BeeGFS sur NetApp avec E-Series Storage

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Sommaire

BeeGFS sur NetApp avec E-Series Storage

Commencez

Ce qui est inclus dans ce site

Ce site explique comment déployer et gérer BeeGFS sur NetApp à la fois sur les architectures vérifiées NetApp (NVA) et sur les architectures personnalisées. Les designs NVA sont minutieusement testés et fournissent aux clients des configurations de référence et des conseils de dimensionnement afin de réduire les risques de déploiement et d'accélérer la mise sur le marché. NetApp prend également en charge les architectures BeeGFS sous forme de matériel NetApp, ce qui offre à vos clients et partenaires la flexibilité nécessaire à la conception de systèmes de fichiers pour répondre à un large éventail d'exigences. Ces deux approches exploitent Ansible pour le déploiement, et offrent une approche de type appliance pour gérer BeeGFS à n'importe quelle échelle sur un éventail flexible de matériel.

Termes et concepts

Les termes et concepts suivants s'appliquent à la solution BeeGFS sur NetApp.

Pour "Administrer les clusters BeeGFS"plus d'informations sur les termes et concepts propres à l'interaction avec les clusters haute disponibilité BeeGFS, reportez-vous à la section.

Durée	Description
L'IA	L'intelligence artificielle.
Nœud de contrôle Ansible	Machine physique ou virtuelle utilisée pour exécuter l'interface de ligne de commande Ansible.
Inventaire Ansible	Structure de répertoire contenant les fichiers YAML qui sont utilisés pour décrire le cluster BeeGFS HA souhaité.
BMC	Contrôleur de gestion de la carte mère. Parfois appelé processeur de service.
Blocs de nœuds	NetApp E-Series
Clients	Nœuds du cluster HPC exécutant des applications qui doivent utiliser le système de fichiers. Parfois également appelé nœuds de calcul ou nœuds GPU.
DL	Apprentissage profond.
nœuds de fichiers	Serveurs de fichiers BeeGFS.
HAUTE DISPONIBILITÉ	Haute disponibilité.

Durée	Description
HIC	Carte d’interface hôte.
HPC	Informatique hautes performances.
Les workloads de type HPC	Les charges de travail de type HPC se caractérisent généralement par plusieurs nœuds de calcul ou GPU nécessitant l’accès au même dataset en parallèle pour faciliter une tâche de calcul ou d’entraînement distribuée. Ces jeux de données comprennent souvent des fichiers volumineux qui doivent être répartis sur plusieurs nœuds de stockage physique, afin d’éliminer les goulets d’étranglement matériels traditionnels qui empêchent l’accès simultané à un seul fichier.
ML	Apprentissage machine.
NLP	Traitement du langage naturel.
NLU	Compréhension du langage naturel.
NVA	Le programme NetApp Verified Architecture (NVA) propose des configurations de référence et des conseils de dimensionnement pour des charges de travail et des cas d’utilisation spécifiques. Ces solutions sont testées en profondeur, conçues pour réduire les risques de déploiement et accélérer le délai de mise sur le marché.
réseau de stockage/réseau client	Réseau utilisé pour les clients pour communiquer avec le système de fichiers BeeGFS. Il s’agit souvent du même réseau utilisé pour l’interface MPI (Parallel message Passing interface) et d’autres communications d’applications entre les nœuds de cluster HPC.

Utilisez des architectures vérifiées

Présentation et configuration requise

Présentation de la solution

La solution BeeGFS sur NetApp associe le système de fichiers parallèle BeeGFS aux systèmes de stockage NetApp EF600 à une infrastructure fiable, évolutive et économique qui s'adapte aux besoins des workloads les plus exigeants.

Programme NVA

La solution BeeGFS sur NetApp fait partie du programme NVA (NetApp Verified Architecture), qui fournit aux clients des configurations de référence et des conseils de dimensionnement pour des workloads et des cas d'utilisation spécifiques. Les solutions NVA sont minutieusement testées et conçues pour réduire les risques de déploiement et accélérer le délai de mise sur le marché.

Présentation de la conception

La solution BeeGFS sur NetApp est une architecture modulaire qui peut être configurée pour de nombreux workloads exigeants. Que ce soit pour gérer de nombreux fichiers de petite taille, des opérations de fichiers volumineux ou une charge de travail hybride, le système de fichiers peut être personnalisé pour répondre à ces besoins. Grâce à une structure matérielle à deux niveaux, la haute disponibilité est intégrée. Elle permet un basculement indépendant sur plusieurs couches matérielles et garantit des performances prévisibles, même en cas de dégradation partielle du système. Le système de fichiers BeeGFS permet de créer un environnement haute performance et évolutif sur différentes distributions Linux. Il offre aux clients un seul namespace de stockage facilement accessible. Pour en savoir plus, consultez le "présentation de l'architecture".

Cas d'utilisation

Les utilisations suivantes s'appliquent à la solution BeeGFS sur NetApp :

- Systèmes NVIDIA DGX SuperPOD équipés de DGX avec DGX A100, H100, H200 et B200 GPU.
- L'intelligence artificielle (IA), comprenant le machine learning (ML), le deep learning (DL), le traitement du langage naturel à grande échelle (NLP) et la compréhension du langage naturel (NLU). Pour plus d'informations, voir "BeeGFS pour l'IA : faits plutôt que fiction".
- Informatique hautes performances (HPC), y compris les applications accélérées par MPI (interface de transmission de messages) et d'autres techniques informatiques distribuées. Pour plus d'informations, voir "Pourquoi BeeGFS va bien au-delà de l'HPC".
- Charges de travail applicatives caractérisées par :
  - Lecture ou écriture dans des fichiers supérieurs à 1 Go
  - Lecture ou écriture dans le même fichier par plusieurs clients (dizaines, centaines et milliers)
- Jeux de données de plusieurs téraoctets ou de plusieurs pétaoctets.
- Environnements qui nécessitent un seul espace de noms de stockage optimal pour un mélange de fichiers de petite ou de grande taille.

Avantages

Voici les principaux avantages de BeeGFS sur NetApp :

- La disponibilité de conceptions matérielles vérifiées permet l'intégration complète des composants matériels et logiciels pour assurer des performances prévisibles et une fiabilité optimale.
- Déploiement et gestion avec Ansible pour une simplicité et une cohérence à grande échelle.
- Contrôle et observabilité fournis avec l'analyseur de performance E-Series et le plug-in BeeGFS. Pour plus d'informations, voir "Présentation d'un cadre de surveillance des solutions NetApp E-Series".
- Haute disponibilité dotée d'une architecture de disques partagés qui assure la durabilité et la disponibilité des données.
- Prise en charge des fonctionnalités modernes de gestion et d'orchestration des workloads à l'aide de conteneurs et de Kubernetes. Pour plus d'informations, voir "Kubernetes et BeeGFS : un récit d'investissement pérenne".

Présentation de l'architecture

La solution BeeGFS sur NetApp inclut des critères de conception architecturale qui permettent de déterminer l'équipement, le câblage et les configurations qui sont requis pour prendre en charge les workloads validés.

Architecture modulaire

Le système de fichiers BeeGFS peut être déployé et adapté de différentes manières, en fonction des besoins en stockage. Par exemple, certains cas d'utilisation mettant en avant de nombreux fichiers de petite taille bénéficieront d'une performance et d'une capacité supplémentaires de métadonnées, tandis que les cas d'utilisation comportant moins de fichiers volumineux peuvent favoriser une capacité de stockage et des performances supérieures pour le contenu réel des fichiers. Ces considérations ont un impact sur les différentes dimensions du déploiement d'un système de fichiers parallèle, ce qui ajoute de la complexité à la conception et au déploiement d'un système de fichiers.

En réponse à ces défis, NetApp a conçu une architecture d'éléments de base standard qui permet une évolutivité horizontale de chaque catégorie. De façon générale, les éléments de base BeeGFS sont déployés dans l'un des trois profils de configuration suivants :

- Un élément de base unique, incluant la gestion BeeGFS, les métadonnées et les services de stockage
- Des métadonnées BeeGFS plus un élément de base du stockage
- Un élément de base de stockage BeeGFS uniquement

Le seul changement matériel entre ces trois options est l'utilisation de lecteurs plus petits pour les métadonnées BeeGFS. Dans le cas contraire, toutes les modifications de configuration sont appliquées via le logiciel. En outre, avec Ansible comme moteur de déploiement, la configuration du profil souhaité pour un élément de base particulier simplifie les tâches de configuration.

Pour plus de détails, voir Conception matérielle vérifiée.

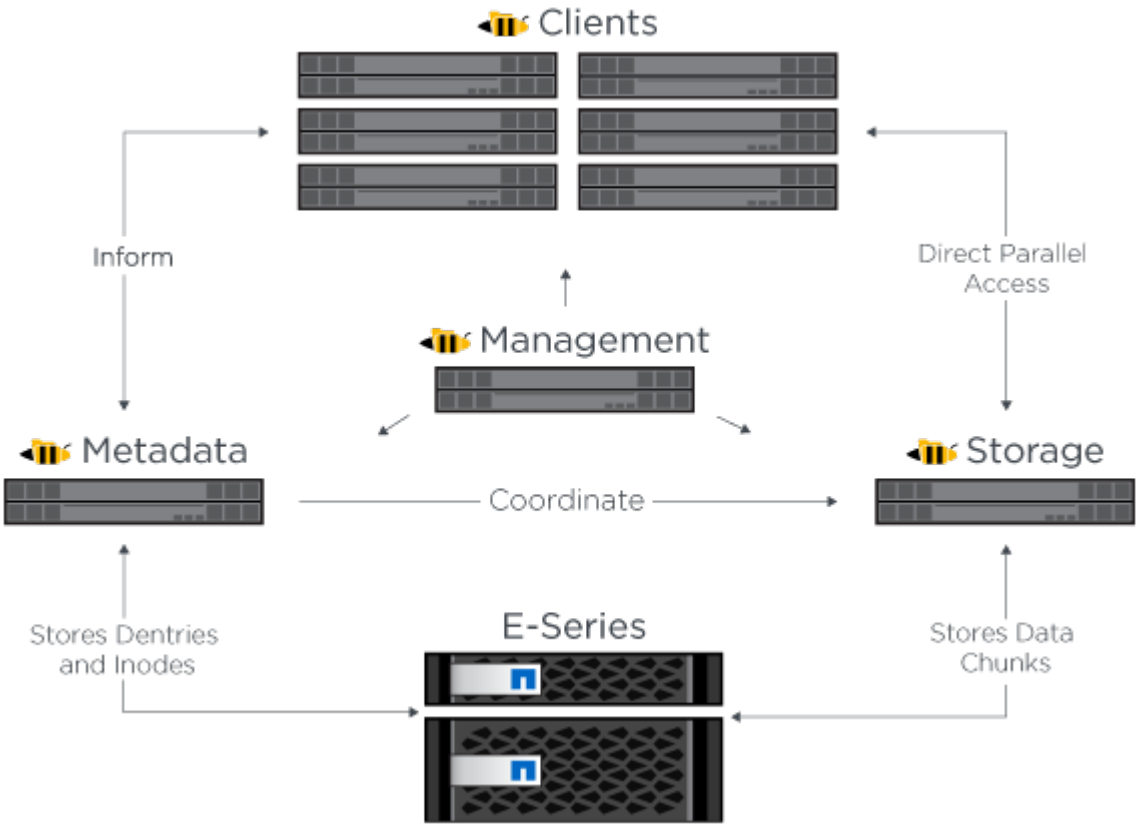
Services de système de fichiers

Le système de fichiers BeeGFS inclut les principaux services suivants :

- **Service de gestion.** registres et contrôle tous les autres services.

- **Service de stockage.** stocke le contenu des fichiers d'utilisateur distribués appelé fichiers de bloc de données.
- **Service de métadonnées.** assure le suivi de la disposition du système de fichiers, du répertoire, des attributs de fichier, etc.
- **Service client.** monte le système de fichiers pour accéder aux données stockées.

La figure suivante présente les composants et les relations de la solution BeeGFS utilisés avec les systèmes NetApp E-Series.



En tant que système de fichiers parallèle, BeeGFS répartit ses fichiers sur plusieurs nœuds de serveur afin de maximiser les performances en lecture/écriture et l'évolutivité. Les nœuds de serveur fonctionnent ensemble pour fournir un système de fichiers unique pouvant être monté et accessible simultanément par d'autres nœuds de serveur, communément appelés *clients*. Ces clients peuvent voir et consommer le système de fichiers distribué de la même manière qu'un système de fichiers local tel que NTFS, XFS ou ext4.

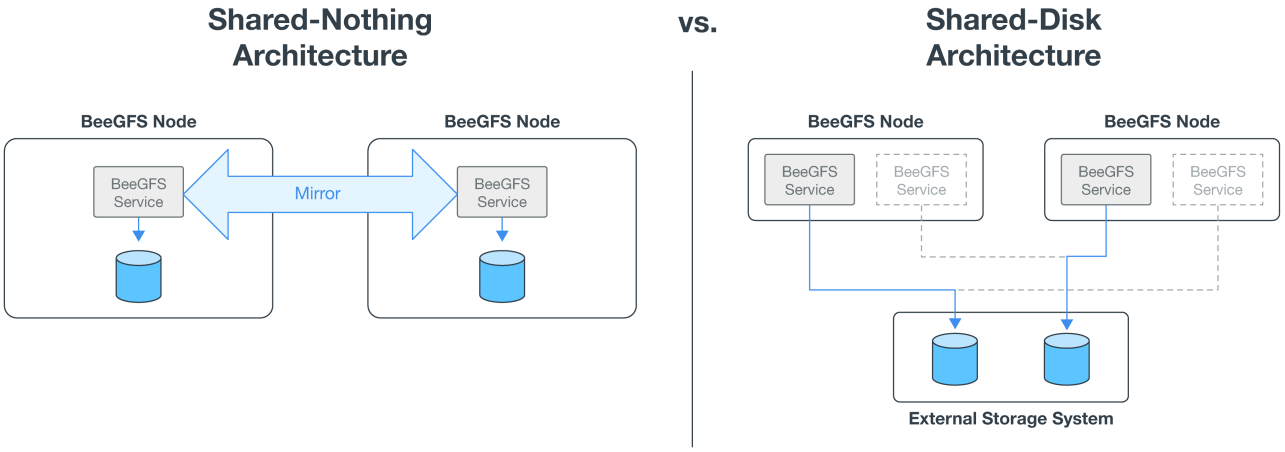
Les quatre services principaux fonctionnent sur un large éventail de distributions Linux prises en charge et communiquent via n'importe quel réseau compatible TCP/IP ou RDMA, y compris InfiniBand (IB), Omni-Path (OPA) et RDMA over Converged Ethernet (RoCE). Les services de serveur BeeGFS (gestion, stockage et métadonnées) sont des démons d'espace utilisateur, alors que le client est un module de noyau natif (sans patchless). Tous les composants peuvent être installés ou mis à jour sans redémarrage. Vous pouvez en outre exécuter n'importe quelle combinaison de services sur le même nœud.

Architecture HAUTE DISPONIBILITÉ

BeeGFS sur NetApp étend les fonctionnalités de la version BeeGFS Enterprise en créant une solution entièrement intégrée avec du matériel NetApp qui offre une architecture haute disponibilité (HA) de disque partagé.

L'édition communautaire BeeGFS peut être utilisée gratuitement. Cependant, l'édition entreprise exige l'achat d'un contrat d'abonnement de support professionnel auprès d'un partenaire comme NetApp. L'édition entreprise permet d'utiliser plusieurs fonctions supplémentaires, notamment la résilience, l'application de quotas et les pools de stockage.

La figure suivante compare les architectures haute disponibilité sans partage et à disque partagé.



Pour plus d'informations, voir "Annonce de la haute disponibilité pour BeeGFS prise en charge par NetApp".

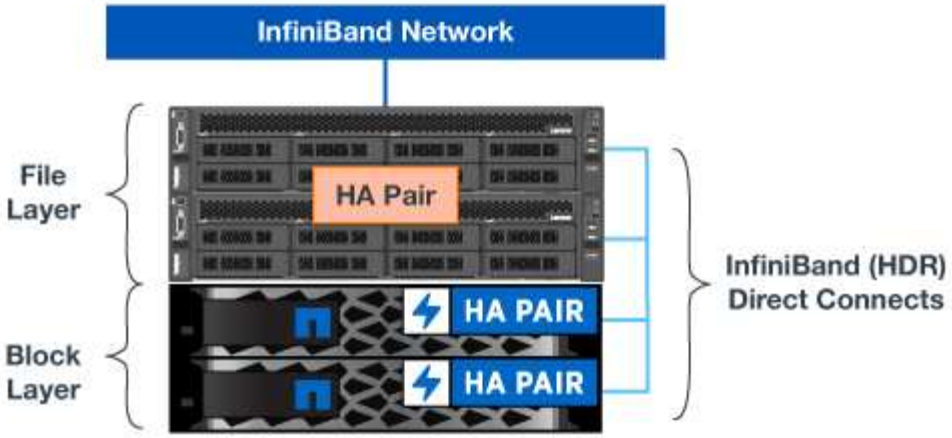
Nœuds vérifiés

La solution BeeGFS sur NetApp a vérifié les nœuds répertoriés ci-dessous.

Nœud	Sous-jacent	Détails
Bloc	Système de stockage EF600 de NetApp	Une baie de stockage 2U 100 % NVMe haute performance conçue pour les workloads exigeants
Fichier	Serveur Lenovo ThinkSystem SR665 V3	Serveur 2U à deux sockets avec PCIe 5.0, deux processeurs AMD EPYC 9124. Pour plus d'informations sur le Lenovo SR665 V3, reportez-vous à la section "Site Web de Lenovo".
	Serveur Lenovo ThinkSystem SR665	Serveur 2U à deux sockets avec PCIe 4.0, deux processeurs AMD EPYC 7003. Pour plus d'informations sur le Lenovo SR665, reportez-vous à la section "Site Web de Lenovo".

Conception matérielle vérifiée

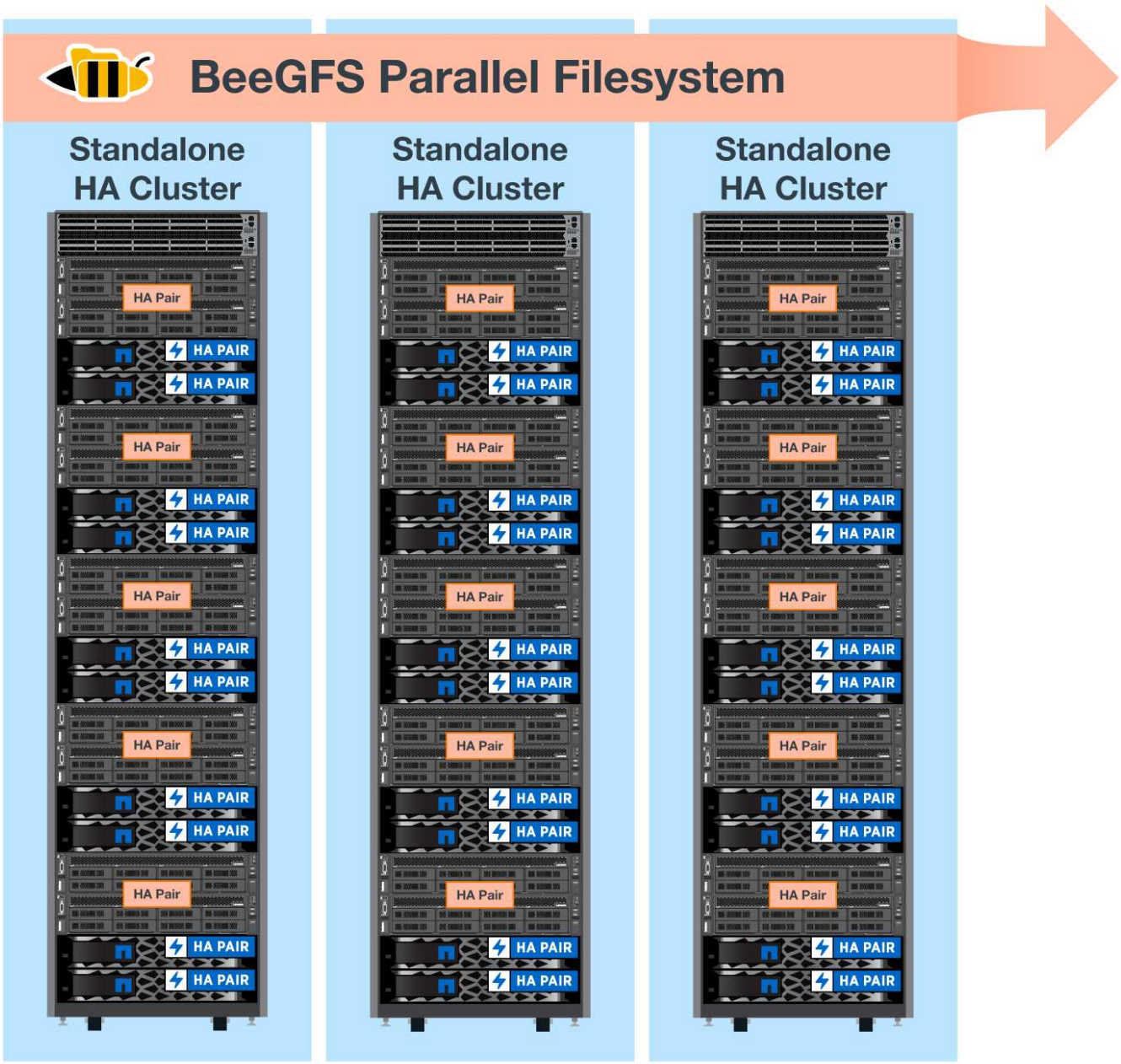
Les éléments de base de la solution (illustrés dans la figure suivante) utilisent les serveurs de nœuds de fichiers vérifiés pour la couche de fichiers BeeGFS et deux systèmes de stockage EF600 comme couche bloc.



La solution BeeGFS sur NetApp s'exécute sur tous les éléments de base du déploiement. Le premier élément de base déployé doit exécuter les services de gestion, de métadonnées et de stockage BeeGFS (également appelés éléments de base). Tous les éléments de base suivants peuvent être configurés via le logiciel pour étendre les métadonnées et les services de stockage, ou pour fournir des services de stockage exclusivement. Cette approche modulaire permet de faire évoluer le système de fichiers en fonction des besoins d'une charge de travail, tout en utilisant les mêmes plateformes matérielles sous-jacentes et la même conception d'éléments de base.

Il est possible de déployer jusqu'à cinq éléments de base pour former un cluster Linux HA autonome. Cela optimise la gestion des ressources avec Pacemaker et maintient une synchronisation efficace avec Corosync. Un ou plusieurs de ces clusters haute disponibilité BeeGFS autonomes sont combinés pour créer un système de fichiers BeeGFS accessible aux clients comme un seul namespace de stockage. Côté matériel, un seul rack 42U peut accueillir jusqu'à cinq éléments de base, ainsi que deux commutateurs InfiniBand 1U pour le réseau de stockage/données. Voir le graphique ci-dessous pour une représentation visuelle.

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.



Ansible

BeeGFS sur NetApp est fourni et déployé à l'aide d'Ansible Automation, qui est hébergé sur GitHub et Ansible Galaxy (la collection BeeGFS est disponible sur "Galaxy Ansible" et "NetApp E-Series GitHub"). Bien qu'Ansible soit principalement testé avec le matériel utilisé pour assembler les éléments de base BeeGFS, vous pouvez le configurer de sorte qu'il s'exécute sur presque tous les serveurs x86 à l'aide d'une distribution Linux prise en charge.

Pour plus d'informations, voir "Déploiement de BeeGFS avec E-Series Storage".

Exigences techniques

Pour implémenter la solution BeeGFS sur NetApp, assurez-vous que votre environnement répond aux exigences technologiques indiquées dans ce document.

Configuration matérielle requise

Avant de commencer, assurez-vous que votre matériel répond aux spécifications suivantes pour un design modulaire deuxième génération de la solution BeeGFS sur NetApp. Les composants exacts d'un déploiement particulier peuvent varier en fonction des besoins du client.

Quantité	Composant matériel	De formation
2	Nœuds de fichiers BeeGFS	Pour atteindre les performances attendues, chaque nœud de fichier doit satisfaire ou dépasser les spécifications des nœuds de fichiers recommandés. **Options de noeud de fichier recommandées :** • **Lenovo ThinkSystem SR665 V3** ◦ **Processeurs:** 2x AMD EPYC 9124 16C 3.0 GHz (configurés comme deux zones NUMA). ◦ **Mémoire :** 256 Go (16 x 16 Go TruDDR5 4800 MHz RDIMM-A) ◦ **Extension PCIe :** quatre emplacements PCIe Gen5 x16 (deux par zone NUMA) ◦ **Divers:** ▪ Deux disques en RAID 1 pour le système d'exploitation (1 To 7200 tr/min SATA) ▪ Port 1 GbE pour la gestion du système d'exploitation intrabande ▪ BMC 1GbE avec API Redfish pour la gestion des serveurs hors bande ▪ Deux blocs d'alimentation remplaçables à chaud et ventilateurs haute performance
2	Nœuds de bloc E-Series (baie EF600)	**Mémoire :** 256 Go (128 Go par contrôleur). **Adaptateur :** 2 ports 200 Go/HDR (NVMe/IB). **Lecteurs :** configurés pour correspondre aux métadonnées et à la capacité de stockage souhaitées.
8	Adaptateurs de carte hôte InfiniBand (pour les nœuds de fichiers).	Les adaptateurs de carte hôte peuvent varier en fonction du modèle de serveur du nœud de fichier. Recommandations pour les nœuds de fichiers vérifiés : • **Lenovo ThinkSystem SR665 V3 Server:** ◦ MCX755106AS-HEAT ConnectX-7, NDR200, QSFP112, 2 ports, PCIe Gen5 x16, adaptateur InfiniBand
1	Switch réseau de stockage	Le commutateur du réseau de stockage doit offrir une vitesse InfiniBand 200 Gbit/s. Modèles de commutateurs recommandés : • **Commutateur NVIDIA QM9700 Quantum 2 NDR InfiniBand** • **Commutateur NVIDIA MQM8700 Quantum HDR InfiniBand**

Exigences de câblage

Connexions directes des nœuds de bloc aux nœuds de fichier.

Quantité	Référence	Longueur
8	MCP1650-H001E30 (câble en cuivre passif NVIDIA, QSFP56, 200 Gbit/s)	1 m

Connexions entre les nœuds de fichiers et le commutateur de réseau de stockage. Sélectionnez l'option de câble appropriée dans le tableau suivant en fonction de votre commutateur de stockage InfiniBand. + la longueur de câble recommandée est de 2 M. toutefois, elle peut varier en fonction de l'environnement du client.

Changer de modèle	Type de câble	Quantité	Référence
NVIDIA QM9700	Fibre active (émetteurs-récepteurs inclus)	2	MMA4Z00-NS (multimode, IB/ETH, 800 Go/s 2x400 Go/s double port OSFP)
		4	MFP7E20-Nxxx (multimode, câble fibre de séparation 4 canaux à deux canaux)
		8	MMA1Z00-NS400 (multimode, IB/ETH, 400 Go/s, QSFP-112 à port unique)
	Cuivre passif	2	MCP7Y40-N002 (câble répartiteur en cuivre passif NVIDIA, InfiniBand 800 Go/s à 4 200 Go/s, OSFP à 4 x QSFP112)
NVIDIA MQM8700	Fibre active	8	MFS1S00-H003E (câble fibre active NVIDIA, InfiniBand 200 Gbit/s, QSFP56)
	Cuivre passif	8	MCP1650-H002E26 (câble en cuivre passif NVIDIA, InfiniBand 200 Gbit/s, QSFP56)

Configuration logicielle et firmware requise

Pour assurer des performances et une fiabilité prévisibles, les versions de la solution BeeGFS sur NetApp sont testées avec des versions spécifiques des composants logiciels et de firmware. Ces versions sont requises pour l'implémentation de la solution.

Configuration requise pour les nœuds de fichiers

Logiciel	Version
Red Hat Enterprise Linux (RHEL)	Serveur physique RHEL 9.4 avec haute disponibilité (2 sockets). **Remarque :** les nœuds de fichiers nécessitent un abonnement Red Hat Enterprise Linux Server valide et le module complémentaire Red Hat Enterprise Linux High Availability.
Noyau Linux	5.14.0-427.42.1.el9_4.x86_64
Micrologiciel HCA	**Micrologiciel ConnectX-7 HCA** Micrologiciel : 28.45.1200 + PXE : 3.7.0500 + UEFI : 14.38.0016 **Micrologiciel ConnectX-6 HCA** Micrologiciel : 20.43.2566 + PXE : 3.7.0500 + UEFI : 14.37.0013

Exigences liées aux nœuds en mode bloc EF600

Logiciel	Version
SANtricity OS	11.90R3
NVSRAM	N6000-890834-D02.dlp
Micrologiciel de lecteur	Dernière version disponible pour les modèles de lecteurs utilisés. Voir la "Site du firmware du disque E-Series".

Configuration requise pour le déploiement de logiciels

Le tableau suivant répertorie les exigences logicielles déployées automatiquement dans le cadre du déploiement BeeGFS basé sur Ansible.

Logiciel	Version
BeeGFS	7.4.6
Corosync	3.1.8-1
Stimulateur cardiaque	2.1.7-5,2
PCS	0.11.7-2
Agents de clôture (sébaste/apc)	4.10.0-62
Pilotes InfiniBand / RDMA	MLNX_OFED_LINUX-23.10-3.2.2.1-LTS

Configuration requise pour le nœud de contrôle Ansible

La solution BeeGFS sur NetApp est déployée et gérée à partir d'un nœud de contrôle Ansible. Pour plus d'informations, reportez-vous à la section "Documentation Ansible".

Les exigences logicielles répertoriées dans les tableaux suivants sont spécifiques à la version de la collection NetApp BeeGFS Ansible indiquée ci-dessous.

Logiciel	Version
Ansible	10.x
Cœur Ansible	>= 2.13.0
Python	3,10
Packs Python supplémentaires	Cryptographie-43.0.0, netaddr-1.3.0, ipaddr-2.2.0
Collection Ansible NetApp E-Series BeeGFS	3.2.0

Examen du design de la solution

Présentation du design

Pour prendre en charge la solution BeeGFS sur NetApp, qui associe le système de fichiers parallèle BeeGFS et les systèmes de stockage NetApp EF600, vous devez utiliser un équipement, un câblage et des configurations spécifiques.

En savoir plus :

- "Configuration matérielle"
- "Configuration logicielle"
- "Vérification de la conception"
- "Instructions de dimensionnement"
- "Réglage des performances"

Architectures dérivées avec des variations de conception et de performances :

- "Élément de base haute capacité"

Configuration matérielle

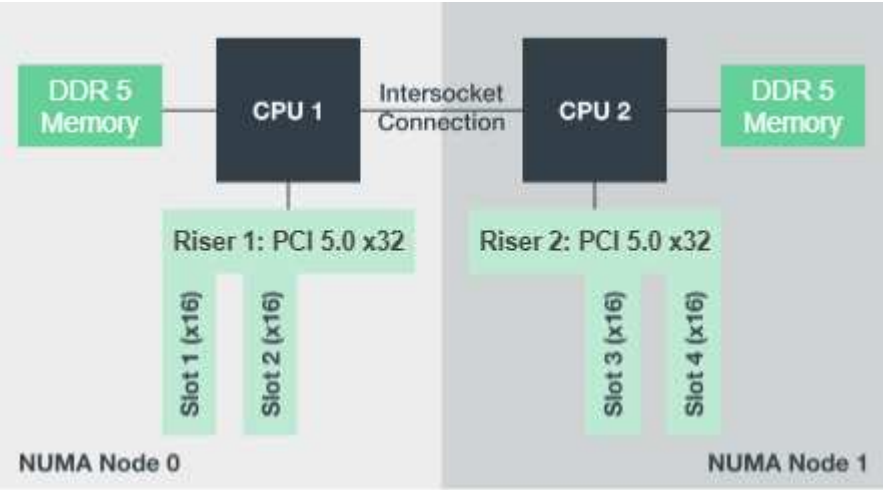
La configuration matérielle de BeeGFS sur NetApp inclut des nœuds de fichiers et le câblage réseau.

Configuration de nœud de fichiers

Les nœuds de fichiers ont deux sockets de CPU configurés en zones NUMA distinctes, qui incluent un accès local à un nombre égal de slots PCIe et de mémoire.

Les adaptateurs InfiniBand doivent être placés dans les connecteurs ou les cartes de montage PCI appropriés, de sorte que la charge de travail soit équilibrée sur les voies PCIe et les canaux de mémoire disponibles. Pour équilibrer la charge de travail, vous pouvez isoler intégralement le travail des services BeeGFS vers un nœud NUMA particulier. L'objectif est d'atteindre les mêmes performances pour chaque nœud de fichiers que s'il s'agissait de deux serveurs à socket unique indépendants.

La figure suivante montre la configuration NUMA du nœud de fichiers.



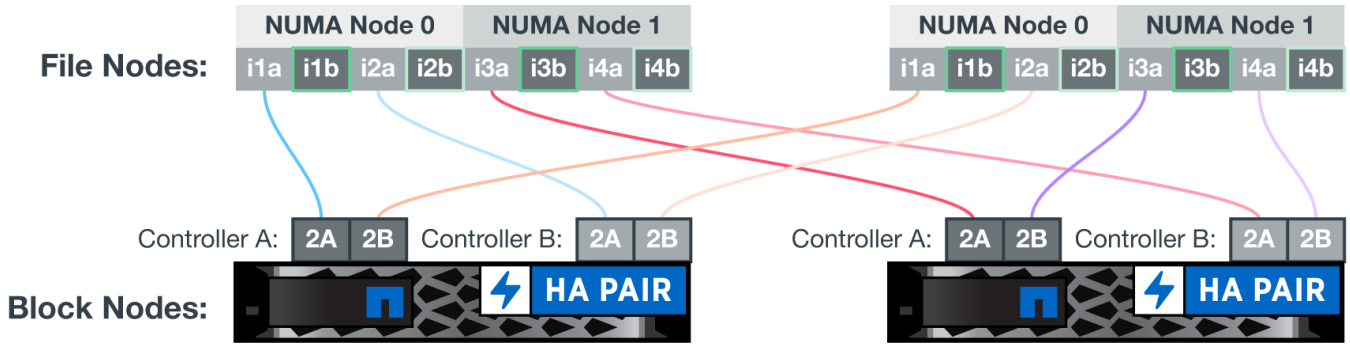
Les processus BeeGFS sont épinglés à une zone NUMA particulière pour s'assurer que les interfaces utilisées se trouvent dans la même zone. Cette configuration évite d'avoir besoin d'un accès à distance via la connexion inter-socket. La connexion inter-sockets est parfois appelée liaison QPI ou GMI2 ; même dans les architectures de processeurs modernes, ils peuvent être un goulot d'étranglement lors de l'utilisation de réseaux haut débit comme HDR InfiniBand.

Configuration des câbles réseau

Dans un élément de base, chaque nœud de fichier est connecté à deux nœuds de bloc grâce à quatre connexions InfiniBand redondantes. En outre, chaque nœud de fichiers dispose de quatre connexions redondantes au réseau de stockage InfiniBand.

Dans la figure suivante, notez que :

- Tous les ports de nœuds de fichiers indiqués en vert sont utilisés pour la connexion au maillage Storage Fabric ; tous les autres ports de nœuds de fichiers sont les connexions directes aux nœuds de blocs.
- Deux ports InfiniBand d'une zone NUMA spécifique se connectent aux contrôleurs A et B du même nœud de bloc.
- Les ports du nœud NUMA 0 se connectent toujours au premier nœud de bloc.
- Les ports du nœud NUMA 1 se connectent au second nœud de bloc.



Lors de l'utilisation de câbles de séparation pour connecter le commutateur de stockage aux nœuds de fichiers, un câble doit se brancher et se connecter aux ports indiqués en vert clair. Un autre câble doit se brancher 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.

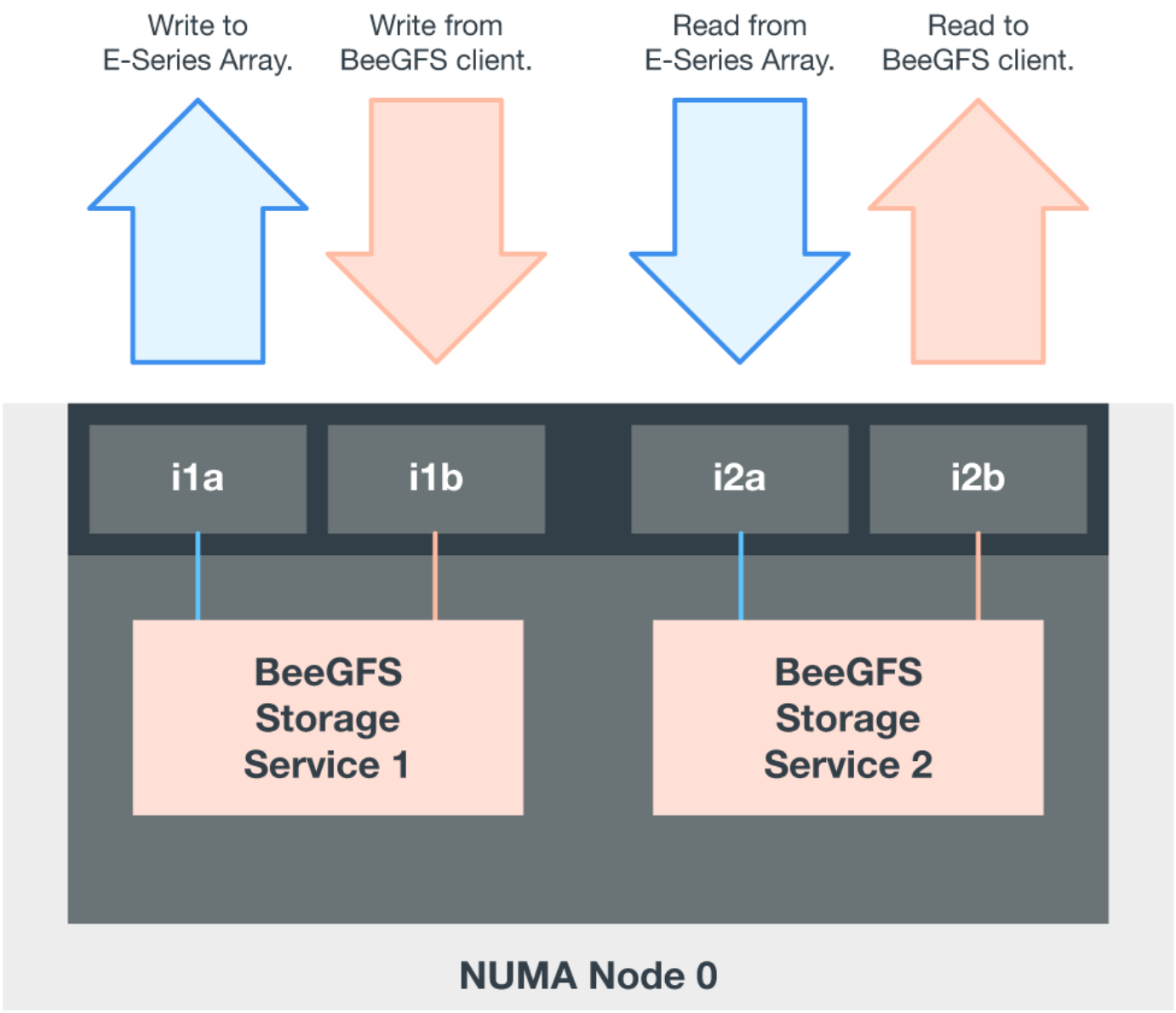
La configuration de câblage illustrée dans la figure permet à chaque service BeeGFS de :

- Exécuter dans la même zone NUMA, quel que soit le nœud de fichier qui exécute le service BeeGFS.
- Disposer de chemins optimaux secondaires au réseau de stockage frontal et aux nœuds de blocs internes, quel que soit l'endroit où une défaillance se produit
- Réduisez l'impact sur la performance si un nœud de fichiers ou un contrôleur d'un nœud de blocs nécessite une maintenance.

Câblage pour exploiter la bande passante

Pour exploiter la bande passante bidirectionnelle PCIe complète, vérifiez que un port de chaque adaptateur InfiniBand se connecte à la structure de stockage, et que l'autre port est connecté à un nœud de bloc.

La figure suivante montre la conception de câblage utilisée pour exploiter la bande passante bidirectionnelle PCIe complète.



Pour chaque service BeeGFS, utilisez la même carte pour connecter le port préféré utilisé pour le trafic client avec le chemin vers le contrôleur de nœuds de bloc qui est le principal propriétaire de ces volumes de services. Pour plus d'informations, voir "Configuration logicielle".

Configuration logicielle

La configuration logicielle de BeeGFS sur NetApp inclut des composants réseau BeeGFS, des nœuds de bloc EF600, des nœuds de fichiers BeeGFS, des groupes de ressources et des services BeeGFS.

Configuration réseau BeeGFS

La configuration du réseau BeeGFS comprend les composants suivants.

- **IP flottantes** les adresses IP flottantes sont un type d'adresse IP virtuelle qui peut être routée

dynamiquement vers n'importe quel serveur du même réseau. Plusieurs serveurs peuvent posséder la même adresse IP flottante, mais elle ne peut être active que sur un seul serveur à la fois.

Chaque service de serveur BeeGFS possède sa propre adresse IP qui peut se déplacer entre les nœuds de fichiers en fonction de l'emplacement d'exécution du service de serveur BeeGFS. Cette configuration IP flottante permet à chaque service de basculer indépendamment vers l'autre nœud de fichiers. Le client a simplement besoin de connaître l'adresse IP d'un service BeeGFS particulier; il n'est pas nécessaire de savoir quel nœud de fichier exécute actuellement ce service.

- **Configuration multi-homing du serveur BeeGFS** pour augmenter la densité de la solution, chaque nœud de fichiers a plusieurs interfaces de stockage avec des adresses IP configurées dans le même sous-réseau IP.

  Des configurations supplémentaires sont nécessaires pour s'assurer que cette configuration fonctionne comme prévu avec la pile réseau Linux, car par défaut, les requêtes à une interface peuvent être traitées sur une autre interface si leurs adresses IP se trouvent dans le même sous-réseau. Outre d'autres inconvénients, ce comportement par défaut rend impossible l'établissement ou la maintenance des connexions RDMA.

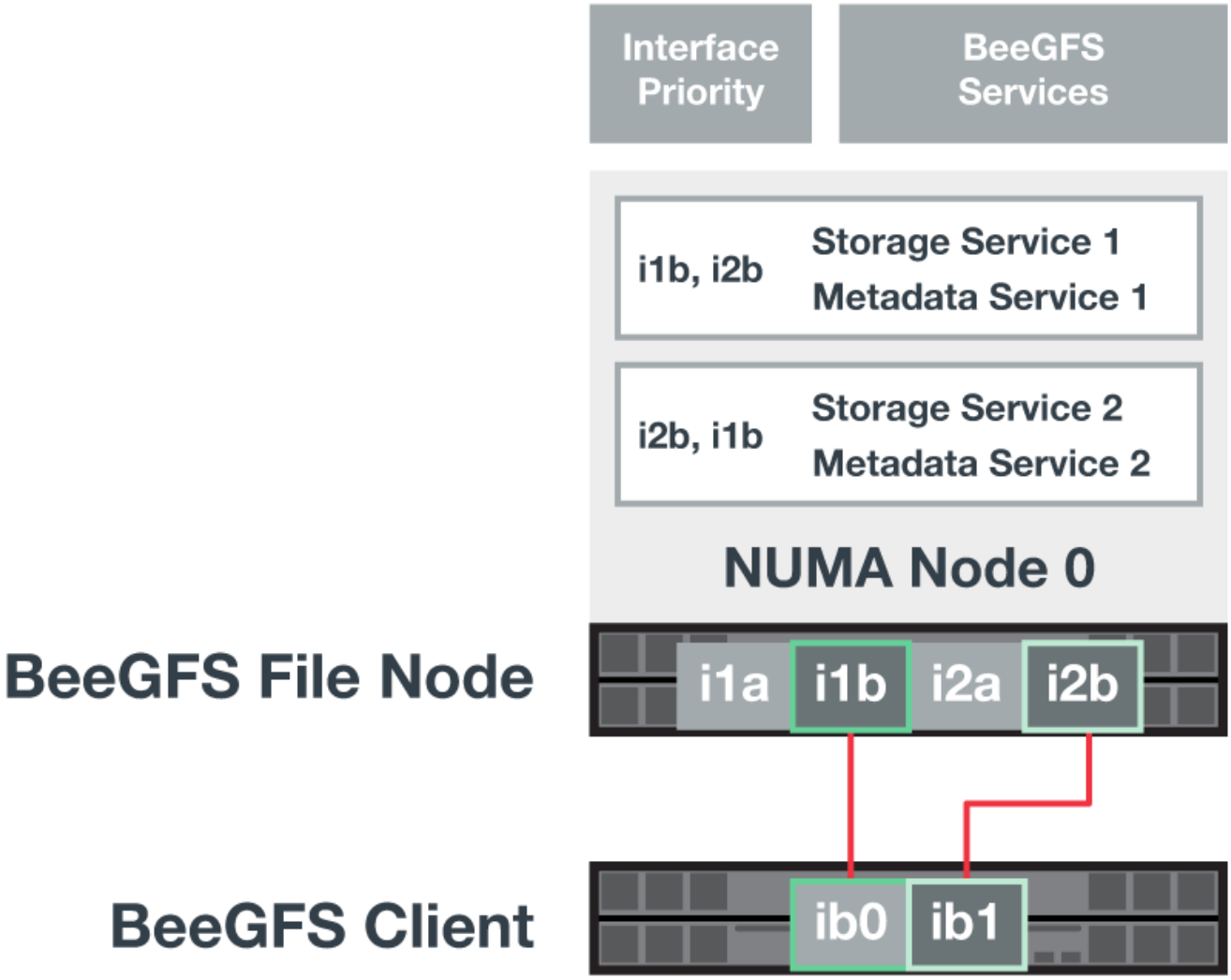
  Le déploiement Ansible gère le serrage du comportement de la RP (reverse path) et du protocole ARP (Address Resolution Protocol), ainsi que la vérification du démarrage et de l'arrêt d'adresses IP flottantes ; les routes et règles IP correspondantes sont créées dynamiquement pour permettre à la configuration réseau multihomée de fonctionner correctement.

- **La configuration multirail du client BeeGFS** *Multi-rail* fait référence à la capacité d'une application à utiliser plusieurs connexions réseau indépendantes, ou « rails », pour améliorer les performances.

  BeeGFS implémente la prise en charge multirail afin de permettre l'utilisation de plusieurs interfaces IB dans un seul sous-réseau IPoIB. Cette fonctionnalité permet notamment l'équilibrage dynamique de la charge entre les cartes réseau RDMA, optimisant ainsi l'utilisation des ressources du réseau. Il s'intègre également au système de stockage NVIDIA GPUDirect (GDS), qui offre une bande passante système accrue et réduit la latence et l'utilisation sur le processeur du client.

  Cette documentation fournit des instructions pour les configurations de sous-réseau IPoIB uniques. Les configurations de sous-réseau Dual IPoIB sont prises en charge, mais ne fournissent pas les mêmes avantages que les configurations à sous-réseau unique.

  La figure suivante montre l'équilibrage du trafic sur plusieurs interfaces client BeeGFS.



Comme chaque fichier de BeeGFS est généralement réparti sur plusieurs services de stockage, la configuration multi-rail permet au client d'atteindre un débit supérieur à celui d'un seul port InfiniBand. Par exemple, l'exemple de code suivant montre une configuration commune de répartition des fichiers qui permet au client d'équilibrer le trafic entre les deux interfaces :

+

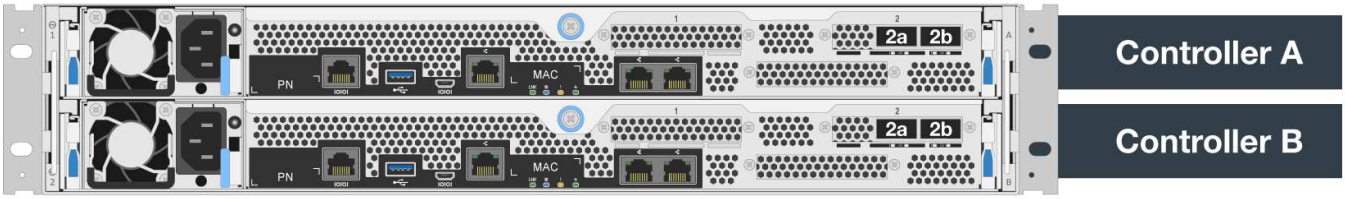
```
root@beegfs01:/mnt/beegfs# beegfs-ctl --getentryinfo myfile
Entry type: file
EntryID: 11D-624759A9-65
Metadata node: meta_01_tgt_0101 [ID: 101]
Stripe pattern details:
+ Type: RAID0
+ Chunksize: 1M
+ Number of storage targets: desired: 4; actual: 4
+ Storage targets:
  + 101 @ stor_01_tgt_0101 [ID: 101]
  + 102 @ stor_01_tgt_0101 [ID: 101]
  + 201 @ stor_02_tgt_0201 [ID: 201]
  + 202 @ stor_02_tgt_0201 [ID: 201]
```

Configuration de nœud en mode bloc EF600

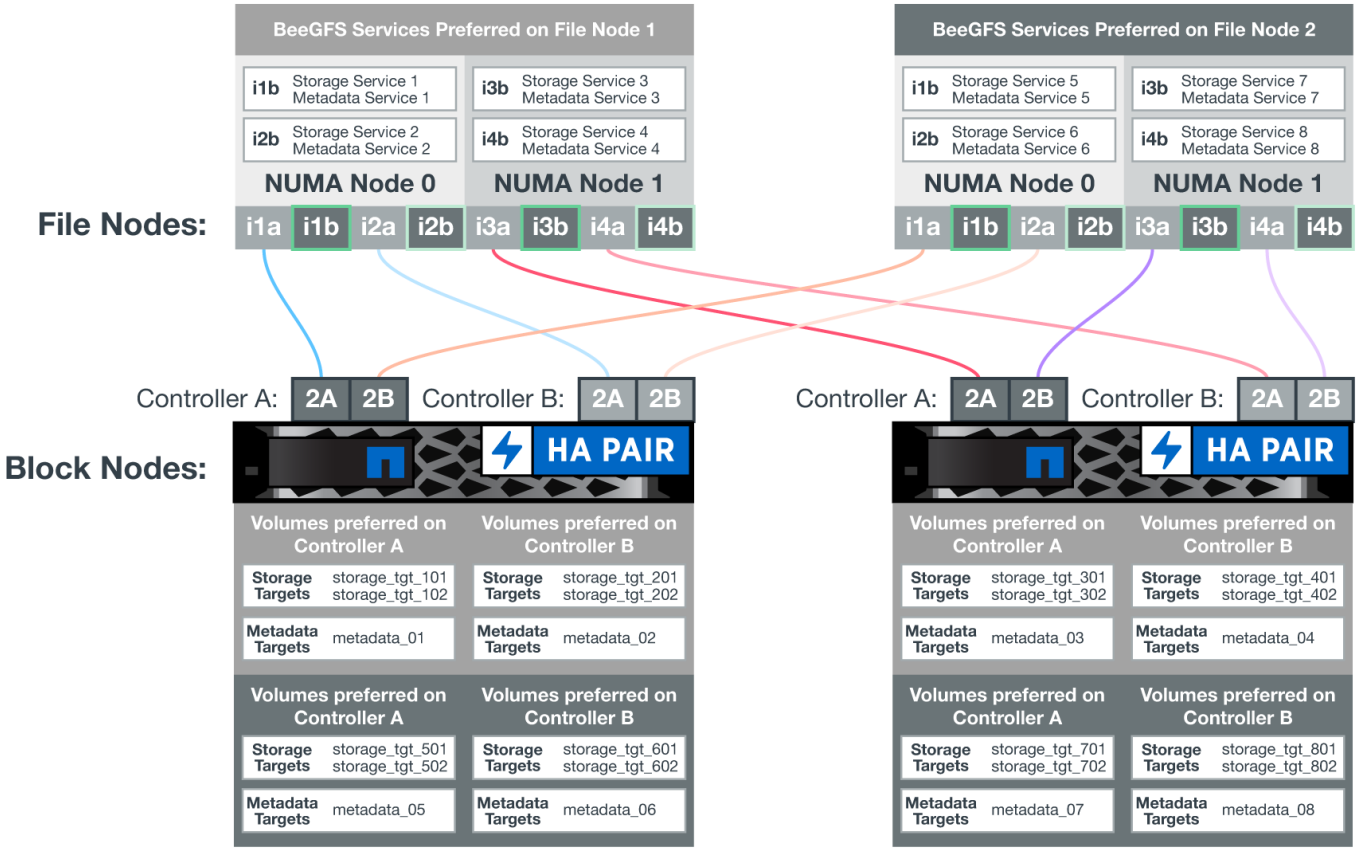
Les nœuds de blocs comprennent deux contrôleurs RAID actifs/actifs avec accès partagé au même ensemble de lecteurs. En général, chaque contrôleur possède la moitié des volumes configurés sur le système, mais il peut reprendre l'autre contrôleur si nécessaire.

Le logiciel de chemins d'accès multiples des nœuds de fichiers détermine le chemin actif et optimisé vers chaque volume et le déplace automatiquement vers l'autre chemin d'accès en cas de défaillance du câble, de l'adaptateur ou du contrôleur.

Le schéma suivant illustre la disposition du contrôleur dans les nœuds de bloc EF600.



Pour faciliter la solution haute disponibilité du disque partagé, les volumes sont mappés sur les deux nœuds de fichiers de manière à ce qu'ils puissent prendre en charge les uns les autres selon les besoins. Le diagramme suivant montre un exemple de configuration du service BeeGFS et de la propriété du volume préféré pour des performances maximales. L'interface à gauche de chaque service BeeGFS indique l'interface préférée que les clients et les autres services utilisent pour le contacter.



Dans l'exemple précédent, les clients et les services serveur préfèrent communiquer avec le service de stockage 1 via l'interface i1b. Le service de stockage 1 utilise l'interface i1a comme chemin préféré pour communiquer avec ses volumes (Storage_tgt_101, 102) sur le contrôleur A du premier nœud de bloc. Cette configuration utilise la bande passante PCIe bidirectionnelle complète disponible pour l'adaptateur InfiniBand et offre de meilleures performances avec un adaptateur HDR InfiniBand à deux ports que le tout avec PCIe 4.0.

Configuration de nœud de fichier BeeGFS

Les nœuds de fichiers BeeGFS sont configurés dans un cluster haute disponibilité (HA) pour faciliter le basculement des services BeeGFS entre plusieurs nœuds de fichiers.

La conception du cluster HA repose sur deux projets Linux HA largement utilisés : Corosync pour l'appartenance à un cluster et Pacemaker pour la gestion des ressources de cluster. Pour plus d'informations, voir "Formation Red Hat pour les modules complémentaires haute disponibilité".

NetApp a rédigé et étendu plusieurs agents de ressources OCF (Open Cluster Framework) pour permettre au cluster de démarrer et de surveiller intelligemment les ressources BeeGFS.

Clusters HA BeeGFS

De façon générale, lorsque vous démarrez un service BeeGFS (avec ou sans HA), quelques ressources doivent être en place :

- Adresses IP où le service est accessible, généralement configurées par Network Manager.
- Les systèmes de fichiers sous-jacents sont utilisés comme cibles de BeeGFS pour stocker des données.

  Celles-ci sont généralement définies dans `/etc/fstab` Et monté par systemd.

- Un service systemd responsable du démarrage des processus BeeGFS lorsque les autres ressources sont prêtes.

  Sans logiciel supplémentaire, ces ressources ne démarrent que sur un seul nœud de fichiers. Par conséquent, si le nœud de fichier passe hors ligne, une partie du système de fichiers BeeGFS est inaccessible.

Comme plusieurs nœuds peuvent démarrer chaque service BeeGFS, Pacemaker doit s'assurer que chaque service et chaque ressource dépendante sont exécutés sur un seul nœud à la fois. Par exemple, si deux nœuds tentent de démarrer le même service BeeGFS, il y a un risque de corruption des données s'ils essaient tous les deux d'écrire sur les mêmes fichiers sur la cible sous-jacente. Pour éviter ce scénario, Pacemaker utilise Corosync pour maintenir en toute fiabilité l'état du cluster global en mode synchrone sur tous les nœuds et établir le quorum.

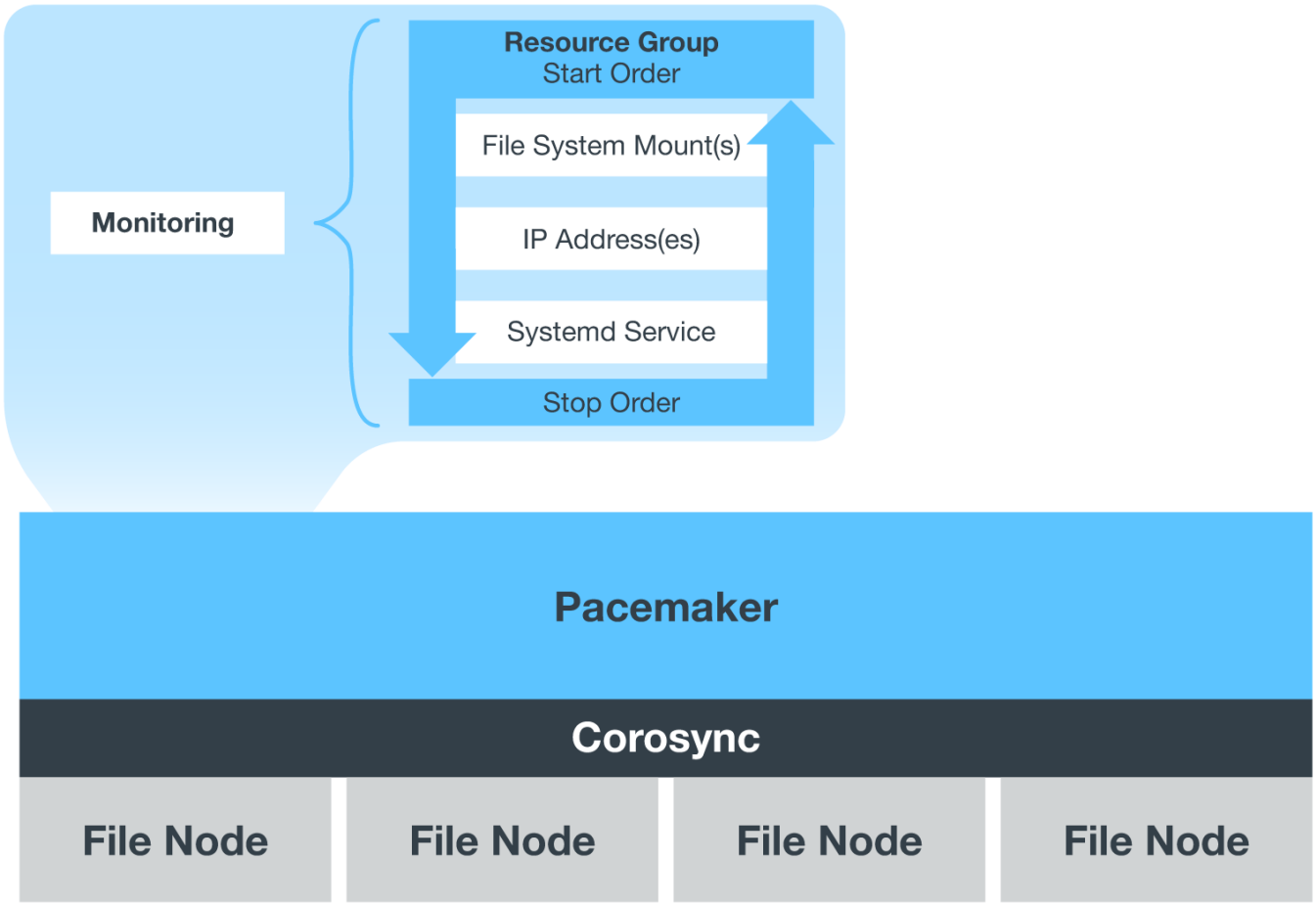
En cas de défaillance dans le cluster, Pacemaker réagit et redémarre les ressources BeeGFS sur un autre nœud. Dans certains cas, il se peut que Pacemaker ne puisse pas communiquer avec le nœud défectueux d'origine pour confirmer que les ressources sont arrêtées. Pour vérifier que le nœud est arrêté avant de redémarrer les ressources BeeGFS ailleurs, Pacemaker déligne le nœud défectueux, idéalement en retirant l'alimentation.

De nombreux agents d'escrime open source sont disponibles pour permettre à Pacemaker de verrouiller un nœud avec une unité de distribution d'alimentation (PDU) ou à l'aide du contrôleur BMC (Baseboard Management Controller) de serveur avec des API telles que Redfish.

Lorsque BeeGFS est exécuté dans un cluster HA, tous les services BeeGFS et les ressources sous-jacentes sont gérés par Pacemaker dans des groupes de ressources. Chaque service BeeGFS et les ressources dont il dépend sont configurés dans un groupe de ressources qui assure le démarrage et l'arrêt des ressources dans le bon ordre et qui sont situés sur le même nœud.

Pour chaque groupe de ressources BeeGFS, Pacemaker exécute une ressource de surveillance BeeGFS personnalisée qui est chargée de détecter les conditions de défaillance et de déclencher intelligemment les basculements lorsqu'un service BeeGFS n'est plus accessible sur un nœud particulier.

La figure suivante montre les services et les dépendances de BeeGFS contrôlés par Pacemaker.



Pour que plusieurs services BeeGFS du même type soient démarrés sur le même nœud, Pacemaker est configuré pour démarrer les services BeeGFS à l'aide de la méthode de configuration Multi-mode. Pour plus d'informations, reportez-vous à la section "Documentation BeeGFS sur Multi-mode".

Comme les services BeeGFS doivent pouvoir démarrer sur plusieurs nœuds, le fichier de configuration pour chaque service (normalement situé à `/etc/beegfs`) Est stocké sur l'un des volumes E-Series utilisés comme cible BeeGFS pour ce service. Cela rend la configuration et les données d'un service BeeGFS accessibles à tous les nœuds qui peuvent avoir besoin d'exécuter le service.

```
# tree stor_01_tgt_0101/ -L 2
stor_01_tgt_0101/
├── data
│   ├── benchmark
│   ├── buddymir
│   ├── chunks
│   ├── format.conf
│   ├── lock.pid
│   ├── nodeID
│   ├── nodeNumID
│   ├── originalNodeID
│   ├── targetID
│   └── targetNumID
└── storage_config
    ├── beegfs-storage.conf
    ├── connInterfacesFile.conf
    └── connNetFilterFile.conf
```

Vérification de la conception

Le design de deuxième génération de la solution BeeGFS sur NetApp a été vérifié à l'aide de trois profils de configuration d'élément de base.

Les profils de configuration incluent les éléments suivants :

- Un élément de base unique, incluant la gestion BeeGFS, les métadonnées et les services de stockage.
- Des métadonnées BeeGFS plus un élément de base de stockage.
- Un élément de base BeeGFS uniquement pour le stockage.

Les éléments de base ont été reliés à deux commutateurs NVIDIA Quantum InfiniBand (MQM8700). Dix clients BeeGFS étaient également connectés aux commutateurs InfiniBand et utilisés pour exécuter des utilitaires de banc d'essai synthétiques.

La figure suivante montre la configuration BeeGFS utilisée pour valider la solution BeeGFS sur NetApp.

Répartition des fichiers BeeGFS

Les systèmes de fichiers parallèles ont notamment pour avantage de répartir les fichiers individuels sur plusieurs cibles de stockage, qui peuvent représenter des volumes sur les mêmes systèmes de stockage sous-jacents ou différents.

Dans BeeGFS, vous pouvez configurer la répartition par répertoire et par fichier pour contrôler le nombre de cibles utilisées pour chaque fichier et pour contrôler la taille chunksize (ou taille de bloc) utilisée pour chaque bande de fichier. Cette configuration permet au système de fichiers de prendre en charge différents types de charges de travail et de profils d'E/S sans avoir à reconfigurer ou à redémarrer des services. Vous pouvez appliquer les paramètres de bande à l'aide du `beegfs-ctl` Outil de ligne de commande ou avec des applications qui utilisent l'API de répartition. Pour plus d'informations, consultez la documentation BeeGFS pour "Répartition" et "API de répartition".

Pour obtenir les meilleures performances, les motifs de bande ont été ajustés tout au long des tests, et les paramètres utilisés pour chaque test sont notés.

Tests de bande passante IOR : plusieurs clients

Les tests de bande passante IOR ont utilisé OpenMPI pour exécuter des travaux parallèles du générateur d'E/S synthétique IOR (disponible à partir de "GitHub HPC") Sur l'ensemble des 10 nœuds clients à un ou plusieurs blocs de construction BeeGFS. Sauf mention contraire :

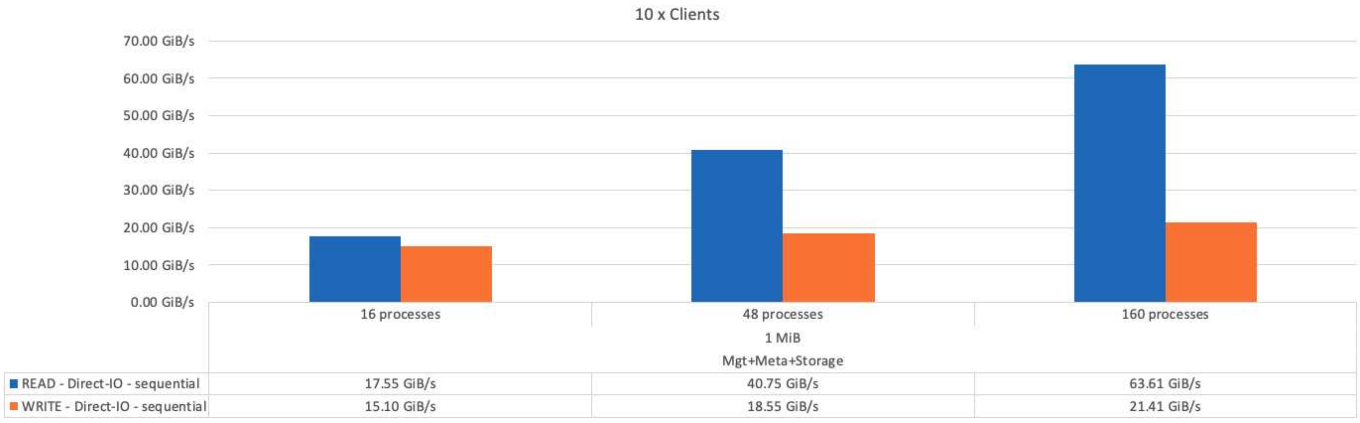
- Tous les tests ont utilisé des E/S directes avec une taille de transfert de 1MiB.
- La répartition des fichiers BeeGFS est définie sur une taille chunksize de 1 Mo et une cible par fichier.

Les paramètres suivants ont été utilisés pour IOR avec le nombre de segments ajusté afin de maintenir la taille de fichier d'agrégat à 5 Tio pour un élément de base et 40 Tio pour trois éléments de base.

```
mpirun --allow-run-as-root --mca btl tcp -np 48 -map-by node -hostfile
10xnodes ior -b 1024k --posix.odirect -e -t 1024k -s 54613 -z -C -F -E -k
```

Un élément de base BeeGFS (gestion, métadonnées et stockage)

La figure suivante montre les résultats du test IOR avec un seul élément de base BeeGFS (gestion, métadonnées et stockage).



Métadonnées BeeGFS + élément de base du stockage

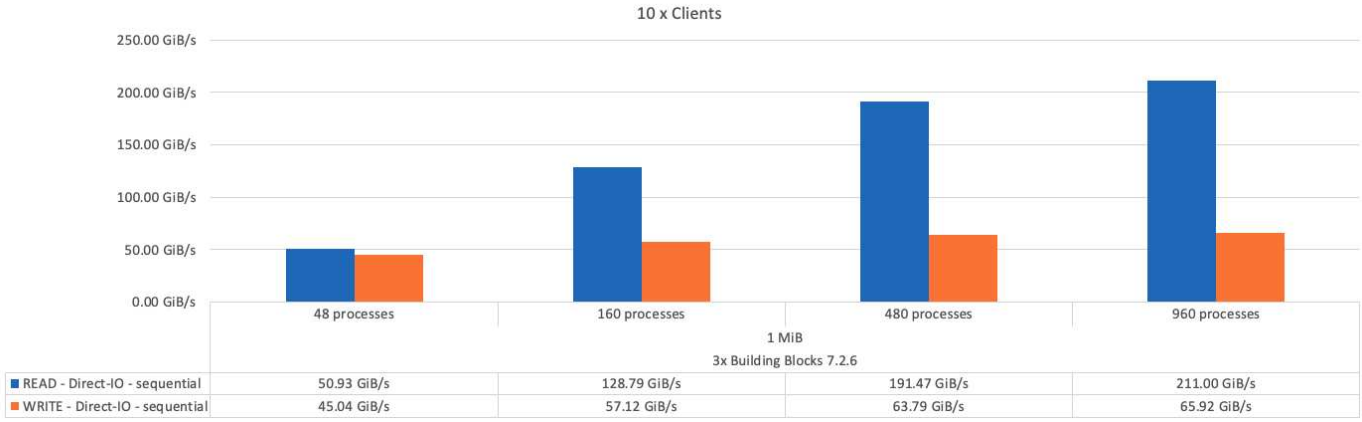
La figure suivante présente les résultats du test IOR avec un seul élément de base de stockage + métadonnées BeeGFS.

Élément de base BeeGFS uniquement pour le stockage

La figure suivante montre les résultats du test IOR avec un seul élément de base BeeGFS Storage uniquement.

Trois éléments de base BeeGFS

La figure suivante montre les résultats du test IOR avec trois éléments de base BeeGFS.



Comme on pouvait s'y attendre, la différence de performances entre l'élément de base et les métadonnées suivantes + l'élément de base du stockage est négligeable. En comparant les métadonnées + l'élément de base du stockage et un élément de base uniquement destiné au stockage, on constate une légère augmentation des performances de lecture en raison des disques supplémentaires utilisés comme cibles de stockage. Toutefois, il n'y a pas de différence significative dans les performances d'écriture. Pour améliorer les performances, vous pouvez ajouter plusieurs éléments de base pour faire évoluer les performances de manière linéaire.

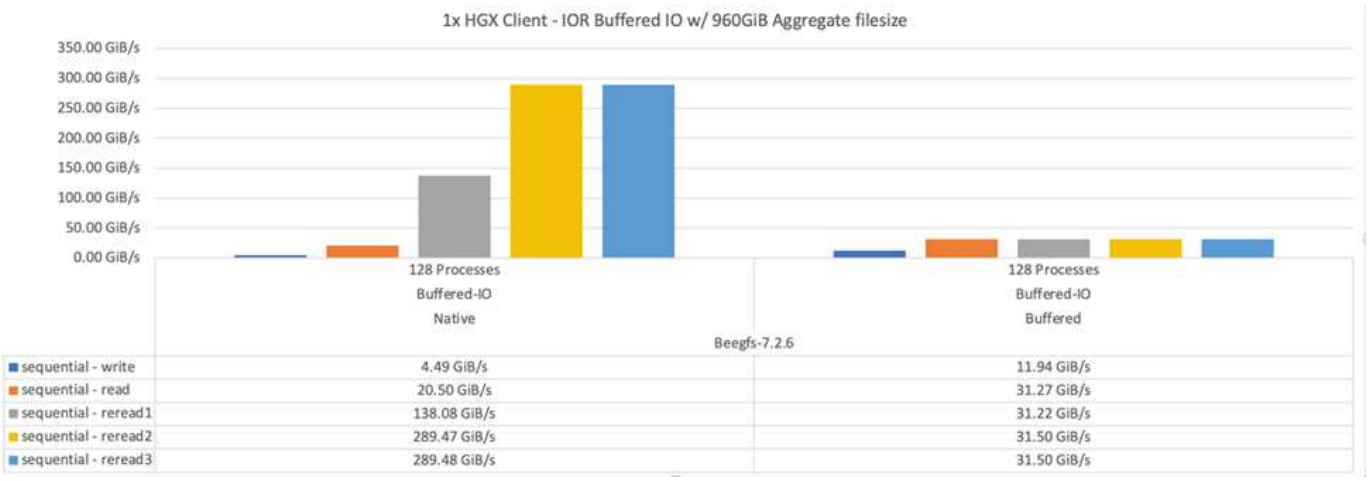
Tests de bande passante IOR : client unique

Le test de bande passante IOR a utilisé OpenMPI pour exécuter plusieurs processus IOR à l'aide d'un seul serveur GPU hautes performances afin d'explorer les performances réalisables pour un même client.

Ce test compare également le comportement et les performances de relecture de BeeGFS lorsque le client est configuré pour utiliser le cache de page du noyau Linux (`tuneFileCacheType = native`) par rapport à la valeur par défaut `buffered` réglage.

Le mode de mise en cache native utilise le cache de page du noyau Linux sur le client, ce qui permet aux opérations de relecture de provenir de la mémoire locale au lieu d'être retransmises sur le réseau.

Le diagramme suivant montre les résultats du test IOR avec trois éléments de base BeeGFS et un seul client.



La répartition BeeGFS pour ces tests a été définie sur une taille chunksize de 1 Mo avec huit cibles par fichier.

Bien que les performances d'écriture et de lecture initiale soient supérieures en mode tampon par défaut, pour

les charges de travail qui relisent plusieurs fois les mêmes données, le mode de mise en cache natif a permis d'optimiser considérablement les performances. Cette amélioration des performances de relecture est importante pour les charges de travail telles que l'apprentissage profond qui relire le même dataset plusieurs fois sur plusieurs séries de tests.

Test de performance des métadonnées

Les tests de performance des métadonnées ont utilisé l'outil MDTest (inclus dans IOR) pour mesurer la performance des métadonnées de BeeGFS. Les tests ont utilisé OpenMPI pour exécuter des travaux parallèles sur les dix nœuds clients.

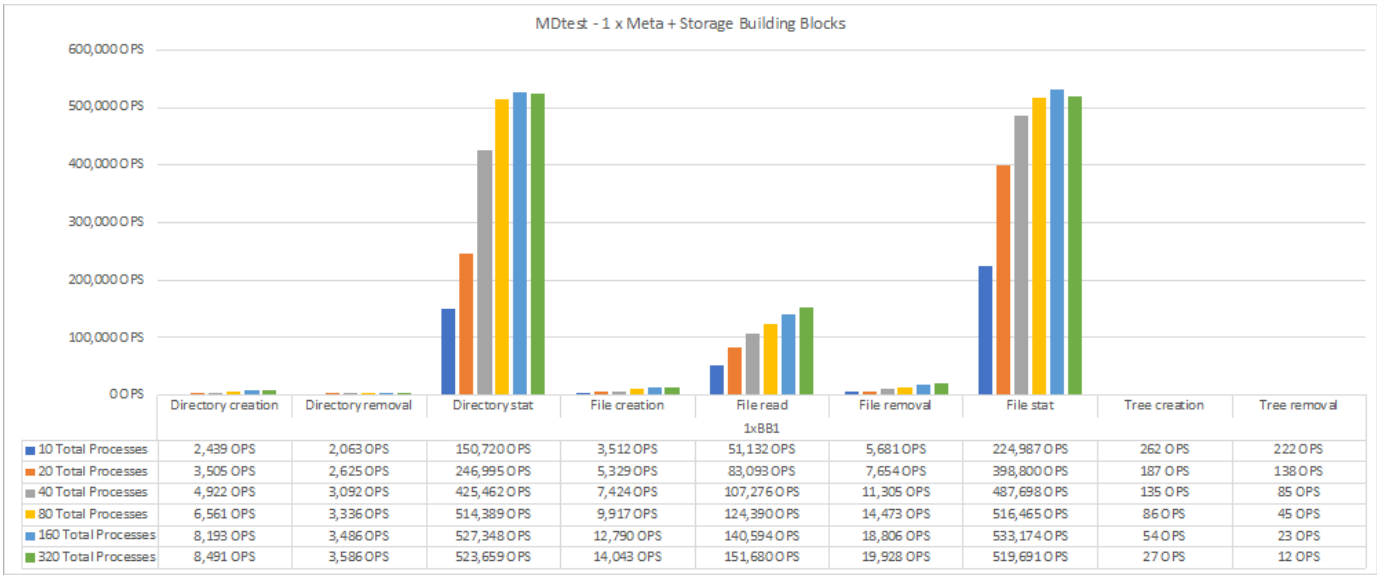
Les paramètres suivants ont été utilisés pour exécuter le test de référence avec le nombre total de processus passe de 10 à 320 par pas de 2x et avec une taille de fichier de 4 ko.

```
mpirun -h 10xnodes –map-by node np $processes mdtest -e 4k -w 4k -i 3 -I
16 -z 3 -b 8 -u
```

Les performances des métadonnées ont été mesurées en premier avec un ou deux blocs de base de stockage + métadonnées afin de montrer l'évolution des performances en ajoutant des éléments de base supplémentaires.

Un seul élément de base de métadonnées BeeGFS + stockage

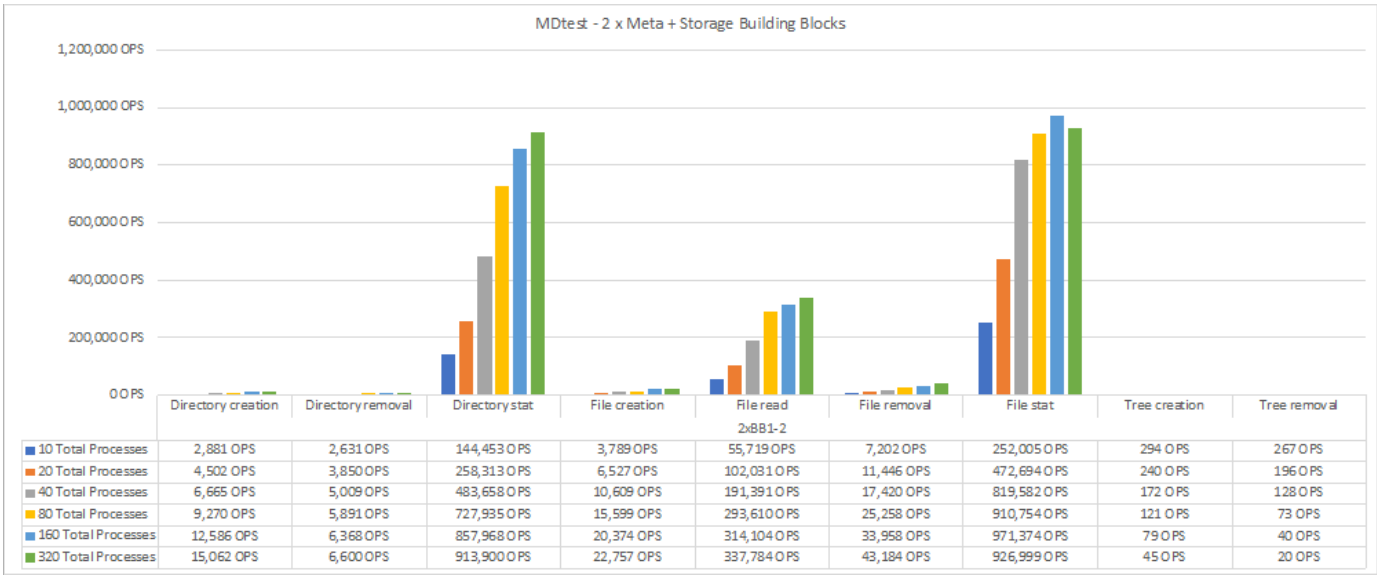
Le diagramme suivant montre les résultats MDTest avec un bloc de construction BeeGFS + stockage.



1xBB1	Directory creation	Directory removal	Directory stat	File creation	File read	File removal	File stat	Tree creation	Tree removal
10 Total Processes	2,439 OPS	2,063 OPS	150,720 OPS	3,512 OPS	51,132 OPS	5,681 OPS	224,987 OPS	262 OPS	222 OPS
20 Total Processes	3,505 OPS	2,625 OPS	246,995 OPS	5,329 OPS	83,093 OPS	7,654 OPS	398,800 OPS	187 OPS	138 OPS
40 Total Processes	4,922 OPS	3,092 OPS	425,462 OPS	7,424 OPS	107,276 OPS	11,305 OPS	487,698 OPS	135 OPS	85 OPS
80 Total Processes	6,561 OPS	3,336 OPS	514,389 OPS	9,917 OPS	124,390 OPS	14,473 OPS	516,465 OPS	86 OPS	45 OPS
160 Total Processes	8,193 OPS	3,486 OPS	527,348 OPS	12,790 OPS	140,594 OPS	18,806 OPS	533,174 OPS	54 OPS	23 OPS
320 Total Processes	8,491 OPS	3,586 OPS	523,659 OPS	14,043 OPS	151,680 OPS	19,928 OPS	519,691 OPS	27 OPS	12 OPS

Deux métadonnées BeeGFS + éléments de base du stockage

Le diagramme suivant montre les résultats MDTest avec deux métadonnées BeeGFS + des modules de stockage.



	Directory creation	Directory removal	Directory stat	File creation	File read	File removal	File stat	Tree creation	Tree removal
					2xBB1-2				
10 Total Processes	2,881 OPS	2,631 OPS	144,453 OPS	3,789 OPS	55,719 OPS	7,202 OPS	252,005 OPS	294 OPS	267 OPS
20 Total Processes	4,502 OPS	3,850 OPS	258,313 OPS	6,527 OPS	102,031 OPS	11,446 OPS	472,694 OPS	240 OPS	196 OPS
40 Total Processes	6,665 OPS	5,009 OPS	483,658 OPS	10,609 OPS	191,391 OPS	17,420 OPS	819,582 OPS	172 OPS	128 OPS
80 Total Processes	9,270 OPS	5,891 OPS	727,935 OPS	15,599 OPS	293,610 OPS	25,258 OPS	910,754 OPS	121 OPS	73 OPS
160 Total Processes	12,586 OPS	6,368 OPS	857,968 OPS	20,374 OPS	314,104 OPS	33,958 OPS	971,374 OPS	79 OPS	40 OPS
320 Total Processes	15,062 OPS	6,600 OPS	913,900 OPS	22,757 OPS	337,784 OPS	43,184 OPS	926,999 OPS	45 OPS	20 OPS

Validation fonctionnelle

Dans le cadre de la validation de cette architecture, NetApp a effectué plusieurs tests fonctionnels :

- Défaillance d'un seul port InfiniBand client en désactivant le port de commutateur.
- Défaillance d'un seul port InfiniBand de serveur en désactivant le port du commutateur.
- Déclenchement d'une mise hors tension immédiate d'un serveur à l'aide du contrôleur BMC.
- Placement normal d'un nœud en veille et basculement de service vers un autre nœud.
- Il est normal de remettre un nœud en ligne et de renvoyer les services vers le nœud d'origine.
- Mise hors tension de l'un des commutateurs InfiniBand à l'aide de la PDU. Tous les tests ont été réalisés alors que les tests de stress étaient en cours avec le `sysSessionChecksEnabled: false` Paramètre défini sur les clients BeeGFS. Aucune erreur ni interruption des E/S n'a été observée.

> Il y a un problème connu (voir "Changement") Lorsque les connexions RDMA BeeGFS client/serveur sont interrompues de façon inattendue, soit par la perte de l'interface principale (comme défini dans la section `connInterfacesFile`) Ou un serveur BeeGFS est défaillant ; les E/S du client actif peuvent se bloquer pendant dix minutes avant de reprendre. Ce problème ne se produit pas lorsque les nœuds BeeGFS sont correctement placés en attente pour la maintenance planifiée ou si TCP est utilisé.

Validation de NVIDIA DGX SuperPOD et BasePOD

NetApp a validé une solution de stockage pour NVIDIA DGX A100 SuperPOD à l'aide d'un système de fichiers BeeGFS constitué de trois éléments de base avec les métadonnées plus le profil de configuration du stockage appliqué. L'effort de qualification a participé au test de la solution décrite par cette architecture NVA avec vingt serveurs GPU DGX A100 exécutant plusieurs bancs d'essai de stockage, d'apprentissage machine et d'apprentissage profond. Basée sur la validation établie avec le DGX A100 SuperPOD de NVIDIA, la solution BeeGFS sur NetApp a été approuvée pour les systèmes DGX SuperPOD H100, H200 et B200. Cette extension repose sur le respect des bancs d'essai et des exigences système précédemment établis et validés avec le système NVIDIA DGX A100

Pour plus d'informations, voir "NVIDIA DGX SuperPOD avec NetApp" et "NVIDIA DGX BasePOD".

Instructions de dimensionnement

La solution BeeGFS inclut des recommandations sur le dimensionnement de la performance et de la capacité qui étaient basées sur des tests de vérification.

L'objectif de l'architecture modulaire est de créer une solution simple à dimensionner en ajoutant plusieurs éléments de base pour répondre aux exigences d'un système BeeGFS particulier. À l'aide des lignes directrices ci-dessous, vous pouvez estimer la quantité et les types de blocs de construction BeeGFS qui sont nécessaires pour répondre aux exigences de votre environnement.

Notez que ces estimations sont les meilleures performances au cas par cas. Les applications de test des performances synthétiques sont écrites et utilisées pour optimiser l'utilisation des systèmes de fichiers sous-jacents d'une manière qui n'est pas forcément possible pour les applications réelles.

Dimensionnement de la performance

Le tableau suivant indique le dimensionnement de performance recommandé.

Profil de configuration	1MiB lit	Écritures 1MiB
Métadonnées + stockage	62 GiBps	21 GiBps
Stockage uniquement	64 GiBps	21 GiBps

Le dimensionnement de la capacité des métadonnées est basé sur la « règle générale » selon laquelle 500 Go de capacité suffisent pour environ 150 millions de fichiers sur BeeGFS. (Pour plus d'informations, consultez la documentation BeeGFS pour "Configuration minimale requise".)

L'utilisation de fonctions telles que les listes de contrôle d'accès et le nombre de répertoires et de fichiers par répertoire affecte également la vitesse de consommation de l'espace de métadonnées. Les estimations de capacité de stockage tiennent compte de la capacité de disque utilisable ainsi que de la surcharge RAID 6 et XFS.

Dimensionnement de la capacité pour les métadonnées + éléments de base du stockage

Le tableau suivant indique le dimensionnement de la capacité recommandé pour les métadonnées et les éléments de base du stockage.

Taille du disque (2+2 RAID 1) groupes de volumes de métadonnées	Capacité des métadonnées (nombre de fichiers)	Taille des disques (RAID 6 8+2) groupes de volumes de stockage	Capacité de stockage (contenu de fichiers)
1,92 TO	1,938,577,200	1,92 TO	51,77 TO
3,84 TO	3,880,388,400	3,84 TO	103,55 TO
7,68 TO	8,125,278,000	7,68 TO	28.74 TO
15,3 TO	17,269,854,000	15,3 TO	460.60 TO

Lors du dimensionnement des métadonnées et des éléments de base de stockage, vous pouvez réduire les coûts en utilisant des disques plus petits pour les groupes de volumes de métadonnées et les groupes de volumes de stockage.

Dimensionnement de la capacité pour les éléments de base uniquement destinés au stockage

Le tableau suivant indique la règle générale de dimensionnement de la capacité pour les éléments de base uniquement liés au stockage.

Taille des disques (RAID 6 10+2) groupes de volumes de stockage	Capacité de stockage (contenu de fichiers)
1,92 TO	59,89 TO
3,84 TO	11980 TO
7,68 TO	251,89TB
15,3 TO	58,55 TO

Les performances et la surcharge liée à la capacité de l'inclusion du service de gestion dans le premier élément de base sont minimales, sauf si le verrouillage global des fichiers est activé.

Réglage des performances

La solution BeeGFS inclut des recommandations sur le réglage de la performance qui étaient basées sur des tests de vérification.

Bien que BeeGFS fournit des performances raisonnables, NetApp a développé un ensemble de paramètres d'ajustement recommandés pour optimiser les performances. Ces paramètres prennent en compte les fonctionnalités des nœuds de bloc E-Series sous-jacents et les exigences spéciales requises pour exécuter BeeGFS dans une architecture HA à disque partagé.

L'ajustement des performances des nœuds de fichiers

Les paramètres de réglage disponibles que vous pouvez configurer sont les suivants :

1. **Paramètres système dans l'UEFI/BIOS des nœuds de fichiers.** pour optimiser les performances, nous vous recommandons de configurer les paramètres système sur le modèle de serveur que vous utilisez comme nœuds de fichiers. Vous configurez les paramètres système lorsque vous configurez vos nœuds de fichiers à l'aide de la configuration du système (UEFI/BIOS) ou des API Redfish fournies par le contrôleur de gestion de la carte mère (BMC).

   Les paramètres système varient en fonction du modèle de serveur que vous utilisez comme nœud de fichier. Les paramètres doivent être configurés manuellement en fonction du modèle de serveur utilisé. Pour savoir comment configurer les paramètres système pour les nœuds de fichiers Lenovo SR665 V3 validés, consultez "Réglez les paramètres du système de nœud de fichiers en fonction des performances" .

2. **Paramètres par défaut pour les paramètres de configuration requis.** les paramètres de configuration requis affectent la configuration des services BeeGFS et la façon dont les volumes E-Series (dispositifs de bloc) sont formatés et montés par Pacemaker. Voici les paramètres de configuration requis :
   - Paramètres de configuration du service BeeGFS

     Vous pouvez remplacer les paramètres par défaut des paramètres de configuration selon vos besoins. Pour connaître les paramètres que vous pouvez ajuster en fonction de vos charges de travail ou de vos cas d'utilisation spécifiques, reportez-vous au "Paramètres de configuration du service BeeGFS".

   - Le formatage de volume et les paramètres de montage sont définis sur les valeurs par défaut

recommandées et ne doivent être ajustés que pour des cas d'utilisation avancés. Les valeurs par défaut sont les suivantes :

- Optimiser le formatage du volume initial en fonction du type de cible (gestion, métadonnées ou stockage, par exemple), de la configuration RAID et de la taille du segment du volume sous-jacent.
- Réglez la manière dont Pacemaker monte chaque volume pour vous assurer que les modifications sont immédiatement transférées vers les nœuds de blocs E-Series. Cela empêche la perte de données en cas d'échec des nœuds de fichier pour les écritures actives.

  Pour connaître les paramètres que vous pouvez ajuster en fonction de vos charges de travail ou de vos cas d'utilisation spécifiques, reportez-vous au "formatage du volume et paramètres de configuration du montage".

3. **Paramètres système du système d'exploitation Linux installé sur les nœuds de fichiers.** Lorsque vous créez l'inventaire Ansible à l'étape 4 de la section , vous pouvez remplacer les paramètres par défaut du système d'exploitation Linux "Créez l'inventaire Ansible".

   Les paramètres par défaut ont été utilisés pour valider la solution BeeGFS sur NetApp, mais vous pouvez les modifier pour s'adapter à vos workloads ou à vos utilisations spécifiques. Voici quelques exemples de paramètres système d'exploitation Linux que vous pouvez modifier :

   - Files d'attente des E/S sur les dispositifs de bloc E-Series.

     Vous pouvez configurer des files d'attente d'E/S sur les périphériques de bloc E-Series utilisés comme cibles BeeGFS pour :

     - Réglez l'algorithme de planification en fonction du type de périphérique (NVMe, HDD, etc.).
     - Augmenter le nombre de demandes en attente.
     - Réglez les tailles des demandes.
     - Optimisez le comportement de lecture anticipée.
   - Paramètres de la mémoire virtuelle.

     Vous pouvez régler les paramètres de la mémoire virtuelle pour des performances de diffusion optimales en continu.

   - Paramètres CPU.

     Vous pouvez régler le régulateur de fréquence de l'UC et d'autres configurations de l'UC pour obtenir des performances maximales.

   - Taille de la demande de lecture.

     Vous pouvez augmenter la taille maximale des demandes de lecture pour les applications HCA NVIDIA.

Réglage des performances des nœuds en mode bloc

En fonction des profils de configuration appliqués à un élément de base BeeGFS particulier, les groupes de volumes configurés sur les nœuds de blocs changent légèrement. Par exemple, avec un nœud de bloc EF600 de 24 disques :

- Pour un seul élément de base, y compris la gestion BeeGFS, les métadonnées et les services de stockage :

- 1 groupe de volumes RAID 10 2+2 pour la gestion BeeGFS et les services de métadonnées
- 2 groupes de volumes RAID 6 8+2 pour les services de stockage BeeGFS

- Pour un élément de base de métadonnées + de stockage BeeGFS :
  - 1 groupe de volumes RAID 10 2+2 pour les services de métadonnées BeeGFS
  - 2 groupes de volumes RAID 6 8+2 pour les services de stockage BeeGFS
- Pour l'élément de base de stockage BeeGFS uniquement :
  - 2 groupes de volumes RAID 6 10+2 pour les services de stockage BeeGFS

> Comme BeeGFS a besoin d'un espace de stockage considérable pour la gestion et les métadonnées par rapport au stockage, une seule option consiste à utiliser des disques plus petits pour les groupes de volumes RAID 10. Les lecteurs plus petits doivent être insérés dans les emplacements de lecteur les plus extérieurs. Pour plus d'informations, reportez-vous à la section "instructions de déploiement".

Tous ces paramètres sont configurés par le déploiement Ansible, et plusieurs autres paramètres sont généralement recommandés pour optimiser les performances/comportements :

- Ajustement de la taille du bloc de cache global à 32Kio et ajustement de la vidage du cache à la demande à 80 %.
- Désactivation de l'équilibrage automatique (en veillant à ce que les attributions de volume du contrôleur restent telles que prévues).
- Activation de la mise en cache de lecture et désactivation de la mise en cache de lecture anticipée
- Activation de la mise en cache d'écriture avec la mise en miroir et demande de sauvegarde sur batterie, les caches sont donc conservés suite à la panne d'un contrôleur de nœud bloc.
- Spécification de l'ordre dans lequel les disques sont affectés aux groupes de volumes, en équilibrant les E/S entre les canaux de disque disponibles.

Élément de base haute capacité

La solution BeeGFS est conçue de façon très performante. Les clients recherchant des cas d'utilisation de grande capacité doivent observer les variations des caractéristiques de conception et de performances décrites ici.

Configuration matérielle et logicielle

Les configurations matérielles et logicielles de l'élément de base haute capacité sont standard, mais les contrôleurs EF600 doivent être remplacés par des contrôleurs EF300, avec une option qui permet de connecter entre 1 et 7 tiroirs d'extension IOM avec 60 disques chacun pour chaque baie de stockage, un total de 2 à 14 tiroirs d'extension par module.

Les clients qui déploient un design d'éléments de base haute capacité n'utilisent probablement que la configuration de type élément de base, qui comprend la gestion BeeGFS, les métadonnées et les services de stockage pour chaque nœud. Pour garantir une rentabilité accrue, les nœuds de stockage haute capacité doivent provisionner des volumes de métadonnées sur les disques NVMe du boîtier de contrôleur EF300 et provisionner les volumes de stockage sur les disques NL-SAS des tiroirs d'extension.

[]

Instructions de dimensionnement

Ces recommandations de dimensionnement supposent que les blocs de base haute capacité sont configurés avec un groupe de volumes SSD NVMe 2+2 pour les métadonnées dans le boîtier EF300 de base et six groupes de volumes NL-SAS 8+2 par plateau d'extension IOM pour le stockage.

Taille du disque (disques durs haute capacité)	Capacité par BB (1 plateau)	Capacité par BB (2 plateaux)	Capacité par BB (3 plateaux)	Capacité par BB (4 plateaux)
4 TO	43TB	878 TO	1317 TO	1756 TO
8 TO	878 TO	1756 TO	2634 TO	3512 TO
10 TO	1097 TO	2195 TO	3292 TO	4390 TO
12 To	1317 TO	2634 TO	3951 TO	5268 TO
16 TO	1756 TO	3512 TO	5268 TO	7024 TO
18 TO	1975 TO	3951 TO	5927 TO	7902 TO

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
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

Nœud de fichier	Port IB	Adresse IP	Nœud de bloc	Port IB	IP physique	IP virtuel
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
Métadonnées BeeGFS	i1b ou i3b	100.127.101.1 - 100.127.101.255
	i2b ou i4b	100.127.102.1 - 100.127.102.255

Objectif	Port InfiniBand	Adresse IP ou plage
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).

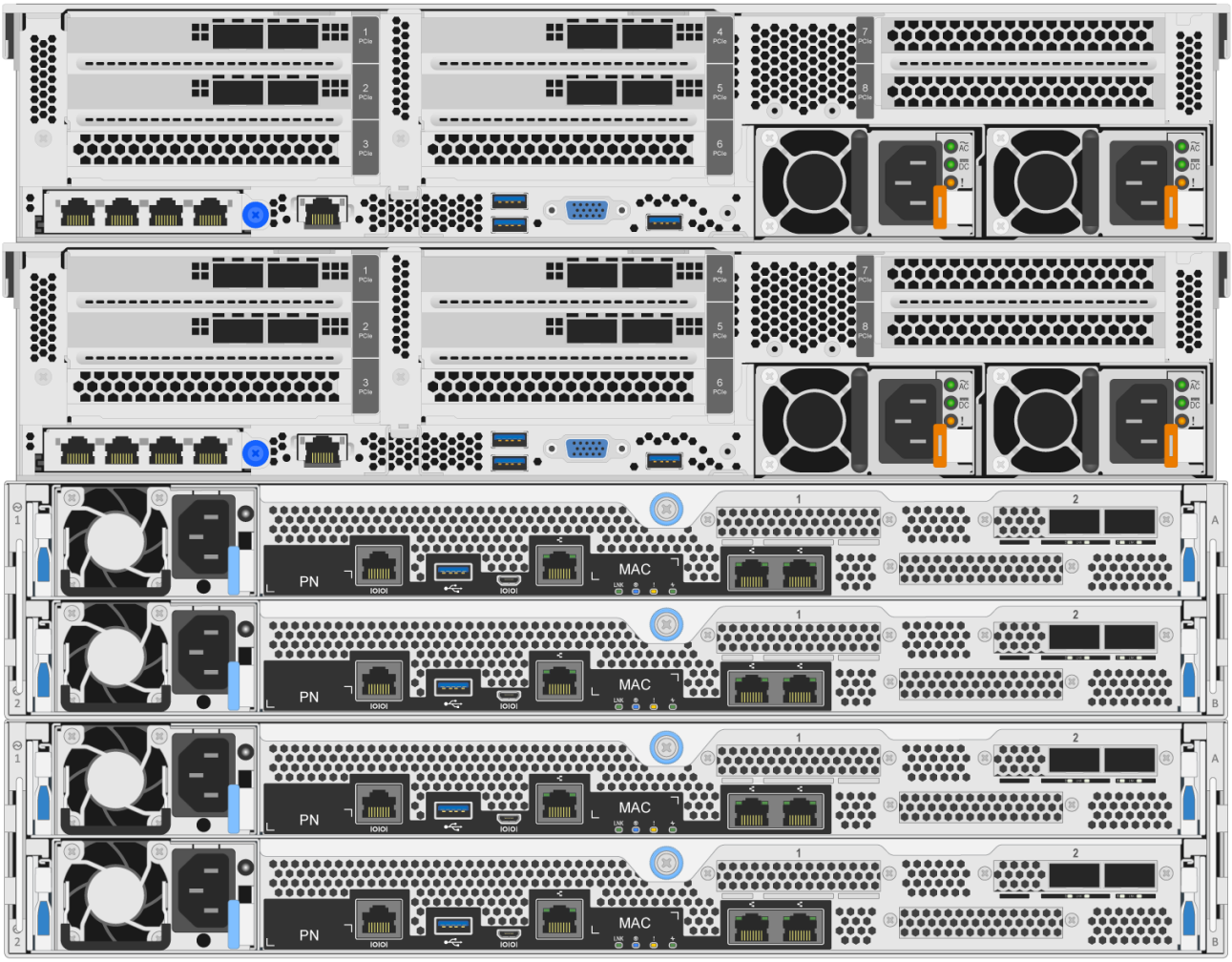
Un minimum de deux éléments de base est requis pour établir le quorum dans le cluster de basculement. Un cluster à deux nœuds présente des limites qui peuvent empêcher un basculement réussi. Vous pouvez configurer un cluster à deux nœuds en incorporant un troisième périphérique comme disjoncteur d'attache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

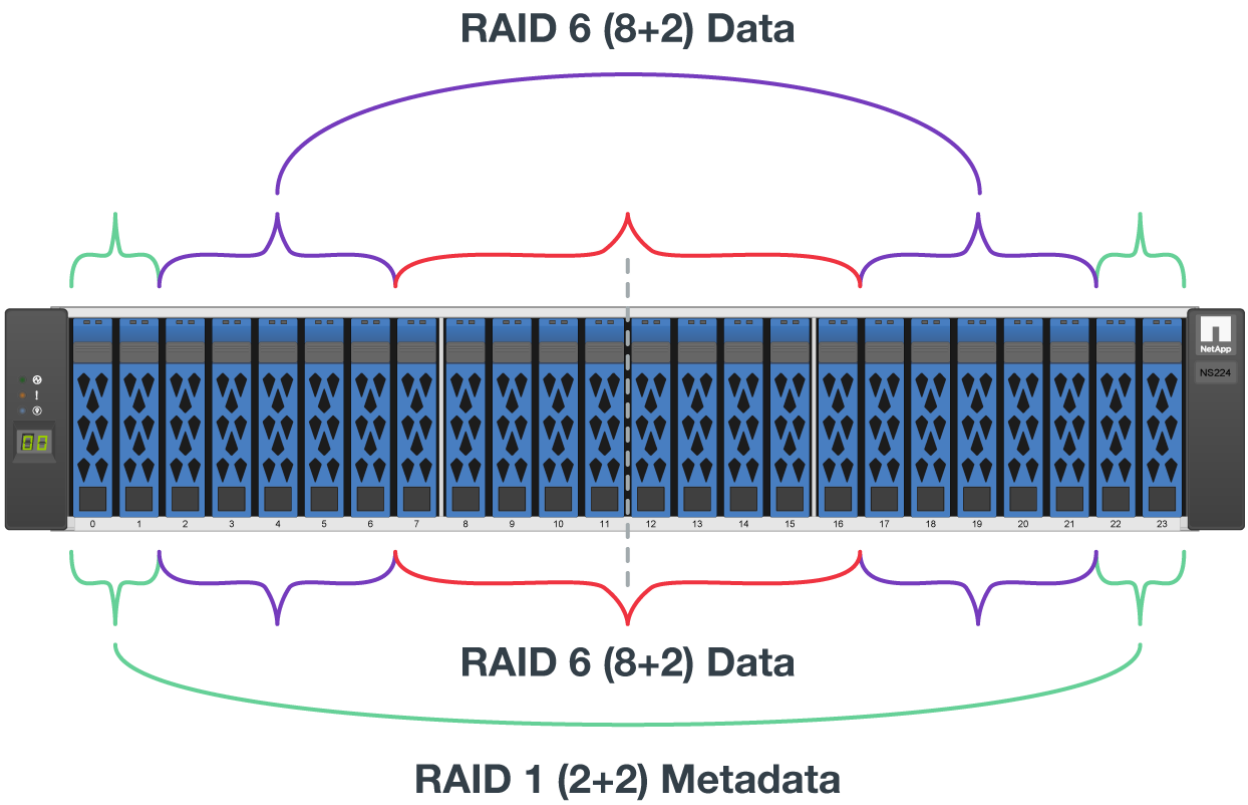
1. Configurez chaque nœud de fichiers BeeGFS avec quatre adaptateurs HCA (Host Channel Adapters) à l'aide des modèles spécifiés dans le "Exigences techniques". Insérez les HCA dans les connecteurs PCIe de votre nœud de fichiers conformément aux spécifications ci-dessous :
   - **Lenovo ThinkSystem SR665 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).

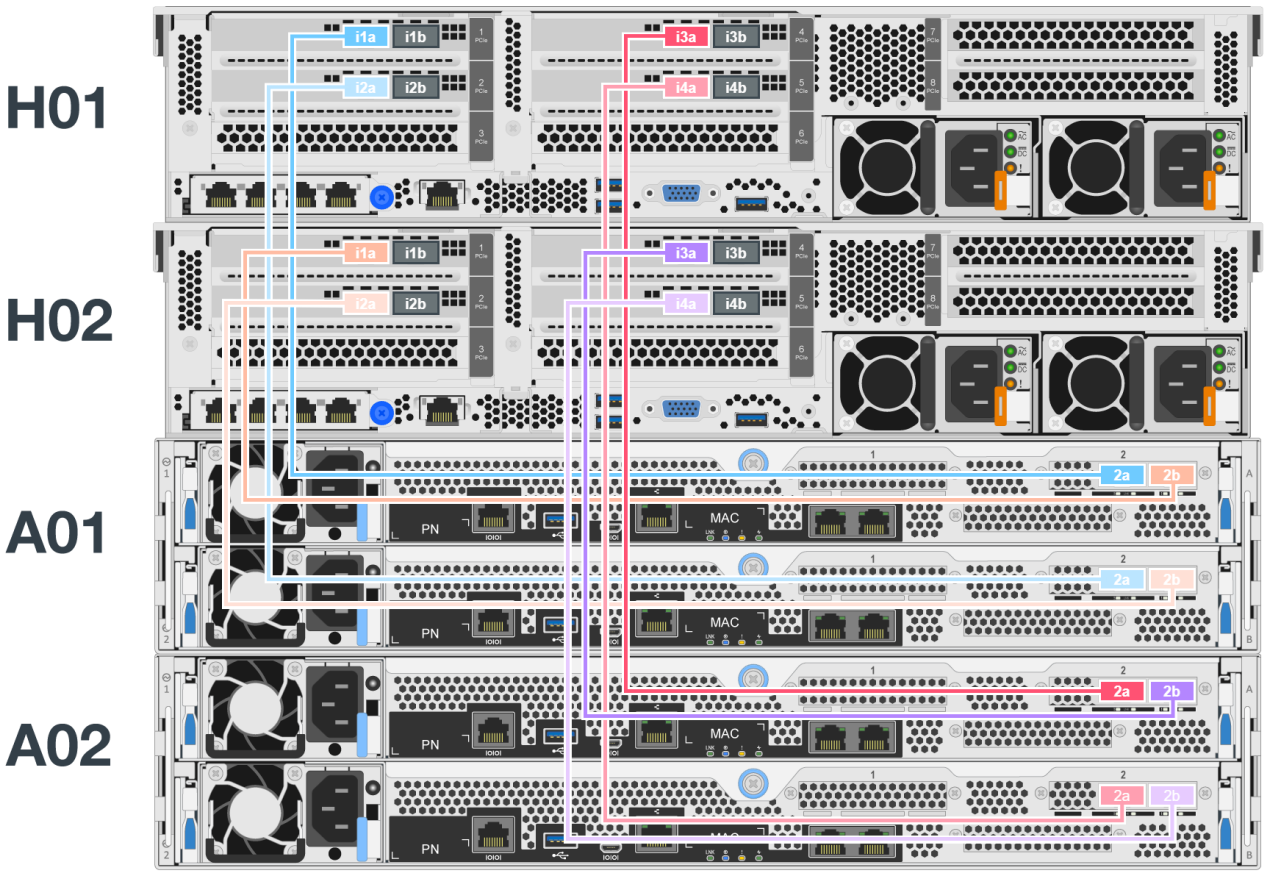


La configuration de l'alimentation électrique pour les cas d'utilisation en production doit généralement utiliser des blocs d'alimentation redondants.

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.

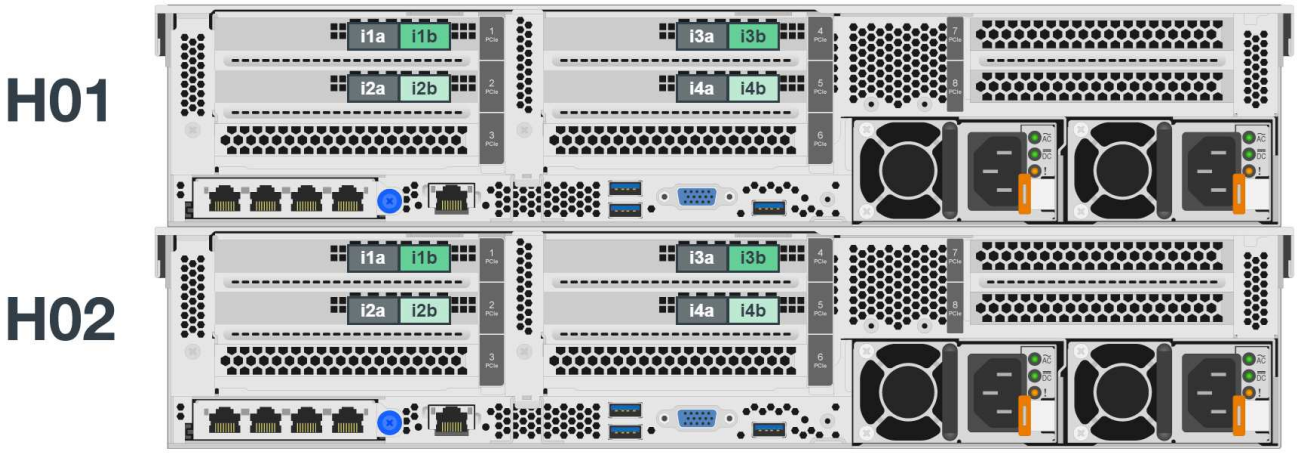


Les nœuds répartis entre plusieurs éléments de base ne sont jamais directement connectés. Chaque élément de base doit être considéré comme une unité autonome et toute communication entre les éléments de base se fait par le biais de commutateurs réseau.

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.

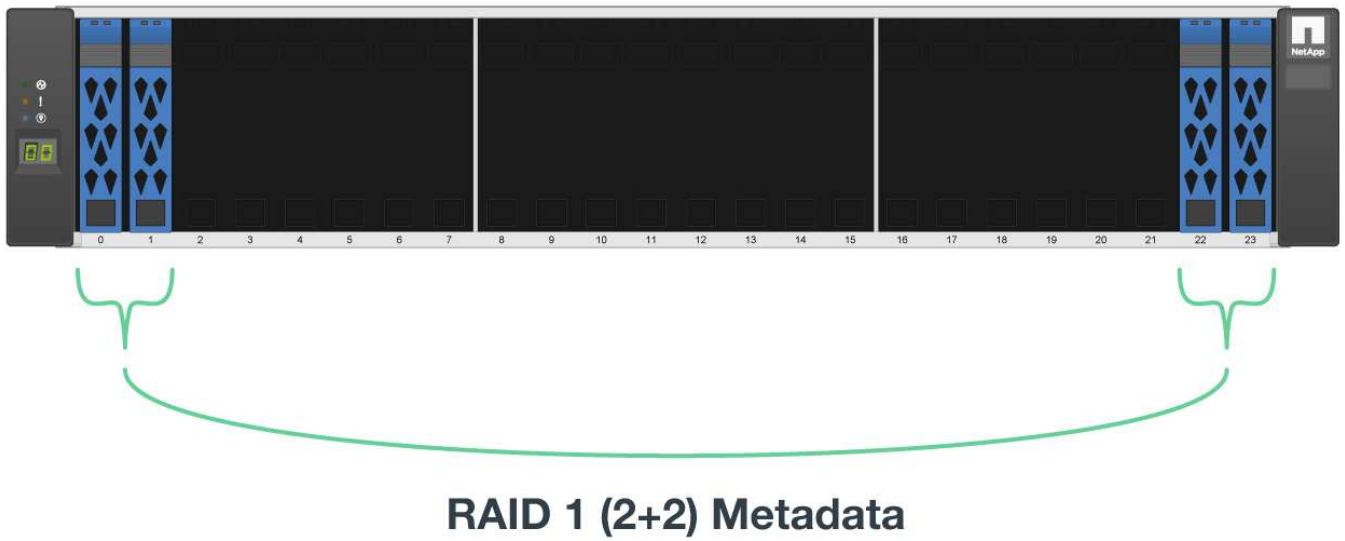
Contrôleurs

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".

Utiliser des architectures personnalisées

Présentation et configuration requise

Utilisez tous les systèmes de stockage NetApp E/EF-Series comme nœuds de bloc BeeGFS et serveurs x86 comme nœuds de fichiers BeeGFS lors du déploiement de clusters haute disponibilité BeeGFS à l'aide d'Ansible.

Les définitions de la terminologie utilisée dans cette section se trouvent à la "termes et concepts" page.

Introduction

Même si "Architectures vérifiées NetApp" nous fournissons des configurations de référence prédéfinies et des conseils sur le dimensionnement, certains clients et partenaires préfèrent peut-être concevoir des architectures personnalisées mieux adaptées à des exigences spécifiques ou à des préférences matérielles spécifiques. L'un des principaux avantages de BeeGFS sur NetApp est la possibilité de déployer des clusters HA à disque partagé BeeGFS à l'aide d'Ansible, en simplifiant la gestion du cluster et en améliorant la fiabilité grâce aux composants HA de NetApp. Le déploiement d'architectures BeeGFS sur NetApp est toujours réalisé à l'aide d'Ansible, tout en conservant une approche de type appliance sur un éventail flexible de matériel.

Cette section présente les étapes générales requises pour déployer des systèmes de fichiers BeeGFS sur du matériel NetApp et pour utiliser Ansible afin de configurer les systèmes de fichiers BeeGFS. Pour en savoir plus sur les bonnes pratiques de conception des systèmes de fichiers BeeGFS et sur les exemples optimisés"Architectures vérifiées NetApp", reportez-vous à la section.

Présentation du déploiement

Le déploiement d'un système de fichiers BeeGFS implique généralement les étapes suivantes :

- Configuration initiale :
  - Installer/raccorder le matériel de fixation.
  - Configurez les nœuds de fichiers et de blocs.
  - Configurez un nœud de contrôle Ansible.
- Définissez le système de fichiers BeeGFS comme un inventaire Ansible.
- Exécutez Ansible sur des nœuds de fichiers et de blocs pour déployer BeeGFS.
  - Vous pouvez également configurer des clients et monter BeeGFS.

Les sections suivantes couvriront ces étapes plus en détail.

Ansible gère toutes les tâches de provisionnement et de configuration des logiciels, y compris :

- Création/mappage de volumes sur des nœuds de blocs.
- Formatage/réglage des volumes sur les nœuds de fichiers.
- Installation/configuration du logiciel sur les nœuds de fichiers.
- Création du cluster HA et configuration des ressources BeeGFS et des services de système de fichiers.

De formation

La prise en charge de BeeGFS dans Ansible est activée "Galaxy Ansible" Ensemble de rôles et de modules qui automatisent le déploiement et la gestion de bout en bout des clusters BeeGFS HA.

BeeGFS est également versionné suivant un schéma de gestion des versions <major>.<minor>.<patch> et la collection conserve les rôles pour chaque version <major>.<minor> prise en charge de BeeGFS, par exemple BeeGFS 7.2 ou BeeGFS 7.3. À mesure que les mises à jour de la collection sont publiées, la version patch dans chaque rôle sera mise à jour pour pointer à la dernière version BeeGFS disponible pour cette branche de publication (exemple : 7.2.8). Chaque version de la collection est également testée et prise en charge avec des distributions et versions Linux spécifiques, actuellement Red Hat pour les nœuds de fichiers et Red Hat et Ubuntu pour les clients. L'exécution d'autres distributions n'est pas prise en charge et l'exécution d'autres versions (en particulier d'autres versions majeures) n'est pas recommandée.

Nœud de contrôle Ansible

Ce nœud contiendra l'inventaire et les manuels de vente utilisés pour gérer BeeGFS. Elle requiert :

- Ansible 6.x (noyau ansible 2.13)
- Python 3.6 (ou version ultérieure)
- Paquets Python (pip) : ipadr et netaddr

Il est également recommandé d'installer SSH sans mot de passe depuis le nœud de contrôle vers tous les nœuds de fichiers et clients BeeGFS.

Nœuds de fichiers BeeGFS

Les nœuds de fichiers doivent exécuter Red Hat Enterprise Linux (RHEL) 9.4 et avoir accès au référentiel HA contenant les packages requis (pacemaker, corosync, fence-agents-all, resource-agents). Par exemple, la commande suivante peut être exécutée pour activer le référentiel approprié sur RHEL 9 :

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

Nœuds clients BeeGFS

Un rôle de client BeeGFS Ansible est disponible pour installer le paquet client BeeGFS et gérer le(s) BeeGFS mount(s). Ce rôle a été testé avec RHEL 9.4 et Ubuntu 22.04.

Si vous n'utilisez pas Ansible pour configurer le client BeeGFS et monter BeeGFS, tout "BeeGFS prend en charge la distribution et le noyau Linux" peut être utilisé.

Configuration initiale

Installez et fixez les câbles

Étapes nécessaires pour installer et câbler le matériel utilisé pour exécuter BeeGFS sur NetApp.

Planifier l'installation

Chaque système de fichiers BeeGFS est composé d'un certain nombre de nœuds de fichiers exécutant des services BeeGFS à l'aide du stockage back-end fourni par un certain nombre de nœuds de blocs. Les nœuds de fichiers sont configurés en un ou plusieurs clusters haute disponibilité pour assurer la tolérance aux pannes des services BeeGFS. Chaque nœud de bloc est déjà une paire haute disponibilité actif-actif. Le nombre minimal de nœuds de fichiers pris en charge dans chaque cluster haute disponibilité est de trois, et le nombre maximum de nœuds de fichiers pris en charge dans chaque cluster est de dix. Les systèmes de fichiers BeeGFS peuvent évoluer au-delà de dix nœuds en déployant plusieurs clusters HA indépendants qui fonctionnent ensemble pour fournir un espace de noms de système de fichiers unique.

Généralement, chaque cluster haute disponibilité est déployé sous la forme d'éléments de base, où un certain nombre de nœuds de fichiers (serveurs x86) sont directement connectés à un certain nombre de nœuds blocs (généralement des systèmes de stockage E-Series). Cette configuration crée un cluster asymétrique où les services BeeGFS peuvent uniquement s'exécuter sur certains nœuds de fichiers qui ont accès au stockage de bloc back-end utilisé pour les cibles BeeGFS. L'équilibre entre des nœuds de fichier à bloc dans chaque élément de base et le protocole de stockage utilisé pour les connexions directes dépend des exigences d'une installation précise.

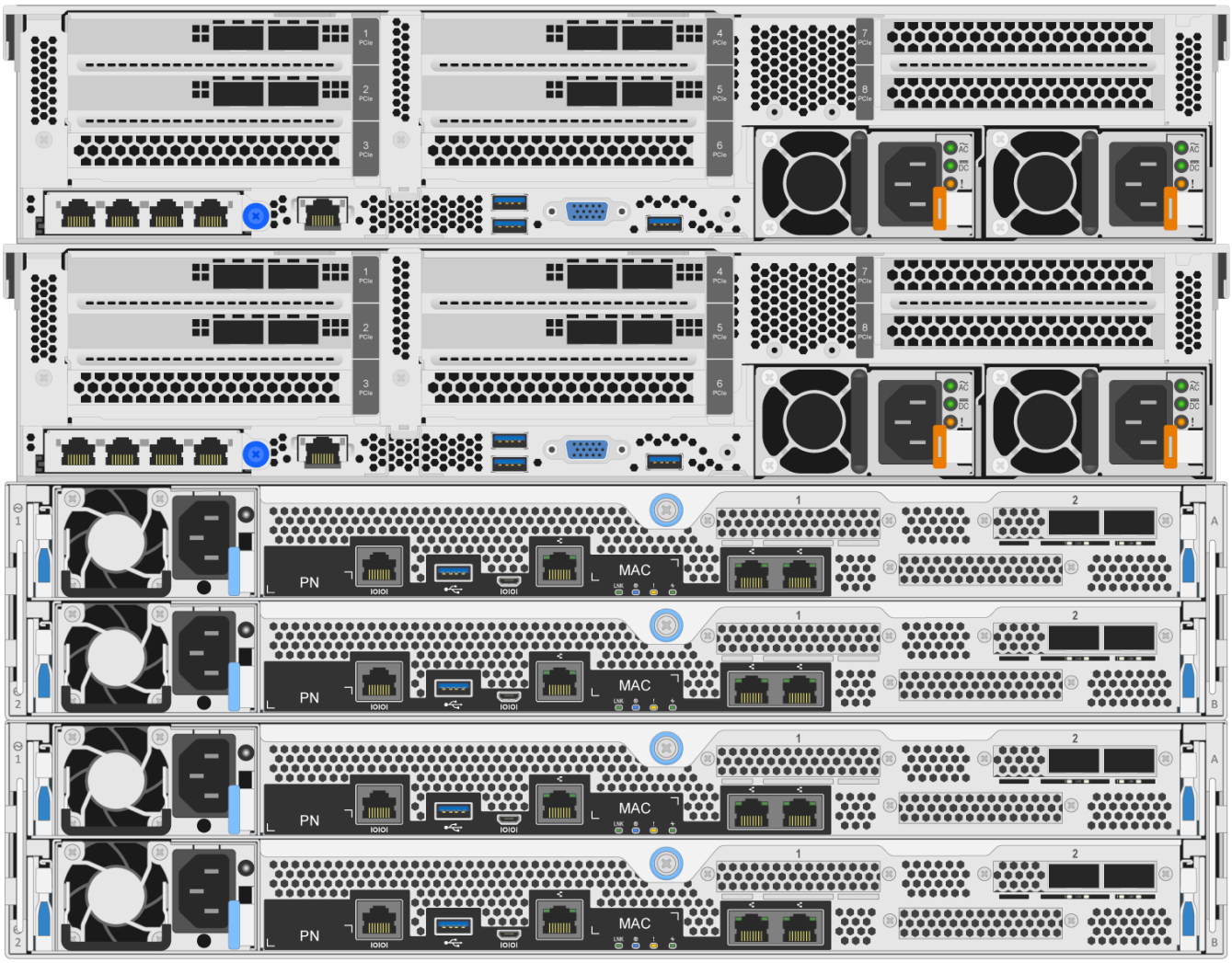
Une autre architecture de cluster haute disponibilité utilise une structure de stockage (également appelée réseau SAN) entre les nœuds de fichiers et de blocs pour établir un cluster symétrique. Cela permet aux services BeeGFS de s'exécuter sur n'importe quel nœud de fichiers d'un cluster HA particulier. En général, les clusters symétriques ne sont pas aussi économiques en raison du matériel SAN supplémentaire, cette documentation suppose l'utilisation d'un cluster asymétrique déployé en tant que série d'un ou plusieurs éléments de base.

Assurez-vous que l'architecture de système de fichiers souhaitée pour un déploiement BeeGFS particulier est bien comprise avant de poursuivre l'installation.

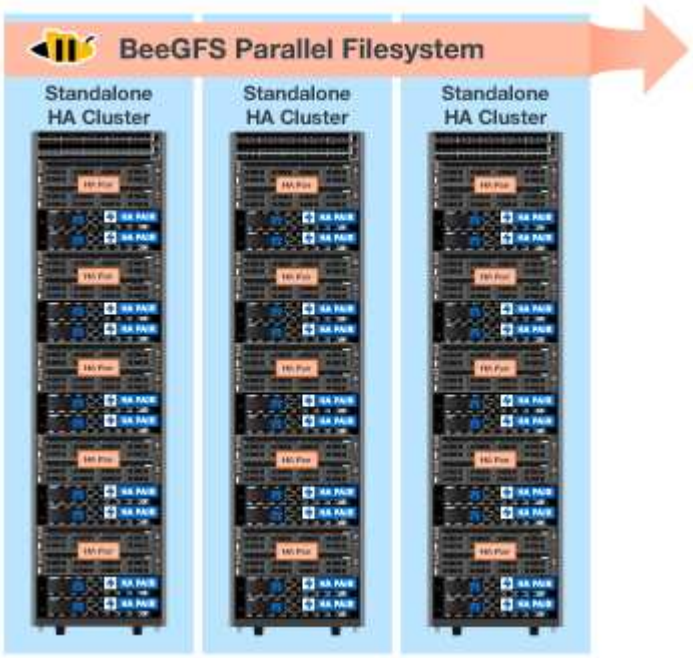
Matériel en rack

Lors de la planification de l'installation, il est important que tous les équipements de chaque élément de base soient montés en rack dans des unités adjacentes. Il est recommandé d'installer les nœuds de fichiers immédiatement au-dessus des nœuds de blocs dans chaque élément de base. Suivez la documentation du ou des modèles de fichier et "bloc" les nœuds que vous utilisez lorsque vous installez des rails et du matériel dans le rack.

Exemple d'un élément de base unique :

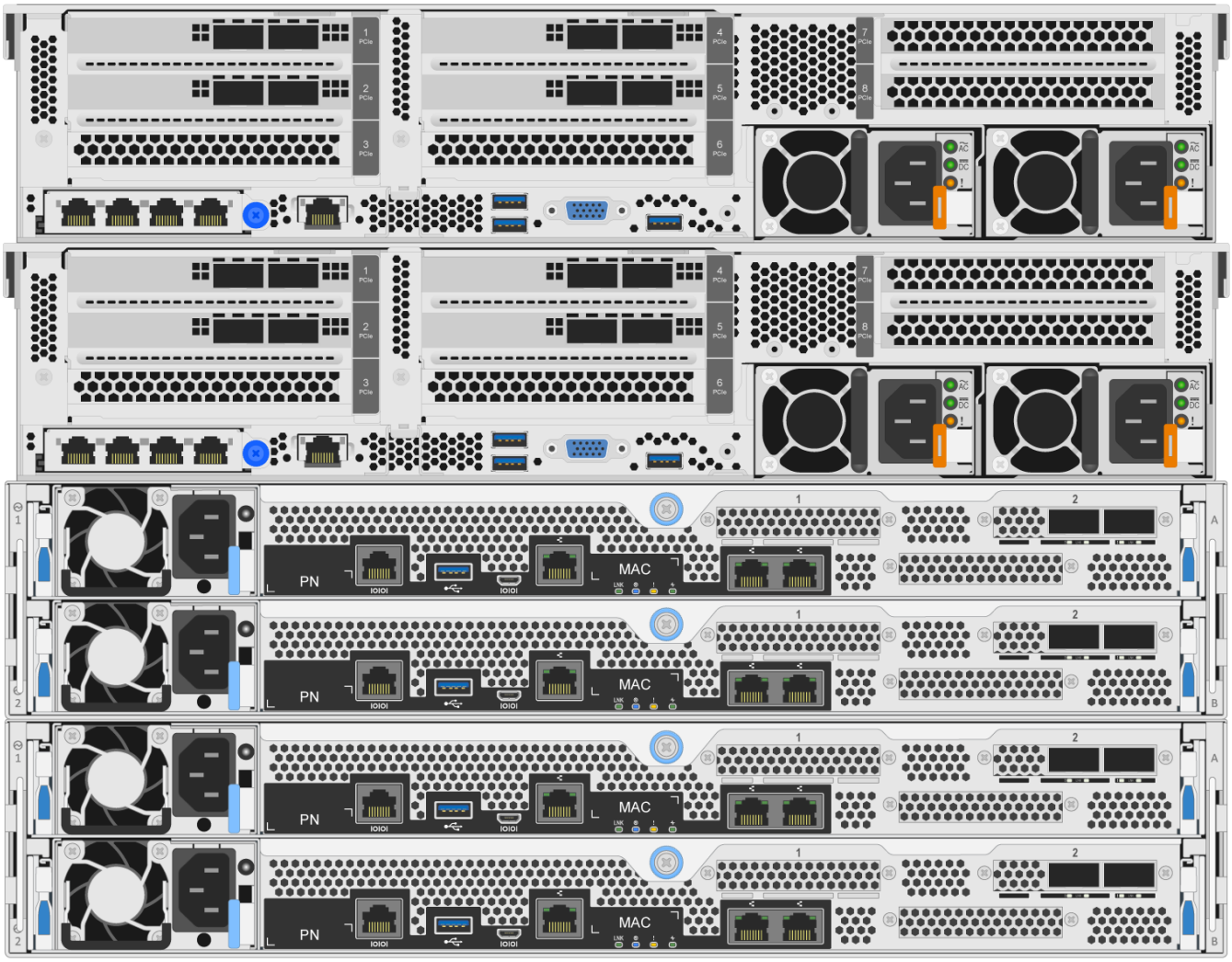


Exemple d’installation BeeGFS où il y a plusieurs éléments de base dans chaque cluster HA et plusieurs clusters HA dans le système de fichiers :



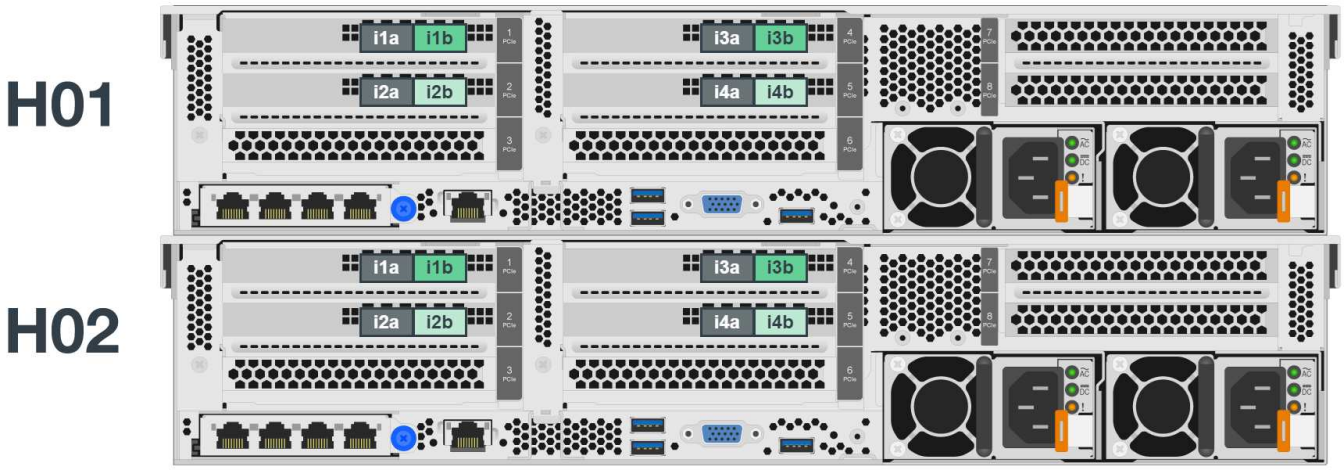
Nœud de bloc et fichier de câble

En général, vous connecterons directement les ports HIC des nœuds de blocs E-Series à l'adaptateur Channel hôte désigné (pour les protocoles InfiniBand) ou aux ports d'adaptateur bus hôte (pour les protocoles Fibre Channel et autres) des nœuds de fichiers. La façon exacte d'établir ces connexions dépendra de l'architecture de système de fichiers souhaitée, voici un exemple "Basé sur l'architecture vérifiée NetApp, BeeGFS sur NetApp":



Nœuds de fichiers de câble vers le réseau client

Chaque nœud de fichier aura un certain nombre de ports InfiniBand ou Ethernet désignés pour le trafic client BeeGFS. Selon l'architecture, chaque nœud de fichiers dispose d'une ou plusieurs connexions à un réseau client/de stockage hautes performances, avec possibilité de recourir à plusieurs commutateurs pour assurer la redondance et augmenter la bande passante. Voici un exemple de câblage client utilisant des commutateurs réseau redondants, où les ports en vert foncé et en vert clair sont connectés à des commutateurs distincts :



Connexion réseau et alimentation de gestion

Établissez toutes les connexions réseau nécessaires pour les réseaux intrabande et hors bande.

Connectez tous les blocs d'alimentation en vous assurant que chaque nœud de fichier et de bloc dispose de connexions à plusieurs unités de distribution d'alimentation pour la redondance (si disponible).

Configurez les nœuds de fichier et de bloc

Étapes manuelles nécessaires pour configurer les nœuds de blocs et de fichiers avant d'exécuter Ansible.

Nœuds de fichiers

Configuration du contrôleur BMC (Baseboard Management Controller)

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 XCC (Lenovo XClarity Controller).

Suivez la documentation du fournisseur du serveur pour activer toutes les licences nécessaires pour accéder à cette fonctionnalité et vérifier que le contrôleur BMC est connecté au réseau et configuré de manière appropriée pour l'accès à distance.

> Si vous souhaitez utiliser Redfish pour l'escrime basé sur BMC, assurez-vous que Redfish est activé et que l'interface BMC est accessible à partir du système d'exploitation installé sur le nœud de fichiers. Une configuration spéciale peut être nécessaire sur le commutateur réseau si le contrôleur BMC et le système d'exploitation partagent la même interface réseau physique.

Réglage des paramètres système

À l'aide de l'interface de configuration du système (BIOS/UEFI), assurez-vous que les paramètres sont définis pour optimiser les performances. Les paramètres exacts et les valeurs optimales varient en fonction du modèle de serveur utilisé. Des conseils sont fournis pour "modèles de nœud de fichier vérifiés", sinon reportez-vous à la documentation du fournisseur du serveur et aux meilleures pratiques en fonction de votre modèle.

Installer un système d'exploitation

Installez un système d'exploitation pris en charge en fonction de la configuration requise pour "ici"le nœud de fichiers indiquée . Reportez-vous aux étapes supplémentaires ci-dessous en fonction de votre distribution Linux.

Red Hat

, voir "Comment enregistrer et souscrire un système RHEL" et "Comment limiter les mises à jour" .

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
```

Configurer le réseau de gestion

Configurez toutes les interfaces réseau nécessaires pour permettre la gestion intrabande du système d'exploitation. Les étapes exactes dépendent de la distribution et de la version spécifiques de Linux utilisées.

Assurez-vous que SSH est activé et que toutes les interfaces de gestion sont accessibles depuis le nœud de contrôle Ansible.

Mettre à jour le micrologiciel HCA et HBA

Assurez-vous que tous les HBA et les HCA exécutent les versions de micrologiciel prises en charge répertoriées sur le "Matrice d'interopérabilité NetApp" et mettez à niveau si nécessaire. Des recommandations supplémentaires pour les adaptateurs NVIDIA ConnectX sont disponibles "ici".

Nœuds de blocs

Suivez les étapes à "Mise en service de la gamme E-Series" pour configurer le port de gestion sur chaque contrôleur de nœud de bloc et définir éventuellement le nom de la matrice de stockage pour chaque système.

Aucune configuration supplémentaire ne s'applique à garantir que tous les nœuds de bloc sont accessibles depuis le nœud de contrôle Ansible. La configuration système restante sera appliquée/gérée à l'aide d'Ansible.

Configurez le nœud de contrôle Ansible

Configurez un nœud de contrôle Ansible pour déployer et gérer le système de fichiers.

Présentation

Un nœud de contrôle Ansible est une machine Linux physique ou virtuelle utilisée pour gérer le cluster. Il doit répondre aux exigences suivantes :

- Rencontrez le "de formation"rôle haute disponibilité BeeGFS, y compris les versions installées d'Ansible, Python et tous les packages Python supplémentaires.
- Rencontrez l'agent "Configuration requise pour le nœud de contrôle Ansible" y compris les versions de système d'exploitation.

- Accès SSH et HTTPS à tous les nœuds de fichiers et de blocs.

Les étapes d'installation détaillées "ici"sont disponibles .

Définissez le système de fichiers BeeGFS

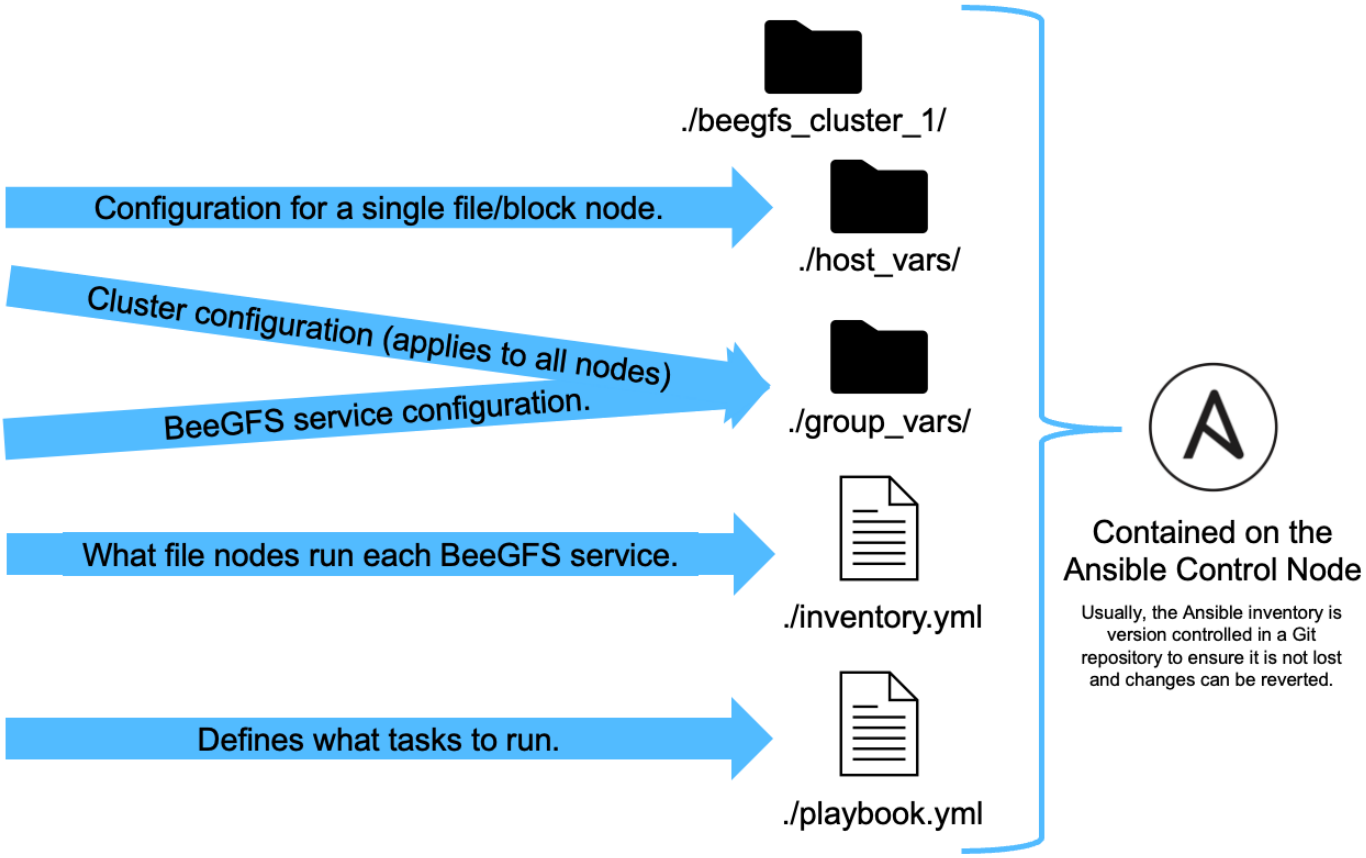
Présentation d'Ansible Inventory

L'inventaire Ansible est un ensemble de fichiers de configuration qui définissent le cluster BeeGFS HA souhaité.

Présentation

Il est recommandé de suivre les pratiques Ansible standard pour l'organisation de votre "inventaire", y compris l'utilisation de "sous-répertoires/fichiers" au lieu de stocker l'intégralité de l'inventaire dans un seul fichier.

L'inventaire Ansible pour un seul cluster BeeGFS HA est organisé comme suit :



Comme un seul système de fichiers BeeGFS peut s'étendre sur plusieurs clusters HA, il est possible aux grandes installations de disposer de plusieurs inventaires Ansible. En règle générale, il n'est pas recommandé de définir plusieurs clusters HA en tant qu'inventaire Ansible unique pour éviter tout problème.

Étapes

1. Sur votre nœud de contrôle Ansible, créez un répertoire vide qui contiendra l'inventaire Ansible pour le

cluster BeeGFS que vous souhaitez déployer.

a. Si votre système de fichiers contient plusieurs clusters haute disponibilité, il est recommandé de créer d'abord un répertoire pour le système de fichiers, puis des sous-répertoires pour l'inventaire représentant chaque cluster haute disponibilité. Par exemple :

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

2. Dans le répertoire contenant l'inventaire du cluster HA que vous souhaitez déployer, créez deux répertoires `group_vars` et `host_vars` et deux fichiers `inventory.yml` et `playbook.yml`.

Les sections suivantes décrivent la définition du contenu de chacun de ces fichiers.

Planifiez le système de fichiers

Planifiez le déploiement du système de fichiers avant de créer l'inventaire Ansible.

Présentation

Avant de déployer le système de fichiers, vous devez définir les adresses IP, les ports et autres configurations requis par tous les nœuds de fichiers, les nœuds de bloc et les services BeeGFS s'exécutant dans le cluster. La configuration exacte varie en fonction de l'architecture du cluster, mais cette section définit les meilleures pratiques et les étapes à suivre généralement applicables.

Étapes

1. Si vous utilisez un protocole de stockage IP (iser, iSCSI, NVMe/IB ou NVMe/RoCE) pour connecter les nœuds de fichiers aux nœuds de bloc, remplissez la fiche suivante pour chaque élément de base. Chaque connexion directe dans un seul bloc de construction doit disposer d'un sous-réseau unique et ne doit pas se chevaucher avec les sous-réseaux utilisés pour la connectivité client-serveur.

Nœud de fichier	Port IB	Adresse IP	Nœud de bloc	Port IB	IP physique	Adresse IP virtuelle (pour EF600 avec HDR IB uniquement)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

Si les nœuds de fichier et de bloc de chaque module sont directement connectés, vous pouvez souvent réutiliser les mêmes adresses IP/schéma pour plusieurs éléments de base.

2. Que vous utilisiez InfiniBand ou RDMA over Converged Ethernet (RoCE) pour le réseau de stockage, remplissez la fiche suivante pour déterminer les plages IP qui seront utilisées pour les services de cluster HA, les services de fichiers BeeGFS et les clients pour communiquer :

Objectif	Port InfiniBand	Adresse IP ou plage
IP(s) de cluster BeeGFS	<INTERFACE(s)>	<RANGE>
Gestion BeeGFS	<INTERFACE(s)>	<IP(s)>
Métadonnées BeeGFS	<INTERFACE(s)>	<RANGE>
Stockage BeeGFS	<INTERFACE(s)>	<RANGE>
Clients BeeGFS	<INTERFACE(s)>	<RANGE>

a. Si vous utilisez un seul sous-réseau IP, une seule feuille de calcul est nécessaire. Sinon, remplissez également la feuille de calcul du second sous-réseau.

3. En fonction de ce qui précède, pour chaque élément de base du cluster, remplissez la feuille de travail suivante définissant les services BeeGFS qu'il exécutera. Pour chaque service, spécifiez le ou les nœuds de fichiers préférés/secondaires, le port réseau, les adresses IP flottantes, l'affectation de zone NUMA (si nécessaire) et le ou les nœuds de bloc qui seront utilisés pour ses cibles. Reportez-vous aux directives suivantes lorsque vous remplissez la fiche :

   a. Spécifiez les services BeeGFS comme l'un ou l'autre `mgmt.yml`, `meta_<ID>.yml`, ou `storage_<ID>.yml` Où ID représente un nombre unique pour tous les services BeeGFS de ce type dans ce système de fichiers. Cette convention simplifie le renvoi à cette feuille de calcul dans les sections suivantes tout en créant des fichiers pour configurer chaque service.

   b. Les ports pour les services BeeGFS ne doivent être uniques qu'à travers un élément de construction particulier. Assurez-vous que les services ayant le même numéro de port ne peuvent jamais être exécutés sur le même nœud de fichier pour éviter les conflits de ports.

   c. Si nécessaire, les services peuvent utiliser des volumes de plusieurs nœuds de blocs et/ou pool de stockage (et tous les volumes ne doivent pas être détenus par le même contrôleur). Plusieurs services peuvent également partager la même configuration de nœud de bloc et/ou de pool de stockage (des volumes individuels seront définis dans une section ultérieure).

Service BeeGFS (nom de fichier)	Nœuds de fichiers	Port	Adresses IP flottantes	Zone NUMA	Nœud de bloc	Pool de stockage	Contrôleur propriétaire
<SERVICE TYPE>_<ID>.yml	<PREFERRED FILE NODE> <SECONDARY FILE NODE(s)>	<PORT>	<INTERFACE>:<IP/SUBNET> <INTERFACE>:<IP/SUBNET>	<NUMA NODE/ZONE>	<BLOCK NODE>	<STORAGE POOL/VOLUME GROUP>	<A OR B>

Pour en savoir plus sur les conventions standard, les bonnes pratiques "et des meilleures pratiques"et les feuilles de calcul fournies, consultez "Définir des éléments de base BeeGFS"les sections et de l'architecture vérifiée BeeGFS sur NetApp.

Définir les nœuds de fichier et de bloc

Configurer des nœuds de fichiers individuels

Spécifiez la configuration des nœuds de fichiers individuels à l'aide de variables hôte (Host_var).

Présentation

Cette section décrit le remplissage d'un `host_vars/<FILE_NODE_HOSTNAME>.yml` fichier pour chaque nœud de fichiers du cluster. Ces fichiers ne doivent contenir qu'une configuration unique à un nœud de fichier particulier. Les points suivants sont généralement utilisés :

- Définition de l'IP ou du nom d'hôte Ansible doit utiliser pour se connecter au nœud.
- Configuration d'interfaces supplémentaires et d'adresses IP de cluster utilisées pour les services de cluster HA (Pacemaker et Corosync) pour communiquer avec d'autres nœuds de fichiers. Par défaut, ces services utilisent le même réseau que l'interface de gestion, mais des interfaces supplémentaires doivent être disponibles pour la redondance. La pratique courante consiste à définir des adresses IP supplémentaires sur le réseau de stockage, ce qui évite d'avoir recours à un cluster ou à un réseau de gestion supplémentaire.
  - Les performances des réseaux utilisés pour la communication en cluster ne sont pas critiques pour les performances du système de fichiers. Avec la configuration de cluster par défaut, un réseau d'au moins 1 Gbit/s fournit généralement des performances suffisantes pour les opérations de cluster telles que la synchronisation des États de nœud et la coordination des modifications de l'état des ressources du cluster. Les réseaux lents/occupés peuvent entraîner des changements d'état des ressources qui prennent plus de temps que d'habitude. Dans des cas extrêmes, les nœuds risquent d'être supprimés du cluster s'ils ne peuvent pas envoyer des signaux cardiaques dans des délais raisonnables.
- Configuration des interfaces utilisées pour la connexion aux nœuds de bloc via le protocole souhaité (par exemple : iSCSI/iser, NVMe/IB, NVMe/RoCE, FCP, etc.)

Étapes

En faisant référence au schéma d'adressage IP défini dans la "Planifiez le système de fichiers" section, pour chaque nœud de fichier du cluster, créez un fichier `host_vars/<FILE_NODE_HOSTNAME>/yml` et remplissez-le comme suit :

1. En haut de la page, indiquez l'IP ou le nom d'hôte Ansible doit utiliser pour SSH sur le nœud et le gérer :

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

2. Configurez des adresses IP supplémentaires qui peuvent être utilisées pour le trafic du cluster :
   a. Si le type de réseau est "InfiniBand (avec IPoib)":

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

   b. Si le type de réseau est "RDMA over Converged Ethernet (RoCE)":

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

c. Si le type de réseau est "Ethernet (TCP uniquement, pas de RDMA)":

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

3. Indiquez quelles adresses IP doivent être utilisées pour le trafic du cluster, avec des adresses IP préférées répertoriées plus haut :

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

IPS configuré à l'étape deux ne sera pas utilisé comme adresses IP de cluster à moins qu'elles ne soient incluses dans le `beegfs_ha_cluster_node_ips` liste. Cela vous permet de configurer des adresses IP/interfaces supplémentaires à l'aide d'Ansible, qui peuvent être utilisées à d'autres fins si nécessaire.

4. Si le nœud de fichiers doit communiquer pour bloquer les nœuds via un protocole IP, les adresses IP doivent être configurées sur l'interface appropriée, ainsi que tous les packages requis pour ce protocole installés/configurés.

a. En cas d'utilisation "ISCSI":

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

b. En cas d'utilisation "Iser":

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

c. En cas d'utilisation "NVMe/IB":

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

d. En cas d'utilisation "NVMe/RoCE":

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

e. Autres protocoles :

   i. En cas d'utilisation "NVMe/FC", il n'est pas nécessaire de configurer des interfaces individuelles. Le déploiement du cluster BeeGFS détecte automatiquement le protocole et installe/configure les exigences selon les besoins. Si vous utilisez une structure pour connecter des nœuds de fichiers et de blocs, assurez-vous que les commutateurs sont correctement zonés en suivant les meilleures pratiques de NetApp et de votre fournisseur de commutateurs.

   ii. L'utilisation de FCP ou de SAS ne nécessite pas l'installation ou la configuration de logiciels supplémentaires. Si vous utilisez FCP, assurez-vous que les commutateurs sont correctement zonés suivant "NetApp" et les meilleures pratiques de votre fournisseur de commutateurs.

   iii. L'utilisation du SRP IB n'est pas recommandée pour le moment. Utilisez les technologies NVMe/IB ou iser selon les systèmes que prennent en charge les nœuds bloc E-Series.

Cliquez sur "ici" par exemple, un fichier d'inventaire complet représentant un nœud de fichier unique.

Avancé : basculement des adaptateurs VPI NVIDIA ConnectX entre Ethernet et InfiniBand

Les adaptateurs NVIDIA ConnectX-Virtual Protocol Interconnect&reg (VPI) prennent en charge les protocoles InfiniBand et Ethernet comme couche de transport. Le passage d'un mode à l'autre n'est pas négocié automatiquement et doit être configuré à l'aide de `mstconfig` l'outil inclus dans `mstflint`, un package open source qui fait partie du "Outils de fermeté NVIDIA (MFT)". La modification du mode des adaptateurs ne doit être effectuée qu'une seule fois. Cette opération peut être effectuée manuellement ou incluse dans l'inventaire Ansible dans le cadre de toute interface configurée à l'aide de la `eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:` section de l'inventaire pour que cette opération soit vérifiée/appliquée automatiquement.

Par exemple, pour modifier le courant d'une interface en mode InfiniBand en Ethernet et pouvoir l'utiliser pour

RoCE :

1. Pour chaque interface que vous souhaitez configurer, spécifiez `mstconfig` en tant que mappage (ou dictionnaire) spécifié `LINK_TYPE_P<N>` où `<N>` Est déterminé par le numéro de port HCA de l'interface. Le `<N>` la valeur peut être déterminée en cours d'exécution `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` Et ajout de 1 au dernier numéro à partir du nom du slot PCI et conversion en décimal.

   a. Par exemple donné `PCI_SLOT_NAME=0000:2f:00.2` (2 + 1 → port HCA 3) → `LINK_TYPE_P3: eth`:

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

Pour plus de détails, reportez-vous au "Documentation de la collection d'hôtes NetApp E-Series" pour le type/protocole d'interface que vous utilisez.

Configurez des nœuds de bloc individuels

Spécifiez la configuration pour les nœuds de bloc individuels à l'aide de variables hôte (Host_var).

Présentation

Cette section décrit le remplissage d'un `host_vars/<BLOCK_NODE_HOSTNAME>.yml` fichier pour chaque nœud de bloc du cluster. Ces fichiers ne doivent contenir qu'une configuration unique à un nœud de bloc particulier. Les points suivants sont généralement utilisés :

- Nom du système (tel qu'il s'affiche dans System Manager).
- L'URL HTTPS pour l'un des contrôleurs (utilisée pour gérer le système à l'aide de son API REST).
- Quels nœuds de fichiers de protocole de stockage utilisent pour se connecter à ce nœud en mode bloc ?
- Configuration des ports HIC (carte d'interface hôte), tels que les adresses IP (si nécessaire).

Étapes

En référençant le schéma d'adressage IP défini dans la "Planifiez le système de fichiers" section, pour chaque nœud de bloc du cluster, créez un fichier `host_vars/<BLOCK_NODE_HOSTNAME>/yml` et remplissez-le comme suit :

1. En haut de la page, spécifier le nom du système et l'URL HTTPS pour l'un des contrôleurs :

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

2. Sélectionner "protocole" les nœuds de fichiers seront utilisés pour se connecter à ce nœud de bloc :

   a. Protocoles pris en charge : `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

   ```
   eseries_initiator_protocol: <PROTOCOL>
   ```

3. Selon le protocole utilisé, les ports HIC peuvent nécessiter des configurations supplémentaires. Si nécessaire, la configuration du port HIC doit être définie de sorte que l'entrée supérieure de la configuration de chaque contrôleur corresponde au port physique le plus à gauche de chaque contrôleur et au port inférieur le plus à droite. Tous les ports nécessitent une configuration valide même s'ils ne sont pas actuellement utilisés.

   > ⓘ Reportez-vous également à la section ci-dessous si vous utilisez des nœuds de bloc EF600 (InfiniBand 200 Go) ou RoCE 200 Go avec les nœuds de bloc EF600.

   a. Pour iSCSI :

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

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

b. Pour iser :

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

c. Pour NVMe/IB :

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

d. Pour NVMe/RoCE :

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

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

e. Les protocoles FC et SAS ne nécessitent pas de configuration supplémentaire. SRP n'est pas recommandé correctement.

Pour plus d'options de configuration des ports HIC et des protocoles hôte, notamment pour configurer iSCSI CHAP, reportez-vous au "documentation" Inclus avec la collection SANtricity. Remarque lors du déploiement de BeeGFS, la configuration du pool de stockage, du volume et d'autres aspects du provisionnement du stockage sont configurés ailleurs et ne doivent pas être définis dans ce fichier.

Cliquez sur "ici" par exemple, un fichier d'inventaire complet représentant un nœud de bloc unique.

Avec l'utilisation de HDR (200 Go) InfiniBand ou RoCE 200 Gb avec les nœuds de bloc NetApp EF600 :

Pour utiliser l'InfiniBand HDR (200 Go) avec la baie EF600, une seconde IP « virtuelle » doit être configurée pour chaque port physique. Vous trouverez ci-dessous un exemple de configuration correcte d'une baie EF600 équipée du système HIC InfiniBand HDR à deux ports :

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

Spécifiez la configuration de nœud de fichier commun

Spécifiez une configuration de nœud de fichier commune à l'aide de variables de groupe (Group_var).

Présentation

La configuration qui doit pommier à tous les noeuds de fichiers est définie à `group_vars/ha_cluster.yml`. Elle comprend généralement :

- Détails sur la connexion et la connexion à chaque noeud de fichier.
- Configuration commune de réseau.
- Indique si les redémarrages automatiques sont autorisés.
- Configuration des États pare-feu et selinux.
- Configuration du cluster, y compris alertes et clôtures.
- Réglage des performances.
- Configuration du service Common BeeGFS.

Les options définies dans ce fichier peuvent également être définies sur des nœuds de fichiers individuels, par exemple si des modèles de matériel mixtes sont utilisés, ou si vous avez des mots de passe différents pour chaque nœud. La configuration des nœuds de fichiers individuels est prioritaire sur la configuration dans ce fichier.

Étapes

Créez le fichier `group_vars/ha_cluster.yml` et remplir comme suit :

1. Indiquez comment le nœud Ansible Control doit s'authentifier auprès des hôtes distants :

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

En particulier pour les environnements de production, ne stockez pas de mots de passe en texte brut. Utilisez plutôt Ansible Vault (voir "Cryptage de contenu avec Ansible Vault") ou le `--ask-become-pass` option lors de l'exécution du manuel de vente. Si le `ansible_ssh_user` est déjà root, alors vous pouvez éventuellement omettre le `ansible_become_password`.

2. Si vous configurez des adresses IP statiques sur des interfaces ethernet ou InfiniBand (par exemple, des adresses IP de cluster) et que plusieurs interfaces se trouvent dans le même sous-réseau IP (par exemple si ib0 utilise 192.168.1.10/24 et ib1 utilise 192.168.1.11/24), Des tables et des règles de routage IP supplémentaires doivent être configurées pour que le support multihomé fonctionne correctement. Activez simplement le crochet de configuration de l'interface réseau fourni comme suit :

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

3. Lors du déploiement du cluster, en fonction du protocole de stockage, les nœuds doivent être redémarrés pour faciliter la détection des dispositifs de blocs distants (volumes E-Series) ou l'application d'autres aspects de la configuration. Par défaut, les nœuds vous demandent avant de redémarrer, mais vous pouvez autoriser les nœuds à redémarrer automatiquement en spécifiant ce qui suit :

```
eseries_common_allow_host_reboot: true
```

   a. Par défaut après un redémarrage, pour vous assurer que les périphériques de bloc et les autres services sont prêts, Ansible attend jusqu'à ce que le système `default.target` est atteinte avant de poursuivre le déploiement. Dans certains cas, lorsque la technologie NVMe/IB est utilisée, il peut s'avérer insuffisant pour initialiser, détecter et se connecter aux périphériques distants. Cela peut entraîner une poursuite prématurée et une défaillance du déploiement automatisé. Pour éviter ce problème lors de l'utilisation de NVMe/IB, définissez également les éléments suivants :

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

4. Plusieurs ports de pare-feu sont nécessaires pour que les services de cluster BeeGFS et HA communiquent. Sauf si vous souhaitez configurer le firwewall manuellement (non recommandé), spécifiez les zones de pare-feu requises et les ports ouverts automatiquement :

```
beegfs_ha_firewall_configure: True
```

5. SELinux n'est pas encore pris en charge et il est recommandé de définir l'état sur Désactivé pour éviter les conflits (en particulier lorsque RDMA est en cours d'utilisation). Définissez ce qui suit pour vous assurer que SELinux est désactivé :

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

6. Configurez l'authentification pour que les nœuds de fichiers puissent communiquer, en ajustant les valeurs par défaut en fonction des règles de votre entreprise :

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

7. Sur la base de "Planifiez le système de fichiers"cette section, spécifiez l'IP de gestion BeeGFS pour ce système de fichiers :

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

> 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.

8. Activez les alertes par e-mail si vous le souhaitez :

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

9. L'activation de l'escrime est fortement recommandée, sinon les services peuvent être bloqués afin de démarrer sur les nœuds secondaires lorsque le nœud principal tombe en panne.
   a. Activer globalement l'escrime en spécifiant les éléments suivants :

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

      i. Remarque tout support "propriété du cluster" peut également être spécifié ici si nécessaire. Ils ne sont généralement pas nécessaires, car le rôle haute disponibilité BeeGFS est livré avec un certain nombre de composants bien testés "valeurs par défaut".

   b. Sélectionnez et configurez ensuite un agent d'escrime :
      i. OPTION 1 : pour activer l'escrime à l'aide des unités de distribution d'alimentation APC :

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

      ii. OPTION 2 : pour activer l'escrime à l'aide des API Redfish fournies par le XCC Lenovo (et d'autres CVM) :

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

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

iii. Pour plus de détails sur la configuration d'autres agents de clôture, reportez-vous au "Documentation Red Hat".

10. Le rôle BeeGFS HA peut appliquer de nombreux paramètres de réglage différents pour optimiser davantage les performances. Ces paramètres incluent notamment l'optimisation de l'utilisation de la mémoire du noyau et le blocage des E/S du périphérique. Le rôle est fourni avec un ensemble raisonnable de "valeurs par défaut" tests basés sur des tests avec des nœuds de bloc NetApp E-Series, mais par défaut, ces tests ne sont pas appliqués sauf si vous spécifiez :

```
beegfs_ha_enable_performance_tuning: True
```

a. Si nécessaire, spécifiez également les modifications apportées au réglage de performance par défaut ici. Pour plus d'informations, reportez-vous à la documentation complète "paramètres d'ajustement des performances" .

11. Pour garantir que les adresses IP flottantes (parfois appelées interfaces logiques) utilisées pour les services BeeGFS peuvent basculer entre les nœuds de fichiers, toutes les interfaces réseau doivent être nommées de façon cohérente. Par défaut, les noms d'interface réseau sont générés par le noyau, qui n'est pas garanti de générer des noms cohérents, même sur des modèles de serveurs identiques avec des cartes réseau installées dans les mêmes slots PCIe. Cela est également utile lors de la création d'inventaires avant le déploiement de l'équipement et la génération de noms d'interfaces connus. Pour garantir des noms de périphériques cohérents, en fonction d'un schéma fonctionnel du serveur ou `lshw -class network -businfo` Sortie, spécifiez le mappage adresse PCIe vers interface logique souhaité comme suit :

a. Pour les interfaces réseau InfiniBand (IPoIB) :

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

b. Pour les interfaces réseau Ethernet :

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

Pour éviter les conflits lorsque les interfaces sont renommées (les empêchant d'être renommées), vous ne devez pas utiliser de noms par défaut potentiels tels que eth0, en9 f0, ib0 ou ibs4f0. la convention de nom la plus courante consiste à utiliser « e » ou « i » pour Ethernet ou InfiniBand, suivi du numéro de connecteur PCIe, ainsi qu'une lettre indiquant le port. Par exemple, le deuxième port d'un adaptateur InfiniBand installé dans le logement 3 est : i3b.

Si vous utilisez un modèle de nœud de fichier vérifié, cliquez sur "ici" Exemples de mappages de port adresse PCIe vers port logique.

12. Spécifiez éventuellement la configuration qui doit s'appliquer à tous les services BeeGFS dans le cluster. Les valeurs de configuration par défaut sont disponibles "ici"et la configuration par service est spécifiée ailleurs :
    a. Service de gestion BeeGFS :

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

    b. Les services de métadonnées BeeGFS :

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

    c. BeeGFS Services de stockage :

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

13. Depuis BeeGFS 7.2.7 et 7.3.1 "authentification de la connexion" doit être configuré ou explicitement désactivé. Il existe quelques façons de configurer ce système à l'aide du déploiement Ansible :
    a. Par défaut, le déploiement configure automatiquement l'authentification de connexion et génère un `connauthfile` Qui sera distribué à tous les nœuds de fichiers et utilisé avec les services BeeGFS. Ce fichier sera également placé/conservé sur le nœud de contrôle Ansible à `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` où il doit être conservé (sécurisé) pour être réutilisé avec les clients qui doivent accéder à ce système de fichiers.
        i. Pour générer une nouvelle clé spécifiez `-e "beegfs_ha_conn_auth_force_new=True` Lors de l'exécution du manuel de vente Ansible. Remarque cette opération est ignorée si une `beegfs_ha_conn_auth_secret` est défini.
        ii. Pour les options avancées, reportez-vous à la liste complète des valeurs par défaut incluses avec le "Rôle BeeGFS HA".

b. Un secret personnalisé peut être utilisé en définissant les éléments suivants dans `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. L'authentification de la connexion peut être entièrement désactivée (NON recommandée) :

```
beegfs_ha_conn_auth_enabled: false
```

Cliquez sur "ici" par exemple, un fichier d'inventaire complet représentant la configuration de nœud de fichier commune.

Utilisation de la technologie InfiniBand HDR (200 Go) avec des nœuds de bloc NetApp EF600 :

Pour utiliser l'InfiniBand HDR (200 Go) avec l'EF600, le gestionnaire de sous-réseau doit prendre en charge la virtualisation. Si les nœuds de fichiers et de blocs sont connectés à l'aide d'un commutateur, celui-ci doit être activé sur le gestionnaire de sous-réseau pour la structure globale.

Si les nœuds de blocs et de fichiers sont directement connectés via InfiniBand, une instance de `opensm` doit être configurée sur chaque nœud de fichiers pour chaque interface directement connectée à un nœud de bloc. Pour ce faire, spécifiez `configure: true` quand "configuration des interfaces de stockage de nœud de fichiers".

Actuellement, la version intégrée de `opensm` fournie avec les distributions Linux prises en charge ne prend pas en charge la virtualisation. Il est nécessaire d'installer et de configurer la version de `opensm` à partir de NVIDIA OpenFabrics Enterprise distribution (OFED). Même si le déploiement avec Ansible est toujours pris en charge, quelques étapes supplémentaires sont requises :

1. À l'aide de curl ou de l'outil de votre choix, téléchargez les packages de la version d'OpenSM répertoriée dans la "exigences technologiques" section du site Web de NVIDIA vers le `<INVENTORY>/packages/` répertoire. Par exemple :

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

2. Sous `group_vars/ha_cluster.yml` définissez la configuration suivante :

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

eseries_ib_opensm_options:
  virt_enabled: "2"
```

Spécifiez la configuration de nœud de bloc commun

Spécifiez une configuration de nœud de bloc commune à l'aide de variables de groupe (Group_var).

Présentation

La configuration qui doit être pommeltée à tous les nœuds de bloc est définie à `group_vars/eseries_storage_systems.yml`. Elle comprend généralement :

- Découvrez comment le nœud de contrôle Ansible doit se connecter aux systèmes de stockage E-Series utilisés comme nœuds de bloc.

- Quelles versions de micrologiciel, de NVSRAM et de micrologiciel de lecteur les nœuds doivent s'exécuter.
- Configuration globale comprenant les paramètres de cache, la configuration de l'hôte et les paramètres de provisionnement des volumes.

> Les options définies dans ce fichier peuvent également être définies sur des nœuds de blocs individuels, par exemple si des modèles de matériel mixtes sont en cours d'utilisation, ou si vous disposez de mots de passe différents pour chaque nœud. La configuration des nœuds de blocs individuels aura priorité sur la configuration dans ce fichier.

Étapes

Créez le fichier `group_vars/eseries_storage_systems.yml` et remplir comme suit :

1. Ansible n'utilise pas SSH pour se connecter aux nœuds de bloc et utilise plutôt des API REST. Pour ce faire, nous devons définir :

```
ansible_connection: local
```

2. Spécifiez le nom d'utilisateur et le mot de passe pour gérer chaque nœud. Le nom d'utilisateur peut éventuellement être omis (et sera par défaut admin), sinon vous pouvez spécifier n'importe quel compte disposant de privilèges d'administrateur. Spécifiez également si les certificats SSL doivent être vérifiés ou ignorés :

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

> La liste des mots de passe en texte clair n'est pas recommandée. Utilisez un coffre-fort Ansible ou fournissez le `eseries_system_password` Lors de l'exécution d'Ansible à l'aide de --extra-var.

3. Spécifiez éventuellement le micrologiciel du contrôleur, la NVSRAM et le micrologiciel du lecteur qui doivent être installés sur les nœuds. Ils devront être téléchargés sur le `packages/` Avant d'exécuter Ansible. Le micrologiciel du contrôleur E-Series et la NVSRAM sont téléchargés "ici" et les firmwares des disques "ici":

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

Si cette configuration est spécifiée, Ansible met automatiquement à jour l'ensemble du firmware, y compris le redémarrage des contrôleurs (si nécessaire) sans invite supplémentaire. Cela ne devrait pas être perturbateur des E/S de l'hôte BeeGFS/O, mais peut entraîner une diminution temporaire des performances.

4. Réglez les paramètres de configuration globale du système par défaut. Les options et valeurs répertoriées ici sont généralement recommandées pour BeeGFS sur NetApp, mais elles peuvent être ajustées si nécessaire :

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

5. Configurez les valeurs par défaut du provisionnement global du volume. Les options et valeurs répertoriées ici sont généralement recommandées pour BeeGFS sur NetApp, mais elles peuvent être ajustées si nécessaire :

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

6. Si nécessaire, ajustez l'ordre dans lequel Ansible sélectionne les disques pour les pools de stockage et les groupes de volumes, en gardant à l'esprit les meilleures pratiques suivantes :

   a. Répertoriez tous les disques (potentiellement plus petits) à utiliser en premier pour les volumes de gestion et/ou de métadonnées, et tous les volumes de stockage en dernier.

b. Veillez à équilibrer l'ordre de sélection des disques sur les canaux disponibles en fonction du ou des modèles de tiroir disque/boîtier(s). Par exemple avec la baie EF600 sans extensions, les disques 0-11 se trouvent sur le canal 1, et les disques 12-23 sur le canal disque. Ainsi, une stratégie pour équilibrer la sélection de l'entraînement est à sélectionner `disk shelf:drive` 99:0, 99:23, 99:1, 99:22, etc Si plusieurs armoires sont disponibles, le premier chiffre représente l'ID de tiroir disque.

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

Cliquez sur "ici" par exemple, un fichier d'inventaire complet représentant la configuration de nœud de bloc commune.

Définir les services BeeGFS

Définissez le service de gestion BeeGFS

Les services BeeGFS sont configurés à l'aide de variables de groupe (Group_var).

Présentation

Cette section décrit la définition du service de gestion BeeGFS. Un seul service de ce type doit exister dans le(s) cluster(s) haute disponibilité pour un système de fichiers particulier. La configuration de ce service inclut la définition des éléments suivants :

- Le type de service (gestion).
- Définition de toute configuration qui ne doit s'appliquer qu'à ce service BeeGFS.
- Configuration d'une ou plusieurs adresses IP flottantes (interfaces logiques) sur lesquelles ce service peut être atteint.
- Spécifier où/comment un volume doit être stocké des données pour ce service (cible de gestion BeeGFS).

Étapes

Créez un nouveau fichier `group_vars/mgmt.yml` et référencez la "Planifiez le système de fichiers" section pour la remplir comme suit :

1. Indiquez ce fichier représente la configuration d'un service de gestion BeeGFS :

```
beegfs_service: management
```

2. Définissez toute configuration qui doit s'appliquer uniquement à ce service BeeGFS. Ce n'est généralement pas nécessaire pour le service de gestion, sauf si vous devez activer des quotas, mais tout paramètre de configuration pris en charge à partir de `beegfs-mgmtd.conf` peut être inclus. Remarque les paramètres suivants sont configurés automatiquement/ailleurs et ne doivent pas être spécifiés ici : `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, et `connNetFilterFile`.

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

3. Configurez une ou plusieurs adresses IP flottantes que les autres services et clients utiliseront pour se connecter à ce service (cela définit automatiquement le BeeGFS `connInterfacesFile` option) :

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

4. Vous pouvez également spécifier un ou plusieurs sous-réseaux IP autorisés qui peuvent être utilisés pour les communications sortantes (cela va automatiquement définir BeeGFS `connNetFilterFile` option) :

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

5. Spécifiez la cible de gestion BeeGFS dans laquelle ce service stockera les données conformément aux directives suivantes :
   a. Le même nom de pool de stockage ou de groupe de volumes peut être utilisé pour plusieurs services/cibles BeeGFS, tout simplement s'assurer d'utiliser la même chose `name`, `raid_level`, `criteria_*`, et `common_*` configuration pour chaque service (les volumes répertoriés doivent être différents).
   b. La taille des volumes doit être indiquée comme pourcentage du groupe pool/volumes de stockage et le total ne doit pas dépasser 100 pour tous les services/volumes utilisant un pool/groupe de volumes spécifique. Remarque : lors de l'utilisation de disques SSD, il est recommandé de laisser un peu d'espace libre dans le groupe de volumes afin d'optimiser les performances et la durée de vie des disques SSD (cliquez "ici" pour plus de détails).
   c. Cliquez sur "ici" pour obtenir la liste complète des options de configuration disponibles pour le `eseries_storage_pool_configuration`. Notez certaines options telles que `state`, `host`, `host_type`, `workload_name`, et `workload_metadata` des noms de volume et de volume sont générés automatiquement et ne doivent pas être spécifiés ici.

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

Cliquez sur "ici" Par exemple un fichier d'inventaire complet représentant un service de gestion BeeGFS.

Définissez le service de métadonnées BeeGFS

Les services BeeGFS sont configurés à l'aide de variables de groupe (Group_var).

Présentation

Cette section décrit la définition du service de métadonnées BeeGFS. Au moins un service de ce type doit exister dans le(s) cluster(s) haute disponibilité pour un système de fichiers particulier. La configuration de ce service inclut la définition des éléments suivants :

- Le type de service (métadonnées).
- Définition de toute configuration qui ne doit s'appliquer qu'à ce service BeeGFS.
- Configuration d'une ou plusieurs adresses IP flottantes (interfaces logiques) sur lesquelles ce service peut être atteint.
- Spécifier où/comment un volume doit être stocké des données pour ce service (cible de métadonnées BeeGFS).

Étapes

En faisant référence "Planifiez le système de fichiers"à la section, créez un fichier à `group_vars/meta_<ID>.yml` pour chaque service de métadonnées du cluster et remplissez-le comme suit :

1. Indiquez ce fichier représente la configuration d'un service de métadonnées BeeGFS :

   ```
   beegfs_service: metadata
   ```

2. Définissez toute configuration qui doit s'appliquer uniquement à ce service BeeGFS. Au minimum, vous devez spécifier le port TCP et UDP de votre choix, mais tout paramètre de configuration pris en charge à partir de `beegfs-meta.conf` peut également être inclus. Remarque les paramètres suivants sont configurés automatiquement/ailleurs et ne doivent pas être spécifiés ici : `sysMgmtdHost`,

`storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, et `connNetFilterFile`.

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

3. Configurez une ou plusieurs adresses IP flottantes que les autres services et clients utiliseront pour se connecter à ce service (cela définit automatiquement le BeeGFS `connInterfacesFile` option) :

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

4. Vous pouvez également spécifier un ou plusieurs sous-réseaux IP autorisés qui peuvent être utilisés pour les communications sortantes (cela va automatiquement définir BeeGFS `connNetFilterFile` option) :

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

5. Spécifiez la cible de métadonnées BeeGFS dans laquelle ce service stockera les données conformément aux directives suivantes (ceci configurera également automatiquement l' `storeMetaDirectory` option) :
   a. Le même nom de pool de stockage ou de groupe de volumes peut être utilisé pour plusieurs services/cibles BeeGFS, tout simplement s'assurer d'utiliser la même chose `name`, `raid_level`, `criteria_*`, et `common_*` configuration pour chaque service (les volumes répertoriés doivent être différents).
   b. La taille des volumes doit être indiquée comme pourcentage du groupe pool/volumes de stockage et le total ne doit pas dépasser 100 pour tous les services/volumes utilisant un pool/groupe de volumes spécifique. Remarque : lors de l'utilisation de disques SSD, il est recommandé de laisser un peu d'espace libre dans le groupe de volumes afin d'optimiser les performances et la durée de vie des disques SSD (cliquez "ici" pour plus de détails).
   c. Cliquez sur "ici" pour obtenir la liste complète des options de configuration disponibles pour le `eseries_storage_pool_configuration`. Notez certaines options telles que `state`, `host`, `host_type`, `workload_name`, et `workload_metadata` des noms de volume et de volume sont générés automatiquement et ne doivent pas être spécifiés ici.

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

Cliquez sur "ici" Exemple de fichier d'inventaire complet représentant un service de métadonnées BeeGFS.

Définissez le service de stockage BeeGFS

Les services BeeGFS sont configurés à l'aide de variables de groupe (Group_var).

Présentation

Cette section décrit la définition du service de stockage BeeGFS. Au moins un service de ce type doit exister dans le(s) cluster(s) haute disponibilité pour un système de fichiers particulier. La configuration de ce service inclut la définition des éléments suivants :

- Le type de service (stockage).
- Définition de toute configuration qui ne doit s'appliquer qu'à ce service BeeGFS.
- Configuration d'une ou plusieurs adresses IP flottantes (interfaces logiques) sur lesquelles ce service peut être atteint.
- Spécifier où/comment le ou les volumes doivent être stockés pour ce service (cibles de stockage BeeGFS).

Étapes

En faisant référence "Planifiez le système de fichiers"à la section, créez un fichier à `group_vars/stor_<ID>.yml` pour chaque service de stockage du cluster et remplissez-le comme suit :

1. Indiquez ce fichier représente la configuration d'un service de stockage BeeGFS :

   ```
   beegfs_service: storage
   ```

2. Définissez toute configuration qui doit s'appliquer uniquement à ce service BeeGFS. Au minimum, vous devez spécifier le port TCP et UDP de votre choix, mais tout paramètre de configuration pris en charge à partir de `beegfs-storage.conf` peut également être inclus. Remarque les paramètres suivants sont configurés automatiquement/ailleurs et ne doivent pas être spécifiés ici : `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`,

`connInterfacesFile`, et `connNetFilterFile`.

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

3. Configurez une ou plusieurs adresses IP flottantes que les autres services et clients utiliseront pour se connecter à ce service (cela définit automatiquement le BeeGFS `connInterfacesFile` option) :

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

4. Vous pouvez également spécifier un ou plusieurs sous-réseaux IP autorisés qui peuvent être utilisés pour les communications sortantes (cela va automatiquement définir BeeGFS `connNetFilterFile` option) :

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

5. Spécifiez la ou les cibles de stockage BeeGFS où ce service stockera les données conformément aux directives suivantes (ceci configurera également automatiquement l' `storeStorageDirectory` option) :
   a. Le même nom de pool de stockage ou de groupe de volumes peut être utilisé pour plusieurs services/cibles BeeGFS, tout simplement s'assurer d'utiliser la même chose `name`, `raid_level`, `criteria_*`, et `common_*` configuration pour chaque service (les volumes répertoriés doivent être différents).
   b. La taille des volumes doit être indiquée comme pourcentage du groupe pool/volumes de stockage et le total ne doit pas dépasser 100 pour tous les services/volumes utilisant un pool/groupe de volumes spécifique. Remarque : lors de l'utilisation de disques SSD, il est recommandé de laisser un peu d'espace libre dans le groupe de volumes afin d'optimiser les performances et la durée de vie des disques SSD (cliquez "ici" pour plus de détails).
   c. Cliquez sur "ici" pour obtenir la liste complète des options de configuration disponibles pour le `eseries_storage_pool_configuration`. Notez certaines options telles que `state`, `host`, `host_type`, `workload_name`, et `workload_metadata` des noms de volume et de volume sont générés automatiquement et ne doivent pas être spécifiés ici.

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

Cliquez sur "ici" Exemple de fichier d'inventaire complet représentant un service de stockage BeeGFS.

Mapper les services BeeGFS sur les nœuds de fichiers

Spécifiez quels nœuds de fichiers peuvent exécuter chaque service BeeGFS à l'aide de l' `inventory.yml` fichier.

Présentation

Cette section explique comment créer le `inventory.yml` fichier. Cela inclut la liste de tous les nœuds de bloc et la spécification des nœuds de fichier pouvant exécuter chaque service BeeGFS.

Étapes

Créez le fichier `inventory.yml` et remplir comme suit :

1. Dans la partie supérieure du fichier, créez la structure d'inventaire Ansible standard :

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

2. Créer un groupe contenant tous les nœuds de bloc participant à ce cluster haute disponibilité :

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

3. Créez un groupe qui contiendra tous les services BeeGFS dans le cluster, ainsi que les nœuds de fichiers qui les exécuteront :

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

4. Pour chaque service BeeGFS du cluster, définissez le ou les nœuds de fichiers préférés et secondaires qui doivent exécuter ce service :

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

Cliquez sur "ici" exemple de fichier d'inventaire complet.

Déployez le système de fichiers BeeGFS

Présentation du PlayBook Ansible

Déploiement et gestion de clusters BeeGFS HA à l'aide d'Ansible.

Présentation

Les sections précédentes ont parcouru les étapes nécessaires à la création d'un inventaire Ansible représentant un cluster BeeGFS HA. Cette section présente l'automatisation Ansible développée par NetApp pour déployer et gérer le cluster.

Ansible : principaux concepts

Avant de commencer, il est utile de connaître quelques concepts clés Ansible :

- Les tâches à exécuter avec un inventaire Ansible sont définies dans ce qu'on appelle **PlayBook**.
  - La plupart des tâches dans Ansible sont conçues pour être **idempotent**, ce qui signifie qu'elles peuvent être exécutées plusieurs fois pour vérifier que la configuration/l'état désiré est toujours appliqué sans briser les choses ou faire des mises à jour inutiles.

- La plus petite unité d'exécution dans Ansible est un **module**.
  - Les playbooks classiques utilisent plusieurs modules.
    - Exemples : télécharger un package, mettre à jour un fichier de configuration, démarrer/activer un service.
  - NetApp distribue des modules pour automatiser les systèmes NetApp E-Series.
- Une automatisation complexe est mieux prédéfinie.
  - Il s'agit essentiellement d'un format standard permettant de distribuer un PlayBook réutilisable.
  - NetApp distribue des rôles pour les hôtes Linux et les systèmes de fichiers BeeGFS.

BeeGFS HA role pour Ansible : concepts clés

Tout l'automatisation nécessaire au déploiement et à la gestion de chaque version de BeeGFS sur NetApp est un rôle Ansible et distribué dans le "NetApp E-Series Ansible Collection pour BeeGFS":

- Ce rôle peut être considéré comme quelque part entre un **installateur** et un * moderne **déploiement/gestion** moteur pour BeeGFS.
  - Applique une infrastructure moderne en tant que pratiques du code et philosophie pour simplifier la gestion de l'infrastructure de stockage à toute échelle.
  - De la même façon que "Priez"le projet permet aux utilisateurs de déployer/gérer une distribution Kubernetes complète pour une infrastructure de calcul scale-out.
- Il s'agit du format **Software-defined** que NetApp utilise pour créer, distribuer et gérer les solutions BeeGFS sur NetApp.
  - S'efforcer de créer une expérience de type appareil sans avoir à distribuer une distribution Linux entière ou une image de grande taille.
  - Inclut des agents de ressources de cluster conformes à la norme OCF (Open Cluster Framework) de NetApp pour les cibles BeeGFS, les adresses IP et la surveillance personnalisés qui fournissent une intégration Pacemaker/BeeGFS intelligente.
- Le rôle ne se limite pas au déploiement de l'« automatisation ». Il est destiné à gérer l'ensemble du cycle de vie du système de fichiers, notamment :
  - Modification et mise à jour de la configuration au niveau du service ou du cluster
  - Automatisation de la réparation et de la restauration de clusters après une résolution des problèmes matériels
  - Simplification du réglage des performances avec des valeurs par défaut définies sur la base de tests approfondis réalisés avec BeeGFS et les volumes NetApp.
  - Vérification et correction de la dérive de configuration.

NetApp fournit également un rôle Ansible pour "Clients BeeGFS", Qui peut éventuellement être utilisé pour installer des systèmes de fichiers BeeGFS et monter des nœuds de calcul/GPU/connexion.

Déployez le cluster BeeGFS HA

Spécifiez les tâches à exécuter pour déployer le cluster BeeGFS HA à l'aide d'un PlayBook.

Présentation

Cette section explique comment assembler le manuel de vente standard utilisé pour déployer/gérer BeeGFS sur NetApp.

Étapes

Créez le manuel de vente Ansible

Créez le fichier `playbook.yml` et remplir comme suit :

1. Commencez par définir un ensemble de tâches (communément appelé « a ») "lecture") Qui ne doit s'exécuter que sur les nœuds de blocs NetApp E-Series. Nous utilisons une tâche de pause avant d'exécuter l'installation (pour éviter les exécutions accidentelles de PlayBook), puis importez la `nar_santricity_management` rôle. Ce rôle permet d'appliquer toute configuration système générale définie dans `group_vars/eseries_storage_systems.yml` ou individuellement `host_vars/<BLOCK NODE>.yml` fichiers.

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

2. Définissez la lecture qui s'exécutera sur tous les nœuds de fichiers et de blocs :

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

3. Dans ce module, nous pouvons définir, en option, un ensemble de « tâches préalables » à exécuter avant de déployer le cluster de haute disponibilité. Cela peut être utile pour vérifier/installer des prérequis comme Python. Nous pouvons également procéder à des vérifications avant vol, par exemple si les balises Ansible fournies sont prises en charge :

```
  pre_tasks:
```

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

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

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

        - name: Create a symbolic link to python from python3.
          raw: ln -s /usr/bin/python3 /usr/bin/python
          become: true
          when: python_version['rc'] != 0
      when: inventory_hostname not in
groups[beegfs_ha_ansible_storage_group]

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

```
      loop: "{{ ansible_run_tags }}"
```

4. Enfin, ce jeu importe le rôle BeeGFS HA pour la version de BeeGFS que vous voulez déployer:

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

> Un rôle BeeGFS HA est maintenu pour chaque version majeure.mineure de BeeGFS prise en charge. Cela permet aux utilisateurs de choisir quand ils souhaitent mettre à niveau des versions majeures/mineures. BeeGFS 7.3.x (`beegfs_7_3`) ou BeeGFS 7.2.x) (`beegfs_7_2`sont actuellement pris en charge. Par défaut, les deux rôles déploieront la dernière version de correctif BeeGFS au moment de la publication, bien que les utilisateurs puissent choisir de la remplacer et de déployer le dernier correctif si nécessaire. Consultez les dernières informations "guide de mise à niveau" pour plus de détails.

5. Facultatif : si vous souhaitez définir d'autres tâches, n'oubliez pas si les tâches doivent être dirigées vers `all` Hôtes (y compris les systèmes de stockage E-Series) ou uniquement les nœuds de fichiers. Si nécessaire, définissez une nouvelle lecture ciblant spécifiquement les nœuds de fichiers à l'aide de `– hosts: ha_cluster`.

Cliquez sur "ici" par exemple de fichier playbook complet.

Installez les collections NetApp Ansible

La collection BeeGFS pour Ansible et toutes les dépendances sont conservées "Galaxy Ansible". Sur votre nœud de contrôle Ansible, exécutez la commande suivante pour installer la dernière version :

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

Bien que cela ne soit pas généralement recommandé, il est également possible d'installer une version spécifique de la collection :

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

À l'aide du manuel de vente

À partir du répertoire de votre nœud de contrôle Ansible contenant le `inventory.yml` et `playbook.yml` exécutez le playbook comme suit :

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

Selon la taille du cluster, le déploiement initial peut prendre plus de 20 minutes. Si le déploiement échoue pour

une raison quelconque, corrigez simplement n'importe quel problème (par exemple, un défaut de câblage, un nœud n'a pas été démarré, etc.), puis redémarrez le PlayBook Ansible.

Lorsque "configuration de nœud de fichier commune"vous spécifiez , si vous choisissez l'option par défaut pour qu'Ansible gère automatiquement l'authentification basée sur la connexion, vous `connAuthFile` pouvez désormais trouver l' utilisé comme secret partagé à l'adresse `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (par défaut). Tous les clients devant accéder au système de fichiers devront utiliser ce secret partagé. Ce traitement est automatique si les clients sont configurés à l'aide de "Rôle client BeeGFS".

Déploiement de clients BeeGFS

Vous pouvez également utiliser Ansible pour configurer les clients BeeGFS et monter le système de fichiers.

Présentation

Pour accéder aux systèmes de fichiers BeeGFS, vous devez installer et configurer le client BeeGFS sur chaque nœud qui doit monter le système de fichiers. Cette section explique comment effectuer ces tâches à l'aide de la disponible "Rôle Ansible".

Étapes

Créez le fichier d'inventaire client

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>
```

3. Vous pouvez également inclure l'une des options suivantes si vous souhaitez utiliser les rôles de la Collection d'hôtes NetApp E-Series pour configurer les interfaces InfiniBand ou Ethernet pour les clients afin de se connecter à des nœuds de fichiers BeeGFS :
   a. Si le type de réseau est "InfiniBand (avec IPoib)":

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

b. Si le type de réseau est "RDMA over Converged Ethernet (RoCE)":

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

c. Si le type de réseau est "Ethernet (TCP uniquement, pas de RDMA)":

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

4. Créez un nouveau fichier `client_inventory.yml` Spécifiez l'utilisateur Ansible doit utiliser pour vous connecter à chaque client, et le mot de passe Ansible doit utiliser pour la réaffectation des privilèges (cette étape nécessite le mot de passe `ansible_ssh_user` être root ou avoir des privilèges sudo) :

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

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

5. Dans le `client_inventory.yml` Fichier, répertorie tous les hôtes qui doivent être configurés comme clients BeeGFS sous `beegfs_clients` Grouper, puis se reporter aux commentaires en ligne et supprimer toute configuration supplémentaire requise pour construire le module de noyau client BeeGFS sur votre système :

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

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

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

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

Lorsque vous utilisez les pilotes OFED de 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".

6. Dans le `client_inventory.yml` Fichier, répertorie les systèmes de fichiers BeeGFS que vous souhaitez monter sous n'importe quel fichier précédemment défini `vars`:

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

7. À partir de BeeGFS 7.2.7 et 7.3.1 "authentification de la connexion" doivent être configurés ou explicitement désactivés. Selon la façon dont vous choisissez de configurer l'authentification basée sur la connexion lors de "configuration de nœud de fichier commune"la spécification , vous devrez peut-être ajuster la configuration de votre client :

   a. Par défaut, le déploiement du cluster haute disponibilité configure automatiquement l'authentification de connexion et génère un `connauthfile` Qui seront placées/conservées sur le nœud de contrôle Ansible à `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. Par défaut, le rôle client BeeGFS est configuré pour lire/distribuer ce fichier aux clients définis dans `client_inventory.yml`, et aucune action supplémentaire n'est nécessaire.

      i. Pour les options avancées, reportez-vous à la liste complète des valeurs par défaut incluses avec le "Rôle client BeeGFS".

   b. Si vous choisissez de spécifier un secret personnalisé avec `beegfs_ha_conn_auth_secret` spécifiez-le dans le `client_inventory.yml` les fichiers ainsi :

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Si vous choisissez de désactiver entièrement l'authentification basée sur la connexion avec `beegfs_ha_conn_auth_enabled`, spécifiez cela dans le `client_inventory.yml` les fichiers ainsi :

```
beegfs_ha_conn_auth_enabled: false
```

Pour obtenir la liste complète des paramètres pris en charge et des détails supplémentaires, reportez-vous au "Documentation complète du client BeeGFS". Pour obtenir un exemple complet d'inventaire client, cliquez sur "ici".

Créez le fichier BeeGFS client PlayBook

1. Créez un nouveau fichier `client_playbook.yml`

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

2. Facultatif : si vous souhaitez utiliser les rôles de la collection d'hôtes NetApp E-Series pour configurer les interfaces pour les clients afin de vous connecter aux systèmes de fichiers BeeGFS, importez le rôle correspondant au type d'interface que vous configurez :
   a. Si vous utilisez InfiniBand (IPoIB) :

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

   b. Si vous utilisez le protocole RDMA over Converged Ethernet (RoCE) :

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

   c. Si vous utilisez Ethernet (TCP uniquement, pas de RDMA) :

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

3. Enfin, importez le rôle client BeeGFS pour installer le logiciel client et configurer les montages du système de fichiers :

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

Pour obtenir un exemple de PlayBook client complet, cliquez sur "ici".

Exécutez le manuel de vente BeeGFS client

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

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

Vérifier le déploiement BeeGFS

Vérifiez le déploiement du système de fichiers avant de mettre le système en production.

Présentation

Avant de placer le système de fichiers BeeGFS en production, effectuez quelques vérifications.

Étapes

1. Connectez-vous à n'importe quel client et exécutez les opérations suivantes pour vous assurer que tous les nœuds attendus sont présents/accessibles, et qu'il n'y a pas d'incohérences ou d'autres problèmes signalés :

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

2. Arrêtez l'ensemble du cluster, puis redémarrez-le. Depuis n'importe quel nœud de fichiers, exécutez ce qui suit :

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

3. Placez chaque nœud en veille et vérifiez que les services BeeGFS peuvent basculer vers un ou plusieurs nœuds secondaires. Pour ce faire, connectez-vous à l'un des nœuds de fichiers et exécutez les opérations suivantes :

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

4. Utilisez des outils d'évaluation des performances tels que IOR et MDTest pour vérifier que les performances du système de fichiers répondent aux attentes. La "Vérification de la conception"section BeeGFS sur l'architecture vérifiée NetApp fournit des exemples de tests et de paramètres courants.

Des tests supplémentaires doivent être effectués en fonction des critères d'acceptation définis pour un site/une installation spécifique.

Déploiement des fonctionnalités et des intégrations

Pilote BeeGFS CSI

Configurer le chiffrement TLS pour BeeGFS v8

Configurez le chiffrement TLS pour sécuriser la communication entre les services de gestion BeeGFS v8 et les clients.

Présentation

BeeGFS v8 introduit la prise en charge de TLS pour le chiffrement des communications réseau entre les outils d'administration (tels que l' `beegfs`utilitaire en ligne de commande) et les services serveur BeeGFS comme Management ou Remote. Ce guide explique comment configurer le chiffrement TLS dans votre cluster BeeGFS à l'aide de trois méthodes de configuration TLS :

- **Utilisation d'une autorité de certification de confiance :** Utilisez des certificats signés par une autorité de certification existante sur votre cluster BeeGFS.
- **Création d'une autorité de certification locale :** Création d'une autorité de certification locale et utilisation de celle-ci pour signer les certificats de vos services BeeGFS. Cette approche convient aux environnements où vous souhaitez gérer votre propre chaîne de confiance sans dépendre d'une autorité de certification externe.
- **TLS désactivé :** Désactivez complètement TLS dans les environnements où le chiffrement n'est pas requis ou pour le dépannage. Cette pratique est déconseillée car elle expose en clair des informations potentiellement sensibles concernant la structure du système de fichiers interne et le fichier de configuration.

Choisissez la méthode la mieux adaptée à votre environnement et à vos politiques organisationnelles. Consultez la "BeeGFS TLS" documentation pour plus de détails.

> Les machines exécutant le `beegfs-client` service n'ont pas besoin de TLS pour monter le système de fichiers BeeGFS. TLS doit être configuré pour utiliser la CLI BeeGFS et d'autres services BeeGFS, tels que remote et sync.

Utilisation d'une autorité de certification de confiance

Si vous avez accès à des certificats émis par une autorité de certification (CA) de confiance—qu'il s'agisse d'une CA d'entreprise interne ou d'un fournisseur tiers—vous pouvez configurer BeeGFS v8 pour utiliser ces certificats signés par la CA au lieu de générer des certificats auto-signés.

Déploiement d'un nouveau cluster BeeGFS v8

Pour un nouveau déploiement de cluster BeeGFS v8, configurez le fichier d'inventaire Ansible `user_defined_params.yml` pour qu'il référence vos certificats signés par une autorité de certification :

```
beegfs_ha_tls_enabled: true

beegfs_ha_ca_cert_src_path: files/beegfs/cert/ca_cert.pem

beegfs_ha_tls_cert_src_path: files/beegfs/cert/mgmtd_tls_cert.pem

beegfs_ha_tls_key_src_path: files/beegfs/cert/mgmtd_tls_key.pem
```

> Si `beegfs_ha_tls_config_options.alt_names` n'est pas vide, Ansible générera automatiquement un certificat TLS auto-signé et une clé, en utilisant les alt_names fournis comme Subject Alternative Names (SANs) dans le certificat. Pour utiliser votre propre certificat et clé TLS personnalisés (tels que spécifiés par `beegfs_ha_tls_cert_src_path` et `beegfs_ha_tls_key_src_path`), vous devez commenter ou supprimer l'intégralité de la section `beegfs_ha_tls_config_options`. Sinon, la génération du certificat auto-signé prendra le dessus et votre certificat et clé personnalisés ne seront pas utilisés.

Configuration d'un cluster BeeGFS v8 existant

Pour un cluster BeeGFS v8 existant, définissez les chemins dans le fichier de configuration des services de gestion BeeGFS sur les certificats signés par l'autorité de certification du nœud de fichiers :

```
tls-cert-file = /path/to/cert.pem
tls-key-file = /path/to/key.pem
```

Configuration des clients BeeGFS v8 avec des certificats signés par une autorité de certification

Pour configurer les clients BeeGFS v8 afin qu'ils fassent confiance aux certificats signés par une autorité de certification à l'aide du pool de certificats système, définissez tls-cert-file = '' dans le fichier de configuration de chaque client. Si le pool de certificats système n'est pas utilisé, indiquez le chemin d'accès à un certificat local en définissant tls-cert-file = <local cert>. Cette configuration permet aux clients d'authentifier les certificats présentés par les services de gestion BeeGFS.

Création d'une autorité de certification locale

Si votre organisation souhaite créer sa propre infrastructure de certificats pour le cluster BeeGFS, vous pouvez créer une autorité de certification (CA) locale pour émettre et signer les certificats de votre cluster BeeGFS. Cette approche consiste à créer une CA qui signe les certificats des services de gestion BeeGFS, lesquels sont ensuite distribués aux clients afin d'établir une chaîne de confiance. Suivez ces instructions pour configurer une CA locale et déployer les certificats sur votre cluster BeeGFS v8 existant ou nouveau.

Déploiement d'un nouveau cluster BeeGFS v8

Pour un nouveau déploiement de BeeGFS v8, le `beegfs_8` rôle Ansible se chargera de créer une autorité de certification locale sur le nœud de contrôle et de générer les certificats nécessaires pour les services de gestion. Cela peut être activé en définissant les paramètres suivants dans le fichier `user_defined_params.yml` d'inventaire Ansible :

```
beegfs_ha_tls_enabled: true

beegfs_ha_ca_cert_src_path: files/beegfs/cert/local_ca_cert.pem

beegfs_ha_tls_cert_src_path: files/beegfs/cert/mgmtd_tls_cert.pem

beegfs_ha_tls_key_src_path: files/beegfs/cert/mgmtd_tls_key.pem

beegfs_ha_tls_config_options:
  alt_names: [<mgmt_service_ip>]
```

Si `beegfs_ha_tls_config_options.alt_names` n'est pas fourni, alors Ansible tentera d'utiliser les certificats existants dans les chemins de certificat/clé spécifiés.

Configuration d'un cluster BeeGFS v8 existant

Pour un cluster BeeGFS existant, vous pouvez intégrer TLS en créant une autorité de certification locale et en générant les certificats nécessaires pour les services de gestion. Mettez à jour les chemins dans le fichier de configuration des services de gestion BeeGFS pour qu'ils pointent vers les certificats nouvellement créés.

Les instructions de cette section sont données à titre indicatif. Il convient de prendre les précautions de sécurité appropriées lors de la manipulation des clés privées et des certificats.

Créer l'autorité de certification

Sur une machine de confiance, créez une autorité de certification locale pour signer les certificats de vos services de gestion BeeGFS. Le certificat de l'autorité de certification sera distribué aux clients pour établir la confiance et permettre une communication sécurisée avec les services BeeGFS.

Les instructions suivantes servent de référence pour la création d'une autorité de certification locale sur un système basé sur RHEL.

1. Installez OpenSSL s'il n'est pas déjà installé :

```
dnf install openssl
```

2. Créez un répertoire de travail pour stocker les fichiers de certificat :

```
mkdir -p ~/beegfs_tls && cd ~/beegfs_tls
```

3. Générez la clé privée de la CA :

```
openssl genrsa -out ca_key.pem 4096
```

4. Créez un fichier de configuration CA nommé `ca.cnf` et ajustez les champs de nom unique pour qu'ils

correspondent à votre organisation :

```
[ req ]
default_bits        = 4096
distinguished_name = req_distinguished_name
x509_extensions     = v3_ca
prompt              = no

[ req_distinguished_name ]
C  = <Country>
ST = <State>
L  = <City>
O  = <Organization>
OU = <OrganizationalUnit>
CN = BeeGFS-CA

[ v3_ca ]
basicConstraints = critical,CA:TRUE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
```

5. Générez le certificat d'autorité de certification. Ce certificat doit être valide pendant toute la durée de vie du système, sinon vous devrez prévoir de régénérer les certificats avant leur expiration. Une fois un certificat expiré, la communication entre certains composants ne sera plus possible et la mise à jour des certificats TLS nécessitera généralement de redémarrer les services pour terminer.

   La commande suivante génère un certificat d'autorité de certification valable 1 an :

   ```
   openssl req -new -x509 -key ca_key.pem -out ca_cert.pem -days 365
   -config ca.cnf
   ```

   Bien que cet exemple utilise une période de validité d'un an par souci de simplicité, vous devez ajuster le `-days` paramètre en fonction des exigences de sécurité de votre organisation et mettre en place un processus de renouvellement de certificat.

Créer des certificats de service de gestion

Générez des certificats pour vos services de gestion BeeGFS et signez-les avec la CA que vous avez créée. Ces certificats seront installés sur les nœuds de fichiers exécutant les services de gestion BeeGFS.

1. Générez la clé privée du service de gestion :

   ```
   openssl genrsa -out mgmtd_tls_key.pem 4096
   ```

2. Créez un fichier de configuration de certificat nommé `tls_san.cnf` avec des Subject Alternative Names

(SAN) pour toutes les adresses IP du service de gestion :

```
[ req ]
default_bits       = 4096
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt             = no

[ req_distinguished_name ]
C  = <Country>
ST = <State>
L  = <City>
O  = <Organization>
OU = <OrganizationalUnit>
CN = beegfs-mgmt

[ req_ext ]
subjectAltName = @alt_names

[ v3_ca ]
subjectAltName = @alt_names
basicConstraints = CA:FALSE

[ alt_names ]
IP.1 = <beegfs_mgmt_service_ip_1>
IP.2 = <beegfs_mgmt_service_ip_2>
```

Mettez à jour les champs de nom unique pour qu'ils correspondent à la configuration de votre CA et les `IP.1` et `IP.2` valeurs avec les adresses IP de votre service de gestion.

3. Générez une demande de signature de certificat (CSR) :

```
openssl req -new -key mgmtd_tls_key.pem -out mgmtd_tls_csr.pem -config
tls_san.cnf
```

4. Signez le certificat avec votre CA (valable 1 an) :

```
openssl x509 -req -in mgmtd_tls_csr.pem -CA ca_cert.pem -CAkey
ca_key.pem -CAcreateserial -out mgmtd_tls_cert.pem -days 365 -sha256
-extensions v3_ca -extfile tls_san.cnf
```

Ajustez la période de validité du certificat (`-days 365` en fonction des politiques de sécurité de votre organisation. De nombreuses organisations exigent un renouvellement des certificats tous les 1-2 ans.

5. Vérifiez que le certificat a été créé correctement :

```
openssl x509 -in mgmtd_tls_cert.pem -text -noout
```

Confirmez que la section Nom alternatif du sujet inclut toutes vos adresses IP de gestion.

Distribuer les certificats aux nœuds de fichiers

Distribuez le certificat d'autorité de certification et les certificats de service de gestion aux nœuds de fichiers et aux clients appropriés.

1. Copiez le certificat d'autorité de certification (CA), le certificat du service de gestion et la clé de ce service sur les nœuds de fichiers exécutant les services de gestion :

```
scp ca_cert.pem mgmtd_tls_cert.pem mgmtd_tls_key.pem
user@beegfs_01:/etc/beegfs/
scp ca_cert.pem mgmtd_tls_cert.pem mgmtd_tls_key.pem
user@beegfs_02:/etc/beegfs/
```

Pointez le service de gestion vers les certificats TLS

Mettez à jour la configuration du service de gestion BeeGFS pour activer TLS et référencer les certificats TLS créés.

1. Depuis un nœud de fichier exécutant le service de gestion BeeGFS, modifiez le fichier de configuration du service de gestion, par exemple à `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml`. Ajoutez ou mettez à jour les paramètres liés à TLS suivants :

```
tls-disable = false
tls-cert-file = "/etc/beegfs/mgmtd_tls_cert.pem"
tls-key-file = "/etc/beegfs/mgmtd_tls_key.pem"
```

2. Prenez les mesures appropriées pour redémarrer en toute sécurité le service de gestion BeeGFS afin que les modifications prennent effet :

```
systemctl restart beegfs-mgmtd
```

3. Vérifiez que le service de gestion a démarré avec succès :

```
journalctl -xeu beegfs-mgmtd
```

Recherchez les entrées de journal indiquant une initialisation TLS réussie et un chargement du certificat.

```
Successfully initialized certificate verification library.
Successfully loaded license certificate: TMP-XXXXXXXXXX
```

Configurer TLS pour les clients BeeGFS v8

Créer et distribuer des certificats signés par la CA locale à tous les clients BeeGFS qui nécessiteront une communication avec les services de gestion BeeGFS.

1. Générez un certificat pour le client en utilisant le même processus que pour le certificat de service de gestion ci-dessus, mais avec l'adresse IP ou le nom d'hôte du client dans le champ Subject Alternative Name (SAN).
2. Copiez à distance et en toute sécurité le certificat du client sur le client et renommez le certificat en `cert.pem` sur le client :

```
scp client_cert.pem user@client:/etc/beegfs/cert.pem
```

3. Redémarrez le service client BeeGFS sur tous les clients :

```
systemctl restart beegfs-client
```

4. Vérifiez la connectivité du client en exécutant une commande `beegfs CLI`, telle que :

```
beegfs health check
```

Désactivation de TLS

TLS peut être désactivé à des fins de dépannage ou si les utilisateurs le souhaitent. Cela est déconseillé car cela expose en clair des informations potentiellement sensibles concernant la structure interne du système de fichiers et le fichier de configuration. Suivez ces instructions pour désactiver TLS sur votre cluster BeeGFS v8 existant ou nouveau.

Déploiement d'un nouveau cluster BeeGFS v8

Pour un nouveau déploiement de cluster BeeGFS, le cluster peut être déployé avec TLS désactivé en définissant le paramètre suivant dans le fichier `user_defined_params.yml` d'inventaire Ansible :

```
beegfs_ha_tls_enabled: false
```

Configuration d'un cluster BeeGFS v8 existant

Pour un cluster BeeGFS v8 existant, modifiez le fichier de configuration du service de gestion. Par exemple, modifiez le fichier à `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml` et définissez :

```
tls-disable = true
```

Prenez les mesures appropriées pour redémarrer en toute sécurité le service de gestion afin que les modifications prennent effet.

Gérer des clusters BeeGFS

Présentation, concepts clés et terminologie

Apprenez à gérer des clusters BeeGFS HA après leur déploiement.

Présentation

Cette section s'adresse aux administrateurs de cluster qui doivent gérer des clusters BeeGFS HA après leur déploiement. Même ceux qui connaissent les clusters haute disponibilité Linux doivent lire attentivement ce guide car il existe un certain nombre de différences dans la gestion du cluster, en particulier concernant la reconfiguration, grâce à l'utilisation d'Ansible.

Concepts clés

Certains de ces concepts sont présentés sur la "termes et concepts"page principale, mais il est utile de les réintroduire dans le contexte d'un cluster BeeGFS HA :

Cluster Node: Un serveur exécutant les services Pacemaker et Corosync et participant au cluster HA.

Nœud de fichiers : Nœud de cluster utilisé pour exécuter un ou plusieurs services de gestion, de métadonnées ou de stockage BeeGFS.

Nœud de bloc : Un système de stockage NetApp E-Series qui fournit un stockage bloc aux nœuds de fichiers. Ces nœuds ne participent pas au cluster BeeGFS HA car ils fournissent leurs propres capacités HA autonomes. Chaque nœud est constitué de deux contrôleurs de stockage qui assurent une haute disponibilité au niveau de la couche bloc.

Service BeeGFS: Un service de gestion, de métadonnées ou de stockage BeeGFS. Chaque nœud de fichiers exécute un ou plusieurs services qui utilisent les volumes du nœud de bloc pour stocker leurs données.

Building Block : Un déploiement standardisé de nœuds de fichiers BeeGFS, de nœuds de blocs E-Series et des services BeeGFS s'exécutent sur eux qui simplifient l'évolution d'un cluster/système de fichiers BeeGFS HA grâce à une architecture vérifiée NetApp. Les clusters haute disponibilité personnalisés sont également pris en charge, mais leur approche consiste souvent à adopter des éléments de base similaires pour simplifier l'évolutivité.

BeeGFS HA Cluster: Nombre évolutif de nœuds de fichiers utilisés pour exécuter les services BeeGFS sauvegardés par des nœuds de blocs pour stocker des données BeeGFS de façon hautement disponible. Repose sur des composants open source éprouvés Pacemaker et Corosync avec Ansible pour le packaging et le déploiement.

Cluster services: désigne les services Pacemaker et Corosync exécutés sur chaque nœud participant au cluster. Notez qu'un nœud n'exécute pas de services BeeGFS et qu'il participe uniquement au cluster comme un nœud « Tiebreaker » s'il n'y a que besoin de deux nœuds de fichiers.

Cluster Resources: pour chaque service BeeGFS s'exécutant dans le cluster, vous verrez une ressource de moniteur BeeGFS et un groupe de ressources contenant des ressources pour les cibles BeeGFS, les adresses IP (IP flottantes) et le service BeeGFS.

Ansible: Un outil de provisionnement logiciel, de gestion de la configuration et de déploiement des applications, permettant ainsi une infrastructure comme du code. Tout cela est possible grâce au package de clusters BeeGFS HA pour simplifier le processus de déploiement, de reconfiguration et de mise à jour de

BeeGFS sur NetApp.

Pcs: Une interface de ligne de commande disponible à partir de n'importe quel nœud de fichiers du cluster utilisé pour interroger et contrôler l'état des nœuds et des ressources du cluster.

Terminologie commune

Basculement: chaque service BeeGFS a un noeud de fichier préféré qu'il fonctionne à moins que ce noeud ne tombe en panne. Lorsqu'un service BeeGFS s'exécute sur un nœud de fichier non préféré/secondaire, il est dit qu'il est en cours de basculement.

Retour arrière: le fait de déplacer les services BeeGFS d'un noeud de fichier non préféré vers leur noeud préféré.

Paire HA : deux nœuds de fichiers qui accèdent au même ensemble de nœuds de bloc sont parfois appelés paire HA. Ce terme est utilisé dans l'ensemble de NetApp pour désigner deux contrôleurs ou nœuds de stockage qui peuvent « prendre le relais » les uns les autres.

Mode Maintenance : désactive la surveillance de toutes les ressources et empêche Pacemaker de déplacer ou de gérer les ressources dans le cluster (voir également la section "mode maintenance").

Cluster HA: un ou plusieurs nœuds de fichiers exécutant les services BeeGFS qui peuvent basculer entre plusieurs nœuds du cluster pour créer un système de fichiers BeeGFS haute disponibilité. Ils sont souvent configurés en paires HA et qui peuvent exécuter un sous-ensemble des services BeeGFS dans le cluster.

Quand utiliser Ansible contre l'outil pcs

Quand devriez-vous utiliser Ansible par rapport à l'outil de ligne de commande pcs pour gérer le cluster HA ?

Toutes les tâches de déploiement et de reconfiguration du cluster doivent être effectuées à l'aide d'Ansible à partir d'un nœud de contrôle Ansible externe. Les modifications temporaires de l'état du cluster (par exemple, placement des nœuds en veille ou en dehors) sont généralement effectuées en se connectant à un nœud du cluster (de préférence un nœud qui n'est pas dégradé ou sur le point de subir des opérations de maintenance) et en utilisant l'outil de ligne de commande pcs.

Tout changement de configuration de cluster, y compris les ressources, les contraintes, les propriétés et les services BeeGFS, doit toujours être effectué à l'aide d'Ansible. Maintenir une copie à jour de l'inventaire et du manuel de vente Ansible (idéalement en contrôle source pour suivre les modifications) fait partie de la maintenance du cluster. Si vous devez modifier la configuration, mettez à jour l'inventaire et exécutez à nouveau le PlayBook Ansible qui importe le rôle BeeGFS HA.

Le rôle HA gère le placement du cluster en mode maintenance, puis les modifications nécessaires avant de redémarrer BeeGFS ou les services du cluster pour appliquer la nouvelle configuration. Le redémarrage complet de nœud n'est généralement pas nécessaire en dehors du déploiement initial. Toutefois, le redémarrage d'Ansible est généralement considéré comme une procédure « sûre », mais toujours recommandé pendant les fenêtres de maintenance ou hors heures si les services BeeGFS doivent redémarrer. Ces redémarrages ne doivent généralement pas provoquer d'erreurs d'application, mais peuvent nuire aux performances (que certaines applications peuvent traiter mieux que d'autres).

Le réexécution Ansible est également une option pour rétablir l'état optimal de l'ensemble du cluster. Dans certains cas, il peut récupérer l'état du cluster plus facilement qu'avec les pièces. Notamment en cas d'urgence où le cluster est hors service, une fois que tous les nœuds sont sauvegardées, Ansible peut récupérer le cluster plus rapidement et de façon plus fiable que toute tentative d'utilisation de pcs.

Vérifiez l'état du cluster

Utilisez les pièces pour voir l'état du bloc d'instruments.

Présentation

Exécution `pcs status` À partir de n'importe quel nœud de cluster est le moyen le plus simple de voir l'état global du cluster et l'état de chaque ressource (par exemple, les services BeeGFS et leurs dépendances). Cette section présente ce que vous trouverez dans les résultats du `pcs status` commande.

Présentation de la sortie de `pcs status`

Courez `pcs status` Sur n'importe quel nœud de cluster où les services de cluster (Pacemaker et Corosync) sont démarrés. Le haut de la sortie affiche un récapitulatif du cluster :

```
[root@beegfs_01 ~]# pcs status
Cluster name: hacluster
Cluster Summary:
  * Stack: corosync
  * Current DC: beegfs_01 (version 2.0.5-9.el8_4.3-ba59be7122) - partition
with quorum
  * Last updated: Fri Jul  1 13:37:18 2022
  * Last change:  Fri Jul  1 13:23:34 2022 by root via cibadmin on
beegfs_01
  * 6 nodes configured
  * 235 resource instances configured
```

La section ci-dessous liste les nœuds du cluster :

```
Node List:
  * Node beegfs_06: standby
  * Online: [ beegfs_01 beegfs_02 beegfs_04 beegfs_05 ]
  * OFFLINE: [ beegfs_03 ]
```

Cela indique notamment tous les nœuds en veille ou hors ligne. Les nœuds en veille font toujours partie du cluster, mais sont marqués comme non éligibles pour l'exécution des ressources. Les nœuds hors ligne indiquent que les services du cluster ne s'exécutent pas sur ce nœud, soit en raison d'un arrêt manuel, soit en raison du redémarrage ou de l'arrêt du nœud.

> Lorsque les nœuds démarrent initialement, les services de cluster sont arrêtés et doivent être démarrés manuellement pour éviter de basculer accidentellement des ressources sur un nœud défaillant.

Si les nœuds sont en attente ou hors ligne en raison d'une raison non administrative (par exemple, une panne), un texte supplémentaire s'affiche à côté de l'état du nœud entre parenthèses. Par exemple, si l'escrime est désactivé et qu'une ressource rencontre une défaillance, vous verrez `Node <HOSTNAME>:`

`standby (on-fail)`. Un autre état possible est `Node <HOSTNAME>: UNCLEAN (offline)`, qui sera brièvement vu comme un nœud est clôturé, mais persistera si l'escrime a échoué indiquant que le cluster ne peut pas confirmer l'état du nœud (cela peut bloquer les ressources de démarrer sur d'autres noeuds).

La section suivante affiche la liste de toutes les ressources du cluster et leur état :

```
Full List of Resources:
  * mgmt-monitor    (ocf::eseries:beegfs-monitor):   Started beegfs_01
  * Resource Group: mgmt-group:
    * mgmt-FS1  (ocf::eseries:beegfs-target):    Started beegfs_01
    * mgmt-IP1  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-IP2  (ocf::eseries:beegfs-ipaddr2):   Started beegfs_01
    * mgmt-service  (systemd:beegfs-mgmtd):  Started beegfs_01
[...]
```

Tout comme les nœuds, un texte supplémentaire s'affiche en regard de l'état de la ressource entre parenthèses s'il y a des problèmes avec la ressource. Par exemple, si Pacemaker demande un arrêt de ressource et qu'il ne s'effectue pas dans le temps alloué, Pacemaker tente de verrouiller le nœud. Si l'escrime est désactivé ou que l'opération d'escrime échoue, l'état de la ressource sera `FAILED <HOSTNAME> (blocked)` Et Pacemaker ne pourra pas démarrer sur un autre nœud.

Il est utile de noter que les clusters BeeGFS HA utilisent un certain nombre d'agents de ressources personnalisées de BeeGFS optimisés pour les OCF. En particulier, le moniteur BeeGFS est responsable du déclenchement d'un basculement lorsque les ressources BeeGFS sur un nœud donné ne sont pas disponibles.

Reconfigurer le cluster HA et BeeGFS

Utilisez Ansible pour reconfigurer le cluster.

Présentation

En général, la reconfiguration d'un aspect du cluster BeeGFS haute disponibilité doit être effectuée en mettant à jour votre inventaire Ansible et en réexécutant `ansible-playbook` la commande. Cela inclut la mise à jour des alertes, la modification de la configuration de l'escrime permanent ou l'ajustement de la configuration du service BeeGFS. Ils sont ajustés à l'aide du `group_vars/ha_cluster.yml` fichier et une liste complète des options se trouve dans la "Spécifiez la configuration de nœud de fichier commun" section.

Pour plus d'informations sur les options de configuration que les administrateurs doivent connaître lors des opérations de maintenance ou de maintenance du cluster, consultez ci-dessous.

Comment désactiver et activer la fonction de fencing

Par défaut, l'escrime est activé/requis lors de la configuration du cluster. Dans certains cas, il peut être souhaitable de désactiver temporairement l'escrime pour s'assurer que les nœuds ne s'arrêtent pas accidentellement lors de certaines opérations de maintenance (par exemple, la mise à niveau du système d'exploitation). Bien que cette fonction puisse être désactivée manuellement, les administrateurs doivent en être conscients des compromis.

OPTION 1 : désactivez l'escrime avec Ansible (recommandé).

Lorsque l'escrime est désactivé à l'aide d'Ansible, l'action en cas d'échec du moniteur BeeGFS passe de « clôture » à « veille ». Cela signifie que si le moniteur BeeGFS détecte une défaillance, il tente de placer le nœud en veille et de basculer tous les services BeeGFS. En dehors du dépannage/test actif, ceci est généralement plus souhaitable que l'option 2. L'inconvénient est que si une ressource ne s'arrête pas sur le nœud d'origine, elle sera bloquée pour commencer ailleurs (c'est pourquoi une clôture est généralement nécessaire pour les grappes de production).

1. Dans votre inventaire Ansible à `groups_vars/ha_cluster.yml` ajoutez la configuration suivante :

```
beegfs_ha_cluster_crm_config_options:
    stonith-enabled: False
```

2. Exécutez à nouveau le manuel de vente Ansible afin d'appliquer les modifications apportées au cluster.

OPTION 2 : désactivez manuellement l'escrime.

Dans certains cas, vous pouvez désactiver temporairement l'escrime sans qu'il soit nécessaire de réexécuter Ansible, afin de faciliter le dépannage ou le test du cluster.

> Dans cette configuration, si le moniteur BeeGFS détecte une défaillance, le cluster tente d'arrêter le groupe de ressources correspondant. Il NE déclenchera PAS un basculement complet ni ne tentera de redémarrer ou de déplacer le groupe de ressources affecté vers un autre hôte. Pour restaurer le système, traitez les problèmes avant de l'exécuter `pcs resource cleanup` ou placez manuellement le nœud en veille.

Étapes :

1. Pour déterminer si l'escrime (stonith) est globalement activé ou désactivé : `pcs property show stonith-enabled`
2. Pour désactiver la séquence d'escrime : `pcs property set stonith-enabled=false`
3. Pour activer la séquence d'escrime : `pcs property set stonith-enabled=true`

> Ce paramètre sera remplacé la prochaine fois que vous exécuterez le playbook Ansible.

Mettez à jour les composants du cluster HA

Mise à niveau des services BeeGFS

Utilisez Ansible pour mettre à jour la version de BeeGFS exécutée sur votre cluster HA.

Présentation

BeeGFS applique un `major.minor.patch` schéma de gestion des versions. Des rôles Ansible haute disponibilité BeeGFS sont fournis pour chaque `major.minor` version prise en charge (par exemple, `beegfs_ha_7_2` et `beegfs_ha_7_3`). Chaque rôle HA est épinglé à la dernière version de correctif BeeGFS disponible au moment de la publication de la collection Ansible.

Ansible doit être utilisé pour toutes les mises à niveau de BeeGFS, y compris le passage entre les versions majeure, mineure et corrective de BeeGFS. Pour mettre à jour BeeGFS, vous devrez d'abord mettre à jour la collection Ansible BeeGFS, ce qui intégrera également les derniers correctifs et améliorations de l'automatisation du déploiement/gestion et du cluster HA sous-jacent. Même après la mise à jour vers la dernière version de la collection, BeeGFS ne sera pas mis à niveau tant que `ansible-playbook` n'aura pas été exécuté avec l' `-e "beegfs_ha_force_upgrade=true"` activé. Pour plus de détails sur chaque mise à niveau, consultez la "Documentation de mise à niveau BeeGFS" pour votre version actuelle.

Si vous effectuez une mise à niveau vers BeeGFS v8, consultez plutôt la "Mise à jour vers BeeGFS v8"procédure.

Chemins de mise à niveau testés

Les voies de mise à niveau suivantes ont été testées et vérifiées :

Version d'origine	Mettre à niveau la version	Multirail	Détails
7.2.6	7.3.2	Oui.	Mise à niveau de la collection beegfs de v3.0.1 à v3.1.0, multirail ajouté
7.2.6	7.2.8	Non	Mise à niveau de la collection beegfs de v3.0.1 à v3.1.0
7.2.8	7.3.1	Oui.	Mise à niveau avec beegfs collection v3.1.0, multirail ajouté
7.3.1	7.3.2	Oui.	Mise à niveau avec beegfs collection v3.1.0
7.3.2	7.4.1	Oui.	Mise à niveau avec beegfs collection v3.2.0
7.4.1	7.4.2	Oui.	Mise à niveau avec beegfs collection v3.2.0
7.4.2	7.4.6	Oui.	Mise à niveau avec beegfs collection v3.2.0
7.4.6	8,0	Oui.	Mettez à niveau en suivant les instructions dans la "Mise à jour vers BeeGFS v8" procédure.
7.4.6	8,1	Oui.	Mettez à niveau en suivant les instructions dans la "Mise à jour vers BeeGFS v8" procédure.
7.4.6	8,2	Oui.	Mettez à niveau en suivant les instructions dans la "Mise à jour vers BeeGFS v8" procédure.

Étapes de mise à niveau BeeGFS

Les sections suivantes expliquent comment mettre à jour la collection BeeGFS Ansible et BeeGFS. Portez une attention particulière à toute étape(s) supplémentaire(s) pour la mise à jour de BeeGFS version majeure ou mineure.

Étape 1 : mise à niveau de la collection BeeGFS

Pour les mises à niveau de collecte avec accès à "Galaxy Ansible", exécutez la commande suivante :

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

Pour les mises à niveau hors ligne de la collection, téléchargez la collection à partir de "Galaxy Ansible" en

cliquant sur le bouton souhaité `Install Version` puis `Download tarball`. Transférez le tarball sur votre nœud de contrôle Ansible, puis exécutez la commande suivante.

```
ansible-galaxy collection install netapp_eseries-beegfs-<VERSION>.tar.gz
--upgrade
```

Voir "Installation de Collections" pour en savoir plus.

Étape 2 : mise à jour de l'inventaire Ansible

Apportez toutes les mises à jour requises ou souhaitées aux fichiers d'inventaire Ansible de votre cluster. Voir la section Notes de mise à niveau de la version ci-dessous pour plus de détails sur vos exigences spécifiques de mise à niveau. Voir la section "Présentation d'Ansible Inventory" pour des informations générales sur la configuration de votre inventaire BeeGFS HA.

Étape 3 : mise à jour du PlayBook Ansible (uniquement en cas de mise à jour des versions principales ou secondaires)

Si vous passez d'une version majeure à une version mineure, dans le `playbook.yml` fichier utilisé pour déployer et gérer le cluster, mettez à jour le nom du `beegfs_ha_<VERSION>` rôle pour refléter la version souhaitée. Par exemple, si vous souhaitez déployer BeeGFS 7.4 `beegfs_ha_7_4`:

```
- hosts: all
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.beegfs
  tasks:
    - name: Ensure BeeGFS HA cluster is setup.
      ansible.builtin.import_role:  # import_role is required for tag
availability.
        name: beegfs_ha_7_4
```

Pour plus de détails sur le contenu de ce fichier PlayBook"Déployez le cluster BeeGFS HA", reportez-vous à la section.

Étape 4 : exécutez la mise à niveau BeeGFS

Pour appliquer la mise à jour BeeGFS :

```
ansible-playbook -i inventory.yml beegfs_ha_playbook.yml -e
"beegfs_ha_force_upgrade=true" --tags beegfs_ha
```

En coulisse, le rôle haute disponibilité BeeGFS gère :

- Assurez-vous que le cluster est dans un état optimal avec chaque service BeeGFS situé sur son nœud préféré.
- Mettre le cluster en mode maintenance.

- Mettre à jour les composants du cluster haute disponibilité (le cas échéant)
- Mettez à niveau chaque nœud de fichiers un par un en procédant comme suit :
  - Mettez le système en veille et basculez ses services vers le nœud secondaire.
  - Mise à jour des packs BeeGFS.
  - Proposer de nouveaux services.
- Déplacez le cluster hors du mode maintenance.

Notes de mise à niveau de la version

Mise à jour de BeeGFS version 7.2.6 ou 7.3.0

Modifications de l'authentification basée sur la connexion

BeeGFS version 7.3.2 et ultérieures nécessitent que l'authentification basée sur la connexion soit configurée. Les services ne démarreront pas sans l'une des options suivantes :

- Spécifier un `connAuthFile`, ou
- Paramétrage `connDisableAuthentication=true` dans le fichier de configuration du service.

Il est fortement recommandé d'activer l'authentification basée sur la connexion pour des raisons de sécurité. Voir "Authentification basée sur la connexion BeeGFS" pour plus d'informations.

Les `beegfs_ha*` rôles génèrent et distribuent automatiquement le fichier d'authentification à :

- Tous les nœuds de fichiers du cluster
- Le nœud de contrôle Ansible à `<playbook_directory>/files/beegfs/<beegfs_mgmt_ip_address>_connAuthFile`

Le `beegfs_client` rôle détectera et appliquera automatiquement ce fichier aux clients lorsqu'il sera présent.

> Si vous n'avez pas utilisé le `beegfs_client` rôle pour configurer les clients, vous devez distribuer manuellement le fichier d'authentification à chaque client et configurer le paramètre `connAuthFile` dans le fichier `beegfs-client.conf`. Lors d'une mise à niveau depuis une version de BeeGFS sans authentification basée sur la connexion, les clients perdront l'accès sauf si vous désactivez l'authentification basée sur la connexion pendant la mise à niveau en définissant `beegfs_ha_conn_auth_enabled: false` dans `group_vars/ha_cluster.yml` (non recommandé).

Pour plus de détails et d'options de configuration alternatives, consultez l'étape de configuration de l'authentification de connexion dans la section "Spécifiez la configuration de nœud de fichier commun".

Mise à jour vers BeeGFS v8

Suivez ces étapes pour mettre à niveau votre cluster BeeGFS HA de la version 7.4.6 à BeeGFS v8.

Présentation

BeeGFS v8 introduit plusieurs changements importants qui nécessitent une configuration supplémentaire

avant la mise à niveau depuis BeeGFS v7. Ce document vous guide dans la préparation de votre cluster aux nouvelles exigences de BeeGFS v8, puis dans la mise à niveau vers BeeGFS v8.

> Avant de procéder à la mise à niveau vers BeeGFS v8, assurez-vous que votre système exécute au moins BeeGFS 7.4.6. Tout cluster exécutant une version antérieure à BeeGFS 7.4.6 doit d'abord "Mise à jour vers la version 7.4.6" avant de poursuivre cette procédure de mise à niveau vers BeeGFS v8.

Principaux changements dans BeeGFS v8

BeeGFS v8 introduit les changements majeurs suivants :

- **Application des licences :** BeeGFS v8 requiert une licence pour utiliser les fonctionnalités premium telles que les pools de stockage, les cibles de stockage distantes, BeeOND, et plus encore. Procurez-vous une licence valide pour votre cluster BeeGFS avant la mise à niveau. Si nécessaire, vous pouvez obtenir une licence d'évaluation temporaire de BeeGFS v8 auprès du "Portail de licences BeeGFS".
- **Migration de la base de données du service de gestion :** Pour activer la configuration avec le nouveau format basé sur TOML dans BeeGFS v8, vous devez migrer manuellement votre base de données du service de gestion BeeGFS v7 vers le format BeeGFS v8 mis à jour.
- **Chiffrement TLS :** BeeGFS v8 introduit TLS pour sécuriser la communication entre les services. Vous devrez générer et distribuer des certificats TLS pour le service de gestion BeeGFS et l' `beegfs`utilitaire en ligne de commande dans le cadre de la mise à niveau.

Pour plus de détails et les modifications supplémentaires apportées à BeeGFS 8, consultez le "Guide de mise à niveau BeeGFS v8.0.0".

> La mise à niveau vers BeeGFS v8 nécessite une interruption du cluster. De plus, les clients BeeGFS v7 ne peuvent pas se connecter aux clusters BeeGFS v8. Coordonnez soigneusement le calendrier de mise à niveau entre le cluster et les clients afin de minimiser l'impact sur les opérations.

Préparez votre cluster BeeGFS pour la mise à niveau

Avant de commencer la mise à niveau, préparez soigneusement votre environnement afin d'assurer une transition en douceur et de minimiser l'interruption.

1. Assurez-vous que votre cluster est dans un état sain, avec tous les services BeeGFS exécutés sur leurs nœuds préférés. À partir d'un nœud de fichiers exécutant les services BeeGFS, vérifiez que toutes les ressources Pacemaker sont exécutées sur leurs nœuds préférés :

   ```
   pcs status
   ```

2. Enregistrez et sauvegardez la configuration de votre cluster.
   a. Consultez le "Documentation de sauvegarde BeeGFS" pour obtenir des instructions sur la sauvegarde de la configuration de votre cluster.
   b. Sauvegardez le répertoire de données de gestion existant :

```
cp -r /mnt/mgmt_tgt_mgmt01/data
/mnt/mgmt_tgt_mgmt01/data_beegfs_v7_backup_$(date +%Y%m%d)
```

c. Exécutez les commandes suivantes depuis un client beegfs et enregistrez leur sortie pour référence :

```
beegfs-ctl --getentryinfo --verbose /path/to/beegfs/mountpoint
```

d. Si vous utilisez la mise en miroir, recueillez des informations détaillées sur l'état :

```
beegfs-ctl --listtargets --longnodes --state --spaceinfo
--mirrorgroups --nodetype=meta
beegfs-ctl --listtargets --longnodes --state --spaceinfo
--mirrorgroups --nodetype=storage
```

3. Préparez vos clients à l'interruption `beegfs-client` des services. Pour chaque client, exécutez :

```
systemctl stop beegfs-client
```

4. Pour chaque cluster Pacemaker, désactivez STONITH. Cela vous permettra de vérifier l'intégrité du cluster après la mise à niveau sans provoquer de redémarrages inutiles des nœuds.

```
pcs property set stonith-enabled=false
```

5. Pour tous les clusters Pacemaker dans l'espace de noms BeeGFS, utilisez PCS pour arrêter le cluster :

```
pcs cluster stop --all
```

Mettez à niveau les packages BeeGFS

Sur tous les nœuds de fichiers du cluster, ajoutez le dépôt de paquets BeeGFS v8 correspondant à votre distribution Linux. Des instructions pour utiliser les dépôts officiels BeeGFS sont disponibles à "page de téléchargement BeeGFS". Sinon, configurez votre dépôt miroir local BeeGFS en conséquence.

La procédure suivante décrit comment procéder à l'aide du dépôt officiel BeeGFS 8.2 sur des nœuds de fichiers RHEL 9. Effectuez les étapes suivantes sur tous les nœuds de fichiers du cluster :

1. Importez la clé GPG de BeeGFS :

```
rpm --import https://www.beegfs.io/release/beegfs_8.2/gpg/GPG-KEY-beegfs
```

2. Importez le dépôt BeeGFS :

```
curl -L -o /etc/yum.repos.d/beegfs-rhel9.repo
https://www.beegfs.io/release/beegfs_8.2/dists/beegfs-rhel9.repo
```

Supprimez tous les dépôts BeeGFS précédemment configurés pour éviter les conflits avec le nouveau dépôt BeeGFS v8.

3. Videz le cache de votre gestionnaire de paquets :

```
dnf clean all
```

4. Sur tous les nœuds de fichiers, mettez à jour les paquets BeeGFS vers BeeGFS 8.2.

```
dnf update beegfs-mgmtd beegfs-storage beegfs-meta libbeegfs-ib
```

Dans un cluster standard, le `beegfs-mgmtd` package ne sera mis à jour que sur les deux premiers nœuds de fichiers.

Mettre à niveau la base de données de gestion

Sur l'un des nœuds de fichiers exécutant le service de gestion BeeGFS, effectuez les étapes suivantes pour migrer la base de données de gestion de BeeGFS v7 vers v8.

1. Lister tous les périphériques NVMe et filtrer selon la cible de gestion :

```
nvme netapp smdevices | grep mgmt_tgt
```

   a. Notez le chemin d'accès au périphérique dans la sortie.

   b. Montez le périphérique cible de gestion sur le point de montage cible de gestion existant (remplacez `/dev/nvmeXnY` par le chemin d'accès à votre périphérique) :

```
mount /dev/nvmeXnY /mnt/mgmt_tgt_mgmt01/
```

2. Importez vos données de gestion BeeGFS 7 dans le nouveau format de base de données en exécutant :

```
/opt/beegfs/sbin/beegfs-mgmtd --import-from
-v7=/mnt/mgmt_tgt_mgmt01/data/
```

   Résultat attendu:

```
Created new database version 3 at "/var/lib/beegfs/mgmtd.sqlite".
Successfully imported v7 management data from
"/mnt/mgmt_tgt_mgmt01/data/".
```

> L'importation automatique peut échouer dans certains cas en raison des exigences de validation plus strictes dans BeeGFS v8. Par exemple, si des cibles sont affectées à des pools de stockage inexistants, l'importation échouera. Si la migration de la base de données échoue, ne procédez pas à la mise à niveau. Contactez le support NetApp pour obtenir de l'aide concernant la résolution des problèmes de migration de la base de données. À titre de solution temporaire, vous pouvez rétrograder les packages BeeGFS v8 et continuer à utiliser BeeGFS v7 pendant que le problème est résolu.

3. Déplacez le fichier SQLite généré vers le point de montage du service de gestion :

```
mv /var/lib/beegfs/mgmtd.sqlite /mnt/mgmt_tgt_mgmt01/data/
```

4. Déplacez le fichier généré `beegfs-mgmtd.toml` vers le point de montage du service de gestion :

```
mv /etc/beegfs/beegfs-mgmtd.toml /mnt/mgmt_tgt_mgmt01/mgmt_config/
```

La préparation du fichier de configuration `beegfs-mgmtd.toml` sera effectuée après avoir terminé les étapes de configuration de la licence et du chiffrement TLS dans les sections suivantes.

Configurer les licences

1. Installez les packages de licence beegfs sur tous les nœuds qui exécutent le service de gestion beegfs. Il s'agit généralement des deux premiers nœuds du cluster :

```
dnf install libbeegfs-license
```

2. Téléchargez votre fichier de licence BeeGFS v8 sur les nœuds de gestion et placez-le à :

```
/etc/beegfs/license.pem
```

Configurer le chiffrement TLS

BeeGFS v8 requiert le chiffrement TLS pour sécuriser les communications entre les services de gestion et les clients. Il existe trois options pour configurer le chiffrement TLS sur les communications réseau entre les services de gestion et les services clients. La méthode recommandée et la plus sécurisée consiste à utiliser des certificats signés par une autorité de certification de confiance. Vous pouvez également créer votre propre autorité de certification locale pour signer les certificats de votre cluster BeeGFS. Pour les environnements où le chiffrement n'est pas requis ou pour le dépannage, TLS peut être entièrement désactivé, bien que cela soit déconseillé car cela expose des informations sensibles au réseau.

Avant de continuer, suivez les instructions du "Configurer le chiffrement TLS pour BeeGFS 8" guide pour configurer le chiffrement pour votre environnement.

Configuration du service de gestion des mises à jour

Préparez le fichier de configuration du service de gestion BeeGFS v8 en transférant manuellement les paramètres de votre fichier de configuration BeeGFS v7 dans le fichier `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml`.

1. Sur le nœud de gestion où la cible de gestion est montée, référencez le `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.conf` fichier de service de gestion pour BeeGFS 7, puis transférez tous les paramètres dans le fichier `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml`. Pour une configuration de base, votre `beegfs-mgmtd.toml` pourrait ressembler à ce qui suit :

   ```
   beemsg-port = 8008
   grpc-port = 8010
   log-level = "info"
   node-offline-timeout = "900s"
   quota-enable = false
   auth-disable = false
   auth-file = "/etc/beegfs/<mgmt_service_ip>_connAuthFile"
   db-file = "/mnt/mgmt_tgt_mgmt01/data/mgmtd.sqlite"
   license-disable = false
   license-cert-file = "/etc/beegfs/license.pem"
   tls-disable = false
   tls-cert-file = "/etc/beegfs/mgmtd_tls_cert.pem"
   tls-key-file = "/etc/beegfs/mgmtd_tls_key.pem"
   interfaces = ['i1b:mgmt_1', 'i2b:mgmt_2']
   ```

   Adaptez tous les chemins selon les besoins pour correspondre à votre environnement et à votre configuration TLS.

2. Sur chaque nœud de fichiers exécutant des services de gestion, modifiez votre fichier de service systemd pour qu'il pointe vers le nouvel emplacement du fichier de configuration.

   ```
   sudo sed -i 's|ExecStart=.*|ExecStart=nice -n -3
   /opt/beegfs/sbin/beegfs-mgmtd --config-file
   /mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml|'
   /etc/systemd/system/beegfs-mgmtd.service
   ```

   a. Recharger systemd :

      ```
      systemctl daemon-reload
      ```

3. Pour chaque nœud de fichier exécutant des services de gestion, ouvrez le port 8010 pour la

communication gRPC du service de gestion.

a. Ajoutez le port 8010/tcp à la zone beegfs :

```
sudo firewall-cmd --zone=beegfs --permanent --add-port=8010/tcp
```

b. Rechargez le pare-feu pour appliquer la modification :

```
sudo firewall-cmd --reload
```

Mettre à jour le script de surveillance BeeGFS

Le script OCF de `beegfs-monitor`Pacemaker nécessite des mises à jour pour prendre en charge le nouveau format de configuration TOML et la gestion des services systemd. Mettez à jour le script sur un nœud du cluster, puis copiez le script mis à jour sur tous les autres nœuds.

1. Créez une sauvegarde du script actuel :

```
cp /usr/lib/ocf/resource.d/eseries/beegfs-monitor
/usr/lib/ocf/resource.d/eseries/beegfs-monitor.bak.$(date +%F)
```

2. Mettez à jour le chemin du fichier de configuration de gestion de `.conf` à `.toml` :

```
sed -i 's|mgmt_config/beegfs-mgmtd\.conf|mgmt_config/beegfs-mgmtd.toml|'
/usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

Sinon, localisez manuellement le bloc suivant dans le script :

```
case $type in
  management)
    conf_path="${configuration_mount}/mgmt_config/beegfs-mgmtd.conf"
    ;;
```

Et remplacez-le par :

```
case $type in
  management)
    conf_path="${configuration_mount}/mgmt_config/beegfs-mgmtd.toml"
    ;;
```

3. Mettez à jour les `get_interfaces()` et `get_subnet_ips()` fonctions pour prendre en charge la configuration TOML :

a. Ouvrez le script dans un éditeur de texte :

```
vi /usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

b. Localisez les deux fonctions : `get_interfaces()` et `get_subnet_ips()`.

c. Supprimez les deux fonctions complètes, en commençant à `get_interfaces()` jusqu'à la fin de `get_subnet_ips()`.

d. Copiez et collez les fonctions mises à jour suivantes à leur place :

```
# Return network communication interface name(s) from the BeeGFS
resource's connInterfaceFile
get_interfaces() {
  # Determine BeeGFS service network IP interfaces.
  if [ "$type" = "management" ]; then
    interfaces_line=$(grep "^interfaces =" "$conf_path")
    interfaces_list=$(echo "$interfaces_line" | sed "s/.*= \[\(.*
\)\]/\1/")
    interfaces=$(echo "$interfaces_list" | tr -d "'" | tr -d " " | tr
',' '\n')

    for entry in $interfaces; do
      echo "$entry" | cut -d ':' -f 1
    done
  else
    connInterfacesFile_path=$(grep "^connInterfacesFile" "$conf_path"
| tr -d "[:space:]" | cut -f 2 -d "=")

    if [ -f "$connInterfacesFile_path" ]; then
      while read -r entry; do
        echo "$entry" | cut -f 1 -d ':'
      done < "$connInterfacesFile_path"
    fi
  fi
}

# Return list containing all the BeeGFS resource's usable IP
addresses. *Note that these are filtered by the connNetFilterFile
entries.
get_subnet_ips() {
  # Determine all possible BeeGFS service network IP addresses.
  if [ "$type" != "management" ]; then
    connNetFilterFile_path=$(grep "^connNetFilterFile" "$conf_path" |
tr -d "[:space:]" | cut -f 2 -d "=")

    filter_ips=""
    if [ -n "$connNetFilterFile_path" ] && [ -e
$connNetFilterFile_path ]; then
      while read -r filter; do
        filter_ips="$filter_ips $(get_ipv4_subnet_addresses $filter)"
      done < $connNetFilterFile_path
    fi

    echo "$filter_ips"
  fi
}
```

e. Enregistrez et quittez l'éditeur de texte.

f. Exécutez la commande suivante pour vérifier le script pour des erreurs de syntaxe avant de poursuivre. L'absence de résultat indique que le script est syntaxiquement correct.

```
bash -n /usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

4. Copiez le script OCF mis à jour `beegfs-monitor` sur tous les autres nœuds du cluster pour garantir la cohérence :

```
scp /usr/lib/ocf/resource.d/eseries/beegfs-monitor
user@node:/usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

Remettre le cluster en ligne

1. Une fois toutes les étapes de mise à niveau précédentes terminées, remettez le cluster en ligne en démarrant les services BeeGFS sur tous les nœuds.

```
pcs cluster start --all
```

2. Vérifiez que le `beegfs-mgmtd` service a démarré correctement :

```
journalctl -xeu beegfs-mgmtd
```

Le résultat attendu comprend des lignes telles que :

```
Started Cluster Controlled beegfs-mgmtd.
Loaded config file from "/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-
mgmtd.toml"
Successfully initialized certificate verification library.
Successfully loaded license certificate: TMP-113489268
Opened database at "/mnt/mgmt_tgt_mgmt01/data/mgmtd.sqlite"
Listening for BeeGFS connections on [::]:8008
Serving gRPC requests on [::]:8010
```

Si des erreurs apparaissent dans les journaux, vérifiez les chemins d'accès au fichier de configuration de gestion et assurez-vous que toutes les valeurs ont été correctement transférées depuis le fichier de configuration BeeGFS 7.

3. Exécutez `pcs status` et vérifiez que le cluster est sain et que les services sont démarrés sur leurs nœuds préférés.
4. Une fois que le cluster est vérifié comme étant sain, réactivez STONITH :

```
pcs property set stonith-enabled=true
```

5. Passez à la section suivante pour mettre à niveau les clients BeeGFS dans le cluster et vérifier l'état de santé du cluster BeeGFS.

Mise à niveau des clients BeeGFS

Après avoir réussi la mise à niveau de votre cluster vers BeeGFS v8, vous devez également mettre à niveau tous les clients BeeGFS.

Les étapes suivantes décrivent le processus de mise à niveau des clients BeeGFS sur un système basé sur Ubuntu.

1. Si ce n'est pas déjà fait, arrêtez le service client BeeGFS :

```
systemctl stop beegfs-client
```

2. Ajoutez le dépôt de paquets BeeGFS v8 pour votre distribution Linux. Des instructions pour utiliser les dépôts officiels BeeGFS se trouvent à "^Page de téléchargement BeeGFS". Sinon, configurez votre dépôt miroir BeeGFS local en conséquence.

   Les étapes suivantes utilisent le dépôt officiel BeeGFS 8.2 sur un système basé sur Ubuntu :

3. Importez la clé GPG de BeeGFS :

```
wget https://www.beegfs.io/release/beegfs_8.2/gpg/GPG-KEY-beegfs -O
/etc/apt/trusted.gpg.d/beegfs.asc
```

4. Téléchargez le fichier du dépôt :

```
wget https://www.beegfs.io/release/beegfs_8.2/dists/beegfs-noble.list -O
/etc/apt/sources.list.d/beegfs.list
```

   Supprimez tous les dépôts BeeGFS précédemment configurés pour éviter les conflits avec le nouveau dépôt BeeGFS v8.

5. Mettez à jour les packages clients BeeGFS :

```
apt-get update
apt-get install --only-upgrade beegfs-client
```

6. Configurez TLS pour le client. TLS est requis pour utiliser la CLI BeeGFS. Consultez la "Configurer le chiffrement TLS pour BeeGFS 8" procédure pour configurer TLS sur le client.
7. Démarrez le service client BeeGFS :

```
systemctl start beegfs-client
```

Vérifier la mise à jour

Après avoir terminé la mise à niveau vers BeeGFS v8, exécutez les commandes suivantes pour vérifier que la mise à niveau a réussi.

1. Vérifiez que l'inode racine appartient bien au même nœud de métadonnées qu'auparavant. Cela devrait se faire automatiquement si vous avez utilisé la `import-from-v7` fonctionnalité dans le service de gestion :

```
beegfs entry info /mnt/beegfs
```

2. Vérifiez que tous les nœuds et cibles sont en ligne et en bon état :

```
beegfs health check
```

> ⓘ Si la vérification de la « capacité disponible » signale que les cibles manquent d'espace libre, vous pouvez ajuster les seuils du « pool de capacité » définis dans le `beegfs-mgmtd.toml` fichier afin qu'ils soient mieux adaptés à votre environnement.

Mise à niveau des packages Pacemaker et Corosync dans un cluster haute disponibilité

Procédez comme suit pour mettre à niveau les packages Pacemaker et Corosync dans un cluster HA.

Présentation

La mise à niveau de Pacemaker et Corosync garantit que le cluster bénéficie de nouvelles fonctionnalités, de nouveaux correctifs de sécurité et d'améliorations des performances.

Approche de mise à niveau

Il existe deux approches recommandées pour la mise à niveau d'un cluster : une mise à niveau par déploiement ou un arrêt complet du cluster. Chaque approche a ses propres avantages et inconvénients. La procédure de mise à niveau peut varier en fonction de la version de votre Pacemaker. Reportez-vous à la documentation de ClusterLabs "Mise à niveau d'un cluster Pacemaker"pour déterminer quelle approche utiliser. Avant de suivre une approche de mise à niveau, vérifiez que :

- Les nouveaux packages Pacemaker et Corosync sont pris en charge dans la solution NetApp BeeGFS.
- Il existe des sauvegardes valides pour votre système de fichiers BeeGFS et la configuration de cluster Pacemaker.
- Le cluster est en bon état.

Mise à jour du déploiement

Cette méthode implique de supprimer chaque nœud du cluster, de le mettre à niveau, puis de le réintégrer dans le cluster jusqu'à ce que tous les nœuds exécutent la nouvelle version. Cette approche assure le fonctionnement continu du cluster, ce qui est idéal pour les clusters haute disponibilité de plus grande taille, mais comporte le risque d'exécuter des versions mixtes lors du processus. Cette approche doit être évitée dans un cluster à deux nœuds.

1. Vérifiez que le cluster est dans un état optimal, chaque service BeeGFS étant exécuté sur le nœud favori. Voir "Vérifiez l'état du cluster" pour plus de détails.
2. Pour la mise à niveau du nœud, mettez-le en mode veille afin d'exploiter (ou de déplacer) tous les services BeeGFS :

   ```
   pcs node standby <HOSTNAME>
   ```

3. Vérifier que les services du nœud ont été vidangés en exécutant :

   ```
   pcs status
   ```

   Assurez-vous qu'aucun service n'est signalé comme `Started` sur le nœud en veille.

   > Selon la taille de votre cluster, le déplacement des services vers le nœud sœur peut prendre quelques secondes, voire quelques minutes. Si un service BeeGFS ne démarre pas sur le nœud Sister, reportez-vous au "Guides de dépannage".

4. Arrêter le cluster sur le nœud :

   ```
   pcs cluster stop <HOSTNAME>
   ```

5. Mettez à niveau les packages Pacemaker, Corosync et pcs sur le nœud :

   > Les commandes du gestionnaire de paquets varient selon le système d'exploitation. Les commandes suivantes sont destinées aux systèmes exécutant RHEL 8 et versions ultérieures.

   ```
   dnf update pacemaker-<version>
   ```

   ```
   dnf update corosync-<version>
   ```

   ```
   dnf update pcs-<version>
   ```

6. Démarrer les services de cluster Pacemaker sur le nœud :

```
pcs cluster start <HOSTNAME>
```

7. Si le `pcs` pack a été mis à jour, réauthentifier le nœud avec le cluster :

```
pcs host auth <HOSTNAME>
```

8. Vérifiez que la configuration du stimulateur est toujours valide avec l' `crm_verify`outil.

> Cette vérification doit être effectuée une seule fois lors de la mise à niveau du cluster.

```
crm_verify -L -V
```

9. Mettre le nœud hors veille :

```
pcs node unstandby <HOSTNAME>
```

10. Retransférez tous les services BeeGFS vers le nœud de votre choix :

```
pcs resource relocate run
```

11. Répétez les étapes précédentes pour chaque nœud du cluster jusqu'à ce que tous les nœuds exécutent les versions Pacemaker, Corosync et pcs souhaitées.
12. Enfin, exécutez `pcs status` et vérifiez que le cluster fonctionne correctement et `Current DC` indique la version du stimulateur souhaitée.

> Si le `Current DC` indique « version limite », un nœud du cluster fonctionne toujours avec la version précédente de Pacemaker et doit être mis à niveau. Si un nœud mis à niveau ne parvient pas à rejoindre le cluster ou si les ressources ne démarrent pas, consultez les journaux du cluster et consultez les notes de mise à jour ou les guides de l'utilisateur Pacemaker pour connaître les problèmes de mise à niveau connus.

Arrêt complet du cluster

Dans cette approche, tous les nœuds et toutes les ressources du cluster sont arrêtés, les nœuds sont mis à niveau, puis le cluster est redémarré. Cette approche est nécessaire si les versions Pacemaker et Corosync ne prennent pas en charge une configuration en version mixte.

1. Vérifiez que le cluster est dans un état optimal, chaque service BeeGFS étant exécuté sur le nœud favori. Voir "Vérifiez l'état du cluster" pour plus de détails.
2. Arrêtez le logiciel de cluster (Pacemaker et Corosync) sur tous les nœuds.

Selon la taille du cluster, l'arrêt de tout le cluster peut prendre quelques secondes, voire quelques minutes.

```
pcs cluster stop --all
```

3. Une fois les services de cluster arrêtés sur tous les nœuds, mettez à niveau les packages Pacemaker, Corosync et pcs sur chaque nœud en fonction de vos besoins.

Les commandes du gestionnaire de paquets varient selon le système d'exploitation. Les commandes suivantes sont destinées aux systèmes exécutant RHEL 8 et versions ultérieures.

```
dnf update pacemaker-<version>
```

```
dnf update corosync-<version>
```

```
dnf update pcs-<version>
```

4. Une fois la mise à niveau de tous les nœuds effectuée, démarrez le logiciel du cluster sur tous les nœuds :

```
pcs cluster start --all
```

5. Si le `pcs` pack a été mis à jour, réauthentifier chaque nœud du cluster :

```
pcs host auth <HOSTNAME>
```

6. Enfin, exécutez `pcs status` et vérifiez que le cluster est en bon état et `Current DC` indique la version correcte du Pacemaker.

Si le `Current DC` indique « version limite », un nœud du cluster fonctionne toujours avec la version précédente de Pacemaker et doit être mis à niveau.

Mettez à jour le micrologiciel de l'adaptateur de nœud de fichier

Procédez comme suit pour mettre à jour les cartes ConnectX-7 du nœud de fichiers vers la dernière version du micrologiciel.

Présentation

La mise à jour du firmware de l'adaptateur ConnectX-7 peut être nécessaire pour prendre en charge un nouveau pilote MLNX_OFED, activer de nouvelles fonctionnalités ou corriger des bogues. Ce guide utilisera

l'utilitaire NVIDIA `mlxfwmanager` pour les mises à jour de la carte en raison de sa facilité d'utilisation et de son efficacité.

Mise à niveau

Ce guide présente deux approches de mise à jour du firmware de la carte ConnectX-7 : une mise à jour en continu et une mise à jour de cluster à deux nœuds. Choisissez l'approche de mise à jour appropriée en fonction de la taille de votre cluster. Avant d'effectuer les mises à jour du micrologiciel, vérifiez que :

- Un pilote MLNX_OFED pris en charge est installé, reportez-vous au "exigences technologiques".
- Il existe des sauvegardes valides pour votre système de fichiers BeeGFS et la configuration de cluster Pacemaker.
- Le cluster est en bon état.

Préparation de la mise à jour du micrologiciel

Il est recommandé d'utiliser l'utilitaire de NVIDIA `mlxfwmanager` pour mettre à jour le micrologiciel de l'adaptateur d'un nœud, qui est fourni avec le pilote MLNX_OFED de NVIDIA. Avant de commencer les mises à jour, téléchargez l'image du micrologiciel de la carte à partir de "Site de support NVIDIA" et stockez-la sur chaque nœud de fichier.

Pour les cartes Lenovo ConnectX-7, utilisez l' `mlxfwmanager_LES`outil, disponible sur la page NVIDIA"Micrologiciel OEM".

Approche de mise à jour par roulement

Cette approche est recommandée pour tout cluster haute disponibilité de plus de deux nœuds. Cette approche implique de mettre à jour le firmware des adaptateurs sur un nœud de fichiers à la fois afin que le cluster haute disponibilité puisse continuer à traiter les demandes, bien qu'il soit recommandé d'éviter de traiter les E/S pendant ce temps.

1. Vérifiez que le cluster est dans un état optimal, chaque service BeeGFS étant exécuté sur le nœud favori. Voir "Vérifiez l'état du cluster" pour plus de détails.
2. Choisissez un nœud de fichiers à mettre à jour et placez-le en mode veille pour drains (ou déplacer) tous les services BeeGFS de ce nœud :

   ```
   pcs node standby <HOSTNAME>
   ```

3. Vérifier que les services du nœud ont été vidangés en exécutant :

   ```
   pcs status
   ```

   Vérifiez qu'aucun service ne signale le `Started` nœud en mode veille.

Selon la taille du cluster, le déplacement des services BeeGFS peut prendre quelques secondes, voire quelques minutes. Si un service BeeGFS ne démarre pas sur le nœud Sister, reportez-vous au "Guides de dépannage".

4. Mettez à jour le micrologiciel de l'adaptateur à l'aide de `mlxfwmanager`.

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

Notez `PCI Device Name` que pour chaque adaptateur recevant des mises à jour de micrologiciel.

5. Réinitialisez chaque carte à l'aide de l' `mlxfwreset`utilitaire pour appliquer le nouveau micrologiciel.

> Certaines mises à jour du micrologiciel peuvent nécessiter un redémarrage pour appliquer la mise à jour. Reportez-vous "Limitations de mlxfwreset de NVIDIA"à pour obtenir des conseils. Si un redémarrage est nécessaire, effectuez un redémarrage au lieu de réinitialiser les adaptateurs.

a. Arrêter le service openhm :

```
systemctl stop opensm
```

b. Exécutez la commande suivante pour chacune des `PCI Device Name` opérations précédemment notées.

```
mlxfwreset -d <pci_device_name> reset -y
```

c. Démarrer le service openhm :

```
systemctl start opensm
```

d. Redémarrez le `eseries_nvme_ib.service` .

```
systemctl restart eseries_nvme_ib.service
```

e. Vérifiez que les volumes de la baie de stockage de la série E sont présents.

```
multipath -ll
```

1. Exécutez `ibstat` et vérifiez que toutes les cartes fonctionnent avec la version de micrologiciel souhaitée :

```
ibstat
```

2. Démarrer les services de cluster Pacemaker sur le nœud :

```
pcs cluster start <HOSTNAME>
```

3. Mettre le nœud hors veille :

```
pcs node unstandby <HOSTNAME>
```

4. Retransférez tous les services BeeGFS vers le nœud de votre choix :

```
pcs resource relocate run
```

Répétez ces étapes pour chaque nœud de fichiers du cluster jusqu'à ce que tous les adaptateurs aient été mis à jour.

Approche de mise à jour des clusters à deux nœuds

Cette approche est recommandée pour les clusters haute disponibilité à deux nœuds uniquement. Cette approche est similaire à une mise à jour propagée, mais elle comprend des étapes supplémentaires pour éviter tout temps d'indisponibilité des services lorsque les services de cluster d'un nœud sont arrêtés.

1. Vérifiez que le cluster est dans un état optimal, chaque service BeeGFS étant exécuté sur le nœud favori. Voir "Vérifiez l'état du cluster" pour plus de détails.
2. Choisissez un nœud de fichiers à mettre à jour et placez le nœud en mode veille, ce qui draine (ou déplace) tous les services BeeGFS de ce nœud :

```
pcs node standby <HOSTNAME>
```

3. Vérifier que les ressources du nœud ont été vidées en exécutant :

```
pcs status
```

Vérifiez qu'aucun service ne signale le `Started` nœud en mode veille.

Selon la taille du cluster, le reporting par les services BeeGFS peut prendre quelques secondes, voire quelques minutes, comme `Started` sur le nœud jumeau. Si un service BeeGFS ne démarre pas, reportez-vous au "Guides de dépannage".

4. Placer le cluster en mode maintenance.

```
pcs property set maintenance-mode=true
```

5. Mettez à jour le micrologiciel de l'adaptateur à l'aide de `mlxfwmanager`.

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

Notez `PCI Device Name` que pour chaque adaptateur recevant des mises à jour de micrologiciel.

6. Réinitialisez chaque carte à l'aide de l' `mlxfwreset`utilitaire pour appliquer le nouveau micrologiciel.

> Certaines mises à jour du micrologiciel peuvent nécessiter un redémarrage pour appliquer la mise à jour. Reportez-vous "Limitations de mlxfwreset de NVIDIA"à pour obtenir des conseils. Si un redémarrage est nécessaire, effectuez un redémarrage au lieu de réinitialiser les adaptateurs.

a. Arrêter le service openhm :

```
systemctl stop opensm
```

b. Exécutez la commande suivante pour chacune des `PCI Device Name` opérations précédemment notées.

```
mlxfwreset -d <pci_device_name> reset -y
```

c. Démarrer le service openhm :

```
systemctl start opensm
```

7. Exécutez `ibstat` et vérifiez que toutes les cartes fonctionnent avec la version de micrologiciel souhaitée :

```
ibstat
```

8. Démarrer les services de cluster Pacemaker sur le nœud :

```
pcs cluster start <HOSTNAME>
```

9. Mettre le nœud hors veille :

```
pcs node unstandby <HOSTNAME>
```

10. Sortir le cluster du mode de maintenance.

```
pcs property set maintenance-mode=false
```

11. Retransférez tous les services BeeGFS vers le nœud de votre choix :

```
pcs resource relocate run
```

Répétez ces étapes pour chaque nœud de fichiers du cluster jusqu'à ce que tous les adaptateurs aient été mis à jour.

Mettez à niveau la baie de stockage E-Series

Suivez ces étapes pour mettre à niveau les composants de la baie de stockage E-Series du cluster HA.

Présentation

En conservant les baies de stockage NetApp E-Series de votre cluster de haute disponibilité à jour avec le dernier firmware, vous bénéficiez de performances optimales et d'une sécurité renforcée. Les mises à jour du micrologiciel de la baie de stockage sont appliquées via le système d'exploitation SANtricity, la NVSRAM et les fichiers de micrologiciel du lecteur.

> Bien que les baies de stockage puissent être mises à niveau avec le cluster haute disponibilité en ligne, il est recommandé de placer le cluster en mode de maintenance pour toutes les mises à niveau.

Étapes de mise à niveau du nœud de bloc

Les étapes suivantes expliquent comment mettre à jour le firmware des baies de stockage à l'aide de la `Netapp_Eseries.Santricity` collection Ansible. Avant de continuer, consultez le "Mise à niveau" pour la mise à jour des systèmes E-Series.

> La mise à niveau vers SANtricity OS 11.80 ou versions ultérieures est possible uniquement à partir de 11.70.5P1. La baie de stockage doit d'abord être mise à niveau vers 11.70.5P1 avant d'appliquer d'autres mises à niveau.

1. Vérifiez que votre nœud de contrôle Ansible utilise la dernière collection SANtricity Ansible.
   - Pour les mises à niveau de collecte avec accès à "Galaxy Ansible", exécutez la commande suivante :

     ```
     ansible-galaxy collection install netapp_eseries.santricity --upgrade
     ```

   - Pour les mises à niveau hors ligne, téléchargez le fichier tarball de "Galaxy Ansible"la collection à partir de , transférez-le vers votre nœud de contrôle et exécutez :

     ```
     ansible-galaxy collection install netapp_eseries-santricity-
     <VERSION>.tar.gz --upgrade
     ```

     Voir "Installation de Collections" pour en savoir plus.

2. Obtenez la dernière version du micrologiciel pour votre matrice de stockage et vos lecteurs.
   a. Téléchargez les fichiers du micrologiciel.
      - **SANtricity OS et NVSRAM :** naviguez jusqu'au "Site de support NetApp" et téléchargez la dernière version de SANtricity OS et NVSRAM pour votre modèle de matrice de stockage.
      - **Microprogramme de lecteur :** naviguez jusqu'au "Site du firmware du disque E-Series"et téléchargez le dernier micrologiciel pour chacun des modèles de lecteur de votre matrice de stockage.
   b. Stockez les fichiers du système d'exploitation SANtricity, de la NVSRAM et du firmware des disques dans le `<inventory_directory>/packages` répertoire du nœud de contrôle Ansible.
3. Si nécessaire, mettez à jour les fichiers d'inventaire Ansible de votre cluster afin d'inclure toutes les baies de stockage (nœuds de bloc) nécessitant des mises à jour. Pour obtenir des conseils, voir "Présentation d'Ansible Inventory"la section.
4. Assurez-vous que le cluster est optimal avec chaque service BeeGFS sur le nœud de votre choix. Voir "Vérifiez l'état du cluster" pour plus de détails.
5. Placez le cluster en mode maintenance en suivant les instructions de "Placer le cluster en mode maintenance"la section .
6. Créez un nouveau PlayBook Ansible nommé `update_block_node_playbook.yml`. Remplissez le manuel avec le contenu suivant en remplaçant le système d'exploitation SANtricity, la NVSRAM et les versions de firmware des disques par le chemin de mise à niveau souhaité :

```
- hosts: eseries_storage_systems
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.santricity
  vars:
    eseries_firmware_firmware: "packages/<SantricityOS>.dlp"
    eseries_firmware_nvsram: "packages/<NVSRAM>.dlp"
    eseries_drive_firmware_firmware_list:
      - "packages/<drive_firmware>.dlp"
    eseries_drive_firmware_upgrade_drives_online: true

  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

7. Pour démarrer les mises à jour, exécutez la commande suivante à partir de votre nœud de contrôle Ansible :

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

8. Une fois le manuel de vente terminé, vérifiez que chaque baie de stockage est dans un état optimal.
9. Déplacez le cluster hors du mode de maintenance et vérifiez qu'il est dans un état optimal, chaque service

BeeGFS étant sur le nœud privilégié.

Entretien et maintenance

Services de basculement/rétablissement

Déplacement des services BeeGFS entre les nœuds du cluster.

Présentation

Les services BeeGFS peuvent basculer entre les nœuds du cluster pour s'assurer que les clients sont en mesure de continuer à accéder au système de fichiers en cas de défaillance d'un nœud ou si vous devez effectuer une maintenance planifiée. Cette section décrit différentes méthodes permettant aux administrateurs d'effectuer une réparation sur le cluster après une reprise d'activité ou de déplacer manuellement les services entre les nœuds.

Étapes

Basculement et rétablissement

Basculement (planifié)

De manière générale, lorsque vous devez mettre un nœud de fichier unique hors ligne pour les opérations de maintenance, vous devez déplacer (ou vidanger) tous les services BeeGFS depuis ce nœud. Pour ce faire, le nœud peut d'abord être en veille :

```
pcs node standby <HOSTNAME>
```

Après vérification de l'utilisation `pcs status` toutes les ressources ont été redémarrées sur le nœud de fichier secondaire, vous pouvez arrêter ou apporter d'autres modifications au nœud si nécessaire.

Restauration (après un basculement planifié)

Lorsque vous êtes prêt à restaurer les services BeeGFS sur le nœud préféré s'exécutent d'abord `pcs status` Et vérifiez dans la « liste de nœuds » que l'état est en veille. Si le nœud a été redémarré, il s'affiche hors ligne jusqu'à ce que vous mettent les services du cluster en ligne :

```
pcs cluster start <HOSTNAME>
```

Une fois le nœud mis en ligne hors veille, grâce à :

```
pcs node unstandby <HOSTNAME>
```

Enfin, transférez tous les services BeeGFS vers leurs nœuds préférés avec :

```
pcs resource relocate run
```

Retour arrière (après basculement non planifié)

Si un nœud présente un défaut matériel ou autre, le cluster haute disponibilité doit réagir automatiquement et déplacer ses services vers un nœud sain, ce qui permet aux administrateurs de prendre des mesures correctives. Avant de continuer, reportez-vous à "dépannage"la section pour déterminer la cause du basculement et résoudre tout problème en suspens. Une fois le nœud mis sous tension et en bon état, vous pouvez continuer à le restaurer.

Lorsqu'un nœud démarre après un redémarrage non planifié (ou planifié), les services de cluster ne sont pas configurés pour démarrer automatiquement. Vous devez donc mettre le nœud en ligne avec :

```
pcs cluster start <HOSTNAME>
```

Ensuite, nettoyez toute défaillance de ressource et réinitialisez l'historique d'escrime du nœud :

```
pcs resource cleanup node=<HOSTNAME>
pcs stonith history cleanup <HOSTNAME>
```

Vérifier dans `pcs status` le nœud est en ligne et fonctionne correctement. Par défaut, les services BeeGFS ne sont pas automatiquement rebasculer afin d'éviter tout déplacement accidentel des ressources vers un nœud malsain. Une fois que vous êtes prêt à renvoyer toutes les ressources du cluster à leurs nœuds préférés avec :

```
pcs resource relocate run
```

Déplacement de services BeeGFS individuels vers d'autres nœuds de fichiers

Déplacer définitivement un service BeeGFS vers un nouveau noeud de fichier

Si vous souhaitez modifier de manière permanente le nœud de fichier favori pour un service BeeGFS, ajustez l'inventaire Ansible de sorte que le nœud préféré soit répertorié en premier et exécutez à nouveau le PlayBook Ansible.

Par exemple, dans cet exemple de `inventory.yml` fichier, beegfs_01 est le nœud de fichiers préféré pour exécuter le service de gestion BeeGFS :

```
        mgmt:
          hosts:
            beegfs_01:
            beegfs_02:
```

Inverser l'ordre ferait que les services de gestion seraient préférés sur beegfs_02:

```
        mgmt:
          hosts:
            beegfs_02:
            beegfs_01:
```

Déplacer temporairement un service BeeGFS vers un autre nœud de fichier

De manière générale, si un nœud est en cours de maintenance, il convient d'utiliser [les étapes de basculement et de retour arrière](#le basculement et la restauration) pour déplacer tous les services hors de ce nœud.

Si vous devez déplacer un service individuel vers un autre nœud de fichiers :

```
pcs resource move <SERVICE>-monitor <HOSTNAME>
```

> Ne spécifiez pas les ressources individuelles ou le groupe de ressources. Spécifiez toujours le nom du moniteur pour le service BeeGFS que vous souhaitez déplacer. Par exemple, pour déplacer le service de gestion BeeGFS vers beegfs_02, exécutez : `pcs resource move mgmt-monitor beegfs_02`. Ce processus peut être répété afin de déplacer un ou plusieurs services hors de leurs nœuds préférés. Vérifiez à l'aide des `pcs status` services qui ont été déplacés/démarrés sur le nouveau nœud.

Pour déplacer un service BeeGFS vers son nœud préféré, effacez d'abord les contraintes de ressources temporaires (en répétant cette étape comme nécessaire pour plusieurs services) :

```
pcs resource clear <SERVICE>-monitor
```

Ensuite, une fois prêt à rapatrier les services sur les nœuds de leur choix :

```
pcs resource relocate run
```

Notez que cette commande permet de transférer tous les services qui ne disposent plus de contraintes temporaires en termes de ressources, situés sur les nœuds de leur choix.

Placer le cluster en mode maintenance

Empêcher le cluster de haute disponibilité de réagir accidentellement aux changements prévus dans l'environnement.

Présentation

Le fait de mettre le cluster en mode maintenance désactive toute la surveillance des ressources et empêche Pacemaker de déplacer ou de gérer des ressources dans le cluster. Toutes les ressources restent exécutées sur leurs nœuds d'origine, peu importe la condition de panne temporaire qui empêcherait leur accès. Voici quelques scénarios recommandés/utiles :

- Maintenance du réseau pouvant interrompre temporairement les connexions entre les nœuds de fichiers et les services BeeGFS.
- Mises à niveau des nœuds de blocs.
- Mises à jour du système d'exploitation de nœud de fichiers, du noyau ou d'autres modules.

En général, la seule raison de placer manuellement le cluster en mode de maintenance est d'éviter que le système ne réagisse à des modifications externes de l'environnement. Si un nœud individuel du cluster nécessite une réparation physique, n'utilisez pas le mode de maintenance et placez simplement ce nœud en veille après la procédure ci-dessus. Notez que le changement d'Ansible place automatiquement le cluster en mode de maintenance pour faciliter la plupart des opérations de maintenance logicielle, y compris les mises à niveau et les modifications de configuration.

Étapes

Pour vérifier si le cluster est en mode maintenance, exécutez :

```
pcs property config
```

La `maintenance-mode` propriété n'apparaît pas si le cluster fonctionne normalement. Si le cluster est actuellement en mode maintenance, la propriété indique `true`. Pour activer le mode maintenance, exécutez :

```
pcs property set maintenance-mode=true
```

Vous pouvez vérifier en exécutant l'état pcs et en vous assurant que toutes les ressources affichent « (non géré) ». Pour mettre le cluster hors mode maintenance, exécutez :

```
pcs property set maintenance-mode=false
```

Arrêtez et démarrez le cluster

Arrêt et démarrage du cluster HA avec élégance.

Présentation

Cette section décrit comment arrêter et redémarrer le cluster BeeGFS. Par exemple, la maintenance électrique ou la migration d'un data Center à l'autre ou d'un rack peut être nécessaire.

Étapes

Si, pour une raison quelconque, vous devez arrêter tout le cluster BeeGFS et arrêter tous les services exécutent :

```
pcs cluster stop --all
```

Il est également possible d'arrêter le cluster sur des nœuds individuels (qui basculeront automatiquement les services vers un autre nœud), bien qu'il soit d'abord recommandé de mettre le nœud en veille (voir la

"basculement" section) :

```
pcs cluster stop <HOSTNAME>
```

Pour démarrer les ressources et les services du cluster sur tous les nœuds, exécutez :

```
pcs cluster start --all
```

Ou démarrer les services sur un nœud spécifique avec :

```
pcs cluster start <HOSTNAME>
```

À ce stade, exécuter `pcs status` Et vérifiez que le cluster et les services BeeGFS démarrent sur tous les nœuds et que les services sont exécutés sur les nœuds que vous attendez.

Selon la taille du cluster, l'arrêt de l'ensemble du cluster peut prendre des secondes ou des minutes, ou s'afficher comme démarré dans `pcs status`. si `pcs cluster <COMMAND>` se bloque pendant plus de cinq minutes, avant d'exécuter « Ctrl+C » pour annuler la commande, connectez-vous à chaque nœud du cluster et utilisez `pcs status` pour voir si les services de cluster (Corosync/Pacemaker) sont toujours en cours d'exécution sur ce nœud. A partir de n'importe quel nœud où le cluster est toujours actif, vous pouvez vérifier les ressources qui bloquent le cluster. Résoudre manuellement le problème et la commande doit être terminée ou réexécutée pour arrêter les services restants.

Remplacer les nœuds de fichiers

Remplacement d'un noeud de fichier si le serveur d'origine est défectueux.

Présentation

Voici un aperçu des étapes nécessaires au remplacement d'un noeud de fichier dans le cluster. Ces étapes présupposent que le nœud de fichier a échoué en raison d'un problème matériel et a été remplacé par un nouveau nœud de fichier identique.

Étapes :

1. Remplacez physiquement le nœud de fichiers et restaurez tout le câblage vers le nœud de bloc et le réseau de stockage.
2. Réinstallez le système d'exploitation sur le nœud de fichier, y compris l'ajout d'abonnements Red Hat.
3. Configurez la mise en réseau BMC et la gestion sur le nœud de fichiers.
4. Mettez à jour l'inventaire Ansible si le nom d'hôte, l'IP, les mappages de l'interface PCIe vers l'interface logique ou tout autre élément modifié concernant le nouveau nœud de fichier. En général, cette opération n'est pas nécessaire si le nœud a été remplacé par le même matériel serveur et que vous utilisez la configuration réseau d'origine.
   a. Par exemple, si le nom d'hôte a changé, créez (ou renommez) le fichier d'inventaire du nœud (`host_vars/<NEW_NODE>.yml`) Puis dans le fichier d'inventaire Ansible (`inventory.yml`),

remplacer le nom de l'ancien nœud par le nouveau nom de nœud :

```
all:
    ...
    children:
    ha_cluster:
        children:
        mgmt:
            hosts:
            node_h1_new:    # Replaced "node_h1" with "node_h1_new"
            node_h2:
```

5. Depuis un des autres nœuds du cluster, supprimer l'ancien nœud : `pcs cluster node remove <HOSTNAME>`.

   NE PAS POURSUIVRE AVANT D'EXÉCUTER CETTE ÉTAPE.

6. Sur le nœud de contrôle Ansible :
   a. Supprimez l'ancienne clé SSH avec :

   ```
   `ssh-keygen -R <HOSTNAME_OR_IP>`
   ```

   b. Configurez SSH sans mot de passe sur le nœud remplacer par :

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

7. Exécutez à nouveau le PlayBook Ansible pour configurer le nœud et l'ajouter au cluster :

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

8. A ce stade, exécuter `pcs status` et vérifiez que le nœud remplacé est maintenant répertorié et que les services sont en cours d'exécution.

Développez ou réduisez le cluster

Ajouter ou supprimer des éléments de base du cluster.

Présentation

Cette section présente divers éléments à prendre en compte et diverses options pour ajuster la taille de votre cluster BeeGFS HA. La taille du cluster est généralement ajustée en ajoutant ou en supprimant des éléments de base, qui sont généralement deux nœuds de fichiers configurés comme une paire haute disponibilité. Il est également possible d'ajouter ou de supprimer des nœuds de fichiers individuels (ou d'autres types de nœuds de cluster) si nécessaire.

Ajout d'un module au cluster

Considérations

Le développement du cluster par l'ajout d'éléments de base supplémentaires est un processus simple. Avant de commencer, les restrictions s'imposent concernant le nombre minimal et maximal de nœuds de cluster dans chaque cluster haute disponibilité. Déterminer si vous devez ajouter des nœuds au cluster haute disponibilité existant ou créer un nouveau cluster haute disponibilité. En général, chaque élément de base se compose de deux nœuds de fichiers, mais trois nœuds représentent le nombre minimum de nœuds par cluster (pour établir le quorum) et dix est le nombre maximum recommandé (testé). Pour les scénarios avancés, il est possible d'ajouter un nœud « Tiebreaker » unique qui n'exécute aucun service BeeGFS lors du déploiement d'un cluster à deux nœuds. Si vous envisagez un tel déploiement, contactez le support NetApp.

Gardez à l'esprit ces restrictions et toute future croissance des clusters lorsque vous décidez d'étendre le cluster. Par exemple, si vous disposez d'un cluster à six nœuds et que vous devez en ajouter quatre autres, il est recommandé de simplement démarrer un nouveau cluster haute disponibilité.

N'oubliez pas qu'un seul système de fichiers BeeGFS peut consister en plusieurs clusters HA indépendants. Les systèmes de fichiers peuvent ainsi continuer à évoluer au-delà des limites recommandées/strictes des composants de cluster haute disponibilité sous-jacents.

Étapes

Lorsque vous ajoutez un élément de base à votre cluster, vous devez créer les `host_vars` fichiers pour chacun des nouveaux nœuds de fichiers et nœuds de blocs (baies E-Series). Les noms de ces hôtes doivent être ajoutés à l'inventaire, ainsi que les nouvelles ressources à créer. Les `group_vars` fichiers correspondants devront être créés pour chaque nouvelle ressource. Voir la "utilisez des architectures personnalisées" section pour plus de détails.

Une fois les fichiers corrects créés, il suffit de relancer l'automatisation à l'aide de la commande :

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

Retrait d'un module du cluster

Il y a plusieurs considérations à garder à l'esprit lorsque vous devez retirer un élément de construction, par exemple :

- Quels sont les services BeeGFS exécutés dans cet élément de base ?
- Les nœuds de fichiers ne sont-ils que ceux qui sont mis hors service et ceux qui doivent être associés à de nouveaux nœuds de fichiers ?
- Si l'ensemble du bloc de construction est retiré, les données doivent-elles être déplacées vers un nouveau bloc de construction, dispersées vers les nœuds existants du cluster ou déplacées vers un nouveau système de fichiers BeeGFS ou un autre système de stockage ?
- Cela peut-il avoir lieu en cas de panne ou doit-il être effectué sans interruption ?
- L'élément de base est-il activement utilisé ou contient-il principalement des données qui ne sont plus actives ?

Étant donné la diversité des points de départ et des États de terminaison, veuillez contacter le support NetApp afin que nous puissions identifier et vous aider à mettre en œuvre la meilleure stratégie en fonction de votre environnement et de vos besoins.

Résoudre les problèmes

Dépannage d'un cluster BeeGFS HA

Présentation

Dans cette section, vous apprendrez à rechercher et à dépanner diverses défaillances et d'autres scénarios possibles liés à l'utilisation d'un cluster BeeGFS HA.

Guides de dépannage

Étude des basculements inattendus

Lorsqu'un nœud est fermé de façon inattendue et que ses services sont déplacés vers un autre nœud, la première étape doit s'assurer que le cluster indique des défaillances de ressource en bas du `pcs status`. En général, rien ne sera présent si l'escrime s'est terminé avec succès et que les ressources ont été redémarrées sur un autre noeud.

Généralement, l'étape suivante consiste à rechercher dans les journaux système à l'aide de `journalctl` Sur l'un des nœuds de fichiers restants (les journaux Pacemaker sont synchronisés sur tous les nœuds). Si vous connaissez l'heure de l'échec, vous pouvez lancer la recherche juste avant l'échec (généralement au moins dix minutes avant l'apparition de l'échec est recommandée) :

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>"
```

Les sections suivantes montrent un texte commun que vous pouvez gresser dans les journaux pour affiner davantage l'enquête.

Étapes à suivre pour rechercher/résoudre

Étape 1 : vérifier si le moniteur BeeGFS a détecté une défaillance :

Si le basculement a été déclenché par le moniteur BeeGFS, une erreur s'affiche (si ce n'est pas le cas, passez à l'étape suivante).

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i unexpected
[...]
Jul 01 15:51:03 beegfs_01 pacemaker-schedulerd[9246]:  warning: Unexpected
result (error: BeeGFS service is not active!) was recorded for monitor of
meta_08-monitor on beegfs_02 at Jul  1 15:51:03 2022
```

Dans cet exemple, BeeGFS service META_08 s'est arrêté pour une raison quelconque. Pour poursuivre le dépannage, nous devons démarrer beegfs_02 et consulter les journaux du service à l'adresse `/var/log/beegfs-meta-meta_08_tgt_0801.log`. Par exemple, le service BeeGFS peut avoir rencontré une erreur d'application en raison d'un problème interne ou d'un problème sur le nœud.

Contrairement aux logs de Pacemaker, les logs des services BeeGFS ne sont pas distribués à tous les nœuds du cluster. Pour examiner ces types de défaillances, les journaux du nœud d'origine où la défaillance est requise.

Les problèmes possibles pouvant être signalés par le moniteur sont les suivants :

- Les cibles ne sont pas accessibles !
  - Description : indique que les volumes de bloc n'ont pas été accessibles.
  - Dépannage :
    - Si le service n'a pas non plus démarré sur le nœud de fichier secondaire, confirmez que le nœud de bloc fonctionne correctement.
    - Vérifiez si des problèmes physiques empêchent l'accès aux nœuds de blocs depuis ce nœud de fichiers, par exemple des adaptateurs ou des câbles InfiniBand défectueux.
- Le réseau est inaccessible !
  - Description : aucun des adaptateurs utilisés par les clients pour se connecter à ce service BeeGFS n'était en ligne.
  - Dépannage :
    - Si plusieurs ou tous les nœuds de fichiers sont affectés, vérifiez s'il y a une défaillance sur le réseau utilisée pour connecter les clients BeeGFS et le système de fichiers.
    - Recherchez les problèmes physiques susceptibles d'empêcher l'accès des clients à partir de ce nœud de fichiers, par exemple des adaptateurs ou des câbles InfiniBand défectueux.
- Le service BeeGFS n'est pas actif!
  - Description : un service BeeGFS s'est arrêté de façon inattendue.
  - Dépannage :
    - Sur le nœud de fichier qui signale l'erreur, vérifiez les journaux du service eGFS impacté pour voir s'il signale une panne. Dans ce cas, ouvrez un dossier auprès du support NetApp afin que le problème puisse être examiné.
    - Si aucune erreur n'est signalée dans le journal BeeGFS, vérifiez les journaux du journal pour voir si systemd a enregistré une raison pour laquelle le service a été arrêté. Dans certains scénarios, le service BeeGFS n'a peut-être pas été donné la possibilité de consigner tous les messages avant la fin du processus (par exemple si quelqu'un a exécuté `kill -9 <PID>`).

Étape 2 : vérifiez si le nœud a quitté le cluster de manière inattendue

Si le nœud a subi une défaillance matérielle catastrophique (par exemple, la carte système est morte) ou s'il y avait une panique du noyau ou un problème logiciel similaire, le moniteur BeeGFS ne signale pas d'erreur. Au lieu de cela, recherchez le nom d'hôte et les messages de Pacemaker indiquant que le nœud a été perdu de façon inattendue :

```
journalctl --since "<YYYY-MM-DD HH:MM:SS>" | grep -i <HOSTNAME>
[...]
Jul 01 16:18:01 beegfs_01 pacemaker-attrd[9245]:  notice: Node beegfs_02
state is now lost
Jul 01 16:18:01 beegfs_01 pacemaker-controld[9247]:  warning:
Stonith/shutdown of node beegfs_02 was not expected
```

Étape 3 : vérifier que Pacemaker a pu verrouiller le nœud

Dans tous les scénarios, Pacemaker tente de limiter le nœud pour vérifier qu'il est réellement hors ligne (les messages exacts peuvent varier en fonction de la cause de l'escrime) :

```
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Cluster
node beegfs_02 will be fenced: peer is no longer part of the cluster
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Node
beegfs_02 is unclean
Jul 01 16:18:02 beegfs_01 pacemaker-schedulerd[9246]:  warning: Scheduling
Node beegfs_02 for STONITH
```

Si l'action de clôture s'effectue correctement, des messages comme :

```
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
[2214070] (call 27 from pacemaker-controld.9247) for host 'beegfs_02' with
device 'fence_redfish_2' returned: 0 (OK)
Jul 01 16:18:14 beegfs_01 pacemaker-fenced[9243]:  notice: Operation 'off'
targeting beegfs_02 on beegfs_01 for pacemaker-
controld.9247@beegfs_01.786df3a1: OK
Jul 01 16:18:14 beegfs_01 pacemaker-controld[9247]:  notice: Peer
beegfs_02 was terminated (off) by beegfs_01 on behalf of pacemaker-
controld.9247: OK
```

Si l'action d'escrime a échoué pour une raison quelconque, les services BeeGFS ne pourront pas redémarrer sur un autre nœud pour éviter la corruption des données. Ce serait un problème à étudier séparément si, par exemple, le dispositif d'escrime (PDU ou BMC) était inaccessible ou mal configuré.

Echec des actions de ressource de l'adresse (en bas de l'état pcs)

Si une ressource requise pour exécuter un service BeeGFS échoue, un basculement est déclenché par le moniteur BeeGFS. Si cela se produit, il est probable qu'aucune « action de ressource ayant échoué » ne soit répertoriée `pcs status` en bas de et vous devez vous reporter aux étapes à suivre "retour arrière après un basculement non planifié"pour savoir comment .

Dans le cas contraire, il ne devrait y avoir que deux scénarios où vous verrez des « actions de ressource échouées ».

Étapes à suivre pour rechercher/résoudre

Scénario 1 : un problème temporaire ou permanent a été détecté avec un agent d'escrime et il a été redémarré ou déplacé vers un autre nœud.

Certains agents d'escrime sont plus fiables que d'autres et chacun mettra en œuvre sa propre méthode de surveillance pour s'assurer que le dispositif d'escrime est prêt. En particulier, l'agent d'escrime de Redfish a été vu pour signaler des actions de ressources échouées comme les suivantes, même s'il se présente toujours commencé :

```
  * fence_redfish_2_monitor_60000 on beegfs_01 'not running' (7):
call=2248, status='complete', exitreason='', last-rc-change='2022-07-26
08:12:59 -05:00', queued=0ms, exec=0ms
```

Un agent d'escrime signalant l'échec des actions de ressources sur un nœud particulier ne devrait pas déclencher un basculement des services BeeGFS s'exécutant sur ce nœud. Il devrait simplement être redémarré automatiquement sur le même nœud ou sur un autre nœud.

Étapes à suivre pour résoudre :

1. Si l'agent d'escrime refuse systématiquement de s'exécuter sur tout ou sous-ensemble de nœuds, vérifiez si ces nœuds peuvent se connecter à l'agent d'escrime et vérifiez que l'agent d'escrime est configuré correctement dans l'inventaire Ansible.
   a. Par exemple, si un agent d'escrime Redfish (BMC) s'exécute sur le même nœud qu'il est responsable de l'escrime, et que la gestion du système d'exploitation et les adresses IP BMC sont sur la même interface physique, certaines configurations de commutateurs réseau ne permettent pas la communication entre les deux interfaces (pour éviter les boucles réseau). Par défaut, le cluster HA tente d'éviter de placer des agents d'escrime sur le nœud qu'ils sont responsables de l'escrime, mais cela peut se produire dans certains scénarios/configurations.
2. Une fois tous les problèmes résolus (ou si le problème semble éphémère), exécutez `pcs resource cleanup` pour réinitialiser les actions de ressources ayant échoué.

Scénario 2 : le moniteur BeeGFS a détecté un problème et déclenché un basculement, mais pour une raison quelconque, les ressources ne peuvent pas démarrer sur un nœud secondaire.

Si l'escrime est activé et que la ressource n'a pas été bloquée pour s'arrêter sur le nœud d'origine (voir la section de dépannage pour « attente (en cas d'échec) »), les raisons les plus probables incluent des problèmes de démarrage de la ressource sur un nœud secondaire car :

- Le nœud secondaire était déjà hors ligne.
- Un problème de configuration physique ou logique a empêché le système secondaire d'accéder aux volumes de bloc utilisés comme cibles BeeGFS.

Étapes à suivre pour résoudre :

1. Pour chaque entrée des actions de ressources ayant échoué :
   a. Confirmez que l'action de ressource échouée était une opération de démarrage.
   b. En fonction de la ressource indiquée et du nœud spécifié dans les actions de ressources ayant échoué :
      i. Recherchez et corrigez tout problème externe qui empêche le nœud de démarrer la ressource

spécifiée. Par exemple, si l'adresse IP BeeGFS (IP flottante) n'a pas démarré, vérifiez qu'au moins une des interfaces requises est connectée/en ligne et câblée au commutateur réseau approprié. Si une cible BeeGFS (périphérique de bloc/volume E-Series) est défectueuse, vérifiez que les connexions physiques vers le(s) nœud(s) du bloc principal sont connectées comme prévu, et vérifiez que les nœuds du bloc sont en bon état.

c. Si aucun problème externe n'est évident et que vous souhaitez en savoir plus sur la cause première, nous vous recommandons d'ouvrir un dossier auprès des services de support de NetApp avant de poursuivre, car les étapes suivantes peuvent compliquer ou empêcher l'analyse des causes profondes (RCA).

2. Après la résolution de tout problème externe :

   a. Commentez tous les nœuds non fonctionnels à partir du fichier Ansible Inventory.yml et exécutez à nouveau le PlayBook Ansible complet pour vous assurer que toute la configuration logique est correctement configurée sur le ou les nœuds secondaires.

      i. Remarque : n'oubliez pas d'annuler la commentaire de ces nœuds et d'exécuter à nouveau le manuel de vente une fois les nœuds sains et vous êtes prêt à revenir en arrière.

   b. Vous pouvez également tenter de restaurer manuellement le cluster :

      i. Remettre en ligne tous les nœuds en utilisant : `pcs cluster start <HOSTNAME>`

      ii. Effacer toutes les actions de ressources ayant échoué à l'aide de : `pcs resource cleanup`

      iii. Exécutez l'état pcs et vérifiez que tous les services commencent comme prévu.

      iv. Si nécessaire, exécutez `pcs resource relocate run` pour renvoyer les ressources vers le nœud de votre choix (s'il est disponible).

Problèmes courants

Les services BeeGFS ne sont pas de basculement ou de retour arrière sur demande

Question probable: le `pcs resource relocate` la commande d'exécution a été exécutée mais n'a jamais réussi.

Comment vérifier : Exécuter `pcs constraint --full` Et recherchez les contraintes d'emplacement avec un ID de `pcs-relocate-<RESOURCE>`.

Comment résoudre : Exécuter `pcs resource relocate clear` puis repassage `pcs constraint --full` pour vérifier que les contraintes supplémentaires sont supprimées.

Un nœud dans l'état pcs affiche "attente (on-fail)" lorsque l'escrime est désactivé

Problème probable : Pacemaker n'a pas pu confirmer avec succès que toutes les ressources ont été arrêtées sur le nœud qui a échoué.

Comment résoudre:

1. Courez `pcs status` enfin, recherchez les ressources qui ne sont pas « démarrées » et affichez les erreurs en bas de la page et résolvez les problèmes.
2. Pour rétablir l'exécution en ligne du nœud `pcs resource cleanup --node=<HOSTNAME>`.

Après un basculement inattendu, les ressources indiquent « Started (on-fail) » (démarré (on-fail)) dans l'état pcs (pcs) lorsque l'escrime est activé

Problème probable : Un problème s'est produit qui a déclenché un basculement, mais Pacemaker n'a pas pu vérifier que le nœud était clôturé. Cela pourrait se produire parce que l'escrime était mal configuré ou qu'il y avait un problème avec l'agent d'escrime (par exemple : l'unité de distribution d'alimentation était déconnectée du réseau).

Comment résoudre:

1. Vérifiez que le nœud est réellement hors tension.

   Si le nœud que vous spécifiez n'est pas réellement arrêté, mais que vous exécutez les services ou les ressources du cluster, une corruption des données ou une défaillance du cluster se produit.

2. Confirmer manuellement l'escrime avec : `pcs stonith confirm <NODE>`

À ce stade, les services devraient finir le basculement et être redémarrés sur un autre noeud en bon état.

Tâches courantes de dépannage

Redémarrez chaque service BeeGFS

Normalement, si un service BeeGFS doit être redémarré (par exemple pour faciliter une modification de la configuration), il doit être fait en mettant à jour l'inventaire Ansible et en exécutant de nouveau le manuel de vente. Dans certains cas, il peut être souhaitable de redémarrer des services individuels pour accélérer le dépannage, par exemple pour modifier le niveau de journalisation sans avoir à attendre l'exécution du manuel de vente dans son intégralité.

Sauf si des modifications manuelles sont également ajoutées à l'inventaire Ansible, elles seront rétablies au prochain exécution du PlayBook Ansible.

Option 1 : redémarrage contrôlé par le système

S'il y a un risque que le service BeeGFS ne redémarre pas correctement avec la nouvelle configuration, tout d'abord placer le cluster en mode maintenance pour empêcher le moniteur BeeGFS de détecter le service est arrêté et déclencher un basculement non souhaité :

```
pcs property set maintenance-mode=true
```

Si nécessaire, modifiez la configuration des services à l'adresse `/mnt/<SERVICE_ID>/_config/beegfs-.conf` (exemple : `/mnt/meta_01_tgt_0101/metadata_config/beegfs-meta.conf`) puis utilisez systemd pour le redémarrer :

```
systemctl restart beegfs-*@<SERVICE_ID>.service
```

Exemple : `systemctl restart beegfs-meta@meta_01_tgt_0101.service`

Option 2 : redémarrage contrôlé par le stimulateur cardiaque

Si vous n'êtes pas préoccupé par la nouvelle configuration, le service peut s'arrêter de façon inattendue (par exemple, en modifiant simplement le niveau de journalisation), ou vous êtes dans une fenêtre de maintenance et ne vous préoccupez pas des temps d'arrêt, il vous suffit de redémarrer le moniteur BeeGFS pour le service que vous voulez redémarrer :

```
pcs resource restart <SERVICE>-monitor
```

Par exemple, pour redémarrer le service de gestion BeeGFS : `pcs resource restart mgmt-monitor`

Mentions légales

Les mentions légales donnent accès aux déclarations de copyright, aux marques, aux brevets, etc.

Droits d’auteur

"https://www.netapp.com/company/legal/copyright/"

Marques déposées

NetApp, le logo NETAPP et les marques mentionnées sur la page des marques commerciales NetApp sont des marques commerciales de NetApp, Inc. Les autres noms de sociétés et de produits peuvent être des marques commerciales de leurs propriétaires respectifs.

"https://www.netapp.com/company/legal/trademarks/"

Brevets

Vous trouverez une liste actuelle des brevets appartenant à NetApp à l’adresse suivante :

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Politique de confidentialité

"https://www.netapp.com/company/legal/privacy-policy/"

Source ouverte

Les fichiers de notification fournissent des informations sur les droits d’auteur et les licences de tiers utilisés dans le logiciel NetApp.

"Remarque : pour les systèmes d’exploitation SANtricity E-Series/EF-Series"

Informations sur le copyright

Informations sur les marques commerciales