



Define BeeGFS services

BeeGFS on NetApp with E-Series Storage

NetApp

January 27, 2026

Table of Contents

- Define BeeGFS services 1
 - Define the BeeGFS management service 1
 - Overview 1
 - Steps 1
 - Define the BeeGFS metadata service 2
 - Overview 2
 - Steps 3
 - Define the BeeGFS storage service 4
 - Overview 4
 - Steps 4

Define BeeGFS services

Define the BeeGFS management service

BeeGFS services are configured using group variables (group_vars).

Overview

This section walks through defining the BeeGFS management service. Only one service of this type should exist in the HA cluster(s) for a particular file system. Configuring this service includes defining:

- The service type (management).
- Defining any configuration that should only apply to this BeeGFS service.
- Configuring one or more floating IPs (logical interfaces) where this service can be reached.
- Specifying where/how a volume should be to store data for this service (the BeeGFS management target).

Steps

Create a new file `group_vars/mgmt.yml` and referencing the [Plan the File System](#) section populate it as follows:

1. Indicate this file represents the configuration for a BeeGFS management service:

```
beegfs_service: management
```

2. Define any configuration that should apply only to this BeeGFS service. This is not typically required for the management service unless you need to enable quotas, however any supported configuration parameter from `beegfs-mgmt.conf` can be included. Note the following parameters are configured automatically/elsewhere and should not be specified here: `storeMgmtDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, and `connNetFilterFile`.

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

3. Configure one or more floating IPs that other services and clients will use to connect to this service (this will automatically set the BeeGFS `connInterfacesFile` option):

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

4. Optionally, specify one or more allowed IP subnets which may be used for outgoing communication (this will automatically set the BeeGFS `connNetFilterFile` option):

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

5. Specify the BeeGFS management target where this service will store data according to the following guidelines:

- The same storage pool or volume group name can be used for multiple BeeGFS services/targets, simply ensure to use the same name, `raid_level`, `criteria_*`, and `common_*` configuration for each (the volumes listed for each service should be different).
- Volume sizes should be specified as a percentage of the storage pool/volume group and the total should not exceed 100 across all services/volumes using a particular storage pool/volume group. Note when using SSDs it is recommended to leave some free space in the volume group to maximize SSD performance and wear life (click [here](#) for more details).
- Click [here](#) for a full list of configuration options available for the `eseries_storage_pool_configuration`. Note some options such as `state`, `host`, `host_type`, `workload_name`, and `workload_metadata` and volume names are generated automatically and should not be specified here.

```
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
```

Click [here](#) for an example of a complete inventory file representing a BeeGFS management service.

Define the BeeGFS metadata service

BeeGFS services are configured using group variables (`group_vars`).

Overview

This section walks through defining the BeeGFS metadata service. At least one service of this type should exist in the HA cluster(s) for a particular file system. Configuring this service includes defining:

- The service type (metadata).
- Defining any configuration that should only apply to this BeeGFS service.
- Configuring one or more floating IPs (logical interfaces) where this service can be reached.

- Specifying where/how a volume should be to store data for this service (the BeeGFS metadata target).

Steps

Referencing the [Plan the File System](#) section, create a file at `group_vars/meta_<ID>.yaml` for each metadata service in the cluster, and populate them as follows:

1. Indicate this file represents the configuration for a BeeGFS metadata service:

```
beegfs_service: metadata
```

2. Define any configuration that should apply only to this BeeGFS service. At minimum you must specify the desired TCP and UDP port, however any supported configuration parameter from `beegfs-meta.conf` can also be included. Note the following parameters are configured automatically/elsewhere and should not be specified here: `sysMgmtHost`, `storeMetaDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, and `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 one or more floating IPs that other services and clients will use to connect to this service (this will automatically set the BeeGFS `connInterfacesFile` option):

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

4. Optionally, specify one or more allowed IP subnets which may be used for outgoing communication (this will automatically set the BeeGFS `connNetFilterFile` option):

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

5. Specify the BeeGFS metadata target where this service will store data according to the following guidelines (this will also automatically configure the `storeMetaDirectory` option):
 - a. The same storage pool or volume group name can be used for multiple BeeGFS services/targets, simply ensure to use the same name, `raid_level`, `criteria_*`, and `common_*` configuration for each (the volumes listed for each service should be different).
 - b. Volume sizes should be specified as a percentage of the storage pool/volume group and the total should not exceed 100 across all services/volumes using a particular storage pool/volume group. Note

when using SSDs it is recommended to leave some free space in the volume group to maximize SSD performance and wear life (click [here](#) for more details).

- c. Click [here](#) for a full list of configuration options available for the `eseries_storage_pool_configuration`. Note some options such as `state`, `host`, `host_type`, `workload_name`, and `workload_metadata` and volume names are generated automatically and should not be specified here.

```
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
```

Click [here](#) for an example of a complete inventory file representing a BeeGFS metadata service.

Define the BeeGFS storage service

BeeGFS services are configured using group variables (`group_vars`).

Overview

This section walks through defining the BeeGFS storage service. At least one service of this type should exist in the HA cluster(s) for a particular file system. Configuring this service includes defining:

- The service type (storage).
- Defining any configuration that should only apply to this BeeGFS service.
- Configuring one or more floating IPs (logical interfaces) where this service can be reached.
- Specifying where/how volume(s) should be to store data for this service (the BeeGFS storage targets).

Steps

Referencing the [Plan the File System](#) section, create a file at `group_vars/stor_<ID>.yaml` for each storage service in the cluster, and populate them as follows:

1. Indicate this file represents the configuration for a BeeGFS storage service:

```
beegfs_service: storage
```

2. Define any configuration that should apply only to this BeeGFS service. At minimum you must specify the desired TCP and UDP port, however any supported configuration parameter from `beegfs-storage.conf` can also be included. Note the following parameters are configured automatically/elsewhere and should not be specified here: `sysMgmtHost`, `storeStorageDirectory`, `connAuthFile`, `connDisableAuthentication`, `connInterfacesFile`, and `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 one or more floating IPs that other services and clients will use to connect to this service (this will automatically set the BeeGFS `connInterfacesFile` option):

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

4. Optionally, specify one or more allowed IP subnets which may be used for outgoing communication (this will automatically set the BeeGFS `connNetFilterFile` option):

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

5. Specify the BeeGFS storage target(s) where this service will store data according to the following guidelines (this will also automatically configure the `storeStorageDirectory` option):
 - a. The same storage pool or volume group name can be used for multiple BeeGFS services/targets, simply ensure to use the same name, `raid_level`, `criteria_*`, and `common_*` configuration for each (the volumes listed for each service should be different).
 - b. Volume sizes should be specified as a percentage of the storage pool/volume group and the total should not exceed 100 across all services/volumes using a particular storage pool/volume group. Note when using SSDs it is recommended to leave some free space in the volume group to maximize SSD performance and wear life (click [here](#) for more details).
 - c. Click [here](#) for a full list of configuration options available for the `eseries_storage_pool_configuration`. Note some options such as `state`, `host`, `host_type`, `workload_name`, and `workload_metadata` and volume names are generated automatically and should not be specified here.

```

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

```

Click [here](#) for an example of a complete inventory file representing a BeeGFS storage service.

Copyright information

Copyright © 2026 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.