솔루션 구축

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

This PDF was generated from https://docs.netapp.com/ko-kr/beegfs/second-gen/beegfs-deploy-overview.html on January 27, 2026. Always check docs.netapp.com for the latest.

목차

솔루션 구축

구축 개요

NetApp 기반 BeeGFS는 NetApp의 BeeGFS 구성 요소 설계와 함께 Ansible을 사용하여 검증된 파일 및 블록 노드에 배포할 수 있습니다.

Ansible 컬렉션 및 역할

NetApp 기반 BeeGFS 솔루션은 애플리케이션 배포를 자동화하는 주요 IT 자동화 엔진인 Ansible을 사용하여 배포됩니다. Ansible에서는 배포하려는 BeeGFS 파일 시스템을 모델링하는 인벤토리라고 하는 일련의 파일을 사용합니다.

Ansible을 사용하면 NetApp과 같은 회사에서 Ansible Galaxy에서 제공되는 컬렉션을 사용하여 기본 제공 기능을 확장할 수 있습니다(참조 "NetApp E-Series BeeGFS 컬렉션"). 컬렉션에는 특정 기능 또는 작업(예: E-Series 볼륨 생성)을 수행하는 모듈과 여러 모듈 및 기타 역할을 호출할 수 있는 역할이 포함됩니다. 이 자동화된 방식을 통해 BeeGFS 파일 시스템 및 기본 HA 클러스터를 구축하는 데 필요한 시간을 줄일 수 있습니다. 또한 클러스터와 BeeGFS 파일 시스템의 유지보수 및 확장을 간소화합니다.

자세한 내용은 을 참조하십시오 "Ansible 인벤토리에 대해 알아보십시오".

NetApp 솔루션에 BeeGFS를 구축하는 데 다양한 단계가 포함되므로 NetApp에서는 수동으로 솔루션 구축을 지원하지 않습니다.

BeeGFS 구성 요소에 대한 구성 프로필입니다

배포 절차는 다음과 같은 구성 프로파일을 다룹니다.

- 관리, 메타데이터 및 스토리지 서비스를 포함하는 하나의 기본 구성 요소입니다.
- 메타데이터와 스토리지 서비스가 포함된 두 번째 구성 요소입니다.
- 스토리지 서비스만 포함하는 세 번째 구성 요소입니다.

이러한 프로필을 통해 NetApp BeeGFS 구성 요소에 대한 권장 구성 프로필 전체 범위를 볼 수 있습니다. 각 구현 시 메타데이터 및 스토리지 구성 요소 또는 스토리지 서비스 전용 구성 요소의 수는 용량 및 성능 요구사항에 따라 달라질 수 있습니다.

배포 단계 개요

배포에는 다음과 같은 고급 작업이 포함됩니다.

하드웨어 구축

1. 각 구성 요소를 물리적으로 조립합니다.
2. 랙 및 케이블 하드웨어. 자세한 절차는 를 참조하십시오 "하드웨어 구축".

소프트웨어 구축

1. "파일 및 블록 노드 설정".

- 파일 노드에서 BMC IP를 구성합니다
- 지원되는 운영 체제를 설치하고 파일 노드에서 관리 네트워킹을 구성합니다
- 블록 노드에서 관리 IP를 구성합니다

2. "Ansible 제어 노드를 설정합니다".
3. "성능을 위해 시스템 설정을 조정합니다".
4. "Ansible 인벤토리를 작성합니다".
5. "BeeGFS 구성 요소에 대한 Ansible 인벤토리를 정의합니다".
6. "Ansible을 사용하여 BeeGFS 구축".
7. "BeeGFS 클라이언트를 구성합니다".

> ⓘ 배포 절차에는 텍스트를 파일로 복사해야 하는 몇 가지 예제가 포함되어 있습니다. 특정 배포에 맞게 수정해야 하거나 수정할 수 있는 모든 내용에 대해 "#" 또는 "//" 문자로 표시된 인라인 주석에 주의 깊게 접근하세요. 예를 들면 다음과 같습니다.
>
> ```
> `beegfs_ha_ntp_server_pools:  # THIS IS AN EXAMPLE OF A COMMENT!
>   - "pool 0.pool.ntp.org iburst maxsources 3"
>   - "pool 1.pool.ntp.org iburst maxsources 3"`
> ```

배포 권장 사항의 차이가 있는 파생 아키텍처:

- "고용량 빌딩 블록"

Ansible 인벤토리에 대해 알아보십시오

배포를 시작하기 전에 NetApp 기반 BeeGFS 솔루션을 배포하기 위해 Ansible을 구성 및 사용하는 방법에 대해 자세히 알아보십시오.

Ansible 인벤토리는 BeeGFS 시스템에 배포될 수 있는 파일 및 블록 노드를 나열하는 디렉토리 구조입니다. 여기에는 원하는 BeeGFS 파일 시스템을 설명하는 호스트, 그룹 및 변수가 포함됩니다. Ansible 인벤토리를 Ansible 제어 노드에 저장해야 합니다. Ansible 플레이북을 실행하는 데 사용되는 파일 및 블록 노드에 액세스할 수 있는 머신입니다. 샘플 재고는 에서 다운로드할 "NetApp E-Series BeeGFS GitHub를 참조하십시오" 수 있습니다.

Ansible 모듈 및 역할

Ansible 인벤토리에 설명된 구성을 적용하려면 엔드 투 엔드 솔루션을 구축하는 NetApp E-Series Ansible 컬렉션(에서 사용 가능)에 제공되는 다양한 Ansible 모듈 및 역할을 "NetApp E-Series BeeGFS GitHub를 참조하십시오" 사용하십시오.

NetApp E-Series Ansible 컬렉션에서 각 역할은 NetApp 솔루션 기반의 BeeGFS를 완벽하게 구축하는 데 있습니다. 이 역할은 NetApp E-Series SANtricity, 호스트 및 BeeGFS 컬렉션을 사용하여 HA(High Availability)를 통해 BeeGFS 파일 시스템을 구성할 수 있습니다. 그런 다음 스토리지를 프로비저닝하고 매핑하고 클러스터 스토리지를 사용할 준비가 되었는지 확인할 수 있습니다.

역할에 맞는 심층적인 문서가 제공되지만, 구축 절차에서는 제2세대 BeeGFS 구성 요소 설계를 사용하여 NetApp 검증

아키텍처를 구축하는 데 역할을 사용하는 방법에 대해 설명합니다.

Ansible에 대한 사전 경험이 사전 필수 요소가 될 수 있도록 구축 단계에서 자세한 정보를 제공하려고 하지만, Ansible 및 관련 용어에 친숙해야 합니다.

BeeGFS HA 클러스터의 인벤토리 레이아웃

Ansible 인벤토리 구조를 사용하여 BeeGFS HA 클러스터를 정의합니다.

이전의 Ansible 경험을 보유한 사용자는 BeeGFS HA 역할이 각 호스트에 적용되는 변수(또는 사실)를 파악하기 위한 사용자 지정 방법을 구현한다는 점을 알아야 합니다. 이 설계는 Ansible 인벤토리를 구성하여 여러 서버에서 실행할 수 있는 리소스를 설명하는 과정을 간소화합니다.

Ansible 인벤토리는 일반적으로 및 `group_vars` 의 파일과 함께 `inventory.yml` 호스트를 특정 그룹(그리고 잠재적으로 다른 그룹에 할당하는 파일)으로 `host_vars` 구성됩니다.

본 하위 섹션의 내용을 포함하는 파일을 만들지 마십시오. 이 내용은 예제로만 제공됩니다.

이 구성은 구성 프로필을 기반으로 사전 결정됩니다. 하지만 다음과 같이 Ansible 인벤토리로 모든 내용을 레이아웃하는 방법을 전반적으로 이해해야 합니다.

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        netapp01:
        netapp02:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
        meta_01:  # Group representing a metadata service with ID 01.
          hosts:
            beegfs_01:  # This service is preferred on the first file
node.
            beegfs_02:  # And can failover to the second file node.
        meta_02:  # Group representing a metadata service with ID 02.
          hosts:
            beegfs_02:  # This service is preferred on the second file
node.
            beegfs_01: # And can failover to the first file node.
```

각 서비스에 대해 해당 구성을 설명하는 group_vars 아래에 추가 파일이 생성됩니다.

```
# meta_01 - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: 8015
  connMetaPortUDP: 8015
  tuneBindToNumaZone: 0
floating_ips:
  - i1b: <IP>/<SUBNET_MASK>
  - i2b: <IP>/<SUBNET_MASK>
# Type of BeeGFS service the HA resource group will manage.
beegfs_service: metadata # Choices: management, metadata, storage.
# What block node should be used to create a volume for this service:
beegfs_targets:
  netapp01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.25
            owning_controller: A
```

이 레이아웃을 통해 각 리소스에 대한 BeeGFS 서비스, 네트워크 및 스토리지 구성을 단일 위치에서 정의할 수 있습니다. BeeGFS 역할은 이러한 인벤토리 구조를 기반으로 각 파일 및 블록 노드에 필요한 구성을 집계합니다.

> 각 서비스의 BeeGFS 숫자 및 문자열 노드 ID는 그룹 이름을 기준으로 자동으로 구성됩니다. 따라서 그룹 이름이 고유해야 하는 일반적인 Ansible 요구 사항 외에도 BeeGFS 서비스를 나타내는 그룹은 해당 그룹이 나타내는 BeeGFS 서비스 유형에 고유한 번호로 끝나야 합니다. 예를 들어, meta_01 및 stor_01은 허용되지만 metadata_01 및 meta_01은 허용되지 않습니다.

모범 사례를 검토합니다

NetApp 솔루션에 BeeGFS를 구축할 때는 모범 사례 지침을 따르십시오.

표준 규약

Ansible 인벤토리 파일을 물리적으로 조립하고 생성할 때는 다음 표준 규칙을 따르십시오(자세한 내용은 을 참조하십시오 "Ansible 인벤토리를 작성합니다")를 클릭합니다.

- 파일 노드 호스트 이름은 랙 상단에 더 적은 숫자가 있고 하단에 더 높은 숫자가 있는 순서대로 번호가 지정됩니다(H01-HN).

  예를 들어 명명 규칙은 `[location][row][rack]hN` 다음과 같습니다 `beegfs_01`.

- 각 블록 노드는 각각 고유한 호스트 이름을 가진 두 개의 스토리지 컨트롤러로 구성됩니다.

스토리지 어레이 이름은 Ansible 인벤토리의 일부로 전체 블록 스토리지 시스템을 나타내는 데 사용됩니다. 스토리지 배열 이름은 순서대로 번호(A01-AN)로 지정되어야 하며, 개별 컨트롤러의 호스트 이름은 해당 명명 규칙에서 파생됩니다.

예를 들어, 이라는 이름의 블록 노드는 `ictad22a01` 일반적으로 및 와 같은 각 컨트롤러에 대해 구성된 호스트 이름을 가질 수 `ictad22a01-a ictad22a01-b`있지만, `Ansible` 인벤토리에서 로 지칭됩니다 `netapp_01`.

- 동일한 빌딩 블록 내의 파일 및 블록 노드는 동일한 번호 지정 체계를 공유하며, 랙의 서로 인접해 있으며 두 파일 노드 모두 위에 있고 두 블록 노드 바로 아래에 있습니다.

  예를 들어 첫 번째 빌딩 블록에서 파일 노드 H01 및 H02는 모두 블록 노드 A01 및 A02에 직접 연결됩니다. 위에서 아래로 호스트 이름은 H01, H02, A01 및 A02입니다.

- 빌딩 블록은 호스트 이름을 기준으로 순차적으로 설치되므로 번호가 낮은 호스트 이름은 랙 상단에, 번호가 높은 호스트 이름은 하단에 표시됩니다.

  이는 랙 스위치 상단으로 연결되는 케이블의 길이를 최소화하고 문제 해결을 단순화하기 위한 표준 배포 방법을 정의하는 것입니다. 랙 안정성 문제로 인해 이것이 허용되지 않는 데이터 센터의 경우, 맨 아래부터 랙을 채우는 역작업이 허용됩니다.

InfiniBand 스토리지 네트워크 구성

각 파일 노드의 InfiniBand 포트 중 절반은 블록 노드에 직접 연결하는 데 사용됩니다. 나머지 절반은 InfiniBand 스위치에 연결되며 BeeGFS 클라이언트-서버 연결에 사용됩니다. BeeGFS 클라이언트 및 서버에 사용되는 IPoIB 서브넷의 크기를 결정할 때 예상되는 컴퓨팅/GPU 클러스터 및 BeeGFS 파일 시스템 확장을 고려해야 합니다. 권장 IP 범위를 벗어나야 하는 경우, 단일 빌딩 블록의 각 직접 접속은 고유한 서브넷을 가지며 클라이언트-서버 접속에 사용되는 서브넷과 중복되지 않는다는 점에 유의하십시오.

직접 연결

각 빌딩 블록 내의 파일 및 블록 노드는 항상 직접 연결에 다음 표의 IP를 사용합니다.

이 주소 지정 체계는 다음 규칙을 따릅니다. 세 번째 옥텟은 항상 홀수이거나 짝수이며, 이는 파일 노드가 홀수인지 아니면 짝수인지에 따라 다릅니다.

파일 노드	**IB** 포트	**IP** 주소입니다	블록 노드	**IB** 포트	물리적 **IP**	가상 **IP**
홀수(h1)	i1a	192.168.1.10	홀수(C1)	2A	192.168.1.100	192.168.1.101
홀수(h1)	i2a	192.168.3.10	홀수(C1)	2A	192.168.3.100	192.168.3.101
홀수(h1)	i3a	192.168.5.10	짝수(C2)	2A	192.168.5.100	192.168.5.101
홀수(h1)	i4a	192.168.7.10	짝수(C2)	2A	192.168.7.100	192.168.7.101
짝수(H2)	i1a	192.168.2.10	홀수(C1)	2B	192.168.2.100	192.168.2.101
짝수(H2)	i2a	192.168.4.10	홀수(C1)	2B	192.168.4.100	192.168.4.101
짝수(H2)	i3a	192.168.6.10	짝수(C2)	2B	192.168.6.100	192.168.6.101
짝수(H2)	i4a	192.168.8.10	짝수(C2)	2B	192.168.8.100	192.168.8.101

BeeGFS 클라이언트-서버 IPoIB 주소 지정 체계

각 파일 노드에서 여러 BeeGFS 서버 서비스(관리, 메타데이터 또는 스토리지)를 실행합니다. 각 서비스가 다른 파일 노드로 독립적으로 페일오버할 수 있도록 각 서비스마다 고유한 IP 주소가 구성되며 이 주소는 두 노드 간에 자유롭게 움직일 수 있습니다(LIF라고도 함).

필수 사항은 아니지만 이 구축 환경에서 이러한 연결에 다음 IPoIB 서브넷 범위가 사용 중인 것으로 가정하며 다음 규칙을 적용하는 표준 주소 지정 체계를 정의합니다.

- 두 번째 옥텟은 파일 노드 InfiniBand 포트가 홀수인지 또는 짝수인지에 따라 항상 홀수이거나 짝수 입니다.
- BeeGFS 클러스터 IP는 항상 xxx입니다. 127.100.yyy 또는 xxx.128.100.yyy.

> ⓘ 대역 내 OS 관리에 사용되는 인터페이스 외에도 클러스터 심장 박동 및 동기화를 위한 Corosync에서 추가 인터페이스를 사용할 수 있습니다. 따라서 단일 인터페이스가 손실되어도 전체 클러스터가 다운되지 않습니다.

- BeeGFS Management 서비스는 항상 xxx.yyy.101.0 또는 xxx.yyy.102.0 중 입니다.
- BeeGFS 메타데이터 서비스는 항상 xxx.yyy.101.zzz 또는 xxx.yyy.102.zzz입니다.
- BeeGFS 스토리지 서비스는 항상 `xxx.yyy.103.zzz` 또는 `xxx.yyy.104.zzz`입니다.
- 100.xxx.1.1 ~ 100.xxx.99.255 범위의 주소는 고객용으로 예약되어 있습니다.

IPoIB 단일 서브넷 주소 지정 체계

이 배포 가이드에서는 에 나와 있는 이점을 감안하여 단일 서브넷 스키마를 "소프트웨어 아키텍처"활용합니다.

서브넷**: 100.127.0.0/16**

다음 표에는 단일 서브넷의 범위가 나와 있습니다. 100.127.0.0/16.

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클러스터 IP입니다	i1b 또는 i4b	100.127.100.1-100.127.100.255
BeeGFS 관리	i1b	100.127.101.0
	i2b	100.127.102.0
BeeGFS 메타데이터	i1b 또는 i3b	100.127.101.1 - 100.127.101.255
	i2b 또는 i4b	100.127.102.1 - 100.127.102.255
BeeGFS 스토리지	i1b 또는 i3b	100.127.103.1 - 100.127.103.255
	i2b 또는 i4b	100.127.104.1 - 100.127.104.255
BeeGFS 클라이언트	(클라이언트에 따라 다름)	100.127.1.1 - 100.127.99.255

IPoIB 두 개의 서브넷 주소 지정 체계

두 개의 서브넷 주소 지정 체계는 더 이상 권장되지 않지만 여전히 구현할 수 있습니다. 권장되는 두 개의 서브넷 구성표는 아래 표를 참조하십시오.

서브넷 **A: 100.127.0.0/16**

다음 표에는 서브넷 A:100.127.0.0/16의 범위가 나와 있습니다.

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클러스터 IP입니다	i1b	100.127.100.1-100.127.100.255
BeeGFS 관리	i1b	100.127.101.0
BeeGFS 메타데이터	i1b 또는 i3b	100.127.101.1 - 100.127.101.255
BeeGFS 스토리지	i1b 또는 i3b	100.127.103.1 - 100.127.103.255
BeeGFS 클라이언트	(클라이언트에 따라 다름)	100.127.1.1 - 100.127.99.255

서브넷 **B: 100.128.0.0/16**

다음 표에는 서브넷 B:100.128.0.0/16의 범위가 나와 있습니다.

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클러스터 IP입니다	i4b	100.128.100.1-100.128.100.255
BeeGFS 관리	i2b	100.128.102.0
BeeGFS 메타데이터	i2b 또는 i4b	100.128.102.1-100.128.102.255
BeeGFS 스토리지	i2b 또는 i4b	100.128.104.1 - 100.128.104.255
BeeGFS 클라이언트	(클라이언트에 따라 다름)	100.128.1.1-100.128.99.255

위 범위에 있는 모든 IP가 이 NetApp 검증 아키텍처에 사용되는 것은 아닙니다. 또한 IP 주소를 사전 할당하여 일관된 IP 주소 지정 체계를 사용하여 파일 시스템을 쉽게 확장할 수 있는 방법을 보여 줍니다. 이 스키마에서는 BeeGFS 파일 노드 및 서비스 ID가 잘 알려진 IP 범위의 네 번째 옥텟과 일치합니다. 필요한 경우 파일 시스템을 255개 노드 또는 서비스 이상으로 확장할 수 있습니다.

하드웨어 구축

각 구성 요소는 HDR(200GB) InfiniBand 케이블을 사용하여 두 블록 노드에 직접 연결된 검증된 x86 파일 노드 2개로 구성됩니다.

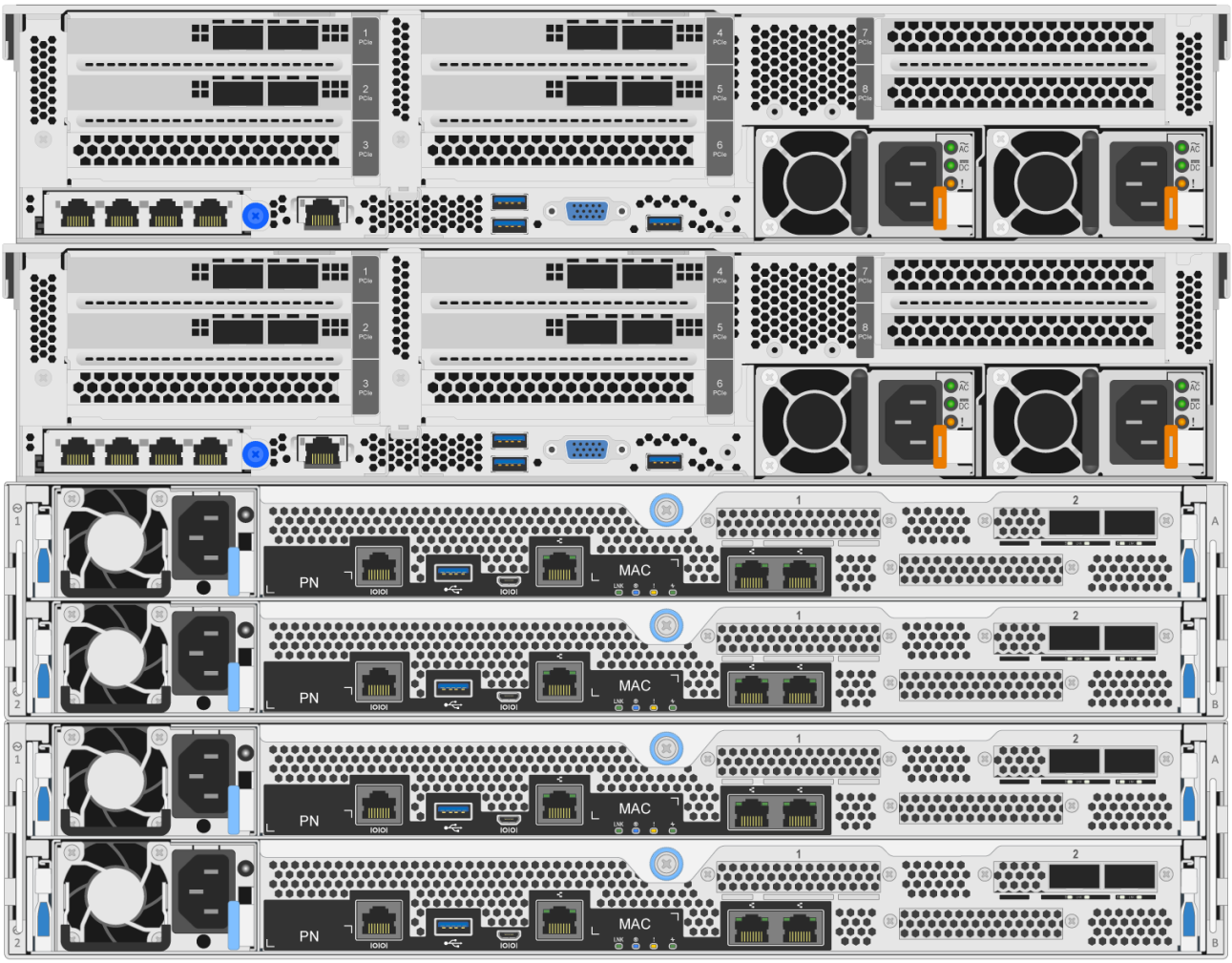
장애 조치 클러스터에서 쿼럼을 설정하려면 최소 2개의 구성 요소가 필요합니다. 2노드 클러스터에는 성공적인 페일오버를 수행할 수 없는 제한 사항이 있습니다. 세 번째 장치를 Tiebreaker로 통합하여 2노드 클러스터를 구성할 수 있지만, 이 문서에서는 그러한 설계에 대해 설명하지 않습니다.

BeeGFS 메타데이터와 스토리지 서비스를 모두 실행하는 데 사용되는지 아니면 스토리지 서비스만 실행하는 데 사용되는지에 관계없이 클러스터의 각 구성 요소에 대해 다음 단계는 동일합니다.

단계

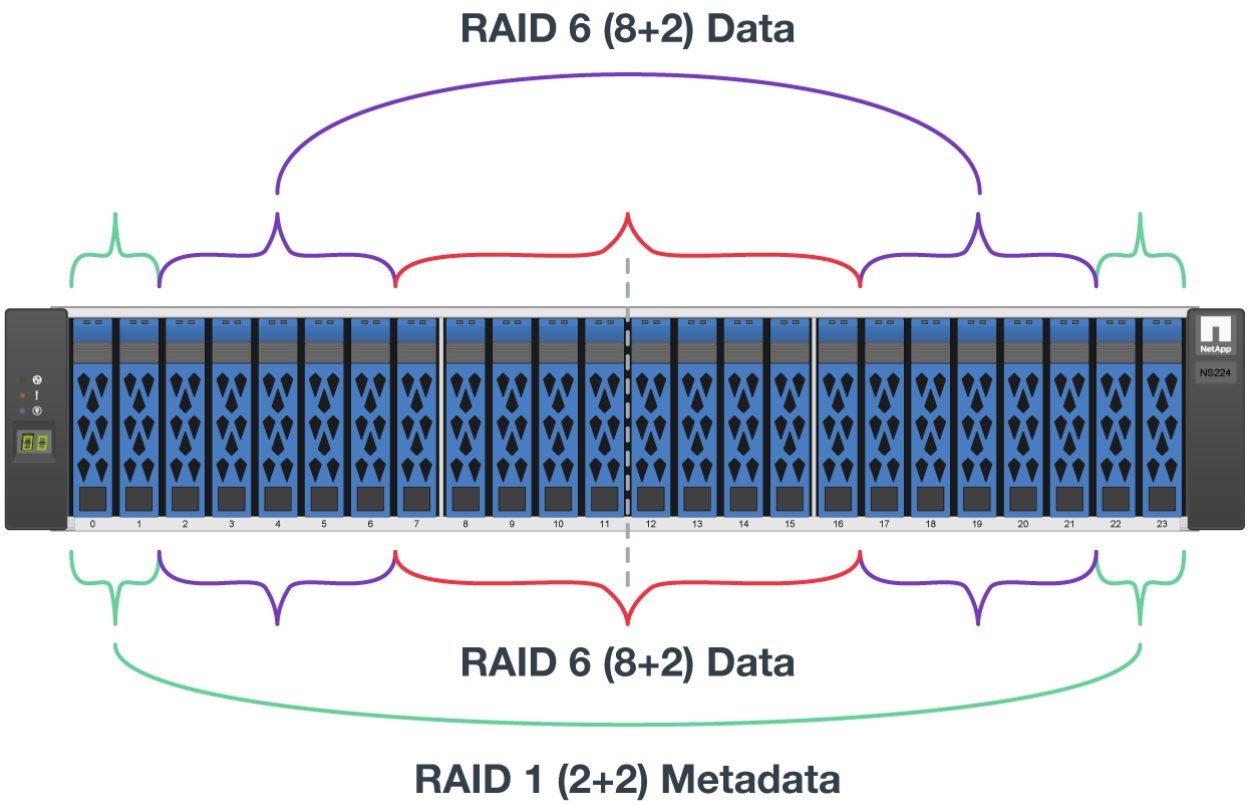
1. 에 지정된 모델을 사용하여 4개의 HCA(호스트 채널 어댑터)로 각 BeeGFS 파일 노드를 "기술 요구사항" 설정합니다. 아래 사양에 따라 파일 노드의 PCIe 슬롯에 HCA를 삽입합니다.
   - * Lenovo ThinkSystem SR665 V3 서버: * PCIe 슬롯 1, 2, 4 및 5를 사용합니다.
   - * Lenovo ThinkSystem SR665 서버: * PCIe 슬롯 2, 3, 5 및 6을 사용합니다.
2. 이중 포트 200GB 호스트 인터페이스 카드(HIC)로 각 BeeGFS 블록 노드를 구성하고 두 스토리지 컨트롤러 각각에 HIC를 설치합니다.

두 개의 BeeGFS 파일 노드가 BeeGFS 블록 노드 위에 있도록 구성 요소를 랙에 설치하십시오. 다음 그림은 Lenovo ThinkSystem SR665 V3 서버를 파일 노드로 사용하는 BeeGFS 구성 요소의 올바른 하드웨어 구성을 보여줍니다(후면).

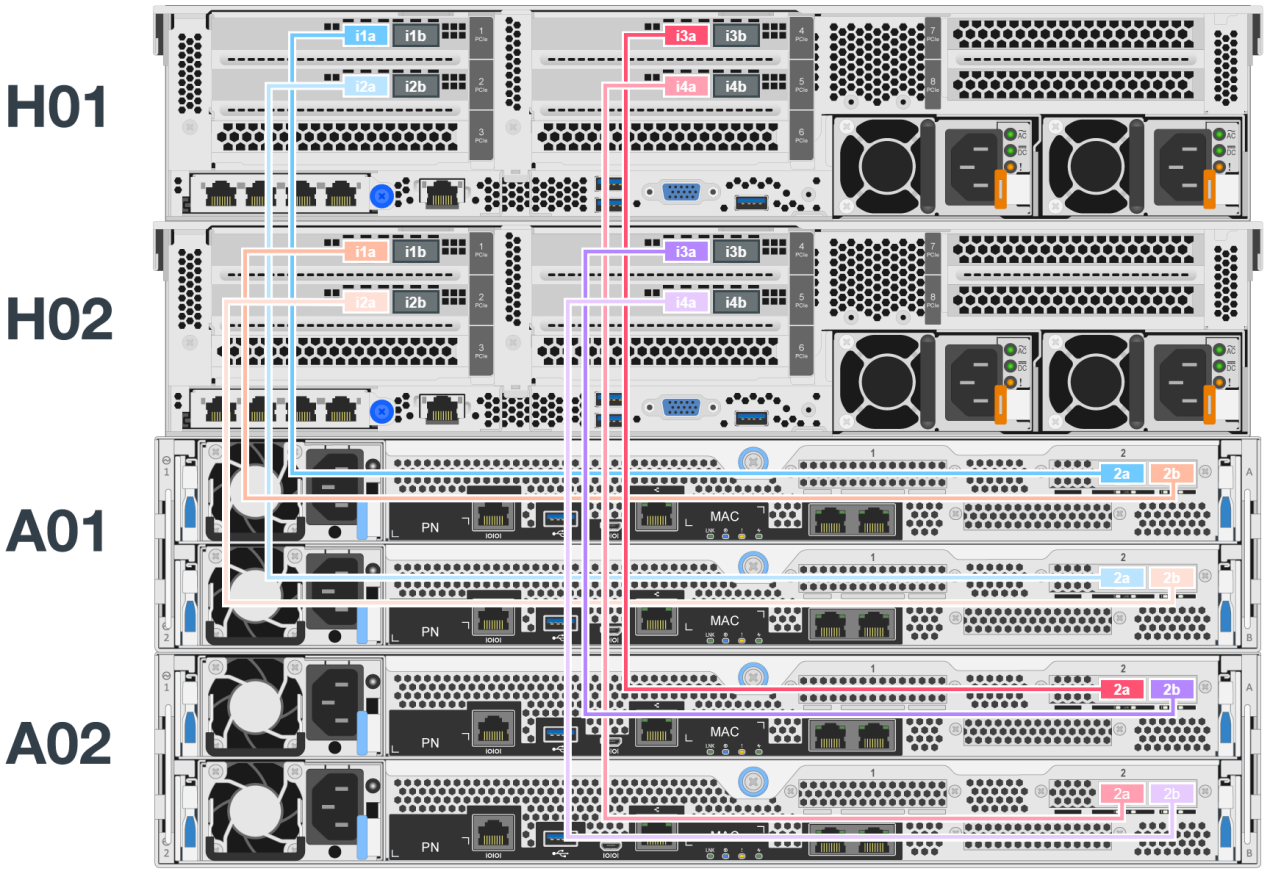


운영 활용 사례에 대한 전원 공급 장치 구성은 일반적으로 중복 PSU를 사용해야 합니다.

3. 필요한 경우 각 BeeGFS 블록 노드에 드라이브를 설치합니다.
   a. 빌딩 블록을 사용하여 BeeGFS 메타데이터 및 스토리지 서비스를 실행하고 작은 드라이브를 메타데이터 볼륨에 사용하는 경우 아래 그림과 같이 가장 바깥쪽 드라이브 슬롯에 채워졌는지 확인합니다.
   b. 모든 구성 요소 구성에서 드라이브 엔클로저가 완전히 채워지지 않은 경우 최적의 성능을 위해 슬롯 0–11 및 12–23에 동일한 수의 드라이브가 채워졌는지 확인하십시오.



4. 다음 그림에 표시된 토폴로지와 일치하도록 를 사용하여 블록 및 파일 노드를 "1m InfiniBand HDR 200GB 직접 연결 구리 케이블"연결합니다.

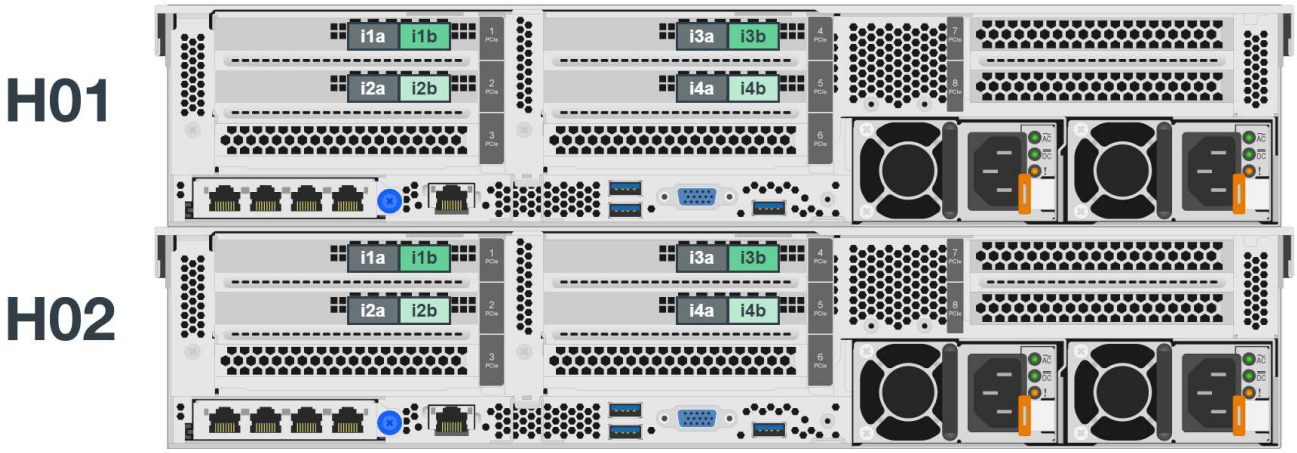


여러 빌딩 블록의 노드는 직접 연결되지 않습니다. 각 구성 요소는 독립형 장치로 취급해야 하며 구성 요소 간의 모든 통신은 네트워크 스위치를 통해 이루어집니다.

5. InfiniBand 스토리지 스위치에 해당하는 를 사용하여 파일 노드의 나머지 InfiniBand 포트를 스토리지 네트워크의 InfiniBand 스위치에 "2M InfiniBand 케이블" 연결합니다.

   Splitter 케이블을 사용하여 스토리지 스위치를 파일 노드에 연결하는 경우 케이블 하나가 스위치에서 분기되어 밝은 녹색으로 표시된 포트에 연결되어야 합니다. 다른 스플리터 케이블이 스위치에서 나와서 어두운 녹색으로 표시된 포트에 연결해야 합니다.

   또한 중복 스위치가 있는 스토리지 네트워크의 경우 연한 녹색으로 표시된 포트가 하나의 스위치에 연결되고 진한 녹색의 포트는 다른 스위치에 연결해야 합니다.



6. 필요한 경우 동일한 케이블 연결 지침에 따라 추가 구성 요소를 조립합니다.

   단일 랙에 구축할 수 있는 총 구성 요소 수는 각 사이트의 사용 가능한 전력 및 냉각에 따라 다릅니다.

소프트웨어 배포

파일 노드 및 블록 노드 설정

대부분의 소프트웨어 구성 작업은 NetApp에서 제공하는 Ansible 컬렉션을 사용하여 자동화되지만, 각 서버의 BMC(베이스보드 관리 컨트롤러)에서 네트워킹을 구성하고 각 컨트롤러의 관리 포트를 구성해야 합니다.

파일 노드 설정

1. 각 서버의 BMC(베이스보드 관리 컨트롤러)에서 네트워킹을 구성합니다.

   검증된 Lenovo SR665 V3 파일 노드에 대한 네트워킹을 구성하는 방법은 을 참조하십시오 "Lenovo ThinkSystem 설명서".

서비스 프로세서라고도 하는 베이스보드 관리 컨트롤러(BMC)는 다양한 서버 플랫폼에 내장되어 운영 체제가 설치되어 있지 않거나 액세스할 수 없는 경우에도 원격 액세스를 제공할 수 있는 대역외 관리 기능의 일반 이름입니다. 공급업체는 일반적으로 고유한 브랜딩으로 이 기능을 마케팅합니다. 예를 들어, Lenovo SR665에서 BMC는 _Lenovo XClarity Controller(XCC)_라고 합니다.

2. 최대 성능을 위해 시스템 설정을 구성합니다.

   UEFI 설정(이전의 BIOS)을 사용하거나 많은 BMC에서 제공하는 Redfish API를 사용하여 시스템 설정을 구성합니다. 시스템 설정은 파일 노드로 사용되는 서버 모델에 따라 달라집니다.

   검증된 Lenovo SR665 V3 파일 노드에 대한 시스템 설정을 구성하는 방법을 알아보려면 다음을 참조하세요. "성능을 위해 시스템 설정을 조정합니다" .

3. Red Hat Enterprise Linux(RHEL) 9.4를 설치하고 Ansible 제어 노드에서 SSH 연결을 포함하여 운영 체제를 관리하는 데 사용되는 호스트 이름과 네트워크 포트를 구성합니다.

   지금은 InfiniBand 포트에 IP를 구성하지 마십시오.

   엄밀히 요구되지는 않지만, 이후의 섹션에서는 호스트 이름이 순차적으로 번호가 매겨진 것으로 간주하고(예: h1-hn) 홀수 호스트와 짝수 번호의 호스트에서 완료해야 하는 작업을 참조합니다.

4. Red Hat Subscription Manager를 사용하여 시스템을 등록하고 구독하면 공식 Red Hat 저장소에서 필요한 패키지를 설치할 수 있고 지원되는 Red Hat 버전으로 업데이트를 제한할 수 있습니다. `subscription-manager release --set=9.4` . 자세한 내용은 및 을 "RHEL 시스템을 등록하고 가입하는 방법" "업데이트 제한 방법"참조하십시오.

5. 고가용성을 위해 필요한 패키지가 포함된 Red Hat 리포지토리를 활성화합니다.

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

6. 모든 HCA 펌웨어를 "파일 노드 어댑터 펌웨어를 업데이트합니다"가이드 사용에 권장되는 버전으로 "기술 요구 사항"업데이트합니다.

블록 노드 설정

각 컨트롤러의 관리 포트를 구성하여 EF600 블록 노드를 설정합니다.

1. 각 EF600 컨트롤러의 관리 포트를 구성합니다.

   포트 구성에 대한 자세한 내용은 를 "E-Series 문서 센터 를 참조하십시오" 참조하십시오.

2. 필요에 따라 각 시스템의 스토리지 어레이 이름을 설정합니다.

   이름을 설정하면 이후 섹션에서 각 시스템을 쉽게 참조할 수 있습니다. 배열 이름 설정에 대한 자세한 내용은 을 "E-Series 문서 센터 를 참조하십시오" 참조하십시오.

엄밀히 요구되지는 않지만, 후속 주제는 스토리지 배열 이름이 순차적으로 번호가 매겨진 것으로 간주하고(예: C1-CN) 홀수 대 짝수 번호의 시스템에서 완료해야 하는 단계를 참조합니다.

성능을 위해 파일 노드 시스템 설정을 조정합니다

성능을 최대화하려면 파일 노드로 사용하는 서버 모델에서 시스템 설정을 구성하는 것이 좋습니다.

시스템 설정은 파일 노드로 사용하는 서버 모델에 따라 달라집니다. 이 항목에서는 검증된 Lenovo ThinkSystem SR665 서버 파일 노드에 대한 시스템 설정을 구성하는 방법에 대해 설명합니다.

UEFI 인터페이스를 사용하여 시스템 설정을 조정합니다

Lenovo SR665 V3 서버의 시스템 펌웨어에는 UEFI 인터페이스를 통해 설정할 수 있는 다양한 튜닝 매개변수가 포함되어 있습니다. 이러한 튜닝 매개 변수는 서버의 작동 방식 및 서버 성능에 미치는 모든 측면에 영향을 줄 수 있습니다.

UEFI 설정 > 시스템 설정 * 에서 다음 시스템 설정을 조정합니다.

작동 모드 메뉴

* 시스템 설정 *	* 로 변경합니다
작동 모드	맞춤형
cTDP	수동
cTDP 설명서	350
패키지 전력 제한	수동
효율성 모드	사용 안 함
Global-Cstate-Control	사용 안 함
SOC P 상태	P0
DF C 상태	사용 안 함
P - 상태	사용 안 함
메모리 전원 끄기 활성화	사용 안 함
소켓당 NUMA 노드	NPS1

Device and I/O ports(장치 및 **I/O** 포트) 메뉴

* 시스템 설정 *	* 로 변경합니다
IOMMU	사용 안 함

전원 메뉴

* 시스템 설정 *	* 로 변경합니다
PCIe 전원 브레이크	사용 안 함

프로세서 메뉴

* 시스템 설정 *	* 로 변경합니다
글로벌 C 상태 제어	사용 안 함
DF C 상태	사용 안 함
SMT 모드	사용 안 함
CPPC	사용 안 함

Redfish API를 사용하여 시스템 설정을 조정합니다

UEFI 설정 외에도 Redfish API를 사용하여 시스템 설정을 변경할 수 있습니다.

```
curl --request PATCH \
  --url https://<BMC_IP_ADDRESS>/redfish/v1/Systems/1/Bios/Pending \
  --user <BMC_USER>:<BMC- PASSWORD> \
  --header 'Content-Type: application/json' \
  --data '{
"Attributes": {
"OperatingModes_ChooseOperatingMode": "CustomMode",
"Processors_cTDP": "Manual",
"Processors_PackagePowerLimit": "Manual",
"Power_EfficiencyMode": "Disable",
"Processors_GlobalC_stateControl": "Disable",
"Processors_SOCP_states": "P0",
"Processors_DFC_States": "Disable",
"Processors_P_State": "Disable",
"Memory_MemoryPowerDownEnable": "Disable",
"DevicesandIOPorts_IOMMU": "Disable",
"Power_PCIePowerBrake": "Disable",
"Processors_GlobalC_stateControl": "Disable",
"Processors_DFC_States": "Disable",
"Processors_SMTMode": "Disable",
"Processors_CPPC": "Disable",
"Memory_NUMANodesperSocket":"NPS1"
}
}
'
```

Redfish 스키마에 대한 자세한 내용은 를 참조하십시오 "DMTF 웹 사이트".

Ansible 제어 노드를 설정합니다

Ansible 제어 노드를 설정하려면 NetApp 솔루션 기반 BeeGFS 솔루션용으로 구축된 모든 파일 및 블록 노드에 대한 네트워크 액세스가 가능한 가상 머신 또는 물리적 머신을 지정해야 합니다.

권장되는 패키지 버전 목록은 를 "기술 요구사항"참조하십시오. 다음 단계는 Ubuntu 22.04에서 테스트되었습니다. 선호하는 Linux 배포와 관련된 단계는 를 "Ansible 설명서"참조하십시오.

1. Ansible 제어 노드에서 다음 Python 및 Python 가상 환경 패키지를 설치합니다.

```
sudo apt-get install python3 python3-pip python3-setuptools python3.10-venv
```

2. Python 가상 환경을 만듭니다.

```
python3 -m venv ~/pyenv
```

3. 가상 환경을 활성화합니다.

```
source ~/pyenv/bin/activate
```

4. 활성화된 가상 환경 내에 필요한 Python 패키지를 설치합니다.

```
pip install ansible netaddr cryptography passlib
```

5. Ansible Galaxy를 사용하여 BeeGFS 컬렉션을 설치합니다.

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

6. 설치된 Ansible, Python 및 BeeGFS 컬렉션 버전이 과 일치하는지 확인합니다"기술 요구사항".

```
ansible --version
ansible-galaxy collection list netapp_eseries.beegfs
```

7. Ansible이 Ansible 제어 노드에서 원격 BeeGFS 파일 노드에 액세스할 수 있도록 암호 없는 SSH를 설정합니다.
   a. 필요한 경우 Ansible 제어 노드에서 공용 키 쌍을 생성합니다.

```
ssh-keygen
```

   b. 각 파일 노드에 암호 없는 SSH를 설정합니다.

```
ssh-copy-id <ip_or_hostname>
```

! 블록 노드에 암호 없는 SSH를 * 설정하지 마십시오. 이 작업은 지원되거나 필요하지 않습니다.

Ansible 인벤토리를 작성합니다

파일 및 블록 노드의 구성을 정의하려면 구축할 BeeGFS 파일 시스템을 나타내는 Ansible 인벤토리를 생성합니다. 인벤토리는 원하는 BeeGFS 파일 시스템을 설명하는 호스트, 그룹 및 변수를 포함합니다.

1단계**:** 모든 빌딩 블록에 대한 설정을 정의합니다

개별적으로 적용할 수 있는 구성 프로파일에 관계없이 모든 구성 요소에 적용되는 구성을 정의합니다.

시작하기 전에

- 배포에 사용할 서브넷 주소 지정 체계를 선택합니다. 에 나와 있는 이점 때문에 "소프트웨어 아키텍처"단일 서브넷 주소 지정 체계를 사용하는 것이 좋습니다.

단계

1. Ansible 제어 노드에서 Ansible 인벤토리 및 플레이북 파일을 저장하는 데 사용할 디렉토리를 식별하십시오.

   별도로 언급하지 않는 한, 이 단계에서 만든 모든 파일과 디렉터리와 다음 단계는 이 디렉터리를 기준으로 생성됩니다.

2. 다음 하위 디렉터리를 만듭니다.

   HOST_VAR'입니다

   group_vars입니다

   '패키지'

3. 클러스터 암호에 대한 하위 디렉토리를 생성하고 Ansible Vault로 암호화하여 파일을 보호합니다(참조 "Ansible Vault로 콘텐츠 암호화").

   a. 하위 디렉터리를 `group_vars/all`만듭니다.

   b. `group_vars/all`디렉토리에서 레이블이 지정된 암호 파일을 `passwords.yml` 만듭니다.

   c. 구성에 따라 모든 사용자 이름 및 암호 매개 변수를 대체하여 를 다음과 같이 입력합니다 `passwords.yml file`.

```
# Credentials for storage system's admin password
eseries_password: <PASSWORD>

# Credentials for BeeGFS file nodes
ssh_ha_user: <USERNAME>
ssh_ha_become_pass: <PASSWORD>

# Credentials for HA cluster
ha_cluster_username: <USERNAME>
ha_cluster_password: <PASSWORD>
ha_cluster_password_sha512_salt: randomSalt

# Credentials for fencing agents
# OPTION 1: If using APC Power Distribution Units (PDUs) for fencing:
# Credentials for APC PDUs.
apc_username: <USERNAME>
apc_password: <PASSWORD>

# OPTION 2: If using the Redfish APIs provided by the Lenovo XCC (and
other BMCs) for fencing:
# Credentials for XCC/BMC of BeeGFS file nodes
bmc_username: <USERNAME>
bmc_password: <PASSWORD>
```

d. 실행 `ansible-vault encrypt passwords.yml` 후 메시지가 표시되면 볼트 암호를 설정합니다.

2단계**:** 개별 파일 및 블록 노드에 대한 설정을 정의합니다

개별 파일 노드 및 개별 구성 요소 노드에 적용되는 구성을 정의합니다.

1. 'host_vars/'에서 다음 내용으로 이름이 '<HOSTNAME>.yml'인 각 BeeGFS 파일 노드에 대한 파일을 만듭니다. BeeGFS 클러스터 IP 및 호스트 이름에 대해 채울 콘텐츠에 대한 메모는 홀수와 짝수로 끝나는 것이 좋습니다.

   처음에는 파일 노드 인터페이스 이름이 여기에 나열된 것과 일치합니다(예: ib0 또는 ibs1f0). 이러한 사용자 정의 이름은 에 구성되어 있습니다 4단계: 모든 파일 노드에 적용할 구성을 정의합니다.

```
ansible_host: "<MANAGEMENT_IP>"
eseries_ipoib_interfaces:  # Used to configure BeeGFS cluster IP
addresses.
  - name: i1b
    address: 100.127.100. <NUMBER_FROM_HOSTNAME>/16
  - name: i4b
    address: 100.127.100. <NUMBER_FROM_HOSTNAME>/16
beegfs_ha_cluster_node_ips:
  - <MANAGEMENT_IP>
  - <i1b_BEEGFS_CLUSTER_IP>
  - <i4b_BEEGFS_CLUSTER_IP>
# NVMe over InfiniBand storage communication protocol information
# For odd numbered file nodes (i.e., h01, h03, ..):
eseries_nvme_ib_interfaces:
  - name: i1a
    address: 192.168.1.10/24
    configure: true
  - name: i2a
    address: 192.168.3.10/24
    configure: true
  - name: i3a
    address: 192.168.5.10/24
    configure: true
  - name: i4a
    address: 192.168.7.10/24
    configure: true
# For even numbered file nodes (i.e., h02, h04, ..):
# NVMe over InfiniBand storage communication protocol information
eseries_nvme_ib_interfaces:
  - name: i1a
    address: 192.168.2.10/24
    configure: true
  - name: i2a
    address: 192.168.4.10/24
    configure: true
  - name: i3a
    address: 192.168.6.10/24
    configure: true
  - name: i4a
    address: 192.168.8.10/24
    configure: true
```

BeeGFS 클러스터를 이미 구축한 경우, NVMe/IB에 사용되는 클러스터 IP 및 IP를 포함하여 정적으로 구성된 IP 주소를 추가하거나 변경하기 전에 클러스터를 중지해야 합니다. 이러한 변경 사항이 적절히 적용되고 클러스터 작업을 방해하지 않도록 이 작업이 필요합니다.

2. 'host_vars/'에서 '<HOSTNAME>.yml'이라는 이름의 각 BeeGFS 블록 노드에 대한 파일을 생성하고 다음 내용으로 채웁니다.

   홀수와 짝수로 끝나는 스토리지 배열 이름에 대한 내용을 입력할 때 특히 주의해야 합니다.

   각 블록 노드에 대해 하나의 파일을 생성하고 두 컨트롤러 중 하나의 "<management_ip>"를 지정합니다 (일반적으로 A).

```
eseries_system_name: <STORAGE_ARRAY_NAME>
eseries_system_api_url: https://<MANAGEMENT_IP>:8443/devmgr/v2/
eseries_initiator_protocol: nvme_ib
# For odd numbered block nodes (i.e., a01, a03, ..):
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.1.101
    - 192.168.2.101
    - 192.168.1.100
    - 192.168.2.100
  controller_b:
    - 192.168.3.101
    - 192.168.4.101
    - 192.168.3.100
    - 192.168.4.100
# For even numbered block nodes (i.e., a02, a04, ..):
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.5.101
    - 192.168.6.101
    - 192.168.5.100
    - 192.168.6.100
  controller_b:
    - 192.168.7.101
    - 192.168.8.101
    - 192.168.7.100
    - 192.168.8.100
```

3단계**:** 모든 파일 및 블록 노드에 적용되어야 하는 설정을 정의합니다

그룹에 해당하는 파일 이름으로 group_vars 아래에 있는 호스트 그룹에 공통된 구성을 정의할 수 있습니다. 이렇게 하면 여러 위치에서 공유 구성이 반복되지 않습니다.

이 작업에 대해

호스트는 둘 이상의 그룹에 있을 수 있으며 런타임 시 Ansible은 변수 우선 순위 규칙에 따라 특정 호스트에 적용되는 변수를 선택합니다. (이 규칙에 대한 자세한 내용은 용 Ansible 설명서를 참조하십시오 "변수 사용"참조)

호스트 대 그룹 지정은 이 절차의 마지막을 위해 생성되는 실제 Ansible 인벤토리 파일에 정의됩니다.

단계

Ansible에서는 모든 호스트에 적용할 구성을 '모두'라는 그룹으로 정의할 수 있습니다. 다음 내용으로 group_vars/all.yml 파일을 만듭니다.

```
ansible_python_interpreter: /usr/bin/python3
beegfs_ha_ntp_server_pools:  # Modify the NTP server addressess if
desired.
  - "pool 0.pool.ntp.org iburst maxsources 3"
  - "pool 1.pool.ntp.org iburst maxsources 3"
```

4단계**:** 모든 파일 노드에 적용할 구성을 정의합니다

파일 노드의 공유 구성은 ha_cluster라는 그룹에 정의됩니다. 이 섹션의 단계에서는 group_vars/ha_cluster.yml 파일에 포함되어야 하는 구성을 작성합니다.

단계

1. 파일 맨 위에서 파일 노드의 'SUDO' 사용자로 사용할 암호를 포함하여 기본값을 정의합니다.

```
### ha_cluster Ansible group inventory file.
# Place all default/common variables for BeeGFS HA cluster resources
below.
### Cluster node defaults
ansible_ssh_user: {{ ssh_ha_user }}
ansible_become_password: {{ ssh_ha_become_pass }}
eseries_ipoib_default_hook_templates:
  - 99-multihoming.j2   # This is required for single subnet
deployments, where static IPs containing multiple IB ports are in the
same IPoIB subnet. i.e: cluster IPs, multirail, single subnet, etc.
# If the following options are specified, then Ansible will
automatically reboot nodes when necessary for changes to take effect:
eseries_common_allow_host_reboot: true
eseries_common_reboot_test_command: "! systemctl status
eseries_nvme_ib.service || systemctl --state=exited | grep
eseries_nvme_ib.service"
eseries_ib_opensm_options:
  virt_enabled: "2"
  virt_max_ports_in_process: "0"
```

> 가 이미 있는 `root` 경우 `ansible_ssh_user` 필요에 따라 를 생략하고 플레이북을 실행할 때 옵션을 지정할 `--ask-become-pass` 수 있습니다 `ansible_become_password`.

2. 필요에 따라 고가용성(HA) 클러스터의 이름을 구성하고 클러스터 내 통신을 위한 사용자를 지정합니다.

   전용 IP 주소 지정 체계를 수정하는 경우 기본 "begfs_ha_mgmtd_floating_ip"도 업데이트해야 합니다. 나중에 BeeGFS 관리 리소스 그룹에 대해 구성한 것과 일치해야 합니다.

"begfs_ha_alert_email_list"를 사용하여 클러스터 이벤트에 대한 경고를 수신할 e-메일을 하나 이상 지정합니다.

```
### Cluster information
beegfs_ha_firewall_configure: True
eseries_beegfs_ha_disable_selinux: True
eseries_selinux_state: disabled
# The following variables should be adjusted depending on the desired
configuration:
beegfs_ha_cluster_name: hacluster                    # BeeGFS HA cluster
name.
beegfs_ha_cluster_username: "{{ ha_cluster_username }}" # Parameter for
BeeGFS HA cluster username in the passwords file.
beegfs_ha_cluster_password: "{{ ha_cluster_password }}" # Parameter for
BeeGFS HA cluster username's password in the passwords file.
beegfs_ha_cluster_password_sha512_salt: "{{
ha_cluster_password_sha512_salt }}" # Parameter for BeeGFS HA cluster
username's password salt in the passwords file.
beegfs_ha_mgmtd_floating_ip: 100.127.101.0          # BeeGFS management
service IP address.
# Email Alerts Configuration
beegfs_ha_enable_alerts: True
beegfs_ha_alert_email_list: ["email@example.com"]  # E-mail recipient
list for notifications when BeeGFS HA resources change or fail.  Often a
distribution list for the team responsible for managing the cluster.
beegfs_ha_alert_conf_ha_group_options:
      mydomain: “example.com”
# The mydomain parameter specifies the local internet domain name. This
is optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com).
# Adjusting the following parameters is optional:
beegfs_ha_alert_timestamp_format: "%Y-%m-%d %H:%M:%S.%N" #%H:%M:%S.%N
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
```

중복된 것처럼 보이지만 BeeGFS 파일 시스템을 단일 HA 클러스터 이상으로 확장하는 경우 "begfs_ha_mgmtd_floating_ip"가 중요합니다. 이후 HA 클러스터는 추가 BeeGFS 관리 서비스 없이 구축되고 첫 번째 클러스터에서 제공하는 관리 서비스를 가리키도록 구축됩니다.

3. 펜싱 에이전트를 구성합니다. (자세한 내용은 을 참조하십시오 "Red Hat High Availability 클러스터에서 펜싱을 구성합니다".) 다음 출력에서는 일반적인 펜싱 에이전트를 구성하는 예를 보여 줍니다. 다음 옵션 중 하나를 선택합니다.

   이 단계에서는 다음 사항에 유의하십시오.

- 기본적으로 펜싱은 활성화되어 있지만 fencing_agent_를 구성해야 합니다.
- pcmk_host_map 또는 pcmk_host_list에 지정된 '<HOSTNAME>'은(는) Ansible 인벤토리의 호스트 이름과 일치해야 합니다.
- 특히 운영 환경에서는 펜싱 없이 BeeGFS 클러스터를 실행할 수 없습니다. 이는 주로 블록 디바이스와 같은 리소스 종속성이 포함된 BeeGFS 서비스가 문제로 인해 페일오버될 때 파일 시스템 손상 또는 기타 바람직하지 않거나 예기치 않은 동작으로 이어질 수 있는 여러 노드에 의한 동시 액세스 위험이 발생하지 않도록 하기 위한 것입니다. 펜싱을 비활성화해야 하는 경우 BeeGFS HA 역할의 시작 가이드의 일반 참고를 참조하여 ha_cluster_crm_config_options ["STONITH -enabled"]"를 false 로 설정합니다.
- 사용 가능한 노드 레벨 펜싱 장치가 여러 개 있으며 BeeGFS HA 역할은 Red Hat HA 패키지 리포지토리에서 사용 가능한 펜싱 에이전트를 구성할 수 있습니다. 가능한 경우 무정전 전원 공급 장치(UPS) 또는 랙 배전 장치(rPDU)를 통해 작동하는 펜싱 에이전트를 사용합니다. BMC(베이스보드 관리 컨트롤러) 또는 서버에 내장된 기타 표시등 출력 장치와 같은 일부 펜싱 에이전트가 특정 장애 시나리오에서 Fence 요청에 응답하지 않을 수 있기 때문입니다.

```
### Fencing configuration:
# OPTION 1: To enable fencing using APC Power Distribution Units
(PDUs):
beegfs_ha_fencing_agents:
 fence_apc:
   - ipaddr: <PDU_IP_ADDRESS>
     login: "{{ apc_username }}" # Parameter for APC PDU username in
the passwords file.
     passwd: "{{ apc_password }}" # Parameter for APC PDU password in
the passwords file.
     pcmk_host_map:
"<HOSTNAME>:<PDU_PORT>,<PDU_PORT>;<HOSTNAME>:<PDU_PORT>,<PDU_PORT>"
# OPTION 2: To enable fencing using the Redfish APIs provided by the
Lenovo XCC (and other BMCs):
redfish: &redfish
  username: "{{ bmc_username }}" # Parameter for XCC/BMC username in
the passwords file.
  password: "{{ bmc_password }}" # Parameter for XCC/BMC password in
the passwords file.
    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
# For details on configuring other fencing agents see
https://access.redhat.com/documentation/en-
us/red_hat_enterprise_linux/9/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

4. Linux OS에서 권장되는 성능 조정을 활성화합니다.

   일반적으로 성능 매개 변수에 대한 기본 설정은 대부분의 사용자가 찾지만 선택적으로 특정 작업 부하에 대한 기본 설정을 변경할 수 있습니다. 따라서 이러한 권장 사항은 BeeGFS 역할에 포함되지만 기본적으로 설정되어 있지 않으므로 사용자가 파일 시스템에 적용된 튜닝에 대해 알 수 있습니다.

   성능 조정을 활성화하려면 다음을 지정하십시오.

```
### Performance Configuration:
beegfs_ha_enable_performance_tuning: True
```

5. (선택 사항) 필요에 따라 Linux OS에서 성능 조정 매개 변수를 조정할 수 있습니다.

   조정할 수 있는 사용 가능한 튜닝 매개 변수의 전체 목록은 에서 BeeGFS HA 역할의 성능 튜닝 기본값 섹션을 참조하십시오 "E-Series BeeGFS GitHub 사이트". 이 파일의 클러스터에 있는 모든 노드 또는 개별 노드의 파일에 대해 기본값을 재정의할 수 `host_vars` 있습니다.

6. 블록 노드와 파일 노드 간에 전체 200GB/HDR 연결을 허용하려면 NVIDIA Open Fabrics Enterprise Distribution(MLNX_OFED)의 OpenSM(Open Subnet Manager) 패키지를 사용하십시오. 나열된 MLNX_OFED 버전은 "파일 노드 요구 사항" 권장 OpenSM 패키지와 함께 제공됩니다. Ansible을 사용한 배포가 지원되지만, 먼저 모든 파일 노드에 MLNX_OFED 드라이버를 설치해야 합니다.

   a. group_vars/ha_cluster.yml에 다음 파라미터를 입력합니다(필요에 따라 패키지 조정).

   ```
   ### OpenSM package and configuration information
   eseries_ib_opensm_options:
     virt_enabled: "2"
     virt_max_ports_in_process: "0"
   ```

7. 논리적 InfiniBand 포트 식별자를 기본 PCIe 디바이스에 일관되게 매핑하도록 'udev' 규칙을 구성합니다.

   udev 규칙은 BeeGFS 파일 노드로 사용되는 각 서버 플랫폼의 PCIe 토폴로지에 고유해야 합니다.

   검증된 파일 노드에 대해 다음 값을 사용합니다.

```
### Ensure Consistent Logical IB Port Numbering
# OPTION 1: Lenovo SR665 V3 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:01:00.0": i1a
  "0000:01:00.1": i1b
  "0000:41:00.0": i2a
  "0000:41:00.1": i2b
  "0000:81:00.0": i3a
  "0000:81:00.1": i3b
  "0000:a1:00.0": i4a
  "0000:a1:00.1": i4b

# OPTION 2: Lenovo SR665 PCIe address-to-logical IB port mapping:
eseries_ipoib_udev_rules:
  "0000:41:00.0": i1a
  "0000:41:00.1": i1b
  "0000:01:00.0": i2a
  "0000:01:00.1": i2b
  "0000:a1:00.0": i3a
  "0000:a1:00.1": i3b
  "0000:81:00.0": i4a
  "0000:81:00.1": i4b
```

8. (선택 사항) 메타데이터 대상 선택 알고리즘을 업데이트합니다.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

> 검증 테스트에서는 일반적으로 성능 벤치마킹 중에 테스트 파일이 모든 BeeGFS 스토리지 대상에 고르게 분산되도록 하기 위해 "랜덤 로빈"이 사용되었습니다(벤치마킹을 위한 자세한 내용은 BeeGFS 사이트 참조) "BeeGFS 시스템을 벤치마킹합니다")를 클릭합니다. 실제 환경에서 사용하면 낮은 번호의 대상이 높은 번호의 목표보다 빠르게 채워질 수 있습니다. 기본 '무작위 배정' 값을 사용하기만 하면 사용 가능한 모든 대상을 활용하는 동시에 우수한 성능을 제공하는 것으로 나타났습니다.

5단계**:** 공통 블록 노드에 대한 구성을 정의합니다

블록 노드의 공유 구성은 eseries_storage_systems라는 그룹에 정의되어 있습니다. 이 섹션의 단계에서는 group_vars/eseries_storage_systems.yml 파일에 포함되어야 하는 구성을 작성합니다.

단계

1. Ansible 연결을 로컬로 설정하고 시스템 암호를 제공하며 SSL 인증서를 확인해야 하는지 여부를 지정합니다. (일반적으로 Ansible은 SSH를 사용하여 관리 호스트에 연결하지만, 블록 노드로 사용되는 NetApp E-Series 스토리지 시스템의 경우 모듈은 통신에 REST API를 사용합니다.) 파일 맨 위에 다음을 추가합니다.

```
### eseries_storage_systems Ansible group inventory file.
# Place all default/common variables for NetApp E-Series Storage Systems
here:
ansible_connection: local
eseries_system_password: {{ eseries_password }} # Parameter for E-Series
storage array password in the passwords file.
eseries_validate_certs: false
```

2. 최적의 성능을 보장하기 위해 에 블록 노드에 대해 나열된 버전을 설치합니다 "기술 요구사항".

   에서 해당 파일을 다운로드합니다 "NetApp Support 사이트". 수동으로 업그레이드하거나 Ansible 제어 노드의 'packages/' 디렉토리에 추가한 다음, Ansible을 사용하여 업그레이드하려면 "eseries_storage_systems.yml"에 다음 매개 변수를 입력합니다.

```
# Firmware, NVSRAM, and Drive Firmware (modify the filenames as needed):
eseries_firmware_firmware: "packages/RCB_11.80GA_6000_64cc0ee3.dlp"
eseries_firmware_nvsram: "packages/N6000-880834-D08.dlp"
```

3. 에서 블록 노드에 설치된 드라이브에 사용할 수 있는 최신 드라이브 펌웨어를 "NetApp Support 사이트 "다운로드하여 설치합니다. 수동으로 업그레이드하거나 Ansible 제어 노드의 디렉토리에 포함시킨 다음 Ansible을 사용하여 업그레이드하려면 에 다음 매개 변수를 채울 수 있습니다 `packages/eseries_storage_systems.yml` .

```
eseries_drive_firmware_firmware_list:
  - "packages/<FILENAME>.dlp"
eseries_drive_firmware_upgrade_drives_online: true
```

> eseries_drive_firmware_upgrade_drives_online을 "false"로 설정하면 업그레이드 속도가 빨라지지만 BeeGFS가 구축되기 전에는 수행할 수 없습니다. 이 설정은 응용 프로그램 오류를 방지하기 위해 업그레이드 전에 드라이브에 대한 모든 I/O를 중지하도록 하기 때문입니다. 볼륨을 구성하기 전에 온라인 드라이브 펌웨어 업그레이드를 수행하는 것이 여전히 빠르지만 나중에 문제가 발생하지 않도록 항상 이 값을 "참"으로 설정하는 것이 좋습니다.

4. 성능을 최적화하려면 글로벌 구성을 다음과 같이 변경합니다.

```
# Global Configuration Defaults
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.
```

5. 최적의 볼륨 프로비저닝 및 동작을 위해 다음 매개 변수를 지정합니다.

```
# Storage Provisioning Defaults
eseries_volume_size_unit: pct
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
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"
```

'eseries_storage_pool_usable_drives'에 지정된 값은 NetApp EF600 블록 노드에만 해당되며 드라이브가 새 볼륨 그룹에 할당되는 순서를 제어합니다. 이 주문을 통해 각 그룹에 대한 입출력이 백엔드 드라이브 채널에 균등하게 분산됩니다.

BeeGFS 구성 요소에 대한 Ansible 인벤토리를 정의합니다

일반 Ansible 인벤토리 구조를 정의한 후 BeeGFS 파일 시스템의 각 구성 요소에 대한 구성을 정의합니다.

이러한 구축 지침은 관리, 메타데이터 및 스토리지 서비스를 포함한 기본 구성 요소로 구성된 파일 시스템, 메타데이터 및 스토리지 서비스를 포함하는 두 번째 구성 요소, 세 번째 스토리지 전용 구성 요소로 이루어진 파일 시스템을 구축하는 방법을 보여 줍니다.

이 단계에서는 전체 BeeGFS 파일 시스템의 요구 사항을 충족하도록 NetApp BeeGFS 구성 요소를 구성하는 데 사용할 수 있는 모든 일반 구성 프로필을 보여 주기 위한 것입니다.

이 섹션과 후속 섹션에서 필요에 따라 조정하여 구축할 BeeGFS 파일 시스템을 나타내는 인벤토리를 생성합니다. 특히, 스토리지 네트워크에 대해 각 블록 또는 파일 노드를 나타내는 Ansible 호스트 이름과 원하는 IP 주소 지정 스키마를 사용하여 BeeGFS 파일 노드 및 클라이언트의 수에 맞게 확장할 수 있도록 합니다.

1단계: Ansible 인벤토리 파일을 만듭니다

단계

1. 새 'inventory.yml' 파일을 만든 다음, 필요에 따라 'eseries_storage_systems' 아래에 있는 호스트를 대체하여 구축의 블록 노드를 나타냅니다. 이름은 host_VAR/<filename>.yml에 사용되는 이름과 일치해야 합니다.

```
# BeeGFS HA (High Availability) cluster inventory.
all:
  children:
    # Ansible group representing all block nodes:
    eseries_storage_systems:
      hosts:
        netapp_01:
        netapp_02:
        netapp_03:
        netapp_04:
        netapp_05:
        netapp_06:
    # Ansible group representing all file nodes:
    ha_cluster:
      children:
```

다음 섹션에서는 클러스터에서 실행할 BeeGFS 서비스를 나타내는 "ha_cluster" 아래에 Ansible 그룹을 추가로 생성합니다.

2단계**:** 관리**,** 메타데이터 및 스토리지 구성 요소에 대한 인벤토리를 구성합니다

클러스터 또는 기본 구성 요소에서 첫 번째 구성 요소는 메타데이터 및 스토리지 서비스와 함께 BeeGFS 관리 서비스를 포함해야 합니다.

단계

1. inventory.yml에서 ha_cluster:Children 아래에 다음 매개 변수를 입력합니다.

```
      # beegfs_01/beegfs_02 HA Pair (mgmt/meta/storage building block):
        mgmt:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_01:
          hosts:
            beegfs_01:
            beegfs_02:
        stor_01:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_02:
          hosts:
            beegfs_01:
            beegfs_02:
        stor_02:
```

```
    hosts:
      beegfs_01:
      beegfs_02:
  meta_03:
    hosts:
      beegfs_01:
      beegfs_02:
  stor_03:
    hosts:
      beegfs_01:
      beegfs_02:
  meta_04:
    hosts:
      beegfs_01:
      beegfs_02:
  stor_04:
    hosts:
      beegfs_01:
      beegfs_02:
  meta_05:
    hosts:
      beegfs_02:
      beegfs_01:
  stor_05:
    hosts:
      beegfs_02:
      beegfs_01:
  meta_06:
    hosts:
      beegfs_02:
      beegfs_01:
  stor_06:
    hosts:
      beegfs_02:
      beegfs_01:
  meta_07:
    hosts:
      beegfs_02:
      beegfs_01:
  stor_07:
    hosts:
      beegfs_02:
      beegfs_01:
  meta_08:
    hosts:
      beegfs_02:
```

```
            beegfs_01:
        stor_08:
          hosts:
            beegfs_02:
            beegfs_01:
```

2. 'group_vars/mgmt.yml' 파일을 생성하고 다음을 포함합니다.

```
# mgmt - BeeGFS HA Management Resource Group
# OPTIONAL: Override default BeeGFS management configuration:
# beegfs_ha_beegfs_mgmtd_conf_resource_group_options:
#  <beegfs-mgmt.conf:key>:<beegfs-mgmt.conf:value>
floating_ips:
  - i1b: 100.127.101.0/16
  - i2b: 100.127.102.0/16
beegfs_service: management
beegfs_targets:
  netapp_01:
    eseries_storage_pool_configuration:
      - name: beegfs_m1_m2_m5_m6
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 1
            owning_controller: A
```

3. group_vars/ 아래에서 다음 템플릿을 사용하여 META_08을 통해 자원 그룹 META_01에 대한 파일을 만든 다음 아래 표를 참조하여 각 서비스에 대한 자리 표시자 값을 입력합니다.

```
# meta_0X - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <PORT>
  connMetaPortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET> # Example: i1b:192.168.120.1/16
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: metadata
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb:  128
        volumes:
          - size: 21.25 # SEE NOTE BELOW!
            owning_controller: <OWNING CONTROLLER>
```

볼륨 크기는 전체 스토리지 풀(볼륨 그룹이라고도 함)의 백분율로 지정됩니다. SSD 오버 프로비저닝을 위해 각 풀에 여유 용량을 남겨 두는 것이 좋습니다(자세한 내용은 참조) "NetApp EF600 어레이 소개")를 클릭합니다. 스토리지 풀 'begfs_m1_m2_m5_m6'도 관리 서비스를 위해 풀 용량의 1%를 할당합니다. 따라서 스토리지 풀의 메타데이터 볼륨에 대해 1.92TB 또는 3.84TB 드라이브를 사용할 때 Beegfs_M1_m2_M5_M6의 경우 이 값을 21.25로 설정하고, 7.65TB 드라이브의 경우 이 값을 22.25로 설정하고, 15.3TB 드라이브의 경우 이 값을 23.75로 설정합니다.

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
meta_01.yml	8015	i1b:100.127.101.1 / 16 i2b:100.127.102.1/16	0	netapp_01를 참조하십시오	Beegfs_m1_m2_m5_m6	A
meta_02.yml	8025	i2b:100.127.102.2 / 16 i1b:100.127.101.2/16	0	netapp_01를 참조하십시오	Beegfs_m1_m2_m5_m6	B
meta_03.yml	8035	i3b:100.127.101.3 / 16 i4b:100.127.102.3/16	1	netapp_02를 참조하십시오	Beegfs_m3_M4_M7_M8	A
meta_04.yml	8045	i4b:100.127.102.4 / 16 i3b:100.127.101.4/16	1	netapp_02를 참조하십시오	Beegfs_m3_M4_M7_M8	B

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
meta_05.yml	8055	i1b:100.127.101.5 / 16 i2b:100.127.102.5/16	0	netapp_01를 참조하십시오	Beegfs_m1_m2_m5_m6	A
meta_06.yml	8065	i2b:100.127.102.6 / 16 i1b:100.127.101.6/16	0	netapp_01를 참조하십시오	Beegfs_m1_m2_m5_m6	B
meta_07.yml	8075	i3b:100.127.101.7 / 16 i4b:100.127.102.7/16	1	netapp_02를 참조하십시오	Beegfs_m3_M4_M7_M8	A
META_08.월	8085	i4b:100.127.102.8 / 16 i3b:100.127.101.8/16	1	netapp_02를 참조하십시오	Beegfs_m3_M4_M7_M8	B

4. group_vars/ 아래에서 다음 템플릿을 사용하여 'tor_08'을 통해 리소스 그룹 tor_01에 대한 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# stor_0X - BeeGFS HA Storage Resource
Groupbeegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 10
        common_volume_configuration:
          segment_size_kb: 512        volumes:
          - size: 21.50 # See note below!             owning_controller:
<OWNING CONTROLLER>
          - size: 21.50            owning_controller: <OWNING
CONTROLLER>
```

올바른 크기는 을 참조하십시오 "권장되는 스토리지 풀 오버 프로비저닝 비율".

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_01.대칭	8013	i1b:100.127.103.1 / 16 i2b:100.127.104.1/16	0	netapp_01를 참조하십시오	Beegfs_s1_s2	A
STOR_02.월	8023	i2b:100.127.104.2 / 16 i1b:100.127.103.2/16	0	netapp_01를 참조하십시오	Beegfs_s1_s2	B
STOR_03.월	8033	i3b:100.127.103.3 / 16 i4b:100.127.104.3/16	1	netapp_02를 참조하십시오	Beegfs_S3_S4	A
STOR_04.yml	8043	i4b:100.127.104.4 / 16 i3b:100.127.103.4/16	1	netapp_02를 참조하십시오	Beegfs_S3_S4	B
STOR_05.월	8053	i1b:100.127.103.5 / 16 i2b:100.127.104.5/16	0	netapp_01를 참조하십시오	Beegfs_S5_S6	A
STOR_06.대칭	8063	i2b:100.127.104.6 / 16 i1b:100.127.103.6/16	0	netapp_01를 참조하십시오	Beegfs_S5_S6	B
STOR_07.월	8073	i3b:100.127.103.7 / 16 i4b:100.127.104.7/16	1	netapp_02를 참조하십시오	Beegfs_S7_s8	A
STOR_08.월	8083	i4b:100.127.104.8 / 16 i3b:100.127.103.8/16	1	netapp_02를 참조하십시오	Beegfs_S7_s8	B

3단계**:** 메타데이터 **+** 스토리지 구성 요소에 대한 인벤토리를 구성합니다

다음 단계에서는 BeeGFS 메타데이터 + 스토리지 구성 요소에 대한 Ansible 인벤토리를 설정하는 방법을 설명합니다.

단계

1. 'inventory.yml'에서 기존 설정 아래에 다음 파라미터를 입력합니다.

```
        meta_09:
          hosts:
            beegfs_03:
            beegfs_04:
```

```
        stor_09:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_10:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_10:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_11:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_11:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_12:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_12:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_13:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_13:
          hosts:
            beegfs_04:
            beegfs_03:
        meta_14:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_14:
          hosts:
            beegfs_04:
            beegfs_03:
        meta_15:
          hosts:
```

```
            beegfs_04:
            beegfs_03:
        stor_15:
          hosts:
            beegfs_04:
            beegfs_03:
        meta_16:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_16:
          hosts:
            beegfs_04:
            beegfs_03:
```

2. group_vars/ 아래에서 다음 템플릿을 사용하여 META_16을 통해 자원 그룹 META_09 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# meta_0X - BeeGFS HA Metadata Resource Group
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <PORT>
  connMetaPortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: metadata
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid1
        criteria_drive_count: 4
        common_volume_configuration:
          segment_size_kb: 128
        volumes:
          - size: 21.5 # SEE NOTE BELOW!
            owning_controller: <OWNING CONTROLLER>
```

올바른 크기는 을 참조하십시오 "권장되는 스토리지 풀 오버 프로비저닝 비율".

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
META_09.대칭	8015	i1b:100.127.101.9 / 16 i2b:100.127.102.9/16	0	netapp_03를 참조하십시오	Beegfs_m9_M10_M13_M14	A
META_10.월	8025	i2b:100.127.102.10 / 16 i1b:100.127.101.10/16	0	netapp_03를 참조하십시오	Beegfs_m9_M10_M13_M14	B
meta_11.yml	8035	i3b:100.127.101.11 / 16 i4b:100.127.102.11/16	1	netapp_04를 참조하십시오	Beegfs_M11_M12_M15_M16	A
META_12.월	8045	i4b:100.127.102.12 / 16 i3b:100.127.101.12/16	1	netapp_04를 참조하십시오	Beegfs_M11_M12_M15_M16	B
META_13.월	8055	i1b:100.127.101.13 / 16 i2b:100.127.102.13/16	0	netapp_03를 참조하십시오	Beegfs_m9_M10_M13_M14	A
meta_14.yml	8065	i2b:100.127.102.14 / 16 i1b:100.127.101.14/16	0	netapp_03를 참조하십시오	Beegfs_m9_M10_M13_M14	B
META_15.월	8075	i3b:100.127.101.15 / 16 i4b:100.127.102.15/16	1	netapp_04를 참조하십시오	Beegfs_M11_M12_M15_M16	A
meta_16.yml	8085	i4b:100.127.102.16 / 16 i3b:100.127.101.16/16	1	netapp_04를 참조하십시오	Beegfs_M11_M12_M15_M16	B

3. group_vars/ 아래에서 다음 템플릿을 사용하여 'tor_16'을 통해 리소스 그룹 tor_09에 대한 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# stor_0X - BeeGFS HA Storage Resource Group
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 10
        common_volume_configuration:
          segment_size_kb: 512        volumes:
          - size: 21.50 # See note below!
            owning_controller: <OWNING CONTROLLER>
          - size: 21.50            owning_controller: <OWNING
CONTROLLER>
```

사용할 올바른 크기를 보려면 다음을 참조하세요. "권장되는 스토리지 풀 오버 프로비저닝 비율" ..

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_09.대칭	8013	i1b:100.127.103.9 / 16 i2b:100.127.104.9/16	0	netapp_03를 참조하십시오	Beegfs_S9_S10	A
STOR_10.월	8023	i2b:100.127.104.10 / 16 i1b:100.127.103.10/16	0	netapp_03를 참조하십시오	Beegfs_S9_S10	B
STOR_11.월	8033	i3b:100.127.103.11 / 16 i4b:100.127.104.11/16	1	netapp_04를 참조하십시오	Beegfs_s11_s12	A
STOR_12.월	8043	i4b:100.127.104.12 / 16 i3b:100.127.103.12/16	1	netapp_04를 참조하십시오	Beegfs_s11_s12	B

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_13.월	8053	i1b:100.127.103.13 / 16 i2b:100.127.104.13/16	0	netapp_03를 참조하십시오	Beegfs_S13_s14	A
STOR_14.월	8063	i2b:100.127.104.14 / 16 i1b:100.127.103.14/16	0	netapp_03를 참조하십시오	Beegfs_S13_s14	B
STOR_15.월	8073	i3b:100.127.103.15 / 16 i4b:100.127.104.15/16	1	netapp_04를 참조하십시오	Beegfs_s15_S16	A
STOR_16.월	8083	i4b:100.127.104.16 / 16 i3b:100.127.103.16/16	1	netapp_04를 참조하십시오	Beegfs_s15_S16	B

4단계**:** 스토리지 전용 구성 요소에 대한 인벤토리를 구성합니다

다음 단계에서는 BeeGFS 스토리지 전용 구성 요소에 대한 Ansible 인벤토리를 설정하는 방법을 설명합니다. 메타데이터 + 스토리지에 대한 구성을 설정하는 것과 스토리지 전용 구성 블록에 대한 구성을 설정하는 것의 주된 차이점은 모든 메타데이터 리소스 그룹이 생략되고 각 스토리지 풀에 대해 "criteria_drive_count"를 10에서 12로 변경하는 것입니다.

단계

1. 'inventory.yml'에서 기존 설정 아래에 다음 파라미터를 입력합니다.

```
# beegfs_05/beegfs_06 HA Pair (storage only building block):
  stor_17:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_18:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_19:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_20:
    hosts:
      beegfs_05:
      beegfs_06:
  stor_21:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_22:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_23:
    hosts:
      beegfs_06:
      beegfs_05:
  stor_24:
    hosts:
      beegfs_06:
      beegfs_05:
```

2. group_vars/ 아래에서 다음 템플릿을 사용하여 'tor_24'를 통해 리소스 그룹 tor_17에 대한 파일을 만든 다음 예제를 참조하여 각 서비스의 자리 표시자 값을 입력합니다.

```
# stor_0X - BeeGFS HA Storage Resource Group
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <PORT>
  connStoragePortUDP: <PORT>
  tuneBindToNumaZone: <NUMA ZONE>
floating_ips:
  - <PREFERRED PORT:IP/SUBNET>
  - <SECONDARY PORT:IP/SUBNET>
beegfs_service: storage
beegfs_targets:
  <BLOCK NODE>:
    eseries_storage_pool_configuration:
      - name: <STORAGE POOL>
        raid_level: raid6
        criteria_drive_count: 12
        common_volume_configuration:
          segment_size_kb: 512
        volumes:
          - size: 21.50 # See note below!
            owning_controller: <OWNING CONTROLLER>
          - size: 21.50
            owning_controller: <OWNING CONTROLLER>
```

사용할 올바른 크기를 보려면 다음을 참조하세요. "권장되는 스토리지 풀 오버 프로비저닝 비율" .

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_17.월	8013	i1b:100.127.103.17 / 16 i2b:100.127.104.17/16	0	netapp_05를 참조하십시오	Beegfs_S17_s18	A
STOR_18.월	8023	i2b:100.127.104.18 / 16 i1b:100.127.103.18/16	0	netapp_05를 참조하십시오	Beegfs_S17_s18	B
STOR_19.대칭	8033	i3b:100.127.103.19 / 16 i4b:100.127.104.19/16	1	netapp_06를 참조하십시오	Beegfs_S19_S20	A
STOR_20.월	8043	i4b:100.127.104.20 / 16 i3b:100.127.103.20/16	1	netapp_06를 참조하십시오	Beegfs_S19_S20	B

파일 이름입니다	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
STOR_21.대칭	8053	i1b:100.127.103.21 / 16 i2b:100.127.104.21/16	0	netapp_05를 참조하십시오	Beegfs_s21_S22	A
STOR_22.월	8063	i2b:100.127.104.22 / 16 i1b:100.127.103.22/16	0	netapp_05를 참조하십시오	Beegfs_s21_S22	B
STOR_23.월	8073	i3b:100.127.103.23 / 16 i4b:100.127.104.23/16	1	netapp_06를 참조하십시오	Beegfs_S23_S24	A
STOR_24.월	8083	i4b:100.127.104.24 / 16 i3b:100.127.103.24/16	1	netapp_06를 참조하십시오	Beegfs_S23_S24	B

BeeGFS 구축

구성 배포 및 관리에는 Ansible이 실행해야 하는 작업이 포함된 하나 이상의 플레이북을 실행해야 전체 시스템을 원하는 상태로 되돌릴 수 있습니다.

모든 작업이 단일 Playbook에 포함될 수 있지만 복잡한 시스템의 경우 이를 관리하기가 매우 복잡해집니다. Ansible을 사용하면 재사용 가능한 플레이북 및 관련 콘텐츠(예: 기본 변수, 작업, 처리기)를 패키지하는 방법으로 역할을 만들고 배포할 수 있습니다. 자세한 내용은 용 Ansible 설명서를 참조하십시오 "역할".

역할은 종종 관련 역할 및 모듈이 포함된 Ansible 컬렉션의 일부로 배포됩니다. 따라서, 이러한 플레이북은 다양한 NetApp E-Series Ansible 컬렉션에서 주로 다양한 역할을 하지만

> 2노드 클러스터를 사용하여 쿼럼을 설정할 때 문제를 완화하기 위해 별도의 쿼럼 장치를 Tiebreaker로 구성하지 않는 한 현재 BeeGFS를 구축하려면 최소 2개의 구성 요소(4개의 파일 노드)가 필요합니다.

단계

1. 새로운 '플레이북.yml' 파일을 생성하고 다음을 포함합니다.

```
# BeeGFS HA (High Availability) cluster playbook.
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

```
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
  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 }}"
  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: Verify the BeeGFS HA cluster is properly deployed.
      ansible.builtin.import_role:
        name: netapp_eseries.beegfs.beegfs_ha_7_4
```

ⓘ 이 플레이북에서는 파일 노드에 Python 3이 설치되어 있는지 확인하는 몇 가지 "pre_tasks"를 실행하고 제공되는 Ansible 태그가 지원되는지 확인합니다.

2. BeeGFS를 배포할 준비가 되면 재고 및 플레이북 파일과 함께 'Ansible-Playbook' 명령을 사용하십시오.

   구축 과정에서 모든 "pre_tasks"를 실행한 다음 실제 BeeGFS 구축을 진행하기 전에 사용자 확인을 묻는 메시지가 표시됩니다.

   다음 명령을 실행하여 필요에 따라 포크 수를 조정합니다(아래 참고 참조).

```
ansible-playbook -i inventory.yml playbook.yml --forks 20
```

   ⓘ 대규모 배포의 경우 `forks` 매개 변수를 사용하여 기본 포크 수(5)를 재정의하면 Ansible이 병렬로 구성하는 호스트 수를 늘리는 것이 좋습니다. (자세한 내용은 을 참조하십시오 "플레이북 실행 제어".) 최대값 설정은 Ansible 제어 노드에서 사용할 수 있는 처리 능력에 따라 다릅니다. 위의 20개 예는 CPU 4개(인텔® 제온® 골드 6146 CPU @ 3.20GHz)가 장착된 가상 Ansible 컨트롤 노드에서 실행되었습니다.

   Ansible 제어 노드와 BeeGFS 파일 및 블록 노드 간의 구축 및 네트워크 성능 크기에 따라 구축 시간이 달라질 수 있습니다.

BeeGFS 클라이언트를 구성합니다

컴퓨팅 또는 GPU 노드와 같이 BeeGFS 파일 시스템에 액세스해야 하는 모든 호스트에 BeeGFS 클라이언트를 설치하고 구성해야 합니다. 이 작업에서는 Ansible 및 BeeGFS 컬렉션을 사용할 수 있습니다.

단계

1. 필요한 경우, Ansible 제어 노드에서 BeeGFS 클라이언트로 구성하려는 각 호스트에 대해 암호 없는 SSH를 설정합니다.

   'ssh-copy-id <user>@<HOSTNAME_OR_IP>'를 참조하십시오

2. 'host_vars/'에서 다음 내용으로 이름이 '<HOSTNAME>.yml'인 각 BeeGFS 클라이언트에 대한 파일을 만들어 사용자 환경에 맞는 올바른 정보로 자리 표시자 텍스트를 채웁니다.

```
# BeeGFS Client
ansible_host: <MANAGEMENT_IP>
# OPTIONAL: If you want to use the NetApp E-Series Host Collection's
IPoIB role to configure InfiniBand interfaces for clients to connect to
BeeGFS file systems:
eseries_ipoib_interfaces:
  - name: <INTERFACE>
    address: <IP>/<SUBNET_MASK> # Example: 100.127.1.1/16
  - name: <INTERFACE>
    address: <IP>/<SUBNET_MASK>
```

두 개의 서브넷 주소 지정 체계를 사용하여 구축하는 경우 각 클라이언트에서 두 개의 스토리지 IPoIB 서브넷에 각각 하나씩 두 개의 InfiniBand 인터페이스를 구성해야 합니다. 여기에 나열된 각 BeeGFS 서비스에 대해 예제 서브넷과 권장 범위를 사용하는 경우, 클라이언트에는 ~ 의 범위에서 인터페이스 하나와 ~ 의 범위로 구성된 인터페이스가 있어야 `100.127.1.0 100.127.99.255 100.128.1.0 `100.128.99.255`합니다.

3. 새 파일 'client_inventory.yml'를 만든 다음 맨 위에 다음 매개 변수를 입력합니다.

```
# BeeGFS client inventory.
all:
  vars:
    ansible_ssh_user: <USER> # This is the user Ansible should use to
connect to each client.
    ansible_become_password: <PASSWORD> # This is the password Ansible
will use for privilege escalation, and requires the ansible_ssh_user be
root, or have sudo privileges.
The defaults set by the BeeGFS HA role are based on the testing
performed as part of this NetApp Verified Architecture and differ from
the typical BeeGFS client defaults.
```

암호를 일반 텍스트로 저장하지 마십시오. 대신 Ansible Vault를 사용하십시오(의 Ansible 설명서 참조) "Ansible Vault로 콘텐츠 암호화") 또는 플레이북이 실행될 때 '--Ask-when-pass' 옵션을 사용합니다.

4. 'client_inventory.yml' 파일에서 Beegfs_clients' 그룹 아래에 BeeGFS 클라이언트로 구성해야 하는 모든 호스트를 나열한 다음 BeeGFS 클라이언트 커널 모듈을 구축하는 데 필요한 추가 구성을 지정합니다.

```
  children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        beegfs_01:
        beegfs_02:
        beegfs_03:
        beegfs_04:
        beegfs_05:
        beegfs_06:
        beegfs_07:
        beegfs_08:
        beegfs_09:
        beegfs_10:
      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.
```

NVIDIA OFED 드라이버를 사용하는 경우 가 `beegfs_client_ofed_include_path` Linux 설치에 대한 올바른 "헤더 포함 경로"를 가리키는지 확인합니다. 자세한 내용은 의 BeeGFS 설명서를 "RDMA 지원"참조하십시오.

5. client_inventory.yml 파일에 미리 정의된 VAR의 하단에 마운트할 BeeGFS 파일 시스템을 나열합니다.

```
        beegfs_client_mounts:
          - sysMgmtdHost: 100.127.101.0 # 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
```

Beegfs_client_config는 테스트된 설정을 나타냅니다. 모든 옵션에 대한 종합적인 개요는 netapp_eseries.beegfs` 컬렉션의 "begfs_client" 역할에 포함된 설명서를 참조하십시오. 여기에는 여러 개의 BeeGFS 파일 시스템을 마운트하거나 동일한 BeeGFS 파일 시스템을 여러 번 마운트하는 방법에 대한 세부 정보가 포함됩니다.

6. 새 'client_Playbook.yml' 파일을 만든 후 다음 매개 변수를 입력합니다.

```
# BeeGFS client playbook.
- hosts: beegfs_clients
  any_errors_fatal: true
  gather_facts: true
  collections:
    - netapp_eseries.beegfs
    - netapp_eseries.host
  tasks:
    - name: Ensure IPoIB is configured
      import_role:
        name: ipoib
    - name: Verify the BeeGFS clients are configured.
      import_role:
        name: beegfs_client
```

필요한 IB/RDMA 드라이버와 IP를 해당 IPoIB 인터페이스에 이미 설치한 경우 'NetApp_eseries.host' 수집 및 'IPoIB' 역할을 가져오지 마십시오.

7. 클라이언트를 설치 및 구축하고 BeeGFS를 마운트하려면 다음 명령을 실행합니다.

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

8. BeeGFS 파일 시스템을 운영 환경에 배치하기 전에 모든 클라이언트에 로그인하고 "begfs-fsck—checkfs"를 실행하여 모든 노드에 연결할 수 있고 보고된 문제가 없는지 확인하는 것이 좋습니다.

5가지 구성 요소 이상으로 확장

5개의 빌딩 블록(10개의 파일 노드) 이상으로 확장하기 위해 페이스 메이커 및 Corosync를 구성할 수 있습니다. 그러나 더 큰 클러스터에는 결점이 있으며, 결국 심장박동기 및 Corosync로 인해 최대 32개의 노드가 필요합니다.

NetApp은 최대 10개 노드까지 BeeGFS HA 클러스터를 테스트했습니다. 따라서 개별 클러스터를 이 제한을 초과하여 확장하는 것은 권장되거나 지원되지 않습니다. 하지만 BeeGFS 파일 시스템은 여전히 10개 노드 이상으로 확장되어야 하며 NetApp은 BeeGFS on NetApp 솔루션에서 이 점을 고려했습니다.

각 파일 시스템에서 구성 요소의 하위 집합이 포함된 여러 HA 클러스터를 구축함으로써 기본 HA 클러스터링 메커니즘에 대한 권장 제한 또는 하드 제한과는 별개로 전체 BeeGFS 파일 시스템을 확장할 수 있습니다. 이 시나리오에서는 다음을 수행합니다.

- 추가 HA 클러스터를 나타내는 새 Ansible 인벤토리를 생성한 다음 다른 관리 서비스 구성 생략합니다. 대신 각 추가 클러스터 ha_cluster.yml의 "begfs_ha_mgmtd_floating_ip" 변수를 첫 번째 BeeGFS 관리 서비스의 IP에 지정합니다.
- 동일한 파일 시스템에 HA 클러스터를 추가할 때는 다음 사항을 확인하십시오.
  - BeeGFS 노드 ID는 고유합니다.

- group_vars의 각 서비스에 해당하는 파일 이름은 모든 클러스터에서 고유합니다.
- BeeGFS 클라이언트 및 서버 IP 주소는 모든 클러스터에서 고유합니다.
- 추가 클러스터를 구축 또는 업데이트하기 전에 BeeGFS 관리 서비스가 포함된 첫 번째 HA 클러스터가 실행되고 있습니다.

- 각 HA 클러스터에 대한 인벤토리를 각각 자체 디렉토리 트리에서 유지 관리합니다.

  하나의 디렉토리 트리에서 여러 클러스터의 인벤토리 파일을 혼합하려고 하면 BeeGFS HA 역할이 특정 클러스터에 적용된 구성을 집계하는 방식에 문제가 발생할 수 있습니다.

> 새 HA 클러스터를 생성하기 전에 각 HA 클러스터를 5개의 구성 요소로 확장할 필요가 없습니다. 대부분의 경우 클러스터당 더 적은 수의 구성 요소를 사용하는 것이 더 쉽게 관리할 수 있습니다. 한 가지 접근 방식은 각 단일 랙의 구성 요소를 HA 클러스터로 구성하는 것입니다.

권장되는 스토리지 풀 오버 프로비저닝 비율

2세대 구성 요소에 대해 스토리지 풀당 표준 볼륨 4개를 따르는 경우 다음 표를 참조하십시오.

이 표에는 각 BeeGFS 메타데이터 또는 스토리지 타겟에 대한 "eseries_storage_pool_configuration"의 볼륨 크기로 사용할 권장 비율이 나와 있습니다.

드라이브 크기	크기
1.92TB	18
3.84TB	21.5
7.68TB	22.5
15.3TB	24

> 위의 지침은 관리 서비스가 포함된 스토리지 풀에 적용되지 않으며, 관리 데이터에 대해 스토리지 풀의 1%를 할당하기 위해 위의 크기를 .25%까지 줄여야 합니다.

이러한 값이 어떻게 결정되었는지 확인하려면 을 참조하십시오 "TR-4800: 부록 A: SSD 내구성 및 오버 프로비저닝 이해".

고용량 구성 요소입니다

표준 BeeGFS 솔루션 구축 가이드에서는 고성능 워크로드 요구 사항에 대한 절차 및 권장 사항을 간략하게 설명합니다. 고용량 요구 사항을 충족하려는 고객은 여기에 설명된 구축 및 권장 사항의 변화를 관찰해야 합니다.

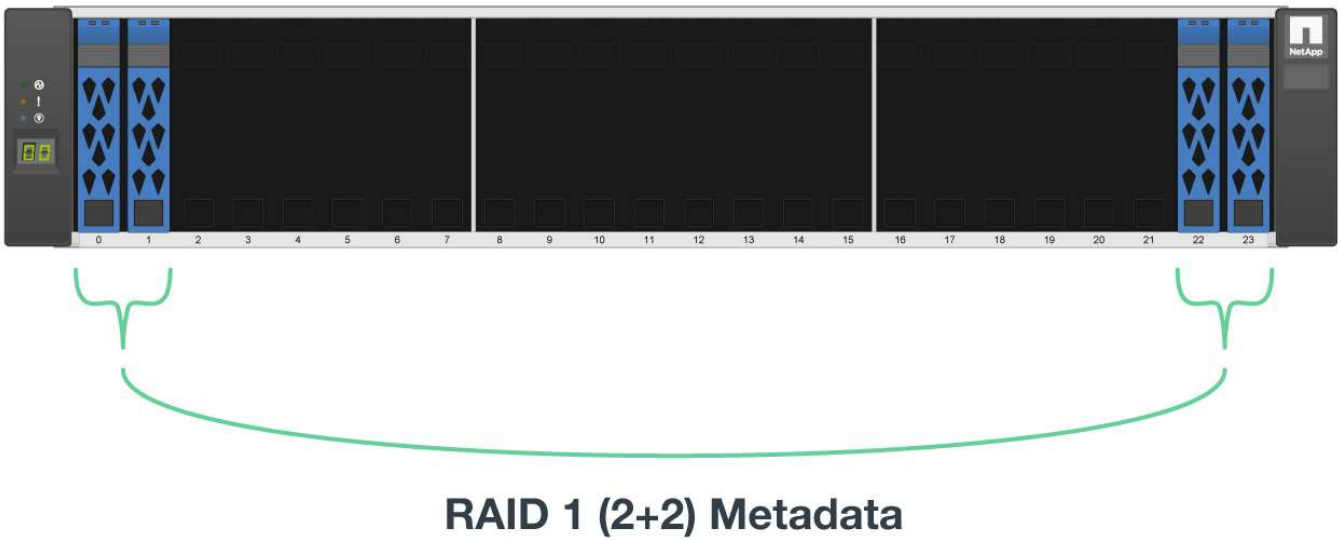
컨트롤러

고용량 구성 요소의 경우 EF600 컨트롤러를 EF300 컨트롤러로 교체해야 하며, 각 컨트롤러는 SAS 확장을 위해 Cascade HIC를 설치합니다. 각 블록 노드는 스토리지 엔클로저에 BeeGFS 메타데이터 스토리지를 위한 최소한의 NVMe SSD를 포함하고 BeeGFS 스토리지 볼륨용 NL-SAS HDD로 채워진 확장 셸프에 연결됩니다.

File Node to Block 노드 구성은 동일하게 유지됩니다.

드라이브 배치

BeeGFS 메타데이터 스토리지를 위해 각 블록 노드에 최소 4개의 NVMe SSD가 필요합니다. 이러한 드라이브는 인클로저의 가장 바깥쪽 슬롯에 위치해야 합니다.



확장 트레이

스토리지 어레이당 1-7, 60 드라이브 확장 트레이를 사용하여 대용량 구성 요소를 사이징할 수 있습니다.

각 확장 트레이 케이블 연결 지침은 "드라이브 쉘프의 EF300 케이블 연결을 참조하십시오".

저작권 정보

Copyright © 2026 NetApp, Inc. All Rights Reserved. 미국에서 인쇄됨 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.