Update HA cluster components

BeeGFS on NetApp with E-Series Storage

NetApp
January 27, 2026

Table of Contents

Update HA cluster components

Upgrade BeeGFS services

Use Ansible to update the BeeGFS version running on your HA cluster.

Overview

BeeGFS follows a `major.minor.patch` versioning scheme. BeeGFS HA Ansible roles are provided for each supported `major.minor` version (e.g., `beegfs_ha_7_2` and `beegfs_ha_7_3`). Each HA role is pinned to the latest BeeGFS patch version available at the time of the Ansible collection's release.

Ansible should be used for all BeeGFS upgrades, including moving between major, minor, and patch versions of BeeGFS. To update BeeGFS you will first need to update the BeeGFS Ansible collection, which will also pull in the latest fixes and enhancements to the deployment/management automation and underlying HA cluster. Even after updating to the latest version of the collection, BeeGFS will not be upgraded until `ansible-playbook` is ran with the `-e "beegfs_ha_force_upgrade=true"` set. For additional details about each upgrade refer to the BeeGFS Upgrade documentation for your current version.

If you are upgrading to BeeGFS v8, see the Upgrade to BeeGFS v8 procedure instead.

Tested upgrade paths

The following upgrade paths have been tested and verified:

Original Version	Upgrade Version	Multirail	Details
7.2.6	7.3.2	Yes	Upgrading beegfs collection from v3.0.1 to v3.1.0, multirail added
7.2.6	7.2.8	No	Upgrading beegfs collection from v3.0.1 to v3.1.0
7.2.8	7.3.1	Yes	Upgrade using beegfs collection v3.1.0, multirail added
7.3.1	7.3.2	Yes	Upgrade using beegfs collection v3.1.0
7.3.2	7.4.1	Yes	Upgrade using beegfs collection v3.2.0
7.4.1	7.4.2	Yes	Upgrade using beegfs collection v3.2.0
7.4.2	7.4.6	Yes	Upgrade using beegfs collection v3.2.0
7.4.6	8.0	Yes	Upgrade using the instructions in the Upgrade to BeeGFS v8 procedure.
7.4.6	8.1	Yes	Upgrade using the instructions in the Upgrade to BeeGFS v8 procedure.
7.4.6	8.2	Yes	Upgrade using the instructions in the Upgrade to BeeGFS v8 procedure.

BeeGFS upgrade steps

The following sections provide steps to update the BeeGFS Ansible collection and BeeGFS itself. Pay special attention to any extra step(s) for updating BeeGFS major or minor versions.

Step 1: Upgrade BeeGFS collection

For collection upgrades with access to Ansible Galaxy, run the following command:

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

For offline collection upgrades, download the collection from Ansible Galaxy by clicking on the desired `Install Version` and then `Download tarball`. Transfer the tarball to your Ansible control node and run the following command.

```
ansible-galaxy collection install netapp_eseries-beegfs-<VERSION>.tar.gz
--upgrade
```

See Installing Collections for more information.

Step 2: Update Ansible inventory

Make any required or desired updates to your cluster's Ansible inventory files. See the Version upgrade notes section below for details about your specific upgrade requirements. See the Ansible Inventory Overview section for general information on configuring your BeeGFS HA inventory.

Step 3: Update Ansible playbook (when updating major or minor versions only)

If you are moving between major or minor versions, in the `playbook.yml` file used to deploy and maintain the cluster, update the name of the `beegfs_ha_<VERSION>` role to reflect the desired version. For example, if you wanted to deploy BeeGFS 7.4 this would be `beegfs_ha_7_4`:

```
- hosts: all
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.beegfs
  tasks:
    - name: Ensure BeeGFS HA cluster is setup.
      ansible.builtin.import_role:  # import_role is required for tag
availability.
        name: beegfs_ha_7_4
```

For more details on the contents of this playbook file see the Deploy the BeeGFS HA cluster section.

Step 4: Run the BeeGFS upgrade

To apply the BeeGFS update:

```
ansible-playbook -i inventory.yml beegfs_ha_playbook.yml -e
"beegfs_ha_force_upgrade=true" --tags beegfs_ha
```

Behind the scenes the BeeGFS HA role will handle:

- Ensure the cluster is in an optimal state with each BeeGFS service located on its preferred node.
- Put the cluster in maintenance mode.
- Update the HA cluster components (if needed).
- Upgrade each file node one at a time as follows:
  - Place it into standby and failover its services to the secondary node.
  - Upgrade BeeGFS packages.
  - Fallback back services.
- Move the cluster out of maintenance mode.

Version upgrade notes

Upgrading from BeeGFS version 7.2.6 or 7.3.0

Changes to connection based authentication

BeeGFS version 7.3.2 and later require connection based authentication to be configured. Services will not start without either:

- Specifying a `connAuthFile`, or
- Setting `connDisableAuthentication=true` in the service's configuration file.

It is highly recommended to enable connection based authentication for security. See BeeGFS Connection Based Authentication for more information.

The `beegfs_ha*` roles automatically generate and distribute the authentication file to:

- All file nodes in the cluster
- The Ansible control node at `<playbook_directory>/files/beegfs/<beegfs_mgmt_ip_address>_connAuthFile`

The `beegfs_client` role will automatically detect and apply this file to clients when it's present.

If you did not use the `beegfs_client` role to configure clients, you must manually distribute the authentication file to each client and configure the `connAuthFile` setting in the `beegfs-client.conf` file. When upgrading from a BeeGFS version without connection based authentication, clients will lose access unless you disable connection based authentication during the upgrade by setting `beegfs_ha_conn_auth_enabled: false` in `group_vars/ha_cluster.yml` (not recommended).

For additional details and alternate configuration options, see the connection authentication configuration step in the Specify Common File Node Configuration section.

Upgrade to BeeGFS v8

Follow these steps to upgrade your BeeGFS HA cluster from version 7.4.6 to BeeGFS v8.

Overview

BeeGFS v8 introduces several significant changes which require additional setup before upgrading from BeeGFS v7. This document guides you through preparing your cluster for BeeGFS v8's new requirements, and then upgrading to BeeGFS v8.

Before upgrading to BeeGFS v8, ensure your system is running at least BeeGFS 7.4.6. Any cluster running a release previous to BeeGFS 7.4.6 must first upgrade to version 7.4.6 before proceeding with this BeeGFS v8 upgrade procedure.

Key changes in BeeGFS v8

BeeGFS v8 introduces the following major changes:

- **License enforcement:** BeeGFS v8 requires a license to use premium features such as storage pools, remote storage targets, BeeOND, and more. Acquire a valid license for your BeeGFS cluster before upgrading. If needed, you can obtain a temporary BeeGFS v8 evaluation license from the BeeGFS License Portal.
- **Management Service Database Migration:** To enable configuration with the new TOML-based format in BeeGFS v8, you must manually migrate your BeeGFS v7 management service database to the updated BeeGFS v8 format.
- **TLS encryption:** BeeGFS v8 introduces TLS for secure communication between services. You will need to generate and distribute TLS certificates for the BeeGFS management service and the `beegfs` command-line utility as part of the upgrade.

For more details and additional changes in BeeGFS 8, see the BeeGFS v8.0.0 Upgrade Guide.

Upgrading to BeeGFS v8 requires cluster downtime. In addition, BeeGFS v7 clients cannot connect to BeeGFS v8 clusters. Coordinate the upgrade timing between the cluster and clients carefully to minimize impact on operations.

Prepare your BeeGFS cluster for the upgrade

Before starting the upgrade, carefully prepare your environment to ensure a smooth transition and minimize downtime.

1. Ensure your cluster is in a healthy state, with all BeeGFS services running on their preferred nodes. From a file node running BeeGFS services, verify all Pacemaker resources are running on their preferred nodes:

   ```
   pcs status
   ```

2. Record and back up your cluster configuration.
   a. Refer to the BeeGFS Backup documentation for instructions on backing up your cluster configuration.
   b. Back up the existing management data directory:

```
cp -r /mnt/mgmt_tgt_mgmt01/data
/mnt/mgmt_tgt_mgmt01/data_beegfs_v7_backup_$(date +%Y%m%d)
```

c. Run the following commands from a beegfs client and save their output for reference:

```
beegfs-ctl --getentryinfo --verbose /path/to/beegfs/mountpoint
```

d. If using mirroring, gather detailed state information:

```
beegfs-ctl --listtargets --longnodes --state --spaceinfo
--mirrorgroups --nodetype=meta
beegfs-ctl --listtargets --longnodes --state --spaceinfo
--mirrorgroups --nodetype=storage
```

3. Prepare your clients for downtime and stop `beegfs-client` services. For each client, run:

```
systemctl stop beegfs-client
```

4. For each Pacemaker cluster, disable STONITH. This will allow you to validate the integrity of the cluster after the upgrade without triggering unnecessary node reboots.

```
pcs property set stonith-enabled=false
```

5. For all Pacemaker clusters in the BeeGFS namespace, use PCS to stop the cluster:

```
pcs cluster stop --all
```

Upgrade the BeeGFS packages

On all file nodes in the cluster, add the BeeGFS v8 package repository for your Linux distribution. Instructions for using the official BeeGFS repositories can be found at the BeeGFS download page. Otherwise, configure your local beegfs mirror repository accordingly.

The following steps walkthrough using the official BeeGFS 8.2 repository on RHEL 9 file nodes. Perform the following steps on all file nodes in the cluster:

1. Import the BeeGFS GPG key:

```
rpm --import https://www.beegfs.io/release/beegfs_8.2/gpg/GPG-KEY-beegfs
```

2. Import the BeeGFS repository:

```
curl -L -o /etc/yum.repos.d/beegfs-rhel9.repo
https://www.beegfs.io/release/beegfs_8.2/dists/beegfs-rhel9.repo
```

Remove any previously configured BeeGFS repositories to avoid conflicts with the new BeeGFS v8 repository.

3. Clean your package manager cache:

```
dnf clean all
```

4. On all file nodes, update the BeeGFS packages to BeeGFS 8.2.

```
dnf update beegfs-mgmtd beegfs-storage beegfs-meta libbeegfs-ib
```

In a standard cluster, the `beegfs-mgmtd` package will only update on the first two file nodes.

Upgrade the management database

On one of the file nodes running the BeeGFS management service, perform the following steps to migrate the management database from BeeGFS v7 to v8.

1. List all NVMe devices and filter for the management target:

```
nvme netapp smdevices | grep mgmt_tgt
```

   a. Note the device path from the output.

   b. Mount the management target device to the existing management target mount point (replace `/dev/nvmeXnY` with your device path):

```
mount /dev/nvmeXnY /mnt/mgmt_tgt_mgmt01/
```

2. Import your BeeGFS 7 management data into the new database format by running:

```
/opt/beegfs/sbin/beegfs-mgmtd --import-from
-v7=/mnt/mgmt_tgt_mgmt01/data/
```

   Expected output:

```
Created new database version 3 at "/var/lib/beegfs/mgmtd.sqlite".
Successfully imported v7 management data from
"/mnt/mgmt_tgt_mgmt01/data/".
```

> The automatic import may not succeed in all cases due to stricter validation requirements in BeeGFS v8. For example, if targets are assigned to non-existent storage pools, the import will fail. If the migration fails, do not proceed with the upgrade. Contact NetApp support for assistance with resolving the database migration issues. As an interim solution, you can downgrade the BeeGFS v8 packages and continue running BeeGFS v7 while the issue is addressed.

3. Move the generated SQLite file to the management service mount:

```
mv /var/lib/beegfs/mgmtd.sqlite /mnt/mgmt_tgt_mgmt01/data/
```

4. Move the generated `beegfs-mgmtd.toml` to the management service mount:

```
mv /etc/beegfs/beegfs-mgmtd.toml /mnt/mgmt_tgt_mgmt01/mgmt_config/
```

Preparing the `beegfs-mgmtd.toml` configuration file will be done after completing the licensing and TLS configuration steps in the next sections.

Configure licensing

1. Install the beegfs license packages on all nodes that run the beegfs management service. This is typically the first two nodes of the cluster:

```
dnf install libbeegfs-license
```

2. Download your BeeGFS v8 license file to the management nodes and place it at:

```
/etc/beegfs/license.pem
```

Configure TLS encryption

BeeGFS v8 requires TLS encryption for secure communication between management services and clients. There are three options for configuring TLS encryption on network communications between management services and client services. The recommended and most secure method is to use certificates signed by a trusted Certificate Authority. Alternatively, you can create your own local CA to sign certificates for your BeeGFS cluster. For environments where encryption is not required or for troubleshooting, TLS can be disabled entirely, though this is discouraged as it exposes sensitive information to the network.

Before proceeding, follow the instructions in the Configure TLS Encryption for BeeGFS 8 guide to setup TLS

encryption for your environment.

Update management service configuration

Prepare the BeeGFS v8 management service configuration file by manually transferring settings from your BeeGFS v7 configuration file into the `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml` file.

1. On the management node with the management target mounted, reference the `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.conf` management service file for BeeGFS 7, then transfer all the settings to the `/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml` file. For a basic setup, your `beegfs-mgmtd.toml` may look like the following:

```
beemsg-port = 8008
grpc-port = 8010
log-level = "info"
node-offline-timeout = "900s"
quota-enable = false
auth-disable = false
auth-file = "/etc/beegfs/<mgmt_service_ip>_connAuthFile"
db-file = "/mnt/mgmt_tgt_mgmt01/data/mgmtd.sqlite"
license-disable = false
license-cert-file = "/etc/beegfs/license.pem"
tls-disable = false
tls-cert-file = "/etc/beegfs/mgmtd_tls_cert.pem"
tls-key-file = "/etc/beegfs/mgmtd_tls_key.pem"
interfaces = ['i1b:mgmt_1', 'i2b:mgmt_2']
```

   Adjust all paths as necessary to match your environment and TLS configuration.

2. On each file node running management services, modify your systemd service file to point to the new configuration file location.

```
sudo sed -i 's|ExecStart=.*|ExecStart=nice -n -3
/opt/beegfs/sbin/beegfs-mgmtd --config-file
/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-mgmtd.toml|'
/etc/systemd/system/beegfs-mgmtd.service
```

   a. Reload systemd:

```
systemctl daemon-reload
```

3. For each file node running management services, open port 8010 for the management service's gRPC communication.

   a. Add port 8010/tcp to the beegfs zone:

```
sudo firewall-cmd --zone=beegfs --permanent --add-port=8010/tcp
```

b. Reload the firewall to apply the change:

```
sudo firewall-cmd --reload
```

Update the BeeGFS monitor script

The Pacemaker `beegfs-monitor` OCF script requires updates to support the new TOML configuration format and systemd service management. Update the script on one node in the cluster, then copy the updated script to all other nodes.

1. Create a backup of the current script:

```
cp /usr/lib/ocf/resource.d/eseries/beegfs-monitor
/usr/lib/ocf/resource.d/eseries/beegfs-monitor.bak.$(date +%F)
```

2. Update the management configuration file path from `.conf` to `.toml`:

```
sed -i 's|mgmt_config/beegfs-mgmtd\.conf|mgmt_config/beegfs-mgmtd.toml|'
/usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

Alternatively, manually locate the following block in the script:

```
case $type in
  management)
    conf_path="${configuration_mount}/mgmt_config/beegfs-mgmtd.conf"
    ;;
```

And replace it with:

```
case $type in
  management)
    conf_path="${configuration_mount}/mgmt_config/beegfs-mgmtd.toml"
    ;;
```

3. Update the `get_interfaces()` and `get_subnet_ips()` functions to support TOML configuration:

a. Open the script in a text editor:

```
vi /usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

b. Locate the two functions: `get_interfaces()` and `get_subnet_ips()`.

c. Delete both entire functions, starting at `get_interfaces()` to the end of `get_subnet_ips()`.

d. Copy and paste the following updated functions in their place:

```
# Return network communication interface name(s) from the BeeGFS
resource's connInterfaceFile
get_interfaces() {
  # Determine BeeGFS service network IP interfaces.
  if [ "$type" = "management" ]; then
    interfaces_line=$(grep "^interfaces =" "$conf_path")
    interfaces_list=$(echo "$interfaces_line" | sed "s/.*= \[\(.*
\)\]/\1/")
    interfaces=$(echo "$interfaces_list" | tr -d "'" | tr -d " " | tr
',' '\n')

    for entry in $interfaces; do
      echo "$entry" | cut -d ':' -f 1
    done
  else
    connInterfacesFile_path=$(grep "^connInterfacesFile" "$conf_path"
| tr -d "[:space:]" | cut -f 2 -d "=")

    if [ -f "$connInterfacesFile_path" ]; then
      while read -r entry; do
        echo "$entry" | cut -f 1 -d ':'
      done < "$connInterfacesFile_path"
    fi
  fi
}

# Return list containing all the BeeGFS resource's usable IP
addresses. *Note that these are filtered by the connNetFilterFile
entries.
get_subnet_ips() {
  # Determine all possible BeeGFS service network IP addresses.
  if [ "$type" != "management" ]; then
    connNetFilterFile_path=$(grep "^connNetFilterFile" "$conf_path" |
tr -d "[:space:]" | cut -f 2 -d "=")

    filter_ips=""
    if [ -n "$connNetFilterFile_path" ] && [ -e
$connNetFilterFile_path ]; then
      while read -r filter; do
        filter_ips="$filter_ips $(get_ipv4_subnet_addresses $filter)"
      done < $connNetFilterFile_path
    fi

    echo "$filter_ips"
  fi
}
```

e. Save and exit the text editor.

f. Run the following command to check the script for syntax errors before proceeding. No output indicates the script is syntactically correct.

```
bash -n /usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

4. Copy the updated `beegfs-monitor` OCF script to all other nodes in the cluster to ensure consistency:

```
scp /usr/lib/ocf/resource.d/eseries/beegfs-monitor
user@node:/usr/lib/ocf/resource.d/eseries/beegfs-monitor
```

Bring the cluster back online

1. Once all the previous upgrade steps have been completed, bring the cluster back online by starting the BeeGFS services on all nodes.

```
pcs cluster start --all
```

2. Verify the `beegfs-mgmtd` service started successfully:

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

Expected output includes lines such as:

```
Started Cluster Controlled beegfs-mgmtd.
Loaded config file from "/mnt/mgmt_tgt_mgmt01/mgmt_config/beegfs-
mgmtd.toml"
Successfully initialized certificate verification library.
Successfully loaded license certificate: TMP-113489268
Opened database at "/mnt/mgmt_tgt_mgmt01/data/mgmtd.sqlite"
Listening for BeeGFS connections on [::]:8008
Serving gRPC requests on [::]:8010
```

If errors appear in the journal logs, review the management configuration file paths and ensure all values were correctly transferred from the BeeGFS 7 configuration file.

3. Run `pcs status` and verify the cluster is healthy and services are started on their preferred nodes.

4. Once the cluster is verified to be healthy, re-enable STONITH:

```
pcs property set stonith-enabled=true
```

5. Proceed to the next section to upgrade the BeeGFS clients on in the cluster and check the BeeGFS cluster's health.

Upgrade BeeGFS clients

After successfully upgrading your cluster to BeeGFS v8, you must also upgrade all BeeGFS clients.

The following steps outline the process to upgrade BeeGFS clients on an Ubuntu-based system.

1. If not already done, stop the BeeGFS client service:

```
systemctl stop beegfs-client
```

2. Add the BeeGFS v8 package repository for your Linux distribution. Instructions for using the official BeeGFS repositories can be found at the ^BeeGFS download page. Otherwise, configure your local BeeGFS mirror repository accordingly.

   The following steps use the official BeeGFS 8.2 repository on an Ubuntu-based system:

3. Import the BeeGFS GPG key:

```
wget https://www.beegfs.io/release/beegfs_8.2/gpg/GPG-KEY-beegfs -O
/etc/apt/trusted.gpg.d/beegfs.asc
```

4. Download the repository file:

```
wget https://www.beegfs.io/release/beegfs_8.2/dists/beegfs-noble.list -O
/etc/apt/sources.list.d/beegfs.list
```

   Remove any previously configured BeeGFS repositories to avoid conflicts with the new BeeGFS v8 repository.

5. Upgrade the BeeGFS client packages:

```
apt-get update
apt-get install --only-upgrade beegfs-client
```

6. Configure TLS for the client. TLS is required to use the BeeGFS CLI. Reference the Configure TLS Encryption for BeeGFS 8 procedure to configure TLS on the client.
7. Start the BeeGFS client service:

```
systemctl start beegfs-client
```

Verify the upgrade

After finishing the upgrade to BeeGFS v8, run the following commands to verify the upgrade was successful.

1. Verify the root inode is owned by the same metadata node as before. This should happen automatically if you used the `import-from-v7` functionality in the management service:

```
beegfs entry info /mnt/beegfs
```

2. Verify all nodes and targets are online and in a good state:

```
beegfs health check
```

> If the "Available Capacity" check warns that targets are low on free space, you can adjust the "capacity pool" thresholds defined in the `beegfs-mgmtd.toml` file so they are better suited to your environment.

Upgrade Pacemaker and Corosync packages in an HA cluster

Follow these steps to upgrade Pacemaker and Corosync packages in an HA cluster.

Overview

Upgrading Pacemaker and Corosync ensures the cluster benefits from new features, security patches, and performance improvements.

Upgrade approach

There are two recommended approaches to upgrading a cluster: a rolling upgrade or a complete cluster shutdown. Each approach has its own advantages and disadvantages. Your upgrade procedure may vary depending on your Pacemaker release version. Refer to ClusterLabs' Upgrading a Pacemaker Cluster documentation to determine which approach to use. Prior to following an upgrade approach, verify that:

- The new Pacemaker and Corosync packages are supported within the NetApp BeeGFS solution.
- Valid backups exist for your BeeGFS filesystem and Pacemaker cluster configuration.
- The cluster is in a healthy state.

Rolling upgrade

This method involves removing each node from the cluster, upgrading it, and then reintroducing it into the cluster until all nodes run the new version. This approach keeps the cluster operational, which is ideal for larger HA clusters, but carries the risk of running mixed versions during the process. This approach should be avoided in a two node cluster.

1. Confirm that the cluster is in an optimal state, with each BeeGFS service running on its preferred node. Refer to Examine the state of the cluster for details.

2. For the node to be upgraded, place it into standby mode to drain (or move) all BeeGFS services:

```
pcs node standby <HOSTNAME>
```

3. Verify that the node's services have drained by running:

```
pcs status
```

   Ensure no services are reported as `Started` on the node in standby.

   > Depending on your cluster size, it may take seconds or minutes for services to move to the sister node. If a BeeGFS service fails to start on the sister node, refer to the Troubleshooting Guides.

4. Shut down the cluster on the node:

```
pcs cluster stop <HOSTNAME>
```

5. Upgrade the Pacemaker, Corosync, and pcs packages on the node:

   > Package manager commands will vary by operating system. The following commands are for systems running RHEL 8 and onward.

```
dnf update pacemaker-<version>
```

```
dnf update corosync-<version>
```

```
dnf update pcs-<version>
```

6. Start Pacemaker cluster services on the node:

```
pcs cluster start <HOSTNAME>
```

7. If the `pcs` package was updated, reauthenticate the node with the cluster:

```
pcs host auth <HOSTNAME>
```

8. Verify the Pacemaker configuration is still valid with the `crm_verify` tool.

This only needs to be verified once during the cluster upgrade.

```
crm_verify -L -V
```

9. Bring the node out of standby:

```
pcs node unstandby <HOSTNAME>
```

10. Relocate all BeeGFS services back to their preferred node:

```
pcs resource relocate run
```

11. Repeat the previous steps for each node in the cluster until all nodes are running the desired Pacemaker, Corosync, and pcs versions.
12. Finally, run `pcs status` and verify the cluster is healthy and the `Current DC` reports the desired Pacemaker version.

If the `Current DC` reports 'mixed-version', then a node in the cluster is still running with the previous Pacemaker version and needs to be upgraded. If any upgraded node is unable to rejoin the cluster or if resources fail to start, check the cluster logs and consult the Pacemaker release notes or user guides for known upgrade issues.

Complete cluster shutdown

In this approach, all cluster nodes and resources are shut down, the nodes are upgraded, and then the cluster is restarted. This approach is necessary if the Pacemaker and Corosync versions do not support a mixed-version configuration.

1. Confirm that the cluster is in an optimal state, with each BeeGFS service running on its preferred node. Refer to Examine the state of the cluster for details.
2. Shut down the cluster software (Pacemaker and Corosync) on all nodes.

Depending on the cluster size, it may take seconds or minutes for the entire cluster to stop.

```
pcs cluster stop --all
```

3. Once cluster services have shut down on all nodes, upgrade the Pacemaker, Corosync, and pcs packages on each node according to your requirements.

Package manager commands will vary by operating system. The following commands are for systems running RHEL 8 and onward.

```
dnf update pacemaker-<version>
```

```
dnf update corosync-<version>
```

```
dnf update pcs-<version>
```

4. After upgrading all nodes, start the cluster software on all nodes:

```
pcs cluster start --all
```

5. If the `pcs` package was updated, reauthenticate each node in the cluster:

```
pcs host auth <HOSTNAME>
```

6. Finally, run `pcs status` and verify the cluster is healthy and the `Current DC` reports the correct Pacemaker version.

> If the `Current DC` reports 'mixed-version', then a node in the cluster is still running with the previous Pacemaker version and needs to be upgraded.

Update file node adapter firmware

Follow these steps to update the file node's ConnectX-7 adapters to the latest firmware.

Overview

Updating the ConnectX-7 adapter firmware may be required to support a new MLNX_OFED driver, enable new features, or fix bugs. This guide will use NVIDIA's `mlxfwmanager` utility for adapter updates due to its ease of use and efficiency.

Upgrade considerations

This guide covers two approaches to updating ConnectX-7 adapter firmware: a rolling update and a two-node cluster update. Choose the appropriate update approach according to your cluster's size. Before performing firmware updates, verify that:

- A supported MLNX_OFED driver is installed, refer to the technology requirements.
- Valid backups exist for your BeeGFS filesystem and Pacemaker cluster configuration.
- The cluster is in a healthy state.

Firmware update preparation

It is recommended to use NVIDIA's `mlxfwmanager` utility to update a node's adapter firmware, which is bundled with NVIDIA's MLNX_OFED driver. Prior to starting the updates, download the adapter's firmware image from NVIDIA's support site and store it on each file node.

For Lenovo ConnectX-7 adapters, use the `mlxfwmanager_LES` tool, which is available on NVIDIA's OEM firmware page.

Rolling update approach

This approach is recommended for any HA cluster with more than two nodes. This approach involves updating adapter firmware on one file node at a time, allowing the HA cluster to keep servicing requests, though it is recommended to avoid servicing I/O during this time.

1. Confirm that the cluster is in an optimal state, with each BeeGFS service running on its preferred node. Refer to Examine the state of the cluster for details.
2. Choose a file node to update and place it into standby mode, which drains (or moves) all BeeGFS services from that node:

```
pcs node standby <HOSTNAME>
```

3. Verify the node's services have drained by running:

```
pcs status
```

   Verify no services are reporting as `Started` on the node in standby.

   Depending on cluster size, it may take seconds or minutes for BeeGFS services to move to the sister node. If a BeeGFS service fails to start on the sister node, refer to the Troubleshooting Guides.

4. Update the adapter firmware using `mlxfwmanager`.

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

   Note the `PCI Device Name` for each adapter receiving firmware updates.

5. Reset each adapter using the `mlxfwreset` utility to apply the new firmware.

   Some firmware updates may require a reboot to apply the update. Refer to NVIDIA's mlxfwreset limitations for guidance. If a reboot is required, perform a reboot instead of resetting the adapters.

   a. Stop the opensm service:

```
systemctl stop opensm
```

b. Execute the following command for each `PCI Device Name` previously noted.

```
mlxfwreset -d <pci_device_name> reset -y
```

c. Start the opensm service:

```
systemctl start opensm
```

d. Restart the `eseries_nvme_ib.service`.

```
systemctl restart eseries_nvme_ib.service
```

e. Verify the E-Series storage array's volumes are present.

```
multipath -ll
```

1. Run `ibstat` and verify all adapters are running at the desired firmware version:

```
ibstat
```

2. Start Pacemaker cluster services on the node:

```
pcs cluster start <HOSTNAME>
```

3. Bring the node out of standby:

```
pcs node unstandby <HOSTNAME>
```

4. Relocate all BeeGFS services back to their preferred node:

```
pcs resource relocate run
```

Repeat these steps for each file node in the cluster until all adapters have been updated.

Two node cluster update approach

This approach is recommended for HA clusters with only two nodes. This approach is similar to a rolling update but includes additional steps to prevent service downtime when one node's cluster services are stopped.

1. Confirm that the cluster is in an optimal state, with each BeeGFS service running on its preferred node. Refer to Examine the state of the cluster for details.
2. Choose a file node to update and place the node in standby mode, which drains (or moves) all BeeGFS services from that node:

```
pcs node standby <HOSTNAME>
```

3. Verify the node's resources have drained by running:

```
pcs status
```

   Verify no services are reporting as `Started` on the node in standby.

   Depending on cluster size, it may take seconds or minutes for BeeGFS services to report as `Started` on the sister node. If a BeeGFS service fails to start, refer to the Troubleshooting Guides.

4. Place the cluster into maintenance mode.

```
pcs property set maintenance-mode=true
```

5. Update the adapter firmware using `mlxfwmanager`.

```
mlxfwmanager -i <path/to/firmware.bin> -u
```

   Note the `PCI Device Name` for each adapter receiving firmware updates.

6. Reset each adapter using the `mlxfwreset` utility to apply the new firmware.

   Some firmware updates may require a reboot to apply the update. Refer to NVIDIA's mlxfwreset limitations for guidance. If a reboot is required, perform a reboot instead of resetting the adapters.

   a. Stop the opensm service:

```
systemctl stop opensm
```

   b. Execute the following command for each `PCI Device Name` previously noted.

```
mlxfwreset -d <pci_device_name> reset -y
```

c. Start the opensm service:

```
systemctl start opensm
```

7. Run `ibstat` and verify all adapters are running at the desired firmware version:

```
ibstat
```

8. Start Pacemaker cluster services on the node:

```
pcs cluster start <HOSTNAME>
```

9. Bring the node out of standby:

```
pcs node unstandby <HOSTNAME>
```

10. Take the cluster out of maintenance mode.

```
pcs property set maintenance-mode=false
```

11. Relocate all BeeGFS services back to their preferred node:

```
pcs resource relocate run
```

Repeat these steps for each file node in the cluster until all adapters have been updated.

Upgrade E-Series storage array

Follow these steps to upgrade the HA cluster's E-Series storage array's components.

Overview

Keeping your HA cluster's NetApp E-Series storage arrays up-to-date with the latest firmware ensures optimal performance and improved security. Firmware updates for the storage array are applied through SANtricity OS, NVSRAM, and drive firmware files.

> While the storage arrays can be upgraded with the HA cluster online, it is recommended to place the cluster into maintenance mode for all upgrades.

Block node upgrade steps

The following steps outline how to update the storage arrays's firmware using the `Netapp_Eseries.Santricity` Ansible collection. Before proceeding, review the Upgrade considerations for updating E-Series systems.

> Upgrading to SANtricity OS 11.80 or later releases is only possible from 11.70.5P1. The storage array must first be upgraded to 11.70.5P1 before applying further upgrades.

1. Validate your Ansible control node is using the latest Santricity Ansible Collection.
   - For collection upgrades with access to Ansible Galaxy, run the following command:

     ```
     ansible-galaxy collection install netapp_eseries.santricity --upgrade
     ```

   - For offline upgrades, download the collection tarball from Ansible Galaxy, transfer it to your control node, and execute:

     ```
     ansible-galaxy collection install netapp_eseries-santricity-
     <VERSION>.tar.gz --upgrade
     ```

     See Installing Collections for more information.

2. Obtain the latest firmware for your storage array and drives.
   a. Download the firmware files.
      - **SANtricity OS and NVSRAM:** Navigate to the NetApp support site and download the latest release of SANtricity OS and NVSRAM for your storage array model.
      - **Drive Firmware:** Navigate to the E-Series disk firmware site and download the latest firmware for each of your storage array's drive models.
   b. Store the SANtricity OS, NVSRAM, and drive firmware files in your Ansible control node's `<inventory_directory>/packages` directory.
3. If necessary, update your cluster's Ansible inventory files to include all storage arrays (block nodes) requiring updates. For guidance, see the Ansible Inventory Overview section.
4. Ensure the cluster is in an optimal state with each BeeGFS service on its preferred node. Refer to Examine the state of the cluster for details.
5. Place the cluster in maintenance mode following the instructions in Place the cluster in maintenance mode.
6. Create a new Ansible playbook named `update_block_node_playbook.yml`. Populate the playbook with the following content, replacing the Santricity OS, NVSRAM, and drive firmware versions to your desired upgrade path:

```
- hosts: eseries_storage_systems
  gather_facts: false
  any_errors_fatal: true
  collections:
    - netapp_eseries.santricity
  vars:
    eseries_firmware_firmware: "packages/<SantricityOS>.dlp"
    eseries_firmware_nvsram: "packages/<NVSRAM>.dlp"
    eseries_drive_firmware_firmware_list:
      - "packages/<drive_firmware>.dlp"
    eseries_drive_firmware_upgrade_drives_online: true

  tasks:
    - name: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

7. To start the updates, execute the following command from your Ansible control node:

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

8. After the playbook completes, verify each storage array is in an optimal state.
9. Move the cluster out of maintenance mode and validate the cluster is in an optimal state with each BeeGFS service is on its preferred node.

Copyright information

Trademark information