



Deploying BlueField Software Using PXE

Table of contents

Deploying BlueField Software Using BFB with PXE	9
Deploying BlueField Software Using ISO with PXE	13

The BlueField boot flow is similar to a standard server, allowing to boot via an external PXE/HTTP server.

Providing a BFB or ISO image via PXE server enables updating the BlueField software.

The update flow for the two software update image types, [BFB](#) and [ISO](#), is slightly different as elaborated on in the following subpages.

BFB Update vs. ISO Update

- PXE booting the BlueField device with a BFB file updates all components included in the BFB image at the end of the automated update process

Note

The BFB image is available in two formats:

- BF-Bundle – includes BlueField firmware, BlueField Arm OS, and DOCA
- BF-FW-Bundle – includes BlueField firmware only

- PXE booting the BlueField device with the ISO image provides the same result as BF-Bundle installation using standard installation method for Ubuntu OS

Note

Make sure that the BlueField's firmware version and the DOCA version running on BlueField Arm cores belong to the same release.

Setting BlueField to PXE Boot

Users may set the BlueField UEFI for PXE boot using any of the following methods:

- Direct access to BlueField UEFI menu (see section "[Setting Next-boot Device in BlueField UEFI Menu](#)")
- Out-of-band, using the Redfish interface over the BlueField BMC management LAN
- Using `efibootmgr` in the BlueField Linux OS

Regardless of the method, the following is a list of PXE devices in the BlueField UEFI which are compatible with the software upgrade procedure described in this chapter:

- `NET-NIC_P0-IPV4`
- `NET-NIC_P0-IPV6`
- `NET-NIC_P1-IPV4`
- `NET-NIC_P1-IPV6`
- `NET-00B-IPV4`
- `NET-00B-IPV6`

Setting Next-boot Device in BlueField UEFI Menu

The following steps detail the PXE deployment sequence:

1. Enter into the BlueField UEFI Setup menu.

Info

Refer to section "[Accessing the UEFI Menu](#)" for instructions.

2. Navigate to the Boot Manager menu.
3. Select the relevant PXE devices to boot with, depending on how BlueField is connected to the PXE network.



Setting BlueField to PXE Boot Using Redfish

To set boot option ID using Redfish:

```
curl -k -u '<username>:<password>' -X PATCH
https://<bmc_ip>/redfish/v1/Systems/Bluefield/Settings
-d '{
  "Boot": {
    "BootSourceOverrideEnabled": "Once",
    "BootSourceOverrideMode": "UEFI",
    "BootSourceOverrideTarget": "Pxe",
    "UefiTargetBootSourceOverride": "Boot000",
    "BootNext": "",
    "AutomaticRetryConfig": "Disabled"
  }
}' | jq
```

Info

Notice the `UefiTargetBootSourceOverride` and `BootSourceOverrideTarget` fields.

Info

Refer to the [BlueField BMC User Manual](#) for more information.

Note

To set up a PXE server, please refer to the documentation provided by the distribution vendor. For example, to install Ubuntu 20.04 or later, see official [Ubuntu 20.04 documentation](#).

Setting BlueField to PXE Boot Using efibootmgr

To set boot option ID using efibootmgr:

```
efibootmgr -n 0000
```

This sets a one-time next-boot to device 0000 (`NET-NIC_P0-IPV4`).

PXE/DHCP Server Configuration Tips

BlueField includes information in the DHCP vendor class identifier and vendor specific information options to help the DHCP server identify BlueField and select which boot image to serve (e.g., shim, grub). This can be especially useful for the PXE server to serve a specific shim based on the currently running BlueField OS and shim version to work around the installed UEFI secure boot SBAT entries and to reduce issues with BlueField upgrades and downgrades.

The DHCP vendor class ID will start with `NVIDIA/BF`.

The DHCP vendor specific information will contain:

- BF type (i.e. BF1, BF2, BF3)
- UEFI firmware version
- BF-bundle version

The following is an example of the DHCP vendor information supplied by a BlueField running an older OS version using the Linux `dhcpcdump` tool:

```
bf:~# dhcpcdump -i tmfifo_net0
...
OPTION: 53 ( 1) DHCP message type      1 (DHCPDISCOVER)
...
OPTION: 97 ( 17) UUID/GUID             009c2debc0368611
...-...6...
...
...
...
OPTION: 94 ( 3) Client NDI             010300
OPTION: 93 ( 2) Client System          000b
OPTION: 60 ( 13) Vendor class identifier NVIDIA/BF/PXE
OPTION: 43 (131) Vendor specific info  8005424633000081 ..BF3...
                                     30426c7565466965
0BlueFie
                                     6c643a342e382e30
ld:4.8.0
```

```

ge79e 2d322d6765373965 -2-
dirt 3037662d64697274 07f-
y..... 7900000000000000
..... 0000000000000000
..HDOCA_ 008248444f43415f
2.5.0_BS 322e352e305f4253
P_4.5.0_ 505f342e352e305f
Ubuntu_2 5562756e74755f32
1.2 322e30342d312e32 2.04-
..... 3032333131303800 0231108 .
..... 0000000000000000
..... 0000000000000000
..... 0000000000000000
..... 000000  ...
...

```

A DHCP server may include some of the following example information in its `dhcpd.conf` to match against the Bluefield above. Note that matching against the vendor specific information using a regex requires setting the substring start offset so that the search occurs after previous strings have ended:

```

...
# Match against vendor class ID

```

```

class "PXEClient" {
    match if substring (option vendor-class-identifier, 0, 13) =
"NVIDIA/BF/PXE";
    filename "/shimaa64.efi";
}

# Match against BF Type
class "BfType" {
    match if substring (option vendor-encapsulated-options, 0, 200)
~= "BF3";
    filename "/shimaa64.efi";
}

# Match against UEFI FW version
class "BfFwVer" {
    match if substring (option vendor-encapsulated-options, 6, 200)
~= "4.8.0-2-ge79e07f";
    filename "/shimaa64.efi";
}

# Match against BF Bundle version
class "BfBundleVer" {
    match if substring (option vendor-encapsulated-options, 55, 200)
~= "2.5.0-BSP_4.5.0_Ubuntu_22.04";
    filename "/shimaa64.efi";
}
...

```

Troubleshooting PXE Boot Issues

More information about how to troubleshoot PXE boot issues can be found in [NVIDIA BlueField Troubleshooting Guide](#).

Deploying BlueField Software Using BFB with PXE

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

PXE installation is not supported for NIC mode on NVIDIA® BlueField®-3.

The following are the steps to prepare a PXE server to deploy a BFB bundle:

1. Convert the BFB image file to PXE bootable image(s). Run:

```
# mlx-mkbfm -x <BFB>
```

For example:

```
# mlx-mkbfm -x DOCA_2.7.0_BSP_4.7.0_Ubuntu_22.04-  
<version>.bfm
```

(i) Note

`mlx-mkbf` is a Python script that can be found in BlueField release tarball under the `/bin` directory or in the BlueField Arm file system `/usr/bin/mlx-mkbf`.

2. Copy the output of the 2 files, `dump-image-v0` and `dump-initramfs-v0`, into the PXE server `tftp` path.

(i) Info

To use a kernel that is compressed with LZMA (as included in BFB as `dump-image-v0`) for PXE installation, it must be uncompressed before use:

```
$ file dump-image-v0
```

If the output says `LZMA compressed data`, proceed to uncompress.

To uncompress the kernel, run:

```
$ text xz -dc < dump-image-v0 > vmlinuz
```

This will uncompress the LZMA kernel image to a usable format called `vmlinuz`.

3. Create a boot entry in the PXE server. For example:

```

/var/lib/tftpboot/grub.cfg

set default=0
set timeout=5
menuentry 'Bluefield_Ubuntu_22_04_From_BFB' --class red --
class gnu-linux --class gnu --class os {
    linux (tftp)/ubuntu22.04/dump-image-v0 ro ip=dhcp
console=hvc0 console=ttyAMA0
    initrd (tftp)/ubuntu22.04/dump-initramfs-v0
}

```

If additional parameters must be set, use the `bf.cfg` configuration file, then add the `bfks` parameter to the Linux command line in the `grub.cfg` above.

```

menuentry 'Ubuntu22.04 From BFB with bf.cfg' --class red --
class gnu-linux --class gnu --class os {
    linux (tftp)/ubuntu22.04/dump-image-v0 console=hvc0
console=ttyAMA0 bfnet=oob_net0:dhcp
bfks=http://15.22.82.40/bfks
    initrd (tftp)/ubuntu22.04/dump-initramfs-v0
}

```

`bfks` is a BASH script that runs alongside BFB's `install.sh` script at the beginning of the BFB installation process. Here is an example of `bfks` that creates a `/etc/bf.cfg` file:

```

cat > /etc/bf.cfg << 'EOF'
DEBUG=yes
ubuntu_PASSWORD=' $1$3B0RlrfX$T1Hry93NFUJzg3Nya00rE1 '
EOF

```

4. Define DHCP:

```
/etc/dhcp/dhcpd.conf

allow booting;
allow bootp;

subnet 192.168.100.0 netmask 255.255.255.0 {
    range 192.168.100.10 192.168.100.20;
    option broadcast-address 192.168.100.255;
    option routers 192.168.100.1;
    option domain-name-servers <ip-address-list>
    option domain-search <domain-name-list>;
    next-server 192.168.100.1;
    filename "/BOOTAA64.EFI";
}

# Specify the IP address for this client.
host tmfifo_pxe_client {
    hardware ethernet 00:1a:ca:ff:ff:01;
    fixed-address 192.168.100.2;
}

subnet 20.7.0.0 netmask 255.255.0.0 {
    range 20.7.8.10 20.7.254.254;
    next-server 20.7.6.6;
    filename "/BOOTAA64.EFI";
}
```

Deploying BlueField Software Using ISO with PXE

BlueField software, including Ubuntu OS, NIC firmware and BMC software, can be deployed using an ISO image similar to the standard Ubuntu ISO deployment method. The BlueField ISO image is based on the standard Ubuntu ISO image for Arm64, with an updated kernel and added DOCA packages.

PXE booting the BlueField device with the ISO image results in Arm OS installation (including DOCA packages), NIC firmware and BMC firmware update (if `BMC_PASSWORD` is provided). This is equivalent to using the corresponding version of bf-bundle BFB.

An Ubuntu ISO image can be used if the RShim interface is not available or if there is an existing deployment system in place that can handle a standard Ubuntu ISO image.

PXE Server Setup

Mount the ISO:

```
$ mount bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso /mnt
$ cp /mnt/casper/vmlinuz /var/lib/tftpboot/boot/
$ cp /mnt/casper/initrd /var/lib/tftpboot/boot/
```

Example of `grub.cfg`:

```
menuentry "Install BF OS" {
    linux /boot/vmlinuz autoinstall fsck.mode=skip no-snapd
    console=hvc0 console=ttyAMA0 earlycon=pl011,0x13010000
    net.ifnames=0 biosdevname=0 iommu.passthrough=1 ip=dhcp
    url=http://<HTTP server IP>/jammy/ISO/bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso
    bfnet=eth0:dhcp bfks=http://<HTTP server IP>/jammy/ISO/bfks
```

```
    initrd /boot/initrd
}
```

The `bf.cfg` file can be used to customize the installation procedure. To create `bf.cfg` on the BlueField to be used for the installation use the `bfks` parameter to point to the script located on HTTP server that will create `bf.cfg` file:

bfks example:

```
cat > /etc/bf.cfg << 'EOF'
BMC_PASSWORD="..."
EOF
```

Standard automatic Ubuntu installation using `autoinstall.yaml` is also supported. See [Introduction to autoinstall - Ubuntu installation documentation](#).

Example of `autoinstall.yaml` that can be used to customize the installation and modify `bf.cfg`:

Example of a `grub.cfg` with `autoinstall.yaml`:

```
menuentry "Install BF OS" {
    linux /boot/vmlinuz autoinstall fsck.mode=skip no-snapd
    console=hvc0 console=ttyAMA0 earlycon=pl011,0x13010000
    net.ifnames=0 biosdevname=0 iommu.passthrough=1 ip=dhcp
    url=http://<HTTP server IP>/jammy/ISO/bf-bundle-2.7.0085-1-2024-06-14-22-36-50.iso force-
    ai=http://<HTTP server IP>/jammy/ISO/autoinstall.yaml cloud-config-url=/dev/null
    initrd /boot/initrd
}
```

Example of `autoinstall.yaml`:

```
version: 1

apt:
  preserve_sources_list: false
  conf: |
    Dpkg::Options {
      "--force-confdef";
      "--force-confold";
    };

storage:
  swap:
    size: 0
  grub:
    reorder_uefi: true
  config:
  - id: nvme0n1
    type: disk
    ptable: gpt
    path: /dev/nvme0n1
    name: osdisk
    wipe: superblock-recursive

  - id: nvme0n1-part1
    type: partition
    device: nvme0n1
    number: 1
    size: 50MB
    flag: boot
    grub_device: true

  - id: nvme0n1-part1-fs1
    type: format
    fstype: fat32
```

```
label: efi
volume: nvme0n1-part1

- id: nvme0n1-part2
  type: partition
  device: nvme0n1
  number: 2
  size: -1

- id: nvme0n1-part2-fs1
  type: format
  fstype: ext4
  label: root
  volume: nvme0n1-part2

- id: nvme0n1-mount
  type: mount
  path: /
  device: nvme0n1-part2-fs1
  options: defaults
  passno: 0
  fstype: auto

- id: nvme0n1-boot-mount
  type: mount
  path: /boot/efi
  device: nvme0n1-part1-fs1
  options: umask=0077
  passno: 1

reporting:
  builtin:
    type: print

# Add user-data so that subiquity doesn't complain about us not
# having a identity section
```

```
user-data:
  debug:
    verbose: true
  write_files:
    - path: /etc/iptables/rules.v4
      permissions: '0644'
      owner: 'root:root'
      content: |
        *mangle
        :PREROUTING ACCEPT [45:3582]
        :INPUT ACCEPT [45:3582]
        :FORWARD ACCEPT [0:0]
        :OUTPUT ACCEPT [36:4600]
        :POSTROUTING ACCEPT [36:4600]
        :KUBE-IPTABLES-HINT - [0:0]
        :KUBE-KUBELET-CANARY - [0:0]
        COMMIT
        *filter
        :INPUT ACCEPT [41:3374]
        :FORWARD ACCEPT [0:0]
        :OUTPUT ACCEPT [32:3672]
        :DOCKER-USER - [0:0]
        :KUBE-FIREWALL - [0:0]
        :KUBE-KUBELET-CANARY - [0:0]
        :LOGGING - [0:0]
        :POSTROUTING - [0:0]
        :PREROUTING - [0:0]
        -A INPUT -j KUBE-FIREWALL
        -A INPUT -p tcp -m tcp --dport 111 -j REJECT --reject-
with icmp-port-unreachable
        -A INPUT -p udp -m udp --dport 111 -j REJECT --reject-
with icmp-port-unreachable
        -A INPUT -i lo -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -d 127.0.0.0/8 -m mark --mark 0xb -m comment --
comment MD_IPTABLES -j DROP
```

```

        -A INPUT -m mark --mark 0xb -m state --state
RELATED,ESTABLISHED -m comment --comment MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp ! --dport 22 ! --tcp-flags
FIN,SYN,RST,ACK SYN -m mark --mark 0xb -m state --state NEW -m
comment --comment MD_IPTABLES -j DROP
        -A INPUT -f -m mark --mark 0xb -m comment --comment
MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags
FIN,SYN,RST,PSH,ACK,URG FIN,SYN,RST,PSH,ACK,URG -m mark --mark
0xb -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags
FIN,SYN,RST,PSH,ACK,URG NONE -m mark --mark 0xb -m comment --
comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m state --state INVALID -m
comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m tcp --tcp-flags RST RST -m mark --mark
0xb -m hashlimit --hashlimit-above 2/sec --hashlimit-burst 2 --
hashlimit-mode srcip --hashlimit-name hashlimit_0 --hashlimit-
htable-expire 30000 -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m mark --mark 0xb -m state --state NEW -
m hashlimit --hashlimit-above 50/sec --hashlimit-burst 50 --
hashlimit-mode srcip --hashlimit-name hashlimit_1 --hashlimit-
htable-expire 30000 -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -p tcp -m mark --mark 0xb -m conntrack --ctstate
NEW -m hashlimit --hashlimit-above 60/sec --hashlimit-burst 20 --
hashlimit-mode srcip --hashlimit-name hashlimit_2 --hashlimit-
htable-expire 30000 -m comment --comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m recent --rcheck --seconds
86400 --name portscan --mask 255.255.255.255 --rsource -m comment --
comment MD_IPTABLES -j DROP
        -A INPUT -m mark --mark 0xb -m recent --remove --name
portscan --mask 255.255.255.255 --rsource -m comment --comment
MD_IPTABLES
        -A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m
conntrack --ctstate NEW -m recent --set --name DEFAULT --mask
255.255.255.255 --rsource -m comment --comment MD_IPTABLES

```

```
-A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m
contrack --ctstate NEW -m recent --update --seconds 60 --
hitcount 50 --name DEFAULT --mask 255.255.255.255 --rsource -m
comment --comment MD_IPTABLES -j DROP
```

```
-A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m
contrack --ctstate NEW -m recent --set --name DEFAULT --mask
255.255.255.255 --rsource -m comment --comment MD_IPTABLES
```

```
-A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m
contrack --ctstate NEW -m recent --update --seconds 60 --
hitcount 10 --name DEFAULT --mask 255.255.255.255 --rsource -m
comment --comment MD_IPTABLES -j DROP
```

```
-A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m
contrack --ctstate NEW -m recent --set --name DEFAULT --mask
255.255.255.255 --rsource -m comment --comment MD_IPTABLES
```

```
-A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m
contrack --ctstate NEW -m recent --update --seconds 60 --
hitcount 100 --name DEFAULT --mask 255.255.255.255 --rsource -m
comment --comment MD_IPTABLES -j DROP
```

```
-A INPUT -p tcp -m tcp --dport 22 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
-A INPUT -p tcp -m tcp --dport 443 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
-A INPUT -p tcp -m tcp --dport 179 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
-A INPUT -p udp -m udp --dport 68 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
-A INPUT -p udp -m udp --dport 122 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
-A INPUT -p udp -m udp --dport 161 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```
-A INPUT -p udp -m udp --dport 6306 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 69 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 389 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 389 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 1812:1813 -m mark --mark 0xb
-m contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 49 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --dport 49 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p tcp -m tcp --sport 53 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 500 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 4500 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
-A INPUT -p udp -m udp --dport 1293 -m mark --mark 0xb -m
contrack --ctstate NEW,ESTABLISHED -m comment --comment
MD_IPTABLES -j ACCEPT
```

```

        -A INPUT -p tcp -m tcp --dport 1293 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 1707 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 1707 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -i lo -p udp -m udp --dport 3786 -m conntrack --
        ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j
        ACCEPT
        -A INPUT -i lo -p udp -m udp --dport 33000 -m conntrack --
        ctstate NEW,ESTABLISHED -m comment --comment MD_IPTABLES -j
        ACCEPT
        -A INPUT -p icmp -m mark --mark 0xb -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --sport 5353 --dport 5353 -m mark --
        mark 0xb -m conntrack --ctstate NEW,ESTABLISHED -m comment --
        comment MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 33434:33523 -m mark --mark
        0xb -m comment --comment MD_IPTABLES -j REJECT --reject-with
        icmp-port-unreachable
        -A INPUT -p udp -m udp --dport 123 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 514 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -p udp -m udp --dport 67 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        MD_IPTABLES -j ACCEPT
        -A INPUT -p tcp -m tcp --dport 60102 -m mark --mark 0xb -m
        conntrack --ctstate NEW,ESTABLISHED -m comment --comment
        "MD_IPTABLES: Feature HA port" -j ACCEPT

```

```

        -A INPUT -m mark --mark 0xb -m comment --comment
MD_IPTABLES -j LOGGING
        -A FORWARD -j DOCKER-USER
        -A OUTPUT -o oob_net0 -m comment --comment MD_IPTABLES -j
ACCEPT
        -A DOCKER-USER -j RETURN
        -A LOGGING -m mark --mark 0xb -m comment --comment
MD_IPTABLES -j NFLOG --nflog-prefix "IPTables-Dropped:" --nflog-group
3
        -A LOGGING -m mark --mark 0xb -m comment --comment
MD_IPTABLES -j DROP
        -A PREROUTING -i oob_net0 -m comment --comment
MD_IPTABLES -j MARK --set-xmark 0xb/0xffffffff
        -A PREROUTING -p tcp -m tcpmss ! --mss 536:65535 -m tcp ! -
-dport 22 -m mark --mark 0xb -m conntrack --ctstate NEW -m comment
--comment MD_IPTABLES -j DROP
COMMIT
*nat
:PREROUTING ACCEPT [1:320]
:INPUT ACCEPT [1:320]
:OUTPUT ACCEPT [8:556]
:POSTROUTING ACCEPT [8:556]
:KUBE-KUBELET-CANARY - [0:0]
:KUBE-MARK-DROP - [0:0]
:KUBE-MARK-MASQ - [0:0]
:KUBE-POSTROUTING - [0:0]
-A POSTROUTING -m comment --comment "kubernetes postrouting rules" -
j KUBE-POSTROUTING
-A KUBE-MARK-DROP -j MARK --set-xmark 0x8000/0x8000
-A KUBE-MARK-MASQ -j MARK --set-xmark 0x4000/0x4000
-A KUBE-POSTROUTING -m mark ! --mark 0x4000/0x4000 -j RETURN
-A KUBE-POSTROUTING -j MARK --set-xmark 0x4000/0x0
-A KUBE-POSTROUTING -m comment --comment "kubernetes service traffic
requiring SNAT" -j MASQUERADE --random-fully
COMMIT
users:

```

```

- name: ubuntu
  lock_passwd: False
  groups: adm, audio, cdrom, dialout, dip, floppy, lxd,
netdev, plugdev, sudo, video
  sudo: ALL=(ALL) NOPASSWD:ALL
  shell: /bin/bash
  plain_text_passwd: 'ubuntu'
chpasswd:
  list: |
    ubuntu:ubuntu
  expire: True
no_ssh_fingerprints: true
runcmd:
  - [ /usr/sbin/netfilter-persistent, start ]
  - [
/opt/mellanox/doca/services/telemetry/import_doca_telemetry.sh ]
  - [ /usr/bin/bfrshlog, "INFO: DPU is ready" ]

late-commands:
  # write release file
  - |
    cat << EOF > /target/etc/bf-release
    BF_NAME="Mellanox Bluefield"
    BF_PRETTY_NAME="Mellanox Bluefield"
    BF_SWBUILD_TIMESTAMP="2024-06-12-12-47-25"
    BF_SWBUILD_VERSION="2.7.0085-1"
    BF_COMMIT_ID="7fce146"
    BF_PLATFORM="BlueField SoC"
    BF_SERIAL_NUMBER="1332723060006"
    EOF

  # mount cdrom
  - mkdir -p /target/tmp/cdrom
  - mount --bind /cdrom /target/tmp/cdrom || true
  - |
    cat << EOF > /target/etc/apt/sources.list

```

```

deb [check-date=no] file:///tmp/cdrom/jammy main restricted
EOF

# avoid running flash kernel after install kernel
- mkdir -p /target/run/systemd
- echo docker > /target/run/systemd/container

# Install packages
- curtin in-target -- apt update -y
- curtin in-target -- apt remove -y --purge `dpkg --get-selections | grep
openipmi | awk '{print $2}'`
- curtin in-target -- /bin/bash -c "DEBIAN_FRONTEND=noninteractive
RUN_FW_UPDATER=no apt-get install --no-install-recommends -y acpid bc binutils bridge-utils build-
essential cracklib-runtime dc docker.io flash-kernel i2c-tools ifenslave iperf3 iptables-persistent iputils-
arping iputils-ping iputils-tracepath kexec-tools libpam-pwquality libssl-dev lldpad lm-sensors net-tools
network-manager nfs-common nvme-cli openssh-server python3.10 python3-pyinotify python3-pip
rasdaemon rsync sbsigntool shim-signed tcpdump watchdog doca-runtime doca-devel containerd kubelet
runc nv-common-apis nvidia-repo-keys linux-bluefield-modules-bluefield linux-image-5.15.0-1042-bluefield"

# rewrite sources
- |
cat << EOF > /target/etc/apt/sources.list
deb http://ports.ubuntu.com/ubuntu-ports/ jammy main restricted universe multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-updates main restricted universe
multiverse
deb http://ports.ubuntu.com/ubuntu-ports/ jammy-security main restricted universe multiverse
EOF

# Allow cloud-init to configure networking
- find /target/etc/cloud/cloud.cfg.d/ -type f ! -name README !
-name 05_logging.cfg ! -name 90_dpkg.cfg -delete || true;
- curtin in-target -- cloud-init clean

# Post-installation steps
# Create bf.cfg
- |
cat << EOF > /target/etc/bf.cfg

```

```

# UPDATE_ATF_UEFI - Updated ATF/UEFI (Default: yes)
# Relevant for PXE installation only as while using RSHIM interface
ATF/UEFI
# will always be updated using capsule method
UPDATE_ATF_UEFI="yes"

#####
# BMC Component Update

#####
# BMC_USER - User name to be used to access BMC (Default:
root)
BMC_USER="root"

# BMC_PASSWORD - Password used by the BMC user to access BMC
(Default: None)
BMC_PASSWORD=""

# BMC_IP_TIMEOUT - Maximum time in seconds to wait for the
connection to the
# BMC to be established (Default: 600)
BMC_IP_TIMEOUT=600

# BMC_TASK_TIMEOUT - Maximum time in seconds to wait for BMC
task (BMC/CEC
# Firmware update) to complete (Default: 1800)
BMC_TASK_TIMEOUT=1800

# UPDATE_BMC_FW - Update BMC firmware (Default: yes)
UPDATE_BMC_FW="yes"

# BMC_REBOOT - Reboot BMC after BMC firmware update to apply
the new version
# (Default: no). Note that the BMC reboot will reset the BMC
console.

```

```

BMC_REBOOT="no"

# UPDATE_CEC_FW - Update CEC firmware (Default: yes)
UPDATE_CEC_FW="yes"

# UPDATE_DPU_GOLDEN_IMAGE - Update BlueField Golden Image
(Default: yes)
UPDATE_DPU_GOLDEN_IMAGE="yes"

# UPDATE_NIC_FW_GOLDEN_IMAGE- Update NIC firmware Golden
Image (Default: yes)
UPDATE_NIC_FW_GOLDEN_IMAGE="yes"

# pre_bmc_components_update - Shell function called by BFB's
install.sh before
# updating BMC components (no communication to the BMC is
established at this
# point)

# post_bmc_components_update - Shell function called by BFB's
install.sh after
# updating BMC components

#####
# NIC Firmware update

#####
# WITH_NIC_FW_UPDATE - Update NIC Firmware (Default: no)
WITH_NIC_FW_UPDATE="yes"
EOF

# Run post-installation script to update ATF/UEFI, NIC
firmware and BMC components
- curtin in-target -- /bin/bash -c "device=/dev/nvme0n1 /usr/local/sbin/bfiso-
post-install.sh || true"

```

```
- curtin in-target -- systemctl disable snapd
```

PXE Sequence with Redfish

HTTP boot configuration can be done using the BlueField BMC's Redfish interface.

ISO upgrade via Redfish to set UEFI HTTPs/PXE boot by setting UEFI first boot source.

To set the UEFI first boot source using Redfish:

1. Follow the instructions under section "[Deploying BlueField Software Using BFB with PXE](#)".
2. Check the current boot override settings by performing a GET on the `ComputerSystem` schema over 1GbE to the BlueField BMC. Look for the `"Boot"` property.

```
curl -k -X GET -u root:<password> https://<BF-BMC-IP>/redfish/v1/Systems/<SystemID>/ | python3 -m json.tool
{
  ...
  "Boot": {
    "BootNext": "",
    "BootOrderPropertySelection": "BootOrder",
    "BootSourceOverrideEnabled": "Disabled",
    "BootSourceOverrideMode": "UEFI",
    "BootSourceOverrideTarget": "None",
    "UefiTargetBootSourceOverride": "None",
    .....
  },
  ....
  "BootSourceOverrideEnabled@Redfish.AllowableValues": [
    "Once",
    "Continuous",
    "Disabled"
```

```

    ],
    "BootSourceOverrideTarget@Redfish.AllowableValues": [
        "None",
        "Pxe",
        "UefiHttp",
        "UefiShell",
        "UefiTarget",
        "UefiBootNext"
    ],
    ....
}

```

Info

Boot override enables overriding the first boot source, either once or continuously.

3. The example output above shows the `BootSourceOverrideEnabled` property is `Disabled` and `BootSourceOverrideTarget` is `None`. The `BootSourceOverrideMode` property should always be set to `UEFI`. Allowable values of `BootSourceOverrideEnabled` and `BootSourceOverrideTarget` are defined in the metadata (`BootSourceOverrideEnabled@Redfish.AllowableValues` and `BootSourceOverrideTarget@Redfish.AllowableValues` respectively).
4. If `BootSourceOverrideEnabled` is set to `Once`, then boot override is disabled after the first boot, and any related properties are reset to their former values to avoid repetition. If it is set to `Continuous`, then on every reboot the BlueField keeps performing boot override (HTTPBoot).
5. To perform boot override, perform a PATCH to pending settings URI over the 1GbE to the BlueField BMC.

```
curl -k -X PATCH -d '{"Boot":
{"BootSourceOverrideEnabled": "Once",
"BootSourceOverrideMode": "UEFI", "BootSourceOverrideTarget":
"UefiHttp", "HttpBootUri": "http://<HTTP-Server-
Ip>/Image.iso"}}' -u root:<password> https://<BF-BMC-
IP>/redfish/v1/Systems/<SystemID>/Settings | python3 -m
json.tool
```

For example:

```
curl -k -X GET -u root:<password> https://<BF-BMC-
IP>/redfish/v1/Systems/<SystemID>/ | python3 -m json.tool
{
...
"Boot": {
    "BootNext": "",
    "BootOrderPropertySelection": "BootOrder",
    "BootSourceOverrideEnabled": "Once",
    "BootSourceOverrideMode": "UEFI",
    "BootSourceOverrideTarget": "UefiHttp",
    "UefiTargetBootSourceOverride": "None",
    .....
},
.....
}
```

6. After performing the above PATCH successfully, reboot the BlueField using the Redfish Manager schema over the 1GbE to the BlueField BMC:

```
curl -k -u root:<password> -H "Content-Type:
application/json" -X POST https://<BF-BMC-
```

```
IP>/redfish/v1/Systems/Bluefield/Actions/ComputerSystem.Reset  
-d '{"ResetType" : "GracefulRestart"}
```

7. Once UEFI has completed, check whether the settings are applied by performing a GET on `ComputerSystem` schema over the 1GbE OOB to the BlueField BMC.

Note

The `HttpBootUri` property is parsed by the Redfish server and the URI is presented to the BlueField as part of DHCP lease when the BlueField performs the HTTP boot.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation (“NVIDIA”) makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer (“Terms of Sale”). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer’s own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer’s sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer’s product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, “MATERIALS”) ARE BEING PROVIDED “AS IS.” NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF

ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA and the NVIDIA logo are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

© Copyright 2025, NVIDIA. PDF Generated on 12/15/2025