Déployez le système de fichiers BeeGFS

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Sommaire

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.

Informations sur le copyright

Informations sur les marques commerciales