Déploiement de la solution

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Sommaire

Déploiement de la solution

Présentation du déploiement

BeeGFS sur NetApp peut être déployé sur des nœuds de blocs et de fichiers validés à l'aide d'Ansible et de la conception d'éléments de base BeeGFS de NetApp.

Collections et rôles Ansible

La solution BeeGFS sur NetApp est déployée à l'aide d'Ansible, un moteur d'automatisation IT couramment utilisé pour automatiser les déploiements d'applications. Ansible utilise une série de fichiers appelée collectivement l'inventaire, qui modélise le système de fichiers BeeGFS que vous souhaitez déployer.

Ansible permet à des entreprises comme NetApp de développer leur activité grâce aux fonctionnalités intégrées à l'aide de collections disponibles sur Ansible Galaxy (voir "Collection NetApp E-Series BeeGFS"). Elles comprennent des modules qui exécutent des fonctions ou des tâches spécifiques (telles que la création d'un volume E-Series) ainsi que des rôles pouvant appeler plusieurs modules ou d'autres rôles. Cette approche automatisée réduit la durée de déploiement du système de fichiers BeeGFS et du cluster HA sous-jacent. De plus, il simplifie la maintenance et l'extension du cluster et du système de fichiers BeeGFS.

Pour plus de détails, voir "Découvrez l'inventaire Ansible".

> Comme de nombreuses étapes sont nécessaires pour déployer la solution BeeGFS sur NetApp, NetApp ne prend pas en charge le déploiement manuel de la solution.

Profils de configuration pour les éléments de base BeeGFS

Les procédures de déploiement couvrent les profils de configuration suivants :

- Un seul élément de base inclut des services de gestion, de métadonnées et de stockage.
- Un second élément de base qui inclut les métadonnées et les services de stockage.
- Un troisième élément inclut uniquement des services de stockage.

Ces profils illustrent la gamme complète de profils de configuration recommandés pour les éléments de base NetApp BeeGFS. Pour chaque déploiement, le nombre d'éléments de base de métadonnées et de stockage ou de services de stockage uniquement peut varier en fonction des exigences de capacité et de performances.

Présentation des étapes de déploiement

Le déploiement implique plusieurs tâches générales :

Déploiement matériel

1. Assembler physiquement chaque élément de bâtiment.
2. Installez le rack et le matériel de câblage. Pour des procédures détaillées, voir "Déployez le matériel".

De déploiement logiciel

1. "Configurez les nœuds de fichiers et de blocs".
    - Configurez les adresses IP BMC sur les nœuds de fichiers
    - Installez un système d'exploitation pris en charge et configurez la mise en réseau de gestion sur les

nœuds de fichiers

- Configurez les adresses IP de gestion sur les nœuds de bloc

2. "Configurez un nœud de contrôle Ansible".
3. "Réglez les paramètres du système en fonction des performances".
4. "Créez l'inventaire Ansible".
5. "Définissez l'inventaire Ansible pour les éléments de base BeeGFS".
6. "Déploiement de BeeGFS avec Ansible".
7. "Configurer les clients BeeGFS".

Les procédures de déploiement incluent plusieurs exemples où du texte doit être copié dans un fichier. Portez une attention particulière à tous les commentaires en ligne indiqués par les caractères « # » ou « // » pour tout ce qui doit ou peut être modifié pour un déploiement spécifique. Par exemple :

```
`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"`
```

Architectures dérivées avec variations dans les recommandations de déploiement :

- "Élément de base haute capacité"

Découvrez l'inventaire Ansible

Avant de commencer un déploiement, familiarisez-vous avec la configuration et l'utilisation d'Ansible pour déployer la solution BeeGFS sur NetApp.

L'inventaire Ansible est une structure de répertoires répertoriant les nœuds de fichiers et de blocs sur lesquels le système de fichiers BeeGFS doit être déployé. Il inclut les hôtes, les groupes et les variables qui décrivent le système de fichiers BeeGFS souhaité. L'inventaire Ansible doit être stocké sur le nœud de contrôle Ansible, qui est de toute machine ayant accès aux nœuds de fichiers et de blocs utilisés pour exécuter le PlayBook Ansible. Les inventaires d'échantillons peuvent être téléchargés à partir du "NetApp E-Series BeeGFS GitHub".

Modules et rôles Ansible

Pour appliquer la configuration décrite dans l'inventaire Ansible, utilisez les différents modules et rôles Ansible fournis dans la collection NetApp E-Series (disponible dans le "NetApp E-Series BeeGFS GitHub") qui déploient la solution de bout en bout.

Chaque rôle de la collection NetApp E-Series Ansible est un déploiement de bout en bout complet de la solution BeeGFS sur NetApp. Les rôles utilisent les collections NetApp E-Series SANtricity, Host et BeeGFS qui vous permettent de configurer le système de fichiers BeeGFS avec la haute disponibilité. Vous pouvez ensuite provisionner et mapper le stockage, et vérifier que le stockage du cluster est prêt à être utilisé.

Bien que la documentation approfondie soit fournie avec les rôles, les procédures de déploiement décrivent comment utiliser le rôle de déploiement d'une architecture vérifiée NetApp à l'aide de la conception de l'élément de base BeeGFS deuxième génération.

Bien que la procédure de déploiement tenter d'offrir suffisamment de détails pour que l'expérience précédente avec Ansible ne soit pas une condition préalable, vous devez avoir quelques connaissances de Ansible et de la terminologie connexe.

Disposition de l'inventaire pour un cluster BeeGFS HA

Définissez un cluster BeeGFS haute disponibilité à l'aide de la structure d'inventaire Ansible.

Toute personne ayant déjà de l'expérience Ansible doit savoir que le rôle BeeGFS HA implémente une méthode personnalisée pour identifier les variables (ou les faits) qui s'appliquent à chaque hôte. Cette conception simplifie la structuration de l'inventaire Ansible afin de décrire les ressources pouvant s'exécuter sur plusieurs serveurs.

Un inventaire Ansible comprend généralement les fichiers dans `host_vars` et `group_vars`, ainsi qu'un `inventory.yml` fichier qui attribue des hôtes à des groupes spécifiques (et potentiellement des groupes à d'autres groupes).

Ne créez pas de fichiers contenant le contenu de cette sous-section, qui est uniquement un exemple.

Bien que cette configuration soit prédéterminée en fonction du profil de configuration, vous devez généralement comprendre comment tout s'établit comme un inventaire Ansible, comme suit :

```
# 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.
```

Pour chaque service, un fichier supplémentaire est créé sous `group_vars` description de sa configuration :

```
# 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
```

Cette disposition permet de définir le service, le réseau et la configuration de stockage BeeGFS pour chaque ressource à un seul emplacement. En arrière-plan, le rôle BeeGFS rassemble la configuration nécessaire pour chaque fichier et nœud de bloc en fonction de cette structure d'inventaire.

> L'ID de noeud de type BeeGFS numérique et chaîne pour chaque service est automatiquement configuré en fonction du nom du groupe. Ainsi, en plus de l'exigence générale Ansible pour que les noms de groupe soient uniques, les groupes représentant un service BeeGFS doivent se terminer par un nombre unique pour le type de service BeeGFS que le groupe représente. Par exemple, META_01 et stor_01 sont autorisés, mais Metadata_01 et META_01 ne le sont pas.

Passez en revue les bonnes pratiques

Suivez les bonnes pratiques pour déployer la solution BeeGFS sur NetApp.

Conventions standard

Lors du montage physique et de la création du fichier d'inventaire Ansible, respecter les conventions standard suivantes (pour plus d'informations, voir "Créez l'inventaire Ansible").

- Les noms d'hôte de nœud de fichiers sont numérotés séquentiellement (h01-HN), les chiffres inférieurs se trouvant en haut du rack et les chiffres plus élevés en bas.

  Par exemple, la convention de dénomination `[location][row][rack]hN` ressemble à : `beegfs_01`.

- Chaque nœud de bloc comprend deux contrôleurs de stockage, chacun avec son propre nom d'hôte.

  Un nom de baie de stockage est utilisé pour désigner l'ensemble du système de stockage en mode bloc dans le cadre d'un inventaire Ansible. Les noms de matrice de stockage doivent être numérotés de façon séquentielle (a01 - an), et les noms d'hôte des contrôleurs individuels sont dérivés de cette convention de nommage.

  Par exemple, un nœud de bloc nommé `ictad22a01` généralement peut avoir des noms d'hôte configurés pour chaque contrôleur comme et , mais être référencé dans un inventaire Ansible comme `ictad22a01-a ictad22a01-b netapp_01`.

- Les nœuds de fichiers et de blocs du même bloc de construction partagent le même schéma de numérotation et sont adjacents les uns aux autres dans le rack, les deux nœuds de fichiers étant situés en haut et les deux nœuds de bloc directement en dessous.

  Par exemple, dans le premier module, les nœuds de fichiers h01 et h02 sont tous deux connectés directement aux nœuds de bloc a01 et a02. De haut en bas, les noms d'hôte sont h01, h02, a01 et a02.

- Les blocs de construction sont installés dans un ordre séquentiel en fonction de leurs noms d'hôtes, de sorte que les noms d'hôtes aux numéros inférieurs se trouvent en haut du rack et les noms d'hôtes aux numéros supérieurs se trouvent en bas.

  L'objectif est de réduire la longueur du câble qui passe en haut des commutateurs du rack et de définir une pratique de déploiement standard afin de simplifier le dépannage. Pour les centres de données où cela n'est pas autorisé en raison de problèmes liés à la stabilité des racks, l'inverse est certainement autorisé, en remplissant le rack par le bas vers le haut.

Configuration du réseau de stockage InfiniBand

Moitié des ports InfiniBand sur chaque nœud de fichiers sont utilisés pour se connecter directement aux nœuds de bloc. L'autre moitié est connectée aux commutateurs InfiniBand et est utilisée pour la connectivité client-serveur BeeGFS. Pour déterminer la taille des sous-réseaux IPoIB utilisés pour les clients et les serveurs BeeGFS, vous devez tenir compte de la croissance prévue de votre cluster Compute/GPU et de votre système de fichiers BeeGFS. Si vous devez vous écarter des plages IP recommandées, n'oubliez pas que chaque connexion directe d'un bloc de construction unique possède un sous-réseau unique et qu'il n'y a pas de chevauchement avec les sous-réseaux utilisés pour la connectivité client-serveur.

Connexions directes

Les nœuds de fichiers et de blocs dans chaque bloc de construction utilisent toujours les adresses IP du tableau suivant pour leurs connexions directes.

Ce schéma d'adressage adhère à la règle suivante : le troisième octet est toujours impair ou pair, ce qui dépend du fait que le nœud de fichier est impair ou pair.

Nœud de fichier	Port IB	Adresse IP	Nœud de bloc	Port IB	IP physique	IP virtuel
Impair (h1)	i1a	192.168.1.10	Impair (c1)	2a	192.168.1.100	192.168.1.101
Impair (h1)	i2a	192.168.3.10	Impair (c1)	2a	192.168.3.100	192.168.3.101
Impair (h1)	i3a	192.168.5.10	Pair (c2)	2a	192.168.5.100	192.168.5.101

Nœud de fichier	Port IB	Adresse IP	Nœud de bloc	Port IB	IP physique	IP virtuel
Impair (h1)	i4a	192.168.7.10	Pair (c2)	2a	192.168.7.100	192.168.7.101
Pair (h2)	i1a	192.168.2.10	Impair (c1)	2b	192.168.2.100	192.168.2.101
Pair (h2)	i2a	192.168.4.10	Impair (c1)	2b	192.168.4.100	192.168.4.101
Pair (h2)	i3a	192.168.6.10	Pair (c2)	2b	192.168.6.100	192.168.6.101
Pair (h2)	i4a	192.168.8.10	Pair (c2)	2b	192.168.8.100	192.168.8.101

Schémas d'adressage IPoIB client-serveur BeeGFS

Chaque nœud de fichier exécute plusieurs services BeeGFS Server (gestion, métadonnées ou stockage). Pour permettre à chaque service de basculer de manière indépendante vers l'autre nœud de fichiers, chaque service est configuré avec des adresses IP uniques qui peuvent basculer entre les deux nœuds (parfois appelées interfaces logiques ou LIF).

Bien qu'il ne soit pas obligatoire, ce déploiement suppose que les plages de sous-réseau IPoIB suivantes sont utilisées pour ces connexions et définit un modèle d'adressage standard qui applique les règles suivantes :

- Le second octet est toujours impair ou pair, en fonction du fait que le port InfiniBand du nœud de fichiers est impair ou pair.
- Les adresses IP du cluster BeeGFS sont toujours `xxx. 127.100.yyy` ou `xxx.128.100.yyy`.

> En plus de l'interface utilisée pour la gestion du système d'exploitation intrabande, Corosync peut utiliser des interfaces supplémentaires pour la synchronisation et la battements cardiaques du cluster. Cela permet de s'assurer que la perte d'une interface unique n'entraîne pas l'arrêt complet du cluster.

- Le service BeeGFS Management est toujours à `xxx.yyy.101.0` ou `xxx.yyy.102.0`.
- Les services de métadonnées de BeeGFS sont toujours à `xxx.yyy.101.zzz` ou `xxx.yyy.102.zzz`.
- Les services de stockage BeeGFS sont toujours en `xxx.yyy.103.zzz` ou `xxx.yyy.104.zzz`.
- Adresses dans la plage `100.xxx.1.1` à `100.xxx.99.255` sont réservés aux clients.

Schéma d'adressage de sous-réseau unique IPoIB

Ce guide de déploiement utilisera un schéma de sous-réseau unique compte tenu des avantages répertoriés dans le "architecture logicielle".

Sous-réseau : 100.127.0.0/16

Le tableau suivant fournit la plage pour un sous-réseau unique : 100.127.0.0/16.

Objectif	Port InfiniBand	Adresse IP ou plage
Cluster BeeGFS IP	i1b ou i4b	100.127.100.1 - 100.127.100.255
Gestion BeeGFS	i1b	100.127.101.0
	i2b	100.127.102.0

Objectif	Port InfiniBand	Adresse IP ou plage
Métadonnées BeeGFS	i1b ou i3b	100.127.101.1 - 100.127.101.255
	i2b ou i4b	100.127.102.1 - 100.127.102.255
Stockage BeeGFS	i1b ou i3b	100.127.103.1 - 100.127.103.255
	i2b ou i4b	100.127.104.1 - 100.127.104.255
Clients BeeGFS	(varie selon le client)	100.127.1.1 - 100.127.99.255

IPoIB deux schémas d'adressage de sous-réseau

Un schéma d'adressage à deux sous-réseaux n'est plus recommandé, mais peut encore être implémenté. Reportez-vous aux tableaux ci-dessous pour connaître les deux schémas de sous-réseau recommandés.

Sous-réseau A : 100.127.0.0/16

Le tableau suivant indique la plage pour le sous-réseau A : 100.127.0.0/16.

Objectif	Port InfiniBand	Adresse IP ou plage
Cluster BeeGFS IP	i1b	100.127.100.1 - 100.127.100.255
Gestion BeeGFS	i1b	100.127.101.0
Métadonnées BeeGFS	i1b ou i3b	100.127.101.1 - 100.127.101.255
Stockage BeeGFS	i1b ou i3b	100.127.103.1 - 100.127.103.255
Clients BeeGFS	(varie selon le client)	100.127.1.1 - 100.127.99.255

Sous-réseau B : 100.128.0.0/16

Le tableau suivant indique la plage pour le sous-réseau B : 100.128.0.0/16.

Objectif	Port InfiniBand	Adresse IP ou plage
Cluster BeeGFS IP	i4b	100.128.100.1 - 100.128.100.255
Gestion BeeGFS	i2b	100.128.102.0
Métadonnées BeeGFS	i2b ou i4b	100.128.102.1 - 100.128.102.255
Stockage BeeGFS	i2b ou i4b	100.128.104.1 - 100.128.104.255
Clients BeeGFS	(varie selon le client)	100.128.1.1 - 100.128.99.255

Toutes les adresses IP comprises dans les plages ci-dessus ne sont pas utilisées dans cette architecture vérifiée NetApp. Ils montrent comment les adresses IP peuvent être pré-allouées pour faciliter l'extension du système de fichiers à l'aide d'un schéma d'adressage IP cohérent. Dans ce schéma, les nœuds de fichiers BeeGFS et les ID de service correspondent au quatrième octet d'une plage bien connue d'adresses IP. Le système de fichiers peut évidemment évoluer au-delà de 255 nœuds ou services si nécessaire.

Déployez le matériel

Chaque module comprend deux nœuds de fichiers x86 validés directement connectés à

deux nœuds de bloc à l'aide de câbles InfiniBand HDR (200 Go).

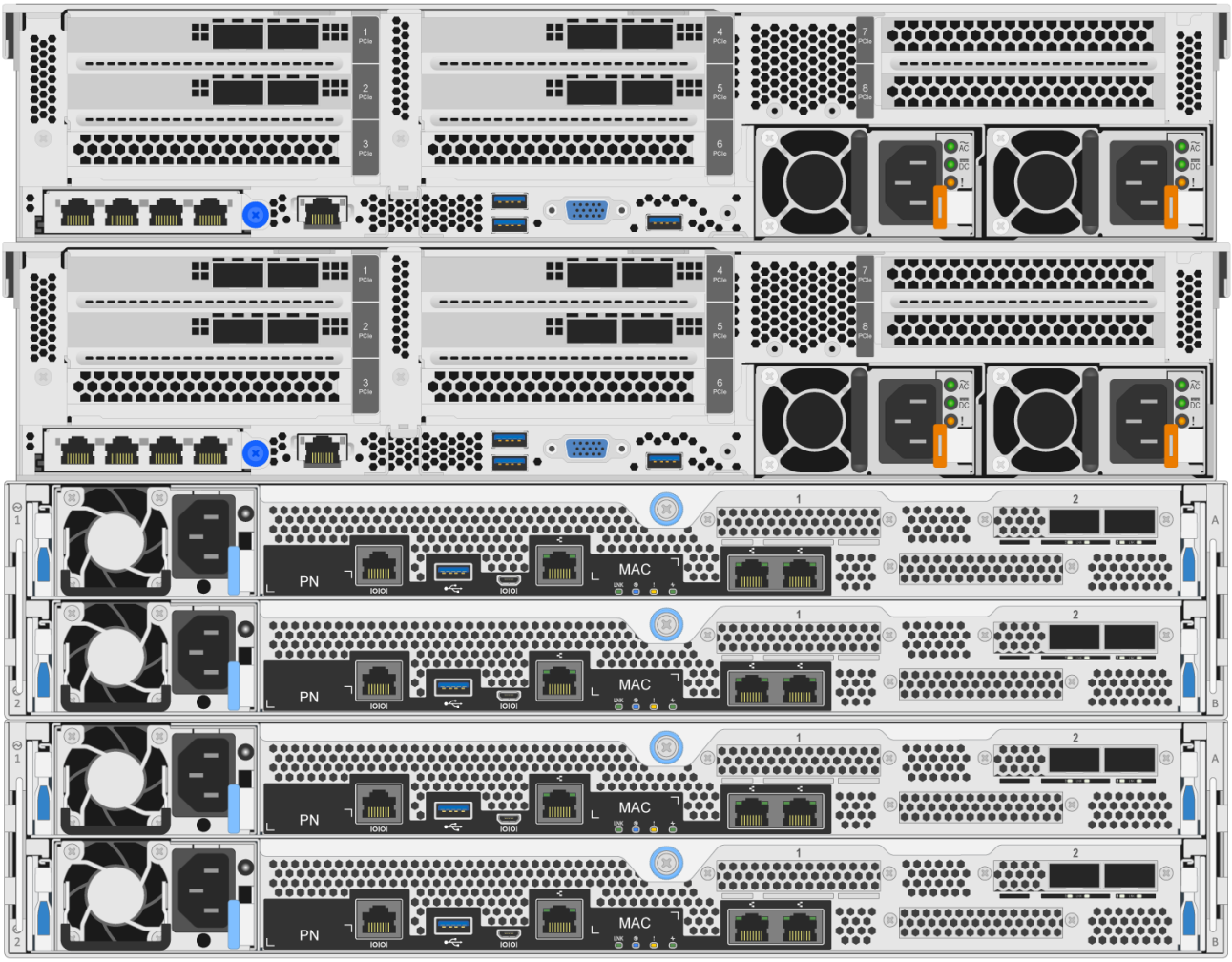
> Un minimum de deux éléments de base est requis pour établir le quorum dans le cluster de basculement. Un cluster à deux nœuds présente des limites qui peuvent empêcher un basculement réussi. Vous pouvez configurer un cluster à deux nœuds en incorporant un troisième périphérique comme disjoncteur d'attache ; cependant, cette documentation ne décrit pas cette conception.

Les étapes suivantes sont identiques pour chaque élément du cluster, qu'il soit utilisé pour exécuter les métadonnées et les services de stockage BeeGFS ou uniquement des services de stockage, sauf indication contraire.

Étapes

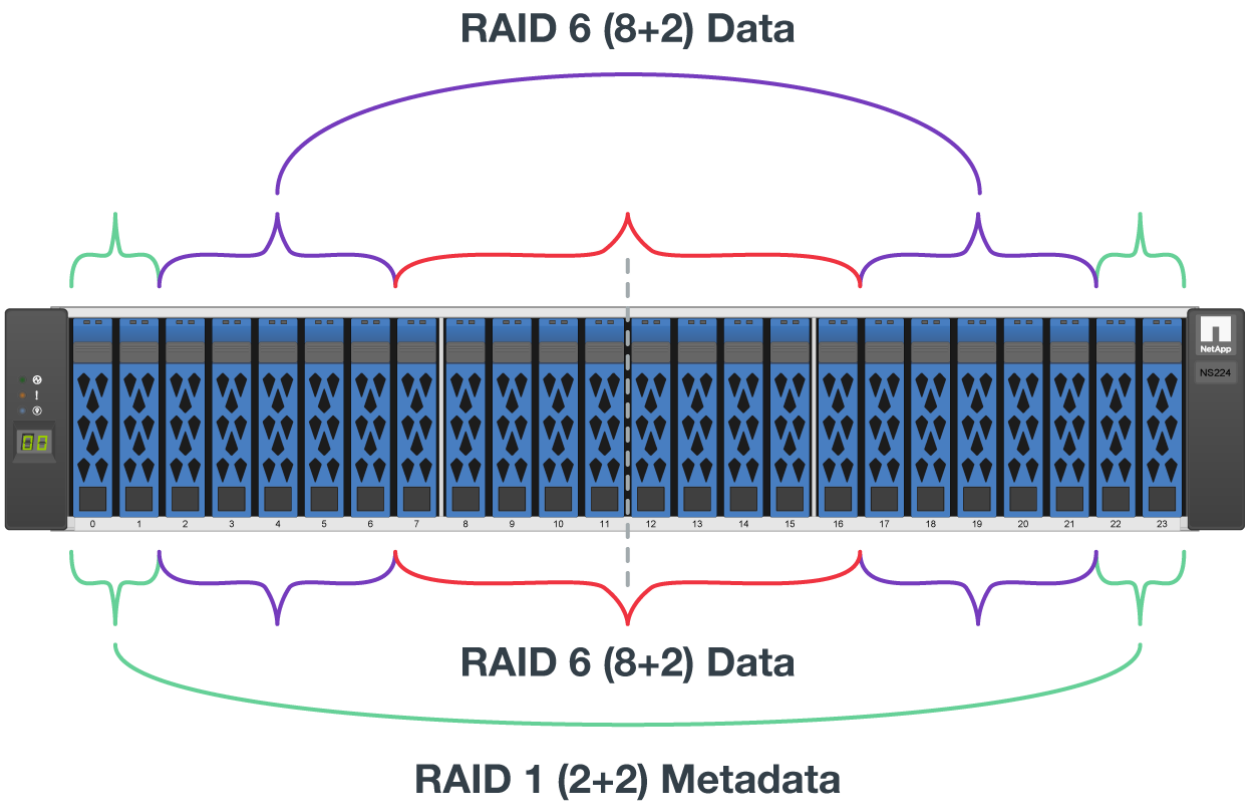
1. Configurez chaque nœud de fichiers BeeGFS avec quatre adaptateurs HCA (Host Channel Adapters) à l'aide des modèles spécifiés dans le "Exigences techniques". Insérez les HCA dans les connecteurs PCIe de votre nœud de fichiers conformément aux spécifications ci-dessous :
   - **Lenovo ThinkSystem SR665 V3 Server:** utilisez les emplacements PCIe 1, 2, 4 et 5.
   - **Lenovo ThinkSystem SR665 Server:** utilisez les emplacements PCIe 2, 3, 5 et 6.
2. Configurez chaque nœud de bloc BeeGFS avec une carte d'interface hôte (HIC) à deux ports 200 Go et installez la HIC dans chacun de ses deux contrôleurs de stockage.

   Placez les éléments de base de façon à ce que les deux nœuds de fichier BeeGFS se trouvent au-dessus des nœuds de bloc BeeGFS. La figure suivante présente la configuration matérielle correcte pour l'élément de base BeeGFS utilisant les serveurs Lenovo ThinkSystem SR665 V3 comme nœuds de fichiers (vue arrière).

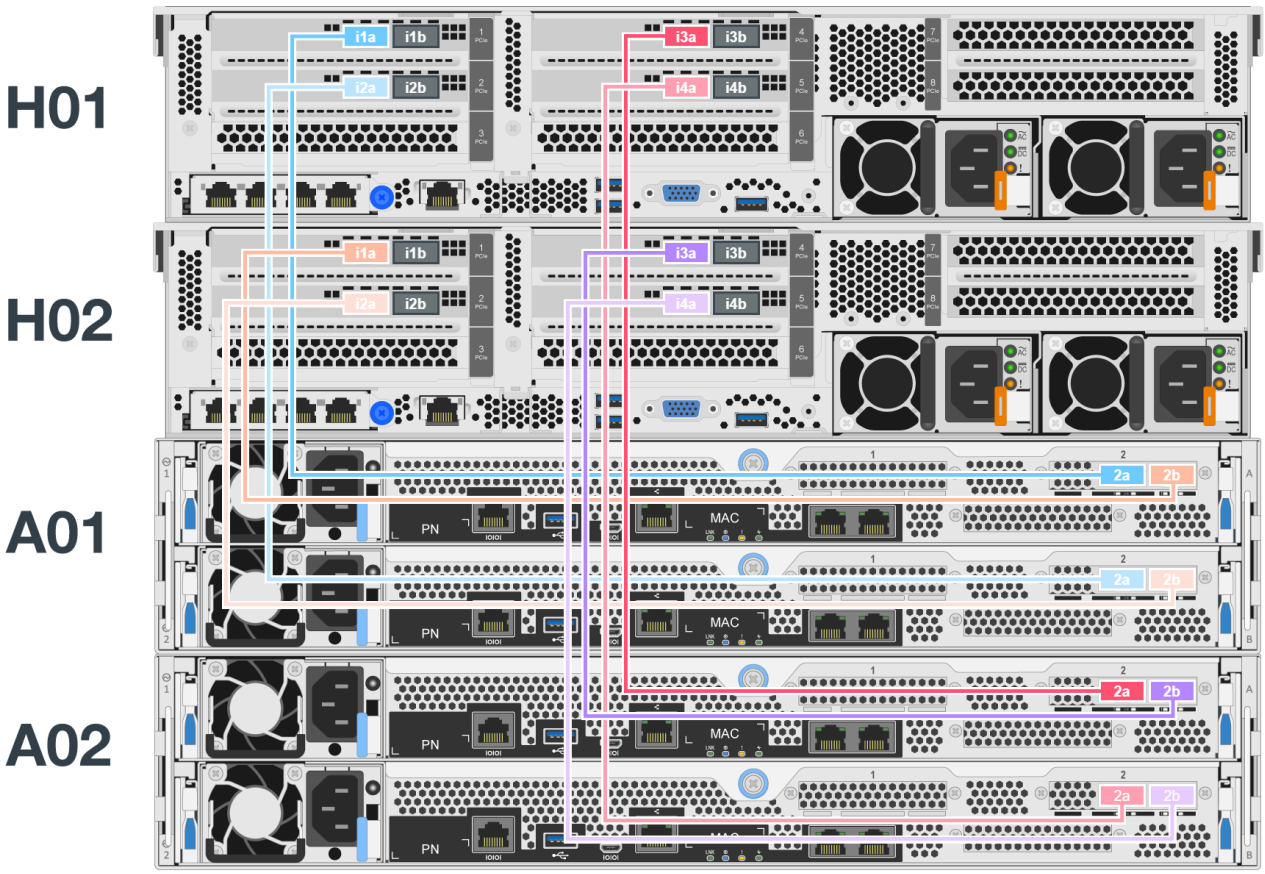


La configuration de l'alimentation électrique pour les cas d'utilisation en production doit généralement utiliser des blocs d'alimentation redondants.

3. Si nécessaire, installez les lecteurs dans chacun des nœuds de bloc BeeGFS.
   a. Si le module sera utilisé pour exécuter des métadonnées et des services de stockage BeeGFS et des disques plus petits sont utilisés pour les volumes de métadonnées, vérifiez qu'ils sont renseignés dans les emplacements de disque les plus à l'extérieur, comme indiqué dans la figure ci-dessous.
   b. Pour toutes les configurations d'éléments de base, si un boîtier de disque n'est pas plein, assurez-vous qu'un nombre égal de disques est utilisé dans les emplacements 0–11 et 12–23 pour des performances optimales.



4. Connectez les nœuds de bloc et de fichier à l'aide de "Câbles en cuivre à connexion directe InfiniBand 200 Gbit/s HDR"la , de manière à ce qu'ils correspondent à la topologie illustrée dans la figure suivante.

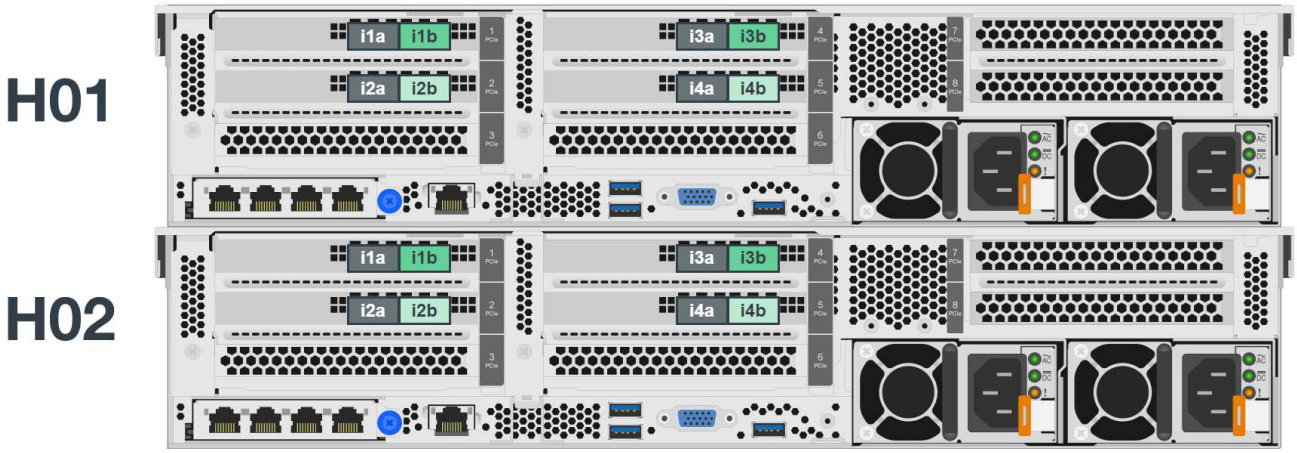


Les nœuds répartis entre plusieurs éléments de base ne sont jamais directement connectés. Chaque élément de base doit être considéré comme une unité autonome et toute communication entre les éléments de base se fait par le biais de commutateurs réseau.

5. Connectez les ports InfiniBand restants du nœud de fichiers au commutateur InfiniBand du réseau de stockage en utilisant le "Câbles InfiniBand de 2 M." commutateur de stockage InfiniBand spécifique à votre commutateur de stockage InfiniBand.

   Lorsque vous utilisez des câbles de séparation pour connecter le commutateur de stockage à des nœuds de fichiers, un câble doit être branché à partir du commutateur et se connecter aux ports indiqués en vert clair. Un autre câble de séparation doit être branché à l'extérieur du commutateur et se connecter aux ports indiqués en vert foncé.

   En outre, pour les réseaux de stockage avec commutateurs redondants, les ports indiqués en vert clair doivent se connecter à un commutateur, tandis que les ports en vert foncé doivent se connecter à un autre commutateur.



6. Au besoin, assembler des éléments de construction supplémentaires en suivant les mêmes directives de câblage.

Le nombre total d'éléments de base pouvant être déployés dans un rack unique dépend de l'alimentation et du refroidissement disponibles sur chaque site.

Déployez des logiciels

Configurez les nœuds de fichiers et les nœuds en mode bloc

Si la plupart des tâches de configuration logicielle sont automatisées au moyen des collections Ansible fournies par NetApp, vous devez configurer la mise en réseau sur le contrôleur de gestion de la carte de base (BMC) de chaque serveur et configurer le port de gestion sur chaque contrôleur.

Configurez les nœuds de fichiers

1. Configurez la mise en réseau sur le contrôleur de gestion de la carte mère (BMC) de chaque serveur.

Pour savoir comment configurer la mise en réseau pour les nœuds de fichiers Lenovo SR665 V3 validés, consultez le "Documentation Lenovo ThinkSystem".

> Un contrôleur de gestion de la carte mère (BMC), parfois appelé processeur de service, est le nom générique de la fonctionnalité de gestion hors bande intégrée dans diverses plates-formes de serveurs qui fournissent un accès à distance même si le système d'exploitation n'est pas installé ou accessible. Les fournisseurs vendent généralement cette fonctionnalité avec leur propre marque. Par exemple, sur le Lenovo SR665, le contrôleur BMC est appelé le contrôleur XClarity (XCC)_ de _Lenovo.

2. Configurez les paramètres du système pour des performances maximales.

   Vous configurez les paramètres système à l'aide de la configuration UEFI (anciennement appelée BIOS) ou en utilisant les API Redfish fournies par de nombreux BMCs. Les paramètres système varient en fonction du modèle de serveur utilisé comme nœud de fichier.

   Pour savoir comment configurer les paramètres système pour les nœuds de fichiers Lenovo SR665 V3 validés, consultez "Réglez les paramètres du système en fonction des performances" .

3. Installez Red Hat Enterprise Linux (RHEL) 9.4 et configurez le nom d'hôte et le port réseau utilisés pour gérer le système d'exploitation, y compris la connectivité SSH à partir du nœud de contrôle Ansible.

   Ne configurez pas d'adresses IP sur l'un des ports InfiniBand pour le moment.

   > Bien qu'il ne soit pas strictement nécessaire, les sections suivantes présument que les noms d'hôte sont numérotés séquentiellement (comme h1-HN) et font référence aux tâches qui doivent être effectuées sur les hôtes impairs et pairs.

4. Utilisez Red Hat Subscription Manager pour enregistrer et abonner le système afin de permettre l'installation des packages requis à partir des référentiels officiels Red Hat et de limiter les mises à jour à la version prise en charge de Red Hat : `subscription-manager release --set=9.4` . Pour obtenir des instructions, voir "Comment enregistrer et souscrire un système RHEL" et "Comment limiter les mises à jour".
5. Activez le référentiel Red Hat contenant les packages requis pour la haute disponibilité.

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

6. Mettez à jour tous les micrologiciels HCA à la version recommandée dans le "Exigences technologiques"Guide d'utilisation"Mettez à jour le micrologiciel de l'adaptateur de nœud de fichier".

Configurez les nœuds en mode bloc

Configurez les nœuds en mode bloc EF600 en configurant le port de gestion sur chaque contrôleur.

1. Configurez le port de gestion sur chaque contrôleur EF600.

   Pour obtenir des instructions sur la configuration des ports, consultez le "Centre de documentation E-Series".

2. Vous pouvez également définir le nom de la matrice de stockage pour chaque système.

La définition d'un nom peut faciliter la référence à chaque système dans les sections suivantes. Pour obtenir des instructions sur la définition du nom de la matrice, reportez-vous à la "Centre de documentation E-Series".

Bien qu'il ne soit pas strictement nécessaire, les rubriques suivantes présument que les noms des matrices de stockage sont numérotés de façon séquentielle (comme c1 - CN) et font référence aux étapes à suivre sur les systèmes pairs ou impairs.

Réglez les paramètres du système de nœud de fichiers en fonction des performances

Pour optimiser les performances, nous vous recommandons de configurer les paramètres système sur le modèle de serveur que vous utilisez en tant que nœuds de fichiers.

Les paramètres système varient en fonction du modèle de serveur que vous utilisez comme nœud de fichier. Cette rubrique décrit comment configurer les paramètres système des nœuds de fichiers serveur Lenovo ThinkSystem SR665 validés.

Utilisez l'interface UEFI pour régler les paramètres du système

Le micrologiciel système du serveur Lenovo SR665 V3 contient de nombreux paramètres de réglage qui peuvent être définis via l'interface UEFI. Ces paramètres de réglage peuvent affecter tous les aspects du fonctionnement du serveur et de son fonctionnement.

Sous **Configuration UEFI > Paramètres système**, réglez les paramètres système suivants :

Menu mode de fonctionnement

Paramètres système	Changer en
Mode de fonctionnement	Personnalisées
CTDP	Manuel
Manuel CTDP	350
Limite de puissance de l'ensemble	Manuel
Mode efficacité	Désactiver
Contrôle global-état-contrôlé	Désactiver
États P SOC	P0
DF États C.	Désactiver
État P.	Désactiver

Paramètres système	Changer en
Activation de la mise hors tension de la mémoire	Désactiver
Nœuds NUMA par socket	NPS1

Menu périphériques et ports d'E/S.

Paramètres système	Changer en
IOMMU	Désactiver

Menu d'alimentation

Paramètres système	Changer en
Frein d'alimentation PCIe	Désactiver

Menu processeurs

Paramètres système	Changer en
Contrôle global de l'état C.	Désactiver
DF États C.	Désactiver
Mode SMT	Désactiver
PC	Désactiver

Utilisez l'API Redfish pour régler les paramètres du système

En plus de l'utilisation de la configuration UEFI, vous pouvez utiliser l'API Redfish pour modifier les paramètres du système.

```
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"
}
}
'
```

Pour plus d'informations sur le schéma Redfish, reportez-vous au "Site Web DMTF".

Configurez un nœud de contrôle Ansible

Pour configurer un nœud de contrôle Ansible, vous devez désigner une machine virtuelle ou physique qui accède au réseau à tous les nœuds de blocs et de fichiers déployés pour la solution BeeGFS sur NetApp.

Consultez le "Exigences techniques" pour obtenir la liste des versions de package recommandées. Les étapes suivantes ont été testées sur Ubuntu 22.04. Pour connaître les étapes spécifiques à votre distribution Linux préférée, consultez le "Documentation Ansible".

1. À partir de votre nœud de contrôle Ansible, installez les packages Python et Python Virtual Environment suivants.

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

2. Créez un environnement virtuel Python.

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

3. Activer l'environnement virtuel.

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

4. Installez les packages Python requis dans l'environnement virtuel activé.

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

5. Installez la collection BeeGFS à l'aide d'Ansible Galaxy.

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

6. Vérifiez que les versions installées d'Ansible, Python et de la collection BeeGFS correspondent aux"Exigences techniques"

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

7. Configurez SSH sans mot de passe pour permettre à Ansible d'accéder aux nœuds de fichiers BeeGFS distants à partir du nœud de contrôle Ansible.
   a. Le cas échéant, générez une paire de clés publiques sur le nœud de contrôle Ansible.

```
ssh-keygen
```

   b. Configurez SSH sans mot de passe sur chacun des nœuds de fichiers.

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

Do **NOT** configurez SSH sans mot de passe sur les nœuds de bloc. Cela n'est ni pris en charge ni obligatoire.

Créez l'inventaire Ansible

Pour définir la configuration des nœuds de fichiers et de blocs, vous créez un inventaire Ansible qui représente le système de fichiers BeeGFS que vous souhaitez déployer. L'inventaire inclut les hôtes, les groupes et les variables décrivant le système de fichiers BeeGFS souhaité.

Étape 1 : définir la configuration de tous les éléments de base

Définissez la configuration qui s'applique à tous les blocs de construction, quel que soit le profil de configuration que vous pouvez appliquer individuellement.

Avant de commencer

- Choisissez un schéma d'adressage de sous-réseau pour votre déploiement. En raison des avantages répertoriés dans le "architecture logicielle", il est recommandé d'utiliser un schéma d'adressage de sous-réseau unique.

Étapes

1. Sur votre nœud de contrôle Ansible, identifiez un répertoire à utiliser pour stocker les fichiers d'inventaire et de PlayBook Ansible.

   Sauf indication contraire, tous les fichiers et répertoires créés dans cette étape et les étapes suivantes sont créés par rapport à ce répertoire.

2. Créez les sous-répertoires suivants :

   `host_vars`

   `group_vars`

   `packages`

3. Créez un sous-répertoire pour les mots de passe de cluster et sécurisez le fichier en le chiffrant à l'aide d'Ansible Vault (voir "Cryptage de contenu avec Ansible Vault") :

   a. Créez le sous-répertoire `group_vars/all`.

   b. Dans le `group_vars/all` répertoire, créez un fichier de mots de passe intitulé `passwords.yml`.

   c. Remplissez le `passwords.yml file` avec les paramètres suivants, en remplaçant tous les paramètres de nom d'utilisateur et de mot de passe en fonction de votre configuration :

```
# 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. Exécutez `ansible-vault encrypt passwords.yml` et définissez un mot de passe de coffre-fort lorsque vous y êtes invité.

Étape 2 : définir la configuration des nœuds de fichiers et de blocs individuels

Définissez la configuration qui s'applique aux nœuds de fichiers individuels et aux nœuds d'élément de base individuels.

1. Sous `host_vars/`, Créez un fichier pour chaque noeud de fichier BeeGFS nommé `<HOSTNAME>.yml` Avec le contenu suivant, en portant une attention particulière aux notes concernant le contenu à remplir pour les adresses IP de cluster BeeGFS et les noms d'hôte se terminant par des nombres impairs et impairs.

   Initialement, les noms d'interface de nœud de fichier correspondent à ce qui est répertorié ici (comme ib0 ou ibs1f0). Ces noms personnalisés sont configurés dans Étape 4 : définissez la configuration qui doit s'appliquer à tous les nœuds de fichiers.

```
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
```

Si vous avez déjà déployé le cluster BeeGFS, vous devez arrêter le cluster avant d'ajouter ou de modifier des adresses IP configurées de manière statique, y compris les adresses IP et IP du cluster utilisées pour NVMe/IB. Cette modification est nécessaire afin que ces modifications prennent effet correctement et ne perturbent pas les opérations du cluster.

2. Sous `host_vars/`, Créez un fichier pour chaque noeud de bloc BeeGFS nommé `<HOSTNAME>.yml` et remplissez-le avec le contenu suivant.

   Faites particulièrement attention aux remarques concernant le contenu à remplir pour les noms de matrices de stockage se terminant par des nombres pairs ou impairs.

   Pour chaque noeud de bloc, créez un fichier et spécifiez `<MANAGEMENT_IP>` Pour un des deux contrôleurs (généralement Un).

```
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
```

Étape 3 : définissez une configuration à appliquer à tous les nœuds de fichiers et de blocs

Vous pouvez définir une configuration commune à un groupe d'hôtes sous `group_vars` dans un nom de fichier correspondant au groupe. Cela empêche de répéter une configuration partagée à plusieurs endroits.

Description de la tâche

Les hôtes peuvent se trouver dans plusieurs groupes et au moment de l'exécution, Ansible choisit les variables qui s'appliquent à un hôte donné en fonction de ses règles de priorité de variable. (Pour plus d'informations sur ces règles, consultez la documentation Ansible pour "Utilisation de variables".)

Les affectations hôte-groupe sont définies dans le fichier d'inventaire Ansible réel, créé à la fin de cette procédure.

Étape

Dans Ansible, vous pouvez définir n'importe quelle configuration que vous souhaitez appliquer à tous les hôtes dans un groupe appelé `All`. Créez le fichier `group_vars/all.yml` avec le contenu suivant :

```
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"
```

Étape 4 : définissez la configuration qui doit s'appliquer à tous les nœuds de fichiers

La configuration partagée pour les nœuds de fichiers est définie dans un groupe appelé `ha_cluster`. Les étapes de cette section créent la configuration qui doit être incluse dans le `group_vars/ha_cluster.yml` fichier.

Étapes

1. En haut du fichier, définissez les valeurs par défaut, y compris le mot de passe à utiliser comme `sudo` utilisateur sur les nœuds de fichiers.

```
### 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"
```

> Si `ansible_ssh_user` est déjà `root`, vous pouvez omettre l' `ansible_become_password` et spécifier l' `--ask-become-pass`option lors de l'exécution du PlayBook.

2. Vous pouvez également configurer un nom pour le cluster haute disponibilité (HA) et spécifier un utilisateur pour les communications intra-cluster.

   Si vous modifiez le schéma d'adressage IP privé, vous devez également mettre à jour le schéma par défaut `beegfs_ha_mgmtd_floating_ip`. Ceci doit correspondre à ce que vous configurez plus tard pour le groupe de ressources BeeGFS Management.

   Spécifiez un ou plusieurs e-mails qui doivent recevoir des alertes pour les événements du cluster à l'aide de `beegfs_ha_alert_email_list`.

```
### 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
```

Tout en apparence redondant, `beegfs_ha_mgmtd_floating_ip` Est important lorsque vous faites évoluer le système de fichiers BeeGFS au-delà d'un seul cluster HA. Les clusters HA suivants sont déployés sans service de gestion BeeGFS et point supplémentaires sur le service de gestion fourni par le premier cluster.

3. Configurer un agent d'escrime. (Pour plus de détails, voir "Configurer l'escrime dans un cluster Red Hat haute disponibilité".) Le résultat suivant présente des exemples de configuration des agents de clôture courants. Choisissez l'une de ces options.

   Pour cette étape, gardez à l'esprit que :

- Par défaut, l'escrime est activé, mais vous devez configurer un *agent* d'escrime.
- Le `<HOSTNAME>` spécifié dans le `pcmk_host_map` ou `pcmk_host_list` Doit correspondre au nom d'hôte dans l'inventaire Ansible.
- L'utilisation du cluster BeeGFS sans escrime n'est pas prise en charge, particulièrement en production. Cela permet de s'assurer que les services BeeGFS, y compris les dépendances de ressources comme les périphériques de bloc, basculent en raison d'un problème, il n'y a aucun risque d'accès simultané par plusieurs nœuds qui entraînent une corruption du système de fichiers ou tout autre comportement indésirable ou inattendu. Si l'escrime doit être désactivé, reportez-vous aux notes générales du guide de démarrage et de mise en place du rôle BeeGFS HA `beegfs_ha_cluster_crm_config_options["stonith-enabled"]` à faux dans `ha_cluster.yml`.
- Plusieurs dispositifs d'escrime au niveau des nœuds sont disponibles, et le rôle BeeGFS HA peut configurer n'importe quel agent d'escrime disponible dans le référentiel de package Red Hat HA. Si possible, utilisez un agent d'escrime qui fonctionne via l'alimentation sans coupure (UPS) ou l'unité de distribution de l'alimentation en rack (RPDU), Parce que certains agents d'escrime, tels que le contrôleur de gestion de la carte mère (BMC) ou d'autres dispositifs d'éclairage intégrés au serveur, peuvent ne pas répondre à la demande de clôture dans certains scénarios de panne.

```
### 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. Activez le réglage des performances recommandé dans le système d’exploitation Linux.

   Si de nombreux utilisateurs trouvent les paramètres par défaut des paramètres de performance qui fonctionnent généralement bien, vous pouvez également modifier les paramètres par défaut d’une charge de travail donnée. Ainsi, ces recommandations sont incluses dans le rôle BeeGFS, mais ne sont pas activées par défaut pour s’assurer que les utilisateurs connaissent le réglage appliqué à leur système de fichiers.

   Pour activer le réglage des performances, spécifiez :

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

5. (Facultatif) vous pouvez régler les paramètres d'ajustement des performances dans le système d'exploitation Linux selon vos besoins.

   Pour obtenir une liste complète des paramètres de réglage disponibles que vous pouvez ajuster, consultez la section Réglages par défaut des performances du rôle haute disponibilité BeeGFS dans la section "E-Series site GitHub BeeGFS". Les valeurs par défaut peuvent être remplacées pour tous les nœuds du cluster dans ce fichier ou pour le `host_vars` fichier d'un nœud individuel.

6. Pour permettre une connectivité 200 Go/HDR complète entre les nœuds de bloc et de fichier, utilisez le progiciel Open Subnet Manager (OpenSM) de NVIDIA Open Fabrics Enterprise distribution (MLNX_OFED). La version MLNX_OFED de la présente "configuration requise pour le nœud de fichiers" est fournie avec les packages OpenSM recommandés. Bien que le déploiement à l'aide d'Ansible soit pris en charge, vous devez d'abord installer le pilote MLNX_OFED sur tous les nœuds de fichiers.

   a. Remplissez les paramètres suivants dans `group_vars/ha_cluster.yml` (réglez les colis si nécessaire) :

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

7. Configurer le `udev` Règle pour assurer un mappage cohérent des identificateurs de port InfiniBand logiques aux périphériques PCIe sous-jacents.

   Le `udev` La règle doit être unique à la topologie PCIe de chaque plate-forme de serveur utilisée comme nœud de fichier BeeGFS.

   Utilisez les valeurs suivantes pour les nœuds de fichiers vérifiés :

```
### 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. (Facultatif) mettre à jour l'algorithme de sélection de cible de métadonnées.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

> Lors des tests de vérification, `randomrobin` Est généralement utilisé pour s'assurer que les fichiers de test étaient répartis de façon égale sur toutes les cibles de stockage BeeGFS pendant l'évaluation des performances (pour plus d'informations sur l'analyse comparative, consultez le site BeeGFS pour "Analyse comparative d'un système BeeGFS"). Avec une utilisation réelle, il est possible que les cibles numérotées soient plus rapidement que les cibles numérotées plus élevées. Omission `randomrobin` et il suffit d'utiliser la valeur par défaut `randomized` la valeur a été indiquée pour fournir de bonnes performances tout en utilisant toujours toutes les cibles disponibles.

Étape 5 : définir la configuration pour le nœud de bloc commun

La configuration partagée pour les nœuds de bloc est définie dans un groupe appelé `eseries_storage_systems`. Les étapes de cette section créent la configuration qui doit être incluse dans le `group_vars/ eseries_storage_systems.yml` fichier.

Étapes

1. Définissez la connexion Ansible sur local, indiquez le mot de passe système et spécifiez si les certificats

SSL doivent être vérifiés. (Normalement, Ansible utilise SSH pour la connexion aux hôtes gérés, mais dans le cas des systèmes de stockage NetApp E-Series utilisés comme nœuds de bloc, les modules utilisent l'API REST pour la communication.) En haut du fichier, ajoutez ce qui suit :

```
### 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. Pour assurer des performances optimales, installez les versions répertoriées pour les nœuds de bloc dans "Exigences techniques".

   Téléchargez les fichiers correspondants à partir du "Site de support NetApp". Vous pouvez les mettre à niveau manuellement ou les inclure dans le `packages/` Répertoire du nœud de contrôle Ansible, puis remplissez les paramètres suivants dans `eseries_storage_systems.yml` Pour la mise à niveau avec Ansible :

   ```
   # 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. Téléchargez et installez le dernier micrologiciel de lecteur disponible pour les lecteurs installés sur vos nœuds de bloc à partir du "Site de support NetApp". Vous pouvez les mettre à niveau manuellement ou les inclure dans `packages/` le répertoire du nœud de contrôle Ansible, puis remplir les paramètres suivants dans `eseries_storage_systems.yml` pour la mise à niveau à l'aide d'Ansible :

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

   > Réglage `eseries_drive_firmware_upgrade_drives_online` à `false` Accélère la mise à niveau, mais ne doit pas être effectuée avant le déploiement de BeeGFS. En effet, ce paramètre nécessite l'arrêt de toutes les E/S des disques avant la mise à niveau afin d'éviter les erreurs d'application. Bien que la mise à niveau en ligne du micrologiciel des lecteurs avant la configuration des volumes soit toujours rapide, nous vous recommandons de toujours définir cette valeur sur `true` pour éviter tout problème par la suite.

4. Pour optimiser les performances, effectuez les modifications suivantes de la configuration globale :

```
# 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. Pour optimiser le provisionnement et le comportement des volumes, spécifiez les paramètres suivants :

```
# 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"
```

> La valeur spécifiée pour `eseries_storage_pool_usable_drives` Est spécifique aux nœuds de bloc NetApp EF600 et contrôle l'ordre dans lequel les disques sont affectés aux nouveaux groupes de volumes. Cette commande permet de s'assurer que les E/S de chaque groupe sont réparties de manière homogène entre les canaux des disques back-end.

Définissez l'inventaire Ansible pour les éléments de base BeeGFS

Après avoir défini la structure d'inventaire générale Ansible, définissez la configuration de chaque élément de base dans le système de fichiers BeeGFS.

Ces instructions de déploiement montrent comment déployer un système de fichiers composé d'un élément de base, incluant la gestion, les métadonnées et les services de stockage, un deuxième élément de base avec des métadonnées et des services de stockage, et un troisième élément de base uniquement dédié au stockage.

Ces étapes sont destinées à afficher la gamme complète des profils de configuration standard que vous pouvez utiliser pour configurer les éléments de base NetApp BeeGFS de façon à répondre aux exigences du système de fichiers global BeeGFS.

> Dans les sections suivantes et ceci, ajustez selon les besoins pour générer l'inventaire représentant le système de fichiers BeeGFS que vous voulez déployer. Utilisez notamment des noms d'hôte Ansible qui représentent chaque nœud de bloc ou de fichier et le schéma d'adressage IP souhaité pour le réseau de stockage, afin de vous assurer qu'il peut évoluer jusqu'au nombre de nœuds de fichiers et de clients BeeGFS.

Étape 1 : créez le fichier d'inventaire Ansible

Étapes

1. Créer un nouveau `inventory.yml` file, puis insérez les paramètres suivants en remplaçant les hôtes sous `eseries_storage_systems` si nécessaire pour représenter les nœuds en mode bloc dans votre déploiement. Les noms doivent correspondre au nom utilisé pour `host_vars/<FILENAME>.yml`.

```
# 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:
```

   Dans les sections suivantes, vous allez créer des groupes Ansible supplémentaires sous `ha_cluster` Qui représentent les services BeeGFS que vous voulez exécuter dans le cluster.

Étape 2 : configurer l'inventaire d'un élément de base de gestion, de métadonnées et de stockage

Le premier élément de base ou du cluster doit inclure le service de gestion BeeGFS ainsi que les services de métadonnées et de stockage :

Étapes

1. Dans `inventory.yml`, remplissez les paramètres suivants sous `ha_cluster: children`:

```
      # 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. Créez le fichier `group_vars/mgmt.yml` et inclure les éléments suivants :

```
# 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. Sous `group_vars/`, créez des fichiers pour les groupes de ressources `meta_01` à `meta_08` à l'aide du modèle suivant, puis remplissez les valeurs des espaces réservés pour chaque service faisant référence au tableau ci-dessous :

```
# 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>
```

La taille du volume est indiquée sous forme de pourcentage du pool de stockage global (également appelé groupe de volumes). NetApp recommande fortement de laisser une certaine capacité libre dans chaque pool afin d'autoriser le sur-provisionnement SSD (pour plus d'informations, voir "Présentation de la baie NetApp EF600"). Le pool de stockage, `beegfs_m1_m2_m5_m6`, alloue également 1% de la capacité du pool pour le service de gestion. Ainsi, pour les volumes de métadonnées dans le pool de stockage, `beegfs_m1_m2_m5_m6`, Si vous utilisez des disques de 1,92 To ou 3,84 To, définissez cette valeur sur `21.25`; Pour les lecteurs 7,65 To, définissez cette valeur sur `22.25`; Et pour les disques de 15,3 To, définissez cette valeur sur `23.75`.

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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
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

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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. Sous `group_vars/`, créez des fichiers pour les groupes de ressources `stor_01` à `stor_08` à l'aide du modèle suivant, puis remplissez les valeurs de paramètre fictif pour chaque service référençant l'exemple :

```
# 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>
```

Pour connaître la taille correcte à utiliser, reportez-vous à la section "Pourcentages de surprovisionnement recommandés pour le pool de stockage".

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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
stor_04.yml	8043	i4b:100.127.104.4/16 i3b:100.127.103.4/16	1	netapp_02	beegfs_s3_s4	B

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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

Étape 3 : configurer l'inventaire d'un élément de base métadonnées + stockage

Elles expliquent comment configurer un inventaire Ansible pour un élément de base de stockage + de métadonnées BeeGFS.

Étapes

1. Dans `inventory.yml`, remplissez les paramètres suivants sous la configuration existante :

```
        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. Sous `group_vars/`, créez des fichiers pour les groupes de ressources `meta_09` à `meta_16` à l'aide du modèle suivant, puis remplissez les valeurs de paramètre fictif pour chaque service référençant l'exemple :

```
# 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>
```

Pour connaître la taille correcte à utiliser, reportez-vous à la section "Pourcentages de surprovisionnement recommandés pour le pool de stockage".

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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
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

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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. Sous `group_vars/`, créez des fichiers pour les groupes de ressources `stor_09` à `stor_16` à l'aide du modèle suivant, puis remplissez les valeurs de paramètre fictif pour chaque service référençant l'exemple :

```
# 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>
```

Pour connaître la taille correcte à utiliser, voir "Pourcentages de surprovisionnement recommandés pour le pool de stockage" ..

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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	A
stor_12.yml	8043	i4b:100.127.104.12/16 i3b:100.127.103.12/16	1	netapp_04	beegfs_s11_s12	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

Étape 4 : configurer l'inventaire pour un élément de base stockage uniquement

Procédure de configuration d'un inventaire Ansible pour un élément de base BeeGFS Storage uniquement. La différence majeure entre l'installation de la configuration pour un bloc de métadonnées + stockage et un bloc modulaire uniquement destiné au stockage, c'est l'omission de tous les groupes de ressources de métadonnées et la modification `criteria_drive_count` de 10 à 12 pour chaque pool de stockage.

Étapes

1. Dans `inventory.yml`, remplissez les paramètres suivants sous la configuration existante :

```
# 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. Sous `group_vars/`, créez des fichiers pour les groupes de ressources `stor_17` à `stor_24` à l'aide du modèle suivant, puis remplissez les valeurs de paramètre fictif pour chaque service référençant l'exemple :

```
# 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>
```

Pour connaître la taille correcte à utiliser, voir "Pourcentages de surprovisionnement recommandés pour le pool de stockage" .

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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
stor_20.yml	8043	i4b:100.127.104.20/16 i3b:100.127.103.20/16	1	netapp_06	beegfs_s19_s20	B

Nom du fichier	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
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

Déployez BeeGFS

Un déploiement et une gestion de la configuration impliquent d'exécuter un ou plusieurs playbooks contenant les tâches Ansible requises pour exécuter et placer le système global dans l'état souhaité.

Même si toutes les tâches peuvent être incluses dans un seul manuel de vente, il est difficile pour les systèmes complexes de gérer cette tâche très rapidement. Ansible vous permet de créer et de distribuer des rôles comme un moyen de packaging des playbooks réutilisables et du contenu associé (par exemple, variables par défaut, tâches et gestionnaires). Pour plus d'informations, consultez la documentation Ansible pour "Rôles".

Les rôles sont souvent distribués dans le cadre d'une collection Ansible contenant des rôles et des modules associés. Donc, ces playbooks importent principalement plusieurs rôles distribués dans les différentes collections NetApp E-Series Ansible.

> Actuellement, au moins deux éléments de base (quatre nœuds de fichiers) sont nécessaires pour déployer BeeGFS, à moins qu'un périphérique quorum distinct soit configuré comme un disjoncteur d'attache pour limiter les problèmes lors de l'établissement du quorum avec un cluster à deux nœuds.

Étapes

1. Créer un nouveau `playbook.yml` classez et incluez les éléments suivants :

```
# 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
```

Ce PlayBook s'exécute `pre_tasks` Vérifiez que Python 3 est installé sur les nœuds de fichiers et vérifiez que les balises Ansible fournies sont prises en charge.

2. Utilisez le `ansible-playbook` Commande avec les fichiers d'inventaire et de PlayBook lorsque vous êtes prêt à déployer BeeGFS.

   Le déploiement va s'exécuter tout `pre_tasks`, Puis demander confirmation de l'utilisateur avant de poursuivre le déploiement BeeGFS.

   Exécuter la commande suivante en réglant le nombre de fourches selon les besoins (voir la remarque ci-dessous) :

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

   `forks`Pour les déploiements de plus grande envergure, il est recommandé de remplacer le nombre par défaut de fourches (5) à l'aide du paramètre afin d'augmenter le nombre d'hôtes configurés en parallèle par Ansible. (Pour plus d'informations, voir "Contrôle de l'exécution de PlayBook".) Le paramètre valeur maximale dépend de la puissance de traitement disponible sur le nœud de contrôle Ansible. L'exemple ci-dessus de 20 a été exécuté sur un nœud de contrôle Ansible virtuel avec 4 processeurs (Intel® Xeon® Gold 6146 CPU à 3,20 GHz).

   Selon la taille du déploiement et les performances réseau entre le nœud de contrôle Ansible et les nœuds de fichier et bloc BeeGFS, la durée de déploiement peut varier.

Configurer les clients BeeGFS

Vous devez installer et configurer le client BeeGFS sur tous les hôtes qui doivent accéder au système de fichiers BeeGFS, comme les nœuds de calcul ou les nœuds GPU. Pour

cette tâche, vous pouvez utiliser Ansible et la collection BeeGFS.

Étapes

1. Si nécessaire, configurez une connexion SSH sans mot de passe depuis le nœud de contrôle Ansible vers chacun des hôtes que vous souhaitez configurer comme clients BeeGFS :

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

2. Sous `host_vars/`, Créez un fichier pour chaque client BeeGFS nommé `<HOSTNAME>.yml` avec le contenu suivant, en renseignant le texte de l'espace réservé contenant les informations correctes pour votre environnement :

   ```
   # 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>
   ```

   > En cas de déploiement avec un schéma d'adressage de sous-réseau à deux, deux interfaces InfiniBand doivent être configurées sur chaque client, une dans chacun des deux sous-réseaux IPoIB de stockage. Si vous utilisez les exemples de sous-réseaux et les plages recommandées pour chaque service BeeGFS répertorié ici, une interface doit être configurée dans la plage `100.127.1.0` `100.127.99.255` à et l'autre dans `100.128.1.0` à `100.128.99.255`.

3. Créez un nouveau fichier `client_inventory.yml`, puis remplissez les paramètres suivants en haut :

   ```
   # 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.
   ```

Ne stockez pas les mots de passe en texte brut. Utilisez plutôt Ansible Vault (consultez la documentation Ansible pour "Cryptage de contenu avec Ansible Vault") ou utilisez l' `--ask-become-pass` option lors de l'exécution du manuel de vente.

4. Dans le `client_inventory.yml` Fichier, répertorie tous les hôtes qui doivent être configurés comme clients BeeGFS sous `beegfs_clients` Définissez ensuite toute configuration supplémentaire requise pour générer le module de noyau client BeeGFS.

```
  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.
```

Lorsque vous utilisez les pilotes OFED NVIDIA, assurez-vous que `beegfs_client_ofed_include_path` pointe vers le "header include path" correct pour votre installation Linux. Pour plus d'informations, consultez la documentation BeeGFS pour "Prise en charge de RDMA".

5. Dans le `client_inventory.yml` Fichier, répertorie les systèmes de fichiers BeeGFS que vous souhaitez monter au bas de tout ce qui a été défini précédemment `vars`.

```
        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
```

Le `beegfs_client_config` représente les paramètres testés. Reportez-vous à la documentation fournie avec le `netapp_eseries.beegfs` collection `beegfs_client` rôle pour une vue d'ensemble complète de toutes les options. Cela inclut le montage de plusieurs systèmes de fichiers BeeGFS ou le montage du même système de fichiers BeeGFS plusieurs fois.

6. Créer un nouveau `client_playbook.yml` puis remplissez les paramètres suivants :

```
# 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
```

Ignorer l'importation du `netapp_eseries.host` collecte et `ipoib` Rôle si vous avez déjà installé les pilotes IB/RDMA requis et configuré les adresses IP sur les interfaces IPoIB appropriées.

7. Pour installer et construire le client et monter BeeGFS, exécutez la commande suivante :

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

8. Avant de placer le système de fichiers BeeGFS en production, nous vous recommandons **fortement** de vous connecter à n'importe quel client et de l'exécuter `beegfs-fsck --checkfs` afin de garantir que tous les nœuds sont accessibles et qu'aucun problème n'est signalé.

Faites évoluer votre infrastructure au-delà de cinq éléments de base

Vous pouvez configurer Pacemaker et Corosync pour qu'elle dépasse cinq éléments de base (10 nœuds de fichiers). Toutefois, il y a des inconvénients pour les grands clusters, et finalement Pacemaker et Corosync imposent un maximum de 32 nœuds.

NetApp n'a testé que les clusters BeeGFS HA pour jusqu'à 10 nœuds. Il n'est pas recommandé ou pris en charge de faire évoluer des clusters individuels au-delà de cette limite. Toutefois, les systèmes de fichiers BeeGFS nécessitent une évolutivité bien supérieure à 10 nœuds, et NetApp en est responsable dans la solution BeeGFS sur NetApp.

En déployant plusieurs clusters HA contenant un sous-ensemble des éléments de base de chaque système de fichiers, vous pouvez faire évoluer le système de fichiers BeeGFS indépendamment de toutes les limites recommandées ou strictes sur les mécanismes de mise en cluster HA sous-jacents. Dans ce scénario, procédez comme suit :

- Créez un nouvel inventaire Ansible représentant les clusters HA supplémentaires, puis ignorez la configuration d'un autre service de gestion. Pointez plutôt le `beegfs_ha_mgmtd_floating_ip` variable

dans chaque cluster supplémentaire `ha_cluster.yml` Vers l'IP pour le premier service de gestion BeeGFS.

- Lorsque vous ajoutez des clusters haute disponibilité supplémentaires sur le même système de fichiers, vérifiez ce qui suit :
  - Les ID de nœud BeeGFS sont uniques.
  - Les noms de fichiers correspondant à chaque service sous `group_vars` est unique dans tous les clusters.
  - Les adresses IP du client et du serveur BeeGFS sont uniques dans tous les clusters.
  - Le premier cluster HA contenant le service de gestion BeeGFS est exécuté avant de tenter de déployer ou de mettre à jour des clusters supplémentaires.
- Maintenir les inventaires pour chaque cluster HA séparément dans leur propre arborescence de répertoires.

  La combinaison des fichiers d'inventaire de plusieurs clusters dans une arborescence de répertoires peut entraîner des problèmes avec la façon dont le rôle BeeGFS HA agrège la configuration appliquée à un cluster donné.

> Chaque cluster haute disponibilité n'a aucune exigence à évoluer jusqu'à cinq éléments de base avant d'en créer un nouveau. Dans bien des cas, la gestion requiert moins d'éléments de base par cluster est plus simple. Une approche consiste à configurer les éléments de base de chaque rack en tant que cluster haute disponibilité.

Pourcentages de surprovisionnement recommandés pour le pool de stockage

Lorsque vous suivez la configuration standard des quatre volumes par pool de stockage pour les éléments de base de deuxième génération, consultez le tableau suivant.

Ce tableau fournit les pourcentages recommandés à utiliser comme taille de volume dans `eseries_storage_pool_configuration` Pour chaque cible de stockage ou de métadonnées BeeGFS :

Taille du disque	Taille
1,92 TO	18
3,84 TO	21.5
7,68 TO	22.5
15,3 TO	24

> Les recommandations ci-dessus ne s'appliquent pas au pool de stockage contenant le service de gestion, ce qui devrait réduire la taille ci-dessus de .25 % pour allouer 1 % du pool de stockage aux données de gestion.

Pour comprendre comment ces valeurs ont été déterminées, reportez-vous à la section "Tr-4800 : Annexe A : compréhension de la longévité et du sur-provisionnement des disques SSD".

Élément de base haute capacité

Le guide de déploiement de la solution BeeGFS standard décrit les procédures et les recommandations pour répondre aux exigences des workloads de haute performance. Les clients cherchant à répondre aux besoins en capacité élevés doivent observer les variations du déploiement et les recommandations décrites ici.

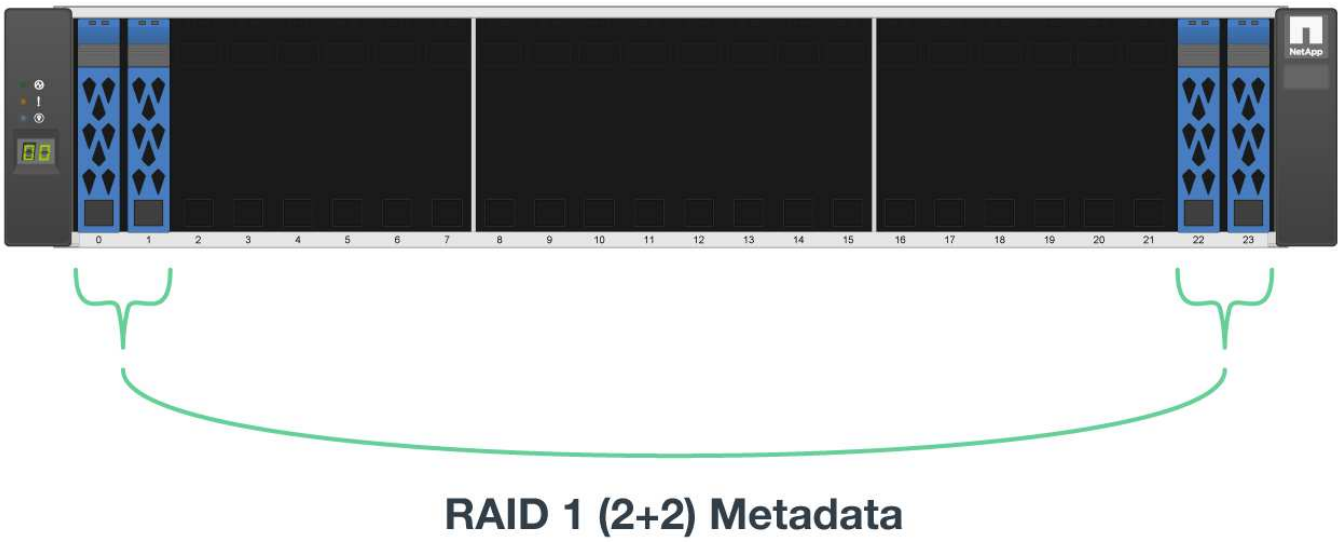
Contrôleurs

Pour les éléments de base haute capacité, les contrôleurs EF600 doivent être remplacés par des contrôleurs EF300, chacun avec une HIC Cascade installée pour l'extension SAS. Chaque nœud de bloc aura un nombre minimal de SSD NVMe dans le boîtier de la baie pour le stockage de métadonnées BeeGFS et sera relié à des tiroirs d'extension dotés de disques durs NL-SAS pour les volumes de stockage BeeGFS.

La configuration du noeud fichier à noeud bloc reste la même.

Placement des disques

Chaque nœud de bloc exige un minimum de 4 SSD NVMe pour le stockage de métadonnées BeeGFS. Ces lecteurs doivent être placés dans les emplacements les plus extérieurs du boîtier.



Bacs d’extension

L’élément de base haute capacité peut être dimensionné avec 1-7, 60 tiroirs d’extension de disque par matrice de stockage.

Pour obtenir des instructions sur le câble de chaque bac d’extension, "Consultez la section câblage EF300 pour les tiroirs disques".