

Agenda

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



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)



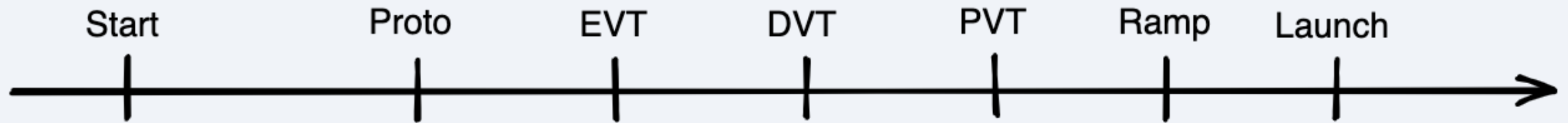
pebble®



Shipping on time

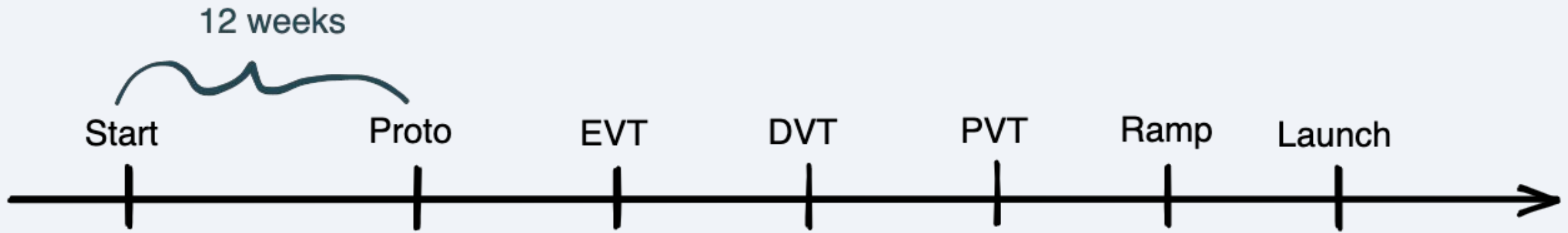


New Product Introduction (NPI) Timeline



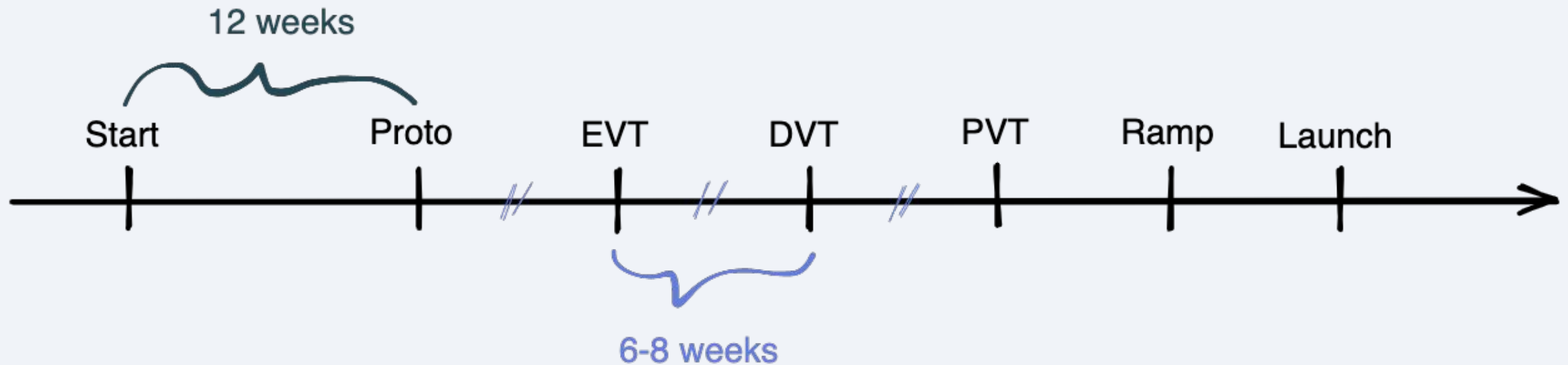
- **EVT** - Engineering Validation Test
- **DVT** - Design Validation Test
- **PVT** - Production Validation Test

NPI Timeline



Proto: Figure out what you want to build

NPI Timeline

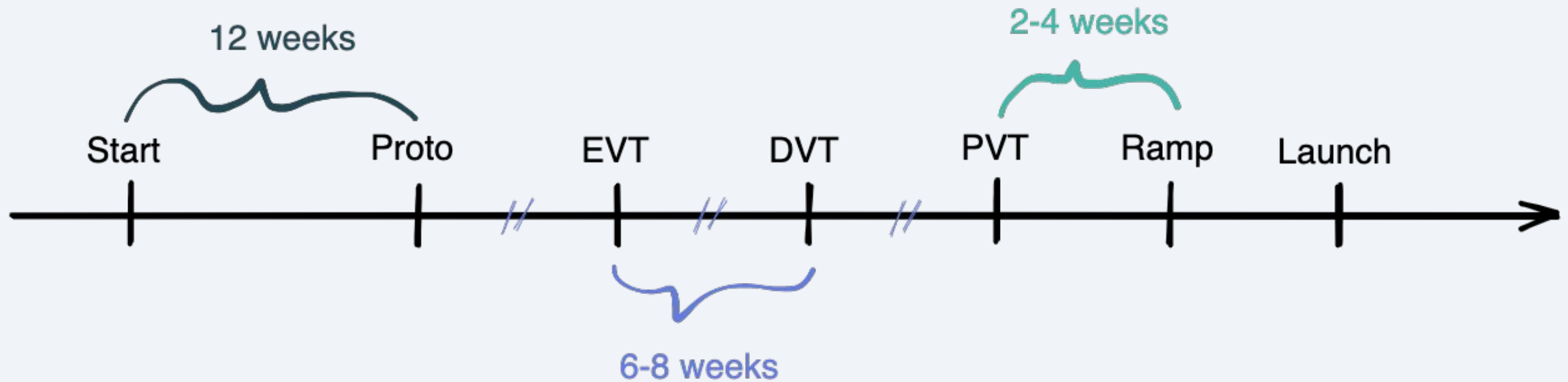


EVT: A handful of configurations, engineering design finalized

DVT: One final configuration, all manufacturing stations pass

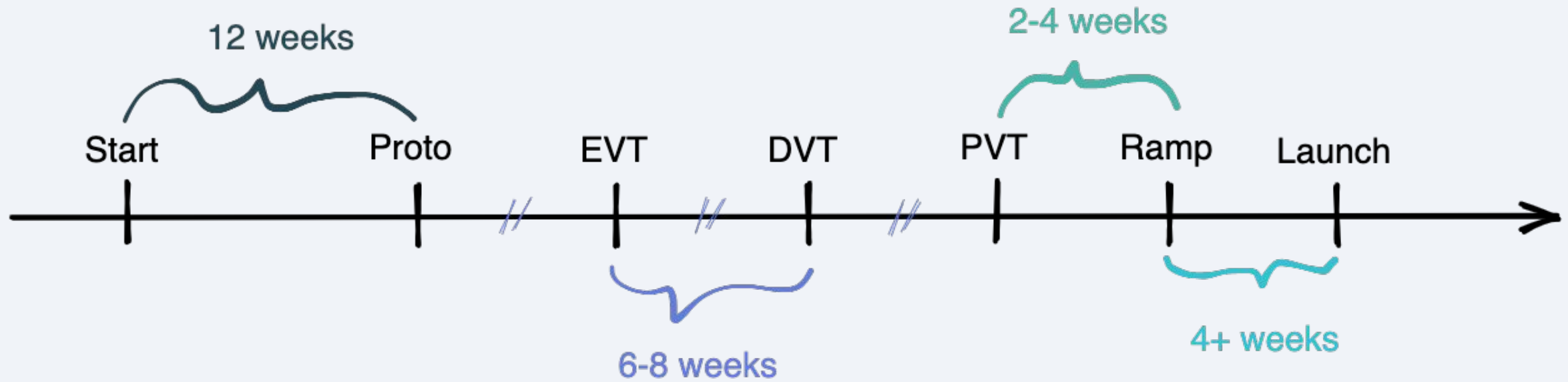
PVT: Manufacturing line operates at yield & speed

NPI Timeline



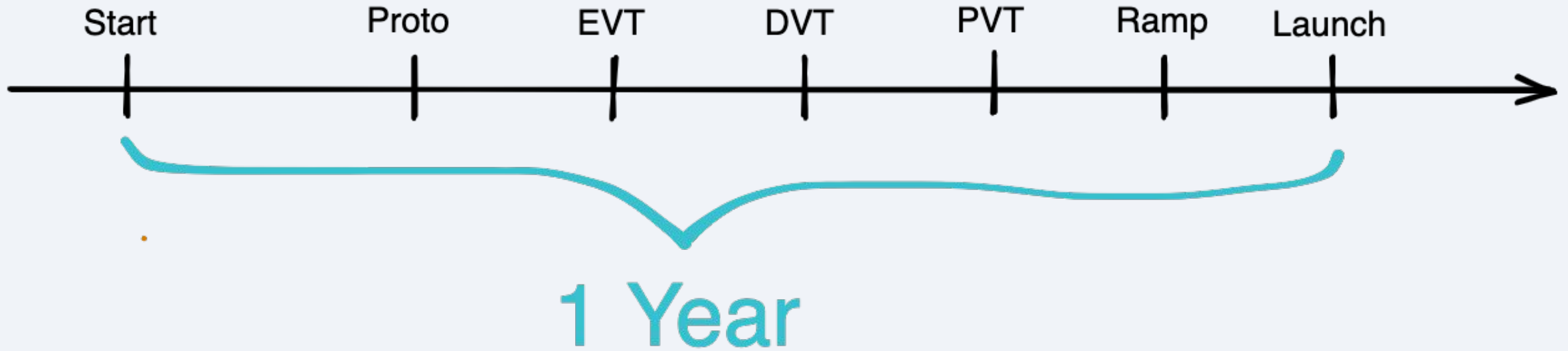
Ramp: Full scale manufacturing, start accumulating inventory for launch

NPI Timeline



Launch 🍷: Devices on shelves, available for purchase

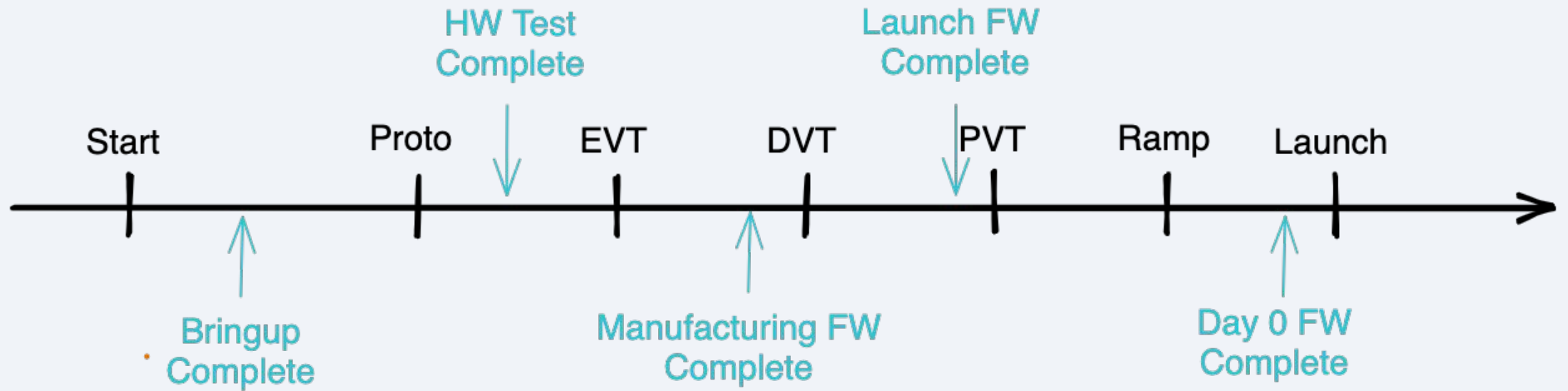
NPI Timeline



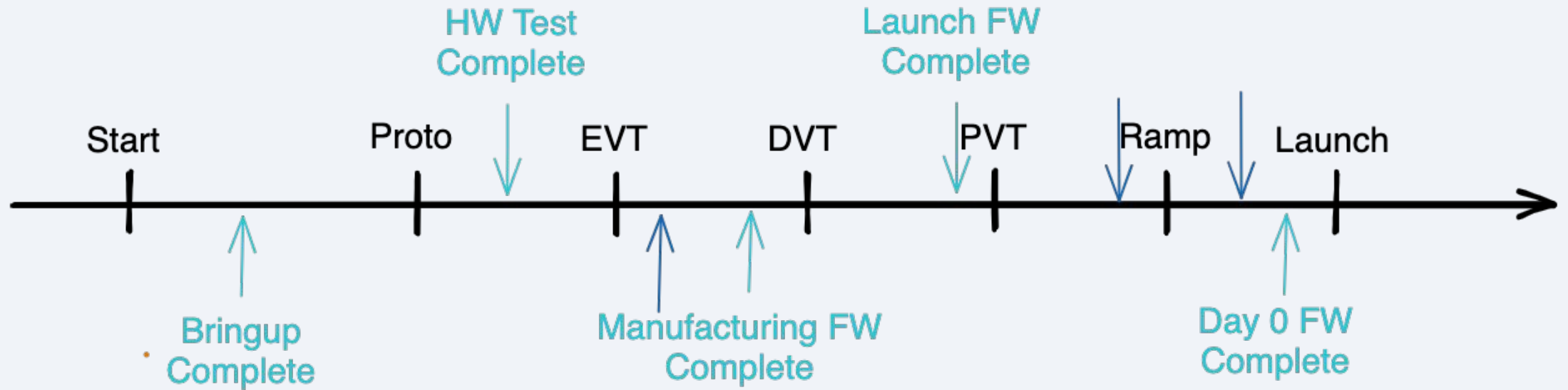


Poll #1

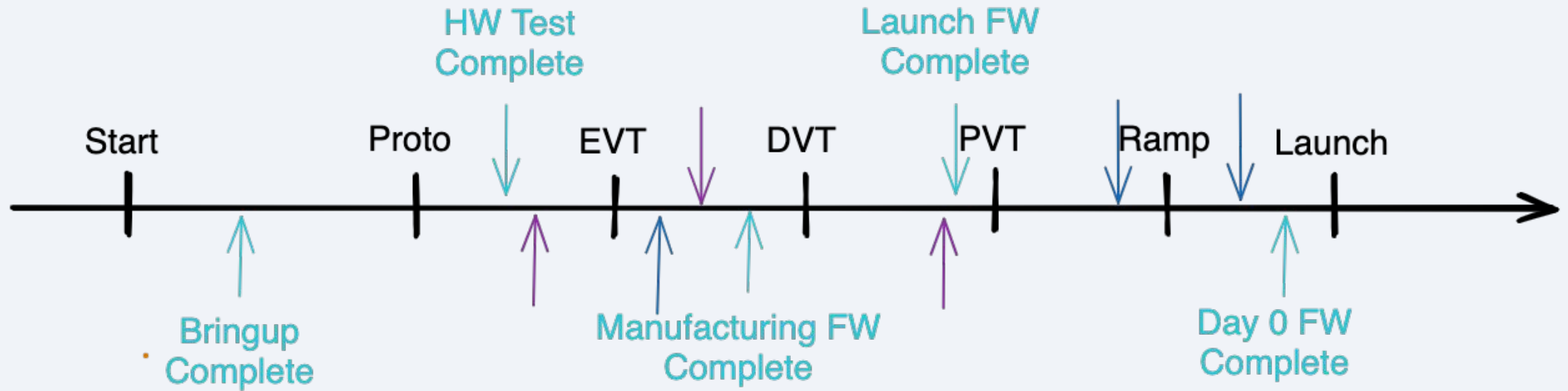
What about firmware?



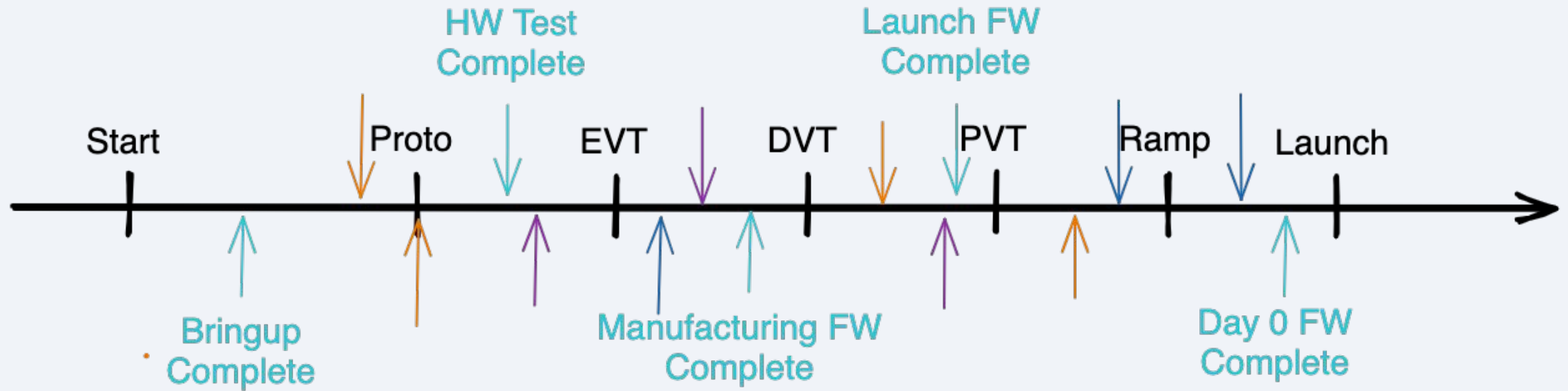
What about marketing?



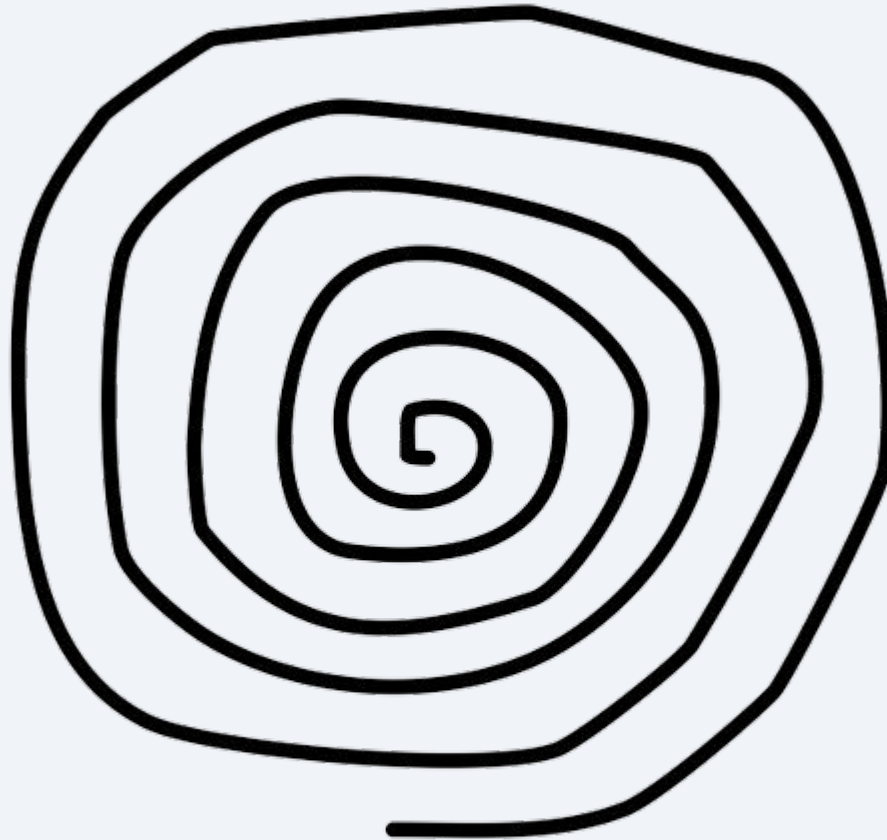
What about factory automation?



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

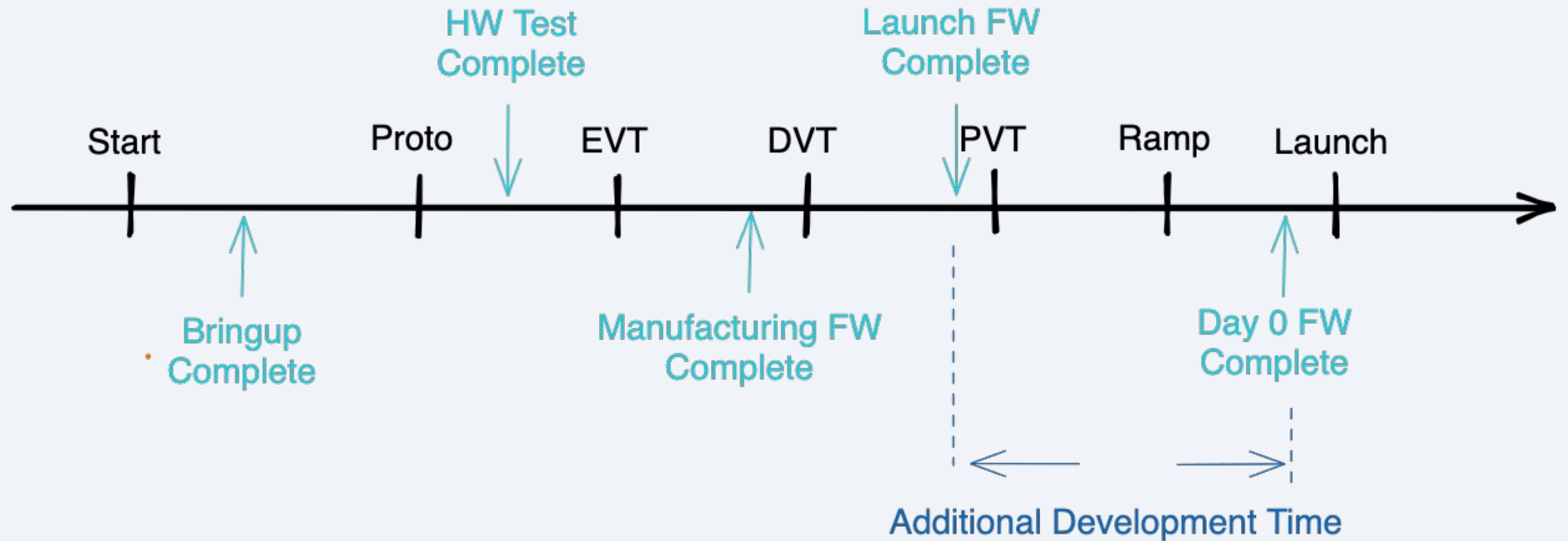
What it is

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



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

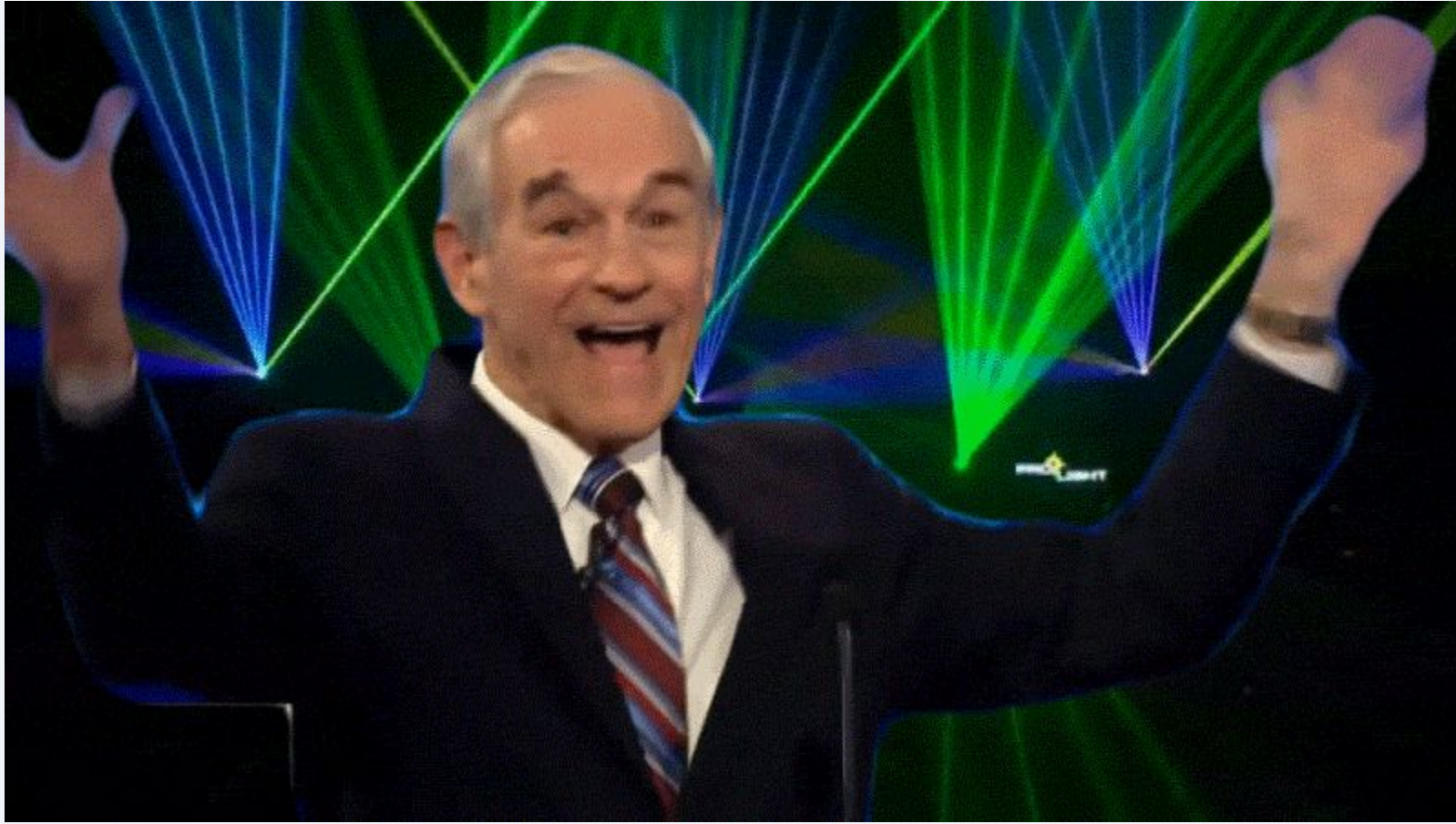
- Iterate on the application FW without impacting the manufacturing FW
- Continue working on app FW after DVT when factory processes are locked
- Save code space

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**

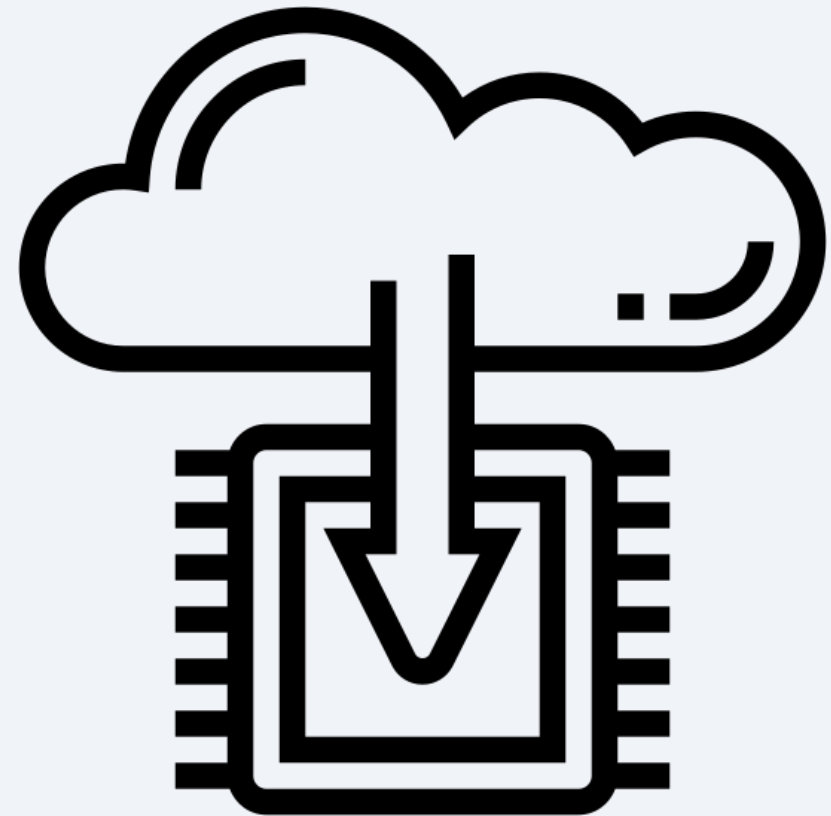


Robust OTA

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:

Cohorts		
Cohort	Devices	Release
Beta beta	14	No Release ✎
default	0	No Release ✎
Internal internal	4	0.9.0 ✎
Production prod	18	1.0.0 ✎

Staged Rollouts

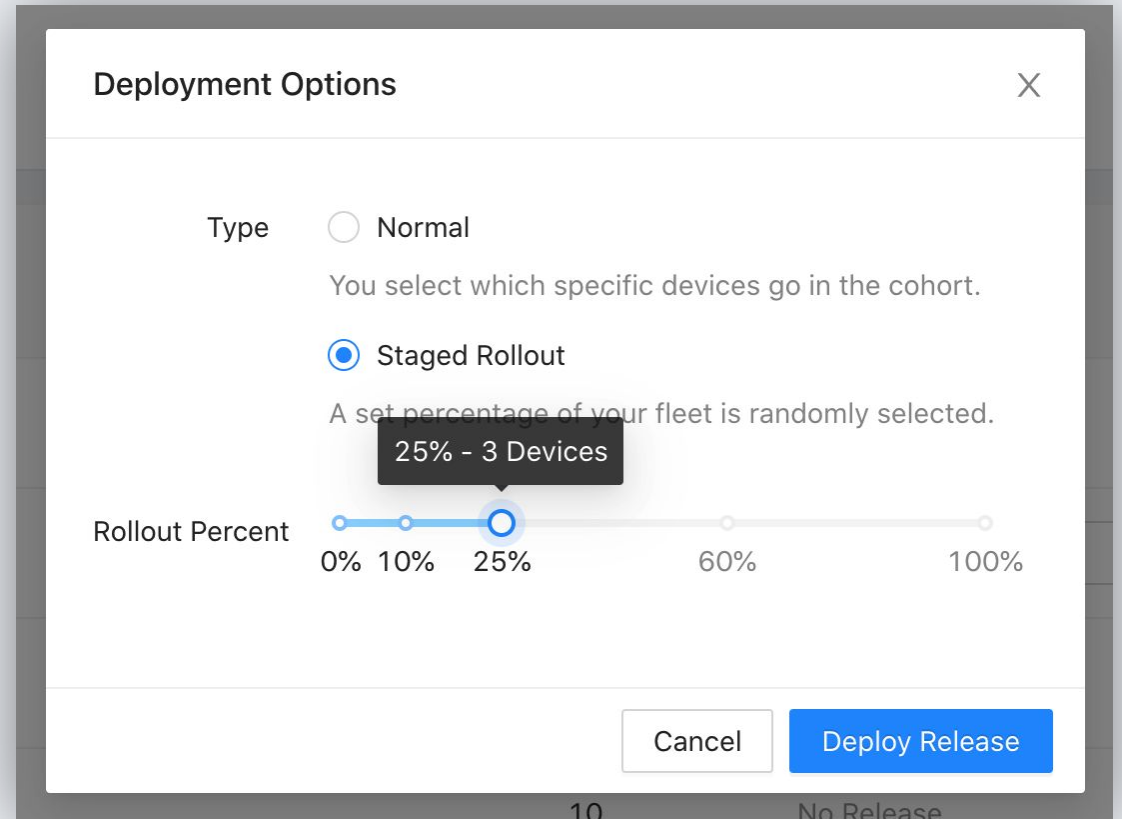
What it is

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:



The screenshot shows a 'Deployment Options' dialog box with a close button (X) in the top right corner. Under the 'Type' section, the 'Staged Rollout' option is selected with a blue radio button. Below it, a description reads: 'A set percentage of your fleet is randomly selected.' A slider for 'Rollout Percent' is positioned at 25%, with a tooltip above it indicating '25% - 3 Devices'. The slider has markers at 0%, 10%, 25%, 60%, and 100%. At the bottom right, there are two buttons: 'Cancel' and 'Deploy Release'.

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:

Create Full Release

* Version

1.5.0

Revision

Revision for release in backing Version Control System

Notes

This release implements a migration from 1.x firmware to 2.x firmware. All devices must first upgrade to 1.5.0 before they can upgrade to 2.0.0

Must Pass Through

When checked and the [release is activated](#), a device on a lower version will be forced to "pass through" this release before an update is allowed to a release with a version greater than this one. Typically this setting is not needed.

Cancel Create

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

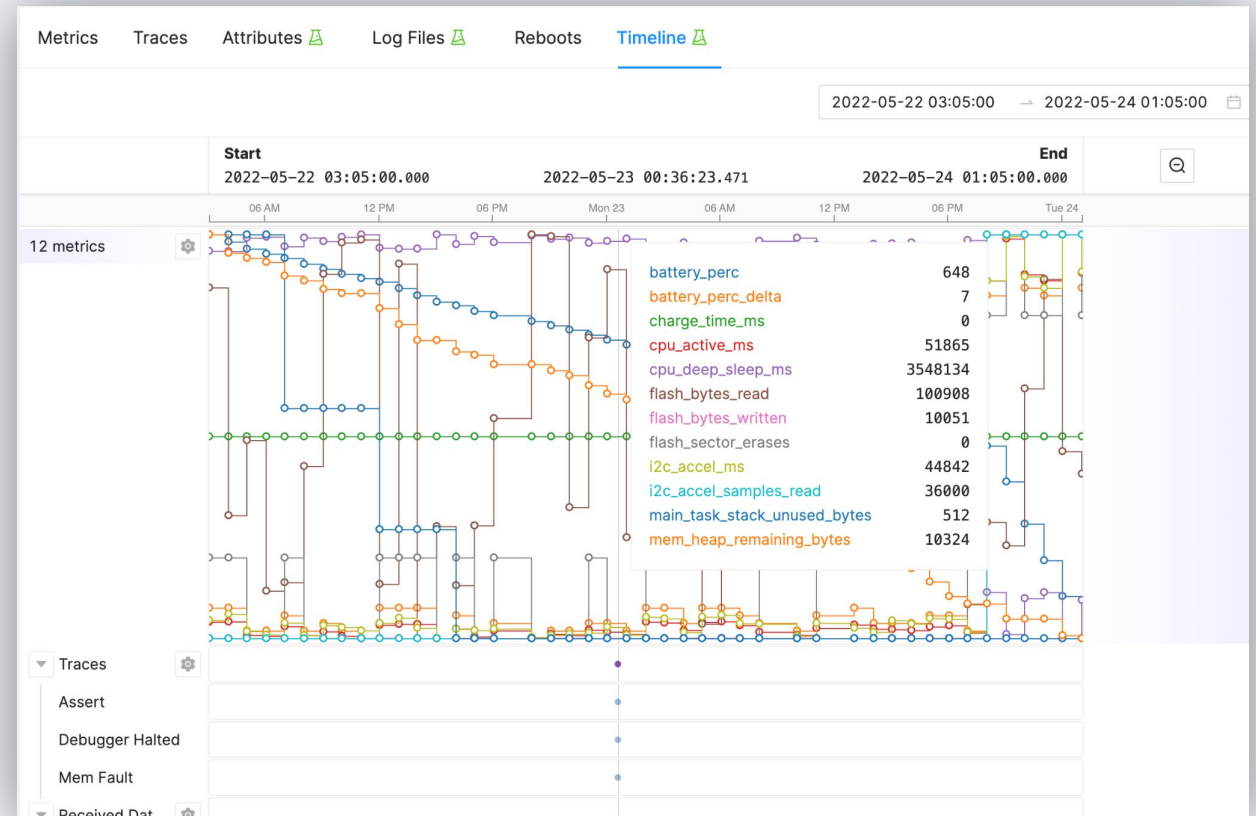
What it is

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

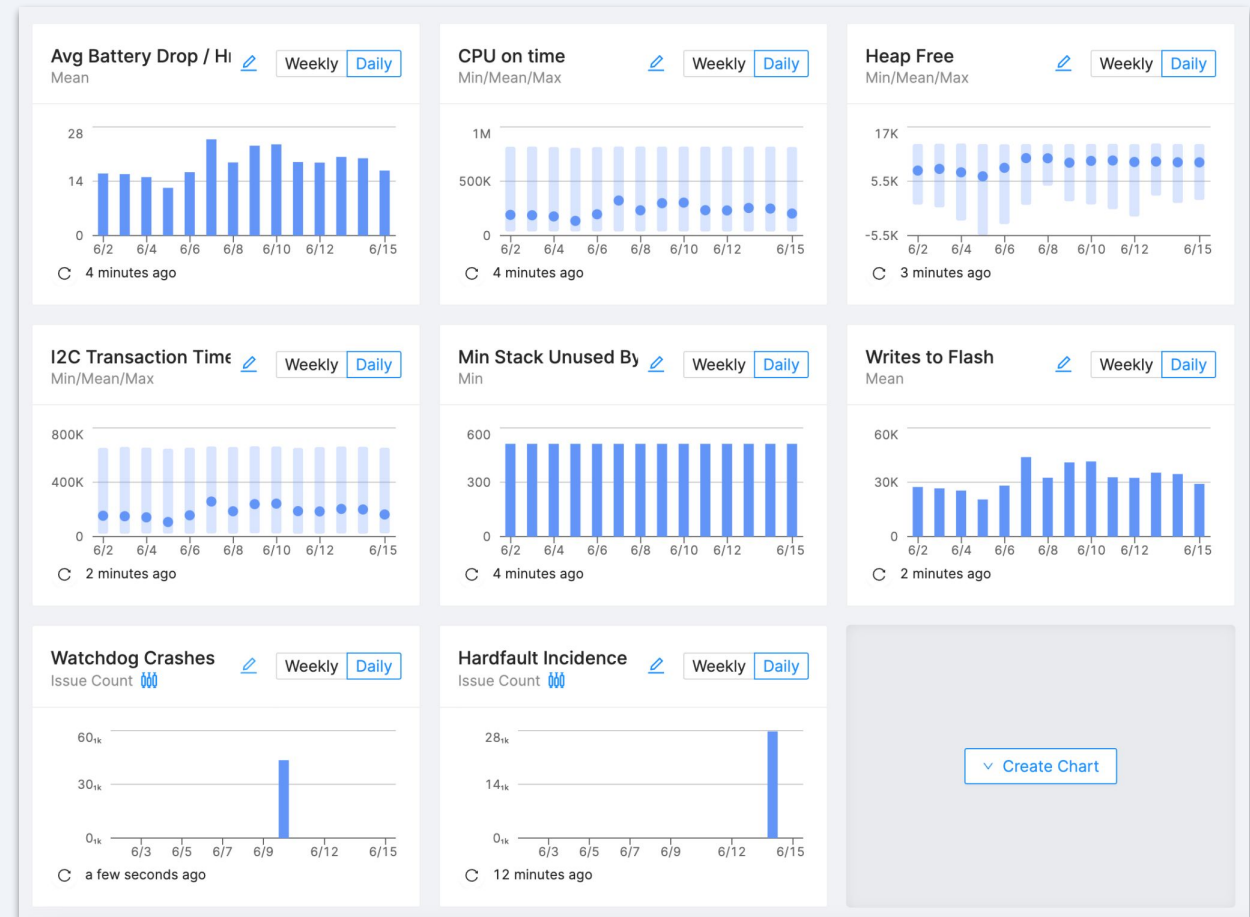
What it is

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:

Create Alert ✕

* Title

Description

Enabled

Type

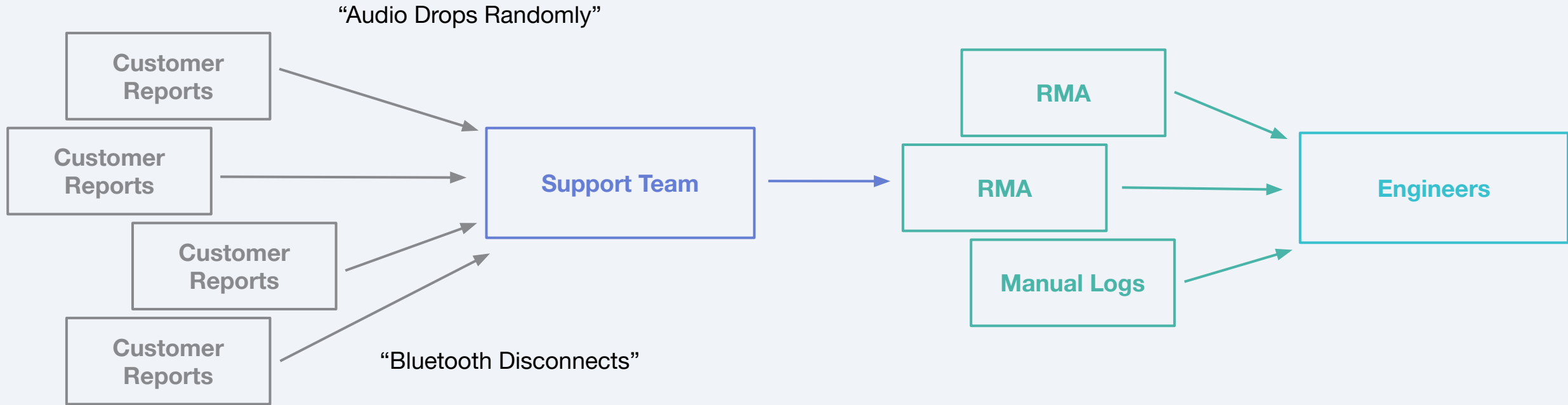
Device Fleet

Condition

cpu_active_ms > 2700

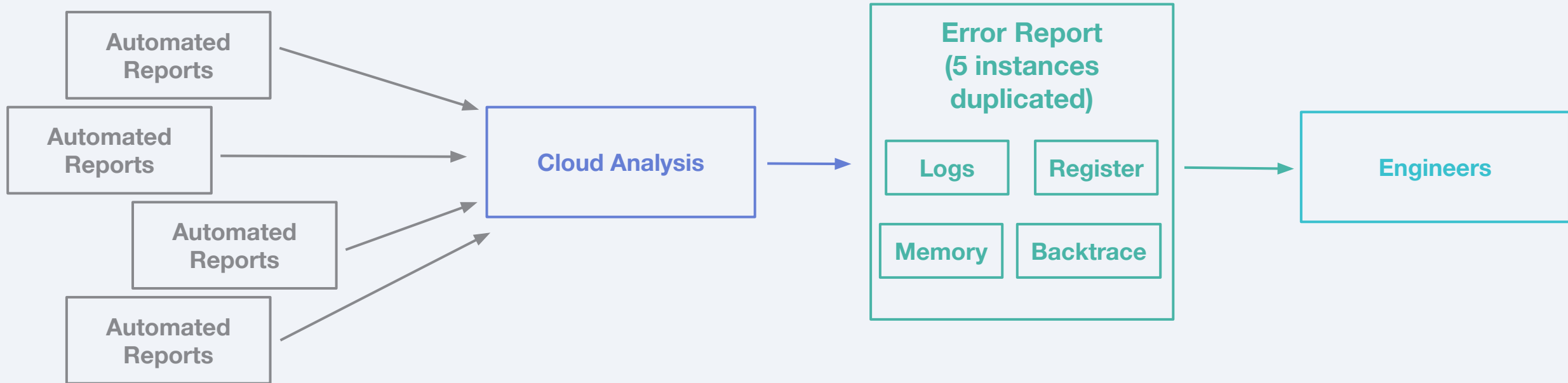
Mean

Remote Debugging



2+ Weeks

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:

The screenshot displays the Memfault Smart Sink interface for an issue titled "Mem Fault at compute_fft [Stack Overflow in accel-workq]". The interface includes a navigation bar with "Details", "All traces", "Comments", and "Merged issues". A table shows device information: Device (DEMOSERIALNUMBER), Cohort (default), Software (1.0.0-md5+a1c641ba (main)), and Hardware (DEVBOARD). The "State" section shows "Threads" for "accel-workq (2)" with a "STACK OVERFLOW" error and "RUNNING" status. The "Registers & Locals" section lists variables: dft_out, i, num_samples, raw_samples, tmp, and registers Sr0 through Sr8. The "Memory Viewer" section shows a memory dump with addresses and hex values.

Device	Value
Device	DEMOSERIALNUMBER
Cohort	default
Software	1.0.0-md5+a1c641ba (main)
Hardware	DEVBOARD

Thread	State
accel-workq (2)	STACK OVERFLOW RUNNING
Thread 3	SUSPENDED
idle (4)	READY
logging (5)	SUSPENDED
net_mgmt (6)	BLOCKED
rx_workq (7)	BLOCKED

Variable	Value
dft_out	0x2000a900 <my_stack_area+1344>
i	400
num_samples	536912536
raw_samples	0x3128115f
tmp	{1, 222, 7, 84}
Sr0	long 536912536 (0x2000a298)
Sr1	long 1372324912 (0x51cc0430)
Sr2	long 1372324919 (0x51cc0437)
Sr3	long 536912832 (0x2000a3c0)
Sr4	long 536912508 (0x2000a27c)
Sr5	long 536914136 (0x2000a8d8)
Sr6	long 0 (0x00000000)
Sr7	long 536912488 (0x2000a268)
Sr8	long 0 (0x00000000)



Poll #2

Memfault: IoT Reliability Platform

