Definire il file system BeeGFS

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Sommario

Definire il file system BeeGFS

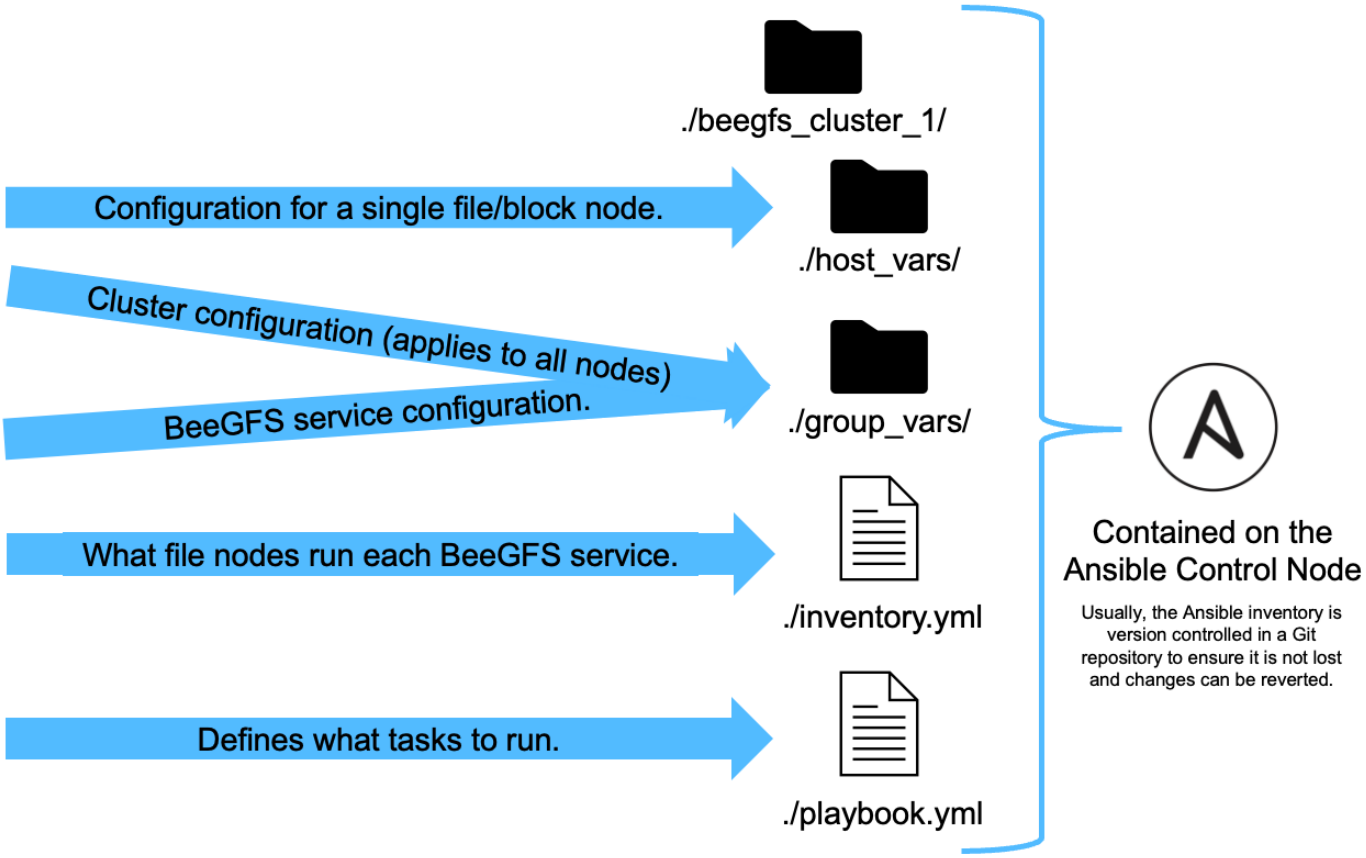
Panoramica di Ansible Inventory

L'inventario Ansible è un insieme di file di configurazione che definiscono il cluster BeeGFS ha desiderato.

Panoramica

Si consiglia di seguire le procedure standard di Ansible per organizzare il "inventario", incluso l'utilizzo di "sottodirectory/file" invece di memorizzare l'intero inventario in un file.

L'inventario Ansible per un singolo cluster BeeGFS ha è organizzato come segue:



Poiché un singolo file system BeeGFS può comprendere più cluster ha, è possibile che installazioni di grandi dimensioni dispongano di più inventari Ansible. In genere, non è consigliabile cercare di definire più cluster ha come un singolo inventario Ansible per evitare problemi.

Fasi

1. Nel nodo di controllo Ansible creare una directory vuota contenente l'inventario Ansible per il cluster BeeGFS che si desidera implementare.
   a. Se il file system conterrà più cluster ha, si consiglia di creare prima una directory per il file system, quindi sottodirectory per l'inventario che rappresenta ciascun cluster ha. Ad esempio:

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

2. Nella directory contenente l'inventario per il cluster ha che si desidera implementare, creare due directory `group_vars` e. `host_vars` e due file `inventory.yml` e. `playbook.yml`.

Nelle sezioni seguenti viene illustrata la definizione del contenuto di ciascuno di questi file.

Pianificare il file system

Pianificare l'implementazione del file system prima di creare l'inventario Ansible.

Panoramica

Prima di implementare il file system, è necessario definire gli indirizzi IP, le porte e le altre configurazioni richieste da tutti i nodi di file, i nodi di blocco e i servizi BeeGFS in esecuzione nel cluster. Anche se la configurazione esatta varia in base all'architettura del cluster, questa sezione definisce le Best practice e i passaggi da seguire che sono generalmente applicabili.

Fasi

1. Se si utilizza un protocollo di storage basato su IP (come iSER, iSCSI, NVMe/IB o NVMe/RoCE) per connettere i nodi di file ai nodi di blocco, compilare il seguente foglio di lavoro per ciascun building block. Ogni connessione diretta in un singolo building block deve avere una subnet univoca e non deve esserci alcuna sovrapposizione con le subnet utilizzate per la connettività client-server.

Nodo del file	Porta IB	Indirizzo IP	Nodo del blocco	Porta IB	IP fisico	Virtual IP (solo per EF600 con HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

> ⓘ Se i nodi di file e blocchi di ciascun building block sono collegati direttamente, spesso è possibile riutilizzare gli stessi IP/schema per più building block.

2. Indipendentemente dall'utilizzo di InfiniBand o RDMA su RoCE (Converged Ethernet) per la rete di storage, compilare il seguente foglio di lavoro per determinare gli intervalli IP che verranno utilizzati per i servizi cluster ha, i file service BeeGFS e i client per comunicare:

Scopo	**Porta InfiniBand**	**Indirizzo IP o intervallo**
IP cluster BeeGFS	<INTERFACE(s)>	<RANGE>
Gestione di BeeGFS	<INTERFACE(s)>	<IP(s)>
Metadati BeeGFS	<INTERFACE(s)>	<RANGE>

Scopo	Porta InfiniBand	Indirizzo IP o intervallo
Storage BeeGFS	<INTERFACE(s)>	<RANGE>
Client BeeGFS	<INTERFACE(s)>	<RANGE>

a. Se si utilizza una singola subnet IP, è necessario un solo foglio di lavoro, altrimenti compilare un foglio di lavoro per la seconda subnet.

3. In base a quanto sopra, per ciascun building block del cluster, compilare il seguente foglio di lavoro che definisce i servizi BeeGFS che verranno eseguiti. Per ogni servizio, specificare i nodi di file preferiti/secondari, la porta di rete, gli IP mobili, l'assegnazione di zona NUMA (se necessario) e i nodi di blocco da utilizzare per le destinazioni. Per compilare il foglio di lavoro, fare riferimento alle seguenti linee guida:

   a. Specificare i servizi BeeGFS come uno dei due `mgmt.yml`, `meta_<ID>.yml`, o. `storage_<ID>.yml` Dove ID rappresenta un numero univoco per tutti i servizi BeeGFS di quel tipo in questo file system. Questa convenzione semplifica il riferimento a questo foglio di lavoro nelle sezioni successive durante la creazione di file per configurare ciascun servizio.

   b. Le porte per i servizi BeeGFS devono essere univoche solo in un particolare building block. Assicurarsi che i servizi con lo stesso numero di porta non possano mai essere eseguiti sullo stesso nodo di file per evitare conflitti di porta.

   c. Se necessario, i servizi possono utilizzare volumi da più di un nodo a blocchi e/o pool di storage (e non tutti i volumi devono essere di proprietà dello stesso controller). Più servizi possono anche condividere lo stesso nodo a blocchi e/o la stessa configurazione del pool di storage (i singoli volumi verranno definiti in una sezione successiva).

Servizio BeeGFS (nome file)	Nodi di file	Porta	IP mobili	Zona NUMA	Nodo del blocco	Pool di storage	Controller proprietario
<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>

Per ulteriori informazioni sulle convenzioni standard, sulle Best practice e sui fogli di lavoro di esempio compilati"best practice", fare riferimento alle "Definire i blocchi di base BeeGFS"sezioni e di BeeGFS sull'architettura verificata NetApp.

Definire i nodi di file e blocchi

Configurare singoli nodi di file

Specificare la configurazione per i singoli nodi di file utilizzando le variabili host (host_vars).

Panoramica

In questa sezione viene illustrata la compilazione di un `host_vars/<FILE_NODE_HOSTNAME>.yml` per ogni nodo di file nel cluster. Questi file devono contenere solo configurazioni univoche per un particolare nodo di

file. Ciò comprende in genere:

- Definizione dell'IP o del nome host che Ansible deve utilizzare per connettersi al nodo.
- Configurazione di interfacce aggiuntive e IP del cluster utilizzati per i servizi cluster ha (Pacemaker e Corosync) per comunicare con altri nodi di file. Per impostazione predefinita, questi servizi utilizzano la stessa rete dell'interfaccia di gestione, ma devono essere disponibili interfacce aggiuntive per la ridondanza. La pratica comune consiste nel definire IP aggiuntivi sulla rete di storage, evitando la necessità di un cluster aggiuntivo o di una rete di gestione.
  - Le performance di qualsiasi rete utilizzata per la comunicazione in cluster non sono critiche per le performance del file system. Con la configurazione predefinita del cluster in genere, almeno una rete a 1Gb GB/s fornirà prestazioni sufficienti per le operazioni del cluster, come la sincronizzazione degli stati dei nodi e il coordinamento delle modifiche dello stato delle risorse del cluster. Le reti lente/occupate possono richiedere più tempo del solito per le modifiche dello stato delle risorse e, in casi estremi, potrebbero causare l'eliminazione dei nodi dal cluster se non riescono a inviare heartbeat in un intervallo di tempo ragionevole.
- Configurazione delle interfacce utilizzate per la connessione ai nodi a blocchi sul protocollo desiderato (ad esempio: ISCSI/iSER, NVMe/IB, NVMe/RoCE, FCP, ecc.)

Fasi

Facendo riferimento allo schema di indirizzamento IP definito nella "Pianificare il file system"sezione, per ogni nodo file nel cluster creare un file `host_vars/<FILE_NODE_HOSTNAME>/yml` e compilarlo come segue:

1. In alto, specificare l'IP o il nome host che Ansible deve utilizzare per SSH al nodo e gestirlo:

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

2. Configurare IP aggiuntivi che possono essere utilizzati per il traffico del cluster:
   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>
```

3. Indicare gli IP da utilizzare per il traffico del cluster, con gli IP preferiti elencati più in alto:

```
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.
```

> Gli IPS configurati nella fase due non verranno utilizzati come IP del cluster a meno che non siano inclusi in `beegfs_ha_cluster_node_ips` elenco. Ciò consente di configurare IP/interfacce aggiuntive utilizzando Ansible che possono essere utilizzati per altri scopi, se necessario.

4. Se il nodo del file deve comunicare per bloccare i nodi su un protocollo basato su IP, gli IP dovranno essere configurati sull'interfaccia appropriata e tutti i pacchetti richiesti per quel protocollo installati/configurati.
   a. Se in uso "ISCSI":

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

   b. Se in uso "Er":

```
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. Se in uso "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. Se in uso "NVMe/RoCE":

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

e. Altri protocolli:

i. Se in uso "NVMe/FC", la configurazione di singole interfacce non è richiesta. L'implementazione del cluster BeeGFS rileverà automaticamente il protocollo e installerà/configurerà i requisiti in base alle necessità. Se si utilizza un fabric per connettere file e nodi a blocchi, assicurarsi che gli switch siano correttamente suddivisi in zone seguendo le Best practice di NetApp e del vendor di switch.

ii. L'utilizzo di FCP o SAS non richiede l'installazione o la configurazione di software aggiuntivo. Se si utilizza FCP, assicurarsi che gli switch siano correttamente zonati "NetApp" e le best practice del tuo fornitore di switch.

iii. Al momento non si consiglia l'utilizzo di IB SRP. Utilizzare NVMe/IB o iSER a seconda di ciò che i nodi a blocchi e-Series supportano.

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta un singolo nodo di file.

Advanced (Avanzate): Passaggio tra le modalità Ethernet e InfiniBand degli adattatori VPI NVIDIA ConnectX

Gli adattatori NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) supportano sia InfiniBand che Ethernet come layer di trasporto. Il passaggio da una modalità all’altra non viene negoziato automaticamente e deve essere configurato utilizzando <code>mstconfig</code> lo strumento incluso in <code>mstflint</code>, un pacchetto open source che fa parte di "Strumenti per la firma NVIDIA (MFT)". La modifica della modalità degli adattatori deve essere eseguita una sola volta. Questa operazione può essere eseguita manualmente o inclusa nell’inventario Ansible come parte di qualsiasi interfaccia configurata utilizzando la <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> sezione dell’inventario, per farla controllare/applicare automaticamente.

Ad esempio, per modificare una corrente di interfaccia in modalità InfiniBand su Ethernet in modo che possa essere utilizzata per RoCE:

1. Per ogni interfaccia che si desidera configurare, specificare `mstconfig` come un mapping (o dizionario) che specifica `LINK_TYPE_P<N>` dove `<N>` È determinato dal numero di porta dell'HCA per l'interfaccia. Il `<N>` il valore può essere determinato eseguendo `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` E aggiungendo 1 all'ultimo numero dal nome dello slot PCI e convertendo in decimale.

   a. Ad esempio `PCI_SLOT_NAME=0000:2f:00.2` (2 + 1 → porta HCA 3) → `LINK_TYPE_P3: eth`:

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

Per ulteriori informazioni, consultare "Documentazione della raccolta di host NetApp e-Series" per il tipo di interfaccia/protocollo in uso.

Configurare singoli nodi a blocchi

Specificare la configurazione per i singoli nodi di blocco utilizzando le variabili host (host_vars).

Panoramica

In questa sezione viene illustrata la compilazione di un `host_vars/<BLOCK_NODE_HOSTNAME>.yml` file per ciascun nodo del blocco nel cluster. Questi file devono contenere solo una configurazione univoca per un nodo di blocco specifico. Ciò comprende in genere:

- Il nome del sistema (visualizzato in System Manager).
- L'URL HTTPS per uno dei controller (utilizzato per gestire il sistema utilizzando l'API REST).
- Quali nodi di file del protocollo di storage utilizzano per connettersi a questo nodo a blocchi.
- Configurazione delle porte della scheda di interfaccia host (HIC), ad esempio gli indirizzi IP (se necessario).

Fasi

Facendo riferimento allo schema di indirizzamento IP definito nella "Pianificare il file system"sezione, per ogni nodo di blocco nel cluster creare un file `host_vars/<BLOCK_NODE_HOSTNAME>/yml` e compilarlo come segue:

1. Nella parte superiore, specificare il nome del sistema e l'URL HTTPS per uno dei controller:

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

2. Selezionare "protocollo" i nodi di file utilizzeranno per connettersi a questo nodo di blocco:
   a. Protocolli supportati: `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

```
eseries_initiator_protocol: <PROTOCOL>
```

3. A seconda del protocollo in uso, le porte HIC potrebbero richiedere una configurazione aggiuntiva. Se necessario, definire la configurazione della porta HIC in modo che la voce superiore nella configurazione di

ciascun controller corrisponda alla porta fisica più a sinistra su ciascun controller e la porta inferiore alla porta più a destra. Tutte le porte richiedono una configurazione valida anche se non sono attualmente in uso.

> Consultare anche la sezione seguente se si utilizza HDR (200 GB) InfiniBand o 200GB RoCE con nodi a blocchi EF600.

a. Per 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. Per 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. Per 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. Per 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. I protocolli FC e SAS non richiedono ulteriori configurazioni. SRP non è consigliato correttamente.

Per ulteriori opzioni di configurazione delle porte HIC e dei protocolli host, inclusa la possibilità di configurare iSCSI CHAP, fare riferimento a. "documentazione" Incluso nella raccolta SANtricity. Nota durante l'implementazione di BeeGFS, il pool di storage, la configurazione del volume e altri aspetti del provisioning dello storage verranno configurati altrove e non devono essere definiti in questo file.

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta un singolo nodo a blocchi.

Utilizzando HDR (200 GB) InfiniBand o 200GB RoCE con i nodi a blocchi NetApp EF600:

Per utilizzare HDR (200 GB) InfiniBand con EF600, è necessario configurare un secondo IP "virtuale" per ciascuna porta fisica. Di seguito è riportato un esempio del modo corretto di configurare un EF600 dotato di

InfiniBand HDR HIC a doppia porta:

```
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)
```

Specificare la configurazione del nodo file comune

Specificare la configurazione del nodo file comune utilizzando le variabili di gruppo (group_vars).

Panoramica

La configurazione che deve essere utilizzata per tutti i nodi di file è definita in `group_vars/ha_cluster.yml`. In genere include:

- Dettagli su come connettersi e accedere a ciascun nodo di file.
- Configurazione di rete comune.
- Se sono consentiti riavvii automatici.
- Modalità di configurazione degli stati firewall e selinux.
- Configurazione del cluster, inclusi avvisi e scherma.
- Tuning delle performance.
- Configurazione comune del servizio BeeGFS.

> Le opzioni impostate in questo file possono essere definite anche su singoli nodi di file, ad esempio se sono in uso modelli hardware misti o se si dispone di password diverse per ciascun nodo. La configurazione sui singoli nodi di file avrà la precedenza sulla configurazione in questo file.

Fasi

Creare il file `group_vars/ha_cluster.yml` e compilarlo come segue:

1. Indicare come il nodo Ansible Control deve autenticare con gli host remoti:

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

In particolare per gli ambienti di produzione, non memorizzare le password in testo normale. Utilizzare invece Ansible Vault (vedere "Crittografia del contenuto con Ansible Vault") o il `--ask-become-pass` quando si esegue il playbook. Se il `ansible_ssh_user` è già root, quindi è possibile omettere il `ansible_become_password`.

2. Se si configurano IP statici sulle interfacce ethernet o InfiniBand (ad esempio gli IP del cluster) e più interfacce si trovano nella stessa subnet IP (ad esempio se ib0 utilizza 192.168.1.10/24 e ib1 utilizza 192.168.1.11/24), Per il corretto funzionamento del supporto multi-homed, è necessario configurare ulteriori tabelle e regole di routing IP. È sufficiente attivare il gancio di configurazione dell'interfaccia di rete fornito come segue:

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

3. Durante l'implementazione del cluster, a seconda del protocollo di storage potrebbe essere necessario riavviare i nodi per facilitare il rilevamento dei dispositivi a blocchi remoti (volumi e-Series) o applicare altri aspetti della configurazione. Per impostazione predefinita, i nodi richiedono prima del riavvio, ma è possibile consentire il riavvio automatico dei nodi specificando quanto segue:

```
eseries_common_allow_host_reboot: true
```

   a. Per impostazione predefinita, dopo un riavvio, per garantire che i dispositivi a blocchi e gli altri servizi siano pronti, Ansible attenderà fino al sistema `default.target` viene raggiunto prima di continuare con l'implementazione. In alcuni scenari, quando NVMe/IB è in uso, potrebbe non essere abbastanza lungo da inizializzare, rilevare e connettersi a dispositivi remoti. Ciò può causare la continuità prematura dell'implementazione automatica e il malfunzionamento. Per evitare questo problema quando si utilizza NVMe/IB, definire anche quanto segue:

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

4. Per comunicare con i servizi cluster BeeGFS e ha sono necessarie diverse porte firewall. A meno che non si desideri configurare il firwewall manualmente (non consigliato), specificare quanto segue per creare le zone firewall richieste e aprire automaticamente le porte:

```
beegfs_ha_firewall_configure: True
```

5. Al momento SELinux non è supportato e si consiglia di impostare lo stato su Disabled (disattivato) per evitare conflitti (soprattutto quando RDMA è in uso). Impostare quanto segue per assicurarsi che SELinux sia disattivato:

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

6. Configurare l'autenticazione in modo che i file node siano in grado di comunicare, regolando le impostazioni predefinite in base alle policy aziendali:

```
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. In base alla "Pianificare il file system" sezione specificare l'IP di gestione BeeGFS per questo file system:

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

> Anche se apparentemente ridondante, `beegfs_ha_mgmtd_floating_ip` È importante quando si scala il file system BeeGFS oltre un singolo cluster ha. I cluster ha successivi vengono implementati senza un servizio di gestione BeeGFS aggiuntivo e puntano al servizio di gestione fornito dal primo cluster.

8. Attivare gli avvisi e-mail se lo si desidera:

```
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. Si consiglia vivamente di abilitare la scherma, altrimenti i servizi possono essere bloccati per l'avvio sui nodi secondari quando il nodo primario non funziona.
   a. Abilitare la scherma a livello globale specificando quanto segue:

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

i. Nota se necessario, è anche possibile specificare qui tutti i supporti "proprietà del cluster" . La regolazione di questi non è tipicamente necessaria, poiché il ruolo di BeeGFS ha viene fornito con un certo numero di ben testati "valori predefiniti".

b. Selezionare e configurare un agente di scherma:

i. OPZIONE 1: Per abilitare la recinzione utilizzando le PDU (Power Distribution Unit) 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. OPZIONE 2: Per abilitare la scherma utilizzando le API Redfish fornite da Lenovo XCC (e da altri BMC):

```
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. Per ulteriori informazioni sulla configurazione di altri agenti di scherma, fare riferimento alla "Documentazione di Red Hat".

10. Il ruolo BeeGFS ha può applicare diversi parametri di tuning per ottimizzare ulteriormente le performance. Questi includono l'ottimizzazione dell'utilizzo della memoria del kernel e l'i/o dei dispositivi a blocchi, tra gli altri parametri. Il ruolo viene fornito con una serie ragionevole di "valori predefiniti" in base al test con i nodi di blocco NetApp E-Series, ma per impostazione predefinita questi non vengono applicati a meno che non si specifichi:

```
beegfs_ha_enable_performance_tuning: True
```

a. Se necessario, specificare qui eventuali modifiche all'ottimizzazione predefinita delle prestazioni. Per ulteriori informazioni, consultare la documentazione completa "parametri di ottimizzazione delle performance" .

11. Per garantire che gli indirizzi IP mobili (talvolta noti come interfacce logiche) utilizzati per i servizi BeeGFS possano eseguire il failover tra i nodi di file, tutte le interfacce di rete devono essere denominate in modo coerente. Per impostazione predefinita, i nomi delle interfacce di rete vengono generati dal kernel, che non è garantito per generare nomi coerenti, anche su modelli di server identici con adattatori di rete installati negli stessi slot PCIe. Ciò è utile anche quando si creano inventari prima dell'implementazione dell'apparecchiatura e si conoscono i nomi delle interfacce generate. Per garantire nomi di dispositivi coerenti, in base a un diagramma a blocchi del server o. `lshw -class network -businfo` Output, specificare il mapping indirizzo PCIe desiderato per l'interfaccia logica come segue:

a. Per le interfacce di rete InfiniBand (IPoIB):

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

b. Per le interfacce di rete Ethernet:

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

Per evitare conflitti quando le interfacce vengono rinominate (impedendone la ridenominazione), non utilizzare nomi predefiniti potenziali come eth0, ens9f0, ib0 o ibs4f0. Una convenzione di denominazione comune prevede l'utilizzo di 'e' o 'i' per Ethernet o InfiniBand, seguito dal numero dello slot PCIe e da una lettera che indica la porta. Ad esempio, la seconda porta di un adattatore InfiniBand installato nello slot 3 è: I3b.

Se si utilizza un modello di nodo di file verificato, fare clic su "qui" Esempio di mapping indirizzo-porta logica PCIe.

12. Specificare facoltativamente la configurazione da applicare a tutti i servizi BeeGFS nel cluster. È possibile trovare i valori di configurazione predefiniti "qui"e specificare altrove la configurazione per servizio:

a. Servizio di gestione BeeGFS:

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

b. Servizi di metadati BeeGFS:

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

c. Servizi di storage BeeGFS:

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

13. A partire da BeeGFS 7.2.7 e 7.3.1 "autenticazione della connessione" deve essere configurato o disabilitato esplicitamente. Esistono alcuni modi per configurarlo utilizzando la distribuzione basata su Ansible:

a. Per impostazione predefinita, l'implementazione configura automaticamente l'autenticazione della connessione e genera un `connauthfile` Che verranno distribuiti a tutti i nodi di file e utilizzati con i servizi BeeGFS. Questo file verrà anche posizionato/mantenuto nel nodo di controllo Ansible all'indirizzo `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` dove deve essere mantenuto (in modo sicuro) per il riutilizzo con i client che devono accedere a questo file system.

i. Per generare una nuova chiave, specificare `-e "beegfs_ha_conn_auth_force_new=True` Quando si esegue il playbook Ansible. Nota: Questa operazione viene ignorata se si seleziona `beegfs_ha_conn_auth_secret` è definito.

ii. Per le opzioni avanzate, fare riferimento all'elenco completo dei valori predefiniti inclusi nella "Ruolo BeeGFS ha".

b. È possibile utilizzare un segreto personalizzato definendo quanto segue in `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. L'autenticazione della connessione può essere disattivata completamente (NON consigliata):

```
beegfs_ha_conn_auth_enabled: false
```

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta la configurazione di un nodo di file comune.

Utilizzo di HDR (200 GB) InfiniBand con i nodi a blocchi NetApp EF600:

Per utilizzare HDR (200 GB) InfiniBand con EF600, il gestore di subnet deve supportare la virtualizzazione. Se i nodi di file e blocchi sono collegati mediante uno switch, questo deve essere attivato nel gestore di subnet per il fabric complessivo.

Se i nodi di file e blocchi sono connessi direttamente con InfiniBand, un'istanza di `opensm` deve essere configurata su ogni nodo di file per ogni interfaccia connessa direttamente a un nodo di blocco. Per eseguire questa operazione, specificare `configure: true` quando "configurazione delle interfacce di storage dei nodi di file".

Attualmente la versione in arrivo di `opensm` fornita con le distribuzioni Linux supportate non supporta la

virtualizzazione. È invece necessario installare e configurare la versione di `opensm` da NVIDIA OpenFabrics Enterprise Distribution (OFED). Sebbene la distribuzione con Ansible sia ancora supportata, sono necessari alcuni passaggi aggiuntivi:

1. Utilizzando curl o lo strumento desiderato, scaricare i pacchetti per la versione di opensm elencati nella "requisiti tecnologici" sezione dal sito web di NVIDIA alla `<INVENTORY>/packages/` directory. Ad esempio:

```
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. Sotto `group_vars/ha_cluster.yml` definire la seguente configurazione:

```
### 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"
```

Specificare la configurazione del nodo del blocco comune

Specificare la configurazione del nodo a blocchi comune utilizzando le variabili di gruppo (group_vars).

Panoramica

La configurazione che deve essere utilizzata per tutti i nodi a blocchi è definita in `group_vars/eseries_storage_systems.yml`. In genere include:

- Dettagli su come il nodo di controllo Ansible deve connettersi ai sistemi storage e-Series utilizzati come nodi a blocchi.

- Quali versioni del firmware, DI NVSRAM e del firmware del disco devono essere eseguite dai nodi.
- Configurazione globale, incluse le impostazioni della cache, la configurazione dell'host e le modalità di provisioning dei volumi.

> Le opzioni impostate in questo file possono essere definite anche su singoli nodi a blocchi, ad esempio se sono in uso modelli hardware misti o se si dispone di password diverse per ciascun nodo. La configurazione sui singoli nodi a blocchi avrà la precedenza sulla configurazione in questo file.

Fasi

Creare il file `group_vars/eseries_storage_systems.yml` e compilarlo come segue:

1. Ansible non utilizza SSH per connettersi ai nodi a blocchi, ma le API REST. Per ottenere questo risultato, dobbiamo stabilire:

```
ansible_connection: local
```

2. Specificare il nome utente e la password per gestire ciascun nodo. Il nome utente può essere omesso (e l'impostazione predefinita è admin), altrimenti è possibile specificare qualsiasi account con privilegi di amministratore. Specificare inoltre se i certificati SSL devono essere verificati o ignorati:

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

> Si sconsiglia di elencare le password in testo non crittografato. Utilizzare Ansible vault o fornire il `eseries_system_password` Quando si esegue Ansible con --extra-vars.

3. Facoltativamente, specificare il firmware del controller, NVSRAM e il firmware del disco da installare sui nodi. Questi dovranno essere scaricati su `packages/` Prima di eseguire Ansible. È possibile scaricare il firmware del controller e-Series e NVSRAM "qui" e firmware del disco "qui":

```
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".
```

Se viene specificata questa configurazione, Ansible aggiorna automaticamente tutto il firmware, incluso il riavvio dei controller (se necessario) con senza richieste aggiuntive. Si prevede che ciò non causi interruzioni per BeeGFS/host i/o, ma può causare una temporanea diminuzione delle performance.

4. Regolare le impostazioni predefinite della configurazione globale del sistema. Le opzioni e i valori elencati di seguito sono generalmente consigliati per BeeGFS su NetApp, ma possono essere modificati se necessario:

```
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. Configurare le impostazioni predefinite per il provisioning di volumi globali. Le opzioni e i valori elencati di seguito sono generalmente consigliati per BeeGFS su NetApp, ma possono essere modificati se necessario:

```
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. Se necessario, modificare l'ordine in cui Ansible selezionerà le unità per i pool di storage e i gruppi di volumi tenendo presente le seguenti Best practice:
   a. Elencare tutte le unità (potenzialmente più piccole) che devono essere utilizzate per i volumi di gestione e/o metadati e i volumi di storage durano.
   b. Assicurarsi di bilanciare l'ordine di selezione dei dischi tra i canali disponibili in base ai modelli di shelf di dischi/enclosure di dischi. Ad esempio, con EF600 e senza espansioni, i dischi 0-11 si trovano sul canale 1 del disco e i dischi 12-23 sul canale del disco. Pertanto, è necessario scegliere una strategia per bilanciare la selezione del disco `disk shelf:drive` 99:0, 99:23, 99:1, 99:22, ecc. In caso di più enclosure, la prima cifra rappresenta l'ID dello shelf del disco.

```
# 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"
```

Fare clic su "qui" per un esempio di un file di inventario completo che rappresenta la configurazione di un nodo a blocchi comune.

Definire i servizi BeeGFS

Definire il servizio di gestione BeeGFS

I servizi BeeGFS vengono configurati utilizzando variabili di gruppo (group_vars).

Panoramica

In questa sezione viene illustrata la definizione del servizio di gestione BeeGFS. Nel cluster ha per un file system specifico dovrebbe esistere un solo servizio di questo tipo. La configurazione di questo servizio include la definizione di:

- Il tipo di servizio (gestione).
- Definizione di qualsiasi configurazione applicabile solo a questo servizio BeeGFS.
- Configurazione di uno o più IP mobili (interfacce logiche) in cui è possibile raggiungere questo servizio.
- Specificare dove/come deve essere un volume per memorizzare i dati per questo servizio (la destinazione di gestione di BeeGFS).

Fasi

Creare un nuovo file `group_vars/mgmt.yml` e fare riferimento alla "Pianificare il file system" sezione compilarlo come segue:

1. Indicare che questo file rappresenta la configurazione per un servizio di gestione BeeGFS:

```
beegfs_service: management
```

2. Definire qualsiasi configurazione da applicare solo a questo servizio BeeGFS. Questo non è generalmente richiesto per il servizio di gestione a meno che non sia necessario attivare le quote, tuttavia qualsiasi parametro di configurazione supportato da `beegfs-mgmtd.conf` può essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, e. `connNetFilterFile`.

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

3. Configurare uno o più IP mobili che altri servizi e client useranno per connettersi a questo servizio (in questo modo si imposterà automaticamente il BeeGFS `connInterfacesFile` opzione):

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

4. Facoltativamente, specificare una o più subnet IP consentite che possono essere utilizzate per la comunicazione in uscita (in questo modo si imposterà automaticamente BeeGFS `connNetFilterFile` opzione):

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

5. Specificare l'obiettivo di gestione di BeeGFS in cui il servizio memorizzerà i dati in base alle seguenti linee guida:
   a. Lo stesso nome del pool di storage o del gruppo di volumi può essere utilizzato per più servizi/destinazioni BeeGFS, assicurandosi semplicemente di utilizzare lo stesso nome `name`, `raid_level`, `criteria_*`, e. `common_*` configurazione per ciascun servizio (i volumi elencati per ciascun servizio devono essere diversi).
   b. Le dimensioni dei volumi devono essere specificate come percentuale del gruppo di pool/volumi di storage e il totale non deve superare 100 per tutti i servizi/volumi che utilizzano un particolare gruppo di pool/volumi di storage. Nota quando si utilizzano le unità SSD, si consiglia di lasciare spazio libero nel gruppo di volumi per massimizzare le prestazioni e la durata dell'unità SSD (fare clic "qui" per ulteriori dettagli).
   c. Fare clic su "qui" per un elenco completo delle opzioni di configurazione disponibili per `eseries_storage_pool_configuration`. Notare alcune opzioni, ad esempio `state`, `host`, `host_type`, `workload_name`, e. `workload_metadata` i nomi dei volumi e vengono generati automaticamente e non devono essere specificati qui.

```
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
```

Fare clic su "qui" Per un esempio di un file di inventario completo che rappresenta un servizio di gestione BeeGFS.

Definire il servizio di metadati BeeGFS

I servizi BeeGFS vengono configurati utilizzando variabili di gruppo (group_vars).

Panoramica

In questa sezione viene illustrata la definizione del servizio di metadati BeeGFS. Almeno un servizio di questo tipo dovrebbe esistere nei cluster ha per un particolare file system. La configurazione di questo servizio include la definizione di:

- Il tipo di servizio (metadati).
- Definizione di qualsiasi configurazione applicabile solo a questo servizio BeeGFS.
- Configurazione di uno o più IP mobili (interfacce logiche) in cui è possibile raggiungere questo servizio.
- Specificare dove/come deve essere un volume per memorizzare i dati per questo servizio (la destinazione dei metadati BeeGFS).

Fasi

Facendo riferimento alla "Pianificare il file system" sezione, creare un file su `group_vars/meta_<ID>.yml` per ogni servizio di metadati nel cluster e compilarli come segue:

1. Indica che questo file rappresenta la configurazione per un servizio di metadati BeeGFS:

```
beegfs_service: metadata
```

2. Definire qualsiasi configurazione da applicare solo a questo servizio BeeGFS. È necessario specificare almeno la porta TCP e UDP desiderata, tuttavia qualsiasi parametro di configurazione supportato da `beegfs-meta.conf` può anche essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `sysMgmtdHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, e. `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. Configurare uno o più IP mobili che altri servizi e client useranno per connettersi a questo servizio (in questo modo si imposterà automaticamente il BeeGFS `connInterfacesFile` opzione):

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

4. Facoltativamente, specificare una o più subnet IP consentite che possono essere utilizzate per la

comunicazione in uscita (in questo modo si imposterà automaticamente BeeGFS `connNetFilterFile` opzione):

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

5. Specificare la destinazione dei metadati BeeGFS in cui il servizio memorizzerà i dati in base alle seguenti linee guida (questa operazione configurerà automaticamente anche `storeMetaDirectory` opzione):
   a. Lo stesso nome del pool di storage o del gruppo di volumi può essere utilizzato per più servizi/destinazioni BeeGFS, assicurandosi semplicemente di utilizzare lo stesso nome `name`, `raid_level`, `criteria_*`, e. `common_*` configurazione per ciascun servizio (i volumi elencati per ciascun servizio devono essere diversi).
   b. Le dimensioni dei volumi devono essere specificate come percentuale del gruppo di pool/volumi di storage e il totale non deve superare 100 per tutti i servizi/volumi che utilizzano un particolare gruppo di pool/volumi di storage. Nota quando si utilizzano le unità SSD, si consiglia di lasciare spazio libero nel gruppo di volumi per massimizzare le prestazioni e la durata dell'unità SSD (fare clic "qui" per ulteriori dettagli).
   c. Fare clic su "qui" per un elenco completo delle opzioni di configurazione disponibili per `eseries_storage_pool_configuration`. Notare alcune opzioni, ad esempio `state`, `host`, `host_type`, `workload_name`, e. `workload_metadata` i nomi dei volumi e vengono generati automaticamente e non devono essere specificati qui.

```
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
```

Fare clic su "qui" Per un esempio di un file di inventario completo che rappresenta un servizio di metadati BeeGFS.

Definire il servizio di storage BeeGFS

I servizi BeeGFS vengono configurati utilizzando variabili di gruppo (group_vars).

Panoramica

In questa sezione viene illustrata la definizione del servizio di storage BeeGFS. Almeno un servizio di questo

tipo dovrebbe esistere nei cluster ha per un particolare file system. La configurazione di questo servizio include la definizione di:

- Il tipo di servizio (storage).
- Definizione di qualsiasi configurazione applicabile solo a questo servizio BeeGFS.
- Configurazione di uno o più IP mobili (interfacce logiche) in cui è possibile raggiungere questo servizio.
- Specificare dove/come devono essere i volumi per memorizzare i dati per questo servizio (le destinazioni di storage BeeGFS).

Fasi

Facendo riferimento alla "Pianificare il file system" sezione, creare un file all'indirizzo `group_vars/stor_<ID>.yml` per ciascun servizio di archiviazione nel cluster e compilarli come segue:

1. Indicare che questo file rappresenta la configurazione per un servizio di storage BeeGFS:

```
beegfs_service: storage
```

2. Definire qualsiasi configurazione da applicare solo a questo servizio BeeGFS. È necessario specificare almeno la porta TCP e UDP desiderata, tuttavia qualsiasi parametro di configurazione supportato da `beegfs-storage.conf` può anche essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, e. `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. Configurare uno o più IP mobili che altri servizi e client useranno per connettersi a questo servizio (in questo modo si imposterà automaticamente il BeeGFS `connInterfacesFile` opzione):

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

4. Facoltativamente, specificare una o più subnet IP consentite che possono essere utilizzate per la comunicazione in uscita (in questo modo si imposterà automaticamente BeeGFS `connNetFilterFile` opzione):

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

5. Specificare le destinazioni di storage BeeGFS in cui il servizio memorizzerà i dati in base alle seguenti linee guida (questa operazione configurerà automaticamente anche `storeStorageDirectory` opzione):
   a. Lo stesso nome del pool di storage o del gruppo di volumi può essere utilizzato per più servizi/destinazioni BeeGFS, assicurandosi semplicemente di utilizzare lo stesso nome `name`, `raid_level`, `criteria_*`, e. `common_*` configurazione per ciascun servizio (i volumi elencati per ciascun servizio devono essere diversi).
   b. Le dimensioni dei volumi devono essere specificate come percentuale del gruppo di pool/volumi di storage e il totale non deve superare 100 per tutti i servizi/volumi che utilizzano un particolare gruppo di pool/volumi di storage. Nota quando si utilizzano le unità SSD, si consiglia di lasciare spazio libero nel gruppo di volumi per massimizzare le prestazioni e la durata dell'unità SSD (fare clic "qui" per ulteriori dettagli).
   c. Fare clic su "qui" per un elenco completo delle opzioni di configurazione disponibili per `eseries_storage_pool_configuration`. Notare alcune opzioni, ad esempio `state`, `host`, `host_type`, `workload_name`, e. `workload_metadata` i nomi dei volumi e vengono generati automaticamente e non devono essere specificati qui.

```
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
```

Fare clic su "qui" Esempio di un file di inventario completo che rappresenta un servizio di storage BeeGFS.

Mappare i servizi BeeGFS ai nodi di file

Specificare quali nodi di file possono eseguire ciascun servizio BeeGFS utilizzando `inventory.yml` file.

Panoramica

In questa sezione viene illustrato come creare `inventory.yml` file. Ciò include l'elenco di tutti i nodi a blocchi e la specifica dei nodi di file che possono eseguire ciascun servizio BeeGFS.

Fasi

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

1. Dall'inizio del file, creare la struttura di inventario Ansible standard:

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

2. Creare un gruppo contenente tutti i nodi a blocchi che partecipano a questo cluster ha:

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

3. Creare un gruppo che conterrà tutti i servizi BeeGFS nel cluster e i nodi di file che li eseguiranno:

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

4. Per ogni servizio BeeGFS nel cluster, definire il nodo/i file preferito/i e secondario/i che deve eseguire tale servizio:

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

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

Informazioni sul copyright

Informazioni sul marchio commerciale