



ソフトウェアを導入 BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

目次

ソフトウェアを導入.....	1
ファイルノードとブロックノードをセットアップ.....	1
ファイルノードをセットアップします.....	1
ブロックノードをセットアップする.....	2
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.....	2
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.....	2
Redfish APIを使用して、システム設定を調整します.....	4
Ansibleコントロールノードをセットアップします.....	4
Ansibleインベントリを作成します.....	6
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.....	6
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.....	7
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.....	9
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.....	10
手順5：共通ブロックノードの設定を定義する.....	16
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.....	18
手順1：Ansibleインベントリファイルを作成する.....	19
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.....	19
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.....	25
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.....	30
BeeGFSを導入します.....	33
BeeGFSクライアントを設定します.....	36

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が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["テクノロジー要件"](#)。

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。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

システム設定	「」に変更します
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_ChoseOperatingMode": "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>.yaml」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クラスターの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: ilb
    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>.yaml」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_mgmt_d_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_mgmtd_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

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_systemes.yml'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_nvram: "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ります。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、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'の下の子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、クラスターで実行する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

クラスター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_mgmt_d_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、["NetApp EF600アレイの概要"](#)）を参照してください。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%も割り当てますが、'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ドライブの場合には22.25'15.3TBドライブの場合には'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 : 100.127.10 1.3/16 i4b : 100.127.10 2.3 /16	1.	netapp_02	beegfs_m3_ m4_m7_M8	A
meta_04.yml	8045	i4b : 100.127.10 2.4/16 i3b : 100.127.10 1.4 /16	1.	netapp_02	beegfs_m3_ m4_m7_M8	B
meta_05.yml	8055	i1b : 100.127.10 1.5/16 i2b : 100.127.10 2.5 /16	0	netapp_01	beegfs_m1_ m2_m5_m6	A
meta_06.yml	8065	i2b : 100.127.10 2.6/16 i1b : 100.127.10 1.6 /16	0	netapp_01	beegfs_m1_ m2_m5_m6	B
meta_07.yml	8075	i3b : 100.127.10 1.7/16 i4b : 100.127.10 2.7 /16	1.	netapp_02	beegfs_m3_ m4_m7_M8	A
meta_08.yml	8085	i4b : 100.127.10 2.8/16 i3b : 100.127.10 1.8 /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 : 100.127.10 4.4/16 i3b : 100.127.10 3.4 /16	1.	netapp_02	beegfs_s3_s4	B
STOR_05.yml	8053	i1b : 100.127.10 3.5/16 i2b : 100.127.10 4.5 /16	0	netapp_01	beegfs_s5_s6	A
STOR_06.yml	8063	i2b : 100.127.10 4.6/16 i1b : 100.127.10 3.6 /16	0	netapp_01	beegfs_s5_s6	B
STOR_07.yml	8073	i3b : 100.127.10 3.7/16 i4b : 100.127.10 4.7 /16	1.	netapp_02	beegfs_s7_s8	A
STOR_08.yml	8083	i4b : 100.127.10 4.8/16 i3b : 100.127.10 3.8 /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 : 100.127.10 2.10/16 i1b : 100.127.10 1.10 /16	0	netapp_03	beegfs_m9_ m10_m13_M 14	B
meta_11.yml	8035	i3b : 100.127.10 1.11/16 i4b : 100.127.10 2.11 /16	1.	netapp_04	BEegfs_M11_ M12_M15_M 16	A
meta_12.yml	8045	i4b : 100.127.10 2.12/16 i3b : 100.127.10 1.12 /16	1.	netapp_04	BEegfs_M11_ M12_M15_M 16	B
meta_13.yml	8055	i1b : 100.127.10 1.13/16 i2b : 100.127.10 2.13 /16	0	netapp_03	beegfs_m9_ m10_m13_M 14	A
meta_14.yml	8065	i2b : 100.127.10 2.14/16 i1b : 100.127.10 1.14 /16	0	netapp_03	beegfs_m9_ m10_m13_M 14	B
meta_15.yml	8075	i3b : 100.127.10 1.15/16 i4b : 100.127.10 2.15 /16	1.	netapp_04	BEegfs_M11_ M12_M15_M 16	A
meta_16.yml	8085	i4b : 100.127.10 2.16/16 i3b : 100.127.10 1.16 /16	1.	netapp_04	BEegfs_M11_ M12_M15_M 16	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 : 100.127.10 4.12/16 i3b : 100.127.10 3.12 /16	1.	netapp_04	beegfs_s11_s 12を指定します	B
STOR_13.yml	8053	i1b : 100.127.10 3.13/16 i2b : 100.127.10 4.13 /16	0	netapp_03	beegfs_S13_ s14	A
STOR_14.yml	8063	i2b : 100.127.10 4.14/16 i1b : 100.127.10 3.14 /16	0	netapp_03	beegfs_S13_ s14	B
STOR_15.yml	8073	i3b : 100.127.10 3.15/16 i4b : 100.127.10 4.15 /16	1.	netapp_04	beegfs_s15_s 16	A
STOR_16.yml	8083	i4b : 100.127.10 4.16/16 i3b : 100.127.10 3.16 /16	1.	netapp_04	beegfs_s15_s 16	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 : 100.127.10 3.17/16 i2b : 100.127.10 4.17 /16	0	netapp_05	beegfs_s17_s 18	A
STOR_18.yml	8023	i2b : 100.127.10 4.18/16 i1b : 100.127.10 3.18 /16	0	netapp_05	beegfs_s17_s 18	B
STOR_19.yml	8033	i3b : 100.127.10 3.19/16 i4b : 100.127.10 4.19 /16	1.	netapp_06	beegfs_s19_s 20	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 : 100.127.10 4.20/16 i3b : 100.127.10 3.20 /16	1.	netapp_06	beegfs_s19_s 20	B
STOR_21. yml	8053	i1b : 100.127.10 3.21/16 i2b : 100.127.10 4.21 /16	0	netapp_05	beegfs_S21_ s22	A
STOR_22.yml	8063	i2b : 100.127.10 4.22/16 i1b : 100.127.10 3.22 /16	0	netapp_05	beegfs_S21_ s22	B
STOR_23.yml	8073	i3b : 100.127.10 3.23/16 i4b : 100.127.10 4.23 /16	1.	netapp_06	beegfs_S23_ s24	A
STOR_24.yml	8083	i4b : 100.127.10 4.24/16 i3b : 100.127.10 3.24 /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 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_tasks」を実行します。

2. BeeGFSを導入する準備ができたなら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>.yaml」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.yaml」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 -become-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:
      - sysMgmtHost: 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

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。