージターゲットを指定します（これにより、も自動的に設定されます） `storeStorageDirectory` オプション）：

   a. 複数のBeeGFSサービス/ターゲットに同じストレージプール名またはボリュームグループ名を使用できます。必ず同じ名前を使用してください `name`、 `raid_level`、 `criteria_*`および `common_*` それぞれの構成（サービスごとに表示されるボリュームは異なるはずです）。

   b. ボリュームサイズは、ストレージプール/ボリュームグループの割合として指定します。また、特定のストレージプール/ボリュームグループを使用するすべてのサービス/ボリュームで、合計サイズが100を超えないようにします。注SSDを使用する場合は、SSDのパフォーマンスと寿命を最大限にするために、ボリュームグループに空きスペースをいくらか残しておくことを推奨します（クリックして"こちらをご覧ください"詳細を確認）。

   c. をクリックします "こちらをご覧ください" で使用可能なすべての設定オプションのリストを表示する

には、を参照してください `eseries_storage_pool_configuration`。などのオプションに注意してください `state`、 `host`、 `host_type`、 `workload_name`および `workload_metadata` ボリューム名は自動的に生成されるため、ここでは指定しないでください。

```
beegfs_targets:
  <BLOCK_NODE>: # The name of the block node as found in the Ansible
inventory. Ex: netapp_01
    eseries_storage_pool_configuration:
      - name: <NAME> # Ex: beegfs_s1_s2
        raid_level: <LEVEL> # One of: raid1, raid5, raid6,
raidDiskPool
        criteria_drive_count: <DRIVE COUNT> # Ex. 4
        common_volume_configuration:
          segment_size_kb: <SEGMENT SIZE> # Ex. 128
        volumes:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
        # Multiple storage targets are supported / typical:
          - size: <PERCENT> # Percent of the pool or volume group to
allocate to this volume. Ex. 1
            owning_controller: <CONTROLLER> # One of: A, B
```

をクリックします "こちらをご覧ください" たとえば、BeeGFSストレージサービスを表す完全なインベントリファイルの例を示します。

BeeGFSサービスをファイルノードにマッピングします

を使用して、各BeeGFSサービスを実行できるファイルノードを指定します `inventory.yml` ファイル。

概要

このセクションでは、を作成する方法について説明します `inventory.yml` ファイル。これには、すべてのブロックノードのリストを表示し、各BeeGFSサービスを実行できるファイルノードを指定することも含まれます。

手順

ファイルを作成します `inventory.yml` 次のように入力します。

1. ファイルの上部から、標準のAnsibleインベントリ構造を作成します。

```
# BeeGFS HA (High_Availability) cluster inventory.
all:
  children:
```

2. このHAクラスタに含まれるすべてのブロックノードを含むグループを作成します。

```
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        <BLOCK NODE HOSTNAME>:
        <BLOCK NODE HOSTNAME>:
        # Additional block nodes as needed.
```

3. クラスタ内のすべてのBeeGFSサービスとそれらを実行するファイルノードを含むグループを作成します。

```
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
```

4. クラスタ内のBeeGFSサービスごとに、そのサービスを実行する優先ファイルノードとセカンダリファイルノードを定義します。

```
        <SERVICE>: # Ex. "mgmt", "meta_01", or "stor_01".
          hosts:
            <FILE NODE HOSTNAME>:
            <FILE NODE HOSTNAME>:
            # Additional file nodes as needed.
```

をクリックします "こちらをご覧ください" 完全なインベントリファイルの例を示します。

BeeGFSファイルシステムを導入します

Ansible Playbookの概要

Ansibleを使用したBeeGFS HAクラスタの導入と管理

概要

前のセクションでは、BeeGFS HAクラスタを表すAnsibleインベントリを構築するために必要な手順を説明しました。このセクションでは、ネットアップがクラスタの導入と管理を行うAnsibleによる自動化を紹介します。

Ansible：重要な概念

開始する前に、Ansibleの主要な概念を理解しておくと役立ちます。

- Ansibleインベントリに対して実行されるタスクは、* Playbook *と呼ばれるもので定義されています。
  - Ansibleのほとんどのタスクは*べき等値*となるように設計されているため、何度も実行して、必要な構成や状態が適用されていることを確認することができます。その際、作業を中断したり、不要な更新を加える必要はありません。
- Ansibleで実行される最小単位は*モジュール*です。
  - 一般的なプレイブックでは、複数のモジュールを使用
    - 例：パッケージのダウンロード、構成ファイルの更新、サービスの開始/有効化
  - NetApp Eシリーズシステムを自動化するために、モジュールを配布
- 複雑な自動化はロールとしてより適切にパッケージ化されています。
  - 基本的には、再利用可能なプレイブックを配布するための標準形式です。
  - LinuxホストとBeeGFSファイルシステムに役割を配布します。

AnsibleのBeeGFS HAロール：主な概念

ネットアップ上のBeeGFSの各バージョンの導入と管理に必要なすべての自動化機能がAnsibleのロールとしてパッケージ化され、の一部として提供されます "BeeGFSに対応したNetApp EシリーズAnsibleコレクション"：

- この役割は、BeeGFS用の*インストーラ*と最新の*導入/管理*エンジンの間にあると考えることができます。
  - コードの手法や理念として最新のインフラを活用し、あらゆる規模のストレージインフラをシンプルに管理できます。
  - この"久保スプレー"プロジェクトでは、スケールアウトコンピューティングインフラ向けにKubernetesディストリビューション全体を導入/保守できるようになります。
- この役割は、ネットアップのソリューションでBeeGFSをパッケージ化、配布、保守するためにネットアップが使用する*ソフトウェア定義*形式です。
  - Linuxディストリビューション全体や大きなイメージを配布することなく、「アプライアンスのような」エクスペリエンスを実現できるように努力してください。
  - カスタムのBeeGFSターゲットとIPアドレスに対応したネットアップがオーサリングしたOpen Cluster Framework（OCF）準拠のクラスタリソースエージェントで構成され、高度なPacemakerとBeeGFSを統合するための監視機能が提供されます。
- この役割は、単に導入を「自動化」するものではなく、以下を含むファイルシステムのライフサイクル全体を管理することを目的としています。
  - サービス単位またはクラスタ全体の設定変更および更新を適用する。
  - ハードウェアの問題が解決されたあとのクラスタの修復とリカバリの自動化
  - BeeGFSとネットアップのボリュームを使用した広範なテストに基づいてデフォルト値を設定することで、パフォーマンスの調整を簡易化
  - 構成のずれの検証と修正

ネットアップは、向けのAnsibleのロールも提供しています "BeeGFSクライアント"必要に応じて、BeeGFSの

インストールとファイルシステムのマウントを行い、/GPU/ログインノードを計算します。

BeeGFS HAクラスタを導入します

プレイブックを使用してBeeGFS HAクラスタを導入するために実行するタスクを指定します。

概要

このセクションでは、ネットアップでBeeGFSを導入/管理するために使用する標準的なプレイブックを組み立てる方法について説明します。

手順

Ansible Playbookを作成

ファイルを作成します `playbook.yml` 次のように入力します。

1. 最初に、一連のタスクを定義します（一般的には、と呼ばれます） "再生"）が実行されるのはNetApp Eシリーズのブロックノードだけです。インストールを実行する前に確認を求めて（誤ってプレイブックが実行されないように）、をインポートします `nar_santricity_management` ロール。このロールは、で定義されている一般的なシステム構成の適用を処理します `group_vars/eseries_storage_systems.yml` または個人 `host_vars/<BLOCK NODE>.yml` ファイル。

```
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  tasks:
    - name: Verify before proceeding.
      pause:
        prompt: "Are you ready to proceed with running the BeeGFS HA
role? Depending on the size of the deployment and network performance
between the Ansible control node and BeeGFS file and block nodes this
can take awhile (10+ minutes) to complete."
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

2. すべてのファイルノードおよびブロックノードに対して実行する再生を定義します。

```
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
```

3. このアプローチでは、必要に応じて、HAクラスタを導入する前に実行する一連の「事前タスク」を定義できます。これは、Pythonなどの前提条件を確認してインストールするのに役立ちます。また、提供されたAnsibleタグがサポートされていることを確認するなど、任意のプリフライトチェックを実行することもできます。

```
  pre_tasks:
    - name: Ensure a supported version of Python is available on all
file nodes.
      block:
        - name: Check if python is installed.
          failed_when: false
          changed_when: false
          raw: python --version
          register: python_version

        - name: Check if python3 is installed.
          raw: python3 --version
          failed_when: false
          changed_when: false
          register: python3_version
          when: 'python_version["rc"] != 0 or (python_version["stdout"]
| regex_replace("Python ", "")) is not version("3.0", ">=")'

        - name: Install python3 if needed.
          raw: |
            id=$(grep "^ID=" /etc/*release* | cut -d= -f 2 | tr -d '"')
            case $id in
              ubuntu) sudo apt install python3 ;;
              rhel|centos) sudo yum -y install python3 ;;
              sles) sudo zypper install python3 ;;
            esac
          args:
            executable: /bin/bash
          register: python3_install
          when: python_version['rc'] != 0 and python3_version['rc'] != 0
          become: true

        - name: Create a symbolic link to python from python3.
          raw: ln -s /usr/bin/python3 /usr/bin/python
```

```
          become: true
          when: python_version['rc'] != 0
      when: inventory_hostname not in
groups[beegfs_ha_ansible_storage_group]

    - name: Verify any provided tags are supported.
      fail:
        msg: "{{ item }} tag is not a supported BeeGFS HA tag. Rerun
your playbook command with --list-tags to see all valid playbook tags."
      when: 'item not in ["all", "storage", "beegfs_ha",
"beegfs_ha_package", "beegfs_ha_configure",
"beegfs_ha_configure_resource", "beegfs_ha_performance_tuning",
"beegfs_ha_backup", "beegfs_ha_client"]'
      loop: "{{ ansible_run_tags }}"
```

4. 最後に、導入するBeeGFSのバージョンに応じてBeeGFS HAロールをインポートします。

```
  tasks:
    - name: Verify the BeeGFS HA cluster is properly deployed.
      import_role:
        name: beegfs_ha_7_4 # Alternatively specify: beegfs_ha_7_3.
```

> BeeGFS HAロールは、サポートされるメジャーマイナーバージョンのBeeGFSごとに維持されます。これにより、ユーザはメジャー/マイナーバージョンをいつアップグレードするかを選択できます。現在、BeeGFS 7.3.x(`beegfs_7_3`）またはBeeGFS 7.2.x(`beegfs_7_2`）のいずれかがサポートされています。デフォルトでは、どちらのロールでも最新のBeeGFSパッチバージョンがリリース時に導入されますが、必要に応じてこれを上書きして最新のパッチを導入することもできます。"アップグレードガイド"詳細については、最新のを参照してください。

5. オプション：追加のタスクを定義する場合は、タスクの指示を考慮してください `all` ホスト（Eシリーズストレージシステムを含む）またはファイルノードのみ。必要に応じて、を使用して、ファイルノードを対象とした新しいプレイを定義します `- hosts: ha_cluster`。

をクリックします "こちらをご覧ください" に、完全なPlaybookファイルの例を示します。

NetApp Ansibleコレクションをインストールします

AnsibleのBeeGFSコレクションとすべての依存関係は維持されます "Ansible Galaxy"。Ansibleコントロールノードで次のコマンドを実行して最新バージョンをインストールします。

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

通常は推奨されませんが、コレクションの特定のバージョンをインストールすることもできます。

```
ansible-galaxy collection install netapp_eseries.beegfs:
==<MAJOR>.<MINOR>.<PATCH>
```

Playbookを実行してください

を含むAnsibleコントロールノードのディレクトリから `inventory.yml` および `playbook.yml` ファイルでは、次のようにプレイブックを実行します。

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

クラスタのサイズによっては、初期導入に20分以上かかることがあります。何らかの理由で導入が失敗した場合は、問題を修正し（ケーブルの接続ミス、ノードの起動など）、Ansibleプレイブックを再起動するだけです。

を指定するときに"共通ファイルノード構成"、接続ベースの認証をAnsibleで自動的に管理するデフォルトオプションを選択した場合、 `connAuthFile`共有シークレットとして使用されているが `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile`（デフォルト）に表示されるようになりました。ファイルシステムにアクセスする必要があるクライアントは、この共有シークレットを使用する必要があります。これは、クライアントがを使用して設定されている場合に自動的に処理され"BeeGFSクライアントの役割"ます。

BeeGFSクライアントを導入します

また、Ansibleを使用してBeeGFSクライアントを設定し、ファイルシステムをマウントすることもできます。

概要

BeeGFSファイルシステムにアクセスするには、ファイルシステムをマウントする必要のある各ノードにBeeGFSクライアントをインストールして設定する必要があります。このセクションでは、使用可能なを使用してこれらのタスクを実行する方法について説明します "Ansibleのロール"。

手順

クライアントインベントリファイルを作成します

1. 必要に応じて、Ansibleコントロールノードから、BeeGFSクライアントとして設定する各ホストにパスワードなしのSSHを設定します。

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

2. の下 `host_vars/`をクリックし、という名前のBeeGFSクライアントごとにファイルを作成します `<HOSTNAME>.yml` 次の内容を使用して、プレースホルダテキストに環境に適した情報を入力します。

```
# BeeGFS Client
ansible_host: <MANAGEMENT_IP>
```

3. NetApp Eシリーズホストコレクションのロールを使用して、クライアントがBeeGFSファイルノードに接続するためのInfiniBandインターフェイスまたはイーサネットインターフェイスを設定する場合は、オプションで次のいずれかを指定します。

   a. ネットワークタイプがの場合 "InfiniBand（IPoIBを使用）"：

   ```
   eseries_ipoib_interfaces:
   - name: <INTERFACE>  # Example: ib0 or i1b
     address: <IP/SUBNET> # Example: 100.127.100.1/16
   - name: <INTERFACE>  # Additional interfaces as needed.
     address: <IP/SUBNET>
   ```

   b. ネットワークタイプがの場合 "RDMA over Converged Ethernet（RoCE）"：

   ```
   eseries_roce_interfaces:
   - name: <INTERFACE>  # Example: eth0.
     address: <IP/SUBNET> # Example: 100.127.100.1/16
   - name: <INTERFACE>  # Additional interfaces as needed.
     address: <IP/SUBNET>
   ```

   c. ネットワークタイプがの場合 "イーサネット（TCPのみ、RDMAなし）"：

   ```
   eseries_ip_interfaces:
   - name: <INTERFACE>  # Example: eth0.
     address: <IP/SUBNET> # Example: 100.127.100.1/16
   - name: <INTERFACE>  # Additional interfaces as needed.
     address: <IP/SUBNET>
   ```

4. 新しいファイルを作成します `client_inventory.yml` さらに、Ansibleが各クライアントに接続するために使用するユーザを指定します。また、パスワードがAnsibleで権限の昇格（これにはが必要です `ansible_ssh_user` rootにするか、sudo権限を持っているか）：

```
# BeeGFS client inventory.
all:
  vars:
    ansible_ssh_user: <USER>
    ansible_become_password: <PASSWORD>
```

パスワードをプレーンテキストで保存しないでください。代わりにAnsible Vaultを使用します（を参照してください） "Ansibleのドキュメント" Ansible Vaultを使用してコンテンツを暗号化する場合）またはを使用します `--ask-become-pass` プレイブックを実行する際のオプション。

5. を参照してください `client_inventory.yml` ファイルに、の下でBeeGFSクライアントとして設定する必要があるすべてのホストをリストします `beegfs_clients` グループ化し、インラインコメントを参照して、BeeGFSクライアントカーネルモジュールをシステムに構築するために必要な追加設定のコメントを外します。

```
children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        <CLIENT HOSTNAME>:
        # Additional clients as needed.

      vars:
        # OPTION 1: If you're using the NVIDIA OFED drivers and they are
already installed:
        #eseries_ib_skip: True # Skip installing inbox drivers when
using the IPoIB role.
        #beegfs_client_ofed_enable: True
        #beegfs_client_ofed_include_path:
"/usr/src/ofa_kernel/default/include"

        # OPTION 2: If you're using inbox IB/RDMA drivers and they are
already installed:
        #eseries_ib_skip: True # Skip installing inbox drivers when
using the IPoIB role.

        # OPTION 3: If you want to use inbox IB/RDMA drivers and need
them installed/configured.
        #eseries_ib_skip: False # Default value.
        #beegfs_client_ofed_enable: False # Default value.
```

NVIDIA OFEDドライバを使用する場合は、beegfs_client_ofed_include_pathが、使用しているLinuxのインストールに適した「ヘッダーインクルードパス」を指定していることを確認してください。詳細については、BeeGFSのドキュメントを参照してください "RDMAのサポート"。

6. を参照してください `client_inventory.yml` ファイルで、以前に定義した任意の下にマウントするBeeGFSファイルシステムを一覧表示します `vars`：

```
        beegfs_client_mounts:
          - sysMgmtdHost: <IP ADDRESS>  # Primary IP of the BeeGFS
management service.
            mount_point: /mnt/beegfs    # Path to mount BeeGFS on the
client.
            connInterfaces:
              - <INTERFACE> # Example: ibs4f1
              - <INTERFACE>
            beegfs_client_config:
              # Maximum number of simultaneous connections to the same
node.
              connMaxInternodeNum: 128 # BeeGFS Client Default: 12
              # Allocates the number of buffers for transferring IO.
              connRDMABufNum: 36 # BeeGFS Client Default: 70
              # Size of each allocated RDMA buffer
              connRDMABufSize: 65536 # BeeGFS Client Default: 8192
              # Required when using the BeeGFS client with the shared-
disk HA solution.
              # This does require BeeGFS targets be mounted in the
default "sync" mode.
              # See the documentation included with the BeeGFS client
role for full details.
              sysSessionChecksEnabled: false
        # Specify additional file system mounts for this or other file
systems.
```

7. BeeGFS 7.2.7および7.3.1以降で"接続認証"は、設定または明示的に無効にする必要があります。を指定するときに接続ベースの認証を設定する方法によっては"共通ファイルノード構成"、クライアント設定の調整が必要になる場合があります。

   a. デフォルトでは、HAクラスタ環境で自動的に接続認証が設定され、が生成されます `connauthfile` に配置/管理されます `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`。デフォルトでは、BeeGFSクライアントの役割は、で定義したクライアントにこのファイルを読み取り/配布するように設定されています `client_inventory.yml`追加のアクションは必要ありません。

      i. 詳細オプションについては、に付属のすべてのデフォルト設定を参照してください "BeeGFSクライアントの役割"。

   b. でカスタムシークレットを指定する場合は、を使用します `beegfs_ha_conn_auth_secret` で指定します `client_inventory.yml` ファイルも同様：

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. で接続ベースの認証を完全に無効にする場合は、を使用します `beegfs_ha_conn_auth_enabled`` で、を指定します `client_inventory.yml` ファイルも同様：

```
beegfs_ha_conn_auth_enabled: false
```

サポートされるパラメータの一覧およびその他の詳細については、を参照してください "BeeGFSクライアントの完全なドキュメント"。クライアントインベントリの完全な例については、をクリックしてください "こちらをご覧ください"。

BeeGFS Client Playbookファイルを作成します

1. 新しいファイルを作成します `client_playbook.yml`

```
# BeeGFS client playbook.
- hosts: beegfs_clients
  any_errors_fatal: true
  gather_facts: true
  collections:
    - netapp_eseries.beegfs
    - netapp_eseries.host
  tasks:
```

2. オプション：NetApp Eシリーズホストコレクションのロールを使用して、クライアントがBeeGFSファイルシステムに接続するためのインターフェイスを設定する場合は、設定するインターフェイスタイプに対応するロールをインポートします。
   a. InfiniBand（IPoIB）を使用している場合は、次の手順を実行します。

```
    - name: Ensure IPoIB is configured
      import_role:
        name: ipoib
```

   b. を使用している環境でRDMA over Converged Ethernet（RoCE）を使用している場合：

```
    - name: Ensure IPoIB is configured
      import_role:
        name: roce
```

   c. 使用しているネットワークがイーサネット（TCPのみ、RDMAはなし）の場合：

```
    - name: Ensure IPoIB is configured
      import_role:
        name: ip
```

3. 最後に、BeeGFSクライアントの役割をインポートしてクライアントソフトウェアをインストールし、フ

ファイルシステムをマウントします。

```
    # REQUIRED: Install the BeeGFS client and mount the BeeGFS file
system.
    - name: Verify the BeeGFS clients are configured.
      import_role:
        name: beegfs_client
```

クライアントのプレイブックの完全な例については、をクリックしてください "こちらをご覧ください"。

BeeGFS Client Playbookを実行します

クライアントをインストール/ビルドしてBeeGFSをマウントするには、次のコマンドを実行します。

```
ansible-playbook -i client_inventory.yml client_playbook.yml
```

BeeGFSの導入を確認します

ファイルシステムを本番環境に導入する前に、ファイルシステムの導入を確認してください。

概要

BeeGFSファイルシステムを本番環境に移行する前に、いくつかの検証チェックを実行します。

手順

1. クライアントにログインして次のコマンドを実行し、想定されるすべてのノードが存在するか到達可能であり、不整合やその他の問題が報告されていないことを確認します。

```
beegfs-fsck --checkfs
```

2. クラスタ全体をシャットダウンし、再起動します。任意のファイルノードから、次のコマンドを実行します。

```
pcs cluster stop --all # Stop the cluster on all file nodes.
pcs cluster start --all  # Start the cluster on all file nodes.
pcs status # Verify all nodes and services are started and no failures
are reported (the command may need to be reran a few times to allow time
for all services to start).
```

3. 各ノードをスタンバイにし、BeeGFSサービスがセカンダリノードにフェイルオーバーできることを確認します。このログインを任意のファイルノードに行うには、次のコマンドを実行します。

```
pcs status # Verify the cluster is healthy at the start.
pcs node standby <FILE NODE HOSTNAME> # Place the node under test in
standby.
pcs status # Verify services are started on a secondary node and no
failures are reported.
pcs node unstandby <FILE NODE HOSTNAME> # Take the node under test out
of standby.
pcs status # Verify the file node is back online and no failures are
reported.
pcs resource relocate run # Move all services back to their preferred
nodes.
pcs status # Verify services have moved back to the preferred node.
```

4. IORやMDTestなどのパフォーマンスベンチマークツールを使用して、ファイルシステムのパフォーマンスが期待どおりであることを確認します。BeeGFSで使用される一般的なテストとパラメータの例については"設計検証"、「BeeGFS on NetApp Verified Architecture」を参照してください。

追加テストは、特定のサイト/設置環境に対して定義された受け入れ基準に基づいて実施する必要があります。