Deploy Features and Integrations

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Table of Contents

Deploy Features and Integrations

BeeGFS CSI Driver

Configure TLS Encryption for BeeGFS v8

Configure TLS encryption to secure communication between BeeGFS v8 management services and clients.

Overview

BeeGFS v8 introduces TLS support for encrypting network communications between administrative tools (such as the `beegfs` command-line utility) and BeeGFS server services like Management or Remote. This guide covers configuring TLS encryption in your BeeGFS cluster using three TLS configuration methods:

- **Using a trusted Certificate Authority:** Use existing CA-signed certificates on your BeeGFS cluster.
- **Creating a local Certificate Authority:** Creating a local Certificate Authority and using it to sign certificates for your BeeGFS services. This approach is suitable for environments where you want to manage your own trust chain without relying on an external CA.
- **TLS Disabled:** Disable TLS entirely for environments where encryption is not required or for troubleshooting. This is discouraged as it exposes potentially sensitive information about the internal file system structure and configuration as cleartext.

Choose the method that best suits your environment and organizational policies. Refer to the BeeGFS TLS documentation for additional details.

Machines running the `beegfs-client` service do not require TLS to mount the BeeGFS filesystem. TLS must be set up to utilize the BeeGFS CLI and other beegfs services, such as remote and sync.

Using a Trusted Certificate Authority

If you have access to certificates issued by a trusted Certificate Authority (CA)—whether from an internal enterprise CA or a third-party provider—you can configure BeeGFS v8 to use these CA-signed certificates instead of generating self-signed ones.

Deploying a new BeeGFS v8 cluster

For a new BeeGFS v8 cluster deployment, configure the Ansible inventory's `user_defined_params.yml` file to reference your CA-signed certificates:

```
beegfs_ha_tls_enabled: true

beegfs_ha_ca_cert_src_path: files/beegfs/cert/ca_cert.pem

beegfs_ha_tls_cert_src_path: files/beegfs/cert/mgmtd_tls_cert.pem

beegfs_ha_tls_key_src_path: files/beegfs/cert/mgmtd_tls_key.pem
```

If `beegfs_ha_tls_config_options.alt_names` is not empty, Ansible will automatically generate a self-signed TLS certificate and key, using the provided alt_names as Subject Alternative Names (SANs) in the certificate. To use your own custom TLS certificate and key (as specified by `beegfs_ha_tls_cert_src_path` and `beegfs_ha_tls_key_src_path`), you must comment out or remove the entire `beegfs_ha_tls_config_options` section. Otherwise, the self-signed certificate generation will take precedence, and your custom certificate and key will not be used.

Configuring an existing BeeGFS v8 cluster

For an existing BeeGFS v8 cluster, set the paths in the BeeGFS management services' configuration file to the file node's CA-signed certificates:

```
tls-cert-file = /path/to/cert.pem
tls-key-file = /path/to/key.pem
```

Configuring BeeGFS v8 clients with CA-signed certificates

To configure BeeGFS v8 clients to trust CA-signed certificates using the system's certificate pool, set tls-cert-file = "" in each client's configuration. If the system certificate pool is not being used, provide the path to a local certificate by setting tls-cert-file = <local cert>. This setup allows clients to authenticate the certificates presented by BeeGFS management services.

Creating a local Certificate Authority

If your organization wants to create its own certificate infrastructure for the BeeGFS cluster, you can create a local Certificate Authority (CA) to issue and sign certificates for your BeeGFS cluster. This approach involves creating a CA that signs certificates for BeeGFS management services, which are then distributed to clients to establish a trust chain. Follow these instructions to set up a local CA and deploy certificates on your existing or new BeeGFS v8 cluster.

Deploying a new BeeGFS v8 cluster

For a new BeeGFS v8 deployment, the `beegfs_8` Ansible role will handle creating a local CA on the control node and generating the necessary certificates for the management services. This can be enabled by setting the following parameters in the Ansible inventory's `user_defined_params.yml` file:

```
beegfs_ha_tls_enabled: true

beegfs_ha_ca_cert_src_path: files/beegfs/cert/local_ca_cert.pem

beegfs_ha_tls_cert_src_path: files/beegfs/cert/mgmtd_tls_cert.pem

beegfs_ha_tls_key_src_path: files/beegfs/cert/mgmtd_tls_key.pem

beegfs_ha_tls_config_options:
  alt_names: [<mgmt_service_ip>]
```

If `beegfs_ha_tls_config_options.alt_names` is not supplied, then Ansible will attempt to use existing certificates in the specified certificate/key paths.

Configuring an existing BeeGFS v8 cluster

For an existing BeeGFS cluster, you can integrate TLS by creating a local Certificate Authority and generating the necessary certificates for the management services. Update the paths in the BeeGFS management services' configuration file to point to the newly created certificates.

Instructions in this section are to be used as a reference. Proper security precautions should be taken when handling private keys and certificates.

Create the Certificate Authority

On a trusted machine, create a local Certificate Authority to sign certificates for your BeeGFS management services. The CA certificate will be distributed to clients to establish trust and enable secure communication with BeeGFS services.

The following instructions are a reference for creating a local Certificate Authority on a RHEL-based system.

1. Install OpenSSL if it is not already installed:

   ```
   dnf install openssl
   ```

2. Create a working directory to store certificate files:

   ```
   mkdir -p ~/beegfs_tls && cd ~/beegfs_tls
   ```

3. Generate the CA private key:

   ```
   openssl genrsa -out ca_key.pem 4096
   ```

4. Create a CA configuration file named `ca.cnf` and adjust the distinguished name fields to match your organization:

```
[ req ]
default_bits        = 4096
distinguished_name  = req_distinguished_name
x509_extensions     = v3_ca
prompt              = no

[ req_distinguished_name ]
C  = <Country>
ST = <State>
L  = <City>
O  = <Organization>
OU = <OrganizationalUnit>
CN = BeeGFS-CA

[ v3_ca ]
basicConstraints = critical,CA:TRUE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
```

5. Generate the CA certificate. This certificate should be valid for the life of the system, otherwise you will need to plan to regenerate certificates before they expire. Once a certificate expires, communication between some components will not be possible and updating TLS certificates will typically require restarting services to complete.

   The following command generates a CA certificate valid for 1 year:

   ```
   openssl req -new -x509 -key ca_key.pem -out ca_cert.pem -days 365
   -config ca.cnf
   ```

   ⓘ While this example uses a 1-year validity period for simplicity, you should adjust the `-days` parameter according to your organization's security requirements and establish a certificate renewal process.

Create management service certificates

Generate certificates for your BeeGFS management services and sign them with the CA you created. These certificates will be installed on the file nodes running BeeGFS management services.

1. Generate the management service private key:

   ```
   openssl genrsa -out mgmtd_tls_key.pem 4096
   ```

2. Create a certificate configuration file named `tls_san.cnf` with Subject Alternative Names (SANs) for all management service IP addresses:

```
[ req ]
default_bits        = 4096
distinguished_name  = req_distinguished_name
req_extensions      = req_ext
prompt              = no

[ req_distinguished_name ]
C  = <Country>
ST = <State>
L  = <City>
O  = <Organization>
OU = <OrganizationalUnit>
CN = beegfs-mgmt

[ req_ext ]
subjectAltName = @alt_names

[ v3_ca ]
subjectAltName = @alt_names
basicConstraints = CA:FALSE

[ alt_names ]
IP.1 = <beegfs_mgmt_service_ip_1>
IP.2 = <beegfs_mgmt_service_ip_2>
```

Update the distinguished name fields to match your CA configuration and the `IP.1` and `IP.2` values with your management service IP addresses.

3. Generate a Certificate Signing Request (CSR):

```
openssl req -new -key mgmtd_tls_key.pem -out mgmtd_tls_csr.pem -config
tls_san.cnf
```

4. Sign the certificate with your CA (valid for 1 year):

```
openssl x509 -req -in mgmtd_tls_csr.pem -CA ca_cert.pem -CAkey
ca_key.pem -CAcreateserial -out mgmtd_tls_cert.pem -days 365 -sha256
-extensions v3_ca -extfile tls_san.cnf
```

Adjust the certificate validity period (`-days 365`) based on your organization's security policies. Many organizations require certificate rotation every 1-2 years.

5. Verify the certificate was created correctly:

```
openssl x509 -in mgmtd_tls_cert.pem -text -noout
```

Confirm that the Subject Alternative Name section includes all your management IP addresses.

Distribute certificates to file nodes

Distribute the CA certificate and management service certificates to the appropriate file nodes and clients.

1. Copy the CA certificate and management service certificate and key to the file nodes running management services:

```
scp ca_cert.pem mgmtd_tls_cert.pem mgmtd_tls_key.pem
user@beegfs_01:/etc/beegfs/
scp ca_cert.pem mgmtd_tls_cert.pem mgmtd_tls_key.pem
user@beegfs_02:/etc/beegfs/
```

Point the management service to the TLS certificates

Update the BeeGFS management service configuration to enable TLS and reference the created TLS certificates.

1. From a file node running the BeeGFS management service, edit the management service configuration file, for example at `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml`. Add or update the following TLS-related parameters:

```
tls-disable = false
tls-cert-file = "/etc/beegfs/mgmtd_tls_cert.pem"
tls-key-file = "/etc/beegfs/mgmtd_tls_key.pem"
```

2. Take appropriate action to safely restart the BeeGFS management service for the changes to take effect:

```
systemctl restart beegfs-mgmtd
```

3. Verify the management service started successfully:

```
journalctl -xeu beegfs-mgmtd
```

Look for log entries indicating successful TLS initialization and certificate loading.

```
Successfully initialized certificate verification library.
Successfully loaded license certificate: TMP-XXXXXXXXXX
```

Configure TLS for BeeGFS v8 clients

Create and distribute certificates signed by the local CA to all BeeGFS clients that will be requiring communication with BeeGFS management services.

1. Generate a certificate for the client using the same process as the management service certificate above, but with the client's IP address or hostname in the Subject Alternative Name (SAN) field.
2. Secure remote copy the client's certificate to the client and rename the certificate to `cert.pem` on the client:

```
scp client_cert.pem user@client:/etc/beegfs/cert.pem
```

3. Restart the BeeGFS client service on all clients:

```
systemctl restart beegfs-client
```

4. Verify client connectivity by executing a `beegfs CLI` command, such as:

```
beegfs health check
```

Disabling TLS

TLS can be disabled for troubleshooting or if desired by the users. This is discouraged as it exposes potentially sensitive information about the internal file system structure and configuration in cleartext form. Follow these instructions to disable TLS on your existing or new BeeGFS v8 cluster.

Deploying a new BeeGFS v8 cluster

For a new BeeGFS cluster deployment, the cluster can be deployed with TLS disabled by setting the following parameter in the Ansible inventory's `user_defined_params.yml` file:

```
beegfs_ha_tls_enabled: false
```

Configuring an existing BeeGFS v8 cluster

For an existing BeeGFS v8 cluster, edit the management service configuration file. For example, edit the file at `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml` and set:

```
tls-disable = true
```

Take appropriate action to safely restart the management service for the changes to take effect.

Copyright information

Trademark information