



# **Implementierung von Software**

## **BeeGFS on NetApp with E-Series Storage**

NetApp  
January 27, 2026

This PDF was generated from <https://docs.netapp.com/de-de/beegfs/second-gen/beegfs-deploy-setup-nodes.html> on January 27, 2026. Always check docs.netapp.com for the latest.

# Inhalt

Implementierung von Software .....	1
Einrichten von Datei-Nodes und Block-Nodes .....	1
Richten Sie die Datei-Nodes ein .....	1
Richten Sie Block-Nodes ein .....	2
Optimieren Sie die System-Einstellungen des File Node für die Performance .....	2
Über die UEFI-Schnittstelle können Sie die Systemeinstellungen anpassen .....	2
Verwenden Sie die Redfish-API, um die Systemeinstellungen anzupassen .....	4
Richten Sie einen Ansible-Steuerungsknoten ein .....	4
Erstellen des Ansible-Inventars .....	6
Schritt 1: Konfiguration für alle Bausteine definieren .....	6
Schritt: Konfiguration für einzelne Datei- und Block-Nodes definieren .....	7
Schritt 3: Definieren Sie die Konfiguration, die für alle Datei- und Block-Nodes gelten soll .....	9
Schritt 4: Definieren Sie die Konfiguration, die für alle Datei-Knoten gelten soll .....	10
Schritt 5: Definieren Sie die Konfiguration für den gemeinsamen Block-Node .....	16
Definieren Sie den Ansible-Bestand für BeeGFS-Bausteine .....	18
Schritt: Die Ansible-Bestandsdatei erstellen .....	19
Schritt: Inventar für einen Management-, Metadaten- und Storage-Baustein konfigurieren .....	19
Schritt 3: Konfigurieren Sie den Bestand für einen Baustein Metadaten + Speicher .....	25
Schritt 4: Konfigurieren Sie den Bestand für einen nur-Storage-Baustein .....	29
BeeGFS bereitstellen .....	32
Konfigurieren Sie BeeGFS-Clients .....	35

# Implementierung von Software

## Einrichten von Datei-Nodes und Block-Nodes

Während die meisten Software-Konfigurationsaufgaben mithilfe der von NetApp zur Verfügung gestellten Ansible Sammlungen automatisiert werden, müssen Sie das Netzwerk auf dem Baseboard Management Controller (BMC) jedes Servers konfigurieren und den Management-Port auf jedem Controller konfigurieren.

### Richten Sie die Datei-Nodes ein

1. Konfigurieren Sie das Netzwerk auf dem Baseboard Management Controller (BMC) jedes Servers.

Informationen zum Konfigurieren der Netzwerkkonfiguration für die validierten Lenovo SR665 V3-Dateiknoten finden Sie unter "[Lenovo ThinkSystem Dokumentation](#)".



Ein Baseboard Management Controller (BMC), der manchmal als Service-Prozessor bezeichnet wird, ist der generische Name für die Out-of-Band-Management-Funktion, die in verschiedenen Server-Plattformen integriert ist, die Remote-Zugriff bieten können, selbst wenn das Betriebssystem nicht installiert ist oder nicht zugänglich ist. Anbieter vermarkten diese Funktionalität in der Regel mit ihrem eigenen Branding. Auf dem Lenovo SR665 wird beispielsweise der BMC als *Lenovo XClarity Controller (XCC)* bezeichnet.

2. Konfigurieren Sie die Systemeinstellungen für maximale Performance.

Sie konfigurieren die Systemeinstellungen über das UEFI-Setup (früher BIOS) oder über die Redfish APIs, die von vielen BMCs bereitgestellt werden. Die Systemeinstellungen variieren je nach Servermodell, das als Dateiknoten verwendet wird.

Informationen zum Konfigurieren der Systemeinstellungen für die validierten Lenovo SR665 V3-Dateiknoten finden Sie unter "[Passen Sie die Systemeinstellungen für die Performance an](#)".

3. Installieren Sie Red Hat Enterprise Linux (RHEL) 9.4 und konfigurieren Sie den Hostnamen und den Netzwerkport, die zur Verwaltung des Betriebssystems verwendet werden, einschließlich der SSH-Konnektivität vom Ansible-Steuerknoten.

Konfigurieren Sie derzeit keine IPs auf einem der InfiniBand-Ports.



Die nachfolgenden Abschnitte gehen davon aus, dass die Hostnamen sequenziell nummeriert sind (z. B. h1-HN), und beziehen sich auf Aufgaben, die auf ungeraden oder gar nummerierten Hosts ausgeführt werden sollten.

4. Verwenden Sie den Red Hat Subscription Manager, um das System zu registrieren und zu abonnieren, damit die erforderlichen Pakete aus den offiziellen Red Hat-Repositories installiert werden können und um Updates auf die unterstützte Version von Red Hat zu beschränken: `subscription-manager release --set=9.4`. Anweisungen hierzu finden Sie unter "[Registrieren und Abonnieren eines RHEL Systems](#)" und "[Einschränken von Aktualisierungen](#)".
5. Aktivieren Sie das Red hat Repository mit den für hohe Verfügbarkeit erforderlichen Paketen.

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

6. Aktualisieren Sie alle HCA-Firmware auf die in Verwendung des ["Aktualisiert die Datei-Node-Adapter-Firmware"](#) Handbuchs empfohlene Version ["Technologieanforderungen erfüllt"](#).

## Richten Sie Block-Nodes ein

Richten Sie die EF600 Block-Nodes ein, indem Sie den Managementport pro Controller konfigurieren.

1. Konfigurieren Sie den Managementport an jedem EF600 Controller.

Anweisungen zum Konfigurieren von Ports finden Sie im ["E-Series Documentation Center"](#).

2. Legen Sie optional den Speicher-Array-Namen für jedes System fest.

Durch das Festlegen eines Namens kann es einfacher sein, in den nachfolgenden Abschnitten auf jedes System zu verweisen. Anweisungen zum Festlegen des Arraynamens finden Sie im ["E-Series Documentation Center"](#).



Folgende Themen setzen voraus, dass die Namen der Speicherarrays nacheinander nummeriert sind (z. B. c1 - CN), und beziehen sich auf die Schritte, die auf ungeraden und nicht gerade nummerierten Systemen ausgeführt werden sollten.

## Optimieren Sie die System-Einstellungen des File Node für die Performance

Um die Leistung zu maximieren, empfehlen wir, die Systemeinstellungen auf dem Servermodell zu konfigurieren, das Sie als Dateiknoten verwenden.

Die Systemeinstellungen variieren je nach Servermodell, das Sie als Dateiknoten verwenden. In diesem Thema wird beschrieben, wie die Systemeinstellungen für die validierten Lenovo ThinkSystem SR665-Serverdateiknoten konfiguriert werden.

### Über die UEFI-Schnittstelle können Sie die Systemeinstellungen anpassen

Die System-Firmware des Lenovo SR665 V3-Servers enthält zahlreiche Tuning-Parameter, die über die UEFI-Schnittstelle eingestellt werden können. Diese Optimierungsparameter können sich auf alle Aspekte der Serverfunktionen und die Leistung des Servers auswirken.

Passen Sie unter **UEFI Setup > Systemeinstellungen** die folgenden Systemeinstellungen an:

#### Menü „Betriebsmodus“

Systemeinstellung	Wechseln Sie zu
Betriebsmodus	Individuell

Systemeinstellung	Wechseln Sie zu
CTDP	Manuell
CTDP-Handbuch	350
Maximale Leistung Des Pakets	Manuell
Effizienzmodus	Deaktivieren
Global-Cstate-Control	Deaktivieren
SOC P-Staaten	P0
DF-C-Staaten	Deaktivieren
P-Zustand	Deaktivieren
Speicherabschaltstrom Aktivieren	Deaktivieren
NUMA-Knoten pro Socket	NPS1

#### Menü „Geräte“ und „E/A-Anschlüsse“

Systemeinstellung	Wechseln Sie zu
IOMMU	Deaktivieren

#### Ein-/aus-Menü

Systemeinstellung	Wechseln Sie zu
PCIe-Power-Brake	Deaktivieren

#### Menü „Prozessoren“

Systemeinstellung	Wechseln Sie zu
Global C-State Control	Deaktivieren
DF-C-Staaten	Deaktivieren
SMT-Modus	Deaktivieren
CPPC	Deaktivieren

## Verwenden Sie die Redfish-API, um die Systemeinstellungen anzupassen

Zusätzlich zur Verwendung von UEFI Setup können Sie die Redfish API verwenden, um Systemeinstellungen zu ändern.

```
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_ChoseOperatingMode": "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"
}
}
'
```

Ausführliche Informationen zum Schema Redfish finden Sie im ["DMTF-Website"](#).

## Richten Sie einen Ansible-Steuerungsknoten ein

Zum Einrichten eines Ansible-Steuerknotens müssen Sie eine virtuelle oder physische Maschine mit Netzwerkzugriff auf alle Datei- und Block-Nodes zuweisen, die für die BeeGFS auf NetApp Lösung implementiert werden.

Eine Liste der empfohlenen Paketversionen finden Sie im ["Technische Anforderungen"](#). Die folgenden Schritte wurden auf Ubuntu 22.04 getestet. Für spezifische Schritte zu Ihrer bevorzugten Linux-Distribution, siehe ["Ansible-Dokumentation"](#).

1. Installieren Sie über Ihren Ansible Control Node die folgenden Pakete für Python und Python Virtual Environment.

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

2. Erstellen Sie eine virtuelle Python-Umgebung.

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

3. Aktivieren Sie die virtuelle Umgebung.

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

4. Installieren Sie die erforderlichen Python-Pakete in der aktivierten virtuellen Umgebung.

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

5. Installieren Sie die BeeGFS-Sammlung mit Ansible Galaxy.

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

6. Überprüfen Sie, ob die installierten Versionen von Ansible, Python und BeeGFS-Sammlung mit der übereinstimmen "[Technische Anforderungen](#)".

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

7. Richten Sie passwortloses SSH ein, damit Ansible vom Ansible-Steuerungsknoten aus auf die Remote-BeeGFS-Datei-Nodes zugreifen kann.

- a. Generieren Sie auf dem Ansible-Steuerungsknoten, falls erforderlich, ein Paar öffentlicher Schlüssel.

```
ssh-keygen
```

- b. Richten Sie passwortloses SSH für jeden der Dateiknoten ein.

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



Richten Sie bei den Blockknoten eine passwortlose SSH ein. Dies wird weder unterstützt noch erforderlich.

# Erstellen des Ansible-Inventars

Um die Konfiguration für Datei- und Block-Nodes zu definieren, erstellen Sie einen Ansible-Bestand für das BeeGFS-Dateisystem, das bereitgestellt werden soll. Der Bestand umfasst Hosts, Gruppen und Variablen, die das gewünschte BeeGFS-Dateisystem beschreiben.

## Schritt 1: Konfiguration für alle Bausteine definieren

Legen Sie die Konfiguration fest, die für alle Bausteine gilt, unabhängig davon, welches Konfigurationsprofil Sie für sie einzeln anwenden können.

### Bevor Sie beginnen

- Wählen Sie ein Subnetz-Adressierungsschema für Ihre Bereitstellung aus. Aufgrund der im aufgeführten Vorteile "[Softwarearchitektur](#)" wird empfohlen, ein einziges Subnetz-Adressierungsschema zu verwenden.

### Schritte

1. Geben Sie auf dem Ansible-Steuerungsknoten ein Verzeichnis an, das Sie zum Speichern der Bestands- und Playbook-Dateien in Ansible verwenden möchten.

Sofern nicht anders angegeben, werden alle in diesem Schritt erstellten Dateien und Verzeichnisse und die folgenden Schritte relativ zu diesem Verzeichnis erstellt.

2. Folgende Unterverzeichnisse erstellen:

```
host_vars
```

```
group_vars
```

```
packages
```

3. Erstellen Sie ein Unterverzeichnis für Cluster-Passwörter und sichern Sie die Datei durch Verschlüsselung mit Ansible Vault (siehe "[Verschlüsseln von Inhalten mit Ansible Vault](#)"):
  - a. Erstellen Sie das Unterverzeichnis `group_vars/all`.
  - b. Erstellen Sie im `group_vars/all` Verzeichnis eine Passwortdatei mit der Bezeichnung `passwords.yml`.
  - c. Füllen Sie den `passwords.yml` file mit den folgenden Angaben aus, und ersetzen Sie alle Benutzernamen- und Kennwortparameter entsprechend Ihrer Konfiguration:



```

# 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. Führen Sie aus `ansible-vault encrypt passwords.yml`, und legen Sie ein Vault-Kennwort fest, wenn Sie dazu aufgefordert werden.

## Schritt: Konfiguration für einzelne Datei- und Block-Nodes definieren

Legen Sie die Konfiguration für einzelne Datei-Nodes und einzelne Baustein-Nodes fest.

1. Unter `host_vars/` Erstellen Sie für jeden BeeGFS-Dateiknoten eine Datei mit dem Namen `<HOSTNAME>.yaml` Mit dem folgenden Inhalt, besondere Aufmerksamkeit auf die Notizen über den Inhalt für BeeGFS Cluster-IPs und Host-Namen enden in ungerade oder gerade Zahlen.

Zunächst stimmen die Schnittstellennamen der Dateiknoten mit dem überein, was hier aufgeführt ist (z. B. `ib0` oder `ibs1f0`). Diese benutzerdefinierten Namen werden in konfiguriert [die für alle Datei-Knoten gelten soll](#).

```

ansible_host: "<MANAGEMENT_IP>"
eseries_ipoib_interfaces:  # Used to configure BeeGFS cluster IP
addresses.
  - name: ilb
    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>
  - <ilb_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

```



Wenn Sie bereits das BeeGFS-Cluster implementiert haben, müssen Sie das Cluster beenden, bevor Sie statisch konfigurierte IP-Adressen hinzufügen oder ändern, einschließlich Cluster-IPs und IPs für NVMe/IB. Dies ist erforderlich, damit diese Änderungen ordnungsgemäß wirksam werden und Cluster-Vorgänge nicht unterbrechen.

2. Unter `host_vars/` Erstellen Sie für jeden BeeGFS-Block-Knoten eine Datei mit dem Namen `<HOSTNAME>.yaml` Und geben Sie den folgenden Inhalt ein.

Achten Sie besonders auf die Hinweise zum Inhalt, die für Speicher-Array-Namen ausgefüllt werden müssen, die mit ungeraden oder geraden Zahlen enden.

Erstellen Sie für jeden Block-Node eine Datei, und geben Sie den an `<MANAGEMENT_IP>` Für einen der beiden Controller (normalerweise 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
```

### Schritt 3: Definieren Sie die Konfiguration, die für alle Datei- und Block-Nodes gelten soll

Unter können Sie die gemeinsame Konfiguration für eine Gruppe von Hosts definieren `group_vars` In einem

Dateinamen, der der Gruppe entspricht. Dadurch wird verhindert, dass eine gemeinsame Konfiguration an mehreren Orten wiederholt wird.

### Über diese Aufgabe

Hosts können sich in mehr als einer Gruppe befinden. Ansible zur Laufzeit wählt Ansible aus, welche Variablen auf Basis seiner variablen Rangfolge für einen bestimmten Host gelten. (Weitere Informationen zu diesen Regeln finden Sie in der Ansible-Dokumentation für "[Variablen verwenden](#)".)

Host-zu-Gruppe-Zuweisungen werden in der tatsächlichen Ansible-Bestandsdatei definiert, die gegen Ende dieses Vorgangs erstellt wird.

### Schritt

In Ansible können alle Konfigurationen, die auf alle Hosts angewendet werden sollen, in einer Gruppe mit dem Namen definiert werden `All`. Erstellen Sie die Datei `group_vars/all.yml` Mit folgenden Inhalten:

```
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"
```

## Schritt 4: Definieren Sie die Konfiguration, die für alle Datei-Knoten gelten soll

Die gemeinsame Konfiguration für Dateiknoten ist in einer Gruppe mit dem Namen definiert `ha_cluster`. In den Schritten in diesem Abschnitt wird die Konfiguration erstellt, die in der enthalten sein sollte `group_vars/ha_cluster.yml` Datei:

### Schritte

1. Legen Sie oben in der Datei die Standardeinstellungen fest, einschließlich des Kennworts, das als verwendet werden soll `sudo` Benutzer auf den Datei-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"

```



Wenn der `ansible_ssh_user` bereits `root` ist, können Sie optional die auslassen und beim Ausführen des Playbook die `ansible_become_password` Option angeben `--ask -become-pass`.

2. Konfigurieren Sie optional einen Namen für den Hochverfügbarkeits-Cluster und geben Sie einen Benutzer für die Cluster-interne Kommunikation an.

Wenn Sie das private IP-Adressschema ändern, müssen Sie auch die Standardeinstellung aktualisieren `beegfs_ha_mgmt floating_ip`. Dies muss mit dem übereinstimmen, was Sie später für die BeeGFS Management Ressourcengruppe konfigurieren.

Geben Sie eine oder mehrere E-Mails an, die Warnmeldungen für Cluster-Ereignisse mit empfangen sollen `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

```



Während scheinbar redundant, `beegfs_ha_mgmtd_floating_ip` ist wichtig, wenn Sie das BeeGFS-Dateisystem über einen einzelnen HA-Cluster hinaus skalieren. Nachfolgende HA-Cluster werden ohne zusätzlichen BeeGFS-Managementservice bereitgestellt und Punkt am Managementservice des ersten Clusters.

3. Konfigurieren Sie einen Fechtagenten. (Weitere Informationen finden Sie unter ["Konfigurieren Sie Fechten in einem Red hat High Availability Cluster"](#).) Die folgende Ausgabe zeigt Beispiele für die Konfiguration gängiger Fencing-Agenten. Wählen Sie eine dieser Optionen.

Beachten Sie bei diesem Schritt Folgendes:

- Standardmäßig ist Fencing aktiviert, Sie müssen jedoch einen Fencing\_Agent\_ konfigurieren.
- Der <HOSTNAME> Angegeben in `pcmk_host_map` Oder `pcmk_host_list` Der Hostname in der Ansible-Bestandsaufnahme entspricht.
- Das BeeGFS-Cluster ohne Fencing wird insbesondere in der Produktion nicht unterstützt. Dies soll weitgehend sicherstellen, wenn BeeGFS-Services, einschließlich aller Ressourcenabhängigkeiten wie Blockgeräte, Failover aufgrund eines Problems durchführen, es besteht keine Möglichkeit des gleichzeitigen Zugriffs durch mehrere Nodes, die zu einer Beschädigung des Filesystems oder anderen unerwünschten oder unerwarteten Verhalten führen. Wenn das Fencing deaktiviert werden muss, lesen Sie die allgemeinen Hinweise in der BeeGFS HA-Rolle „erste Schritte“-Anleitung und „Set“ `beegfs_ha_cluster_crm_config_options["stonith-enabled"]` Mit FALSE innen `ha_cluster.yml`.
- Es sind mehrere Fencinggeräte auf Node-Ebene verfügbar, und die BeeGFS HA-Rolle kann jeden Fencingagenten konfigurieren, der im Red hat HA Package Repository verfügbar ist. Wenn möglich, verwenden Sie einen Zaunagenten, der über die unterbrechungsfreie Stromversorgung (USV) oder die Rack-Stromverteilereinheit (rPDU) arbeitet. Da einige Fencing-Agenten wie der Baseboard-Management-Controller (BMC) oder andere Lights-Out-Geräte, die in den Server integriert sind, möglicherweise nicht auf die Zaunanforderung unter bestimmten Ausfallszenarien reagieren.

```

### 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. Aktivieren Sie die empfohlene Performance-Optimierung im Linux-Betriebssystem.

Viele Benutzer finden die Standardeinstellungen für die Performance-Parameter zwar im Allgemeinen gut, Sie können jedoch optional die Standardeinstellungen für einen bestimmten Workload ändern. Daher sind diese Empfehlungen in die BeeGFS-Rolle enthalten, jedoch sind sie nicht standardmäßig aktiviert, um sicherzustellen, dass Benutzer die auf ihr Dateisystem angewendete Einstellung kennen.

Um das Performance-Tuning zu aktivieren, geben Sie Folgendes an:



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

5. (Optional) Sie können die Leistungsparameter im Linux-Betriebssystem nach Bedarf anpassen.

Eine umfassende Liste der verfügbaren Tuning-Parameter, die Sie anpassen können, finden Sie im Abschnitt Performance Tuning Defaults der BeeGFS HA-Rolle in "[E-Series BeeGFS GitHub-Website](#)". Die Standardwerte können für alle Knoten im Cluster in dieser Datei oder für die Datei eines einzelnen Knotens überschrieben werden `host_vars`.

6. Um vollständige 200 GB/HDR-Konnektivität zwischen Block- und Dateiknoten zu ermöglichen, verwenden Sie das OpenSM-Paket (Open Subnetz Manager) aus der NVIDIA Open Fabrics Enterprise Distribution (MLNX\_OFED). Die MLNX\_OFED-Version in der Liste wird im Lieferumfang der "[Anforderungen an den Datei-Node](#)" empfohlenen OpenSM-Pakete enthalten. Obwohl die Implementierung mit Ansible unterstützt wird, müssen Sie zuerst den MLNX\_OFED-Treiber auf allen Datei-Nodes installieren.
- a. Füllen Sie die folgenden Parameter in `group_vars/ha_cluster.yml` (Passen Sie Pakete nach Bedarf an):

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

7. Konfigurieren Sie die `udev` Regel zur Sicherstellung einer konsistenten Zuordnung von logischen InfiniBand-Port-IDs zu zugrunde liegenden PCIe-Geräten.

Der `udev` Die Regel muss für die PCIe-Topologie jeder Serverplattform, die als BeeGFS-Datei-Node verwendet wird, eindeutig sein.

Für verifizierte Dateiknoten folgende Werte verwenden:

```

### 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) Aktualisieren des Metadaten-Zielauswahlalgorithmus.

```

beegfs_ha_beegfs_meta_conf_ha_group_options:
    tuneTargetChooser: randomrobin

```



Während der Verifizierungstests `randomrobin` wurde in der Regel verwendet, um sicherzustellen, dass Testdateien während des Performance-Benchmarking gleichmäßig auf alle BeeGFS-Speicherziele verteilt wurden (weitere Informationen zu Benchmarking finden Sie auf der BeeGFS-Website für "[Benchmarking eines BeeGFS-Systems](#)"). Bei der realen Welt könnte dies dazu führen, dass sich die niedrigeren nummerierten Ziele schneller füllen als die höher nummerierten Ziele. Auslassung `randomrobin` und nur mit dem Standard `randomized` Der Wert zeigt sich, dass er eine gute Leistung bietet und gleichzeitig alle verfügbaren Ziele nutzt.

## Schritt 5: Definieren Sie die Konfiguration für den gemeinsamen Block-Node

Die gemeinsame Konfiguration für Block-Knoten wird in einer Gruppe mit dem Namen definiert `eseries_storage_systems`. In den Schritten in diesem Abschnitt wird die Konfiguration erstellt, die in der enthalten sein sollte `group_vars/ eseries_storage_systems.yml` Datei:

### Schritte

1. Setzen Sie die Ansible-Verbindung auf Local, geben Sie das Systemkennwort ein und geben Sie an, ob

SSL-Zertifikate verifiziert werden sollen. (Normalerweise verwendet Ansible SSH für die Verbindung zu gemanagten Hosts. Bei Storage-Systemen der NetApp E-Series, die als Block-Nodes verwendet werden, verwenden die Module JEDOCHE die REST-API für die Kommunikation.) Fügen Sie oben in der Datei Folgendes hinzu:

```
### 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. Installieren Sie die für Block-Nodes in aufgeführten Versionen, um eine optimale Performance zu gewährleisten "[Technische Anforderungen](#)".

Laden Sie die entsprechenden Dateien aus dem herunter "[NetApp Support Website](#)". Sie können sie entweder manuell aktualisieren oder sie in das einbeziehen `packages/` Verzeichnis des Ansible-Steuerknotens, und füllen Sie dann die folgenden Parameter in aus `eseries_storage_systems.yml` So führen Sie ein Upgrade mit Ansible durch:

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

3. Laden Sie die neueste Laufwerksfirmware herunter, die für die in Ihren Blockknoten installierten Laufwerke verfügbar ist, und installieren Sie sie im "[NetApp Support Website](#)". Sie können sie entweder manuell aktualisieren oder in das Verzeichnis des Ansible-Steuerknotens aufnehmen `packages/` . Dann füllen Sie die folgenden Parameter in aus `eseries_storage_systems.yml` , um das Upgrade mit Ansible auszuführen:

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



Einstellung `eseries_drive_firmware_upgrade_drives_online` Bis `false` Beschleunigt das Upgrade, sollte aber erst nach dem Einsatz von BeeGFS durchgeführt werden. Der Grund dafür ist, dass bei dieser Einstellung sämtliche I/O-Vorgänge auf den Laufwerken vor dem Upgrade angehalten werden müssen, um Applikationsfehler zu vermeiden. Obwohl ein Online-Laufwerk-Firmware-Upgrade vor der Konfiguration von Volumes noch schnell durchgeführt wird, empfehlen wir Ihnen, diesen Wert immer auf `true` zu setzen Um später Probleme zu vermeiden.

4. Nehmen Sie zur Optimierung der Leistung folgende Änderungen an der globalen Konfiguration vor:

```
# 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. Um eine optimale Bereitstellung und ein optimales Verhalten von Volumes zu gewährleisten, geben Sie folgende Parameter an:

```
# 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"
```



Der für angegebene Wert `eseries_storage_pool_usable_drives` Gibt einen spezifischen Block-Node der NetApp EF600 an und steuert die Reihenfolge, in der Laufwerke neuen Volume-Gruppen zugewiesen werden. Durch diese Bestellung wird sichergestellt, dass der I/O zu jeder Gruppe gleichmäßig über die Kanäle des Backend-Laufwerks verteilt wird.

## Definieren Sie den Ansible-Bestand für BeeGFS-Bausteine

Definieren Sie nach der Definition der allgemeinen Ansible-Bestandsstruktur die Konfiguration für jeden Baustein im BeeGFS-Dateisystem.

Diese Implementierungsanleitungen zeigen, wie Sie ein Filesystem implementieren, das aus einem Grundbaustein besteht, einschließlich Management-, Metadaten- und Storage-Services, einem zweiten Baustein mit Metadaten und Storage-Services und einem dritten Baustein nur für Storage.

Diese Schritte sollen den gesamten Bereich typischer Konfigurationsprofile anzeigen, mit denen Sie NetApp BeeGFS-Bausteine konfigurieren können, um die Anforderungen des gesamten BeeGFS-Dateisystems zu erfüllen.



Passen Sie in diesen und nachfolgenden Abschnitten nach Bedarf an, um den Bestand zu erstellen, der das BeeGFS-Dateisystem darstellt, das Sie bereitstellen möchten. Verwenden Sie insbesondere Ansible-Hostnamen, die jeden Block- oder Datei-Node darstellen, und das gewünschte IP-Adressschema für das Storage-Netzwerk, um sicherzustellen, dass es auf die Anzahl der BeeGFS-Datei-Nodes und -Clients skaliert werden kann.

## Schritt: Die Ansible-Bestandsdatei erstellen

### Schritte

1. Erstellen Sie eine neue `inventory.yml` Datei, und fügen Sie dann die folgenden Parameter ein, ersetzen Sie die Hosts unter `eseries_storage_systems` Nach Bedarf zur Darstellung der Block-Nodes in Ihrer Implementierung. Die Namen sollten dem Namen entsprechen, für den sie verwendet werden `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 den nachfolgenden Abschnitten werden unter weitere Ansible-Gruppen erstellt `ha_cluster` Die die BeeGFS-Dienste darstellen, die im Cluster ausgeführt werden sollen.

## Schritt: Inventar für einen Management-, Metadaten- und Storage-Baustein konfigurieren

Der erste Baustein im Cluster- oder Basis-Baustein muss den BeeGFS-Managementservice sowie Metadaten- und Storage-Services umfassen:

### Schritte

1. In `inventory.yml`, Befüllen Sie die folgenden Parameter unter `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. Erstellen Sie die Datei `group_vars/mgmt.yml` Und geben Sie Folgendes an:

```

# 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. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `meta_01` Bis `meta_08` Verwenden Sie die folgende Vorlage und füllen Sie dann die Platzhalterwerte für jeden Service aus, indem Sie auf die folgende Tabelle verweisen:

```
# 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>
```



Die Volume-Größe wird als Prozentsatz des gesamten Storage-Pools angegeben (auch als Volume-Gruppe bezeichnet). NetApp empfiehlt, freie Kapazitäten in jedem Pool zu belassen, um Platz für die SSD-Überprovisionierung zu haben (weitere Informationen finden Sie unter "[Einführung in das NetApp EF600 Array](#)"). Der Storage-Pool, beegfs\_m1\_m2\_m5\_m6, weist auch 1% der Kapazität des Pools für den Management-Service. Somit für Metadaten-Volumes im Storage-Pool beegfs\_m1\_m2\_m5\_m6, Wenn 1,92-TB- oder 3,84-TB-Laufwerke verwendet werden, setzen Sie diesen Wert auf 21.25; Für 7,5-TB-Laufwerke setzen Sie diesen Wert auf 22.25; Und für 15,3-TB-Laufwerke ist dieser Wert auf festgelegt 23.75.

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
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



Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
meta_04.yml	8045	l4b: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	l4b:100.127.1 02.8/16 i3b:100.127.1 01.8/16	1	netapp_02	Beegfs_m3_ m4_m7_m8	B

4. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `stor_01` Bis `stor_08` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```



Informationen zur richtigen Größe finden Sie unter "[Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools](#)".

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_01.yml	8013	i1b:100.127.1 03.1/16 i2b:100.127.1 04.1/16	0	netapp_01	Beegfs_s1_s 2	A
stor_02.yml	8023	i2b:100.127.1 04.2/16 i1b:100.127.1 03.2/16	0	netapp_01	Beegfs_s1_s 2	B
stor_03.yml	8033	i3b:100.127.1 03.3/16 i4b:100.127.1 04.3/16	1	netapp_02	Beegfs_s3_s 4	A
stor_04.yml	8043	i4b:100.127.1 04.4/16 i3b:100.127.1 03.4/16	1	netapp_02	Beegfs_s3_s 4	B

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
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

### Schritt 3: Konfigurieren Sie den Bestand für einen Baustein Metadaten + Speicher

In diesen Schritten wird beschrieben, wie ein Ansible-Inventar für BeeGFS-Metadaten + Storage-Baustein eingerichtet wird.

#### Schritte

1. In `inventory.yml`, Befüllen Sie die folgenden Parameter unter der vorhandenen Konfiguration:

```

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. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `meta_09` Bis `meta_16` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```



Informationen zur richtigen Größe finden Sie unter "[Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools](#)".

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
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

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
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_m14	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_m14	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_m16	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_m16	B

3. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `stor_09` Bis `stor_16` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:

```
# 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>
```



Die richtige Größe finden Sie unter "[Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools](#)" ..

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_09.yml	8013	i1b:100.127.1 03.9/16 i2b:100.127.1 04.9/16	0	netapp_03	Beegfs_s9_s10	A
stor_10.yml	8023	i2b:100.127.1 04.10/16 i1b:100.127.1 03.10/16	0	netapp_03	Beegfs_s9_s10	B
stor_11.yml	8033	i3b:100.127.1 03.11/16 i4b:100.127.1 04.11/16	1	netapp_04	Beegfs_s11_s12	A
stor_12.yml	8043	i4b:100.127.1 04.12/16 i3b:100.127.1 03.12/16	1	netapp_04	Beegfs_s11_s12	B
stor_13.yml	8053	i1b:100.127.1 03.13/16 i2b:100.127.1 04.13/16	0	netapp_03	Beegfs_s13_s14	A
stor_14.yml	8063	i2b:100.127.1 04.14/16 i1b:100.127.1 03.14/16	0	netapp_03	Beegfs_s13_s14	B
stor_15.yml	8073	i3b:100.127.1 03.15/16 i4b:100.127.1 04.15/16	1	netapp_04	Beegfs_s15_s16	A
stor_16.yml	8083	i4b:100.127.1 04.16/16 i3b:100.127.1 03.16/16	1	netapp_04	Beegfs_s15_s16	B

## Schritt 4: Konfigurieren Sie den Bestand für einen nur-Storage-Baustein

In diesen Schritten wird beschrieben, wie Sie einen Ansible-Bestand für einen einzigen BeeGFS-Baustein einrichten. Der Hauptunterschied zwischen der Konfiguration für Metadaten + Storage und einem rein Storage-basierten Baustein besteht darin, dass alle Metadaten-Ressourcengruppen und Änderungen nicht mehr berücksichtigt werden `criteria_drive_count` Von 10 bis 12 für jeden Speicherpool.

### Schritte

1. In `inventory.yml`, Befüllen Sie die folgenden Parameter unter der vorhandenen Konfiguration:

```
# 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. Unter `group_vars/`, Dateien für Ressourcengruppen erstellen `stor_17` Bis `stor_24` Füllen Sie anschließend die Platzhalterwerte für jeden Service aus, indem Sie auf das Beispiel verweisen:



```
# 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>
```



Die richtige Größe finden Sie unter "[Empfohlene Prozentsätze für die Überprovisionierung von Storage-Pools](#)".

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_17.yml	8013	i1b:100.127.1 03.17/16 i2b:100.127.1 04.17/16	0	netapp_05	Beegfs_s17_ s18	A
stor_18.yml	8023	i2b:100.127.1 04.18/16 i1b:100.127.1 03.18/16	0	netapp_05	Beegfs_s17_ s18	B
stor_19.yml	8033	i3b:100.127.1 03.19/16 i4b:100.127.1 04.19/16	1	netapp_06	Beegfs_s19_ s20	A
stor_20.yml	8043	i4b:100.127.1 04.20/16 i3b:100.127.1 03.20/16	1	netapp_06	Beegfs_s19_ s20	B

Dateiname	Port	Fließende IPs	NUMA-Zone	Block-Node	Storage-Pool	Controller, der die LUN besitzt
stor_21.yml	8053	i1b:100.127.1 03.21/16 i2b:100.127.1 04.21/16	0	netapp_05	Beegfs_s21_s22	A
stor_22.yml	8063	i2b:100.127.1 04.22/16 i1b:100.127.1 03.22/16	0	netapp_05	Beegfs_s21_s22	B
stor_23.yml	8073	i3b:100.127.1 03.23/16 i4b:100.127.1 04.23/16	1	netapp_06	Beegfs_s23_s24	A
stor_24.yml	8083	i4b:100.127.1 04.24/16 i3b:100.127.1 03.24/16	1	netapp_06	Beegfs_s23_s24	B

## BeeGFS bereitstellen

Zur Implementierung und zum Management der Konfiguration werden ein oder mehrere Playbooks ausgeführt, die die Aufgaben enthalten, die Ansible ausführen muss, und das gesamte System in den gewünschten Zustand bringen.

Zwar können alle Aufgaben in einem einzigen Playbook enthalten sein, doch bei komplexen Systemen ist dies schnell und schwerfällig. Mit Ansible können Sie Rollen erstellen und verteilen, um wiederverwendbare Playbooks und verwandte Inhalte (z. B. Standardvariablen, Aufgaben und Handler) zu verpacken. Weitere Informationen finden Sie in der Ansible-Dokumentation für ["Rollen"](#).

Rollen werden häufig im Rahmen einer Ansible Sammlung mit zugehörigen Rollen und Modulen verteilt. Daher importieren diese Playbooks in erster Linie mehrere Rollen, die in den verschiedenen NetApp E-Series Ansible Sammlungen verteilt sind.



Derzeit sind mindestens zwei Bausteine (vier Datei-Nodes) für die Bereitstellung von BeeGFS erforderlich, es sei denn, ein separates Quorum-Gerät ist als Tiebreaker konfiguriert, um Probleme beim Einrichten von Quorum mit einem Cluster mit zwei Nodes zu minimieren.

### Schritte

1. Erstellen Sie eine neue `playbook.yml` Datei und schließen Sie Folgendes ein:

```
# 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.
    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

```



In diesem Playbook wird ein paar ausgeführt `pre_tasks` Überprüfen Sie, ob Python 3 auf den Datei-Nodes installiert ist, und überprüfen Sie, ob die angegebenen Ansible-Tags unterstützt werden.

2. Verwenden Sie die `ansible-playbook` Befehl mit den Inventar- und Playbook-Dateien, wenn Sie bereit sind BeeGFS zu implementieren.

Die Bereitstellung wird komplett ausgeführt `pre_tasks`, Und dann zur Bestätigung des Benutzers aufgefordert, bevor mit der tatsächlichen BeeGFS-Bereitstellung.

Führen Sie den folgenden Befehl aus, indem Sie die Anzahl der Gabeln nach Bedarf anpassen (siehe Hinweis unten):

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



Insbesondere bei größeren Bereitstellungen `forks` wird empfohlen, die Standardanzahl von Gabeln (5) mit dem Parameter zu überschreiben, um die Anzahl der Hosts zu erhöhen, die Ansible parallel konfiguriert. (Weitere Informationen finden Sie unter "[Kontrolle der Playbook-Ausführung](#)".) Die Einstellung für den maximalen Wert hängt von der Verarbeitungsleistung ab, die auf dem Ansible-Steuerungsknoten verfügbar ist. Das oben genannte Beispiel von 20 wurde auf einem virtuellen Ansible-Steuerungsknoten mit 4 CPUs (Intel® Xeon® Gold 6146 CPU @ 3,20 GHz) ausgeführt.

Je nach Größe der Implementierung und Netzwerk-Performance zwischen dem Ansible Control Node und BeeGFS File- und Block-Nodes kann die Implementierungszeit variieren.

# Konfigurieren Sie BeeGFS-Clients

Sie müssen den BeeGFS-Client auf allen Hosts installieren und konfigurieren, die Zugriff auf das BeeGFS-Dateisystem benötigen, z. B. Compute- oder GPU-Nodes. Für diese Aufgabe können Sie Ansible und die BeeGFS-Sammlung verwenden.

## Schritte

1. Richten Sie bei Bedarf über den Ansible-Steuerungsknoten passwortlose SSH für jeden Host ein, den Sie als BeeGFS-Clients konfigurieren möchten:

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

2. Unter `host_vars/` Erstellen Sie für jeden BeeGFS-Client eine Datei mit dem Namen `<HOSTNAME>.yaml` Füllen Sie den Platzhaltertext mit den korrekten Informationen für Ihre Umgebung aus:

```
# 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>
```



Bei der Bereitstellung mit zwei Subnetzadressierungsschemata müssen auf jedem Client zwei InfiniBand-Schnittstellen konfiguriert werden, eine in jedem der beiden Storage-IPoIB-Subnetze. Wenn Sie die Beispiel-Subnetze und empfohlenen Bereiche für jeden hier aufgeführten BeeGFS-Dienst verwenden, sollten Clients eine Schnittstelle im Bereich von 100.127.1.0 bis 100.127.99.255 und die andere in bis 100.128.1.0 bis 100.128.99.255 konfigurieren.

3. Erstellen Sie eine neue Datei `client_inventory.yaml`, Und dann füllen Sie die folgenden Parameter an der Spitze:

```
# 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.
```



Speichern Sie Passwörter nicht im Klartext. Verwenden Sie stattdessen den Ansible Vault (siehe Ansible-Dokumentation für "[Verschlüsseln von Inhalten mit Ansible Vault](#)") Oder verwenden Sie die `--ask-become-pass` Option beim Ausführen des Playbooks.

4. Im `client_inventory.yml` Datei, Listen Sie alle Hosts auf, die als BeeGFS-Clients unter dem konfiguriert werden sollen `beegfs_clients` Gruppe, und geben Sie dann alle zusätzlichen Konfigurationen an, die zum Erstellen des BeeGFS-Client-Kernelmoduls erforderlich sind.

```

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.

```



Wenn Sie die NVIDIA OFED-Treiber verwenden, stellen Sie sicher, dass `beegfs_client_ofed_include_path` sie auf den richtigen „Header include path“ für Ihre Linux-Installation verweist. Weitere Informationen finden Sie in der BeeGFS-Dokumentation für ["RDMA-Unterstützung"](#).

5. Im `client_inventory.yml` Datei, Listen Sie die BeeGFS-Dateisysteme auf, die am unteren Rand eines zuvor definierten gemountet werden sollen `vars`.

```

    beegfs_client_mounts:
      - sysMgmtHost: 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

```



Der `beegfs_client_config` Stellt die Einstellungen dar, die getestet wurden. Lesen Sie die Dokumentation im `netapp_eseries.beegfs` Kollektion's `beegfs_client` Rolle für einen umfassenden Überblick über alle Optionen. Dies enthält Details zum Mounten mehrerer BeeGFS-Dateisysteme oder zum mehrere Male das gleiche BeeGFS-Dateisystem.

6. Erstellen Sie eine neue `client_playbook.yml` Datei, und füllen Sie dann die folgenden Parameter aus:



```
# 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
```



Beim Importieren des weglassen `netapp_eseries.host` Sammlung und `ipoib` Rolle, wenn Sie bereits die erforderlichen IB/RDMA-Treiber und IP-Adressen auf den entsprechenden IPoIB-Schnittstellen installiert haben.

7. Führen Sie den folgenden Befehl aus, um den Client zu installieren und zu erstellen und BeeGFS zu mounten:

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

8. Bevor Sie das BeeGFS-Dateisystem in Produktion setzen, empfehlen wir **\* dringend\***, sich bei allen Clients anzumelden und zu starten `beegfs-fsck --checkfs` Um sicherzustellen, dass alle Knoten erreichbar sind und keine Probleme gemeldet werden.

## Copyright-Informationen

Copyright © 2026 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtsinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGliche EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFTE SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGEND EINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

ERLÄUTERUNG ZU „RESTRICTED RIGHTS“: Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

## Markeninformationen

NETAPP, das NETAPP Logo und die unter <http://www.netapp.com/TM> aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.