

# Remote Debugging & Device Observability: How Memfault & Diamond Kinetics worked together to fix firmware bugs

Mike Ressler, CTO, Diamond Kinetics (@mressler)

François Baldassari, Founder & CEO, Memfault (@baldassarifr)



Diamond Kinetics is the market leader in mobile motion technology that enables player development, superior equipment fitting, objective scouting and recruiting, and engagement-driven entertainment.

## Products

- **SwingTracker** - baseball and softball swing analysis and development tool.
- **PitchTracker** - motion analytics and easy-to-understand metrics, data and pitching analysis tools previously available only with expensive hardware system.

### Gen 2

- **Chip:** CSR8670
- **Connectivity:** BLE
- **OS:** Qualcomm Audio
- **Language:** Assembly

### Gen 3 (latest)

- **Chip:** Dialog
- **Connectivity:** BLE
- **OS:** FreeRTOS
- **Language:** C



# How it works



BLE

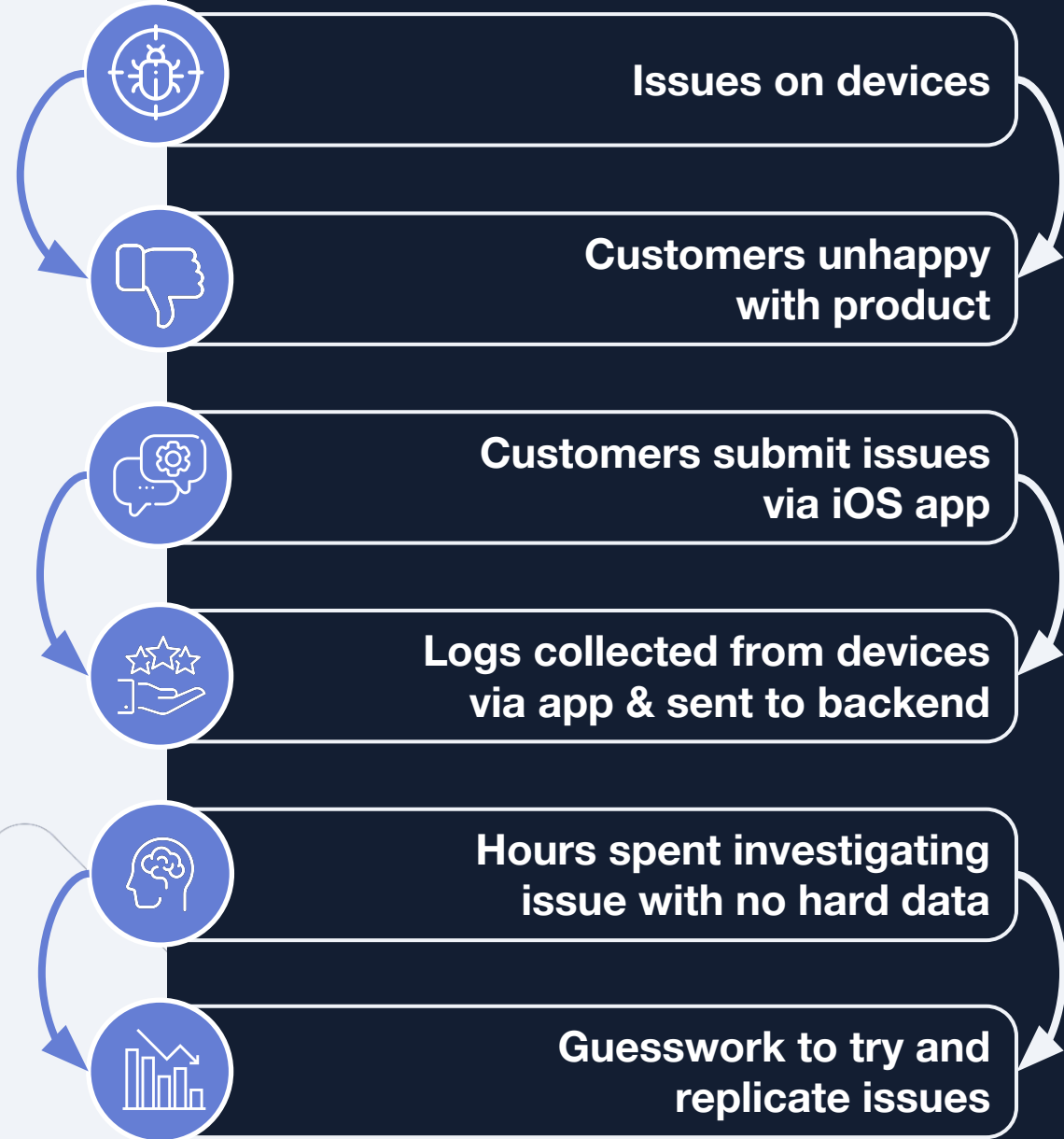


HTTP



# Once out in the field, you must rely on “psychic debugging”.

Diamond Kinetics had to solve bugs based on iOS app logs, customer reports, and replicating issues in a controlled setting.



# Debugging Challenges

No data on crash logs -  
No stack traces &  
embedded running  
custom assembly on a  
CSR8670.



No analytics into  
BLE connectivity,  
battery and other  
metrics



Updating devices  
felt risky and  
error-prone



Too much  
wasted testing  
during dev  
phase



**Developers on average  
spend 40% of the  
develop cycle debugging**

Source: <https://www.designnews.com/electronics/defining-right-debugging-mindset-faults-errors-and-bugs>

# We decided to work with Memfault

## ◇ Tight OS Integration

Were able to build Memfault in to FreeRTOS from Day 1 and had a working dev complete firmware within 2 weeks.

## ◇ Development Accelerant

Used Memfault as data transport layer & had working data pipeline with parsing on iOS app side in one night instead of weeks or more.

## ◇ Agile IoT Development

Small roll outs, observe issues, and respond appropriately - it's like web app development, but for embedded devices.

## ◇ Flexible Framework

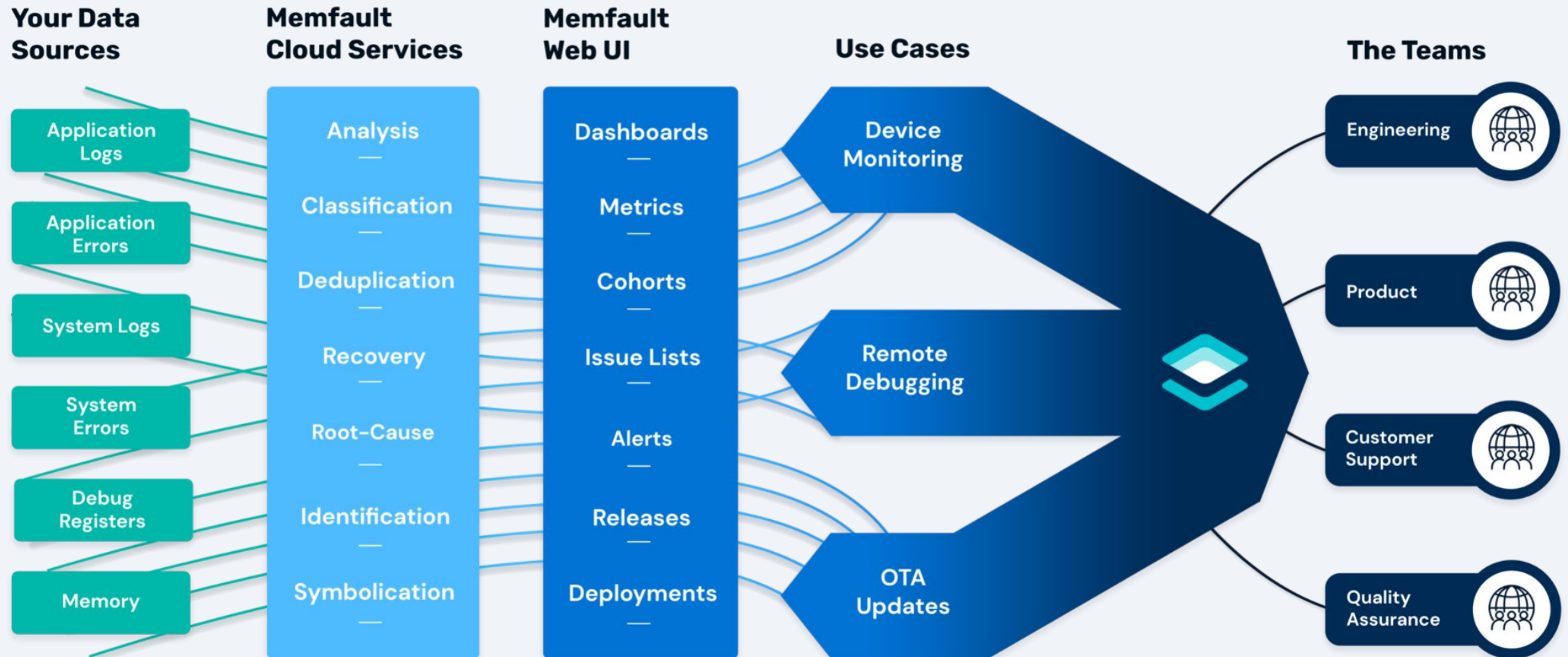
Cohorts enabled us to maintain compatible app version and firmware versions with custom implementations.

# How it works

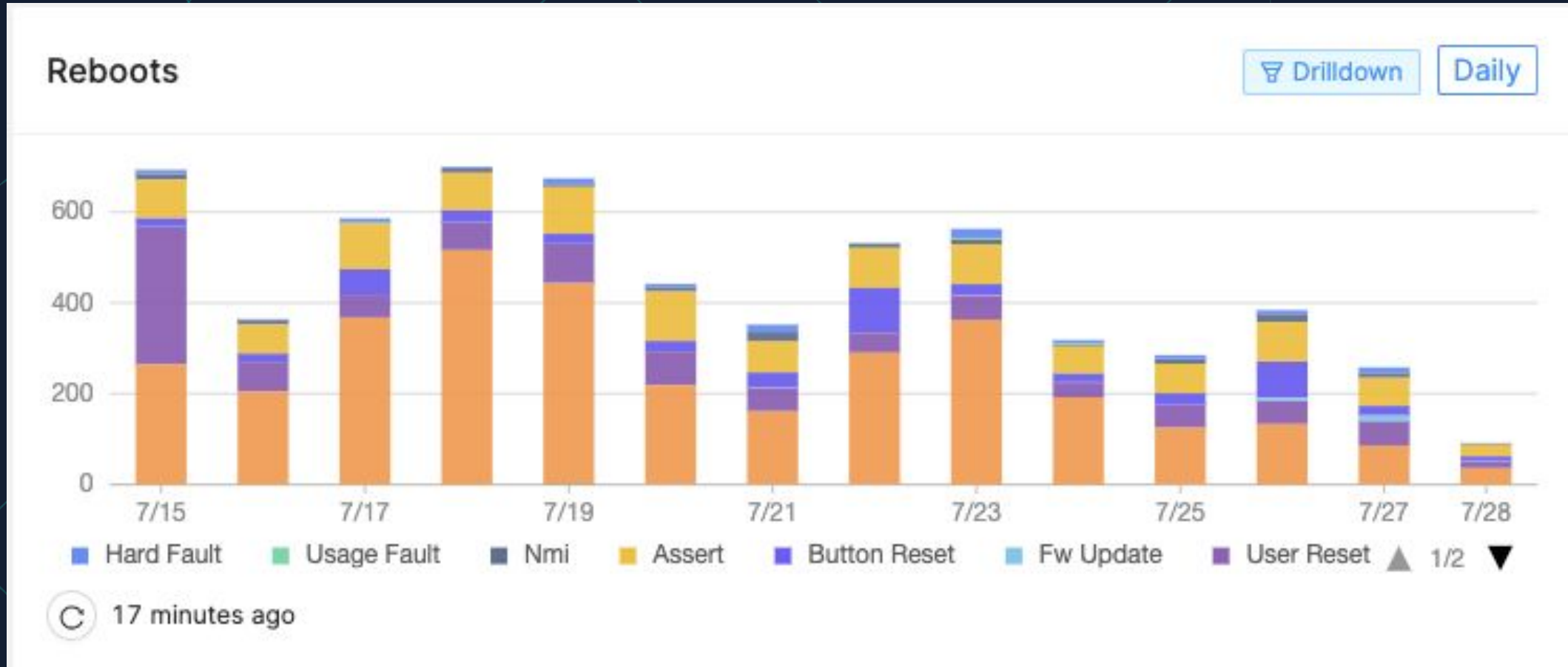




# How Memfault works



# What did we find?



# What did we find?

TrackerV3 / Issues

## Assert at chargingPanic [↗](#)

[Assert](#)

[Resolve](#) [Merge](#)

First Seen: 7 months ago 16

[Details](#) [All traces](#) [Comments 0](#) [Merge](#)

TrackerV3 / Issues


## Hard Fault at ble\_controller\_error [↗](#)

[Resolve](#) [Merge](#)

[Hard Fault](#)

[FB](#) [HB](#) [+6](#)

First Seen	Last Seen	Devices	Traces
7 months ago	an hour ago	2,284	33,823



[Details](#) [All traces](#) [Comments 1](#) [Merged issues](#)

TrackerV3 / Issues


## Assert at MainTask\_performDelayed [Stack Overflow in Transfer Pipe] [↗](#)

[Assert](#)

[Resolve](#) [Merge](#)

[FB](#) [JR](#)

First Seen	Last Seen	Devices	Traces
7 months ago	an hour ago	334	463



[Details](#) [All traces](#) [Comments 0](#) [Merged issues](#)

# What did we find?

The screenshot displays a debugger interface with two main panels. The left panel, titled 'Threads', lists various threads and their states. The right panel, titled 'Registers & Locals', shows the current register values.

**Threads Panel:**

- External Interrupt 4 - Exception Number 20 (2) **ACTIVE INTERRUPT**
  - 0 ble\_controller\_error in .../da14690/src/arch\_main.c at line 419
  - 1 sys\_cmac\_on\_error\_handler in .../da14690/src/arch\_main.c at line 424
  - 2 pm\_get\_sys\_wakeup\_cycles in .../sys\_power\_mgr\_da1469x.c at line 1665
  - 3 0xfffffbc
- IDLE (3) **RUNNING**
- AD\_IRQ\_SNC (4) **SUSPENDED**
- AD\_SNC (5) **SUSPENDED**
- BLE Task (6) **SUSPENDED**
- CH\_NOK (7) **SUSPENDED**
- CH\_OK (8) **SUSPENDED**
- Main Task (9) **SUSPENDED**
- SW\_FSM (10) **SUSPENDED**
- Save Pipe (11) **SUSPENDED**
- Sen Stream Pipe (12) **SUSPENDED**
- Sensor Sampling (13) **BLOCKED**
- Tmr Svc (14) **BLOCKED**
- Transfer Pipe (15) **SUSPENDED**
- VBUS (16) **SUSPENDED**
- bleA (17) **SUSPENDED**
- bleM (18) **SUSPENDED**

**Registers & Locals Panel:**

- R \$r0 = long 1 (0x00000001)
- R \$r1 = long 536875655 (0x20001287)
- R \$r2 = long -2147482623 (0x80000401)
- R \$r3 = long 536936928 (0x200101e0)
- R \$r4 = long 1073750016 (0x40002000)
- R \$r5 = long 3240 (0x0000ca8)
- R \$r6 = long 0 (0x00000000)
- R \$r7 = long 3241 (0x0000ca9)
- R \$r8 = long 7621078 (0x007449d6)
- R \$r9 = long 0 (0x00000000)
- R \$r10 = long 1 (0x00000001)
- R \$r11 = long 18 (0x00000012)
- R \$r12 = long 9169 (0x000023d1)
- R \$sp = void \* 0x20021248 <\_\_StackLimit+976> (0x20021248)
- R \$lr = long 52113 (0x0000cb91)
- R \$pc = void (\*)() 0xcb86 <ble\_controller\_error+2> (0x0000cb86)
- R \$xpsr = long 16777236 (0x01000014)
- R \$fpscr = long 0 (0x00000000)
- R \$msp = void \* 0x20021248 <\_\_StackLimit+976> (0x20021248)
- R \$psp = void \* 0x20013d28 <ucHeap+1592> (0x20013d28)

# What did we find?

## Fixed connectivity bug (#1 crash) in BLE chip using Memfault

### ◆ Caught issue during dev testing

No more guesswork and “What was I doing when this crashed?” Diamond Kinetics had data needed to root case the issue and send a fix.

### ◆ Went to market with issue

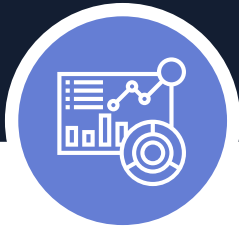
Did not have to delay shipment with since we were aware of it and confident we could solve it. Instead, developed a systemic workaround with Dialog.

### ◆ Fixed the issue

Shared “Bluetooth processor” stack traces from Memfault with Dialog to help them patch firmware.

**We reduced the number  
of resets per device by  
90%**

# Why I wouldn't go to market on an IoT product without Memfault



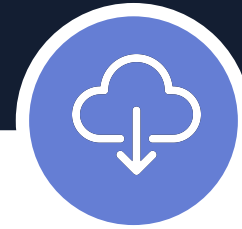
## More Confidence in Releases

Knowing we have real-time alerts with full visibility into any issue assures us that we can resolve issues quickly cutting down the need to test for weeks.



## Happier support & engineering teams

Support and engineering get to work with hard data rather than guesses. They are happier and more productive.



## Ship products and updates faster

Controlled, rolling firmware deployments enables us to observe new issues quickly leading to a decrease in manual QA and more rapid deployment schedule.

**“Never let the fear of  
striking out keep you  
from playing the game.”**

**- Babe Ruth**



**“Never let the fear of bugs  
keep you from shipping  
your product. - Babe Ruth”**

***- Mike Ressler***

# Connect with us



[mressler@diamondkinetics.com](mailto:mressler@diamondkinetics.com)



[linkedin.com/in/mressler/](https://www.linkedin.com/in/mressler/)



@mressler



[francois@memfault.com](mailto:francois@memfault.com)



[linkedin.com/in/francois-baldassari/](https://www.linkedin.com/in/francois-baldassari/)



@baldassarifr

**Questions?**