Memfault

Device Firmware Update Best Practices

François Baldassari, Founder & CEO, Memfault

Webinar | February 11, 2021

The Speaker

François Baldassari

Founder & CEO, Memfault

- Passion: tooling and automation in software engineering
- Previously a Firmware Engineer @ Pebble, Oculus, Sun Microsystems
- Can find my thoughts and content on Memfault's Interrupt blog (interrupt.memfault.com)

pebble Memfault oculus

Agenda Today

1. Firmware Update Architecture
2. Firmware Update Distribution

Why do we need DFU?

You are bound to ship bugs

10-100 defects per 1000 lines of code, unless you're in the Elite 1%

-Ganssle

Group	Avg bugs per KLOC
All	120.8
Top 25%	61.9
Top 10%	28.9
Top 1%	11.2

*Source: http://www.ganssle.com/tem/tem299.html

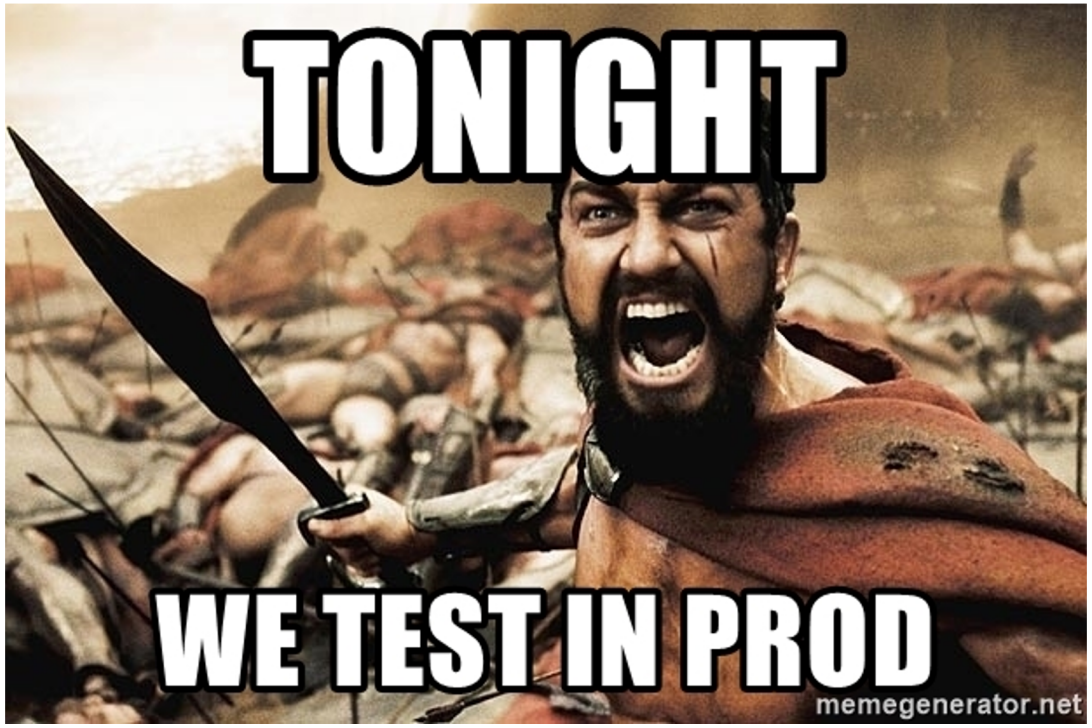
Even the best do

"This is the **third upgrade version** since Curiosity's landing on Mars 16 months ago [...]. An earlier switch to version 11 prompted an unintended reboot on Nov. 7 and a return to version 10, but the latest transition went smoothly."

*Source: https://www.nasa.gov/jpl/msl/mars-rover-curiosity-20131220/

Law of Large Numbers

Some issues will only be found in production



SDKs and Protocols

It's **not just your code** you have to worry about!

Will change at an accelerating pace

- Chip SDK
- Protocol implementations
- 3rd party libraries

- Maintaining compatibility or keeping up with SSL certs will require future updates

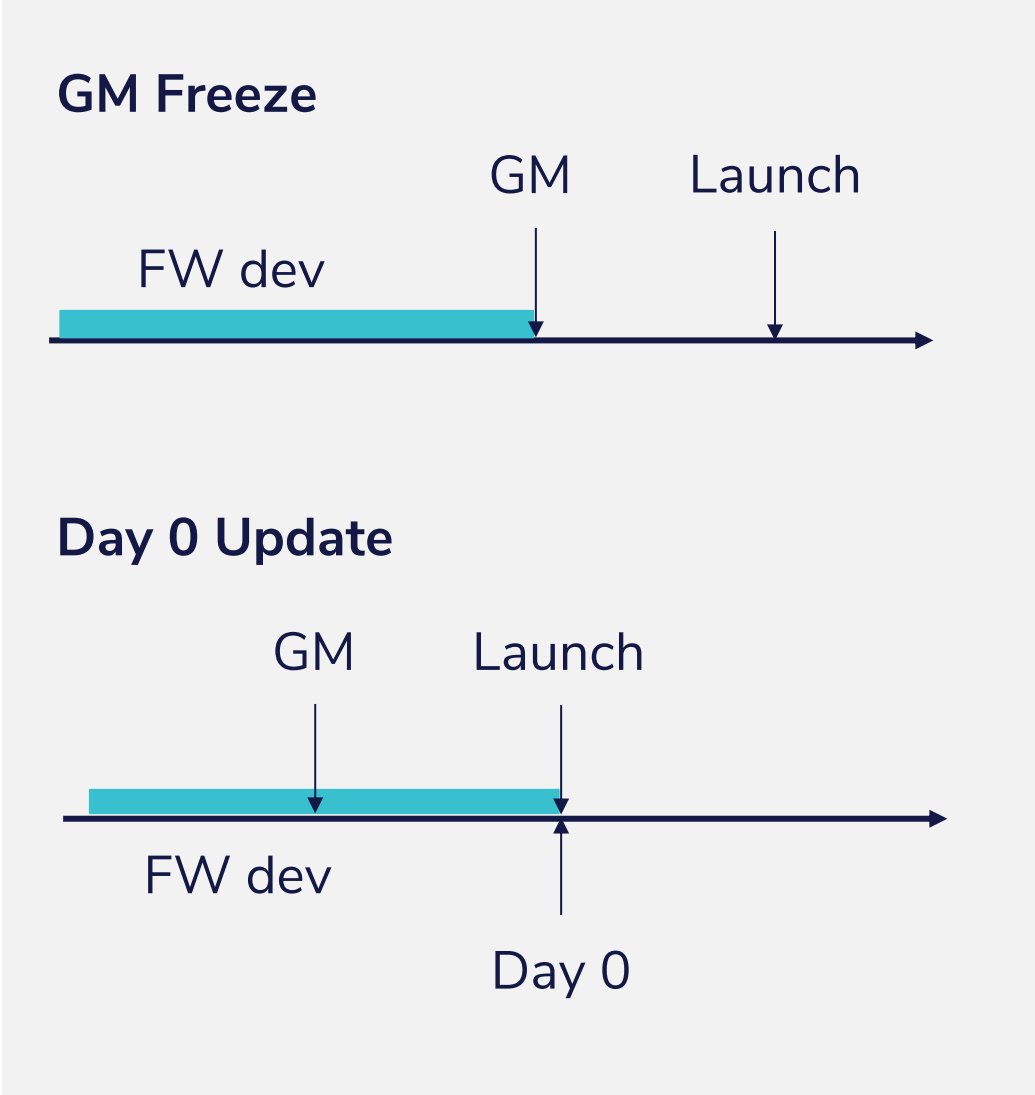
Keep up with SDKs and protocols

"The Bluetooth Low Energy implementation in Cypress PSoC Creator BLE 4.2 component versions before 3.64 generates [a] predictable or brute-forceable random number **[which] allows an attacker (in radio range) to perform a MITM attack during BLE pairing.**"

*Source: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-11957

Ship better products, faster

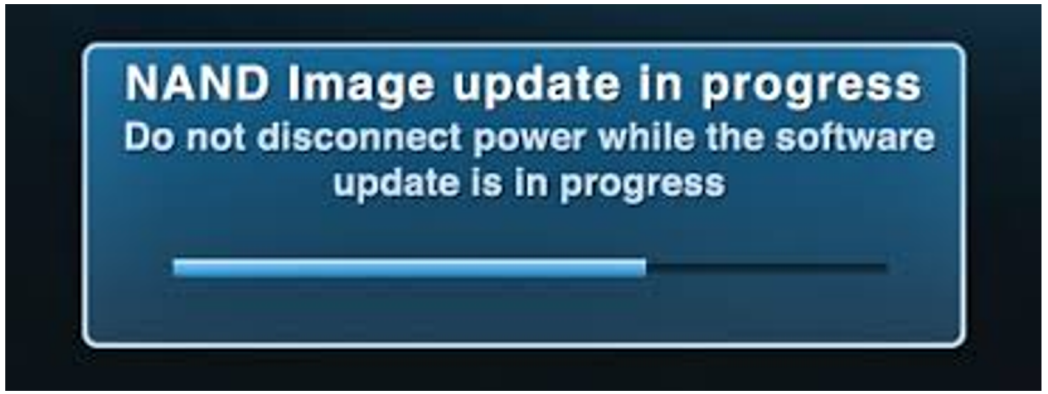
- Agile development workflows: ship MVP and iterate
- Increasingly, products enter MP unfinished, and get completed in a day-0 update
- Complex algorithms can always be improved!



Firmware Update Architecture (for MCU)

What could possibly go wrong?

- DFU is one of the more sensitive sub-systems in your firmware
- Done poorly, there is a real risk to brick devices



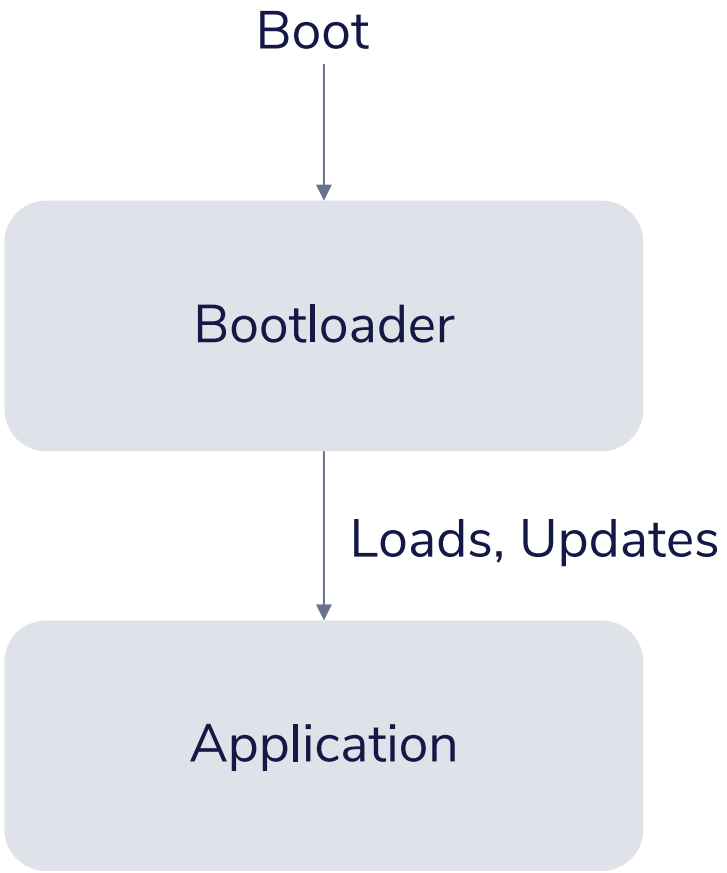
Principles for Safe OTA design

1. DFU should be separate from the application
2. DFU should be updateable
3. Failure mid-DFU should be recoverable
4. Version your non-volatile data

1. DFU should be separate from the application

- App code needs to change regularly
- DFU code should be as stable as possible

→ *to avoid inadvertently breaking DFU with a bad update, keep the DFU code separate from the app*

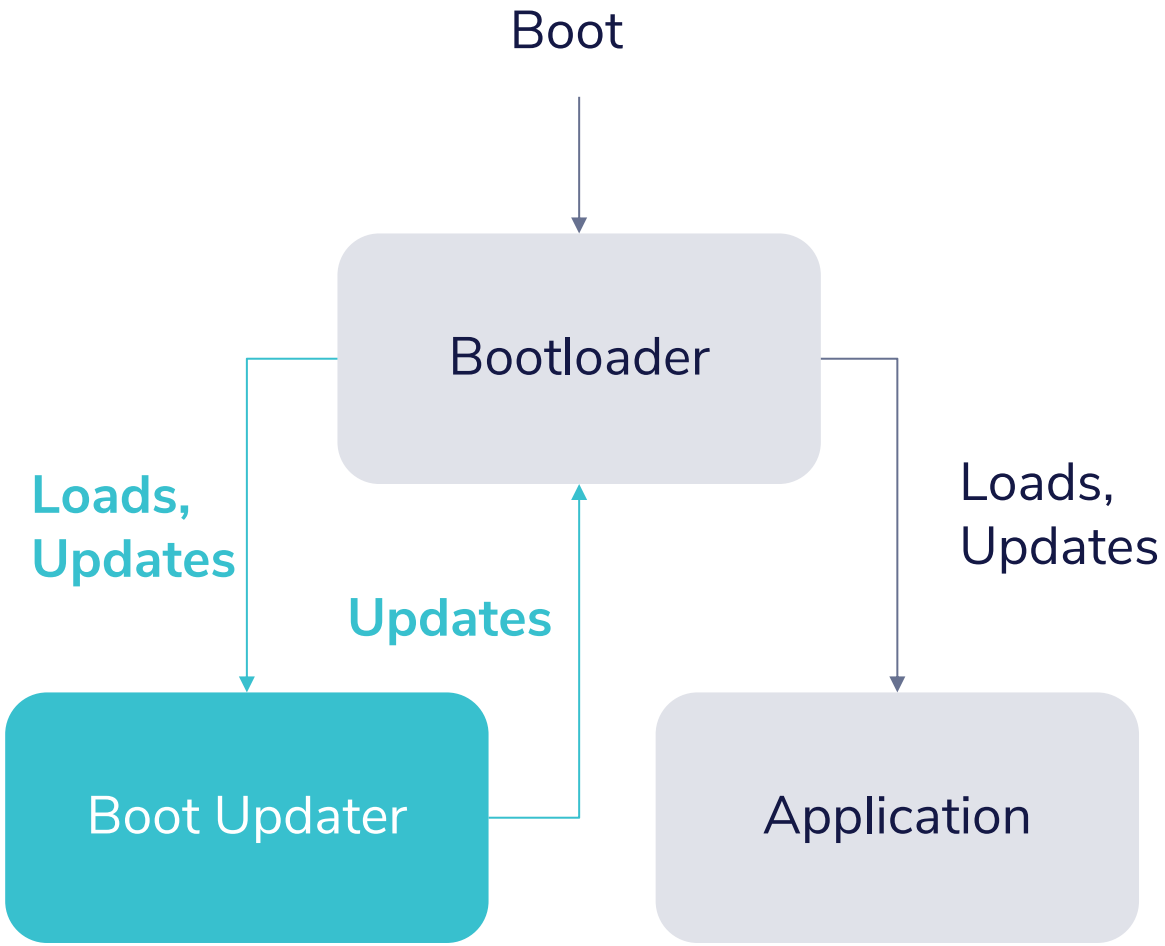


2. DFU should be updateable

Bugs in our DFU codes are inevitable

- Need a second DFU process to update our bootloader, but *outside* of the application

→ *Create a third program, the "DFU updater"*

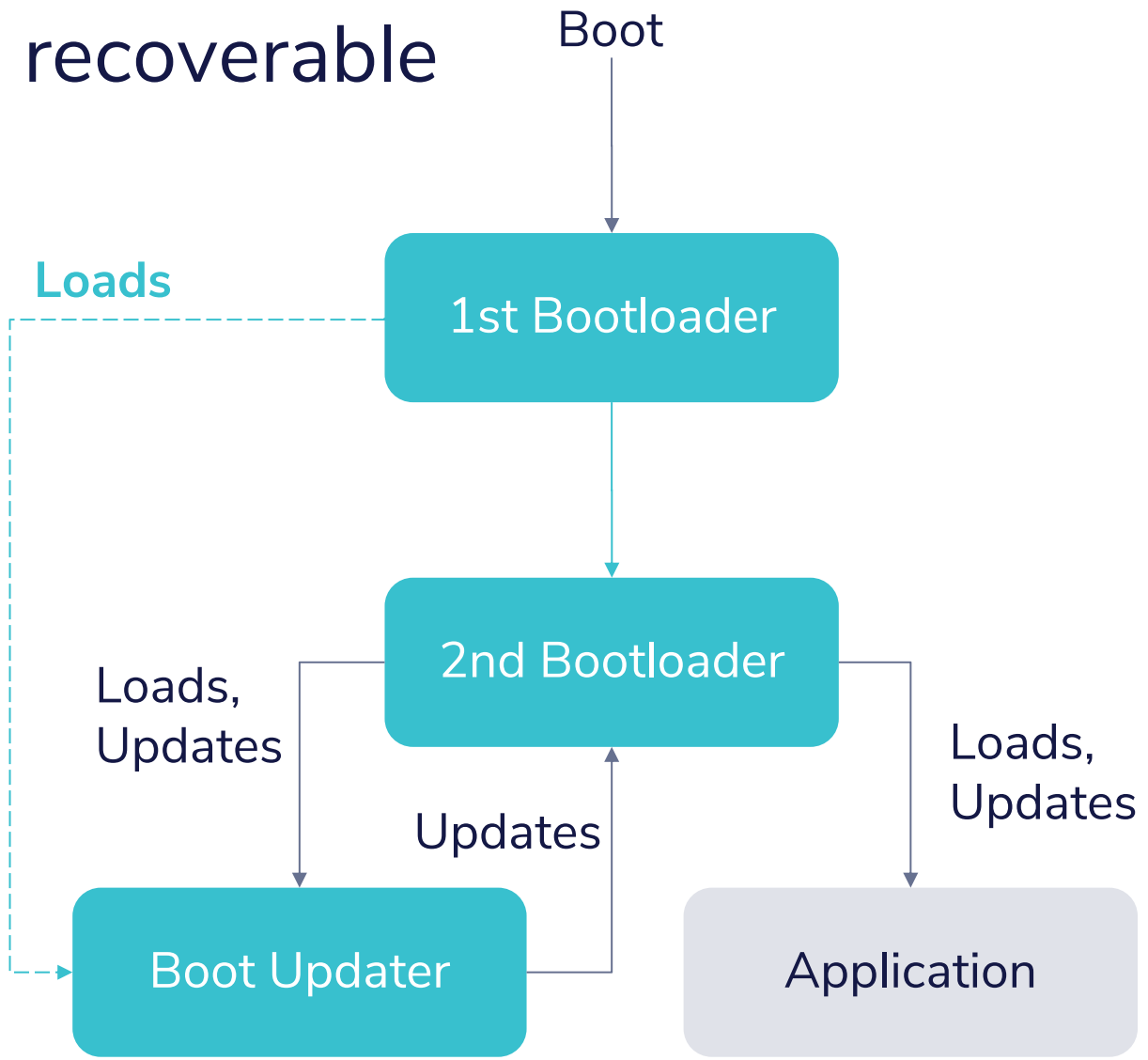


3. Failure mid-DFU should be recoverable

What do we do if Loader is mid-update?

- Need a way to fallback to our DFU-updater

→ *Add an immutable primary bootloader*



4. Version your non-volatile data

- Non-volatile data will need to be migrated
- Ensure all data structures have:
  - Version field
  - Commit-bit
  - Erase-bit
- Keep a migration function for every version increment
  - (e.g. 1->2, 2->3, 3->4, ...etc.)

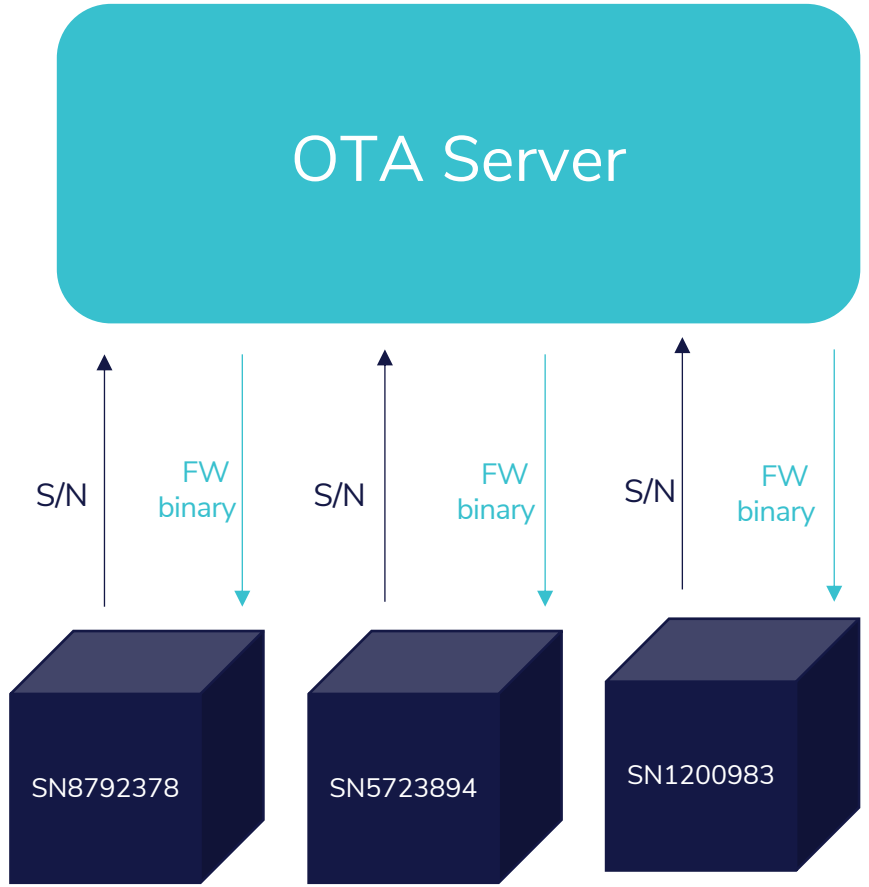
```
struct
__attribute__((packed))
{
  uint8_t version;
  uint8_t commit_bit : 1;
  uint8_t erase_bit : 1;
  ...
} foo;
```

Distributing Updates at Scale

Right Firmware to Right Device

A few different firmware images based on hardware configuration

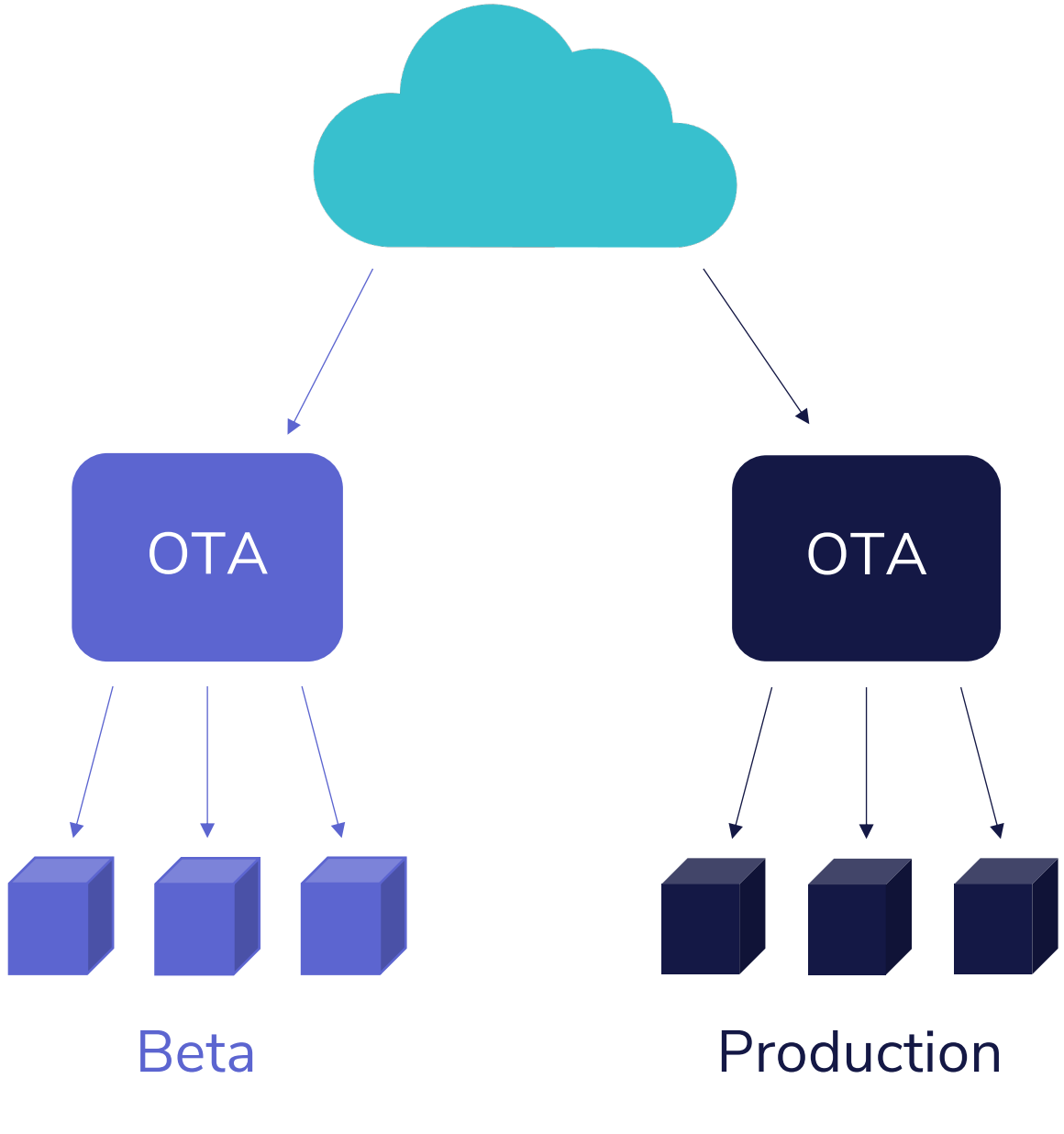
- *Simplest solution:* host different firmware at different paths.
  - E.g., http://foo.com/firwmares/hw-version-latest.bin
- ***Better solution:*** create rules for which firmware to send based on serial number
  - This enables beta releases, experiments, and limited rollouts



Staged Rollouts

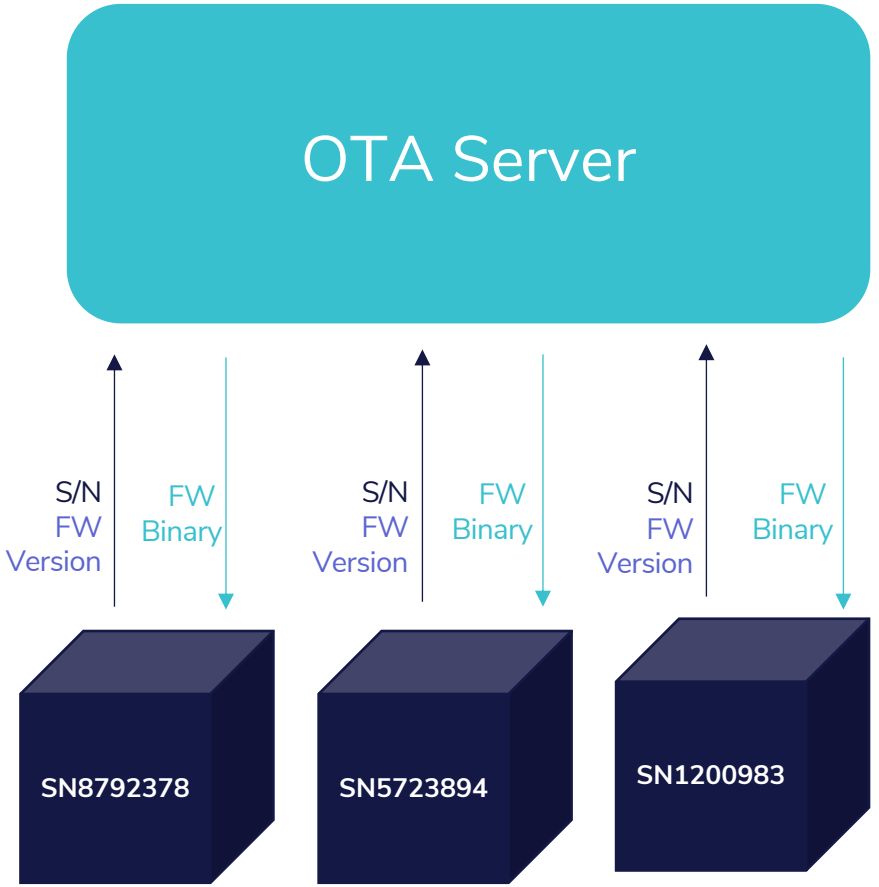
Second best thing to catching a bug before launch, is to **limit the blast radius**

1. Split your fleet into several deployment cohorts and manually deploy to them one at a time
2. *Better Option:* automatic



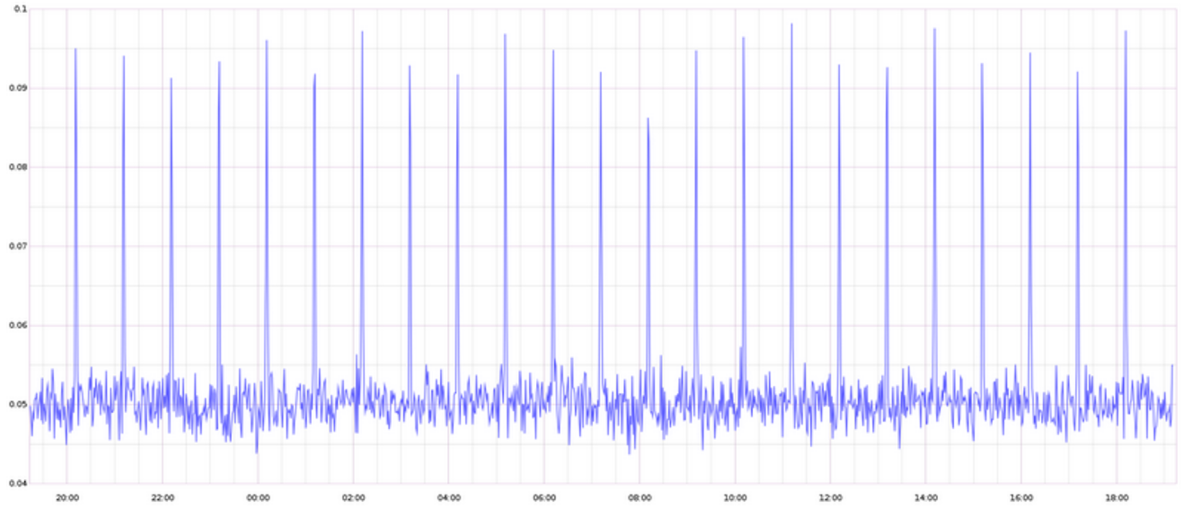
Must-pass-through Releases

- Some complex migrations are not forward compatible and may require a step through for devices that were not updated recently
  - E.g., upgrading from 1.2 to 3.8 might require 1.2→2.0→3.0→3.8
- We cannot do this with statically hosted binaries alone



Global Firmware Delivery

- Requires global infrastructure to do quickly and efficiently
- Watch out for bandwidth costs!
  - AWS S3: $0.05-$0.10 per GB
  - Cloudfront: $0.02-$0.17 per GB
- Add noise / spread out requests to avoid thundering herd problem



Global Firmware Delivery

- Requires global infrastructure to do quickly and efficiently
- Watch out for bandwidth costs!
  - AWS S3: $0.05-$0.10 per GB
  - Cloudfront: $0.02-$0.17 per GB
- Add noise / spread out requests to avoid thundering herd problem

Pulling an Update

- Inevitably, you'll need to **pause or cancel** a rollout in progress
- Ideally, this is a single button click, even better-**automated** based on error rates
- Make sure caches in your infrastructure do not continue serving a bad update

Memfault for OTA

- ✓ Global CDN at bulk bandwidth price
- ✓ Cohorts
- ✓ Must-pass-through releases
- ✓ Staged Rollouts
- ✓ Release scheduling
- ✓ Single button abort

... and more!

Trusted By

logitech

proxy

WHOOP

ŌURA

PANIC

skip

Memfault

Connect with François

@baldassarifr

https://www.linkedin.com/in/francois-baldassari/

francois@memfault.com

https://interrupt.memfault.com

Interrupt Blog Resources

- Device Firmware Cookbook
- Secure Firmware Updates with Code Signing
- From Zero to main(): How to Write a Bootloader from Scratch

Memfault