Deploying BlueField Software Using BFB from Host

Table of contents

ⓘ **Info**

It is recommended to upgrade your BlueField product to the latest software and firmware versions available to benefit from new features and latest bug fixes.

ⓘ **Note**

This procedure assumes that a NVIDIA® BlueField® networking platform (DPU or SuperNIC) has already been installed in a server according to the instructions detailed in the BlueField device's hardware user guide.

The following table lists an overview of the steps required to install Ubuntu BFB on your BlueField:

Step	Procedure	Link to Section
1	Uninstall previous DOCA on host (if exists)	Uninstall Previous Software from Host
2	Install RShim on the host	Install RShim on Host
3	Verify that RShim is running on the host	Ensure RShim Running on Host
4	Install the Ubuntu BFB image	BFB Installation
5	Verify installation completed successfully	Verify BFB is Installed
6	Upgrade the firmware on your BlueField	Firmware Upgrade

Uninstall Previous Software from Host

If an older DOCA software version is installed on your host, make sure to uninstall it before proceeding with the installation of the new version:

Ubuntu	host# for f in $(dpkg --list | grep doca | awk '{print $2}'); do echo $f ; apt remove --purge $f -y ; done host# sudo apt-get autoremove
CentOS/RHEL	host# for f in $(rpm -qa |grep -i doca) ; do yum -y remove $f; done host# yum autoremove host# yum makecache

Install RShim on Host

Before installing the RShim driver, verify that the RShim devices, which will be probed by the driver, are listed under lsusb or lspci.

```
lspci | grep -i nox
```

Output example:

```
27:00.0 Ethernet controller: Mellanox Technologies MT42822 BlueField-2 integrated ConnectX-6 Dx network controller
27:00.1 Ethernet controller: Mellanox Technologies MT42822 BlueField-2 integrated ConnectX-6 Dx network controller
27:00.2 Non-Volatile memory controller: Mellanox Technologies NVMe SNAP Controller
27:00.3 DMA controller: Mellanox Technologies MT42822 BlueField-2 SoC Management Interface // This is the RShim PF
```

RShim is compiled as part of the doca-runtimepackage in the doca-host-repo-ubuntu<version>_amd64 file (.deb or .rpm).

To install doca-runtime:

OS	Procedure

Ubuntu/Debian	1. Download the DOCA Runtime host package from the "Installation Files" section in the *NVIDIA DOCA Installation Guide for Linux*.
	2. Unpack the deb repo. Run:
	`host# sudo dpkg -i doca-host-repo-ubuntu<version>_amd64.deb`
	3. Perform apt update. Run:
	`host# sudo apt-get update`
	4. Run `apt install` for DOCA runtime package.
	`host# sudo apt install doca-runtime`

CentOS/RHEL 7.x	1. Download the DOCA runtime host package from the "Installation Files" section in the *NVIDIA DOCA Installation Guide for Linux*.
	2. Unpack the RPM repo. Run:
	`host# sudo rpm -Uvh doca-host-repo-rhel<version>.x86_64.rpm`
	3. Enable new yum repos. Run:
	`host# sudo yum makecache`
	4. Run `yum install` to install DOCA runtime package.
	`host# sudo yum install doca-runtime`

CentOS/RHEL 8.x or Rocky 8.6	1. Download the DOCA runtime host package from the "Installation Files" section in the *NVIDIA DOCA Installation Guide for Linux*. 2. Unpack the RPM repo. Run: `host# sudo rpm -Uvh doca-host-repo-rhel<version>.x86_64.rpm` 3. Enable new dnf repos. Run:

OS	Procedure

	`host# sudo dnf makecache`
	4. Run dnf install to install DOCA runtime.
	`host# sudo dnf install doca-runtime`

Ensure RShim Running on Host

1. Verify RShim status. Run:

   ```
   sudo systemctl status rshim
   ```

   Expected output:

   ```
   active (running)
   ...
   Probing pcie-0000:<BlueField's PCIe Bus address on host>
   create rshim pcie-0000:<BlueField's PCIe Bus address on host>
   rshim<N> attached
   ```

   Where <N> denotes RShim enumeration starting with 0 (then 1, 2, etc.) for every additional BlueField installed on the server.

   If the text "another backend already attached" is displayed, users will not be able to use RShim on the host. Please refer to "RShim Troubleshooting and How-Tos" to troubleshoot RShim issues.

   1. If the previous command displays inactive or another error, restart RShim service. Run:

```
sudo systemctl restart rshim
```

    2. Verify RShim status again. Run:

```
sudo systemctl status rshim
```

If this command does not display "active (running)", then refer to "RShim Troubleshooting and How-Tos".

2. Display the current setting. Run:

```
# cat /dev/rshim<N>/misc | grep DEV_NAME
DEV_NAME        pcie-0000:04:00.2
```

This output indicates that the RShim service is ready to use.

Installing Ubuntu on BlueField

BFB Installation

Note

Check the BFB version installed on your BlueField-2. If the version is 1.5.0 or lower, please see known issue #3600716 under Known Issues section.

Info

To upgrade the BMC firmware using BFB, the user must provide the current BMC credentials in the bf.cfg. Refer to "Customizing BlueField Software Deployment Using bf.cfg" for more information.

ⓘ **Note**

Upgrading the BlueField networking platform using BFB Bundle updates the NIC firmware by default. NIC firmware upgrade triggers a NIC reset flow via `mlxfwreset` in the BlueField Arm.

If this reset flow cannot complete or is not supported on your setup, `bfb-install` alerts about it at the end of the installation. In this case, perform a BlueField system-level reset.

To skip NIC firmware upgrade during BFB Bundle installation , provide the parameter `WITH_NIC_FW_UPDATE=no` in the `bf.cfg` text file when running `bfb-install` .

A pre-built BFB of Ubuntu 22.04 with DOCA Runtime and DOCA packages installed is available on the NVIDIA DOCA SDK developer zone page.

ⓘ **Note**

All new BlueField-2 devices and all BlueField-3 are secure boot enabled, hence all the relevant SW images (ATF/UEFI, Linux Kernel and Drivers) must be signed in order to boot. All formally published SW images are signed.

⚠ **Warning**

When installing the BFB bundle in NIC mode, users must perform the following:

1. Prior to installing the BFB bundle, users must unbind each NIC port, using its PCIe function address. For example:

   ```
   [host]# lspci -d 15b3:
   21:00.0 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
   integrated ConnectX-7 network controller (rev 01)
   21:00.1 Ethernet controller: Mellanox Technologies MT43244 BlueField-3
   integrated ConnectX-7 network controller (rev 01)
   21:00.2 DMA controller: Mellanox Technologies MT43244 BlueField-3
   SoC Management Interface (rev 01)

   [host]# echo 0000:21:00.0 > /sys/bus/pci/drivers/mlx5_core/unbind
   [host]# echo 0000:21:00.1 > /sys/bus/pci/drivers/mlx5_core/unbind
   ```

   If there are multiple BlueField devices to be updated in the server, repeat this step on all of them, before starting BFB bundle installations.

2. After the BFB bundle installation is done, users must perform a warm reboot on the host.

To install Ubuntu BFB, run on the host side:

```
# bfb-install -h
syntax: bfb-install --bfb|-b <BFBFILE> [--config|-c <bf.cfg>] \
  [--rootfs|-f <rootfs.tar.xz>] --rshim|-r <rshimN> [--help|-h]
```

The bfb-install utility is installed by the RShim package.

This utility script pushes the BFB image and optional configuration (bf.cfg file) to the BlueField side and checks and prints the BFB installation progress. To see the BFB installation progress, please install the pv Linux tool.

⚠ **Warning**

BFB image installation must complete before restarting the system/BlueField. Doing so may result in anomalous behavior of the BlueField (e.g., it may not be accessible using SSH). If this happens, re-initiate the update process with bfb-install to recover the BlueField.

The following is an output example of Ubuntu 20.04 installation with the bfb-install script assuming pv has been installed.

```
# bfb-install --bfb <BlueField-BSP>.bfb --config bf.cfg --rshim rshim0 Pushing bfb + cfg
1.46GiB 0:01:11 [20.9MiB/s] [                                              <=>                      ]
Collecting BlueField booting status. Press Ctrl+C to stop…
 INFO[PSC]: PSC BL1 START
 INFO[BL2]: start
 INFO[BL2]: boot mode (rshim)
 INFO[BL2]: VDDQ: 1120 mV
 INFO[BL2]: DDR POST passed
 INFO[BL2]: UEFI loaded
 INFO[BL31]: start
 INFO[BL31]: lifecycle Production
 INFO[BL31]: MB8: VDD adjustment complete
 INFO[BL31]: VDD: 743 mV
 INFO[BL31]: power capping disabled
 INFO[BL31]: runtime
 INFO[UEFI]: eMMC init
 INFO[UEFI]: eMMC probed
 INFO[UEFI]: UPVS valid
 INFO[UEFI]: PMI: updates started
 INFO[UEFI]: PMI: total updates: 1
 INFO[UEFI]: PMI: updates completed, status 0
 INFO[UEFI]: PCIe enum start
 INFO[UEFI]: PCIe enum end
 INFO[UEFI]: UEFI Secure Boot (disabled)
 INFO[UEFI]: exit Boot Service
 INFO[MISC]: : Found bf.cfg
 INFO[MISC]: : Ubuntu installation started
 INFO[MISC]: bfb_pre_install
```

```
INFO[MISC]: Installing OS image
INFO[MISC]: : Changing the default password for user ubuntu
INFO[MISC]: : Running bfb_modify_os from bf.cfg
INFO[MISC]: : Ubuntu installation finished
```

Verify BFB is Installed

After installation of the Ubuntu OS is complete, the following note appears in /dev/rshim0/misc on first boot:

```
...
INFO[MISC]: Linux up
INFO[MISC]: DPU is ready
```

"DPU is ready" indicates that all the relevant services are up and users can login the system.

After the installation of the Ubuntu 20.04 BFB, the configuration detailed in the following sections is generated.

Note

Make sure all the services (including cloud-init) are started on BlueField and to perform a graceful shutdown before power cycling the host server.

BlueField OS image version is stored under /etc/mlnx-release in the BlueField:

```
# cat /etc/mlnx-release
bf-bundle-2.7.0-<version>_ubuntu-22.04_prod
```

Firmware Upgrade

To upgrade firmware:

1. Access the BlueField using one of the available interfaces (RShim console, BMC console, SSH via oob_net0 or tmfifo_net0 interfaces).
2. Upgrade the firmware on BlueField. Run:

   ```
   sudo /opt/mellanox/mlnx-fw-updater/mlnx_fw_updater.pl --force-fw-update
   ```

   Example output:

   ```
   Device #1:
   ----------

     Device Type:      BlueField-2
     [...]
     Versions:         Current        Available
        FW             <Old_FW>       <New_FW>
   ```

   > **Note**
   >
   > **Important!** To apply NVConfig changes, stop here and follow the steps in section "Updating NVConfig Params". In this case, the following step #3 is redundant.

3. Perform a BlueField system reboot for the upgrade to take effect.

Updating NVConfig Params from Host

1. Optional. To reset the BlueField NIC firmware configuration (aka Nvconfig params) to their factory default values, run the following from the BlueField ARM OS or from the host OS:

```
# sudo mlxconfig -d /dev/mst/<MST device> -y reset

Reset configuration for device /dev/mst/<MST device>? (y/n) [n] : y
Applying... Done!
-I- Please reboot machine to load new configurations.
```

> **Note**
>
> For now, please ignore tool's instruction to reboot

> **Note**
>
> To learn what MST device the BlueField has on your setup, run:
>
> ```
> mst start
> mst status
> ```
>
> Example output taken on a multiple BlueField host:

```
// The MST device corresponds with PCI Bus address.

MST modules:
------------
    MST PCI module is not loaded
    MST PCI configuration module loaded

MST devices:
------------
/dev/mst/mt41692_pciconf0         - PCI configuration cycles access.
                                   domain:bus:dev.fn=0000:03:00.0 addr.reg=88
data.reg=92 cr_bar.gw_offset=-1
                                   Chip revision is: 01
/dev/mst/mt41692_pciconf1         - PCI configuration cycles access.
                                   domain:bus:dev.fn=0000:83:00.0 addr.reg=88
data.reg=92 cr_bar.gw_offset=-1
                                   Chip revision is: 01
/dev/mst/mt41686_pciconf0         - PCI configuration cycles access.
                                   domain:bus:dev.fn=0000:a3:00.0 addr.reg=88
data.reg=92 cr_bar.gw_offset=-1
                                   Chip revision is: 01
```

The MST device IDs for the BlueField-2 and BlueField-3 devices in this example are /dev/mst/mt41686_pciconf0 and /dev/mst/mt41692_pciconf0 respectively.

2. (Optional) Enable NVMe emulation. Run:

```
sudo mlxconfig -d <MST device> -y s NVME_EMULATION_ENABLE=1
```

3. Skip this step if your BlueField is Ethernet only. Please refer to section "Supported Platforms and Interoperability" under the Release Notes to learn your BlueField type.

   If you have an InfiniBand-and-Ethernet-capable BlueField, the default link type of the ports will be configured to IB. If you want to change the link type to Ethernet, please run the following configuration:

```
sudo mlxconfig -d <MST device> -y s LINK_TYPE_P1=2 LINK_TYPE_P2=2
```

4. Perform a BlueField system-level reset for the new settings to take effect.

> **Note**
>
> After modifying files on the BlueField, run the command `sync` to flush file system buffers to eMMC/SSD flash memory to avoid data loss during reboot or power cycle.

Default Ports and OVS Configuration

The /sbin/mlnx_bf_configure script runs automatically with ib_umad kernel module loaded (see /etc/modprobe.d/mlnx-bf.conf) and performs the following configurations:

1. Ports are configured with switchdev mode and software steering.
2. RDMA device isolation in network namespace is enabled.
3. Two scalable function (SF) interfaces are created (one per port) if BlueField is configured with Embedded CPU mode (default):

```
# mlnx-sf -a show

SF Index: pci/0000:03:00.0/229408
  Parent PCI dev: 0000:03:00.0
  Representor netdev: en3f0pf0sf0
  Function HWADDR: 02:a9:49:7e:34:29
  Function trust: off
  Function roce: true
  Function eswitch: NA
  Auxiliary device: mlx5_core.sf.2
    netdev: enp3s0f0s0
    RDMA dev: mlx5_2

SF Index: pci/0000:03:00.1/294944
  Parent PCI dev: 0000:03:00.1
  Representor netdev: en3f1pf1sf0
  Function HWADDR: 02:53:8f:2c:8a:76
  Function trust: off
  Function roce: true
  Function eswitch: NA
  Auxiliary device: mlx5_core.sf.3
    netdev: enp3s0f1s0
    RDMA dev: mlx5_3
```

The parameters for these SFs are defined in configuration file /etc/mellanox/mlnx-sf.conf.

```
/sbin/mlnx-sf --action create --device 0000:03:00.0 --sfnum 0 --hwaddr 02:61:f6:21:32:8c
/sbin/mlnx-sf --action create --device 0000:03:00.1 --sfnum 0 --hwaddr 02:30:13:6a:2d:2c
```

> ⓘ **Note**
>
> To avoid repeating a MAC address in the your network, the SF MAC address is set randomly upon BFB installation. You may choose to configure a different MAC address that better suit your network needs.

4. Two OVS bridges are created:

```
# ovs-vsctl show
f08652a8-92bf-4000-ba0b-7996c772aff6
    Bridge ovsbr2
        Port ovsbr2
            Interface ovsbr2
                type: internal
        Port p1
```

```
            Interface p1
        Port en3f1pf1sf0
            Interface en3f1pf1sf0
        Port pf1hpf
            Interface pf1hpf
    Bridge ovsbr1
        Port p0
            Interface p0
        Port pf0hpf
            Interface pf0hpf
        Port ovsbr1
            Interface ovsbr1
                type: internal
        Port en3f0pf0sf0
            Interface en3f0pf0sf0
    ovs_version: "2.14.1"
```

The parameters for these bridges are defined in configuration file /etc/mellanox/mlnx-ovs.conf:

```
CREATE_OVS_BRIDGES="yes"
OVS_BRIDGE1="ovsbr1"
OVS_BRIDGE1_PORTS="p0 pf0hpf en3f0pf0sf0"
OVS_BRIDGE2="ovsbr2"
OVS_BRIDGE2_PORTS="p1 pf1hpf en3f1pf1sf0"
OVS_HW_OFFLOAD="yes"
OVS_START_TIMEOUT=30
```

> **Note**
>
> If failures occur in `/sbin/mlnx_bf_configure` or configuration changes happen (e.g. switching to separated host mode) OVS bridges are not created even if `CREATE_OVS_BRIDGES="yes"`.

5. OVS HW offload is configured.

Default Network Interface Configuration

Network interfaces are configured using the netplan utility:

```
# cat /etc/netplan/50-cloud-init.yaml
# This file is generated from information provided by the datasource.  Changes
# to it will not persist across an instance reboot.  To disable cloud-init's
# network configuration capabilities, write a file
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
  ethernets:
    tmfifo_net0:
      addresses:
      - 192.168.100.2/30
      dhcp4: false
      nameservers:
        addresses:
        - 192.168.100.1
      routes:
      -   metric: 1025
        to: 0.0.0.0/0
        via: 192.168.100.1
    oob_net0:
      dhcp4: true
  renderer: NetworkManager
  version: 2

# cat /etc/netplan/60-mlnx.yaml
network:
 ethernets:
  enp3s0f0s0:
   dhcp4: 'true'
  enp3s0f1s0:
   dhcp4: 'true'
 renderer: networkd
 version: 2
```

BlueField devices also have a local IPv6 (LLv6) derived from the MAC address via the STD stack mechanism. For a default MAC, 00:1A:CA:FF:FF:01, the LLv6 address would be fe80::21a:caff:feff:ff01.

For multi-device support, the LLv6 address works with SSH for any number of BlueField devices in the same host by including the interface name in the SSH command:

```
host]# systemctl restart rshim
// wait 10 seconds
host]# ssh -6 ubuntu@fe80::21a:caff:feff:ff01%tmfifo_net<n>
```

> **Note**
>
> If `tmfifo_net<n>` on the host does not have an LLv6 address, restart the RShim driver:
>
> ```
> systemctl restart rshim
> ```

Ubuntu Boot Time Optimizations

To improve the boot time, the following optimizations were made to Ubuntu OS image:

```
# cat /etc/systemd/system/systemd-networkd-wait-online.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/bin/nm-online -s -q --timeout=5

# cat /etc/systemd/system/NetworkManager-wait-online.service.d/override.conf
[Service]
ExecStart=
ExecStart=/usr/lib/systemd/systemd-networkd-wait-online --timeout=5

# cat /etc/systemd/system/networking.service.d/override.conf
[Service]
TimeoutStartSec=5
ExecStop=
```

```
ExecStop=/sbin/ifdown -a --read-environment --exclude=lo --force --ignore-errors
```

This configuration may affect network interface configuration if DHCP is used. If a network device fails to get configuration from the DHCP server, then the timeout value in the two files above must be increased.

Grub Configuration:

Setting the Grub timeout at 2 seconds with `GRUB_TIMEOUT=2` under `/etc/default/grub`. In conjunction with the `GRUB_TIMEOUT_STYLE=countdown` parameter, Grub will show the countdown of 2 seconds in the console before booting Ubuntu. Please note that, with this short timeout, the standard Grub method for entering the Grub menu (i.e., SHIFT or Esc) does not work. Function key F4 can be used to enter the Grub menu.

System Services:

docker.service is disabled in the default Ubuntu OS image as it dramatically affects boot time.

The `kexec` utility can be used to reduce the reboot time. Script `/usr/sbin/kexec_reboot` is included in the default Ubuntu 20.04 OS image to run corresponding `kexec` commands.

```
# kexec_reboot
```

DHCP Client Configuration

```
/etc/dhcp/dhclient.conf:
send vendor-class-identifier "NVIDIA/BF/DP";
interface "oob_net0" {
  send vendor-class-identifier "NVIDIA/BF/OOB";
      }
```

Ubuntu Dual Boot Support

BlueField may be installed with support for dual boot. That is, two identical images of the BlueField OS may be installed using BFB.

The following is a proposed SSD partitioning layout for 119.24 GB SSD:

```
Device              Start       End   Sectors  Size Type
/dev/nvme0n1p1       2048    104447    102400   50M EFI System
/dev/nvme0n1p2     104448 114550086 114445639 54.6G Linux filesystem
/dev/nvme0n1p3 114550087 114652486    102400   50M EFI System
/dev/nvme0n1p4 114652487 229098125 114445639 54.6G Linux filesystem
/dev/nvme0n1p5 229098126 250069645  20971520   10G Linux filesystem
```

Where:

- `/dev/nvme0n1p1` – boot EFI partition for the first OS image
- `/dev/nvme0n1p2` – root FS partition for the first OS image
- `/dev/nvme0n1p3` – boot EFI partition for the second OS image
- `/dev/nvme0n1p4` – root FS partition for the second OS image
- `/dev/nvme0n1p5` – common partition for both OS images

For example, the following is a proposed eMMC partitioning layout for a 64GB eMMC:

```
Device              Start       End  Sectors  Size Type
/dev/mmcblk0p1       2048    104447   102400   50M EFI System
/dev/mmcblk0p2     104448  50660334 50555887 24.1G Linux filesystem
/dev/mmcblk0p3  50660335  50762734   102400   50M EFI System
/dev/mmcblk0p4  50762735 101318621 50555887 24.1G Linux filesystem
/dev/mmcblk0p5 101318622 122290141 20971520   10G Linux filesystem
```

Where:

- `/dev/mmcblk0p1` – boot EFI partition for the first OS image
- `/dev/mmcblk0p2` – root FS partition for the first OS image
- `/dev/mmcblk0p3` – boot EFI partition for the second OS image
- `/dev/mmcblk0p4` – root FS partition for the second OS image

- /dev/mmcblk0p5 – common partition for both OS images

> **Note**
>
> The common partition can be used to store BFB files that will be used for OS image update on the non-active OS partition.

Installing Ubuntu OS Image Using Dual Boot

> **Note**
>
> For software upgrade procedure, please refer to section "Upgrading Ubuntu OS Image Using Dual Boot".

Add the values below to the bf.cfg configuration file (see section "bf.cfg Parameters" for more information).

```
DUAL_BOOT=yes
```

If the eMMC size is ≤16GB, dual boot support is disabled by default, but it can be forced by setting the following parameter in bf.cfg:

```
FORCE_DUAL_BOOT=yes
```

To modify the default size of the /common partition, add the following parameter:

```
COMMON_SIZE_SECTORS=<number-of-sectors>
```

The number of sectors is the size in bytes divided by the block size (512). For example, for 10GB, the COMMON_SIZE_SECTORS=$((10*2**30/512)).

After assigning size for the /common partition, what remains is divided equally between the two OS images.

```
# bfb-install --bfb <BFB> --config bf.cfg --rshim rshim0
```

This will result in the Ubuntu OS image to be installed twice on the BlueField.

> **Note**
>
> For comprehensive list of the supported parameters to customize bf.cfg during BFB installation, refer to section "bf.cfg Parameters".

Upgrading Ubuntu OS Image Using Dual Boot

1. Download the new BFB to the BlueField into the /common partition. Use bfb_tool.py script to install the new BFB on the inactive BlueField partition:

   ```
   /opt/mellanox/mlnx_snap/exec_files/bfb_tool.py --op fw_activate_bfb --bfb <BFB>
   ```

2. Reset BlueField to load the new OS image:

   ```
   /sbin/shutdown -r 0
   ```

   BlueField should now boot into the new OS image.

Use efibootmgr utility to manage the boot order if necessary.

- Change the boot order with:

```
# efibootmgr -o
```

- Remove stale boot entries with:

```
# efibootmgr -b <E> -B
```

  Where `<E>` is the last character of the boot entry (i.e., `Boot000<E>`). You can find that by running:

```
# efibootmgr
BootCurrent: 0040
Timeout: 3 seconds
BootOrder: 0040,0000,0001,0002,0003
Boot0000* NET-NIC_P0-IPV4
Boot0001* NET-NIC_P0-IPV6
Boot0002* NET-NIC_P1-IPV4
Boot0003* NET-NIC_P1-IPV6
Boot0040* focal0
....2
```

Note

Modifying the boot order with `efibootmgr -o` does not remove unused boot options. For example, changing a boot order from 0001,0002, 0003 to just 0001 does not actually remove 0002 and 0003. 0002 and 0003 need to be explicitly removed using `efibootmgr -B` .