

How to Install the NVMe Host Module Patch on Compute Nodes under CentOS

Provides the steps to installing the NVMe host module patch on compute nodes using

The steps in this section apply to CentOS 7, Kernel 5.x and must be completed on each compute node. They have been tested on CentOS 7.9 and CentOS 8.3.

Step 1. Compile the NVMe host kernel modules

To compile the NVMe host kernel modules, complete these following:

a) Install common development tools and other required components:

```
yum -y groupinstall "Development Tools"

yum -y install ncurses-devel hmaccalc zlib-devel binutils-devel elfutils-libelf-devel
```

b) If you are running CentOS 7, install these additional tools:

```
yum -y install centos-release-scl

yum -y install devtoolset-8-gcc

scl enable devtoolset-8 -- bash

wget https://www.openssl.org/source/openssl-1.1.0i.tar.gz

tar -zxf openssl-1.1.0i.tar.gz

cd openssl-1.1.0i

./config

make

make install

cp /usr/local/lib64/libcrypto.so.1.1 /lib64

cd ..
```

c) Install kernel source and development packages for the target kernel (e.g. kernel 5.10.61):

```
rpm -i kernel-devel-5.10.61-1.x86_64.rpm

rpm -i kernel-headers-5.10.61-1.x86_64.rpm

rpm -i kernel-5.10.61-1.src.rpm

tar xzf /root/rpmbuild/SOURCES/kernel-5.10.61.tar.gz
```

d) Apply the patch received from Kioxia:

```
SRC_DIR=/root/rpmbuild/SOURCES/kernel-5.10.61/drivers/nvme/host

PATH_TO_PATCH=<path-to-the-patch-dir>/nvmf_host_fast_io_fail_patch_5.10.61-1.1-1.patch

cd ${SRC_DIR}

patch --dry-run -p1 -i ${PATH_TO_PATCH}

patch -p1 -i ${PATH_TO_PATCH}
```

e) Define **MODULE_DESCRIPTION** to identify the applied patch:

```
BUILD_TIME=$(date +%y%m%d%H%M)

DESCRIPTION="\nvme host Kioxia patch (${BUILD_TIME})\"

sed -i "/MODULE_DESCRIPTION.*/d" core.c

echo -e "\nMODULE_DESCRIPTION(${DESCRIPTION});" >> core.c
```

f) Build the patched kernel modules.

```
KDIR=/lib/modules/5.10.61/build/

make -C ${KDIR} M=${SRC_DIR}
```

Note: If the make fails, try to upgrade the GNU Compiler Collection (GCC) as shown below:

```
wget https://ftp.mirrorservice.org/sites/sourceware.org/pub/gcc/releases/gcc-8.5.0/gcc-8.5.0.tar.gz

tar xzf gcc-8.5.0.tar.gz

cd gcc-8.5.0

yum -y install bzip2

yum install libmpc-devel./configure --disable-multilib --enable-languages=c,c++

make -j 4

make install
```

g) Sign the modules

```
cd ${SRC_DIR}
KOS=$(ls *.ko)
for module in ${KOS[*]}; do echo "sign ${module}"; strip --strip-unneeded ${module}; ${KDIR}/scripts/sign-file sha256
```

NOTE: If the signing fails, then generate keys for using it with the kernel module you want to sign and copy them to, /usr/src/kernels/\${uname -r}/certs/:

```
openssl req -new -x509 -newkey rsa:2048 -keyout signing_key.pem -outform DER -out signing_key.x509 -nodes -subj /CN=kernel
cp signing_key.x509 signing_key.pem /usr/src/kernels/${uname -r}/certs/
```

h) Confirm the five kernel modules listed below are produced:

- nvme-core.ko
- nvme.ko
- nvme-fabrics.ko
- nvme-tcp.ko
- nvme-rdma.ko

Step 2. Install the patched modules

a) Assuming the patched modules of Step 1 are placed in the current directory, verify that their version matches currently running kernel version:

```
modinfo *.ko | grep vermagic | awk '{print $2}'

uname -r
```

b) Define bash variables:

```
KVER=$(uname -r)

SAVED_DIR="/root/tmp/orig_$(uname -r)"

MODULES_DIR="/lib/modules/$KVER/kernel/drivers/nvme/host"
```

c) Create a backup of the **initramfs** image:

```
cp /boot/initramfs-"$KVER".img /boot/saved-initramfs-"$KVER".img
```

d) Backup the original modules and replace them with the patched version:

```
mkdir -p ${SAVED_DIR}

cd ${MODULES_DIR}

tar -cvf ${SAVED_DIR}/"$KVER"_kos.tar ./*.ko*

rm -f ${MODULES_DIR}/*.ko*

cp -f ${SRC_DIR}/*.ko ${MODULES_DIR}/
```

e) Regenerate dependencies and **initrd**:

```
/sbin/depmod -ae -F /boot/System.map-$(uname -r)

dracut --force -H
```

f) If nvme (pci) module cannot be unloaded, reboot the machine:

```
reboot
```

Otherwise, unload the original modules and load the patched version:

- ```
modprobe -r nvme-tcp

modprobe -r nvme-rdma

modprobe -r nvme-fc
```

modprobe -r nvme-fabrics  
modprobe -r nvme  
modprobe -r nvme-core  
modprobe nvme-core  
modprobe nvme  
modprobe nvme-fabrics  
modprobe nvme-tcp  
modprobe nvme-rdma

Step 3. Install the NVMe patch

To install the NVMe patch, complete these following:

a) Confirm you have disabled Secure Boot from your firmware (BIOS). Note that you will of course lose the protection provided by UEFI Secure Boot.

b) Install the rpm of the NVMe host patch:

```
rpm -i nvme_host_fast_io_fail_patch_4.18.0-305.7.1.el8_4-1.0-1.x86_64.rpm
```

c) Verify that patch has been installed by running:

```
modinfo nvme-core|grep description
```

It should return:

```
nvme host Kioxia patch (nvme-host-fast-io-fail-patch-5.4.0-91-generic-1.0-0-all), then patch is installed,
```

You are now ready to use the compute nodes with KumoScale. If you need to uninstall the NVMe patch and restore the original kernel modules, see [How to uninstall the NVMe patch and restore original kernel modules.](#)