



Utilizzare architetture personalizzate

BeeGFS on NetApp with E-Series Storage

NetApp

January 27, 2026

Sommario

- Utilizzare architetture personalizzate 1
 - Panoramica e requisiti 1
 - Introduzione 1
 - Panoramica sull’implementazione 1
 - Requisiti 2
 - Configurazione iniziale 2
 - Installazione e cavo hardware 2
 - Impostare i nodi di file e blocchi 6
 - Impostare Ansible Control Node 7
 - Definire il file system BeeGFS 8
 - Panoramica di Ansible Inventory 8
 - Pianificare il file system 9
 - Definire i nodi di file e blocchi 10
 - Definire i servizi BeeGFS 28
 - Mappare i servizi BeeGFS ai nodi di file 33
 - Implementare il file system BeeGFS 34
 - Panoramica di Ansible Playbook 34
 - Implementare il cluster BeeGFS ha 36
 - Implementare i client BeeGFS 39
 - Verificare l’implementazione di BeeGFS 44

Utilizzare architetture personalizzate

Panoramica e requisiti

Utilizzare qualsiasi sistema storage NetApp e/EF-Series come nodi a blocchi BeeGFS e server x86 come nodi di file BeeGFS quando si implementano cluster ad alta disponibilità BeeGFS utilizzando Ansible.



Le definizioni per la terminologia utilizzata in questa sezione "[termini e concetti](#)" sono disponibili nella pagina.

Introduzione

Sebbene "[Architetture verificate da NetApp](#)" offrano configurazioni di riferimento predefinite e linee guida sul dimensionamento, alcuni clienti e partner potrebbero preferire la progettazione di architetture personalizzate più adatte a specifici requisiti o preferenze hardware. Uno dei principali vantaggi della scelta di BeeGFS su NetApp è la capacità di implementare cluster ha a disco condiviso BeeGFS utilizzando Ansible, semplificando la gestione dei cluster e migliorando l'affidabilità con i componenti ha creati da NetApp. L'implementazione di architetture BeeGFS personalizzate su NetApp viene ancora eseguita utilizzando Ansible, mantenendo un approccio simile all'appliance su una gamma flessibile di hardware.

In questa sezione vengono descritti i passaggi generali necessari per implementare i file system BeeGFS sull'hardware NetApp e l'utilizzo di Ansible per configurare i file system BeeGFS. Per informazioni dettagliate sulle Best practice relative alla progettazione dei file system BeeGFS ed esempi ottimizzati, fare riferimento alla "[Architetture verificate da NetApp](#)" sezione.

Panoramica sull'implementazione

In genere, l'implementazione di un file system BeeGFS richiede i seguenti passaggi:

- Configurazione iniziale:
 - Installazione/cavo hardware.
 - Impostare i nodi di file e blocchi.
 - Impostare un nodo di controllo Ansible.
- Definire il file system BeeGFS come un inventario Ansible.
- Eseguire Ansible su file e nodi a blocchi per implementare BeeGFS.
 - Facoltativamente per configurare i client e montare BeeGFS.

Le sezioni successive tratterà questi passaggi in modo più dettagliato.

Ansible gestisce tutte le attività di provisioning e configurazione del software, tra cui:



- Creazione/mappatura di volumi su nodi a blocchi.
- Formattazione/messa a punto di volumi su nodi di file.
- Installazione/configurazione del software sui nodi di file.
- Stabilire il cluster ha e configurare le risorse BeeGFS e i servizi del file system.

Requisiti

Il supporto per BeeGFS in Ansible viene rilasciato il "[Ansible Galaxy](#)" Insieme di ruoli e moduli che automatizzano l'implementazione e la gestione end-to-end dei cluster BeeGFS ha.

BeeGFS è dotato di versioni che seguono uno schema di controllo delle versioni di <major>.<minor>.<patch> e l'insieme mantiene i ruoli per ogni versione supportata di <major>.<minor> di BeeGFS, ad esempio BeeGFS 7.2 o BeeGFS 7.3. Man mano che vengono rilasciati gli aggiornamenti della raccolta, la versione della patch in ciascun ruolo verrà aggiornata in modo da indicare l'ultima versione di BeeGFS disponibile per tale ramo di release (esempio: 7.2.8). Ogni versione della raccolta è inoltre testata e supportata con specifiche distribuzioni e versioni di Linux, attualmente Red Hat per i nodi di file e Red Hat e Ubuntu per i client. L'esecuzione di altre distribuzioni non è supportata e l'esecuzione di altre versioni (in particolare altre versioni principali) non è consigliata.

Nodo di controllo Ansible

Questo nodo conterrà l'inventario e i playbook utilizzati per gestire BeeGFS. Richiede:

- Ansible 6.x (ansible-core 2.13)
- Python 3.6 (o versione successiva)
- Pacchetti Python (pip): Ipaddr e netaddr

Si consiglia inoltre di impostare SSH senza password dal nodo di controllo a tutti i nodi di file e client BeeGFS.

Nodi di file BeeGFS

I nodi file devono eseguire Red Hat Enterprise Linux (RHEL) 9.4 e avere accesso al repository HA contenente i pacchetti richiesti (pacemaker, corosync, fence-agents-all, resource-agents). Ad esempio, è possibile eseguire il seguente comando per abilitare il repository appropriato su RHEL 9:

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

Nodi client BeeGFS

È disponibile un ruolo Ansible del client BeeGFS per installare il pacchetto client BeeGFS e gestire i mount BeeGFS. Questo ruolo è stato testato con RHEL 9.4 e Ubuntu 22.04.

Se non si utilizza Ansible per configurare il client BeeGFS e montare BeeGFS, qualsiasi "[BeeGFS supporta la distribuzione e il kernel Linux](#)" può essere utilizzato.

Configurazione iniziale

Installazione e cavo hardware

Procedure necessarie per installare e collegare l'hardware utilizzato per eseguire BeeGFS su NetApp.

Pianificare l'installazione

Ciascun file system BeeGFS è costituito da un certo numero di nodi di file che eseguono i servizi BeeGFS utilizzando lo storage back-end fornito da un certo numero di nodi a blocchi. I file node sono configurati in uno o più cluster ad alta disponibilità per fornire la fault tolerance per i servizi BeeGFS. Ogni nodo a blocchi è già una coppia ha attiva/attiva. Il numero minimo di nodi di file supportati in ciascun cluster ha è tre e il numero massimo di nodi di file supportati in ciascun cluster è dieci. I file system BeeGFS possono scalare oltre dieci nodi implementando più cluster ha indipendenti che lavorano insieme per fornire un singolo namespace del file system.

In genere, ogni cluster ha viene implementato come una serie di "building block" in cui alcuni nodi di file (server x86) sono collegati direttamente a un certo numero di nodi a blocchi (in genere sistemi storage e-Series). Questa configurazione crea un cluster asimmetrico, in cui i servizi BeeGFS possono essere eseguiti solo su alcuni nodi di file che hanno accesso allo storage a blocchi di back-end utilizzato per le destinazioni BeeGFS. Il bilanciamento dei nodi file-to-block in ciascun building block e del protocollo di storage in uso per le connessioni dirette dipende dai requisiti di una particolare installazione.

Un'architettura di cluster ha alternativa utilizza un fabric di storage (noto anche come SAN (Storage Area Network) tra i nodi di file e blocchi per stabilire un cluster simmetrico. Ciò consente l'esecuzione dei servizi BeeGFS su qualsiasi nodo di file in un cluster ha specifico. Poiché i cluster generalmente simmetrici non sono così convenienti a causa dell'hardware SAN aggiuntivo, questa documentazione presuppone l'utilizzo di un cluster asimmetrico implementato come una serie di uno o più building block.

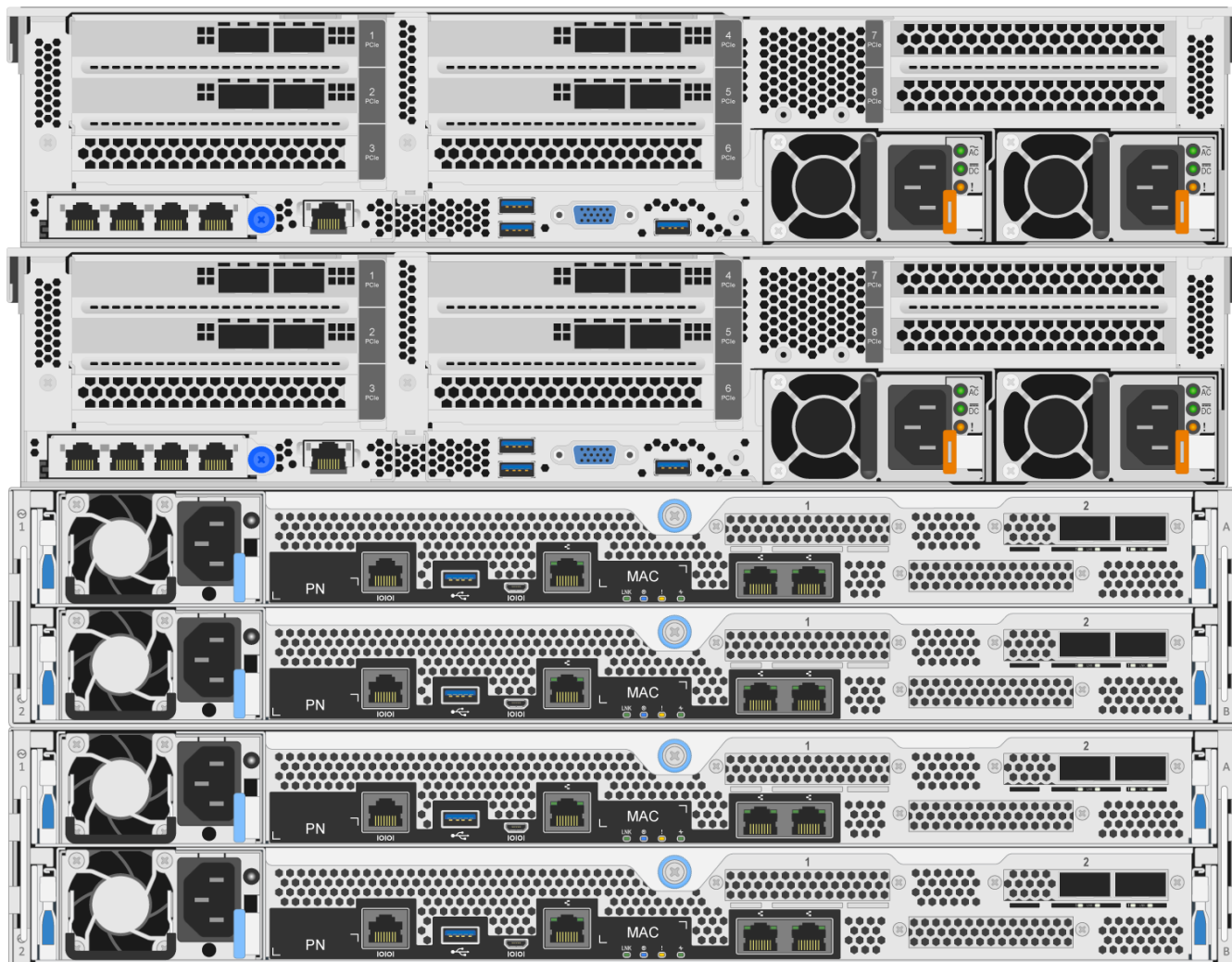


Assicurarsi che l'architettura del file system desiderata per una particolare implementazione di BeeGFS sia ben compresa prima di procedere con l'installazione.

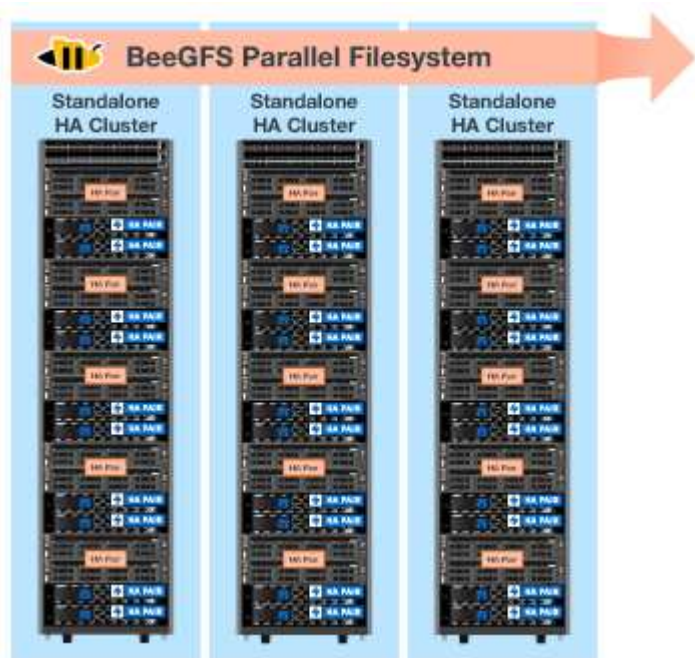
Hardware per rack

Quando si pianifica l'installazione, è importante che tutte le apparecchiature di ciascun building block siano installate in rack adiacenti. La procedura consigliata prevede il racking dei nodi di file immediatamente sopra i nodi di blocco in ciascun building block. Seguire la documentazione relativa ai modelli di file e. "blocco" nodi utilizzati durante l'installazione di guide e hardware nel rack.

Esempio di un singolo building block:

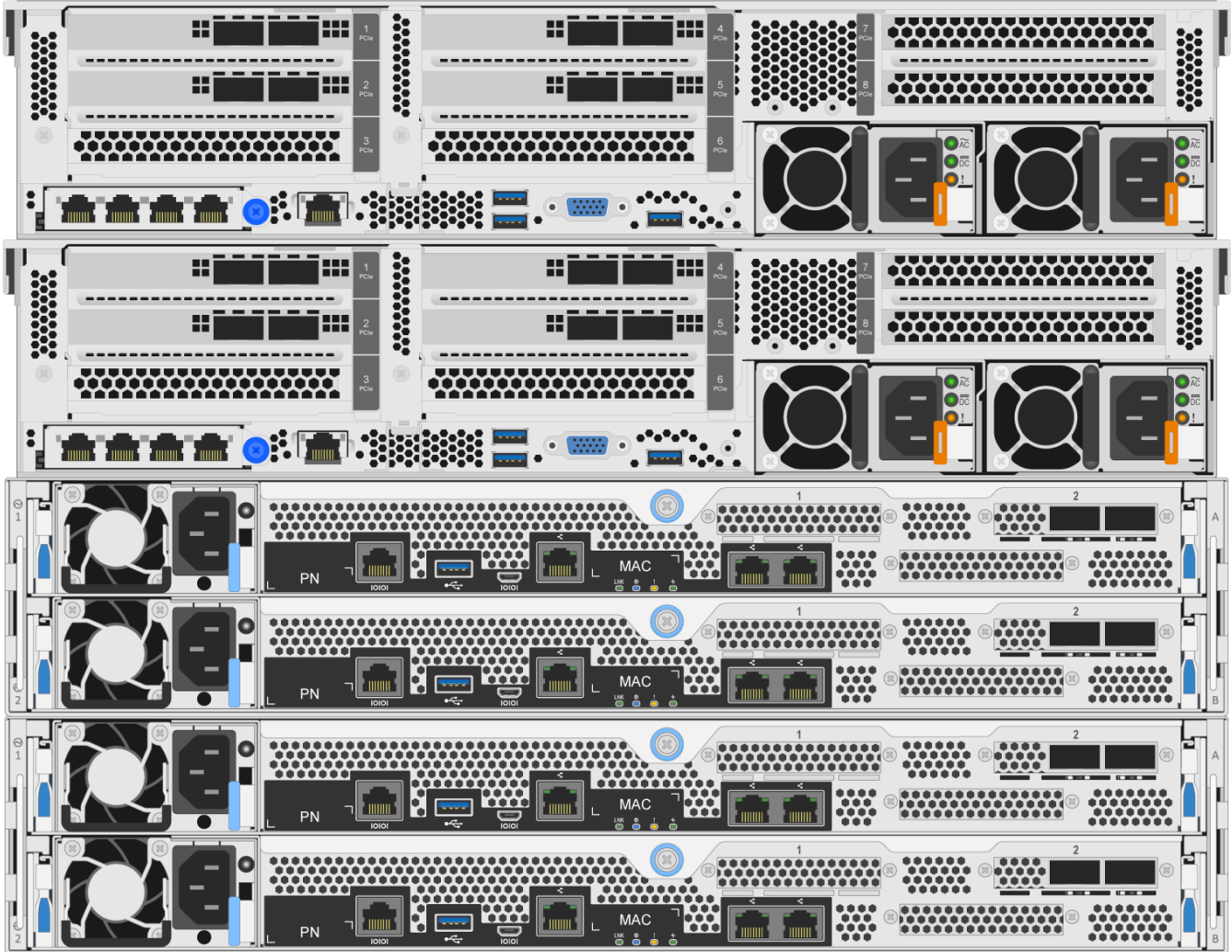


Esempio di un'installazione BeeGFS di grandi dimensioni in cui sono presenti più elementi di base in ciascun cluster ha e più cluster ha nel file system:



Nodi di blocco e file via cavo

In genere, le porte HIC dei nodi a blocchi e-Series vengono collegate direttamente alle porte dell'adattatore del canale host designato (per i protocolli InfiniBand) o dell'adattatore del bus host (per Fibre Channel e altri protocolli) dei nodi di file. Il modo esatto per stabilire queste connessioni dipenderà dall'architettura del file system desiderata, ecco un esempio "[Basato sull'architettura verificata di seconda generazione di BeeGFS su NetApp](#)":

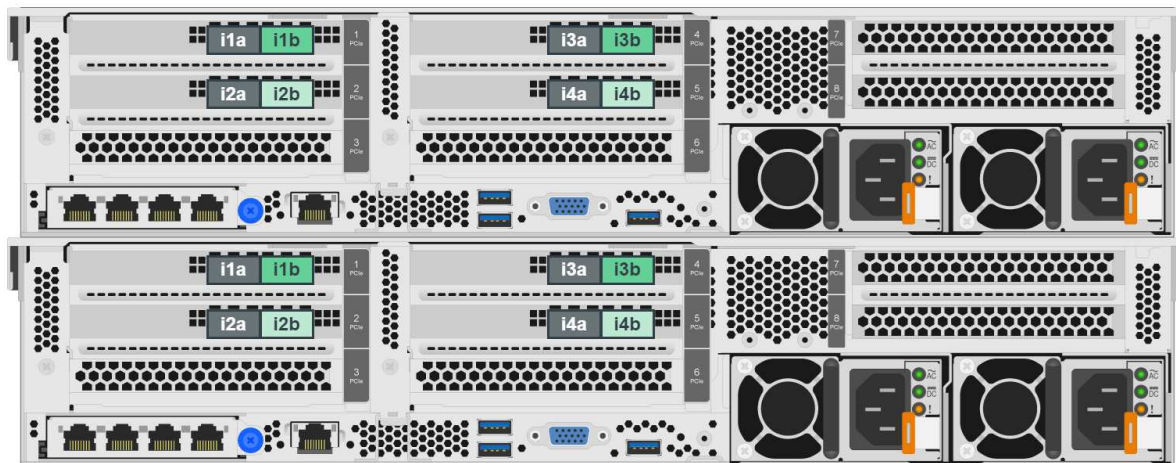


Collegare i nodi di file alla rete client

Ogni nodo di file avrà un certo numero di porte InfiniBand o Ethernet designate per il traffico del client BeeGFS. A seconda dell'architettura, ciascun nodo di file disporrà di una o più connessioni a una rete client/storage dalle performance elevate, potenzialmente a più switch per la ridondanza e l'aumento della larghezza di banda. Di seguito viene riportato un esempio di cablaggio del client che utilizza switch di rete ridondanti, in cui le porte evidenziate in verde scuro e verde chiaro sono collegate a switch separati:

H01

H02



Gestione delle connessioni rete e alimentazione

Stabilire le connessioni di rete necessarie per le reti in-band e out-of-band.

Collegare tutti gli alimentatori assicurandosi che ciascun nodo di file e blocchi sia collegato a più unità di distribuzione dell'alimentazione per la ridondanza (se disponibile).

Impostare i nodi di file e blocchi

Operazioni manuali necessarie per impostare i nodi di file e blocchi prima di eseguire Ansible.

Nodi di file

Configurare Baseboard Management Controller (BMC)

Un BMC (Baseboard Management Controller), a volte chiamato Service Processor, è il nome generico della funzionalità di gestione out-of-band integrata in varie piattaforme server che possono fornire accesso remoto anche se il sistema operativo non è installato o accessibile. I vendor in genere commercializzano questa funzionalità con un proprio marchio. Ad esempio, su Lenovo SR665, BMC viene definito XCC (Lenovo XClarity Controller).

Seguire la documentazione del vendor del server per abilitare le licenze necessarie per accedere a questa funzionalità e assicurarsi che il BMC sia connesso alla rete e configurato in modo appropriato per l'accesso remoto.



Se si desidera utilizzare Redfish per la schermata basata su BMC, assicurarsi che Redfish sia attivato e che l'interfaccia BMC sia accessibile dal sistema operativo installato nel nodo file. Potrebbe essere necessaria una configurazione speciale sullo switch di rete se BMC e il sistema operativo condividono la stessa interfaccia di rete fisica.

Mettere a punto le impostazioni di sistema

Utilizzando l'interfaccia del programma di configurazione del sistema (BIOS/UEFI), assicurarsi che le impostazioni siano impostate per massimizzare le prestazioni. Le impostazioni esatte e i valori ottimali variano in base al modello di server in uso. Vengono fornite indicazioni per ["modelli di file node verificati"](#), altrimenti fare riferimento alla documentazione del fornitore del server e alle procedure consigliate basate sul modello in uso.

Installare un sistema operativo

Installare un sistema operativo supportato in base ai requisiti del nodo file elencati ["qui"](#). Fare riferimento a eventuali passaggi aggiuntivi riportati di seguito in base alla distribuzione Linux.

Red Hat

, vedere ["Come registrarsi e sottoscrivere un sistema RHEL"](#) E ["Come limitare gli aggiornamenti"](#) .

Abilitare il repository Red Hat contenente i pacchetti richiesti per l'alta disponibilità:

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

Configurare la rete di gestione

Configurare le interfacce di rete necessarie per consentire la gestione in banda del sistema operativo. I passaggi esatti dipendono dalla distribuzione e dalla versione di Linux in uso.



Assicurarsi che SSH sia attivato e che tutte le interfacce di gestione siano accessibili dal nodo di controllo Ansible.

Aggiornare il firmware HCA e HBA

Assicurarsi che tutti gli HBA e gli HCA eseguano le versioni del firmware supportate elencate sul ["Matrice di interoperabilità NetApp"](#)e, se necessario, aggiornarle. Ulteriori raccomandazioni per gli adattatori NVIDIA ConnectX sono disponibili ["qui"](#).

Nodi a blocchi

Seguire i passi da a. ["Inizia a lavorare con e-Series"](#) per configurare la porta di gestione su ciascun controller di nodi a blocchi e, facoltativamente, impostare il nome dell'array di storage per ciascun sistema.



Non è necessaria alcuna configurazione aggiuntiva oltre a garantire che tutti i nodi a blocchi siano accessibili dal nodo di controllo Ansible. La configurazione di sistema rimanente verrà applicata/mantenuta utilizzando Ansible.

Impostare Ansible Control Node

Impostare un nodo di controllo Ansible per implementare e gestire il file system.

Panoramica

Un nodo di controllo Ansible è una macchina Linux fisica o virtuale utilizzata per gestire il cluster. Deve soddisfare i seguenti requisiti:

- Soddisfare il ["requisiti"](#) ruolo di ha BeeGFS, incluse le versioni installate di Ansible, Python e qualsiasi altro pacchetto Python.
- Incontra il funzionario ["Requisiti dei nodi di controllo Ansible"](#) incluse le versioni del sistema operativo.
- Avere accesso SSH e HTTPS a tutti i nodi di file e blocchi.

La procedura di installazione dettagliata è disponibile ["qui"](#).

Definire il file system BeeGFS

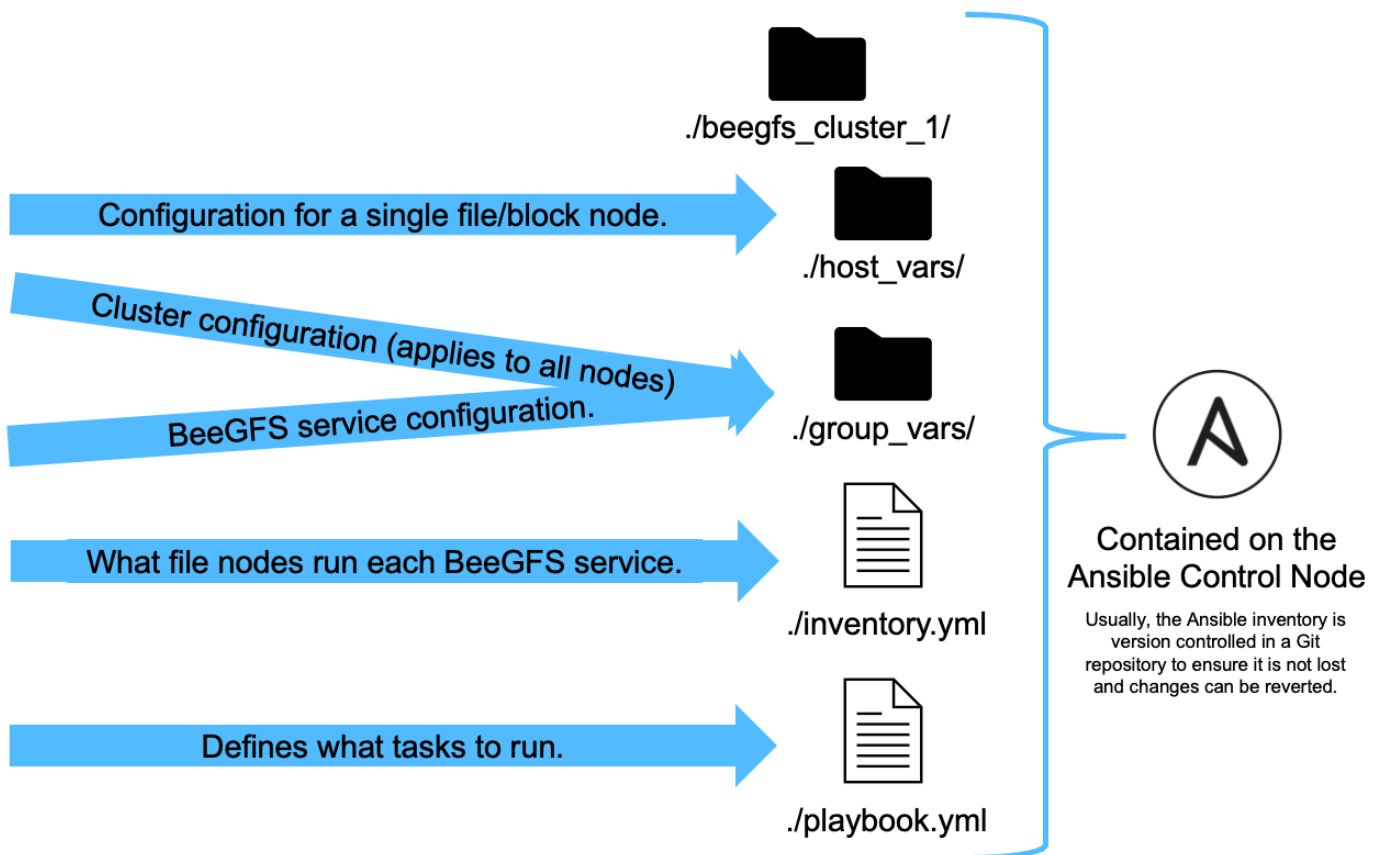
Panoramica di Ansible Inventory

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)>

Scopo	Porta InfiniBand	Indirizzo IP o intervallo
Metadati BeeGFS	<INTERFACE(s)>	<RANGE>
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>/yaml` 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 ilb
  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 `mstconfig` lo strumento incluso in `mstflint`, un pacchetto open source che fa parte di <https://docs.nvidia.com/networking/display/mftv4270/mft+supported+configurations+and+parameters> 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 `eseries-
[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:` 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>.yaml` 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>.yaml` 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 schermo.
- 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 firewall 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 schermo:
 - 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 schermo 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 schermo, 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/<sysMgmtHost>_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 omissso (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_nvram: "packages/<FILENAME>.dlp" # Ex.
"packages/N6000-880834-D08.dlp"
eseries_drive_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, 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"`

```
# 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-mgmt.conf` può essere incluso. Nota: I seguenti parametri vengono configurati automaticamente/altrove e non devono essere specificati qui: `storeMgmtDirectory`, `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:
- 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).
 - 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).
 - 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>.yaml` 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: `sysMgmtHost`, `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>.yaml` 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: `sysMgmtHost`, `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):
- 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).
 - 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).
 - 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.

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 il ["Kubespray"](#) progetto consente agli utenti di implementare/mantenere 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_4 # Alternatively specify: beegfs_ha_7_3.

```



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('beegfs_7_2' sono supportati BeeGFS 7,3.x) o BeeGFS 7,2.x). 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. Per "[guida all'upgrade](#)" ulteriori dettagli, fare riferimento alle informazioni più recenti.

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 "[configurazione di un nodo di file comune](#)" si specifica, se si sceglie l'opzione predefinita per far sì che Ansible gestisca automaticamente l'autenticazione basata sulla connessione, `connAuthFile` è ora possibile trovare un usato come segreto condiviso all'indirizzo

`<playbook_dir>/files/beegfs/<sysMgmtHost>_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 con "[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 ilb
  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 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.
```



Quando si utilizzano i driver NVIDIA OFED, assicurarsi che `beegfs_client_ofed_include_PATH` punti al corretto "header include PATH" per l'installazione Linux. Per ulteriori informazioni, vedere 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:
  - sysMgmtHost: <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 esplicitamente disattivato. 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 regolare 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/<sysMgmtHost>_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 si trovano nella ["Verifica del progetto"](#) sezione di BeeGFS su architettura verificata NetApp.

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

Informazioni sul copyright

Copyright © 2026 NetApp, Inc. Tutti i diritti riservati. Stampato negli Stati Uniti d'America. Nessuna porzione di questo documento soggetta a copyright può essere riprodotta in qualsiasi formato o mezzo (grafico, elettronico o meccanico, inclusi fotocopie, registrazione, nastri o storage in un sistema elettronico) senza previo consenso scritto da parte del detentore del copyright.

Il software derivato dal materiale sottoposto a copyright di NetApp è soggetto alla seguente licenza e dichiarazione di non responsabilità:

IL PRESENTE SOFTWARE VIENE FORNITO DA NETAPP "COSÌ COM'È" E SENZA QUALSIVOGLIA TIPO DI GARANZIA IMPLICITA O ESPRESSA FRA CUI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, GARANZIE IMPLICITE DI COMMERCIALIZZABILITÀ E IDONEITÀ PER UNO SCOPO SPECIFICO, CHE VENGONO DECLINATE DAL PRESENTE DOCUMENTO. NETAPP NON VERRÀ CONSIDERATA RESPONSABILE IN ALCUN CASO PER QUALSIVOGLIA DANNO DIRETTO, INDIRETTO, ACCIDENTALE, SPECIALE, ESEMPLARE E CONSEGUENZIALE (COMPRESI, A TITOLO ESEMPLIFICATIVO E NON ESAUSTIVO, PROCUREMENT O SOSTITUZIONE DI MERCI O SERVIZI, IMPOSSIBILITÀ DI UTILIZZO O PERDITA DI DATI O PROFITTI OPPURE INTERRUZIONE DELL'ATTIVITÀ AZIENDALE) CAUSATO IN QUALSIVOGLIA MODO O IN RELAZIONE A QUALUNQUE TEORIA DI RESPONSABILITÀ, SIA ESSA CONTRATTUALE, RIGOROSA O DOVUTA A INSOLVENZA (COMPRESA LA NEGLIGENZA O ALTRO) INSORTA IN QUALSIASI MODO ATTRAVERSO L'UTILIZZO DEL PRESENTE SOFTWARE ANCHE IN PRESENZA DI UN PREAVVISO CIRCA L'EVENTUALITÀ DI QUESTO TIPO DI DANNI.

NetApp si riserva il diritto di modificare in qualsiasi momento qualunque prodotto descritto nel presente documento senza fornire alcun preavviso. NetApp non si assume alcuna responsabilità circa l'utilizzo dei prodotti o materiali descritti nel presente documento, con l'eccezione di quanto concordato espressamente e per iscritto da NetApp. L'utilizzo o l'acquisto del presente prodotto non comporta il rilascio di una licenza nell'ambito di un qualche diritto di brevetto, marchio commerciale o altro diritto di proprietà intellettuale di NetApp.

Il prodotto descritto in questa guida può essere protetto da uno o più brevetti degli Stati Uniti, esteri o in attesa di approvazione.

LEGENDA PER I DIRITTI SOTTOPOSTI A LIMITAZIONE: l'utilizzo, la duplicazione o la divulgazione da parte degli enti governativi sono soggetti alle limitazioni indicate nel sottoparagrafo (b)(3) della clausola Rights in Technical Data and Computer Software del DFARS 252.227-7013 (FEB 2014) e FAR 52.227-19 (DIC 2007).

I dati contenuti nel presente documento riguardano un articolo commerciale (secondo la definizione data in FAR 2.101) e sono di proprietà di NetApp, Inc. Tutti i dati tecnici e il software NetApp forniti secondo i termini del presente Contratto sono articoli aventi natura commerciale, sviluppati con finanziamenti esclusivamente privati. Il governo statunitense ha una licenza irrevocabile limitata, non esclusiva, non trasferibile, non cedibile, mondiale, per l'utilizzo dei Dati esclusivamente in connessione con e a supporto di un contratto governativo statunitense in base al quale i Dati sono distribuiti. Con la sola esclusione di quanto indicato nel presente documento, i Dati non possono essere utilizzati, divulgati, riprodotti, modificati, visualizzati o mostrati senza la previa approvazione scritta di NetApp, Inc. I diritti di licenza del governo degli Stati Uniti per il Dipartimento della Difesa sono limitati ai diritti identificati nella clausola DFARS 252.227-7015(b) (FEB 2014).

Informazioni sul marchio commerciale

NETAPP, il logo NETAPP e i marchi elencati alla pagina <http://www.netapp.com/TM> sono marchi di NetApp, Inc. Gli altri nomi di aziende e prodotti potrebbero essere marchi dei rispettivi proprietari.