Memfault

Launching an IoT Device

A Blueprint to Success

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

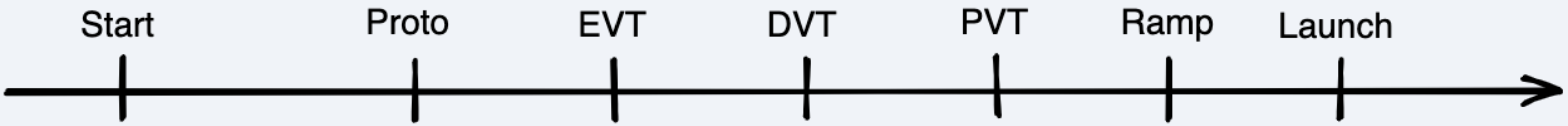
Agenda

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

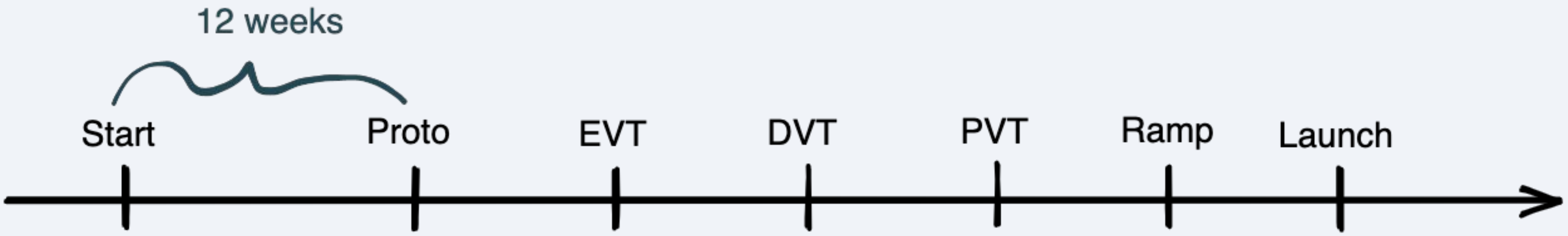
Shipping on time

⏱️

NPI Timeline

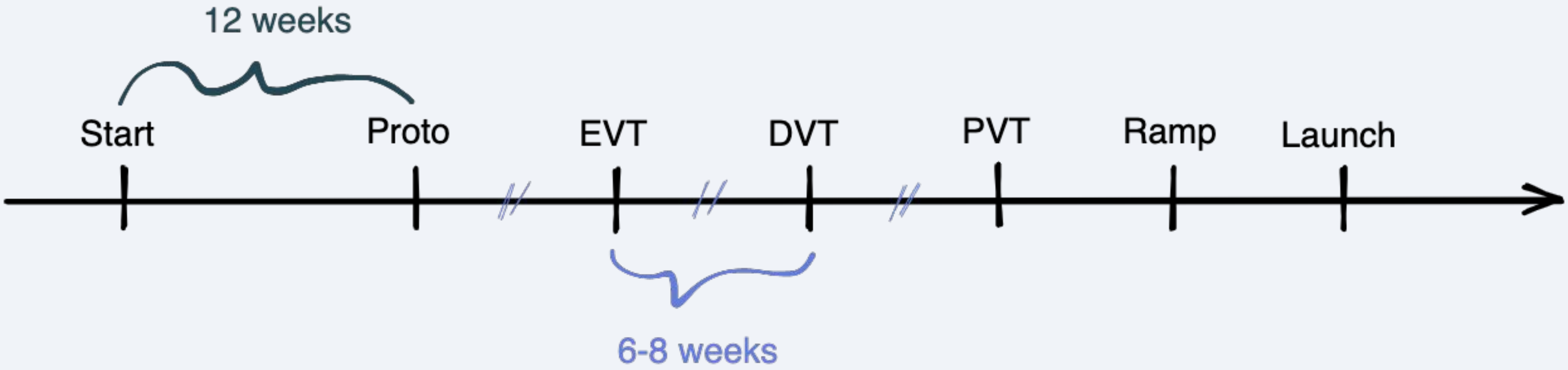


NPI Timeline



Proto: Figure out what you want to build

NPI Timeline

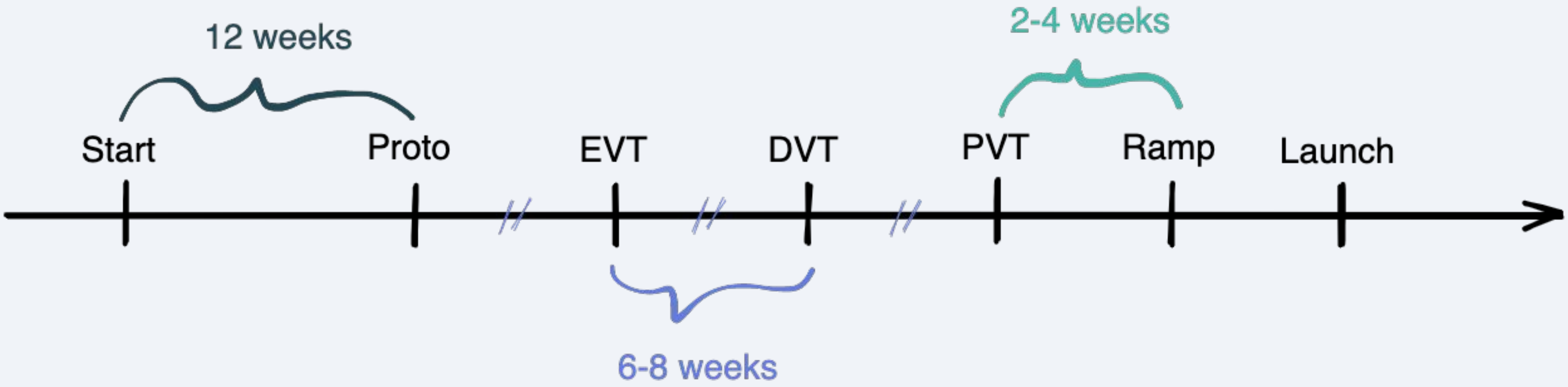


EVT: A handful of configurations, engineering design finalized

DVT: One final configuration, all manufacturing stations pass

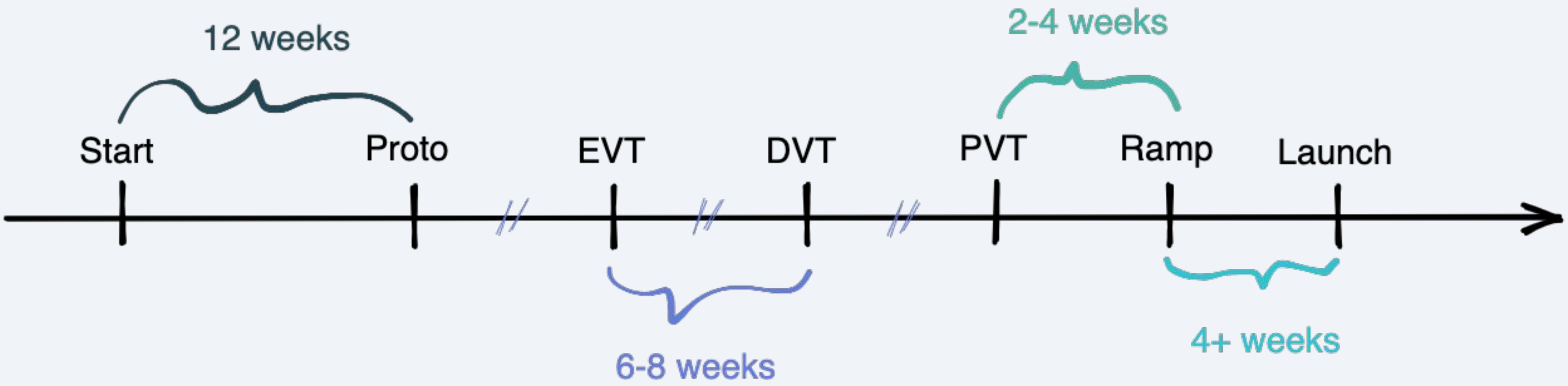
PVT: Manufacturing line operates at yield & speed

NPI Timeline



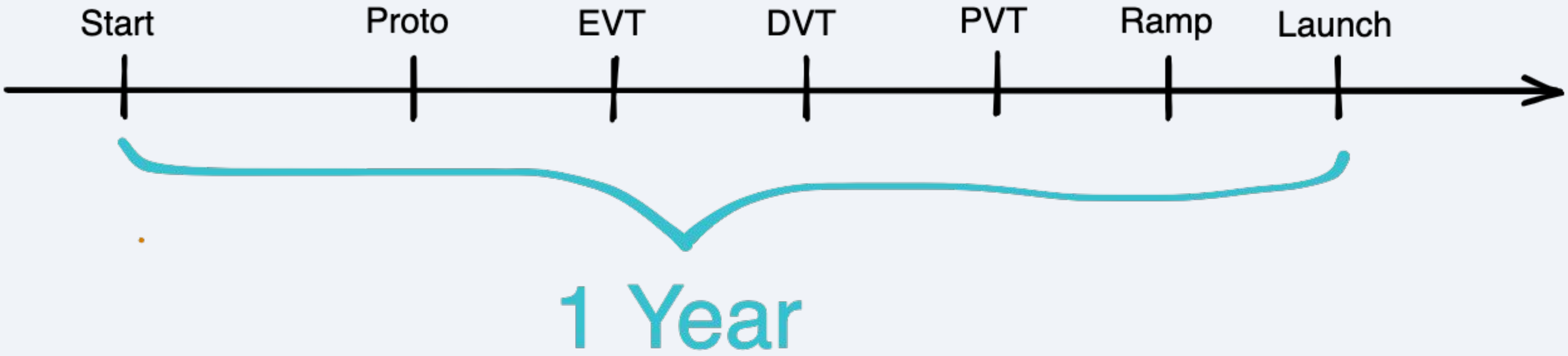
Ramp: Full scale manufacturing, start accumulating inventory for launch

NPI Timeline



Launch 🍾: Devices on shelves, available for purchase

NPI Timeline



Poll #1

How long did NPI take on your last product?

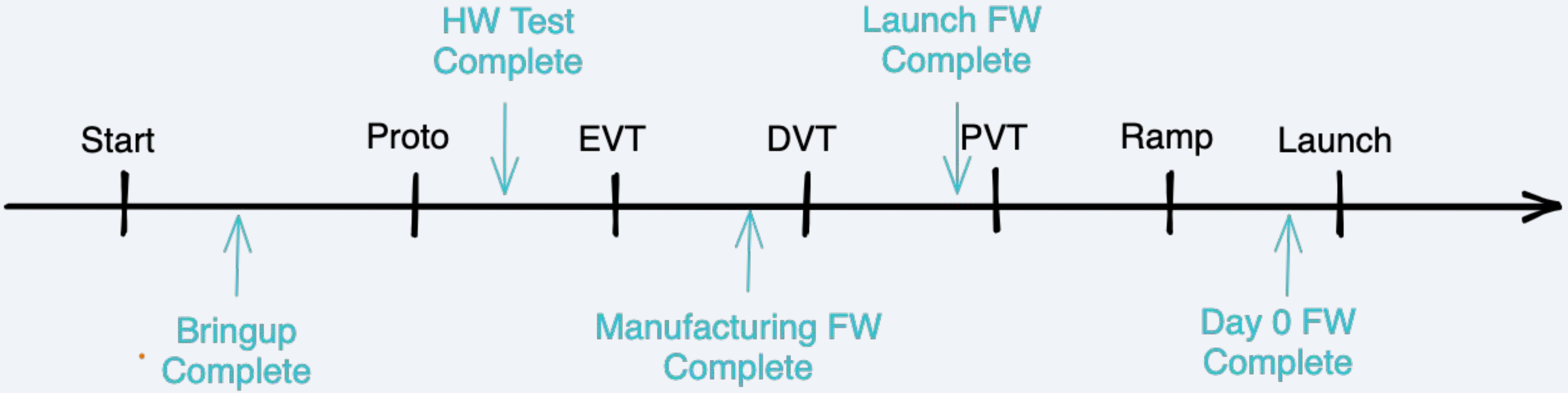
A. <= 12 months

B. <= 18 months

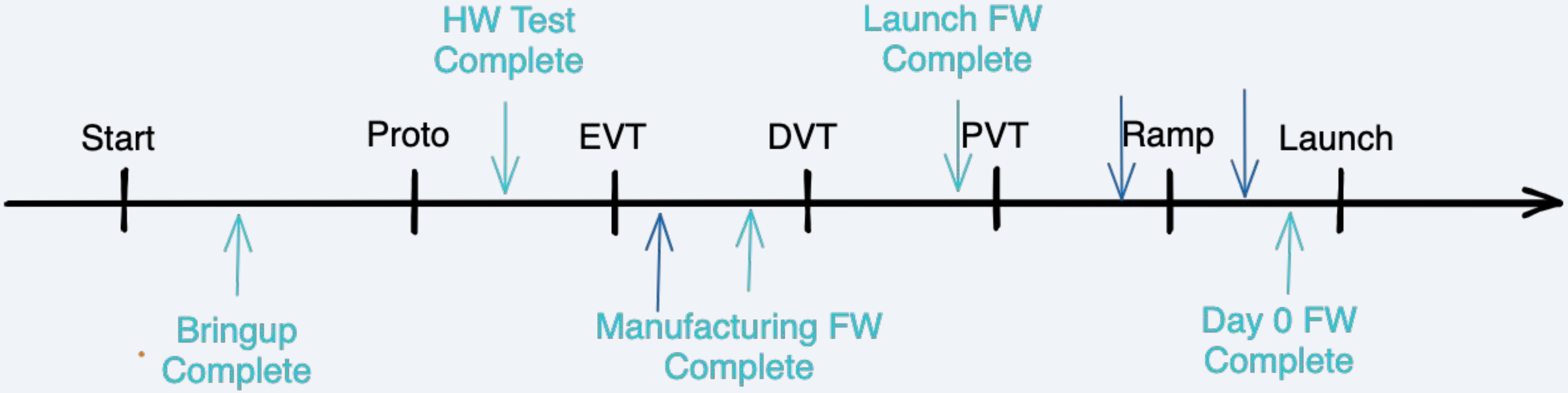
C. <= 2 years

D. > 2 years

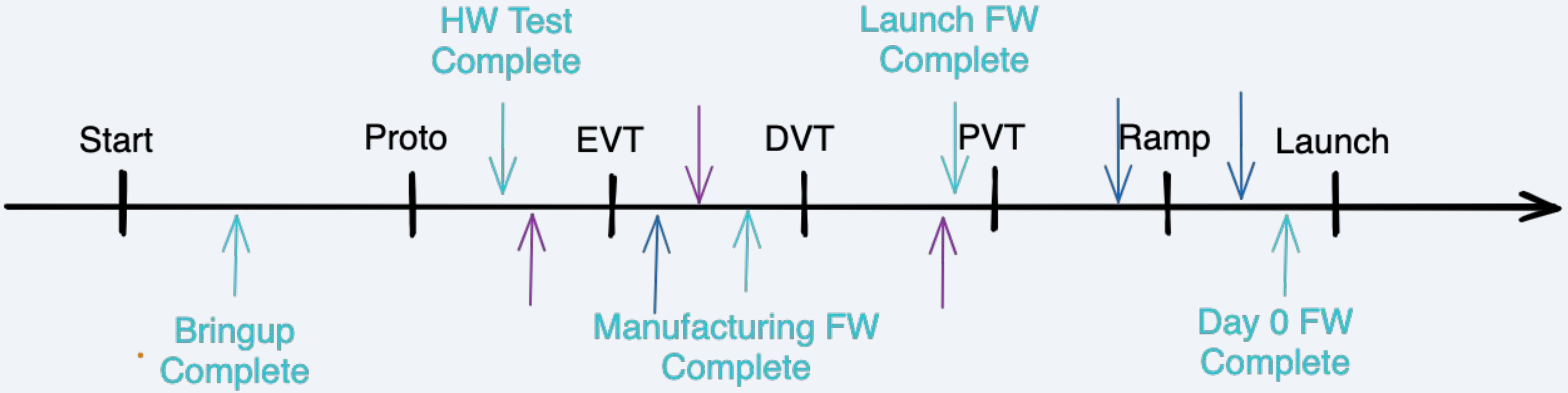
What about firmware?



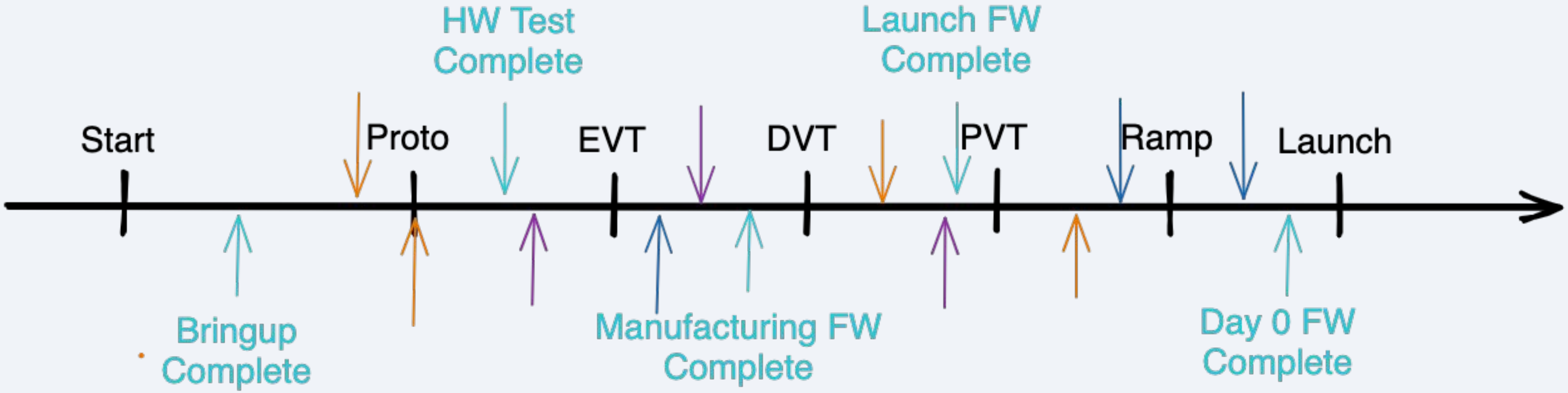
What about marketing?



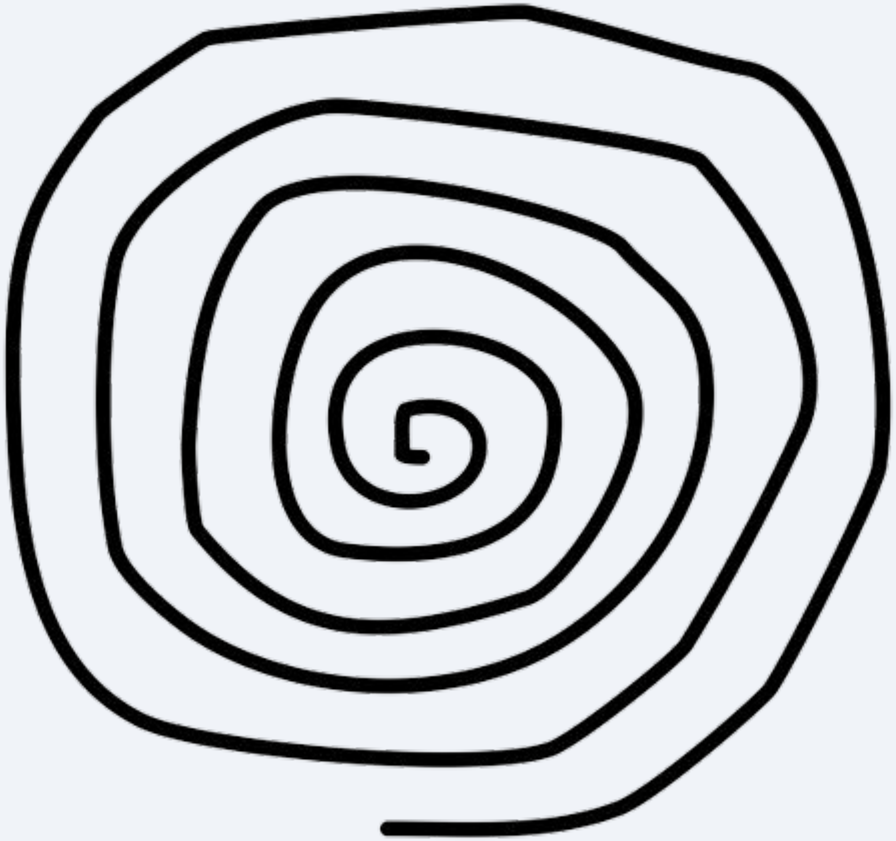
What about factory automation?



What about cloud software?



Avoid a dependency spiral



Decoupling SW & HW Timelines

1. Test Driven Development
2. Day-0 Updates
3. A Strong HAL
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

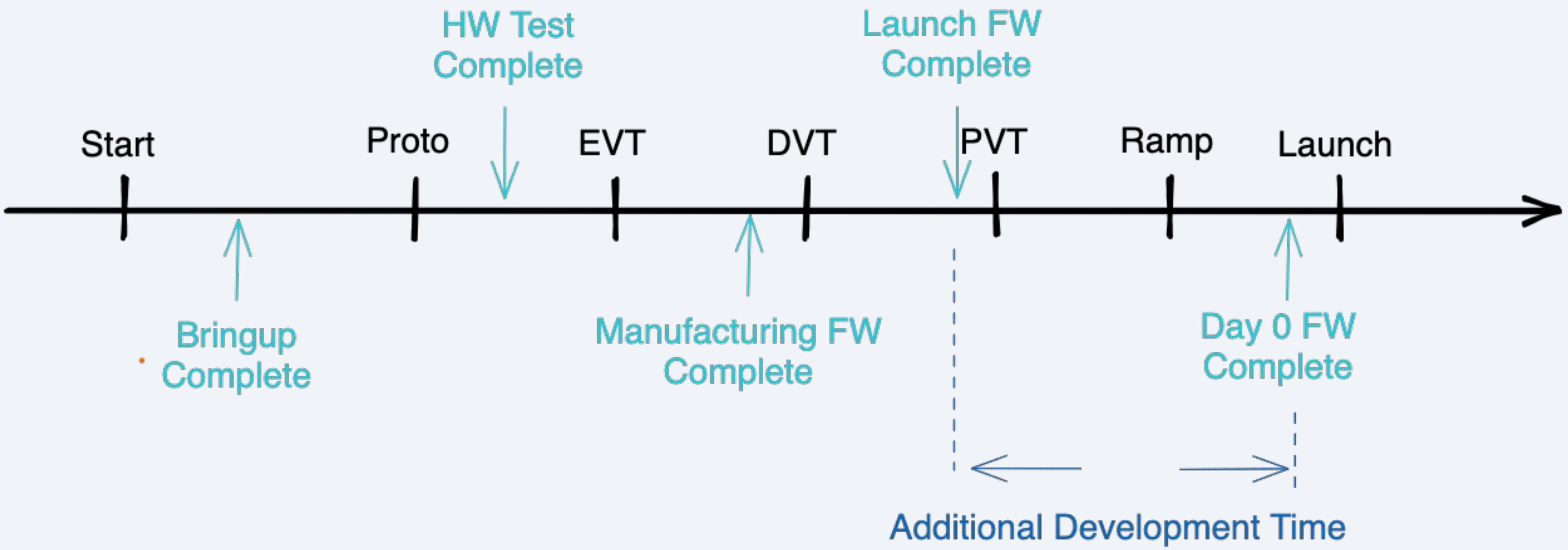
What it is

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

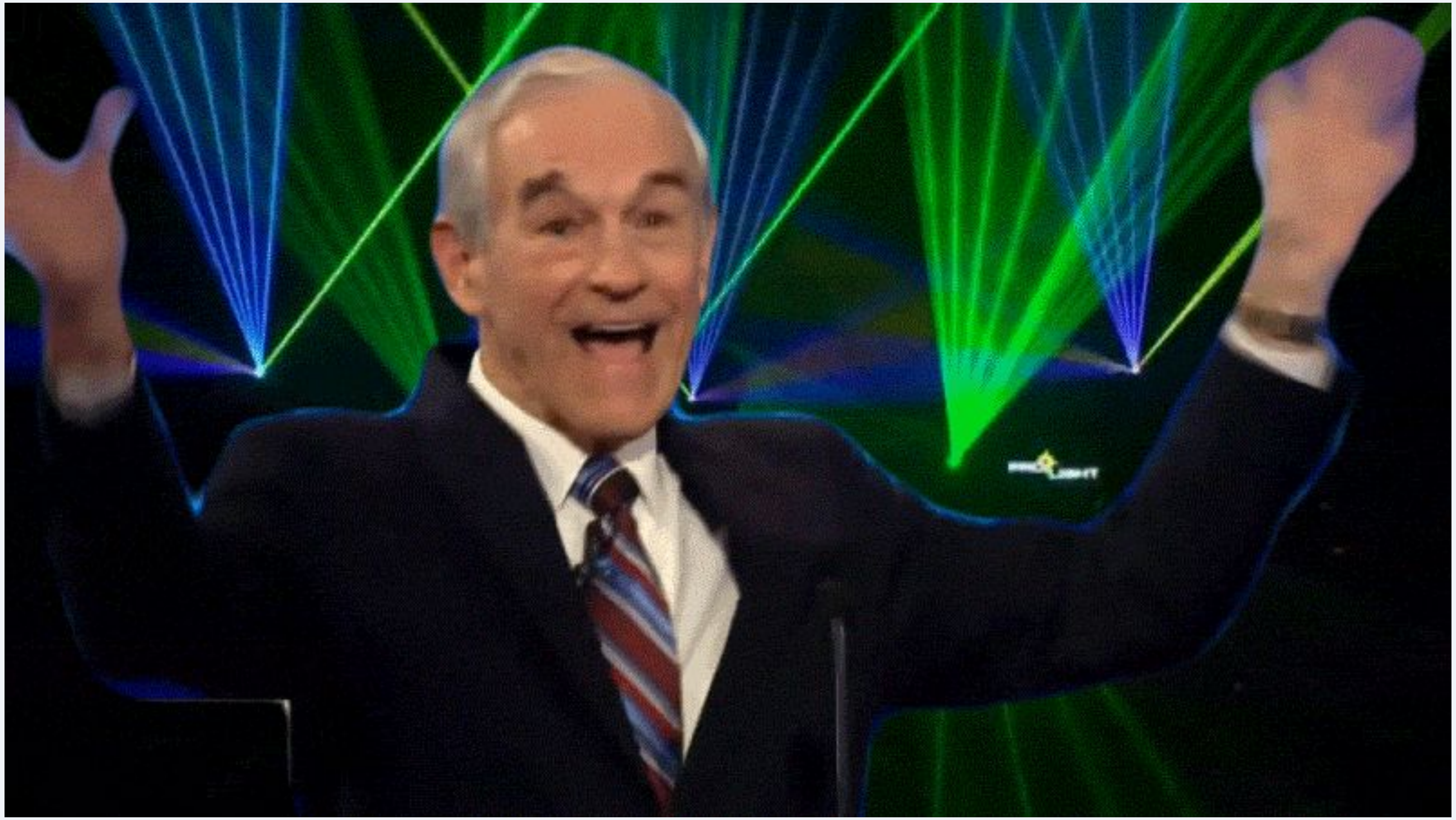
- Iterate on the application FW without impacting the manufacturing FW
- Continue working on app FW after DVT when factory processes are locked
- Save code space

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 Fleet Reliability Engineering

Robust OTA

Performance Metrics

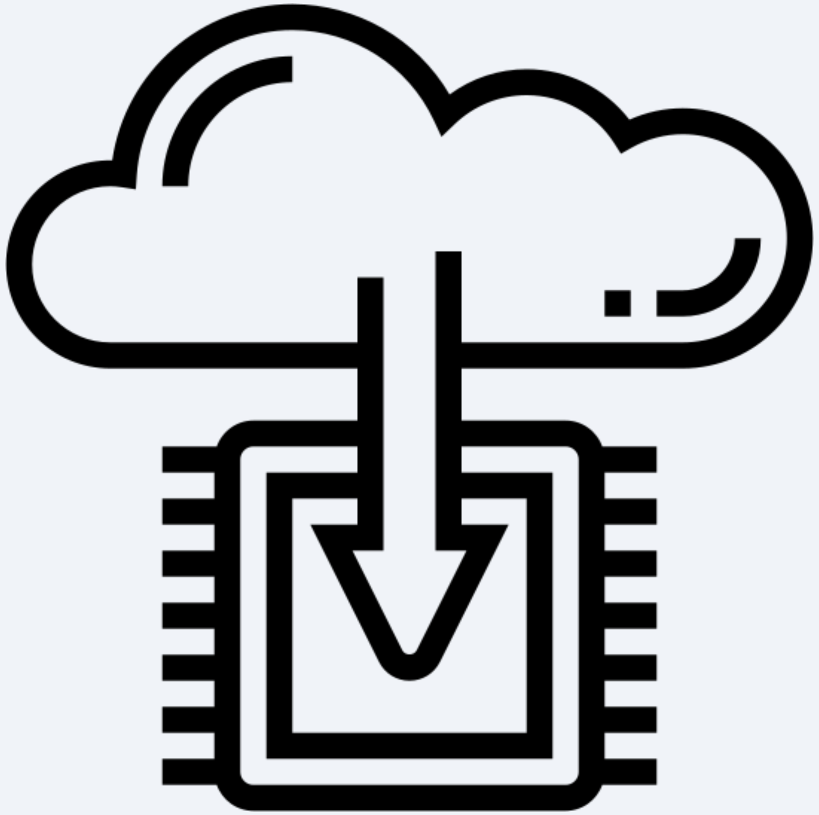
Remote Debugging

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

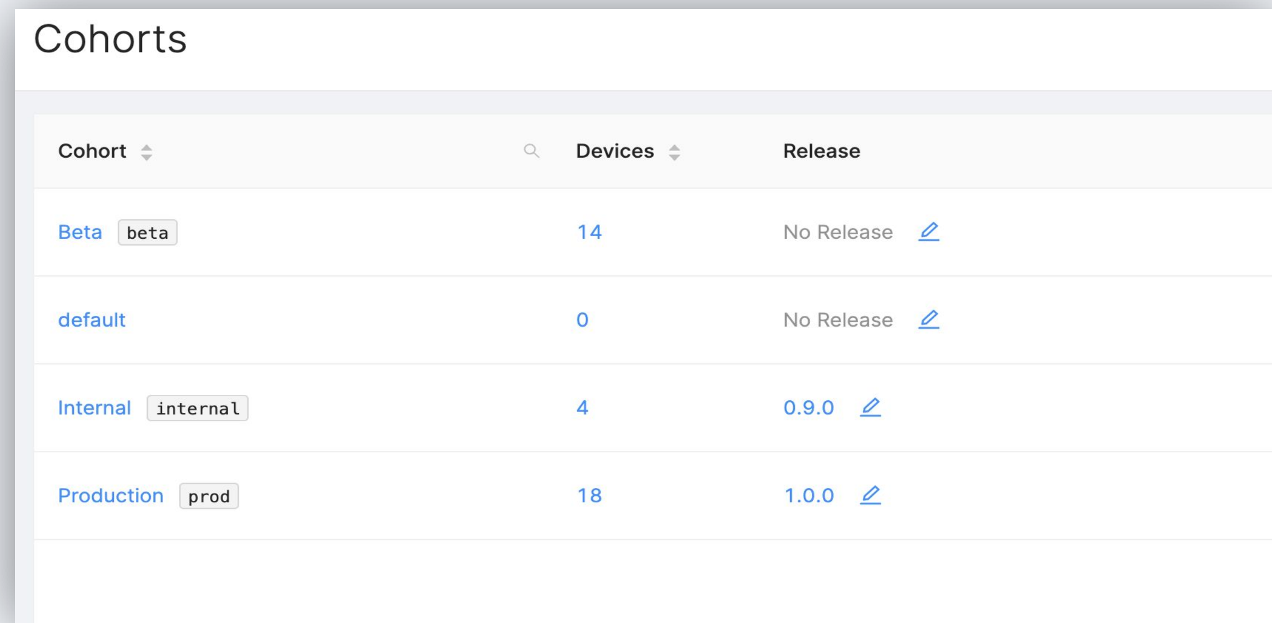
What it is

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

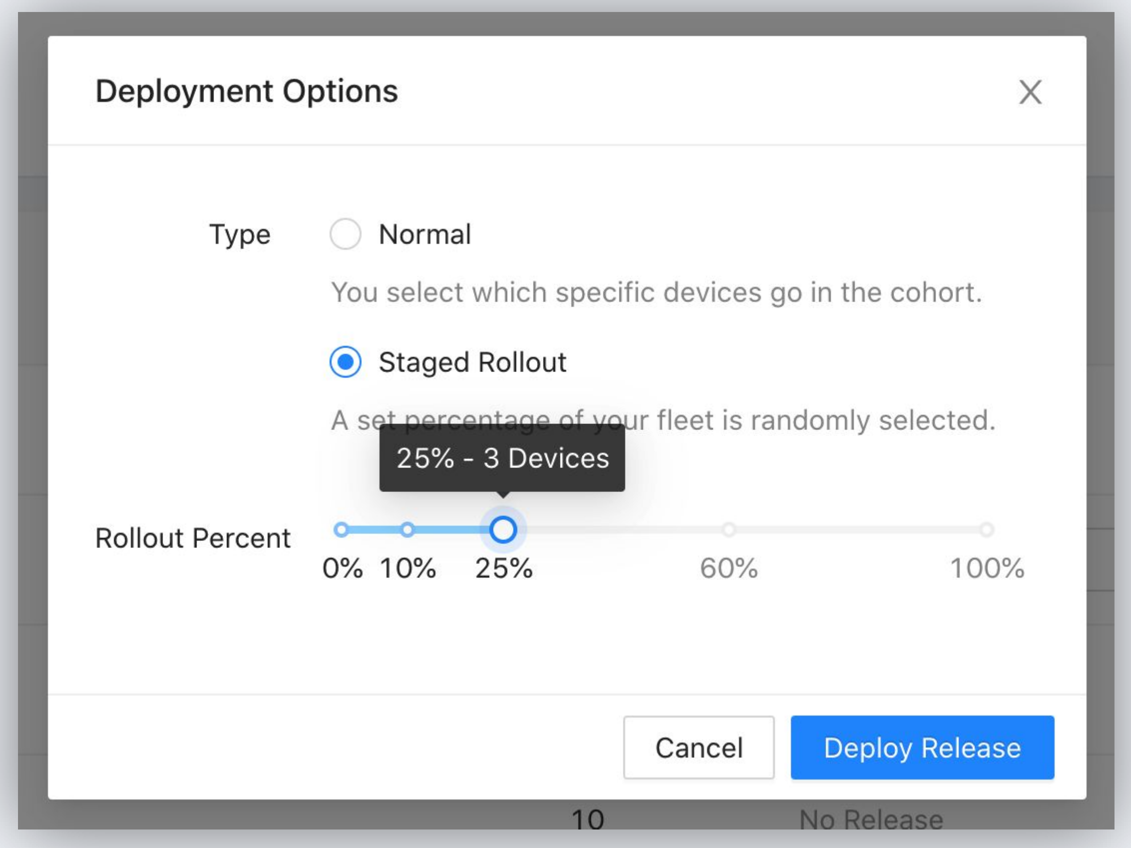
What it is

The ability to roll out a new release to an incrementally larger sub-set 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:



Deployment Options

Type ○ Normal

You select which specific devices go in the cohort.

● Staged Rollout

A set percentage of your fleet is randomly selected.

25% - 3 Devices

Rollout Percent 0% 10% 25% 60% 100%

10 No Release

Must-Pass Through

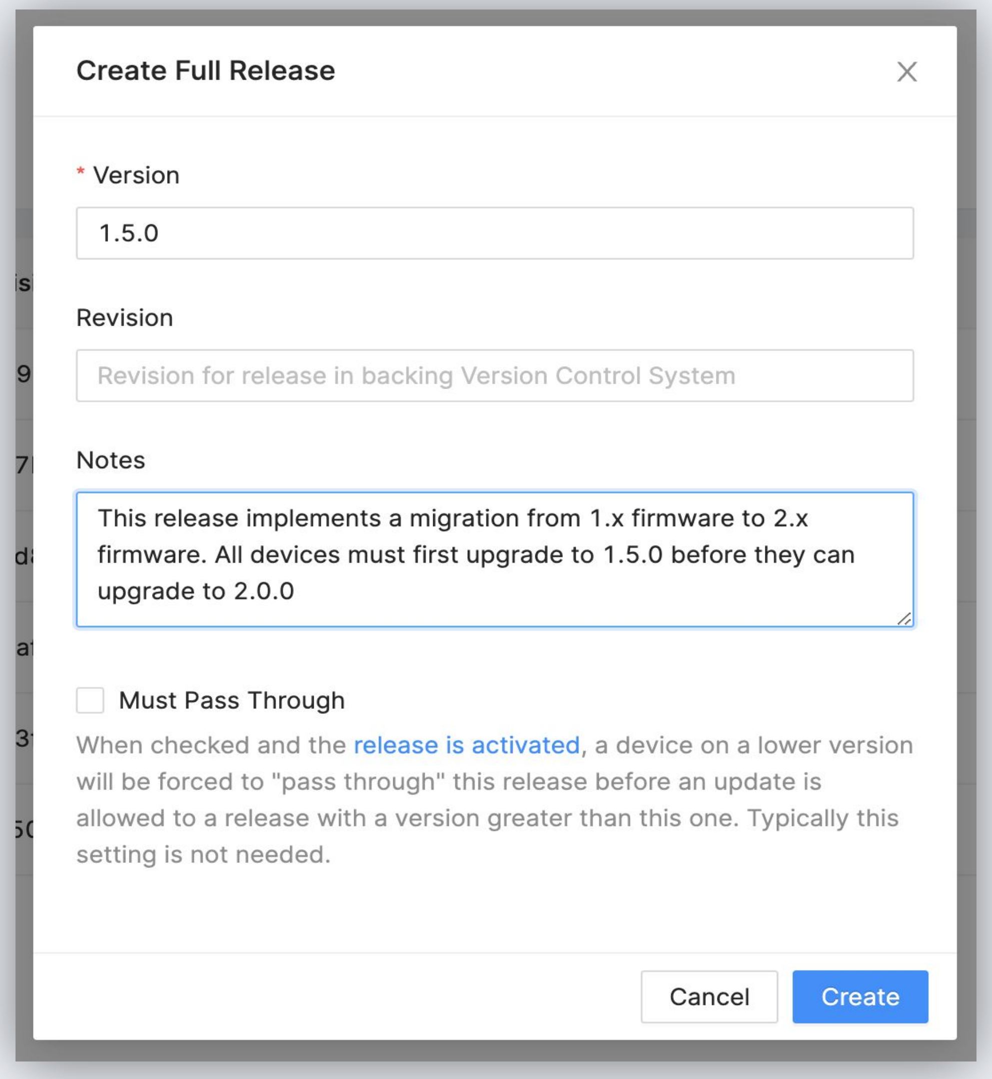
What it is

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.

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

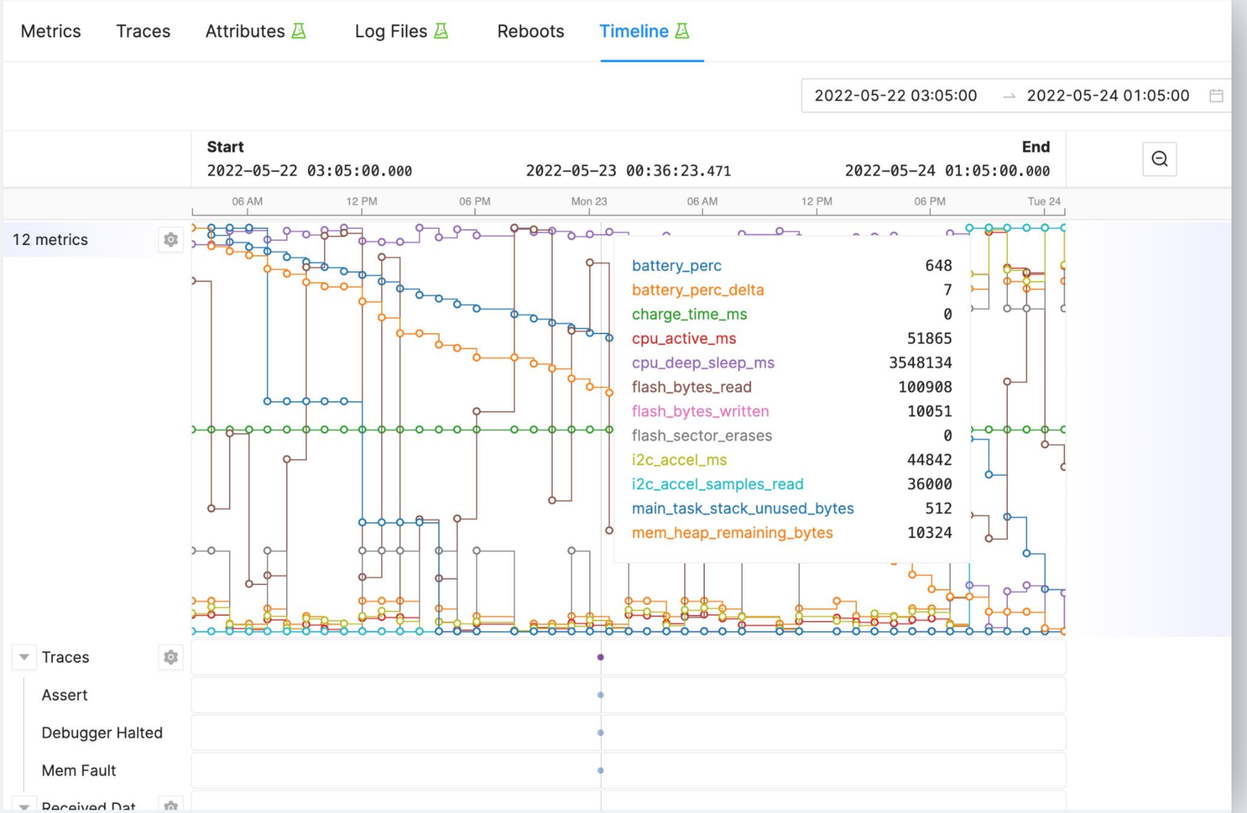
What it is

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:



2022-05-22 03:05:00 → 2022-05-24 01:05:00

Start 2022-05-22 03:05:00.000

2022-05-23 00:36:23.471

End 2022-05-24 01:05:00.000

battery_perc	648
battery_perc_delta	7
charge_time_ms	0
cpu_active_ms	51865
cpu_deep_sleep_ms	3548134
flash_bytes_read	100908
flash_bytes_written	10051
flash_sector_erases	0
i2c_accel_ms	44842
i2c_accel_samples_read	36000
main_task_stack_unused_bytes	512
mem_heap_remaining_bytes	10324

Aggregates and Dashboards

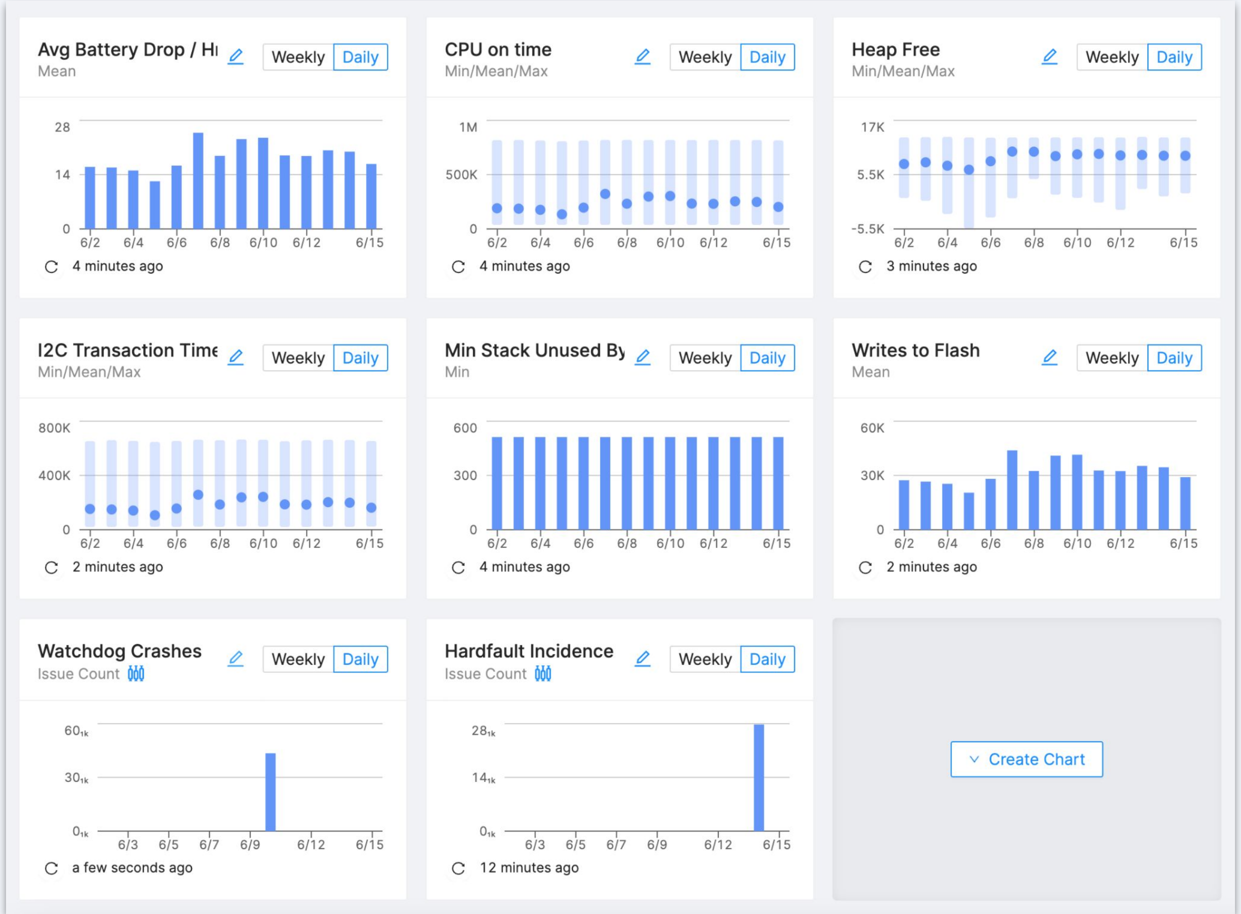
What it is

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

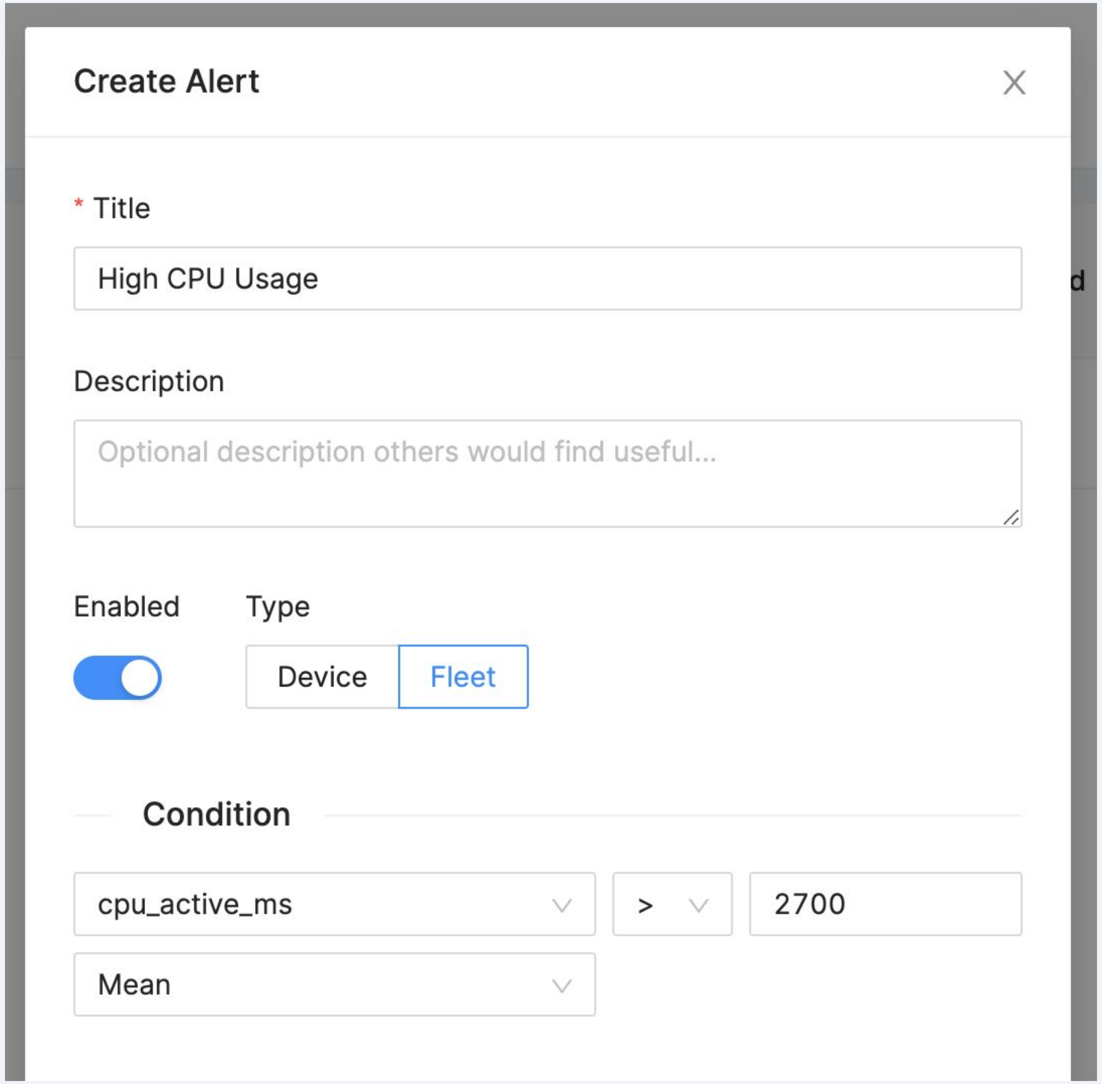
What it is

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

High CPU Usage

Description

Optional description others would find useful...

Enabled: ●

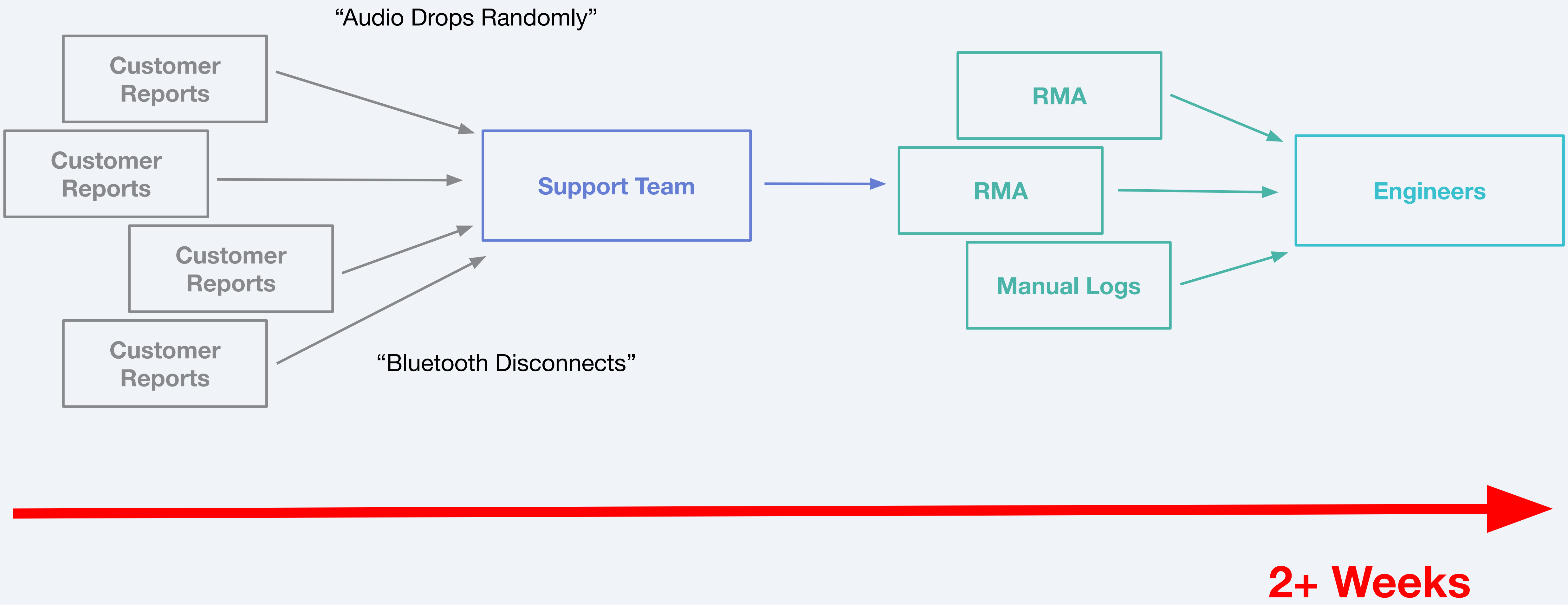
Type: Device | Fleet

Condition

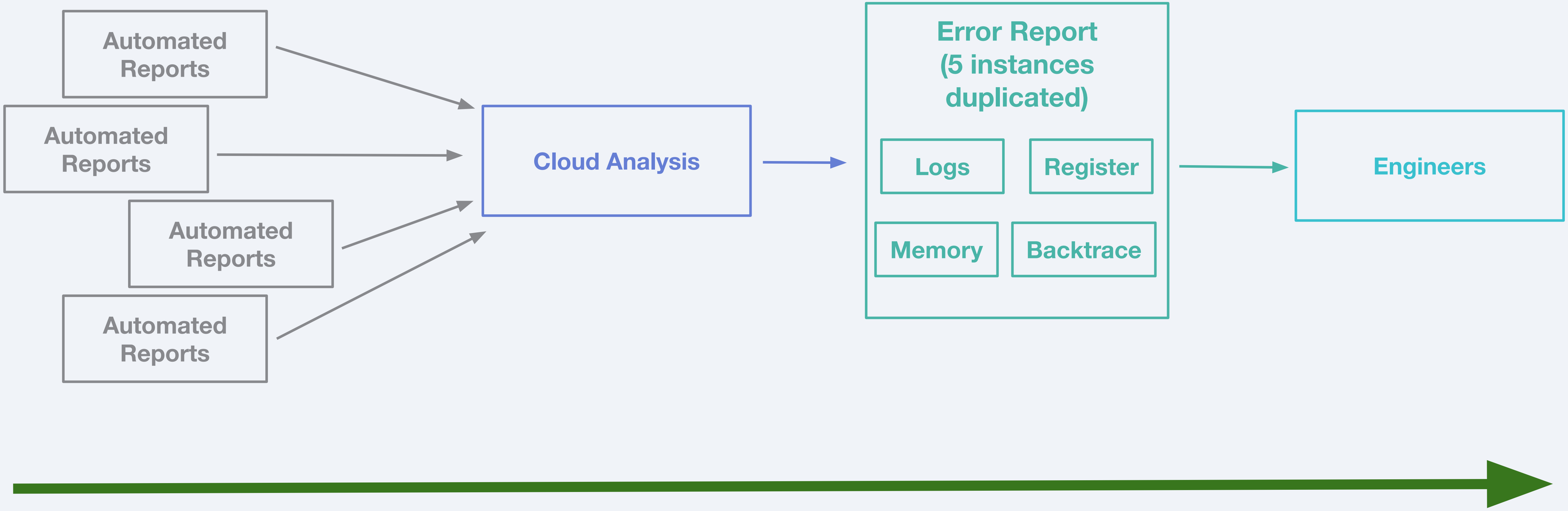
cpu_active_ms > 2700

Mean

Remote Debugging



Remote Debugging



2 minutes

Coredumps

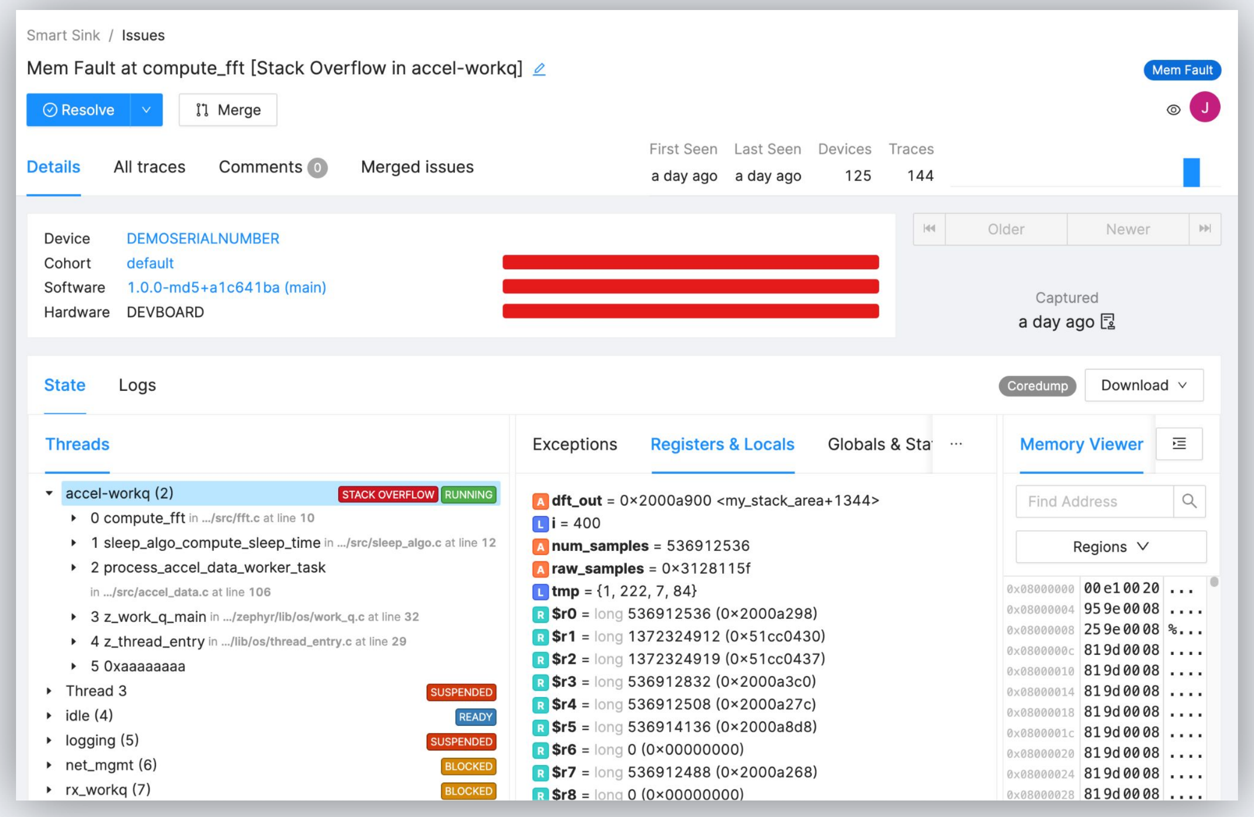
What it is

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:



Smart Sink / Issues

Mem Fault at compute_fft [Stack Overflow in accel-workq]

First Seen	Last Seen	Devices	Traces
a day ago	a day ago	125	144

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

[redacted]

Captured
a day ago

```
▾ accel-workq (2)                                        STACK OVERFLOW RUNNING
    ▸ 0 compute_fft in .../src/fft.c at line 10
    ▸ 1 sleep_algo_compute_sleep_time in .../src/sleep_algo.c at line 12
    ▸ 2 process_accel_data_worker_task
      in .../src/accel_data.c at line 106
    ▸ 3 z_work_q_main in .../zephyr/lib/os/work_q.c at line 32
    ▸ 4 z_thread_entry in .../lib/os/thread_entry.c at line 29
    ▸ 5 0xaaaaaaaa
▸ Thread 3                                                          SUSPENDED
▸ idle (4)                                                              READY
▸ logging (5)                                                       SUSPENDED
▸ net_mgmt (6)                                                        BLOCKED
▸ rx_workq (7)                                                        BLOCKED
```

```
A dft_out = 0x2000a900 <my_stack_area+1344>
L i = 400
A num_samples = 536912536
A raw_samples = 0x3128115f
L tmp = {1, 222, 7, 84}
R $r0 = long 536912536 (0x2000a298)
R $r1 = long 1372324912 (0x51cc0430)
R $r2 = long 1372324919 (0x51cc0437)
R $r3 = long 536912832 (0x2000a3c0)
R $r4 = long 536912508 (0x2000a27c)
R $r5 = long 536914136 (0x2000a8d8)
R $r6 = long 0 (0x00000000)
R $r7 = long 536912488 (0x2000a268)
R $r8 = long 0 (0x00000000)
```

```
0x08000000  00 e1 00 20  ...
0x08000004  95 9e 00 08  ....
0x08000008  25 9e 00 08  %...
0x0800000c  81 9d 00 08  ....
0x08000010  81 9d 00 08  ....
0x08000014  81 9d 00 08  ....
0x08000018  81 9d 00 08  ....
0x0800001c  81 9d 00 08  ....
0x08000020  81 9d 00 08  ....
0x08000024  81 9d 00 08  ....
0x08000028  81 9d 00 08  ....
```

Poll #2

Which infrastructure do you have in place?

Check all that apply…

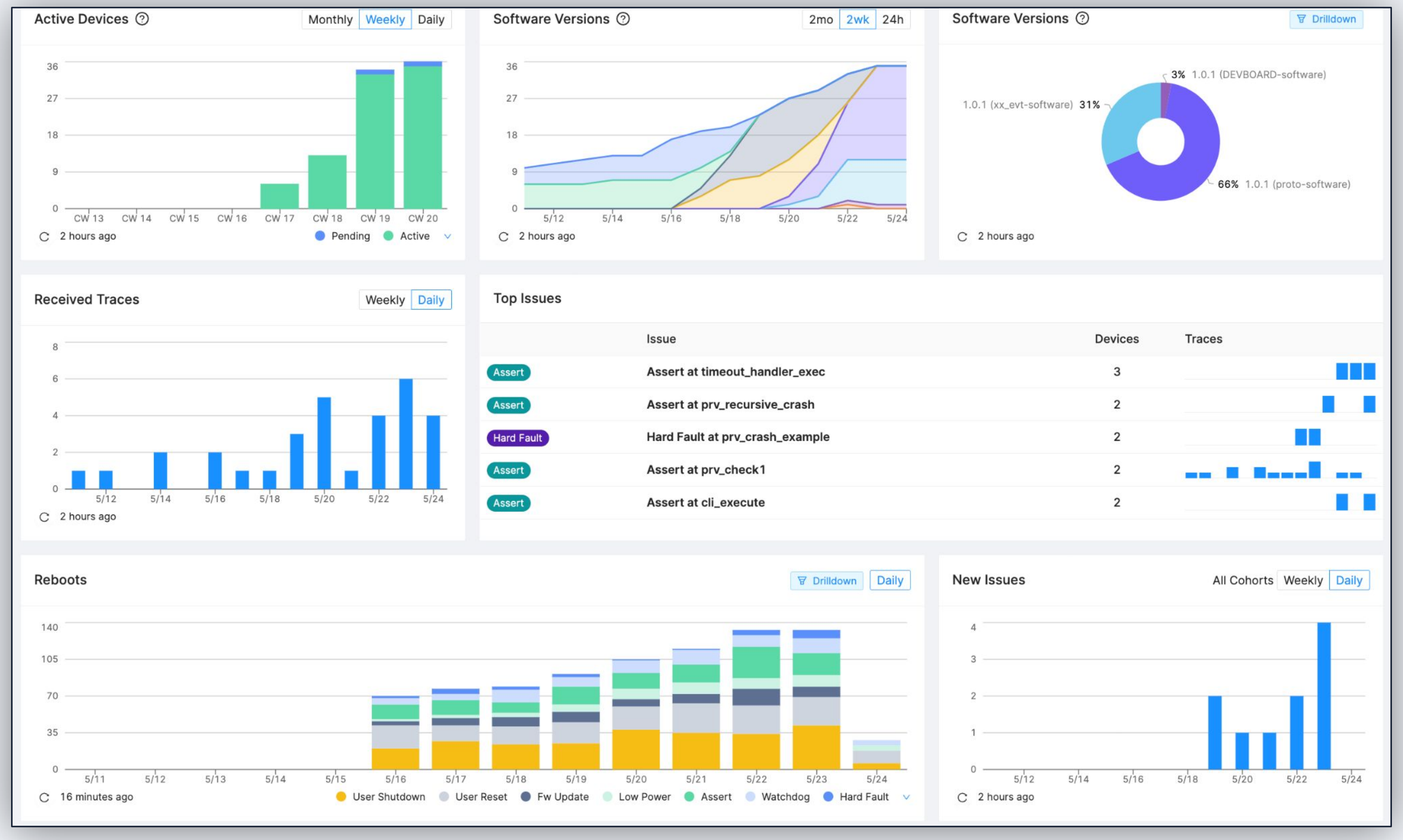
A. **OTA**

B. **Metrics**

C. **Remote Debugging**

D. **None of the above**

Memfault: Fleet Reliability Engineering Platform



Top Issues

	Issue	Devices	Traces
Assert	Assert at timeout_handler_exec	3	
Assert	Assert at prv_recursive_crash	2	
Hard Fault	Hard Fault at prv_crash_example	2	
Assert	Assert at prv_check1	2	
Assert	Assert at cli_execute	2	

Q&A

Memfault