Memfault

De-Risk Product Launches with Device Reliability Engineering

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

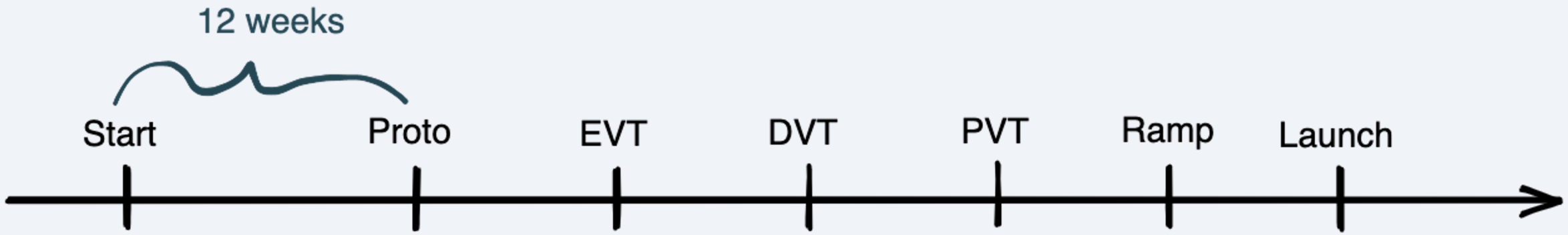
Agenda

- Shipping on Time
- De-Risking Launch
- Q & A

Shipping on time

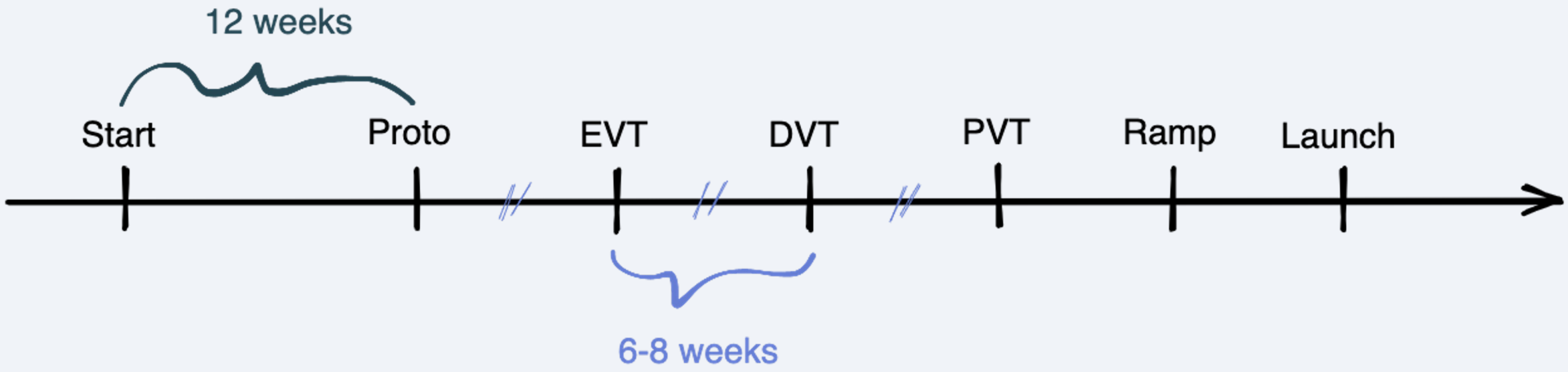
NPI Timeline

NPI Timeline



Proto: Figure out what you want to build

NPI Timeline

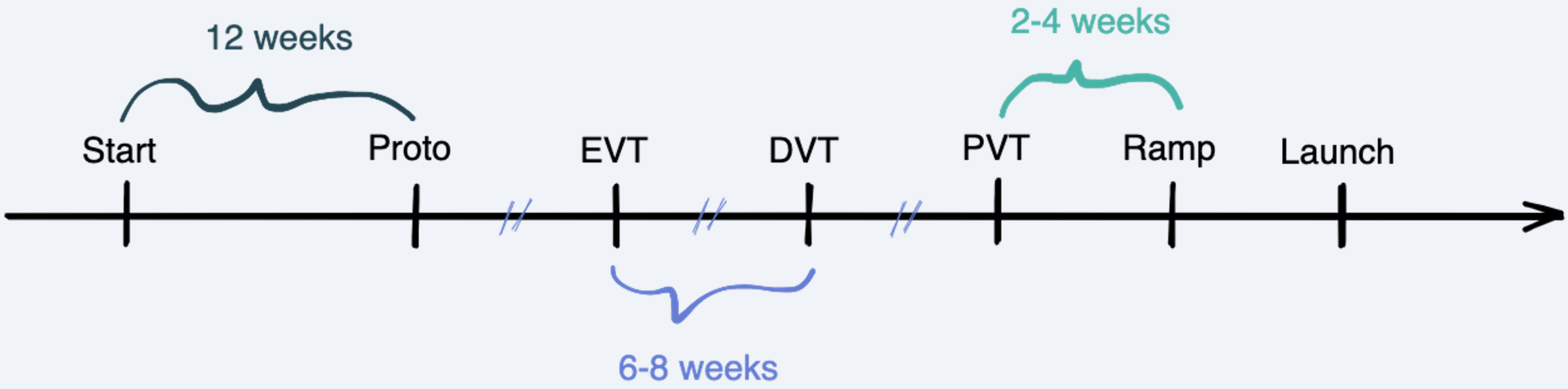


EVT: A handful of configurations, engineering design finalized

DVT: One final configuration, all manufacturing stations pass

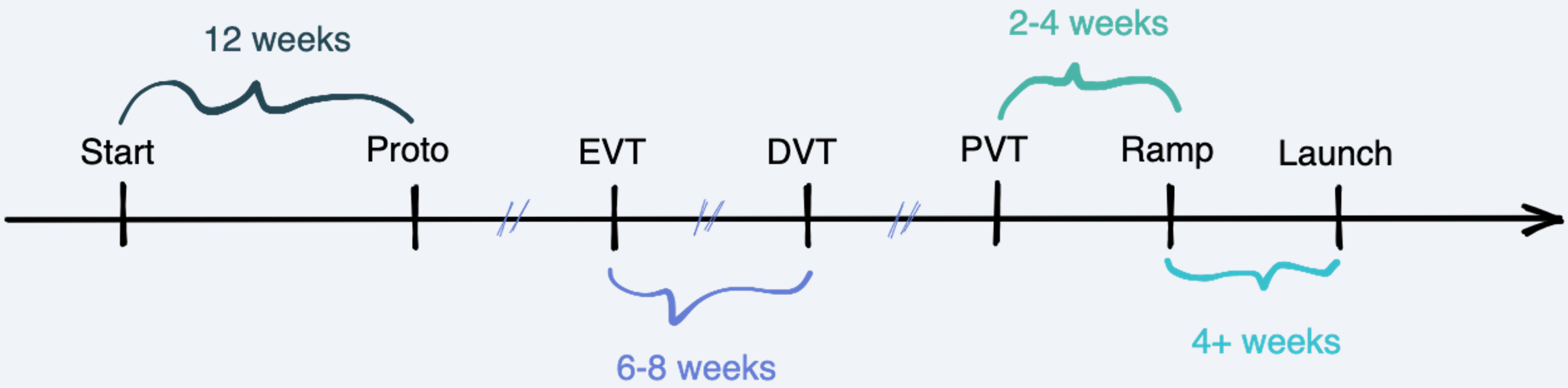
PVT: Manufacturing line operates at yield & speed

NPI Timeline



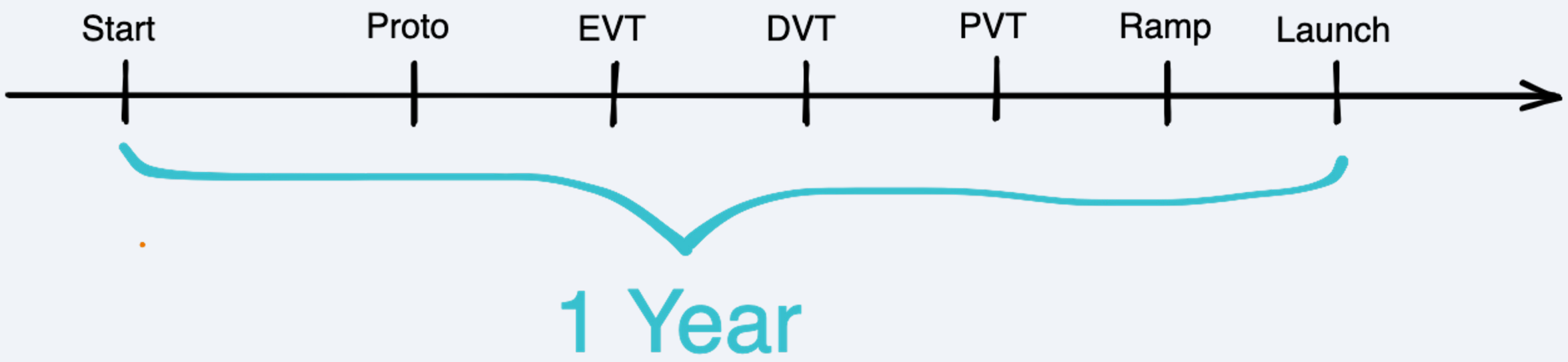
Ramp: Full scale manufacturing, start accumulating inventory for launch

NPI Timeline



Launch 🍾: Devices on shelves, available for purchase

NPI Timeline



Poll #1

How long did NPI take on your last product?

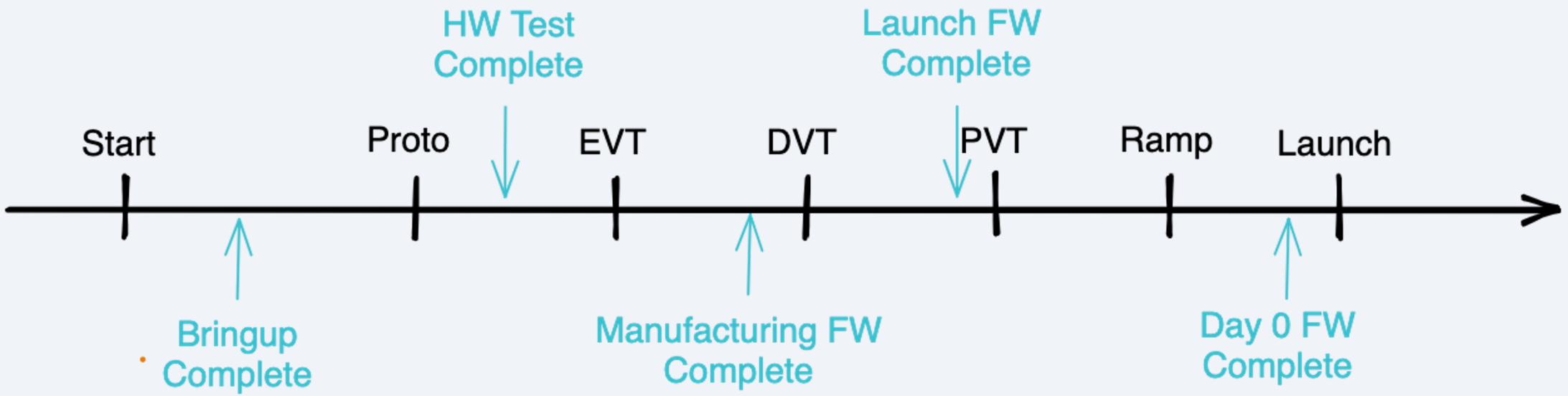
A. <= 12 months

B. <= 18 months

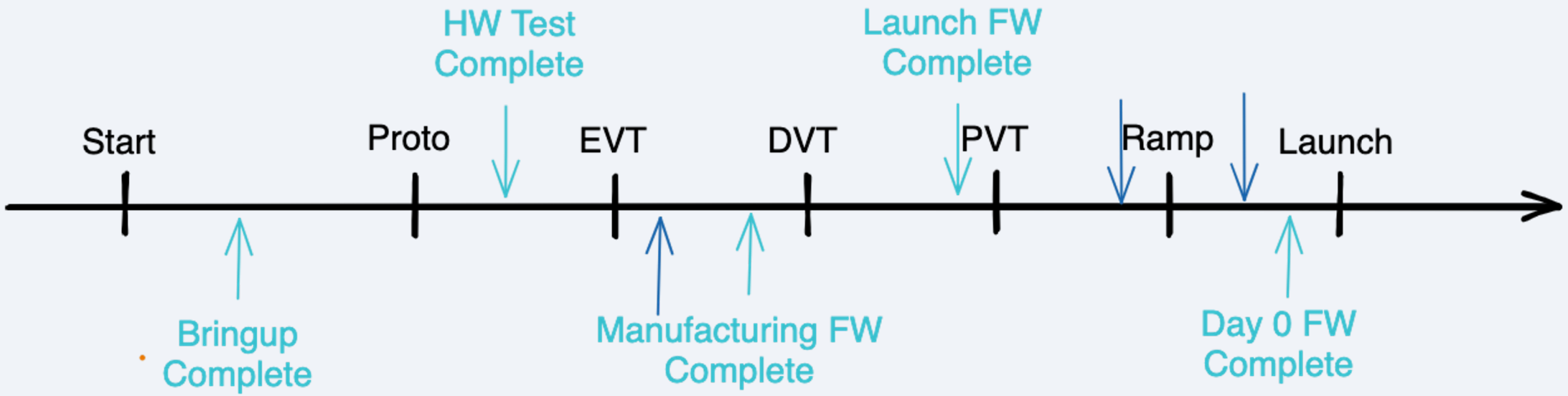
C. <= 2 years

D. > 2 years

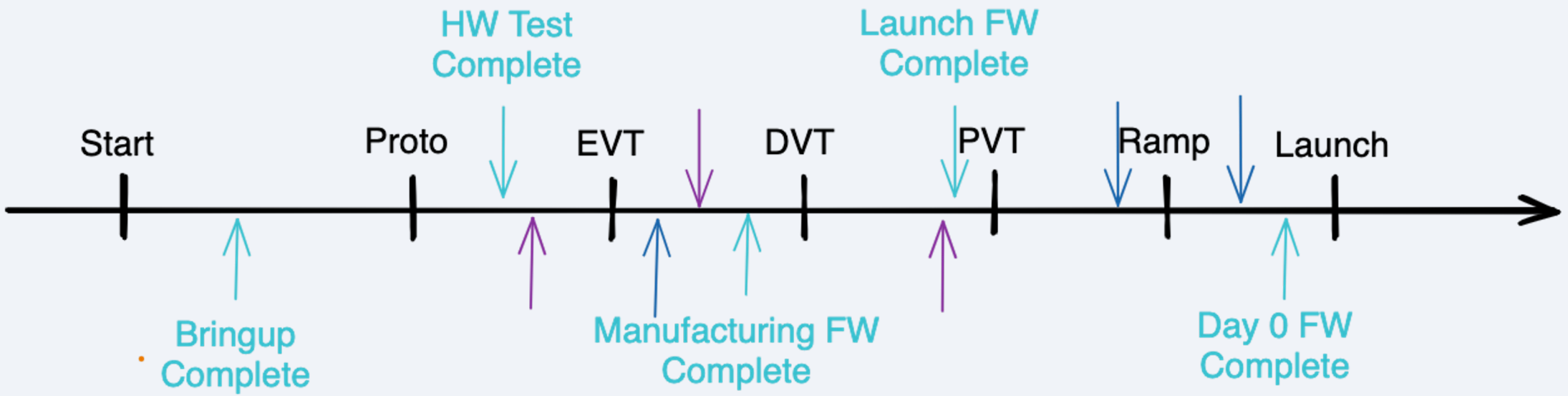
What about firmware?



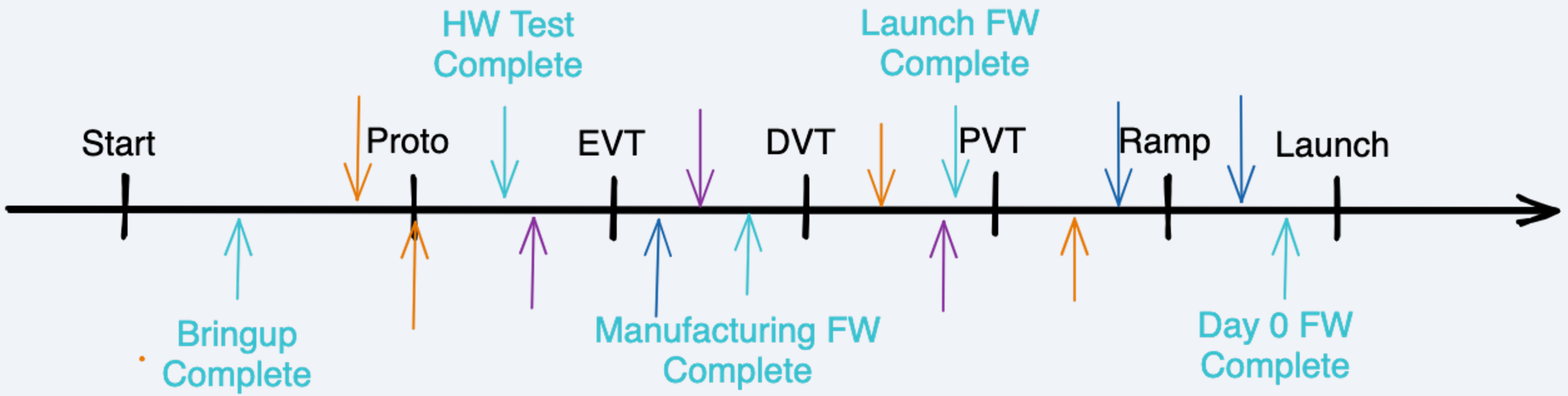
What about marketing?



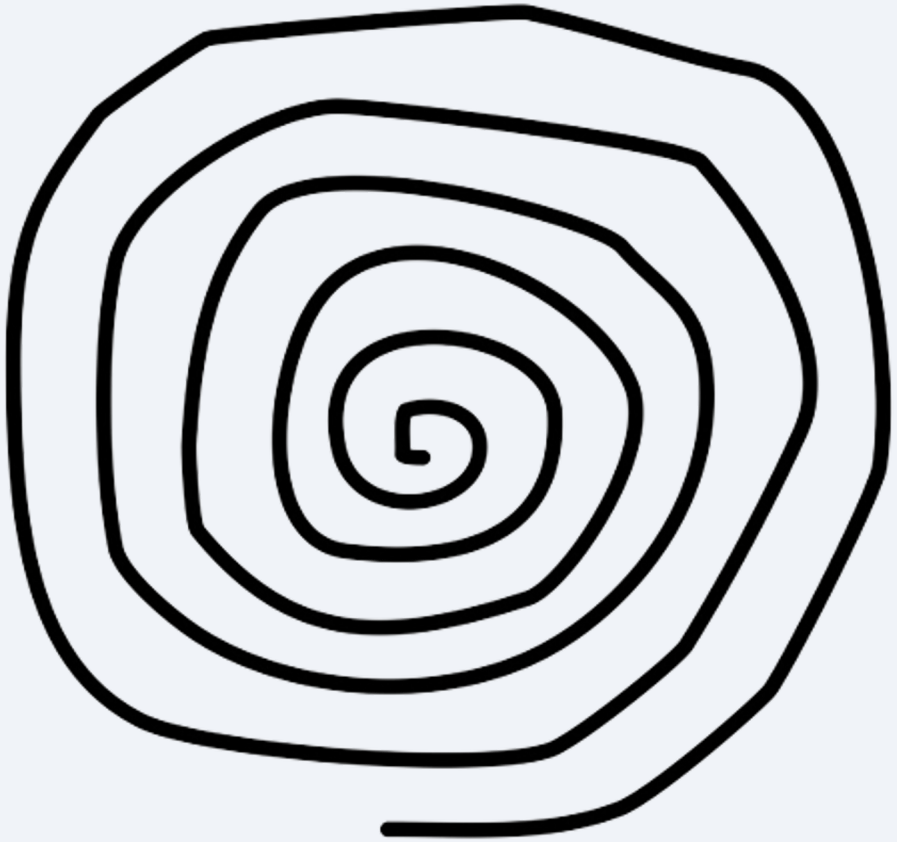
What about factory automation?



What about cloud software?



Avoid a dependency spiral



Decoupling SW & HW Timelines

1. Test Driven Development
2. Day-0 Updates
3. Hardware Abstraction Layer
4. Splitting Manufacturing and App Firmware

Test-Driven Development

What it is

Building firmware against a software test harness rather than real hardware. This can include the use of unit testing frameworks (e.g. CppUTest) and simulators (e.g. Renode).

Learn more

- https://interrupt.memfault.com/blog/unit-testing-basics
- https://interrupt.memfault.com/blog/intro-to-renode

Why Do It

- Allows for development to proceed before hardware is ready
- Faster iteration speed
- Creates a robust set of tests which can be reused to support development

Day-0 Update

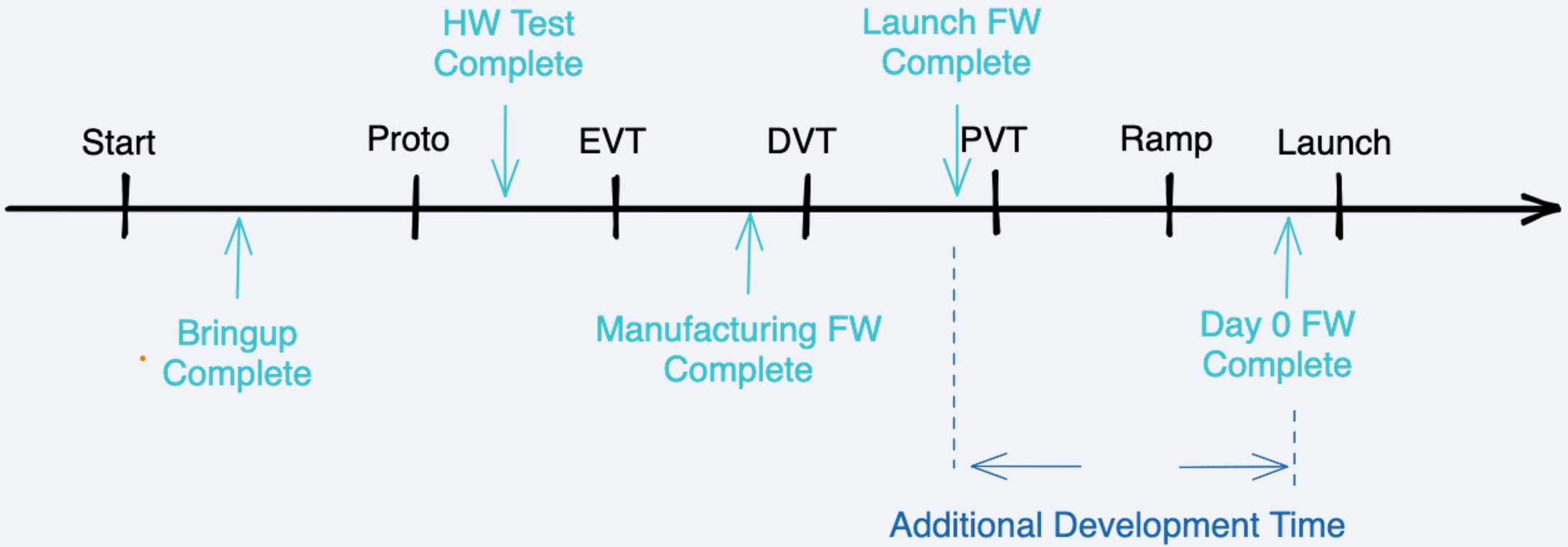
What it is

Preparing a software update applied to the devices at unboxing. This update needs to be ready by the time devices are in customers' hands rather than at manufacturing.

Why Do It

- Decouple dependency between ramp and software GM
- Extend software development schedule by >4 weeks

Day-0 Update


HW Test Complete
Launch FW Complete
Start
Proto
EVT
DVT
PVT
Ramp
Launch
Bringup Complete
Manufacturing FW Complete
Day 0 FW Complete
Additional Development Time

A Strong HAL

What it is

Use a cross-platform operating system and hardware abstraction layer that can easily be ported to new hardware. The Zephyr project is an excellent option with strong backing from semiconductor and device manufacturers.

Learn more

- https://www.zephyrproject.org/

Why Do It

- Decouple firmware from the underlying hardware
- Create optionality in the event of supply chain constraints
- Lay the ground for code re-use on future programs

Splitting Manufacturing and App Firmware

What it is

Use a purpose built firmware on the manufacturing line which changes very rarely and is completely separate from the application firmware. Load the app firmware at the last test station on the line.

Why Do It

- Iterate on the application FW without impacting the manufacturing FW
- Continue working on app FW after DVT when factory processes are locked
- Save code space

But!!

Watch out for dependencies between app and manufacturing firmware (e.g. sensor configuration.

De-risking Launch

Congratulations, you've launched!

Not so fast...

- Bugs
- RMAs
- Security Issues
- Missing Features
- Customer Complaints

This Will Happen to You!

- Ganssle: “10-100 defects per 1000 lines of code”
- Some of these issues will be severe, some will be security flaws
- Law of large numbers: some issues will only be found in production

> “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.”

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

De-Risk with Device Reliability Engineering

Robust OTA

Performance Monitoring

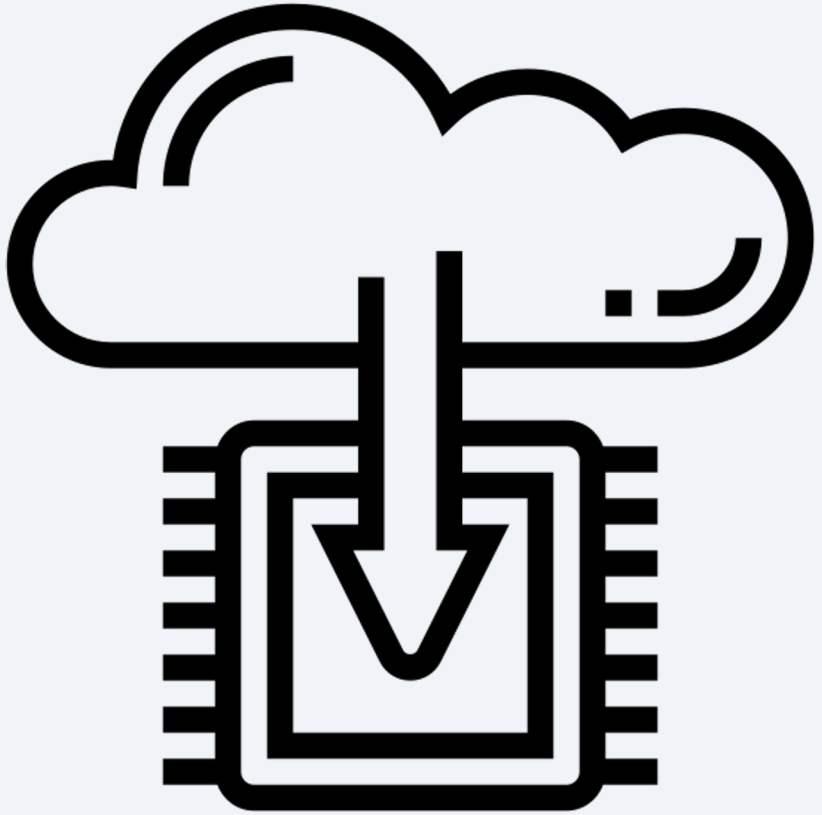
Remote Debugging

Robust OTA

OTA is your insurance policy against issues

It needs excellent test coverage!

At the very least, your system should support **cohorts**, **staged rollout**, and must-pass-through releases



Cohorts

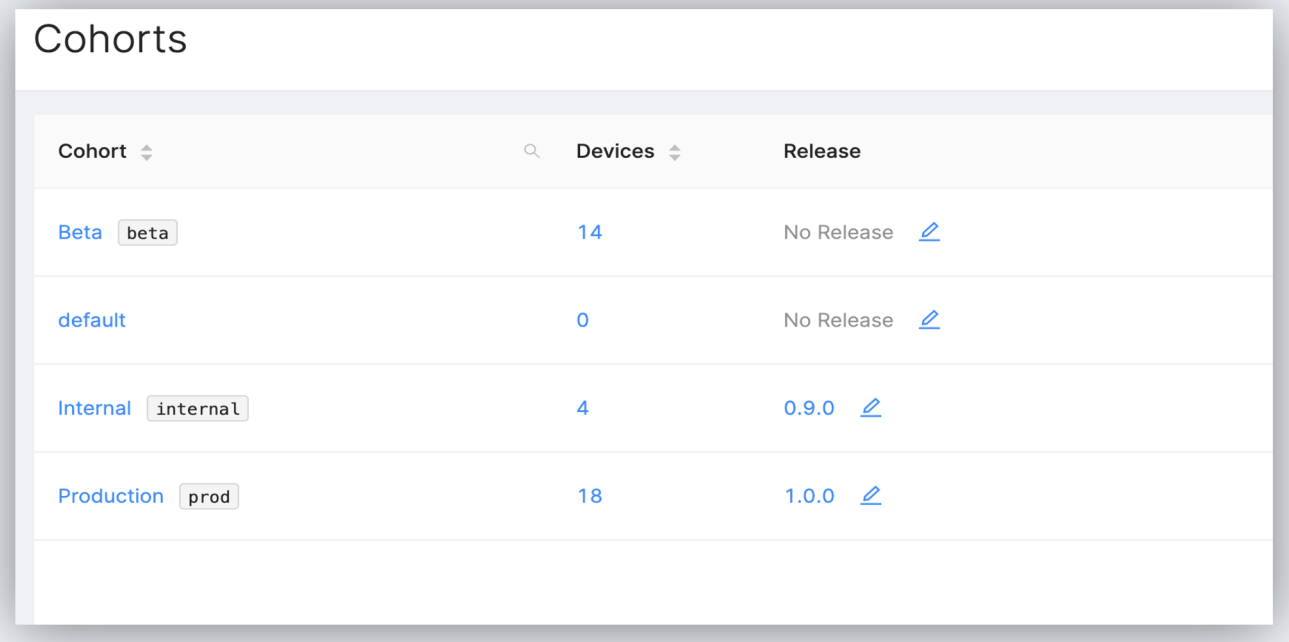
What it is

Grouping your devices, and updating each group separately

Why You Need It

Cohorts are a simple way to enable beta tests, A/B tests, and other forms of experimentation

Cohorts with Memfault:



Staged Rollouts

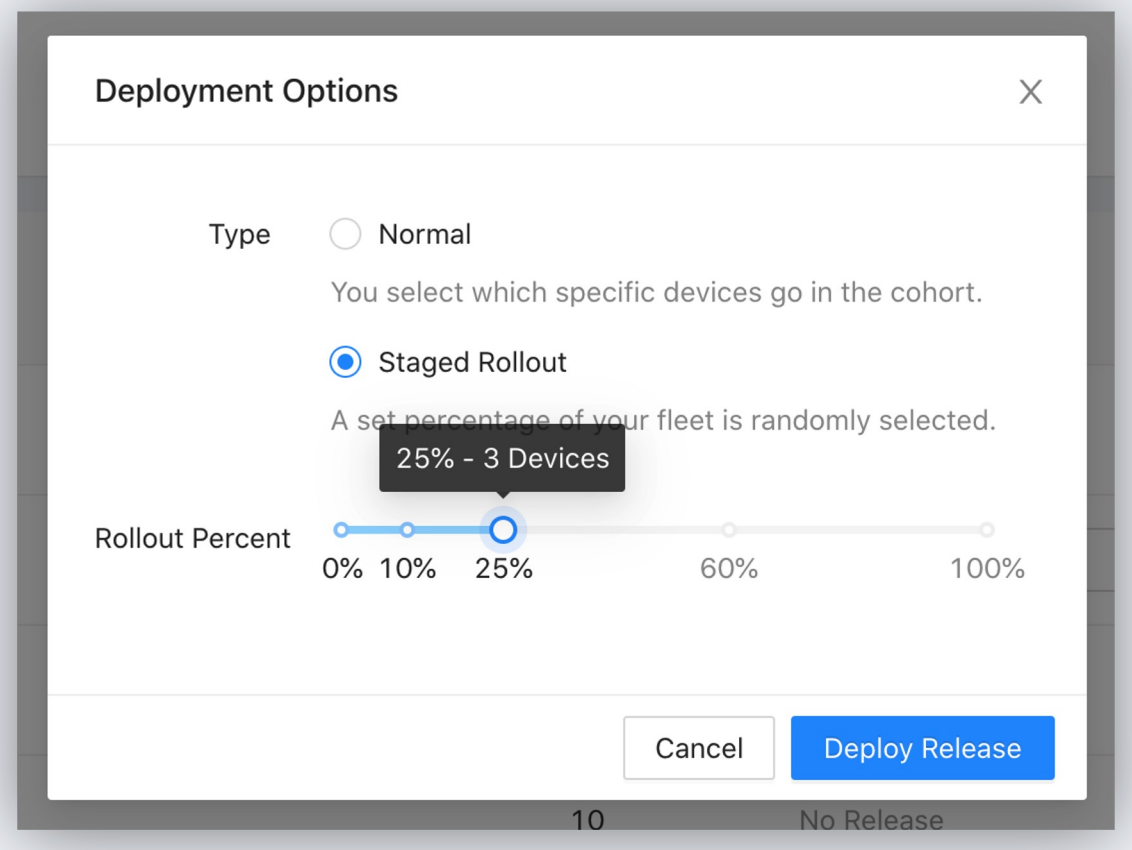
What it is

The ability to roll out a new release to an incrementally larger subset of the fleet.

Why You Need It

Every release introduces risk. By rollout out updates incrementally, you limit the blast radius of any new issue that comes up.

Staged rollouts with Memfault:



Must-Pass Through

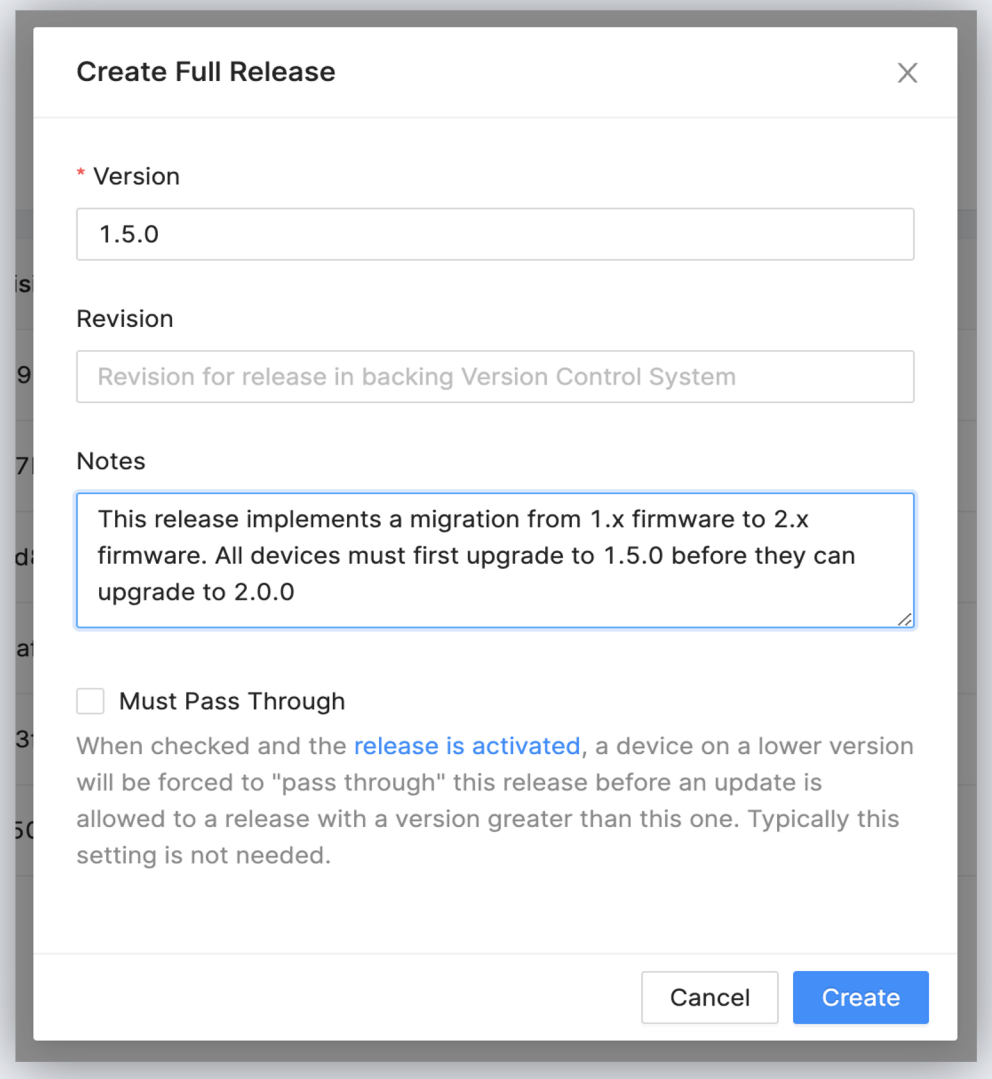
What it is

A release which must be loaded on the device before future releases can be installed.

Why You Need It

Some complex migrations may not be forward compatible. For example, upgrading from 1.2 to 3.8 might require multiple steps:
1.2 → 2.0 → 3.0 → 3.8

Must-pass-through with Memfault:



Performance Metrics

“How are my devices doing?”

- Connectivity
- Battery Life
- Memory Usage
- Sensor Performance
- System Responsiveness

This system must be:

1. Low overhead (no device impact)
2. Easy to extend
3. Privacy preserving

Individual Device Metrics

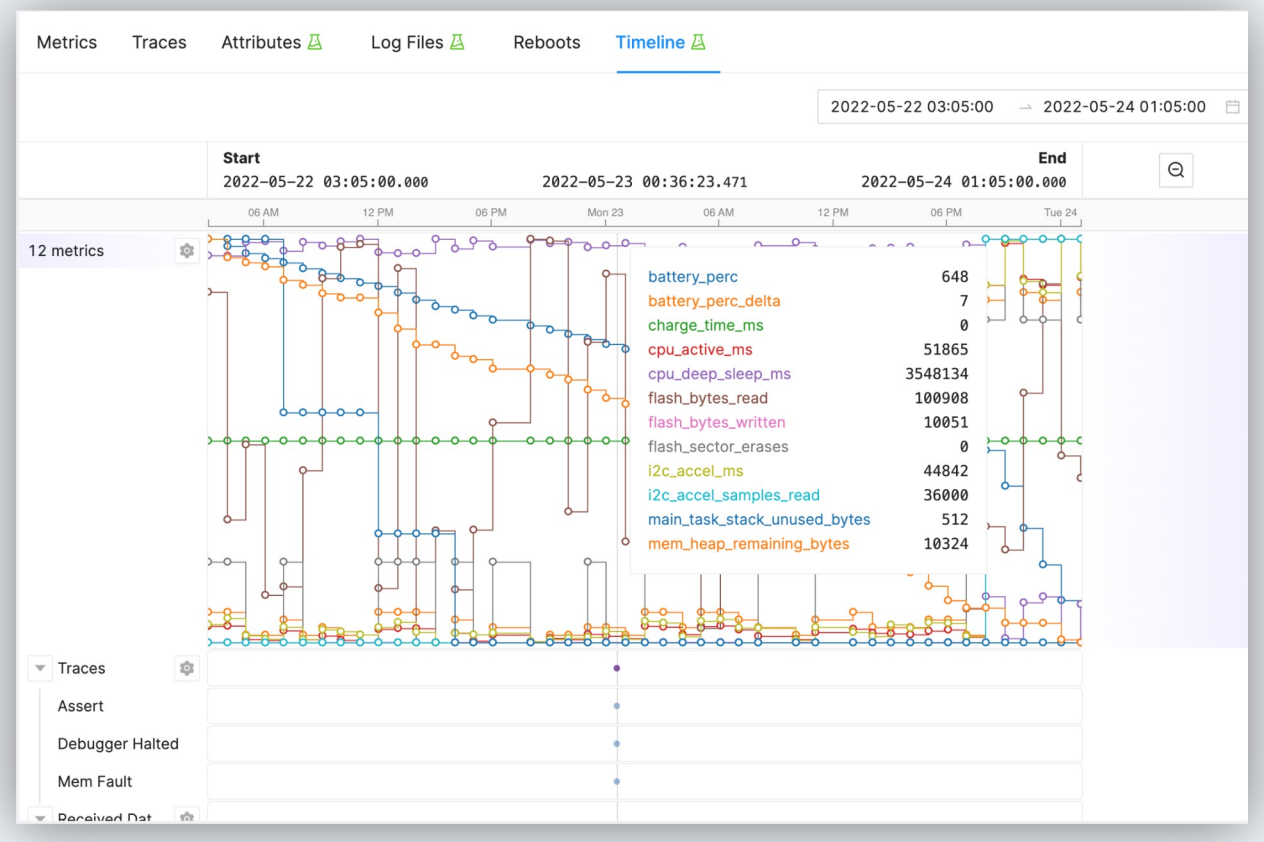
What it is

Collection of datapoints from devices at regular intervals.

Why You Need It

To investigate specific reports of devices misbehaving, either by customer support or engineering teams

Device Metrics with Memfault:



Aggregates and Dashboards

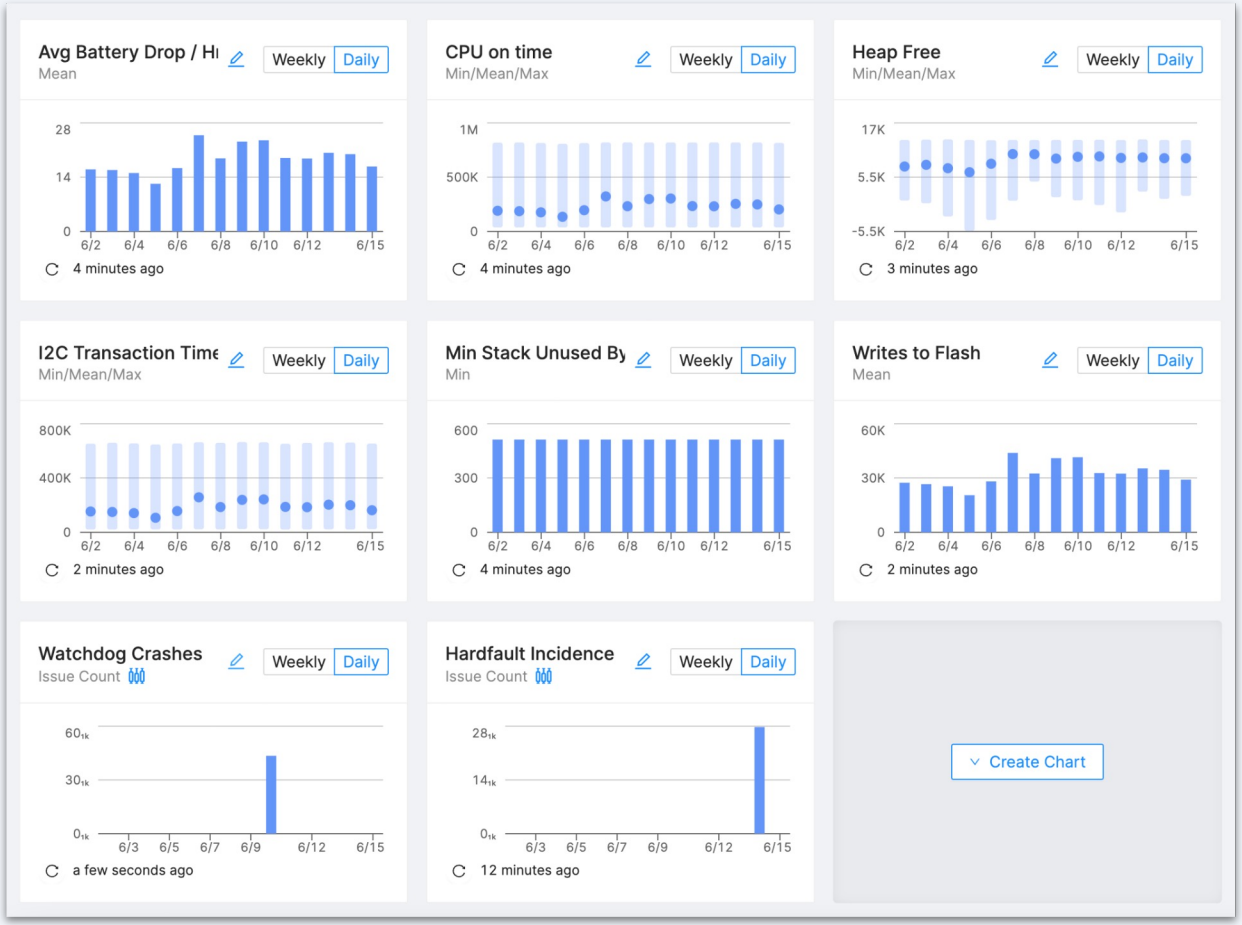
What it is

Dashboards aggregating individual data into high level charts

Why You Need It

To understand overall fleet performance and quickly identify trends in the data

Dashboards with Memfault:



Alerts

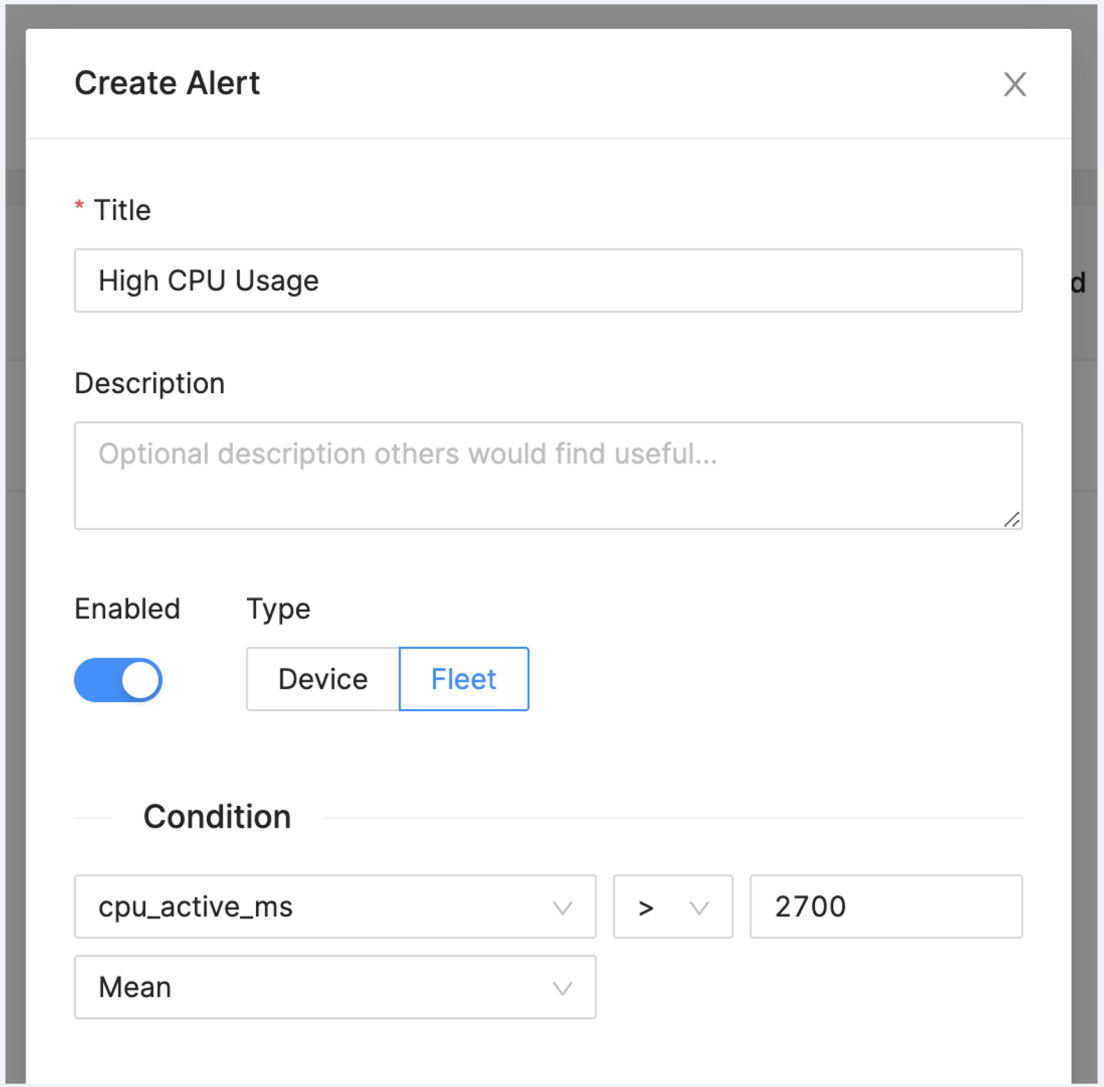
What it is

Alerts to email, slack or incident management platforms when certain conditions are met

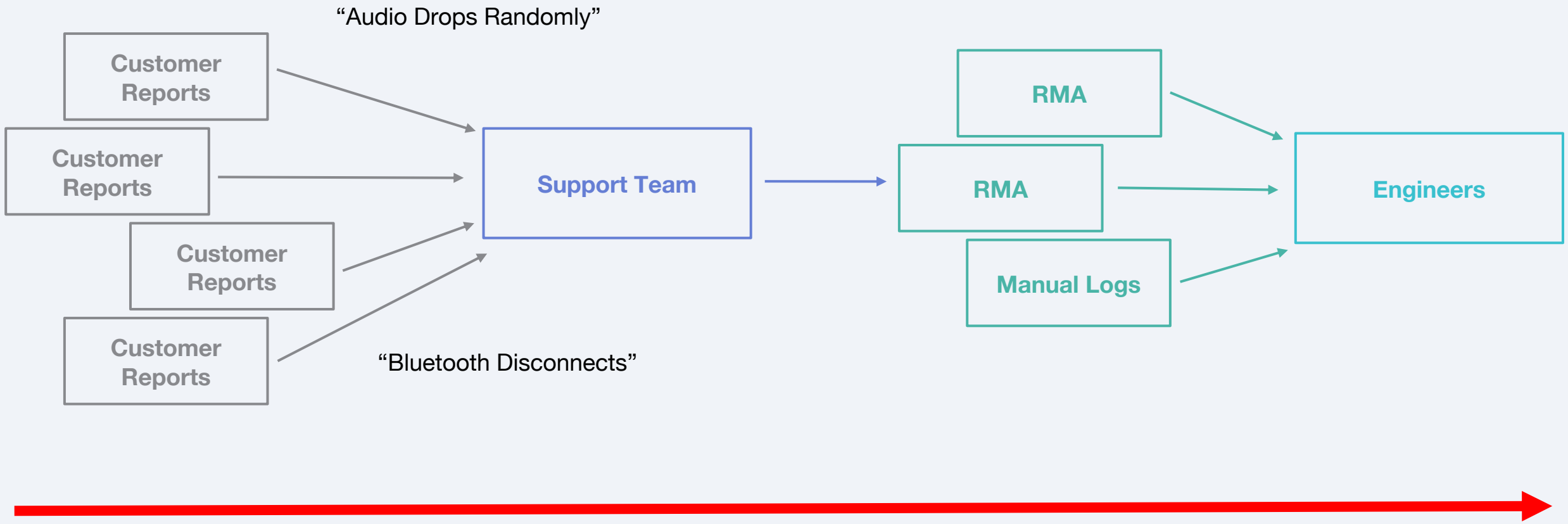
Why You Need It

To bring issues to your attention quickly, rather than wait for the next time you look at the charts

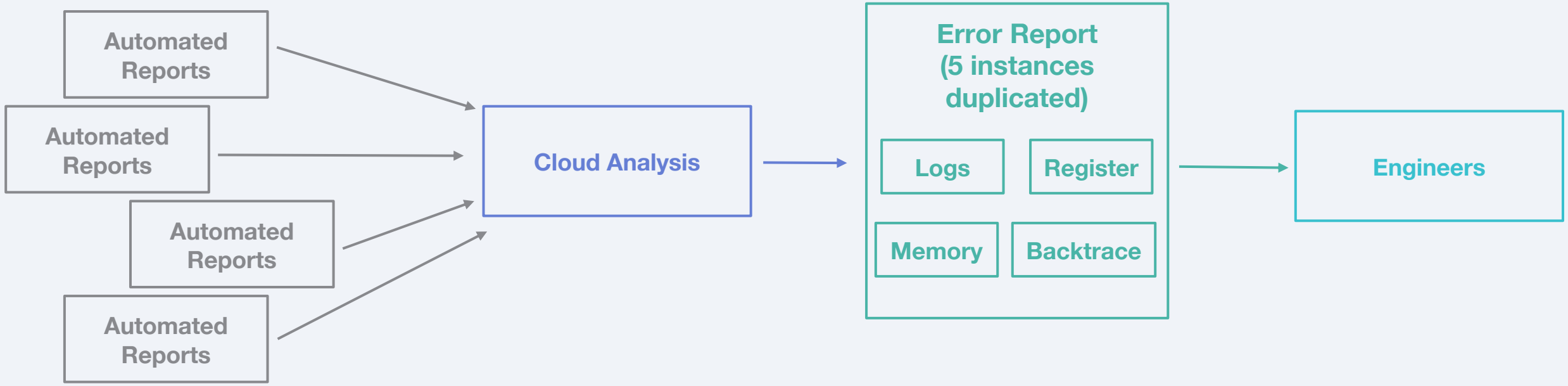
Alerts with Memfault:



Remote Debugging



Remote Debugging



2 minutes

Coredumps

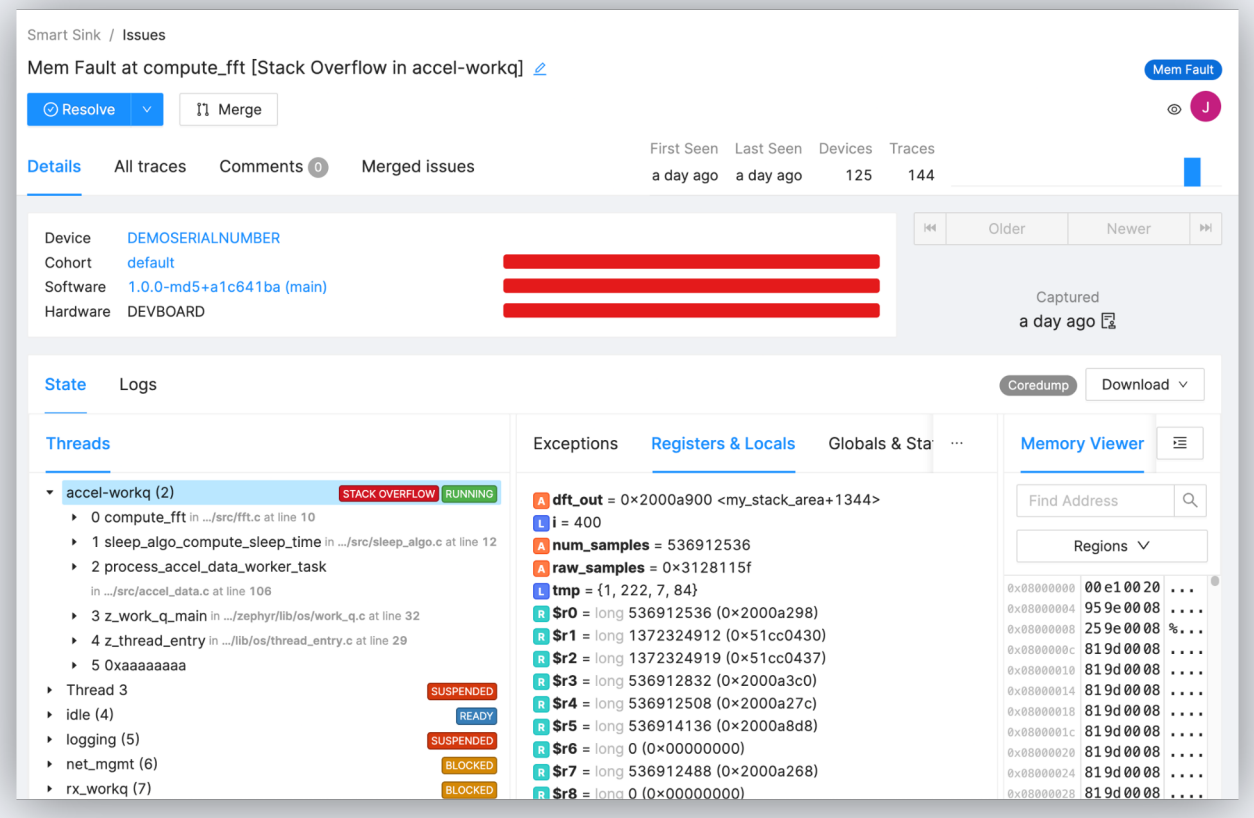
What it is

Automatically collect detailed diagnostics data as soon as an issue occurs

Why You Need It

Give your engineers the information they need to resolve the problem quickly, without an RMA or sending out a technician

Coredumps with Memfault:



Poll #2

Which infrastructure do you have in place today?

Check all that apply…

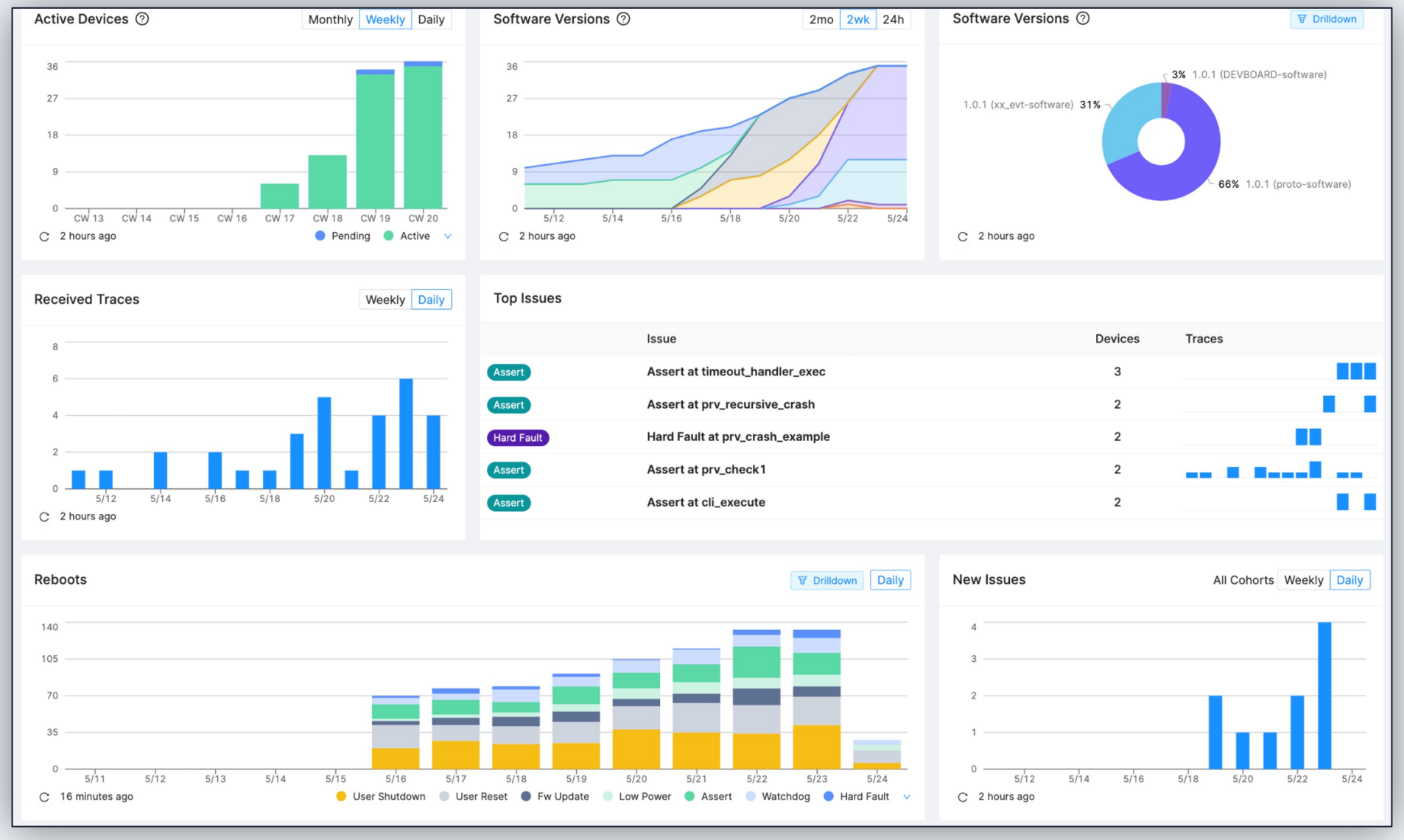
A. OTA

B. Metrics

C. Remote Debugging

D. None of the above

Memfault: IoT Reliability Platform



Q&A

Would appreciate filling out the survey at the end.

It will appear in browser when the webinar is over and we will have it in the follow-up email.

Memfault