# 解決策 を導入します

## BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

# 目次

# 解決策 を導入します

## 導入の概要

NetApp上のBeeGFSは、NetAppのBeeGFSビルディングブロック設計とAnsibleを使用して、検証済みのファイルノードとブロックノードに導入できます。

### Ansibleのコレクションとロール

BeeGFS on NetAppソリューションは、アプリケーションの導入を自動化する一般的なIT自動化エンジンであるAnsibleを使用して導入されます。Ansibleでは、総称してインベントリと呼ばれる一連のファイルを使用して、導入するBeeGFSファイルシステムをモデル化します。

Ansibleを使用すると、NetAppなどの企業は、Ansible Galaxyで利用可能なコレクションを使用して、組み込み機能を拡張できます（を参照 "NetApp EシリーズBeeGFSのコレクション"）。コレクションには、特定の機能やタスク（Eシリーズボリュームの作成など）を実行するモジュールや、複数のモジュールやその他のロールを呼び出すことができるロールが含まれます。この自動化されたアプローチにより、BeeGFSファイルシステムと基盤となるHAクラスタの導入に要する時間が短縮されます。さらに、クラスタとBeeGFSファイルシステムの保守と拡張が容易になります。

詳細については、を参照してください "Ansibleのインベントリを確認できます"。

NetApp解決策 へのBeeGFSの導入には多数の手順が含まれるため、解決策 の手動による導入はサポートされません。

### BeeGFSビルディングブロックの構成プロファイル

導入手順では、次の設定プロファイルについて説明します。

- 管理、メタデータ、ストレージサービスを含む1つのベースとなるビルディングブロックです。
- メタデータとストレージサービスを含む2つ目のビルディングブロック。
- ストレージサービスのみを含む3つ目のビルディングブロック。

これらのプロファイルは、NetApp BeeGFSビルディングブロックに推奨されるすべての構成プロファイルを示しています。メタデータとストレージのビルディングブロックまたはストレージサービスのみのビルディングブロックの数は、環境ごとに容量とパフォーマンスの要件に応じて変わる場合があります。

### 導入手順の概要

の導入では、次の作業を実行します。

ハードウェアの導入

1. 各ビルディングブロックを物理的に組み立てます。
2. ラックに設置してケーブルを配線する。詳細な手順については、を参照してください "ハードウェアを導入"。

ソフトウェアの導入

1. "ファイルノードとブロックノードをセットアップ"。
   - ファイルノードにBMCのIPを設定します
   - 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、ファイルノードに管理ネットワークを設定します
   - ブロックノードに管理IPを設定します
2. "Ansibleコントロールノードをセットアップします"。
3. "パフォーマンスのシステム設定を調整します"。
4. "Ansibleインベントリを作成します"。
5. "BeeGFSビルディングブロックのAnsibleインベントリを定義します"。
6. "Ansibleを使用してBeeGFSを導入します"。
7. "BeeGFSクライアントを設定します"。

> 展開手順には、テキストをファイルにコピーする必要があるいくつかの例が含まれています。特定のデプロイメントに合わせて変更する必要がある、または変更できる内容については、「#」または「//」文字で示されるインラインコメントに注意してください。例:
>
> ```
> `beegfs_ha_ntp_server_pools:  # THIS IS AN EXAMPLE OF A COMMENT!
>   - "pool 0.pool.ntp.org iburst maxsources 3"
>   - "pool 1.pool.ntp.org iburst maxsources 3"`
> ```

導入に関する推奨事項にバリエーションを伴う派生アーキテクチャ：

- "大容量ビルディングブロック"

# Ansibleのインベントリを確認できます

導入を開始する前に、Ansibleがどのように設定され、BeeGFS on NetAppソリューションの導入に使用されるかについて理解しておいてください。

Ansibleインベントリは、導入するBeeGFSファイルシステムのファイルノードとブロックノードをリストしたディレクトリ構造です。これには、目的のBeeGFSファイルシステムを記述するホスト、グループ、および変数が含まれます。Ansibleインベントリは、Ansibleコントロールノードに格納する必要があります。コントロールノードとは、Ansibleプレイブックの実行に使用されるファイルノードとブロックノードにアクセスできる任意のマシンです。サンプルインベントリはからダウンロードできます "NetApp EシリーズBeeGFS GitHub"。

## Ansibleのモジュールとロール

Ansibleインベントリに記載されている構成を適用するには、エンドツーエンドのソリューションを導入するNetApp EシリーズAnsibleコレクション（から入手可能）に含まれているさまざまなAnsibleモジュールとロールを使用し "NetApp EシリーズBeeGFS GitHub"ます。

NetApp EシリーズAnsibleコレクションの各ロールは、NetApp解決策 上のBeeGFSをエンドツーエンドで導入します。これらのロールでは、NetApp E-Series SANtricity 、Host、およびBeeGFSの各コレクションを使用

して、HA（High Availability）を使用してBeeGFSファイルシステムを設定できます。その後、ストレージをプロビジョニングしてマッピングし、クラスタストレージを使用できる状態にします。

ロールには詳細なドキュメントが含まれていますが、導入手順では、第2世代のBeeGFSビルディングブロック設計を使用して、ロールを使用してNetApp Verified Architectureを導入する方法について説明します。

導入手順でAnsibleの使用経験が十分に細かい情報を提供できるようにすることは前提条件ではありませんが、Ansibleと関連する用語についてある程度理解している必要があります。

## BeeGFS HAクラスタのインベントリレイアウト

Ansibleのインベントリ構造を使用してBeeGFS HAクラスタを定義します。

Ansibleの使用経験がある方は、BeeGFS HAロールには、各ホストに適用される変数（ファクト）を検出するためのカスタムメソッドが実装されていることに注意してください。この設計により、Ansibleインベントリの構造化が簡素化され、複数のサーバで実行できるリソースが記述されます。

Ansibleインベントリは通常、およびのファイルと `inventory.yml`、 `group_vars`ホストを特定のグループ（場合によっては他のグループ）に割り当てるファイルで構成され `host_vars`ます。

このサブセクションの内容を含むファイルは、例としてのみ作成しないでください。

この構成は構成プロファイルに基づいて事前定義されていますが、以下に示すように、すべてがAnsibleインベントリとしてどのようにレイアウトされるかについて一般的に理解しておく必要があります。

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        netapp01:
        netapp02:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
        meta_01:  # Group representing a metadata service with ID 01.
          hosts:
            beegfs_01:  # This service is preferred on the first file
node.
            beegfs_02:  # And can failover to the second file node.
        meta_02:  # Group representing a metadata service with ID 02.
          hosts:
            beegfs_02:  # This service is preferred on the second file
node.
            beegfs_01: # And can failover to the first file node.
```

サービスごとに'構成を記述するgroup_varsの下に追加ファイルが作成されます

```
# meta_01 - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: 8015
  connMetaPortUDP: 8015
  tuneBindToNumaZone: 0
floating_ips:
  - i1b: <IP>/<SUBNET_MASK>
  - i2b: <IP>/<SUBNET_MASK>
# Type of BeeGFS service the HA resource group will manage.
beegfs_service: metadata # Choices: management, metadata, storage.
# What block node should be used to create a volume for this service:
beegfs_targets:
  netapp01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.25
            owning_controller: A
```

このレイアウトでは、各リソースのBeeGFSサービス、ネットワーク、ストレージの構成を1か所で定義できます。バックグラウンドでは、BeeGFSロールは、このインベントリ構造に基づいて各ファイルおよびブロックノードに必要な設定を集約します。

> 各サービスのBeeGFS数値と文字列ノードIDは、グループ名に基づいて自動的に設定されます。したがって、グループ名を一意にするための一般的なAnsibleの要件に加えて、BeeGFSサービスを表すグループは、グループが表すBeeGFSサービスのタイプに一意の番号で終わる必要があります。たとえば、meta_01とstor_01は許可されますが、meta_01とmeta_01は許可されません。

# ベストプラクティスを確認

NetApp解決策 にBeeGFSを導入する際は、ベストプラクティスのガイドラインに従ってください。

## 標準規則

Ansibleインベントリファイルを物理的に構築して作成する場合は、以下の標準的な規則に従ってください（詳細については、を参照してください） "Ansibleインベントリを作成します"）。

- ファイル・ノードのホスト名は'順番に番号が付けられ（H01-HN）'ラックの一番上に番号が小さい'下に数

字が大きい

たとえば、命名規則は `[location][row][rack]hN` 次のようになります。 `beegfs_01`

- 各ブロックノードは2台のストレージコントローラで構成され、コントローラごとに独自のホスト名が付けられます。

  ストレージアレイ名は、Ansibleインベントリの一部としてブロックストレージシステム全体を参照するために使用されます。ストレージアレイ名は順番に（A01-AN）番号を付け、各コントローラのホスト名はこの命名規則に基づいて付けます。

  たとえば、という名前のブロックノードは `ictad22a01` 通常、コントローラごとにホスト名を設定できます（やなど）があります `ictad22a01-a` `ictad22a01-b`が、Ansibleインベントリではと呼ばれます。 `netapp_01`

- 同じビルディングブロック内のファイルノードとブロックノードは、同じ番号方式を共有し、ラック内で互いに隣接しています。一番上に両方のファイルノードがあり、その下に両方のブロックノードが直接配置されています。

  たとえば、最初のビルディングブロックでは、ファイルノードH01とH02がどちらもブロックノードA01とA02に直接接続されています。ホスト名は、H01、H02、A01、およびA02です。

- ビルディングブロックは、ホスト名に基づいて順に配置されるため、番号の小さいホスト名はラックの上部に、番号の大きいホスト名は下部に配置されます。

  ここでは、ケーブルをラックスイッチの上部まで配線する時間を最小限に抑え、トラブルシューティングを簡単にするための標準的な導入方法を定義します。ラックの安定性に問題があるためにこれが許可されないデータセンターでは、逆の場合は、下から上にラックにデータを入力することができます。

## InfiniBandストレージネットワーク構成

各ファイルノードの半分のInfiniBandポートをブロックノードに直接接続します。残りの半分はInfiniBandスイッチに接続され、BeeGFSクライアント/サーバ接続に使用されます。BeeGFSクライアントおよびサーバに使用するIPoIBサブネットのサイズを決定する際には、コンピューティング/ GPUクラスタとBeeGFSファイルシステムの予想される増加を考慮する必要があります。推奨されるIP範囲から外れる必要がある場合は、1つのビルディングブロック内の各直接接続に一意のサブネットがあり、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ことに注意してください。

直接接続

各ビルディングブロック内のファイルノードとブロックノードは、常に次の表のIPを使用して直接接続されます。

> このアドレス指定方式は、次のルールに従っています。3番目のオクテットは常に奇数または偶数で、ファイルノードが奇数であるか偶数であるかによって異なります。

| 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 |
|---|---|---|---|---|---|---|
| 奇数（h1） | i1a | 192.168.1.10 | 奇数（c1） | 2A | 192.168.1.100 | 192.168.1.101 |
| 奇数（h1） | i2a | 192.168.3.10 | 奇数（c1） | 2A | 192.168.3.100 | 192.168.3.101 |

| 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 |
|---|---|---|---|---|---|---|
| 奇数（h1） | i3a | 192.168.5.10 | 偶数（C2） | 2A | 192.168.5.100 | 192.168.5.101 |
| 奇数（h1） | i4a | 192.168.7.10 | 偶数（C2） | 2A | 192.168.7.100 | 192.168.7.101 |
| 偶数（h2） | i1a | 192.168.2.10 | 奇数（c1） | 2B | 192.168.2.100 | 192.168.2.101 |
| 偶数（h2） | i2a | 192.168.4.10 | 奇数（c1） | 2B | 192.168.4.100 | 192.168.4.101 |
| 偶数（h2） | i3a | 192.168.6.10 | 偶数（C2） | 2B | 192.168.6.100 | 192.168.6.101 |
| 偶数（h2） | i4a | 192.168.8.10 | 偶数（C2） | 2B | 192.168.8.100 | 192.168.8.101 |

**BeeGFS**クライアント/サーバ**IPoIB**アドレス方式

各ファイルノードは、複数のBeeGFSサーバサービス（管理、メタデータ、またはストレージ）を実行します。各サービスを他のファイルノードに個別にフェイルオーバーできるようにするために、各サービスには両方のノード間で共有できる一意のIPアドレス（論理インターフェイスまたはLIFとも呼ばれます）が設定されます。

必須ではありませんが、この配置では、次のIPoIBサブネット範囲がこれらの接続に使用されていることが前提となり、次の規則を適用する標準アドレッシング方式を定義します。

- 2番目のオクテットは、ファイルノードのInfiniBandポートが奇数であるか偶数であるかに基づいて常に奇数になります。
- BeeGFSクラスタIPは常にxxxです。127.100.yyyまたは'xxx.128.100.yyy

> インバンドOS管理に使用されるインターフェイスに加えて、Corosyncでクラスタのハートビートおよび同期に追加のインターフェイスを使用できます。これにより、1つのインターフェイスが停止してもクラスタ全体が停止することはありません。

- BeeGFS管理サービスは'常に'xxx.yyy.101.0'または'xxx.yyy.102.0`になります
- BeeGFSメタデータ・サービスは常に「xxx.yyy.101.zzz」または「xxx.yyy.102.zzz」です。
- BeeGFSストレージサービスは、常に `xxx.yyy.103.zzz` または `xxx.yyy.104.zzz`です。
- アドレス範囲は'100.xxx.1.1'～'100.xxx.99.255'で'クライアント用に予約されています

**IPoIB**単一サブネットアドレッシング方式

この導入ガイドでは、に記載されている利点を考慮して、単一のサブネットスキーマを使用し "ソフトウェアアーキテクチャ"ます。

サブネット：**100.127.0.0/16**

次の表に、単一のサブネットの範囲（100.127.0.0/16）を示します。

| 目的 | InfiniBandポート | IPアドレスまたはIP範囲 |
|---|---|---|
| BeeGFS Cluster IP（BeeGFSクラスタIP） | i1bまたはi4b | 100.127.100.1-100.127.100.255 |

| 目的 | **InfiniBand**ポート | **IP**アドレスまたは**IP**範囲 |
| --- | --- | --- |
| BeeGFS Managementの略 | i1b | 100.127.101.0 |
| | i2b | 100.127.102.0 |
| BeeGFSメタデータ | i1bまたはi3b | 100.127.101.1-100.127.101.255 |
| | i2bまたはi4b | 100.127.102.1～100.127.102.255 |
| BeeGFS Storage（BeeGFSストレージ） | i1bまたはi3b | 100.127.103.1-100.127.103.255 |
| | i2bまたはi4b | 100.127.104.1～100.127.104.255 |
| BeeGFSクライアント | （クライアントによって異なる） | 100.127.1.1～100.127.99.255 |

#### IPoIB 2サブネットアドレッシング方式

2つのサブネットアドレッシング方式は推奨されなくなりましたが、実装は可能です。推奨される2つのサブネット方式については、次の表を参照してください。

サブネット**A**：**100.127.0.0.0/16**

次の表に、サブネットAの範囲を示します。100.127.0.0.0/16

| 目的 | **InfiniBand**ポート | **IP**アドレスまたは**IP**範囲 |
| --- | --- | --- |
| BeeGFS Cluster IP（BeeGFSクラスタIP） | i1b | 100.127.100.1-100.127.100.255 |
| BeeGFS Managementの略 | i1b | 100.127.101.0 |
| BeeGFSメタデータ | i1bまたはi3b | 100.127.101.1-100.127.101.255 |
| BeeGFS Storage（BeeGFSストレージ） | i1bまたはi3b | 100.127.103.1-100.127.103.255 |
| BeeGFSクライアント | （クライアントによって異なる） | 100.127.1.1～100.127.99.255 |

サブネット**B**：**100.128.0.0/16**

次の表に、サブネットBの範囲を示します。100.128.0.0/16

| 目的 | **InfiniBand**ポート | **IP**アドレスまたは**IP**範囲 |
| --- | --- | --- |
| BeeGFS Cluster IP（BeeGFSクラスタIP） | i4b | 100.128.100.1~100.128.100.255 |
| BeeGFS Managementの略 | i2b | 100.128.102.0 |
| BeeGFSメタデータ | i2bまたはi4b | 100.128.102.1～100.128.102.255 |
| BeeGFS Storage（BeeGFSストレージ） | i2bまたはi4b | 100.128.104.1～100.128.104.255 |
| BeeGFSクライアント | （クライアントによって異なる） | 100.128.1.1～100.128.99.255 |

このNetApp Verified Architectureでは、上記の範囲のすべてのIPが使用されているわけではありません。一貫したIPアドレッシング方式を使用してファイルシステムを簡単に拡張できるように、IPアドレスを事前に割り当てる方法を示します。この方式では、BeeGFSファイルノードとサービスIDは既知のIP範囲の4番目のオクテットに対応します。ファイルシステムは、必要に応じて255ノード以上のノードやサービスを拡張できます。

# ハードウェアを導入

## 各ビルディングブロックは、HDR（200GB）InfiniBandケーブルを使用して2つのブロックノードに直接接続された、検証済みの2つのx86ファイルノードで構成されます。

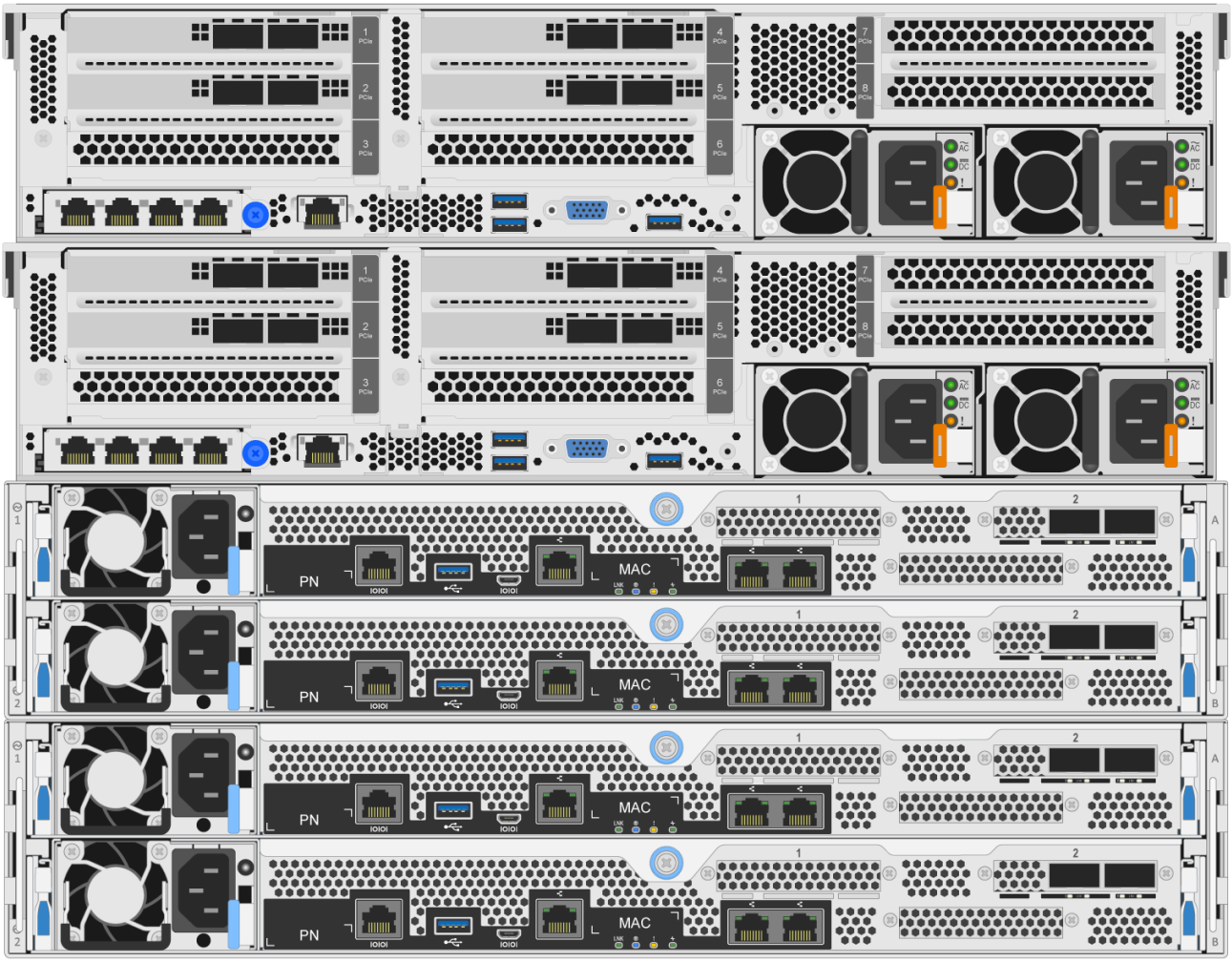
フェイルオーバークラスタでクォーラムを確立するには、少なくとも2つのビルディングブロックが必要です。2ノードクラスタには、フェイルオーバーの正常な実行を妨げる可能性がある制限があります。3つ目のデバイスをTiebreakerとして組み込むことで、2ノードクラスタを構成できますが、このドキュメントではその設計については説明していません。

次の手順は、特に記載がないかぎり、クラスタ内の各ビルディングブロックについて同じです。これは、このビルディングブロックを使用してBeeGFSメタデータサービスとストレージサービスの両方を実行するか、ストレージサービスだけを実行するかに関係ありません。

手順

1. で指定したモデルを使用して、4つのホストチャネルアダプタ（HCA）で各BeeGFSファイルノードをセットアップします。 "技術要件"以下の仕様に従って、HCAをファイルノードのPCIeスロットに挿入します。
   - * Lenovo ThinkSystem SR665 V3サーバー：PCIeスロット1、2、4、5を使用します。
   - * Lenovo ThinkSystem SR665サーバー：PCIeスロット2、3、5、6を使用します。
2. デュアルポートの200GBホストインターフェイスカード（HIC）で各BeeGFSブロックノードを設定し、2台の各ストレージコントローラにHICを取り付けます。

   2つのBeeGFSファイルノードがBeeGFSブロックノードの上になるようにビルディングブロックをラックに配置します。次の図は、Lenovo ThinkSystem SR665 V3サーバをファイルノードとして使用するBeeGFSビルディングブロックの正しいハードウェア構成を示しています（背面図）。

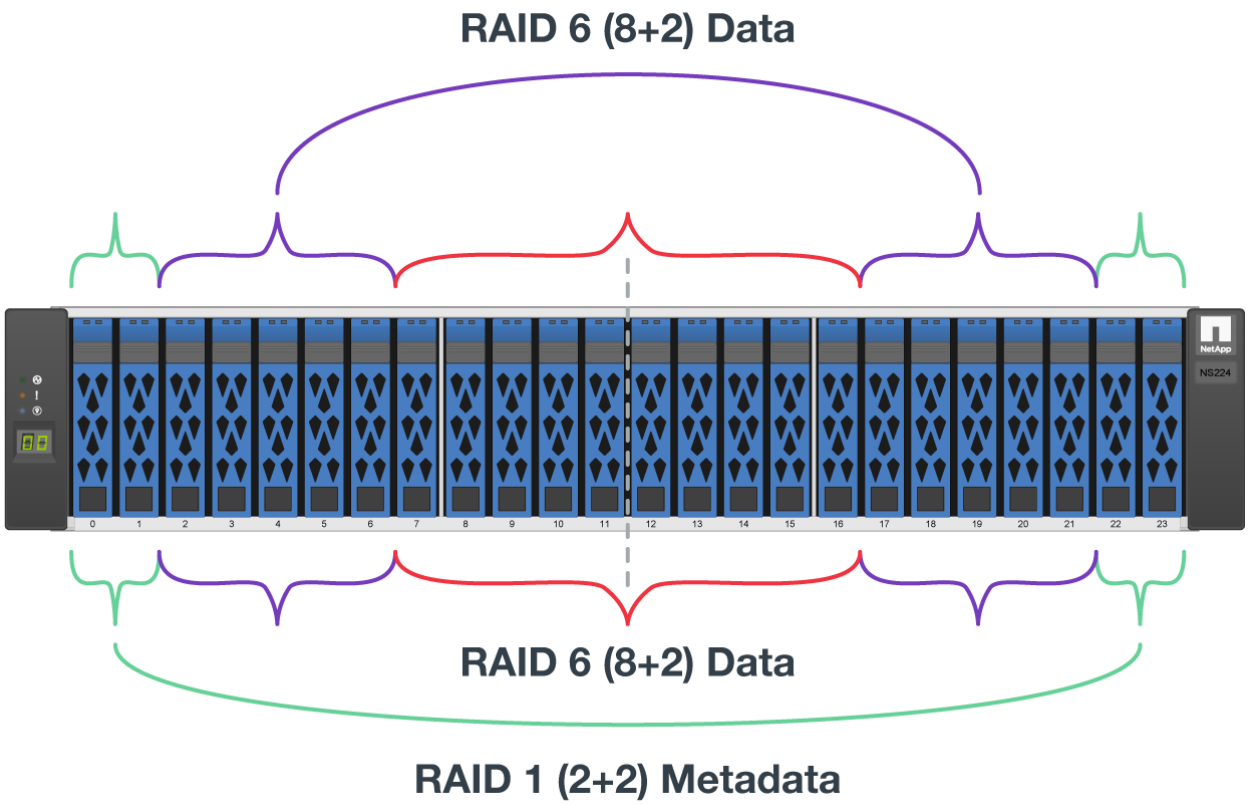



一般に、本番環境では電源装置を冗長PSUにする必要があります。

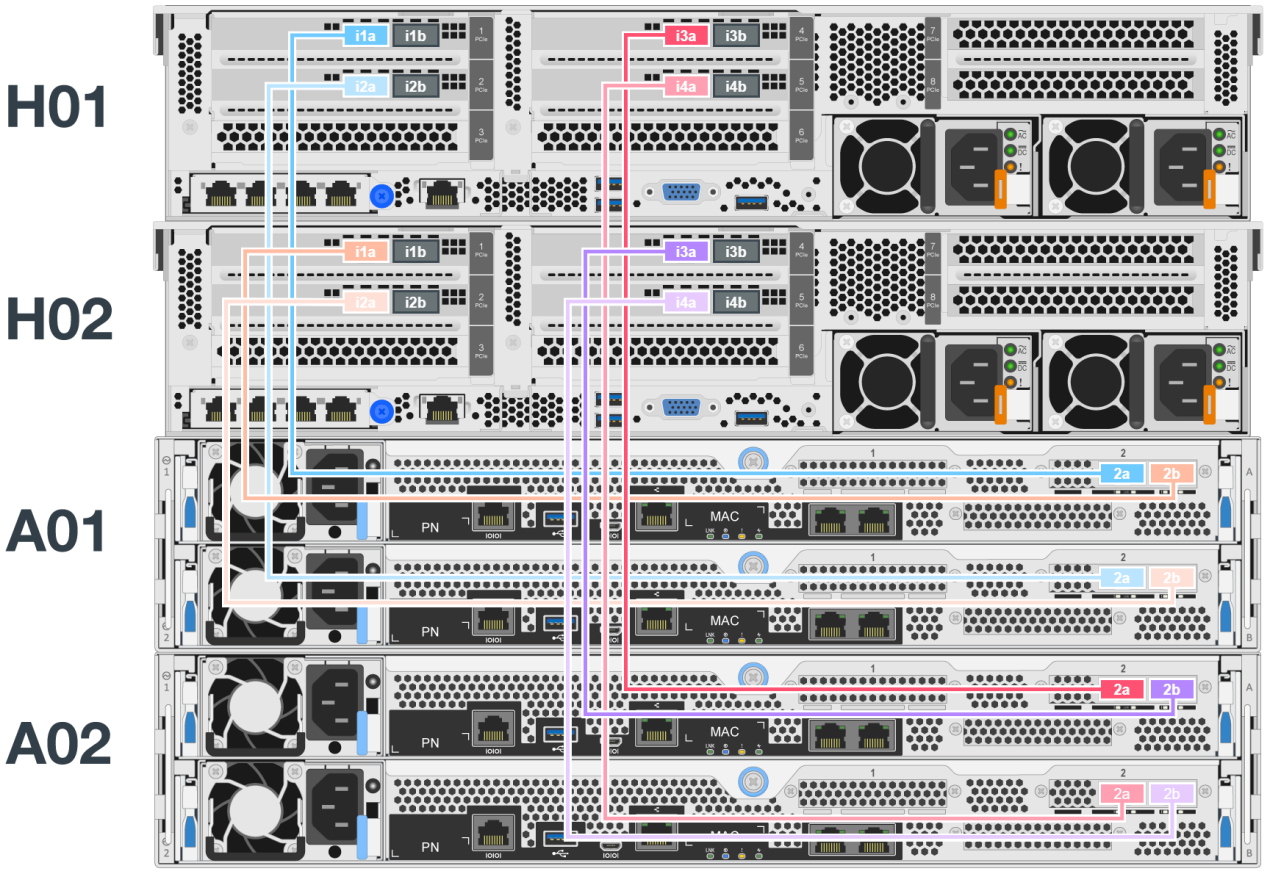
3. 必要に応じて、BeeGFSブロックノードのそれぞれにドライブを取り付けます。

   a. ビルディングブロックを使用してBeeGFSメタデータとストレージサービスを実行し、さらに小さいドライブをメタデータボリュームに使用する場合は、次の図に示すように、最も外側のドライブスロットにそれらが搭載されていることを確認します。

   b. すべてのビルディングブロック構成で、ドライブエンクロージャにフル装備されていない場合は、最適なパフォーマンスを得るために、同じ数のドライブがスロット0$_{11}$および$_{12}$23に装着されていることを確認してください。




4. 次の図に示すトポロジと一致するように、を使用してブロックノードとファイルノードを接続します "1M InfiniBand HDR 200GB直接接続銅ケーブル"。

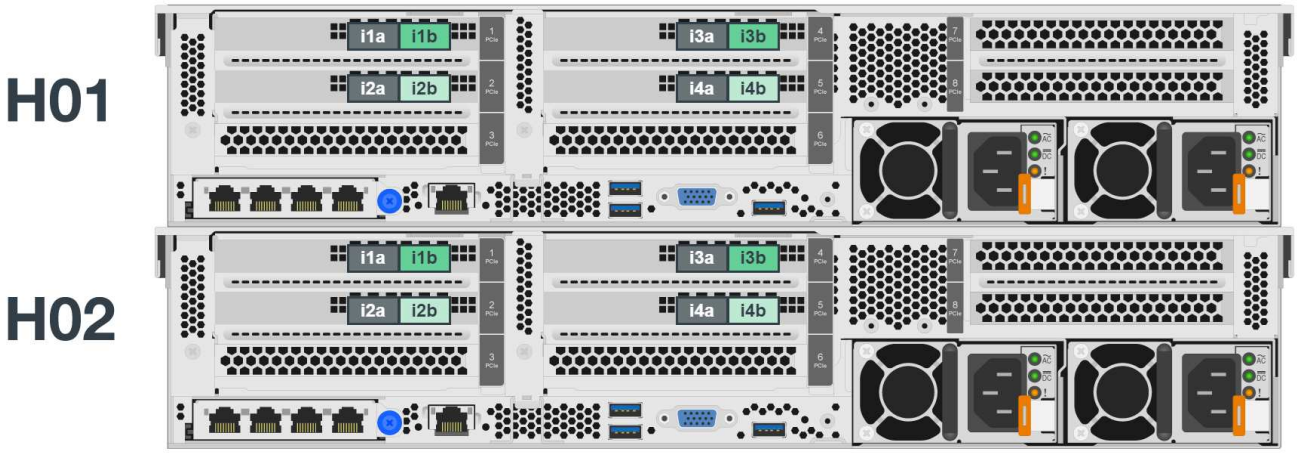


複数のビルディングブロックを横断するノードが直接接続されることはありません。各ビルディングブロックはスタンドアロンユニットとして扱われ、ビルディングブロック間のすべての通信はネットワークスイッチを介して行われます。

5. InfiniBandストレージスイッチに固有のを使用して、ファイルノードの残りのInfiniBandポートをストレージネットワークのInfiniBandスイッチに接続します "2M InfiniBandケーブル" 。

   スプリッタケーブルを使用してストレージスイッチとファイルノードを接続する場合は、スイッチから1本のケーブルが分岐し、ライトグリーンで示されているポートに接続する必要があります。別のスプリッタケーブルがスイッチから分岐し、濃い緑色で示されているポートに接続する必要があります。

   また、冗長スイッチを使用するストレージネットワークの場合、薄い緑色のポートは1つのスイッチに接続し、濃い緑色のポートは別のスイッチに接続します。




6. 必要に応じて、同じケーブル配線ガイドラインに従って追加のビルディングブロックをアセンブルします。

1台のラックに導入できるビルディングブロックの総数は、各サイトで利用可能な電力と冷却量によって異なります。

# ソフトウェアを導入

## ファイルノードとブロックノードをセットアップ

ほとんどのソフトウェア設定タスクはネットアップが提供するAnsibleコレクションを使用して自動化されていますが、各サーバのBaseboard Management Controller（BMC；ベースボード管理コントローラ）でネットワークを設定し、各コントローラの管理ポートを設定する必要があります。

### ファイルノードをセットアップします

1. 各サーバのベースボード管理コントローラ（BMC）のネットワークを設定します。

   検証済みLenovo SR665 V3ファイルノードのネットワーク設定方法については、を参照してください "Lenovo ThinkSystemのドキュメント"。

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。

2. 最大のパフォーマンスを得るためにシステムを設定します。

   システム設定は、UEFIセットアップ（旧BIOS）を使用するか、多くのBMCが提供するRedfish APIを使用して設定します。システム設定は、ファイルノードとして使用するサーバモデルによって異なります。

   検証済みのLenovo SR665 V3ファイルノードのシステム設定を構成する方法については、以下を参照してください。 "パフォーマンスのシステム設定を調整します" 。

3. Red Hat Enterprise Linux (RHEL) 9.4 をインストールし、Ansible コントロール ノードからの SSH 接続を含むオペレーティング システムの管理に使用するホスト名とネットワーク ポートを構成します。

   この時点では、InfiniBandポートにIPを設定しないでください。

   厳密には必須ではありませんが、以降のセクションでは、ホスト名には順番に番号が付けられ（h1-hNなど）、奇数ホストと偶数ホストで完了する必要のあるタスクを参照するようにしています。

4. Red Hat Subscription Manager を使用してシステムを登録およびサブスクライブし、公式 Red Hat リポジトリからの必要なパッケージのインストールを許可し、サポートされているバージョンの Red Hat への更新を制限します。 `subscription-manager release --set=9.4` 。手順については、およびを参照してください "RHELシステムを登録および登録する方法" "更新を制限する方法"。

5. 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
   ```

6. ガイドの使用"ファイルノードアダプタファームウェアの更新"で推奨されているバージョンにすべてのHCAファームウェアを更新します"テクノロジ要件"。

ブロックノードをセットアップする

各コントローラの管理ポートを設定してEF600ブロックノードをセットアップします。

1. 各EF600コントローラに管理ポートを設定します。

   ポートの設定手順については、を参照して "Eシリーズドキュメントセンター"ください。

2. 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。

   名前を設定すると、以降のセクションで各システムを簡単に参照できるようになります。アレイ名の設定手順については、を参照して "Eシリーズドキュメントセンター"ください。

必須ではありませんが、以降のトピックでは、ストレージアレイ名には必ず順番に番号を付け（c1-CNなど）、奇数のシステムでも偶数のシステムでも完了する必要がある手順を参照してください。

## パフォーマンスのファイルノードシステム設定を調整します

## パフォーマンスを最大化するには、ファイルノードとして使用するサーバーモデルでシステム設定を構成することをお勧めします。

システム設定は、ファイルノードとして使用するサーバーモデルによって異なります。この項では、検証済みLenovo ThinkSystem SR665サーバーファイルノードのシステム設定を構成する方法について説明します。

### **UEFI**インターフェイスを使用して、システム設定を調整します

Lenovo SR665 V3 サーバーのシステム ファームウェアには、UEFI インターフェイスを通じて設定できる多数のチューニング パラメーターが含まれています。これらのチューニングパラメータは、サーバの機能とサーバのパフォーマンスのすべての側面に影響を与える可能性があります。

UEFI Setup（UEFIセットアップ）> System Settings（システム設定）*で、次のシステム設定を調整します。

**Operating Mode**（操作モード）メニュー

| システム設定 | 「」に変更します |
|---|---|
| 動作モード | カスタム |
| CTDP | 手動 |
| CTDPマニュアル | 350 |
| パッケージの電力制限 | 手動 |
| 効率モード | 無効にします |
| GLOBAL-Cstate-Controlの略 | 無効にします |
| SOCの状態 | P0 |
| DF C -状態 | 無効にします |
| P状態 | 無効にします |
| Memory Power Down Enable (メモリの電源オフ有効 | 無効にします |
| ソケットごとのNUMAノード | NPS1 |

デバイスと**I/O**ポートのメニュー

| システム設定 | 「」に変更します |
| --- | --- |
| IOMMUを使用します | 無効にします |

電源メニュー

| システム設定 | 「」に変更します |
| --- | --- |
| PCIeパワーブレーキ | 無効にします |

**[**プロセッサ**]**メニュー

| システム設定 | 「」に変更します |
| --- | --- |
| グローバルCステートコントロール | 無効にします |
| DF C -状態 | 無効にします |
| SMTモード | 無効にします |
| CPPC | 無効にします |

**Redfish API**を使用して、システム設定を調整します

UEFIセットアップのほかに、Redfish APIを使用してシステム設定を変更することもできます。

```
curl --request PATCH \
  --url https://<BMC_IP_ADDRESS>/redfish/v1/Systems/1/Bios/Pending \
  --user <BMC_USER>:<BMC- PASSWORD> \
  --header 'Content-Type: application/json' \
  --data '{
"Attributes": {
"OperatingModes_ChooseOperatingMode": "CustomMode",
"Processors_cTDP": "Manual",
"Processors_PackagePowerLimit": "Manual",
"Power_EfficiencyMode": "Disable",
"Processors_GlobalC_stateControl": "Disable",
"Processors_SOCP_states": "P0",
"Processors_DFC_States": "Disable",
"Processors_P_State": "Disable",
"Memory_MemoryPowerDownEnable": "Disable",
"DevicesandIOPorts_IOMMU": "Disable",
"Power_PCIePowerBrake": "Disable",
"Processors_GlobalC_stateControl": "Disable",
"Processors_DFC_States": "Disable",
"Processors_SMTMode": "Disable",
"Processors_CPPC": "Disable",
"Memory_NUMANodesperSocket":"NPS1"
}
}
'
```

Redfishスキーマの詳細については、を参照してください "DMTFのWebサイト"。

## Ansibleコントロールノードをセットアップします

Ansible制御ノードをセットアップするには、BeeGFS on NetAppソリューション用に導入されたすべてのファイルノードとブロックノードにネットワークアクセスできる仮想マシンまたは物理マシンを指定する必要があります。

推奨されるパッケージバージョンのリストについては、を参照して"技術要件"ください。次の手順はUbuntu 22.04でテストされました。使用しているLinuxディストリビューションに固有の手順については、を参照してください "Ansibleのドキュメント"。

1. Ansibleコントロールノードから、次のPythonおよびPython仮想環境パッケージをインストールします。

```
sudo apt-get install python3 python3-pip python3-setuptools python3.10-venv
```

2. Python 仮想環境を作成します。

```
python3 -m venv ~/pyenv
```

3. 仮想環境をアクティブ化します。

```
source ~/pyenv/bin/activate
```

4. アクティブ化された仮想環境内に必要なPythonパッケージをインストールします。

```
pip install ansible netaddr cryptography passlib
```

5. Ansible Galaxyを使用してBeeGFSコレクションをインストールします。

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

6. インストールされているAnsible、Python、BeeGFSコレクションのバージョンがに一致することを確認します"技術要件"。

```
ansible --version
ansible-galaxy collection list netapp_eseries.beegfs
```

7. パスワードレスSSHを設定して、AnsibleがAnsibleコントロールノードからリモートBeeGFSファイルノードにアクセスできるようにします。
   a. Ansibleコントロールノードで、必要に応じて公開鍵のペアを生成します。

```
ssh-keygen
```

   b. 各ファイルノードへのパスワードレスSSHを設定します。

```
ssh-copy-id <ip_or_hostname>
```

ブロックノードにパスワードなしのSSHを設定しないでください。サポートされていません。

## Ansibleインベントリを作成します

ファイルノードとブロックノードの設定を定義するには、導入するBeeGFSファイルシステムを表すAnsibleインベントリを作成します。インベントリには、目的のBeeGFSファイルシステムを記述するホスト、グループ、および変数が含まれます。

## 手順1：すべてのビルディングブロックの構成を定義します

どの構成プロファイルを個別に適用できるかに関係なく、環境 のすべての構成ブロックを定義します。

### 作業を開始する前に

- 導入環境に適したサブネットアドレッシング方式を選択します。に記載されている利点のため、 "ソフトウェアアーキテクチャ"単一のサブネットアドレッシング方式を使用することを推奨します。

### 手順

1. Ansibleの制御ノードで、Ansibleのインベントリファイルとプレイブックファイルの格納に使用するディレクトリを特定します。

   特に記載がないかぎり、この手順および以降の手順で作成するすべてのファイルとディレクトリは、このディレクトリを基準にして作成されます。

2. 次のサブディレクトリを作成します。

   「host_vars」

   'group_vars'

   「パッケージ」

3. クラスタパスワード用のサブディレクトリを作成し、Ansible Vaultでファイルを暗号化して保護します（を参照） "Ansible Vaultを使用したコンテンツの暗号化"。

   a. サブディレクトリを作成し `group_vars/all`ます。

   b. `group_vars/all`ディレクトリに、という名前のパスワードファイルを作成し `passwords.yml`ます。

   c. 設定に応じて、すべてのユーザ名およびパスワードパラメータを置き換えて、に次の情報を入力 `passwords.yml file`します。

```
# Credentials for storage system's admin password
eseries_password: <PASSWORD>

# Credentials for BeeGFS file nodes
ssh_ha_user: <USERNAME>
ssh_ha_become_pass: <PASSWORD>

# Credentials for HA cluster
ha_cluster_username: <USERNAME>
ha_cluster_password: <PASSWORD>
ha_cluster_password_sha512_salt: randomSalt

# Credentials for fencing agents
# OPTION 1: If using APC Power Distribution Units (PDUs) for fencing:
# Credentials for APC PDUs.
apc_username: <USERNAME>
apc_password: <PASSWORD>

# OPTION 2: If using the Redfish APIs provided by the Lenovo XCC (and
other BMCs) for fencing:
# Credentials for XCC/BMC of BeeGFS file nodes
bmc_username: <USERNAME>
bmc_password: <PASSWORD>
```

d. プロンプトが表示されたら、を実行して `ansible-vault encrypt passwords.yml`ボルトパスワードを設定します。

手順**2**：個々のファイルノードとブロックノードの設定を定義する

環境 の個々のファイルノードおよび個々のビルディングブロックノードの構成を定義します。

1. 「host_vars/`」で、「<hostname>.yml」という名前のBeeGFSファイルノードごとに次の内容のファイルを作成します。BeeGFSクラスタのIPおよび奇数で終わるホスト名と偶数で終わるホスト名には、内容に関する注意を払って入力してください。

   最初は、ファイルノードのインターフェイス名が、ここに記載されている名前と一致しています（ib0やibs1f0など）。これらのカスタム名は、で設定します [手順4：すべてのファイルノードに適用する設定を定義する]。

```
ansible_host: "<MANAGEMENT_IP>"
eseries_ipoib_interfaces:  # Used to configure BeeGFS cluster IP
addresses.
  - name: i1b
    address: 100.127.100. <NUMBER_FROM_HOSTNAME>/16
  - name: i4b
    address: 100.127.100. <NUMBER_FROM_HOSTNAME>/16
beegfs_ha_cluster_node_ips:
  - <MANAGEMENT_IP>
  - <i1b_BEEGFS_CLUSTER_IP>
  - <i4b_BEEGFS_CLUSTER_IP>
# NVMe over InfiniBand storage communication protocol information
# For odd numbered file nodes (i.e., h01, h03, ..):
eseries_nvme_ib_interfaces:
  - name: i1a
    address: 192.168.1.10/24
    configure: true
  - name: i2a
    address: 192.168.3.10/24
    configure: true
  - name: i3a
    address: 192.168.5.10/24
    configure: true
  - name: i4a
    address: 192.168.7.10/24
    configure: true
# For even numbered file nodes (i.e., h02, h04, ..):
# NVMe over InfiniBand storage communication protocol information
eseries_nvme_ib_interfaces:
  - name: i1a
    address: 192.168.2.10/24
    configure: true
  - name: i2a
    address: 192.168.4.10/24
    configure: true
  - name: i3a
    address: 192.168.6.10/24
    configure: true
  - name: i4a
    address: 192.168.8.10/24
    configure: true
```

BeeGFSクラスタをすでに導入している場合は、静的に設定されたIPアドレス（NVMe/IBで使用するクラスタIPやIPなど）を追加または変更する前に、クラスタを停止する必要があります。これは、変更が適切に反映され、クラスタの処理が中断されないようにするために必要です。

2. 「host_vars/`」で、「<hostname>.yml」という名前のBeeGFSブロックノードごとにファイルを作成し、次の内容を入力します。

   ストレージアレイ名の末尾が奇数で偶数である場合は、内容に特に注意してください。

   ブロックノードごとに1つのファイルを作成し、2つのコントローラ（通常はA）のうちの1つに「<MANAGEMENT _IP>」を指定します。

```
eseries_system_name: <STORAGE_ARRAY_NAME>
eseries_system_api_url: https://<MANAGEMENT_IP>:8443/devmgr/v2/
eseries_initiator_protocol: nvme_ib
# For odd numbered block nodes (i.e., a01, a03, ..):
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.1.101
    - 192.168.2.101
    - 192.168.1.100
    - 192.168.2.100
  controller_b:
    - 192.168.3.101
    - 192.168.4.101
    - 192.168.3.100
    - 192.168.4.100
# For even numbered block nodes (i.e., a02, a04, ..):
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.5.101
    - 192.168.6.101
    - 192.168.5.100
    - 192.168.6.100
  controller_b:
    - 192.168.7.101
    - 192.168.8.101
    - 192.168.7.100
    - 192.168.8.100
```

手順**3**：すべてのファイルノードとブロックノードに適用する設定を定義する

グループに対応するファイル名に'GROLE_vars'の下にあるホストのグループに共通する構成を定義できますこれにより、複数の場所で共有設定を繰り返す必要がなくなります。

このタスクについて

ホストは複数のグループに含めることができ、実行時に、Ansibleは、変数の優先順位ルールに基づいて、特定のホストに適用する変数を選択します。（これらのルールの詳細については、Ansibleのドキュメントを参照してください "変数を使用します". ）

ホストとグループの割り当ては、実際のAnsibleインベントリファイルに定義されます。このファイルは、この手順 の末尾に作成されます。

ステップ

Ansibleでは、すべてのホストに適用する構成は「all」というグループで定義できます。次の内容で'ファイル'group_vars/all.yml'を作成します

```
ansible_python_interpreter: /usr/bin/python3
beegfs_ha_ntp_server_pools:  # Modify the NTP server addressess if
desired.
  - "pool 0.pool.ntp.org iburst maxsources 3"
  - "pool 1.pool.ntp.org iburst maxsources 3"
```

手順**4**：すべてのファイルノードに適用する設定を定義する

ファイル・ノードの共有構成は'ha_cluster'というグループで定義されますこのセクションの手順では'group_vars/ha_cluster.yml`ファイルに含める必要がある構成を構築します

手順

1. ファイルの最上部で'ファイルノードのsudoユーザーとして使用するパスワードを含むデフォルトを定義します

```
### ha_cluster Ansible group inventory file.
# Place all default/common variables for BeeGFS HA cluster resources
below.
### Cluster node defaults
ansible_ssh_user: {{ ssh_ha_user }}
ansible_become_password: {{ ssh_ha_become_pass }}
eseries_ipoib_default_hook_templates:
  - 99-multihoming.j2   # This is required for single subnet
deployments, where static IPs containing multiple IB ports are in the
same IPoIB subnet. i.e: cluster IPs, multirail, single subnet, etc.
# If the following options are specified, then Ansible will
automatically reboot nodes when necessary for changes to take effect:
eseries_common_allow_host_reboot: true
eseries_common_reboot_test_command: "! systemctl status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
eseries_ib_opensm_options:
  virt_enabled: "2"
  virt_max_ports_in_process: "0"
```

> がすでに存在する `root`場合は `ansible_ssh_user`、必要に応じてを省略し、Playbookの実行時にオプションを指定 `--ask-become-pass`できます `ansible_become_password`。

2. 必要に応じて、ハイアベイラビリティ（HA）クラスタの名前を設定し、クラスタ内通信用のユーザを指定します。

   プライベートIPアドレッシング方式を変更する場合は、デフォルトの「beegfs_ha_mgmtd_floating_ip」も更新する必要があります。これは、後でBeeGFS Managementリソースグループに設定する内容と一致している必要があります。

   「beegfs_alert_email_list」を使用して、クラスタ・イベントのアラートを受信する電子メールを1つ以上指定します。

```
### Cluster information
beegfs_ha_firewall_configure: True
eseries_beegfs_ha_disable_selinux: True
eseries_selinux_state: disabled
# The following variables should be adjusted depending on the desired
configuration:
beegfs_ha_cluster_name: hacluster                    # BeeGFS HA cluster
name.
beegfs_ha_cluster_username: "{{ ha_cluster_username }}" # Parameter for
BeeGFS HA cluster username in the passwords file.
beegfs_ha_cluster_password: "{{ ha_cluster_password }}" # Parameter for
BeeGFS HA cluster username's password in the passwords file.
beegfs_ha_cluster_password_sha512_salt: "{{
ha_cluster_password_sha512_salt }}" # Parameter for BeeGFS HA cluster
username's password salt in the passwords file.
beegfs_ha_mgmtd_floating_ip: 100.127.101.0          # BeeGFS management
service IP address.
# Email Alerts Configuration
beegfs_ha_enable_alerts: True
beegfs_ha_alert_email_list: ["email@example.com"]  # E-mail recipient
list for notifications when BeeGFS HA resources change or fail.  Often a
distribution list for the team responsible for managing the cluster.
beegfs_ha_alert_conf_ha_group_options:
      mydomain: “example.com”
# The mydomain parameter specifies the local internet domain name. This
is optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com).
# Adjusting the following parameters is optional:
beegfs_ha_alert_timestamp_format: "%Y-%m-%d %H:%M:%S.%N" #%H:%M:%S.%N
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
```

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。

3. フェンシングエージェントを設定します。（詳細については、を参照してください "Red Hatハイアベイラビリティクラスタでフェンシングを設定します"）。次の出力は、一般的なフェンシングエージェントの設定例を示しています。次のいずれかのオプションを選択します。

   この手順では、次の点に注意してください。

   ◦ フェンシングはデフォルトで有効になっていますが、フェンシングエージェント_を設定する必要があ

ります。

- 'pcmk_host_map'または'pcmk_host_listに指定されている`<hostname>は'Ansibleインベントリ内のホスト名に対応している必要があります
- フェンシングなしでBeeGFSクラスタを実行することは、特に本番環境ではサポートされません。これは、ブロックデバイスなどのリソース依存関係を含むBeeGFSサービスが問題 によってフェイルオーバーする際に、ファイルシステムの破損やその他の望ましくない動作や予期しない動作を引き起こす複数のノードによる同時アクセスのリスクがないことを主に保証するためです。フェンシングを無効にする必要がある場合は'BeeGFS HAロールの入門ガイドの一般的な注意事項を参照して'ha_cluster.ymlで'beegfs_cluster_crm_config_options[stonith -enabled "]をfalseに設定します
- 複数のノードレベルのフェンシングデバイスがあり、BeeGFS HAロールでは、Red Hat HAパッケージリポジトリで使用可能なフェンシングエージェントを設定できます。可能な場合は、無停電電源装置（UPS）またはラック配電装置（rPDU）を経由するフェンシングエージェントを使用します。 ベースボード管理コントローラ（BMC）などの一部のフェンシングエージェントや、サーバに組み込まれているその他のライトアウトデバイスは、特定の障害シナリオではフェンス要求に応答しない場合があります。

```
### Fencing configuration:
# OPTION 1: To enable fencing using APC Power Distribution Units
(PDUs):
beegfs_ha_fencing_agents:
 fence_apc:
   - ipaddr: <PDU_IP_ADDRESS>
     login: "{{ apc_username }}" # Parameter for APC PDU username in
the passwords file.
     passwd: "{{ apc_password }}" # Parameter for APC PDU password in
the passwords file.
     pcmk_host_map:
"<HOSTNAME>:<PDU_PORT>,<PDU_PORT>;<HOSTNAME>:<PDU_PORT>,<PDU_PORT>"
# OPTION 2: To enable fencing using the Redfish APIs provided by the
Lenovo XCC (and other BMCs):
redfish: &redfish
  username: "{{ bmc_username }}" # Parameter for XCC/BMC username in
the passwords file.
  password: "{{ bmc_password }}" # Parameter for XCC/BMC password in
the passwords file.
    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
# For details on configuring other fencing agents see
https://access.redhat.com/documentation/en-
us/red_hat_enterprise_linux/9/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

4. Linux OSで推奨されるパフォーマンス調整を有効にします。

   多くのユーザはパフォーマンスパラメータのデフォルト設定を確認できますが、特定のワークロードのデフォルト設定は必要に応じて変更できます。そのため、これらの推奨事項はBeeGFSロールに含まれますが、デフォルトでは有効になっていないため、ユーザーはファイルシステムに適用された調整を認識できません。

   パフォーマンス・チューニングを有効にするには’次のように指定

```
### Performance Configuration:
beegfs_ha_enable_performance_tuning: True
```

5. （オプション）Linux OSのパフォーマンス調整パラメータを必要に応じて調整できます。

   調整可能なチューニングパラメータの包括的なリストについては、のBeeGFS HAロールの「Performance Tuning Defaults」セクションを参照してください "EシリーズBeeGFS GitHubサイト"。 デフォルト値は、このファイルのクラスタ内のすべてのノードまたは個 々 のノードのファイルで上書きできます `host_vars` 。

6. ブロックノードとファイルノード間の200GB/HDR接続を完全に許可するには、NVIDIA Open Fabrics Enterprise Distribution（MLNX_OFED）のOpen Subnet Manager（OpenSM）パッケージを使用します。に記載されているMLNX_OFEDバージョンは "ファイルノードの要件" 、推奨されるOpenSMパッケージにバンドルされています。Ansibleを使用した導入もサポートされていますが、最初にすべてのファイルノードにMLNX_OFEDドライバをインストールする必要があります。

   a. 'group_vars/ha_cluster.yml'の次のパラメータを入力します(必要に応じてパッケージを調整します)

   ```
   ### OpenSM package and configuration information
   eseries_ib_opensm_options:
     virt_enabled: "2"
     virt_max_ports_in_process: "0"
   ```

7. 論理InfiniBandポート識別子と基盤となるPCIeデバイスとのマッピングが一貫して行われるように'udev'ルールを設定します

   udevルールは'BeeGFSファイル・ノードとして使用される各サーバ・プラットフォームのPCIeトポロジーに固有のものである必要があります

   検証済みファイルノードには、次の値を使用します。

```
### Ensure Consistent Logical IB Port Numbering
# OPTION 1: Lenovo SR665 V3 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:01:00.0": i1a
  "0000:01:00.1": i1b
  "0000:41:00.0": i2a
  "0000:41:00.1": i2b
  "0000:81:00.0": i3a
  "0000:81:00.1": i3b
  "0000:a1:00.0": i4a
  "0000:a1:00.1": i4b

# OPTION 2: Lenovo SR665 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:41:00.0": i1a
  "0000:41:00.1": i1b
  "0000:01:00.0": i2a
  "0000:01:00.1": i2b
  "0000:a1:00.0": i3a
  "0000:a1:00.1": i3b
  "0000:81:00.0": i4a
  "0000:81:00.1": i4b
```

8. （オプション）メタデータターゲット選択アルゴリズムを更新します。

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

検証テストでは'通常'randomrobinを使用して'パフォーマンス・ベンチマーク中にテスト・ファイルがすべてのBeeGFSストレージ・ターゲットに均等に分散されるようにしました（ベンチマークの詳細については'BeeGFSのサイトを参照してください "BeeGFSシステムのベンチマーク"）。実際に使用されている場合は、原因 の番号が小さいターゲットが、番号の大きいターゲットよりも早くいっぱいになる可能性があります。「randomrobin」を省略し、デフォルトの「randomized」値を使用するだけで、利用可能なすべてのターゲットを利用しながら、優れたパフォーマンスを提供できるようになりました。

手順**5**：共通ブロックノードの設定を定義する

ブロック・ノードの共有構成は'eseries_storage_systems'というグループで定義されますこのセクションの手順では'group_vars/eseries_storage_systems.yml`ファイルに含める必要がある構成を構築します

手順

1. Ansible接続をローカルに設定し、システムパスワードを指定して、SSL証明書を検証するかどうかを指定します。（通常、AnsibleはSSHを使用して管理対象ホストに接続しますが、NetApp Eシリーズストレージシステムがブロックノードとして使用されている場合、モジュールはREST APIを使用して通信します

）。 ファイルの上部に、次の情報を追加します。

```
### eseries_storage_systems Ansible group inventory file.
# Place all default/common variables for NetApp E-Series Storage Systems
here:
ansible_connection: local
eseries_system_password: {{ eseries_password }} # Parameter for E-Series
storage array password in the passwords file.
eseries_validate_certs: false
```

2. 最適なパフォーマンスを確保するには、に記載されているバージョンをブロックノードにインストールします "技術要件"。

   対応するファイルをからダウンロードします "ネットアップサポートサイト"。これらを手動でアップグレードするか'Ansibleコントロール・ノードのパッケージ/ディレクトリに含めてから'eseries_storage_systemesyml'に以下のパラメータを入力して'Ansibleを使用してアップグレードできます

```
# Firmware, NVSRAM, and Drive Firmware (modify the filenames as needed):
eseries_firmware_firmware: "packages/RCB_11.80GA_6000_64cc0ee3.dlp"
eseries_firmware_nvsram: "packages/N6000-880834-D08.dlp"
```

3. から、ブロックノードに取り付けられているドライブで使用可能な最新のドライブファームウェアをダウンロードしてインストールし "ネットアップサポートサイト"ます。手動でアップグレードするか、Ansible制御ノードのディレクトリに追加してから、の次のパラメータを入力してAnsibleを使用してアップグレードできます `packages/ eseries_storage_systems.yml` 。

```
eseries_drive_firmware_firmware_list:
  - "packages/<FILENAME>.dlp"
eseries_drive_firmware_upgrade_drives_online: true
```

> eseries_drive_firmware_upgrade_drivesonlineを'false'に設定すると'アップグレードが高速化されますが'BeeGFSが導入されるまでは実行しないでくださいこれは、アプリケーションエラーを回避するために、アップグレード前にドライブへのすべてのI/Oを停止する必要があるためです。ボリュームを構成する前にオンライン・ドライブ・ファームウェア・アップグレードを実行しても問題が発生しないようにするには'この値を常にtrueに設定することを推奨します

4. パフォーマンスを最適化するには、グローバル構成に対して次の変更を行います。

```
# Global Configuration Defaults
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.
```

5. ボリュームのプロビジョニングと動作を最適化するには、次のパラメータを指定します。

```
# Storage Provisioning Defaults
eseries_volume_size_unit: pct
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
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"
```

ⓘ 「eseries_storage_pool_usable_drives」に指定する値はNetApp EF600ブロックノードに固有であり、新しいボリュームグループにドライブを割り当てる順序を制御します。この順序により、各グループへのI/Oがバックエンドドライブチャネル間で均等に分散されます。

## BeeGFSビルディングブロックのAnsibleインベントリを定義します

### 一般的なAnsibleのインベントリ構造を定義したら、BeeGFSファイルシステムの各ビルディングブロックの設定を定義します。

導入手順では、管理、メタデータ、ストレージサービスなどの基本ビルディングブロックで構成されるファイルシステム、メタデータとストレージサービスを提供する2つ目のビルディングブロック、およびストレージ専用の3つ目のビルディングブロックで構成されるファイルシステムの導入方法を示します。

以下の手順は、BeeGFSファイルシステム全体の要件を満たすようにNetApp BeeGFSビルディングブロックを設定する際に使用する代表的な構成プロファイルをすべて示しています。

ⓘ このセクションと以降のセクションで、必要に応じて調整して、導入するBeeGFSファイルシステムを表すインベントリを作成します。特に、各ブロックまたはファイルノードを表すAnsibleホスト名と、ストレージネットワークに必要なIPアドレス指定方式を使用して、BeeGFSファイルノードとクライアントの数に合わせて拡張できます。

手順**1**：**Ansible**インベントリファイルを作成する

手順

1. 新しい'inventory.yml`ファイルを作成し'以下のパラメータを挿入します配置されたブロック・ノードを表すために'必要に応じて'eseries_storage_systems'の下のホストを置き換えますこれらの名前は'host_vars/<filename>.yml`に使用する名前に対応していなければなりません

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        netapp_01:
        netapp_02:
        netapp_03:
        netapp_04:
        netapp_05:
        netapp_06:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
```

以降のセクションでは、「ha_cluster」の下に、クラスタで実行するBeeGFSサービスを表すAnsibleグループを追加で作成します。

手順**2**：管理、メタデータ、ストレージのビルディングブロックのインベントリを設定する

クラスタまたはベースビルディングブロックの最初のビルディングブロックには、メタデータサービスおよびストレージサービスとともにBeeGFS管理サービスが含まれている必要があります。

手順

1. 'inventory.yml'で'ha_cluster: children'の下に次のパラメータを入力します

```
      # beegfs_01/beegfs_02 HA Pair (mgmt/meta/storage building block):
        mgmt:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_01:
          hosts:
            beegfs_01:
            beegfs_02:
        stor_01:
          hosts:
            beegfs_01:
```

```
    beegfs_02:
meta_02:
  hosts:
    beegfs_01:
    beegfs_02:
stor_02:
  hosts:
    beegfs_01:
    beegfs_02:
meta_03:
  hosts:
    beegfs_01:
    beegfs_02:
stor_03:
  hosts:
    beegfs_01:
    beegfs_02:
meta_04:
  hosts:
    beegfs_01:
    beegfs_02:
stor_04:
  hosts:
    beegfs_01:
    beegfs_02:
meta_05:
  hosts:
    beegfs_02:
    beegfs_01:
stor_05:
  hosts:
    beegfs_02:
    beegfs_01:
meta_06:
  hosts:
    beegfs_02:
    beegfs_01:
stor_06:
  hosts:
    beegfs_02:
    beegfs_01:
meta_07:
  hosts:
    beegfs_02:
    beegfs_01:
stor_07:
```

```
          hosts:
            beegfs_02:
            beegfs_01:
        meta_08:
          hosts:
            beegfs_02:
            beegfs_01:
        stor_08:
          hosts:
            beegfs_02:
            beegfs_01:
```

2. ファイル'group_vars/mgmt.yml'を作成し'以下を含めます

```
# mgmt - BeeGFS HA Management Resource Group
# OPTIONAL: Override default BeeGFS management configuration:
# beegfs_ha_beegfs_mgmtd_conf_resource_group_options:
#  <beegfs-mgmt.conf:key>:<beegfs-mgmt.conf:value>
floating_ips:
  - i1b: 100.127.101.0/16
  - i2b: 100.127.102.0/16
beegfs_service: management
beegfs_targets:
  netapp_01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 1
            owning_controller: A
```

3. 「group_vars/`」の下で、次のテンプレートを使用して「meta_01」から「meta_08」までのリソースグループのファイルを作成し、以下の表を参照する各サービスのプレースホルダ値を入力します。

```
# meta_0X - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <PORT>
  connMetaPortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET> # Example: i1b:192.168.120.1/16
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: metadata
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 21.25 # SEE NOTE BELOW!
            owning_controller: <OWNING CONTROLLER>
```

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とも呼ばれる）全体に対する割合で指定します。SSDのオーバープロビジョニングのためのスペースを確保するために、各プールにある程度の空き容量を確保することを強く推奨します（詳細については、を参照してください） "NetApp EF600アレイの概要"）。ストレージプール'beegfs_m1_m2_m3_m6"は'管理サービス用のプールの容量の1%も割り当てますしたがって'ストレージ・プール内のメタデータ・ボリュームでは'beegfs_m1_m2_m5_m6'1.92TBまたは3.84TBのドライブを使用している場合'この値を21.25'7.65TBドライブの場合は22.25'15.3TBドライブの場合は'23.75'に設定します

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| meta_01.yml | 8015 | i1b：100.127.101.1/16 i2b：100.127.102.1 /16 | 0 | netapp_01 | beegfs_m1_m2_m5_m6 | A |
| meta_02.yml | 8025 | i2b：100.127.102.2/16 i1b：100.127.101.2 /16 | 0 | netapp_01 | beegfs_m1_m2_m5_m6 | B |

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| meta_03.yml | 8035 | i3b：100.127.101.3/16 i4b：100.127.102.3 /16 | 1. | netapp_02 | beegfs_m3_m4_m7_M8 | A |
| meta_04.yml | 8045 | i4b：100.127.102.4/16 i3b：100.127.101.4 /16 | 1. | netapp_02 | beegfs_m3_m4_m7_M8 | B |
| meta_05.yml | 8055 | i1b：100.127.101.5/16 i2b：100.127.102.5 /16 | 0 | netapp_01 | beegfs_m1_m2_m5_m6 | A |
| meta_06.yml | 8065 | i2b：100.127.102.6/16 i1b：100.127.101.6 /16 | 0 | netapp_01 | beegfs_m1_m2_m5_m6 | B |
| meta_07.yml | 8075 | i3b：100.127.101.7/16 i4b：100.127.102.7 /16 | 1. | netapp_02 | beegfs_m3_m4_m7_M8 | A |
| meta_08.yml | 8085 | i4b：100.127.102.8/16 i3b：100.127.101.8 /16 | 1. | netapp_02 | beegfs_m3_m4_m7_M8 | B |

4. 「group_vars/`」の下で、以下のテンプレートを使用して「stor_01」から「stor_08」のリソースグループ用のファイルを作成し、例を参照する各サービスのプレースホルダ値を入力します。

```
# stor_0X - BeeGFS HA Storage Resource
Groupbeegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 10
        common_volume_configuration:
          segment_size_kb: 512        volumes:
          - size: 21.50 # See note below!             owning_controller:
<OWNING CONTROLLER>
          - size: 21.50            owning_controller: <OWNING
CONTROLLER>
```

正しいサイズについては、を参照してください "ストレージプールのオーバープロビジョニングの割合を推奨します"。

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| STOR_01.yml | 8013 | i1b：100.127.103.1/16 i2b：100.127.104.1 /16 | 0 | netapp_01 | beegfs_s1_s2 | A |
| STOR_02.yml | 8023 | i2b：100.127.104.2/16 i1b：100.127.103.2 /16 | 0 | netapp_01 | beegfs_s1_s2 | B |
| STOR_03.yml | 8033 | i3b：100.127.103.3/16 i4b：100.127.104.3 /16 | 1. | netapp_02 | beegfs_s3_s4 | A |

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| STOR_04.yml | 8043 | i4b：100.127.104.4/16 i3b：100.127.103.4 /16 | 1. | netapp_02 | beegfs_s3_s4 | B |
| STOR_05.yml | 8053 | i1b：100.127.103.5/16 i2b：100.127.104.5 /16 | 0 | netapp_01 | beegfs_s5_s6 | A |
| STOR_06.yml | 8063 | i2b：100.127.104.6/16 i1b：100.127.103.6 /16 | 0 | netapp_01 | beegfs_s5_s6 | B |
| STOR_07.yml | 8073 | i3b：100.127.103.7/16 i4b：100.127.104.7 /16 | 1. | netapp_02 | beegfs_s7_s8 | A |
| STOR_08.yml | 8083 | i4b：100.127.104.8/16 i3b：100.127.103.8 /16 | 1. | netapp_02 | beegfs_s7_s8 | B |

手順**3**：メタデータとストレージのビルディングブロックのインベントリを設定する

以下の手順では、BeeGFSメタデータとストレージビルディングブロックにAnsibleインベントリを設定する方法について説明します。

手順

1. 'inventory.yml'で'既存の構成の下に次のパラメータを入力します

```
        meta_09:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_09:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_10:
          hosts:
            beegfs_03:
```

```
        beegfs_04:
    stor_10:
      hosts:
        beegfs_03:
        beegfs_04:
    meta_11:
      hosts:
        beegfs_03:
        beegfs_04:
    stor_11:
      hosts:
        beegfs_03:
        beegfs_04:
    meta_12:
      hosts:
        beegfs_03:
        beegfs_04:
    stor_12:
      hosts:
        beegfs_03:
        beegfs_04:
    meta_13:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_13:
      hosts:
        beegfs_04:
        beegfs_03:
    meta_14:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_14:
      hosts:
        beegfs_04:
        beegfs_03:
    meta_15:
      hosts:
        beegfs_04:
        beegfs_03:
    stor_15:
      hosts:
        beegfs_04:
        beegfs_03:
    meta_16:
```

```
          hosts:
            beegfs_04:
            beegfs_03:
        stor_16:
          hosts:
            beegfs_04:
            beegfs_03:
```

2. 「group_vars/`」の下で、次のテンプレートを使用して「meta_09」から「meta_16」までのリソースグループのファイルを作成し、例を参照する各サービスのプレースホルダ値を入力します。

```
# meta_0X - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <PORT>
  connMetaPortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: metadata
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.5 # SEE NOTE BELOW!
            owning_controller: <OWNING CONTROLLER>
```

正しいサイズについては、を参照してください "ストレージプールのオーバープロビジョニングの割合を推奨します"。

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| meta_09.yml | 8015 | i1b：100.127.101.9/16 i2b：100.127.102.9 /16 | 0 | netapp_03 | beegfs_m9_m10_m13_M14 | A |

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| meta_10.yml | 8025 | i2b：100.127.102.10/16 i1b：100.127.101.10 /16 | 0 | netapp_03 | beegfs_m9_m10_m13_M14 | B |
| meta_11.yml | 8035 | i3b：100.127.101.11/16 i4b：100.127.102.11 /16 | 1. | netapp_04 | BEegfs_M11_M12_M15_M16 | A |
| meta_12.yml | 8045 | i4b：100.127.102.12/16 i3b：100.127.101.12 /16 | 1. | netapp_04 | BEegfs_M11_M12_M15_M16 | B |
| meta_13.yml | 8055 | i1b：100.127.101.13/16 i2b：100.127.102.13 /16 | 0 | netapp_03 | beegfs_m9_m10_m13_M14 | A |
| meta_14.yml | 8065 | i2b：100.127.102.14/16 i1b：100.127.101.14 /16 | 0 | netapp_03 | beegfs_m9_m10_m13_M14 | B |
| meta_15.yml | 8075 | i3b：100.127.101.15/16 i4b：100.127.102.15 /16 | 1. | netapp_04 | BEegfs_M11_M12_M15_M16 | A |
| meta_16.yml | 8085 | i4b：100.127.102.16/16 i3b：100.127.101.16 /16 | 1. | netapp_04 | BEegfs_M11_M12_M15_M16 | B |

3. 「group_vars/」の下で、「stor_09」から「stor_16」までのリソースグループ用のファイルを作成し、例を参照する各サービスのプレースホルダ値を入力します。

```
# stor_0X - BeeGFS HA Storage Resource Group
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 10
        common_volume_configuration:
          segment_size_kb: 512        volumes:
          - size: 21.50 # See note below!
            owning_controller: <OWNING CONTROLLER>
          - size: 21.50            owning_controller: <OWNING
CONTROLLER>
```

適切なサイズについては、 "ストレージプールのオーバープロビジョニングの割合を推奨します" ..

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| STOR_09.yml | 8013 | i1b：100.127.103.9/16 i2b：100.127.104.9 /16 | 0 | netapp_03 | beegfs_s9_s10 | A |
| STOR_10.yml | 8023 | i2b：100.127.104.10/16 i1b：100.127.103.10 /16 | 0 | netapp_03 | beegfs_s9_s10 | B |
| STOR_11.yml | 8033 | i3b：100.127.103.11/16 i4b：100.127.104.11 /16 | 1. | netapp_04 | beegfs_s11_s12を指定します | A |

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| STOR_12.yml | 8043 | i4b：100.127.104.12/16 i3b：100.127.103.12 /16 | 1. | netapp_04 | beegfs_s11_s12を指定します | B |
| STOR_13.yml | 8053 | i1b：100.127.103.13/16 i2b：100.127.104.13 /16 | 0 | netapp_03 | beegfs_S13_s14 | A |
| STOR_14.yml | 8063 | i2b：100.127.104.14/16 i1b：100.127.103.14 /16 | 0 | netapp_03 | beegfs_S13_s14 | B |
| STOR_15.yml | 8073 | i3b：100.127.103.15/16 i4b：100.127.104.15 /16 | 1. | netapp_04 | beegfs_s15_s16 | A |
| STOR_16.yml | 8083 | i4b：100.127.104.16/16 i3b：100.127.103.16 /16 | 1. | netapp_04 | beegfs_s15_s16 | B |

手順**4**：ストレージ専用のビルディングブロックのインベントリを設定する

以下の手順では、BeeGFSストレージ専用ビルディングブロックのAnsibleインベントリを設定する方法について説明します。メタデータとストレージのみのビルディング・ブロックの構成を設定する場合の主な違いは'すべてのメタデータ・リソース・グループを省略し'各ストレージ・プールの基準ドライブ数を10から12に変更することです

手順

1. 'inventory.yml'で'既存の構成の下に次のパラメータを入力します

```
# beegfs_05/beegfs_06 HA Pair (storage only building block):
  stor_17:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_18:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_19:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_20:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_21:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_22:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_23:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_24:
    hosts:
      beegfs_06:
      beegfs_05:
```

2. 「group_vars/」の下で、以下のテンプレートを使用して「`stor_17~`stor_24`のリソースグループのファイルを作成し、例を参照する各サービスのプレースホルダ値を入力します。

```
# stor_0X - BeeGFS HA Storage Resource Group
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 12
        common_volume_configuration:
          segment_size_kb: 512
        volumes:
          - size: 21.50 # See note below!
            owning_controller: <OWNING CONTROLLER>
          - size: 21.50
            owning_controller: <OWNING CONTROLLER>
```

適切なサイズについては、 "ストレージプールのオーバープロビジョニングの割合を推奨します"。

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| STOR_17.yml | 8013 | i1b：100.127.103.17/16 i2b：100.127.104.17 /16 | 0 | netapp_05 | beegfs_s17_s18 | A |
| STOR_18.yml | 8023 | i2b：100.127.104.18/16 i1b：100.127.103.18 /16 | 0 | netapp_05 | beegfs_s17_s18 | B |
| STOR_19.yml | 8033 | i3b：100.127.103.19/16 i4b：100.127.104.19 /16 | 1. | netapp_06 | beegfs_s19_s20 | A |

| ファイル名 | 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 |
|---|---|---|---|---|---|---|
| STOR_20.yml | 8043 | i4b：100.127.104.20/16 i3b：100.127.103.20 /16 | 1. | netapp_06 | beegfs_s19_s20 | B |
| STOR_21.yml | 8053 | i1b：100.127.103.21/16 i2b：100.127.104.21 /16 | 0 | netapp_05 | beegfs_S21_s22 | A |
| STOR_22.yml | 8063 | i2b：100.127.104.22/16 i1b：100.127.103.22 /16 | 0 | netapp_05 | beegfs_S21_s22 | B |
| STOR_23.yml | 8073 | i3b：100.127.103.23/16 i4b：100.127.104.23 /16 | 1. | netapp_06 | beegfs_S23_s24 | A |
| STOR_24.yml | 8083 | i4b：100.127.104.24/16 i3b：100.127.103.24 /16 | 1. | netapp_06 | beegfs_S23_s24 | B |

## BeeGFSを導入します

構成の導入と管理には、Ansibleで実行するタスクが含まれた1つ以上のプレイブックを実行し、システム全体を目的の状態にする必要があります。

すべてのタスクを1つのプレイブックに含めることができますが、複雑なシステムでは、この作業が管理しにくくなります。Ansibleを使用すると、再利用可能なプレイブックと関連コンテンツ（デフォルトの変数、タスク、ハンドラなど）をパッケージ化する方法でロールを作成して配布できます。詳細については、Ansibleのドキュメントを参照してください "ロール"。

多くの場合、ロールは関連するロールとモジュールを含むAnsibleコレクションの一部として配布されます。このため、このプレイブックは、主に、NetApp Eシリーズの各種Ansibleコレクションに分散された複数のロールをインポートするだけです。

> ⓘ 現在、2ノードクラスタとのクォーラムの確立時に問題が発生しないように、別のクォーラムデバイスをTiebreakerとして設定している場合を除き、BeeGFSを導入するには少なくとも2つのビルディングブロック（4つのファイルノード）が必要です。

手順

1. 新しい'playbook.yml`ファイルを作成し'次のものを含めます

```
# BeeGFS HA (High Availability) cluster playbook.
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
  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 }}"
  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: Verify the BeeGFS HA cluster is properly deployed.
      ansible.builtin.import_role:
        name: netapp_eseries.beegfs.beegfs_ha_7_4
```

このプレイブックは、Python 3がファイルノードにインストールされていることを確認し、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いくつかの「pre_ttasks」を実行します。

2. BeeGFSを導入する準備ができたら'Ansibleプレイブックコマンドを使用してインベントリとプレイブックファイルを作成します

   配備ではすべての「pre_tasks」が実行され、ユーザーの確認を求めるプロンプトが表示された後、実際のBeeGFS配備に進みます。

   次のコマンドを実行して、必要に応じてフォークの数を調整します（以下の注記を参照）。

```
ansible-playbook -i inventory.yml playbook.yml --forks 20
```

   特に大規模な環境では、 `forks`Ansibleが並行して構成するホストの数を増やすために、パラメータを使用してデフォルトのフォーク数（5）を上書きすることを推奨します。（詳細については、を参照して "プレイブックの実行を制御する"ください）。最大値の設定は、Ansibleコントロールノードで使用可能な処理能力によって異なります。上記の例では、CPUを4つ搭載した仮想Ansibleコントロールノード（インテル（R）Xeon（R）Gold 6146 CPU @ 3.20GHz）上で20を実行しています。

導入のサイズと、Ansible制御ノードとBeeGFSファイルおよびブロックノードの間のネットワークパフォ

ーマンスによって、導入時間が異なる場合があります。

## BeeGFSクライアントを設定します

コンピューティングノードやGPUノードなど、BeeGFSファイルシステムにアクセスする必要のあるホストに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。このタスクでは、AnsibleとBeeGFSコレクションを使用できます。

手順

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. 「host_vars/`」の下で、「<hostname>.yml」という名前のBeeGFSクライアントごとに、次の内容でファイルを作成し、環境に適した情報をプレースホルダテキストに入力します。

```
# BeeGFS Client
ansible_host: <MANAGEMENT_IP>
# OPTIONAL: If you want to use the NetApp E-Series Host Collection's
IPoIB role to configure InfiniBand interfaces for clients to connect to
BeeGFS file systems:
eseries_ipoib_interfaces:
  - name: <INTERFACE>
    address: <IP>/<SUBNET_MASK> # Example: 100.127.1.1/16
  - name: <INTERFACE>
    address: <IP>/<SUBNET_MASK>
```

   2つのサブネットアドレス方式を導入している場合は、2つのストレージIPoIBサブネットそれぞれに1つずつ、各クライアントに2つのInfiniBandインターフェイスを設定する必要があります。ここに示す各BeeGFSサービスのサブネットの例と推奨範囲を使用する場合は、クライアントのインターフェイスの1つを`の範囲で設定し、もう1つを`の範囲で設定する必要があります `100.127.1.0 100.127.99.255 100.128.1.0 100.128.99.255`。

3. 新しいファイル'client_inventory.yml'を作成し'上部に次のパラメータを設定します

```
# BeeGFS client inventory.
all:
  vars:
    ansible_ssh_user: <USER> # This is the user Ansible should use to
connect to each client.
    ansible_become_password: <PASSWORD> # This is the password Ansible
will use for privilege escalation, and requires the ansible_ssh_user be
root, or have sudo privileges.
The defaults set by the BeeGFS HA role are based on the testing
performed as part of this NetApp Verified Architecture and differ from
the typical BeeGFS client defaults.
```

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（のAnsibleのドキュメントを参照してください） "Ansible Vaultを使用したコンテンツの暗号化")または'プレイブックを実行するときに'--Ask -bece-pass`オプションを使用します

4. 「client_inventory.yml」ファイルで、「beegfs_clients」グループの下に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一覧表示し、BeeGFSクライアントカーネルモジュールの構築に必要な追加の設定を指定します。

```
  children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        beegfs_01:
        beegfs_02:
        beegfs_03:
        beegfs_04:
        beegfs_05:
        beegfs_06:
        beegfs_07:
        beegfs_08:
        beegfs_09:
        beegfs_10:
      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、がLinuxのインストールに適した「ヘッダインクルードパス」を指していることを確認して `beegfs_client_ofed_include_path` 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"。

5. 'client_inventory.yml'ファイルに'以前に定義したすべての変数の一番下にマウントするBeeGFSファイル・システムを一覧表示します

```
        beegfs_client_mounts:
          - sysMgmtdHost: 100.127.101.0 # 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
```

「beegfs_client_config」は、テストされた設定を表します。すべてのオプションの包括的な概要については'netapp_eseries.beegfs`コレクションのbeegfs_client'ロールに付属のマニュアルを参照してくださいこれには、複数のBeeGFSファイルシステムのマウントまたは同じBeeGFSファイルシステムの複数回のマウントに関する詳細が含まれます。

6. 新しい'client_playbook.yml`ファイルを作成し'次のパラメータを設定します

```
# BeeGFS client playbook.
- hosts: beegfs_clients
  any_errors_fatal: true
  gather_facts: true
  collections:
    - netapp_eseries.beegfs
    - netapp_eseries.host
  tasks:
    - name: Ensure IPoIB is configured
      import_role:
        name: ipoib
    - name: Verify the BeeGFS clients are configured.
      import_role:
        name: beegfs_client
```

必要なIB/RDMAドライバをインストールし、適切なIPoIBインターフェイスにIPを設定している場合は、「NetApp_eseries.host」コレクションと「IPoIB」ロールのインポートを省略します。

7. クライアントをインストールしてビルドし、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
```

8. BeeGFSファイル・システムを本番環境に配置する前に'任意のクライアントにログインし'beegfs -ffsck --checkfs'を実行して'すべてのノードにアクセスできることと'問題が報告されないことを確認することを強くお勧めします

# 5つのビルディングブロックを超えた拡張性

PacemakerとCorosyncを5つ以上のビルディングブロック（10個のファイルノード）に拡張できるように設定できます。ただし、大規模なクラスタには欠点があるため、最終的にはPacemakerとCorosyncでは最大32個のノードが必要になります。

ネットアップでは、最大10個のノードについてBeeGFS HAクラスタのみをテストしています。この制限を超える個々のクラスタの拡張は推奨もサポートもされません。ただし、BeeGFSファイルシステムは依然として10ノードをはるかに超える規模に拡張する必要があり、ネットアップはNetApp解決策 のBeeGFSでこれを計上しています。

各ファイルシステムのビルディングブロックのサブセットを含む複数のHAクラスタを導入することで、基盤となるHAクラスタリングメカニズムの推奨制限やハード制限とは無関係に、BeeGFSファイルシステム全体を拡張できます。このシナリオでは、次の手順を実行します。

- 追加のHAクラスタを表す新しいAnsibleインベントリを作成し、別の管理サービスの設定は省略します。代わりに'ha_cluster.yml'追加する各クラスタのbeegfs_ha_gmtd_floating_ip'変数を最初のBeeGFS管理サービスのIPに指定します

- 同じファイルシステムにHAクラスタを追加する場合は、次の点を確認してください。
  - BeeGFSノードIDは一意です。
  - 「group_vars」の下の各サービスに対応するファイル名は、すべてのクラスタで一意です。
  - BeeGFSクライアントとサーバのIPアドレスは'すべてのクラスタで一意です
  - 追加のクラスタを導入または更新する前に、BeeGFS管理サービスを含む最初のHAクラスタが実行されています。
- 各HAクラスタのインベントリを、それぞれのディレクトリツリーで個別に維持します。

  複数のクラスタのインベントリファイルを1つのディレクトリツリーに混在させようとすると、BeeGFS HAロールで特定のクラスタに適用される構成を集約する方法で原因 の問題が発生することがあります。

新しいビルディングブロックを作成する前に、HAクラスタごとに5つのビルディングブロックを拡張する必要はありません。多くの場合、クラスタあたりの使用するビルディングブロック数が少なくて済むため、管理が容易です。1つは、各ラックのビルディングブロックをHAクラスタとして構成する方法です。

# ストレージプールのオーバープロビジョニングの割合を推奨します

## 第2世代のビルディングブロックでストレージプールあたりの標準の4ボリューム構成に従う場合は、次の表を参照してください。

次の表に、BeeGFSメタデータまたはストレージ・ターゲットごとの'eseries_storage_pool_configuration'でボリューム・サイズとして使用する推奨パーセンテージを示します。

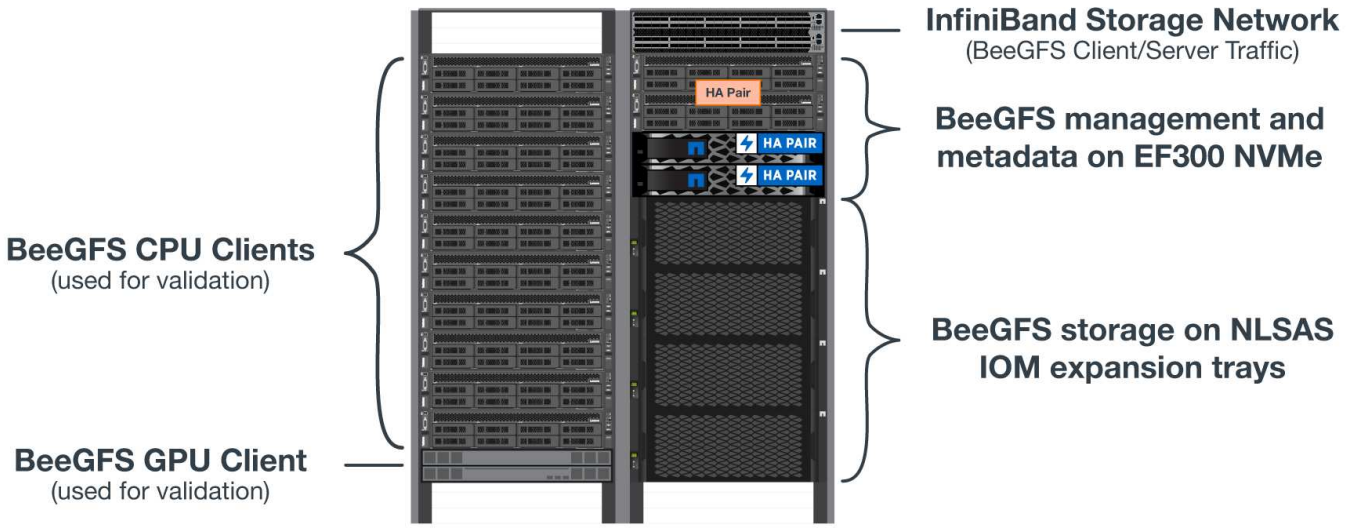
| ドライブサイズ | サイズ |
|---|---|
| 1.92TB | 18 |
| 3.84TB | 21.5 |
| 7.68TB | 22.5インチ |
| 15.3TB | 24 |

上記のガイダンスは、管理サービスが含まれるストレージプールには適用されません。この場合、管理データ用にストレージプールの1%を割り当てるために、25%上のサイズを縮小する必要があります。

これらの値の決定方法については、を参照してください "TR-4800『Appendix A：Understanding SSD持久力とオーバープロビジョニング』"。

# 大容量のビルディングブロック

## 標準のBeeGFS解決策 導入ガイドには、ハイパフォーマンスなワークロードの要件に関する手順と推奨事項が記載されています。大容量の要件を満たすことを検討しているお客様は、ここで紹介する導入方法や推奨事項の違いを確認する必要があります。



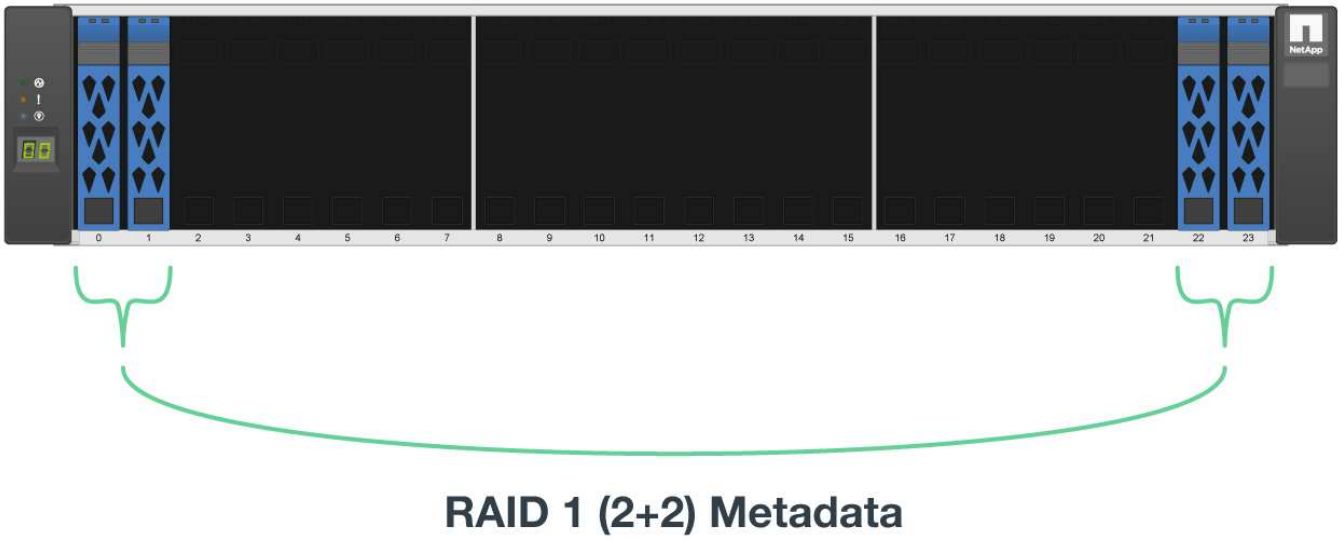

## コントローラ

大容量ビルディングブロックの場合は、EF600コントローラをEF300コントローラに交換し、それぞれのコントローラにSAS拡張用のカスケードHICを取り付ける必要があります。各ブロックノードには、BeeGFSメタデータストレージ用の最小限の数のNVMe SSDがアレイエンクロージャに搭載され、BeeGFSストレージボリューム用にNL-SAS HDDが搭載された拡張シェルフに接続されます。

ファイルノードからブロックノードへの構成は変更されません。

## ドライブの配置

BeeGFSメタデータストレージの各ブロックノードには、少なくとも4本のNVMe SSDが必要です。これらのドライブは、エンクロージャの一番外側のスロットに取り付ける必要があります。




## 拡張トレイ

大容量ビルディングブロックのサイズは、ストレージアレイごとに1~7、60本のドライブ拡張トレイを使用し

て設定できます。

各拡張トレイのケーブル接続手順については、 "ドライブシェルフのEF300ケーブル接続を参照してください"。