Deploy software

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Table of Contents

Deploy software

Set up file nodes and block nodes

While most software configuration tasks are automated using the NetApp-provided Ansible collections, you must configure networking on the baseboard management controller (BMC) of each server and configure the management port on each controller.

Set up file nodes

1. Configure networking on the baseboard management controller (BMC) of each server.

   To learn how to configure networking for the validated Lenovo SR665 V3 file nodes, see the Lenovo ThinkSystem Documentation.

   > ⓘ   A baseboard management controller (BMC), sometimes referred to as a service processor, is the generic name for the out-of-band management capability built into various server platforms that can provide remote access even if the operating system is not installed or accessible. Vendors typically market this functionality with their own branding. For example, on the Lenovo SR665, the BMC is referred to as the *Lenovo XClarity Controller (XCC)*.

2. Configure the system settings for maximum performance.

   You configure the system settings using the UEFI setup (formerly known as the BIOS) or by using the Redfish APIs provided by many BMCs. The system settings vary based on the server model used as a file node.

   To learn how to configure the system settings for the validated Lenovo SR665 V3 file nodes, see Tune system settings for performance.

3. Install Red Hat Enterprise Linux (RHEL) 9.4 and configure the host name and network port used to manage the operating system including SSH connectivity from the Ansible control node.

   Do not configure IPs on any of the InfiniBand ports at this time.

   > ⓘ   While not strictly required, subsequent sections presume that host names are sequentially numbered (such as h1-hN) and refer to tasks that should be completed on odd versus even numbered hosts.

4. Use Red Hat Subscription Manager to register and subscribe the system to allow installation of the required packages from the official Red Hat repositories and to limit updates to the supported version of Red Hat: `subscription-manager release --set=9.4`. For instructions, see How to register and subscribe a RHEL system and How to limit updates.

5. Enable the Red Hat repository containing the packages required for high availability.

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

6. Update all HCA firmware to the version recommended in Technology requirements using the Update file

node adapter firmware guide.

Set up block nodes

Set up the EF600 block nodes by configuring the management port on each controller.

1. Configure the management port on each EF600 controller.

   For instructions on configuring ports, go to the E-Series Documentation Center.

2. Optionally, set the storage array name for each system.

   Setting a name can make it easier to refer to each system in subsequent sections. For instructions on setting the array name, go to the E-Series Documentation Center.

   ⓘ   While not strictly required, subsequent topics presume storage array names are sequentially numbered (such as c1 - cN) and refer to the steps that should be completed on odd versus even numbered systems.

Tune file node system settings for performance

To maximize performance, we recommend configuring the system settings on the server model you use as your file nodes.

The system settings vary depending on the server model you use as your file node. This topic describes how to configure the system settings for the validated Lenovo ThinkSystem SR665 server file nodes.

Use the UEFI interface to tune system settings

The Lenovo SR665 V3 server's system firmware contains numerous tuning parameters that can be set through the UEFI interface. These tuning parameters can affect all aspects of how the server functions and how well the server performs.

Under **UEFI Setup > System Settings**, adjust the following system settings:

Operating Mode menu

System Setting	Change to
Operating Mode	Custom
cTDP	Manual
cTDP Manual	350
Package Power Limit	Manual
Efficiency Mode	Disable

System Setting	Change to
Global-Cstate-Control	Disable
SOC P-states	P0
DF C-States	Disable
P-State	Disable
Memory Power Down Enable	Disable
NUMA Nodes per Socket	NPS1

Devices and I/O ports menu

System Setting	Change to
IOMMU	Disable

Power menu

System Setting	Change to
PCIe Power Brake	Disable

Processors menu

System Setting	Change to
Global C-state Control	Disable
DF C-States	Disable
SMT Mode	Disable
CPPC	Disable

Use Redfish API to tune system settings

In addition to using UEFI Setup, you can use the Redfish API to change system settings.

Example

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

For detailed information on the Redfish schema, see the DMTF website.

Set up an Ansible control node

To set up an Ansible control node, you must designate a virtual or physical machine with network access to all file and block nodes deployed for the BeeGFS on NetApp solution.

Review the Technical requirements for a list of recommended package versions. The following steps were tested on Ubuntu 22.04. For steps specific to your preferred Linux distribution, see the Ansible documentation.

1. From your Ansible control node, install the following Python and Python Virtual Environment packages.

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

2. Create a Python virtual environment.

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

3. Activate the virtual environment.

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

4. Install the required Python packages within the activated virtual environment.

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

5. Install the BeeGFS collection using Ansible Galaxy.

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

6. Verify the installed versions of Ansible, Python, and the BeeGFS collection match the Technical requirements.

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

7. Set up passwordless SSH to allow Ansible to access the remote BeeGFS file nodes from the Ansible control node.
   a. On the Ansible control node, if needed, generate a pair of public keys.

```
ssh-keygen
```

   b. Set up passwordless SSH to each of the file nodes.

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

> ⚠️ Do **not** set up passwordless SSH to the block nodes. This is neither supported nor required.

Create the Ansible inventory

To define the configuration for file and block nodes, you create an Ansible inventory that represents the BeeGFS file system you want to deploy. The inventory includes hosts, groups, and variables describing the desired BeeGFS file system.

Step 1: Define configuration for all building blocks

Define the configuration that applies to all building blocks, regardless of which configuration profile you may apply to them individually.

Before you begin

- Choose a subnet addressing scheme for your deployment. Due to the benefits listed in the software architecture, it is recommended to use a single subnet addressing scheme.

Steps

1. On your Ansible control node, identify a directory that you want to use to store the Ansible inventory and playbook files.

   Unless otherwise noted, all files and directories created in this step and following steps are created relative to this directory.

2. Create the following subdirectories:

   `host_vars`

   `group_vars`

   `packages`

3. Create a subdirectory for cluster passwords and secure the file by encrypting it with Ansible Vault (see Encrypting content with Ansible Vault):

   a. Create the subdirectory `group_vars/all.`

   b. In the `group_vars/all` directory, create a passwords file labeled `passwords.yml.`

   c. Populate the `passwords.yml file` with the following, replacing all username and password parameters according to your configuration:

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

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

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

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

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

    d. Run `ansible-vault encrypt passwords.yml` and set a vault password when prompted.

Step 2: Define configuration for individual file and block nodes

Define the configuration that applies to individual file nodes and individual building block nodes.

1. Under `host_vars/`, create a file for each BeeGFS file node named `<HOSTNAME>.yml` with the following content, paying special attention to the notes regarding content to populate for BeeGFS cluster IPs and host names ending in odd versus even numbers.

   Initially, the file node interface names do match what is listed here (such as ib0 or ibs1f0). These custom names are configured in Step 4: Define configuration that should apply to all file nodes.

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

> ⓘ If you have already deployed the BeeGFS cluster, you must stop the cluster before adding or changing statically configured IP addresses, including cluster IPs and IPs used for NVMe/IB. This is required so these changes take effect properly and do not disrupt cluster operations.

2. Under `host_vars/`, create a file for each BeeGFS block node named `<HOSTNAME>.yml` and populate it with the following content.

   Pay special attention to the notes regarding content to populate for storage array names ending in odd versus even numbers.

   For each block node, create one file and specify the `<MANAGEMENT_IP>` for one of the two controllers (usually A).

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

Step 3: Define configuration that should apply to all file and block nodes

You can define configuration common to a group of hosts under `group_vars` in a file name that corresponds with the group. This prevents repeating a shared configuration in multiple places.

About this task

Hosts can be in more than one group, and at runtime, Ansible chooses what variables apply to a particular host based on its variable precedence rules. (For more information on these rules, see the Ansible documentation for Using variables.)

Host-to-group assignments are defined in the actual Ansible inventory file, which is created towards the end of this procedure.

Step

In Ansible, any configuration you want to apply to all hosts can be defined in a group called `All`. Create the file `group_vars/all.yml` with the following content:

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

Step 4: Define configuration that should apply to all file nodes

The shared configuration for file nodes is defined in a group called `ha_cluster`. The steps in this section build out the configuration that should be included in the `group_vars/ha_cluster.yml` file.

Steps

1. At the top of the file, define the defaults, including the password to use as the `sudo` user on the file nodes.

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

> (i) If the `ansible_ssh_user` is already `root`, then you can optionally omit the `ansible_become_password` and specify the `--ask-become-pass` option when running the playbook.

2. Optionally, configure a name for the high-availability (HA) cluster and specify a user for intra-cluster communication.

   If you are modifying the private IP addressing scheme, you must also update the default `beegfs_ha_mgmtd_floating_ip`. This must match what you configure later for the BeeGFS Management resource group.

   Specify one or more emails that should receive alerts for cluster events using `beegfs_ha_alert_email_list`.

```
### Cluster information
beegfs_ha_firewall_configure: True
eseries_beegfs_ha_disable_selinux: True
eseries_selinux_state: disabled
# The following variables should be adjusted depending on the desired
configuration:
beegfs_ha_cluster_name: hacluster                      # BeeGFS HA cluster
name.
beegfs_ha_cluster_username: "{{ ha_cluster_username }}" # Parameter for
BeeGFS HA cluster username in the passwords file.
beegfs_ha_cluster_password: "{{ ha_cluster_password }}" # Parameter for
BeeGFS HA cluster username's password in the passwords file.
beegfs_ha_cluster_password_sha512_salt: "{{
ha_cluster_password_sha512_salt }}" # Parameter for BeeGFS HA cluster
username's password salt in the passwords file.
beegfs_ha_mgmtd_floating_ip: 100.127.101.0             # BeeGFS management
service IP address.
# Email Alerts Configuration
beegfs_ha_enable_alerts: True
beegfs_ha_alert_email_list: ["email@example.com"]  # E-mail recipient
list for notifications when BeeGFS HA resources change or fail.  Often a
distribution list for the team responsible for managing the cluster.
beegfs_ha_alert_conf_ha_group_options:
     mydomain: "example.com"
# The mydomain parameter specifies the local internet domain name. This
is optional when the cluster nodes have fully qualified hostnames (i.e.
host.example.com).
# Adjusting the following parameters is optional:
beegfs_ha_alert_timestamp_format: "%Y-%m-%d %H:%M:%S.%N" #%H:%M:%S.%N
beegfs_ha_alert_verbosity: 3
#  1) high-level node activity
#  3) high-level node activity + fencing action information + resources
(filter on X-monitor)
#  5) high-level node activity + fencing action information + resources
```

> ⓘ While seemingly redundant, `beegfs_ha_mgmtd_floating_ip` is important when you scale the BeeGFS file system beyond a single HA cluster. Subsequent HA clusters are deployed without an additional BeeGFS management service and point at the management service provided by the first cluster.

3. Configure a fencing agent. (For more details, see Configure fencing in a Red Hat High Availability cluster.) The following output shows examples for configuring common fencing agents. Choose one of these options.

   For this step, be aware that:

- By default, fencing is enabled, but you need to configure a fencing *agent*.

- The `<HOSTNAME>` specified in the `pcmk_host_map` or `pcmk_host_list` must correspond with the hostname in the Ansible inventory.

- Running the BeeGFS cluster without fencing is not supported, particularly in production. This is largely to ensure when BeeGFS services, including any resource dependencies like block devices, fail over due to an issue, there is no risk of concurrent access by multiple nodes that result in file system corruption or other undesirable or unexpected behavior. If fencing must be disabled, refer to the general notes in the BeeGFS HA role's getting started guide and set `beegfs_ha_cluster_crm_config_options["stonith-enabled"]` to false in `ha_cluster.yml`.

- There are multiple node-level fencing devices available, and the BeeGFS HA role can configure any fencing agent available in the Red Hat HA package repository. When possible, use a fencing agent that works through the uninterruptible power supply (UPS) or rack power distribution unit (rPDU), because some fencing agents such as the baseboard management controller (BMC) or other lights-out devices that are built into the server might not respond to the fence request under certain failure scenarios.

```
### Fencing configuration:
# OPTION 1: To enable fencing using APC Power Distribution Units
(PDUs):
beegfs_ha_fencing_agents:
 fence_apc:
   - ipaddr: <PDU_IP_ADDRESS>
     login: "{{ apc_username }}" # Parameter for APC PDU username in
the passwords file.
     passwd: "{{ apc_password }}" # Parameter for APC PDU password in
the passwords file.
     pcmk_host_map:
"<HOSTNAME>:<PDU_PORT>,<PDU_PORT>;<HOSTNAME>:<PDU_PORT>,<PDU_PORT>"
# OPTION 2: To enable fencing using the Redfish APIs provided by the
Lenovo XCC (and other BMCs):
redfish: &redfish
  username: "{{ bmc_username }}" # Parameter for XCC/BMC username in
the passwords file.
  password: "{{ bmc_password }}" # Parameter for XCC/BMC password in
the passwords file.
    ssl_insecure: 1 # If a valid SSL certificate is not available
specify "1".
beegfs_ha_fencing_agents:
  fence_redfish:
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
    - pcmk_host_list: <HOSTNAME>
      ip: <BMC_IP>
      <<: *redfish
# For details on configuring other fencing agents see
https://access.redhat.com/documentation/en-
us/red_hat_enterprise_linux/9/html/configuring_and_managing_high_avai
lability_clusters/assembly_configuring-fencing-configuring-and-
managing-high-availability-clusters.
```

4. Enable recommended performance tuning in the Linux OS.

   While many users find the default settings for the performance parameters generally work well, you can optionally change the default settings for a particular workload. As such, these recommendations are included in the BeeGFS role, but are not enabled by default to ensure users are aware of the tuning applied to their file system.

   To enable performance tuning, specify:

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

5. (Optional) You can adjust the performance tuning parameters in the Linux OS as needed.

   For a comprehensive list of the available tuning parameters that you can adjust, see the Performance Tuning Defaults section of the BeeGFS HA role in E-Series BeeGFS GitHub site. The default values can be overridden for all nodes in the cluster in this file or the `host_vars` file for an individual node.

6. To allow full 200Gb/HDR connectivity between block and file nodes, use the Open Subnet Manager (OpenSM) package from the NVIDIA Open Fabrics Enterprise Distribution (MLNX_OFED). The MLNX_OFED version in listed the file node requirements comes bundled with the recommended OpenSM packages. Although deployment using Ansible is supported, you must first install the MLNX_OFED driver on all file nodes.

   a. Populate the following parameters in `group_vars/ha_cluster.yml` (adjust packages as needed):

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

7. Configure the `udev` rule to ensure consistent mapping of logical InfiniBand port identifiers to underlying PCIe devices.

   The `udev` rule must be unique to the PCIe topology of each server platform used as a BeeGFS file node.

   Use the following values for verified file nodes:

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

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

8. (Optional) Update the metadata target selection algorithm.

```
beegfs_ha_beegfs_meta_conf_ha_group_options:
  tuneTargetChooser: randomrobin
```

> ⓘ In verification testing, `randomrobin` was typically used to ensure that test files were evenly distributed across all BeeGFS storage targets during performance benchmarking (for more information on benchmarking, see the BeeGFS site for Benchmarking a BeeGFS System). With real world use, this might cause lower numbered targets to fill up faster than higher numbered targets. Omitting `randomrobin` and just using the default `randomized` value has been shown to provide good performance while still utilizing all available targets.

Step 5: Define the configuration for the common block node

The shared configuration for block nodes is defined in a group called `eseries_storage_systems`. The steps in this section build out the configuration that should be included in the `group_vars/eseries_storage_systems.yml` file.

Steps

1. Set the Ansible connection to local, provide the system password, and specify if SSL certificates should be verified. (Normally, Ansible uses SSH to connect to managed hosts, but in the case of the NetApp E-Series storage systems used as block nodes, the modules use the REST API for communication.) At the top of

the file, add the following:

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

2. To ensure optimal performance, install the versions listed for block nodes in Technical requirements.

   Download the corresponding files from the NetApp Support site. You can either upgrade them manually or include them in the `packages/` directory of the Ansible control node, and then populate the following parameters in `eseries_storage_systems.yml` to upgrade using Ansible:

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

3. Download and install the latest drive firmware available for the drives installed in your block nodes from the NetApp Support site. You can either upgrade them manually or include them in the `packages/` directory of the Ansible control node, and then populate the following parameters in `eseries_storage_systems.yml` to upgrade using Ansible:

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

> (i) Setting `eseries_drive_firmware_upgrade_drives_online` to `false` will speed up the upgrade, but should not be done until after BeeGFS is deployed. This is because that setting requires stopping all I/O to the drives before the upgrade to avoid application errors. Although performing an online drive firmware upgrade before configuring volumes is still quick, we recommend you always set this value to `true` to avoid issues later.

4. To optimize performance, make the following changes to the global configuration:

```
# Global Configuration Defaults
eseries_system_cache_block_size: 32768
eseries_system_cache_flush_threshold: 80
eseries_system_default_host_type: linux dm-mp
eseries_system_autoload_balance: disabled
eseries_system_host_connectivity_reporting: disabled
eseries_system_controller_shelf_id: 99 # Required.
```

5. To ensure optimal volume provisioning and behavior, specify the following parameters:

```
# Storage Provisioning Defaults
eseries_volume_size_unit: pct
eseries_volume_read_cache_enable: true
eseries_volume_read_ahead_enable: false
eseries_volume_write_cache_enable: true
eseries_volume_write_cache_mirror_enable: true
eseries_volume_cache_without_batteries: false
eseries_storage_pool_usable_drives:
"99:0,99:23,99:1,99:22,99:2,99:21,99:3,99:20,99:4,99:19,99:5,99:18,99:6,
99:17,99:7,99:16,99:8,99:15,99:9,99:14,99:10,99:13,99:11,99:12"
```

> (i) The value specified for `eseries_storage_pool_usable_drives` is specific to NetApp
> EF600 block nodes and controls the order in which drives are assigned to new volume
> groups. This ordering ensures that the I/O to each group is evenly distributed across
> backend drive channels.

Define Ansible inventory for BeeGFS building blocks

After defining the general Ansible inventory structure, define the configuration for each
building block in the BeeGFS file system.

These deployment instructions demonstrate how to deploy a file system that consists of a base building block
including management, metadata, and storage services; a second building block with metadata and storage
services; and a third storage-only building block.

These steps are intended to show the full range of typical configuration profiles that you can use to configure
NetApp BeeGFS building blocks to meet the requirements of the overall BeeGFS file system.

> (i) In this and subsequent sections, adjust as needed to build the inventory representing the
> BeeGFS file system that you want to deploy. In particular, use Ansible host names that
> represent each block or file node and the desired IP addressing scheme for the storage network
> to ensure it can scale to the number of BeeGFS file nodes and clients.

Step 1: Create the Ansible inventory file

Steps

1. Create a new `inventory.yml` file, and then insert the following parameters, replacing the hosts under
   `eseries_storage_systems` as needed to represent the block nodes in your deployment. The names
   should correspond with the name used for `host_vars/<FILENAME>.yml`.

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

In the subsequent sections, you will create additional Ansible groups under `ha_cluster` that represent the BeeGFS services you want to run in the cluster.

Step 2: Configure the inventory for a management, metadata, and storage building block

The first building block in the cluster or base building block must include the BeeGFS management service along with metadata and storage services:

Steps

1. In `inventory.yml`, populate the following parameters under `ha_cluster: children:`

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

```
        stor_02:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_03:
          hosts:
            beegfs_01:
            beegfs_02:
        stor_03:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_04:
          hosts:
            beegfs_01:
            beegfs_02:
        stor_04:
          hosts:
            beegfs_01:
            beegfs_02:
        meta_05:
          hosts:
            beegfs_02:
            beegfs_01:
        stor_05:
          hosts:
            beegfs_02:
            beegfs_01:
        meta_06:
          hosts:
            beegfs_02:
            beegfs_01:
        stor_06:
          hosts:
            beegfs_02:
            beegfs_01:
        meta_07:
          hosts:
            beegfs_02:
            beegfs_01:
        stor_07:
          hosts:
            beegfs_02:
            beegfs_01:
        meta_08:
          hosts:
```

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

2. Create the file `group_vars/mgmt.yml` and include the following:

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

3. Under `group_vars/`, create files for resource groups `meta_01` through `meta_08` using the following template, and then fill in the placeholder values for each service referencing the table below:

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

> (i) The volume size is specified as a percentage of the overall storage pool (also referred to as a volume group). NetApp highly recommends that you leave some free capacity in each pool to allow room for SSD overprovisioning (for more information, see Introduction to NetApp EF600 array). The storage pool, `beegfs_m1_m2_m5_m6`, also allocates 1% of the pool's capacity for the management service. Thus, for metadata volumes in the storage pool, `beegfs_m1_m2_m5_m6`, when 1.92TB or 3.84TB drives are used, set this value to `21.25`; for 7.65TB drives, set this value to `22.25`; and for 15.3TB drives, set this value to `23.75`.

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
meta_01.yml	8015	i1b:100.127.1 01.1/16 i2b:100.127.1 02.1/16	0	netapp_01	beegfs_m1_ m2_m5_m6	A
meta_02.yml	8025	i2b:100.127.1 02.2/16 i1b:100.127.1 01.2/16	0	netapp_01	beegfs_m1_ m2_m5_m6	B
meta_03.yml	8035	i3b:100.127.1 01.3/16 i4b:100.127.1 02.3/16	1	netapp_02	beegfs_m3_ m4_m7_m8	A

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
meta_04.yml	8045	i4b:100.127.1 02.4/16 i3b:100.127.1 01.4/16	1	netapp_02	beegfs_m3_ m4_m7_m8	B
meta_05.yml	8055	i1b:100.127.1 01.5/16 i2b:100.127.1 02.5/16	0	netapp_01	beegfs_m1_ m2_m5_m6	A
meta_06.yml	8065	i2b:100.127.1 02.6/16 i1b:100.127.1 01.6/16	0	netapp_01	beegfs_m1_ m2_m5_m6	B
meta_07.yml	8075	i3b:100.127.1 01.7/16 i4b:100.127.1 02.7/16	1	netapp_02	beegfs_m3_ m4_m7_m8	A
meta_08.yml	8085	i4b:100.127.1 02.8/16 i3b:100.127.1 01.8/16	1	netapp_02	beegfs_m3_ m4_m7_m8	B

4. Under `group_vars/`, create files for resource groups `stor_01` through `stor_08` using the following template, and then fill in the placeholder values for each service referencing the example:

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

ℹ️ For the correct size to use, see Recommended storage pool overprovisioning percentages.

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
stor_01.yml	8013	i1b:100.127.103.1/16 i2b:100.127.104.1/16	0	netapp_01	beegfs_s1_s2	A
stor_02.yml	8023	i2b:100.127.104.2/16 i1b:100.127.103.2/16	0	netapp_01	beegfs_s1_s2	B
stor_03.yml	8033	i3b:100.127.103.3/16 i4b:100.127.104.3/16	1	netapp_02	beegfs_s3_s4	A
stor_04.yml	8043	i4b:100.127.104.4/16 i3b:100.127.103.4/16	1	netapp_02	beegfs_s3_s4	B

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
stor_05.yml	8053	i1b:100.127.1 03.5/16 i2b:100.127.1 04.5/16	0	netapp_01	beegfs_s5_s6	A
stor_06.yml	8063	i2b:100.127.1 04.6/16 i1b:100.127.1 03.6/16	0	netapp_01	beegfs_s5_s6	B
stor_07.yml	8073	i3b:100.127.1 03.7/16 i4b:100.127.1 04.7/16	1	netapp_02	beegfs_s7_s8	A
stor_08.yml	8083	i4b:100.127.1 04.8/16 i3b:100.127.1 03.8/16	1	netapp_02	beegfs_s7_s8	B

Step 3: Configure the inventory for a Metadata + storage building block

These steps describe how to set up an Ansible inventory for a BeeGFS metadata + storage building block.

Steps

1. In `inventory.yml`, populate the following parameters under the existing configuration:

```
meta_09:
  hosts:
    beegfs_03:
    beegfs_04:
stor_09:
  hosts:
    beegfs_03:
    beegfs_04:
meta_10:
  hosts:
    beegfs_03:
    beegfs_04:
stor_10:
  hosts:
    beegfs_03:
    beegfs_04:
meta_11:
  hosts:
    beegfs_03:
    beegfs_04:
```

```
        stor_11:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_12:
          hosts:
            beegfs_03:
            beegfs_04:
        stor_12:
          hosts:
            beegfs_03:
            beegfs_04:
        meta_13:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_13:
          hosts:
            beegfs_04:
            beegfs_03:
        meta_14:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_14:
          hosts:
            beegfs_04:
            beegfs_03:
        meta_15:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_15:
          hosts:
            beegfs_04:
            beegfs_03:
        meta_16:
          hosts:
            beegfs_04:
            beegfs_03:
        stor_16:
          hosts:
            beegfs_04:
            beegfs_03:
```

2. Under `group_vars/`, create files for resource groups `meta_09` through `meta_16` using the following

template, and then fill in the placeholder values for each service referencing the example:

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

ⓘ   For the correct size to use, see Recommended storage pool overprovisioning percentages.

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
meta_09.yml	8015	i1b:100.127.1 01.9/16 i2b:100.127.1 02.9/16	0	netapp_03	beegfs_m9_ m10_m13_m 14	A
meta_10.yml	8025	i2b:100.127.1 02.10/16 i1b:100.127.1 01.10/16	0	netapp_03	beegfs_m9_ m10_m13_m 14	B
meta_11.yml	8035	i3b:100.127.1 01.11/16 i4b:100.127.1 02.11/16	1	netapp_04	beegfs_m11_ m12_m15_m 16	A
meta_12.yml	8045	i4b:100.127.1 02.12/16 i3b:100.127.1 01.12/16	1	netapp_04	beegfs_m11_ m12_m15_m 16	B

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
meta_13.yml	8055	i1b:100.127.1 01.13/16 i2b:100.127.1 02.13/16	0	netapp_03	beegfs_m9_ m10_m13_m 14	A
meta_14.yml	8065	i2b:100.127.1 02.14/16 i1b:100.127.1 01.14/16	0	netapp_03	beegfs_m9_ m10_m13_m 14	B
meta_15.yml	8075	i3b:100.127.1 01.15/16 i4b:100.127.1 02.15/16	1	netapp_04	beegfs_m11_ m12_m15_m 16	A
meta_16.yml	8085	i4b:100.127.1 02.16/16 i3b:100.127.1 01.16/16	1	netapp_04	beegfs_m11_ m12_m15_m 16	B

3. Under `group_vars/`, create files for resource groups `stor_09` through `stor_16` using the following template, and then fill in the placeholder values for each service referencing the example:

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

ⓘ For the correct size to use, see Recommended storage pool overprovisioning percentages..

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
stor_09.yml	8013	i1b:100.127.1 03.9/16 i2b:100.127.1 04.9/16	0	netapp_03	beegfs_s9_s1 0	A
stor_10.yml	8023	i2b:100.127.1 04.10/16 i1b:100.127.1 03.10/16	0	netapp_03	beegfs_s9_s1 0	B
stor_11.yml	8033	i3b:100.127.1 03.11/16 i4b:100.127.1 04.11/16	1	netapp_04	beegfs_s11_s 12	A
stor_12.yml	8043	i4b:100.127.1 04.12/16 i3b:100.127.1 03.12/16	1	netapp_04	beegfs_s11_s 12	B
stor_13.yml	8053	i1b:100.127.1 03.13/16 i2b:100.127.1 04.13/16	0	netapp_03	beegfs_s13_s 14	A
stor_14.yml	8063	i2b:100.127.1 04.14/16 i1b:100.127.1 03.14/16	0	netapp_03	beegfs_s13_s 14	B
stor_15.yml	8073	i3b:100.127.1 03.15/16 i4b:100.127.1 04.15/16	1	netapp_04	beegfs_s15_s 16	A
stor_16.yml	8083	i4b:100.127.1 04.16/16 i3b:100.127.1 03.16/16	1	netapp_04	beegfs_s15_s 16	B

Step 4: Configure the inventory for a storage-only building block

These steps describe how to set up an Ansible inventory for a BeeGFS storage-only building block. The major difference between setting up the configuration for a metadata + storage versus a storage-only building block is the omission of all metadata resource groups and changing `criteria_drive_count` from 10 to 12 for each storage pool.

Steps

1. In `inventory.yml`, populate the following parameters under the existing configuration:

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

2. Under `group_vars/`, create files for resource groups `stor_17` through `stor_24` using the following template, and then fill in the placeholder values for each service referencing the example:

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

ⓘ For the correct size to use, see Recommended storage pool overprovisioning percentages.

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
stor_17.yml	8013	i1b:100.127.1 03.17/16 i2b:100.127.1 04.17/16	0	netapp_05	beegfs_s17_s 18	A
stor_18.yml	8023	i2b:100.127.1 04.18/16 i1b:100.127.1 03.18/16	0	netapp_05	beegfs_s17_s 18	B
stor_19.yml	8033	i3b:100.127.1 03.19/16 i4b:100.127.1 04.19/16	1	netapp_06	beegfs_s19_s 20	A
stor_20.yml	8043	i4b:100.127.1 04.20/16 i3b:100.127.1 03.20/16	1	netapp_06	beegfs_s19_s 20	B

File name	Port	Floating IPs	NUMA zone	Block node	Storage pool	Owning controller
stor_21.yml	8053	i1b:100.127.103.21/16 i2b:100.127.104.21/16	0	netapp_05	beegfs_s21_s22	A
stor_22.yml	8063	i2b:100.127.104.22/16 i1b:100.127.103.22/16	0	netapp_05	beegfs_s21_s22	B
stor_23.yml	8073	i3b:100.127.103.23/16 i4b:100.127.104.23/16	1	netapp_06	beegfs_s23_s24	A
stor_24.yml	8083	i4b:100.127.104.24/16 i3b:100.127.103.24/16	1	netapp_06	beegfs_s23_s24	B

Deploy BeeGFS

Deploying and managing the configuration involves running one or more playbooks that contain the tasks Ansible needs to execute and bring the overall system to the desired state.

While all tasks can be included in a single playbook, for complex systems, this quickly becomes unwieldy to manage. Ansible allows you to create and distribute roles as a way of packaging reusable playbooks and related content (for example: default variables, tasks, and handlers). For more information, see the Ansible documentation for Roles.

Roles are often distributed as part of an Ansible collection containing related roles and modules. Thus, these playbooks primarily just import several roles distributed in the various NetApp E-Series Ansible collections.

> ⓘ Currently, at least two building blocks (four file nodes) are required to deploy BeeGFS, unless a separate quorum device is configured as a tiebreaker to mitigate any issues when establishing quorum with a two-node cluster.

Steps

1. Create a new `playbook.yml` file and include the following:

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

```yaml
      import_role:
        name: nar_santricity_management
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
  pre_tasks:
    - name: Ensure a supported version of Python is available on all
file nodes.
      block:
        - name: Check if python is installed.
          failed_when: false
          changed_when: false
          raw: python --version
          register: python_version
        - name: Check if python3 is installed.
          raw: python3 --version
          failed_when: false
          changed_when: false
          register: python3_version
          when: 'python_version["rc"] != 0 or (python_version["stdout"]
| regex_replace("Python ", "")) is not version("3.0", ">=")'
        - name: Install python3 if needed.
          raw: |
            id=$(grep "^ID=" /etc/*release* | cut -d= -f 2 | tr -d '"')
            case $id in
              ubuntu) sudo apt install python3 ;;
              rhel|centos) sudo yum -y install python3 ;;
              sles) sudo zypper install python3 ;;
            esac
          args:
            executable: /bin/bash
          register: python3_install
          when: python_version['rc'] != 0 and python3_version['rc'] != 0
          become: true
        - name: Create a symbolic link to python from python3.
          raw: ln -s /usr/bin/python3 /usr/bin/python
          become: true
          when: python_version['rc'] != 0
      when: inventory_hostname not in
groups[beegfs_ha_ansible_storage_group]
    - name: Verify any provided tags are supported.
      fail:
        msg: "{{ item }} tag is not a supported BeeGFS HA tag. Rerun
your playbook command with --list-tags to see all valid playbook tags."
```

```
          when: 'item not in ["all", "storage", "beegfs_ha",
  "beegfs_ha_package", "beegfs_ha_configure",
  "beegfs_ha_configure_resource", "beegfs_ha_performance_tuning",
  "beegfs_ha_backup", "beegfs_ha_client"]'
          loop: "{{ ansible_run_tags }}"
      tasks:
        - name: Verify before proceeding.
          pause:
            prompt: "Are you ready to proceed with running the BeeGFS HA
  role? Depending on the size of the deployment and network performance
  between the Ansible control node and BeeGFS file and block nodes this
  can take awhile (10+ minutes) to complete."
        - name: Verify the BeeGFS HA cluster is properly deployed.
          ansible.builtin.import_role:
            name: netapp_eseries.beegfs.beegfs_ha_7_4
```

(i) This playbook runs a few `pre_tasks` that verify Python 3 is installed on the file nodes and check that the Ansible tags provided are supported.

2. Use the `ansible-playbook` command with the inventory and playbook files when you're ready to deploy BeeGFS.

The deployment will run all `pre_tasks`, and then prompt for user confirmation before proceeding with the actual BeeGFS deployment.

Run the following command, adjusting the number of forks as needed (see the note below):

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

(i) Especially for larger deployments, overriding the default number of forks (5) using the `forks` parameter is recommended to increase the number of hosts that Ansible configures in parallel. (For more information, see Controlling playbook execution.) The maximum value setting depends on the processing power available on the Ansible control node. The above example of 20 was run on a virtual Ansible control node with 4 CPUs (Intel® Xeon® Gold 6146 CPU @ 3.20GHz).

Depending on the size of the deployment and network performance between the Ansible control node and BeeGFS file and block nodes, deployment time might vary.

Configure BeeGFS clients

You must install and configure the BeeGFS client on any hosts that need access to the BeeGFS file system, such as compute or GPU nodes. For this task, you can use Ansible and the BeeGFS collection.

Steps

1. If needed, set up passwordless SSH from the Ansible control node to each of the hosts you want to configure as BeeGFS clients:

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

2. Under `host_vars/`, create a file for each BeeGFS client named `<HOSTNAME>.yml` with the following content, filling in the placeholder text with the correct information for your environment:

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

> (i) If deploying with a two subnet addressing scheme, two InfiniBand interfaces must be configured on each client, one in each of the two storage IPoIB subnets. If using the example subnets and recommended ranges for each BeeGFS service listed here, clients should have one interface configured in the range of `100.127.1.0` to `100.127.99.255` and the other in `100.128.1.0` to `100.128.99.255`.

3. Create a new file `client_inventory.yml`, and then populate the following parameters at the top:

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

> (i) Do not store passwords in plain text. Instead, use the Ansible Vault (see the Ansible documentation for Encrypting content with Ansible Vault) or use the `--ask-become-pass` option when running the playbook.

4. In the `client_inventory.yml` file, list all hosts that should be configured as BeeGFS clients under the `beegfs_clients` group, and then specify any additional configuration required to build the BeeGFS client kernel module.

```
  children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        beegfs_01:
        beegfs_02:
        beegfs_03:
        beegfs_04:
        beegfs_05:
        beegfs_06:
        beegfs_07:
        beegfs_08:
        beegfs_09:
        beegfs_10:
      vars:
        # OPTION 1: If you're using the NVIDIA OFED drivers and they are
already installed:
        eseries_ib_skip: True # Skip installing inbox drivers when using
the IPoIB role.
        beegfs_client_ofed_enable: True
        beegfs_client_ofed_include_path:
"/usr/src/ofa_kernel/default/include"
        # OPTION 2: If you're using inbox IB/RDMA drivers and they are
already installed:
        eseries_ib_skip: True # Skip installing inbox drivers when using
the IPoIB role.
        # OPTION 3: If you want to use inbox IB/RDMA drivers and need
them installed/configured.
        eseries_ib_skip: False # Default value.
        beegfs_client_ofed_enable: False # Default value.
```

> (i) When using the NVIDIA OFED drivers, make sure that
> `beegfs_client_ofed_include_path` points to the correct "header include path" for
> your Linux installation. For more information, see the BeeGFS documentation for RDMA
> support.

5. In the `client_inventory.yml` file, list the BeeGFS file systems you want mounted at the bottom of any previously defined `vars`.

```
        beegfs_client_mounts:
          - sysMgmtdHost: 100.127.101.0 # Primary IP of the BeeGFS
management service.
            mount_point: /mnt/beegfs    # Path to mount BeeGFS on the
client.
            connInterfaces:
              - <INTERFACE> # Example: ibs4f1
              - <INTERFACE>
            beegfs_client_config:
              # Maximum number of simultaneous connections to the same
node.
              connMaxInternodeNum: 128 # BeeGFS Client Default: 12
              # Allocates the number of buffers for transferring IO.
              connRDMABufNum: 36 # BeeGFS Client Default: 70
              # Size of each allocated RDMA buffer
              connRDMABufSize: 65536 # BeeGFS Client Default: 8192
              # Required when using the BeeGFS client with the shared-
disk HA solution.
              # This does require BeeGFS targets be mounted in the
default "sync" mode.
              # See the documentation included with the BeeGFS client
role for full details.
              sysSessionChecksEnabled: false
```

> ⓘ The `beegfs_client_config` represents the settings that were tested. See the documentation included with the `netapp_eseries.beegfs` collection's `beegfs_client` role for a comprehensive overview of all options. This includes details around mounting multiple BeeGFS file systems or mounting the same BeeGFS file system multiple times.

6. Create a new `client_playbook.yml` file, and then populate the following parameters:

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

> ⓘ Omit importing the `netapp_eseries.host` collection and `ipoib` role if you have already
> installed the required IB/RDMA drivers and configured IPs on the appropriate IPoIB
> interfaces.

7. To install and build the client and mount BeeGFS, run the following command:

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

8. Before you place the BeeGFS file system in production, we **strongly** recommend that you log in to any
   clients and run `beegfs-fsck --checkfs` to ensure that all nodes are reachable and there are no issues
   reported.