Implementare il file system BeeGFS

BeeGFS on NetApp with E-Series Storage

NetApp
June 18, 2024

Sommario

Implementare il file system BeeGFS

Panoramica di Ansible Playbook

Implementazione e gestione di cluster BeeGFS ha con Ansible.

Panoramica

Nelle sezioni precedenti sono illustrate le fasi necessarie per creare un inventario Ansible che rappresenti un cluster BeeGFS ha. Questa sezione presenta l'automazione Ansible sviluppata da NetApp per implementare e gestire il cluster.

Ansible: Concetti chiave

Prima di procedere, è utile acquisire familiarità con alcuni concetti chiave di Ansible:

- Le attività da eseguire su un inventario Ansible sono definite in ciò che è noto come **playbook**.
  - La maggior parte delle attività di Ansible è progettata per essere **idempotent**, il che significa che possono essere eseguite più volte per verificare che la configurazione/stato desiderato sia ancora applicato senza interrompere le operazioni o eseguire aggiornamenti non necessari.
- La più piccola unità di esecuzione di Ansible è un modulo *.
  - I playbook tipici utilizzano più moduli.
    - Esempi: Scaricare un pacchetto, aggiornare un file di configurazione, avviare/abilitare un servizio.
  - NetApp distribuisce i moduli per automatizzare i sistemi NetApp e-Series.
- L'automazione complessa è meglio integrata come ruolo.
  - Essenzialmente un formato standard per la distribuzione di un playbook riutilizzabile.
  - NetApp distribuisce i ruoli per host Linux e file system BeeGFS.

Ruolo BeeGFS ha per Ansible: Concetti chiave

Tutta l'automazione necessaria per implementare e gestire ogni versione di BeeGFS su NetApp viene fornita come ruolo Ansible e distribuita come parte di "NetApp e-Series Ansible Collection per BeeGFS":

- Questo ruolo può essere considerato tra un **installer** e un moderno motore di **implementazione/gestione** per BeeGFS.
  - Applica l'infrastruttura moderna come pratiche di codice e filosofie per semplificare la gestione dell'infrastruttura di storage su qualsiasi scala.
  - Simile a come "Kubespray" Project consente agli utenti di implementare/gestire un'intera distribuzione Kubernetes per un'infrastruttura di calcolo scale-out.
- Questo ruolo è il formato * software-defined* utilizzato da NetApp per il packaging, la distribuzione e la manutenzione di BeeGFS su soluzioni NetApp.
  - Cerca di creare un'esperienza simile a quella di un'appliance senza dover distribuire un'intera distribuzione Linux o un'immagine di grandi dimensioni.
  - Include agenti di risorse cluster compatibili con Open Cluster Framework (OCF) creati da NetApp per destinazioni BeeGFS personalizzate, indirizzi IP e monitoraggio che forniscono un'integrazione intelligente di Pacemaker/BeeGFS.

- Questo ruolo non è semplicemente "automazione" dell'implementazione ed è destinato a gestire l'intero ciclo di vita del file system, tra cui:
  - Applicazione di modifiche e aggiornamenti della configurazione per servizio o a livello di cluster.
  - Automazione della riparazione e del ripristino del cluster dopo la risoluzione dei problemi hardware.
  - Semplificazione dell'ottimizzazione delle performance con valori predefiniti impostati in base a test approfonditi con volumi BeeGFS e NetApp.
  - Verifica e correzione della deriva della configurazione.

NetApp offre anche un ruolo Ansible per "Client BeeGFS", Che può essere utilizzato facoltativamente per installare BeeGFS e montare file system su nodi di calcolo/GPU/login.

Implementare il cluster BeeGFS ha

Specificare le attività da eseguire per implementare il cluster BeeGFS ha utilizzando un manuale.

Panoramica

Questa sezione descrive come assemblare il manuale standard utilizzato per implementare/gestire BeeGFS su NetApp.

Fasi

Creare il manuale Ansible Playbook

Creare il file `playbook.yml` e compilarlo come segue:

1. Per prima cosa, definire una serie di attività (comunemente denominate a. "gioca") Che dovrebbe essere eseguito solo sui nodi a blocchi NetApp e-Series. Viene utilizzata un'attività di pausa per richiedere conferma prima di eseguire l'installazione (per evitare l'esecuzione accidentale di un playbook), quindi importare `nar_santricity_management` ruolo. Questo ruolo gestisce l'applicazione di qualsiasi configurazione generale del sistema definita in `group_vars/eseries_storage_systems.yml` o individuale `host_vars/<BLOCK NODE>.yml` file.

```
- 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. Definire il gioco che verrà eseguito su tutti i nodi di file e blocchi:

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

3. All'interno di questo gioco è possibile definire facoltativamente un set di "pre-task" che devono essere eseguiti prima di implementare il cluster ha. Questo può essere utile per verificare/installare qualsiasi prerequisito come Python. Possiamo anche effettuare controlli prima del volo, ad esempio verificando che i tag Ansible forniti siano supportati:

```
  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. Infine, questo gioco importa il ruolo BeeGFS ha per la versione di BeeGFS che si desidera implementare:

```
  tasks:
    - name: Verify the BeeGFS HA cluster is properly deployed.
      import_role:
        name: beegfs_ha_7_3 # Alternatively specify: beegfs_ha_7_2.
```

Viene mantenuto un ruolo BeeGFS ha per ciascuna versione principale.minore supportata di BeeGFS. Questo consente agli utenti di scegliere quando aggiornare le versioni principali/secondarie. Attualmente BeeGFS 7.3.x (`beegfs_7_3`) O BeeGFS 7.2.x (`beegfs_7_2`) sono supportati. Per impostazione predefinita, entrambi i ruoli implementeranno la versione più recente delle patch BeeGFS al momento del rilascio, anche se gli utenti possono scegliere di eseguire l'override e distribuire la patch più recente, se lo desiderano. Fare riferimento alla versione più recente "guida all'upgrade" per ulteriori dettagli.

5. Facoltativo: Se si desidera definire attività aggiuntive, tenere presente se le attività devono essere indirizzate a. `all` Host (inclusi i sistemi storage e-Series) o solo i nodi di file. Se necessario, definire un nuovo gioco specifico per i nodi di file utilizzando `- hosts: ha_cluster`.

Fare clic su "qui" per un esempio di un file di playbook completo.

Installare NetApp Ansible Collections

L'insieme BeeGFS per Ansible e tutte le dipendenze vengono mantenute su "Ansible Galaxy". Sul nodo di controllo Ansible eseguire il seguente comando per installare la versione più recente:

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

Sebbene non sia generalmente consigliato, è anche possibile installare una versione specifica della raccolta:

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

Eseguire il Playbook

Dalla directory del nodo di controllo Ansible contenente `inventory.yml` e. `playbook.yml` eseguire il playbook come segue:

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

In base alle dimensioni del cluster, l'implementazione iniziale può richiedere oltre 20 minuti. Se l'implementazione non riesce per qualsiasi motivo, correggere eventuali problemi (ad esempio, cablaggio errato, nodo non avviato, ecc.) e riavviare il playbook Ansible.

Quando si specifica "configurazione di un nodo di file comune", Se si sceglie l'opzione predefinita per fare in modo che Ansible gestisca automaticamente l'autenticazione basata sulla connessione, un `connAuthFile` utilizzato come segreto condiviso è ora disponibile all'indirizzo `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (per impostazione predefinita). Tutti i client che hanno bisogno di accedere al file system dovranno utilizzare questo segreto condiviso. Questo viene gestito automaticamente se i client vengono configurati utilizzando "Ruolo del client BeeGFS".

Implementare i client BeeGFS

In alternativa, è possibile utilizzare Ansible per configurare i client BeeGFS e montare il file system.

Panoramica

L'accesso ai file system BeeGFS richiede l'installazione e la configurazione del client BeeGFS su ciascun nodo che deve montare il file system. In questa sezione viene descritto come eseguire queste attività utilizzando la disponibile "Ruolo Ansible".

Fasi

Creare il file di inventario del client

1. Se necessario, impostare SSH senza password dal nodo di controllo Ansible a ciascuno degli host che si desidera configurare come client BeeGFS:

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

2. Sotto `host_vars/`, Creare un file per ogni client BeeGFS denominato `<HOSTNAME>.yml` con il seguente contenuto, inserendo il testo segnaposto con le informazioni corrette per il tuo ambiente:

```
# BeeGFS Client
ansible_host: <MANAGEMENT_IP>
```

3. Se si desidera utilizzare i ruoli di NetApp e-Series host Collection per configurare le interfacce InfiniBand o Ethernet per consentire ai client di connettersi ai nodi di file BeeGFS, è possibile includere uno dei seguenti elementi:
   a. Se il tipo di rete è "InfiniBand (con 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. Se il tipo di rete è "RDMA su Ethernet convergente (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. Se il tipo di rete è "Ethernet (solo TCP, senza 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. Creare un nuovo file `client_inventory.yml` E specificare l'utente che Ansible deve utilizzare per connettersi a ciascun client e la password che Ansible deve utilizzare per l'escalation dei privilegi (ciò richiede `ansible_ssh_user` essere root o avere privilegi sudo):

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

Non memorizzare le password in testo normale. Utilizzare invece il vault Ansible (vedere la "Documentazione Ansible" Per crittografare il contenuto con Ansible Vault) o utilizzare `--ask-become-pass` quando si esegue il playbook.

5. In `client_inventory.yml` File, elenca tutti gli host che devono essere configurati come client BeeGFS in `beegfs_clients` Fare riferimento ai commenti inline e rimuovere eventuali commenti aggiuntivi necessari per creare il modulo del kernel del client BeeGFS sul sistema:

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

      vars:
        # OPTION 1: If you're using the Mellanox 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.
```

Quando si utilizzano i driver Mellanox OFED, assicurarsi che beegfs_client_ofed_include_path punti al "header include path" corretto per l'installazione di Linux. Per ulteriori informazioni, consultare la documentazione di BeeGFS per "Supporto RDMA".

6. In `client_inventory.yml` Elencare i file system BeeGFS che si desidera montare sotto qualsiasi file definito in precedenza `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. A partire da BeeGFS 7.2.7 e 7.3.1 "autenticazione della connessione" deve essere configurato o disabilitato esplicitamente. A seconda di come si sceglie di configurare l'autenticazione basata sulla connessione quando si specifica "configurazione di un nodo di file comune", potrebbe essere necessario modificare la configurazione del client:
   a. Per impostazione predefinita, l'implementazione del cluster ha configurerà automaticamente l'autenticazione della connessione e genererà un `connauthfile` Che verranno posizionati/mantenuti sul nodo di controllo Ansible in `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. Per impostazione predefinita, il ruolo del client BeeGFS è impostato per leggere/distribuire questo file ai client definiti in `client_inventory.yml`e non sono necessarie ulteriori azioni.
      i. Per le opzioni avanzate, fare riferimento all'elenco completo dei valori predefiniti inclusi in "Ruolo del client BeeGFS".
   b. Se si sceglie di specificare un segreto personalizzato con `beegfs_ha_conn_auth_secret` specificarlo in `client_inventory.yml` anche file:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Se si sceglie di disattivare completamente l'autenticazione basata sulla connessione con `beegfs_ha_conn_auth_enabled`, specificare che in `client_inventory.yml` anche file:

```
beegfs_ha_conn_auth_enabled: false
```

Per un elenco completo dei parametri supportati e ulteriori dettagli, fare riferimento a. "Documentazione completa del client BeeGFS". Per un esempio completo di un inventario client, fare clic su "qui".

Creare il file Playbook del client BeeGFS

1. Creare un nuovo file `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. Facoltativo: Se si desidera utilizzare i ruoli di NetApp e-Series host Collection per configurare le interfacce per la connessione dei client ai file system BeeGFS, importare il ruolo corrispondente al tipo di interfaccia che si sta configurando:
   a. Se si utilizza InfiniBand (IPoIB):

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

   b. Se si utilizza RDMA su Ethernet convergente (RoCE):

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

   c. Se si utilizza Ethernet (solo TCP, senza RDMA):

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

3. Infine, importare il ruolo del client BeeGFS per installare il software client e configurare i supporti del file system:

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

Per un esempio completo di un playbook client, fai clic "qui".

Eseguire il manuale BeeGFS Client Playbook

Per installare/creare il client e montare BeeGFS, eseguire il seguente comando:

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

Verificare l'implementazione di BeeGFS

Verificare l'implementazione del file system prima di mettere il sistema in produzione.

Panoramica

Prima di mettere il file system BeeGFS in produzione, eseguire alcuni controlli di verifica.

Fasi

1. Accedere a qualsiasi client ed eseguire quanto segue per assicurarsi che tutti i nodi previsti siano presenti/raggiungibili e che non siano segnalate incoerenze o altri problemi:

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

2. Arrestare l'intero cluster, quindi riavviarlo. Da qualsiasi nodo di file eseguire quanto segue:

```
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. Mettere ciascun nodo in standby e verificare che i servizi BeeGFS siano in grado di eseguire il failover su nodi secondari. Per eseguire questa operazione, accedere a uno dei nodi di file ed eseguire quanto segue:

```
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. Utilizza strumenti di benchmarking delle performance come IOR e MDTest per verificare che le performance del file system soddisfino le aspettative. Esempi di test e parametri comuni utilizzati con BeeGFS sono disponibili nella "Verifica del progetto" Sezione di BeeGFS su NetApp Verified Architecture.

È necessario eseguire test aggiuntivi in base ai criteri di accettazione definiti per un sito/installazione particolare.

Informazioni sul copyright

Informazioni sul marchio commerciale