Porting fwupd to the BSD Operating Systems

Keep your hardware safe with up-to-date firmware

EuroBSDcon 2021

Norbert Kamiński

3MDEB

Agenda

- $ whoami
- Who we are?
- Overall information about fwupd
- fwupd tool architecture
- LVFS
- Porting fwupd to the *BSDs
- FreeBSD CI for fwupd
- Updating USB devices
- Updating UEFI devices
- Q&A

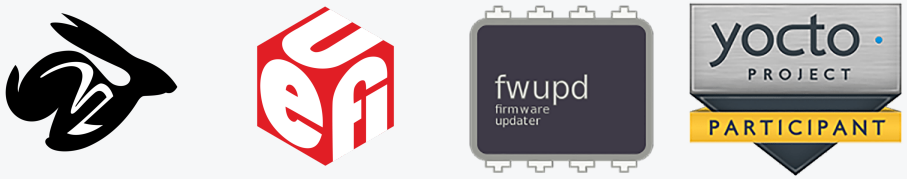
$ whoami

Norbert Kamiński
Embedded Systems Engineer

- open-source contributor:
  - fwupd
  - meta-pcengines
- scope of interests:
  - firmware upgrade tools
  - virtualization
  - firmware security

- norbert.kaminski@3mdeb.com
- linkedin.com/in/norbert-kami%C5%84ski/
- facebook.com/nkaminski3
- @asiderr

Who we are?



- coreboot licensed service providers since 2016
- coreboot project leadership participants
- UEFI Adopters since 2018
- Official consultants for fwupd/LVFS project
- Open Source Firmware enthusiasts and evangelists

Project genesis

- Our clients were asking if there is an easy way to upgrade firmware in BSD distributions
- The community were asking if there is the possibility to port fwupd to BSD distributions
- fwupd port is funded by NLNet foundation - https://nlnet.nl/project/fwdup-BSD/

fwupd - overall information

- Outdated firmware makes devices vulnerable to the different attacks
- fwupd project can query supported hardware for the current firmware versions and also deploy new firmware versions to devices
- LVFS is a secure web service that provides information about available firmware updates. It can be used by the OEM's to upload firmware archives downloaded by the users
- Our mission is to port fwupd to BSDs to make the firmware update process easier for the BSD community

fwupd/LVFS architecture

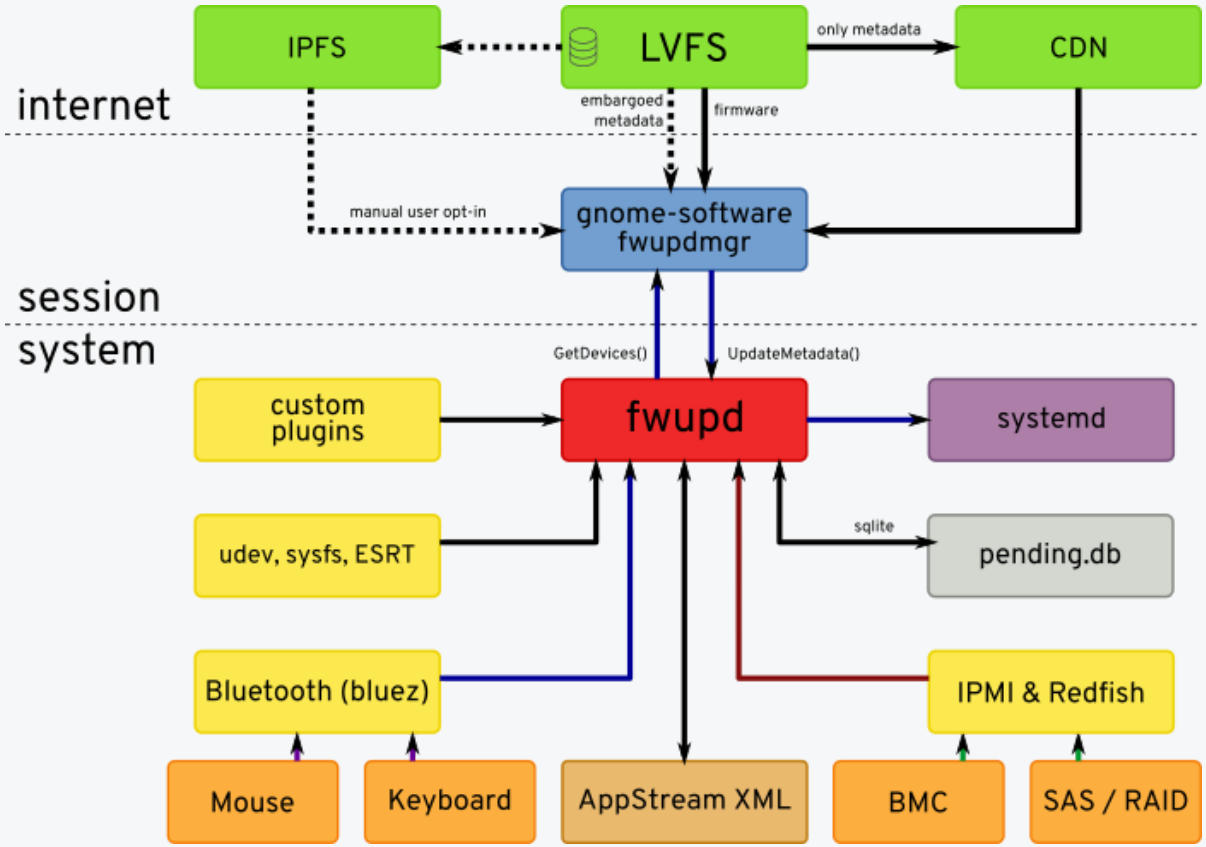


Image source: https://lvfs.readthedocs.io/en/latest/intro.html

Linux Vendor Firmware Service (LVFS)

- The LVFS is a secure web service that is used by OEM's to provide firmware updates
- The LVFS provides metadata that contains information about possible updates
- The firmware updates are packed into cabinet archives. The archive contains the firmware blob, information about the update, and jcat file, which is used to verify the firmware updates
- A manufacturer is signing the firmware and this sign is verified during the update

Compilation of fwupd on the *BSDs

- We focused on four BSD operating systems: DragonflyBSD, FreeBSD, NetBSD, and OpenBSD
- Initially, we wanted to create one pkgsrc package for all four operating systems
- The pkgsrc documentation [1] declares support for each OS, but after proof of concept work and dependencies hell that we met, we decided to create four different packages using the native package managers

[1] https://www.netbsd.org/docs/pkgsrc/introduction.html#intro.platforms

Process of creation fwupd port for *BSDs

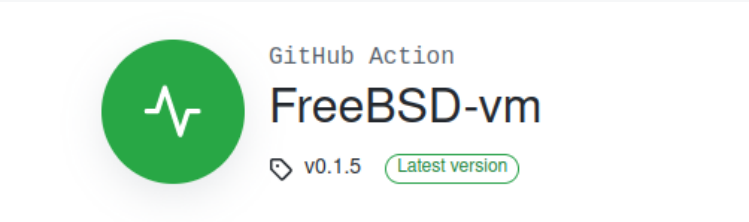
- At first, we have created the ports for the fwupd dependencies (libgusb, libjcat, libxmlb, libefiboot)
- Then we added ifdefs for Linux related parts of the fwupd code
- The last step was an adaptation of the fwupd plugins to the *BSDs needs

Current status of work

- We created fwupd packages for all four operating systems. FreeBSD package is during the upstream process. The rest of the packages will be upstreamed in the close future
- Our initial goal was to develop the fwupd functionalities in the FreeBSD
- Our next steps will be testing fwupd functionalities and solving the problems in the DragonflyBSD

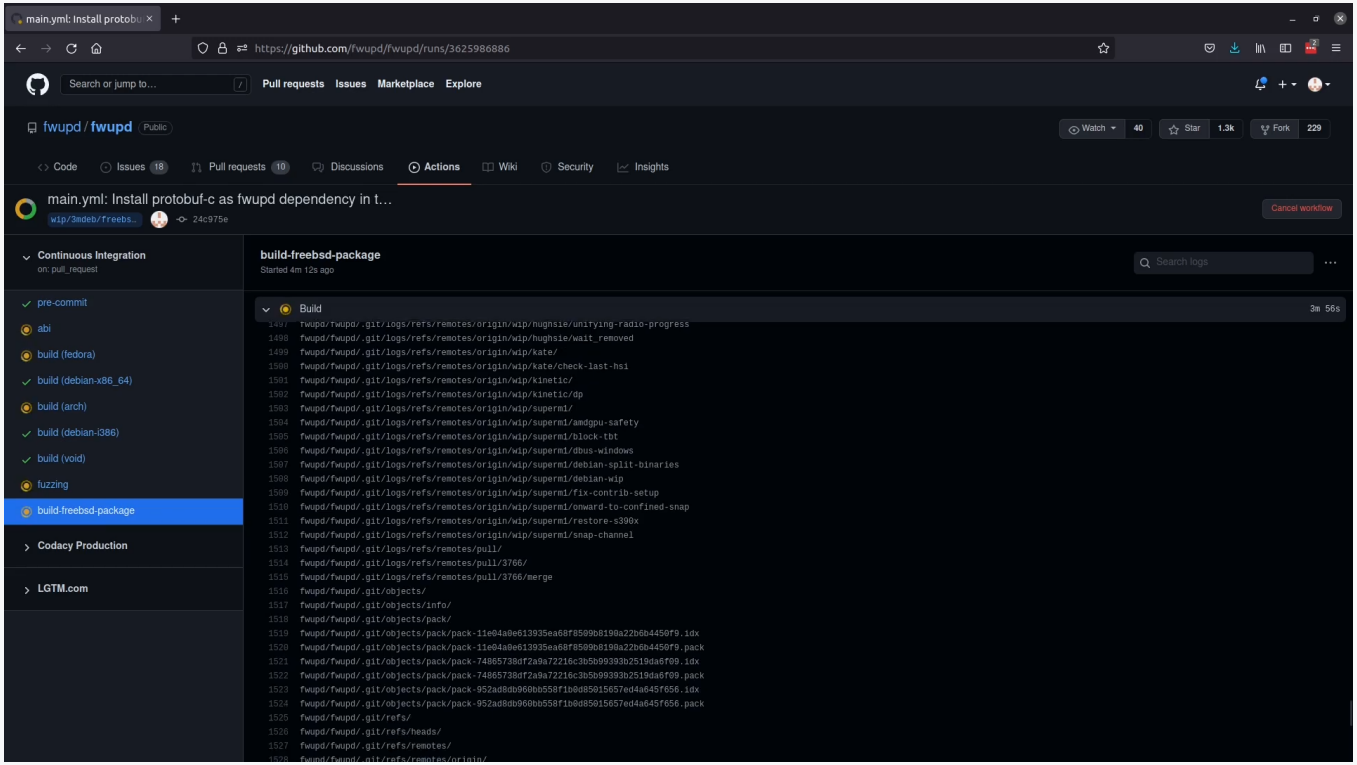
FreeBSD CI for fwupd

- At the beginning of fwupd for *BSDs development, we created FreeBSD Continuous Integration [1] in the fwupd repository
- Continuous Integration let us develop new changes with the confidence that we do not break the build process
- We used GitHub actions FreeBSD virtual machine [2] as a testing environment
- As a result, CI creates a fwupd package that could be installed in the FreeBSD



[1] https://github.com/fwupd/fwupd/blob/master/.github/workflows/main.yml#L80 [2] https://github.com/marketplace/actions/freebsd-vm

FreeBSD CI for fwupd - demo



[1] https://github.com/fwupd/fwupd/runs/3626962351 [2] https://youtu.be/-j20jQ_hpzo

Information about possible updates

- To update the device we need to know exactly what hardware is connected to our PC and if there are possible firmware updates for our devices
- For this purposes, we are using three fwupd commands - `get-devices`, `refresh`, and `get-updates`
- There was a problem with updating fwupd metadata information about possible firmware updates
- It was caused by `memfd_create()` which was not available at FreeBSD 12.2 (It is already added in the FreeBSD 13.0)
- Now fwupd checks if the function is available. If not, it emulates an in-memory file by an unlinked temporary file [1]

[1] https://github.com/fwupd/fwupd/pull/3279/files

fwupdmgr get-devices

```
nkaminski@nkaminski:~/projects/fwupd/build/src $ ./fwupdmgr  get-devices
WARNING: This package has not been validated, it may not work properly.
Unknown Product
│
└─ColorHug2:
      Device ID:          003dd5443e411c857e1a6220d9b68ee3136661ec
      Summary:            An open source display colorimeter
      Current version:    2.0.6
      Vendor:             Hughski Ltd. (USB:0x273F)
      Install Duration:   8 seconds
      GUIDs:              2082b5e0-7a64-478a-b1b2-e3404fab6dad
                          aa4b4156-9732-55db-9500-bf6388508ee3
                          101ee86a-7bea-59fb-9f89-6b6297ceed3b
                          2fa8891f-3ece-53a4-adc4-0dd875685f30
      Device Flags:       • Updatable
                          • Supported on remote server
                          • Device can recover flash failures
```

fwupdmgr refresh

```
nkaminski@nkaminski:~/projects/fwupd/build/src $ ./fwupdmgr refresh --force
Fetching metadata https://cdn.fwupd.org/downloads/firmware.xml.gz
Downloading…            [**************************************]
Fetching signature https://cdn.fwupd.org/downloads/firmware.xml.gz.jcat

Successfully downloaded new metadata: 0 local devices supported
```

fwupdmgr get-updates

```
nkaminski@nkaminski:~/projects/fwupd/build/src $ ./fwupdmgr  get-updates
WARNING: This package has not been validated, it may not work properly.
Unknown Product
│
└─ColorHug2:
  │  Device ID:          003dd5443e411c857e1a6220d9b68ee3136661ec
  │  Summary:            An open source display colorimeter
  │  Current version:    2.0.6
  │  Vendor:             Hughski Ltd. (USB:0x273F)
  │  Install Duration:   8 seconds
  │  GUIDs:              2082b5e0-7a64-478a-b1b2-e3404fab6dad
  │                      aa4b4156-9732-55db-9500-bf6388508ee3
  │                      101ee86a-7bea-59fb-9f89-6b6297ceed3b
  │                      2fa8891f-3ece-53a4-adc4-0dd875685f30
  │  Device Flags:       • Updatable
  │                      • Supported on remote server
  │                      • Device can recover flash failures
  │
  └─ColorHug2 Device Update:
    New version:      2.0.7
    Remote ID:        lvfs
    Summary:          Firmware for the Hughski ColorHug2 Colorimeter
    License:          GPL-2.0+
    Size:             16.4 kB
    Created:          2016-12-28
    Urgency:          Medium
    Source:           https://github.com/hughski/colorhug2-firmware
    Vendor:           Hughski Limited
    Duration:         8 seconds
    Flags:            is-upgrade
    Description:
    This release fixes prevents the firmware returning an error when the remote SHA1 hash was never sent.
```

Updating USB devices

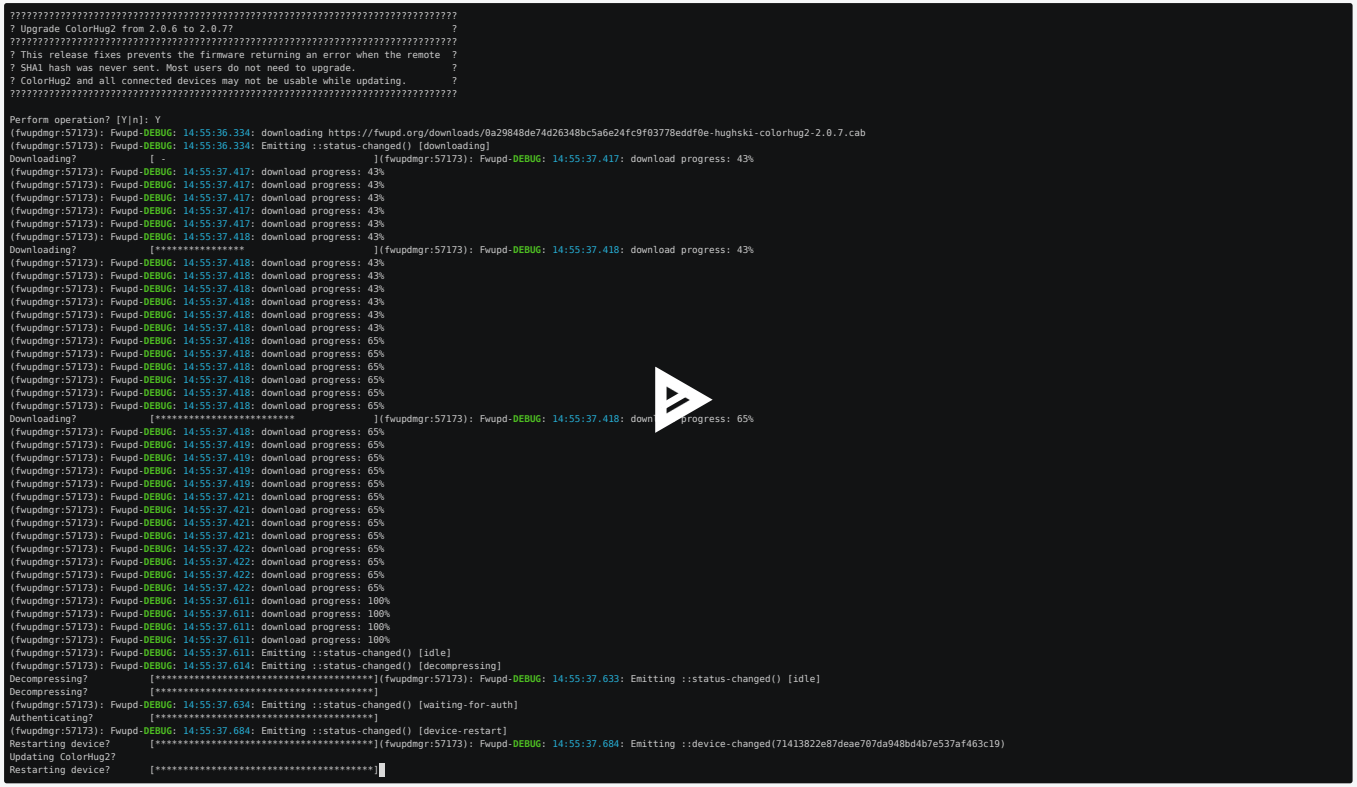
- The next step was enabling firmware updates for USB devices
- We encountered a problem with FreeBSD libusb which prevented USB devices from returning to the operating system after reboot
- fwupd uses the libgusb library, a GLib wrapper around libusb. The usual flow of an update is as follows:
    - Issue command to the device to enter bootloader mode - in the case of ColorHug2, a custom HID-based flashing mode
    - Write an update to the device
    - Upon successful update, return the device to runtime mode
- The issue occurred after the first step

Updating USB devices firmware

- We were unable to reattach the device to the host. After issuing a command to reset the device back to normal operation, the OS would not recognize and reattach it - it would stay gone
- Because libgusb uses libusb asynchronous API, fwupd would close a device after an update before all events had been processed. Upon processing such an event, libusb would detect that the device is gone and mark it with a device_is_gone flag
- This meant that on all future requests, libusb would fail with a LIBUSB_ERROR_NO_DEVICE error
- We fixed that by clearing the device gone flag, in case the device was opened after a re-attach, to allow new transaction [1]

[1] https://cgit.freebsd.org/src/commit/?id=6847ea50196f1a685be408a24f01cb8d407da19c

Updating USB devices firmware - demo



[1] https://asciinema.org/a/G2OT5XvMpv9r10Q6qD5rZBbLA

Updating UEFI devices firmware

- From the perspective of security, UEFI updates are critical, and implementing this functionality for FreeBSD was a priority
- There were a couple parts that had to be implemented to make it work:
  - UEFI ESRT table support in FreeBSD, and support for it in fwupd
  - FreeBSD efivar backend for fwupd - on Linux, efivar support is implemented via a sysfs interface, while FreeBSD has a C API
  - bsdisks support in fwupd
  - adding FreeBSD support in the UEFI update capsule plugin

Updating UEFI devices firmware - UEFI ESRT

- UEFI ESRT (EFI System Resource Table) is a standard interface for firmware updates available since UEFI 2.5. It exposes, among other data, information about the currently installed firmware versions and the status of the last update attempt
- It's used by fwupd for detection and matching available updates. Support for these tables was missing in FreeBSD, so we added it [1]

[1] https://reviews.freebsd.org/D30104

Updating UEFI devices firmware

- fwupd applies firmware updates by installing a small EFI binary along with the update capsule into the ESP and setting the EFI bootnext variable to point to it
- The machine reboots and launches the EFI binary which then calls UpdateCapsule(), which tells the UEFI to apply the capsule. The actual

flashing is handled by the UEFI implementation itself

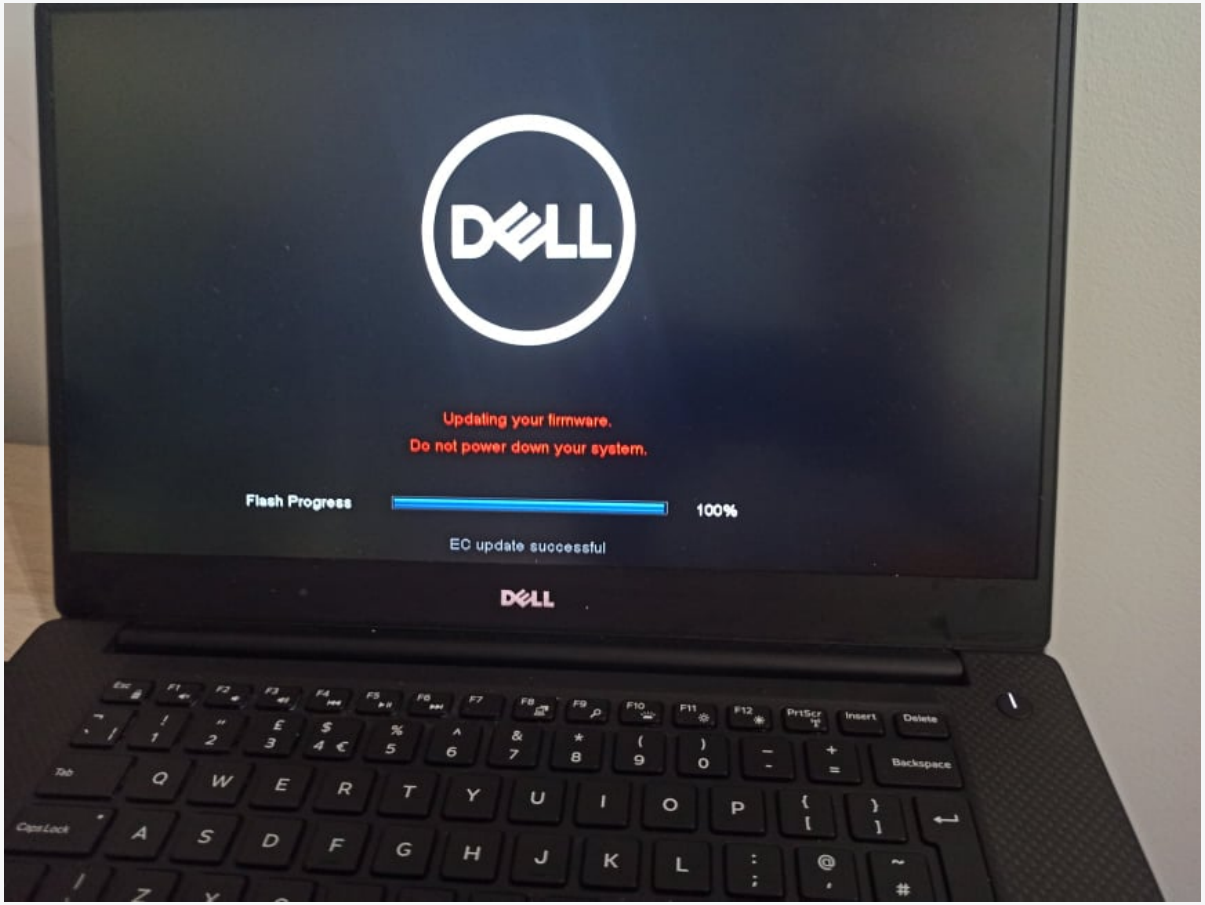
- This requires efivar support, and FreeBSD has a different, programmatic API, so support for it had to be added in fwupd. Furthermore, since FreeBSD has a disk management API that differs slightly from the Linux standard UDisks2 API, support for it also had to be added [1], [2]

[1] https://github.com/fwupd/fwupd/pull/3330 [2] https://github.com/fwupd/fwupd/pull/3318

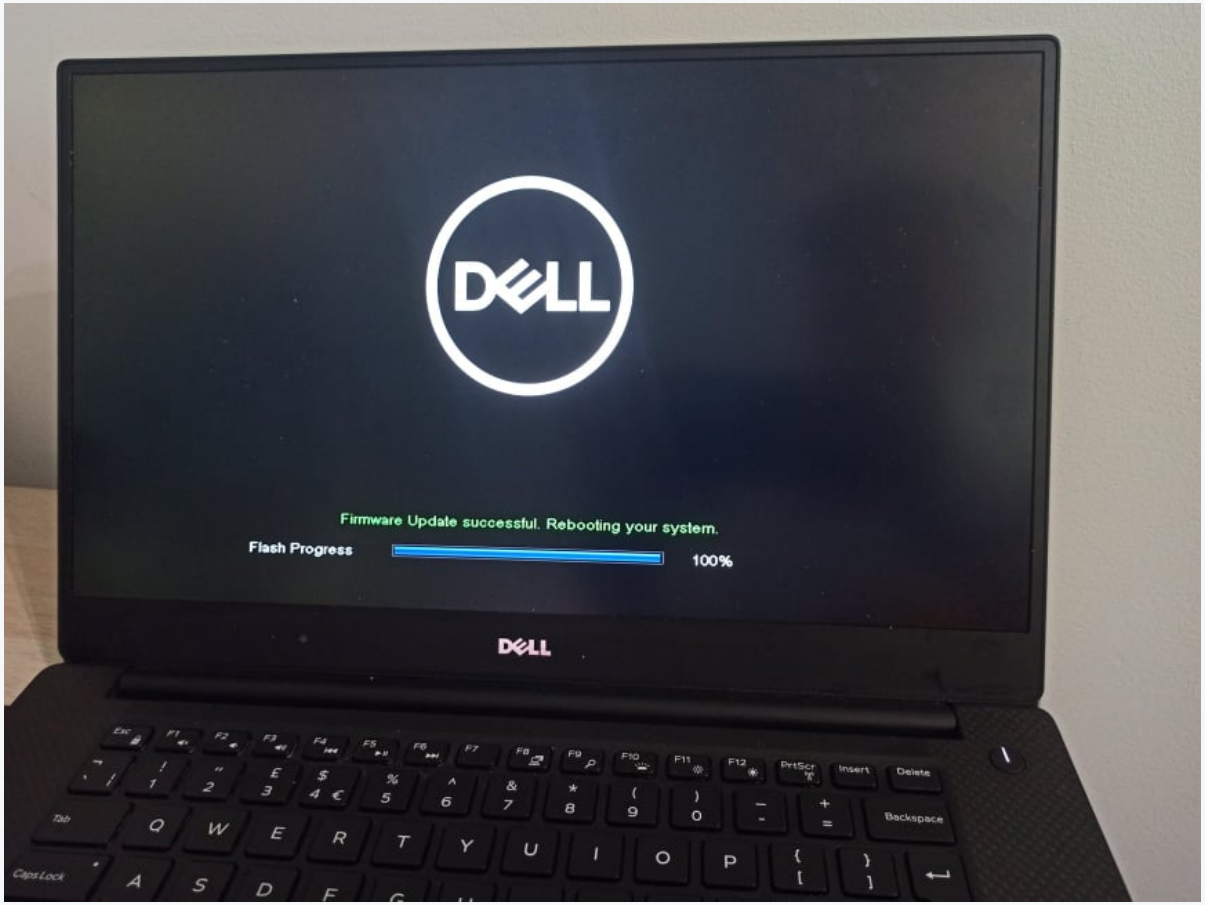
Updating UEFI devices - demo

[1] https://asciinema.org/a/EG2W6t13jeyxyoQIxzc4dmgeQ

Updating UEFI devices - update process



Updating UEFI devices - update process



Contact us

We are open to cooperate and discuss

- contact@3mdeb.com
- facebook.com/3mdeb
- @3mdeb_com
- linkedin.com/company/3mdeb
- https://3mdeb.com
- Book a call
- Sign up for the newsletter

Feel free to contact us if you believe we can help you in any way. We are always open to cooperate and discuss.

Many thanks!

Many thanks to our engineers who make fwupd for BSD possible:

- Michał Kopeć
- Sergii Dmytruk
- Pavel Balaev

Q&A