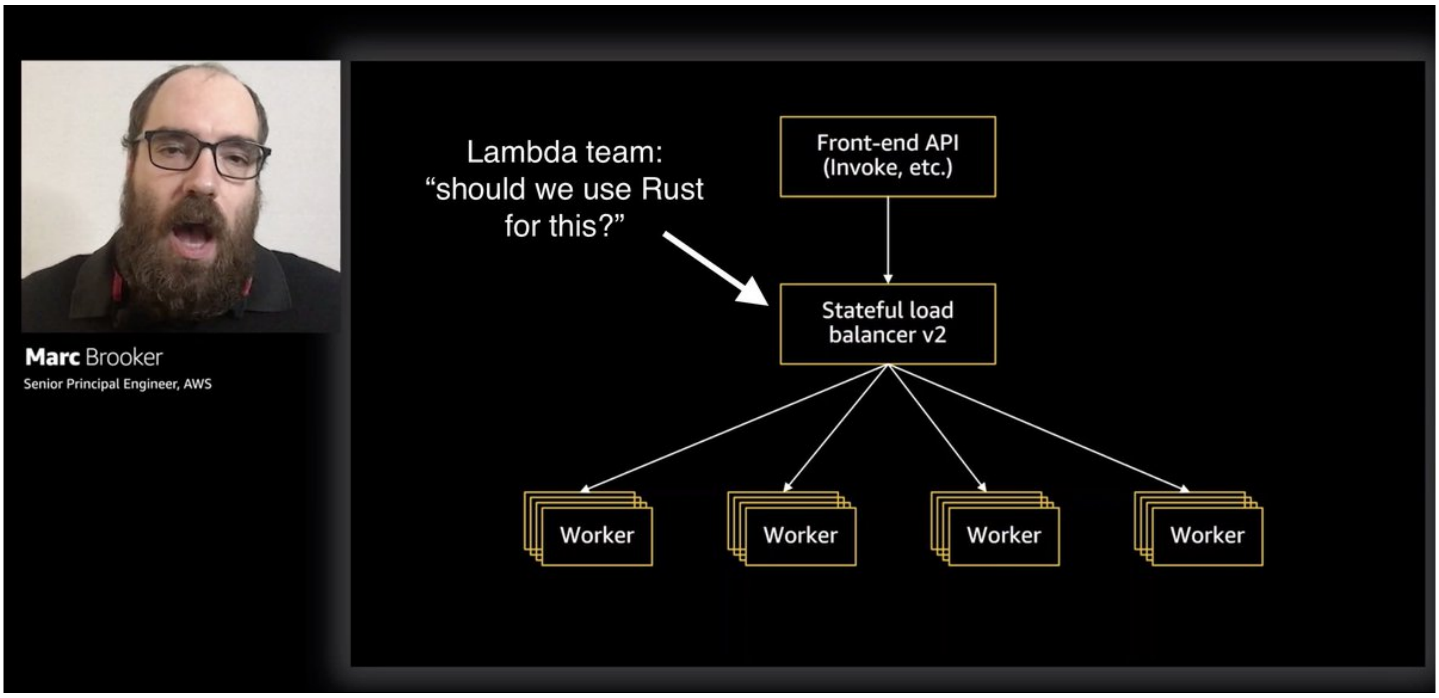
Twitter Thread by Yan Cui is making the AppSync Masterclass

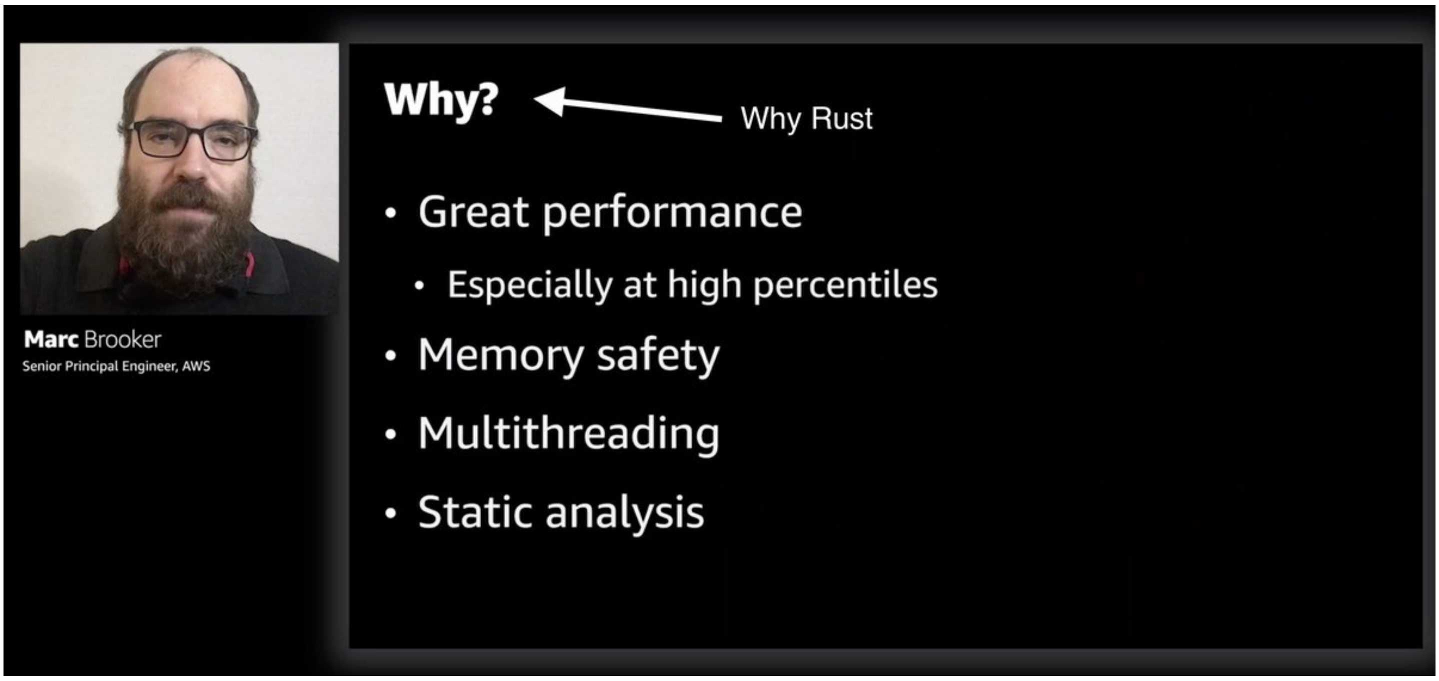
Yan Cui is making the AppSync Masterclass
@theburningmonk

Great session by @MarcJBrooker earlier on building technology standards at Amazon scale, and some interesting tidbits about the secret sauce behind Lambda and how they make technology choices - e.g. in whether to use Rust for the stateful load balancer v2 for Lambda.

■


Marc Brooker
Senior Principal Engineer, AWS
Lambda team: "should we use Rust for this?"
Front-end API (Invoke, etc.)
Stateful load balancer v2
Worker
Worker
Worker
Worker

Nice shout out to some of the benefits of Rust - no GC (good for p99+ percentile latency), memory safety with its ownership system https://t.co/2ShIC786S5 great support for multi-threading (which still works with the ownership system)

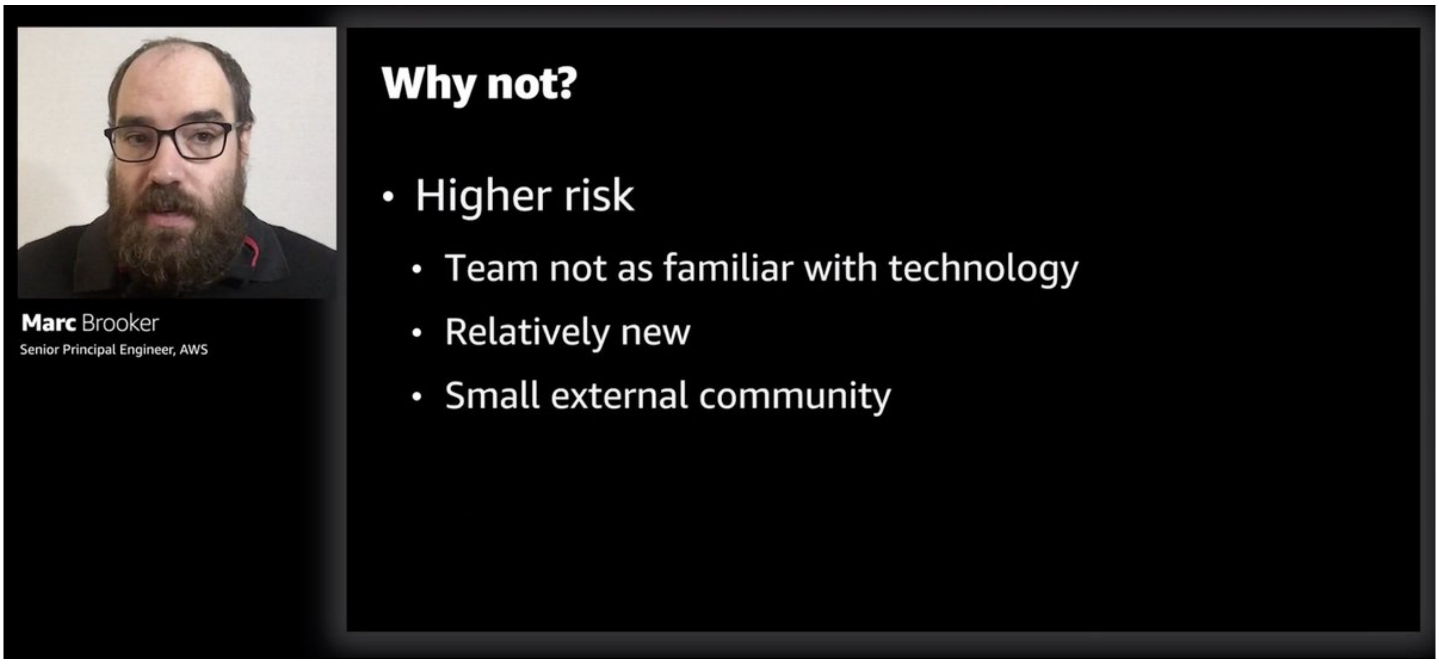


Why? ← Why Rust

- Great performance
  - Especially at high percentiles
- Memory safety
- Multithreading
- Static analysis

And why not to use Rust.

The interesting Q is how to balance technical strengths vs weaknesses that are more organizational.

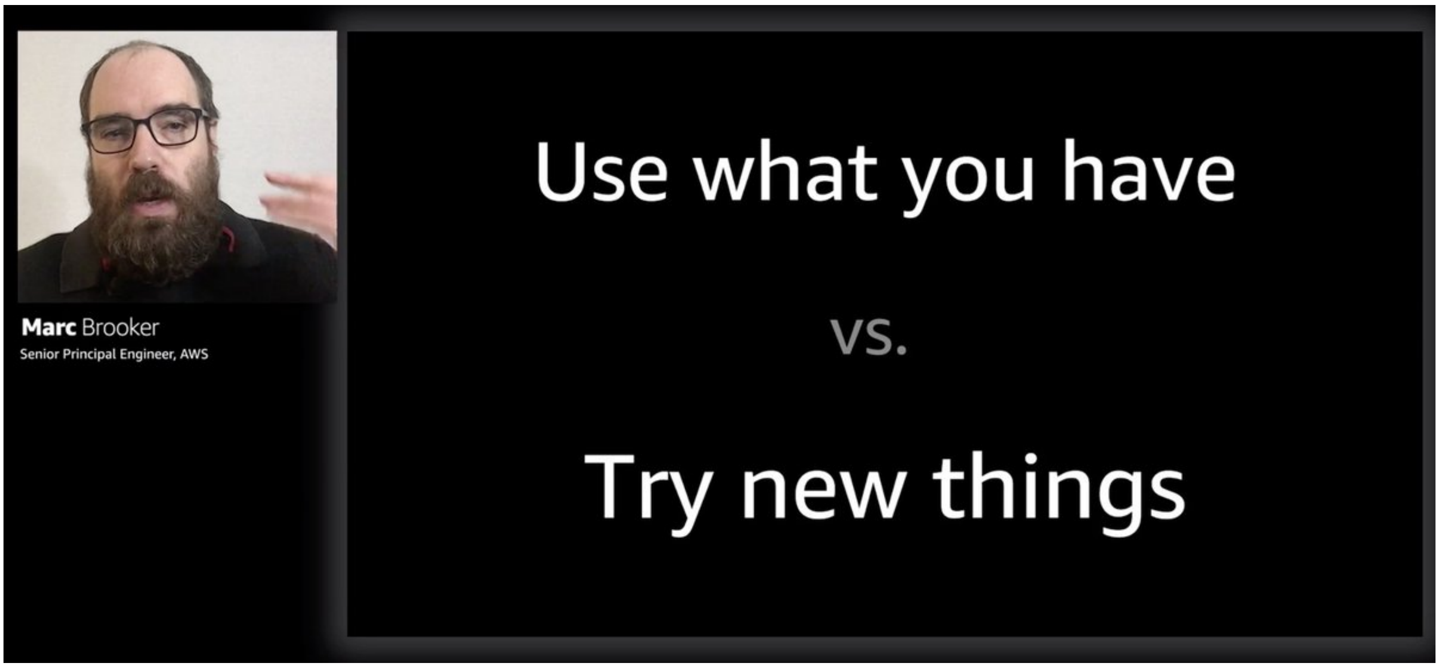


Why not?

- Higher risk
  - Team not as familiar with technology
  - Relatively new
  - Small external community

And it all boils down to this..

which is basically the same question that organizations all over the world have to answer when they consider adopting #serverless technologies like Lambda.


Marc Brooker
Senior Principal Engineer, AWS

Use what you have

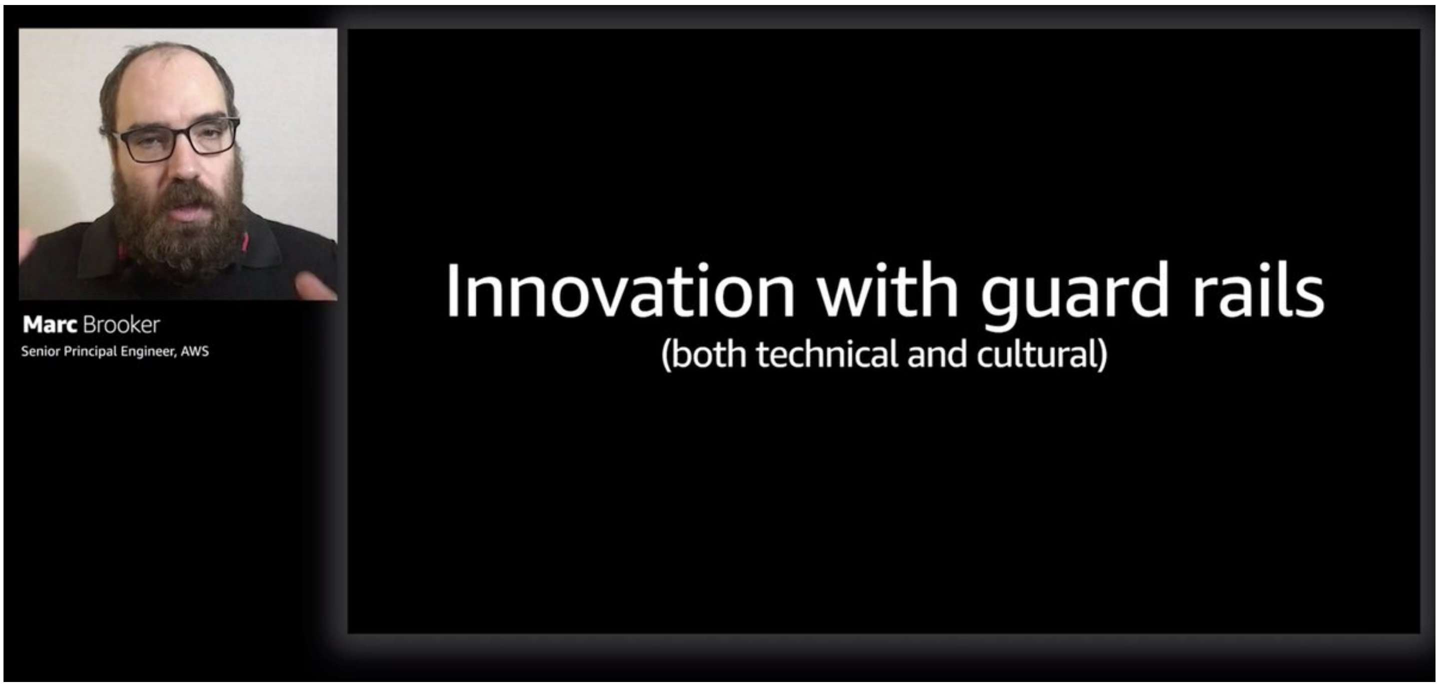
vs.

Try new things

And I love Marc's answer - to innovate (ie. try new things) with guard rails that mitigate the risks.

As a consultant, I often find myself being one of those guard rails for organizations that want to adopt #Serverless

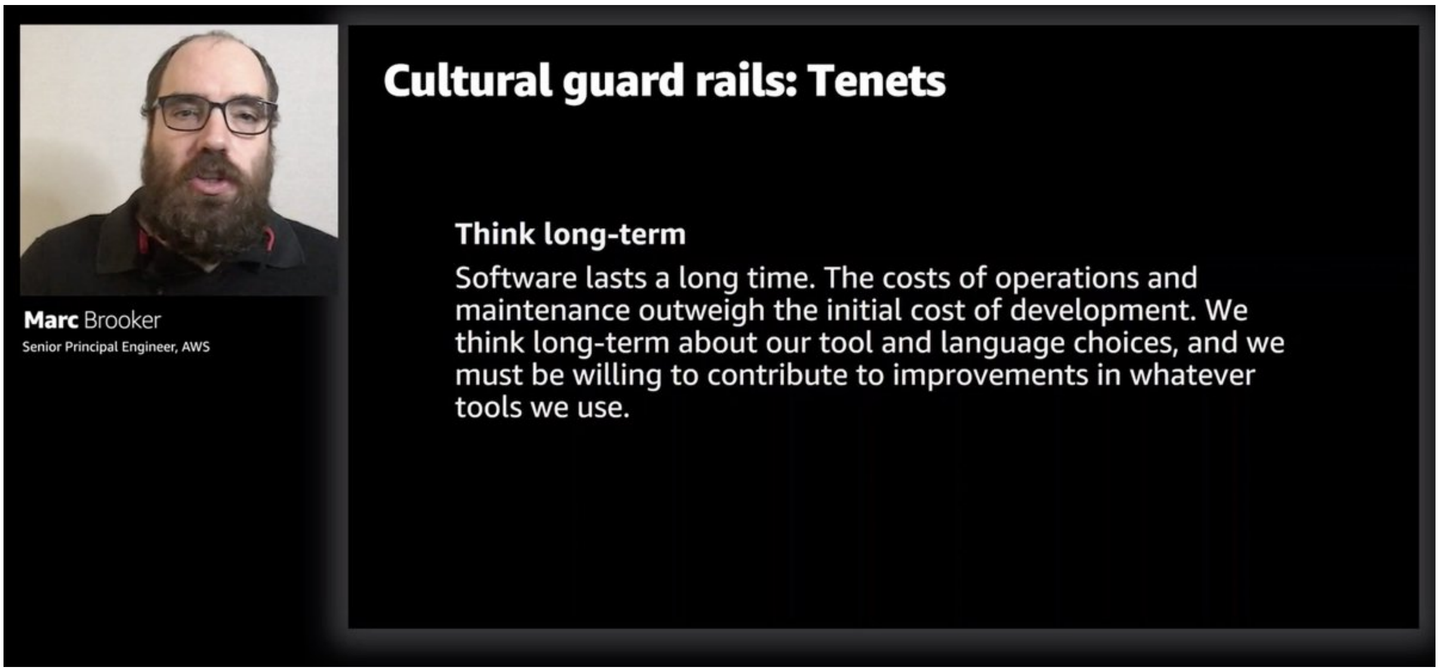
(nice plug, self hi-five! ■)


Marc Brooker
Senior Principal Engineer, AWS

Innovation with guard rails
(both technical and cultural)

Ha, I have heard @heitor_lessa mention "tenets" many times.

This line about avoiding baking language-specific choices into your contract and data is so important. It gives you an easier path to back out of that language choice if it turns out to be wrong.



Cultural guard rails: Tenets

Think long-term

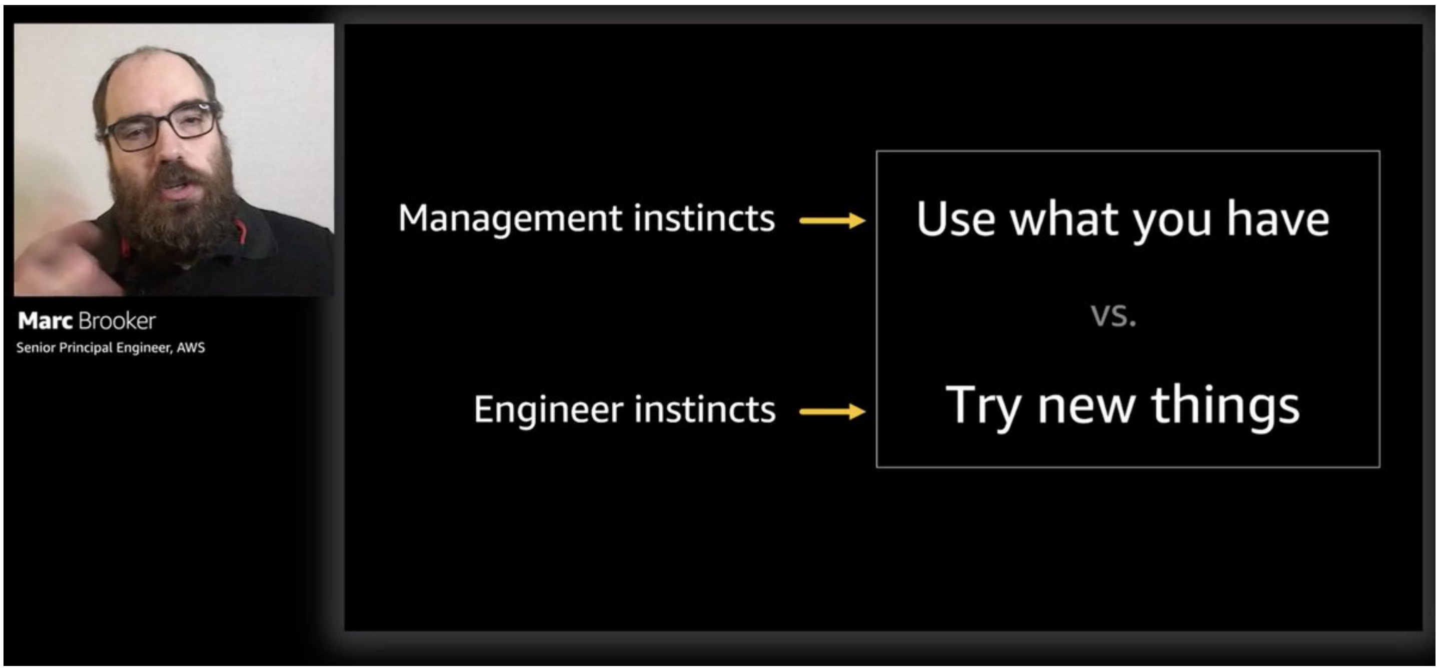
Software lasts a long time. The costs of operations and maintenance outweigh the initial cost of development. We think long-term about our tool and language choices, and we must be willing to contribute to improvements in whatever tools we use.

Which, actually reminds me of what Bezos said in this article about the 2 types of decisions - one-way (aka, "no coming back from this decision!") and two-way doors.

https://t.co/qBh6wgGuz1

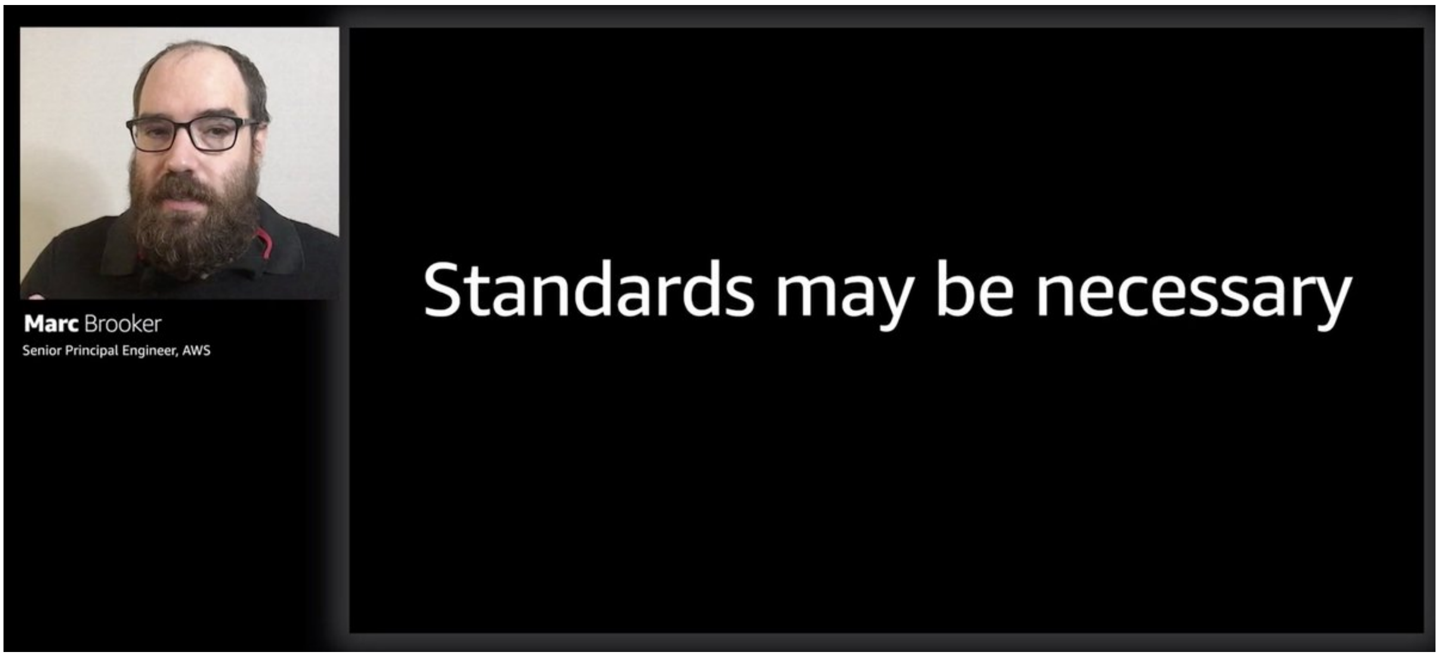
"Baking these tensions into tenets and making it really obvious to everyone means we're upfront about the conversation that we're really having"

■■■



Standards: top-down decision, comes with risk (e.g. limits upside - losing ideas that are better than what's baked into the standards)

"We use standards very sparingly, only in areas where we deeply understand the context and innovation has little upside"


Marc Brooker

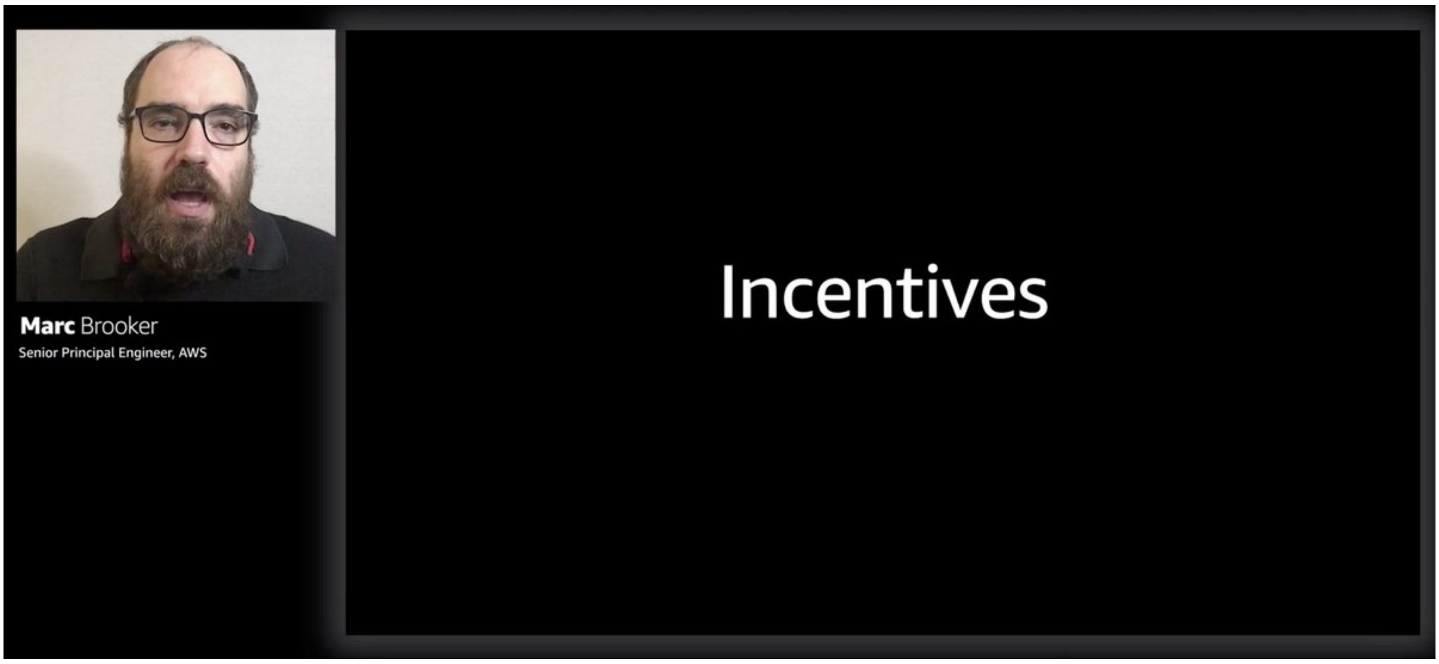
Senior Principal Engineer, AWS

Standards may be necessary

"It all starts with the right incentives"

This, so much this■

Why? because incentives drive outcomes.


Marc Brooker

Senior Principal Engineer, AWS

Incentives

And then there's ownership - because people making these decisions are on the hook for its long term success.

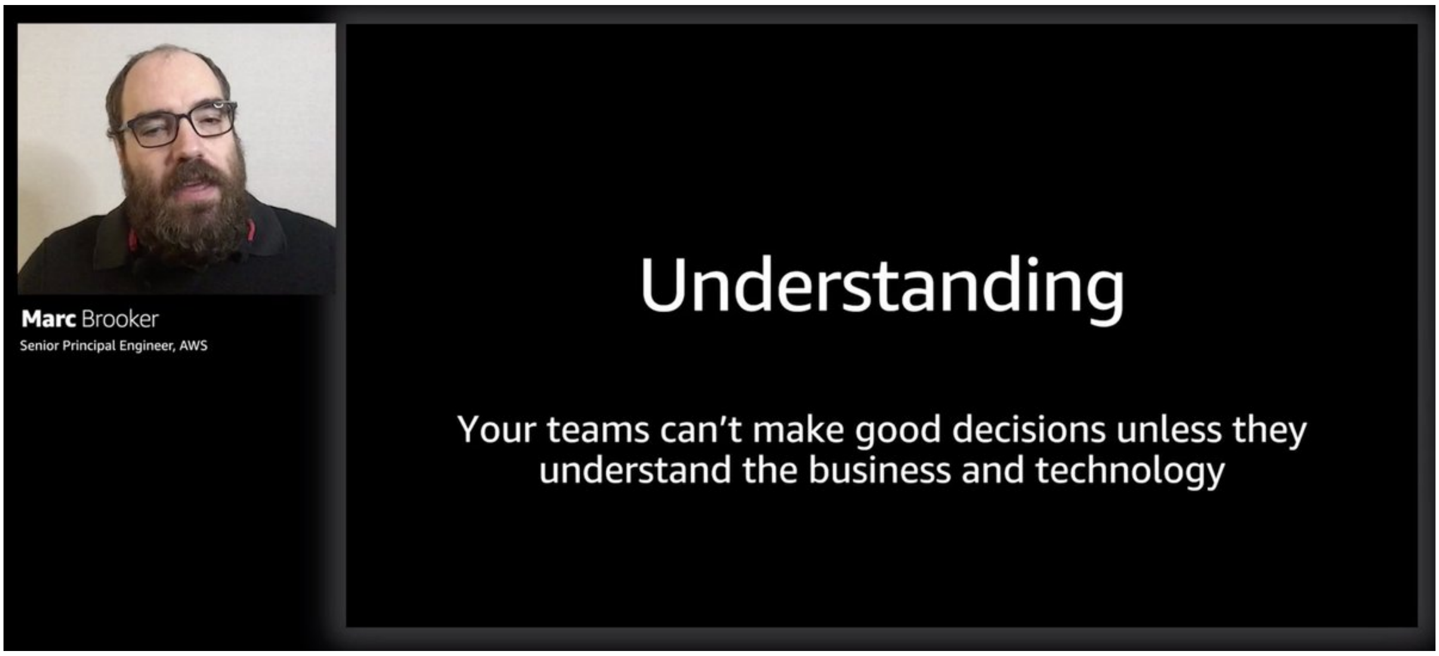
That's why the ivory tower architect is such a bad model - they make all the decisions but you're on the hook for it.

Marc Brooker

Senior Principal Engineer, AWS

Ownership

Yup, 100% agree here. A leader's job is to provide the necessary context so that others can make the best decisions they can. A leader's job is NOT to make all the decisions for others.



Marc Brooker

Senior Principal Engineer, AWS

Understanding

Your teams can't make good decisions unless they understand the business and technology

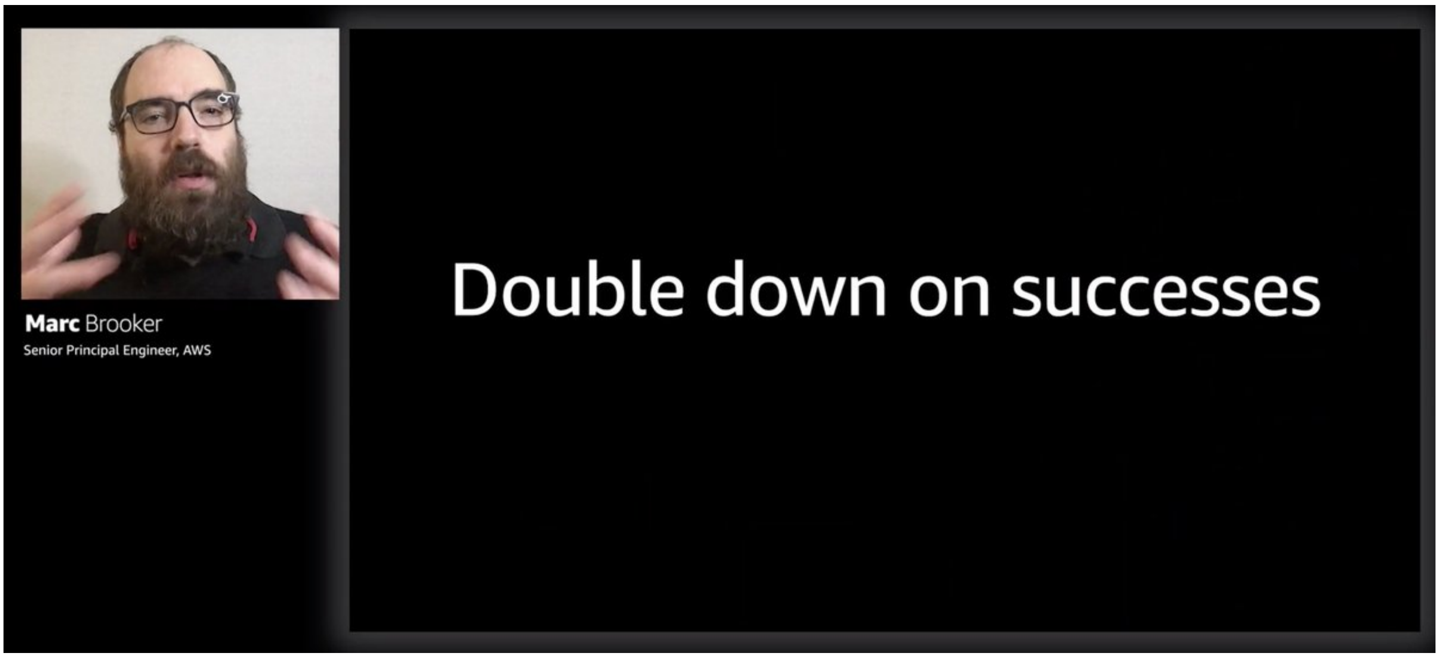
And then Marc describes his job as enabling end-to-end understanding of the business and technology and getting teams talking to each other so they can make the best decisions without those technical standards.

So did they end up using Rust?

Yes!

"When you try new things and they turn out to be successful, then you double down on those. And take the learnings of what's great and make sure you can multiply that"

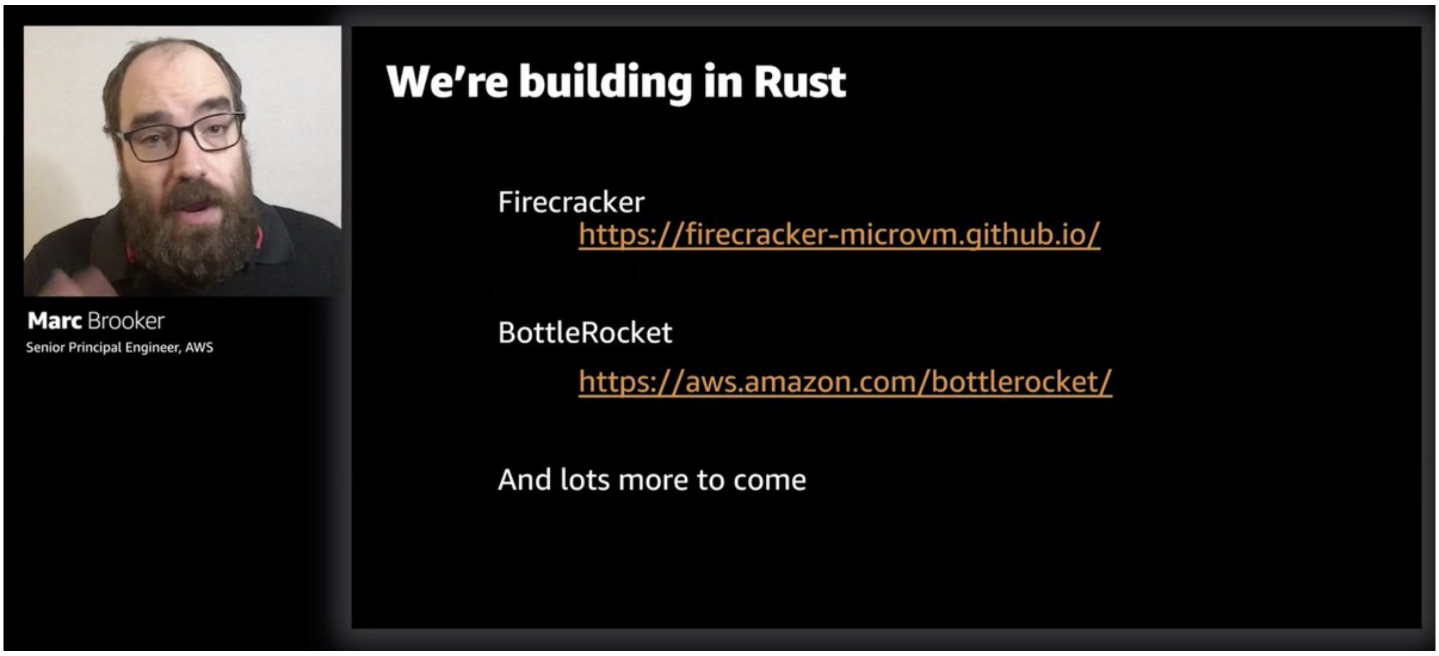
And that's how many organizations has adopted #serverless successfully, starting with one success story.



Marc Brooker
Senior Principal Engineer, AWS

Double down on successes

And that's also been the story of the adoption of Rust at AWS. Both Firecracker and BottleRocket are built with Rust.



Marc Brooker
Senior Principal Engineer, AWS

We're building in Rust

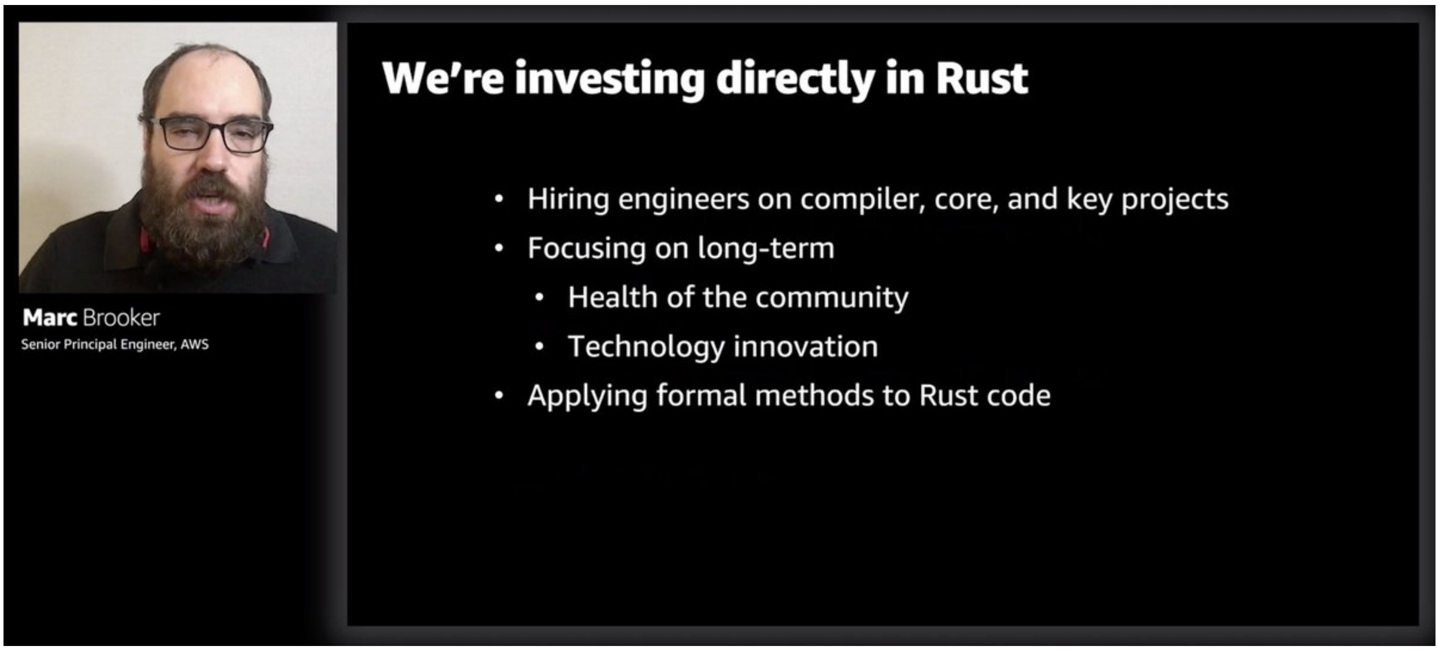
Firecracker
https://firecracker-microvm.github.io/

BottleRocket
https://aws.amazon.com/bottlerocket/

And lots more to come

And great to see they're investing into the community itself, doubling down both internally and externally.

Love to see more details on how formal methods is applied here.



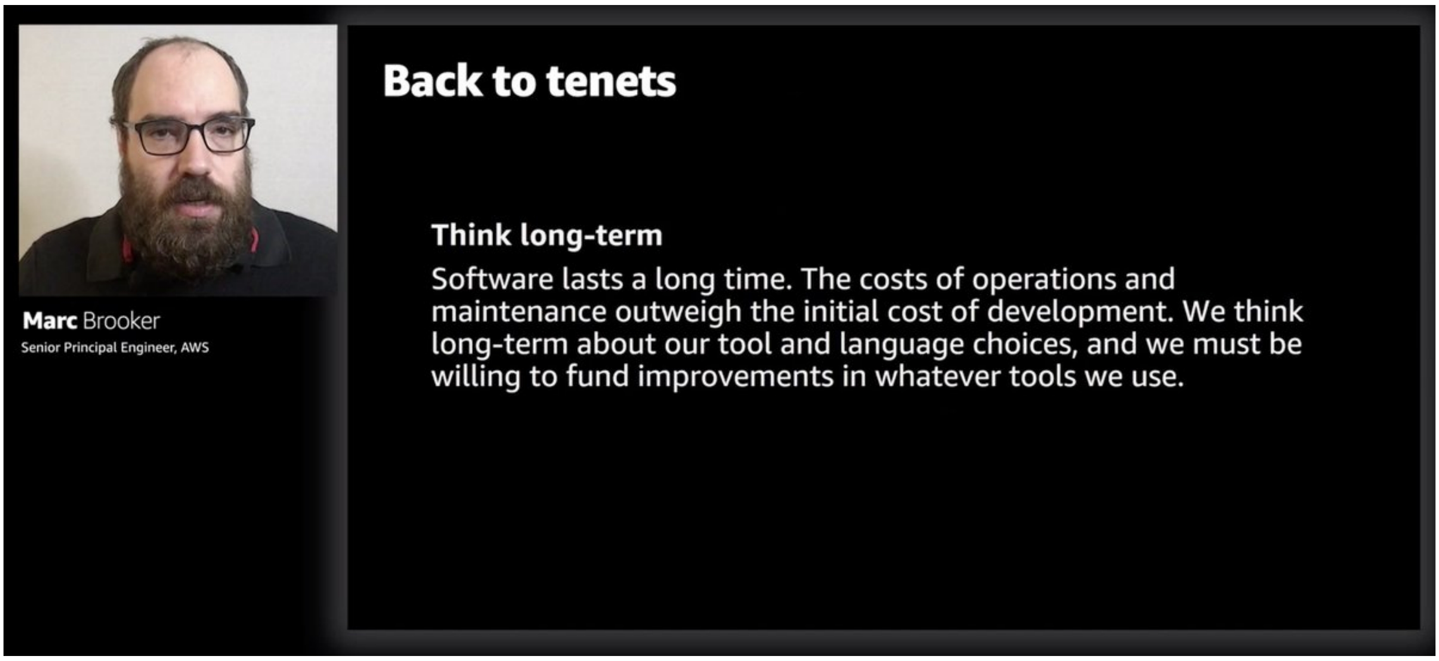
We're investing directly in Rust

- Hiring engineers on compiler, core, and key projects
- Focusing on long-term
  - Health of the community
  - Technology innovation
- Applying formal methods to Rust code

btw, AWS uses formal methods all over the place, I hear that TLA+ is widely used by its service teams. Someone told me that they used it to find a bug in DynamoDB during design that would have resulted in data loss in extremely rare cases.

https://t.co/6p5MfXCyfR

"Building technology standards is a short-term thing that limits a company's creativity. Setting up incentives and helping people understand the decisions they're making and giving them full ownership of those decisions is the way I like to think about tech standards."



Back to tenets

Think long-term
Software lasts a long time. The costs of operations and maintenance outweigh the initial cost of development. We think long-term about our tool and language choices, and we must be willing to fund improvements in whatever tools we use.

Well, that was great!

Make sure to catch this on a replay or when it becomes available on-demand.