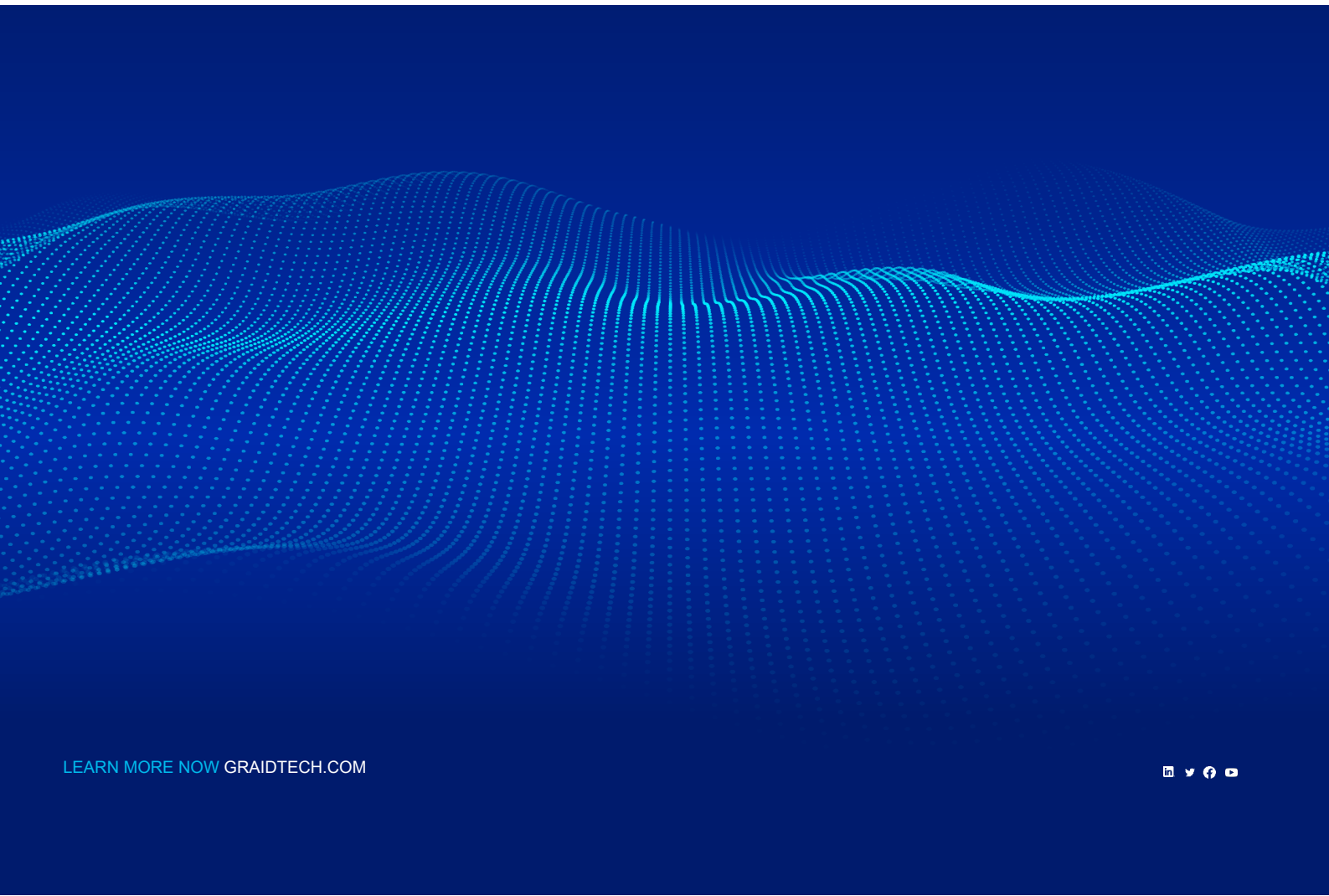
GRAID SupremeRAID™

User Guide for Linux

Version: 1.2.2



Copyright

The term GRAID refers to GRAID Technology Inc. For more information, please visit www.graidtech.com.

GRAID reserves the right to make changes without further notice to any products or content herein to improve reliability, function, or design. GRAID makes no warranty as to the accuracy or completeness of the content or information provided herein, which are provided on an “as is” basis.

No license to GRAID’s or any third party’s intellectual property rights are conveyed hereunder.

Publication: July 1, 2022

Table of Contents

Introduction

GRAID SupremeRAID™ is the most powerful, high-speed data protection solution specially designed for NVMe SSDs. GRAID SupremeRAID™ installs a virtual NVMe controller onto the operating system and integrates a high-performance, AI processor equipped PCIe RAID card into the system to manage the RAID operations of the virtual NVMe controller.

This document explains how to install the SupremeRAID™ software package for Linux and how to manage the RAID components using the command-line interface.

GRAID SupremeRAID™ Specifications

GRAID SupremeRAID™ Driver Specifications	
Supported GRAID Models	SR-1000, SR-1010,SR-1001
Supported RAID levels	RAID 0, 1, 5, 6, 10,
Recommended minimum drive number for each RAID level	RAID 0 : at least two drives RAID 1 : at least two drives RAID 5 : at least three drives RAID 6 : at least four drives RAID 10 : at least four drives
Maximum number of physical drives	32
Maximum number of drive groups	4
Maximum number of virtual drives per drive group	8
Maximum size of the drive group	Defined by the physical drive sizes

Supported Operating Systems

Linux Distro	x86_64	arm64
CentOS	7.9, 8.3, 8.4, 8.5	Not Supported
RHEL	7.9, 8.3, 8.4, 8.5	Not Supported
Rocky Linux	8.5	Not Supported
AlmaLinux	8.5	Not Supported
Ubuntu	20.04	20.04
openSUSE Leap	15.2, 15.3	Not Supported
SLES	15 SP2, 15 SP3	Not Supported

Installation

This section describes installing the GRAID SupremeRAID™ software package for Linux.

Prerequisites

Before installing the software package, ensure that the system meets the following requirements:

1. Minumum system requirements:
    - CPU: 2 GHz or faster with at least 8 cores
    - RAM: 16 GB
    - An available PCIe Gen3 or Gen4 x16 slot
2. The GRAID SupremeRAID™ card is installed into a PCIe x16 slot
3. The IOMMU function is disabled in the system BIOS.
4. The GRAID SupremeRAID™ software package is downloaded from the GRAID, or GRAID partner, website.

Installing on Linux using the Pre-installer

The GRAID pre-installer is an executable file that contains the required dependencies and a setup script that installs the NVIDIA driver. The script makes it easy to prepare the environment and install the GRAID SupremeRAID™ driver in every supported Linux distriubution. Use following steps to prepare the environment and install the GRAID SupremeRAID™ driver using the pre-installer in supported Linux distributions.

All supported distro

From a terminal that does not run the GUI console:

1. Download the latest version of the pre-installer and make it to executable.

```
$ wget https://download.graidtech.com/driver/pre-install/<graid-sr-pre-installer-x.x.x-xx.run>
$ chmod +x <graid-sr-pre-installer-x.x.x-xx.run>
```

2. Execute the pre-installer and follow the instructions to complete the pre-installation process as shown below.

```
[root@graid ~]# sudo ./graid-sr-pre-installer-1.0.0-31.run
Extracting installer files, please wait a few seconds ...
Extracting installer files done.

Starting installer ...
This pre-installer is for CentOS
Running service check
Service check succeeded.

Running system check
System check succeeded.

Running install packages and kernel setting.
Generating grub configuration file ...
Adding boot menu entry for EFI firmware configuration
done
Generating new initramfs...
Generated new initramfs.
Install packages and kernel setting succeeded.

chvt: ioctl VT_ACTIVATE: No such device or address
Disabled Xorg instance.
Nouveau module has been loaded, pre-installer will unload nouveau for NVIDIA driver install.
Unload nouveau module successfully.
Running install NVIDIA Driver. (This step will take a while.)
Mon Oct  4 14:30:35 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA T1000        Off  | 00000000:13:00.0 Off |                  N/A |
| 37%   39C    P0    N/A /  50W |      0MiB /  3911MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
Install NVIDIA Driver succeeded.

This pre-installer will reboot the system for apply previous setting!
Do you want to continue? [Y/n]
```

3. The system must be rebooted after running the pre-installation script. When prompted, type Y to reboot the system.
4. Download the latest version of the SupremeRAID™ driver installer from the GRAID website and make it to executable.

```
$ wget https://download.graidtech.com/driver/../../<graid-sr-installer-x.x.x-xx.run>
$ chmod +x <graid-sr-installer-x.x.x-xx.run>
```

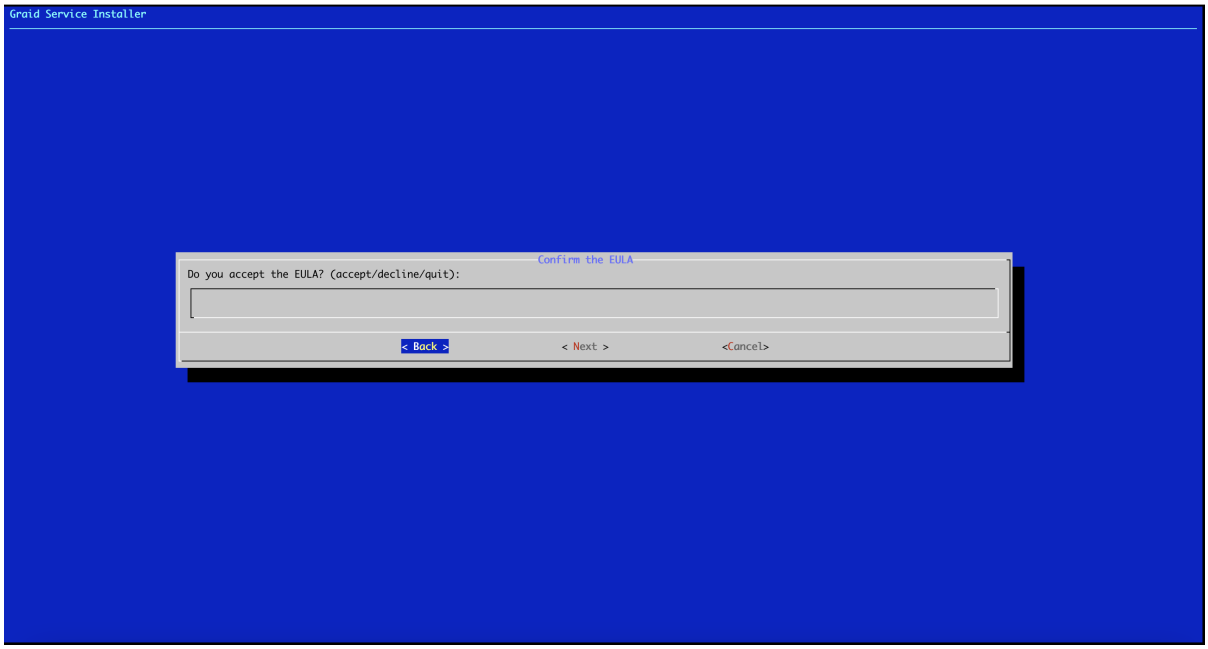
5. Execute the installer and follow the instructions to complete the installation process as shown below.

```
sudo ./<graid-sr-installer-x.x.x-xx.run>
```

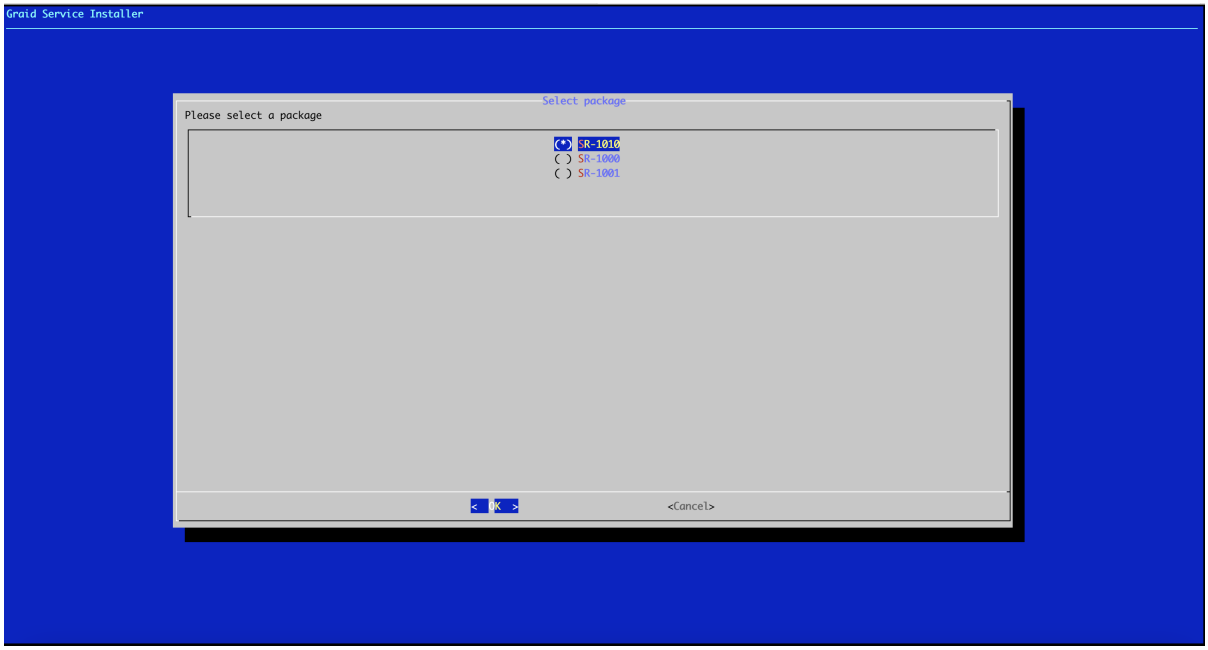
1. Welcome page, please select next and click enter to view the end-user license agreement

2. In end-user license agreement page, you can click space to scroll down the license content, after you review the license, select next and click enter to proceed

3. Type **accept**, click tab, select next and click enter to accept the license agreement



4. Use the up/down key to switch between the models and click space to select the model you want to install. Once the modal has been selected, you select ok and click enter to proceed



6. Apply the SupremeRAID™ license key.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Installing on Linux Manually

If you prefer, you can manually install the dependencies required by the SupremeRAID™ driver.

CentOS, Rocky Linux, AlmaLinux and RHEL

1. Install the package dependencies and build tool for dkms.

   For CentOS, Rocky Linux and AlmaLinux

   ```
   $ sudo yum install --enablerepo=extras epel-release
   $ sudo yum install vim wget make automake gcc gcc-c++ kernel-devel kernel-headers kernel dkms ipmitool
   tar mdadm sg3_utils sqlite-libs automake dialog
   ```

   For RHEL

   ```
   $ sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
   $ sudo yum install  gcc-$(awk -F'=' '/VERSION_ID/{ gsub(/"/,""); print $2}' /etc/os-release) gcc-
   c++-$(awk -F'=' '/VERSION_ID/{ gsub(/"/,""); print $2}' /etc/os-release)
   $ sudo yum install vim wget make automake kernel-devel-$(uname -r) kernel-headers-$(uname -r) dkms
   ipmitool tar mdadm sg3_utils sqlite-libs automake dialog
   ```

2. Add the kernel option. This step prevents the Nouveau driver from loading during installation and disables IOMMU in the system BIOS.

   ```
   $ sudo vim /etc/default/grub
   ```

3. Append ***"iommu=pt"*** and ***"nvme_core.multipath=Y"*** to **GRUB_CMDLINE_LINUX** then update the grub configuration:

   ```
   $ sudo grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

4. Append **"blacklist nouveau"** and **"options nouveau modeset=0"** to the end of the **/etc/modprobe.d/graid-blacklist.conf** file to disable the Nouveau driver and update initramfs.

   ```
   graid@graid-demo:~$ cat graid-blacklist.conf
   blacklist nouveau
   options nouveau modeset=0
   ```

   > **Note:**
   > You might need to manually create the /etc/modprobe.d/graid-blacklist.conf file and append **"blacklist nouveau"** and **"options nouveau modeset=0"**.

   ```
   $ sudo update-initramfs -u
   ```

For CentOS, Rocky Linux and AlmaLinux

Please find out the latest verion of kernel and assign to --kver

```
$ sudo dracut -f --kver `rpm -qa | grep kernel-headers | awk -F'kernel-headers-' {'print $2'}`
```

For RHEL

```
$ sudo dracut -f
```

5. Reboot the system and ensure that the grub configuration was applied. You can check **/proc/cmdline** for the grub configuration in use. For example:

```
[root@graid ~]# cat /proc/cmdline
BOOT_IMAGE=(hd3,gpt8)/vmlinuz-4.18.0-305.17.1.el8_4.x86_64 root=UUID=ba33c54d-74c9-409d-ae05-db27cacd68b3 ro crashkernel=auto resume=UUID=ae5b3808-b657-4593-a598-c5cbc5a87105 rhgb quiet rd.driver.blacklist=nouveau iommu=pt nvme_core.multipath=Y
```

6. Install the NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.86/NVIDIA-Linux-x86_64-470.86.run
$ chmod +x NVIDIA-Linux-x86_64-470.86.run
```

For CentOS

Use the latest version of kernel-headers to install the NVIDIA driver.

```
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms -k `rpm -qa | grep kernel-headers | awk -F'kernel-headers-' {'print $2'}`
```

For RHEL

```
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

Step 3 disables the Nouveau driver. Using the following command, the NVIDIA driver installer can also disable the Nouveau driver. After disabling the Nouveau driver, you must reboot and re-install the NVIDIA driver.

```
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --disable-nouveau
$ sudo reboot
```

7. Confirm that the NVIDIA GPU is working using the ***nvidia-smi*** command. Example output of a successful installation is shown below:

```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

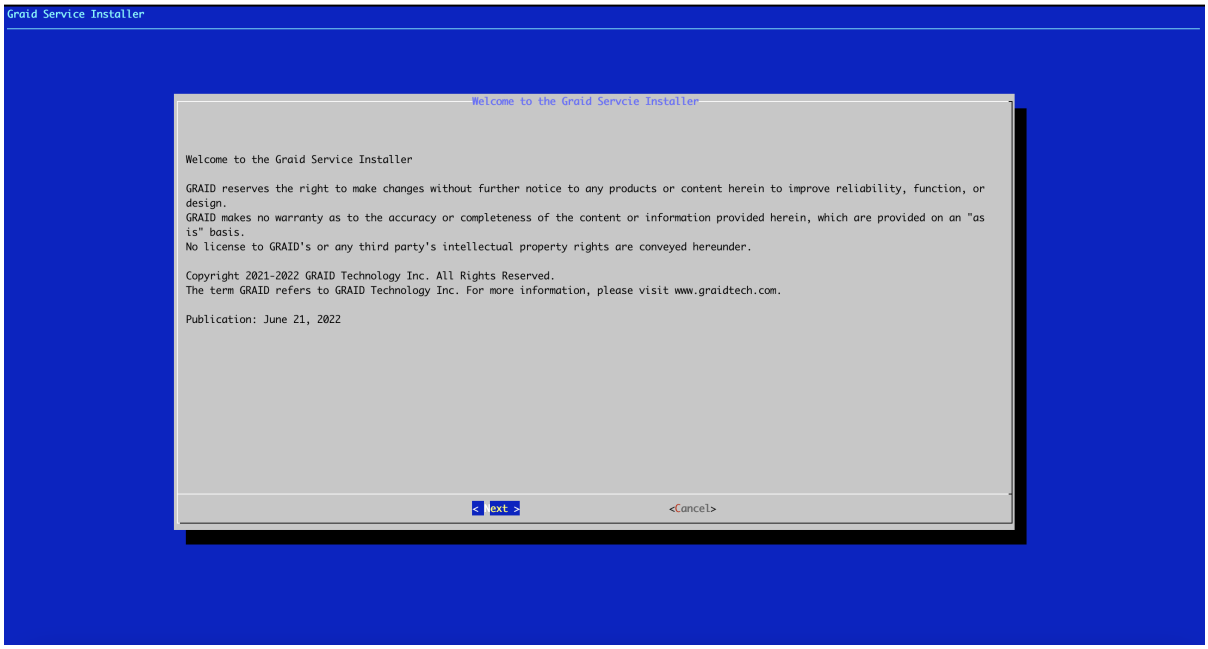
8. Download the latest version of the SupremeRAID™ driver installer from the GRAID website and make it to executable.

```
$ wget https://download.graidtech.com/driver/../../<graid-sr-installer-x.x.x-xx.run>
$ chmod +x <graid-sr-installer-x.x.x-xx.run>
```

9. Execute the installer and follow the instructions to complete the installation process as shown below.

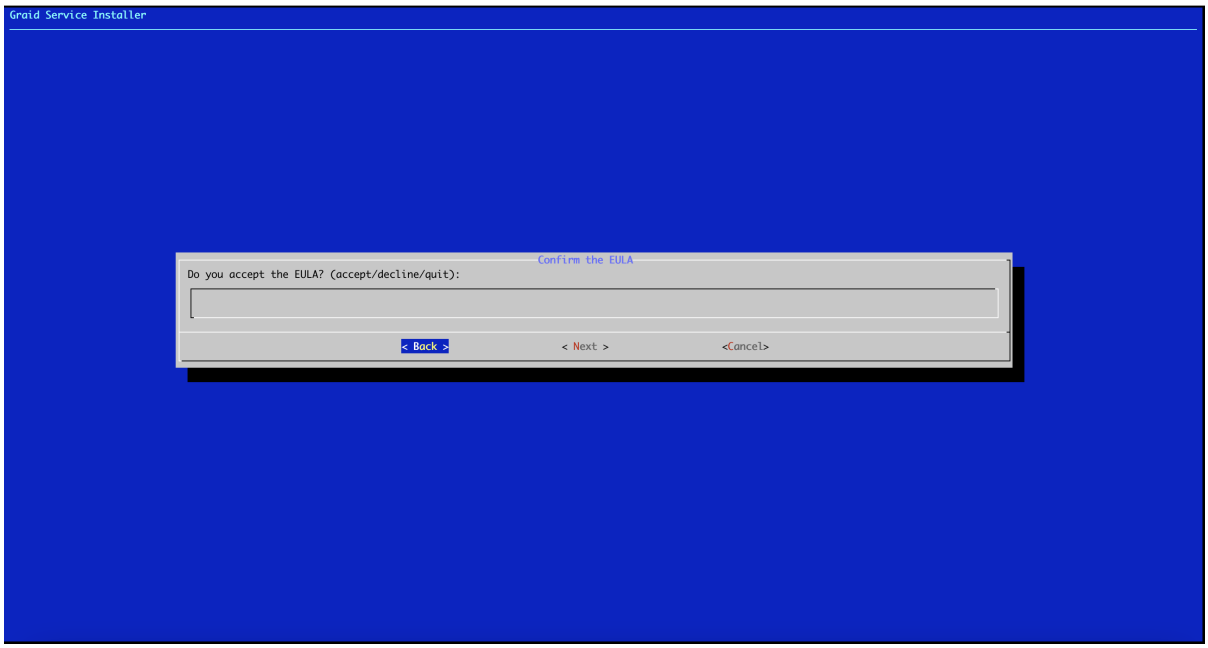
```
sudo ./<graid-sr-installer-x.x.x-xx.run>
```

1. Welcome page, please select next and click enter to view the end-user license agreement

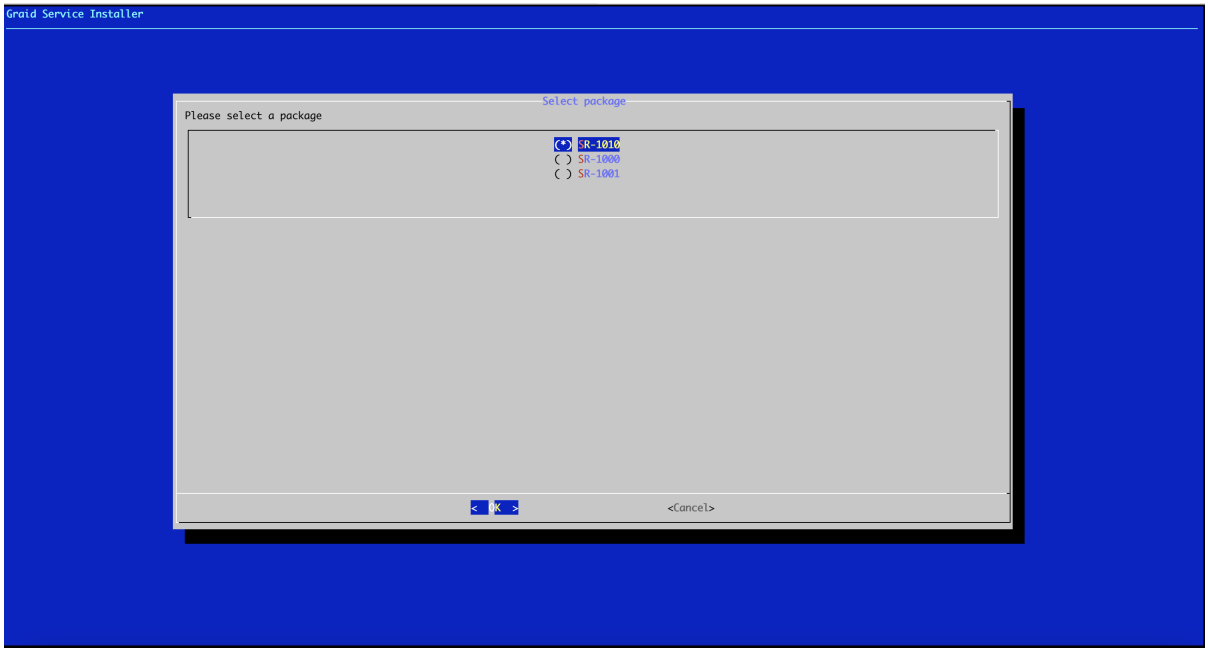


2. In end-user license agreement page, you can click space to scroll down the license content, after you review the license, select next and click enter to proceed

3. Type **accept**, click tab, select next and click enter to accept the license agreement



4. Use the up/down key to switch between the models and click space to select the model you want to install. Once the modal has been selected, you select ok and click enter to proceed



10. Apply the SupremeRAID™ license key.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Ubuntu

1. Install the package dependencies and build tool for dkms.

```
$ sudo apt-get update
$ sudo apt-get install make automake gcc g++ linux-headers-$(uname -r) dkms ipmitool initramfs-tools
tar mdadm libsgutils2-2 libudev-dev libpci3 sqlite automake dialog
```

2. Disable Ubuntu daily upgrade.

```
$ sed -i '/Unattended-Upgrade "1"/ s/"1"/"0"/' /etc/apt/apt.conf.d/20auto-upgrades
$ sed -i '/Update-Package-Lists "1"/ s/"1"/"0"/' /etc/apt/apt.conf.d/20auto-upgrades
```

3. Add the kernel option. This step prevents the Nouveau driver from loading during installation and disables IOMMU in the system BIOS.

```
$ sudo vim /etc/default/grub
```

4. Append ***"iommu=pt"*** and ***"nvme_core.multipath=Y"*** to **GRUB_CMDLINE_LINUX** then update the grub configuration:

```
$ sudo update-grub
```

5. Append **"blacklist nouveau"** and **"options nouveau modeset=0"** to the end of the **/etc/modprobe.d/graid-blacklist.conf** file to disable the Nouveau driver and update initramfs.

```
graid@graid-demo:~$ cat graid-blacklist.conf
blacklist nouveau
options nouveau modeset=0
```

> **Note:**
> You might need to manually create the /etc/modprobe.d/graid-blacklist.conf file and append **"blacklist nouveau"** and **"options nouveau modeset=0"**.

```
$ sudo update-initramfs -u
```

6. Reboot the system and ensure that the grub configuguration was applied. You can check **/proc/cmdline** for the grub configuration in use. For example:

```
root@graid:~/# cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-5.4.0-74-generic root=/dev/mapper/ubuntu--vg-ubuntu--lv ro rd.driver.blacklist=nouveau iommu=pt nvme_core.multipath=Y
```

7. Install the NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.86/NVIDIA-Linux-x86_64-470.86.run
$ chmod +x NVIDIA-Linux-x86_64-470.86.run
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

Step 3 disables the Nouveau driver. Using the following command, the NVIDIA driver installer can also disable the Nouveau driver. After disabling the Nouveau driver, you must reboot and re-install the NVIDIA driver.

```
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --disable-nouveau
$ reboot
```

8. Confirm that the NVIDIA GPU is working using the ***nvidia-smi*** command. Example output of a successful installation is shown below:

```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

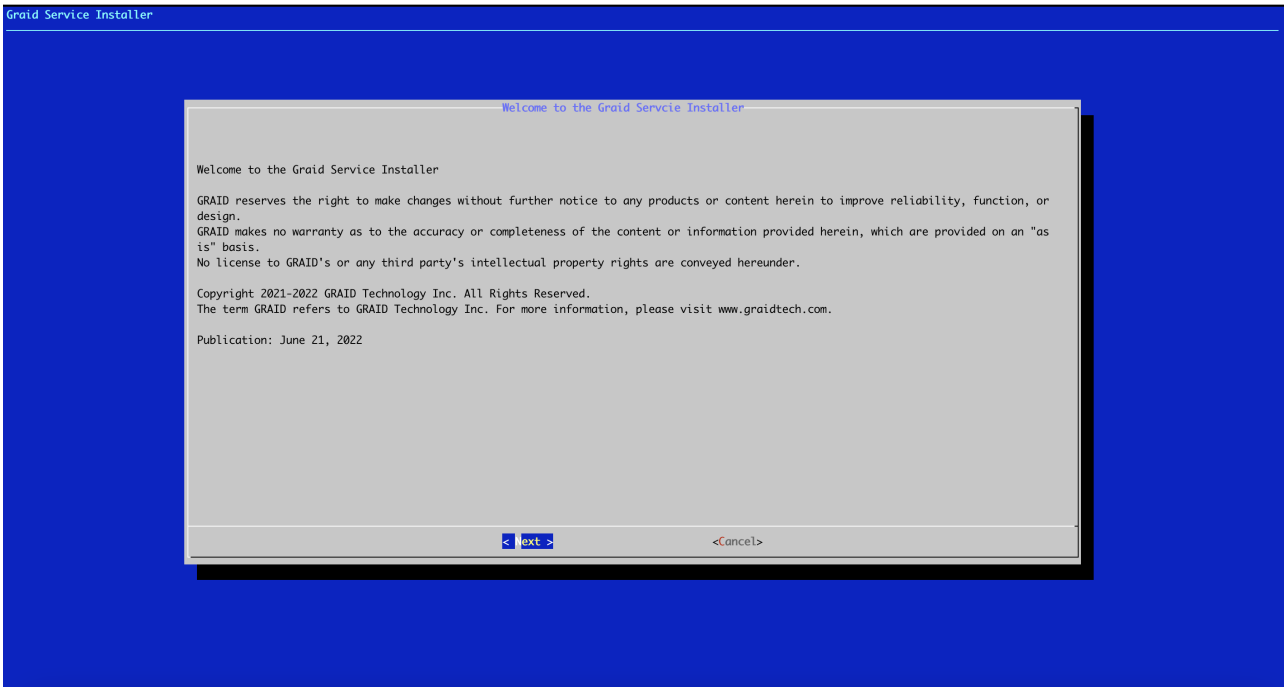
9. Download the latest version of the SupremeRAID™ driver installer from the GRAID website and make it to executable.

```
$ wget https://download.graidtech.com/driver/../../<graid-sr-installer-x.x.x-xx.run>
$ chmod +x <graid-sr-installer-x.x.x-xx.run>
```

10. Execute the installer and follow the instructions to complete the installation process as shown below.

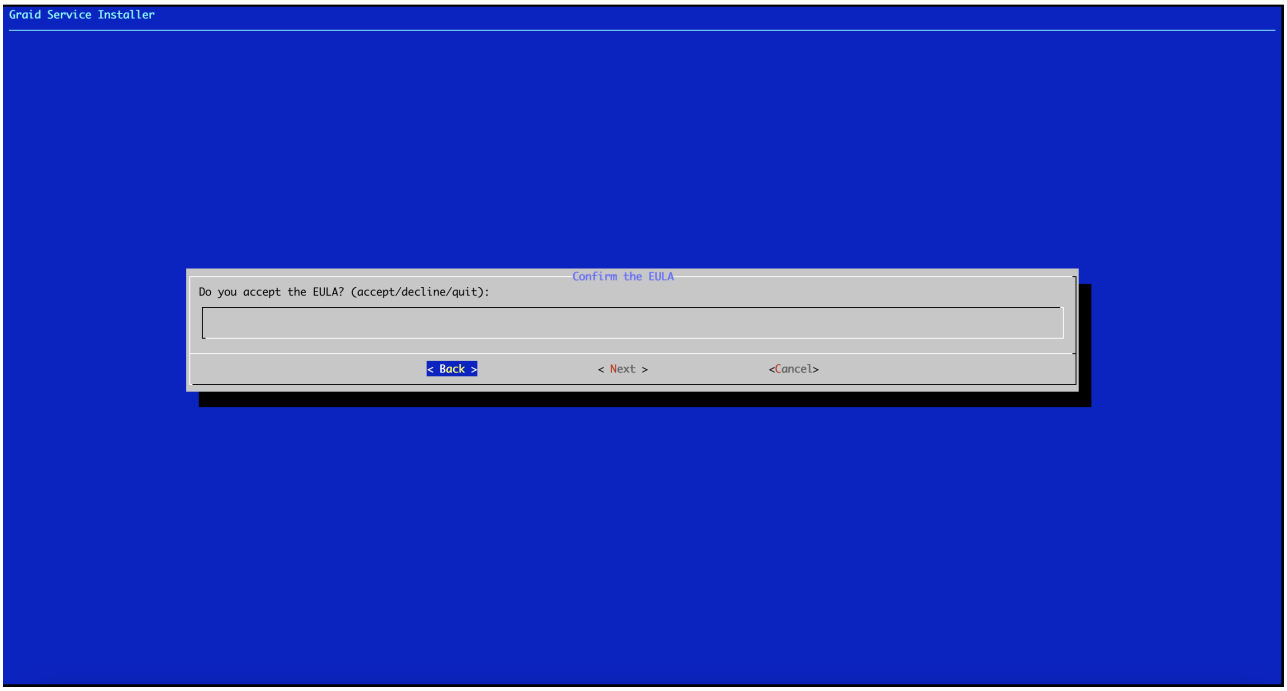
```
sudo ./<graid-sr-installer-x.x.x-xx.run>
```

1. Welcome page, please select next and click enter to view the end-user license agreement

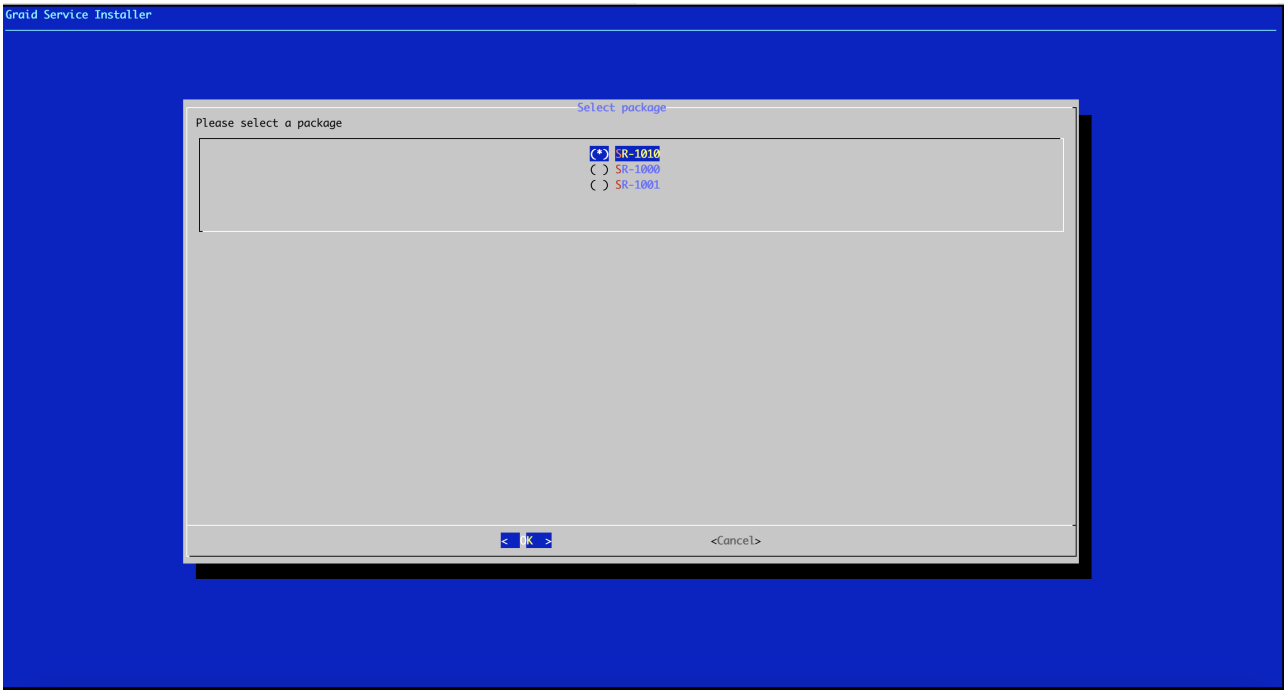


2. In end-user license agreement page, you can click space to scroll down the license content, after you review the license, select next and click enter to proceed

3. Type **accept**, click tab, select next and click enter to accept the license agreement



4. Use the up/down key to switch between the models and click space to select the model you want to install. Once the modal has been selected, you select ok and click enter to proceed



5. Apply the SupremeRAID™ license key.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

openSUSE

1. Install openSUSE and select all online repositories.

2. Install the package dependencies and build tool for dkms.

```
$ sudo zypper addrepo -f https://download.opensuse.org/distribution/leap/15.2/repo/oss/ leap-15.2
$ sudo zypper --gpg-auto-import-keys refresh
$ sudo zypper install sudo vim wget libpci3 dkms ipmitool tar mdadm libsgutils-devel libudev-devel
sqlite3 automake dialog
$ sudo zypper install -C kernel-default-devel=$(uname -r | awk -F"-default" '{print $1}')
```

3. Add the kernel option. This step prevents the Nouveau driver from loading during installation and disables IOMMU in the system BIOS.

```
$ sudo vim /etc/default/grub
```

4. Append ***"iommu=pt"*** and ***"nvme_core.multipath=Y"*** to **GRUB_CMDLINE_LINUX** then update the grub configuration:

```
$ sudo update-grub
```

5. Append **"blacklist nouveau"** to the end of the **/etc/modprobe.d/graid-blacklist.conf** file to disable the Nouveau driver.

```
graid@graid-demo:~$ cat graid-blacklist.conf
blacklist nouveau
options nouveau modeset=0
```

> **Note:**
> You might need to manually create the /etc/modprobe.d/graid-blacklist.conf file and append **"blacklist nouveau"** and **"options nouveau modeset=0"**.

6. Set the **allow_unsupported_modules** option to **1** in the **/etc/modprobe.d/10-unsupported-modules.conf** file and update initrd.

```
$ sudo mkinitrd
```

7. Reboot the system and ensure that the grub configuguration was applied. You can check **/proc/cmdline** for the grub configuration in use. For example:

```
root@graid:~ # cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.3.18-59.5-default root=UUID=7560fe42-0275-4618-b8a0-0785765610c9 modprobe.blacklist=nouveau iommu=pt splash=silent quiet
mitigations=auto nvme_core.multipath=Y
```

8. Install the NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.86/NVIDIA-Linux-x86_64-470.86.run
$ chmod +x NVIDIA-Linux-x86_64-470.86.run
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

Step 3 disables the Nouveau driver. Using the following command, the NVIDIA driver installer can also disable the Nouveau driver. After disabling the Nouveau driver, you must reboot and re-install the NVIDIA driver.

```
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --disable-nouveau
$ reboot
```

9. Confirm that the NVIDIA GPU is working using the ***nvidia-smi*** command. Example output of a successful installation is shown below:

```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

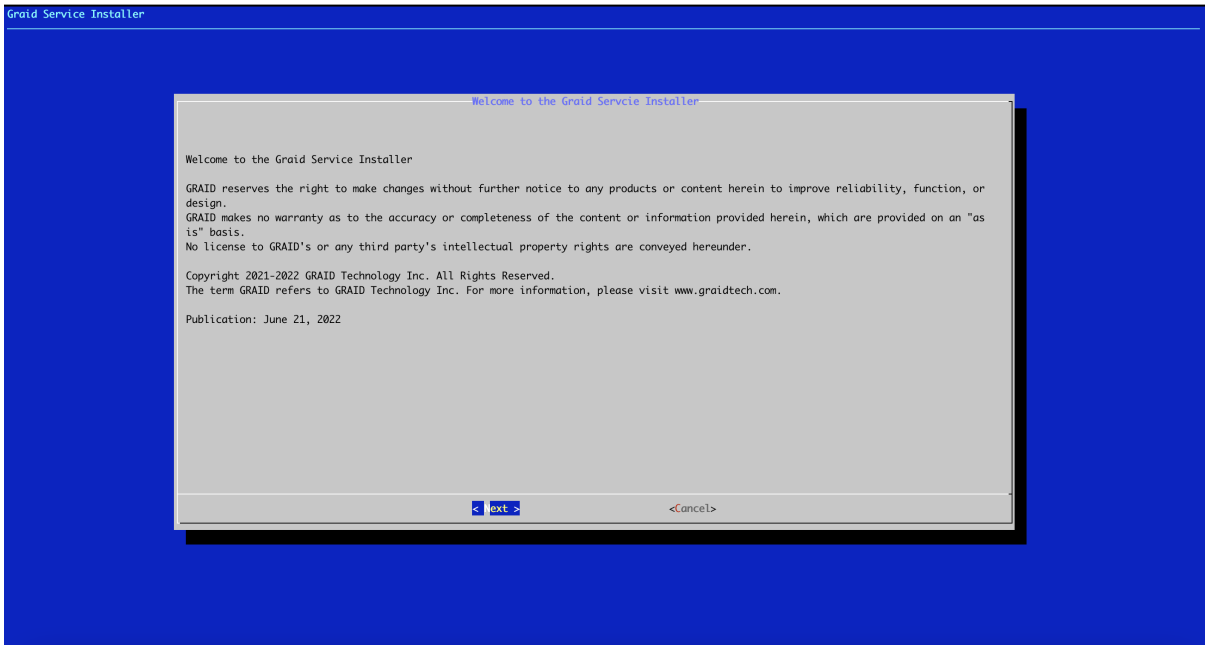
10. Download the latest version of the SupremeRAID™ driver installer from the GRAID website and make it to executable.

```
$ wget https://download.graidtech.com/driver/../../<graid-sr-installer-x.x.x-xx.run>
$ chmod +x <graid-sr-installer-x.x.x-xx.run>
```

11. Execute the installer and follow the instructions to complete the installation process as shown below.

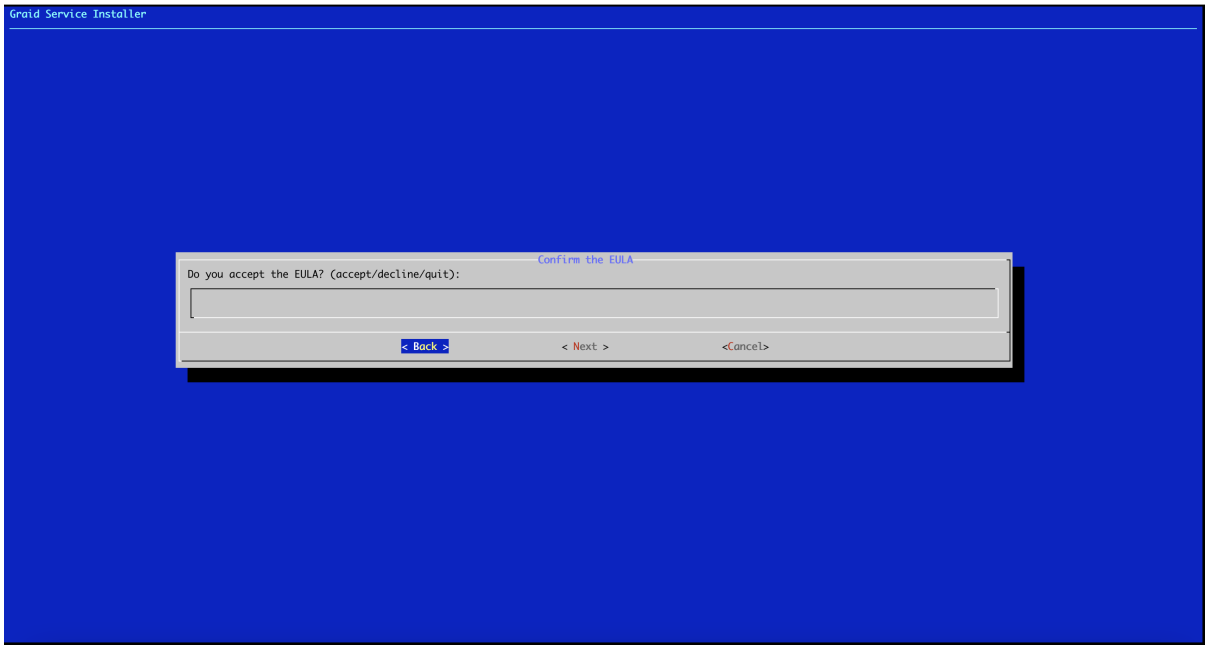
```
sudo ./<graid-sr-installer-x.x.x-xx.run>
```

    1. Welcome page, please select next and click enter to view the end-user license agreement

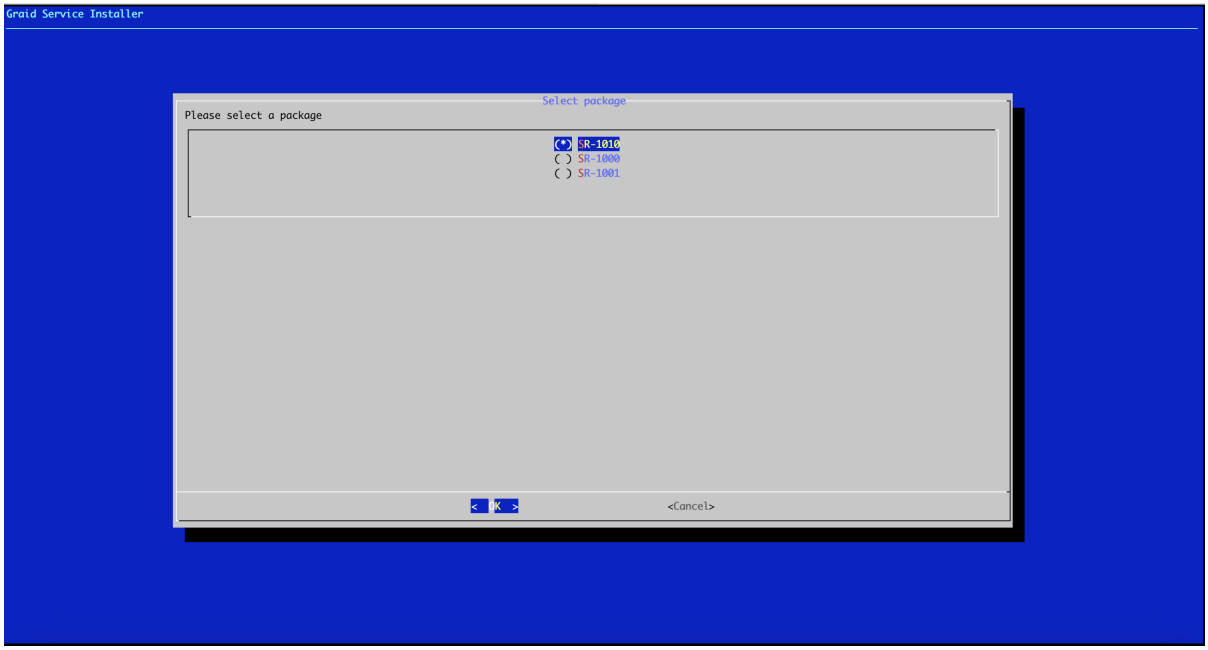


2. In end-user license agreement page, you can click space to scroll down the license content, after you review the license, select next and click enter to proceed

3. Type **accept**, click tab, select next and click enter to accept the license agreement



4. Use the up/down key to switch between the models and click space to select the model you want to install. Once the modal has been selected, you select ok and click enter to proceed



12. Apply the SupremeRAID™ license key.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

SLES

1. Install SLES with the following extensions and modules:

    1. SUSE Package Hub 15 SP2 x86_64
    2. Desktop Applications Module 15 SP2 x86_64
    3. Development Tools Module 15 SP2 x86_64

2. Install the package dependencies and build tool for dkms.

```
$ sudo zypper addrepo -f https://download.opensuse.org/distribution/leap/15.2/repo/oss/ leap-15.2
$ sudo zypper --gpg-auto-import-keys refresh
$ sudo zypper install sudo vim wget libpci3 dkms ipmitool tar mdadm libsgutils-devel libudev-devel
sqlite3 automake dialog
$ sudo zypper install -C kernel-default-devel=$(uname -r | awk -F"-default" '{print $1}')
```

3. Add the kernel option. This step prevents the Nouveau driver from loading during installation and disables IOMMU in the system BIOS.

```
$ sudo vim /etc/default/grub
```

4. Append ***"iommu=pt"*** and ***"nvme_core.multipath=Y"*** to **GRUB_CMDLINE_LINUX** then update the grub configuration:

```
$ sudo update-grub
```

5. Append **"blacklist nouveau"** to the end of the **/etc/modprobe.d/graid-blacklist.conf** file to disable the Nouveau driver.

```
graid@graid-demo:~$ cat graid-blacklist.conf
blacklist nouveau
options nouveau modeset=0
```

> **Note:**
> You might need to manually create the /etc/modprobe.d/graid-blacklist.conf file and append **"blacklist nouveau"** and **"options nouveau modeset=0"**.

6. Set the **allow_unsupported_modules** option to **1** in the **/etc/modprobe.d/10-unsupported-modules.conf** file.

7. Update initrd.

```
$ sudo mkinitrd
```

8. Reboot the system and ensure that the grub configuguration was applied. You can check **/proc/cmdline** for the grub configuration in use. For example:

```
root@graid:~ # cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-5.3.18-59.5-default root=UUID=7560fe42-0275-4618-b8a0-0785765610c9 modprobe.blacklist=nouveau iommu=pt splash=silent quiet
mitigations=auto nvme_core.multipath=Y
```

9. Install the NVIDIA driver.

```
$ wget https://us.download.nvidia.com/XFree86/Linux-x86_64/470.86/NVIDIA-Linux-x86_64-470.86.run
$ chmod +x NVIDIA-Linux-x86_64-470.86.run
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --no-systemd --no-opengl-files --no-nvidia-modprobe --dkms
```

Step 3 disables the Nouveau driver. Using the following command, the NVIDIA driver installer can also disable the Nouveau driver. After disabling the Nouveau driver, you must reboot and re-install the NVIDIA driver.

```
$ sudo ./NVIDIA-Linux-x86_64-470.86.run -s --disable-nouveau
$ reboot
```

10. Confirm that the NVIDIA GPU is working using the ***nvidia-smi***command. Example output of a successful installation is shown below:

```
[root@graid ~]# nvidia-smi
Tue Sep 21 21:27:37 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA T1000        Off  | 00000000:81:00.0 Off |                  N/A |
| 34%   38C    P0    N/A /  50W |      0MiB /  3911MiB |      0%   E. Process |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

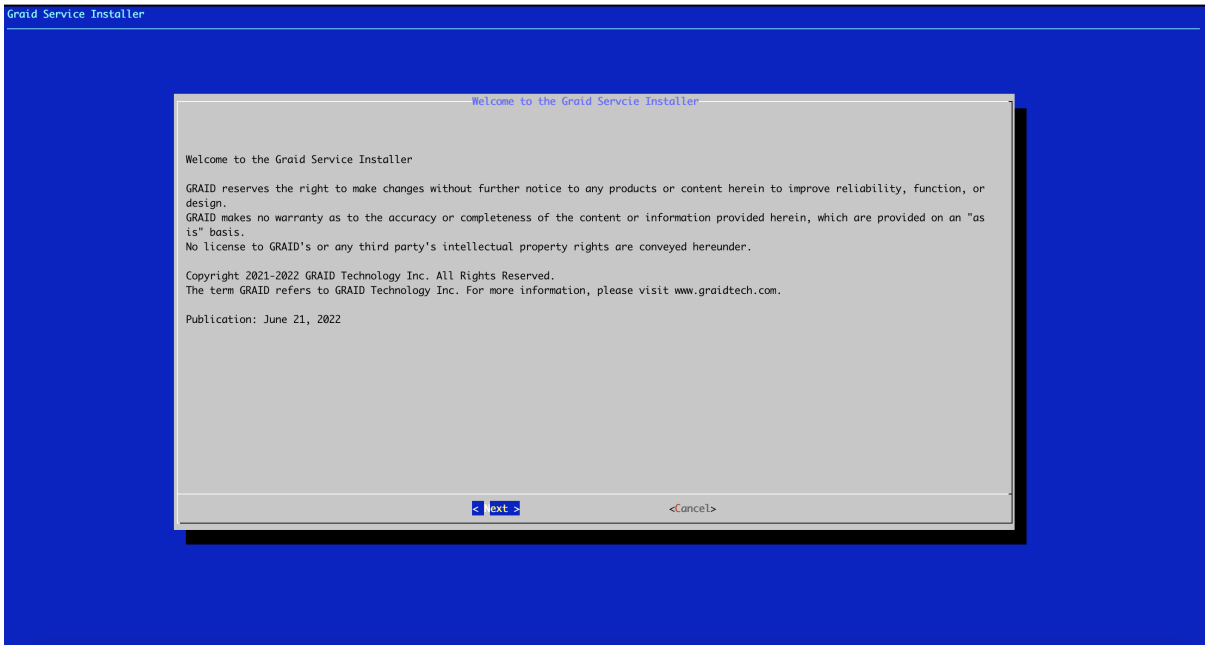
11. Download the latest version of the SupremeRAID™ driver installer from the GRAID website and make it to executable.

```
$ wget https://download.graidtech.com/driver/../../<graid-sr-installer-x.x.x-xx.run>
$ chmod +x <graid-sr-installer-x.x.x-xx.run>
```

12. Execute the installer and follow the instructions to complete the installation process as shown below.

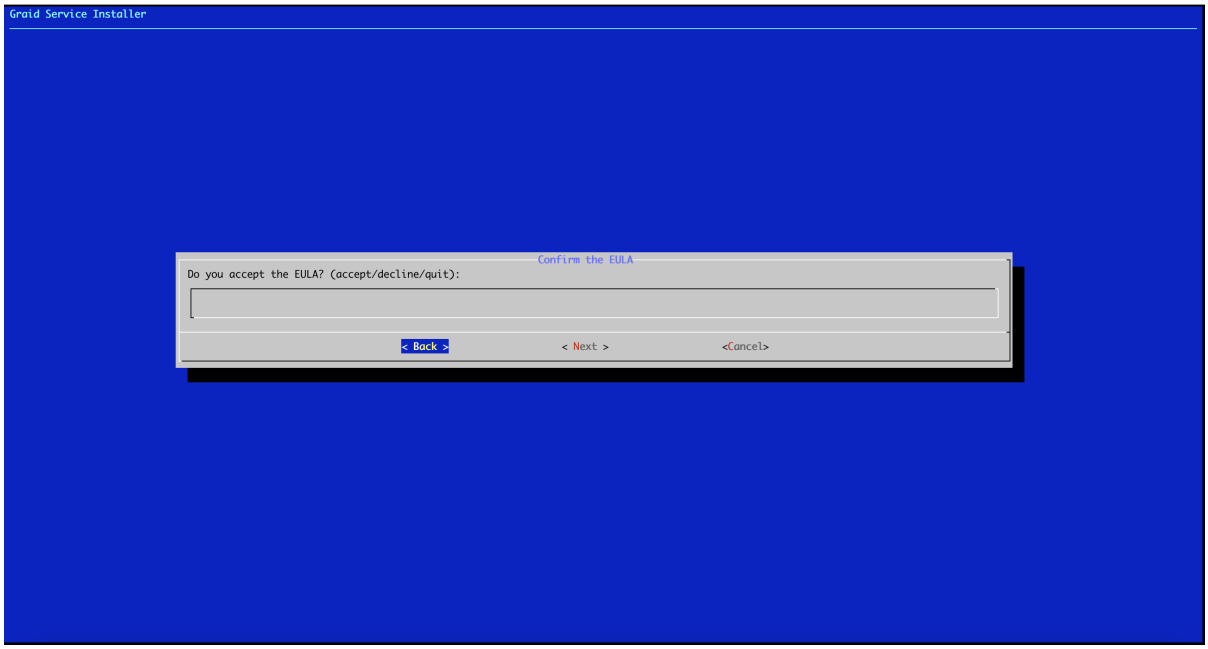
```
sudo ./<graid-sr-installer-x.x.x-xx.run>
```

    1. Welcome page, please select next and click enter to view the end-user license agreement

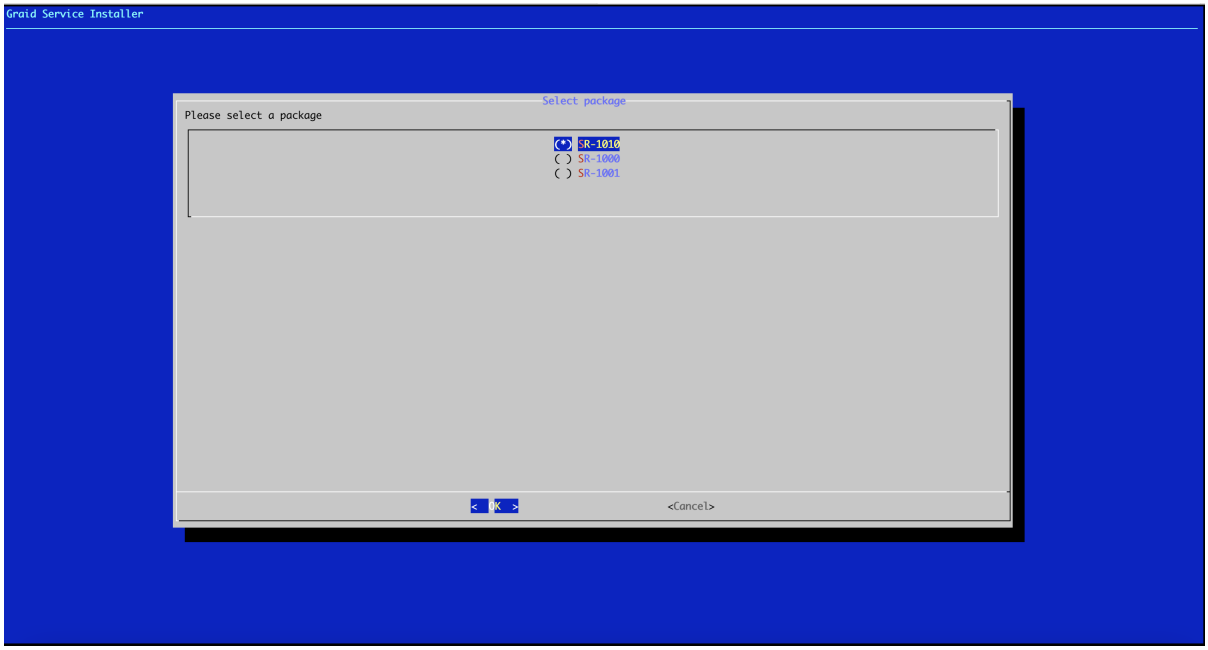


2. In end-user license agreement page, you can click space to scroll down the license content, after you review the license, select next and click enter to proceed

3. Type **accept**, click tab, select next and click enter to accept the license agreement



4. Use the up/down key to switch between the models and click space to select the model you want to install. Once the modal has been selected, you select ok and click enter to proceed



13. Apply the SupremeRAID™ license key.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Upgrading the Software

> **Note:**
> You must exactly follow the steps below to upgrade the software.

To upgrade the software:

1. Stop all applications running on the virtual drive.
2. Stop the management service.

```
$ sudo systemctl stop graid
```

3. Ensure that the GRAID kernel module is unloaded.

```
$ sudo rmmod graid
```

4. Check the NVIDIA driver DKMS status.

```
$ sudo dkms status nvidia
```

> **Note:**
> The NVIDIA driver version that is installed in the kernel must match the GRAID driver version. Perform step 5, Uninstall the NVIDIA Driver, when the versions do not match.

5. **[Optional]** Uninstall the NVIDIA Driver

> **Note:**
> This step is only required when the NVIDIA driver version and the GRAID driver version do not match.

   - Dracut the initramfs.

```
$ dracut --omit-drivers "nvidia graid" -f
```

   - Uninstall the NVIDIA driver.

```
$ sudo ./usr/bin/nvidia-uninstall
```

   - Install New Nvidia driver
   - Reboot

6. Uninstall the package.
   - In Centos, Rocky Linux, AlmaLinux, RHEL, openSUSE and SLES

```
$ sudo rpm -e graid-sr
```

- In Ubuntu

```
$ sudo dpkg -r graid-sr
```

7. Confirm that the GRAID module is unloaded.

```
$ sudo lsmod | grep graid
```

There should not be any ouput.

8. Confirm that the GRAID package is uninstalled.

- In Centos, Rocky Linux, AlmaLinux, RHEL, openSUSE and SLES

```
$ sudo rpm -qa | grep graid
```

- In Ubuntu

```
$ sudo dpkg -l | grep graid
```

There should not be any output.

9. Install the new package.

```
sudo ./<graid-sr-installer-x.x.x-xx.run>
```

10. Start the management service.

```
$ sudo systemctl enable graid
$ sudo systemctl start graid
```

Management

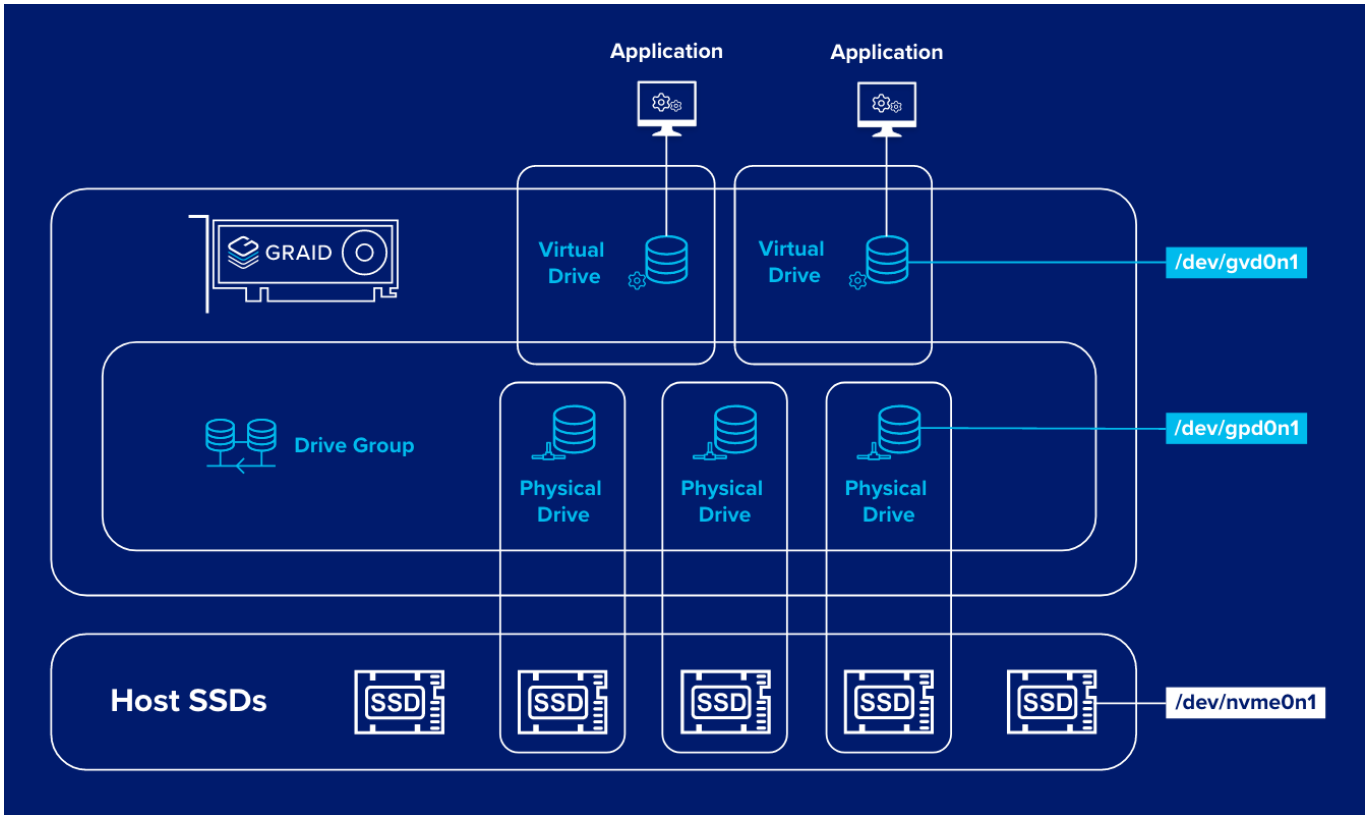
Overview of the GRAID SupremeRAID™ Software Module

There are three major components of the GRAID SupremeRAID™ Software Module:

1. **graidctl** - The command-line management tool.
2. **graid_server** - The management daemon that handles requests from graidctl to control the driver.
3. **graid.ko** - The driver kernel module.

RAID Components

There are three major RAID logical components in SupremeRAID™, the **Physical Drive (PD)**, the **Drive Group (DG)**, and the **Virtual Drive (VD)**.



Physical Drive (PD)

Since NVMe drives are not directly attached to the SupremeRAID™ controller, you must tell the controller which SSDs can be managed. Once an SSD has been created as a physical drive, the SupremeRAID™ driver unbinds the SSD from the operating system, meaning the device node (/dev/nvmeX) will disappear and is no longer accessible. At the same time, a corresponding device node is created (/dev/gpdX) by the SupremeRAID™ driver. You can check the SSD information, such as SSD model or SMART logs, using this device node. To control and access the SSD using /dev/nvmeXn1, you must first delete the corresponding physical drive.

Currently, SupremeRAID™ supports a total of **32** physical drives, regardless of whether the physical drives are created from a native NVMe SSD, a drive connected through NVMe-oF, or a SAS/SATA disk.

Drive Group (DG)

The main component of RAID logic is a RAID group. When the drive group is created, the SupremeRAID™ driver initializes the physical drives with the corresponding RAID mode to ensure that the data and the parity are synchronized. There are two types of the initialization processes.

- Fast Initilization: When all of the physical drives in the drive group (DG) support the deallocate dataset management command, the SupremeRAID™ driver performs fast initialization by default, meaning the drive group state is optimized immediately.
- Background Initialization: Performance will be slightly affected by the initialization traffic, but you can still create the virtual drive and access the virtual drive during a background initialization.

Currently, SupremeRAID™ supports a total of **4** drive groups, with a maximum of **32** physical drives in one drive group.

Virtual Drive (VD)

The virtual drive is equivalent to the RAID volume. You can create multiple virtual drives in the same drive group for multiple applications. The corresponding device node (/dev/nvmeXn1) appears on the operating system when you create a virtual drive, and you can make the file system or running application directly on this device node. Currently, the SupremeRAID™ driver supports a maximum of **8** virtual drives in each drive group.

Overview of graidctl

Syntax

Use the following syntax to run graidctl commands from the terminal window:

```
graidctl [command] [OBJECT_TYPE] [OBJECT_ID] [flags]
```

where **command**, **OBJECT_TYPE**, **OBJECT_ID**, and **flags** are:

- **command**: Specifies the operation to perform on one or more resources, for example create, list, describe, and delete.
- **OBJECT_TYPE**: Specifies the object type. Object types are case-sensitive, for example license, physical_drive, and drive_group.
- **OBJECT_ID**: Specifies the object ID. Some commands support simultaneous operations on multiple objects. You can specify the OBJECT_ID individually, or you can use a dash to describe an OBJECT_ID range.
  For example, to delete physical drives 1, 3, 4, and 5 simultaneously:

  ```
  $ sudo graidctl delete physical_drive 1 3-5
  ```

- **flags**: Specifies optional flags.

  For example:

  - **--force**

    Forces the deletion of a physical drive.

    ```
    $ sudo graidctl delete physical_drive 0 --force
    ```

  - **--json**

    Print output in json format. This flag can also assist with API implementation.

    ```
    $ sudo graidctl list virtual_drive --format json
    ```

To get help, run *graidctl help* from the terminal window.

Managing Licenses

You can apply the license and check license information.

Applying the License

To apply the license and complete the installation, run:

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Output example for invalid and valid licenses is shown below:

```
[graid@graid-demo ~]$ sudo graidctl apply license 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWJ
✔Apply license successfully.
[graid@graid-demo ~]$ sudo graidctl apply license 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWI
✖Apply license failed: LicenseApply: Failed to apply license 34B45F67-5694YXQB-I4LHTCAT-VYW6CXWI
```

> **Note:**
>
> When applying the license, you might need to provide the serial number of the NVIDIA GPU to GRAID Technical Support.
>
> To obtain NVIDIA GPU serial number, run:

```
$ sudo nvidia-smi --query-gpu=name,index,serial --format=csv
```

This command will list all Nvidia cards in your environment and their serial number.

Checking License Information

To obtain the license information, run:

```
$ sudo graidctl describe license
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl describe license
✔Describe license successfully.
License State:      APPLIED
License Key:        3EQ6USVG-855YRIT2-JFH6HEB6-HICCUKZ5
Expiration Days:    Unlimited
Features:
                    PD Number: 32
                    RAID5: true
                    RAID6: true
                    NVMe-over-Fabric: true
```

Output content:

Field	Description
License State	The license state.
License Key	The applied license key.
Feature	The feature set of the license key.
ExpDays	The expiration date of the license key.

The license state:

State	Description
UNAPPLIED	The license was not applied.
APPLIED	A valid license was applied.
INVALID	A valid license was applied, but a valid RAID card cannot be detected.

Managing Remote NVMe-oF Targets

You must connect to the NVMe-oF target before you create physical drives from NVMe-oF devices.

Connecting to a Remote NVMe-oF Target

To connect to a remote NVMe-oF target, run:

```
$ sudo graidctl connect remote_target <transport type> <addr> <address family> <port service id>
```

Required parameters:

Option	Value	Description
transport type		This field specifies the network fabric being used for a NVMe-over-Fabrics network. Current string values include:
	RDMA	The network fabric is an RDMA network (RoCE, iWARP, Infiniband, basic rdma, etc)
	TCP	The network fabric is a TCP/IP network.
	FC	The network fabric is a Fibre Channel network.
ip address		This field specifies the network address of the controller.
address family		This field specifies the network address protocol. Current string values include: ipv4/ipv6
port service		This field specifies the transport service ID.

Output example:

```
[graid@graid-demo ~]$ sudo graidctl connect remote_target rdma 192.168.2.10 ipv4 4420
✔Connect remote target successfully.
✔Connect remote target Target 0 successfully.
```

Listing Connected Remote NVMe-oF Targets

To list all of the connected remote NVMe-oF targets, run:

```
$ sudo graidctl list remote_target
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list remote_target
✔List remote target successfully.
|----|------|---------------|-----------------|-------------|
| ID | TYPE | ADDRESS       | ADDRESS FAMILY | SERVICE ID |
|----|------|---------------|-----------------|-------------|
|  0 | rdma | 192.168.2.10 | ipv4           | 4420       |
|----|------|---------------|-----------------|-------------|
```

> **Note:**
>
> You can control multiple NVMe-of namespaces using GRAID SupremeRAID™ v1.1.0.

Disconnecting from Remote NVMe-oF Targets

To disconnect from an NVMe-oF target, run:

```
$ sudo graidctl disconnect remote_target <target id>
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl disconnect remote_target 0
✔Disconnect remote target successfully.
✔Disconnect remote target  Port 0 successfully.
```

> **Note:**
>
> You cannot delete the target when there are physical drives created from the target.

Exporting NVMe-oF Target Management

You can export the virtual drive to other initiators.

Creating the NVMe-oF Target Port Service

To create the NVme-oF target port service, run:

```
$ sudo graidctl create nvmeof_target <tcp|rdma> <interface> <address family> <srvcid> [flags]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl create nvmeof_target tcp ens160 ipv4 4420
✔Create nvmeof target successfully.
✔Create nvmeof target Port 0 successfully.
```

Exporting NVMe-oF Targets

To export NVMe-oF target devices using the service port that you created, run:

```
$ sudo graidctl export virtual_drive <DG_ID> <VD_ID> [flags]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl export virtual drive 0 0-1 --all
✔Export virtual drive successfully.
✔Export virtual drive VD0 into Port 0 successfully.
✔Export virtual drive successfully.
✔Export virtual drive VD1 into Port 0 successfully.
[graid@graid-demo ~]$ sudo graidctl export virtual drive 0 3 --port-ids=1
✔Export virtual drive successfully.
✔Export virtual drive VD3 into Port 1 successfully.
```

Listing Created NVMe-oF Targets

To list all created NVMe-oF target devices, run:

```
$ sudo graidctl list nvmeof_target
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list nvmeof_target
✔List nvmeof target successfully.
|---------|------|-----------|-------------|----------------|------------|-----------------------------|
| PORT ID | TYPE | INTERFACE | ADDRESS     | ADDRESS FAMILY | SERVICE ID | SUBSYSTEMS                  |
|---------|------|-----------|-------------|----------------|------------|-----------------------------|
|       0 | tcp  | ens160    | 172.16.11.81 | ipv4           | 4420       | DG0/VD0, DG0/VD1            |
|---------|------|-----------|-------------|----------------|------------|-----------------------------|
|       1 | tcp  | ens160    | 172.16.11.81 | ipv4           | 4421       | DG0/VD0, DG0/VD1, DG0/VD3   |
|---------|------|-----------|-------------|----------------|------------|-----------------------------|
```

Deleting the NVMe-oF Target Port Service

To delete the NVMe-oF target port service, run:

```
$ sudo graidctl delete nvmeof_target <PORT_ID> [flags]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl delete nvmeof_target 0
✔Delete nvmeof target successfully.
✔Delete nvmeof target Port 0 successfully.
```

Unexporting NVMe-oF Targets

To unexport an NVMe-of target, run:

```
$ sudo graidctl unexport virtual_drive <DG_ID> <VD_ID> [flags]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl unexport virtual_drive 0 3 -p 1
✔Unexport virtual drive successfully.
✔Unexport virtual drive VD3 from Port 1 successfully.
[graid@graid-demo ~]$ sudo graidctl unexport virtual_drive 0 0-1 --all
✔Unexport virtual drive successfully.
✔Unexport virtual drive VD0 from Port 0 successfully.
✔Unexport virtual drive VD0 from Port 1 successfully.
✔Unexport virtual drive successfully.
✔Unexport virtual drive VD1 from Port 0 successfully.
✔Unexport virtual drive VD1 from Port 1 successfully.
```

Viewing Host Drive Information

Listing the NVMe Drives

To list all the directly attached NVMe drives, or NVMe-oF target drives, that can be used to create physical drives, run:

```
$ sudo graidctl list nvme_drive
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list nvme_drive
✔List nvme drive successfully.
|-------------|--------------|-----------------------------------------------|------|----------|--------------|
| DEVICE PATH | MODEL        | NQN/WWID                                      | NSID | CAPACITY | ADDRESS      |
|-------------|--------------|-----------------------------------------------|------|----------|--------------|
| /dev/nvme0  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A064T1L8 |    1 | 3.2 TB   | 0000:1a:00.0 |
| /dev/nvme1  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8 |    1 | 3.2 TB   | 0000:1b:00.0 |
| /dev/nvme2  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A05KT1L8 |    1 | 3.2 TB   | 0000:1c:00.0 |
| /dev/nvme3  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:X0N0A015T1L8 |    1 | 3.2 TB   | 0000:1d:00.0 |
| /dev/nvme4  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A06QT1L8 |    1 | 3.2 TB   | 0000:11:00.0 |
| /dev/nvme5  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0G0A001T1L8 |    1 | 3.2 TB   | 0000:12:00.0 |
| /dev/nvme6  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A006T1L8 |    1 | 3.2 TB   | 0000:13:00.0 |
| /dev/nvme7  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04WT1L8 |    1 | 3.2 TB   | 0000:20:00.0 |
| /dev/nvme8  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A003T1L8 |    1 | 3.2 TB   | 0000:21:00.0 |
| /dev/nvme9  | KCM61VUL3T20 | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8 |    1 | 3.2 TB   | 0000:22:00.0 |
|-------------|--------------|-----------------------------------------------|------|----------|--------------|
```

Output content:

Field	Description
DEVICE PATH	This field displays the block device path of the drive.
NQN	This field displays the NVMe Qualified Name of the drive.
MODEL	This field displays the model number of the drive.
CAPACITY	This field displays the capacity of the drive.

Listing SAS/SATA Drives

To list all SAS/SATA drives that can be used as physical drives, run:

```
$ sudo graidctl list scsi_drive
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list scsi_drive
✔List scsi drive successfully.
|-------------|-----------------------------------------------|-----------------|----------|
| DEVICE PATH | WWID                                          | MODEL           | CAPACITY |
|-------------|-----------------------------------------------|-----------------|----------|
| /dev/sda    | t10.ATA INTEL SSDSC2KB240G7 BTYS83010GKS240AGN | INTEL SSDSC2KB24 | 240 GB   |
| /dev/sdb    | t10.ATA INTEL SSDSC2KB240G8 BTYF052107VH240AGN | INTEL SSDSC2KB24 | 240 GB   |
|-------------|-----------------------------------------------|-----------------|----------|
```

Output content:

Field	Description
DEVICE PATH	This field displays the block device path for the drive.
WWID	This field displays the Worldwide Identification of the drive.
MODEL	This field displays the model number of the drive.
CAPACITY	This field displays the capacity of the drive.

Managing Physical Drives

Creating a Physical Drive

To create a physical drive, run:

```
$ sudo graidctl create physical_drive <DEVICE_PATH|NQN|WWID>
```

Output example for simultaneously creating multiple physical drives with the device path and NQN:

```
[graid@graid-demo ~]$ sudo graidctl create physical_drive /dev/nvme0-3
✔Create physical drive successfully.
✔Create physical drive PD0 (/dev/nvme0: nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A064T1L8) successfully.
✔Create physical drive PD1 (/dev/nvme1: nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8) successfully.
✔Create physical drive PD2 (/dev/nvme2: nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A05KT1L8) successfully.
✔Create physical drive PD3 (/dev/nvme3: nqn.2019-10.com.kioxia:KCM61VUL3T20:X0N0A015T1L8) successfully.
[graid@graid-demo ~]$ sudo graidctl create physical_drive nqn.2019-10.com.kioxia:KCM61VUL3T20:X0X0A01ET1L8 \
> nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8
✔Create physical drive PD8 (nqn.2019-10.com.kioxia:KCM61VUL3T20:X0X0A01ET1L8) successfully.
✔Create physical drive PD9 (nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8) successfully.
```

> **Note:**
>
> Ensure that the system or other applications are not on the physical drive before creating or replacing it.

Listing the Physical Drives

To list all of the physical drives, run:

```
$ sudo graidctl list physical_drive
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list physical_drive
✔List physical drive successfully.
```

PD ID	DG ID	DEVICE PATH	NQN/WWID	MODEL	CAPACITY	SLOT ID	STATE
0	N/A	/dev/gpd0	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A038T1L8	KCM61VUL3T20	3.2 TB	0	UNCONFIGURED_GOOD
1	N/A	/dev/gpd1	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A06QT1L8	KCM61VUL3T20	3.2 TB	1	UNCONFIGURED_GOOD
2	N/A	/dev/gpd2	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04WT1L8	KCM61VUL3T20	3.2 TB	2	UNCONFIGURED_GOOD
3	N/A	/dev/gpd3	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8	KCM61VUL3T20	3.2 TB	3	UNCONFIGURED_GOOD
4	N/A	/dev/gpd4	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A003T1L8	KCM61VUL3T20	3.2 TB	4	UNCONFIGURED_GOOD
5	N/A	/dev/gpd5	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A005T1L8	KCM61VUL3T20	3.2 TB	5	UNCONFIGURED_GOOD
6	N/A	/dev/gpd6	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0F0A031T1L8	KCM61VUL3T20	3.2 TB	6	UNCONFIGURED_GOOD
7	N/A	/dev/gpd7	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A002T1L8	KCM61VUL3T20	3.2 TB	7	UNCONFIGURED_GOOD
32	4	/dev/nvme0n1	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8	KCM61VUL3T20	3.2 TB	N/A	ONLINE
33	4	/dev/nvme1n1	nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A001T1L8	KCM61VUL3T20	3.2 TB	N/A	ONLINE

Output content:

Field	Description
SLOT ID	This field displays the slot ID of the corresponding NVMe/SAS/SATA drive. Note that the PD ID is not related to the SLOT ID, and that you must set the physical drives using the PD ID.
DG ID	This field displays the drive group ID of the physical drive.
PD ID	This field displays the PD ID. The PD ID is a unique ID provided by the SupremeRAID driver when the physical drive is created. It is not related to any SSD information such as slot ID or NQN. The PD ID is used for all further operations.
NQN/WWID	This field displays the NQN or WWID of corresponding NVMe/SAS/SATA drive.
MODEL	This field displays the model number of the corresponding NVMe/SAS/SATA drive.
CAPACITY	This field displays the capacity of corresponding NVMe/SAS/SATA drive.
STATE	This field displays the physical drive state.

Physical drive STATE:

STATE	Description
ONLINE	The physical drive was added to a drive group and is ready to work.
HOTSPARE	The physical drive is configured as hot spare drive.
FAILED	The physical drive is detected, but it is not functioning normally.
OFFLINE	The physical drive is marked as offline.
REBUILD	The physical drive is being rebuilt.
MISSING	The physical drive cannot be detected.
INCONSISTENT	The data in the physical drive is inconsistent. This condition generally occurs when the physical drive is in the REBUILD state and the system encounters an abnormal crash.
UNCONFIGURED_GOOD	The physical drive did not join a drive group.
UNCONFIGURED_BAD	The physical drive did not join a drive group and it is not functioning normally.

Deleting a Physical Drive

To delete a physical drive, run:

```
$ sudo graidctl delete physical_drive <PD_ID>
```

Output example for deleting multiple physical drives simultaneously:

```
[graid@graid-demo ~]$ sudo graidctl delete physical_drive 2
✖Delete physical drive failed: Failed to delete some PDs.
✖Delete physical drive PD2 failed: rpc error: code = NotFound desc = PD2 is still using by DG0
[graid@graid-demo ~]$ sudo graidctl delete physical_drive 0 1 5-7
✔Delete physical drive successfully.
✔Delete physical drive PD0 successfully.
✔Delete physical drive PD1 successfully.
✔Delete physical drive PD5 successfully.
✔Delete physical drive PD6 successfully.
✔Delete physical drive PD7 successfully.
```

The output shows that a physical drive cannot be deleted when it is part of a drive group.

Describing a Physical Drive

To view detailed information for a physical drive, run:

```
$ sudo graidctl describe physical_drive <PD_ID>
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl describe physical_drive 5
✔Describe physical drive successfully.
PD ID:              5
DG ID:              -1
Slot ID:            15
GUID:               nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A038T1L8
Mode:               KCM61VUL3T20
Capacity:           3.2 TB
State:              HOTSPARE
Device Path:        /dev/nvme9n1
Attributes:
                    hotspare = 0, 1
                    locating = false
```

Locating a Physical Drive

To locate a physical drive, run:

```
$ sudo graidctl edit physical_drive <PD_ID> locating start
```

To stop locating a physical drive, run:

```
$ sudo graidctl edit physical_drive <PD_ID> locating stop
```

Marking a Physical Drive Online or Offline

To mark a physical drive as online or offline, run:

```
$ sudo graidctl edit physical_drive <PD_ID> marker <offline|online>
```

> **Note:**
> Marking a physical drive as offline, even briefly, puts the physical drive in the REBUILD state.

Assigning a Hot Spare Drive

To assign a physical drive as global hot spare, run:

```
$ sudo graidctl edit physical_drive <PD_ID> hotspare global
```

To assign a physical drive as the hot spare for a specific drive group, run:

```
$ sudo graidctl edit physical_drive <PD_ID> hotspare <DG_ID>
```

To assign a physical drive as a hot spare for multiple drive groups, use a comma (,) to separate the drive group IDs.

Replacing a Nearly Worn-Out or Broken SSD

To replace a nearly worn-out or broken SSD:

1. Mark the physical drive as bad using the following command. (You can skip this step if the physical drive is in the **MISSING** or other abnormal state.)

   ```
   $ sudo graidctl edit pd <OLD_PD_ID> marker bad
   ```

2. Replace the NVMe SSD. The state of the prior physical drive will indicate **FAILED**.

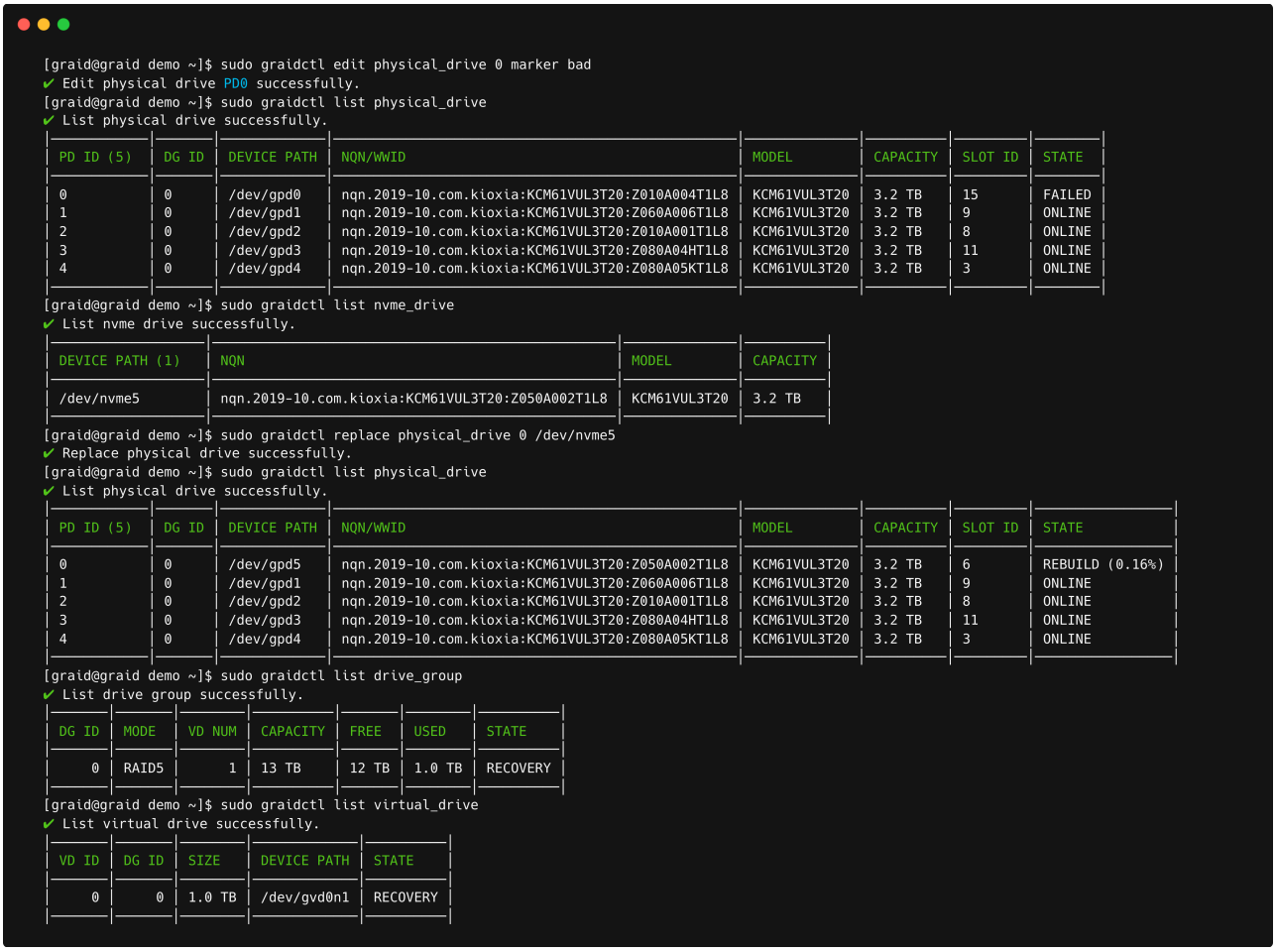
3. Check the NQN of the new SSD.

   ```
   $ sudo graidctl list nvme_drive
   ```

4. Replace the physical drive.

   ```
   $ sudo graidctl replace physical_drive <OLD_PD_ID> <DEVICE_PATH|NQN|WWID>
   ```

Output example:



```
[graid@graid demo ~]$ sudo graidctl edit physical_drive 0 marker bad
✔ Edit physical drive PD0 successfully.
[graid@graid demo ~]$ sudo graidctl list physical_drive
✔ List physical drive successfully.
|————————|————————|————————|————————————————————————————————————|————————————|————————|————————|————————|
| PD ID (5)  | DG ID  | DEVICE PATH | NQN/WWID                                           | MODEL        | CAPACITY | SLOT ID | STATE  |
|————————|————————|————————|————————————————————————————————————|————————————|————————|————————|————————|
| 0          | 0      | /dev/gpd0   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A004T1L8   | KCM61VUL3T20 | 3.2 TB   | 15      | FAILED |
| 1          | 0      | /dev/gpd1   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A006T1L8   | KCM61VUL3T20 | 3.2 TB   | 9       | ONLINE |
| 2          | 0      | /dev/gpd2   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A001T1L8   | KCM61VUL3T20 | 3.2 TB   | 8       | ONLINE |
| 3          | 0      | /dev/gpd3   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8   | KCM61VUL3T20 | 3.2 TB   | 11      | ONLINE |
| 4          | 0      | /dev/gpd4   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A05KT1L8   | KCM61VUL3T20 | 3.2 TB   | 3       | ONLINE |
|————————|————————|————————|————————————————————————————————————|————————————|————————|————————|————————|
[graid@graid demo ~]$ sudo graidctl list nvme_drive
✔ List nvme drive successfully.
|————————————|————————————————————————————————————|————————————|————————|
| DEVICE PATH (1)  | NQN                                          | MODEL        | CAPACITY |
|————————————|————————————————————————————————————|————————————|————————|
| /dev/nvme5       | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8 | KCM61VUL3T20 | 3.2 TB   |
|————————————|————————————————————————————————————|————————————|————————|
[graid@graid demo ~]$ sudo graidctl replace physical_drive 0 /dev/nvme5
✔ Replace physical drive successfully.
[graid@graid demo ~]$ sudo graidctl list physical_drive
✔ List physical drive successfully.
|————————|————————|————————|————————————————————————————————————|————————————|————————|————————|————————————————|
| PD ID (5)  | DG ID  | DEVICE PATH | NQN/WWID                                           | MODEL        | CAPACITY | SLOT ID | STATE          |
|————————|————————|————————|————————————————————————————————————|————————————|————————|————————|————————————————|
| 0          | 0      | /dev/gpd5   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z050A002T1L8   | KCM61VUL3T20 | 3.2 TB   | 6       | REBUILD (0.16%) |
| 1          | 0      | /dev/gpd1   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z060A006T1L8   | KCM61VUL3T20 | 3.2 TB   | 9       | ONLINE         |
| 2          | 0      | /dev/gpd2   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A001T1L8   | KCM61VUL3T20 | 3.2 TB   | 8       | ONLINE         |
| 3          | 0      | /dev/gpd3   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8   | KCM61VUL3T20 | 3.2 TB   | 11      | ONLINE         |
| 4          | 0      | /dev/gpd4   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A05KT1L8   | KCM61VUL3T20 | 3.2 TB   | 3       | ONLINE         |
|————————|————————|————————|————————————————————————————————————|————————————|————————|————————|————————————————|
[graid@graid demo ~]$ sudo graidctl list drive_group
✔ List drive group successfully.
|————————|————————|————————|————————|————————|————————|
| DG ID  | MODE   | VD NUM | CAPACITY | FREE   | USED   | STATE    |
|————————|————————|————————|————————|————————|————————|
|      0 | RAID5  |      1 | 13 TB    | 12 TB  | 1.0 TB | RECOVERY |
|————————|————————|————————|————————|————————|————————|
[graid@graid demo ~]$ sudo graidctl list virtual_drive
✔ List virtual drive successfully.
|————————|————————|————————|————————|————————|
| VD ID  | DG ID  | SIZE   | DEVICE PATH | STATE    |
|————————|————————|————————|————————|————————|
|      0 |      0 | 1.0 TB | /dev/gvd0n1 | RECOVERY |
|————————|————————|————————|————————|————————|
```

> **Note:**
>
> Ensure that the system or other applications are not on the physical drive before creating or replacing it.

Creating the Physical Drive from the NVMe-oF Drive

To create the physical drive from the NVMe-oF drive:

1. Connect to the remote NVMe-oF target.

```
$ sudo graidctl connect remote_target <tcp|rdma|fc> <addr> <address family> <service id>
```

2. Check the NVMe drives from the remote NVMe-oF target.

```
$ sudo graidctl list nvme_drive
```

3. Create the physical drives.

```
$ sudo graidctl create physical_drive <nqn or devpath>...
```

4. Create a RAID5 drive group with 4 physical drives.

```
$ sudo graidctl create drive_group <Mode> <PD_ID>... [flags]
```

Output example:

```
[graid@graid demo~]$ $ sudo graidctl connect remote_target tcp 172.16.11.81 ipv4 4420
✔Connect remote target successfully.
✔Connect remote target Target 0 successfully.
[graid@graid demo~]$ sudo graidctl list nvme_drive
✔List nvme drive successfully.
|─────────────────|───────|──────────────────────────────────────────|──────|──────────|──────────────────────────────────|
| DEVICE PATH (4) | MODEL | NQN/WWID                                 | NSID | CAPACITY | ADDRESS                          |
|─────────────────|───────|──────────────────────────────────────────|──────|──────────|──────────────────────────────────|
| /dev/nvme0n1    | Linux | uuid.b951d877-76af-4dfe-84ee-a45164554fe2 |    1 | 22 GB    | traddr=172.16.11.81,trsvcid=4420 |
| /dev/nvme1n1    | Linux | uuid.6f21ec8f-00ee-4a30-a9b8-413447b8f138 |    1 | 22 GB    | traddr=172.16.11.81,trsvcid=4420 |
| /dev/nvme2n1    | Linux | uuid.34d1d6aa-41fc-4c02-a660-f75429d7d74b |    1 | 22 GB    | traddr=172.16.11.81,trsvcid=4420 |
| /dev/nvme3n1    | Linux | uuid.d846f451-31af-49ae-b3db-8ca90f454c3b |    1 | 22 GB    | traddr=172.16.11.81,trsvcid=4420 |
|─────────────────|───────|──────────────────────────────────────────|──────|──────────|──────────────────────────────────|
[graid@graid demo~]$ sudo graidctl create physical_drive uuid.b951d877-76af-4dfe-84ee-a45164554fe2 /dev/nvme1 /dev/nvme3 uuid.34d1d6aa-41fc-4c02-a660-
f75429d7d74b
✔Create physical drive successfully.
✔Create physical drive PD0 (uuid.b951d877-76af-4dfe-84ee-a45164554fe2) successfully.
✔Create physical drive PD1 (/dev/nvme1: uuid.6f21ec8f-00ee-4a30-a9b8-413447b8f138) successfully.
✔Create physical drive PD2 (/dev/nvme3: uuid.d846f451-31af-49ae-b3db-8ca90f454c3b) successfully.
✔Create physical drive PD3 (uuid.34d1d6aa-41fc-4c02-a660-f75429d7d74b) successfully.
[graid@graid demo~]$ sudo graidctl create drive_group raid5 0-3
✔ Create drive group DG0 successfully.
```

Managing Drive Groups

Creating Drive Groups

To create a drive group or groups, run:

```
$ sudo graidctl create drive_group <RAID_MODE> (PD_IDs) [--background-init]
```

```
[graid@graid-demo ~]$ sudo graidctl create drive_group raid1 0-1
✔Create drive group DG0 successfully.
[graid@graid-demo ~]$ sudo graidctl create drive_group raid5 2-4
✔Create drive group DG1 successfully.
[graid@graid-demo ~]$ sudo graidctl create drive_group raid6 5-9
✔Create drive group DG2 successfully.
```

Required parameters:

Option	Description
RAID_MODE	This field specifies the RAID mode of the drive group. Entries must be all uppercase or all lowercase. (For example, RAID6 or raid6 are both correct)
PD_IDs	This field specifies the ID of the physical drive joining the drive group.

Optional parameters:

Option	Description	Behavior
--background-init, -b	Default option. Use standard methods to initialize the drive group. When all the physical drives in the drive group support the deallocate dataset management command, it is used to synchronize the data, or parity, between the physical drives during the creation of the drive group.	An I/O capable device path similar to /dev/gvd0n1 is created.
--foreground-init, -z	Initialzing foreground.	The virtual drive appears in the system after initialization is complete. Use: $ sudo graidctl list drive_group to check the initialization progress.

Option	Description	Behavior
--force, -f	Force to initialize. Assumes that the drive is free.	The drive group STATE immediately becomes OPTIMAL indicating that the drive group is available for use.

> **Important Note:**
> Wait for the drive group initialization to complete. DO NOT power-off or reboot the system when the drive_group state is INIT/RESYNC/RECOVERY.

Use the command below to check drive_group state:

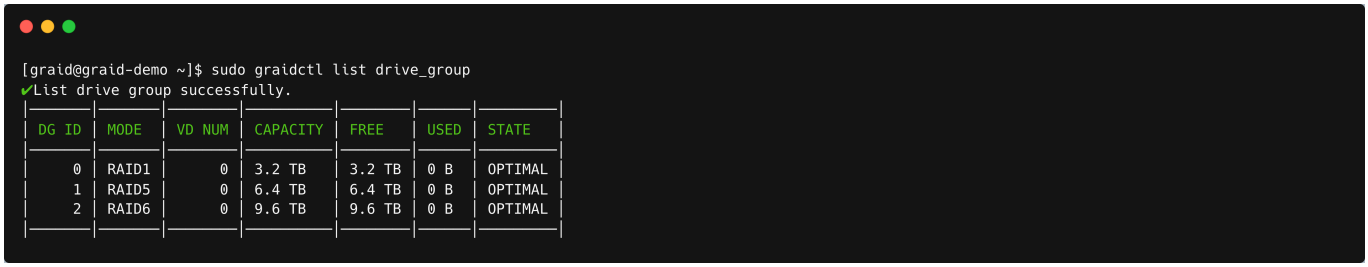
```
$ sudo graidctl list drive_group
```

Listing Drive Groups

To list all drive groups, run:

```
$ sudo graidctl list drive_group
```

Output example:



Output content:

Field	Description
DG ID	This field displays the drive group ID.
MODE	This field displays the drive group RAID mode.
VD NUM	This field displays the number of virtual drives in the drive group.
CAPACITY	This field displays the total usable capacity of the drive group.
FREE	This field displays the unused space of the drive group.
USED	This field displays the used space of the drive group.
STATE	This field displays the drive group state.

STATE	Description
OFFLINE	The drive group does not function normally. This condition is usually caused when the number of damaged physical drives exceeds the limit.
OPTIMAL	The drive group is in optimal state.
OPTIMAL(!)	The drive group is in optimal state but found inconsistency data.
OPTIMAL(cc)	The drive group is in optimal state and the consistency check task is ongoing.
OPTIMAL(cc!)	The drive group is in optimal state and the consistency check task is ongoing but found inconsistent data.
DEGRADED	The drive group is available and ready, but the number of missing or failed physical drives has reached the limit.
PARTIALLY_DEGRADED	The drive group is available and ready for use, but some physical drives are missing or failed.
RECOVERY	The drive group is recovering.
FAILED	The drive group does not function normally.
INIT	The drive group is initializing.
RESYNC	The drive group is re-synchronizing. This condition usually occurs when the system encounters an abnormal crash. Do not replace the physical drive in this state until the re-synchronization process is complete.
RESCUE	The drive group is in rescue mode.
INIT	The drive group is initializing.
RESYNC	The drive group is re-synchronizing. This condition usually occurs when the system encounters an abnormal crash. Do not replace the physical drive in this state until the re-synchronization process is complete.
RESCUE	The drive group is in rescue mode.

Deleting Drive Groups

To delete drive groups, run:

```
$ sudo graidctl delete drive_group <DG_ID>
```

```
[graid@graid-demo ~]$ sudo graidctl delete drive_group 1
✖Delete drive group failed: Failed to delete some DGs.
✖Delete drive group DG1 failed: rpc error: code = FailedPrecondition desc = DG1 still has 1VD(s)
[graid@graid-demo ~]$ sudo graidctl delete drive_group 0 2
✔Delete drive group DG0 successfully.
✔Delete drive group DG2 successfully.
```

You cannot delete a drive group that contains a virtual drive.

In this example, drive group 1 was not deleted because it contains a virtual drive. Drive groups 0 and 2 were deleted successfully.

Describing a Drive Group

To display detailed information for a drive group, run:

```
$ sudo graidctl describe drive_group <DG_ID>
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl describe drive_group 0
✔Describe drive group successfully.
DG ID:              0
Mode:               RAID5
Capacity:           11 GB
Free Space:         1.4 GB
Used Space:         9.2 GB
State:              OPTIMAL
PD IDs:             [0 1 2]
Number of VDs:      3
Attributes:
                    rebuild_speed = high
```

Setting the Drive Group Rebuild Speed

To set the rebuild speed for a drive group, run:

```
$ sudo graidctl edit drive_group <DG_ID> rebuild_speed {low|normal|high}
```

Locating the Physical Drives in the Drive Group

To locate all the physical drives in the drive group, run:

```
$ sudo graidctl edit drive_group <DG_ID> locating start
```

To stop locating all the physical drives in drive group, run:

```
$ sudo graidctl edit drive_group <DG_ID> locating stop
```

Degradation and Recovery

- When multiple drive groups require simultaneous recovery, the drive groups recover individually.
- When multiple physical drives in the same drive group require rebuilding, the physical drives are rebuilt simultaneously.

Rescue Mode

When a damaged drive group is initialized, or when a recovering drive group encounters an abnormal system crash, the data integrity of the drive group is affected. In this event, the drive group is forced offline to prevent data from being written to the drive group. To read the data for the drive group, force the drive group to go online using Rescue mode.

> **Note:**
>
> A drive group in Rescue mode is read-only. Rescue mode cannot be disabled.

To enter the rescue mode, run:

```
$ sudo graidctl edit drive_group <DG_ID> rescue_mode on
```

Managing Virtual Drives

Creating a Virtual Drive

To create a virtual drive, run:

```
$ sudo graidctl create virtual_drive <DG_ID> [<VD_SIZE>]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl create virtual_drive 0
✔Create virtual drive VD0 in DG0 successfully.
[graid@graid-demo ~]$ sudo graidctl create virtual_drive 1 100G
✔Create virtual drive VD0 in DG1 successfully.
[graid@graid-demo ~]$ sudo graidctl create virtual_drive 2 1T
✔Create virtual drive VD0 in DG2 successfully.
```

Listing Virtual Drives

To list virtual drives, run:

```
$ sudo graidctl list virtual_drive [--dg-id=<DG_ID>] [--vd-id=<VD_ID>]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list virtual_drive
✔List virtual drive successfully.
|-------------|-------|--------|--------------------|---------|----------|
| VD ID (12)  | DG ID | SIZE   | DEVICE PATH        | STATE   | EXPORTED |
|-------------|-------|--------|--------------------|---------|----------|
|           0 |     0 | 4.3 GB | = Stripe Cache =   | OPTIMAL | No       |
|           1 |     0 | 4.3 GB | /dev/mapper/dg0vd1 | OPTIMAL | No       |
|           2 |     0 | 105 GB | /dev/gvd2n1        | OPTIMAL | Yes      |
|           3 |     0 | 105 GB | /dev/gvd3n1        | OPTIMAL | Yes      |
|           0 |     4 | 14 GB  | /dev/md125         | OPTIMAL | No       |
|           1 |     4 | 1.1 GB | /dev/md126         | OPTIMAL | No       |
|           2 |     4 | 629 MB | /dev/md124         | OPTIMAL | No       |
|           3 |     4 | 1.7 GB | /dev/md127         | OPTIMAL | No       |
|-------------|-------|--------|--------------------|---------|----------|
```

Output content:

Field	Description
DG ID	This field displays the drive group ID.
VD ID	This field displays the virtual drive ID.
SIZE	This field displays the usable size of the virtual drive.
DEVICE PATH	This field displays the device path of the virtual drive.

Field	Description
NQN	This field displays the NQN of the virtual drive.
STATE	This field displays the virtual drive state. It is identical to the drive group state.
EXPORTED	This field displays whether the virtual drive was exported using NVMe-oF or iSCSI.

> **Note:**
> Do not perform I/O before the virtual drive is initialized and the device path (for example, /dev/gvd0n) is created.

Virtual Drive STATE:

Identical to the drive group state.

STATE	Description
OFFLINE	The drive group does not function normally. This condition is usually caused when the number of damaged physical drives exceeds the limit.
OPTIMAL	The drive group is in the optimal state.
DEGRADED	The drive group is available and ready, but the number of missing or failed physical drives has reached the limit.
PARTIALLY_DEGRADED	The drive group is available and ready for use, but some physical drives are missing or failed.
RECOVERY	The drive group is recovering.
FAILED	The drive group does not function normally.
INIT	The drive group is initializing.
RESYNC	The drive group is re-synchronizing. This condition usually occurs when the system encounters an abnormal crash. Do not replace the physical drive in this state until the re-synchronization process is complete.
RESCUE	The drive group is in rescue mode.

Deleting Virtual Drives

To delete virtual drives, run:

```
$ sudo graidctl delete virtual_drive <DG_ID> <VD_ID> [--force]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl delete virtual_drive 0 0
✔Delete virtual drive VD0 from DG0 successfully.
[graid@graid-demo ~]$ sudo graidctl delete virtual_drive 2 0-1
✔Delete virtual drive VD1 from DG2 successfully.
✔Delete virtual drive VD0 from DG2 successfully.
```

The example shows that a virtual drive being used by the application cannot be deleted without adding the force flag.

Describing a Virtual Group

To check the detailed information for a virtual group, run:

```
$ sudo graidctl describe virtual_drive <DG_ID> <VD_ID>
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl describe virtual_drive 0 4
✔Describe virtual drive successfully.
DG ID:              0
VD ID:              0
Model:              SER0001
NQN:                nqn.2020-05.com.graidtech:SER00013DBDD6BD73EF89C9
DevicePath:         /dev/gvd4n1
Size:               4.3 GB
State:              OPTIMAL
Description:
Exported:
|------|----------------|-------------|------------|----------------|------------|
| PORT | TRANSPORT TYPE | ADDRESS     | INTERFACE  | ADDRESS FAMILY | SERVICE ID |
|------|----------------|-------------|------------|----------------|------------|
|    0 | tcp            | 172.16.11.64 | ens192    | ipv4           | 4420       |
|------|----------------|-------------|------------|----------------|------------|
```

About Consistency Check task

Consistency Check task

The consistency check operation verifies the data is correct in DGs that use RAID levels 1, 5, 6, and 10. For example, in a system with parity, checking consistency means calculating the data on one drive and comparing the results to the contents of the parity drive. The consistency check cannot be performed on RAID 0 because it does not provide data redundancy. Besides, a consistency check can only run when the DG is in OPTIMAL or PARTIALLY_DEGRADED state.

The record of the Consistency Check

The consistency check function will record all events to the event database, and graidctl provides commands to retrieve the events. The maximum number of event entries is limited to 1000. The event entries will be periodically deleted by the system or manually deleted by the user's command.

Enable Consistency Check task by manual

To start consistency check manual, run:

```
$ sudo graidctl start consistency_check manual_task [DG_IDs][--policy|-p <stop_on_error|auto_fix>]
```

Enable consistency check task would add annotations beside the output string of DG state as follows:

DG state	Description
OPTIMAL	Normal state without enable consistency check
OPTIMAL (!)	Inconsistency found
OPTIMAL (cc)	Consistency check ongoing
OPTIMAL (cc!)	Consistency check ongoing and Inconsistency found

Output example:

```
[graid@graid-demo ~]$ sudo graidctl start consistency_check manual_task 0 1  -p stop_on_error
✔Start consistency check successfully.
```

Stop Consistency Check task

To stop the consistency check task, run:

```
$ sudo graidctl stop consistency_check current_task
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl stop consistency_check current_task
✔Stop consistency check successfully.
```

Schedule Consistency Check task

To schedule the consistency check task, run:

```
$ sudo graidctl set consistency_check schedule_mode <off|ontinuously|hourly|daily|weekly|monthly>
<yyyy/mm/dd> <hh>
```

Enable consistency check task would add annotations beside the output string of DG state as follows:

DG state	Description
OPTIMAL	Normal state without enable consistency check
OPTIMAL (!)	Inconsistency found
OPTIMAL (cc)	Consistency check ongoing
OPTIMAL (cc!)	Consistency check ongoing and Inconsistency found

Output example:

```
[graid@graid-demo ~]$ sudo graidctl set consistency_check schedule_mode daily 2022/06/25 10
✔Set consistency check successfully.
```

Describing the Consistency Check task

To check the detailed information for the consistency check task, run:

```
$ sudo graidctl describe consistency_check
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl describe consistency_check
✔Describe consistency check successfully.
Schedule Mode:      daily
Schedule Base:      2022-06-25 10:00:00 +0800 CST
Excluded DGs:       []
Policy:             stop_on_error
Next Schedule:      2022-06-26 10:00:00 +0800 CST
Current Task:       2 DG(s)
                    -DG0: Checking (progress: 28.15%)
                        Start Time: 2022-06-26 09:37:37 +0800 CST
                        End Time:
                    -DG1: Pending
                        Start Time:
                        End Time:
```

Set the Consistency Check policy

To set consistency check policy, run:

```
$ sudo graidctl set consistency_check policy <auto_fix|stop_on_error>
```

> **Note:** The consistency check would runs on all drive_group by defalt, if user wants to exclude some drive group user needs to run “excluded_dgs" command.

Output example:

```
[graid@graid-demo ~]$ sudo graidctl set consistency_check policy auto_fix
✔Set consistency check successfully.
```

Consistency Check exclude certain drive groups.

To exclude some drive groups, run:

```
$ sudo graidctl set consistency_check excluded_dgs <DG_IDs>
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl set consistency_check excluded_dgs 1
✔Set consistency check successfully.
```

List events

To check the detailed information from record, run:

```
$ sudo graidctl list event [-n <max_entries>] [-s <severity>] [-c <component>]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list event -n 10 -s INFO -c DG
✔List event successfully.
[2022-06-22 22:06:29 +0800 CST][INFO][DG][0] State transitted from UNKNOWN to OFFLINE.
[2022-06-22 22:20:07 +0800 CST][INFO][DG][0] Drive group deleted.
[2022-06-22 22:21:13 +0800 CST][INFO][DG][0] State transitted from UNKNOWN to OPTIMAL.
[2022-06-22 22:21:13 +0800 CST][INFO][DG][0] Drive group created.
[2022-06-22 22:28:02 +0800 CST][INFO][DG][0] Drive group deleted.
[2022-06-22 22:28:20 +0800 CST][INFO][DG][0] State transitted from UNKNOWN to OPTIMAL.
[2022-06-22 22:28:20 +0800 CST][INFO][DG][0] Drive group created.
[2022-06-22 22:30:15 +0800 CST][INFO][DG][0] CC has started.
[2022-06-22 23:26:57 +0800 CST][INFO][DG][0] CC has completed.
[2022-06-22 23:26:57 +0800 CST][INFO][DG][0] CC has started.
```

Delete Events

To delete old events, run:

```
$ sudo graidctl delete event [-d yyyy/mm/dd] [-e <num_entries>]
```

Output example:

```
[graid@graid-demo ~]$ sudo graidctl delete event -d 2022/06/25 -e 10
✔Delete event successfully.
```

Creating a RAID-5 Virtual Drive with 5 NVMe SSDs

To create a RAID-5 virtual drive with 5 NVMe SSDs:

1. Create a physical drive.

```
$ sudo graidctl create physical_drive /dev/nvme0-4
```

2. Create a drive group.

```
$ sudo graidctl create drive_group raid5 0-4
```

3. Create a virtual drive.

```
$ sudo graidctl create virtual_drive 0
```

4. Check the device path of the new virtual drive.

```
$ sudo graidctl list virtual_drive --dg-id=0
```

Output example:

```
[graid@graid demo~]$ sudo graidctl create physical_drive /dev/nvme0-4
✔ Create physical drive PD0 successfully.
✔ Create physical drive PD1 successfully.
✔ Create physical drive PD2 successfully.
✔ Create physical drive PD3 successfully.
✔ Create physical drive PD4 successfully.
[graid@graid demo~]$ sudo graidctl create drive_group raid5 0-4
✔ Create drive group DG0 successfully.
[graid@graid demo~]$ sudo graidctl create virtual_drive 0
✔ Create virtual drive VD0 in DG0 successfully.
[graid@graid demo~]$ sudo graidctl list virtual_drive --dg-id=0
✔ List virtual drive successfully.
|———————|———————|———————|———————————|———————|
| VD ID | DG ID | SIZE | DEVICE PATH | STATE |
|———————|———————|———————|———————————|———————|
| 0 | 0 | 1.0 TB | /dev/gvd0n1 | OPTIMAL |
|———————|———————|———————|———————————|———————|
```

Exporting the Virtual Drive as an NVMe-oF Target Drive using RDMA to the Initiator

To export the virtual drive as an NVMe-oF target drive using RDMA to the initiator:

1. Create the RDMA/TCP NVMe-of target port services.

```
$ sudo graidctl create nvmeof_target <tcp|rdma> <interface> <address family> <srvcid> [flags]
```

2. Export a virtual drive as an NVMe-of target.

```
$ sudo graidctl export virtual_drive <DG_ID> <VD_ID>... [flags]
```

3. List all NVMe-of targets.

```
$ sudo graidctl list nvmeof_target [flags]
```

4. Describe the detail information for an NVMe-of target.

```
$ sudo graidctl describe nvmeof_target <PORT_ID> [flags]
```

Output example:

```
[graid@graid demo~]$ sudo graidctl list virtual_drive
✔List virtual drive successfully.
|————————————|————————|————————|————————————|————————|————————————|
| VD ID (1)  | DG ID | SIZE     | DEVICE PATH | STATE    | EXPORTED |
|————————————|————————|————————|————————————|————————|————————————|
|          0 |      0 | 10.0 GB | /dev/gvd0n1 | OPTIMAL | No       |
|————————————|————————|————————|————————————|————————|————————————|
[graid@graid demo~]$ sudo graidctl create nvmeof_target tcp ens160 ipv4 4420
✔Create nvmeof target successfully.
✔Create nvmeof target Port 0 successfully.
[graid@graid demo~]$ sudo graidctl export virtual_drive 0 0 -p 0
✔Export virtual drive successfully.
✔Export virtual drive VD0 into Port 0 successfully.
[graid@graid demo~]$ sudo graidctl list nvmeof_target
✔List nvmeof target successfully.
|————————|——————|————————————|————————————|————————————|————————————|
| PORT ID | TYPE | INTERFACE | ADDRESS       | ADDRESS FAMILY | SERVICE ID | SUBSYSTEMS |
|————————|——————|————————————|————————————|————————————|————————————|
|       0 | tcp  | ens160    | 172.16.11.81 | ipv4           | 4420       | DG0/VD0    |
|————————|——————|————————————|————————————|————————————|————————————|
[graid@graid demo~]$ sudo graidctl describe nvmeof_target 0
✔Describe nvmeof target successfully.
Port:               0
TransportType:      tcp
Address:            172.16.11.81
Interface:          ens160
AddressFamily:      ipv4
ServiceId:          4420
Subsystems:
|————————————————————————————————————————————————————|————————|————————|————————|————————————|
| TARGET NAME                                          | DG ID | VD ID | ENABLE | DEVICE PATH |
|————————————————————————————————————————————————————|————————|————————|————————|————————————|
| nqn.2020-05.com.graidtech:SER0001763D84DEEB0A1F15 |      0 |      0 | Yes    | /dev/gvd0n1 |
|————————————————————————————————————————————————————|————————|————————|————————|————————————|
```

Setting Up a Stripe Cache to Improve the HDD's RAID 5 and RAID 6 Performance

To set up a stripe cache to improve an HDD's RAID 5 and RAID 6 performance:

1. Create a stripe cache with a 4GB virtual drive.

```
$ sudo graidctl create virtual_drive 0 4GB
```

> **Note:**
> A 4GB stripe cache is considered the best practice.
> Use this configuration whenever possible.

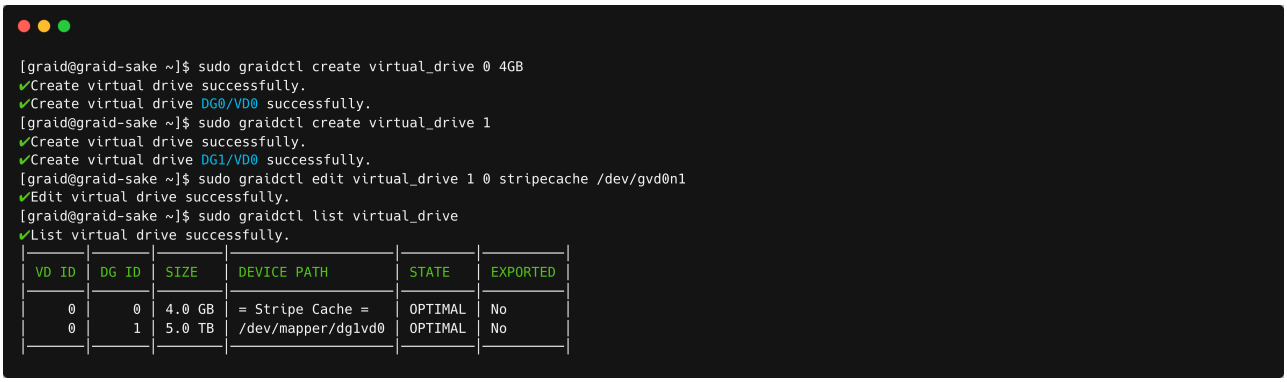
2. Assign a 4GB virtual disk as the stripe cache.

```
$ sudo graidctl edit virtual_drive 1 0 stripecache /dev/gvd0n1
```

3. Check the stripe cache:

```
$ sudo graidctl list virtual_drive
```

The assigned virtual drive is listed as '= Stripe Cache =' in DEVICE PATH column.
Output example:



To flush the stipe cache:

```
$ sudo graidctl edit vd 0 0 stripecache none
```

Basic Troubleshooting

Sequential Read Performance is not as Expected on a New Drive Group

Unlike SAS/SATA hard drives, many NVMe SSDs support the deallocate dataset management command. Using this command, you can reset all data in the NVMe SSD immediately, eliminationg the need to synchronize data between physical drives when creating a drive group.

But for other SSDs, the performance is not as expected when reading unwritten sectors after issuing the deallocate dataset management command. While this behavior also impacts the performance of the new drive group, it does not affect the applications because they do not read sectors that do not contain data.

To test GRAID perfomance, write the entire virtual drive sequentially using a large block size.

Kernel Log Message "failed to set APST feature (-19)" Appears When Creating Physical Drives

Some NVMe SSD models might display a "failed to set APST feature (-19)" message in the kernel log when creating the physical drive.

When SupremeRAID™ creates the physical drive, the SSD is unbound from the operating system so that SupremeRAID™ can control the SSD. During the unbinding process, when the APST feature is enabled, the NVMe driver attempts and fails to set the APST state to SSD, and the error message is issued.

This message is expected and can be ignored. SupremeRAID™ is functioning normally.

Different LED Blink Patterns on the Backplane

You might notice that the HDD/SSD activity indicator blink pattern is different on GRAID SupremeRAID™ than on traditional RAID cards.

GRAID SupremeRAID™ does not require a buffering or caching mechanism to improve read/write performance like traditional RAID cards. This feature causes GRAID SupremeRAID™ indicators to blink differently than traditional RAID cards.

Appendix

Manually Migrating the RAID Configuration Between Hosts

To manually migrate the RAID configuration between hosts:

1. Periodically backup the configuration file **/etc/graid.conf** from the original host. Use cp or scp to move the configuration file to another system.
2. Setup the target host and ensure that the GRAID service is stopped.

> **Note**
> When the target host already contains an installed and running SupremeRAID™ card, stop and restart the service using the **graid.conf** file from the original system.

3. Move all the SSDs from the original host to the new host.
4. Copy the configuration backup file to the new host using the same path.
5. Start the GRAID service directly if the original card also moved to the new host.

```
$ sudo systemctl start graid
```

Otherwise, you must apply the new license.

```
$ sudo graidctl apply license <LICENSE_KEY>
```

Monitoring System Input/Output Statistics for Devices Using iostat

The sysstat package contains the tools most commonly used to monitor I/O statistics in Linux systems. The sysstat package includes the iostat tool, which monitors system I/O device loading by observing the time the devices are active relative to their average transfer rates. The iostat command generates reports that enable you to fine tune the system configuration to better balance the I/O load between physical disks.

For example, to monitor specific devices and display statistics in megabytes per second (MBps):

```
$ iostat –m md124 sda nvme0n1
```

Output example:

```
[graid@graid-demo ~]$ iostat -m md124 sda nvme3n1
Linux 4.18.0-348.7.1.el8_5.x86_64 (graid-demo)  01/06/2022  _x86_64_    (16 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.01    0.00    0.15    0.00    0.00   99.84

Device             tps    MB_read/s    MB_wrtn/s    MB_dscd/s    MB_read    MB_wrtn    MB_dscd
md124             0.00         0.00         0.00         0.00          5          0          0
nvme3n1           0.00         0.00         0.00         0.00          1          0          0
sda               6.35         0.74         0.05         0.00      80843       5208          0
```

sysstat versions v12.3.3 and later

For sysstat versions v12.3.3 and later, the iostat tool includes an "alternative directory" feature that enables you to specify the directory from which to read device statistics.

- Add a "+f" parameter to the tool and use the "/sys/devices/virtual/graid/graid" sysfs device path to read device statistics from both the standard kernel files and from the files in the alternative directory.
- Add a "-f" parameter to the tool and use the "/sys/devices/virtual/graid/graid" sysfs device path to only read device statistics from the files in the alternative directory.

Alternative directory description from the iostat manual page.

```
-f directory
+f directory
       Specify an alternative directory for iostat to read devices statistics. Option -f tells iostat to use only the files located
       in the alternative directory, whereas option +f tells it to use both the standard kernel files and the files located in the
       alternative directory to read device statistics.

       directory is a directory containing files with statistics for devices managed in userspace.  It may contain:

       - a "diskstats" file whose format is compliant with that located in "/proc",
       - statistics for individual devices contained in files whose format is compliant with that of files located in "/sys".

       In particular, the following files located in directory may be used by iostat:

       directory/block/device/stat
       directory/block/device/partition/stat

       partition files must have an entry in directory/dev/block/ directory, e.g.:

       directory/dev/block/major:minor --> ../../block/device/partition
```

To check the iostat version:

```
$ iostat -V
```

Output example:

```
[graid@graid-demo ~]$ iostat -V
sysstat version 12.5.5
(C) Sebastien Godard (sysstat <at> orange.fr)
```

The gpd# statistics are not displayed in the iostat report without appending the "+f" and defining the sysfs path.

```
$ iostat -m +f /sys/devices/virtual/graid/graid gvd0nl md124 sda nvme0n1 gpd3
```

Output example:

```
[graid@graid-demo ~]$ iostat -m gvd0nl md124 sda nvme0n1 gpd3
Linux 4.18.0-348.7.1.el8_5.x86_64 (graid-demo)  01/06/2022  _x86_64_    (16 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.01    0.00    0.14    0.00    0.00   99.84

Device             tps    MB_read/s    MB_wrtn/s    MB_dscd/s    MB_read    MB_wrtn    MB_dscd
gvd0n1            0.68         0.00         0.00         0.00          1          0          0
md124             0.00         0.00         0.00         0.00          5          0          0
nvme0n1           0.00         0.00         0.00         0.00          1          0          0
sda               5.62         0.66         0.03         0.00     118093       5468          0
```

The gpd# statistics are displayed when "+f" is appended and the sysfs path is defined.

```
$ iostat -m +f /sys/devices/virtual/graid/graid gvd0nl md124 sda nvme0n1 gpd3
```

Output example:

```
[graid@graid-demo ~]$ iostat -m +f /sys/devices/virtual/graid/graid gvd0n1 md124 sda nvme0n1 gpd3
Linux 4.18.0-348.7.1.el8_5.x86_64 (graid-demo)  01/06/2022  _x86_64_    (16 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.01    0.00    0.15    0.00    0.00   99.84

Device             tps    MB_read/s    MB_wrtn/s    MB_dscd/s    MB_read    MB_wrtn    MB_dscd
gpd3              0.00         0.00         0.00         0.00          9          0          0
gvd0n1            0.00         0.00         0.00         0.00          2          0          0
md124             0.00         0.00         0.00         0.00          5          0          0
nvme0n1           0.00         0.00         0.00         0.00          1          0          0
sda               6.22         0.72         0.05         0.00      80853       5208          0
```

sysstat versions prior to v12.3.3

For operating systems with sysstat versions prior to v12.3.3 (CentOS for example), GRAID provides an alternate tool called "giostat" to display device statistics.

In the following example, the operating system version of iostat is prior to v12.3.3.

```
$ sudo yum list --installed |grep sysstat
```

Output example:

```
[graid@graid-demo ~]$ sudo yum list --installed |grep sysstat
sysstat.x86_64                                  11.7.3-6.el8                              @appstream
```

The giostat and iostat tools are very similar and their usage is the same. Set the parameter preferences using giostat.

Output example:

```
[graid@graid-demo ~]$ sudo graidctl list physical_drive;sudo graidctl list drive_group ;sudo graidctl list virtual_drive
✔List physical drive successfully.
|------------|-------|-------------|-----------------------------------------------------|--------------|----------|---------|--------------------|
| PD ID (5)  | DG ID | DEVICE PATH | NQN/WWID                                            | MODEL        | CAPACITY | SLOT ID | STATE              |
|------------|-------|-------------|-----------------------------------------------------|--------------|----------|---------|--------------------|
| 0          | 0     | /dev/gpd0   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z0G0A001T1L8    | KCM61VUL3T20 | 3.2 TB   | 12      | ONLINE             |
| 1          | 0     | /dev/gpd3   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z010A004T1L8    | KCM61VUL3T20 | 3.2 TB   | 19      | ONLINE             |
| 2          | 0     | /dev/gpd2   | nqn.2019-10.com.kioxia:KCM61VUL3T20:X0X0A01ET1L8    | KCM61VUL3T20 | 3.2 TB   | 18      | ONLINE             |
| 3          | 0     | /dev/gpd1   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A04HT1L8    | KCM61VUL3T20 | 3.2 TB   | 8       | ONLINE             |
| 4          | N/A   | /dev/gpd4   | nqn.2019-10.com.kioxia:KCM61VUL3T20:Z080A038T1L8    | KCM61VUL3T20 | 3.2 TB   | 0       | UNCONFIGURED_GOOD  |
|------------|-------|-------------|-----------------------------------------------------|--------------|----------|---------|--------------------|
✔List drive group successfully.
|-------|--------|--------|----------|--------|--------|----------|
| DG ID | MODE   | VD NUM | CAPACITY | FREE   | USED   | STATE    |
|-------|--------|--------|----------|--------|--------|----------|
|     0 | RAID6  |      4 | 6.4 TB   | 6.4 TB | 25 GB  | OPTIMAL  |
|-------|--------|--------|----------|--------|--------|----------|
✔List virtual drive successfully.
|------------|-------|---------|-------------|---------|-----------|
| VD ID (4)  | DG ID | SIZE    | DEVICE PATH | STATE   | EXPORTED  |
|------------|-------|---------|-------------|---------|-----------|
|          0 |     0 | 10 GB   | /dev/gvd0n1 | RESYNC  | No        |
|          1 |     0 | 5.0 GB  | /dev/gvd1n1 | RESYNC  | No        |
|          2 |     0 | 5.0 GB  | /dev/gvd2n1 | RESYNC  | No        |
|          3 |     0 | 5.0 GB  | /dev/gvd3n1 | RESYNC  | No        |
|------------|-------|---------|-------------|---------|-----------|
[graid@graid-demo ~]$ giostat -m gvd0n1 gpd3 nvme10n1 sda
Linux 4.18.0-348.2.1.el8_5.x86_64 (graid-demo)  01/06/2022  _x86_64_    (128 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.02    0.00    0.04    0.00    0.00   99.93

Device             tps    MB_read/s    MB_wrtn/s    MB_dscd/s    MB_read    MB_wrtn    MB_dscd
gpd3           1449.98         3.79         4.19         0.00    3355542    3707736          0
gvd0n1            0.05         0.01         0.00         0.00       9530          0          0
nvme10n1          0.00         0.00         0.00         0.00          1          0          0
sda               0.00         0.00         0.00         0.00         15          0          0
```

Setting Up the Auto-mount File Systems on Linux using the GRAID Driver

To set up the auto-mount file systems on Linux using the GRAID driver:

1. Create a virtual drive.

```
$ sudo graidctl create virtual_drive <DG_ID> [size] [flags]
```

2. Format the virtual drive and create a mount point for it.

```
$ sudo mkdir /mnt/<name-of-the-drive>
$ sudo mkfs.<file-system-type> /dev/gvd#n1
$ sudo mount /dev/gvd#n1 /mnt/<name-of-the-drive>/
```

3. Obtain the Name, UUID, and file system type.

```
$ sudo blkid
```

4. Edit the /etc/fstab file.

    1. Edit the /etc/fstab file.

    ```
    $ sudo vim /etc/fstab
    ```

    2. Append one line of code to the end of the file. Use the following format:

    ```
    UUID=<uuid-of-the-drive>  <mount-point>  <file-system-type>  <mount-option>  <dump>  <pass>
    ```

5. Update the mount dependency.

```
$ sudo graidctl update mount_dependency
```

> **Note**
>
> The mount point will auto-mount even after a system reboot.

Output example:

```
[graid@graid-demo ~]$ sudo graidctl create virtual_drive 0
✔Create virtual drive successfully.
✔Create virtual drive DG0/VD4 successfully.
[graid@graid-demo ~]$ sudo mkfs.ext4 /dev/gvd4n1
mke2fs 1.45.6 (20-Mar-2020)
Creating filesystem with 3637248 4k blocks and 909312 inodes
Filesystem UUID: b7aa98b4-d8b7-4123-bbee-e3d0752de973
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

[graid@graid-demo ~]$ sudo mount /dev/gvd4n1 /mnt/auto_mount/
[graid@graid-demo ~]$ sudo blkid
/dev/nvme1n1p1: UUID="cf83c157-7e54-c01c-a145-8c965a6eaaa9" UUID_SUB="7d80b309-0dec-bcd5-a722-6dd5df348f8c" LABEL="localhost:root"
TYPE="linux_raid_member" PARTUUID="11c78e0a-8f0d-4b44-8412-6cbdc80c1956"
/dev/nvme1n1p2: UUID="6795ce8e-8975-9a61-a597-4b5b40976113" UUID_SUB="7b04beee-9b0d-eded-24dc-66c7d77f770f" LABEL="localhost:boot"
TYPE="linux_raid_member" PARTUUID="5c915506-43be-4add-829d-320bb35d8eea"
/dev/nvme1n1p3: UUID="cd36b754-8fb7-636f-7077-8ce6c1172b76" UUID_SUB="0b9a90e9-03d7-6fa8-6efd-ca2388f631af" LABEL="localhost:boot_efi"
TYPE="linux_raid_member" PARTUUID="1723a6eb-bfdb-4b3f-bb73-616c24692162"
/dev/nvme1n1p4: UUID="a946683c-eec8-6ca9-2a90-db1fc7350ad6" UUID_SUB="edff470c-ff13-e0f3-086d-5202da944391" LABEL="localhost:swap"
TYPE="linux_raid_member" PARTUUID="8feefc8a-2045-4268-bdfd-ff546111ce4e"
/dev/nvme0n1p1: UUID="cf83c157-7e54-c01c-a145-8c965a6eaaa9" UUID_SUB="2bde7b08-6d5b-8a99-3236-3ccb16fd7a4d" LABEL="localhost:root"
TYPE="linux_raid_member" PARTUUID="5043925a-322a-45e1-a5c2-a575d05ff6af"
/dev/nvme0n1p2: UUID="6795ce8e-8975-9a61-a597-4b5b40976113" UUID_SUB="3a2afb38-4d29-b31b-1f7e-66abdfa85227" LABEL="localhost:boot"
TYPE="linux_raid_member" PARTUUID="d12aefb3-e844-40b7-8a78-f59a7e3e5bdf"
/dev/nvme0n1p3: UUID="cd36b754-8fb7-636f-7077-8ce6c1172b76" UUID_SUB="70ee39ac-b9ac-5876-ac2b-1d35149958e1" LABEL="localhost:boot_efi"
TYPE="linux_raid_member" PARTUUID="e65af2b7-4a14-4adb-86fa-e97a5c029618"
/dev/nvme0n1p4: UUID="a946683c-eec8-6ca9-2a90-db1fc7350ad6" UUID_SUB="98fc65ee-be6e-425c-cff2-943660d40f9b" LABEL="localhost:swap"
TYPE="linux_raid_member" PARTUUID="dd656e5f-a46b-4647-b034-c853e94c6abe"
/dev/md127: UUID="a069e3d2-04c8-4576-a6d9-af0e06495985" TYPE="swap"
/dev/md126: UUID="1c807345-02fd-49b0-a4ee-444d5beca6c9" BLOCK_SIZE="512" TYPE="xfs"
/dev/md125: UUID="6828ce26-088e-4d72-ae95-89b68b3fd17d" BLOCK_SIZE="512" TYPE="xfs"
/dev/md124: UUID="7520-A102" BLOCK_SIZE="512" TYPE="vfat"
/dev/gvd3n1: UUID="44ae1115-14a5-46f2-b86b-3a533e52f6f8" BLOCK_SIZE="4096" TYPE="ext4"
/dev/nvme1n1: PTUUID="6310375f-ea2c-4df0-af40-59c7726e0736" PTTYPE="gpt"
/dev/nvme0n1: PTUUID="e2fdfb97-da06-44dc-864e-f6bf5a1c4062" PTTYPE="gpt"
/dev/gvd4n1: UUID="b7aa98b4-d8b7-4123-bbee-e3d0752de973" BLOCK_SIZE="4096" TYPE="ext4"
[graid@graid-demo ~]$ sudo vim /etc/fstab
[graid@graid-demo ~]$ sudo cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Mon Nov 15 04:51:50 2021
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=6828ce26-088e-4d72-ae95-89b68b3fd17d /                       xfs     defaults        0 0
UUID=1c807345-02fd-49b0-a4ee-444d5beca6c9 /boot                   xfs     defaults        0 0
UUID=7520-A102          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
UUID=a069e3d2-04c8-4576-a6d9-af0e06495985 none                    swap    defaults        0 0
UUID=b7aa98b4-d8b7-4123-bbee-e3d0752de973 /mnt/auto_mount          ext4    defaults        0 0
[graid@graid-demo ~]$ sudo graidctl update mount_dependency
✔Update mount dependency successfully.
✔Update mount dependency Added mount point: /mnt/auto_mount successfully.
```

6. Remove the mount dependency.

```
$ sudo graidctl update mount_dependency
```

Output example:

```
[graid@graid-demo ~]$ sudo cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Mon Nov 15 04:51:50 2021
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=6828ce26-088e-4d72-ae95-89b68b3fd17d /                       xfs     defaults        0 0
UUID=1c807345-02fd-49b0-a4ee-444d5beca6c9 /boot                   xfs     defaults        0 0
UUID=7520-A102          /boot/efi               vfat    umask=0077,shortname=winnt 0 2
UUID=a069e3d2-04c8-4576-a6d9-af0e06495985 none                    swap    defaults        0 0
#UUID=b7aa98b4-d8b7-4123-bbee-e3d0752de973 /mnt/auto_mount          ext4    defaults        0 0
[graid@graid-demo ~]$ sudo graidctl update mount_dependency
✔Update mount dependency successfully.
✔Update mount dependency Removed mount point: /mnt/auto_mount successfully.
```

> **Note**
>
> To disable the automount point or delete the virtual drive, edit the /etc/fstab file and delete/comment that entry.
>
> Then, execute update mount_dependency to unmount the virtual drive.

Enabling Virtual Machines with GPU Passthrough

You can create virtual machines with GRAID SupremeRAID™ support to maximize performance.

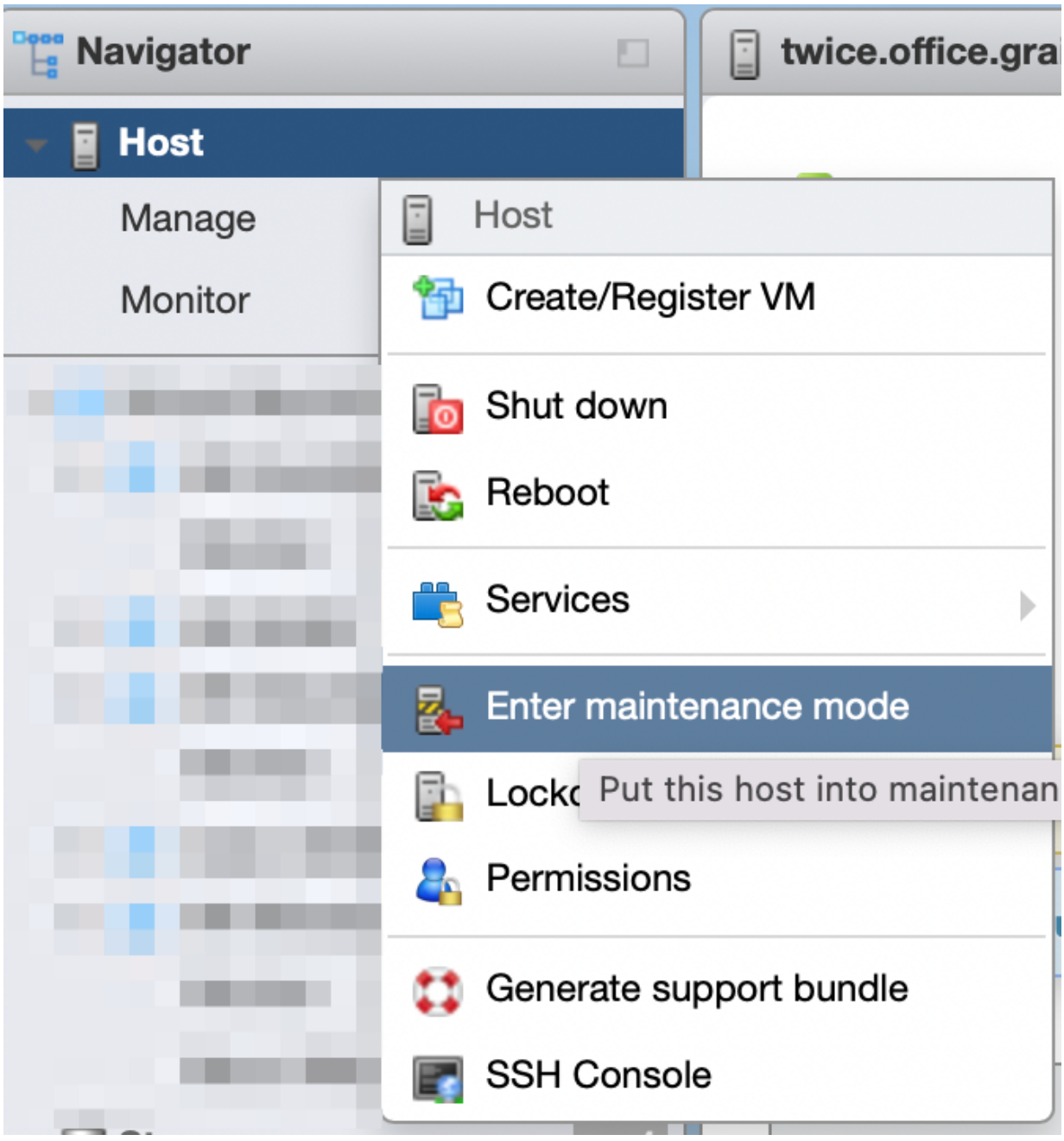
Hypervisor support:

- VMWare ESXi 7.0U3

Configuring Hosts for NVIDIA GPU Device Passthrough

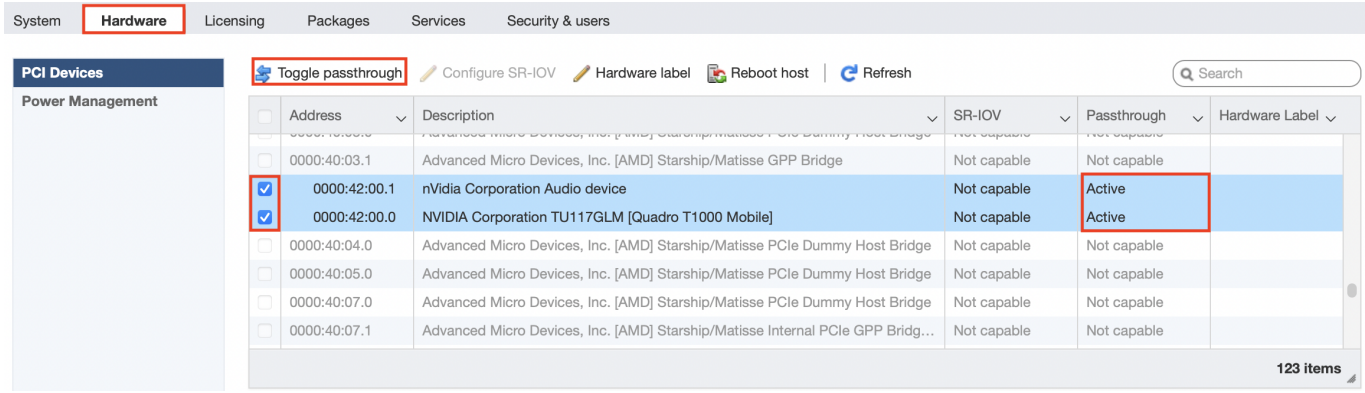
Put the ESXi host into maintenance mode

1. From the **Navigator** menu, select **Host** > **Enter maintenance mode**.



Manage PCI Device Passthrough

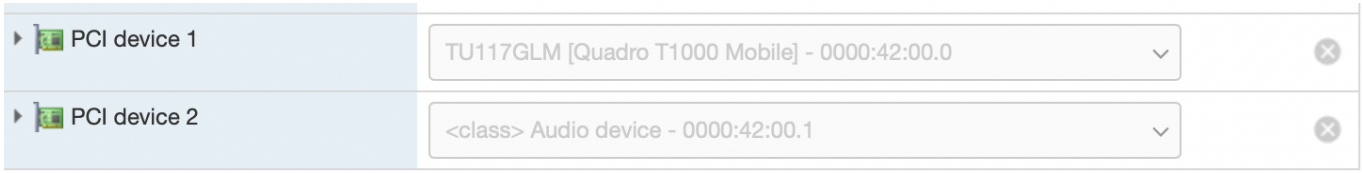
1. From the **Navigator** menu, select **Manage** > **Hardware** > **PCI Devices**. The **Passthrough Configuration** page appears listing all of the available pass-through devices.
2. Select the NVIDIA T1000 (Quadro T1000 Mobile) and its audio device.
3. Click **Toggle passthrough**.
4. Check the Passthrough status. It should be **Active**.



Configuring Virtual Machines

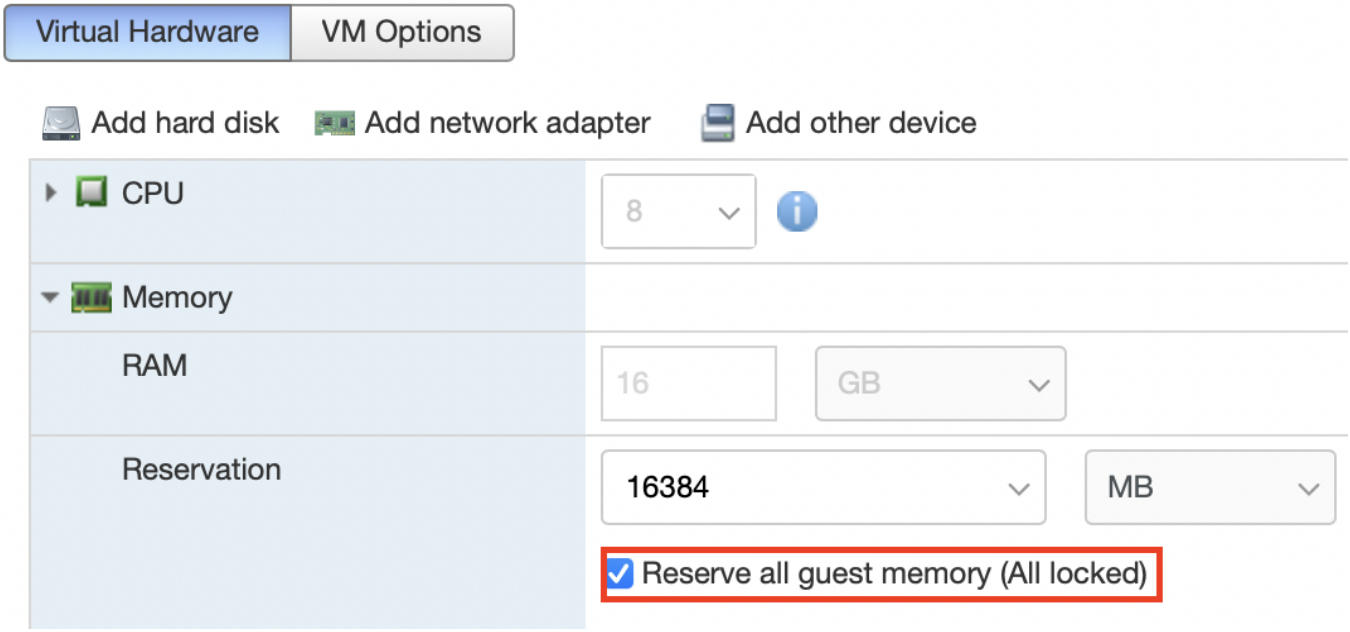
Attach PCI devices to the virtual machine.

1. From the **Edit VM setting** page, choose **Virtual Hardware** > **Add other device** > **PCI device**.
2. Choose Quadro T1000 and its Audio device as the two PCI devices.



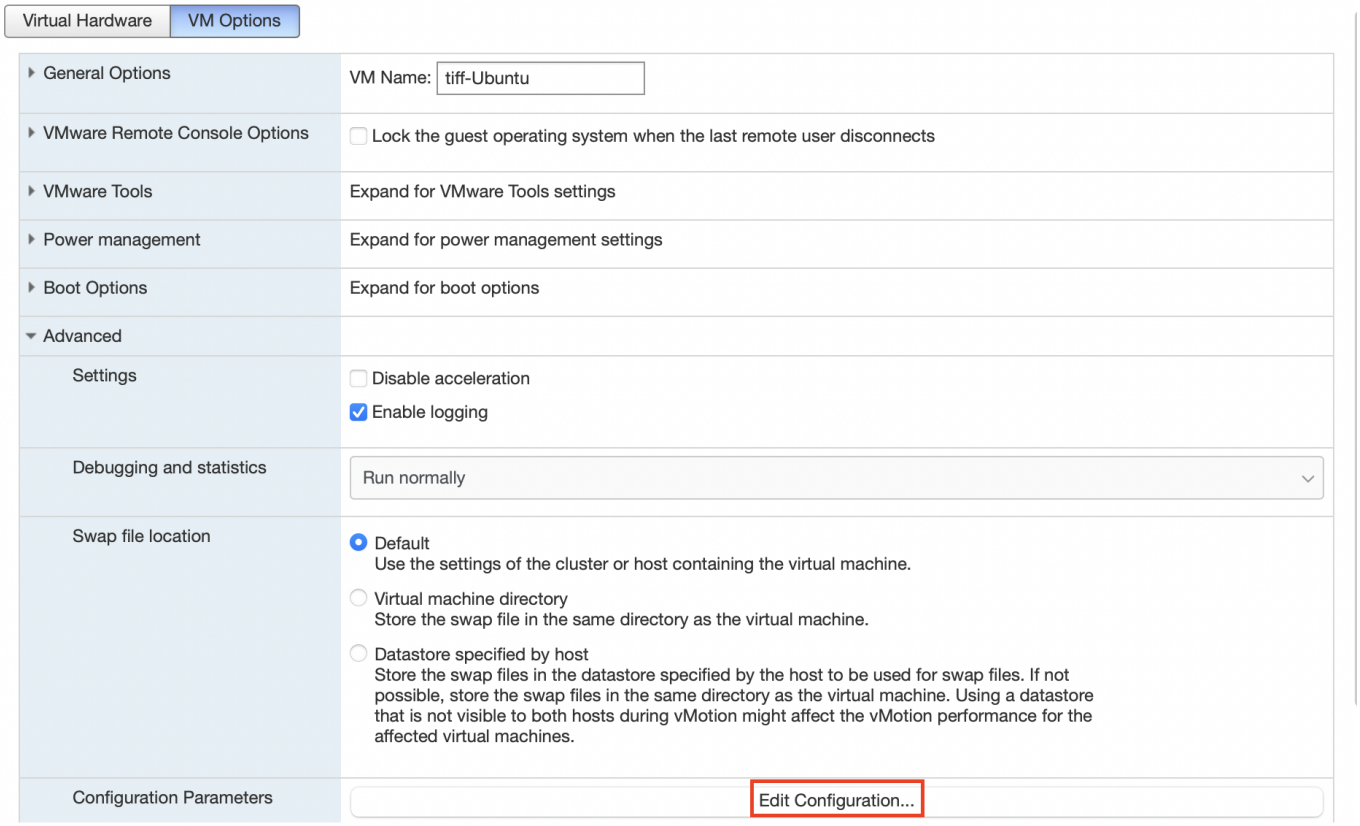
> **Note:**
> When the T1000 PCI device is assigned to the virtual machine, you must set the memory reservation to accomodate the fully configured memory size.

3. Choose **Virtual Hardware** > **Memory**.
4. Check **Reserve all guest memory (All locked)**.



Enabling Point-to-Point (P2P) on the Virtual Machine for Best Performance

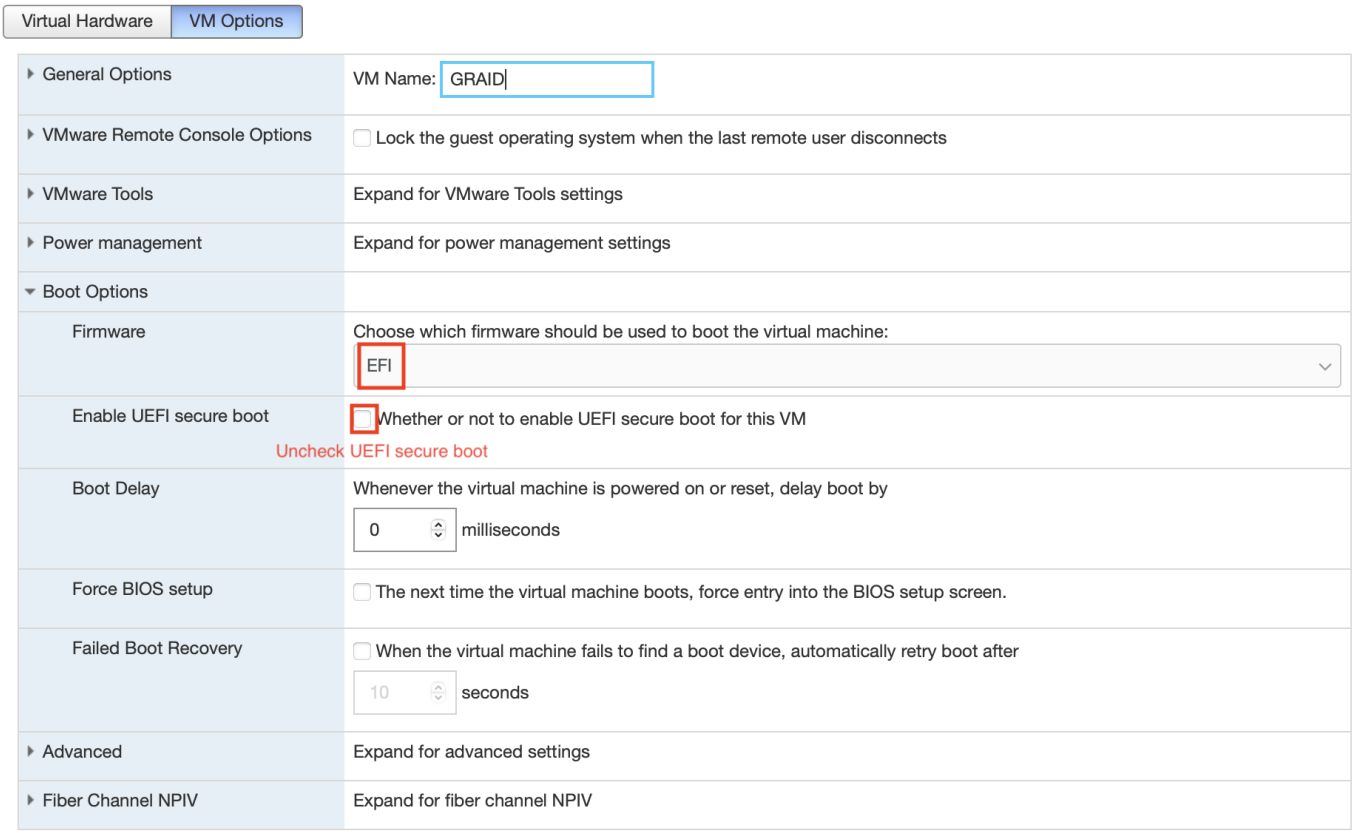
1. From the **Edit VM setting** page, choose **VM Options** > **Advanced** > **Configuration Parameters** > **Edit Configuration...**.



2. Add the following two parameters:

```
hypervisor.cpuid.v0 = “FALSE”
pciPassthru.allowP2P = “TRUE”
```

3. From the **Edit VM setting** page, choose **VM Options** > **Boot Options** > **Firmware** > **EFI**.
4. Uncheck **Whether or not to enable UEFI secure boot for this VM**.



Setting Up a Self-Encrypting Drive (SED)

A SED utilizes native full-disk encryption. GRAID SupremeRAID™ supports SEDs and SED key management. When the SED key is configured, GRAID SupremeRAID™ uses the imported key to unlock the SED.

> **Note:**
> You must configure the SED key using the graidctl tool before creating the physical drives.

Prerequisites

- Collect the NQN/WWID of the NVMe disks. (They are required to import the SED key.)
- Prepare the SED key for each disk.

Limitations

- Only NVMe devices are supported.
- Only the "global" range parameter is supported.

Importing a Single SED Key using NQN/WWID

```
$ sudo graidctl edit config sed <NQN/WWID>
```

```
graid@graid:~$ sudo graidctl edit config sed nqn.2014-08.org.nvmexpress:uuid:52bbdb40-c5bf-f92d-9961-a6368e845bfd
Enter Key:
✔ Edit config sed successfully.
```

Importing a Batched SED Key using NQN/WWID

```
$ sudo graidctl edit config sed file <filename>
file content format:
<NQN1/WWID1>, <KEY1>
<NQN1/WWID1>, <KEY2>
...
<NQNn/WWIDn>, <KEYn>
```

Displaying SED Key Information

```
$ sudo graidctl describe config sed
```

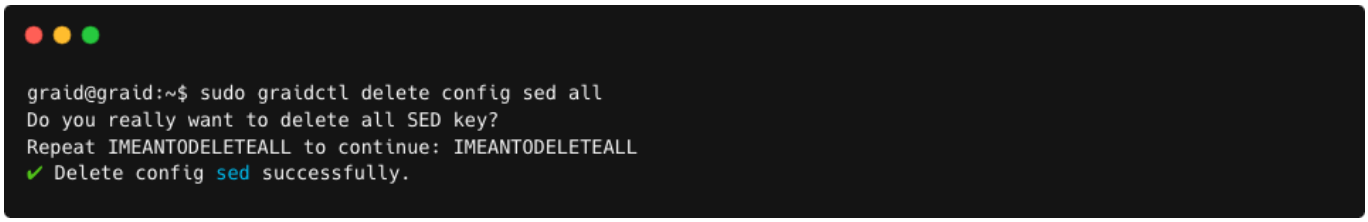
```
graid@graid:~$ sudo graidctl describe config sed
Totally 1 GUIDs have SED key:
nqn.2014-08.org.nvmexpress:uuid:52bbdb40-c5bf-f92d-9961-a6368e845bfd
```

Deleting SED Keys

```
$ graidctl delete config sed <GUID>
```

To delete all keys:

```
$ graidctl delete config sed all
```



Setting Boot-Drive Devices

You can set two NVMe SSDs as RAID1 boot devices and control them using GRAID SupremeRAID™. The methods used to set NVMe SSDs as RAID1 boot devices depend upon the operating system in use.

Prerequisites

- Two NVMe SSDs to set as RAID1 boot devices.

Limitation

- Installation on multiple operating systems is not supported.

Setup by Operating System

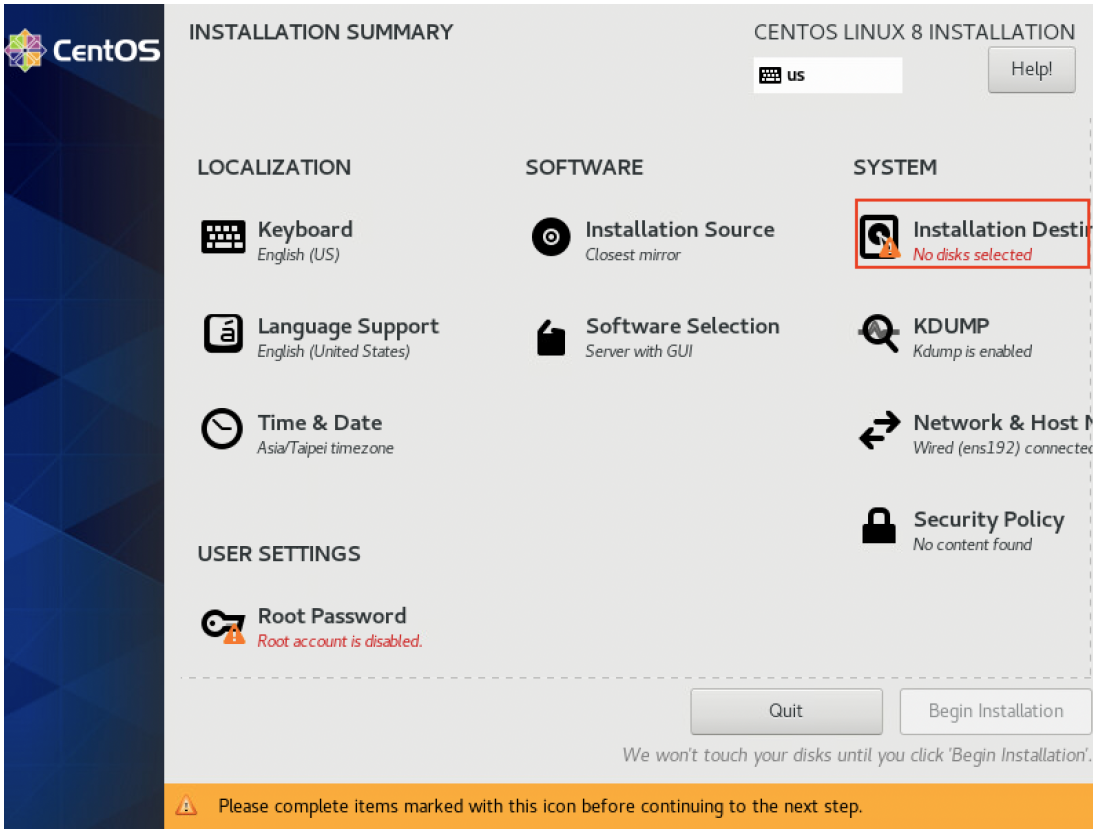
CentOS

Assigning RAID1 Boot Devices Manually

You assign RAID1 boot devices when you install CentOS. If you are not prompted by the CentOS GUI to assign the boot devices, you can assign them manually.

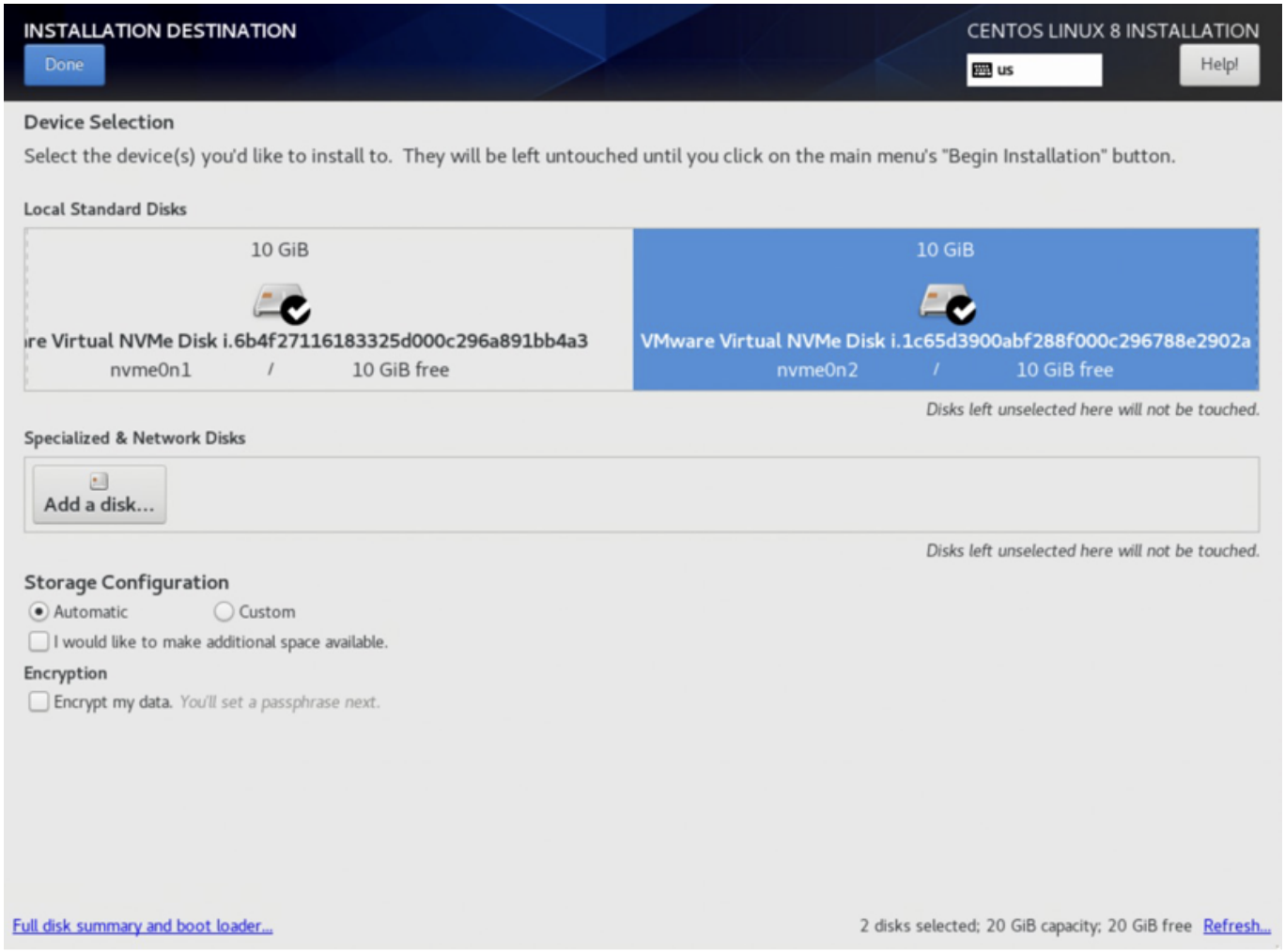
To manually assign the RAID 1 boot devices:

1. From the **INSTALLATION SUMMARY** page, choose **SYSTEM** > **Installation Destination**.



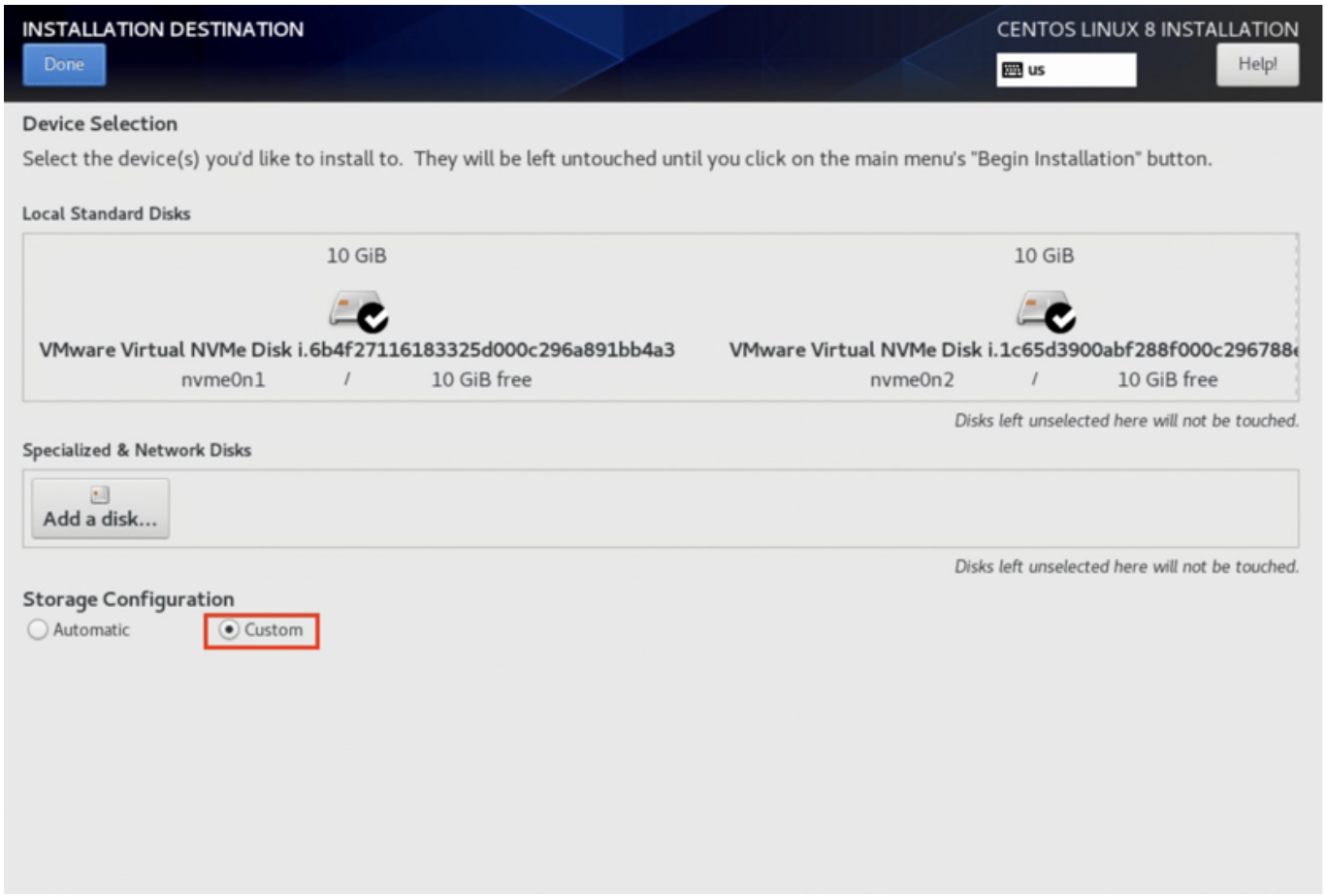
2. From the **INSTALLATION DESTINATION** page, select the two NVMe SSDs that you want to set as RAID1 boot devices.

Tip: Use the “Ctrl” key to select multiple devices.



3. Choose **Custom** for the **Storage Configuration**.

4. Click **Done** to continue.

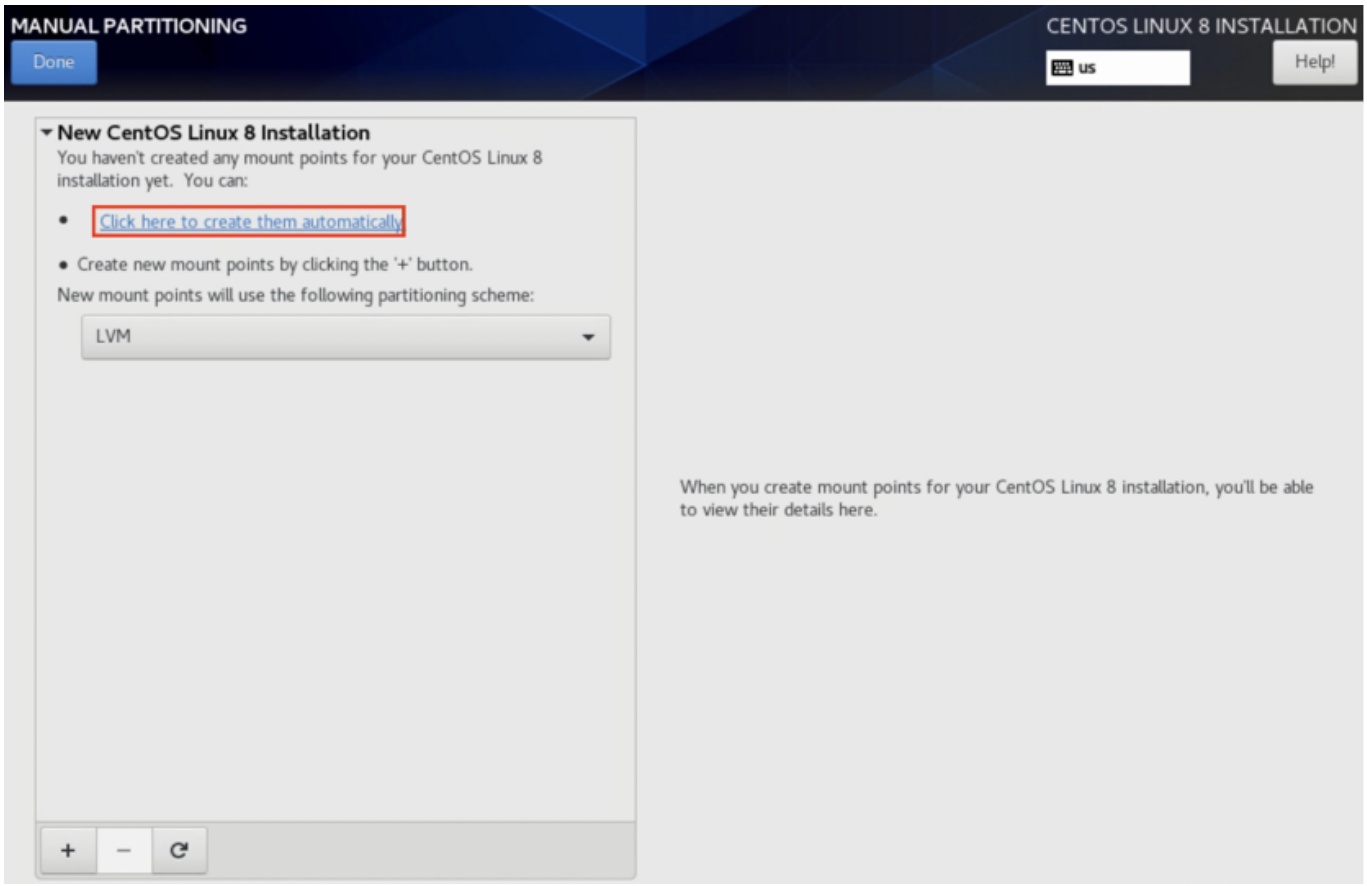


Creating Storage Partitions Manually

You manually create the storage partitions on CentOS systems.

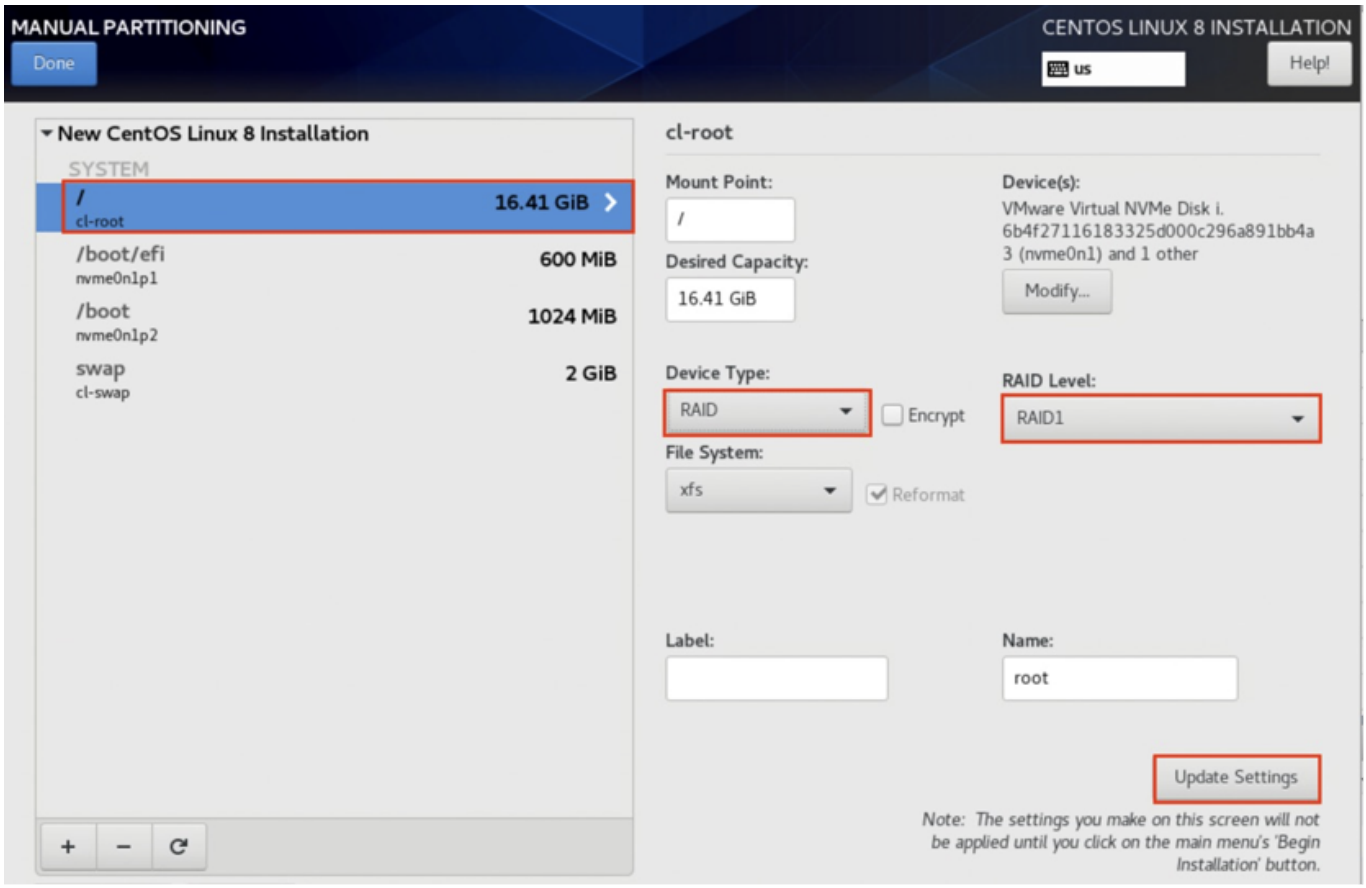
To manually create partitions:

1. From the **MANUAL PARTITIONING** page, choose **New CentOS Linux 8 Installation**.
2. Click **Click here to create them automatically** to create the mount points.



3. Set the **Device Type** to **RAID** and set the **RAID LEVEL** to **RAID 1**.

4. Click **Update Settings**.



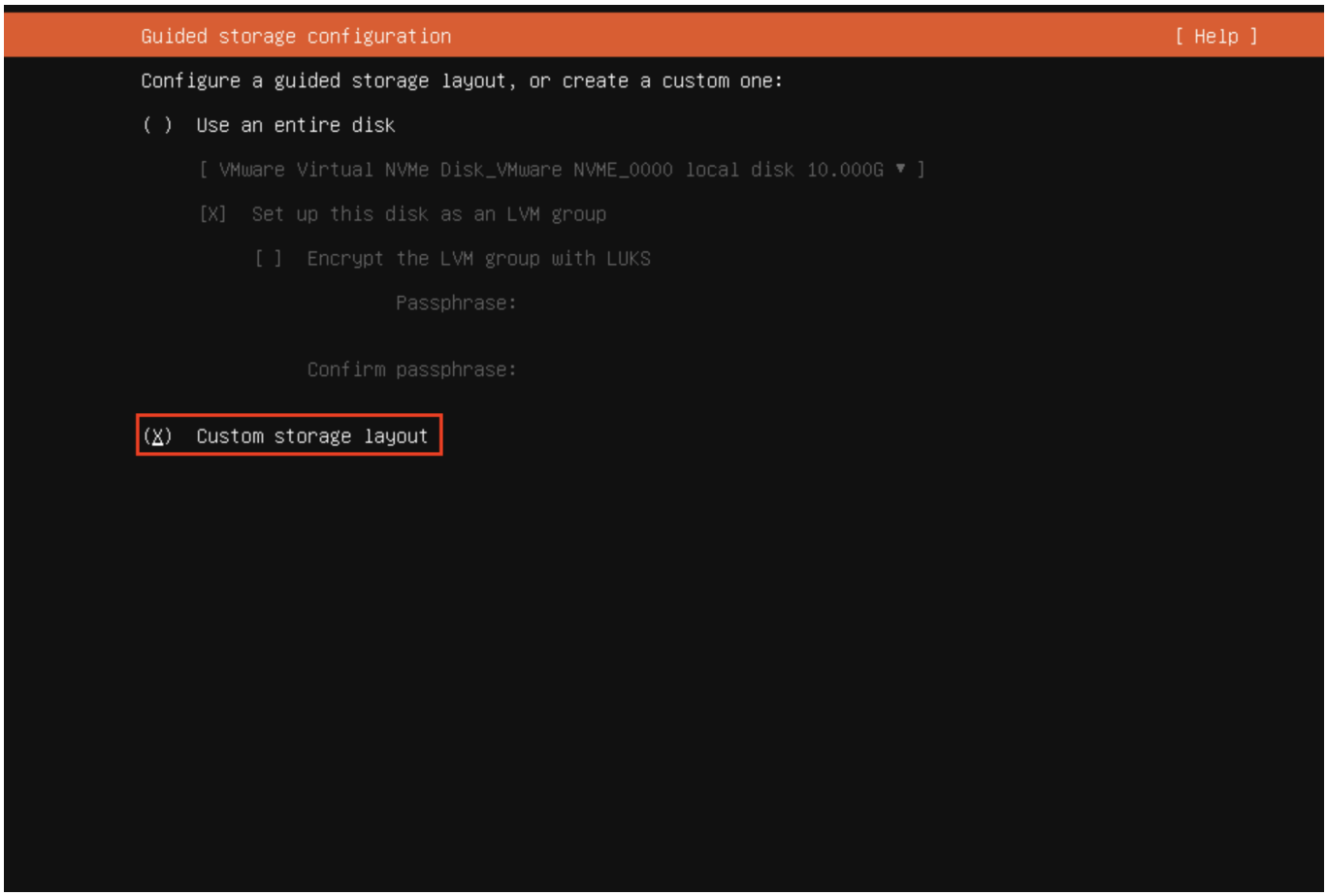
Ubuntu Server 20.04

Creating and Configuring Storage Partitions

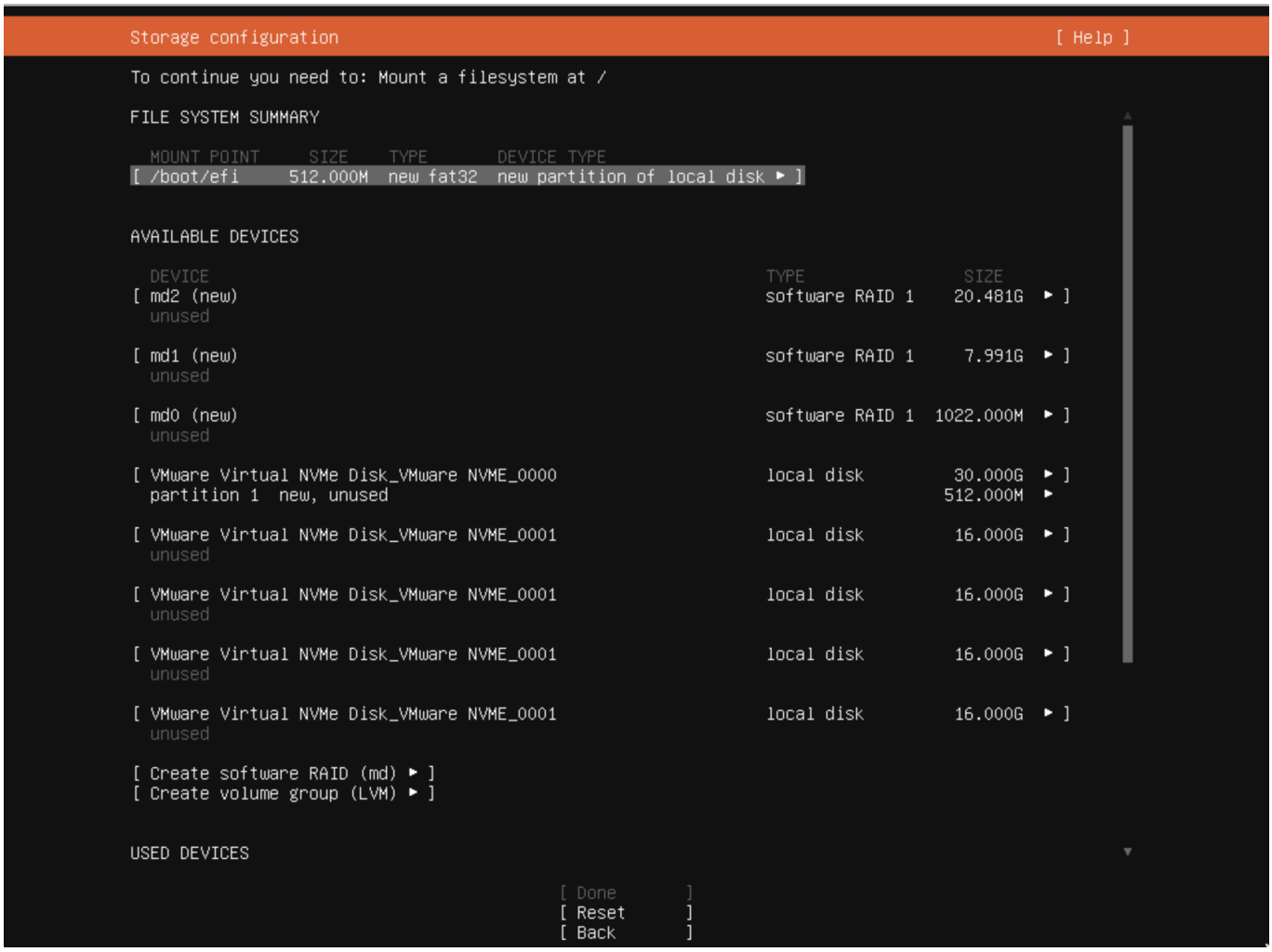
Storage partitions must be created and configured during Ubuntu Server 20.04 installations. The partitions are required for mounting /boot, swap, and root/. Each partition functions as a soft RAID.

To create the storage partitions:

1. From the **Guided storage configuration** page, select **Custom storage layout**.



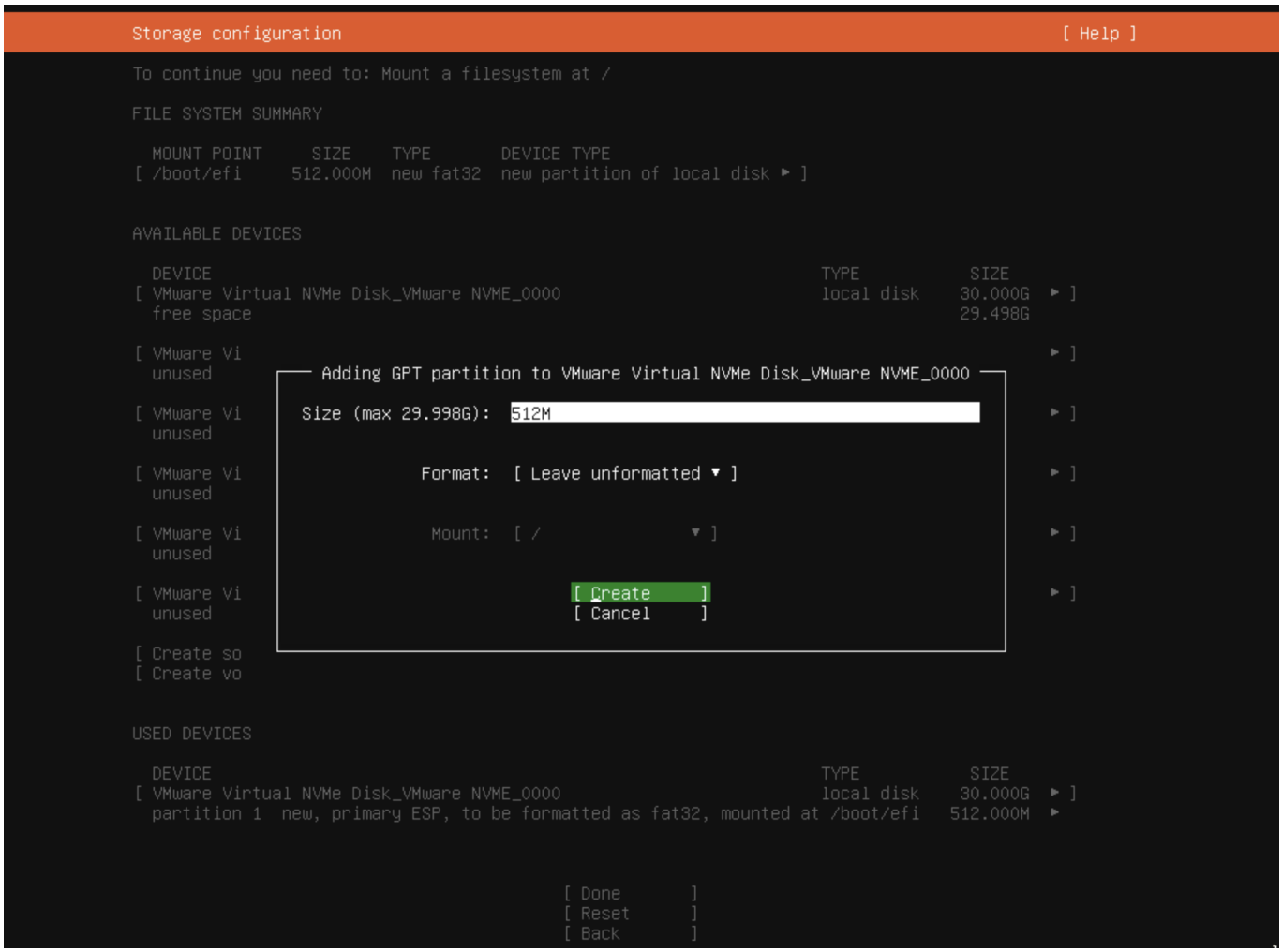
2. From the **Storage configuration** page, select the first disk as the boot disk.



3. From the second **Disk** menu, choose **Add GPT partition** > **Create a partition**.
4. Set the **Size** of the new partition. Use the same size as the boot disk so that the first and second partitions align.
5. For the **Format**, select **[Leave unformatted]**.

> **Note:**
> You must use **[Leave unformatted]**.
> DO NOT mount the partition. Setting RAID1 and mounting partitions on multiple drives (MD) occurs later in the process.

6. Select **[Create]** to create the storage partition.

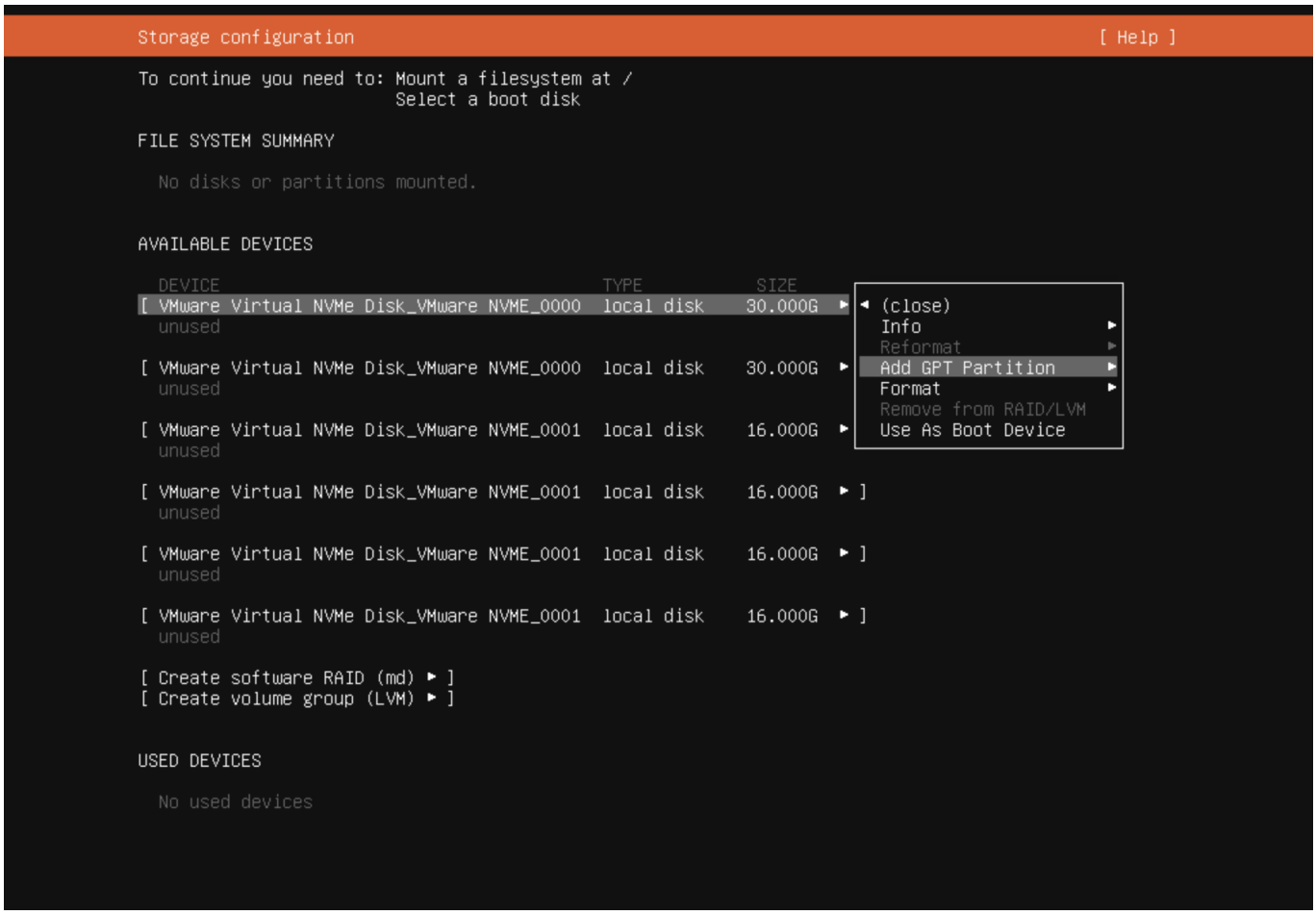


Configuring the Boot Partitions

This process configures the /boot, swap, and root/ partitions on both disks.

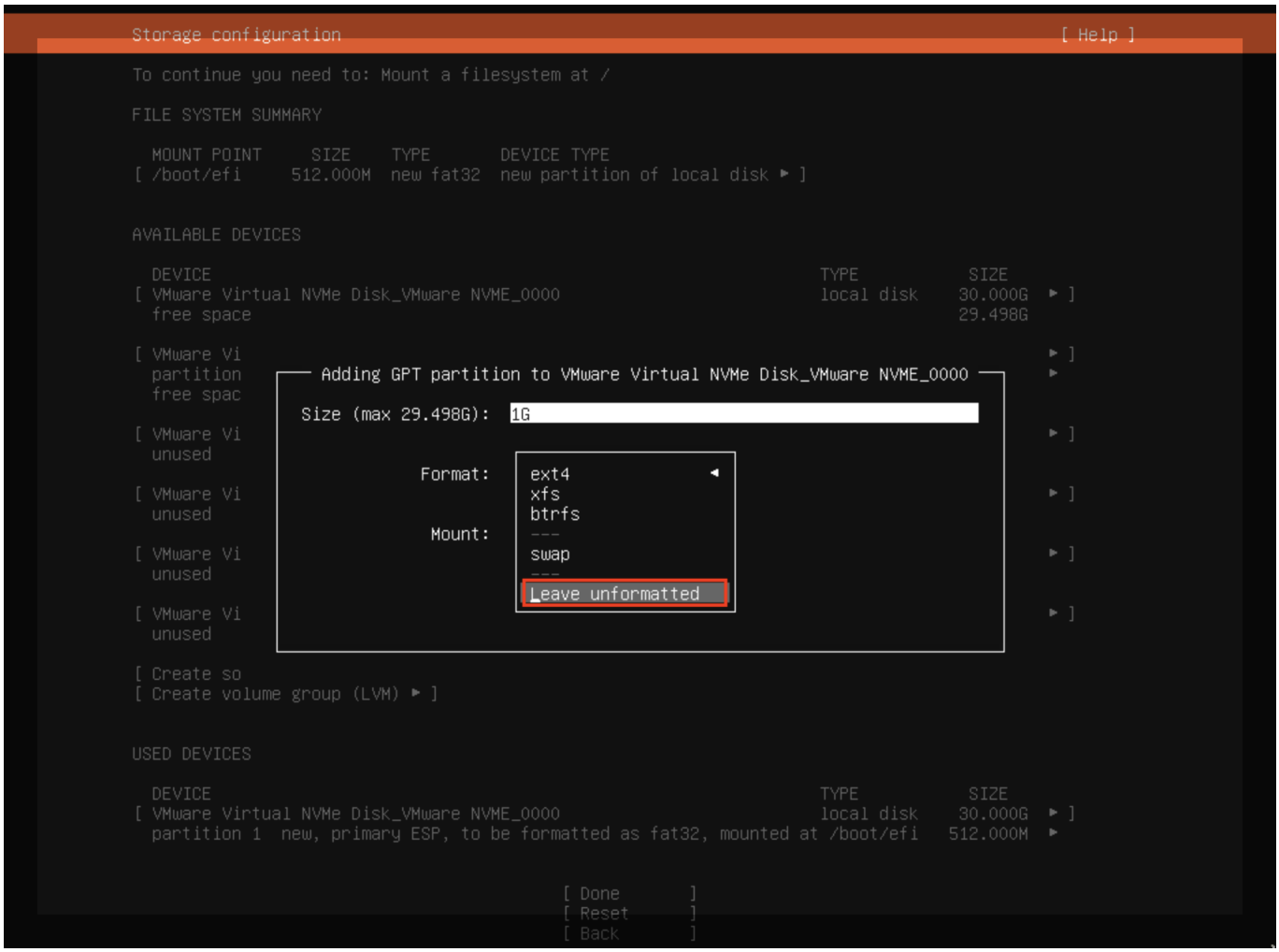
To configure the partitions:

1. From the **Storage configuration** page **Disk** menu, select **Add GPT Partition**.



2. Set the **Size** of the partitions. Use 1G for /boot, the memory size for swap, and the remaining size for root/.

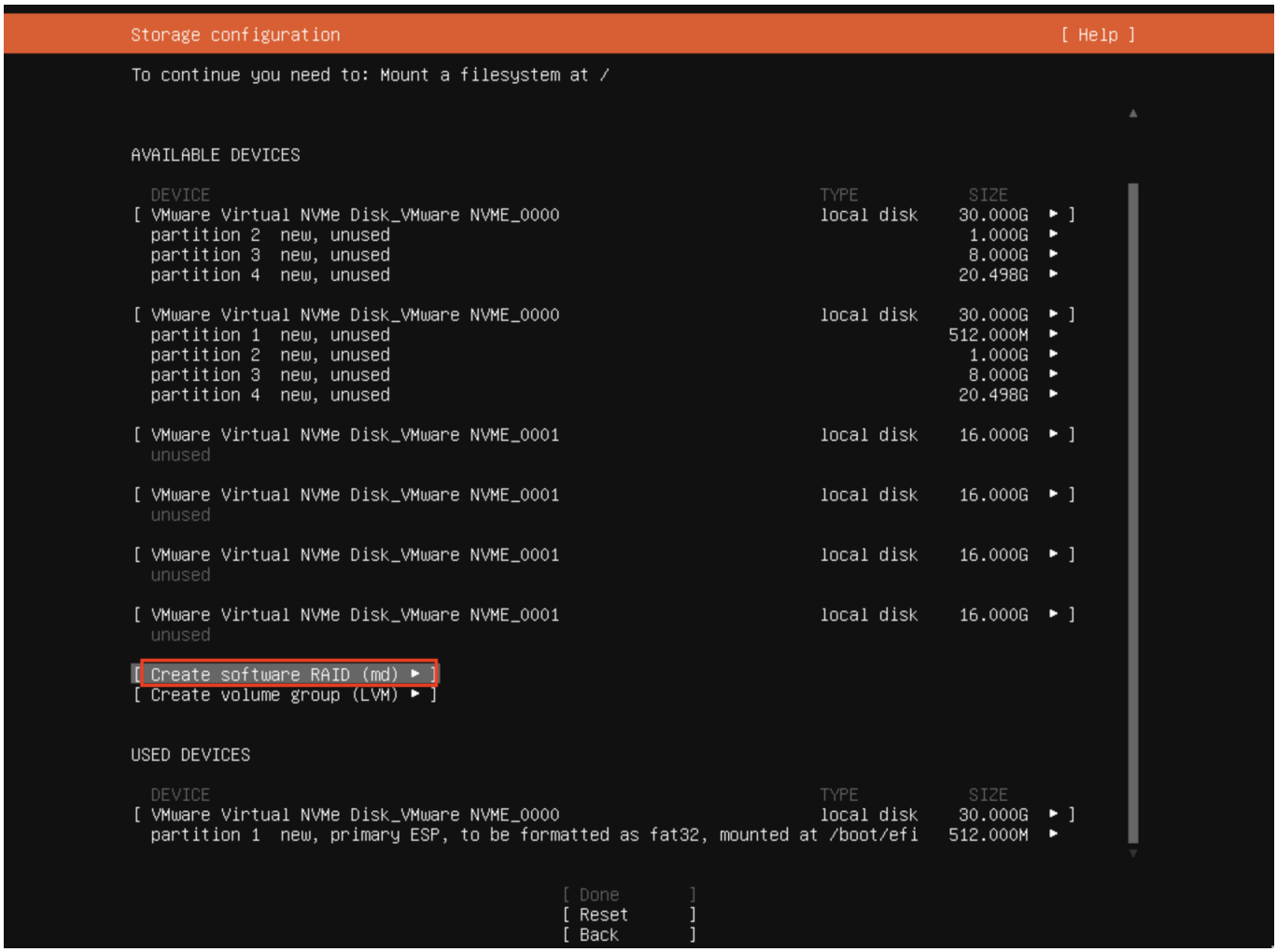
3. For the **Format**, select **[Leave unformatted]**.



Creating a Software RAID for Multiple Devices (MD)

To create the software RAID on multiple devices:

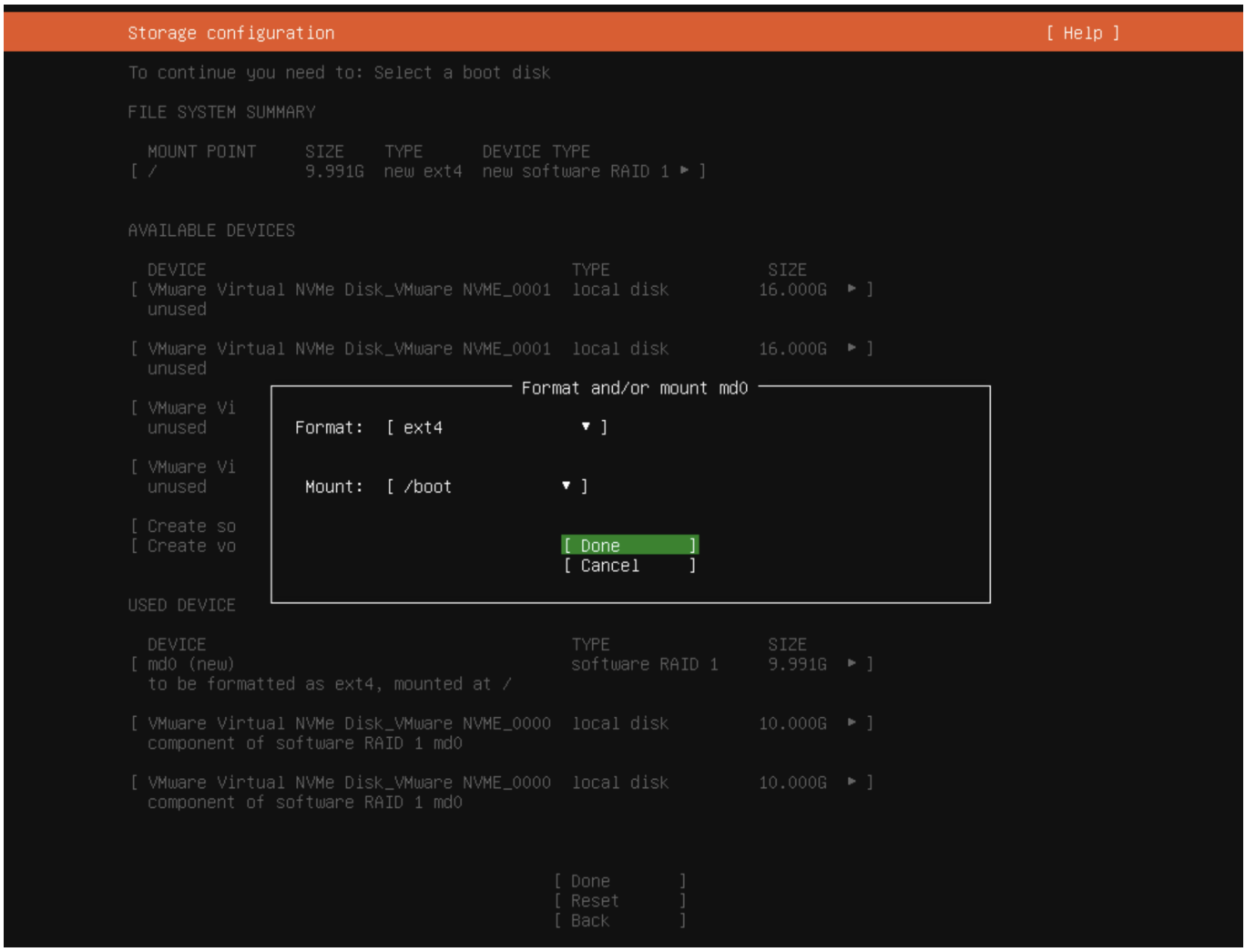
1. From the **Storage configuration** page, select **Create software RAID (md)**.



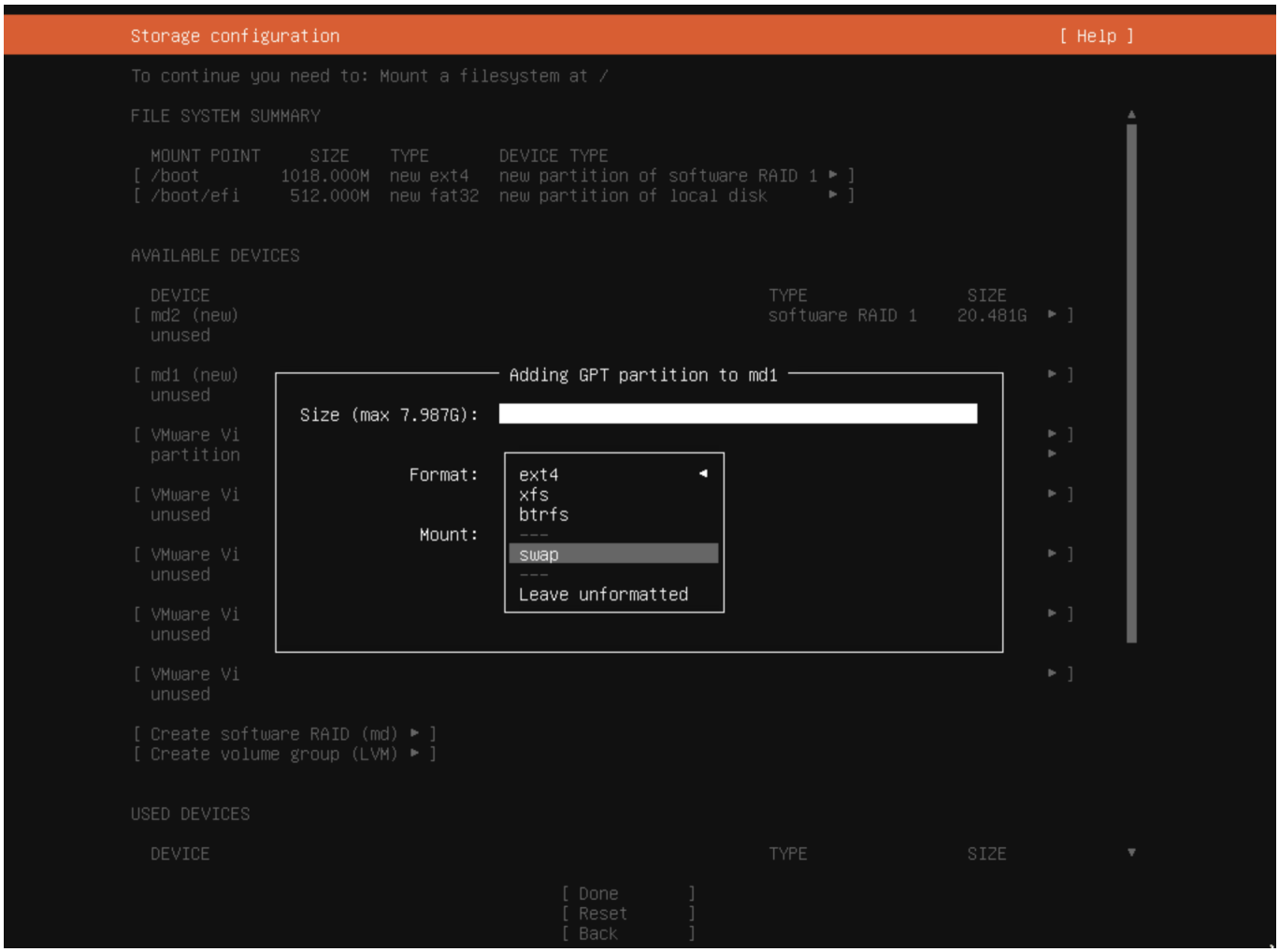
Setting MD as the Mounting Point

To set MD as the mounting point:

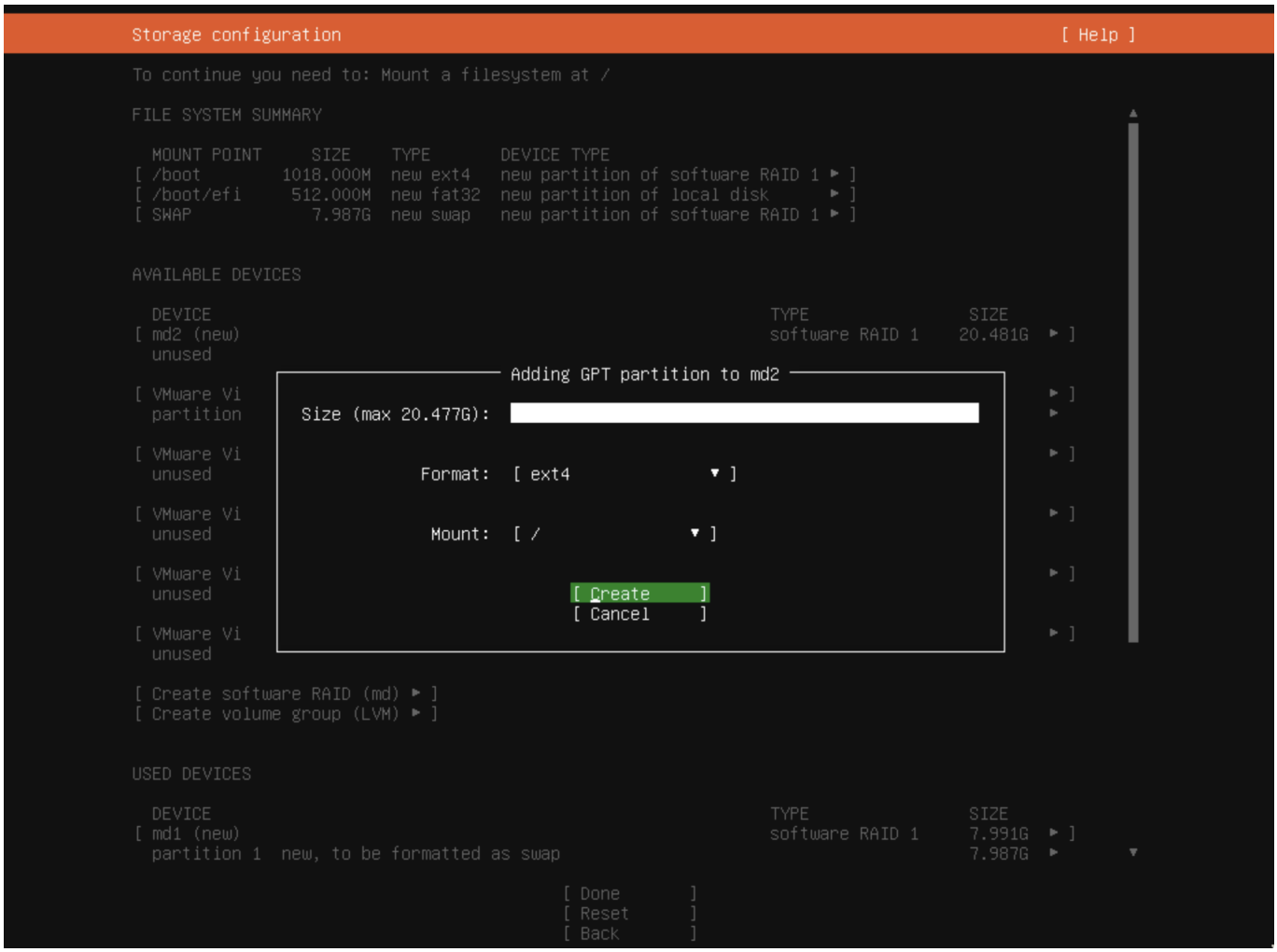
1. From the **Storage configuration** page **Disk** menu, set md0 as the /boot mounting point.



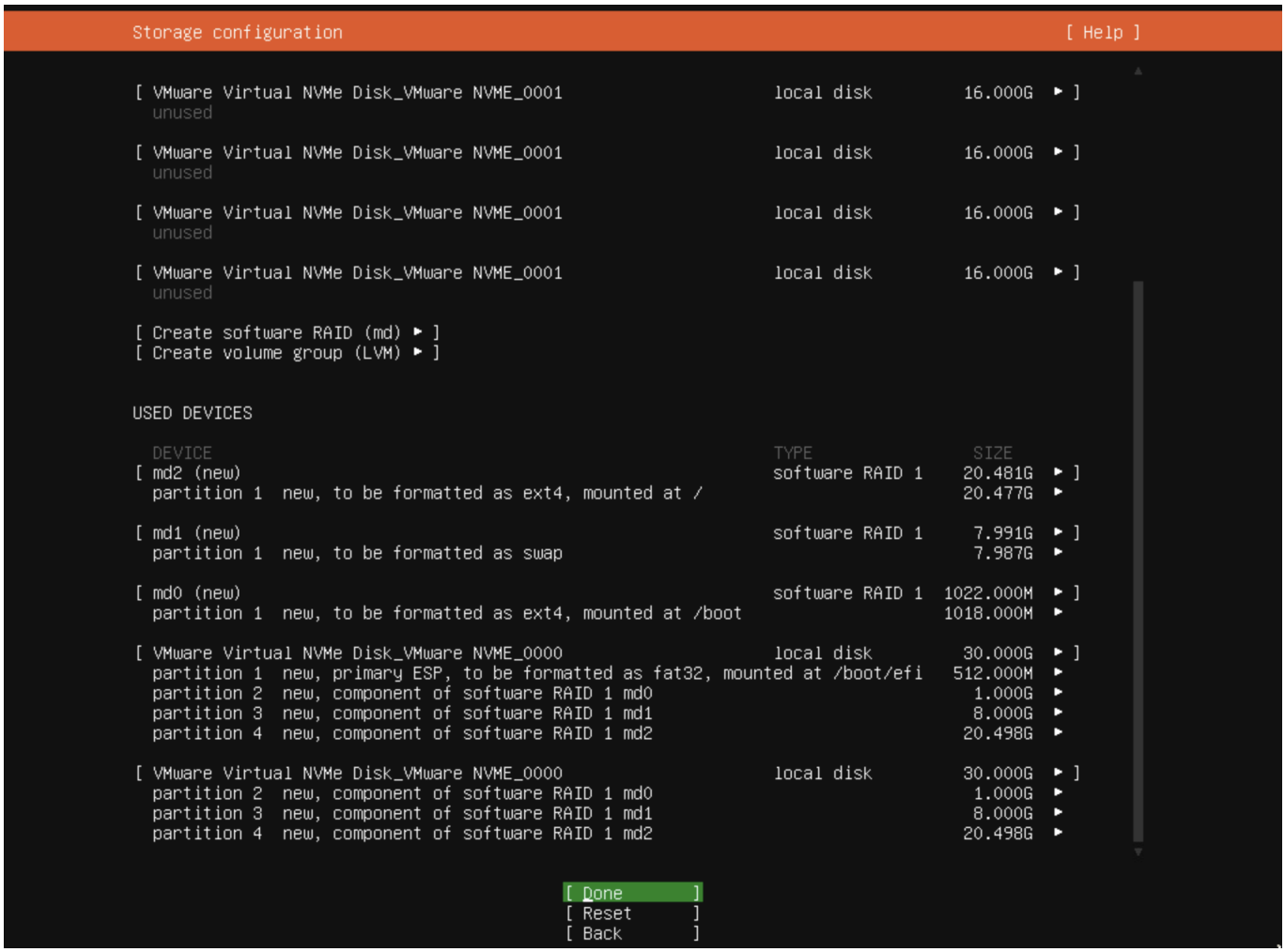
2. From the **Disk** menu, select **Add GPT Partition**, set md1 as the swap mounting point.



3. From the **Disk** menu, select **Add GPT Partition**, set md2 as the root / mounting point.

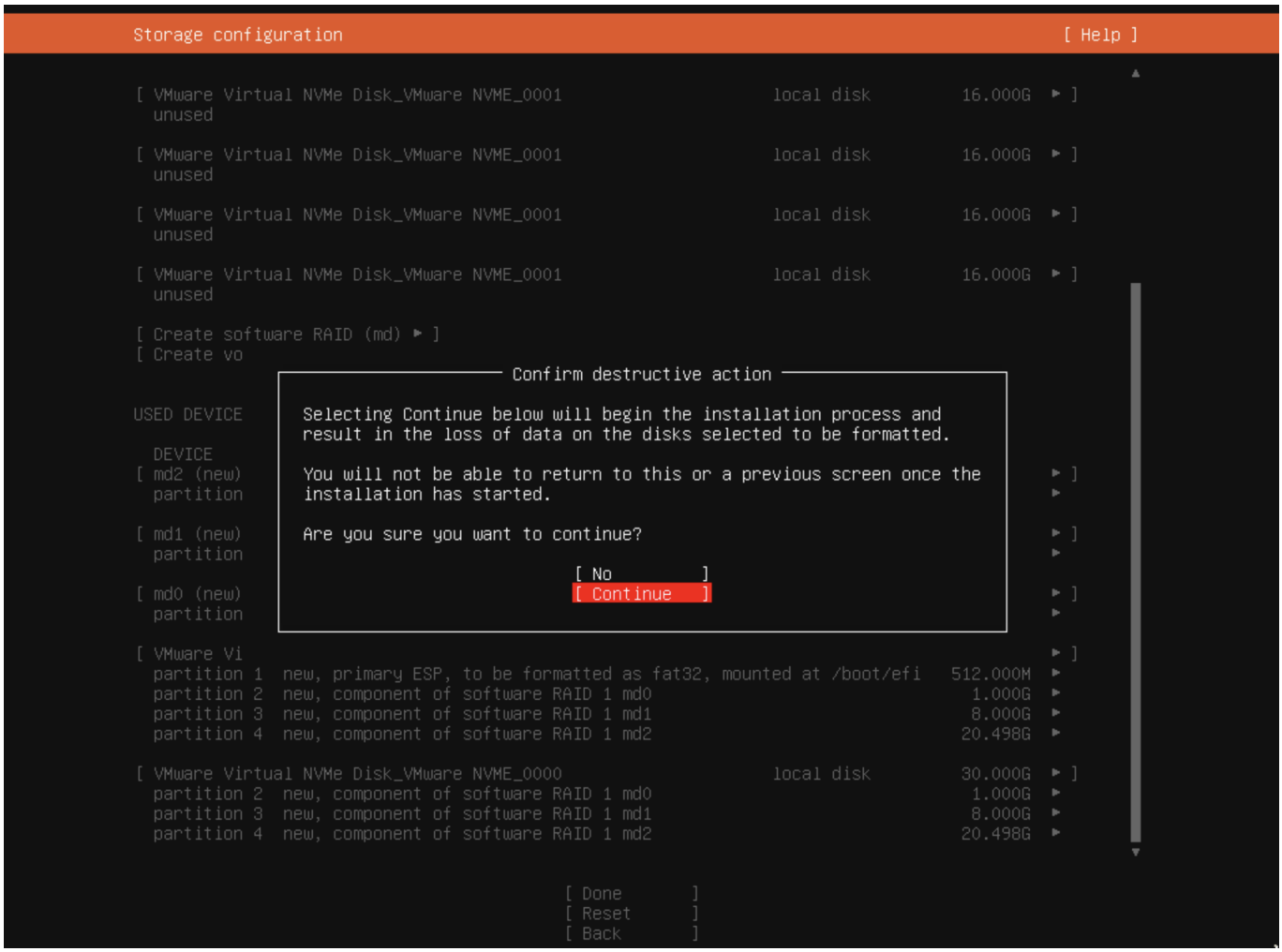


4. Click **Done** when you finish setting the mount points.



5. From the **Confirm destructive action** popup, select **Continue**.

The partition settings are now in effect.

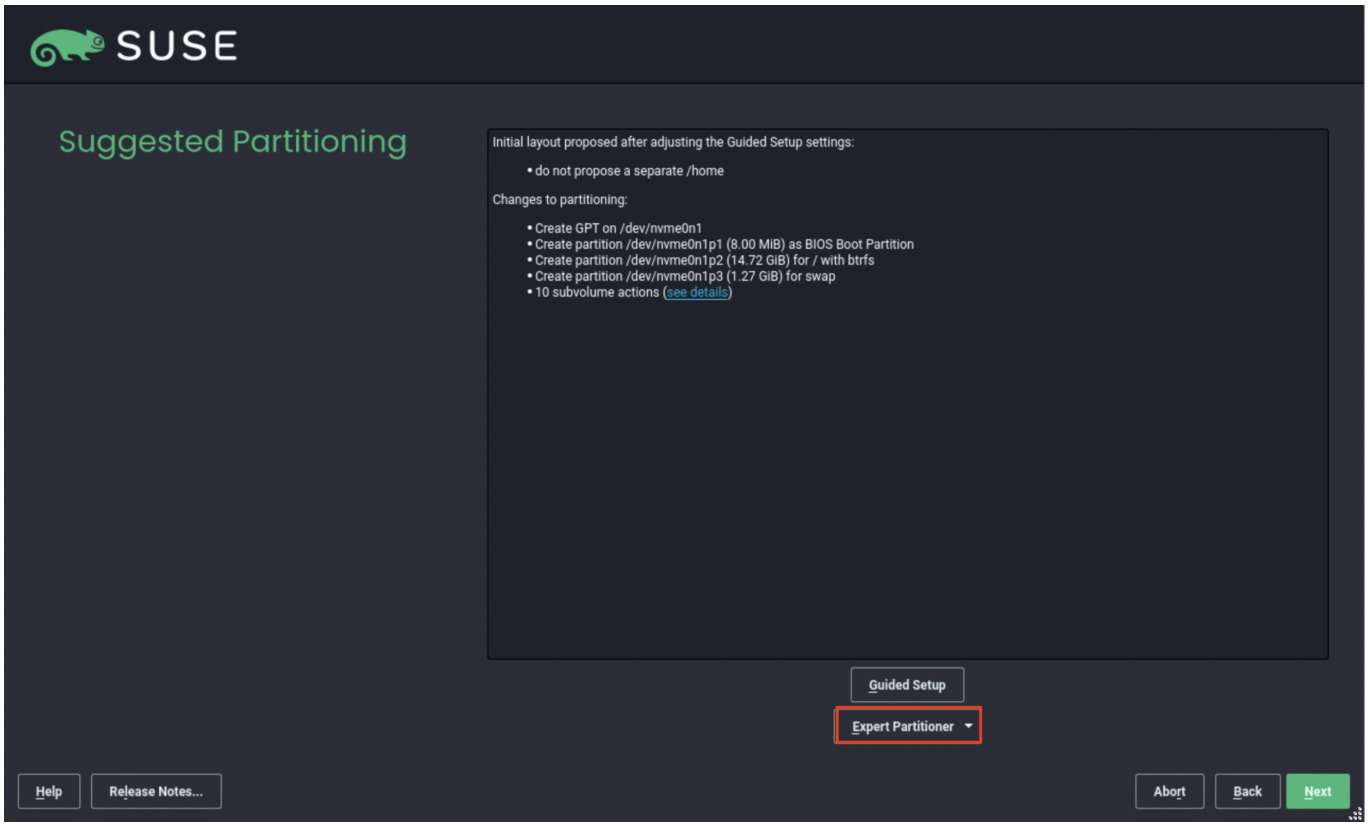


SLES 15 SP2, and SP3

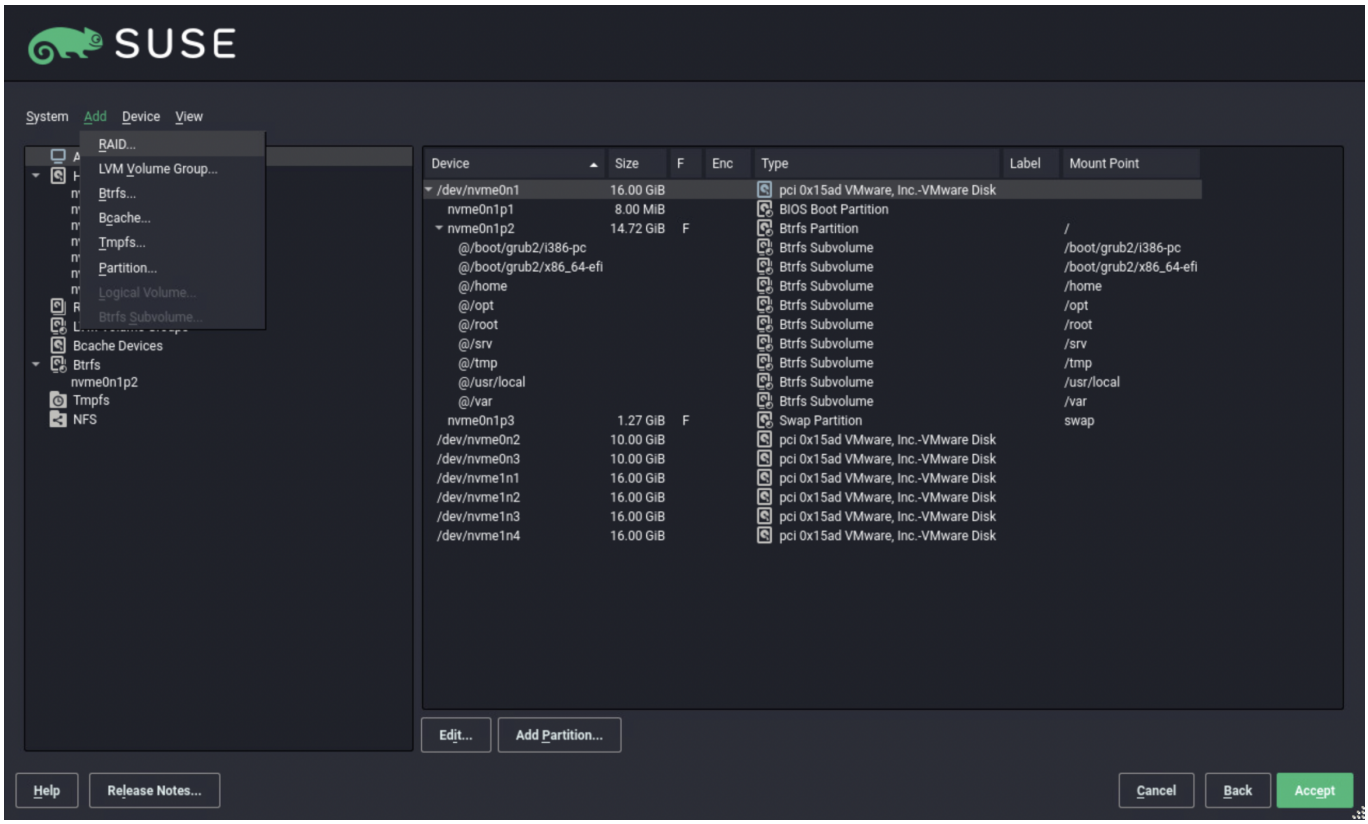
When installing SLES 15 SP2 or SP3, you must manually create RAID1 and configure the partitions.

To manually create RAID1 and configure the partitions:

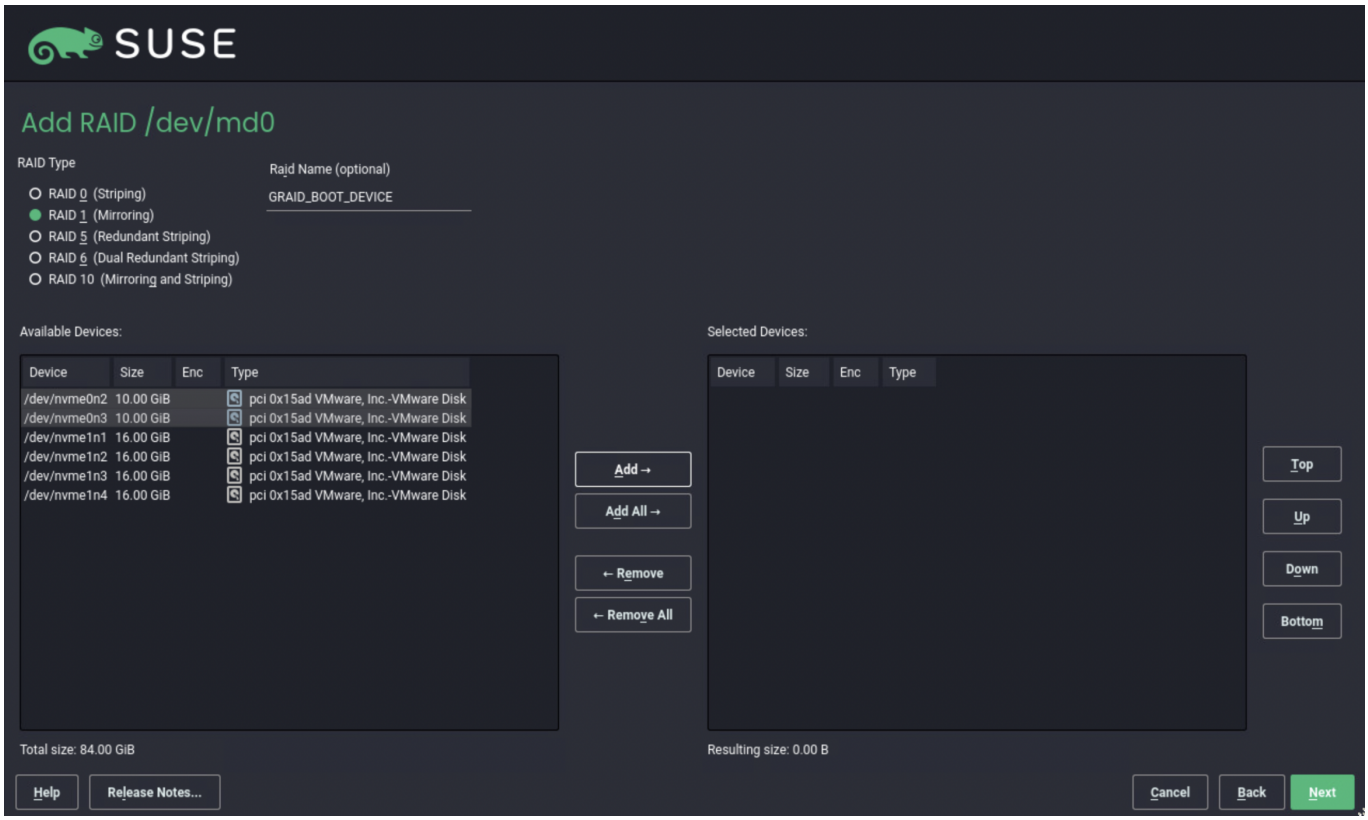
1. From the SUSE **Suggested Partitioning** page, select **Expert Partitioner** > **Next**.



2. From the SUSE **Add** menu, select **Add** > **RAID**.

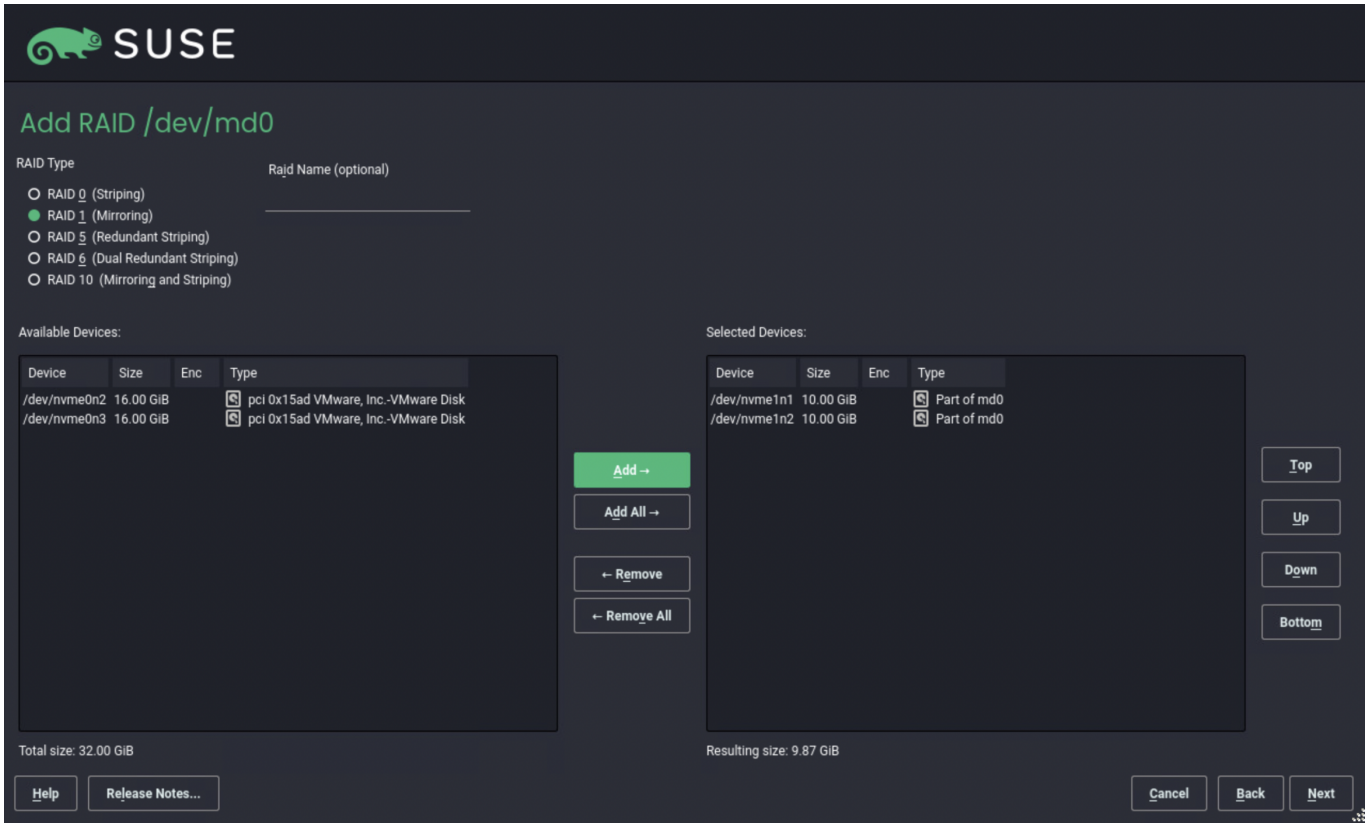


3. From the SUSE **Add RAID** page, select **RAID 1 (Mirroring)** for the **RAID Type**.



4. From the **Selected Devices** list, choose two NVMe disks and click **Add**.

5. Click **Next** to continue with the installation.



Importing and Controlling MD Bootable NVMe RAIDs using SupremeRAID™

After installing the SupremeRAID™ driver and the graidctl utility, SupremeRAID™ can import and control an MD bootable NVMe RAID. This feature makes it easier to swap drives when a bootable drive malfunctions.

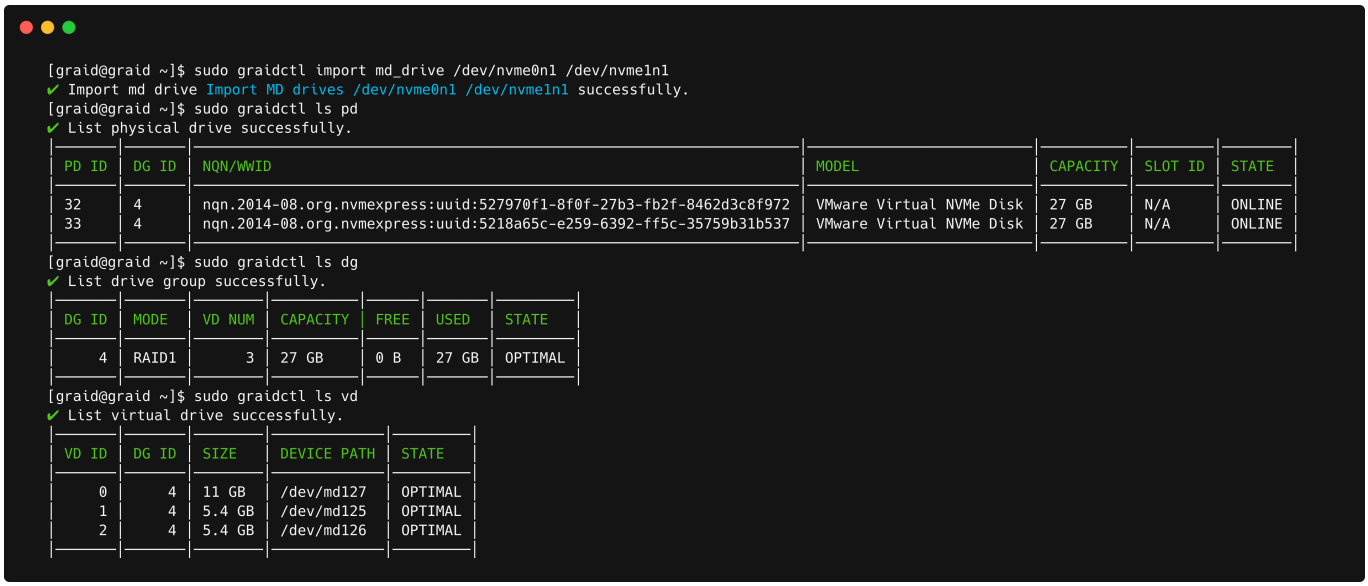
Importing an MD Bootable NVMe RAID

> **Note:**
> Only MD bootable NVMe RAID1 can be imported.

To import MD bootable NVMe RAID1 and control it using the GRAID driver:

```
$ sudo graidctl import md_drive <DEVICE_PATH_0> <DEVICE_PATH_1>
```

Output example:



Replacing an MD Bootable NVMe RAID1

> **Note:**
> Only MD bootable NVMe RAID1 can be replaced.

To replace an MD bootable NVMe RAID1:

1. Replace the old NVMe SSD with the new one. The old physical drive state should indicate **MISSING**.

   ```
   $ sudo graidctl replace md_drive <OLD_MD)PD_ID> <NEW_DEVICE_PATH>
   ```

   Output example MD missing:

```
[graid@graid ~]$ sudo graidctl ls pd
✔ List physical drive successfully.
|-------|-------|-----------------------------------------------------------------|-----------------------|----------|---------|---------|
| PD ID | DG ID | NQN/WWID                                                        | MODEL                 | CAPACITY | SLOT ID | STATE   |
|-------|-------|-----------------------------------------------------------------|-----------------------|----------|---------|---------|
| 32    | 4     | nqn.2014-08.org.nvmexpress:uuid:527970f1-8f0f-27b3-fb2f-8462d3c8f972 | VMware Virtual NVMe Disk | 27 GB    | N/A     | ONLINE  |
| 33    | 4     | nqn.2014-08.org.nvmexpress:uuid:5218a65c-e259-6392-ff5c-35759b31b537 | VMware Virtual NVMe Disk | 27 GB    | N/A     | MISSING |
|-------|-------|-----------------------------------------------------------------|-----------------------|----------|---------|---------|
[graid@graid ~]$ sudo graidctl ls dg
✔ List drive group successfully.
|-------|--------|--------|----------|------|--------|----------|
| DG ID | MODE   | VD NUM | CAPACITY | FREE | USED   | STATE    |
|-------|--------|--------|----------|------|--------|----------|
| 4     | RAID1  | 3      | 27 GB    | 0 B  | 27 GB  | DEGRADED |
|-------|--------|--------|----------|------|--------|----------|
[graid@graid ~]$ sudo graidctl ls vd
✔ List virtual drive successfully.
|-------|-------|---------|-------------|----------|
| VD ID | DG ID | SIZE    | DEVICE PATH | STATE    |
|-------|-------|---------|-------------|----------|
| 0     | 4     | 11 GB   | /dev/md127  | DEGRADED |
| 1     | 4     | 5.4 GB  | /dev/md125  | DEGRADED |
| 2     | 4     | 5.4 GB  | /dev/md126  | DEGRADED |
|-------|-------|---------|-------------|----------|
```

Output example replace drive:

```
[graid@graid ~]$ sudo graidctl replace md_drive 33 /dev/nvme2n1
✔ Replace md drive replaced PD33 with /dev/nvme2n1 successfully.
[graid@graid ~]$ sudo graidctl ls pd
✔ List physical drive successfully.
|-------|-------|-----------------------------------------------------------------|-----------------------|----------|---------|---------|
| PD ID | DG ID | NQN/WWID                                                        | MODEL                 | CAPACITY | SLOT ID | STATE   |
|-------|-------|-----------------------------------------------------------------|-----------------------|----------|---------|---------|
| 32    | 4     | nqn.2014-08.org.nvmexpress:uuid:527970f1-8f0f-27b3-fb2f-8462d3c8f972 | VMware Virtual NVMe Disk | 27 GB    | N/A     | ONLINE  |
| 33    | 4     | nqn.2014-08.org.nvmexpress:uuid:52524729-5a31-13e7-a316-f6e765e16ec8 | VMware Virtual NVMe Disk | 27 GB    | N/A     | REBUILD |
|-------|-------|-----------------------------------------------------------------|-----------------------|----------|---------|---------|
[graid@graid ~]$ sudo graidctl ls dg
✔ List drive group successfully.
|-------|--------|--------|----------|------|--------|----------|
| DG ID | MODE   | VD NUM | CAPACITY | FREE | USED   | STATE    |
|-------|--------|--------|----------|------|--------|----------|
| 4     | RAID1  | 3      | 27 GB    | 0 B  | 27 GB  | REBUILD  |
|-------|--------|--------|----------|------|--------|----------|
[graid@graid ~]$ sudo graidctl ls vd
✔ List virtual drive successfully.
|-------|-------|---------|-------------|-------------------|
| VD ID | DG ID | SIZE    | DEVICE PATH | STATE             |
|-------|-------|---------|-------------|-------------------|
| 0     | 4     | 11 GB   | /dev/md127  | REBUILD (pending) |
| 1     | 4     | 5.4 GB  | /dev/md125  | REBUILD (82.52%)  |
| 2     | 4     | 5.4 GB  | /dev/md126  | REBUILD (pending) |
|-------|-------|---------|-------------|-------------------|
```

The bootable RAID group rebuilds immediately after replacing the drive.

Dismissing an Imported MD Bootable NVMe RAID1

> **Note:**
>
> Only MD bootable NVMe RAID1 can be dismissed.

To dismiss an Imported MD Bootable NVMe RAID1:

```
$ sudo graidctl delete dg <DG_ID>
```

```
[graid@graid ~]$ sudo graidctl ls dg
✔ List drive group successfully.
|-------|--------|--------|----------|------|--------|---------|
| DG ID | MODE   | VD NUM | CAPACITY | FREE | USED   | STATE   |
|-------|--------|--------|----------|------|--------|---------|
| 4     | RAID1  | 3      | 27 GB    | 0 B  | 27 GB  | OPTIMAL |
|-------|--------|--------|----------|------|--------|---------|
[graid@graid ~]$ sudo graidctl delete dg 4
✔ Delete drive group successfully.
```

Compatible NVMe Drives List

The following NVMe drives passed GRAID qualification and can be used with GRAID SupremeRAID™.

GRAID updates this list when new NVMe drives pass the qualification process.

Manufacturer	Series	Interface	FormFactor
Dapustor	H3200	PCIe Gen3 x 4	2.5 inch U.2
Intel	DC P4510	PCIe Gen3 x 4	2.5 inch U.2
Intel	DC P4610	PCIe Gen3 x 4	2.5 inch U.2
Intel	D7-P5316	PCIe Gen4 x 4	2.5 inch U.2
Intel	D7-P5510	PCIe Gen4 x 4	2.5 inch U.2
Intel	Optane™ P5800X	PCIe Gen4 x 4	2.5 inch U.2
Kioxia	CD5	PCIe Gen3 x 4	2.5 inch U.2
Kioxia	CD6	PCIe Gen4 x 4	2.5 inch U.3
Kioxia	CM6	PCIe Gen4 x 4	2.5 inch U.3
Memblaze	P6536	PCIe Gen4 x 4	2.5 inch U.2
Micron	9300	PCIe Gen3 x 4	2.5 inch U.2
Micron	7300	PCIe Gen4 x 4	2.5 inch U.2
Micron	7400	PCIe Gen4 x 4	2.5 inch U.3
Netlist	N1951	PCIe Gen3 x 4	2.5 inch U.2
Samsung	PM963	PCIe Gen3 x 4	2.5 inch U.2
Samsung	PM983	PCIe Gen3 x 4	2.5 inch U.2
Samsung	PM9A3	PCIe Gen4 x 4	2.5 inch U.2
Samsung	PM1733	PCIe Gen4 x 4	2.5 inch U.2
Western Digital Technologies	SN640	PCIe Gen3 x 4	2.5 inch U.2

For the latest information, see the Compatible NVMe Drives List on the GRAID website.