SUSE Linux Enterprise Server

ONTAP SAN Host Utilities

NetApp
January 30, 2026

目次

SUSE Linux Enterprise Server

SUSE Linux Enterprise ServerのONTAPサポートと機能について学ぶ

NVMe over Fabrics (NVMe-oF) を使用したホスト構成でサポートされる機能は、ONTAPおよび SUSE Linux Enterprise Server のバージョンによって異なります。

特徴	**SUSE Linux Enterprise Server** ホストバージョン	**ONTAP**のバージョン
NVMe/TCPはASA r2システムのすべての最適化されたパスを報告します	16	9.16.1以降
NVMe/TCPではトランスポート層セキュリティ（TLS）1.3暗号化がサポートされています	15 SP6以降	9.16.1以降
RHELホストとONTAPコントローラ間のNVMe/TCP経由の安全なインバンド認証がサポートされています。	15 SP4以降	9.12.1以降
永続的検出コントローラ（PDC）は、固有の検出NQNを使用してサポートされます。	15 SP4以降	9.11.1以降
NVMe/TCPはネイティブを使用して名前空間を提供します `nvme-cli`パッケージ	15 SP4以降	9.10.1以降
NVMe と SCSI トラフィックは、NVMe-oF 名前空間の場合は NVMe マルチパス、SCSI LUN の場合は dm-multipath を使用して、同じホスト上でサポートされます。	15 SP1以降	9.4以降

ONTAP は、システム セットアップで実行されているONTAP のバージョンに関係なく、次の SAN ホスト機能をサポートします。

特徴	**SUSE Linux Enterprise Server** ホストバージョン
SANブートはNVMe/FCプロトコルを使用して有効化されます	15 SP7以降
ネイティブNVMeマルチパスはデフォルトで有効になっています	15 SP1以降
その `nvme-cli`パッケージには自動接続スクリプトが含まれており、サードパーティのスクリプトは不要になります。	15 SP1以降

サポートされている構成の詳細については、"Interoperability Matrix Tool" 。

次の手順

SUSE Linux Enterprise Server のバージョンが ... の場合	... について学びましょう。
16	"SUSE Linux Enterprise Server 16 用の NVMe の構成"
15 SPxシリーズ	"SUSE Linux Enterprise Server 15 SPx 用の NVMe の構成"

関連情報

- "ASA r2システムについて学ぶ"
- "NVMeプロトコルの管理について学ぶ"

ONTAPストレージでNVMe-oF用にSUSE Linux Enterprise Server 16を構成する

SUSE Linux Enterprise Server 16ホストは、非対称名前空間アクセス（ANA）を備えたNVMe over Fibre Channel（NVMe/FC）およびNVMe over TCP（NVMe/TCP）プロトコルをサポートしています。ANAは、iSCSIおよびFCP環境における非対称論理ユニットアクセス（ALUA）と同等のマルチパス機能を提供します。

SUSE Linux Enterprise Server 16 用の NVMe over Fabrics (NVMe-oF) ホストを構成する方法について説明します。サポートと機能の詳細については、"ONTAPのサポートと機能"を参照してください。

SUSE Linux Enterprise Server 16 の NVMe-oF には、次の既知の制限があります：

- その `nvme disconnect-all`このコマンドはルートファイルシステムとデータファイルシステムの両方を切断し、システムが不安定になる可能性があります。 NVMe-TCP または NVMe-FC 名前空間を介して SAN から起動するシステムではこれを発行しないでください。
- NetApp sanlun ホスト ユーティリティのサポートは NVMe-oF では利用できません。代わりに、ネイティブに含まれるNetAppプラグインを利用できます。 `nvme-cli` すべての NVMe-oF トランスポート用のパッケージ。

手順1：必要に応じてSANブートを有効にします。

SAN ブートを使用するようにホストを構成すると、展開が簡素化され、スケーラビリティが向上します。使用"Interoperability Matrix Tool"Linux OS、ホスト バス アダプタ (HBA)、HBA ファームウェア、HBA ブート BIOS、およびONTAPバージョンが SAN ブートをサポートしていることを確認します。

手順

1. "NVMe名前空間を作成し、ホストにマッピングする" 。
2. SAN ブート名前空間がマップされているポートに対して、サーバー BIOS で SAN ブートを有効にします。

   HBA BIOS を有効にする方法については、ベンダー固有のマニュアルを参照してください。

3. ホストを再起動し、OS が起動して実行されていることを確認します。

ステップ2: SUSE Linux Enterprise ServerとNVMeソフトウェアをインストールし、構成を確認する

NVMe-oF 用にホストを構成するには、ホストおよび NVMe ソフトウェア パッケージをインストールし、マルチパスを有効にして、ホストの NQN 構成を確認する必要があります。

手順

1. サーバーにSUSE Linux Enterprise Server 16をインストールします。インストールが完了したら、指定されたSUSE Linux Enterprise Server 16カーネルが実行されていることを確認します：

```
uname -r
```

SUSE Linux Enterprise Server カーネルバージョンの例：

```
6.12.0-160000.6-default
```

2. 「 nvme-cli 」パッケージをインストールします。

```
rpm -qa|grep nvme-cli
```

次の例は、 `nvme-cli`パッケージバージョン:

```
nvme-cli-2.11+29.g35e62868-160000.1.1.x86_64
```

3. をインストールします `libnvme` パッケージ：

```
rpm -qa|grep libnvme
```

次の例は、 `libnvme`パッケージバージョン:

```
libnvme1-1.11+17.g6d55624d-160000.1.1.x86_64
```

4. ホスト上で、hostnqn文字列を確認します。 `/etc/nvme/hostnqn` ：

```
cat /etc/nvme/hostnqn
```

次の例は、 `hostnqn`バージョン:

```
nqn.2014-08.org.nvmexpress:uuid:d3b581b4-c975-11e6-8425-0894ef31a074
```

5. ONTAPシステムで、 `hostnqn`文字列が一致する `hostnqn`ONTAPアレイ上の対応するサブシステムの文字列:

```
::> vserver nvme subsystem host show -vserver vs_coexistence_emulex
```

例を示します

```
Vserver Subsystem Priority  Host NQN
------- --------- --------
-------------------------------------------------
vs_coexistence_emulex
        nvme1
                  regular   nqn.2014-
08.org.nvmexpress:uuid:d3b581b4-c975-11e6-8425-0894ef31a074
        nvme10
                  regular   nqn.2014-
08.org.nvmexpress:uuid:d3b581b4-c975-11e6-8425-0894ef31a074
        nvme11
                  regular   nqn.2014-
08.org.nvmexpress:uuid:d3b581b4-c975-11e6-8425-0894ef31a074
        nvme12
                  regular   nqn.2014-
08.org.nvmexpress:uuid:d3b581b4-c975-11e6-8425-0894ef31a074
4 entries were displayed.
```

状況に応じて `hostnqn` 文字列が一致しない場合は、を使用してください `vserver modify` コマンドを使用してを更新します `hostnqn` 対応するONTAP アレイサブシステムで、に一致する文字列を指定します `hostnqn` から文字列 `/etc/nvme/hostnqn` ホスト。

ステップ3: NVMe/FCとNVMe/TCPを構成する

Broadcom/Emulex または Marvell/QLogic アダプタを使用して NVMe/FC を構成するか、手動の検出および接続操作を使用して NVMe/TCP を構成します。

NVMe/FC - ブロードコム/エミュレックス

Broadcom/Emulex FCアダプタ用にNVMe/FCを設定

手順

1. サポートされているアダプタモデルを使用していることを確認します。

   a. モデル名を表示します。

   ```
   cat /sys/class/scsi_host/host*/modelname
   ```

   次の出力が表示されます。

   ```
   SN37A92079
   SN37A92079
   ```

   b. モデルの説明を表示します。

   ```
   cat /sys/class/scsi_host/host*/modeldesc
   ```

   次の出力が表示されます。

   ```
   Emulex SN37A92079 32Gb 2-Port Fibre Channel Adapter
   Emulex SN37A92079 32Gb 2-Port Fibre Channel Adapter
   ```

2. 推奨されるBroadcomを使用していることを確認します `lpfc` ファームウェアおよび受信トレイドライバ：

   a. ファームウェアのバージョンを表示します。

   ```
   cat /sys/class/scsi_host/host*/fwrev
   ```

   次の例はファームウェアのバージョンを示しています。

   ```
   14.4.393.53, sli-4:6:d
   14.4.393.53, sli-4:6:d
   ```

   b. 受信トレイのドライバーのバージョンを表示します。

   ```
   cat /sys/module/lpfc/version
   ```

次の例は、ドライバーのバージョンを示しています。

```
0:14.4.0.11
```

サポートされているアダプタドライバおよびファームウェアバージョンの最新リストについては、を参照してください"Interoperability Matrix Tool"。

3. の想定される出力がに設定されている `3`ことを確認し `lpfc_enable_fc4_type`ます。

```
cat /sys/module/lpfc/parameters/lpfc_enable_fc4_type
```

4. イニシエータポートを表示できることを確認します。

```
cat /sys/class/fc_host/host*/port_name
```

次のような出力が表示されます：

```
0x100000109bdacc75
0x100000109bdacc76
```

5. イニシエータポートがオンラインであることを確認します。

```
cat /sys/class/fc_host/host*/port_state
```

次の出力が表示されます。

```
Online
Online
```

6. NVMe/FCイニシエータポートが有効になっており、ターゲットポートが認識されることを確認します。

```
cat /sys/class/scsi_host/host*/nvme_info
```

出力例を表示します。

```
NVME Initiator Enabled
XRI Dist lpfc0 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc0 WWPN x100000109bdacc75 WWNN x200000109bdacc75
DID x060100 ONLINE
NVME RPORT       WWPN x2001d039ea951c45 WWNN x2000d039ea951c45
DID x080801 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2003d039ea951c45 WWNN x2000d039ea951c45
DID x080d01 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2024d039eab31e9c WWNN x2023d039eab31e9c
DID x020a09 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2026d039eab31e9c WWNN x2023d039eab31e9c
DID x020a08 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2003d039ea5cfc90 WWNN x2002d039ea5cfc90
DID x061b01 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2012d039ea5cfc90 WWNN x2011d039ea5cfc90
DID x061b05 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2005d039ea5cfc90 WWNN x2002d039ea5cfc90
DID x061201 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2014d039ea5cfc90 WWNN x2011d039ea5cfc90
DID x061205 TARGET DISCSRVC ONLINE

NVME Statistics
LS: Xmt 0000017242 Cmpl 0000017242 Abort 00000000
LS XMIT: Err 00000000  CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 0000000000378362 Issue 00000000003783c7 OutIO
0000000000000065
        abort 00000409 noxri 00000000 nondlp 0000003a qdepth
00000000 wqerr 00000000 err 00000000
FCP CMPL: xb 00000409 Err 0000040a

NVME Initiator Enabled
XRI Dist lpfc1 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc1 WWPN x100000109bdacc76 WWNN x200000109bdacc76
DID x062800 ONLINE
NVME RPORT       WWPN x2002d039ea951c45 WWNN x2000d039ea951c45
DID x080701 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2004d039ea951c45 WWNN x2000d039ea951c45
DID x081501 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2025d039eab31e9c WWNN x2023d039eab31e9c
DID x020913 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2027d039eab31e9c WWNN x2023d039eab31e9c
DID x020912 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2006d039ea5cfc90 WWNN x2002d039ea5cfc90
DID x061401 TARGET DISCSRVC ONLINE
```

```
NVME RPORT       WWPN x2015d039ea5cfc90 WWNN x2011d039ea5cfc90
DID x061405 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2004d039ea5cfc90 WWNN x2002d039ea5cfc90
DID x061301 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2013d039ea5cfc90 WWNN x2011d039ea5cfc90
DID x061305 TARGET DISCSRVC ONLINE

NVME Statistics
LS: Xmt 0000017428 Cmpl 0000017428 Abort 00000000
LS XMIT: Err 00000000  CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 00000000003443be Issue 000000000034442a OutIO
000000000000006c
        abort 00000491 noxri 00000000 nondlp 00000086 qdepth
00000000 wqerr 00000000 err 00000000
FCP CMPL: xb 00000491 Err 00000494
```

NVMe/FC - マーベル**/QLogic**

Marvell/QLogicアダプタ用にNVMe/FCを設定します。

手順

1. サポートされているアダプタドライバとファームウェアのバージョンが実行されていることを確認します。

   ```
   cat /sys/class/fc_host/host*/symbolic_name
   ```

   次の例は、ドライバーとファームウェアのバージョンを示しています。

   ```
   QLE2772 FW:v9.15.06 DVR:v10.02.09.400-k-debug
   QLE2772 FW:v9.15.06 DVR:v10.02.09.400-k-debug
   ```

2. 確認します `ql2xnvmeenable` が設定されます。これにより、MarvellアダプタをNVMe/FCイニシエータとして機能させることができます。

   ```
   cat /sys/module/qla2xxx/parameters/ql2xnvmeenable
   ```

   想定される出力は1です。

NVMe/FC

NVMe/TCP プロトコルは自動接続操作をサポートしていません。代わりに、NVMe/TCPサブシステムと名前空間をNVMe/TCPコマンドで検出することができます。 `connect`または `connect-all`手動で操作します。

手順

1. イニシエータポートがサポートされているNVMe/TCP LIFの検出ログページのデータを取得できることを確認します。

```
nvme discover -t tcp -w <host-traddr> -a <traddr>
```

出力例を表示します。

```
nvme discover -t tcp -w 192.168.38.20 -a 192.168.38.10
Discovery Log Number of Records 8, Generation counter 42
=====Discovery Log Entry 0======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  4
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.211.71
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 1======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  3
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.111.71
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 2======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  2
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.211.70
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 3======
trtype:  tcp
```

```
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  1
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.111.70
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 4======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  4
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.211.71
eflags:  none
sectype: none
=====Discovery Log Entry 5======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  3
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.111.71
eflags:  none
sectype: none
=====Discovery Log Entry 6======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  2
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
```

```
le_tcp_sub
traddr:  192.168.211.70
eflags:  none
sectype: none
=====Discovery Log Entry 7======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  1
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.111.70
eflags:  none
sectype: none
localhost:~ #
```

2. NVMe/TCPイニシエータとターゲットLIFの他のすべての組み合わせで、検出ログページのデータを正常に取得できることを確認します。

```
nvme discover -t tcp -w <host-traddr> -a <traddr>
```

例を示します

```
nvme discover -t tcp -w 192.168.38.20 -a 192.168.38.10
nvme discover -t tcp -w 192.168.38.20 -a 192.168.38.11
nvme discover -t tcp -w 192.168.39.20 -a 192.168.39.10
nvme discover -t tcp -w 192.168.39.20 -a 192.168.39.11
```

3. を実行します `nvme connect-all` ノード全体でサポートされているすべてのNVMe/TCPイニシエータ/ターゲットLIFを対象としたコマンド：

```
nvme connect-all -t tcp -w <host-traddr> -a <traddr>
```

例を示します

```
nvme    connect-all -t  tcp -w  192.168.38.20   -a
192.168.38.10
nvme    connect-all -t  tcp -w  192.168.38.20   -a
192.168.38.11
nvme    connect-all -t  tcp -w  192.168.39.20   -a
192.168.39.10
nvme    connect-all -t  tcp -w  192.168.39.20   -a
192.168.39.11
```

NVMe/TCPの設定 `ctrl_loss_tmo timeout`自動的に「オフ」に設定されます。結果として：

- 再試行回数に制限はありません（無期限再試行）。
- 特定の設定を手動で行う必要はありません `ctrl_loss_tmo timeout`使用時の持続時間 `nvme connect`または `nvme connect-all`コマンド（オプション -l ）。
- NVMe/TCP コントローラーは、パス障害が発生した場合でもタイムアウトが発生せず、無期限に接続されたままになります。

ステップ**4:** オプションとして、**udev**ルールの**iopolicy**を変更します。

SUSE Linux Enterprise Server 16以降、NVMe-oFのデフォルトのiopolicyは `queue-depth`に設定されています。iopolicyを `round-robin`に変更したい場合は、udevルールファイルを以下のように変更してください：

手順

1. ルート権限でテキスト エディターで udev ルール ファイルを開きます。

   ```
   /usr/lib/udev/rules.d/71-nvmf-netapp.rules
   ```

   次の出力が表示されます。

   ```
   vi /usr/lib/udev/rules.d/71-nvmf-netapp.rules
   ```

2. 次の例のルールに示すように、 NetApp ONTAPコントローラの iopolicy を設定する行を見つけます。

   ```
   ACTION=="add", SUBSYSTEM=="nvme-subsystem", ATTR{subsystype}=="nvm",
   ATTR{model}=="NetApp ONTAP Controller", ATTR{iopolicy}="queue-depth"
   ```

3. 規則を修正して `queue-depth`が `round-robin`になるようにします：

```
ACTION=="add", SUBSYSTEM=="nvme-subsystem", ATTR{subsystype}=="nvm",
ATTR{model}=="NetApp ONTAP Controller", ATTR{iopolicy}="round-robin"
```

4. udev ルールを再読み込みし、変更を適用します。

```
udevadm control --reload
udevadm trigger --subsystem-match=nvme-subsystem
```

5. サブシステムの現在の iopolicy を確認します。 <subsystem>を置き換えます。例: `nvme-subsys0` 。

```
cat /sys/class/nvme-subsystem/<subsystem>/iopolicy
```

次の出力が表示されます。

```
round-robin
```

新しい iopolicy は、一致するNetApp ONTAPコントローラ デバイスに自動的に適用されます。再起動する必要はありません。

ステップ5: オプションでNVMe/FCの1MB I/Oを有効にする

ONTAP は、識別コントローラ データで最大データ転送サイズ (MDTS) が 8 であると報告します。つまり、最大 I/O 要求サイズは 1 MB までになります。 Broadcom NVMe/FCホストに1MBのI/Oリクエストを発行するには、 `lpfc`の価値 `lpfc_sg_seg_cnt`パラメータをデフォルト値の 64 から 256 に変更します。

この手順は、Qlogic NVMe/FCホストには適用されません。

手順

1. `lpfc_sg_seg_cnt`パラメータを256に設定します。

```
cat /etc/modprobe.d/lpfc.conf
```

次の例のような出力が表示されます。

```
options lpfc lpfc_sg_seg_cnt=256
```

2. コマンドを実行し `dracut -f`、ホストをリブートします。
3. の値が256であることを確認し `lpfc_sg_seg_cnt`ます。

```
cat /sys/module/lpfc/parameters/lpfc_sg_seg_cnt
```

ステップ6: NVMeブートサービスを確認する

その `nvmefc-boot-connections.service`そして `nvmf-autoconnect.service`NVMe/FCに含まれるブートサービス `nvme-cli`パッケージはシステムの起動時に自動的に有効になります。

起動が完了したら、 `nvmefc-boot-connections.service`そして `nvmf-autoconnect.service`ブート サービスが有効になっています。

手順

1. が有効であることを確認し `nvmf-autoconnect.service`ます。

```
systemctl status nvmf-autoconnect.service
```

出力例を表示します。

```
nvmf-autoconnect.service - Connect NVMe-oF subsystems automatically
during boot
  Loaded: loaded (/usr/lib/systemd/system/nvmf-autoconnect.service;
enabled; vendor preset: disabled)
  Active: inactive (dead) since Thu 2024-05-25 14:55:00 IST; 11min
ago
Process: 2108 ExecStartPre=/sbin/modprobe nvme-fabrics (code=exited,
status=0/SUCCESS)
Process: 2114 ExecStart=/usr/sbin/nvme connect-all (code=exited,
status=0/SUCCESS)
Main PID: 2114 (code=exited, status=0/SUCCESS)

systemd[1]: Starting Connect NVMe-oF subsystems automatically during
boot...
nvme[2114]: traddr=nn-0x201700a098fd4ca6:pn-0x201800a098fd4ca6 is
already connected
systemd[1]: nvmf-autoconnect.service: Deactivated successfully.
systemd[1]: Finished Connect NVMe-oF subsystems automatically during
boot.
```

2. が有効であることを確認し `nvmefc-boot-connections.service`ます。

```
systemctl status nvmefc-boot-connections.service
```

出力例を表示します。

```
nvmefc-boot-connections.service - Auto-connect to subsystems on FC-
NVME devices found during boot
   Loaded: loaded (/usr/lib/systemd/system/nvmefc-boot-
connections.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Thu 2024-05-25 14:55:00 IST; 11min
ago
 Main PID: 1647 (code=exited, status=0/SUCCESS)

systemd[1]: Starting Auto-connect to subsystems on FC-NVME devices
found during boot...
systemd[1]: nvmefc-boot-connections.service: Succeeded.
systemd[1]: Finished Auto-connect to subsystems on FC-NVME devices
found during boot.
```

ステップ**7:** マルチパス構成を確認する

カーネル内のNVMeマルチパスステータス、ANAステータス、およびONTAPネームスペースがNVMe-oF構成に対して正しいことを確認します。

手順

1. カーネル内NVMeマルチパスが有効になっていることを確認します。

   ```
   cat /sys/module/nvme_core/parameters/multipath
   ```

   次の出力が表示されます。

   ```
   Y
   ```

2. それぞれのONTAP名前空間の適切な NVMe-oF 設定 (モデルをNetApp ONTAPコントローラに設定し、ロード バランシングiopolicy を queue-depth に設定するなど) がホストに正しく反映されていることを確認します。

   a. サブシステムを表示します。

      ```
      cat /sys/class/nvme-subsystem/nvme-subsys*/model
      ```

      次の出力が表示されます。

```
NetApp ONTAP Controller
NetApp ONTAP Controller
```

b. ポリシーを表示します。

```
cat /sys/class/nvme-subsystem/nvme-subsys*/iopolicy
```

次の出力が表示されます。

```
queue-depth
queue-depth
```

3. ネームスペースが作成され、ホストで正しく検出されたことを確認します。

```
nvme list
```

例を示します

```
Node         SN                   Model
------------------------------------------------------------
/dev/nvme7n1 81Ix2BVuekWcAAAAAAAB   NetApp ONTAP Controller

Namespace Usage    Format              FW              Rev
-----             21.47 GB / 21.47 GB    4 KiB + 0 B   FFFFFFFF
```

4. 各パスのコントローラの状態がliveであり、正しいANAステータスが設定されていることを確認します。

```
nvme list-subsys /dev/<controller_ID>
```

ONTAP 9.16.1 以降、NVMe/FC および NVMe/TCP は ASA r2 システム上のすべての最適化されたパスを報告します。

NVMe/FC

次の出力例は、NVMe/FC を使用する AFF、FAS、ASA システムおよび ASA r2 システム用の 2 ノード ONTAP コントローラでホストされているネームスペースを示しています。

AFF、FAS、ASA出力例を表示します。

```
 nvme-subsys114 - NQN=nqn.1992-
08.com.netapp:sn.9e30b9760a4911f08c87d039eab67a95:subsystem.sles
_161_27
                 hostnqn=nqn.2014-
08.org.nvmexpress:uuid:f6517cae-3133-11e8-bbff-7ed30aef123f
iopolicy=round-robin\
+- nvme114 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2360d039ea359e4a,host_traddr=nn-0x20000090fae0ec88:pn-
0x10000090fae0ec88 live optimized
+- nvme115 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2362d039ea359e4a,host_traddr=nn-0x20000090fae0ec88:pn-
0x10000090fae0ec88 live non-optimized
+- nvme116 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2361d039ea359e4a,host_traddr=nn-0x20000090fae0ec89:pn-
0x10000090fae0ec89 live optimized
+- nvme117 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2363d039ea359e4a,host_traddr=nn-0x20000090fae0ec89:pn-
0x10000090fae0ec89 live non-optimized
```

ASA r2 の出力例を表示

```
nvme-subsys96 - NQN=nqn.1992-
08.om.netapp:sn.b351b2b6777b11f0b3c2d039ea5cfc91:subsystem.nvme2
4
                hostnqn=nqn.2014-
08.org.nvmexpress:uuid:d3b581b4-c975-11e6-8425-0894ef31a074
\
 +- nvme203 fc traddr=nn-0x2011d039ea5cfc90:pn-
0x2015d039ea5cfc90,host_traddr=nn-0x200000109bdacc76:pn-
0x100000109bdacc76 live optimized
 +- nvme25 fc traddr=nn-0x2011d039ea5cfc90:pn-
0x2014d039ea5cfc90,host_traddr=nn-0x200000109bdacc75:pn-
0x100000109bdacc75 live optimized
 +- nvme30 fc traddr=nn-0x2011d039ea5cfc90:pn-
0x2012d039ea5cfc90,host_traddr=nn-0x200000109bdacc75:pn-
0x100000109bdacc75 live optimized
 +- nvme32 fc traddr=nn-0x2011d039ea5cfc90:pn-
0x2013d039ea5cfc90,host_traddr=nn-0x200000109bdacc76:pn-
0x100000109bdacc76 live optimized
```

NVMe/FC

次の出力例は、NVMe/TCP を使用する AFF、FAS、ASA システムおよび ASA r2 システム用の 2 ノード ONTAP コントローラでホストされているネームスペースを示しています。

AFF、FAS、ASA出力例を表示します。

```
nvme-subsys9 - NQN=nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme
10
               hostnqn=nqn.2014-08.org.nvmexpress:uuid:4c4c4544-
0035-5910-804b-b7c04f444d33
\
 +- nvme105 tcp
traddr=192.168.39.10,trsvcid=4420,host_traddr=192.168.39.20,src_
addr=192.168.39.20 live optimized
 +- nvme153 tcp
traddr=192.168.39.11,trsvcid=4420,host_traddr=192.168.39.20,src_
addr=192.168.39.20 live non-optimized
 +- nvme57 tcp
traddr=192.168.38.11,trsvcid=4420,host_traddr=192.168.38.20,src_
addr=192.168.38.20 live non-optimized
 +- nvme9 tcp
traddr=192.168.38.10,trsvcid=4420,host_traddr=192.168.38.20,src_
addr=192.168.38.20 live optimized
```

ASA r2 の出力例を表示

```
nvme-subsys4 - NQN=nqn.1992-
08.com.netapp:sn.17e32b6e8c7f11f09545d039eac03c33:subsystem.Bidi
rectional_DHCP_1_0
               hostnqn=nqn.2014-08.org.nvmexpress:uuid:4c4c4544-
0054-5110-8039-c3c04f523034
\
+- nvme4 tcp
traddr=192.168.20.28,trsvcid=4420,host_traddr=192.168.20.21,src_
addr=192.168.20.21 live optimized
+- nvme5 tcp
traddr=192.168.20.29,trsvcid=4420,host_traddr=192.168.20.21,src_
addr=192.168.20.21 live optimized
+- nvme6 tcp
traddr=192.168.21.28,trsvcid=4420,host_traddr=192.168.21.21,src_
addr=192.168.21.21 live optimized
+- nvme7 tcp
traddr=192.168.21.29,trsvcid=4420,host_traddr=192.168.21.21,src_
addr=192.168.21.21 live optimized
```

5. ネットアッププラグインで、ONTAP ネームスペースデバイスごとに正しい値が表示されていることを確認します。

列（**Column**）

```
nvme netapp ontapdevices -o column
```

例を示します

```
Device            Vserver                   Namespace Path
NSID UUID                                    Size
----------------- ------------------------- ------------------
---- --------------------------------------- ---------
/dev/nvme0n1      vs_coexistence_emulex     ns1                1
79510f05-7784-11f0-b3c2-d039ea5cfc91   21.47GB
```

JSON

```
nvme netapp ontapdevices -o json
```

例を示します

```
{
"ONTAPdevices":[{
      "Device":"/dev/nvme0n1",
      "Vserver":"vs_coexistence_emulex",
      "Namespace_Path":"ns1",
      "NSID":1,
      "UUID":"79510f05-7784-11f0-b3c2-d039ea5cfc91",
      "Size":"21.47GB",
      "LBA_Data_Size":4096,
      "Namespace_Size":5242880
    }  ]
}
```

ステップ**8:** 永続的な検出コントローラを作成する

SUSE Linux Enterprise Server 16ホスト用の永続検出コントローラ（PDC）を作成できます。PDCは、NVMeサブシステムの追加または削除操作と、検出ログページデータの変更を自動的に検出するために必要です。

手順

1. 検出ログページのデータが使用可能で、イニシエータポートとターゲットLIFの組み合わせから取得できることを確認します。

```
nvme discover -t <trtype> -w <host-traddr> -a <traddr>
```

出力例を表示します。

```
Discovery Log Number of Records 8, Generation counter 10
=====Discovery Log Entry 0======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  3
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:discovery
traddr:  192.168.39.10
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 1======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  1
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:discovery
traddr:  192.168.38.10
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 2======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  4
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:discovery
traddr:  192.168.39.11
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 3======
trtype:  tcp
adrfam:  ipv4
```

```
subtype: current discovery subsystem
treq:    not specified
portid:  2
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:discovery
traddr:  192.168.38.11
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 4======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  3
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
traddr:  192.168.39.10
eflags:  none
sectype: none
=====Discovery Log Entry 5======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  1
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
traddr:  192.168.38.10
eflags:  none
sectype: none
=====Discovery Log Entry 6======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  4
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
traddr:  192.168.39.11
eflags:  none
sectype: none
```

```
=====Discovery Log Entry 7======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  2
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
traddr:  192.168.38.11
eflags:  none
sectype: none
```

2. 検出サブシステムのPDCを作成します。

```
nvme discover -t <trtype> -w <host-traddr> -a <traddr> -p
```

次の出力が表示されます。

```
nvme discover -t tcp -w 192.168.39.20 -a 192.168.39.11 -p
```

3. ONTAPコントローラから、PDCが作成されたことを確認します。

```
vserver nvme show-discovery-controller -instance -vserver <vserver_name>
```

出力例を表示します。

```
vserver nvme show-discovery-controller -instance -vserver
vs_tcp_sles16
Vserver Name: vs_tcp_sles16
               Controller ID: 0180h
     Discovery Subsystem NQN: nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:discovery
           Logical Interface: lif3
                        Node: A400-12-171
                   Host NQN: nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b7c04f444d33
          Transport Protocol: nvme-tcp
 Initiator Transport Address: 192.168.39.20
Transport Service Identifier: 8009
             Host Identifier: 4c4c454400355910804bb7c04f444d33
            Admin Queue Depth: 32
       Header Digest Enabled: false
         Data Digest Enabled: false
   Keep-Alive Timeout (msec): 30000
```

ステップ**9:** 安全なインバンド認証を設定する

SUSE Linux Enterprise Server 16ホストとONTAPコントローラ間のNVMe/TCP経由の安全なインバンド認証がサポートされます。

各ホストまたはコントローラは、 `DH-HMAC-CHAP` 安全な認証を設定するためのキー。DH-HMAC-CHAP キーは、NVMe ホストまたはコントローラの NQN と管理者が設定した認証シークレットの組み合わせです。ピアを認証するには、NVMe ホストまたはコントローラはピアに関連付けられたキーを認識する必要があります。

手順

CLI または設定 JSON ファイルを使用して、安全なインバンド認証を設定します。サブシステムごとに異なるDHCHAPキーを指定する必要がある場合は、config JSONファイルを使用する必要があります。

CLI の使用

CLIを使用してセキュアなインバンド認証を設定します。

1. ホストNQNを取得します。

```
cat /etc/nvme/hostnqn
```

2. ホストの dhchap キーを生成します。

   コマンドパラメータの出力を次に示し `gen-dhchap-key`ます。

```
nvme gen-dhchap-key -s optional_secret -l key_length {32|48|64} -m
HMAC_function {0|1|2|3} -n host_nqn
•   -s secret key in hexadecimal characters to be used to initialize
the host key
•   -l length of the resulting key in bytes
•   -m HMAC function to use for key transformation
0 = none, 1- SHA-256, 2 = SHA-384, 3=SHA-512
•   -n host NQN to use for key transformation
```

   次の例では、HMACが3に設定されたランダムDHCHAPキー（SHA-512）が生成されます。

```
nvme gen-dhchap-key -m 3 -n nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b7c04f444d33
DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFGwmhgwd
JWmVoripbWbMJy5eMAbCahN4hhYU=:
```

3. ONTAPコントローラで、ホストを追加し、両方のDHCHAPキーを指定します。

```
vserver nvme subsystem host add -vserver <svm_name> -subsystem
<subsystem> -host-nqn <host_nqn> -dhchap-host-secret
<authentication_host_secret> -dhchap-controller-secret
<authentication_controller_secret> -dhchap-hash-function {sha-
256|sha-512} -dhchap-group {none|2048-bit|3072-bit|4096-bit|6144-
bit|8192-bit}
```

4. ホストは、単方向と双方向の2種類の認証方式をサポートします。ホストで、ONTAPコントローラに接続し、選択した認証方式に基づいてDHCHAPキーを指定します。

```
nvme connect -t tcp -w <host-traddr> -a <tr-addr> -n <host_nqn> -S
<authentication_host_secret> -C <authentication_controller_secret>
```

5. 検証する `nvme connect authentication` ホストとコントローラのDHCHAPキーを確認してコマンドを実行します。
   a. ホストDHCHAPキーを確認します。

   ```
   cat /sys/class/nvme-subsystem/<nvme-subsysX>/nvme*/dhchap_secret
   ```

   に、単方向設定の出力例を示します。

   ```
   # cat /sys/class/nvme-subsystem/nvme-
   subsys1/nvme*/dhchap_secret
   DHHC-1:01:wkwAKk8r9Ip7qECKt7V5aIo/7Y1CH7DWkUfLfMxmseg39DFb:
   DHHC-1:01:wkwAKk8r9Ip7qECKt7V5aIo/7Y1CH7DWkUfLfMxmseg39DFb:
   DHHC-1:01:wkwAKk8r9Ip7qECKt7V5aIo/7Y1CH7DWkUfLfMxmseg39DFb:
   DHHC-1:01:wkwAKk8r9Ip7qECKt7V5aIo/7Y1CH7DWkUfLfMxmseg39DFb:
   ```

   b. コントローラのDHCHAPキーを確認します。

   ```
   cat /sys/class/nvme-subsystem/<nvme-
   subsysX>/nvme*/dhchap_ctrl_secret
   ```

に、双方向設定の出力例を示します。

```
# cat /sys/class/nvme-subsystem/nvme-
subsys6/nvme*/dhchap_ctrl_secret
DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFG
wmhgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:
DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFG
wmhgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:
DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFG
wmhgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:
DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFG
wmhgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:
```

JSON

ONTAPコントローラ構成で複数のNVMeサブシステムを使用できる場合は、コマンドでファイルを `nvme connect-all`使用できます `/etc/nvme/config.json`。

使用 `-o`JSON ファイルを生成するオプション。詳細な構文オプションについては、NVMe connect-all のマニュアル ページを参照してください。

1. JSON ファイルを設定します。

出力例を表示します。

```
# cat /etc/nvme/config.json
[
  {
    "hostnqn":"nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-
5910-804b-b7c04f444d33",
    "hostid":"4c4c4544-0035-5910-804b-b7c04f444d33",
    "dhchap_key":"DHHC-
1:01:wkwAKk8r9Ip7qECKt7V5aIo/7Y1CH7DWkUfLfMxmseg39DFb:",
    "subsystems":[
      {
        "nqn":"nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.inba
nd_bidirectional",
        "ports":[
          {
            "transport":"tcp",
            "traddr":"192.168.38.10",
            "host_traddr":"192.168.38.20",
            "trsvcid":"4420",
            "dhchap_ctrl_key":"DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFGwm
hgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:"
          },
          {
            "transport":"tcp",
            "traddr":"192.168.38.11",
            "host_traddr":"192.168.38.20",
            "trsvcid":"4420",
            "dhchap_ctrl_key":"DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFGwm
hgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:"
          },
          {
            "transport":"tcp",
            "traddr":"192.168.39.11",
            "host_traddr":"192.168.39.20",
            "trsvcid":"4420",
            "dhchap_ctrl_key":"DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFGwm
hgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:"
          },
          {
            "transport":"tcp",
```

```
            "traddr":"192.168.39.10",
            "host_traddr":"192.168.39.20",
            "trsvcid":"4420",
            "dhchap_ctrl_key":"DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFGwm
hgwdJWmVoripbWbMJy5eMAbCahN4hhYU=:"
          }
        ]
      }
    ]
  }
]
```

次の例では、 `dhchap_key`対応する `dhchap_secret`そして `dhchap_ctrl_key`対応する `dhchap_ctrl_secret`。

2. config jsonファイルを使用してONTAPコントローラに接続します。

```
nvme connect-all -J /etc/nvme/config.json
```

出力例を表示します。

```
traddr=192.168.38.10is already connected
traddr=192.168.39.10 is already connected
traddr=192.168.38.11 is already connected
traddr=192.168.39.11 is already connected
traddr=192.168.38.10is already connected
traddr=192.168.39.10 is already connected
traddr=192.168.38.11 is already connected
traddr=192.168.39.11 is already connected
traddr=192.168.38.10is already connected
traddr=192.168.39.10 is already connected
traddr=192.168.38.11 is already connected
traddr=192.168.39.11 is already connected
```

3. 各サブシステムの各コントローラでDHCHAPシークレットが有効になっていることを確認します。

   a. ホストDHCHAPキーを確認します。

```
cat /sys/class/nvme-subsystem/nvme-subsys0/nvme0/dhchap_secret
```

次の例は、dhchap キーを示しています。

```
DHHC-1:01:wkwAKk8r9Ip7qECKt7V5aIo/7Y1CH7DWkUfLfMxmseg39DFb:
```

b. コントローラのDHCHAPキーを確認します。

```
cat /sys/class/nvme-subsystem/nvme-
subsys0/nvme0/dhchap_ctrl_secret
```

次の例のような出力が表示されます。

```
DHHC-
1:03:ohdxI1yIS8gBLwIOubcwl57rXcozYuRgBsoWaBvxEvpDlQHn/7dQ4JjFGwmhgwd
JWmVoripbWbMJy5eMAbCahN4hhYU=:
```

ステップ10: トランスポート層セキュリティを構成する

トランスポート層セキュリティ (TLS) は、NVMe-oF ホストとONTAPアレイ間の NVMe 接続に安全なエンドツーエンドの暗号化を提供します。CLI と設定された事前共有キー (PSK) を使用して TLS 1.3 を設定できます。

ONTAPコントローラで手順を実行するように指定されている場合を除き、SUSE Linux Enterprise Server ホストで次の手順を実行します。

手順

1. 以下のものをお持ちかご確認ください `ktls-utils` 、 `openssl` 、 そして `libopenssl`ホストにインストールされているパッケージ:

a. 確認する `ktls-utils` ：

```
rpm -qa | grep ktls
```

次の出力が表示されます。

```
ktls-utils-0.10+33.g311d943-160000.2.2.x86_64
```

a. SSL パッケージを確認します。

```
rpm -qa | grep ssl
```

出力例を表示します。

```
libopenssl3-3.5.0-160000.3.2.x86_64
openssl-3.5.0-160000.2.2.noarch
openssl-3-3.5.0-160000.3.2.x86_64
libopenssl3-x86-64-v3-3.5.0-160000.3.2.x86_64
```

2. 次の設定が正しいことを確認し `/etc/tlshd.conf` ます。

```
cat /etc/tlshd.conf
```

出力例を表示します。

```
[debug]
loglevel=0
tls=0
nl=0

[authenticate]
#keyrings= <keyring>;<keyring>;<keyring>

[authenticate.client]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>

[authenticate.server]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>
```

3. システム起動時に起動するように有効にし `tlshd` ます。

```
systemctl enable tlshd
```

4. デーモンが実行されていることを確認し `tlshd` ます。

```
systemctl status tlshd
```

出力例を表示します。

```
tlshd.service - Handshake service for kernel TLS consumers
   Loaded: loaded (/usr/lib/systemd/system/tlshd.service; enabled;
preset: disabled)
   Active: active (running) since Wed 2024-08-21 15:46:53 IST; 4h
57min ago
     Docs: man:tlshd(8)
Main PID: 961 (tlshd)
   Tasks: 1
     CPU: 46ms
   CGroup: /system.slice/tlshd.service
       └─961 /usr/sbin/tlshd
Aug 21 15:46:54 RX2530-M4-17-153 tlshd[961]: Built from ktls-utils
0.11-dev on Mar 21 2024 12:00:00
```

5. を使用してTLS PSKを生成し `nvme gen-tls-key`ます。

   a. ホストを確認します:

   ```
   cat /etc/nvme/hostnqn
   ```

   次の出力が表示されます。

   ```
   nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b7c04f444d33
   ```

   b. キーを検証します:

   ```
   nvme gen-tls-key --hmac=1 --identity=1 --subsysnqn= nqn.1992-
   08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
   ```

   次の出力が表示されます。

```
NVMeTLSkey-1:01:C50EsaGtuOp8n5fGE9EuWjbBCtshmfoHx4XTqTJUmydf0gIj:
```

6. ONTAPコントローラで、ONTAPサブシステムにTLS PSKを追加します。

出力例を表示します。

```
nvme subsystem host add -vserver vs_iscsi_tcp -subsystem nvme1 -host
-nqn nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-
b2c04f444d33 -tls-configured-psk NVMeTLSkey-
1:01:C50EsaGtuOp8n5fGE9EuWjbBCtshmfoHx4XTqTJUmydf0gIj:
```

7. TLS PSKをホストカーネルキーリングに挿入します。

```
nvme check-tls-key --identity=1 --subsysnqn=nqn.1992
-08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
--keydata=NVMeTLSkey
-1:01:C50EsaGtuOp8n5fGE9EuWjbBCtshmfoHx4XTqTJUmydf0gIj: --insert
```

次の TLS キーが表示されます。

```
Inserted TLS key 069f56bb
```

PSKは次のように表示されます `NVMe1R01`使用するため `identity v1` TLS ハンドシェイク アルゴリズムから。Identity v1は、ONTAPがサポートする唯一のバージョンです。

8. TLS PSKが正しく挿入されていることを確認します。

```
cat /proc/keys | grep NVMe
```

出力例を表示します。

```
069f56bb I-Q-- 5 perm 3b010000 0 0 psk NVMe1R01 nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
oYVLelmiOwnvDjXKBmrnIgGVpFIBDJtc4hmQXE/36Sw=: 32
```

9. 挿入したTLS PSKを使用してONTAPサブシステムに接続します。

   a. TLS PSK を検証します。

```
nvme connect -t tcp -w 192.168.38.20 -a 192.168.38.10  -n nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
--tls_key=0x069f56bb –tls
```

次の出力が表示されます。

```
connecting to device: nvme0
```

a. list-subsys を検証します。

```
nvme list-subsys
```

出力例を表示します。

```
nvme-subsys0 - NQN=nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
               hostnqn=nqn.2014-08.org.nvmexpress:uuid:4c4c4544-
0035-5910-804b-b2c04f444d33
**
 +- nvme0 tcp
traddr=192.168.38.10,trsvcid=4420,host_traddr=192.168.38.20,src_a
ddr=192.168.38.20 live
```

10. ターゲットを追加し、指定したONTAPサブシステムへのTLS接続を確認します。

```
nvme subsystem controller show -vserver vs_tcp_sles16 -subsystem nvme1
-instance
```

出力例を表示します。

```
(vserver nvme subsystem controller show)
                        Vserver Name: vs_tcp_sles16
                           Subsystem: nvme1
                       Controller ID: 0040h
                   Logical Interface: lif1
                                Node: A400-12-171
                            Host NQN: nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
                  Transport Protocol: nvme-tcp
         Initiator Transport Address: 192.168.38.20
                     Host Identifier:
4c4c454400355910804bb2c04f444d33
                Number of I/O Queues: 2
                    I/O Queue Depths: 128, 128
                   Admin Queue Depth: 32
               Max I/O Size in Bytes: 1048576
           Keep-Alive Timeout (msec): 5000
                       Subsystem UUID: 62203cfd-826a-11f0-966e-
d039eab31e9d
               Header Digest Enabled: false
                 Data Digest Enabled: false
       Authentication Hash Function: sha-256
Authentication Diffie-Hellman Group: 3072-bit
                 Authentication Mode: unidirectional
       Transport Service Identifier: 4420
                        TLS Key Type: configured
                    TLS PSK Identity: NVMe1R01 nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
nqn.1992-
08.com.netapp:sn.9927e165694211f0b4f4d039eab31e9d:subsystem.nvme1
oYVLelmiOwnvDjXKBmrnIgGVpFIBDJtc4hmQXE/36Sw=
                          TLS Cipher: TLS-AES-128-GCM-SHA256
```

手順11：既知の問題を確認する

既知の問題はありません。

ONTAPストレージでNVMe-oF用にSUSE Linux Enterprise Server 15 SPxを構成する

SUSE Linux Enterprise Server 15 SPx ホストは、非対称名前空間アクセス (ANA) を備え

た NVMe over Fibre Channel (NVMe/FC) および NVMe over TCP (NVMe/TCP) プロトコルをサポートします。ANA は、iSCSI および FCP 環境における非対称論理ユニットアクセス (ALUA) と同等のマルチパス機能を提供します。

SUSE Linux Enterprise Server 15 SPx 用の NVMe over Fabrics (NVMe-oF) ホストを構成する方法を学習します。詳細なサポートと機能情報については、 "ONTAPのサポートと機能"。

SUSE Linux Enterprise Server 15 SPx の NVMe-oF には、次の既知の制限があります。

- その `nvme disconnect-all`このコマンドはルートファイルシステムとデータファイルシステムの両方を切断し、システムが不安定になる可能性があります。 NVMe-TCP または NVMe-FC 名前空間を介して SAN から起動するシステムではこれを発行しないでください。
- NetApp sanlun ホスト ユーティリティのサポートは NVMe-oF では利用できません。代わりに、ネイティブに含まれるNetAppプラグインを利用できます。 `nvme-cli` すべての NVMe-oF トランスポート用のパッケージ。
- SUSE Linux Enterprise Server 15 SP6 以前では、NVMe-oF プロトコルを使用した SAN ブートはサポートされていません。

手順1：必要に応じてSANブートを有効にします。

SAN ブートを使用するようにホストを構成すると、展開が簡素化され、スケーラビリティが向上します。使用"Interoperability Matrix Tool"Linux OS、ホスト バス アダプタ (HBA)、HBA ファームウェア、HBA ブート BIOS、およびONTAPバージョンが SAN ブートをサポートしていることを確認します。

手順

1. "NVMe名前空間を作成し、ホストにマッピングする" 。
2. SAN ブート名前空間がマップされているポートに対して、サーバー BIOS で SAN ブートを有効にします。

   HBA BIOS を有効にする方法については、ベンダー固有のマニュアルを参照してください。

3. ホストを再起動し、OS が起動して実行されていることを確認します。

ステップ2: SUSE Linux Enterprise ServerとNVMeソフトウェアをインストールし、構成を確認する

NVMe-oF 用にホストを構成するには、ホストおよび NVMe ソフトウェア パッケージをインストールし、マルチパスを有効にして、ホストの NQN 構成を確認する必要があります。

手順

1. サーバーに SUSE Linux Enterprise Server 15 SPx をインストールします。インストールが完了したら、指定された SUSE Linux Enterprise Server 15 SPx カーネルが実行されていることを確認します。

   ```
   uname -r
   ```

   Rocky Linux カーネルバージョンの例:

```
6.4.0-150700.53.3-default
```

2. 「 nvme-cli 」パッケージをインストールします。

```
rpm -qa|grep nvme-cli
```

次の例は、 `nvme-cli`パッケージバージョン:

```
nvme-cli-2.11+22.gd31b1a01-150700.3.3.2.x86_64
```

3. をインストールします `libnvme` パッケージ：

```
rpm -qa|grep libnvme
```

次の例は、 `libnvme`パッケージバージョン:

```
libnvme1-1.11+4.ge68a91ae-150700.4.3.2.x86_64
```

4. ホスト上で、hostnqn文字列を確認します。 `/etc/nvme/hostnqn` ：

```
cat /etc/nvme/hostnqn
```

次の例は、 `hostnqn`バージョン:

```
nqn.2014-08.org.nvmexpress:uuid:f6517cae-3133-11e8-bbff-7ed30aef123f
```

5. ONTAPシステムで、 `hostnqn`文字列が一致する `hostnqn`ONTAPアレイ上の対応するサブシステムの文字列:

```
::> vserver nvme subsystem host show -vserver vs_coexistence_LPE36002
```

例を示します

```
Vserver Subsystem Priority  Host NQN
------- --------- --------
-------------------------------------------------
vs_coexistence_LPE36002
        nvme
                  regular   nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0056-5410-8048-b9c04f425633
        nvme_1
                  regular   nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0056-5410-8048-b9c04f425633
        nvme_2
                  regular   nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0056-5410-8048-b9c04f425633
        nvme_3
                  regular   nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0056-5410-8048-b9c04f425633
4 entries were displayed.
```

状況に応じて `hostnqn` 文字列が一致しない場合は、を使用してください `vserver modify` コマンドを使用してを更新します `hostnqn` 対応するONTAP アレイサブシステムで、に一致する文字列を指定します `hostnqn` から文字列 `/etc/nvme/hostnqn` ホスト。

ステップ**3: NVMe/FC**と**NVMe/TCP**を構成する

Broadcom/Emulex または Marvell/QLogic アダプタを使用して NVMe/FC を構成するか、手動の検出および接続操作を使用して NVMe/TCP を構成します。

NVMe/FC - ブロードコム/エミュレックス

Broadcom/Emulex FCアダプタ用にNVMe/FCを設定

手順

1. サポートされているアダプタモデルを使用していることを確認します。

   a. モデル名を表示します。

   ```
   cat /sys/class/scsi_host/host*/modelname
   ```

   次の出力が表示されます。

   ```
   LPe36002-M64
   LPe36002-M64
   ```

   b. モデルの説明を表示します。

   ```
   cat /sys/class/scsi_host/host*/modeldesc
   ```

   次の出力が表示されます。

   ```
   Emulex LightPulse LPe36002-M64 2-Port 64Gb Fibre Channel Adapter
   Emulex LightPulse LPe36002-M64 2-Port 64Gb Fibre Channel Adapter
   ```

2. 推奨されるBroadcomを使用していることを確認します `lpfc` ファームウェアおよび受信トレイドライバ：

   a. ファームウェアのバージョンを表示します。

   ```
   cat /sys/class/scsi_host/host*/fwrev
   ```

   次の例はファームウェアのバージョンを示しています。

   ```
   14.4.393.25, sli-4:2:c
   14.4.393.25, sli-4:2:c
   ```

   b. 受信トレイのドライバーのバージョンを表示します。

   ```
   cat /sys/module/lpfc/version
   ```

次の例は、ドライバーのバージョンを示しています。

```
0:14.4.0.8
```

サポートされているアダプタドライバおよびファームウェアバージョンの最新リストについては、を参照してください"Interoperability Matrix Tool"。

3. の想定される出力がに設定されている `3`ことを確認し `lpfc_enable_fc4_type`ます。

```
cat /sys/module/lpfc/parameters/lpfc_enable_fc4_type
```

4. イニシエータポートを表示できることを確認します。

```
cat /sys/class/fc_host/host*/port_name
```

次のような出力が表示されます：

```
0x10000090fae0ec88
0x10000090fae0ec89
```

5. イニシエータポートがオンラインであることを確認します。

```
cat /sys/class/fc_host/host*/port_state
```

次の出力が表示されます。

```
Online
Online
```

6. NVMe/FCイニシエータポートが有効になっており、ターゲットポートが認識されることを確認します。

```
cat /sys/class/scsi_host/host*/nvme_info
```

出力例を表示します。

```
NVME Initiator Enabled
XRI Dist lpfc0 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc0 WWPN x10000090fae0ec88 WWNN x20000090fae0ec88
DID x0a1300 ONLINE
NVME RPORT       WWPN x23b1d039ea359e4a WWNN x23aed039ea359e4a
DID x0a1c01 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x22bbd039ea359e4a WWNN x22b8d039ea359e4a
DID x0a1c0b TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2362d039ea359e4a WWNN x234ed039ea359e4a
DID x0a1c10 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x23afd039ea359e4a WWNN x23aed039ea359e4a
DID x0a1a02 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x22b9d039ea359e4a WWNN x22b8d039ea359e4a
DID x0a1a0b TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2360d039ea359e4a WWNN x234ed039ea359e4a
DID x0a1a11 TARGET DISCSRVC ONLINE

NVME Statistics
LS: Xmt 0000004ea0 Cmpl 0000004ea0 Abort 00000000
LS XMIT: Err 00000000  CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 0000000000102c35 Issue 0000000000102c2d OutIO
fffffffffffffff8
        abort 00000175 noxri 00000000 nondlp 0000021d qdepth
00000000 wqerr 00000007 err 00000000
FCP CMPL: xb 00000175 Err 0000058b

NVME Initiator Enabled
XRI Dist lpfc1 Total 6144 IO 5894 ELS 250
NVME LPORT lpfc1 WWPN x10000090fae0ec89 WWNN x20000090fae0ec89
DID x0a1200 ONLINE
NVME RPORT       WWPN x23b2d039ea359e4a WWNN x23aed039ea359e4a
DID x0a1d01 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x22bcd039ea359e4a WWNN x22b8d039ea359e4a
DID x0a1d0b TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2363d039ea359e4a WWNN x234ed039ea359e4a
DID x0a1d10 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x23b0d039ea359e4a WWNN x23aed039ea359e4a
DID x0a1b02 TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x22bad039ea359e4a WWNN x22b8d039ea359e4a
DID x0a1b0b TARGET DISCSRVC ONLINE
NVME RPORT       WWPN x2361d039ea359e4a WWNN x234ed039ea359e4a
DID x0a1b11 TARGET DISCSRVC ONLINE

NVME Statistics
```

```
LS: Xmt 0000004e31 Cmpl 0000004e31 Abort 00000000
LS XMIT: Err 00000000  CMPL: xb 00000000 Err 00000000
Total FCP Cmpl 00000000001017f2 Issue 00000000001017ef OutIO
fffffffffffffffd
        abort 0000018a noxri 00000000 nondlp 0000012e qdepth
00000000 wqerr 00000004 err 00000000
FCP CMPL: xb 0000018a Err 000005ca
```

NVMe/FC - マーベル**/QLogic**

Marvell/QLogicアダプタ用にNVMe/FCを設定します。

手順

1. サポートされているアダプタドライバとファームウェアのバージョンが実行されていることを確認します。

```
cat /sys/class/fc_host/host*/symbolic_name
```

次の例は、ドライバーとファームウェアのバージョンを示しています。

```
QLE2742 FW:v9.14.00 DVR:v10.02.09.400-k-debug
QLE2742 FW:v9.14.00 DVR:v10.02.09.400-k-debug
```

2. 確認します `ql2xnvmeenable` が設定されます。これにより、MarvellアダプタをNVMe/FCイニシエータとして機能させることができます。

```
cat /sys/module/qla2xxx/parameters/ql2xnvmeenable
```

想定される出力は1です。

NVMe/FC

NVMe/TCP プロトコルは自動接続操作をサポートしていません。代わりに、NVMe/TCPサブシステムと名前空間をNVMe/TCPコマンドで検出することができます。 `connect`または `connect-all`手動で操作します。

手順

1. イニシエータポートがサポートされているNVMe/TCP LIFの検出ログページのデータを取得できることを確認します。

```
nvme discover -t tcp -w <host-traddr> -a <traddr>
```

出力例を表示します。

```
nvme discover -t tcp -w 192.168.111.80 -a 192.168.111.70

Discovery Log Number of Records 8, Generation counter 42
=====Discovery Log Entry 0======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  4
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.211.71
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 1======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  3
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.111.71
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 2======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  2
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.211.70
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 3======
```

```
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  1
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:discovery
traddr:  192.168.111.70
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 4======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  4
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.211.71
eflags:  none
sectype: none
=====Discovery Log Entry 5======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  3
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.111.71
eflags:  none
sectype: none
=====Discovery Log Entry 6======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  2
trsvcid: 4420
subnqn:  nqn.1992-
```

```
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.211.70
eflags:  none
sectype: none
=====Discovery Log Entry 7======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  1
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub
traddr:  192.168.111.70
eflags:  none
sectype: none
localhost:~ #
```

2. NVMe/TCPイニシエータとターゲットLIFの他のすべての組み合わせで、検出ログページのデータを正常に取得できることを確認します。

```
nvme discover -t tcp -w <host-traddr> -a <traddr>
```

例を示します

```
nvme discover -t tcp -w 192.168.111.80 -a 192.168.111.66
nvme discover -t tcp -w 192.168.111.80 -a 192.168.111.67
nvme discover -t tcp -w 192.168.211.80 -a 192.168.211.66
nvme discover -t tcp -w 192.168.211.80 -a 192.168.211.67
```

3. を実行します `nvme connect-all` ノード全体でサポートされているすべてのNVMe/TCPイニシエータ/ターゲットLIFを対象としたコマンド：

```
nvme connect-all -t tcp -w <host-traddr> -a <traddr>
```

例を示します

```
nvme    connect-all -t  tcp -w  192.168.111.80  -a
192.168.111.66
nvme    connect-all -t  tcp -w  192.168.111.80  -a
192.168.111.67
nvme    connect-all -t  tcp -w  192.168.211.80  -a
192.168.211.66
nvme    connect-all -t  tcp -w  192.168.211.80  -a
192.168.211.67
```

SUSE Linux Enterprise Server 15 SP6以降、NVMe/TCPの設定は `ctrl_loss_tmo timeout` 自動的に「オフ」に設定されます。結果として：

- 再試行回数に制限はありません（無期限再試行）。
- 特定の設定を手動で行う必要はありません `ctrl_loss_tmo timeout`使用時の持続時間 `nvme connect`または `nvme connect-all`コマンド（オプション -l ）。
- NVMe/TCP コントローラーは、パス障害が発生した場合でもタイムアウトが発生せず、無期限に接続されたままになります。

ステップ**4:** オプションとして、**udev**ルールの**iopolicy**を変更します。

SUSE Linux Enterprise Server 15 SP6以降、NVMe-oFのデフォルトのiopolicyは次のように設定されています。 `round-robin`。iopolicyを次のように変更したい場合 `queue-depth`、udev ルール ファイルを次のように変更します。

手順

1. ルート権限でテキスト エディターで udev ルール ファイルを開きます。

   ```
   /usr/lib/udev/rules.d/71-nvmf-netapp.rules
   ```

   次の出力が表示されます。

   ```
   vi /usr/lib/udev/rules.d/71-nvmf-netapp.rules
   ```

2. 次の例のルールに示すように、 NetApp ONTAPコントローラの iopolicy を設定する行を見つけます。

   ```
   ACTION=="add", SUBSYSTEM=="nvme-subsystem", ATTR{subsystype}=="nvm",
   ATTR{model}=="NetApp ONTAP Controller", ATTR{iopolicy}="round-robin"
   ```

3. ルールを修正して `round-robin`なる `queue-depth`:

```
ACTION=="add", SUBSYSTEM=="nvme-subsystem", ATTR{subsystype}=="nvm",
ATTR{model}=="NetApp ONTAP Controller", ATTR{iopolicy}="queue-depth"
```

4. udev ルールを再読み込みし、変更を適用します。

```
udevadm control --reload
udevadm trigger --subsystem-match=nvme-subsystem
```

5. サブシステムの現在の iopolicy を確認します。 <subsystem>を置き換えます。例: `nvme-subsys0` 。

```
cat /sys/class/nvme-subsystem/<subsystem>/iopolicy
```

次の出力が表示されます。

```
queue-depth.
```

新しい iopolicy は、一致するNetApp ONTAPコントローラ デバイスに自動的に適用されます。再起動する必要はありません。

ステップ5: オプションでNVMe/FCの1MB I/Oを有効にする

ONTAP は、識別コントローラ データで最大データ転送サイズ (MDTS) が 8 であると報告します。つまり、最大 I/O 要求サイズは 1 MB までになります。 Broadcom NVMe/FCホストに1MBのI/Oリクエストを発行するには、 `lpfc`の価値 `lpfc_sg_seg_cnt`パラメータをデフォルト値の 64 から 256 に変更します。

この手順は、Qlogic NVMe/FCホストには適用されません。

手順

1. `lpfc_sg_seg_cnt`パラメータを256に設定します。

```
cat /etc/modprobe.d/lpfc.conf
```

次の例のような出力が表示されます。

```
options lpfc lpfc_sg_seg_cnt=256
```

2. コマンドを実行し `dracut -f`、ホストをリブートします。

3. の値が256であることを確認し `lpfc_sg_seg_cnt`ます。

```
cat /sys/module/lpfc/parameters/lpfc_sg_seg_cnt
```

ステップ6: NVMeブートサービスを確認する

その `nvmefc-boot-connections.service`そして `nvmf-autoconnect.service`NVMe/FCに含まれるブートサービス `nvme-cli`パッケージはシステムの起動時に自動的に有効になります。

起動が完了したら、 `nvmefc-boot-connections.service`そして `nvmf-autoconnect.service`ブート サービスが有効になっています。

手順

1. が有効であることを確認し `nvmf-autoconnect.service`ます。

```
systemctl status nvmf-autoconnect.service
```

出力例を表示します。

```
nvmf-autoconnect.service - Connect NVMe-oF subsystems automatically
during boot
  Loaded: loaded (/usr/lib/systemd/system/nvmf-autoconnect.service;
enabled; preset: enabled)
  Active: inactive (dead) since Fri 2025-07-04 23:56:38 IST; 4 days
ago
  Main PID: 12208 (code=exited, status=0/SUCCESS)
    CPU: 62ms

Jul 04 23:56:26 localhost systemd[1]: Starting Connect NVMe-oF
subsystems automatically during boot...
Jul 04 23:56:38 localhost systemd[1]: nvmf-autoconnect.service:
Deactivated successfully.
Jul 04 23:56:38 localhost systemd[1]: Finished Connect NVMe-oF
subsystems automatically during boot.
```

2. が有効であることを確認し `nvmefc-boot-connections.service`ます。

```
systemctl status nvmefc-boot-connections.service
```

出力例を表示します。

```
nvmefc-boot-connections.service - Auto-connect to subsystems on FC-
NVME devices found during boot
    Loaded: loaded (/usr/lib/systemd/system/nvmefc-boot-
connections.service; enabled; preset: enabled)
    Active: inactive (dead) since Mon 2025-07-07 19:52:30 IST; 1 day
4h ago
  Main PID: 2945 (code=exited, status=0/SUCCESS)
       CPU: 14ms

Jul 07 19:52:30 HP-DL360-14-168 systemd[1]: Starting Auto-connect to
subsystems on FC-NVME devices found during boot...
Jul 07 19:52:30 HP-DL360-14-168 systemd[1]: nvmefc-boot-
connections.service: Deactivated successfully.
Jul 07 19:52:30 HP-DL360-14-168 systemd[1]: Finished Auto-connect to
subsystems on FC-NVME devices found during boot.
```

ステップ**7:** マルチパス構成を確認する

カーネル内のNVMeマルチパスステータス、ANAステータス、およびONTAPネームスペースがNVMe-oF構成に対して正しいことを確認します。

手順

1. カーネル内NVMeマルチパスが有効になっていることを確認します。

```
cat /sys/module/nvme_core/parameters/multipath
```

次の出力が表示されます。

```
Y
```

2. それぞれのONTAP名前空間の適切な NVMe-oF 設定 (モデルをNetApp ONTAPコントローラに設定し、ロード バランシングiopolicy を queue-depth に設定するなど) がホストに正しく反映されていることを確認します。

   a. サブシステムを表示します。

```
cat /sys/class/nvme-subsystem/nvme-subsys*/model
```

次の出力が表示されます。

```
NetApp ONTAP Controller
NetApp ONTAP Controller
```

b. ポリシーを表示します。

```
cat /sys/class/nvme-subsystem/nvme-subsys*/iopolicy
```

次の出力が表示されます。

```
queue-depth
queue-depth
```

3. ネームスペースが作成され、ホストで正しく検出されたことを確認します。

```
nvme list
```

例を示します

```
Node         SN                   Model
------------------------------------------------------------
/dev/nvme4n1 81Ix2BVuekWcAAAAAAAB   NetApp ONTAP Controller


Namespace Usage    Format             FW             Rev
------------------------------------------------------------
1                 21.47 GB / 21.47 GB   4 KiB + 0 B   FFFFFFFF
```

4. 各パスのコントローラの状態がliveであり、正しいANAステータスが設定されていることを確認します。

NVMe/FC

```
nvme list-subsys /dev/nvme4n5
```

出力例を表示します。

```
nvme-subsys114 - NQN=nqn.1992-
08.com.netapp:sn.9e30b9760a4911f08c87d039eab67a95:subsystem.sles
_161_27
                 hostnqn=nqn.2014-
08.org.nvmexpress:uuid:f6517cae-3133-11e8-bbff-7ed30aef123f
iopolicy=round-robin\
+- nvme114 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2360d039ea359e4a,host_traddr=nn-0x20000090fae0ec88:pn-
0x10000090fae0ec88 live optimized
+- nvme115 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2362d039ea359e4a,host_traddr=nn-0x20000090fae0ec88:pn-
0x10000090fae0ec88 live non-optimized
+- nvme116 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2361d039ea359e4a,host_traddr=nn-0x20000090fae0ec89:pn-
0x10000090fae0ec89 live optimized
+- nvme117 fc traddr=nn-0x234ed039ea359e4a:pn-
0x2363d039ea359e4a,host_traddr=nn-0x20000090fae0ec89:pn-
0x10000090fae0ec89 live non-optimized
```

NVMe/FC

```
nvme list-subsys /dev/nvme9n1
```

出力例を表示します。

```
nvme-subsys9 - NQN=nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.with
_inband_with_json hostnqn=nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
iopolicy=round-robin
\
+- nvme10 tcp
traddr=192.168.111.71,trsvcid=4420,src_addr=192.168.111.80 live
non-optimized
 +- nvme11 tcp
traddr=192.168.211.70,trsvcid=4420,src_addr=192.168.211.80 live
optimized
 +- nvme12 tcp
traddr=192.168.111.70,trsvcid=4420,src_addr=192.168.111.80 live
optimized
 +- nvme9 tcp
traddr=192.168.211.71,trsvcid=4420,src_addr=192.168.211.80 live
non-optimized
```

5. ネットアッププラグインで、ONTAP ネームスペースデバイスごとに正しい値が表示されていることを確認します。

列（ **Column** ）

```
nvme netapp ontapdevices -o column
```

例を示します

```
Device           Vserver                          Namespace Path
NSID UUID                                          Size
---------------- -------------------------
--------------------------------------------------- ----
------------------------------------ ---------
/dev/nvme0n1     vs_161
/vol/fc_nvme_vol1/fc_nvme_ns1                          1
32fd92c7-0797-428e-a577-fdb3f14d0dc3   5.37GB
```

JSON

```
nvme netapp ontapdevices -o json
```

例を示します

```
{
      "Device":"/dev/nvme98n2",
      "Vserver":"vs_161",
      "Namespace_Path":"/vol/fc_nvme_vol71/fc_nvme_ns71",
      "NSID":2,
      "UUID":"39d634c4-a75e-4fbd-ab00-3f9355a26e43",
      "LBA_Size":4096,
      "Namespace_Size":5368709120,
      "UsedBytes":430649344,
    }
  ]
}
```

ステップ8: 永続的な検出コントローラを作成する

SUSE Linux Enterprise Server 15 SPx ホスト用の永続検出コントローラ (PDC) を作成できます。NVMe サブシステムの追加または削除操作と検出ログ ページ データの変更を自動的に検出するには、PDC が必要です。

手順

1. 検出ログページのデータが使用可能で、イニシエータポートとターゲットLIFの組み合わせから取得できることを確認します。

```
nvme discover -t <trtype> -w <host-traddr> -a <traddr>
```

出力例を表示します。

```
Discovery Log Number of Records 8, Generation counter 18
=====Discovery Log Entry 0======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  4
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:discovery
traddr:  192.168.111.66
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 1======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  2
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:discovery
traddr:  192.168.211.66
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 2======
trtype:  tcp
adrfam:  ipv4
subtype: current discovery subsystem
treq:    not specified
portid:  3
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:discovery
traddr:  192.168.111.67
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 3======
trtype:  tcp
adrfam:  ipv4
```

```
subtype: current discovery subsystem
treq:    not specified
portid:  1
trsvcid: 8009
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:discovery
traddr:  192.168.211.67
eflags:  explicit discovery connections, duplicate discovery
information
sectype: none
=====Discovery Log Entry 4======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  4
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:subsystem.pdc
traddr:  192.168.111.66
eflags:  none
sectype: none
=====Discovery Log Entry 5======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  2
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:subsystem.pdc
traddr:  192.168.211.66
eflags:  none
sectype: none
=====Discovery Log Entry 6======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  3
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:subsystem.pdc
traddr:  192.168.111.67
eflags:  none
sectype: none
```

```
=====Discovery Log Entry 7======
trtype:  tcp
adrfam:  ipv4
subtype: nvme subsystem
treq:    not specified
portid:  1
trsvcid: 4420
subnqn:  nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:subsystem.pdc
traddr:  192.168.211.67
eflags:  none
sectype: none
```

2. 検出サブシステムのPDCを作成します。

```
nvme discover -t <trtype> -w <host-traddr> -a <traddr> -p
```

次の出力が表示されます。

```
nvme discover -t tcp -w 192.168.111.80 -a 192.168.111.66 -p
```

3. ONTAPコントローラから、PDCが作成されたことを確認します。

```
vserver nvme show-discovery-controller -instance -vserver <vserver_name>
```

出力例を表示します。

```
vserver nvme show-discovery-controller -instance -vserver vs_pdc

           Vserver Name: vs_pdc
               Controller ID: 0101h
     Discovery Subsystem NQN: nqn.1992-
08.com.netapp:sn.4f7af2bd221811f0afadd039eab0dadd:discovery
           Logical Interface: lif2
                       Node: A400-12-181
                  Host NQN: nqn.2014-
08.org.nvmexpress:uuid:9796c1ec-0d34-11eb-b6b2-3a68dd3bab57
          Transport Protocol: nvme-tcp
 Initiator Transport Address: 192.168.111.80
Transport Service Identifier: 8009
             Host Identifier: 9796c1ec0d3411ebb6b23a68dd3bab57
           Admin Queue Depth: 32
       Header Digest Enabled: false
         Data Digest Enabled: false
   Keep-Alive Timeout (msec): 30000
```

ステップ**9:** 安全なインバンド認証を設定する

SUSE Linux Enterprise Server 15 SPx ホストとONTAPコントローラ間の NVMe/TCP 経由の安全なインバンド認証がサポートされます。

各ホストまたはコントローラは、 `DH-HMAC-CHAP` 安全な認証を設定するためのキー。DH-HMAC-CHAP キーは、NVMe ホストまたはコントローラの NQN と管理者が設定した認証シークレットの組み合わせです。ピアを認証するには、NVMe ホストまたはコントローラはピアに関連付けられたキーを認識する必要があります。

手順

CLI または設定 JSON ファイルを使用して、安全なインバンド認証を設定します。サブシステムごとに異なるDHCHAPキーを指定する必要がある場合は、config JSONファイルを使用する必要があります。

CLI の使用

CLIを使用してセキュアなインバンド認証を設定します。

1. ホストNQNを取得します。

```
cat /etc/nvme/hostnqn
```

2. ホストの dhchap キーを生成します。

   コマンドパラメータの出力を次に示し `gen-dhchap-key`ます。

```
nvme gen-dhchap-key -s optional_secret -l key_length {32|48|64} -m
HMAC_function {0|1|2|3} -n host_nqn
•   -s secret key in hexadecimal characters to be used to initialize
the host key
•   -l length of the resulting key in bytes
•   -m HMAC function to use for key transformation
0 = none, 1- SHA-256, 2 = SHA-384, 3=SHA-512
•   -n host NQN to use for key transformation
```

   次の例では、HMACが3に設定されたランダムDHCHAPキー（SHA-512）が生成されます。

```
nvme gen-dhchap-key -m 3 -n nqn.2014-
08.org.nvmexpress:uuid:e6dade64-216d-11ec-b7bb-7ed30a5482c3
DHHC-
1:03:1CFivw9ccz58gAcOUJrM7Vs98hd2ZHSr+iw+Amg6xZPl5D2Yk+HDTZiUAg1iGgx
TYqnxukqvYedA55Bw3wtz6sJNpR4=:
```

3. ONTAPコントローラで、ホストを追加し、両方のDHCHAPキーを指定します。

```
vserver nvme subsystem host add -vserver <svm_name> -subsystem
<subsystem> -host-nqn <host_nqn> -dhchap-host-secret
<authentication_host_secret> -dhchap-controller-secret
<authentication_controller_secret> -dhchap-hash-function {sha-
256|sha-512} -dhchap-group {none|2048-bit|3072-bit|4096-bit|6144-
bit|8192-bit}
```

4. ホストは、単方向と双方向の2種類の認証方式をサポートします。ホストで、ONTAPコントローラに接続し、選択した認証方式に基づいてDHCHAPキーを指定します。

```
nvme connect -t tcp -w <host-traddr> -a <tr-addr> -n <host_nqn> -S
<authentication_host_secret> -C <authentication_controller_secret>
```

5. 検証する `nvme connect authentication` ホストとコントローラのDHCHAPキーを確認してコマンドを実行します。
   a. ホストDHCHAPキーを確認します。

   ```
   cat /sys/class/nvme-subsystem/<nvme-subsysX>/nvme*/dhchap_secret
   ```

   に、単方向設定の出力例を示します。

   ```
   cat /sys/class/nvme-subsystem/nvme-subsys1/nvme*/dhchap_secret
   DHHC-1:01:iM63E6cX7G5SOKKOju8gmzM53qywsy+C/YwtzxhIt9ZRz+ky:
   DHHC-1:01:iM63E6cX7G5SOKKOju8gmzM53qywsy+C/YwtzxhIt9ZRz+ky:
   DHHC-1:01:iM63E6cX7G5SOKKOju8gmzM53qywsy+C/YwtzxhIt9ZRz+ky:
   DHHC-1:01:iM63E6cX7G5SOKKOju8gmzM53qywsy+C/YwtzxhIt9ZRz+ky:
   ```

   b. コントローラのDHCHAPキーを確認します。

   ```
   cat /sys/class/nvme-subsystem/<nvme-
   subsysX>/nvme*/dhchap_ctrl_secret
   ```

   に、双方向設定の出力例を示します。

   ```
   cat /sys/class/nvme-subsystem/nvme-
   subsys6/nvme*/dhchap_ctrl_secret
   DHHC-
   1:03:1CFivw9ccz58gAcOUJrM7Vs98hd2ZHSr+iw+Amg6xZPl5D2Yk+HDTZiUA
   g1iGgxTYqnxukqvYedA55Bw3wtz6sJNpR4=:
   DHHC-
   1:03:1CFivw9ccz58gAcOUJrM7Vs98hd2ZHSr+iw+Amg6xZPl5D2Yk+HDTZiUA
   g1iGgxTYqnxukqvYedA55Bw3wtz6sJNpR4=:
   DHHC-
   1:03:1CFivw9ccz58gAcOUJrM7Vs98hd2ZHSr+iw+Amg6xZPl5D2Yk+HDTZiUA
   g1iGgxTYqnxukqvYedA55Bw3wtz6sJNpR4=:
   DHHC-
   1:03:1CFivw9ccz58gAcOUJrM7Vs98hd2ZHSr+iw+Amg6xZPl5D2Yk+HDTZiUA
   g1iGgxTYqnxukqvYedA55Bw3wtz6sJNpR4=:
   ```

JSON

ONTAPコントローラ構成で複数のNVMeサブシステムを使用できる場合は、コマンドでファイルを `nvme connect-all`使用できます `/etc/nvme/config.json`。

使用 `-o`JSON ファイルを生成するオプション。詳細な構文オプションについては、NVMe connect-all のマニュアル ページを参照してください。

1. JSON ファイルを設定します。

出力例を表示します。

```
cat /etc/nvme/config.json
[
 {
    "hostnqn":"nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-
5910-804b-b2c04f444d33",
    "hostid":"4c4c4544-0035-5910-804b-b2c04f444d33",
    "dhchap_key":"DHHC-
1:01:i4i789R11sMuHLCY27RVI8XloC\/GzjRwyhxip5hmIELsHrBq:",
    "subsystems":[
      {
        "nqn":"nqn.1992-
08.com.netapp:sn.f8e2af201b7211f0ac2bd039eab67a95:subsystem.samp
le_tcp_sub",
        "ports":[
          {
            "transport":"tcp",
            "traddr":"192.168.111.70",
            "host_traddr":"192.168.111.80",
            "trsvcid":"4420"
            "dhchap_ctrl_key":"DHHC-
1:03:jqgYcJSKp73+XqAf2X6twr9ngBpr2n0MGWbmZIZq4PieKZCoilKGef8lAvh
YS0PNK7T+04YD5CRPjh+m3qjJU++yR8s=:"
          },
                {
                     "transport":"tcp",
                     "traddr":"192.168.111.71",
                     "host_traddr":"192.168.111.80",
                     "trsvcid":"4420",
                     "dhchap_ctrl_key":"DHHC-
1:03:jqgYcJSKp73+XqAf2X6twr9ngBpr2n0MGWbmZIZq4PieKZCoilKGef8lAvh
YS0PNK7T+04YD5CRPjh+m3qjJU++yR8s=:"
                },
                {
                     "transport":"tcp",
                     "traddr":"192.168.211.70",
                     "host_traddr":"192.168.211.80",
                     "trsvcid":"4420",
                     "dhchap_ctrl_key":"DHHC-
1:03:jqgYcJSKp73+XqAf2X6twr9ngBpr2n0MGWbmZIZq4PieKZCoilKGef8lAvh
YS0PNK7T+04YD5CRPjh+m3qjJU++yR8s=:"
                },
                {
                     "transport":"tcp",
```

```
                    "traddr":"192.168.211.71",
                    "host_traddr":"192.168.211.80",
                    "trsvcid":"4420",
                    "dhchap_ctrl_key":"DHHC-
1:03:jqgYcJSKp73+XqAf2X6twr9ngBpr2n0MGWbmZIZq4PieKZCoilKGef8lAvh
YS0PNK7T+04YD5CRPjh+m3qjJU++yR8s=:"
                }
            ]
        }
    ]
  }
]
```

次の例では、 `dhchap_key`対応する `dhchap_secret`そして `dhchap_ctrl_key`対応する `dhchap_ctrl_secret`。

2. config jsonファイルを使用してONTAPコントローラに接続します。

```
nvme connect-all -J /etc/nvme/config.json
```

出力例を表示します。

```
traddr=192.168.211.70 is already connected
traddr=192.168.111.71 is already connected
traddr=192.168.211.71 is already connected
traddr=192.168.111.70 is already connected
traddr=192.168.211.70 is already connected
traddr=192.168.111.70 is already connected
traddr=192.168.211.71 is already connected
traddr=192.168.111.71 is already connected
traddr=192.168.211.70 is already connected
traddr=192.168.111.71 is already connected
traddr=192.168.211.71 is already connected
traddr=192.168.111.70 is already connected
```

3. 各サブシステムの各コントローラでDHCHAPシークレットが有効になっていることを確認します。

   a. ホストDHCHAPキーを確認します。

```
cat /sys/class/nvme-subsystem/nvme-subsys0/nvme0/dhchap_secret
```

次の例は、dhchap キーを示しています。

```
DHHC-1:01:i4i789R11sMuHLCY27RVI8XloC/GzjRwyhxip5hmIELsHrBq:
```

b. コントローラのDHCHAPキーを確認します。

```
cat /sys/class/nvme-subsystem/nvme-
subsys0/nvme0/dhchap_ctrl_secret
```

次の例のような出力が表示されます。

```
DHHC-
1:03:jqgYcJSKp73+XqAf2X6twr9ngBpr2n0MGWbmZIZq4PieKZCoilKGef8lAvhYS0P
NK7T+04YD5CRPjh+m3qjJU++yR8s=:
```

ステップ10: トランスポート層セキュリティを構成する

トランスポート層セキュリティ (TLS) は、NVMe-oF ホストとONTAPアレイ間の NVMe 接続に安全なエンドツーエンドの暗号化を提供します。CLI と設定された事前共有キー (PSK) を使用して TLS 1.3 を設定できます。

ONTAPコントローラで手順を実行するように指定されている場合を除き、SUSE Linux Enterprise Server ホストで次の手順を実行します。

手順

1. 以下のものをお持ちかご確認ください `ktls-utils` 、 `openssl` 、 そして `libopenssl`ホストにインストールされているパッケージ:

a. 確認する `ktls-utils` ：

```
rpm -qa | grep ktls
```

次の出力が表示されます。

```
ktls-utils-0.10+33.g311d943-150700.1.5.x86_64
```

a. SSL パッケージを確認します。

```
rpm -qa | grep ssl
```

出力例を表示します。

```
libopenssl3-3.2.3-150700.3.20.x86_64
openssl-3-3.2.3-150700.3.20.x86_64
libopenssl1_1-1.1.1w-150700.9.37.x86_64
```

2. 次の設定が正しいことを確認し `/etc/tlshd.conf`ます。

```
cat /etc/tlshd.conf
```

出力例を表示します。

```
[debug]
loglevel=0
tls=0
nl=0
[authenticate]
keyrings=.nvme
[authenticate.client]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>
[authenticate.server]
#x509.truststore= <pathname>
#x509.certificate= <pathname>
#x509.private_key= <pathname>
```

3. システム起動時に起動するように有効にし `tlshd`ます。

```
systemctl enable tlshd
```

4. デーモンが実行されていることを確認し `tlshd`ます。

```
systemctl status tlshd
```

出力例を表示します。

```
tlshd.service - Handshake service for kernel TLS consumers
   Loaded: loaded (/usr/lib/systemd/system/tlshd.service; enabled;
preset: disabled)
   Active: active (running) since Wed 2024-08-21 15:46:53 IST; 4h
57min ago
     Docs: man:tlshd(8)
Main PID: 961 (tlshd)
   Tasks: 1
     CPU: 46ms
   CGroup: /system.slice/tlshd.service
       └─961 /usr/sbin/tlshd
Aug 21 15:46:54 RX2530-M4-17-153 tlshd[961]: Built from ktls-utils
0.11-dev on Mar 21 2024 12:00:00
```

5. を使用してTLS PSKを生成し `nvme gen-tls-key`ます。

   a. ホストを確認します:

   ```
   cat /etc/nvme/hostnqn
   ```

   次の出力が表示されます。

   ```
   nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
   ```

   b. キーを検証します:

   ```
   nvme gen-tls-key --hmac=1 --identity=1 --subsysnqn= nqn.1992-
   08.com.netapp:sn.a2d41235b78211efb57dd039eab67a95:subsystem.nvme1
   ```

   次の出力が表示されます。

```
NVMeTLSkey-1:01:C50EsaGtuOp8n5fGE9EuWjbBCtshmfoHx4XTqTJUmydf0gIj:
```

6. ONTAPコントローラで、ONTAPサブシステムにTLS PSKを追加します。

出力例を表示します。

```
nvme subsystem host add -vserver vs_iscsi_tcp -subsystem nvme1 -host
-nqn nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-
b2c04f444d33 -tls-configured-psk NVMeTLSkey-
1:01:C50EsaGtuOp8n5fGE9EuWjbBCtshmfoHx4XTqTJUmydf0gIj:
```

7. TLS PSKをホストカーネルキーリングに挿入します。

```
nvme check-tls-key --identity=1 --subsysnqn=nqn.1992
-08.com.netapp:sn.a2d41235b78211efb57dd039eab67a95:subsystem.nvme1
--keydata=NVMeTLSkey
-1:01:C50EsaGtuOp8n5fGE9EuWjbBCtshmfoHx4XTqTJUmydf0gIj: --insert
```

次の TLS キーが表示されます。

```
Inserted TLS key 22152a7e
```

PSKは次のように表示されます `NVMe1R01`使用するため `identity v1` TLS ハンドシェイク アルゴリズムから。Identity v1は、ONTAPがサポートする唯一のバージョンです。

8. TLS PSKが正しく挿入されていることを確認します。

```
cat /proc/keys | grep NVMe
```

出力例を表示します。

```
069f56bb I--Q---     5 perm 3b010000     0     0 psk       NVMe1R01
nqn.2014-08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
nqn.1992-
08.com.netapp:sn.a2d41235b78211efb57dd039eab67a95:subsystem.nvme1
oYVLelmiOwnvDjXKBmrnIgGVpFIBDJtc4hmQXE/36Sw=: 32
```

9. 挿入したTLS PSKを使用してONTAPサブシステムに接続します。

   a. TLS PSK を検証します。

```
nvme connect -t tcp -w 192.168.111.80 -a 192.168.111.66  -n nqn.1992-
08.com.netapp:sn.a2d41235b78211efb57dd039eab67a95:subsystem.nvme1
--tls_key=0x069f56bb –tls
```

次の出力が表示されます。

```
connecting to device: nvme0
```

a. list-subsys を検証します。

```
nvme list-subsys
```

出力例を表示します。

```
nvme-subsys0 - NQN=nqn.1992-
08.com.netapp:sn.a2d41235b78211efb57dd039eab67a95:subsystem.nvme1
               hostnqn=nqn.2014-08.org.nvmexpress:uuid:4c4c4544-
0035-5910-804b-b2c04f444d33
\
 +- nvme0 tcp
traddr=192.168.111.66,trsvcid=4420,host_traddr=192.168.111.80,src
_addr=192.168.111.80 live
```

10. ターゲットを追加し、指定したONTAPサブシステムへのTLS接続を確認します。

```
nvme subsystem controller show -vserver sles15_tls -subsystem sles15
-instance
```

出力例を表示します。

```
(vserver nvme subsystem controller show)
                           Vserver Name: vs_iscsi_tcp
                           Subsystem: nvme1
                       Controller ID: 0040h
                   Logical Interface: tcpnvme_lif1_1
                                Node: A400-12-181
                            Host NQN: nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
                  Transport Protocol: nvme-tcp
         Initiator Transport Address: 192.168.111.80
                      Host Identifier:
4c4c454400355910804bb2c04f444d33
                Number of I/O Queues: 2
                    I/O Queue Depths: 128, 128
                   Admin Queue Depth: 32
               Max I/O Size in Bytes: 1048576
           Keep-Alive Timeout (msec): 5000
                       Subsystem UUID: 8bbfb403-1602-11f0-ac2b-
d039eab67a95
               Header Digest Enabled: false
                 Data Digest Enabled: false
       Authentication Hash Function: sha-256
Authentication Diffie-Hellman Group: 3072-bit
                 Authentication Mode: unidirectional
       Transport Service Identifier: 4420
                         TLS Key Type: configured
                    TLS PSK Identity: NVMe1R01 nqn.2014-
08.org.nvmexpress:uuid:4c4c4544-0035-5910-804b-b2c04f444d33
nqn.1992-
08.com.netapp:sn.a2d41235b78211efb57dd039eab67a95:subsystem.nvme1
oYVLelmiOwnvDjXKBmrnIgGVpFIBDJtc4hmQXE/36Sw=
                          TLS Cipher: TLS-AES-128-GCM-SHA256
```

手順11：既知の問題を確認する

既知の問題はありません。

著作権に関する情報

商標に関する情報