Utilizar arquitecturas personalizadas

BeeGFS on NetApp with E-Series Storage

NetApp
August 08, 2024

Tabla de contenidos

Utilizar arquitecturas personalizadas

Descripción general y requisitos

Use cualquier sistema de almacenamiento E/EF-Series de NetApp como nodos de bloque BeeGFS y servidores x86 como nodos de archivos BeeGFS cuando ponga en marcha clústeres de alta disponibilidad BeeGFS con Ansible.

Las definiciones de la terminología utilizada en esta sección se pueden encontrar en la "términos y conceptos" página.

Introducción

Aunque "Arquitecturas verificadas de NetApp" al proporcionar configuraciones de referencia predefinidas y orientación sobre la configuración, algunos clientes y partners pueden preferir diseñar arquitecturas personalizadas que se adapten mejor a requisitos concretos o a preferencias de hardware. Una de las principales ventajas de elegir BeeGFS en NetApp es la capacidad de poner en marcha clústeres de alta disponibilidad de disco compartido BeeGFS mediante Ansible, lo que simplifica la gestión del clúster y mejora la fiabilidad con componentes de alta disponibilidad creados por NetApp. La puesta en marcha de arquitecturas BeeGFS personalizadas en NetApp todavía se realiza con Ansible, lo que mantiene un enfoque similar al de los dispositivos en una gama flexible de hardware.

En esta sección, se describen los pasos generales necesarios para poner en marcha sistemas de archivos BeeGFS en hardware de NetApp y el uso de Ansible para configurar los sistemas de archivos BeeGFS. Para obtener más información sobre las mejores prácticas relacionadas con el diseño de sistemas de archivos BeeGFS y ejemplos optimizados, consulte "Arquitecturas verificadas de NetApp" sección.

Visión General de la implementación

Generalmente, la implementación de un sistema de archivos BeeGFS incluye los siguientes pasos:

- Configuración inicial:
  - Instalar/cablear la tornillería.
  - Configure los nodos de archivos y bloques.
  - Configure un nodo de control Ansible.
- Defina el sistema de archivos BeeGFS como un inventario de Ansible.
- Ejecute Ansible contra los nodos de archivos y bloques para poner en marcha BeeGFS.
  - Opcionalmente para configurar clientes y montar BeeGFS.

Las siguientes secciones tratarán estos pasos con más detalle.

Ansible gestiona todas las tareas de configuración y aprovisionamiento del software, incluidas:

- Crear/asignar volúmenes en nodos de bloques.
- Formateo/ajuste de volúmenes en nodos de archivo.
- Instalar/configurar software en nodos de archivos.
- Establecer el clúster de alta disponibilidad y configurar los recursos de BeeGFS y los servicios del sistema de archivos.

Requisitos

Ya está disponible el soporte para BeeGFS en Ansible "Galaxia de ansible" Como un conjunto de funciones y módulos que automatizan la implementación y gestión integral de clústeres de alta disponibilidad de BeeGFS.

BeeGFS se versión siguiendo un esquema de control de versiones <major>.<minor>.<patch> y la colección mantiene roles para cada versión compatible de <major>.<minor> de BeeGFS, por ejemplo BeeGFS 7.2 o BeeGFS 7.3. A medida que se publican actualizaciones de la colección, la versión de revisión de cada rol se actualizará para señalar la última versión disponible de BeeGFS para esa rama de versión (por ejemplo: 7.2.8). Cada versión de la colección también está probada y compatible con distribuciones y versiones específicas de Linux, actualmente Red Hat para nodos de archivos, y RedHat y Ubuntu para clientes. No se admite la ejecución de otras distribuciones; no se recomienda ejecutar otras versiones (especialmente otras versiones principales).

Nodo de control de Ansible

Este nodo contendrá el inventario y los libros de estrategia utilizados para gestionar BeeGFS. Requiere:

- Ansible 6.x (núcleo de Ansible 2.13)
- Python 3.6 (o posterior)
- Paquetes de Python (pip): Ipaddr y netaddr

También es recomendable configurar SSH sin contraseñas desde el nodo de control en todos los clientes y nodos de archivos BeeGFS.

Nodos de archivo BeeGFS

Los nodos de archivo deben ejecutar RedHat 9.3 y tener acceso al repositorio ha que contenga los paquetes requeridos (marcapaso, corosync, valla-agents-all, Resource-agents). Por ejemplo, se puede ejecutar el siguiente comando para habilitar el repositorio apropiado en RedHat 9:

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

Nodos de cliente BeeGFS

Hay disponible un rol de Ansible para el cliente BeeGFS para instalar el paquete de cliente BeeGFS y gestionar los montajes BeeGFS. Esta función se ha probado con RedHat 8.4 y Ubuntu 22.04.

Si no utiliza Ansible para configurar el cliente BeeGFS y montar BeeGFS, any "Distribución y kernel de Linux compatibles con BeeGFS" puede utilizarse.

Configuración inicial

Instale y conecte el cableado de la tornillería

Pasos necesarios para instalar y cablear el hardware utilizado para ejecutar BeeGFS en NetApp.

Planifique la instalación

Cada sistema de archivos BeeGFS consistirá en un número determinado de nodos de archivo en los que se ejecutan servicios BeeGFS mediante el almacenamiento de fondo proporcionado por un número determinado de nodos de bloque. Los nodos de archivos están configurados en uno o varios clústeres de alta disponibilidad para proporcionar tolerancia a fallos en los servicios BeeGFS. Cada nodo de bloque ya es un par de alta disponibilidad activo-activo. El número mínimo de nodos de archivo admitidos en cada clúster de alta disponibilidad es de tres y el número máximo de nodos de archivo admitidos en cada clúster es de diez. Los sistemas de archivos BeeGFS pueden escalar más allá de diez nodos mediante la puesta en marcha de varios clústeres de alta disponibilidad independientes que funcionan en conjunto para proporcionar un espacio de nombres único del sistema de archivos.

Normalmente, cada clúster de alta disponibilidad se implementa como una serie de «elementos básicos» donde hay algún número de nodos de archivo (servidores x86) conectados directamente a un cierto número de nodos de bloques (normalmente sistemas de almacenamiento E-Series). Esta configuración crea un clúster asimétrico, en el que los servicios BeeGFS sólo pueden ejecutarse en determinados nodos de archivo que tienen acceso al almacenamiento de bloques de fondo utilizado para los destinos BeeGFS. El equilibrio de los nodos de archivo a bloque en cada elemento básico y el protocolo de almacenamiento que se utiliza para las conexiones directas dependen de los requisitos de una instalación concreta.

Una arquitectura de cluster de alta disponibilidad alternativa utiliza una estructura de almacenamiento (también conocida como red de área de almacenamiento o SAN) entre los nodos de archivo y de bloque para establecer un cluster simétrico. Esto permite que los servicios de BeeGFS se ejecuten en cualquier nodo de archivo en un clúster de alta disponibilidad en particular. Como los clústeres simétricos en general no son tan rentables debido al hardware SAN adicional, esta documentación presupone el uso de un clúster asimétrico desplegado como una serie de uno o más bloques de construcción.

Asegúrese de que la arquitectura del sistema de archivos deseada para un despliegue de BeeGFS en particular está bien comprendida antes de continuar con la instalación.

Hardware en rack

Al planificar la instalación, es importante que todo el equipo de cada bloque de construcción esté montado en rack adyacente. La práctica recomendada es que los nodos de archivo estén montados en rack inmediatamente encima de los nodos de bloque en cada bloque básico. Siga la documentación de los modelos de archivo y. "bloque" los nodos que usa mientras instala rieles y hardware en el rack.

Ejemplo de un solo elemento básico:

[ejemplo de bloque de construcción]

Ejemplo de una instalación de BeeGFS de gran tamaño en la que hay varios elementos básicos en cada clúster de alta disponibilidad y varios clústeres de alta disponibilidad en el sistema de archivos:

[Ejemplo de implementación de BeeGFS]

Nodos de archivos de cable y bloques

Normalmente, se conectan directamente los puertos HIC de los nodos de bloque de E-Series al adaptador de canal de host designado (para protocolos InfiniBand) o al adaptador de bus de host (para el canal de fibra y otros protocolos) de los nodos de archivo. La forma exacta de establecer estas conexiones dependerá de la arquitectura del sistema de archivos deseada, aquí se muestra un ejemplo "Basado en BeeGFS de segunda generación en la arquitectura verificada de NetApp":

[Ejemplo de archivo BeeGFS para bloquear el cableado de nodos]

Conecte los nodos de archivo a la red de cliente

Cada nodo de archivo tendrá un número de puertos InfiniBand o Ethernet designados para el tráfico del cliente BeeGFS. Dependiendo de la arquitectura que tenga cada nodo de archivo tendrá una o varias conexiones a una red cliente/almacenamiento de alto rendimiento, potencialmente a varios switches para obtener redundancia y un mayor ancho de banda. A continuación se muestra un ejemplo de cableado de cliente mediante switches de red redundantes, donde los puertos resaltados en verde oscuro frente al verde claro se conectan a switches separados:

[Ejemplo de cableado de cliente BeeGFS]

Conecte la red y la alimentación de la administración

Establezca las conexiones de red necesarias para la red dentro y fuera de banda.

Conecte todas las fuentes de alimentación asegurándose de que cada nodo de archivo y bloque tenga conexiones a varias unidades de distribución de alimentación para obtener redundancia (si está disponible).

Configure los nodos de archivos y bloques

Pasos manuales necesarios para configurar nodos de archivos y bloques antes de ejecutar Ansible.

Nodos de archivos

Configurar el controlador de administración de la placa base (BMC)

Un controlador de administración en placa base (BMC), conocido a veces como procesador de servicios, es el nombre genérico para la capacidad de administración fuera de banda integrada en varias plataformas de servidor que pueden proporcionar acceso remoto aunque el sistema operativo no esté instalado o sea accesible. Los proveedores suelen comercializar esta funcionalidad con su propia Marca. Por ejemplo, en el Lenovo SR665, el BMC se conoce como controlador XClaridad Lenovo (XCC).

Siga la documentación del proveedor del servidor para habilitar las licencias necesarias para acceder a esta funcionalidad y asegurarse de que el BMC está conectado a la red y configurado de forma adecuada para el acceso remoto.

> ⓘ Si desea utilizar la cercado basada en BMC con Redfish, asegúrese de que Redfish esté activado y de que se pueda acceder a la interfaz BMC desde el sistema operativo instalado en el nodo de archivo. Es posible que se requiera una configuración especial en el conmutador de red si el BMC y el sistema operativo comparten la misma interfaz de red física.

Ajuste la configuración del sistema

Utilizando la interfaz de configuración del sistema (BIOS/UEFI), asegúrese de que los ajustes se han establecido para maximizar el rendimiento. La configuración exacta y los valores óptimos variarán en función del modelo de servidor que se esté utilizando. Se proporciona orientación "modelos de nodos de archivos verificados", de lo contrario, consulte la documentación del proveedor del servidor y las mejores prácticas basadas en su modelo.

Instale un sistema operativo

Instale un sistema operativo compatible según los requisitos del nodo de archivo enumerados "aquí". Consulte los pasos adicionales que se indican a continuación según su distribución de Linux.

Red Hat

Utilice RedHat Subscription Manager para registrar y suscribirse al sistema para permitir la instalación de los paquetes necesarios desde los repositorios oficiales de Red Hat y para limitar las actualizaciones a la versión compatible de Red Hat: `subscription-manager release --set=<MAJOR_VERSION>.<MINOR_VERSION>`. Para ver instrucciones, consulte "Cómo registrar y suscribirse a un sistema RHEL" y.. "Cómo limitar las actualizaciones".

Active el repositorio de Red Hat que contiene los paquetes necesarios para la alta disponibilidad:

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

Configure la red de gestión

Configure las interfaces de red necesarias para permitir la administración en banda del sistema operativo. Los pasos exactos dependerán de la distribución y versión específicas de Linux que se esté utilizando.

> Compruebe que SSH esté habilitado y que todas las interfaces de gestión sean accesibles desde el nodo de control de Ansible.

Actualice el firmware de HCA y HBA

Comprobar que todos los HBA y HCA estén ejecutando versiones de firmware compatibles mostradas en la "Matriz de interoperabilidad de NetApp" y actualícelo si es necesario. También se pueden encontrar recomendaciones adicionales para los adaptadores NVIDIA ConnectX "aquí".

Nodos de bloques

Siga los pasos a. "Póngase en marcha con E-Series" para configurar el puerto de gestión en cada controladora del nodo de bloque y, opcionalmente, establezca el nombre de la cabina de almacenamiento para cada sistema.

> No será necesario realizar ninguna configuración adicional más allá de garantizar que todos los nodos de bloques sean accesibles desde el nodo de control de Ansible. La configuración del sistema restante se aplicará/mantendrá con Ansible.

Configure el nodo de control de Ansible

Configure un nodo de control de Ansible para poner en marcha y gestionar el sistema de archivos.

Descripción general

Un nodo de control de Ansible es una máquina física o virtual Linux que se usa para gestionar el clúster. Debe cumplir los siguientes requisitos:

- Conozca la "requisitos" Para la función de alta disponibilidad de BeeGFS, incluidas las versiones instaladas de Ansible, Python y cualquier paquete adicional de Python.
- Conozca al funcionario "Requisitos del nodo de control de Ansible" incluye las versiones del sistema operativo.
- Tienen acceso SSH y HTTPS a todos los nodos de archivos y bloques.

Encontrará los pasos detallados para la instalación "aquí".

Defina el sistema de archivos BeeGFS

Descripción general del inventario de Ansible

El inventario de Ansible es un conjunto de archivos de configuración que definen el clúster de alta disponibilidad de BeeGFS deseado.

Descripción general

Se recomienda seguir las prácticas de Ansible estándar para organizar el "inventario", incluido el uso de "subdirectorios/archivos" en lugar de almacenar todo el inventario en un archivo.

El inventario de Ansible para un único clúster de alta disponibilidad de BeeGFS está organizado de la siguiente forma:

[Descripción general del inventario de Ansible]

> Dado que un único sistema de archivos BeeGFS puede abarcar varios clústeres de alta disponibilidad, es posible que las instalaciones de gran tamaño tengan varios inventarios de Ansible. Por lo general, no se recomienda intentar definir varios clústeres de alta disponibilidad como un único inventario de Ansible para evitar problemas.

Pasos

1. En el nodo de control de Ansible, cree un directorio vacío que contendrá el inventario de Ansible para el clúster de BeeGFS que desea implementar.
   a. Si su sistema de archivos contendrá en algún momento varios clústeres de alta disponibilidad, se recomienda crear primero un directorio para el sistema de archivos y, a continuación, subdirectorios para el inventario que represente cada clúster de alta disponibilidad. Por ejemplo:

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

2. En el directorio que contiene el inventario del clúster de alta disponibilidad que desea implementar, cree dos directorios `group_vars` y.. `host_vars` y dos archivos `inventory.yml` y.. `playbook.yml`.

En las siguientes secciones se describe la definición del contenido de cada uno de estos archivos.

Planifique el sistema de archivos

Planifique la puesta en marcha del sistema de archivos antes de crear el inventario de Ansible.

Descripción general

Antes de implementar el sistema de archivos, debe definir qué direcciones IP, puertos y otra configuración serán necesarias para todos los nodos de archivos, nodos de bloques y servicios BeeGFS que se ejecuten en el clúster. Aunque la configuración exacta variará en función de la arquitectura del clúster, esta sección define las prácticas recomendadas y los pasos a seguir que son aplicables.

Pasos

1. Si utiliza un protocolo de almacenamiento basado en IP (como Iser, iSCSI, NVMe/IB o NVMe/roce) para conectar nodos de archivos a nodos de bloques, rellene la siguiente hoja de datos para cada elemento básico. Cada conexión directa en un único bloque de creación debería tener una subred única y no debería haber superposición con subredes utilizadas para la conectividad cliente-servidor.

Nodo de archivo	Puerto IB	Dirección IP	Nodo de bloques	Puerto IB	IP física	Virtual IP (solo para EF600 con HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

> Si los nodos de archivo y bloque de cada bloque de creación están conectados directamente, a menudo puede reutilizar las mismas IP/esquema para varios bloques de creación.

2. Independientemente de si utiliza InfiniBand o RDMA sobre Ethernet convergente (roce) para la red de almacenamiento, rellene la siguiente hoja de datos para determinar los rangos de IP que se usarán para los servicios de clúster de alta disponibilidad, los servicios de archivos BeeGFS y los clientes para comunicarse:

Específico	**Puerto InfiniBand**	**Dirección IP o rango**
IP(s) de clúster de BeeGFS	<INTERFACE(s)>	<RANGE>
Gestión de BeeGFS	<INTERFACE(s)>	<IP(s)>

Específico	Puerto InfiniBand	Dirección IP o rango
Metadatos de BeeGFS	<INTERFACE(s)>	<RANGE>
Almacenamiento de BeeGFS	<INTERFACE(s)>	<RANGE>
Clientes BeeGFS	<INTERFACE(s)>	<RANGE>

a. Si utiliza una sola subred IP, solo será necesaria una hoja de datos; de lo contrario, rellene también una hoja de cálculo para la segunda subred.

3. En función de lo anterior, para cada bloque de creación del clúster, rellene la siguiente hoja de trabajo que define qué servicios de BeeGFS se ejecutará. Para cada servicio, especifique los nodos de archivo preferidos/secundarios, el puerto de red, las IP flotantes, la asignación de zonas NUMA (si es necesario) y qué nodos de bloque se usarán para sus destinos. Consulte las siguientes directrices al rellenar la hoja de trabajo:

   a. Especifique los servicios BeeGFS como cualquiera de los dos `mgmt.yml`, `meta_<ID>.yml`, o. `storage_<ID>.yml` Donde ID representa un número único en todos los servicios BeeGFS de ese tipo en este sistema de archivos. Esta convención simplificará la referencia a esta hoja de trabajo en secciones posteriores mientras crea archivos para configurar cada servicio.

   b. Los puertos para los servicios BeeGFS sólo deben ser únicos en un bloque de construcción en particular. Asegúrese de que los servicios con el mismo número de puerto no pueden ejecutarse nunca en el mismo nodo de archivo para evitar conflictos de puertos.

   c. Si los servicios necesarios pueden utilizar volúmenes de más de un nodo de bloque o un pool de almacenamiento (y no todos los volúmenes deben ser propiedad de la misma controladora). Múltiples servicios también pueden compartir la misma configuración de nodo de bloque o pool de almacenamiento (se definirán los volúmenes individuales en una sección posterior).

Servicio BeeGFS (nombre de archivo)	Nodos de archivos	Puerto	IP flotantes	Zona NUMA	Nodo de bloques	Del banco de almacenamiento	Controladora propietaria
<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>

Para obtener más información sobre las convenciones estándar, las mejores prácticas y las hojas de trabajo de ejemplo rellenadas, consulte la "mejores prácticas" y.. "Defina los bloques de creación de BeeGFS" Secciones de BeeGFS en Arquitectura verificada de NetApp.

Defina los nodos de archivo y bloque

Configurar nodos de archivos individuales

Especifique la configuración de los nodos de archivos individuales con variables de host (host_var).

Descripción general

Esta sección recorre la relleno de un `host_vars/<FILE_NODE_HOSTNAME>.yml` archivo para cada nodo de archivo del clúster. Estos archivos sólo deben contener una configuración exclusiva de un nodo de archivo concreto. Esto incluye normalmente:

- Definición de la IP o el nombre de host que Ansible debe usar para conectarse al nodo.
- Configurar interfaces adicionales e IP de clúster utilizadas para los servicios de clúster de alta disponibilidad (Pacemaker y Corosync) para comunicarse con otros nodos de archivo. De forma predeterminada, estos servicios utilizan la misma red que la interfaz de gestión, pero deberían estar disponibles interfaces adicionales para la redundancia. La práctica común es definir IP adicionales en la red de almacenamiento, lo que evita la necesidad de un clúster o una red de gestión adicionales.
  - El rendimiento de cualquier red utilizada para la comunicación del clúster no es crítico en cuanto al rendimiento del sistema de archivos. Con la configuración de clúster predeterminada, por lo general, al menos una red de 1GB GB/s proporcionará suficiente rendimiento para las operaciones de clúster, como la sincronización de estados de nodo y la coordinación de cambios de estado de recursos de clúster. Las redes lentas/ocupadas pueden hacer que los cambios en el estado de los recursos tarden más de lo habitual y, en casos extremos, podrían resultar en que los nodos se expulsen del clúster si no pueden enviar latidos en un período de tiempo razonable.
- Configurar las interfaces utilizadas para conectarse a los nodos de bloques sobre el protocolo deseado (por ejemplo, iSCSI/Iser, NVMe/IB, NVMe/roce, FCP, etc.)

Pasos

Hacer referencia al esquema de direccionamiento IP definido en "Planifique el sistema de archivos" para cada nodo de archivo del clúster, cree un archivo `host_vars/<FILE_NODE_HOSTNAME>/yml` y rellenarlo de la siguiente manera:

1. En la parte superior, especifique la IP o el nombre de host que Ansible debe usar a SSH del nodo y gestiónelo:

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

2. Configure las IP adicionales que se puedan usar para el tráfico del clúster:
   a. Si el tipo de red es "InfiniBand (uso de IPoIB)":

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

   b. Si el tipo de red es "RDMA sobre 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. Si el tipo de red es "Ethernet (solo TCP, sin 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. Indique qué IP se deben utilizar para el tráfico del clúster con las IP preferidas más alta:

```
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.
```

Los IPS configurados en el paso dos no se utilizarán como IP de clúster a menos que estén incluidos en el `beegfs_ha_cluster_node_ips` lista. Esto le permite configurar IP/interfaces adicionales con Ansible que pueden utilizarse para otros fines si así lo desea.

4. Si el nodo de archivo tiene que comunicarse con los nodos de bloque a través de un protocolo basado en IP, se deberán configurar las IP en la interfaz adecuada y con todos los paquetes necesarios para instalar y configurar ese protocolo.

a. Si se utiliza "ISCSI":

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

b. Si se utiliza "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. Si se utiliza "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. Si se utiliza "NVMe/roce":

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

e. Otros protocolos:

   i. Si se utiliza "NVMe/FC", no es necesario configurar interfaces individuales. La implementación del clúster BeeGFS detectará automáticamente los requisitos de protocolo e instalará/configurará según sea necesario. Si utiliza una estructura para conectar nodos de archivos y bloques, asegúrese de que los switches se dividen correctamente siguiendo las prácticas recomendadas de NetApp y del proveedor del switch.

   ii. El uso de FCP o SAS no requiere la instalación ni la configuración de software adicional. Si utiliza FCP, asegúrese de que los switches se dividen correctamente a continuación "NetApp" y las prácticas recomendadas de su proveedor del switch.

   iii. No se recomienda el uso de SRP IB en este momento. Utilice NVMe/IB o Iser en función de lo que admita su nodo de bloque E-Series.

Haga clic en "aquí" para obtener un ejemplo de un archivo de inventario completo que representa un solo nodo de archivo.

Avanzado: Alternar los adaptadores VPI NVIDIA ConnectX entre Ethernet y modo InfiniBand

Los adaptadores NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) admiten InfiniBand y Ethernet como capa de transporte. El cambio entre modos no se negocia automáticamente y debe configurarse mediante la <code>mstconfig</code> herramienta incluida en <code>mstflint</code>, un paquete de código abierto que forma parte de "Herramientas de Firmare de NVIDIA (MFT)". El cambio del modo de los adaptadores solo debe realizarse una vez. Esto se puede hacer manualmente, o incluir en el inventario de Ansible como parte de cualquier interfaz configurada usando la <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> sección del inventario, para que se

active/aplique automáticamente.

Por ejemplo, para cambiar una interfaz actual en modo InfiniBand a Ethernet, se puede utilizar para roce:

1. Especifique para cada interfaz que desee configurar `mstconfig` como una asignación (o diccionario) que especifica `LINK_TYPE_P<N>` donde `<N>` Viene determinado por el número de puerto de HCA de la interfaz. La `<N>` el valor se puede determinar ejecutando `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` Y agregando 1 al último número desde el nombre de la ranura PCI y convirtiendo a decimal.
   a. Por ejemplo dado `PCI_SLOT_NAME=0000:2f:00.2` (2 + 1 → puerto HCA 3) → `LINK_TYPE_P3: eth`:

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

Para obtener información adicional, consulte "Documentación de la colección de hosts E-Series de NetApp" para el tipo/protocolo de interfaz que utiliza.

Configure nodos de bloques individuales

Especifique la configuración de los nodos de bloque individuales con variables de host (host_var).

Descripción general

Esta sección recorre la relleno de un `host_vars/<BLOCK_NODE_HOSTNAME>.yml` archivo de cada nodo de bloque del clúster. Estos archivos sólo deben contener una configuración exclusiva de un nodo de bloque determinado. Esto incluye normalmente:

- El nombre del sistema (como se muestra en System Manager).
- La URL de HTTPS para una de las controladoras (se utiliza para gestionar el sistema mediante su API DE REST).
- Qué nodos de archivo de protocolo de almacenamiento utilizan para conectarse a este nodo de bloque.
- Configurar los puertos de tarjeta de interfaz del host (HIC), como las direcciones IP (si son necesarias).

Pasos

Hacer referencia al esquema de direccionamiento IP definido en "Planifique el sistema de archivos" para cada nodo de bloque del clúster, cree un archivo `host_vars/<BLOCK_NODE_HOSTNAME>/yml` y rellenarlo de la siguiente manera:

1. En la parte superior, especifique el nombre del sistema y la URL de HTTPS para una de las controladoras:

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

2. Seleccione la "protocolo" los nodos de archivo utilizarán para conectarse a este nodo de bloque:
   a. Protocolos compatibles: `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

```
eseries_initiator_protocol: <PROTOCOL>
```

3. Según el protocolo en uso, los puertos HIC pueden necesitar una configuración adicional. Si es necesario, se debe definir la configuración de puertos de HIC para que la entrada superior de la configuración de cada controladora se corresponda con el puerto físico más a la izquierda de cada controladora y el puerto inferior al puerto que se encuentra en el extremo derecho. Todos los puertos requieren una configuración válida incluso si no están en uso actualmente.

> Consulte también la siguiente sección si utiliza InfiniBand HDR (200 GB) o roce de 200 GB con nodos de bloque EF600.

   a. Para 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. Para 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. Para 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. Para 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. Los protocolos FC y SAS no requieren configuración adicional. SRP no se recomienda correctamente.

Para obtener opciones adicionales para configurar los puertos HIC y los protocolos de host, incluida la capacidad de configurar CHAP iSCSI, consulte la "documentación" Incluido con la colección SANtricity. Tenga en cuenta que al implementar BeeGFS, el pool de almacenamiento, la configuración de volumen y otros aspectos del aprovisionamiento del almacenamiento se configurarán en otra parte y no se deberán definir en este archivo.

Haga clic en "aquí" para obtener un ejemplo de un archivo de inventario completo que representa un solo nodo de bloque.

Mediante InfiniBand HDR (200 GB) o roce de 200 GB con nodos de bloque de EF600 de NetApp:

Para utilizar InfiniBand HDR (200 GB) con EF600, se debe configurar una segunda IP "virtual" para cada puerto físico. A continuación se muestra un ejemplo de la forma correcta de configurar un EF600 equipado con la HIC HDR InfiniBand de doble puerto:

```
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)
```

Especifique la configuración de nodos de archivos comunes

Especifique la configuración de nodos de archivos comunes mediante variables de grupo (group_var).

Descripción general

La configuración que debería Apple para todos los nodos de archivo se define en `group_vars/ha_cluster.yml`. Normalmente incluye:

- Información detallada sobre cómo conectarse e iniciar sesión en cada nodo de archivo.
- Configuración de red común.
- Si se permiten reinicios automáticos.
- Cómo deben configurarse los estados de firewall y selinux.
- Configuración de clústeres, incluidas las alertas y las cercas.
- Ajuste del rendimiento.
- Configuración de servicio de BeeGFS común.

Las opciones establecidas en este archivo también pueden definirse en nodos de archivos individuales; por ejemplo, si se están usando modelos de hardware mixtos o tiene contraseñas diferentes para cada nodo. La configuración en nodos de archivo individuales tendrá prioridad sobre la configuración de este archivo.

Pasos

Cree el archivo `group_vars/ha_cluster.yml` y rellenarlo de la siguiente manera:

1. Indique cómo debe autenticarse el nodo de Ansible Control con los hosts remotos:

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

Especialmente en entornos de producción, no almacene contraseñas en texto sin formato. En su lugar, use Ansible Vault (consulte "Cifrado de contenido con Ansible Vault") o el `--ask-become-pass` al ejecutar el libro de estrategia. Si la `ansible_ssh_user` es ya el usuario raíz. puede omitir de forma opcional el `ansible_become_password`.

2. Si está configurando IP estáticas en interfaces ethernet o InfiniBand (por ejemplo, IP de clúster) y varias interfaces se encuentran en la misma subred IP (por ejemplo, si ib0 está usando 192.168.1.10/24 e ib1 está usando 192.168.1.11/24), Para que el soporte multihost funcione correctamente, se deben configurar reglas y tablas de enrutamiento IP adicionales. Sólo tiene que activar el enlace de configuración de la interfaz de red proporcionado de la siguiente forma:

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

3. Al poner en marcha el clúster, según el protocolo de almacenamiento, puede que sean necesarios el reinicio de los nodos para facilitar la detección de dispositivos de bloques remotos (volúmenes de E-Series) o aplicar otros aspectos de la configuración. De forma predeterminada, los nodos se preguntará antes de reiniciar, pero puede permitir que los nodos se reinicien automáticamente especificando lo siguiente:

```
eseries_common_allow_host_reboot: true
```

   a. De forma predeterminada, después de un reinicio, para asegurarse de que los dispositivos de bloque y otros servicios estén listos, Ansible esperará hasta el sistema `default.target` se alcanza antes de continuar con la implementación. En algunos casos, cuando se utiliza NVMe/IB, es posible que este no sea el tiempo suficiente para inicializar, detectar y conectarse a dispositivos remotos. Esto puede provocar que la implementación automatizada continúe prematuramente y falle. Para evitar esto cuando se usa NVMe/IB, también se debe definir lo siguiente:

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

4. Para comunicarse con los servicios de clúster de BeeGFS y ha se necesitan varios puertos de firewall. A menos que desee configurar manualmente el firmwwall (no se recomienda), especifique lo siguiente para crear zonas de firewall necesarias y abrir puertos automáticamente:

```
beegfs_ha_firewall_configure: True
```

5. En este momento, SELinux no es compatible y se recomienda que el estado se configure como desactivado para evitar conflictos (especialmente cuando RDMA está en uso). Establezca lo siguiente para asegurarse de que SELinux esté desactivado:

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

6. Configure la autenticación de modo que los nodos de archivo puedan comunicarse, ajustando los valores predeterminados según sea necesario según las directivas de su organización:

```
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. Según la "Planifique el sistema de archivos" Especifique la IP de administración de BeeGFS para este sistema de archivos:

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

> Aunque aparentemente redundante, `beegfs_ha_mgmtd_floating_ip` Es importante cuando escala el sistema de archivos BeeGFS más allá de un único clúster de alta disponibilidad. Los clústeres de alta disponibilidad posteriores se ponen en marcha sin un servicio de gestión de BeeGFS adicional y se señalan en el servicio de gestión proporcionado por el primer clúster.

8. Habilite las alertas de correo electrónico si lo desea:

```
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. Se recomienda encarecidamente habilitar la delimitación; de lo contrario, se puede bloquear que los servicios se inicien en nodos secundarios cuando se produzca un error en el nodo principal.
   a. Active la delimitación de forma global especificando lo siguiente:

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

      i. Nota Cualquier compatible "propiedad del clúster" también se puede especificar aquí si es necesario. No suele ser necesario ajustar estos ajustes, ya que el papel BeeGFS HA se entrega con una serie de pruebas bien probadas "valores predeterminados".

   b. A continuación, seleccione y configure un agente de cercado:
      i. OPCIÓN 1: Para habilitar la cercado mediante unidades de distribución de energía (PDU) 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. OPCIÓN 2: Para habilitar la esgrima mediante las API Redfish proporcionadas por Lenovo XCC (y otros BMCs):

```
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. Para obtener más información sobre la configuración de otros agentes de cercado, consulte la "Documentación de redhat".

10. El rol de ha de BeeGFS puede aplicar muchos parámetros de ajuste diferentes para ayudar a optimizar aún más el rendimiento. Entre ellos se incluyen la optimización de la utilización de la memoria del núcleo y la E/S del dispositivo en bloque, entre otros parámetros. El rol se incluye con un conjunto razonable de "valores predeterminados" basado en pruebas con nodos de bloque NetApp E-Series, pero de forma predeterminada estos no se aplican a menos que especifique:

```
beegfs_ha_enable_performance_tuning: True
```

a. Si es necesario, también especifique aquí cualquier cambio en el ajuste del rendimiento predeterminado. Consulte la documentación completa "parámetros de ajuste del rendimiento" para obtener más información.

11. Para garantizar que las direcciones IP flotantes (a veces conocidas como interfaces lógicas) utilizadas para los servicios BeeGFS puedan conmutar por error entre nodos de archivos, todas las interfaces de red deben tener un nombre coherente. De forma predeterminada, el kernel genera nombres de interfaz de red, lo cual no garantiza la generación de nombres coherentes, incluso en modelos de servidor idénticos con adaptadores de red instalados en las mismas ranuras PCIe. Esto también es útil cuando se crean inventarios antes de que el equipo se despliegue y se conozcan los nombres de las interfaces generadas. Para garantizar nombres de dispositivos coherentes, basados en un diagrama de bloque del servidor o. `lshw -class network -businfo` Output, especifique la asignación de dirección PCIe a interfaz lógica deseada del siguiente modo:

a. Para interfaces de red InfiniBand (IPoIB):

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

b. Para interfaces de red Ethernet:

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

> Para evitar conflictos cuando se cambia el nombre de las interfaces (evitando que se le cambie el nombre), no debe utilizar ningún nombre predeterminado potencial como eth0, ens9f0, ib0 o ibs4f0. Una convención de nomenclatura común consiste en usar "e" o "i" para Ethernet o InfiniBand, seguido del número de ranura PCIe y una letra para indicar el puerto. Por ejemplo, el segundo puerto de un adaptador InfiniBand instalado en la ranura 3 sería: I3b.

> Si va a utilizar un modelo de nodo de archivos verificado, haga clic en "aquí" Asignaciones de puerto lógico a dirección PCIe de ejemplo.

12. Opcionalmente, especifique la configuración que debe aplicarse a todos los servicios de BeeGFS del clúster. Se pueden encontrar valores de configuración por defecto "aquí"y la configuración por servicio se especifica en otro lugar:
    a. Servicio de gestión de BeeGFS:

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

    b. Servicios de metadatos BeeGFS:

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

    c. Servicios de almacenamiento de BeeGFS:

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

13. A partir de BeeGFS 7.2.7 y 7.3.1 "autenticación de conexión" se debe configurar o deshabilitar explícitamente. Hay algunas formas de configurar esto con la puesta en marcha basada en Ansible:
    a. De forma predeterminada, la implementación configurará automáticamente la autenticación de conexión y generará un `connauthfile` Se distribuirá a todos los nodos de archivos y se utilizará con los servicios BeeGFS. Este archivo también se colocará/mantendrá en el nodo de control Ansible en `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` donde se debe mantener (de forma segura) para reutilizarlo con clientes que necesiten acceder a este sistema de archivos.
        i. Para generar una nueva clave, especifique `-e "beegfs_ha_conn_auth_force_new=True` Al ejecutar el libro de estrategia de Ansible. Nota esto se ignora si un `beegfs_ha_conn_auth_secret` está definido.
        ii. Para opciones avanzadas, consulte la lista completa de valores predeterminados incluidos con el "Rol de BeeGFS ha".

b. Se puede utilizar un secreto personalizado definiendo lo siguiente en `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. La autenticación de conexión se puede deshabilitar completamente (NO se recomienda):

```
beegfs_ha_conn_auth_enabled: false
```

Haga clic en "aquí" para obtener un ejemplo de un archivo de inventario completo que representa la configuración común de nodos de archivos.

Usar InfiniBand HDR (200 GB) con nodos de bloque de EF600 de NetApp:

Para utilizar InfiniBand HDR (200 GB) con EF600, el administrador de subredes debe admitir la virtualización. Si los nodos de archivos y bloques se conectan mediante un switch, deberá habilitarse en el administrador de subredes de la estructura general.

Si los nodos de bloque y archivo se conectan directamente mediante InfiniBand, una instancia de `opensm` debe configurarse en cada nodo de archivo para cada interfaz conectada directamente a un nodo de bloque. Esto se realiza especificando `configure: true` cuando "configurar las interfaces de almacenamiento del nodo de archivo".

Actualmente, la versión de bandeja de entrada de `opensm` incluida con distribuciones de Linux compatibles no admite la virtualización. En su lugar, es necesario instalar y configurar la versión de `opensm` desde la distribución empresarial de OpenFabrics (OFED) de NVIDIA. A pesar de que todavía se admite la puesta en marcha con Ansible, se requieren algunos pasos adicionales:

1. Utilizando curl o la herramienta que desee, descargue los paquetes para la versión de OpenSM enumerados en la "requisitos tecnológicos" sección desde el sitio web de NVIDIA al `<INVENTORY>/packages/` directorio. Por ejemplo:

```
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/rhel9.3/x86_64/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm

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/rhel9.3/x86_64/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
```

2. Inferior `group_vars/ha_cluster.yml` defina la siguiente configuración:

```
### 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-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
          "packages/opensm-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"
```

Especifique la configuración de nodo de bloque común

Especifique la configuración de nodos de bloque común con las variables de grupo (group_var).

Descripción general

La configuración que debería Apple a todos los nodos de bloque se define en `group_vars/eseries_storage_systems.yml`. Normalmente incluye:

- Detalles sobre cómo el nodo de control Ansible debe conectarse a los sistemas de almacenamiento E-Series que se utilizan como nodos de bloques.

- Las versiones de firmware, NVSRAM y de unidad que deben ejecutar los nodos.
- Configuración global, que incluye la configuración de caché, la configuración de hosts y la configuración de cómo deben aprovisionarse los volúmenes.

> Las opciones establecidas en este archivo también pueden definirse en nodos de bloques individuales; por ejemplo, si se están utilizando modelos de hardware mixtos o tiene contraseñas diferentes para cada nodo. La configuración en nodos de bloque individuales tendrá prioridad sobre la configuración de este archivo.

Pasos

Cree el archivo `group_vars/eseries_storage_systems.yml` y rellenarlo de la siguiente manera:

1. Ansible no utiliza SSH para conectarse a los nodos de bloques y, en su lugar, utiliza API DE REST. Para lograrlo, debemos establecer:

```
ansible_connection: local
```

2. Especifique el nombre de usuario y la contraseña para gestionar cada nodo. El nombre de usuario puede omitirse opcionalmente (y, de forma predeterminada, admin), si no es posible especificar cualquier cuenta con privilegios de administrador. Especifique también si los certificados SSL deben verificarse o ignorarse:

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

> No se recomienda enumerar las contraseñas en texto sin formato. Use el almacén de Ansible o proporcione el `eseries_system_password` Al ejecutar Ansible con distribuidores de valor añadido de --extra.

3. Opcionalmente, especifique qué firmware de la controladora, NVSRAM y firmware de la unidad se debe instalar en los nodos. Deberá descargarse en el `packages/` directorio antes de ejecutar Ansible. El firmware de la controladora E-Series y NVSRAM se pueden descargar "aquí" y el firmware de la unidad "aquí":

```
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".
```

Si se especifica esta configuración, Ansible actualizará automáticamente todo el firmware, incluido el reinicio de las controladoras (si es necesario) sin ningún aviso adicional. Se espera que esto no sea disruptivo para la I/o del host de BeeGFS, pero podría provocar un descenso temporal del rendimiento.

4. Ajuste los valores predeterminados de configuración global del sistema. BeeGFS en NetApp suele recomendar las opciones y valores que se incluyen en esta lista, pero se pueden ajustar en caso necesario:

```
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. Configure las opciones predeterminadas de aprovisionamiento de volúmenes globales. BeeGFS en NetApp suele recomendar las opciones y valores que se incluyen en esta lista, pero se pueden ajustar en caso necesario:

```
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. Si es necesario, ajuste el orden en el que Ansible seleccionará las unidades para los pools de almacenamiento y los grupos de volúmenes, teniendo en cuenta las siguientes prácticas recomendadas:
   a. Enumere cualquier unidad (potencialmente menor) que se deben usar para los volúmenes de metadatos o gestión primero, y los volúmenes de almacenamiento en último lugar.
   b. Asegúrese de equilibrar el orden de selección de las unidades en los canales de unidad disponibles según los modelos de bandeja de discos/compartimento de unidades. Por ejemplo, con EF600 y sin expansiones, las unidades 0-11 están en el canal de unidades 1 y las unidades 12-23 están en el canal de unidades. Por lo tanto, una estrategia para equilibrar la selección de conducción es seleccionar `disk shelf:drive` 99:0, 99:23, 99:1, 99:22, etc. En el caso de que haya más de un compartimento, el primer dígito representa el ID de bandeja de unidades.

```
# 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"
```

Haga clic en "aquí" para obtener un ejemplo de un archivo de inventario completo que representa la configuración común de nodos de bloques.

Defina los servicios BeeGFS

Defina el servicio de gestión de BeeGFS

Los servicios BeeGFS se configuran mediante variables de grupo (Group_var).

Descripción general

En esta sección se describe la definición del servicio de gestión de BeeGFS. En los clústeres de alta disponibilidad solo debe haber un servicio de este tipo para un sistema de archivos concreto. La configuración de este servicio incluye la definición:

- El tipo de servicio (gestión).
- Definir cualquier configuración que sólo se debe aplicar a este servicio BeeGFS.
- Configuración de una o varias IP flotantes (interfaces lógicas) en las que se puede acceder a este servicio.
- Especificar dónde y cómo debe almacenar un volumen datos para este servicio (el objetivo de gestión de BeeGFS).

Pasos

Cree un archivo nuevo `group_vars/mgmt.yml` y haciendo referencia a la "Planifique el sistema de archivos" sección rellenarla de la siguiente forma:

1. Indique que este archivo representa la configuración de un servicio de administración de BeeGFS:

```
beegfs_service: management
```

2. Defina cualquier configuración que se deba aplicar sólo a este servicio BeeGFS. Esto no suele ser necesario para el servicio de gestión a menos que necesite habilitar cuotas, sin embargo, con cualquier parámetro de configuración admitido de `beegfs-mgmtd.conf` se puede incluir. Nota los siguientes parámetros se configuran automáticamente u otros lugares y no se deben especificar aquí: `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, y. `connNetFilterFile`.

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

3. Configure uno o varios IP flotantes que utilizarán otros servicios y clientes para conectarse a este servicio (esto establecerá automáticamente BeeGFS `connInterfacesFile` opción):

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

4. Opcionalmente, especifique una o varias subredes IP permitidas que se pueden utilizar para la comunicación saliente (esto establecerá automáticamente BeeGFS `connNetFilterFile` opción):

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

5. Especifique el objetivo de gestión de BeeGFS en el que este servicio almacenará datos de acuerdo con las siguientes directrices:
   a. Se puede utilizar el mismo nombre de pool de almacenamiento o grupo de volúmenes para varios servicios/objetivos de BeeGFS; asegúrese de utilizar el mismo `name`, `raid_level`, `criteria_*`, y. `common_*` la configuración de cada uno (los volúmenes enumerados para cada servicio deben ser diferentes).
   b. Los tamaños de los volúmenes se deben especificar como un porcentaje del pool de almacenamiento/grupo de volúmenes y el total no debe ser superior a 100 en todos los servicios/volúmenes que utilizan un pool de almacenamiento/grupo de volúmenes en particular. Nota cuando se usan SSD, se recomienda dejar espacio libre en el grupo de volúmenes para maximizar el rendimiento de SSD y la vida útil (haga clic en "aquí" para obtener más detalles).
   c. Haga clic en "aquí" para obtener una lista completa de las opciones de configuración disponibles para `eseries_storage_pool_configuration`. Tenga en cuenta algunas opciones como `state`, `host`, `host_type`, `workload_name`, y. `workload_metadata` y los nombres de volúmenes se generan automáticamente y no se deben especificar aquí.

```
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
```

Haga clic en "aquí" Para obtener un ejemplo de un archivo de inventario completo que representa un servicio de administración de BeeGFS.

Defina el servicio de metadatos BeeGFS

Los servicios BeeGFS se configuran mediante variables de grupo (Group_var).

Descripción general

En esta sección se describe la definición del servicio de metadatos de BeeGFS. Al menos debe haber un servicio de este tipo en los clústeres de alta disponibilidad para un sistema de archivos determinado. La configuración de este servicio incluye la definición:

- El tipo de servicio (metadatos).
- Definir cualquier configuración que sólo se debe aplicar a este servicio BeeGFS.
- Configuración de una o varias IP flotantes (interfaces lógicas) en las que se puede acceder a este servicio.
- Especificar dónde y cómo debe almacenar un volumen datos para este servicio (el objetivo de metadatos BeeGFS).

Pasos

Haciendo referencia a "Planifique el sistema de archivos" cree un archivo en `group_vars/meta_<ID>.yml` para cada servicio de metadatos del clúster y rellene los siguientes pasos:

1. Indique que este archivo representa la configuración de un servicio de metadatos BeeGFS:

```
beegfs_service: metadata
```

2. Defina cualquier configuración que se deba aplicar sólo a este servicio BeeGFS. Al mínimo debe especificar el puerto TCP y UDP deseado, sin embargo, cualquier parámetro de configuración compatible desde `beegfs-meta.conf` también se puede incluir. Nota los siguientes parámetros se configuran automáticamente u otros lugares y no se deben especificar aquí: `sysMgmtdHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, y. `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. Configure uno o varios IP flotantes que utilizarán otros servicios y clientes para conectarse a este servicio (esto establecerá automáticamente BeeGFS `connInterfacesFile` opción):

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

4. Opcionalmente, especifique una o varias subredes IP permitidas que se pueden utilizar para la comunicación saliente (esto establecerá automáticamente BeeGFS `connNetFilterFile` opción):

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

5. Especifique el destino de metadatos BeeGFS en el que este servicio almacenará datos de acuerdo con las siguientes directrices (también configurará automáticamente el `storeMetaDirectory` opción):
   a. Se puede utilizar el mismo nombre de pool de almacenamiento o grupo de volúmenes para varios servicios/objetivos de BeeGFS; asegúrese de utilizar el mismo `name`, `raid_level`, `criteria_*`, y. `common_*` la configuración de cada uno (los volúmenes enumerados para cada servicio deben ser diferentes).
   b. Los tamaños de los volúmenes se deben especificar como un porcentaje del pool de almacenamiento/grupo de volúmenes y el total no debe ser superior a 100 en todos los servicios/volúmenes que utilizan un pool de almacenamiento/grupo de volúmenes en particular. Nota cuando se usan SSD, se recomienda dejar espacio libre en el grupo de volúmenes para maximizar el rendimiento de SSD y la vida útil (haga clic en "aquí" para obtener más detalles).
   c. Haga clic en "aquí" para obtener una lista completa de las opciones de configuración disponibles para `eseries_storage_pool_configuration`. Tenga en cuenta algunas opciones como `state`, `host`, `host_type`, `workload_name`, y. `workload_metadata` y los nombres de volúmenes se generan automáticamente y no se deben especificar aquí.

```
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
```

Haga clic en "aquí" Para obtener un ejemplo de un archivo de inventario completo que representa un servicio de metadatos BeeGFS.

Defina el servicio de almacenamiento BeeGFS

Los servicios BeeGFS se configuran mediante variables de grupo (Group_var).

Descripción general

En esta sección se describe la definición del servicio de almacenamiento de BeeGFS. Al menos debe haber un servicio de este tipo en los clústeres de alta disponibilidad para un sistema de archivos determinado. La

configuración de este servicio incluye la definición:

- El tipo de servicio (almacenamiento).
- Definir cualquier configuración que sólo se debe aplicar a este servicio BeeGFS.
- Configuración de una o varias IP flotantes (interfaces lógicas) en las que se puede acceder a este servicio.
- Especificar dónde/cómo deben ser los volúmenes para almacenar datos para este servicio (los destinos de almacenamiento BeeGFS).

Pasos

Haciendo referencia a "Planifique el sistema de archivos" cree un archivo en `group_vars/stor_<ID>.yml` para cada servicio de almacenamiento del clúster y rellene los siguientes pasos:

1. Indique este archivo que representa la configuración de un servicio de almacenamiento BeeGFS:

```
beegfs_service: storage
```

2. Defina cualquier configuración que se deba aplicar sólo a este servicio BeeGFS. Al mínimo debe especificar el puerto TCP y UDP deseado, sin embargo, cualquier parámetro de configuración compatible desde `beegfs-storage.conf` también se puede incluir. Nota los siguientes parámetros se configuran automáticamente u otros lugares y no se deben especificar aquí: `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, y. `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. Configure uno o varios IP flotantes que utilizarán otros servicios y clientes para conectarse a este servicio (esto establecerá automáticamente BeeGFS `connInterfacesFile` opción):

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

4. Opcionalmente, especifique una o varias subredes IP permitidas que se pueden utilizar para la comunicación saliente (esto establecerá automáticamente BeeGFS `connNetFilterFile` opción):

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

5. Especifique los objetivos de almacenamiento de BeeGFS en los que este servicio almacenará datos de acuerdo con las siguientes directrices (también configurará automáticamente el `storeStorageDirectory` opción):

   a. Se puede utilizar el mismo nombre de pool de almacenamiento o grupo de volúmenes para varios servicios/objetivos de BeeGFS; asegúrese de utilizar el mismo `name`, `raid_level`, `criteria_*`, y. `common_*` la configuración de cada uno (los volúmenes enumerados para cada servicio deben ser diferentes).

   b. Los tamaños de los volúmenes se deben especificar como un porcentaje del pool de almacenamiento/grupo de volúmenes y el total no debe ser superior a 100 en todos los servicios/volúmenes que utilizan un pool de almacenamiento/grupo de volúmenes en particular. Nota cuando se usan SSD, se recomienda dejar espacio libre en el grupo de volúmenes para maximizar el rendimiento de SSD y la vida útil (haga clic en "aquí" para obtener más detalles).

   c. Haga clic en "aquí" para obtener una lista completa de las opciones de configuración disponibles para `eseries_storage_pool_configuration`. Tenga en cuenta algunas opciones como `state`, `host`, `host_type`, `workload_name`, y. `workload_metadata` y los nombres de volúmenes se generan automáticamente y no se deben especificar aquí.

```
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
```

Haga clic en "aquí" Para obtener un ejemplo de un archivo de inventario completo que representa un servicio de almacenamiento BeeGFS.

Asigne servicios BeeGFS a los nodos de archivo

Especifique qué nodos de archivo pueden ejecutar cada servicio BeeGFS mediante `inventory.yml` archivo.

Descripción general

En esta sección se explica cómo crear la `inventory.yml` archivo. Esto incluye una lista de todos los nodos de bloque y especificar qué nodos de archivo pueden ejecutar cada servicio BeeGFS.

Pasos

Cree el archivo `inventory.yml` y rellenarlo de la siguiente manera:

1. Desde la parte superior del archivo, cree la estructura de inventario estándar de Ansible:

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

2. Cree un grupo que contenga todos los nodos de bloques que participan en este clúster de alta disponibilidad:

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

3. Cree un grupo que contendrá todos los servicios BeeGFS del clúster y los nodos de archivo que los ejecutarán:

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

4. Para cada servicio BeeGFS del clúster, defina los nodos de archivos preferidos y secundarios que deben ejecutar ese servicio:

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

Haga clic en "aquí" para obtener un ejemplo de un archivo de inventario completo.

Implemente el sistema de archivos BeeGFS

Descripción general del libro de estrategia de Ansible

Puesta en marcha y gestión de clústeres de alta disponibilidad de BeeGFS mediante Ansible.

Descripción general

En las secciones anteriores se han realizado los pasos necesarios para crear un inventario de Ansible que represente un clúster de alta disponibilidad de BeeGFS. En esta sección se presenta la automatización de Ansible desarrollada por NetApp para poner en marcha y gestionar el clúster.

Ansible: Conceptos clave

Antes de continuar, es útil estar familiarizado con algunos conceptos clave de Ansible:

- Las tareas que se deben ejecutar con un inventario de Ansible se definen en lo que se conoce como **playbook**.
  - La mayoría de las tareas en Ansible están diseñadas para ser **idempotente**, lo que significa que pueden ejecutarse varias veces para verificar que la configuración/estado deseada todavía se aplica sin romper las cosas ni hacer actualizaciones innecesarias.
- La unidad más pequeña de ejecución en Ansible es un **módulo**.
  - Los libros de estrategia habituales utilizan varios módulos.
    - Ejemplos: Descargue un paquete, actualice un archivo de configuración, inicie/habilite un servicio.
  - NetApp distribuye módulos para automatizar los sistemas E-Series de NetApp.
- La automatización compleja se empaquetará mejor como rol.
  - Básicamente, un formato estándar para distribuir un libro de aplicaciones reutilizable.
  - NetApp distribuye roles para hosts Linux y sistemas de archivos BeeGFS.

Rol de ha de BeeGFS para Ansible: Conceptos clave

Toda la automatización necesaria para poner en marcha y gestionar cada versión de BeeGFS en NetApp se presenta como un rol de Ansible y se distribuye como parte de la "Colección de Ansible E-Series de NetApp para BeeGFS":

- Este papel se puede considerar como un lugar entre un motor de **instalador** y un motor de **implementación/administración** moderno para BeeGFS.
  - Aplica una infraestructura moderna como prácticas de código y filosofías para simplificar la gestión de infraestructura de almacenamiento a cualquier escala.
  - Similar a cómo "Kubespray" El proyecto permite a los usuarios poner en marcha/mantener una distribución completa de Kubernetes para la infraestructura informática de escalado horizontal.
- Este rol es el formato **definido por software** que utiliza NetApp para empaquetar, distribuir y mantener BeeGFS en soluciones NetApp.
  - Esforzarse por crear una experiencia “similar a un dispositivo” sin necesidad de distribuir una distribución entera de Linux o una imagen grande.
  - Incluye agentes de recursos en clúster compatibles con Open Cluster Framework (OCF) de NetApp

para objetivos BeeGFS personalizados, direcciones IP y supervisión que proporcionan una integración inteligente con Pacemaker/BeeGFS.

- Esta función no se limita a la puesta en marcha de la «automatización», sino que está destinada a gestionar todo el ciclo de vida del sistema de archivos, incluidos:
  - Aplicar cambios y actualizaciones de configuración por servicio o para todo el clúster.
  - Automatizar la reparación y recuperación del clúster después de resolver problemas de hardware.
  - Simplificación del ajuste del rendimiento con valores predeterminados establecidos en función de amplias pruebas con BeeGFS y volúmenes de NetApp.
  - Verificación y corrección de deriva de configuración.

NetApp también proporciona un rol de Ansible para "Clientes de BeeGFS", Que se puede utilizar opcionalmente para instalar BeeGFS y montar sistemas de archivos en nodos de cálculo/GPU/inicio de sesión.

Ponga en marcha el clúster de alta disponibilidad de BeeGFS

Especifique qué tareas se deben ejecutar para poner en marcha el clúster de alta disponibilidad de BeeGFS mediante un libro de aplicaciones.

Descripción general

En esta sección se describe cómo montar el libro de estrategia estándar utilizado para poner en marcha/gestionar BeeGFS en NetApp.

Pasos

Cree el libro de aplicaciones de Ansible

Cree el archivo `playbook.yml` y rellenarlo de la siguiente manera:

1. Defina primero un conjunto de tareas (comúnmente denominado a) "repr") Que solo se debe ejecutar en los nodos de bloques E-Series de NetApp. Utilizamos una tarea de pausa para preguntar antes de ejecutar la instalación (para evitar la ejecución accidental de playbook) y, a continuación, importar la `nar_santricity_management` función. Esta función se ocupa de aplicar cualquier configuración general del sistema definida en `group_vars/eseries_storage_systems.yml` o individual `host_vars/<BLOCK NODE>.yml` archivos.

```
- 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. Defina la reproducción que se ejecutará en todos los nodos de archivos y bloques:

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

3. En esta aplicación podemos definir opcionalmente un conjunto de «tareas previas» que se deben ejecutar antes de poner en marcha el clúster de alta disponibilidad. Esto puede ser útil para verificar/instalar requisitos previos como Python. También podemos inyectar comprobaciones previas al vuelo, por ejemplo, verificar que las etiquetas de Ansible proporcionadas son compatibles:

```
  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. Por último, este juego importa el rol de ha de BeeGFS para la versión de BeeGFS que desea implementar:

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

Se mantiene un rol de ha de BeeGFS para cada versión principal.secundaria de BeeGFS admitida. Esto permite a los usuarios elegir cuándo desean actualizar versiones principales o secundarias. Actualmente BeeGFS 7.3.x (`beegfs_7_3`) O BeeGFS 7.2.x. (`beegfs_7_2`) son compatibles. De forma predeterminada, ambos roles implementarán la versión más reciente del parche de BeeGFS en el momento de su publicación, aunque los usuarios pueden optar por anular este parche e implementar el último parche si lo desean. Consulte la información más reciente "guía de actualización" para obtener más detalles.

5. Opcional: Si desea definir tareas adicionales, tenga en cuenta si las tareas deben ser dirigidas `all` Los hosts (incluidos los sistemas de almacenamiento E-Series) o solo los nodos de archivos. Si es necesario, defina una nueva reproducción dirigida específicamente a los nodos de archivo mediante `- hosts: ha_cluster`.

Haga clic en "aquí" por ejemplo, un archivo de libro de estrategia completo.

Instale las colecciones Ansible de NetApp

Se mantiene la colección BeeGFS para Ansible y todas las dependencias "Galaxia de ansible". En su nodo de control de Ansible, ejecute el siguiente comando para instalar la versión más reciente:

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

Aunque normalmente no se recomienda, también es posible instalar una versión específica de la colección:

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

Ejecute el libro de aplicaciones

Desde el directorio del nodo de control de Ansible que contiene el `inventory.yml` y.. `playbook.yml` ejecute el libro de estrategia de la siguiente forma:

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

Según el tamaño del clúster, la puesta en marcha inicial puede tardar más de 20 minutos. Si se produce un error en la puesta en marcha por algún motivo, solo tiene que corregir los problemas (p. ej., un cableado incorrecto, nodo no se inició, etc.) y reiniciar el libro de estrategia de Ansible.

Al especificar "configuración común de nodos de archivos", Si elige la opción predeterminada para que Ansible administre automáticamente la autenticación basada en la conexión, un `connAuthFile` usado como secreto compartido ahora se puede encontrar en `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (de forma predeterminada). Cualquier cliente que necesite acceder al sistema de archivos tendrá que utilizar este secreto compartido. Esto se controla automáticamente si los clientes se configuran mediante el "Función de cliente de BeeGFS".

Implemente clientes BeeGFS

Opcionalmente, Ansible se puede usar para configurar los clientes de BeeGFS y montar

el sistema de archivos.

Descripción general

Para acceder a los sistemas de archivos BeeGFS es necesario instalar y configurar el cliente BeeGFS en cada nodo que necesite montar el sistema de archivos. En esta sección se documenta cómo realizar estas tareas mediante el útil disponible "Rol de Ansible".

Pasos

Cree el archivo de inventario de cliente

1. Si es necesario, configure SSH sin contraseñas desde el nodo de control de Ansible a cada uno de los hosts que desea configurar como clientes BeeGFS:

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

2. Inferior `host_vars/`, Cree un archivo para cada cliente BeeGFS denominado `<HOSTNAME>.yml` con el siguiente contenido, rellene el texto del marcador de posición con la información correcta para su entorno:

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

3. Incluya de manera opcional uno de los siguientes elementos si desea utilizar los roles de recogida de hosts E-Series de NetApp para configurar las interfaces InfiniBand o Ethernet para que los clientes se conecten a los nodos de archivos BeeGFS:
   a. Si el tipo de red es "InfiniBand (uso de IPoIB)":

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

   b. Si el tipo de red es "RDMA sobre 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. Si el tipo de red es "Ethernet (solo TCP, sin 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. Cree un archivo nuevo `client_inventory.yml` Y especifique el usuario que Ansible debe usar para conectarse a cada cliente y la contraseña que Ansible debe usar para el escalado de privilegios (esto requiere `ansible_ssh_user` sea raíz o tenga privilegios sudo):

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

> No almacene contraseñas en texto sin formato. En su lugar, utilice Ansible Vault (consulte "Documentación de Ansible" Para cifrar contenido con Ansible Vault) o usar el `--ask -become-pass` al ejecutar el libro de estrategia.

5. En la `client_inventory.yml` File, enumera todos los hosts que deben configurarse como clientes BeeGFS en `beegfs_clients` Agrupe y, a continuación, consulte los comentarios en línea y elimine los comentarios de cualquier configuración adicional necesaria para crear el módulo de kernel de cliente BeeGFS en su 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.
```

Cuando utilice los controladores OFED de NVIDIA, asegúrese de que beegfs_CLIENT_ofed_INCLUDE_PATH apunte a la ruta de acceso de inclusión de encabezado correcta para la instalación de Linux. Para obtener más información, consulte la documentación de BeeGFS para "Compatibilidad con RDMA".

6. En la `client_inventory.yml` Archivo, enumere los sistemas de archivos BeeGFS que desea montar en cualquiera de los definidos previamente `vars`:

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

7. A partir de BeeGFS 7.2.7 y 7.3.1 "autenticación de conexión" se debe configurar o deshabilitar explícitamente. Dependiendo de cómo elija configurar la autenticación basada en la conexión al especificar "configuración común de nodos de archivos", es posible que tenga que ajustar la configuración del cliente:
   a. De forma predeterminada, la puesta en marcha del clúster de alta disponibilidad configurará automáticamente la autenticación de conexiones y generará un `connauthfile` Que se colocará/mantendrá en el nodo de control de Ansible en `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. De forma predeterminada, la función de cliente BeeGFS está configurada para leer/distribuir este archivo a los clientes definidos en `client_inventory.yml`, y no se necesita ninguna acción adicional.
      i. Para obtener información sobre las opciones avanzadas, consulte la lista completa de valores predeterminados que se incluyen con la "Función de cliente de BeeGFS".
   b. Si decide especificar un secreto personalizado con `beegfs_ha_conn_auth_secret` especifique en la `client_inventory.yml` también archivo:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. Si decide deshabilitar la autenticación basada en conexión completamente con `beegfs_ha_conn_auth_enabled`, especifique que en la `client_inventory.yml` también

archivo:

```
beegfs_ha_conn_auth_enabled: false
```

Para obtener una lista completa de los parámetros admitidos y detalles adicionales, consulte la "Documentación completa del cliente de BeeGFS". Para ver un ejemplo completo de un inventario de cliente, haga clic en "aquí".

Cree el archivo del libro de aplicaciones del cliente BeeGFS

1. Cree un archivo nuevo `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. Opcional: Si desea utilizar los roles de la recogida de hosts de E-Series de NetApp para configurar interfaces para que los clientes se conecten a sistemas de archivos BeeGFS, importe el rol correspondiente al tipo de interfaz que está configurando:
   a. Si utiliza InfiniBand (IPoIB):

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

   b. Si utiliza RDMA over Converged Ethernet (roce):

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

   c. Si utiliza Ethernet (solo TCP, no RDMA):

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

3. Por último, importe la función de cliente de BeeGFS para instalar el software cliente y configurar los montajes del sistema de archivos:

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

Para ver un ejemplo completo de un libro de aplicaciones del cliente, haga clic en "aquí".

Ejecute el libro de aplicaciones del cliente BeeGFS

Para instalar/crear el cliente y montar BeeGFS, ejecute el siguiente comando:

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

Verifique la implementación de BeeGFS

Compruebe la implementación del sistema de archivos antes de colocar el sistema en producción.

Descripción general

Antes de poner el sistema de archivos BeeGFS en producción, realice algunas comprobaciones de verificación.

Pasos

1. Inicie sesión en cualquier cliente y ejecute lo siguiente para garantizar que todos los nodos esperados estén presentes o sean accesibles, no se han notificado inconsistencias ni otros problemas:

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

2. Apague todo el clúster y, a continuación, reinícielo. Desde cualquier nodo de archivo ejecute lo siguiente:

```
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. Coloque cada nodo en espera y compruebe que los servicios de BeeGFS pueden conmutar por error a nodos secundarios. Para realizar este inicio de sesión en cualquiera de los nodos de archivo y ejecute lo siguiente:

```
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. Utilizar herramientas de evaluación del rendimiento como IOR y MDTest para verificar que el rendimiento del sistema de archivos cumple las expectativas. En la se pueden encontrar ejemplos de pruebas y parámetros comunes utilizados con BeeGFS "Verificación del diseño" De BeeGFS en Arquitectura verificada de NetApp.

Las pruebas adicionales deben realizarse en función de los criterios de aceptación definidos para una instalación/emplazamiento particular.