Defina o sistema de arquivos BeeGFS

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Índice

Defina o sistema de arquivos BeeGFS

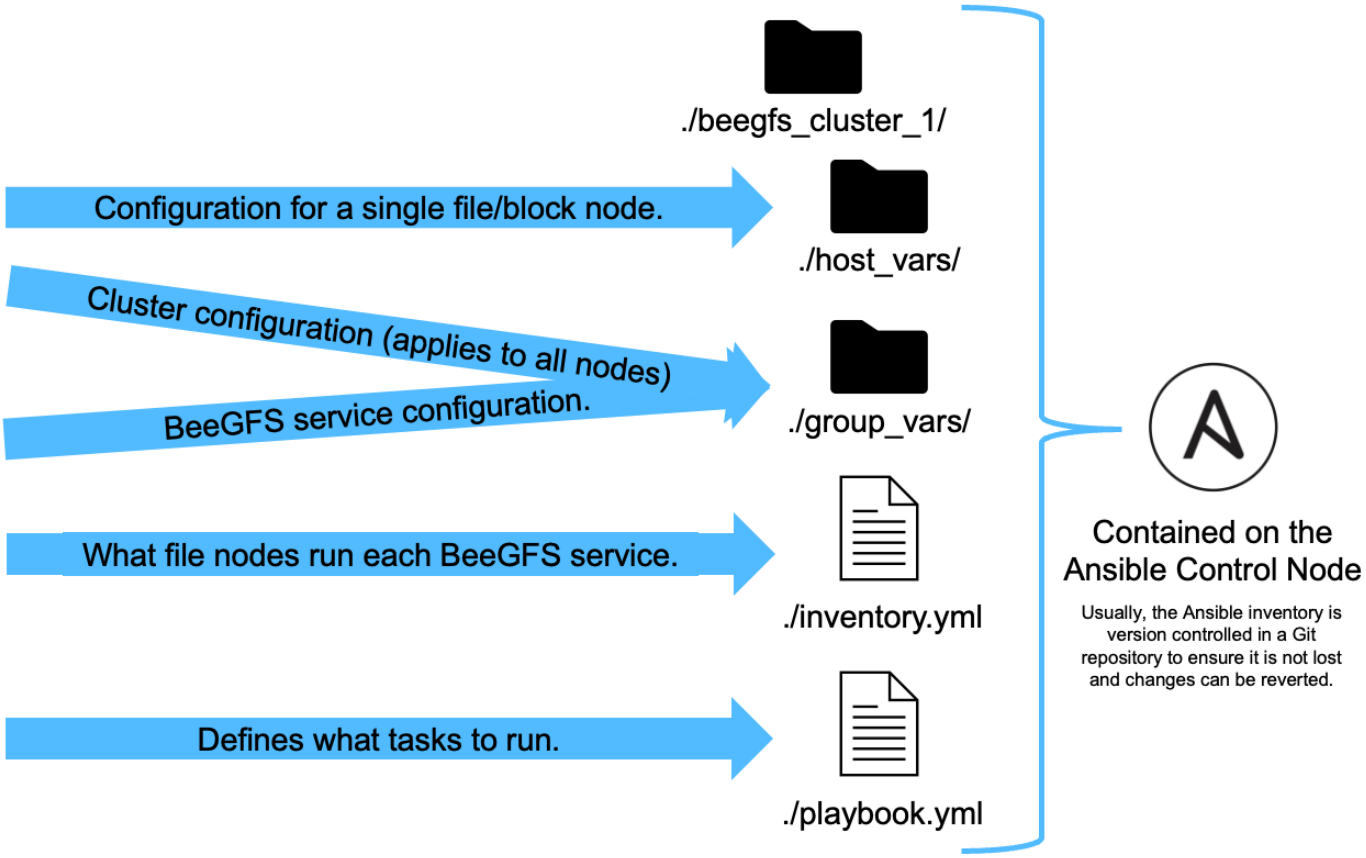
Visão geral do Ansible Inventory

O inventário do Ansible é um conjunto de arquivos de configuração que definem o cluster de HA do BeeGFS desejado.

Visão geral

Recomenda-se seguir as práticas padrão do Ansible para organizar o "inventário", incluindo o uso do , "subdiretórios/ficheiros"em vez de armazenar todo o inventário em um arquivo.

O inventário do Ansible para um único cluster BeeGFS HA está organizado da seguinte forma:



Como um único sistema de arquivos BeeGFS pode abranger vários clusters de HA, é possível que grandes instalações tenham vários inventários do Ansible. Geralmente, não é recomendável definir vários clusters de HA como um único inventário do Ansible para evitar problemas.

Passos

1. No nó de controle do Ansible, crie um diretório vazio que conterá o inventário do Ansible para o cluster BeeGFS que você deseja implantar.
   a. Se o seu sistema de arquivos eventualmente contiver vários clusters de HA, é recomendável criar primeiro um diretório para o sistema de arquivos e, em seguida, subdiretórios para o inventário que

representa cada cluster de HA. Por exemplo:

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

2. No diretório que contém o inventário do cluster HA que deseja implantar, crie dois diretórios `group_vars` e `host_vars` dois arquivos `inventory.yml` e `playbook.yml`.

As seções a seguir descrevem o conteúdo de cada um desses arquivos.

Planeie o sistema de ficheiros

Planeje a implantação do sistema de arquivos antes de desenvolver o inventário do Ansible.

Visão geral

Antes de implantar o sistema de arquivos, você deve definir quais endereços IP, portas e outras configurações serão exigidas por todos os nós de arquivo, nós de bloco e serviços BeeGFS executados no cluster. Embora a configuração exata varie com base na arquitetura do cluster, esta seção define as práticas recomendadas e as etapas a seguir que geralmente são aplicáveis.

Passos

1. Se você estiver usando um protocolo de storage baseado em IP (como iSER, iSCSI, NVMe/IB ou NVMe/RoCE) para conectar nós de arquivo a nós de bloco, preencha a Planilha a seguir para cada bloco básico. Cada conexão direta em um único bloco de construção deve ter uma sub-rede única, e não deve haver sobreposição com sub-redes usadas para conetividade cliente-servidor.

Nó de arquivo	Porta de IB	Endereço IP	Nó de bloco	Porta de IB	IP físico	IP virtual (apenas para EF600K com HDR IB)
<HOSTNAME>	<PORT>	<IP/SUBNET>	<HOSTNAME>	<PORT>	<IP/SUBNET>	<IP/SUBNET>

> ⓘ Se os nós de arquivo e bloco em cada bloco de construção estiverem diretamente conetados, você pode frequentemente reutilizar os mesmos IPs/esquema para vários blocos de construção.

2. Independentemente de você estar usando InfiniBand ou RDMA em Converged Ethernet (RoCE) para a rede de storage, preencha a seguinte Planilha para determinar os intervalos de IP que serão usados para serviços de cluster de HA, serviços de arquivos BeeGFS e clientes para se comunicar:

Finalidade	**Porta InfiniBand**	**Endereço IP ou intervalo**
IP(s) de cluster do BeeGFS	<INTERFACE(s)>	<RANGE>

Finalidade	Porta InfiniBand	Endereço IP ou intervalo
Gestão BeeGFS	<INTERFACE(s)>	<IP(s)>
Metadados BeeGFS	<INTERFACE(s)>	<RANGE>
Storage BeeGFS	<INTERFACE(s)>	<RANGE>
Clientes BeeGFS	<INTERFACE(s)>	<RANGE>

a. Se você estiver usando uma única sub-rede IP, apenas uma Planilha será necessária, caso contrário, também preencha uma Planilha para a segunda sub-rede.

3. Com base no exposto, para cada componente básico no cluster, preencha a seguinte Planilha que define quais serviços BeeGFS ele será executado. Para cada serviço, especifique o(s) nó(s) de arquivo preferencial/secundário(s), a porta de rede, IP(s) flutuante(s), a atribuição de zona NUMA (se necessário) e que nó(s) de bloco serão usados para seus destinos. Consulte as seguintes diretrizes ao preencher a Planilha:

   a. Especifique os serviços BeeGFS como `mgmt.yml` , `meta_<ID>.yml` `storage_<ID>.yml` ou onde o ID representa um número exclusivo em todos os serviços BeeGFS desse tipo neste sistema de arquivos. Esta convenção simplificará a referência a esta Planilha em seções subsequentes ao criar arquivos para configurar cada serviço.

   b. As portas para serviços BeeGFS precisam ser exclusivas em um componente básico específico. Certifique-se de que os serviços com o mesmo número de porta nunca possam ser executados no mesmo nó de arquivo para evitar conflitos de porta.

   c. Se necessário, os serviços podem usar volumes de mais de um nó de bloco e/ou pool de storage (e nem todos os volumes precisam pertencer à mesma controladora). Vários serviços também podem compartilhar o mesmo nó de bloco e/ou configuração de pool de storage (volumes individuais serão definidos em uma seção posterior).

Serviço BeeGFS (nome do arquivo)	Nós de arquivo	Porta	IPs flutuantes	Zona NUMA	Nó de bloco	Pool de storage	Controlador proprietário
<SERVICE TYPE>_<ID>.yml	NÓ(S) DE ARQUIVO SECUNDÁRIO(S)>	<PORT>	<INTERFACE>:<IP/SUBNET> <INTERFACE>:<IP/SUBNET>	<NUMA NODE/ZONE>	<BLOCK NODE>	POOL DE ARMAZENAMENTO/ GRUPO DE VOLUME>	A OU B>

Para obter mais detalhes sobre convenções padrão, práticas recomendadas e planilhas de exemplo preenchidas, consulte "práticas recomendadas"as seções e "Defina os componentes básicos do BeeGFS"do BeeGFS na arquitetura verificada do NetApp.

Definir nós de arquivo e bloco

Configurar nós de arquivo individuais

Especifique a configuração para nós de arquivo individuais usando variáveis de host (host_vars).

Visão geral

Esta seção aborda o preenchimento de um `host_vars/<FILE_NODE_HOSTNAME>.yml` arquivo para cada nó de arquivo no cluster. Esses arquivos só devem conter configuração exclusiva de um nó de arquivo específico. Isso geralmente inclui:

- Definindo o IP ou o nome do host que o Ansible deve usar para se conetar ao nó.
- Configuração de interfaces adicionais e IPs de cluster usados para serviços de cluster HA (Pacemaker e Corosync) para se comunicar com outros nós de arquivo. Por padrão, esses serviços usam a mesma rede que a interface de gerenciamento, mas interfaces adicionais devem estar disponíveis para redundância. A prática comum é definir IPs adicionais na rede de armazenamento, evitando a necessidade de um cluster adicional ou rede de gerenciamento.
  - O desempenho de qualquer rede usada para comunicação em cluster não é crítico para o desempenho do sistema de arquivos. Com a configuração padrão do cluster, geralmente, pelo menos uma rede de 1GB GB/s fornecerá desempenho suficiente para operações do cluster, como a sincronização dos estados dos nós e a coordenação das alterações do estado dos recursos do cluster. Redes lentas/ocupadas podem fazer com que as alterações de estado de recursos demorem mais tempo do que o normal, e em casos extremos podem resultar em nós sendo despejados do cluster se não puderem enviar batimentos cardíacos em um período de tempo razoável.
- Configuração de interfaces usadas para conexão a nós de bloco pelo protocolo desejado (por exemplo: ISCSI/iSER, NVMe/IB, NVMe/RoCE, FCP, etc.)

Passos

Fazendo referência ao esquema de endereçamento IP definido na "Planeie o sistema de ficheiros"seção, para cada nó de arquivo no cluster, crie um arquivo `host_vars/<FILE_NODE_HOSTNAME>/yml` e o preencha da seguinte forma:

1. Na parte superior, especifique o IP ou o nome de host que o Ansible deve usar para SSH para o nó e gerenciá-lo:

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

2. Configure IPs adicionais que podem ser usados para tráfego de cluster:
   a. Se o tipo de rede for "InfiniBand (usando IPoIB)":

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

   b. Se o tipo de rede for "RDMA em Ethernet convergente (RoCE)":

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

c. Se o tipo de rede for "Ethernet (apenas TCP, sem RDMA)":

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

3. Indique quais IPs devem ser usados para tráfego de cluster, com IPs preferenciais listados acima:

```
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.
```

Os IPS configurados na etapa dois não serão usados como IPs de cluster, a menos que sejam incluídos na `beegfs_ha_cluster_node_ips` lista. Isso permite configurar IPs/interfaces adicionais usando o Ansible que podem ser usados para outros fins, se desejado.

4. Se o nó de arquivo precisar se comunicar com nós de bloco por um protocolo baseado em IP, os IPs precisarão ser configurados na interface apropriada e quaisquer pacotes necessários para esse protocolo instalado/configurado.

a. Se estiver a utilizar "ISCSI":

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

b. Se estiver a utilizar "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. Se estiver a utilizar "NVMe/IB":

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

d. Se estiver a utilizar "NVMe/RoCE":

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

e. Outros protocolos:

   i. Se estiver usando "NVMe/FC"o , a configuração de interfaces individuais não é necessária. A implantação do cluster do BeeGFS detetará automaticamente o protocolo e instalará/configurará os requisitos conforme necessário. Se você estiver usando uma malha para conectar nós de arquivo e bloco, verifique se os switches estão adequadamente zoneados seguindo as práticas recomendadas do fornecedor de switch e do NetApp.

   ii. O uso de FCP ou SAS não requer a instalação ou configuração de software adicional. Se estiver usando FCP, verifique se os switches estão adequadamente zoneados seguindo "NetApp" e as práticas recomendadas do fornecedor de switch.

   iii. O uso do SRP IB não é recomendado neste momento. Use o NVMe/IB ou iSER dependendo do suporte dos nós de bloco do e-Series.

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um único nó de arquivo.

Avançado: Alternando os adaptadores VPI do NVIDIA ConnectX entre o modo Ethernet e InfiniBand

Os adaptadores NVIDIA ConnectX-Virtual Protocol Interconnect® (VPI) suportam InfiniBand e Ethernet como a camada de transporte. A troca entre modos não é negociada automaticamente e deve ser configurada usando a <code>mstconfig</code> ferramenta incluída no <code>mstflint</code>, um pacote de código aberto que faz parte "Ferramentas NVIDIA Firmare (MFT)"do . Alterar o modo dos adaptadores só precisa ser feito uma vez. Isso pode ser feito manualmente ou incluído no inventário do Ansible como parte de qualquer interface configurada usando a <code>eseries-[ib|ib_iser|ipoib|nvme_ib|nvme_roce|roce]_interfaces:</code> seção do inventário, para que ele seja verificado/aplicado automaticamente.

Por exemplo, para alterar uma interface atual no modo InfiniBand para Ethernet, para que possa ser usada no RoCE:

1. Para cada interface que você deseja configurar especifique `mstconfig` como um mapeamento (ou dicionário) que especifica `LINK_TYPE_P<N>` onde `<N>` é determinado pelo número de porta do HCA para a interface. O `<N>` valor pode ser determinado executando `grep PCI_SLOT_NAME /sys/class/net/<INTERFACE_NAME>/device/uevent` e adicionando 1 ao último número do nome do slot PCI e convertendo para decimal.
   a. Por exemplo, fornecido `PCI_SLOT_NAME=0000:2f:00.2` (2 e 1 → porta HCA 3) → `LINK_TYPE_P3: eth`:

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

Para obter mais detalhes, consulte a para obter informações "Documentação da coleção de hosts do NetApp e-Series"sobre o tipo/protocolo de interface que está a utilizar.

Configurar nós de bloco individual

Especifique a configuração para nós de bloco individuais usando variáveis de host (host_vars).

Visão geral

Esta seção aborda o preenchimento de um `host_vars/<BLOCK_NODE_HOSTNAME>.yml` arquivo para cada nó de bloco no cluster. Esses arquivos só devem conter configuração exclusiva de um nó de bloco específico. Isso geralmente inclui:

- O nome do sistema (conforme apresentado no System Manager).
- O URL HTTPS para um dos controladores (usado para gerenciar o sistema usando sua API REST).
- Quais nós de arquivo de protocolo de storage usam para se conectar a esse nó de bloco.
- Configuração de portas de placa de interface do host (HIC), como endereços IP (se necessário).

Passos

Fazendo referência ao esquema de endereçamento IP definido na "Planeie o sistema de ficheiros"seção, para cada nó de bloco no cluster, crie um arquivo `host_vars/<BLOCK_NODE_HOSTNAME>/yml` e o preencha da seguinte forma:

1. Na parte superior, especifique o nome do sistema e o URL HTTPS para um dos controladores:

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

2. Selecione os "protocolo"nós de arquivo que serão usados para se conetar a esse nó de bloco:

   a. Protocolos suportados: `auto` `iscsi` , , `fc`, `sas`, , , `ib_srp`, `ib_iser` `nvme_ib`, , `nvme_fc`, `nvme_roce`.

   ```
   eseries_initiator_protocol: <PROTOCOL>
   ```

3. Dependendo do protocolo em uso, as portas HIC podem exigir configuração adicional. Quando necessário, a configuração da porta HIC deve ser definida de modo que a entrada superior na configuração para cada controlador corresponda com a porta física mais à esquerda em cada controlador, e a porta inferior a porta mais à direita. Todas as portas requerem uma configuração válida mesmo que não estejam em uso no momento.

   Consulte também a seção abaixo se você estiver usando HDR (200GB) InfiniBand ou 200GB RoCE com EF600 nós de bloco.

   a. Para iSCSI:

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

# Alternatively the following common port configuration can be
defined for all ports and omitted above:
eseries_controller_iscsi_port_state: enabled         # Generally
specifies whether a controller port definition should be applied
Choices: enabled, disabled
eseries_controller_iscsi_port_config_method: dhcp    # General port
configuration method definition for both controllers. Choices:
static, dhcp
eseries_controller_iscsi_port_gateway:               # General port
IPv4 gateway for both controllers.
eseries_controller_iscsi_port_subnet_mask:           # General port
IPv4 subnet mask for both controllers.
eseries_controller_iscsi_port_mtu: 9000              # General port
maximum transfer units (MTU) for both controllers. Any value greater
than 1500 (bytes).
```

b. Para iSER:

```
eseries_controller_ib_iser_port:
  controller_a:    # Ordered list of controller A channel address
definition.
    -              # Port IPv4 address for channel 1
    - (...)        # So on and so forth
  controller_b:    # Ordered list of controller B channel address
definition.
```

c. Para NVMe/IB:

```
eseries_controller_nvme_ib_port:
  controller_a:    # Ordered list of controller A channel address
definition.
    -              # Port IPv4 address for channel 1
    - (...)        # So on and so forth
  controller_b:    # Ordered list of controller B channel address
definition.
```

d. Para NVMe/RoCE:

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

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

e. Os protocolos FC e SAS não exigem configuração adicional. SRP não é recomendado corretamente.

Para opções adicionais para configurar portas HIC e protocolos host, incluindo a capacidade de configurar CHAP iSCSI, consulte o "documentação" incluído com a coleção SANtricity. Observação ao implantar o BeeGFS, o pool de storage, a configuração de volume e outros aspectos do provisionamento de storage serão configurados em outro lugar e não deverão ser definidos nesse arquivo.

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um nó de bloco único.

Usando InfiniBand HDR (200GBK) ou RoCE de 200GB GB com nós de bloco NetApp EF600:

Para usar o InfiniBand HDR (200GB) com o EF600, um segundo IP "virtual" deve ser configurado para cada porta física. Abaixo está um exemplo da maneira correta de configurar um EF600 equipado com a porta dupla InfiniBand HDR HIC:

```
eseries_controller_nvme_ib_port:
  controller_a:
    - 192.168.1.101   # Port 2a (virtual)
    - 192.168.2.101   # Port 2b (virtual)
    - 192.168.1.100   # Port 2a (physical)
    - 192.168.2.100   # Port 2b (physical)
  controller_b:
    - 192.168.3.101   # Port 2a (virtual)
    - 192.168.4.101   # Port 2b (virtual)
    - 192.168.3.100   # Port 2a (physical)
    - 192.168.4.100   # Port 2b (physical)
```

Especifique a Configuração do nó de ficheiro Comum

Especifique a configuração de nó de arquivo comum usando variáveis de grupo (group_vars).

Visão geral

A configuração que deve ser Apple para todos os nós de arquivo é definida em `group_vars/ha_cluster.yml`. Geralmente inclui:

- Detalhes sobre como conetar e fazer login em cada nó de arquivo.
- Configuração de rede comum.
- Se as reinicializações automáticas são permitidas.
- Como o firewall e os estados selinux devem ser configurados.
- Configuração de cluster, incluindo alertas e cercas.
- Ajuste de desempenho.
- Configuração comum do serviço BeeGFS.

As opções definidas neste arquivo também podem ser definidas em nós de arquivo individuais, por exemplo, se modelos de hardware mistos estiverem em uso ou se você tiver senhas diferentes para cada nó. A configuração em nós de arquivo individuais terá precedência sobre a configuração neste arquivo.

Passos

Crie o arquivo `group_vars/ha_cluster.yml` e preencha-o da seguinte forma:

1. Indique como o nó Ansible Control deve se autenticar com os hosts remotos:

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

Particularmente para ambientes de produção, não armazene senhas em texto simples. Em vez disso, use o Ansible Vault ("Criptografia de conteúdo com o Ansible Vault"consulte) ou a `--ask-become-pass` opção ao executar o manual de estratégia. Se o `ansible_ssh_user` já for root, você pode omitir opcionalmente o `ansible_become_password`.

2. Se você estiver configurando IPs estáticos em interfaces ethernet ou InfiniBand (por exemplo, IPs de cluster) e várias interfaces estiverem na mesma sub-rede IP (por exemplo, se ib0 estiver usando 192.168.1.10/24 e IB1 estiver usando 192.168.1.11/24), tabelas e regras de roteamento IP adicionais devem ser configuradas para que o suporte multi-homed funcione corretamente. Basta ativar o gancho de configuração da interface de rede fornecido da seguinte forma:

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

3. Ao implantar o cluster, dependendo do protocolo de storage, pode ser necessário reiniciar os nós para facilitar a descoberta de dispositivos de bloco remoto (volumes e-Series) ou aplicar outros aspetos da configuração. Por padrão, os nós serão solicitados antes da reinicialização, mas você pode permitir que os nós sejam reiniciados automaticamente especificando o seguinte:

```
eseries_common_allow_host_reboot: true
```

   a. Por padrão, após uma reinicialização, para garantir que os dispositivos de bloco e outros serviços estejam prontos, o Ansible aguardará até que o systemd `default.target` seja alcançado antes de continuar com a implantação. Em alguns cenários em que o NVMe/IB está em uso, isso pode não ser longo o suficiente para inicializar, descobrir e se conetar a dispositivos remotos. Isto pode resultar na continuação prematura e falha da implementação automatizada. Para evitar isso ao usar o NVMe/IB, defina o seguinte:

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

4. Várias portas de firewall são necessárias para que os serviços do cluster BeeGFS e HA se comuniquem. A menos que você deseje configurar o firwewall manualmente (não recomendado), especifique o seguinte para que as zonas de firewall necessárias sejam criadas e as portas abertas automaticamente:

```
beegfs_ha_firewall_configure: True
```

5. Neste momento, o SELinux não é suportado, e é recomendável que o estado seja definido como desativado para evitar conflitos (especialmente quando o RDMA está em uso). Defina o seguinte para garantir que o SELinux esteja desativado:

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

6. Configure a autenticação para que os nós de arquivo possam se comunicar, ajustando os padrões conforme necessário com base nas políticas da sua organização:

```
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. Com base na "Planeie o sistema de ficheiros" seção, especifique o IP de gerenciamento do BeeGFS para este sistema de arquivos:

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

> Embora pareça redundante, `beegfs_ha_mgmtd_floating_ip` é importante quando você escala o sistema de arquivos BeeGFS além de um único cluster de HA. Os clusters de HA subsequentes são implantados sem um serviço de gerenciamento BeeGFS adicional e apontam para o serviço de gerenciamento fornecido pelo primeiro cluster.

8. Ative alertas de e-mail, se desejado:

```
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. A ativação do esgrima é fortemente recomendada, caso contrário, os serviços podem ser bloqueados de iniciar em nós secundários quando o nó primário falhar.
   a. Ative a vedação globalmente especificando o seguinte:

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

      i. Observação qualquer suporte "propriedade cluster" também pode ser especificado aqui, se necessário. Normalmente, não é necessário ajustá-los, uma vez que a função BeeGFS HA é fornecida com uma série de testes bem testados"predefinições".

   b. Em seguida, selecione e configure um agente de esgrima:
      i. Opção 1: Para ativar a vedação utilizando unidades de distribuição de energia (PDUs) da APC:

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

      ii. Opção 2: Para habilitar o esgrima usando as APIs do Redfish fornecidas pelo Lenovo XCC (e outros BMCs):

```
redfish: &redfish
  username: <BMC_USERNAME>
  password: <BMC_PASSWORD>
  ssl_insecure: 1 # If a valid SSL certificate is not available
specify "1".

beegfs_ha_fencing_agents:
  fence_redfish:
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
```

iii. Para obter detalhes sobre a configuração de outros agentes de vedação, consulte o "Documentação do Red Hat".

10. A função BeeGFS HA pode aplicar muitos parâmetros de ajuste diferentes para ajudar a otimizar ainda mais a performance. Estes incluem a otimização da utilização da memória do kernel e a e/S do dispositivo de bloco, entre outros parâmetros. A função é fornecida com um conjunto razoável de "predefinições" com base em testes com os nós de bloco do NetApp e-Series, mas por padrão estes não são aplicados a menos que você especifique:

```
beegfs_ha_enable_performance_tuning: True
```

a. Se necessário, especifique também quaisquer alterações ao ajuste de desempenho padrão aqui. Consulte a documentação completa "parâmetros de ajuste de desempenho"para obter detalhes adicionais.

11. Para garantir que os endereços IP flutuantes (às vezes conhecidos como interfaces lógicas) usados para serviços BeeGFS possam fazer failover entre nós de arquivo, todas as interfaces de rede devem ser nomeadas de forma consistente. Por padrão, os nomes de interface de rede são gerados pelo kernel, o que não é garantido para gerar nomes consistentes, mesmo em modelos de servidor idênticos com adaptadores de rede instalados nos mesmos slots PCIe. Isso também é útil ao criar inventários antes que o equipamento seja implantado e os nomes de interface gerados sejam conhecidos. Para garantir nomes de dispositivos consistentes, com base em um diagrama de blocos do servidor ou `lshw -class network -businfo` saída, especifique o mapeamento de interface lógica de endereço PCIe desejado da seguinte forma:

a. Para interfaces de rede InfiniBand (IPoIB):

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

b. Para interfaces de rede Ethernet:

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

Para evitar conflitos quando as interfaces são renomeadas (impedindo-as de serem renomeadas), você não deve usar nomes padrão potenciais como eth0, ens9f0, ib0 ou ibs4f0. Uma convenção de nomenclatura comum é usar 'e' ou 'i' para Ethernet ou InfiniBand, seguido do número do slot PCIe e uma letra para indicar a porta. Por exemplo, a segunda porta de um adaptador InfiniBand instalado no slot 3 seria: i3b.

Se você estiver usando um modelo de nó de arquivo verificado, clique "aqui" em exemplo mapeamentos de endereço PCIe para porta lógica.

12. Especifique opcionalmente a configuração que deve ser aplicada a todos os serviços BeeGFS no cluster. Os valores de configuração padrão podem ser "aqui"encontrados e a configuração por serviço é especificada em outro lugar:
    a. Serviço de gerenciamento BeeGFS:

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

    b. Serviços de metadados BeeGFS:

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

    c. Serviços BeeGFS Storage:

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

13. A partir do BeeGFS 7.2.7 e 7.3.1 "autenticação de conexão"devem ser configurados ou explicitamente desativados. Há algumas maneiras de configurar isso usando a implantação baseada no Ansible:
    a. Por padrão, a implantação configurará automaticamente a autenticação de conexão e gerará uma `connauthfile` que será distribuída para todos os nós de arquivos e usada com os serviços BeeGFS. Esse arquivo também será colocado/mantido no nó de controle do Ansible `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile` onde ele deve ser mantido (com segurança) para reutilização com clientes que precisam acessar esse sistema de arquivos.
        i. Para gerar uma nova chave, especifique `-e "beegfs_ha_conn_auth_force_new=True` ao executar o manual de estratégia do Ansible. Nota isto é ignorado se a `beegfs_ha_conn_auth_secret` estiver definida.
        ii. Para opções avançadas, consulte a lista completa de padrões incluídos no "BeeGFS HA função".
    b. Um segredo personalizado pode ser usado definindo o seguinte em `ha_cluster.yml`:

```
beegfs_ha_conn_auth_secret: <SECRET>
```

c. A autenticação de conexão pode ser totalmente desativada (NÃO recomendada):

```
beegfs_ha_conn_auth_enabled: false
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa a configuração comum do nó de arquivo.

Usando InfiniBand HDR (200GBK) com nós de bloco NetApp EF600:

Para usar o InfiniBand HDR (200GB) com o EF600, o gerenciador de sub-rede deve suportar a virtualização. Se os nós de arquivo e bloco estiverem conetados usando um switch, isso precisará ser ativado no gerenciador de sub-rede para a malha geral.

Se os nós de bloco e arquivo estiverem diretamente conetados usando InfiniBand, uma instância de `opensm` deve ser configurada em cada nó de arquivo para cada interface diretamente conetada a um nó de bloco. Isso é feito especificando `configure: true` quando "configurando interfaces de storage de nós de arquivo".

Atualmente, a versão da caixa de entrada `opensm` fornecida com distribuições Linux suportadas não suporta virtualização. Em vez disso, é necessário instalar e configurar a versão do `opensm` a partir do NVIDIA OpenFabrics Enterprise Distribution (OFED). Embora a implantação usando Ansible ainda seja compatível, algumas etapas adicionais são necessárias:

1. Usando curl ou a ferramenta desejada, baixe os pacotes para a versão do OpenSM listados na "requisitos de tecnologia"seção do site do NVIDIA para `<INVENTORY>/packages/` o diretório. Por exemplo:

```
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. Em `group_vars/ha_cluster.yml` Definir a seguinte configuração:

```
### 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"
```

Especifique Common Block Node Configuration

Especifique a configuração de nó de bloco comum usando variáveis de grupo (group_vars).

Visão geral

A configuração que deve ser Apple para todos os nós de bloco é definida em `group_vars/eseries_storage_systems.yml`. Geralmente inclui:

- Detalhes sobre como o nó de controle do Ansible deve se conectar aos sistemas de storage e-Series usados como nós de bloco.

- Quais versões de firmware, NVSRAM e firmware da unidade os nós devem ser executados.
- Configuração global, incluindo configurações de cache, configuração de host e configurações de como os volumes devem ser provisionados.

> As opções definidas neste arquivo também podem ser definidas em nós de bloco individuais, por exemplo, se modelos de hardware mistos estiverem em uso ou se você tiver senhas diferentes para cada nó. A configuração em nós de bloco individuais terá precedência sobre a configuração neste arquivo.

Passos

Crie o arquivo `group_vars/eseries_storage_systems.yml` e preencha-o da seguinte forma:

1. O Ansible não usa SSH para se conectar a nós de bloco e, em vez disso, usa APIs REST. Para conseguir isso, devemos definir:

```
ansible_connection: local
```

2. Especifique o nome de usuário e a senha para gerenciar cada nó. O nome de usuário pode ser omitido opcionalmente (e será padrão para admin), caso contrário, você pode especificar qualquer conta com admin Privileges. Especifique também se os certificados SSL devem ser verificados ou ignorados:

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

> Listar senhas em texto simples não é recomendado. Use o Ansible Vault ou forneça o `eseries_system_password` ao executar o Ansible usando --extra-vars.

3. Opcionalmente, especifique o firmware da controladora, NVSRAM e da unidade que devem ser instalados nos nós. Eles precisarão ser baixados para `packages/` o diretório antes de executar o Ansible. O firmware da controladora e-Series e a NVSRAM podem ser baixados "aqui" e o firmware da unidade "aqui":

```
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".
```

Se essa configuração for especificada, o Ansible atualizará automaticamente todo o firmware, incluindo a reinicialização de controladores (se necessário) sem prompts adicionais. Espera-se que isso não cause interrupções na e/S do BeeGFS/host, mas possa causar uma diminuição temporária na performance.

4. Ajuste os padrões globais de configuração do sistema. As opções e valores listados aqui são comumente recomendados para BeeGFS no NetApp, mas podem ser ajustados se necessário:

```
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. Configurar padrões globais de provisionamento de volume. As opções e valores listados aqui são comumente recomendados para BeeGFS no NetApp, mas podem ser ajustados se necessário:

```
eseries_volume_size_unit: pct # Required by default. This allows volume
capacities to be specified as a percentage, simplifying putting together
the inventory.
eseries_volume_read_cache_enable: true
eseries_volume_read_ahead_enable: false
eseries_volume_write_cache_enable: true
eseries_volume_write_cache_mirror_enable: true
eseries_volume_cache_without_batteries: false
```

6. Se necessário, ajuste a ordem pela qual o Ansible selecionará unidades para pools de storage e grupos de volumes tendo em mente as práticas recomendadas a seguir:
   a. Liste as unidades (potencialmente menores) que devem ser usadas primeiro para gerenciamento e/ou volumes de metadados e, por último, os volumes de storage.
   b. Certifique-se de equilibrar a ordem de seleção da unidade entre os canais de unidade disponíveis com base no(s) modelo(s) de compartimento de disco/compartimento de unidade. Por exemplo, com o EF600 e sem expansões, as unidades 0-11 estão no canal de unidade 1 e as unidades 12-23 estão no canal de unidade. Assim, uma estratégia para equilibrar a seleção da unidade é selecionar `disk shelf:drive` 99:0, 99:23, 99:1, 99:22, etc. caso haja mais de um compartimento, o primeiro dígito representa a ID do compartimento da unidade.

```
# 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"
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa a configuração de nó

de bloco comum.

Defina os serviços BeeGFS

Definir o serviço de gerenciamento BeeGFS

Os serviços BeeGFS são configurados usando variáveis de grupo (group_vars).

Visão geral

Esta seção descreve a definição do serviço de gerenciamento BeeGFS. Apenas um serviço desse tipo deve existir no(s) cluster(s) HA para um sistema de arquivos específico. A configuração deste serviço inclui a definição de:

- O tipo de serviço (gerenciamento).
- Definição de qualquer configuração que só se aplique a este serviço BeeGFS.
- Configurar um ou mais IPs flutuantes (interfaces lógicas) onde este serviço pode ser alcançado.
- Especificar onde/como um volume deve ser armazenado os dados para esse serviço (o destino de gerenciamento do BeeGFS).

Passos

Crie um novo arquivo `group_vars/mgmt.yml` e consulte a "Planeie o sistema de ficheiros" seção preencha-o da seguinte forma:

1. Indicar que esse arquivo representa a configuração de um serviço de gerenciamento BeeGFS:

```
beegfs_service: management
```

2. Defina qualquer configuração que se aplique somente a esse serviço BeeGFS. Normalmente, isso não é necessário para o serviço de gerenciamento, a menos que você precise ativar cotas, no entanto, qualquer parâmetro de configuração suportado do `beegfs-mgmtd.conf` pode ser incluído. Observação os seguintes parâmetros são configurados automaticamente/em outro lugar e não devem ser especificados aqui: `storeMgmtdDirectory` `connAuthFile` , , `connDisableAuthentication`, , `connInterfacesFile` E `connNetFilterFile`.

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

3. Configure um ou mais IPs flutuantes que outros serviços e clientes usarão para se conetar a esse serviço (isso definirá automaticamente a opção BeeGFS `connInterfacesFile`):

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

4. Opcionalmente, especifique uma ou mais sub-redes IP permitidas que podem ser usadas para comunicação de saída (isso definirá automaticamente a opção BeeGFS `connNetFilterFile`):

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

5. Especifique o destino de gerenciamento do BeeGFS no qual esse serviço armazenará dados de acordo com as seguintes diretrizes:
   a. O mesmo pool de storage ou nome de grupo de volumes pode ser usado para vários serviços/destinos do BeeGFS. Basta usar a mesma `name` `criteria_*` configuração , `raid_level` e e `common_*` para cada um (os volumes listados para cada serviço devem ser diferentes).
   b. Os tamanhos de volume devem ser especificados como uma porcentagem do pool de armazenamento/grupo de volumes e o total não deve exceder 100 em todos os serviços/volumes usando um pool de armazenamento/grupo de volumes específico. Observação ao usar SSDs, é recomendável deixar algum espaço livre no grupo de volumes para maximizar o desempenho do SSD e a vida útil do desgaste (clique "aqui" para obter mais detalhes).
   c. Clique "aqui" em para obter uma lista completa das opções de configuração disponíveis para o `eseries_storage_pool_configuration`. Observação algumas opções, como `state` , `host`, , `host_type`, `workload_name` `workload_metadata` e e nomes de volume são geradas automaticamente e não devem ser especificadas aqui.

```
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
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um serviço de gerenciamento BeeGFS.

Defina o serviço de metadados do BeeGFS

Os serviços BeeGFS são configurados usando variáveis de grupo (group_vars).

Visão geral

Esta seção descreve a definição do serviço de metadados do BeeGFS. Pelo menos um serviço desse tipo deve existir no(s) cluster(s) HA para um sistema de arquivos específico. A configuração deste serviço inclui a

definição de:

- O tipo de serviço (metadados).
- Definição de qualquer configuração que só se aplique a este serviço BeeGFS.
- Configurar um ou mais IPs flutuantes (interfaces lógicas) onde este serviço pode ser alcançado.
- Especificar onde/como um volume deve ser armazenado os dados para esse serviço (o destino de metadados do BeeGFS).

Passos

Fazendo referência à "Planeie o sistema de ficheiros" seção, crie um arquivo em `group_vars/meta_<ID>.yml` para cada serviço de metadados no cluster e preencha-os da seguinte forma:

1. Indicar que esse arquivo representa a configuração de um serviço de metadados do BeeGFS:

```
beegfs_service: metadata
```

2. Defina qualquer configuração que se aplique somente a esse serviço BeeGFS. No mínimo, você deve especificar a porta TCP e UDP desejada, no entanto, qualquer parâmetro de configuração suportado `beegfs-meta.conf` também pode ser incluído. Observação os seguintes parâmetros são configurados automaticamente/em outro lugar e não devem ser especificados aqui: `sysMgmtdHost` `storeMetaDirectory` , , `connAuthFile`, `connDisableAuthentication`, , `connInterfacesFile` E `connNetFilterFile`.

```
beegfs_ha_beegfs_meta_conf_resource_group_options:
  connMetaPortTCP: <TCP PORT>
  connMetaPortUDP: <UDP PORT>
  tuneBindToNumaZone: <NUMA ZONE> # Recommended if using file nodes with
multiple CPU sockets.
```

3. Configure um ou mais IPs flutuantes que outros serviços e clientes usarão para se conetar a esse serviço (isso definirá automaticamente a opção BeeGFS `connInterfacesFile`):

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

4. Opcionalmente, especifique uma ou mais sub-redes IP permitidas que podem ser usadas para comunicação de saída (isso definirá automaticamente a opção BeeGFS `connNetFilterFile`):

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

5. Especifique o destino de metadados do BeeGFS no qual esse serviço armazenará dados de acordo com as diretrizes a seguir (isso também configurará a opção automaticamente `storeMetaDirectory`):
   a. O mesmo pool de storage ou nome de grupo de volumes pode ser usado para vários serviços/destinos do BeeGFS. Basta usar a mesma `name` `criteria_*` configuração , `raid_level` e e `common_*` para cada um (os volumes listados para cada serviço devem ser diferentes).
   b. Os tamanhos de volume devem ser especificados como uma porcentagem do pool de armazenamento/grupo de volumes e o total não deve exceder 100 em todos os serviços/volumes usando um pool de armazenamento/grupo de volumes específico. Observação ao usar SSDs, é recomendável deixar algum espaço livre no grupo de volumes para maximizar o desempenho do SSD e a vida útil do desgaste (clique "aqui" para obter mais detalhes).
   c. Clique "aqui" em para obter uma lista completa das opções de configuração disponíveis para o `eseries_storage_pool_configuration`. Observação algumas opções, como `state` , `host`, , `host_type`, `workload_name` `workload_metadata` e e nomes de volume são geradas automaticamente e não devem ser especificadas aqui.

```
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
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um serviço de metadados BeeGFS.

Defina o serviço de storage BeeGFS

Os serviços BeeGFS são configurados usando variáveis de grupo (group_vars).

Visão geral

Esta seção descreve a definição do serviço de storage BeeGFS. Pelo menos um serviço desse tipo deve existir no(s) cluster(s) HA para um sistema de arquivos específico. A configuração deste serviço inclui a definição de:

- O tipo de serviço (armazenamento).
- Definição de qualquer configuração que só se aplique a este serviço BeeGFS.
- Configurar um ou mais IPs flutuantes (interfaces lógicas) onde este serviço pode ser alcançado.
- Especificar onde/como os volumes devem ser armazenados para esse serviço (os destinos de storage do BeeGFS).

Passos

Fazendo referência à "Planeie o sistema de ficheiros" seção, crie um arquivo em `group_vars/stor_<ID>.yml` para cada serviço de armazenamento no cluster e preencha-os da seguinte forma:

1. Indicar que esse arquivo representa a configuração de um serviço de storage BeeGFS:

```
beegfs_service: storage
```

2. Defina qualquer configuração que se aplique somente a esse serviço BeeGFS. No mínimo, você deve especificar a porta TCP e UDP desejada, no entanto, qualquer parâmetro de configuração suportado `beegfs-storage.conf` também pode ser incluído. Observação os seguintes parâmetros são configurados automaticamente/em outro lugar e não devem ser especificados aqui: `sysMgmtdHost` `storeStorageDirectory` , , `connAuthFile`, `connDisableAuthentication`, , `connInterfacesFile` E `connNetFilterFile`.

```
beegfs_ha_beegfs_storage_conf_resource_group_options:
  connStoragePortTCP: <TCP PORT>
  connStoragePortUDP: <UDP PORT>
  tuneBindToNumaZone: <NUMA ZONE> # Recommended if using file nodes with
multiple CPU sockets.
```

3. Configure um ou mais IPs flutuantes que outros serviços e clientes usarão para se conetar a esse serviço (isso definirá automaticamente a opção BeeGFS `connInterfacesFile`):

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

4. Opcionalmente, especifique uma ou mais sub-redes IP permitidas que podem ser usadas para comunicação de saída (isso definirá automaticamente a opção BeeGFS `connNetFilterFile`):

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

5. Especifique o(s) destino(s) de storage BeeGFS em que esse serviço armazenará dados de acordo com as diretrizes a seguir (isso também configurará automaticamente a `storeStorageDirectory` opção):
   a. O mesmo pool de storage ou nome de grupo de volumes pode ser usado para vários serviços/destinos do BeeGFS. Basta usar a mesma `name` `criteria_*` configuração , `raid_level` e e `common_*` para cada um (os volumes listados para cada serviço devem ser diferentes).
   b. Os tamanhos de volume devem ser especificados como uma porcentagem do pool de armazenamento/grupo de volumes e o total não deve exceder 100 em todos os serviços/volumes usando um pool de armazenamento/grupo de volumes específico. Observação ao usar SSDs, é

recomendável deixar algum espaço livre no grupo de volumes para maximizar o desempenho do SSD e a vida útil do desgaste (clique "aqui" para obter mais detalhes).

c. Clique "aqui" em para obter uma lista completa das opções de configuração disponíveis para o `eseries_storage_pool_configuration`. Observação algumas opções, como `state` , `host`, , `host_type`, `workload_name` `workload_metadata` e e nomes de volume são geradas automaticamente e não devem ser especificadas aqui.

```
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
```

Clique "aqui" para ver um exemplo de um arquivo de inventário completo que representa um serviço de storage BeeGFS.

Mapear os serviços BeeGFS para nós de arquivo

Especifique quais nós de arquivo podem executar cada serviço BeeGFS usando o `inventory.yml` arquivo.

Visão geral

Esta seção descreve como criar o `inventory.yml` arquivo. Isso inclui listar todos os nós de bloco e especificar quais nós de arquivo podem executar cada serviço BeeGFS.

Passos

Crie o arquivo `inventory.yml` e preencha-o da seguinte forma:

1. Na parte superior do arquivo, crie a estrutura de inventário padrão do Ansible:

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

2. Crie um grupo que contenha todos os nós de bloco participantes deste cluster HA:

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

3. Crie um grupo que contenha todos os serviços BeeGFS no cluster e os nós de arquivo que os executarão:

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

4. Para cada serviço BeeGFS no cluster, defina o(s) nó(s) de arquivo preferencial e secundário(s) que deve(m) executar esse serviço:

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

Clique "aqui" para ver um exemplo de um arquivo de inventário completo.

Informações sobre direitos autorais

Informações sobre marcas comerciais