Verwenden Sie individuelle Architekturen

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Inhalt

Verwenden Sie individuelle Architekturen

Überblick und Anforderungen

Verwenden Sie alle NetApp E/EF-Series Storage-Systeme als BeeGFS-Block-Nodes und x86-Server als BeeGFS-Datei-Nodes, wenn Sie BeeGFS High Availability-Cluster mithilfe von Ansible implementieren.

Definitionen für die in diesem Abschnitt verwendete Terminologie finden Sie auf der "Begriffe und Konzepte" Seite.

Einführung

"NetApp-verifizierte Architekturen"Einige Kunden und Partner bieten zwar vordefinierte Referenzkonfigurationen und Hinweise zur Größenbestimmung, möchten aber möglicherweise lieber benutzerdefinierte Architekturen entwickeln, die sich besser an die speziellen Anforderungen oder Hardwarepräferenzen anpassen. Einer der Hauptvorteile der Entscheidung für BeeGFS auf NetApp ist die Möglichkeit, BeeGFS HA-Cluster auf Shared-Festplatten mit Ansible zu implementieren. Dadurch wird das Cluster-Management vereinfacht und die Zuverlässigkeit mithilfe von NetApp entwickelten HA-Komponenten gesteigert. Die Implementierung benutzerdefinierter BeeGFS-Architekturen auf NetApp erfolgt noch immer mithilfe von Ansible und den Appliance-ähnlichen Ansatz auf einem flexiblen Spektrum an Hardware.

Dieser Abschnitt beschreibt die allgemeinen Schritte, die zur Implementierung von BeeGFS-Dateisystemen auf NetApp Hardware und zur Verwendung von Ansible zur Konfiguration von BeeGFS-Dateisystemen erforderlich sind. Details zu Best Practices für das Design von BeeGFS-Dateisystemen und optimierten Beispielen finden Sie im "NetApp-verifizierte Architekturen" Abschnitt.

Implementierungsübersicht

Die Bereitstellung eines BeeGFS-Dateisystems umfasst im Allgemeinen die folgenden Schritte:

- Ersteinrichtung:
  - Hardware installieren/verkabeln.
  - Richten Sie Datei- und Block-Nodes ein.
  - Richten Sie einen Ansible-Steuerungsknoten ein.
- Definieren Sie das BeeGFS-Dateisystem als Ansible-Inventar.
- Ausführen von Ansible für Datei- und Block-Nodes zur Implementierung von BeeGFS
  - Optional zum Einrichten von Clients und Mounten BeeGFS.

In den nachfolgenden Abschnitten werden diese Schritte näher beschrieben.

Ansible übernimmt alle Softwareprovisionierung- und Konfigurationsaufgaben, z. B.:

- Erstellen/Zuordnen von Volumes auf Block-Nodes
- Formatieren/Tuning von Volumes auf Datei-Nodes
- Installation/Konfiguration von Software auf Datei-Nodes
- Einrichten des HA-Clusters und Konfigurieren von BeeGFS-Ressourcen und File-System-Services

Anforderungen

Unterstützung für BeeGFS in Ansible ist veröffentlicht am "Ansible-Galaxie" Als Sammlung von Rollen und Modulen zur Automatisierung der End-to-End-Implementierung und des Managements von BeeGFS HA-Clustern.

BeeGFS selbst ist nach einem <major>.<minor>.<patch> Versioning Schema versioniert und die Sammlung pflegt Rollen für jede unterstützte <major>.<minor> Version von BeeGFS, zum Beispiel BeeGFS 7.2 oder BeeGFS 7.3. Da Updates für die Sammlung veröffentlicht werden, wird die Patch-Version in jeder Rolle aktualisiert, um auf die neueste verfügbare BeeGFS-Version für diesen Release Branch (Beispiel: 7.2.8) zu verweisen. Jede Version der Sammlung wird auch mit bestimmten Linux-Distributionen und -Versionen getestet und unterstützt, derzeit Red Hat für Dateiknoten und Red Hat und Ubuntu für Clients. Die Ausführung anderer Distributionen wird nicht unterstützt, und die Ausführung anderer Versionen (insbesondere anderer Hauptversionen) wird nicht empfohlen.

Ansible-Steuerungsknoten

Dieser Node enthält Inventar und Playbooks, die zum Managen von BeeGFS verwendet werden. Dazu benötigen Sie:

- Ansible, 6.x (ansible-Core, 2.13)
- Python 3.6 (oder höher)
- Python (Pip) Pakete: Ipaddr und netaddr

Es empfiehlt sich auch, passwortlose SSH vom Steuerungsknoten auf alle BeeGFS Datei-Knoten und -Clients einzurichten.

BeeGFS-Dateiknoten

Dateiknoten müssen Red Hat Enterprise Linux (RHEL) 9.4 ausführen und Zugriff auf das HA-Repository mit den erforderlichen Paketen (Pacemaker, Corosync, Fence-Agents-All, Resource-Agents) haben. Beispielsweise kann der folgende Befehl ausgeführt werden, um das entsprechende Repository unter RHEL 9 zu aktivieren:

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

BeeGFS Client-Knoten

Eine BeeGFS-Client-Ansible-Rolle steht zur Verfügung, um das BeeGFS-Client-Paket zu installieren und BeeGFS-Mount(s) zu verwalten. Diese Rolle wurde mit RHEL 9.4 und Ubuntu 22.04 getestet.

Wenn Sie nicht Ansible verwenden, um den BeeGFS-Client einzurichten und BeeGFS zu mounten, any "BeeGFS unterstützte Linux-Distribution und Kernel" Kann verwendet werden.

Ersteinrichtung

Installieren und verkabeln Sie die Hardware

Schritte erforderlich, um Hardware zu installieren und zu verkabeln, die zum Ausführen von BeeGFS auf NetApp verwendet wird.

Planen Sie die Installation

Jedes BeeGFS-Dateisystem besteht aus einer Anzahl von Datei-Nodes, auf denen BeeGFS-Dienste über Backend-Storage ausgeführt werden, der von einer Anzahl von Block-Nodes bereitgestellt wird. Die Datei-Nodes sind in einem oder mehreren Hochverfügbarkeits-Clustern konfiguriert, um Fehlertoleranz für BeeGFS-Services zu bieten. Jeder Block-Node ist bereits ein aktiv/aktiv-HA-Paar. Die Mindestanzahl unterstützter File-Nodes in jedem HA-Cluster beträgt drei und die maximale Anzahl unterstützter File-Nodes in jedem Cluster ist zehn. BeeGFS-Filesysteme können über zehn Nodes hinaus skaliert werden, indem mehrere unabhängige HA-Cluster implementiert werden, die zusammen einen Single Filesystem Namespace bieten.

Normalerweise wird jedes HA-Cluster als eine Reihe von „Bausteinen“ bereitgestellt, in denen einige File-Nodes (x86-Server) direkt mit einer Reihe von Block-Nodes verbunden sind (in der Regel E-Series Storage-Systeme). Diese Konfiguration erzeugt ein asymmetrisches Cluster, in dem BeeGFS-Services nur auf bestimmten Datei-Nodes ausgeführt werden können, die Zugriff auf den Back-End-Block-Storage haben, der für BeeGFS-Ziele verwendet wird. Die Balance zwischen Datei- und Block-Nodes in jedem Baustein und dem für die direkte Verbindung verwendeten Storage-Protokoll hängen von den Anforderungen einer bestimmten Installation ab.

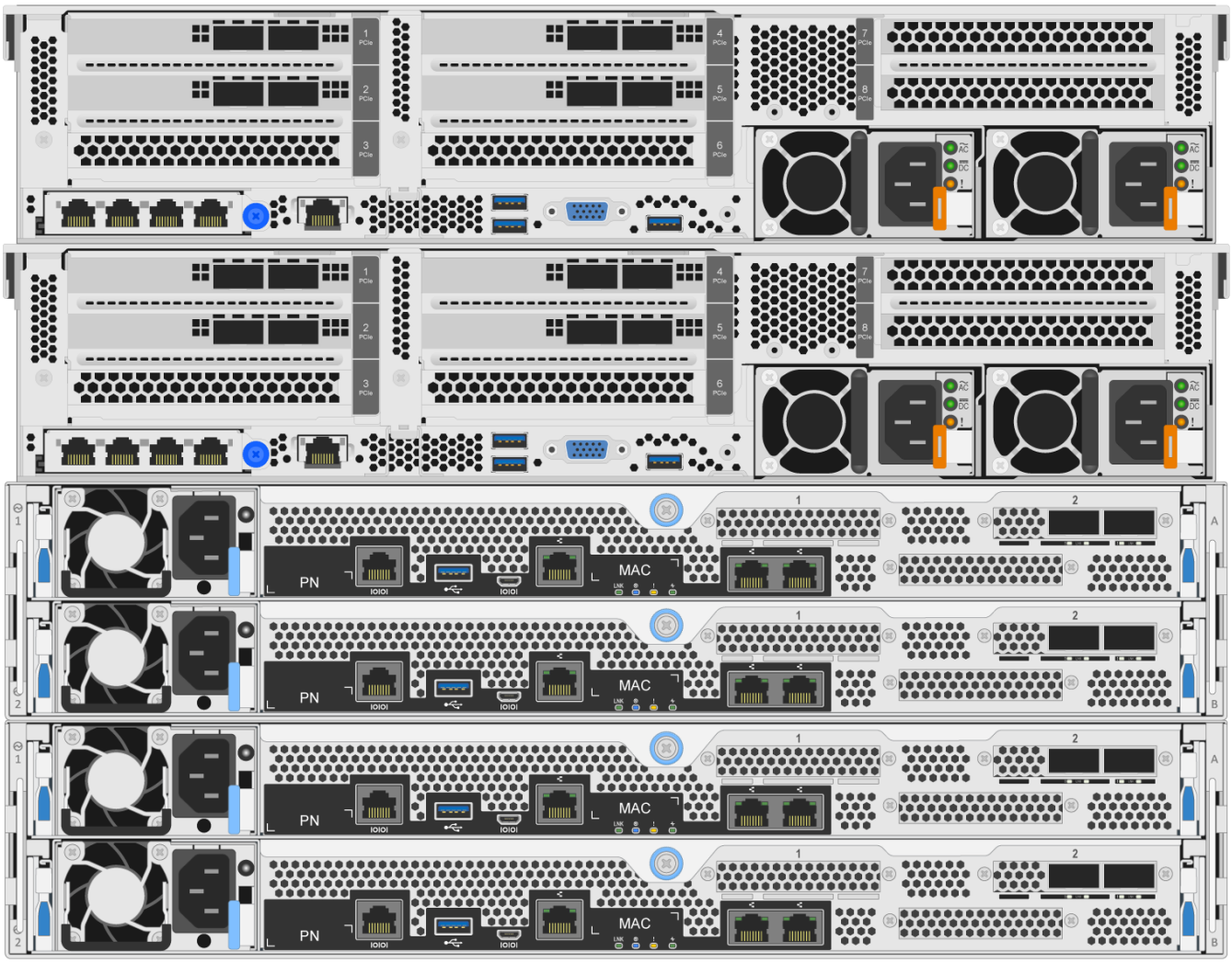
Eine alternative HA-Cluster-Architektur verwendet ein Storage-Fabric (auch als Storage Area Network oder SAN bekannt) zwischen den Datei- und Block-Nodes, um ein symmetrisches Cluster herzustellen. So können BeeGFS-Services auf jedem Datei-Node in einem bestimmten HA-Cluster ausgeführt werden. Da symmetrische Cluster aufgrund der zusätzlichen SAN-Hardware nicht so kostengünstig sind, setzt diese Dokumentation den Einsatz eines asymmetrischen Clusters voraus, der als eine Reihe von einem oder mehreren Bausteinen implementiert wird.

Stellen Sie sicher, dass die gewünschte Dateisystemarchitektur für eine bestimmte BeeGFS-Bereitstellung gut verstanden wird, bevor Sie mit der Installation fortfahren.

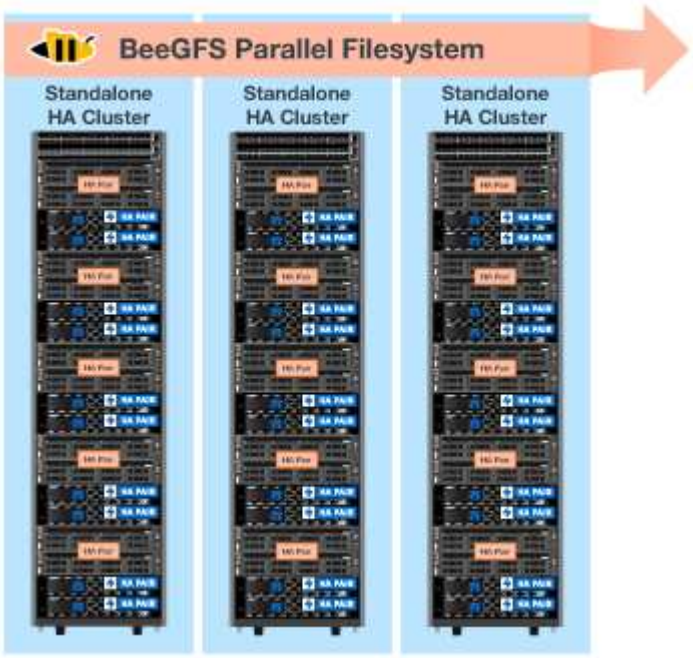
Rack-Hardware

Bei der Planung der Installation ist es wichtig, dass alle Geräte in jedem Baustein in benachbarten Rack-Einheiten verfügbar sind. Als Best Practice empfiehlt es sich, Datei-Nodes sofort über Block-Nodes in jedem Baustein verfügbar zu machen. Befolgen Sie die Dokumentation für die Modelle der Datei und "Block-Storage" Knoten, die Sie verwenden, wenn Sie Schienen und Hardware im Rack installieren.

Beispiel für einen einzelnen Baustein:

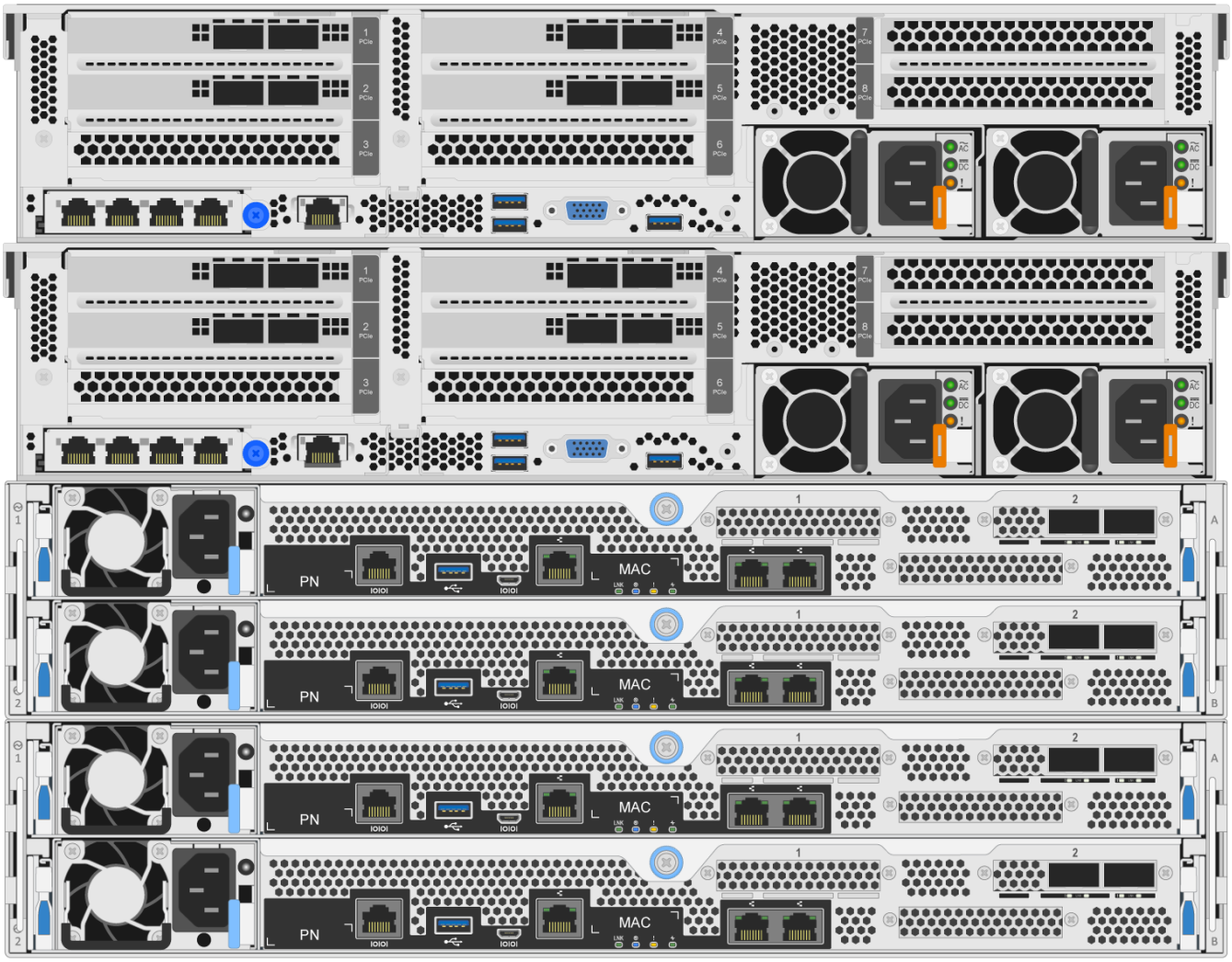


Beispiel für eine große BeeGFS-Installation, bei der es in jedem HA-Cluster mehrere Bausteine und mehrere HA-Cluster im Filesystem gibt:



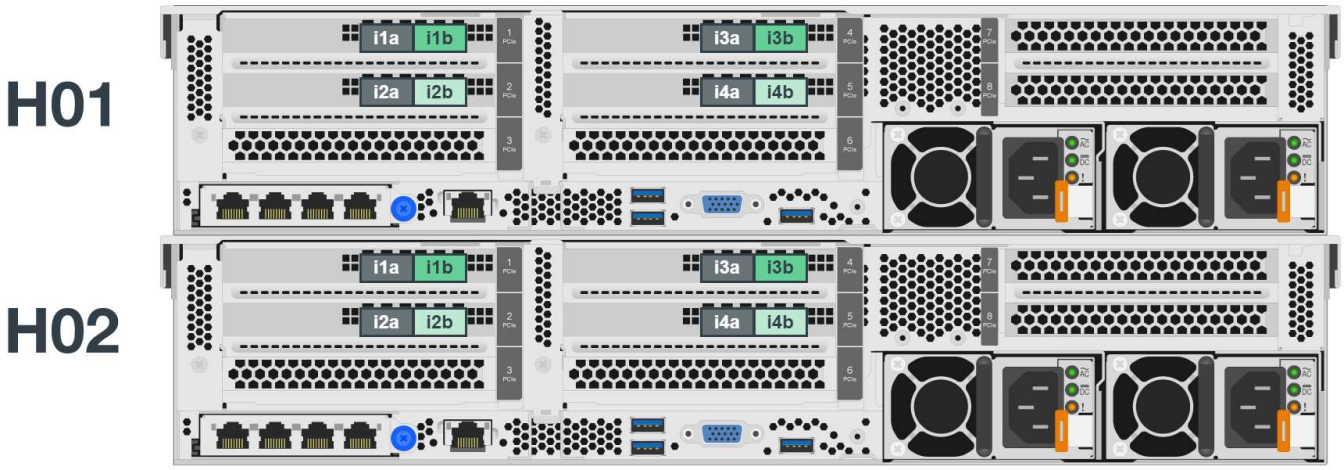
Kabeldatei- und Blockknoten

Sie werden die HIC-Ports der Block-Nodes der E-Series normalerweise mit dem vorgesehenen Host Channel Adapter (für InfiniBand-Protokolle) oder den Host-Bus-Adaptern (für Fibre Channel und andere Protokolle) der Datei-Nodes verbinden. Die genaue Art und Weise, diese Verbindungen herzustellen, hängt von der gewünschten Dateisystemarchitektur ab, hier ist ein Beispiel"Basierend auf BeeGFS der zweiten Generation auf NetApp Verified Architecture":



Dateiknoten mit dem Client-Netzwerk verkabeln

Jeder Datei-Node verfügt über eine bestimmte Anzahl von InfiniBand- oder Ethernet-Ports für BeeGFS-Client-Traffic. Je nach Architektur verfügt jeder Datei-Node über eine oder mehrere Verbindungen zu einem hochperformanten Client-/Storage-Netzwerk, möglicherweise zu mehreren Switches für Redundanz und höhere Bandbreite. Hier sehen Sie ein Beispiel für die Client-Verkabelung mithilfe redundanter Netzwerk-Switches, bei denen die in Dunkelgrün bzw. hellgrün hervorgehobenen Ports mit separaten Switches verbunden sind:



Verbindung zwischen Management-Netzwerk und Stromversorgung

Stellen Sie alle erforderlichen Netzwerkverbindungen für in-Band- und Out-of-Band-Netzwerke her.

Schließen Sie alle Netzteile an, um sicherzustellen, dass jeder Datei- und Block-Knoten Verbindungen zu mehreren Stromverteilungs-Einheiten hat, um Redundanz zu gewährleisten (falls verfügbar).

Datei- und Block-Knoten einrichten

Manuelle Schritte zur Einrichtung von Datei- und Block-Nodes vor der Ausführung von Ansible erforderlich

File-Nodes

Konfigurieren des Baseboard Management Controllers (BMC)

Ein Baseboard Management Controller (BMC), der manchmal als Service-Prozessor bezeichnet wird, ist der generische Name für die Out-of-Band-Management-Funktion, die in verschiedenen Server-Plattformen integriert ist, die Remote-Zugriff bieten können, selbst wenn das Betriebssystem nicht installiert ist oder nicht zugänglich ist. Anbieter vermarkten diese Funktionalität in der Regel mit ihrem eigenen Branding. Auf dem Lenovo SR665 wird beispielsweise der BMC als Lenovo XClarity Controller (XCC) bezeichnet.

Befolgen Sie die Dokumentation des Serveranbieters, um alle erforderlichen Lizenzen für den Zugriff auf diese Funktionalität zu aktivieren und sicherzustellen, dass der BMC mit dem Netzwerk verbunden und für den Remote-Zugriff entsprechend konfiguriert ist.

> Wenn ein BMC-basiertes Fechten mit Redfish gewünscht wird, stellen Sie sicher, dass Redfish aktiviert ist und die BMC-Schnittstelle über das auf dem Dateiknoten installierte Betriebssystem zugänglich ist. Auf dem Netzwerk-Switch kann eine spezielle Konfiguration erforderlich sein, wenn BMC und der Betrieb dieselbe physische Netzwerkschnittstelle nutzen.

Systemeinstellungen Einstellen

Stellen Sie mithilfe der Benutzeroberfläche des System-Setup (BIOS/UEFI) sicher, dass Einstellungen auf maximale Leistung eingestellt sind. Die genauen Einstellungen und optimalen Werte variieren je nach verwendetes Servermodell. Es wird eine Anleitung zur Verfügung gestellt"Verifizierte Datei-Node-Modelle", andernfalls beziehen Sie sich auf die Dokumentation und Best Practices des Serverherstellers, die auf Ihrem Modell basieren.

Installieren Sie ein Betriebssystem

Installieren Sie ein unterstütztes Betriebssystem basierend auf den aufgeführten Dateiknoten"Hier" -Anforderungen. Beachten Sie die nachfolgenden Schritte, die auf Ihrer Linux-Distribution basieren.

Red Hat

Verwenden Sie den Red Hat Subscription Manager, um das System zu registrieren und zu abonnieren, damit die erforderlichen Pakete aus den offiziellen Red Hat-Repositorys installiert werden können und um Updates auf die unterstützte Version von Red Hat zu beschränken: `subscription-manager release --set=<MAJOR_VERSION>.<MINOR_VERSION>` . Anweisungen finden Sie unter "Registrieren und Abonnieren eines RHEL Systems" Und "Einschränken von Aktualisierungen" .

Red hat Repository mit den für hohe Verfügbarkeit erforderlichen Paketen aktivieren:

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

Managementnetzwerk Konfigurieren

Konfigurieren Sie alle erforderlichen Netzwerkschnittstellen für die bandinterne Verwaltung des Betriebssystems. Die genauen Schritte hängen von der jeweiligen Linux-Distribution und der verwendeten Version ab.

> Vergewissern Sie sich, dass SSH aktiviert ist und alle Managementoberflächen über den Ansible Kontroll-Node zugänglich sind.

Aktualisieren der HCA- und HBA-Firmware

Stellen Sie sicher, dass auf allen HBAs und HCAs unterstützte Firmware-Versionen ausgeführt "NetApp Interoperabilitätsmatrix"werden, die auf dem aufgeführt sind, und aktualisieren Sie ggf.. Weitere Empfehlungen für NVIDIA ConnectX Adapter finden Sie "Hier".

Block-Nodes

Befolgen Sie die Schritte zu "Die Inbetriebnahme ist möglich mit E-Series" Um den Managementport an jedem Block Node Controller zu konfigurieren und optional den Namen des Storage-Arrays für jedes System festzulegen.

> Es ist keine zusätzliche Konfiguration erforderlich, die darüber hinaus sicherstellt, dass alle Block-Nodes über den Ansible-Kontroll-Node zugänglich sind. Die verbleibende Systemkonfiguration wird mit Ansible angewendet/gewartet.

Ansible-Steuerungsknoten Einrichten

Richten Sie einen Ansible-Steuerungsknoten ein, um das Dateisystem zu implementieren und zu managen.

Überblick

Ein Ansible-Steuerungsknoten ist eine physische oder virtuelle Linux-Maschine, die zum Verwalten des

Clusters verwendet wird. Er muss folgende Anforderungen erfüllen:

- Lernen Sie die "Anforderungen"Rolle für die BeeGFS HA kennen, einschließlich der installierten Versionen von Ansible, Python und zusätzlichen Python-Paketen.
- Treffen Sie den Beamten "Ansible-Control-Node-Anforderungen" Einschließlich Betriebssystemversionen.
- SSH- und HTTPS-Zugriff auf alle Datei- und Block-Nodes

Detaillierte Installationsschritte finden Sie "Hier".

Definieren Sie das BeeGFS-Dateisystem

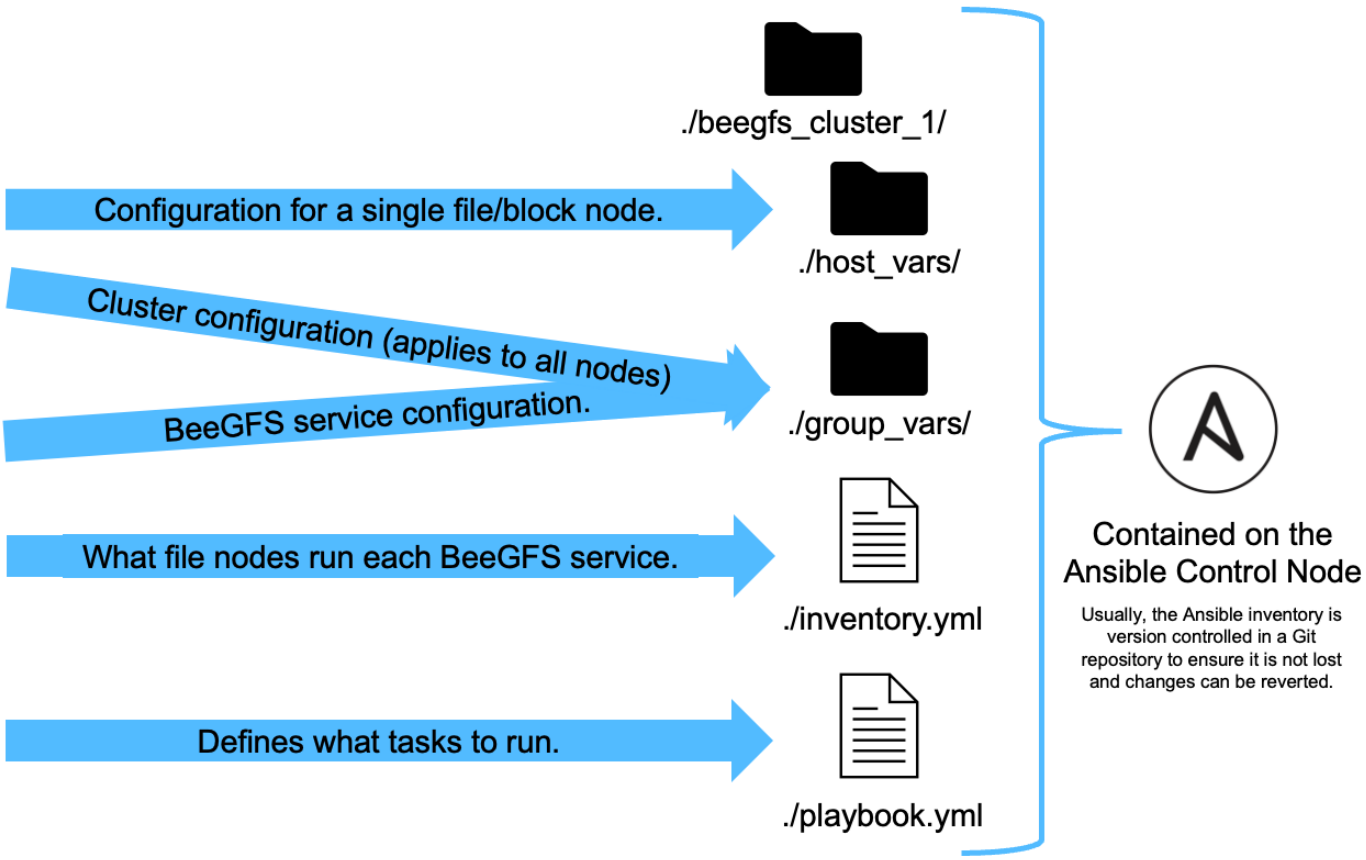
Ansible-Bestandsübersicht

Der Ansible-Bestand ist ein Satz von Konfigurationsdateien, die das gewünschte BeeGFS HA-Cluster definieren.

Überblick

Es wird empfohlen, die standardmäßigen Ansible-Methoden für die Organisation des zu befolgen "Inventar", Einschließlich der Verwendung von "Unterverzeichnisse/Dateien" Anstatt den gesamten Bestand in einer Datei zu speichern.

Der Ansible-Bestand für ein einzelnes BeeGFS HA-Cluster ist wie folgt organisiert:



Da ein einzelnes BeeGFS-Dateisystem mehrere HA-Cluster umfassen kann, können große Installationen mehrere Ansible-Inventare durchführen. Im Allgemeinen wird es nicht empfohlen, mehrere HA-Cluster als einen einzelnen Ansible-Bestand zu definieren, um Probleme zu vermeiden.

Schritte

1. Erstellen Sie auf Ihrem Ansible-Kontroll-Node ein leeres Verzeichnis, das den Ansible-Bestand für das BeeGFS-Cluster enthält, das bereitgestellt werden soll.
   a. Wenn Ihr Filesystem schließlich mehrere HA-Cluster enthalten soll/kann, wird empfohlen, zuerst ein Verzeichnis für das Dateisystem zu erstellen und dann Unterverzeichnisse für den Bestand, der die einzelnen HA-Cluster darstellt, einzutragen. Beispiel:

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

2. Erstellen Sie im Verzeichnis, das den Bestand für den HA-Cluster enthält, den Sie bereitstellen möchten, zwei Verzeichnisse `group_vars` Und `host_vars` Und zwei Dateien `inventory.yml` Und `playbook.yml`.

Die folgenden Abschnitte gehen durch die Definition des Inhalts jeder dieser Dateien.

Planen Sie das Dateisystem

Planen Sie die Filesystem-Implementierung, bevor Sie den Ansible-Bestand aufbauen.

Überblick

Vor der Bereitstellung des Dateisystems sollten Sie festlegen, welche IP-Adressen, Ports und andere Konfigurationen für alle Datei-Nodes, Block-Nodes und BeeGFS-Services im Cluster erforderlich sind. Während die genaue Konfiguration je nach Architektur des Clusters variiert, werden in diesem Abschnitt Best Practices und Schritte definiert, die allgemein anwendbar sind.

Schritte

1. Wenn Sie zum Verbinden von Datei-Nodes mit Block-Nodes ein IP-basiertes Storage-Protokoll (z. B. iSER, iSCSI, NVMe/IB oder NVMe/RoCE) verwenden, füllen Sie für jeden Baustein das folgende Arbeitsblatt aus. Jede direkte Verbindung in einem einzelnen Baustein sollte ein eigenes Subnetz haben, und es sollte keine Überschneidung mit Subnetzen geben, die für die Client-Server-Konnektivität verwendet werden.

Datei-Node	IB-Port	IP-Adresse	Block-Node	IB-Port	Physische IP-Adresse	Virtuelle IP (nur für EF600 mit HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

Wenn Datei- und Block-Nodes in jedem Baustein direkt verbunden sind, können für mehrere Bausteine oft dieselben IPs/Schemas verwendet werden.

2. Füllen Sie unabhängig davon aus, ob Sie InfiniBand oder RDMA over Converged Ethernet (RoCE) für das Storage-Netzwerk verwenden, das folgende Arbeitsblatt aus, um die IP-Bereiche zu ermitteln, die für HA-Cluster-Services, BeeGFS-Fileservices und Clients verwendet werden, um zu kommunizieren:

Zweck	InfiniBand-Port	IP-Adresse oder Bereich
BeeGFS Cluster-IP(s)	<INTERFACE(s)>	<RANGE>
BeeGFS Management	<INTERFACE(s)>	<IP(s)>
BeeGFS-Metadaten	<INTERFACE(s)>	<RANGE>
BeeGFS-Speicherung	<INTERFACE(s)>	<RANGE>
BeeGFS-Clients	<INTERFACE(s)>	<RANGE>

   a. Wenn Sie ein einzelnes IP-Subnetz verwenden, ist nur ein Arbeitsblatt erforderlich. Füllen Sie andernfalls auch ein Arbeitsblatt für das zweite Subnetz aus.

3. Füllen Sie für jeden Baustein im Cluster das folgende Arbeitsblatt aus, um festzulegen, welche BeeGFS-Services ausgeführt werden sollen. Geben Sie für jeden Service die bevorzugten/sekundären Dateiknoten, Netzwerkport, fließende IP(s), NUMA-Zonenzuweisung (falls erforderlich) und welche Block-Nodes für ihre Ziele verwendet werden. Beachten Sie beim Ausfüllen des Arbeitsblatts die folgenden Richtlinien:

   a. Geben Sie BeeGFS-Dienste als entweder an `mgmt.yml`, `meta_<ID>.yml`, Oder `storage_<ID>.yml` Wobei ID eine eindeutige Nummer für alle BeeGFS-Dienste dieses Typs in diesem Dateisystem darstellt. Dieses Übereinkommen erleichtert die Rückverweisen auf dieses Arbeitsblatt in den nachfolgenden Abschnitten, während Dateien für die Konfiguration der einzelnen Dienste erstellt werden.

   b. Ports für BeeGFS-Dienste müssen nur in einem bestimmten Baustein einzigartig sein. Stellen Sie sicher, dass Dienste mit derselben Portnummer nicht jemals auf demselben Dateiknoten ausgeführt werden können, um Portkonflikte zu vermeiden.

   c. Bei Bedarf können Services Volumes von mehr als einem Block-Node und/oder Storage-Pool nutzen (und nicht alle Volumes müssen Eigentum desselben Controllers sein). Zudem können mehrere Services denselben Block-Node und/oder dieselbe Storage-Pool-Konfiguration nutzen (einzelne Volumes werden in einem späteren Abschnitt definiert).

BeeGFS-Dienst (Dateiname)	File-Nodes	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
<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>

Weitere Details zu Standardkonventionen, Best Practices und ausgefüllten Arbeitsblättern finden Sie in den "Best Practices in sich vereint""Definieren Sie BeeGFS-Bausteine"Abschnitten und der BeeGFS on NetApp Verified Architecture.

Datei- und Blockknoten definieren

Konfigurieren Einzelner Dateiknoten

Legen Sie die Konfiguration für einzelne Datei-Nodes mithilfe von Host-Variablen fest (Host_vars).

Überblick

In diesem Abschnitt wird das Ausfüllen von erläutert `host_vars/<FILE_NODE_HOSTNAME>.yml` Datei für jeden Datei-Node im Cluster. Diese Dateien sollten nur die für einen bestimmten Dateiknoten spezifische Konfiguration enthalten. Hierzu zählen folgende allgemein:

- Die Definition der IP oder des Hostnamen Ansible sollte für die Verbindung mit dem Node verwendet werden.
- Konfiguration zusätzlicher Schnittstellen und Cluster-IPs, die für HA-Cluster-Services (Pacemaker und Corosync) zur Kommunikation mit anderen Datei-Nodes verwendet werden Standardmäßig verwenden diese Dienste dasselbe Netzwerk wie die Managementoberfläche, aber für Redundanz sollten zusätzliche Schnittstellen verfügbar sein. Es ist üblich, zusätzliche IPs im Storage-Netzwerk zu definieren, ohne dass ein zusätzliches Cluster- oder Management-Netzwerk erforderlich ist.
  - Die Performance aller Netzwerke, die für die Cluster-Kommunikation verwendet werden, ist für die Performance des Filesystems nicht von entscheidender Bedeutung. Bei der Standardkonfiguration des Clusters bietet ein Netzwerk mit mindestens 1 GB/s im Allgemeinen ausreichende Performance für Cluster-Vorgänge, z. B. die Synchronisierung von Node-Status und die Koordinierung von Änderungen des Clusterressourcenstatus. Langsame/überlastete Netzwerke können dazu führen, dass Änderungen des Ressourcenzustands länger dauern als üblich, und in extremen Fällen können Nodes aus dem Cluster entfernt werden, wenn sie in einem angemessenen Zeitrahmen keine Herzschläge senden können.
- Konfigurieren von Schnittstellen, die für die Verbindung zu Block-Nodes über das gewünschte Protokoll verwendet werden (z. B. iSCSI/iSER, NVMe/IB, NVMe/RoCE, FCP usw.)

Schritte

Wenn Sie auf das im "Planen Sie das Dateisystem" Abschnitt definierte IP-Adressierungsschema verweisen, erstellen Sie für jeden Dateiknoten im Cluster eine Datei `host_vars/<FILE_NODE_HOSTNAME>/yml` und füllen Sie sie wie folgt aus:

1. Geben Sie oben die IP oder den Hostnamen an, den Ansible für den Node mit SSH verwenden und verwalten soll:

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

2. Konfigurieren Sie zusätzliche IPs, die für den Cluster-Datenverkehr verwendet werden können:
   a. Wenn der Netzwerktyp ist "InfiniBand (mit IPoIB)":

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

b. Wenn der Netzwerktyp ist "RDMA über Converged Ethernet (RoCE)":

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

c. Wenn der Netzwerktyp ist "Ethernet (nur TCP, kein 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. Geben Sie an, welche IPs für Cluster-Datenverkehr verwendet werden sollen, wobei die bevorzugten IPs höher aufgeführt sind:

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

IPS, die in Schritt zwei konfiguriert sind, werden nicht als Cluster-IPs verwendet, es sei denn, sie sind im enthalten `beegfs_ha_cluster_node_ips` Liste. So können mithilfe von Ansible zusätzliche IPs/Schnittstellen konfiguriert werden, die bei Bedarf für andere Zwecke verwendet werden können.

4. Wenn der Datei-Node über ein IP-basiertes Protokoll mit Block-Nodes kommunizieren muss, müssen IPs auf der entsprechenden Schnittstelle konfiguriert werden. Für dieses installierte/konfigurierte Protokoll sind alle Pakete erforderlich.

a. Bei Verwendung von "ISCSI":

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

b. Bei Verwendung von "ISER":

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

c. Bei Verwendung von "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. Bei Verwendung von "NVMe/RoCE":

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

e. Andere Protokolle:

i. Bei Verwendung von "NVMe/FC", Die Konfiguration einzelner Schnittstellen ist nicht erforderlich. Die BeeGFS-Cluster-Implementierung erkennt das Protokoll automatisch und installiert/konfiguriert die Anforderungen nach Bedarf. Wenn Sie eine Fabric zum Verbinden von Datei- und Block-Nodes verwenden, stellen Sie sicher, dass die Switches im Rahmen der Best Practices von NetApp und dem Switch-Anbieter ordnungsgemäß begrenzt sind.

ii. Bei der Verwendung von FCP oder SAS muss keine zusätzliche Software installiert oder konfiguriert werden. Wenn Sie FCP verwenden, stellen Sie sicher, dass die Switches Folgendes ordnungsgemäß begrenzt sind "NetApp" Und die Best Practices Ihres Switch-Anbieters berücksichtigen.

iii. Die Verwendung von IB-SRP wird derzeit nicht empfohlen. NVMe/IB oder iSER nutzen, je nachdem, was die Block-Nodes der E-Series unterstützen.

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen einzelnen Dateiknoten darstellt.

Erweitert: Zwischen Ethernet- und InfiniBand-Modus können NVIDIA ConnectX VPI-Adapter eingesetzt werden

NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) Adapter unterstützen sowohl InfiniBand als auch Ethernet als Transportebene. Das Umschalten zwischen den Modi wird nicht automatisch ausgehandelt und muss mit dem in enthaltenen Werkzeug konfiguriert werden <code>mstconfig</code> <code>mstflint</code>, einem Open-Source-Paket, das Teil des ist "NVIDIA Firmare Tools (MFT)". Das Ändern des Modus der Adapter muss nur einmal vorgenommen werden. Dies kann manuell vorgenommen oder als Teil aller Schnittstellen, die mithilfe des Abschnitts des Bestands konfiguriert wurden, in das Ansible-Inventar aufgenommen <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> werden, um es automatisch prüfen/anwenden zu lassen.

So kann beispielsweise die aktuelle Schnittstellenspannung im InfiniBand-Modus in Ethernet geändert werden, damit sie für RoCE verwendet werden kann:

1. Für jede Schnittstelle, die Sie angeben möchten `mstconfig` Als Zuordnung (oder Wörterbuch), das angibt `LINK_TYPE_P<N>` Wo `<N>` Wird durch die Anschlussnummer des HCA für die Schnittstelle bestimmt. Der `<N>` Wert kann durch Ausführen bestimmt werden `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` Und fügen Sie 1 zur letzten Nummer aus dem PCI-Steckplatznamen hinzu und konvertieren Sie auf dezimal.
   a. Beispiel angegeben `PCI_SLOT_NAME=0000:2f:00.2` (2 + 1 → HCA-Port 3) → `LINK_TYPE_P3: eth`:

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

Weitere Details finden Sie im "Dokumentation der NetApp E-Series Host-Sammlung" Für den Schnittstellentyp/das Protokoll, das Sie verwenden.

Konfigurieren Einzelner Blockknoten

Legen Sie die Konfiguration für einzelne Block-Nodes mithilfe von Host-Variablen fest (Host_vars).

Überblick

In diesem Abschnitt wird das Ausfüllen von erläutert `host_vars/<BLOCK_NODE_HOSTNAME>.yml` Datei für jeden Block-Node im Cluster. Diese Dateien sollten nur die Konfiguration enthalten, die für einen bestimmten Block-Node eindeutig ist. Hierzu zählen folgende allgemein:

- Der Systemname (wie in System Manager angezeigt).
- Die HTTPS-URL für einen der Controller (wird zum Verwalten des Systems mit seiner REST-API verwendet).
- Welche Storage-Protokoll-Datei-Nodes verwenden für die Verbindung zu diesem Block-Node?

- Konfigurieren von Ports für die Host-Schnittstelle (HIC), z. B. IP-Adressen (falls erforderlich)

Schritte

Wenn Sie auf das im "Planen Sie das Dateisystem" Abschnitt definierte IP-Adressierungsschema verweisen, erstellen Sie für jeden Block-Node im Cluster eine Datei `host_vars/<BLOCK_NODE_HOSTNAME>/yml` und füllen Sie sie wie folgt aus:

1. Geben Sie oben den Systemnamen und die HTTPS-URL für einen Controller an:

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

2. Wählen Sie die aus "Protokoll" Dateiknoten werden für die Verbindung zu diesem Block-Knoten verwendet:
   a. Unterstützte Protokolle: `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

```
eseries_initiator_protocol: <PROTOCOL>
```

3. Je nach verwendetem Protokoll erfordern die HIC-Ports unter Umständen zusätzliche Konfigurationen. Bei Bedarf sollte die HIC-Port-Konfiguration definiert werden, sodass der oberste Eintrag in der Konfiguration für jeden Controller dem physischen, am meisten linken Port auf jedem Controller entspricht, und der untere Port dem fast rechten Port. Alle Ports erfordern eine gültige Konfiguration, auch wenn sie derzeit nicht verwendet werden.

   ⓘ Wenn Sie HDR (200 GB) InfiniBand oder 200 GB RoCE mit EF600 Block-Nodes verwenden, sehen Sie den folgenden Abschnitt.

   a. Für 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. Für 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. Für 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. Für 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. FC- und SAS-Protokolle erfordern keine zusätzliche Konfiguration. SRP wird nicht richtig empfohlen.

Weitere Optionen zum Konfigurieren von HIC-Ports und Hostprotokollen, einschließlich der Möglichkeit zum Konfigurieren von iSCSI-CHAP finden Sie im "Dokumentation" In der SANtricity Kollektion enthalten. Hinweis: Bei der Bereitstellung von BeeGFS werden der Speicherpool, die Volume-Konfiguration und andere Aspekte des Bereitstellungsspeicher an anderer Stelle konfiguriert und in dieser Datei nicht definiert.

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen einzelnen Block-Knoten darstellt.

Mit HDR (200 GB) InfiniBand oder 200 GB RoCE mit NetApp EF600 Block-Nodes:

Um HDR (200 GB) InfiniBand mit der EF600 zu verwenden, muss für jeden physischen Port eine zweite „virtuelle“ IP konfiguriert werden. Nachfolgend sehen Sie ein Beispiel für die korrekte Konfiguration eines EF600 mit Dual-Port InfiniBand HDR HIC:

```
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)
```

Festlegen Der Konfiguration Des Gemeinsamen Dateiknotens

Geben Sie unter Verwendung von Gruppenvariablen (Group_vars) die Konfiguration allgemeiner Dateiknoten an.

Überblick

Konfiguration, die auf alle Datei-Nodes Apfel soll, wird bei definiert `group_vars/ha_cluster.yml`. Dazu gehören in der Regel:

- Details zur Verbindung und Anmeldung zu den einzelnen Dateiknoten.
- Gängige Netzwerkkonfiguration.
- Gibt an, ob ein automatischer Neustart zulässig ist.
- Wie Firewall- und selinux-Status konfiguriert werden sollen.
- Cluster-Konfiguration mit Warn- und Fechten
- Performance-Optimierung:
- Allgemeine BeeGFS-Servicekonfiguration.

Die in dieser Datei festgelegten Optionen können auch auf einzelnen Datei-Nodes definiert werden, z. B. wenn gemischte Hardware-Modelle verwendet werden oder Sie unterschiedliche Passwörter für jeden Knoten haben. Die Konfiguration auf einzelnen Datei-Knoten hat Vorrang vor der Konfiguration in dieser Datei.

Schritte

Erstellen Sie die Datei `group_vars/ha_cluster.yml` Und füllen Sie es wie folgt aus:

1. Geben Sie an, wie sich der Ansible Control-Node mit den Remote-Hosts authentifizieren soll:

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

Speichern Sie Passwörter insbesondere für Produktionsumgebungen nicht im Klartext. Verwenden Sie stattdessen Ansible Vault (siehe "Verschlüsseln von Inhalten mit Ansible Vault") Oder der `--ask-become-pass` Option beim Ausführen des Playbooks. Wenn der `ansible_ssh_user` Ist bereits root, dann können Sie optional auslassen `ansible_become_password`.

2. Wenn Sie statische IPs in ethernet- oder InfiniBand-Schnittstellen konfigurieren (zum Beispiel Cluster-IPs) und mehrere Schnittstellen im gleichen IP-Subnetz sind (z. B. wenn ib0 192.168.1.10/24 verwendet und ib1 192.168.1.11/24 verwendet), Zusätzliche IP-Routing-Tabellen und -Regeln müssen so eingerichtet sein, dass Multi-Homed-Unterstützung ordnungsgemäß funktioniert. Aktivieren Sie einfach den mitgelieferten Konfigurationshaken für Netzwerkschnittstellen wie folgt:

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

3. Bei der Implementierung des Clusters müssen je nach Storage-Protokoll Nodes neu gestartet werden, um die Erkennung von Remote-Block-Geräten (E-Series Volumes) zu erleichtern oder andere Aspekte der Konfiguration anzuwenden. Standardmäßig werden vor dem Neubooten von Nodes angezeigt. Sie können Nodes jedoch durch Angabe des folgenden Verfahrens automatisch neu starten:

```
eseries_common_allow_host_reboot: true
```

   a. Standardmäßig nach einem Neustart, um sicherzustellen, dass Block-Geräte und andere Services bereit sind Ansible wartet, bis das System `default.target` Erreicht wird, bevor die Implementierung fortgesetzt wird. In manchen Szenarien, in denen NVMe/IB verwendet wird, ist dies möglicherweise nicht lang genug, um Remote-Geräte zu initialisieren, zu erkennen und eine Verbindung zu herstellen. Dies kann dazu führen, dass die automatisierte Implementierung vorzeitig ausfällt und ausfällt. Um dies bei der Nutzung von NVMe/IB zu vermeiden, müssen Sie außerdem Folgendes definieren:

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

4. Für die Kommunikation mit BeeGFS und HA-Cluster-Services sind mehrere Firewall-Ports erforderlich. Wenn Sie die Firwewall nicht manuell konfigurieren möchten (nicht empfohlen), geben Sie Folgendes an, damit erforderliche Firewall-Zonen erstellt und Ports automatisch geöffnet werden:

```
beegfs_ha_firewall_configure: True
```

5. Derzeit wird SELinux nicht unterstützt, und es wird empfohlen, den Status auf deaktiviert zu setzen, um Konflikte zu vermeiden (insbesondere, wenn RDMA verwendet wird). Stellen Sie Folgendes ein, um sicherzustellen, dass SELinux deaktiviert ist:

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

6. Konfigurieren Sie die Authentifizierung so, dass Dateiknoten kommunizieren können und passen Sie die Standardeinstellungen entsprechend Ihren Unternehmensrichtlinien an:

```
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. "Planen Sie das Dateisystem"Legen Sie auf der Grundlage des Abschnitts die BeeGFS-Management-IP für dieses Dateisystem fest:

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

> Während scheinbar redundant, `beegfs_ha_mgmtd_floating_ip` Ist wichtig, wenn Sie das BeeGFS-Dateisystem über einen einzelnen HA-Cluster hinaus skalieren. Nachfolgende HA-Cluster werden ohne zusätzlichen BeeGFS-Managementservice bereitgestellt und Punkt am Managementservice des ersten Clusters.

8. Aktivieren Sie bei Bedarf E-Mail-Alarme:

```
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. Es wird dringend empfohlen, Fechten zu aktivieren, da bei Ausfall des primären Knotens Services vom Starten auf sekundären Knoten blockiert werden können.
   a. Aktivieren Sie das globale Fechten, indem Sie Folgendes angeben:

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

      i. Hinweis: Bei Bedarf können auch alle unterstützten "Cluster-Eigenschaft" Daten hier angegeben werden. Diese Anpassungen sind in der Regel nicht notwendig, da die BeeGFS HA-Rolle mit einer Reihe gut getesteter ausgeliefert "Standardwerte"wird.

   b. Wählen Sie anschließend einen Fechten-Agent aus und konfigurieren Sie ihn:
      i. OPTION 1: Ermöglicht das Fechten mit APC Power Distribution Units (PDUs):

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

      ii. OPTION 2: Ermöglicht das Fechten mit den vom Lenovo XCC (und anderen BMCs) bereitgestellten Redfish APIs:

```
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. Weitere Informationen zum Konfigurieren anderer Fencing-Agenten finden Sie im "Red Hat-Dokumentation".

10. Die BeeGFS HA-Rolle kann viele verschiedene Tuning-Parameter anwenden, um die Leistung weiter zu optimieren. Dazu gehören unter anderem die Optimierung der Kernel-Speicherauslastung und die E/A von Blockgeräten. Die Rolle wird mit einem angemessenen Satz von basierend auf Tests mit NetApp E-Series Block-Nodes ausgeliefert "Standardwerte" . Diese werden standardmäßig jedoch nicht angewendet, es sei denn, Sie geben Folgendes an:

```
beegfs_ha_enable_performance_tuning: True
```

a. Geben Sie bei Bedarf auch hier Änderungen an der Standard-Performance-Optimierung an. Weitere Informationen finden Sie in der vollständigen "Parameter für die Performance-Optimierung" Dokumentation.

11. Damit schwebende IP-Adressen (manchmal auch als logische Schnittstellen bekannt), die für BeeGFS-Dienste verwendet werden, zwischen Datei-Nodes ausfallen können, müssen alle Netzwerkschnittstellen konsistent benannt werden. Standardmäßig werden Netzwerkschnittstellennamen vom Kernel generiert, was nicht garantiert ist, dass konsistente Namen generiert werden, auch bei identischen Servermodellen mit Netzwerkadaptern, die in denselben PCIe-Steckplätzen installiert sind. Dies ist auch nützlich, wenn Vorräte erstellt werden, bevor das Gerät bereitgestellt wird und generierte Schnittstellennamen bekannt sind. Um konsistente Gerätenamen auf der Grundlage eines Blockdiagramms des Servers oder sicherzustellen `lshw -class network -businfo` Ausgabe, geben Sie die gewünschte PCIe-Adresse-zu-logische Schnittstellenzuordnung wie folgt an:

a. Für InfiniBand (IPoIB)-Netzwerkschnittstellen:

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

b. Bei Ethernet-Netzwerkschnittstellen:

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

Um Konflikte zu vermeiden, wenn Schnittstellen umbenannt werden (um zu verhindern, dass sie umbenannt werden), sollten Sie keine möglichen Standardnamen wie eth0, ens9f0, ib0 oder ibs4f0 verwenden. Eine häufige Namenskonvention besteht darin, „e“ oder „i“ für Ethernet oder InfiniBand zu verwenden, gefolgt von der PCIe-Steckplatznummer und einem Buchstaben zur Angabe des Ports. Zum Beispiel wäre der zweite Port eines InfiniBand-Adapters, der in Steckplatz 3 installiert ist: i3b.

Wenn Sie ein verifiziertes Datei-Node-Modell verwenden, klicken Sie auf "Hier" Beispiel für Zuordnungen von PCIe-Adressen zu logischen Ports

12. Geben Sie optional die Konfiguration an, die für alle BeeGFS-Dienste im Cluster gelten soll. Die Standardkonfigurationswerte können gefunden werden "Hier", und die Konfiguration pro Service wird an anderer Stelle angegeben:

    a. BeeGFS Management-Service:

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

    b. BeeGFS Metadata Services:

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

    c. BeeGFS Storage-Services:

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

13. Ab BeeGFS 7.2.7 und 7.3.1 "Verbindungsauthentifizierung" Muss konfiguriert oder explizit deaktiviert werden. Es gibt einige Konfigurationsmöglichkeiten, die mit der Ansible-basierten Implementierung konfiguriert werden können:

    a. Standardmäßig konfiguriert die Bereitstellung die Verbindungsauthentifizierung automatisch und erstellt ein `connauthfile` Die auf alle Datei-Nodes verteilt und mit den BeeGFS-Diensten verwendet werden. Diese Datei wird auch auf dem Ansible-Steuerungsknoten in abgelegt/gepflegt `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` Wo Sie Daten für die Wiederverwendung mit Clients, die auf dieses Filesystem zugreifen müssen, aufbewahren (sicher).

        i. Zum Generieren eines neuen Schlüssels angeben `-e "beegfs_ha_conn_auth_force_new=True` Wenn Sie das Ansible-Playbook ausführen. Beachten Sie, dass dies bei einem ignoriert wird `beegfs_ha_conn_auth_secret` Definiert ist.

        ii. Weitere Optionen finden Sie in der vollständigen Liste der Standardwerte, die im enthalten

"BeeGFS HA-Rolle"sind.

b. Ein benutzerdefiniertes Geheimnis kann verwendet werden, indem Sie Folgendes in definieren `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. Die Verbindungsauthentifizierung kann vollständig deaktiviert werden (NICHT empfohlen):

```
beegfs_ha_conn_auth_enabled: false
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die die allgemeine Konfiguration des Dateiknoten darstellt.

Verwendung von HDR (200 GB) InfiniBand mit NetApp EF600 Block-Nodes:

Um HDR (200 GB) InfiniBand mit der EF600 zu verwenden, muss der Subnetzmanager die Virtualisierung unterstützen. Wenn Datei- und Block-Knoten über einen Switch verbunden sind, muss dies im Subnetz Manager für die Gesamtstruktur aktiviert sein.

Wenn Block- und Datei-Nodes direkt über InfiniBand verbunden sind, `opensm` muss auf jedem Datei-Node für jede Schnittstelle, die direkt mit einem Block-Node verbunden ist, eine Instanz von konfiguriert werden. Dies geschieht durch Angabe von `configure: true` wann "Konfigurieren von File-Node-Storage-Schnittstellen".

Derzeit unterstützt die Inbox-Version von `opensm` , die mit unterstützten Linux-Distributionen ausgeliefert wurde, keine Virtualisierung. Stattdessen ist es erforderlich, dass Sie die Version von über die NVIDIA OpenFabrics Enterprise Distribution (OFED) installieren und konfigurieren `opensm` . Obwohl die Implementierung mit Ansible weiterhin unterstützt wird, sind einige weitere Schritte erforderlich:

1. Laden Sie die Pakete für die Version von OpenSM, die im Abschnitt von der NVIDIA-Website aufgeführt sind, mithilfe von Curl oder Ihrem gewünschten Tool in das Verzeichnis herunter "Technologieanforderungen erfüllt" `<INVENTORY>/packages/` . Beispiel:

```
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. Unter `group_vars/ha_cluster.yml` Definieren Sie die folgende Konfiguration:

```
### 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"
```

Festlegen Der Konfiguration Allgemeiner Blockknoten

Geben Sie unter Verwendung von Gruppenvariablen (Group_vars) die allgemeine Konfiguration von Blockknoten an.

Überblick

Die Konfiguration, die auf alle Block-Nodes Apfel soll, wird auf definiert `group_vars/eseries_storage_systems.yml`. Dazu gehören in der Regel:

- Details dazu, wie der Ansible-Kontroll-Node mit als Block-Nodes verwendeten E-Series Storage-Systemen verbunden werden soll.

- Welche Firmware-, NVSRAM- und Laufwerk-Firmware-Versionen die Nodes ausführen sollten.
- Globale Konfiguration einschließlich Cache-Einstellungen, Host-Konfiguration und Einstellungen für die Bereitstellung von Volumes

> Die in dieser Datei festgelegten Optionen können auch auf einzelnen Block-Nodes definiert werden, z. B. wenn gemischte Hardware-Modelle verwendet werden oder Sie unterschiedliche Passwörter für jeden Knoten haben. Die Konfiguration auf einzelnen Block-Knoten hat Vorrang vor der Konfiguration in dieser Datei.

Schritte

Erstellen Sie die Datei `group_vars/eseries_storage_systems.yml` Und füllen Sie es wie folgt aus:

1. Ansible verwendet SSH nicht für die Verbindung mit Block-Nodes und verwendet stattdessen REST-APIs. Um dies zu erreichen, müssen wir Folgendes festlegen:

```
ansible_connection: local
```

2. Geben Sie den Benutzernamen und das Passwort an, um jeden Knoten zu verwalten. Der Benutzername kann optional ausgelassen werden (und wird standardmäßig auf admin gesetzt), andernfalls können Sie jedes Konto mit Administratorrechten angeben. Geben Sie außerdem an, ob SSL-Zertifikate überprüft oder ignoriert werden sollen:

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

> Es wird nicht empfohlen, Kennwörter im Klartext zu verwenden. Verwenden Sie einen Ansible-Vault, oder stellen Sie die bereit `eseries_system_password` Wenn Sie Ansible mit --extra-Vars verwenden.

3. Geben Sie optional an, welche Controller-Firmware, NVSRAM und Laufwerk-Firmware auf den Nodes installiert werden soll. Diese müssen auf das heruntergeladen werden `packages/` Verzeichnis vor der Ausführung von Ansible. NVSRAM und E-Series Controller Firmware können heruntergeladen werden "Hier" Und Laufwerk-Firmware "Hier":

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

Wenn diese Konfiguration angegeben wird, aktualisiert Ansible automatisch alle Firmware einschließlich des Neubootens von Controllern (falls erforderlich), ohne zusätzliche Eingabeaufforderungen. Dies wird für BeeGFS/Host-I/O voraussichtlich ohne Unterbrechung ausgeführt, kann jedoch zu einer vorübergehenden Abnahme der Performance führen.

4. Passen Sie die Standardeinstellungen für die globale Systemkonfiguration an. Die hier aufgeführten Optionen und Werte werden häufig für BeeGFS auf NetApp empfohlen, können jedoch bei Bedarf angepasst werden:

```
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. Konfigurieren Sie die Standardeinstellungen für die globale Volume-Bereitstellung. Die hier aufgeführten Optionen und Werte werden häufig für BeeGFS auf NetApp empfohlen, können jedoch bei Bedarf angepasst werden:

```
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. Passen Sie bei Bedarf die Reihenfolge an, in der Ansible Laufwerke für Storage Pools und Volume-Gruppen wählt, und berücksichtigen Sie die folgenden Best Practices:

   a. Nennen Sie alle (möglicherweise kleineren) Laufwerke, die zuerst für Management- und/oder Metadaten-Volumes verwendet werden sollten, und Storage Volumes zuletzt.

b. Stellen Sie sicher, dass Sie die Reihenfolge der Festplattenauswahl auf der Grundlage der Modelle für Festplatten-Shelfs/Festplattengehäuse ausgleichen, um die Laufwerksauswahl über verfügbare Laufwerkskanäle auszugleichen. Beispielsweise befinden sich Laufwerke 0-11 mit der EF600 und ohne Erweiterungen auf Laufwerkskanal 1 und Laufwerke 12-23 auf dem Laufwerkskanal. Daher ist eine Strategie zur Balance der Antriebsauswahl zu wählen `disk shelf:drive` 99:0, 99:23, 99:1, 99:22 usw. Wenn mehr als ein Gehäuse vorhanden ist, steht die erste Ziffer für die Laufwerk-Shelf-ID.

```
# 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"
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die die allgemeine Block-Node-Konfiguration darstellt.

BeeGFS-Dienste definieren

Definieren Sie den BeeGFS-Managementdienst

BeeGFS-Dienste werden mit Gruppenvariablen (Group_vars) konfiguriert.

Überblick

In diesem Abschnitt wird die Definition des BeeGFS-Managementservice erläutert. In den HA-Clustern für ein bestimmtes Dateisystem sollte nur ein Service dieses Typs vorhanden sein. Die Konfiguration dieses Services umfasst die folgenden Punkte:

- Der Servicetyp (Management).
- Definieren von Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen.
- Konfiguration einer oder mehrerer fließender IPs (logische Schnittstellen), an denen dieser Service erreicht werden kann.
- Geben Sie an, wo/wie ein Volume Daten für diesen Service speichern soll (das BeeGFS-Managementziel).

Schritte

Erstellen Sie eine neue Datei `group_vars/mgmt.yml`, und verweisen Sie auf den "Planen Sie das Dateisystem" Abschnitt. Füllen Sie diese wie folgt aus:

1. Geben Sie diese Datei an, um die Konfiguration für einen BeeGFS-Managementdienst anzuzeigen:

```
beegfs_service: management
```

2. Definieren Sie alle Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen. Dies ist in der Regel nicht für den Management-Service erforderlich, es sei denn, Sie müssen Quoten aktivieren, jedoch alle unterstützten Konfigurationsparameter von `beegfs-mgmtd.conf` Kann enthalten sein. Beachten Sie, dass die folgenden Parameter automatisch/an anderer Stelle konfiguriert werden und hier nicht angegeben werden sollten: `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, und `connNetFilterFile`.

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

3. Konfigurieren Sie eine oder mehrere unverankerte IPs, die andere Dienste und Clients verwenden, um eine Verbindung zu diesem Dienst herzustellen (dadurch wird das BeeGFS automatisch festgelegt `connInterfacesFile` Option):

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

4. Geben Sie optional ein oder mehrere zulässige IP-Subnetze an, die für die ausgehende Kommunikation verwendet werden können (dadurch wird automatisch das BeeGFS eingestellt `connNetFilterFile` Option):

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

5. Geben Sie das BeeGFS-Managementziel an, auf dem dieser Service Daten gemäß den folgenden Richtlinien speichert:

   a. Für mehrere BeeGFS Services/Ziele kann derselbe Speicherpool oder Volume-Gruppenname verwendet werden. Stellen Sie einfach sicher, dass er dasselbe verwendet `name`, `raid_level`, `criteria_*`, und `common_*` Konfiguration für jede einzelne (die für jeden Service aufgeführten Volumes sollten unterschiedlich sein).

   b. Volume-Größen sollten als Prozentsatz der Storage-Pool/Volume-Gruppe angegeben werden. Die Summe sollte bei allen Services/Volumes, die über einen bestimmten Storage-Pool/Volume-Gruppe verfügen, nicht mehr als 100 übersteigen. Hinweis: Bei der Verwendung von SSDs wird empfohlen, freien Speicherplatz in der Volume-Gruppe zu belassen, um die SSD-Performance und den SSD-Verschleiß "Hier"zu maximieren (klicken Sie für weitere Details).

   c. Klicken Sie Auf "Hier" Eine vollständige Liste der für das verfügbaren Konfigurationsoptionen finden Sie unter `eseries_storage_pool_configuration`. Notieren Sie einige Optionen wie z. B. `state`, `host`, `host_type`, `workload_name`, und `workload_metadata` Und Volume-Namen werden automatisch generiert und sollten hier nicht angegeben werden.

```
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
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen BeeGFS-Managementdienst darstellt.

Definieren Sie den BeeGFS-Metadatendienst

BeeGFS-Dienste werden mit Gruppenvariablen (Group_vars) konfiguriert.

Überblick

In diesem Abschnitt wird die Definition des BeeGFS-Metadatendienstes erläutert. In den HA-Clustern für ein bestimmtes Dateisystem sollte mindestens ein Service dieses Typs vorhanden sein. Die Konfiguration dieses Services umfasst die folgenden Punkte:

- Der Servicetyp (Metadaten).
- Definieren von Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen.
- Konfiguration einer oder mehrerer fließender IPs (logische Schnittstellen), an denen dieser Service erreicht werden kann.
- Festlegen, wo/wie ein Volume Daten für diesen Service speichern soll (das BeeGFS-Metadatenziel).

Schritte

"Planen Sie das Dateisystem"Erstellen Sie `group_vars/meta_<ID>.yml` für jeden Metadatendienst im Cluster eine Datei unter, und füllen Sie sie wie folgt aus, um auf den Abschnitt zu verweisen:

1. Geben Sie an, dass diese Datei die Konfiguration für einen BeeGFS-Metadatendienst darstellt:

   ```
   beegfs_service: metadata
   ```

2. Definieren Sie alle Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen. Mindestens müssen Sie den gewünschten TCP- und UDP-Port angeben, jedoch alle unterstützten Konfigurationsparameter von `beegfs-meta.conf` Kann ebenfalls enthalten sein. Beachten Sie, dass die folgenden Parameter automatisch/an anderer Stelle konfiguriert werden und hier nicht angegeben werden sollten: `sysMgmtdHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`,

`connInterfacesFile`, und `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. Konfigurieren Sie eine oder mehrere unverankerte IPs, die andere Dienste und Clients verwenden, um eine Verbindung zu diesem Dienst herzustellen (dadurch wird das BeeGFS automatisch festgelegt `connInterfacesFile` Option):

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

4. Geben Sie optional ein oder mehrere zulässige IP-Subnetze an, die für die ausgehende Kommunikation verwendet werden können (dadurch wird automatisch das BeeGFS eingestellt `connNetFilterFile` Option):

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

5. Geben Sie das BeeGFS-Metadatenziel an, bei dem dieser Dienst Daten gemäß den folgenden Richtlinien speichert (dies konfiguriert auch automatisch den `storeMetaDirectory` Option):

   a. Für mehrere BeeGFS Services/Ziele kann derselbe Speicherpool oder Volume-Gruppenname verwendet werden. Stellen Sie einfach sicher, dass er dasselbe verwendet `name`, `raid_level`, `criteria_*`, und `common_*` Konfiguration für jede einzelne (die für jeden Service aufgeführten Volumes sollten unterschiedlich sein).

   b. Volume-Größen sollten als Prozentsatz der Storage-Pool/Volume-Gruppe angegeben werden. Die Summe sollte bei allen Services/Volumes, die über einen bestimmten Storage-Pool/Volume-Gruppe verfügen, nicht mehr als 100 übersteigen. Hinweis: Bei der Verwendung von SSDs wird empfohlen, freien Speicherplatz in der Volume-Gruppe zu belassen, um die SSD-Performance und den SSD-Verschleiß "Hier"zu maximieren (klicken Sie für weitere Details).

   c. Klicken Sie Auf "Hier" Eine vollständige Liste der für das verfügbaren Konfigurationsoptionen finden Sie unter `eseries_storage_pool_configuration`. Notieren Sie einige Optionen wie z. B. `state`, `host`, `host_type`, `workload_name`, und `workload_metadata` Und Volume-Namen werden automatisch generiert und sollten hier nicht angegeben werden.

```
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
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen BeeGFS-Metadatendienst darstellt.

Definieren Sie den BeeGFS-Speicherdienst

BeeGFS-Dienste werden mit Gruppenvariablen (Group_vars) konfiguriert.

Überblick

In diesem Abschnitt wird die Definition des BeeGFS-Speicherdienstes erläutert. In den HA-Clustern für ein bestimmtes Dateisystem sollte mindestens ein Service dieses Typs vorhanden sein. Die Konfiguration dieses Services umfasst die folgenden Punkte:

- Den Servicetyp (Storage).
- Definieren von Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen.
- Konfiguration einer oder mehrerer fließender IPs (logische Schnittstellen), an denen dieser Service erreicht werden kann.
- Geben Sie an, wo/wie Volumen(en) Daten für diesen Dienst speichern sollen (die BeeGFS-Speicherziele).

Schritte

"Planen Sie das Dateisystem"Erstellen Sie `group_vars/stor_<ID>.yml` für jeden Storage-Service im Cluster eine Datei unter, und füllen Sie sie wie folgt aus, um auf den Abschnitt Bezug zu nehmen:

1. Geben Sie diese Datei für die Konfiguration eines BeeGFS-Speicherdienstes an:

   ```
   beegfs_service: storage
   ```

2. Definieren Sie alle Konfigurationen, die nur für diesen BeeGFS-Dienst gelten sollen. Mindestens müssen Sie den gewünschten TCP- und UDP-Port angeben, jedoch alle unterstützten Konfigurationsparameter von `beegfs-storage.conf` Kann ebenfalls enthalten sein. Beachten Sie, dass die folgenden Parameter automatisch/an anderer Stelle konfiguriert werden und hier nicht angegeben werden sollten: `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, und `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. Konfigurieren Sie eine oder mehrere unverankerte IPs, die andere Dienste und Clients verwenden, um eine Verbindung zu diesem Dienst herzustellen (dadurch wird das BeeGFS automatisch festgelegt `connInterfacesFile` Option):

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

4. Geben Sie optional ein oder mehrere zulässige IP-Subnetze an, die für die ausgehende Kommunikation verwendet werden können (dadurch wird automatisch das BeeGFS eingestellt `connNetFilterFile` Option):

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

5. Geben Sie die BeeGFS-Speicherziele an, in denen dieser Service Daten gemäß den folgenden Richtlinien speichert (dies konfiguriert auch automatisch den `storeStorageDirectory` Option):

   a. Für mehrere BeeGFS Services/Ziele kann derselbe Speicherpool oder Volume-Gruppenname verwendet werden. Stellen Sie einfach sicher, dass er dasselbe verwendet `name`, `raid_level`, `criteria_*`, und `common_*` Konfiguration für jede einzelne (die für jeden Service aufgeführten Volumes sollten unterschiedlich sein).

   b. Volume-Größen sollten als Prozentsatz der Storage-Pool/Volume-Gruppe angegeben werden. Die Summe sollte bei allen Services/Volumes, die über einen bestimmten Storage-Pool/Volume-Gruppe verfügen, nicht mehr als 100 übersteigen. Hinweis: Bei der Verwendung von SSDs wird empfohlen, freien Speicherplatz in der Volume-Gruppe zu belassen, um die SSD-Performance und den SSD-Verschleiß "Hier"zu maximieren (klicken Sie für weitere Details).

   c. Klicken Sie Auf "Hier" Eine vollständige Liste der für das verfügbaren Konfigurationsoptionen finden Sie unter `eseries_storage_pool_configuration`. Notieren Sie einige Optionen wie z. B. `state`, `host`, `host_type`, `workload_name`, und `workload_metadata` Und Volume-Namen werden automatisch generiert und sollten hier nicht angegeben werden.

```
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
```

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei, die einen BeeGFS-Speicherdienst darstellt.

Zuordnen von BeeGFS-Services zu Datei-Nodes

Geben Sie an, welche Datei-Nodes jeden BeeGFS-Dienst mit dem ausführen können `inventory.yml` Datei:

Überblick

In diesem Abschnitt wird die Erstellung des erläutert `inventory.yml` Datei: Dazu gehören alle Block-Nodes und die Angabe, welche Datei-Nodes jeden BeeGFS-Service ausführen können.

Schritte

Erstellen Sie die Datei `inventory.yml` Und füllen Sie es wie folgt aus:

1. Erstellen Sie oben in der Datei die Ansible-Standardinventarstruktur:

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

2. Erstellen Sie eine Gruppe mit allen Block-Nodes, die an diesem HA-Cluster teilnehmen:

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

3. Erstellen Sie eine Gruppe, die alle BeeGFS-Dienste im Cluster und die Datei-Nodes enthält, auf denen sie ausgeführt werden:

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

4. Definieren Sie für jeden BeeGFS-Service im Cluster die bevorzugten und sekundären Dateiknoten, die diesen Service ausführen sollen:

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

Klicken Sie Auf "Hier" Beispiel für eine komplette Bestandsdatei.

Stellen Sie das BeeGFS-Dateisystem bereit

Ansible – Playbook-Überblick

BeeGFS HA-Cluster implementieren und managen mithilfe von Ansible

Überblick

In den vorherigen Abschnitten wurden die Schritte aufgeführt, mit denen ein Ansible-Inventar erstellt werden konnte, der ein BeeGFS HA-Cluster darstellt. In diesem Abschnitt wird die von NetApp entwickelte Ansible-Automatisierung für die Implementierung und das Management des Clusters vorgestellt.

Ansible – Wichtige Konzepte

Bevor Sie fortfahren, ist es hilfreich, sich mit ein paar Schlüsselkonzepten von Ansible vertraut zu machen:

- Aufgaben, die für einen Ansible-Bestand ausgeführt werden müssen, werden in einem sogenannten **Playbook** definiert.
  - Die meisten Aufgaben in Ansible sind **idempotent**, d. h., sie können mehrmals ausgeführt werden, um zu überprüfen, ob die gewünschte Konfiguration/der gewünschte Zustand noch angewendet wird, ohne

dass Dinge zu stören oder unnötige Updates zu machen.

- Die kleinste Ausführungseinheit in Ansible ist ein **Modul**.
  - Typische Playbooks nutzen mehrere Module.
    - Beispiele: Laden Sie ein Paket herunter, aktualisieren Sie eine Konfigurationsdatei, starten/aktivieren Sie einen Dienst.
  - NetApp verteilt Module zur Automatisierung von NetApp E-Series Systemen.
- Komplexe Automatisierungsoptionen sind besser als Rollen integriert.
  - Im Wesentlichen ein Standardformat zur Verteilung eines wiederverwendbaren Playbooks.
  - NetApp verteilt Rollen für Linux-Hosts und BeeGFS-Filesysteme.

BeeGFS HA-Rolle für Ansible: Schlüsselkonzepte

Die gesamte Automatisierung, die für das Implementieren und Managen jeder Version von BeeGFS auf NetApp erforderlich ist, ist als Ansible-Rolle verpackt und im Rahmen der verteilt "NetApp E-Series Ansible Collection für BeeGFS":

- Diese Rolle kann als irgendwo zwischen einem **Installer** und einer modernen **Deployment/Management** Engine für BeeGFS gedacht werden.
  - Nutzt moderne Infrastruktur als Code-Praktiken und -Philosophien um das Management der Storage-Infrastruktur in jeder Größenordnung zu vereinfachen
  - Das "Kubesbete"Projekt ermöglicht Benutzern die Implementierung und Wartung einer gesamten Kubernetes-Distribution für die Scale-out-Computing-Infrastruktur.
- Dabei handelt es sich um das **softwaredefinierte**-Format von NetApp zur Verpackung, Verteilung und Wartung von BeeGFS auf NetApp Lösungen.
  - Versuchen Sie, eine „Appliance-ähnliche“ Erfahrung zu schaffen, ohne eine gesamte Linux-Distribution oder ein großes Bild zu verteilen.
  - Dazu gehören die von NetApp entwickelten Open Cluster Framework (OCF)-konformen Cluster-Ressourcen-Agents für benutzerdefinierte BeeGFS-Ziele, IP-Adressen und Monitoring für intelligente Pacemaker/BeeGFS-Integration.
- Diese Rolle spielt nicht einfach die Implementierung der „Automatisierung“ und soll den gesamten Lebenszyklus des Filesystems managen, darunter:
  - Konfigurationsänderungen und Updates pro Service oder Cluster-weite Konfiguration
  - Automatisierung von Cluster-Reparatur und Recovery nach Behebung von Hardware-Problemen
  - Vereinfachte Performance-Optimierung mit Standardeinstellungen, die auf umfangreichen Tests mit BeeGFS und NetApp Volumes basieren
  - Überprüfung und Korrektur von Konfigurationstendenzen.

NetApp bietet auch eine Ansible-Rolle für "BeeGFS-Clients", Die optional verwendet werden kann, um BeeGFS zu installieren und mounten Sie Dateisysteme auf Compute/GPU/Login-Nodes.

Implementieren Sie das BeeGFS HA-Cluster

Geben Sie an, welche Aufgaben ausgeführt werden sollen, um BeeGFS HA-Cluster mithilfe eines Playbooks zu implementieren.

Überblick

In diesem Abschnitt wird beschrieben, wie Sie das Standard-Playbook zur Bereitstellung/zum Managen von BeeGFS auf NetApp zusammenstellen können.

Schritte

Erstellen Sie das Ansible Playbook

Erstellen Sie die Datei `playbook.yml` Und füllen Sie es wie folgt aus:

1. Definieren Sie zunächst einen Satz von Aufgaben (allgemein als A bezeichnet) "Spielen") Die nur auf Block-Nodes der NetApp E-Series ausgeführt werden sollte. Wir verwenden eine Pause-Aufgabe, um vor dem Ausführen der Installation eine Aufforderung zu geben (um versehentliche Playbook-Läufe zu vermeiden), und importieren dann die `nar_santricity_management` Rolle: Diese Rolle übernimmt die Anwendung aller in definierten allgemeinen Systemkonfiguration `group_vars/eseries_storage_systems.yml` Oder einzeln `host_vars/<BLOCK NODE>.yml` Dateien:

```
- 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. Definieren Sie die Wiedergabe, die für alle Datei- und Blockknoten ausgeführt wird:

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

3. In diesem Sales Play können wir optional einen Satz von „Voraufgaben“ definieren, die vor der Bereitstellung des HA-Clusters ausgeführt werden sollten. Dies kann nützlich sein, um alle Voraussetzungen wie Python zu überprüfen/zu installieren. Zudem können Überprüfungen vor dem Flug durchgeführt werden, beispielsweise die Unterstützung der bereitgestellten Ansible-Tags:

```
```

```
  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. Schließlich importiert dieses Spiel die BeeGFS HA-Rolle für die Version von BeeGFS, die Sie bereitstellen möchten:

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

> Für jede unterstützte Major.Minor Version von BeeGFS wird eine BeeGFS HA-Rolle beibehalten. Auf diese Weise können Benutzer festlegen, wann ein Upgrade von Major-/Minor-Versionen durchgeführt werden soll. Derzeit (`beegfs_7_3`(`beegfs_7_2`werden BeeGFS 7.3.x) oder BeeGFS 7.2.x) unterstützt. Standardmäßig werden beide Rollen zum Zeitpunkt der Veröffentlichung die neueste BeeGFS-Patch-Version bereitstellen. Benutzer können dies jedoch überschreiben und gegebenenfalls den neuesten Patch bereitstellen. "Upgrade-Leitfaden"Weitere Informationen finden Sie auf dem neuesten Stand.

5. Optional: Wenn Sie zusätzliche Aufgaben definieren möchten, sollten Sie beachten, ob die Aufgaben an geleitet werden sollen `all` Hosts (einschließlich der E-Series Storage-Systeme) oder nur die Datei-Nodes Definieren Sie bei Bedarf ein neues Spiel speziell für Dateiknoten mit `- hosts: ha_cluster`.

Klicken Sie Auf "Hier" Beispiel für eine vollständige Playbook-Datei.

NetApp Ansible Sammlungen installieren

Die BeeGFS-Sammlung für Ansible, und alle Abhängigkeiten werden aufrechterhalten "Ansible-Galaxie". Führen Sie auf Ihrem Ansible-Steuerungsknoten den folgenden Befehl aus, um die neueste Version zu installieren:

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

Obwohl nicht in der Regel empfohlen, ist es auch möglich, eine bestimmte Version der Sammlung zu installieren:

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

Führen Sie das Playbook aus

Aus dem Verzeichnis auf Ihrem Ansible-Steuerungsknoten, der den enthält `inventory.yml` Und `playbook.yml` Dateien, führen Sie das Playbook wie folgt aus:

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

Basierend auf der Cluster-Größe kann die ursprüngliche Implementierung 20+ Minuten dauern. Wenn die Implementierung aus irgendeinem Grund fehlschlägt, korrigieren Sie einfach Probleme (z. B. Fehlverkabelung, Knoten wurde nicht gestartet usw.) und starten Sie das Ansible Playbook neu.

"Allgemeine Konfiguration der Datei-Nodes"Wenn Sie angeben , wenn Sie die Standardoption wählen, damit Ansible die verbindungsbasierte Authentifizierung automatisch verwaltet, `connAuthFile` kann ein als gemeinsamer Schlüssel verwendet jetzt unter `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (standardmäßig) gefunden werden. Alle Clients, die auf das Dateisystem zugreifen müssen, müssen diesen gemeinsam genutzten Schlüssel verwenden. Dies wird automatisch verarbeitet, wenn Clients über die konfiguriert werden"BeeGFS-Client-Rolle".

Bereitstellen von BeeGFS-Clients

Optional kann Ansible verwendet werden, um BeeGFS-Clients zu konfigurieren und das Dateisystem zu mounten.

Überblick

Für den Zugriff auf BeeGFS-Dateisysteme muss der BeeGFS-Client auf jedem Node installiert und konfiguriert werden, der das Dateisystem bereitstellen muss. In diesem Abschnitt wird beschrieben, wie Sie diese Aufgaben mit dem verfügbaren ausführen "Ansible-Rolle".

Schritte

Erstellen Sie die Client-Bestandsdatei

1. Richten Sie bei Bedarf über den Ansible-Steuerungsknoten passwortlose SSH für jeden Host ein, den Sie als BeeGFS-Clients konfigurieren möchten:

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

2. Unter `host_vars/`Erstellen Sie für jeden BeeGFS-Client eine Datei mit dem Namen `<HOSTNAME>.yml` Füllen Sie den Platzhaltertext mit den korrekten Informationen für Ihre Umgebung aus:

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

3. Geben Sie optional eine der folgenden Optionen ein, wenn Sie die Rollen der NetApp E-Series Host Collection verwenden möchten, um InfiniBand- oder Ethernet-Schnittstellen für Clients zu konfigurieren, damit eine Verbindung mit BeeGFS File-Nodes hergestellt werden kann:

   a. Wenn der Netzwerktyp ist "InfiniBand (mit IPoIB)":

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

b. Wenn der Netzwerktyp ist "RDMA über Converged Ethernet (RoCE)":

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

c. Wenn der Netzwerktyp ist "Ethernet (nur TCP, kein 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. Erstellen Sie eine neue Datei `client_inventory.yml` Geben Sie den Benutzer an, den Ansible für die Verbindung mit den einzelnen Clients verwenden soll, und das Passwort, das Ansible zur Eskalation von Berechtigungen verwenden soll (dies erfordert `ansible_ssh_user` „Root“ oder „Sudo“-Berechtigungen besitzen):

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

Speichern Sie Passwörter nicht im Klartext. Verwenden Sie stattdessen den Ansible-Vault (siehe "Ansible-Dokumentation" Für Inhalte mit Ansible Vault) oder verwenden Sie den `--ask-become-pass` Option beim Ausführen des Playbooks.

5. Im `client_inventory.yml` Datei, Listen Sie alle Hosts auf, die als BeeGFS-Clients unter dem konfiguriert werden sollen `beegfs_clients` Gruppe, und dann beachten Sie die Inline-Kommentare und Uncomment zusätzliche Konfiguration erforderlich, um das BeeGFS-Client-Kernel-Modul auf Ihrem System zu erstellen:

```
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.
```

> Wenn Sie die NVIDIA OFED-Treiber verwenden, stellen Sie sicher, dass beegfs_Client_ofed_include_PATH auf den korrekten „Header include path“ für Ihre Linux-Installation verweist. Weitere Informationen finden Sie in der BeeGFS-Dokumentation für "RDMA-Unterstützung".

6. Im `client_inventory.yml` Datei, Listen Sie die BeeGFS-Dateisysteme auf, die Sie unter einem zuvor definierten gemountet haben möchten `vars`:

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

7. Ab BeeGFS 7.2.7 und 7.3.1 "Verbindungsauthentifizierung" müssen konfiguriert oder explizit deaktiviert sein. Je nachdem, wie Sie die verbindungsbasierte Authentifizierung bei der Angabe konfigurieren"Allgemeine Konfiguration der Datei-Nodes", müssen Sie möglicherweise Ihre Clientkonfiguration anpassen:

   a. Standardmäßig konfiguriert die HA-Cluster-Implementierung die Verbindungsauthentifizierung automatisch und generiert einen `connauthfile` Die auf dem Ansible-Kontroll-Node bei platziert/gewartet werden `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. Standardmäßig ist die BeeGFS-Client-Rolle so eingerichtet, dass sie diese Datei an die in definierten Clients liest/verteilt `client_inventory.yml`, Und es ist keine zusätzliche Aktion erforderlich.

      i. Weitere Optionen finden Sie in der vollständigen Liste der Standardwerte, die im enthalten sind "BeeGFS-Client-Rolle".

   b. Wenn Sie ein benutzerdefiniertes Geheimnis mit angeben `beegfs_ha_conn_auth_secret` Geben Sie ihn im an `client_inventory.yml` Außerdem:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Wenn Sie die verbindungsbasierte Authentifizierung vollständig mit deaktivieren `beegfs_ha_conn_auth_enabled`, Geben Sie das im an `client_inventory.yml` Außerdem:

```
beegfs_ha_conn_auth_enabled: false
```

Eine vollständige Liste der unterstützten Parameter und weitere Details finden Sie im "Vollständige BeeGFS-Client-Dokumentation". Klicken Sie für ein vollständiges Beispiel eines Clientbestands auf "Hier".

Erstellen Sie die BeeGFS Client Playbook-Datei

1. Erstellen Sie eine neue Datei `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. Optional: Wenn Sie die Rollen der NetApp E-Series Host Collection verwenden möchten, um Schnittstellen für Clients zu konfigurieren, mit denen sich eine Verbindung zu BeeGFS-Dateisystemen herstellen lässt, importieren Sie die Rolle entsprechend dem Schnittstellentyp, den Sie konfigurieren:

   a. Wenn Sie InfiniBand (IPoIB) verwenden:

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

   b. Bei Verwendung von RDMA over Converged Ethernet (RoCE):

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

   c. Wenn Sie Ethernet verwenden (nur TCP, kein RDMA):

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

3. Schließlich importieren Sie die BeeGFS-Client-Rolle, um die Client-Software zu installieren und das Dateisystem-Mounts einzurichten:

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

Klicken Sie zum vollständigen Beispiel für ein Client-Playbook auf "Hier".

Führen Sie das BeeGFS Client Playbook aus

Führen Sie den folgenden Befehl aus, um den Client zu installieren/zu erstellen und BeeGFS zu mounten:

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

Überprüfen Sie die BeeGFS-Bereitstellung

Überprüfen Sie die Bereitstellung des Dateisystems, bevor Sie das System in die Produktion bringen.

Überblick

Bevor Sie das BeeGFS-Dateisystem in Produktion setzen, führen Sie einige Überprüfungsprüfungen durch.

Schritte

1. Melden Sie sich bei einem Client an und führen Sie Folgendes aus, um sicherzustellen, dass alle erwarteten Knoten vorhanden/erreichbar sind, und es werden keine Inkonsistenzen oder andere Probleme gemeldet:

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

2. Fahren Sie den gesamten Cluster herunter und starten Sie ihn dann neu. Führen Sie von jedem beliebigen Dateiknoten Folgendes aus:

   ```
   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. Platzieren Sie jeden Knoten in den Standby-Modus, und überprüfen Sie, ob BeeGFS-Dienste einen Failover auf sekundäre Knoten ausführen können. So melden Sie sich an einem beliebigen Dateiknoten an und führen Sie Folgendes aus:

```
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. Verwenden Sie Tools zum Leistungsvergleich wie IOR und MDTest, um zu überprüfen, ob die Performance des Dateisystems den Erwartungen entspricht. Beispiele für häufige Tests und Parameter, die mit BeeGFS verwendet werden"Design-Verifizierung", finden Sie im Abschnitt BeeGFS on NetApp Verified Architecture.

Zusätzliche Tests sollten auf Grundlage der Abnahmekriterien durchgeführt werden, die für einen bestimmten Standort/eine bestimmte Installation definiert sind.