Deploy the BeeGFS file system

BeeGFS on NetApp with E-Series Storage

NetApp
March 21, 2024

Table of Contents

Deploy the BeeGFS file system 1
Ansible Playbook Overview 1
Deploy the BeeGFS HA cluster 2
Deploy BeeGFS clients 5
Verify the BeeGFS deployment 10

Deploy the BeeGFS file system

Ansible Playbook Overview

Deploying and managing BeeGFS HA clusters using Ansible.

Overview

The previous sections walked through the steps needed to build an Ansible inventory representing a BeeGFS HA cluster. This section introduces the Ansible automation developed by NetApp to deploy and manage the cluster.

Ansible: Key Concepts

Before proceeding, it is helpful to be familiar with a few key Ansible concepts:

- Tasks to execute against an Ansible inventory are defined in what is known as a **playbook**.
  - Most tasks in Ansible are designed to be **idempotent**, meaning they can be run multiple times to verify the desired configuration/state is still applied without breaking things or making unnecessary updates.
- The smallest unit of execution in Ansible is a **module**.
  - Typical playbooks use multiple modules.
    - Examples: Download a package, update a config file, start/enable a service.
  - NetApp distributes modules to automate NetApp E-Series systems.
- Complex automation is better packaged as a role.
  - Essentially a standard format to distribute a reusable playbook.
  - NetApp distributes roles for Linux hosts and BeeGFS file systems.

BeeGFS HA role for Ansible: Key Concepts

All the automation needed to deploy and manage each version of BeeGFS on NetApp is packaged as an Ansible role and distributed as part of the NetApp E-Series Ansible Collection for BeeGFS:

- This role can be thought of as somewhere between an **installer** and modern **deployment/management** engine for BeeGFS.
  - Applies modern infrastructure as code practices and philosophies to simplify managing storage infrastructure at any scale.
  - Similar to how the Kubespray project allows users to deploy/maintain an entire Kubernetes distribution for scale out compute infrastructure.
- This role is the **software-defined** format NetApp uses to package, distribute, and maintain BeeGFS on NetApp solutions.
  - Strive to create an “appliance-like” experience without needing to distribute an entire Linux distribution or large image.
  - Includes NetApp authored Open Cluster Framework (OCF) compliant cluster resource agents for custom BeeGFS targets, IP addresses, and monitoring that provide intelligent Pacemaker/BeeGFS integration.

- This role is not simply deployment "automation" and is intended to manage the entire file system lifecycle including:
  - Applying per-service or cluster-wide configuration changes and updates.
  - Automating cluster healing and recovery after hardware issues are resolved.
  - Simplifying performance tuning with default values set based on extensive testing with BeeGFS and NetApp volumes.
  - Verifying and correcting configuration drift.

NetApp also provides an Ansible role for BeeGFS clients, that can optionally be used to install BeeGFS and mount file systems to compute/GPU/login nodes.

Deploy the BeeGFS HA cluster

Specify what tasks should run to deploy the BeeGFS HA cluster using a playbook.

Overview

This section covers how to assemble the standard playbook used to deploy/manage BeeGFS on NetApp.

Steps

Create the Ansible Playbook

Create the file `playbook.yml` and populate it as follows:

1. First define a set of tasks (commonly referred to as a play) that should only run on NetApp E-Series block nodes. We use a pause task to prompt before running the installation (to avoid accidental playbook runs), then import the `nar_santricity_management` role. This role handles applying any general system configuration defined in `group_vars/eseries_storage_systems.yml` or individual `host_vars/<BLOCK NODE>.yml` files.

```
- hosts: eseries_storage_systems
  gather_facts: false
  collections:
    - netapp_eseries.santricity
  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: Configure NetApp E-Series block nodes.
      import_role:
        name: nar_santricity_management
```

2. Define the play that will run against all file and block nodes:

```
- hosts: all
  any_errors_fatal: true
  gather_facts: false
  collections:
    - netapp_eseries.beegfs
```

3. Within this play we can optionally define a set of "pre-tasks" that should run before deploying the HA cluster. This can be useful to verify/install any prerequisites like Python. We can also inject any pre-flight checks, for example verifying the provided Ansible tags are supported:

```
  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 }}"
```

4. Lastly, this play imports the BeeGFS HA role for the version of BeeGFS you want to deploy:

```
  tasks:
    - name: Verify the BeeGFS HA cluster is properly deployed.
      import_role:
        name: beegfs_ha_7_3 # Alternatively specify: beegfs_ha_7_2.
```

> A BeeGFS HA role is maintained for each supported major.minor version of BeeGFS. This allows users to choose when they want to upgrade major/minor versions. Currently either BeeGFS 7.3.x (`beegfs_7_3`) or BeeGFS 7.2.x (`beegfs_7_2`) are supported. By default both roles will deploy the latest BeeGFS patch version at the time of release, though users can opt to override this and deploy the latest patch if desired. Refer to the latest upgrade guide for more details.

5. Optional: If you wish to define additional tasks, keep in mind if the tasks should be directed to `all` hosts (including the E-Series storage systems) or only the file nodes. If needed define a new play specifically targeting file nodes using `- hosts: ha_cluster`.

Click here for an example of a complete playbook file.

Install the NetApp Ansible Collections

The BeeGFS collection for Ansible and all dependencies are maintained on Ansible Galaxy. On your Ansible control node run the following command to install the latest version:

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

Though not typically recommended, it is also possible to install a specific version of the collection:

```
ansible-galaxy collection install netapp_eseries.beegfs:
==<MAJOR>.<MINOR>.<PATCH>
```

Run the Playbook

From the directory on your Ansible control node containing the `inventory.yml` and `playbook.yml` files, run the playbook as follows:

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

Based on the size of the cluster the initial deployment can take 20+ minutes. If the deployment fails for any reason, simply correct any issues (e.g., miscabling, node wasn't started, etc.), and restart the Ansible playbook.

When specifying common file node configuration, if you choose the default option to have Ansible automatically manage connection based authentication, a `connAuthFile` used as a shared secret can now be found at `<playbook_dir>/files/beegfs/<sysMgmtdHost>_connAuthFile` (by default). Any clients needing to access the file system will need to use this shared secret. This is handled automatically if clients are configured using the BeeGFS client role.

Deploy BeeGFS clients

Optionally, Ansible can be used to configure BeeGFS clients and mount the file system.

Overview

Accessing BeeGFS file systems requires installing and configuring the BeeGFS client on each node that needs to mount the file system. This section documents how to perform these tasks using the available Ansible role.

Steps

Create the Client Inventory File

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>
   ```

3. Optionally include one of the following if you want to use the NetApp E-Series Host Collection's roles to configure InfiniBand or Ethernet interfaces for clients to connect to BeeGFS file nodes:

   a. If the network type is InfiniBand (using IPoIB):

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

b. If the network type is RDMA over Converged Ethernet (RoCE):

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

c. If the network type is Ethernet (TCP only, no RDMA):

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

4. Create a new file `client_inventory.yml` and specify the user Ansible should use to connect to each client, and the password Ansible should use for privilege escalation (this requires `ansible_ssh_user` be root, or have sudo privileges):

```
# BeeGFS client inventory.
all:
  vars:
    ansible_ssh_user: <USER>
    ansible_become_password: <PASSWORD>
```

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.

5. In the `client_inventory.yml` file, list all hosts that should be configured as BeeGFS clients under the `beegfs_clients` group, and then refer to the inline comments and uncomment any additional configuration required to build the BeeGFS client kernel module on your system:

```
children:
    # Ansible group representing all BeeGFS clients:
    beegfs_clients:
      hosts:
        <CLIENT HOSTNAME>:
        # Additional clients as needed.

      vars:
        # OPTION 1: If you're using the Mellanox 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.
```

When using the Mellanox 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.

6. In the `client_inventory.yml` file, list the BeeGFS file systems you want mounted under any previously defined `vars`:

```
        beegfs_client_mounts:
          - sysMgmtdHost: <IP ADDRESS>  # 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
        # Specify additional file system mounts for this or other file
systems.
```

7. As of BeeGFS 7.2.7 and 7.3.1 connection authentication must be configured or explicitly disabled. Depending how you choose to configure connection based authentication when specifying common file node configuration, you may need to adjust your client configuration:
   a. By default the HA cluster deployment will automatically configure connection authentication, and generate a `connauthfile` that will be placed/maintained on the Ansible control node at `<INVENTORY>/files/beegfs/<sysMgmtdHost>_connAuthFile`. By default the BeeGFS client role is setup to read/distribute this file to the clients defined in `client_inventory.yml`, and no additional action is needed.
      i. For advanced options refer to the full list of defaults included with the BeeGFS client role.
   b. If you choose to specify a custom secret with `beegfs_ha_conn_auth_secret` specify it in the `client_inventory.yml` file as well:

      ```
      beegfs_ha_conn_auth_secret: <SECRET>
      ```

   c. If you choose to disable connection based authentication entirely with `beegfs_ha_conn_auth_enabled`, specify that in the `client_inventory.yml` file as well:

```
beegfs_ha_conn_auth_enabled: false
```

For a full list of supported parameters and additional details refer to the full BeeGFS client documentation. For a complete example of a client inventory click here.

Create the BeeGFS Client Playbook File

1. Create a new file `client_playbook.yml`

```
# BeeGFS client playbook.
- hosts: beegfs_clients
  any_errors_fatal: true
  gather_facts: true
  collections:
    - netapp_eseries.beegfs
    - netapp_eseries.host
  tasks:
```

2. Optional: If you want to use the NetApp E-Series Host Collection's roles to configure interfaces for clients to connect to BeeGFS file systems, import the role corresponding with the interface type you are configuring:
   a. If you are using are using InfiniBand (IPoIB):

```
    - name: Ensure IPoIB is configured
      import_role:
        name: ipoib
```

   b. If you are using are using RDMA over Converged Ethernet (RoCE):

```
    - name: Ensure IPoIB is configured
      import_role:
        name: roce
```

   c. If you are using are using Ethernet (TCP only, no RDMA):

```
    - name: Ensure IPoIB is configured
      import_role:
        name: ip
```

3. Lastly import the BeeGFS client role to install the client software and setup the file system mounts:

```
    # REQUIRED: Install the BeeGFS client and mount the BeeGFS file
system.
    - name: Verify the BeeGFS clients are configured.
      import_role:
        name: beegfs_client
```

For a complete example of a client playbook click here.

Run the BeeGFS Client Playbook

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

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

Verify the BeeGFS deployment

Verify the file system deployment before placing the system in production.

Overview

Before you place the BeeGFS file system in production, perform a few verification checks.

Steps

1. Login to any client and run the following to ensure all expected nodes are present/reachable, and there are no inconsistencies or other issues reported:

```
beegfs-fsck --checkfs
```

2. Shutdown the entire cluster, then restart it. From any file node run the following:

```
pcs cluster stop --all # Stop the cluster on all file nodes.
pcs cluster start --all  # Start the cluster on all file nodes.
pcs status # Verify all nodes and services are started and no failures
are reported (the command may need to be reran a few times to allow time
for all services to start).
```

3. Place each node in standby and verify BeeGFS services are able to failover to secondary node(s). To do this login to any of the file nodes and run the following:

```
pcs status # Verify the cluster is healthy at the start.
pcs node standby <FILE NODE HOSTNAME> # Place the node under test in
standby.
pcs status # Verify services are started on a secondary node and no
failures are reported.
pcs node unstandby <FILE NODE HOSTNAME> # Take the node under test out
of standby.
pcs status # Verify the file node is back online and no failures are
reported.
pcs resource relocate run # Move all services back to their preferred
nodes.
pcs status # Verify services have moved back to the preferred node.
```

4. Use performance benchmarking tools such as IOR and MDTest to verify the file system performance meets expectations. Examples of common tests and parameters used with BeeGFS can be found in the Design Verification section of the BeeGFS on NetApp Verified Architecture.

Additional tests should be performed based on the acceptance criteria defined for a particular site/installation.

Copyright information

Trademark information