



# **BeeGFS** ファイルシステムを定義します

## BeeGFS on NetApp with E-Series Storage

NetApp  
January 27, 2026

# 目次

BeeGFSファイルシステムを定義します	1
Ansibleのインベントリの概要	1
概要	1
手順	1
ファイルシステムを計画	2
概要	2
手順	2
ファイルノードとブロックノードを定義します	3
個々のファイルノードを設定します	4
個々のブロックノードを設定します	7
Common File Node Configurationを指定します	11
Common Block Node Configurationを指定します	18
BeeGFSサービスを定義します	21
BeeGFS管理サービスを定義します	21
BeeGFSメタデータサービスを定義します	22
BeeGFSストレージサービスを定義します	24
BeeGFSサービスをファイルノードにマッピングします	26
概要	26
手順	26

# BeeGFSファイルシステムを定義します

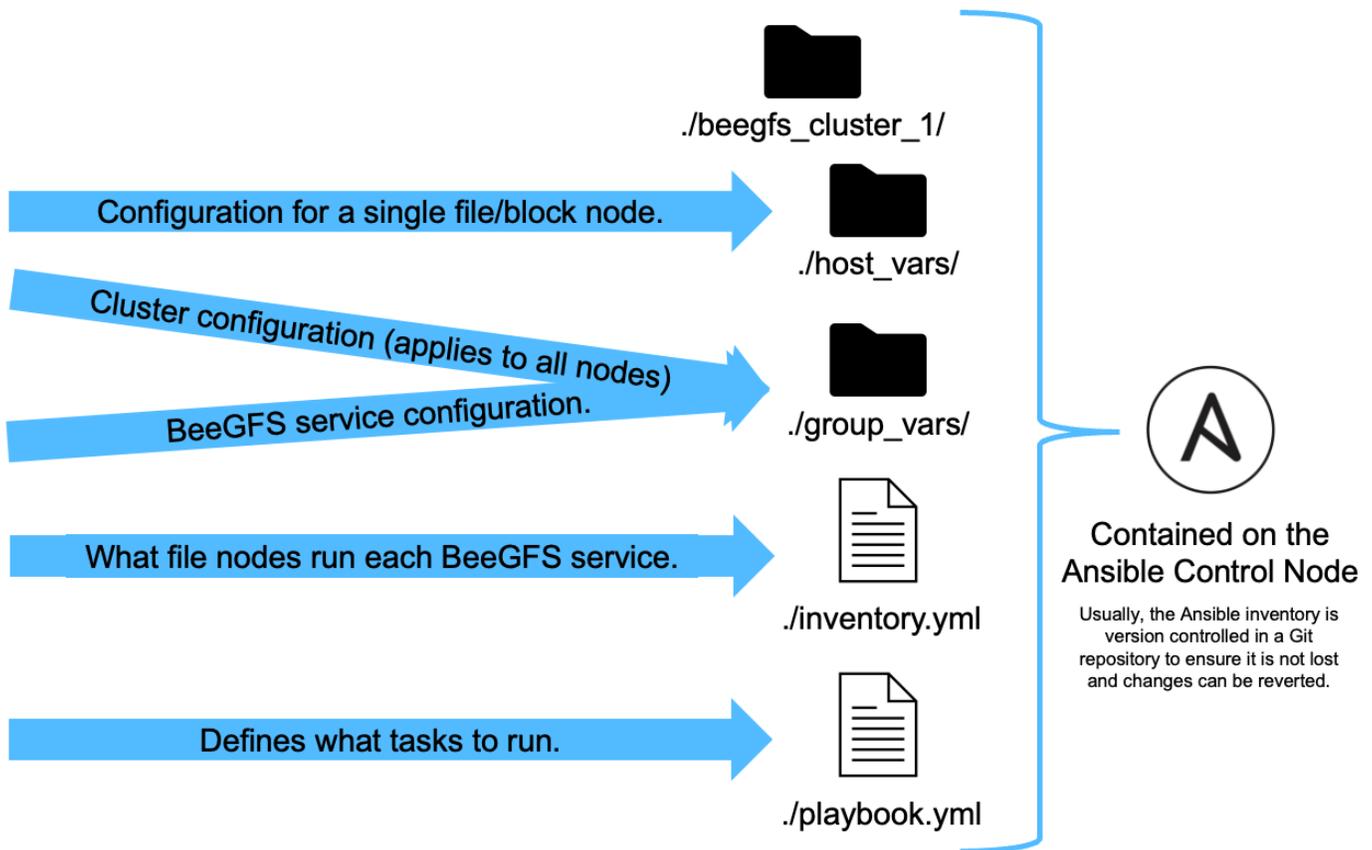
## Ansibleのインベントリの概要

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のできるため、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、のについて説明します host\_vars/<FILE\_NODE\_HOSTNAME>.yaml 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>/yaml、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 ilb  
  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 &reg; (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 `mstconfig` `mstflint` `<a href="https://docs.nvidia.com/networking/display/mftv4270/mft+supported+configurations+and+parameters" target="_blank">"NVIDIAファームウェアツール(MFT)"</a>`。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>.yaml` 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>/yaml`、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、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ない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。firewall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_mgmt_d_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 **"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

- 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_mgmt_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/<sysMgmtHost>\_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ルール"](#) 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 ["テクノロジー要件"](#) <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、NVRAM、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せん。

## 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_nvram: "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、コントローラのレポート（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（小容量の可能性のある）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ン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<sub>11</sub>はドライブチャンネル1に、ドライブ23はドライブチャン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-mgmgtd.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。 storeMgmgtdDirectory、 connAuthFile、 connDisableAuthentication、 connInterfacesFile`および `connNetFilterFile。

```
beegfs_ha_beegfs_mgmgtd_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>.yaml`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。 `sysMgmtHost`、`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、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>.yaml` 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.
  ilb: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](#)

[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。

## 著作権に関する情報

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S.このドキュメントは著作権によって保護されています。著作権所有者の書面による事前承諾がある場合を除き、画像媒体、電子媒体、および写真複写、記録媒体、テープ媒体、電子検索システムへの組み込みを含む機械媒体など、いかなる形式および方法による複製も禁止します。

ネットアップの著作物から派生したソフトウェアは、次に示す使用許諾条項および免責条項の対象となります。

このソフトウェアは、ネットアップによって「現状のまま」提供されています。ネットアップは明示的な保証、または商品性および特定目的に対する適合性の暗示的保証を含み、かつこれに限定されないいかなる暗示的な保証も行いません。ネットアップは、代替品または代替サービスの調達、使用不能、データ損失、利益損失、業務中断を含み、かつこれに限定されない、このソフトウェアの使用により生じたすべての直接的損害、間接的損害、偶発的損害、特別損害、懲罰的損害、必然的損害の発生に対して、損失の発生の可能性が通知されていたとしても、その発生理由、根拠とする責任論、契約の有無、厳格責任、不法行為（過失またはそうでない場合を含む）にかかわらず、一切の責任を負いません。

ネットアップは、ここに記載されているすべての製品に対する変更を随時、予告なく行う権利を保有します。ネットアップによる明示的な書面による合意がある場合を除き、ここに記載されている製品の使用により生じる責任および義務に対して、ネットアップは責任を負いません。この製品の使用または購入は、ネットアップの特許権、商標権、または他の知的所有権に基づくライセンスの供与とはみなされません。

このマニュアルに記載されている製品は、1つ以上の米国特許、その他の国の特許、および出願中の特許によって保護されている場合があります。

権利の制限について：政府による使用、複製、開示は、DFARS 252.227-7013（2014年2月）およびFAR 5252.227-19（2007年12月）のRights in Technical Data -Noncommercial Items（技術データ - 非商用品目に関する諸権利）条項の(b)(3)項、に規定された制限が適用されます。

本書に含まれるデータは商用製品および/または商用サービス（FAR 2.101の定義に基づく）に関係し、データの所有権はNetApp, Inc.にあります。本契約に基づき提供されるすべてのネットアップの技術データおよびコンピュータソフトウェアは、商用目的であり、私費のみで開発されたものです。米国政府は本データに対し、非独占的かつ移転およびサブライセンス不可で、全世界を対象とする取り消し不能の制限付き使用权を有し、本データの提供の根拠となった米国政府契約に関連し、当該契約の裏付けとする場合にのみ本データを使用できます。前述の場合を除き、NetApp, Inc.の書面による許可を事前に得ることなく、本データを使用、開示、転載、改変するほか、上演または展示することはできません。国防総省にかかる米国政府のデータ使用权については、DFARS 252.227-7015(b)項（2014年2月）で定められた権利のみが認められます。

## 商標に関する情報

NetApp、NetAppのロゴ、<http://www.netapp.com/TM>に記載されているマークは、NetApp, Inc.の商標です。その他の会社名と製品名は、それを所有する各社の商標である場合があります。