맞춤형 아키텍처를 사용합니다

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

목차

맞춤형 아키텍처를 사용합니다

개요 및 요구 사항

Ansible을 사용하여 BeeGFS 고가용성 클러스터를 구축할 때 모든 NetApp E/EF-Series 스토리지 시스템을 BeeGFS 블록 노드 및 x86 서버로 BeeGFS 파일 노드로 사용할 수 있습니다.

이 섹션에서 사용되는 용어에 대한 정의는 "용어 및 개념"페이지에서 확인할 수 있습니다.

소개

"NetApp 검증 아키텍처"사전 정의된 참조 구성 및 사이징 지침을 제공하는 한편, 일부 고객 및 파트너는 특정 요구 사항이나 하드웨어 선호도에 보다 적합한 맞춤형 아키텍처를 설계하기를 원할 수 있습니다. NetApp에서 BeeGFS를 선택할 때의 주요 이점 중 하나는 Ansible을 사용하여 BeeGFS 공유 디스크 HA 클러스터를 구축하여 NetApp에서 작성한 HA 구성요소를 통해 클러스터 관리를 단순화하고 안정성을 높일 수 있다는 것입니다. NetApp에 맞춤형 BeeGFS 아키텍처를 구축할 때는 Ansible을 사용하여 유연한 하드웨어 제품군에서 어플라이언스 같은 접근 방식을 유지 관리합니다.

이 섹션에서는 NetApp 하드웨어에서 BeeGFS 파일 시스템을 구축하고 Ansible을 사용하여 BeeGFS 파일 시스템을 구성하는 데 필요한 일반적인 단계를 간략하게 설명합니다. BeeGFS 파일 시스템 설계와 관련된 모범 사례 및 최적화된 예제에 대한 자세한 "NetApp 검증 아키텍처"내용은 섹션을 참조하십시오.

구축 개요

일반적으로 BeeGFS 파일 시스템을 구축하려면 다음 단계를 수행해야 합니다.

- 초기 설정:
  - 하드웨어 설치/케이블 연결
  - 파일 및 블록 노드 설정
  - Ansible 제어 노드를 설정합니다.
- BeeGFS 파일 시스템을 Ansible 인벤토리로 정의합니다.
- 파일 및 블록 노드 기준으로 Ansible을 실행하여 BeeGFS 구축
  - 선택적으로 클라이언트를 설정하고 BeeGFS를 마운트합니다.

다음 섹션에서는 이러한 단계에 대해 자세히 설명합니다.

Ansible은 다음을 비롯한 모든 소프트웨어 프로비저닝 및 구성 작업을 처리합니다.

- 블록 노드에서 볼륨 생성/매핑
- 파일 노드에서 볼륨 포맷/튜닝
- 파일 노드에 소프트웨어 설치/구성
- HA 클러스터 설정 및 BeeGFS 리소스 및 파일 시스템 서비스 구성

요구 사항

Ansible에서 BeeGFS에 대한 지원은 에서 제공됩니다 "Ansible 갤럭시" BeeGFS HA 클러스터의 포괄적인 구축 및 관리를 자동화하는 역할 및 모듈 모음입니다.

BeeGFS 자체는 <major>.<minor>.<patch> 버전 관리 체계에 따라 버전이 지정되며, 이 컬렉션은 지원되는 각 <major>.<minor> 버전의 BeeGFS에 대한 역할을 유지합니다. 예를 들어 BeeGFS 7.2 또는 BeeGFS 7.3과 같은 BeeGFS 버전이 있습니다. 컬렉션에 대한 업데이트가 릴리스되면 각 역할의 패치 버전이 해당 릴리즈 지점에 대해 사용 가능한 최신 BeeGFS 버전을 가리키도록 업데이트됩니다(예: 7.2.8). 각 버전의 컬렉션은 특정 Linux 배포판과 버전에서도 테스트되어 지원됩니다. 현재 파일 노드의 경우 Red Hat이고, 클라이언트의 경우 Red Hat과 Ubuntu입니다. 다른 배포판 실행이 지원되지 않으며 다른 버전(특히 다른 주요 버전)을 실행하는 것은 권장되지 않습니다.

Ansible 컨트롤 노드

이 노드에는 BeeGFS 관리에 사용되는 인벤토리 및 Playbook이 포함됩니다. 다음과 같은 요구 사항이 있습니다.

- Ansible 6.x(Ansible-코어 2.13)
- Python 3.6 (이상)
- Python(PIP) 패키지: ipaddr 및 netaddr

또한 제어 노드에서 모든 BeeGFS 파일 노드 및 클라이언트로 암호 없는 SSH를 설정하는 것이 좋습니다.

BeeGFS 파일 노드

파일 노드는 Red Hat Enterprise Linux(RHEL) 9.4를 실행해야 하며, 필수 패키지(pacemaker, corosync, fence-agents-all, resource-agents)가 포함된 HA 저장소에 액세스할 수 있어야 합니다. 예를 들어, 다음 명령을 실행하여 RHEL 9에서 해당 저장소를 활성화할 수 있습니다.

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

BeeGFS 클라이언트 노드

BeeGFS 클라이언트 Ansible 역할을 사용하여 BeeGFS 클라이언트 패키지를 설치하고 BeeGFS 마운트를 관리할 수 있습니다. 이 역할은 RHEL 9.4 및 Ubuntu 22.04에서 테스트되었습니다.

Anabilities를 사용하여 BeeGFS 클라이언트를 설정하고 BeeGFS를 마운트하지 않는 경우 "BeeGFS는 Linux 배포 및 커널을 지원했습니다" 사용할 수 있습니다.

초기 설정

하드웨어 설치 및 케이블 연결

NetApp에서 BeeGFS를 실행하는 데 사용되는 하드웨어를 설치하고 케이블을 연결하는 데 필요한 단계입니다.

설치 계획

각 BeeGFS 파일 시스템은 몇 개의 블록 노드에서 제공하는 백엔드 스토리지를 사용하여 BeeGFS 서비스를 실행하는 몇 개의 파일 노드로 구성됩니다. 파일 노드는 하나 이상의 고가용성 클러스터로 구성되어 BeeGFS 서비스에 대한 내결함성을 제공합니다. 각 블록 노드는 이미 액티브/액티브 HA 쌍입니다. 각 HA 클러스터에서 지원되는 파일 노드의 최소 수는 3개이고, 각 클러스터에서 지원되는 파일 노드의 최대 수는 10개입니다. BeeGFS 파일 시스템은 함께 작동하여 단일 파일 시스템 네임스페이스를 제공하는 여러 독립 HA 클러스터를 구축함으로써 10개 노드 이상으로 확장할 수 있습니다.

일반적으로 각 HA 클러스터는 몇 개의 파일 노드(x86 서버)가 일부 블록 노드(일반적으로 E-Series 스토리지 시스템)에 직접 연결되는 일련의 "구성 요소"로 구축됩니다. 이 구성은 비대칭형 클러스터를 생성하며, BeeGFS 서비스는 BeeGFS 타겟에 사용되는 백엔드 블록 스토리지에 대한 액세스 권한이 있는 특정 파일 노드에서만 실행할 수 있습니다. 각 구성 요소에서 파일-블록 노드와 직접 연결에 사용되는 스토리지 프로토콜의 균형은 특정 설치 요구 사항에 따라 달라집니다.

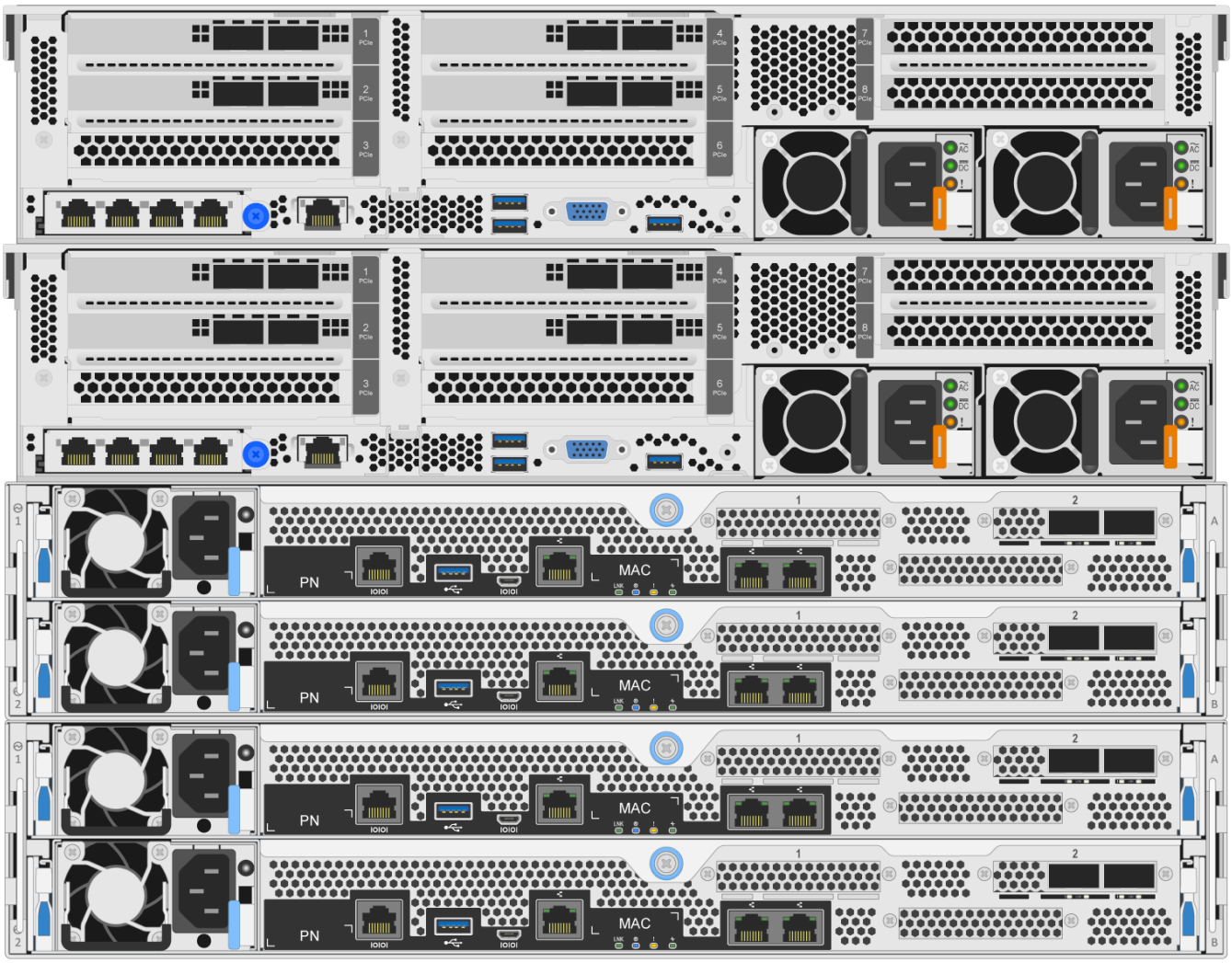
대체 HA 클러스터 아키텍처는 파일 노드와 블록 노드 간에 스토리지 패브릭(SAN이라고도 함)을 사용하여 대칭 클러스터를 설정합니다. 따라서 특정 HA 클러스터의 모든 파일 노드에서 BeeGFS 서비스를 실행할 수 있습니다. 일반적으로 대칭형 클러스터는 추가 SAN 하드웨어로 인해 비용 효율적이지 않기 때문에 이 문서에서는 하나 이상의 빌딩 블록으로 구성된 일련의 비대칭 클러스터를 사용한다고 가정합니다.

> 설치를 계속하기 전에 특정 BeeGFS 구축에 대해 원하는 파일 시스템 아키텍처를 충분히 이해해야 합니다.

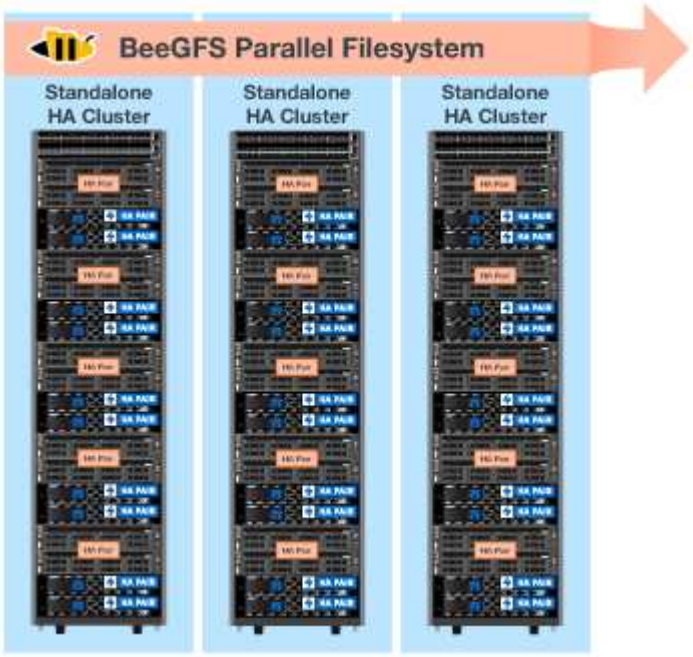
랙 하드웨어

설치를 계획할 때 각 구성 요소에 있는 모든 장비가 인접한 랙 유닛에 랙에 장착되어 있어야 합니다. 각 구성 요소에서 파일 노드를 바로 블록 노드 위에 랙에 마운트하는 것이 모범 사례입니다. 파일 및 의 모델에 대한 설명서를 따릅니다 "블록" 레일 및 하드웨어를 랙에 설치할 때 사용 중인 노드입니다.

단일 구성 요소 예:

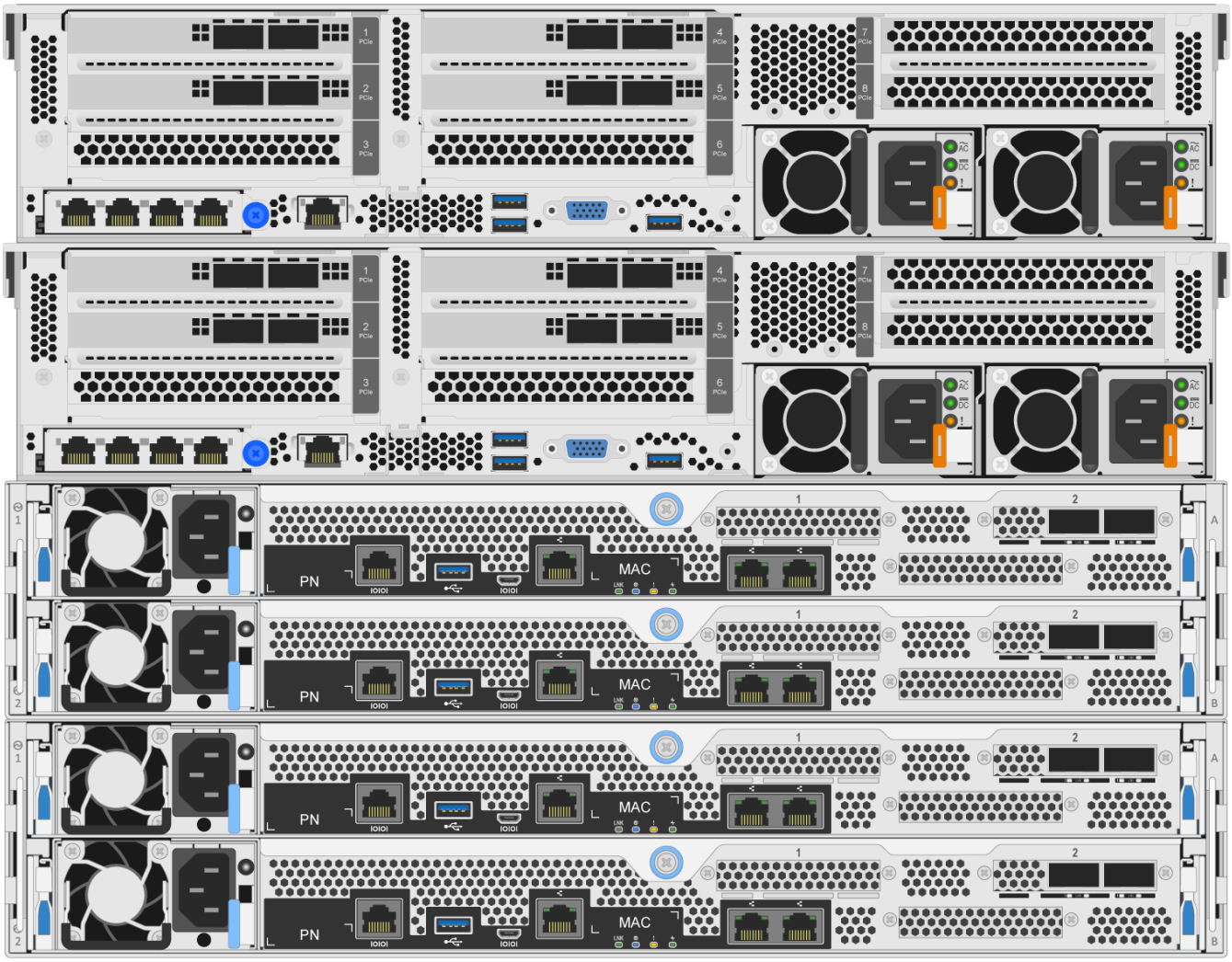


각 HA 클러스터에 여러 개의 구성 요소가 있고 파일 시스템에 여러 HA 클러스터가 있는 대규모 BeeGFS 설치의 예:



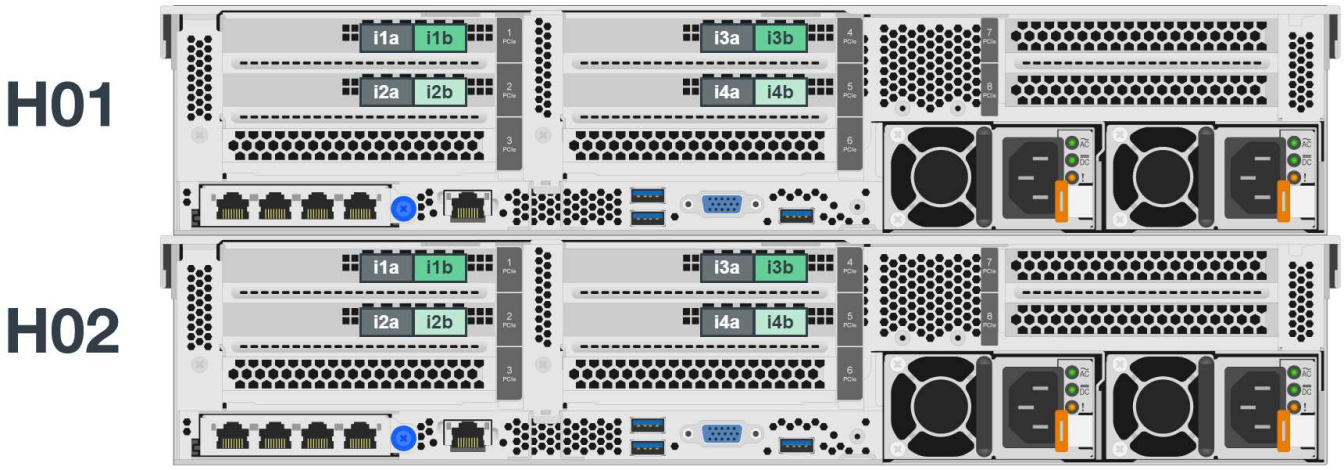
케이블 파일 및 블록 노드

일반적으로 E-Series 블록 노드의 HIC 포트를 파일 노드의 지정된 호스트 채널 어댑터(InfiniBand 프로토콜의 경우) 또는 호스트 버스 어댑터(파이버 채널 및 기타 프로토콜의 경우) 포트에 직접 연결합니다. 이러한 접속을 설정하는 정확한 방법은 원하는 파일 시스템 아키텍처에 따라 다릅니다. 예를 들면 다음과 "NetApp 검증 아키텍처에서 2세대 BeeGFS 기반"같습니다.



클라이언트 네트워크에 파일 노드를 케이블로 연결합니다

각 파일 노드에는 BeeGFS 클라이언트 트래픽용으로 지정된 몇 개의 InfiniBand 또는 이더넷 포트가 있습니다. 아키텍처에 따라 각 파일 노드는 고성능 클라이언트/스토리지 네트워크에 하나 이상의 연결을 갖게 되며, 이중화 및 대역폭 향상을 위해 여러 스위치에 연결할 수 있습니다. 다음은 이중화된 네트워크 스위치를 사용하는 클라이언트 케이블 연결의 예입니다. 이 때 어두운 녹색으로 강조 표시된 포트와 밝은 녹색으로 강조 표시된 포트는 별도의 스위치에 연결됩니다.



연결 관리 네트워킹 및 전원

대역내 및 대역외 네트워크에 필요한 네트워크 연결을 설정합니다.

각 파일 및 블록 노드가 중복성을 위해 여러 전원 분배 장치에 연결되어 있는지 확인하기 위해 모든 전원 공급 장치를 연결합니다(사용 가능한 경우).

파일 및 블록 노드 설정

Ansible을 실행하기 전에 파일 및 블록 노드를 설정하는 데 필요한 수동 단계

파일 노드

베이스보드 관리 컨트롤러(BMC) 구성

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.

서버 공급업체의 설명서에 따라 이 기능에 액세스하는 데 필요한 모든 라이센스를 활성화하고 BMC가 네트워크에 연결되고 원격 액세스에 맞게 구성되었는지 확인합니다.

> Redfish를 사용하여 BMC 기반 펜싱을 사용하려면 Redfish가 활성화되어 있고 파일 노드에 설치된 OS에서 BMC 인터페이스에 액세스할 수 있어야 합니다. BMC와 운영 체제가 동일한 물리적 네트워크 인터페이스를 공유하는 경우 네트워크 스위치에 특별한 구성이 필요할 수 있습니다.

시스템 설정을 조정합니다

시스템 설정(BIOS/UEFI) 인터페이스를 사용하여 성능을 최대화하도록 설정이 설정되어 있는지 확인합니다. 정확한 설정과 최적의 값은 사용 중인 서버 모델에 따라 달라집니다. 에 대한 지침이 "파일 노드 모델을 확인했습니다"제공되며, 그렇지 않은 경우 모델에 따라 서버 공급업체의 설명서 및 모범 사례를 참조하십시오.

운영 체제를 설치합니다

나열된 파일 노드 요구 사항에 따라 지원되는 운영 체제를 "여기"설치합니다. Linux 배포판에 따라 아래의 추가 단계를 참조하십시오.

Red Hat

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=<MAJOR_VERSION>.<MINOR_VERSION>` . 지침은 다음을 참조하세요. "RHEL 시스템을 등록하고 가입하는 방법" 그리고 "업데이트 제한 방법" .

고가용성을 위해 필요한 패키지가 포함된 Red Hat 리포지토리를 활성화합니다.

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

관리 네트워크를 구성합니다

운영 체제의 대역 내 관리를 허용하는 데 필요한 네트워크 인터페이스를 구성합니다. 정확한 단계는 사용 중인 특정 Linux 배포 및 버전에 따라 다릅니다.

SSH가 활성화되어 있고 Ansible 제어 노드에서 모든 관리 인터페이스에 액세스할 수 있는지 확인합니다.

HCA 및 **HBA** 펌웨어를 업데이트합니다

모든 HBA 및 HCA가 에 나열된 지원되는 펌웨어 버전을 "NetApp 상호 운용성 매트릭스"실행하고 있는지 확인하고 필요한 경우 업그레이드합니다. NVIDIA ConnectX 어댑터에 대한 추가 권장 사항을 찾을 "여기"수 있습니다.

블록 노드

의 단계를 따릅니다 "E-Series와 함께 가동 및 운영합니다" 각 블록 노드 컨트롤러에서 관리 포트를 구성하고 선택적으로 각 시스템의 스토리지 어레이 이름을 설정합니다.

Ansible 제어 노드에서 모든 블록 노드에 액세스할 수 있도록 보장하는 추가 구성은 필요하지 않습니다. 나머지 시스템 구성은 Ansible을 사용하여 적용/유지합니다.

Ansible Control Node 설정

파일 시스템을 배포 및 관리하기 위해 Ansible 제어 노드를 설정합니다.

개요

Ansible 제어 노드는 클러스터를 관리하는 데 사용되는 물리적 또는 가상 Linux 시스템입니다. 다음 요구 사항을 충족해야 합니다.

- "요구 사항"Ansible, Python 및 추가 Python 패키지의 설치 버전을 비롯한 BeeGFS HA 역할을 소개합니다.
- 공무원도 만나세요 "Ansible 제어 노드 요구사항" 운영 체제 버전을 포함합니다.
- 모든 파일 및 블록 노드에 대한 SSH 및 HTTPS 액세스 권한 보유

자세한 설치 단계를 찾을 수 "여기"있습니다.

BeeGFS 파일 시스템을 정의합니다

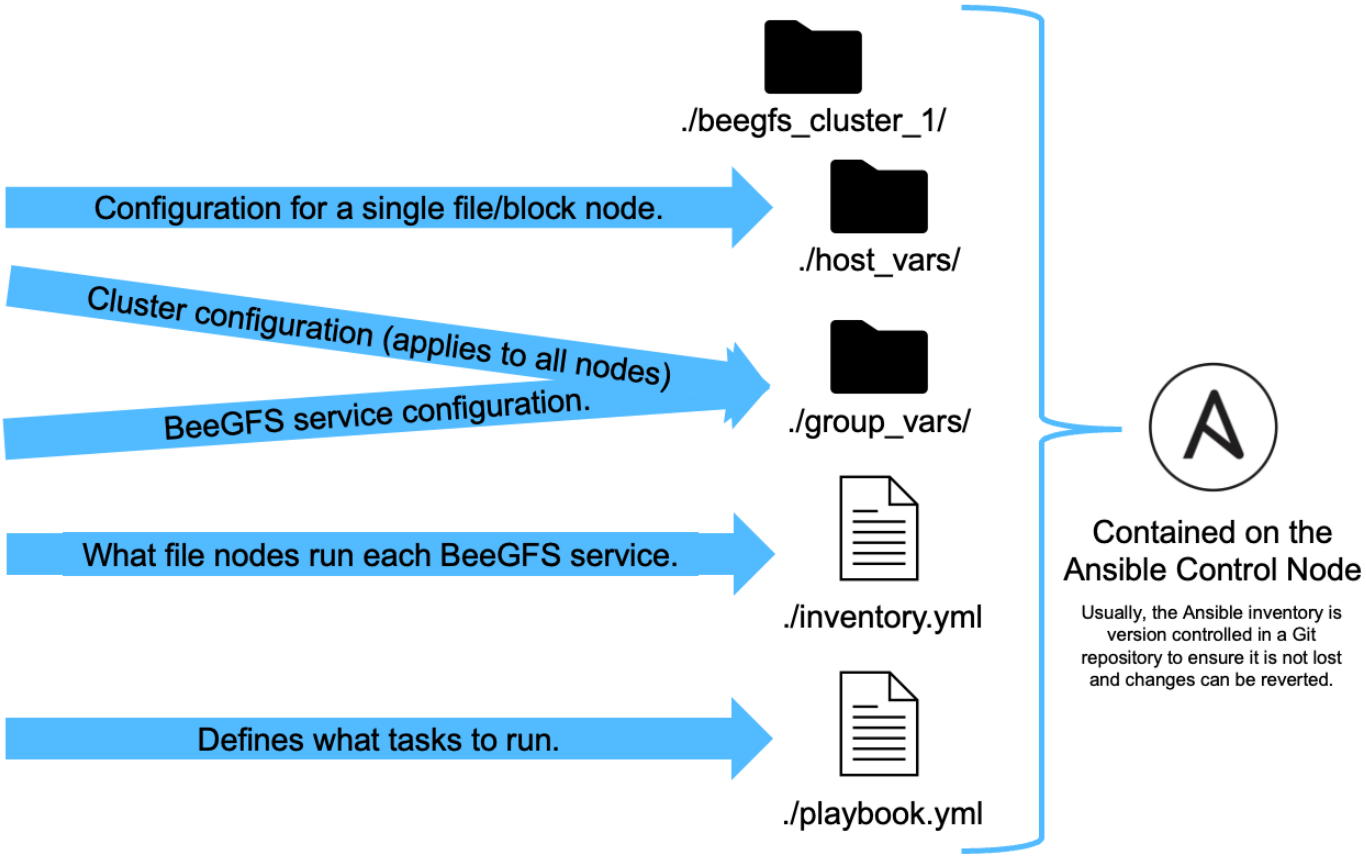
Ansible 인벤토리 개요

Ansible 인벤토리는 원하는 BeeGFS HA 클러스터를 정의하는 구성 파일 세트입니다.

개요

을 구성하기 위한 표준 Ansible 관행을 따르는 것이 좋습니다 "인벤토리"을 참조하십시오 "하위 디렉토리/파일" 전체 재고를 한 파일에 저장하는 대신

단일 BeeGFS HA 클러스터의 Ansible 인벤토리는 다음과 같이 구성됩니다.



단일 BeeGFS 파일 시스템이 여러 HA 클러스터에 걸쳐 있을 수 있으므로 대규모 설치의 경우 여러 Ansible 재고가 있을 수 있습니다. 일반적으로 문제를 피하기 위해 여러 HA 클러스터를 단일 Ansible 인벤토리로 정의하지 않는 것이 좋습니다.

단계

1. Ansible 제어 노드에서 구축할 BeeGFS 클러스터의 Ansible 인벤토리가 포함된 빈 디렉토리를 생성합니다.
   a. 파일 시스템에 여러 HA 클러스터가 포함될 수 있는 경우 먼저 파일 시스템에 대한 디렉토리를 생성한 다음 각 HA 클러스터를 나타내는 인벤토리에 대한 하위 디렉토리를 생성하는 것이 좋습니다. 예를 들면 다음과 같습니다.

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

2. 구축할 HA 클러스터의 인벤토리가 포함된 디렉토리에서 두 개의 디렉토리를 생성합니다 `group_vars` 및 `host_vars` 두 개의 파일이 있습니다 `inventory.yml` 및 `playbook.yml`.

다음 섹션에서는 이러한 각 파일의 내용을 정의하는 방법을 설명합니다.

파일 시스템 계획

Ansible 재고를 구축하기 전에 파일 시스템 배포를 계획하십시오.

개요

파일 시스템을 구축하기 전에 클러스터에서 실행 중인 모든 파일 노드, 블록 노드 및 BeeGFS 서비스에 필요한 IP 주소, 포트 및 기타 구성을 정의해야 합니다. 정확한 구성은 클러스터의 아키텍처에 따라 다르지만 이 섹션에서는 일반적으로 적용되는 모범 사례와 후속 단계를 정의합니다.

단계

1. IP 기반 스토리지 프로토콜(예: iSER, iSCSI, NVMe/IB 또는 NVMe/RoCE)을 사용하여 파일 노드를 블록 노드에 연결하는 경우, 각 구성 요소에 대해 다음 워크시트를 작성하십시오. 단일 빌딩 블록의 각 직접 접속은 고유한 서브넷을 가져야 하며, 클라이언트-서버 접속에 사용되는 서브넷과 중복되지 않아야 합니다.

파일 노드	IB 포트	IP 주소입니다	블록 노드	IB 포트	물리적 IP	가상 IP(HDR IB를 사용하는 EF600용)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

> 각 빌딩 블록의 파일 및 블록 노드가 직접 연결된 경우 여러 빌딩 블록에 동일한 IP/스키마를 재사용할 수 있습니다.

2. 스토리지 네트워크에 InfiniBand 또는 RoCE(RDMA over Converged Ethernet)를 사용하는 경우, 다음 워크시트를 작성하여 HA 클러스터 서비스, BeeGFS 파일 서비스 및 클라이언트가 통신할 수 있는 IP 범위를 결정하십시오.

목적	**InfiniBand** 포트입니다	**IP** 주소 또는 범위입니다
BeeGFS 클러스터 IP	<INTERFACE(s)>	<RANGE>
BeeGFS 관리	<INTERFACE(s)>	<IP(s)>
BeeGFS 메타데이터	<INTERFACE(s)>	<RANGE>
BeeGFS 스토리지	<INTERFACE(s)>	<RANGE>
BeeGFS 클라이언트	<INTERFACE(s)>	<RANGE>

a. 단일 IP 서브넷을 사용하는 경우 워크시트가 하나만 필요합니다. 그렇지 않으면 두 번째 서브넷에 대한 워크시트도 작성하십시오.

3. 위의 내용을 토대로 클러스터의 각 구성 요소에 대해 실행할 BeeGFS 서비스를 정의하는 다음 워크시트를 작성하십시오. 각 서비스에 대해 기본 설정/보조 파일 노드, 네트워크 포트, 부동 IP, NUMA 영역 할당(필요한 경우) 및 대상에 사용할 블록 노드를 지정합니다. 워크시트를 작성할 때 다음 지침을 참조하십시오.

   a. BeeGFS 서비스 중 하나를 지정합니다 `mgmt.yml`, `meta_<ID>.yml`, 또는 `storage_<ID>.yml` 여기서 ID는 이 파일 시스템에서 해당 유형의 모든 BeeGFS 서비스에 대한 고유 번호입니다. 이 규칙은 각 서비스를 구성하는 파일을 생성하는 동안 다음 섹션에서 이 워크시트를 간단하게 참조할 수 있도록 합니다.

   b. BeeGFS 서비스의 포트는 특정 구성 요소 전체에서 고유해야 합니다. 포트 충돌을 방지하기 위해 동일한 포트 번호의 서비스를 동일한 파일 노드에서 실행할 수 없도록 합니다.

   c. 필요한 서비스가 둘 이상의 블록 노드 및/또는 스토리지 풀의 볼륨을 사용할 수 있는 경우(모든 볼륨을 동일한 컨트롤러가 소유할 필요는 없음) 또한 여러 서비스에서 동일한 블록 노드 및/또는 스토리지 풀 구성을 공유할 수 있습니다(개별 볼륨은 이후 섹션에서 정의).

BeeGFS 서비스(파일 이름)	파일 노드	포트	유동 **IP**	**NUMA** 영역	블록 노드	스토리지 풀	소유 컨트롤러
<SERVICE TYPE>_<ID>.대칭	<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>

표준 규칙, 모범 사례 및 작성된 예제 워크시트에 대한 자세한 내용은 "모범 사례""BeeGFS 구성 요소를 정의합니다 "NetApp 검증 아키텍처에서 BeeGFS의 및 섹션을 참조하십시오.

파일 및 블록 노드 정의

개별 파일 노드를 구성합니다

호스트 변수(host_vars)를 사용하여 개별 파일 노드의 구성을 지정합니다.

개요

이 섹션에서는 를 채우는 방법을 설명합니다 `host_vars/<FILE_NODE_HOSTNAME>.yml` 클러스터에 있는 각 파일 노드에 대한 파일입니다. 이러한 파일에는 특정 파일 노드에 고유한 구성만 포함되어야 합니다. 여기에는 일반적으로 다음이 포함됩니다.

- Ansible에서 노드에 연결하는 데 사용해야 하는 IP 또는 호스트 이름 정의
- 다른 파일 노드와 통신하기 위해 HA 클러스터 서비스(박동조율기 및 Corosync)에 사용되는 추가 인터페이스 및 클러스터 IP를 구성합니다. 기본적으로 이러한 서비스는 관리 인터페이스와 동일한 네트워크를 사용하지만 이중화를 위해 추가 인터페이스를 사용할 수 있어야 합니다. 일반적으로 추가 클러스터 또는 관리 네트워크가 필요하지 않도록 스토리지 네트워크에 추가 IP를 정의하는 것이 좋습니다.
  - 클러스터 통신에 사용되는 모든 네트워크의 성능은 파일 시스템 성능에 중요하지 않습니다. 기본 클러스터 구성에서 일반적으로 1Gb/s 이상의 네트워크는 노드 상태 동기화 및 클러스터 리소스 상태 변경 조정과 같은 클러스터 작업에 충분한 성능을 제공합니다. 느리거나 사용량이 많은 네트워크는 리소스 상태 변경이 평소보다 오래 걸릴 수 있으며, 극단적인 경우 적절한 시간 내에 하트비트를 전송할 수 없는 경우 클러스터에서 노드가

제거될 수 있습니다.

- 원하는 프로토콜을 통해 블록 노드에 연결하는 데 사용되는 인터페이스 구성(예: iSCSI/iSER, NVMe/IB, NVMe/RoCE, FCP 등)

단계

"파일 시스템 계획"섹션에 정의된 IP 주소 지정 체계를 참조하여 클러스터의 각 파일 노드에 대해 파일을 `host_vars/<FILE_NODE_HOSTNAME>/yml` 생성하고 다음과 같이 채웁니다.

1. 맨 위에서 Ansible이 노드에 SSH를 통해 사용하고 관리해야 하는 IP 또는 호스트 이름을 지정합니다.

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

2. 클러스터 트래픽에 사용할 수 있는 추가 IP 구성:
   a. 네트워크 유형이 인 경우 "InfiniBand(IPoIB 사용)":

```
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. 네트워크 유형이 인 경우 "RoCE(RDMA over Converged Ethernet)":

```
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. 네트워크 유형이 인 경우 "이더넷(TCP 전용, RDMA 없음)":

```
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. 클러스터 트래픽에 사용해야 하는 IP를 표시하고 기본 IP가 더 높게 나열됨:

```
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.
```

2단계에서 구성된 IPS는 에 포함되지 않는 한 클러스터 IP로 사용되지 않습니다 `beegfs_ha_cluster_node_ips` 목록. 따라서 필요한 경우 다른 목적으로 사용할 수 있는 Ansible을 사용하여 추가 IP/인터페이스를 구성할 수 있습니다.

4. 파일 노드가 IP 기반 프로토콜을 통해 블록 노드와 통신해야 하는 경우 IP를 적절한 인터페이스와 해당 프로토콜에 필요한 모든 패키지를 설치/구성해야 합니다.

   a. 를 사용하는 경우 "iSCSI":

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

   b. 를 사용하는 경우 "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. 를 사용하는 경우 "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. 를 사용하는 경우 "NVMe/RoCE":

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

   e. 기타 프로토콜:

i. 를 사용하는 경우 "NVMe/FC", 개별 인터페이스를 구성할 필요가 없습니다. BeeGFS 클러스터 배포는 자동으로 프로토콜을 감지하고 필요에 따라 요구 사항을 설치/구성합니다. 패브릭을 사용하여 파일 및 블록 노드를 연결하는 경우, 스위치가 NetApp과 스위치 공급업체의 모범 사례에 따라 적절히 조닝되었는지 확인하십시오.

ii. FCP 또는 SAS를 사용하는 경우 추가 소프트웨어를 설치하거나 구성할 필요가 없습니다. FCP를 사용하는 경우 스위치가 다음에 적절하게 조닝(zoning)되어 있는지 확인합니다 "넷엡" 스위치 공급업체의 모범 사례를 소개합니다.

iii. 현재 IB SRP 사용은 권장되지 않습니다. E-Series 블록 노드에서 지원하는 것에 따라 NVMe/IB 또는 iSER을 사용합니다.

을 클릭합니다 "여기" 단일 파일 노드를 나타내는 전체 인벤토리 파일의 예

고급: 이더넷과 **InfiniBand** 모드 간에 **NVIDIA ConnectX VPI** 어댑터를 전환합니다

NVIDIA ConnectX-Virtual Protocol Interconnect & reg;(VPI) 어댑터는 전송 계층으로 InfiniBand와 이더넷을 모두 지원합니다. 모드 간 전환은 자동으로 조정되지 않으며 에 포함된 오픈 소스 패키지인 에 포함된 도구를 사용하여 구성해야 `mstconfig` `mstflint` "NVIDIA 펌웨어 도구(MFT)"합니다. 어댑터 모드 변경은 한 번만 수행하면 됩니다. 이 작업을 수동으로 수행하거나 인벤토리 섹션을 사용하여 구성된 인터페이스의 일부로 Ansible 인벤토리에 포함시켜 자동으로 확인/적용할 수 `eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:` 있습니다.

예를 들어, InfiniBand 모드에서 인터페이스 전류를 이더넷으로 변경하여 RoCE에 사용할 수 있습니다.

1. 구성할 각 인터페이스에 대해 지정합니다 `mstconfig` 를 지정하는 매핑(또는 사전)으로 지정합니다 `LINK_TYPE_P<N>` 위치 `<N>` 인터페이스에 대한 HCA의 포트 번호로 결정됩니다. 를 클릭합니다 `<N>` 값은 를 실행하여 확인할 수 있습니다 `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` PCI 슬롯 이름의 성에 1을 추가하고 10진수로 변환합니다.

   a. 예를 들어, 를 입력합니다 `PCI_SLOT_NAME=0000:2f:00.2` (2+1 → HCA 포트 3) → `LINK_TYPE_P3: eth`:

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

자세한 내용은 를 참조하십시오 "NetApp E-Series 호스트 컬렉션의 문서입니다" 사용 중인 인터페이스 유형 /프로토콜의 경우.

개별 블록 노드를 구성합니다

호스트 변수(host_vars)를 사용하여 개별 블록 노드의 구성을 지정합니다.

개요

이 섹션에서는 를 채우는 방법을 설명합니다 `host_vars/<BLOCK_NODE_HOSTNAME>.yml` 클러스터에 있는 각 블록 노드에 대한 파일입니다. 이러한 파일에는 특정 블록 노드에 고유한 설정만 포함되어야 합니다. 여기에는

일반적으로 다음이 포함됩니다.

- 시스템 이름(System Manager에 표시됨)
- 컨트롤러 중 하나에 대한 HTTPS URL(REST API를 사용하여 시스템을 관리하는 데 사용됨)
- 이 블록 노드에 연결하기 위해 사용하는 스토리지 프로토콜 파일 노드
- IP 주소(필요한 경우)와 같은 HIC(호스트 인터페이스 카드) 포트 구성

단계

"파일 시스템 계획"섹션에 정의된 IP 주소 지정 체계를 참조하여 클러스터의 각 블록 노드에 대해 파일을 `host_vars/<BLOCK_NODE_HOSTNAME>/yml` 생성하고 다음과 같이 채웁니다.

1. 맨 위에서 컨트롤러 중 하나에 대한 시스템 이름과 HTTPS URL을 지정합니다.

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

2. 를 선택합니다 "프로토콜" 파일 노드는 를 사용하여 이 블록 노드에 접속합니다.
   a. 지원되는 프로토콜: `auto`, `iscsi`, `fc`, `sas`, `ib_srp`, `ib_iser`, `nvme_ib`, `nvme_fc`, `nvme_roce`.

```
eseries_initiator_protocol: <PROTOCOL>
```

3. 사용 중인 프로토콜에 따라 HIC 포트는 추가 구성이 필요할 수 있습니다. 필요한 경우 HIC 포트 구성을 정의하여 각 컨트롤러 구성의 상위 항목이 각 컨트롤러의 물리적 가장 왼쪽 포트 및 하단 포트가 가장 오른쪽 포트와 대응하도록 해야 합니다. 모든 포트는 현재 사용되지 않는 경우에도 유효한 구성이 필요합니다.

   EF600 블록 노드와 함께 HDR(200GB) InfiniBand 또는 200GB RoCE를 사용하는 경우에는 아래 섹션도 참조하십시오.

   a. iSCSI의 경우:

```
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. iSER의 경우:

```
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. NVMe/IB의 경우:

```
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. NVMe/RoCE의 경우:

```
eseries_controller_nvme_roce_port:
  controller_a:         # Ordered list of controller A channel
definition.
    - state:            # Whether the port should be enabled.
      config_method:    # Port configuration method Choices: static,
dhcp
      address:          # Port IPv4 address
      subnet_mask:      # Port IPv4 subnet_mask
      gateway:          # Port IPv4 gateway
      mtu:              # Port IPv4 mtu
      speed:            # Port IPv4 speed
  controller_b:         # Ordered list of controller B channel
definition.
    - (...)             # Same as controller A but for controller B

# Alternatively the following common port configuration can be
defined for all ports and omitted above:
eseries_controller_nvme_roce_port_state: enabled         # Generally
specifies whether a controller port definition should be applied
Choices: enabled, disabled
eseries_controller_nvme_roce_port_config_method: dhcp    # General
port configuration method definition for both controllers. Choices:
static, dhcp
eseries_controller_nvme_roce_port_gateway:               # General
port IPv4 gateway for both controllers.
eseries_controller_nvme_roce_port_subnet_mask:           # General
port IPv4 subnet mask for both controllers.
eseries_controller_nvme_roce_port_mtu: 4200              # General
port maximum transfer units (MTU). Any value greater than 1500
(bytes).
eseries_controller_nvme_roce_port_speed: auto            # General
interface speed. Value must be a supported speed or auto for
automatically negotiating the speed with the port.
```

e. FC 및 SAS 프로토콜은 추가 구성이 필요하지 않습니다. SRP가 올바르게 권장되지 않습니다.

iSCSI CHAP 구성 기능을 포함하여 HIC 포트 및 호스트 프로토콜을 구성하는 추가 옵션은 을 참조하십시오 "문서화" SANtricity 컬렉션에 포함되어 있습니다. 참고 BeeGFS를 구축할 때 스토리지 풀, 볼륨 구성 및 스토리지 용량 할당의 기타 측면은 다른 위치에 구성되며 이 파일에 정의하면 안 됩니다.

을 클릭합니다 "여기" 단일 블록 노드를 나타내는 전체 인벤토리 파일의 예

HDR(200GB) InfiniBand 또는 **200GB RoCE**와 **NetApp EF600** 블록 노드 사용**:**

EF600에서 HDR(200GB) InfiniBand를 사용하려면 각 물리적 포트에 대해 두 번째 "가상" IP를 구성해야 합니다. 다음은 이중 포트 InfiniBand HDR HIC가 장착된 EF600을 구성하는 올바른 방법의 예입니다.

```
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)
```

일반 파일 노드 구성을 지정합니다

그룹 변수(group_vars)를 사용하여 일반 파일 노드 구성을 지정합니다.

개요

모든 파일 노드에 대해 사과해야 하는 구성은 에 정의되어 있습니다 `group_vars/ha_cluster.yml`. 일반적으로 다음과 같은 기능이 있습니다.

- 각 파일 노드에 연결 및 로그인하는 방법에 대한 세부 정보
- 공통 네트워킹 구성
- 자동 재부팅이 허용되는지 여부
- 방화벽 및 SELinux 상태를 구성하는 방법
- 경고 및 펜싱을 포함한 클러스터 구성
- 성능 튜닝:
- 공통 BeeGFS 서비스 구성

> ⓘ 이 파일에 설정된 옵션은 혼합 하드웨어 모델을 사용 중이거나 각 노드에 대해 다른 암호를 사용하는 경우와 같이 개별 파일 노드에서도 정의할 수 있습니다. 개별 파일 노드의 구성은 이 파일의 구성보다 우선합니다.

단계

파일을 만듭니다 `group_vars/ha_cluster.yml` 다음과 같이 채웁니다.

1. Ansible Control 노드가 원격 호스트에서 인증해야 하는 방법을 나타냅니다.

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

> 특히 프로덕션 환경에서는 암호를 일반 텍스트로 저장하지 마십시오. 대신 Ansible Vault를 사용하십시오(참조 "Ansible Vault로 콘텐츠 암호화") 또는 을 누릅니다 `--ask-become-pass` 옵션을 클릭합니다. 를 누릅니다 `ansible_ssh_user` 이(가) 이미 루트이므로 필요에 따라 를 생략할 수 있습니다 `ansible_become_password`.

2. 이더넷 또는 InfiniBand 인터페이스(예: 클러스터 IP)에서 정적 IP를 구성하고 여러 인터페이스가 동일한 IP 서브넷에 있는 경우(예: ib0이 192.168.1.10/24를 사용하고 ib1이 192.168.1.11/24를 사용 중인 경우) 멀티홈 지원이 제대로 작동하려면 추가 IP 라우팅 테이블 및 규칙을 설정해야 합니다. 다음과 같이 제공된 네트워크 인터페이스 구성 후크를 활성화하기만 하면 됩니다.

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

3. 클러스터를 구축할 때 스토리지 프로토콜에 따라 노드를 재부팅하여 원격 블록 장치(E-Series 볼륨)를 쉽게 검색하거나 다른 구성 요소를 적용해야 할 수 있습니다. 기본적으로 노드를 재부팅하기 전에 프롬프트가 표시되지만 다음을 지정하여 노드를 자동으로 다시 시작할 수 있습니다.

```
eseries_common_allow_host_reboot: true
```

   a. 재부팅 후 기본적으로 블록 장치 및 기타 서비스가 준비되도록 하려면 Ansible이 시스템이 준비될 때까지 기다립니다 `default.target` 에 도달한 후 배포를 계속합니다. NVMe/IB가 사용 중인 일부 시나리오에서는 이 시간이 부족하여 원격 장치를 초기화, 검색 및 연결할 수 없습니다. 이로 인해 자동화된 배포가 조기에 계속 진행되어 실패할 수 있습니다. NVMe/IB를 사용할 때 이를 방지하려면 다음을 정의합니다.

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

4. BeeGFS 및 HA 클러스터 서비스가 통신하려면 다양한 방화벽 포트가 필요합니다. 첫 번째 명령을 수동으로 구성하지 않는 한(권장하지 않음) 필요한 방화벽 영역을 만들고 포트를 자동으로 열리도록 다음을 지정합니다.

```
beegfs_ha_firewall_configure: True
```

5. 이때 SELinux는 지원되지 않으며, 특히 RDMA를 사용하는 경우 충돌을 피하기 위해 상태를 비활성화로 설정하는 것이 좋습니다. SELinux가 비활성화되었는지 확인하려면 다음을 설정합니다.

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

6. 파일 노드가 통신할 수 있도록 인증을 구성하고 조직의 정책에 따라 필요에 따라 기본값을 조정합니다.

```
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. "파일 시스템 계획"섹션을 기반으로 이 파일 시스템에 대한 BeeGFS 관리 IP를 지정합니다.

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

> ⓘ 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.

8. 원하는 경우 e-메일 알림 활성화:

```
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. 펜싱을 사용하는 것이 좋습니다. 그렇지 않으면 기본 노드에 장애가 발생할 때 보조 노드에서 서비스가 시작되지

않도록 차단할 수 있습니다.

a. 다음을 지정하여 펜싱을 전역적으로 활성화합니다.

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

i. 참고 필요한 경우 여기에서 지원되는 모든 항목을 "클러스터 속성" 지정할 수도 있습니다. BeeGFS HA의 역할은 잘 테스트된 많은 테스트를 거치므로 이러한 조정이 필요하지 않습니다. "기본값"

b. 그런 다음 펜싱 에이전트를 선택하고 구성합니다.

i. 옵션 1: APC PDU(Power Distribution Unit)를 사용하여 펜싱 활성화하기:

```
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. 옵션 2: Lenovo XCC(및 기타 BMC)에서 제공하는 Redfish API를 사용하여 펜싱을 활성화하려면

```
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. 다른 펜싱 에이전트 구성에 대한 자세한 내용은 을 "Red Hat 문서"참조하십시오.

10. BeeGFS HA 역할은 다양한 튜닝 매개 변수를 적용하여 성능을 더욱 최적화할 수 있습니다. 여기에는 커널 메모리 활용도 최적화 및 블록 디바이스 입출력 등이 포함됩니다. 이 역할은 NetApp E-Series 블록 노드를 사용한 테스트를 기반으로 하는 적절한 집합을 "기본값" 제공하지만, 기본적으로 다음을 지정하지 않으면 적용되지 않습니다.

```
beegfs_ha_enable_performance_tuning: True
```

a. 필요한 경우 여기에서 기본 성능 튜닝에 대한 변경 사항도 지정합니다. 자세한 내용은 전체 설명서를 "성능 튜닝 매개 변수" 참조하십시오.

11. BeeGFS 서비스에 사용되는 부동 IP 주소(논리 인터페이스라고도 함)가 파일 노드 간에 페일오버할 수 있도록 모든 네트워크 인터페이스의 이름이 일관되게 지정되어야 합니다. 기본적으로 네트워크 인터페이스 이름은 동일한 PCIe 슬롯에 네트워크 어댑터가 설치된 동일한 서버 모델에서도 일관된 이름을 생성한다는 보장이 없는 커널에 의해 생성됩니다. 이 기능은 장비를 구축하고 생성된 인터페이스 이름을 알 수 있도록 하기 전에 인벤토리를 생성할 때도 유용합니다. 서버 또는 의 블록 다이어그램을 기반으로 일관된 장치 이름을 보장합니다 `lshw -class network -businfo` 출력에서 원하는 PCIe 주소-논리 인터페이스 매핑을 다음과 같이 지정합니다.

a. InfiniBand(IPoIB) 네트워크 인터페이스의 경우:

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

b. 이더넷 네트워크 인터페이스의 경우:

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

인터페이스의 이름을 바꿀 때(이름을 바꿀 수 없음) 충돌을 방지하려면 eth0, ens9f0, ib0 또는 ibs4f0과 같은 잠재적인 기본 이름을 사용하지 않아야 합니다. 일반적인 명명 규칙은 이더넷 또는 InfiniBand의 경우 'e' 또는 'i'를 사용하고 그 뒤에 PCIe 슬롯 번호와 해당 포트를 나타내는 문자를 사용하는 것입니다. 예를 들어 슬롯 3에 설치된 InfiniBand 어댑터의 두 번째 포트는 i3b입니다.

검증된 파일 노드 모델을 사용하는 경우 를 클릭합니다 "여기" PCIe 주소와 논리적 포트 매핑의 예

12. 선택적으로 클러스터의 모든 BeeGFS 서비스에 적용할 구성을 지정합니다. 기본 구성 값을 찾을 수 "여기"있으며 서비스별 구성은 다른 곳에 지정됩니다.

a. BeeGFS 관리 서비스:

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

b. BeeGFS 메타데이터 서비스:

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

c. BeeGFS 스토리지 서비스:

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

13. BeeGFS 7.2.7 및 7.3.1 "연결 인증" 구성 또는 명시적으로 비활성화해야 합니다. Ansible 기반 배포를 사용하여 다음과 같은 몇 가지 방법으로 이를 구성할 수 있습니다.

a. 기본적으로 배포는 연결 인증을 자동으로 구성하고 을 생성합니다 `connauthfile` 모든 파일 노드에 배포되고 BeeGFS 서비스와 함께 사용됩니다. 이 파일은 또한 의 Ansible 제어 노드에 배치/유지됩니다 `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` 이 파일 시스템을 액세스해야 하는 클라이언트에서 재사용하기 위해 안전하게 유지해야 하는 경우

i. 새 키 지정을 생성하려면 다음을 지정합니다 `-e "beegfs_ha_conn_auth_force_new=True` Ansible 플레이북을 실행할 때, 참고 의 경우 이 작업은 무시됩니다 `beegfs_ha_conn_auth_secret` 정의됩니다.

ii. 고급 옵션은 에 포함된 전체 기본값 목록을 "BeeGFS HA 역할입니다"참조하십시오.

b. 에서 다음을 정의하여 사용자 지정 암호를 사용할 수 있습니다 `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. 연결 인증은 완전히 비활성화할 수 있습니다(권장하지 않음).

```
beegfs_ha_conn_auth_enabled: false
```

을 클릭합니다 "여기" 일반 파일 노드 구성을 나타내는 전체 인벤토리 파일의 예

NetApp EF600 블록 노드에서 HDR(200GB) InfiniBand 사용:

EF600에서 HDR(200GB) InfiniBand를 사용하려면 서브넷 관리자가 가상화를 지원해야 합니다. 스위치를 사용하여 파일 및 블록 노드를 연결하는 경우 전체 패브릭의 서브넷 관리자 관리자에서 이 기능을 활성화해야 합니다.

블록 및 파일 노드가 InfiniBand를 사용하여 `opensm` 직접 연결된 경우 블록 노드에 직접 연결된 각 인터페이스에 대해 각 파일 노드에서 의 인스턴스를 구성해야 합니다. 이 작업은 `configure: true` 시기를 지정하여"파일 노드 스토리지 인터페이스를 구성하는 중입니다" 수행합니다.

현재 지원되는 Linux 배포판과 함께 제공된 의 받은 편지함 버전은 `opensm` 가상화를 지원하지 않습니다. 대신 OFED(NVIDIA OpenFabrics Enterprise Distribution)에서 의 버전을 설치하고 구성해야 `opensm` 합니다. Ansible을 사용한 구축도 여전히 지원되지만, 몇 가지 추가 단계가 필요합니다.

1. curl 또는 원하는 툴을 사용하여 NVIDIA 웹 사이트에서 디렉토리로 섹션에 나열된 OpenSM 버전의 패키지를 다운로드합니다. "기술 요구 사항" `<INVENTORY>/packages/` 예를 들면 다음과 같습니다.

```
curl -o packages/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.4/x86_64/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
curl -o packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
https://linux.mellanox.com/public/repo/mlnx_ofed/23.10-
3.2.2.0/rhel9.4/x86_64/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
```

2. 아래에서 `group_vars/ha_cluster.yml` 다음 구성을 정의합니다.

```
### OpenSM package and configuration information
eseries_ib_opensm_allow_upgrades: true
eseries_ib_opensm_skip_package_validation: true
eseries_ib_opensm_rhel_packages: []
eseries_ib_opensm_custom_packages:
  install:
    - files:
        add:
          "packages/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
          "packages/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm": "/tmp/"
    - packages:
        add:
          - /tmp/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
          - /tmp/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
  uninstall:
    - packages:
        remove:
          - opensm
          - opensm-libs
      files:
        remove:
          - /tmp/opensm-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm
          - /tmp/opensm-libs-5.17.2.MLNX20240610.dc7c2998-
0.1.2310322.x86_64.rpm

eseries_ib_opensm_options:
  virt_enabled: "2"
```

일반 블록 노드 구성을 지정합니다

그룹 변수(group_vars)를 사용하여 일반 블록 노드 구성을 지정합니다.

개요

모든 블록 노드에 대해 사과해야 하는 구성은 에 정의되어 있습니다
`group_vars/eseries_storage_systems.yml`. 일반적으로 다음과 같은 기능이 있습니다.

- Ansible 제어 노드를 블록 노드로 사용되는 E-Series 스토리지 시스템에 연결하는 방법에 대한 자세한 정보
- 노드에서 실행해야 하는 펌웨어, NVSRAM 및 드라이브 펌웨어 버전

- 캐시 설정, 호스트 구성 및 볼륨 프로비저닝 방법에 대한 설정을 포함한 글로벌 구성

> 이 파일에 설정된 옵션은 혼합 하드웨어 모델을 사용 중이거나 각 노드에 대해 다른 암호를 사용하는 경우와 같이 개별 블록 노드에서도 정의할 수 있습니다. 개별 블록 노드의 구성은 이 파일의 구성보다 우선합니다.

단계

파일을 만듭니다 `group_vars/eseries_storage_systems.yml` 다음과 같이 채웁니다.

1. Ansible은 SSH를 사용하여 블록 노드에 연결하지 않고 REST API를 사용합니다. 이를 위해 다음을 설정해야 합니다.

```
ansible_connection: local
```

2. 각 노드를 관리할 사용자 이름과 암호를 지정합니다. 사용자 이름은 선택적으로 생략할 수 있으며 기본적으로 admin이 됩니다. 그렇지 않으면 관리자 권한이 있는 계정을 지정할 수 있습니다. 또한 SSL 인증서를 확인해야 하는지 무시해야 하는지 여부를 지정합니다.

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

> 암호를 일반 텍스트로 나열하는 것은 권장되지 않습니다. Ansible 볼트를 사용하거나 을 제공합니다 `eseries_system_password` VAR을(를) 사용하여 Ansible을 실행하는 경우

3. 노드에 설치할 컨트롤러 펌웨어, NVSRAM 및 드라이브 펌웨어를 선택적으로 지정합니다. 이러한 파일은 로 다운로드해야 합니다 `packages/` Ansible을 실행하기 전 디렉토리: E-Series 컨트롤러 펌웨어 및 NVSRAM을 다운로드할 수 있습니다 "여기" 및 드라이브 펌웨어를 업데이트합니다 "여기":

```
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".
```

> 이 구성을 지정하면 Ansible이 추가 프롬프트 없이 컨트롤러 재부팅(필요한 경우)을 비롯한 모든 펌웨어를 자동으로 업데이트합니다. 이는 BeeGFS/호스트 입출력에 영향을 줄 수 있지만 일시적으로 성능이 저하될 수 있습니다.

4. 글로벌 시스템 구성 기본값을 조정합니다. 여기에 나열된 옵션과 값은 NetApp 기반의 BeeGFS에 일반적으로 권장되지만 필요한 경우 조정할 수 있습니다.

```
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. 글로벌 볼륨 프로비저닝 기본값을 구성합니다. 여기에 나열된 옵션과 값은 NetApp 기반의 BeeGFS에 일반적으로 권장되지만 필요한 경우 조정할 수 있습니다.

```
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. 필요한 경우 Ansible에서 스토리지 풀 및 볼륨 그룹을 위한 드라이브를 선택하는 순서를 조정하고 다음 모범 사례를 염두에 두십시오.

   a. 관리 및/또는 메타데이터 볼륨에 먼저 사용되어야 하는 (잠재적으로 작은) 드라이브와 스토리지 볼륨을 나열합니다.

   b. 디스크 쉘프/드라이브 엔클로저 모델을 기준으로 사용 가능한 드라이브 채널 간에 드라이브 선택 순서를 조정해야 합니다. 예를 들어 EF600과 확장 없는 경우 드라이브 0-11은 드라이브 채널 1에 있고 드라이브 12-23은 드라이브 채널에 있습니다. 따라서 드라이브 선택의 균형을 맞추는 전략은 선택입니다 `disk shelf:drive` 99:0, 99:23, 99:1, 99:22 등 하나 이상의 엔클로저가 있는 경우 첫 번째 숫자는 드라이브 쉘프 ID를 나타냅니다.

```
# 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"
```

을 클릭합니다 "여기" 일반 블록 노드 구성을 나타내는 전체 인벤토리 파일의 예

BeeGFS 서비스를 정의합니다

BeeGFS 관리 서비스를 정의합니다

BeeGFS 서비스는 그룹 변수(group_VAR)를 사용하여 구성됩니다.

개요

이 섹션에서는 BeeGFS 관리 서비스 정의에 대해 설명합니다. 특정 파일 시스템에 대한 HA 클러스터에는 이 유형의 서비스 하나만 있어야 합니다. 이 서비스를 구성하는 데는 다음 사항이 포함됩니다.

- 서비스 유형(관리)
- 이 BeeGFS 서비스에만 적용해야 하는 구성을 정의합니다.
- 이 서비스에 연결할 수 있는 하나 이상의 부동 IP(논리 인터페이스)를 구성합니다.
- 이 서비스에 대한 데이터를 저장할 볼륨 위치/방법 지정(BeeGFS 관리 타겟)

단계

새 파일을 `group_vars/mgmt.yml` 만들고 "파일 시스템 계획"섹션을 참조하여 다음과 같이 채웁니다.

1. 이 파일이 BeeGFS 관리 서비스의 구성을 나타냅니다.

```
beegfs_service: management
```

2. 이 BeeGFS 서비스에만 적용해야 하는 구성을 정의합니다. 할당량을 설정해야 하는 경우가 아니면 일반적으로 관리 서비스에 필요하지 않지만 에서 지원되는 구성 매개 변수는 필요하지 않습니다 `beegfs-mgmtd.conf` 포함될 수 있습니다. 참고 다음 매개 변수는 자동으로/다른 위치에 구성되며 여기에서 지정하면 안 됩니다. `storeMgmtdDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, 및 `connNetFilterFile`.

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

3. 다른 서비스 및 클라이언트가 이 서비스에 연결하는 데 사용할 하나 이상의 부동 IP를 구성합니다. 이렇게 하면 BeeGFS가 자동으로 설정됩니다 `connInterfacesFile` 옵션):

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

4. 선택적으로, 나가는 통신에 사용할 수 있는 하나 이상의 허용된 IP 서브넷을 지정합니다(이 경우 BeeGFS가 자동으로 설정됩니다) `connNetFilterFile` 옵션):

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

5. 다음 지침에 따라 이 서비스가 데이터를 저장할 BeeGFS 관리 대상을 지정합니다.
   a. 여러 BeeGFS 서비스/타겟에 동일한 스토리지 풀 또는 볼륨 그룹 이름을 사용할 수 있으므로 동일한 스토리지 풀 또는 볼륨 그룹 이름을 사용하기만 하면 됩니다 `name`, `raid_level`, `criteria_*`, 및 `common_*` 각 서비스에 대한 구성(각 서비스에 대해 나열된 볼륨은 서로 달라야 함)
   b. 볼륨 크기는 스토리지 풀/볼륨 그룹의 백분율로 지정해야 하며, 특정 스토리지 풀/볼륨 그룹을 사용하는 모든 서비스/볼륨에서 합계가 100을 초과해서는 안 됩니다. 참고 SSD를 사용할 경우 SSD 성능 및 마모 수명을 최대화하기 위해 볼륨 그룹에 여유 공간을 두는 것이 좋습니다("여기"자세한 내용을 보려면 클릭).
   c. 을 클릭합니다 "여기" 에서 사용할 수 있는 전체 구성 옵션 목록을 확인하십시오 `eseries_storage_pool_configuration`. 과 같은 일부 옵션을 확인합니다 `state`, `host`, `host_type`, `workload_name`, 및 `workload_metadata` 및 볼륨 이름은 자동으로 생성되며 여기에서 지정할 수 없습니다.

```
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
```

을 클릭합니다 "여기" BeeGFS 관리 서비스를 나타내는 전체 인벤토리 파일의 예

BeeGFS 메타데이터 서비스를 정의합니다

BeeGFS 서비스는 그룹 변수(group_VAR)를 사용하여 구성됩니다.

개요

이 섹션에서는 BeeGFS 메타데이터 서비스 정의에 대해 설명합니다. 특정 파일 시스템에 대한 HA 클러스터에 이 유형의 서비스가 하나 이상 있어야 합니다. 이 서비스를 구성하는 데는 다음 사항이 포함됩니다.

- 서비스 유형(메타데이터)
- 이 BeeGFS 서비스에만 적용해야 하는 구성을 정의합니다.
- 이 서비스에 연결할 수 있는 하나 이상의 부동 IP(논리 인터페이스)를 구성합니다.
- 이 서비스에 대한 데이터를 저장할 볼륨 위치/방법 지정(BeeGFS 메타데이터 타겟)

단계

"파일 시스템 계획"섹션을 참조하여 `group_vars/meta_<ID>.yml` 클러스터의 각 메타데이터 서비스에 대해 에서 파일을 생성하고 다음과 같이 채웁니다.

1. 이 파일이 BeeGFS 메타데이터 서비스에 대한 구성을 나타냅니다.

```
beegfs_service: metadata
```

2. 이 BeeGFS 서비스에만 적용해야 하는 구성을 정의합니다. 최소한 원하는 TCP 및 UDP 포트를 지정해야 하지만 에서 지원되는 구성 매개 변수는 모두 지정해야 합니다 `beegfs-meta.conf` 또한 포함될 수 있습니다. 참고 다음 매개 변수는 자동으로/다른 위치에 구성되며 여기에서 지정하면 안 됩니다. `sysMgmtdHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, 및 `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. 다른 서비스 및 클라이언트가 이 서비스에 연결하는 데 사용할 하나 이상의 부동 IP를 구성합니다. 이렇게 하면 BeeGFS가 자동으로 설정됩니다 `connInterfacesFile` 옵션):

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

4. 선택적으로, 나가는 통신에 사용할 수 있는 하나 이상의 허용된 IP 서브넷을 지정합니다(이 경우 BeeGFS가 자동으로 설정됩니다) `connNetFilterFile` 옵션):

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

5. 다음 지침에 따라 이 서비스가 데이터를 저장할 BeeGFS 메타데이터 타겟을 지정합니다. 이렇게 하면 가 자동으로 구성됩니다 `storeMetaDirectory` 옵션):

   a. 여러 BeeGFS 서비스/타겟에 동일한 스토리지 풀 또는 볼륨 그룹 이름을 사용할 수 있으므로 동일한 스토리지 풀 또는 볼륨 그룹 이름을 사용하기만 하면 됩니다 `name`, `raid_level`, `criteria_*`, 및 `common_*` 각 서비스에 대한 구성(각 서비스에 대해 나열된 볼륨은 서로 달라야 함)

   b. 볼륨 크기는 스토리지 풀/볼륨 그룹의 백분율로 지정해야 하며, 특정 스토리지 풀/볼륨 그룹을 사용하는 모든 서비스/볼륨에서 합계가 100을 초과해서는 안 됩니다. 참고 SSD를 사용할 경우 SSD 성능 및 마모 수명을 최대화하기 위해 볼륨 그룹에 여유 공간을 두는 것이 좋습니다("여기"자세한 내용을 보려면 클릭).

c. 을 클릭합니다 "여기" 에서 사용할 수 있는 전체 구성 옵션 목록을 확인하십시오 `eseries_storage_pool_configuration`. 과 같은 일부 옵션을 확인합니다 `state`, `host`, `host_type`, `workload_name`, 및 `workload_metadata` 및 볼륨 이름은 자동으로 생성되며 여기에서 지정할 수 없습니다.

```
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
```

을 클릭합니다 "여기" BeeGFS 메타데이터 서비스를 나타내는 전체 인벤토리 파일의 예

BeeGFS 스토리지 서비스를 정의합니다

BeeGFS 서비스는 그룹 변수(group_VAR)를 사용하여 구성됩니다.

개요

이 섹션에서는 BeeGFS 스토리지 서비스 정의를 안내합니다. 특정 파일 시스템에 대한 HA 클러스터에 이 유형의 서비스가 하나 이상 있어야 합니다. 이 서비스를 구성하는 데는 다음 사항이 포함됩니다.

- 서비스 유형(스토리지)
- 이 BeeGFS 서비스에만 적용해야 하는 구성을 정의합니다.
- 이 서비스에 연결할 수 있는 하나 이상의 부동 IP(논리 인터페이스)를 구성합니다.
- 이 서비스에 대한 데이터를 저장할 볼륨 위치/방법 지정(BeeGFS 스토리지 타겟)

단계

"파일 시스템 계획"섹션을 참조하여 `group_vars/stor_<ID>.yml` 클러스터의 각 스토리지 서비스에 대해 에서 파일을 생성하고 다음과 같이 채웁니다.

1. 이 파일이 BeeGFS 스토리지 서비스에 대한 구성을 나타냅니다.

```
beegfs_service: storage
```

2. 이 BeeGFS 서비스에만 적용해야 하는 구성을 정의합니다. 최소한 원하는 TCP 및 UDP 포트를 지정해야 하지만 에서 지원되는 구성 매개 변수는 모두 지정해야 합니다 `beegfs-storage.conf` 또한 포함될 수 있습니다. 참고

다음 매개 변수는 자동으로/다른 위치에 구성되며 여기에서 지정하면 안 됩니다. `sysMgmtdHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, 및 `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. 다른 서비스 및 클라이언트가 이 서비스에 연결하는 데 사용할 하나 이상의 부동 IP를 구성합니다. 이렇게 하면 BeeGFS가 자동으로 설정됩니다 `connInterfacesFile` 옵션):

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

4. 선택적으로, 나가는 통신에 사용할 수 있는 하나 이상의 허용된 IP 서브넷을 지정합니다(이 경우 BeeGFS가 자동으로 설정됩니다) `connNetFilterFile` 옵션):

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

5. 다음 지침에 따라 이 서비스가 데이터를 저장할 BeeGFS 스토리지 타겟을 지정합니다. 이 경우에도 가 자동으로 구성됩니다 `storeStorageDirectory` 옵션):

   a. 여러 BeeGFS 서비스/타겟에 동일한 스토리지 풀 또는 볼륨 그룹 이름을 사용할 수 있으므로 동일한 스토리지 풀 또는 볼륨 그룹 이름을 사용하기만 하면 됩니다 `name`, `raid_level`, `criteria_*`, 및 `common_*` 각 서비스에 대한 구성(각 서비스에 대해 나열된 볼륨은 서로 달라야 함)

   b. 볼륨 크기는 스토리지 풀/볼륨 그룹의 백분율로 지정해야 하며, 특정 스토리지 풀/볼륨 그룹을 사용하는 모든 서비스/볼륨에서 합계가 100을 초과해서는 안 됩니다. 참고 SSD를 사용할 경우 SSD 성능 및 마모 수명을 최대화하기 위해 볼륨 그룹에 여유 공간을 두는 것이 좋습니다("여기"자세한 내용을 보려면 클릭).

   c. 을 클릭합니다 "여기" 에서 사용할 수 있는 전체 구성 옵션 목록을 확인하십시오 `eseries_storage_pool_configuration`. 과 같은 일부 옵션을 확인합니다 `state`, `host`, `host_type`, `workload_name`, 및 `workload_metadata` 및 볼륨 이름은 자동으로 생성되며 여기에서 지정할 수 없습니다.

```
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
```

을 클릭합니다 "여기" BeeGFS 스토리지 서비스를 나타내는 전체 인벤토리 파일의 예

BeeGFS 서비스를 파일 노드에 매핑합니다

를 사용하여 각 BeeGFS 서비스를 실행할 수 있는 파일 노드를 지정합니다 `inventory.yml` 파일.

개요

이 섹션에서는 을 생성하는 방법을 안내합니다 `inventory.yml` 파일. 여기에는 모든 블록 노드를 나열하고 각 BeeGFS 서비스를 실행할 수 있는 파일 노드를 지정하는 작업이 포함됩니다.

단계

파일을 만듭니다 `inventory.yml` 다음과 같이 채웁니다.

1. 파일 상단에서 표준 Ansible 인벤토리 구조를 생성합니다.

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

2. 이 HA 클러스터에 참여하는 모든 블록 노드를 포함하는 그룹을 생성합니다.

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

3. 클러스터의 모든 BeeGFS 서비스를 포함할 그룹과 해당 서비스를 실행할 파일 노드를 생성합니다.

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

4. 클러스터의 각 BeeGFS 서비스에 대해 해당 서비스를 실행해야 하는 기본 파일 노드 및 보조 파일 노드를 정의합니다.

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

을 클릭합니다 "여기" 전체 재고 파일의 예

BeeGFS 파일 시스템을 구축합니다

Ansible 플레이북 개요

Ansible을 사용하여 BeeGFS HA 클러스터 구축 및 관리

개요

이전 섹션에서는 BeeGFS HA 클러스터를 나타내는 Ansible 재고를 구축하는 데 필요한 단계를 안내했습니다. 이 섹션에서는 클러스터를 구축 및 관리하기 위해 NetApp에서 개발한 Ansible 자동화를 소개합니다.

Ansible: 주요 개념

진행하기 전에 Ansible의 몇 가지 주요 개념을 숙지하는 것이 좋습니다.

- Ansible 인벤토리에 대해 실행할 작업은 * 플레이북 * 으로 알려진 내용에 정의됩니다.
  - Ansible에서 수행하는 대부분의 작업은 * idemtiple * 으로 설계되어 여러 번 실행할 수 있으므로, 원하는 구성/상태가 중단 없이 적용되었는지, 불필요한 업데이트를 하지 않고 그대로 적용되는지를 확인할 수 있습니다.
- Ansible에서 실행할 수 있는 가장 작은 단위는 * 모듈 * 입니다.

    - 일반적인 플레이북에서는 여러 모듈을 사용합니다.
        - 예: 패키지 다운로드, 구성 파일 업데이트, 서비스 시작/활성화
    - NetApp은 모듈을 분산하여 NetApp E-Series 시스템을 자동화합니다.
- 복잡한 자동화는 하나의 역할로 더 잘 패키징됩니다.
    - 기본적으로 재사용 가능한 플레이북을 배포하기 위한 표준 형식입니다.
    - NetApp은 Linux 호스트 및 BeeGFS 파일 시스템에 대한 역할을 분산합니다.

Ansible용 BeeGFS HA 역할: 주요 개념

NetApp에서 각 버전의 BeeGFS를 구축 및 관리하는 데 필요한 모든 자동화 기능이 Ansible 역할로 패키지되어 의 일부로 배포됩니다 "BeeGFS용 NetApp E-Series Ansible 컬렉션":

- 이 역할은 * 설치 프로그램 * 과 BeeGFS용 최신 * 구축/관리 * 엔진 사이의 어딘가에 있다고 생각할 수 있습니다.
    - 최신 인프라를 코드 사례 및 철학으로 적용하여 모든 규모의 스토리지 인프라 관리를 단순화합니다.
    - "구베기도"사용자가 스케일아웃 컴퓨팅 인프라를 위한 전체 Kubernetes 배포를 배포/유지 관리할 수 있는 프로젝트와 비슷합니다.
- NetApp에서 NetApp 솔루션에서 BeeGFS를 패키징, 배포 및 유지 관리하는 데 사용하는 * 소프트웨어 정의 * 형식입니다.
    - 전체 Linux 배포판이나 큰 이미지를 배포할 필요 없이 "어플라이언스와 유사한" 환경을 조성하기 위해 노력합니다.
    - 지능형 Pacemaker/BeeGFS 통합을 제공하는 맞춤형 BeeGFS 타겟, IP 주소, 모니터링을 위해 NetApp에서 작성한 OCF(Open Cluster Framework) 규격 클러스터 리소스 에이전트가 포함되어 있습니다.
- 이 역할은 단순히 "자동화"를 구축하는 것이 아니라 다음을 포함한 전체 파일 시스템 수명주기를 관리하는 데 사용됩니다.
    - 서비스별 또는 클러스터 전체 구성 변경 및 업데이트 적용
    - 하드웨어 문제가 해결된 후 클러스터 복구 및 복구 자동화
    - BeeGFS 및 NetApp 볼륨을 사용한 광범위한 테스트 결과를 기준으로 설정된 기본값으로 성능 조정을 단순화합니다.
    - 구성 드리프트 확인 및 수정

NetApp은 또한 Ansible 역할을 제공합니다 "BeeGFS 클라이언트"필요에 따라 BeeGFS를 설치하고 파일 시스템을 컴퓨팅/GPU/로그인 노드에 마운트하는 데 사용할 수 있습니다.

BeeGFS HA 클러스터를 구축합니다

Playbook을 사용하여 BeeGFS HA 클러스터를 구축하기 위해 실행해야 할 작업을 지정합니다.

개요

이 섹션에서는 NetApp에서 BeeGFS를 구축/관리하는 데 사용되는 표준 플레이북을 취합하는 방법에 대해 설명합니다.

단계

Ansible 플레이북을 작성합니다

파일을 만듭니다 `playbook.yml` 다음과 같이 채웁니다.

1. 먼저 작업 집합(일반적으로 라고 함)을 정의합니다 "재생") NetApp E-Series 블록 노드에서만 실행되어야 합니다. 일시 중지 작업을 사용하여 설치를 실행하기 전에 메시지를 표시한 다음(우발적인 플레이북 실행을 피하기 위해) 을(를) 가져옵니다 `nar_santricity_management` 역할. 이 역할은 에 정의된 모든 일반 시스템 구성을 적용하는 작업을 처리합니다 `group_vars/eseries_storage_systems.yml` 있습니다 `host_vars/<BLOCK NODE>.yml` 파일.

```
- 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. 모든 파일 및 블록 노드에 대해 실행할 플레이를 정의합니다.

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

3. 이 플레이에서는 HA 클러스터를 구축하기 전에 실행해야 하는 "사전 작업" 세트를 선택적으로 정의할 수 있습니다. Python과 같은 필수 구성 요소를 확인/설치하는 데 유용할 수 있습니다. 제공된 Ansible 태그가 지원되는지 확인하는 등 비행 전 점검을 삽입할 수도 있습니다.

```
  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. 마지막으로, 이 플레이는 구축할 BeeGFS 버전에 대한 BeeGFS HA 역할을 가져옵니다.

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

지원되는 각 주요 버전 BeeGFS에 대해 BeeGFS HA 역할이 유지됩니다. 따라서 사용자는 주/부 버전을 업그레이드할 시기를 선택할 수 있습니다. 현재 BeeGFS 7.3.x(`beegfs_7_3`) 또는 BeeGFS 7.2.(`beegfs_7_2`x)가 지원됩니다. 기본적으로 두 역할 모두 릴리즈 시점에 최신 BeeGFS 패치 버전을 구축합니다. 하지만 사용자가 원할 경우 이를 무시하고 최신 패치를 배포할 수 있습니다. "업그레이드 가이드"자세한 내용은 최신 을 참조하십시오.

5. 선택 사항: 추가 작업을 정의하려면 작업이 에 지정되어야 하는지 여부를 염두에 두십시오 `all` 호스트(E-Series 스토리지 시스템 포함) 또는 파일 노드만 포함됩니다. 필요한 경우 를 사용하여 파일 노드를 대상으로 하는 새로운 플레이를 정의합니다 `- hosts: ha_cluster`.

을 클릭합니다 "여기" 전체 플레이북 파일의 예

NetApp Ansible Collections를 설치합니다

Ansible용 BeeGFS 컬렉션 및 모든 종속 항목이 에 유지됩니다 "Ansible 갤럭시". Ansible 제어 노드에서 다음 명령을 실행하여 최신 버전을 설치합니다.

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

일반적으로 권장하지는 않지만 컬렉션의 특정 버전을 설치할 수도 있습니다.

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

Playbook을 실행합니다

이 포함된 Ansible 제어 노드의 디렉토리에서 `inventory.yml` 및 `playbook.yml` 파일을 실행하고 다음과 같이 플레이북을 실행합니다.

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

클러스터의 크기에 따라 초기 구축에는 20분 이상 걸릴 수 있습니다. 어떠한 이유로든 구축에 실패하는 경우, 잘못된 케이블 연결, 노드 시작 등 문제를 해결하고 Ansible 플레이북을 다시 시작하십시오.

를 지정할 때 "공통 파일 노드 구성"Ansible에서 연결 기반 인증을 자동으로 관리하도록 기본 옵션을 선택한 경우 `connAuthFile` 공유 암호로 사용되는 을 `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (기본적으로) 에서 찾을 수 있습니다. 파일 시스템에 액세스해야 하는 모든 클라이언트는 이 공유 암호를 사용해야 합니다. 이 작업은 클라이언트를 를 사용하여 구성한 경우 자동으로 "BeeGFS 클라이언트 역할입니다"처리됩니다.

BeeGFS 클라이언트를 구축합니다

선택적으로 Ansible을 사용하여 BeeGFS 클라이언트를 구성하고 파일 시스템을 마운트할 수 있습니다.

개요

BeeGFS 파일 시스템을 액세스하려면 파일 시스템을 마운트해야 하는 각 노드에서 BeeGFS 클라이언트를 설치 및 구성해야 합니다. 이 섹션에서는 사용 가능한 를 사용하여 이러한 작업을 수행하는 방법을 설명합니다 "Ansible 역할".

단계

클라이언트 인벤토리 파일을 생성합니다

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

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

2. 아래에서 `host_vars/`에서 각 `BeeGFS` 클라이언트에 대한 파일을 생성합니다 `<HOSTNAME>.yml` 다음 콘텐츠를 사용하여 환경에 맞는 올바른 정보로 자리 표시자 텍스트를 입력합니다.

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

3. 선택적으로 NetApp E-Series 호스트 컬렉션의 역할을 사용하여 클라이언트가 BeeGFS 파일 노드에 연결할 수 있도록 InfiniBand 또는 이더넷 인터페이스를 구성하려면 다음 중 하나를 포함합니다.

   a. 네트워크 유형이 인 경우 "InfiniBand(IPoIB 사용)":

```
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. 네트워크 유형이 인 경우 "RoCE(RDMA over Converged Ethernet)":

```
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. 네트워크 유형이 인 경우 "이더넷(TCP 전용, RDMA 없음)":

```
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. 새 파일을 만듭니다 `client_inventory.yml` 그리고 Ansible이 각 클라이언트에 연결하는 데 사용해야 하는 사용자 지정과 Ansible이 권한 에스컬레이션을 위해 사용해야 하는 암호(이 경우 필요)를 지정합니다 `ansible_ssh_user` 루트 또는 sudo 권한 보유):

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

암호를 일반 텍스트로 저장하지 마십시오. 대신 Ansible Vault를 사용하십시오(참조 "Ansible 설명서" Ansible Vault로 콘텐츠 암호화)를 사용하거나 을 사용합니다 `--ask-become-pass` 옵션을 클릭합니다.

5. 에 있습니다 `client_inventory.yml` File(파일): 에 BeeGFS 클라이언트로 구성해야 하는 모든 호스트를 나열합니다 `beegfs_clients` 그룹화한 다음 인라인 주석을 참조하여 시스템에서 BeeGFS 클라이언트 커널 모듈을 구축하는 데 필요한 추가 구성을 제거합니다.

```
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.
```

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

6. 에 있습니다 `client_inventory.yml` 파일, 이전에 정의한 모든 파일 아래에 마운트할 BeeGFS 파일 시스템을 나열합니다 `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. BeeGFS 7.2.7 및 7.3.1부터 "연결 인증"구성하거나 명시적으로 사용하지 않도록 설정해야 합니다. 를 지정할 때 연결 기반 인증을 구성하는 방법에 따라 "공통 파일 노드 구성"클라이언트 구성을 조정해야 할 수도 있습니다.

   a. 기본적으로 HA 클러스터 배포는 연결 인증을 자동으로 구성하고 를 생성합니다 `connauthfile` 이 정보는 에서 Ansible 제어 노드에 배치/유지됩니다 `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. 기본적으로 BeeGFS 클라이언트 역할은 에 정의된 클라이언트에 이 파일을 읽고 배포하도록 설정되어 있습니다 `client_inventory.yml`추가 조치가 필요하지 않습니다.

      i. 고급 옵션은 에 포함된 기본값 전체 목록을 참조하십시오 "BeeGFS 클라이언트 역할입니다".

   b. 을 사용하여 사용자 지정 암호를 지정하도록 선택한 경우 `beegfs_ha_conn_auth_secret` 에서 지정합니다 `client_inventory.yml` 파일 또한:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

   c. 을 사용하여 연결 기반 인증을 완전히 사용하지 않도록 선택하는 경우 `beegfs_ha_conn_auth_enabled`에서 를 지정합니다 `client_inventory.yml` 파일 또한:

```
beegfs_ha_conn_auth_enabled: false
```

지원되는 매개 변수의 전체 목록과 추가 세부 정보는 를 참조하십시오 "전체 BeeGFS 클라이언트 문서". 클라이언트 인벤토리의 전체 예제를 보려면 을 클릭합니다 "여기".

BeeGFS Client Playbook File을 생성합니다

1. 새 파일을 만듭니다 `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. 선택 사항: NetApp E-Series Host Collection의 역할을 사용하여 클라이언트가 BeeGFS 파일 시스템에 연결할 수 있도록 인터페이스를 구성하려면 구성 중인 인터페이스 유형에 해당하는 역할을 가져옵니다.

   a. InfiniBand(IPoIB)를 사용하는 경우:

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

   b. RoCE(RDMA over Converged Ethernet)를 사용 중인 경우:

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

   c. 를 사용 중인 경우 이더넷(TCP 전용, RDMA 없음)을 사용합니다.

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

3. 마지막으로 BeeGFS 클라이언트 역할을 가져와 클라이언트 소프트웨어를 설치하고 파일 시스템 마운트를 설정합니다.

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

클라이언트 플레이북의 전체 예제를 보려면 을 클릭합니다 "여기".

BeeGFS Client Playbook을 실행합니다

클라이언트를 설치/구축하고 BeeGFS를 마운트하려면 다음 명령을 실행합니다.

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

BeeGFS 구축을 확인합니다

시스템을 운영 환경에 배치하기 전에 파일 시스템 구축을 확인하십시오.

개요

BeeGFS 파일 시스템을 운영 환경에 배치하기 전에 몇 가지 검증 검사를 수행하십시오.

단계

1. 모든 클라이언트에 로그인하고 다음을 실행하여 모든 예상 노드가 존재하고 연결 가능한지, 불일치 또는 보고된 다른 문제가 없는지 확인합니다.

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

2. 전체 클러스터를 종료한 다음 재시작합니다. 모든 파일 노드에서 다음을 실행합니다.

```
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. 각 노드를 스탠바이에 배치하고 BeeGFS 서비스가 보조 노드로 페일오버할 수 있는지 확인합니다. 이 작업을 수행하려면 파일 노드에 로그인하고 다음을 실행합니다.

```
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. IOR 및 MDTest와 같은 성능 벤치마킹 툴을 사용하여 파일 시스템 성능이 기대에 부합하는지 확인합니다. BeeGFS와 함께 사용되는 일반 테스트 및 매개 변수의 예는 "설계 Verification"NetApp 검증 아키텍처의 BeeGFS 섹션에서 찾을 수 있습니다.

특정 현장/설치에 대해 정의된 수용 기준에 따라 추가 테스트를 수행해야 합니다.