AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

OPN301

Using Rust to minimize environmental impact

Shane Miller (she/her)
Senior Engineering Manager
Amazon Web Services

Carl Lerche (he/him)
Principal Engineer
Amazon Web Services

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

What are we going to talk about?

Energy efficiency in the cloud

Energy-efficient programming languages

Rust customer success stories

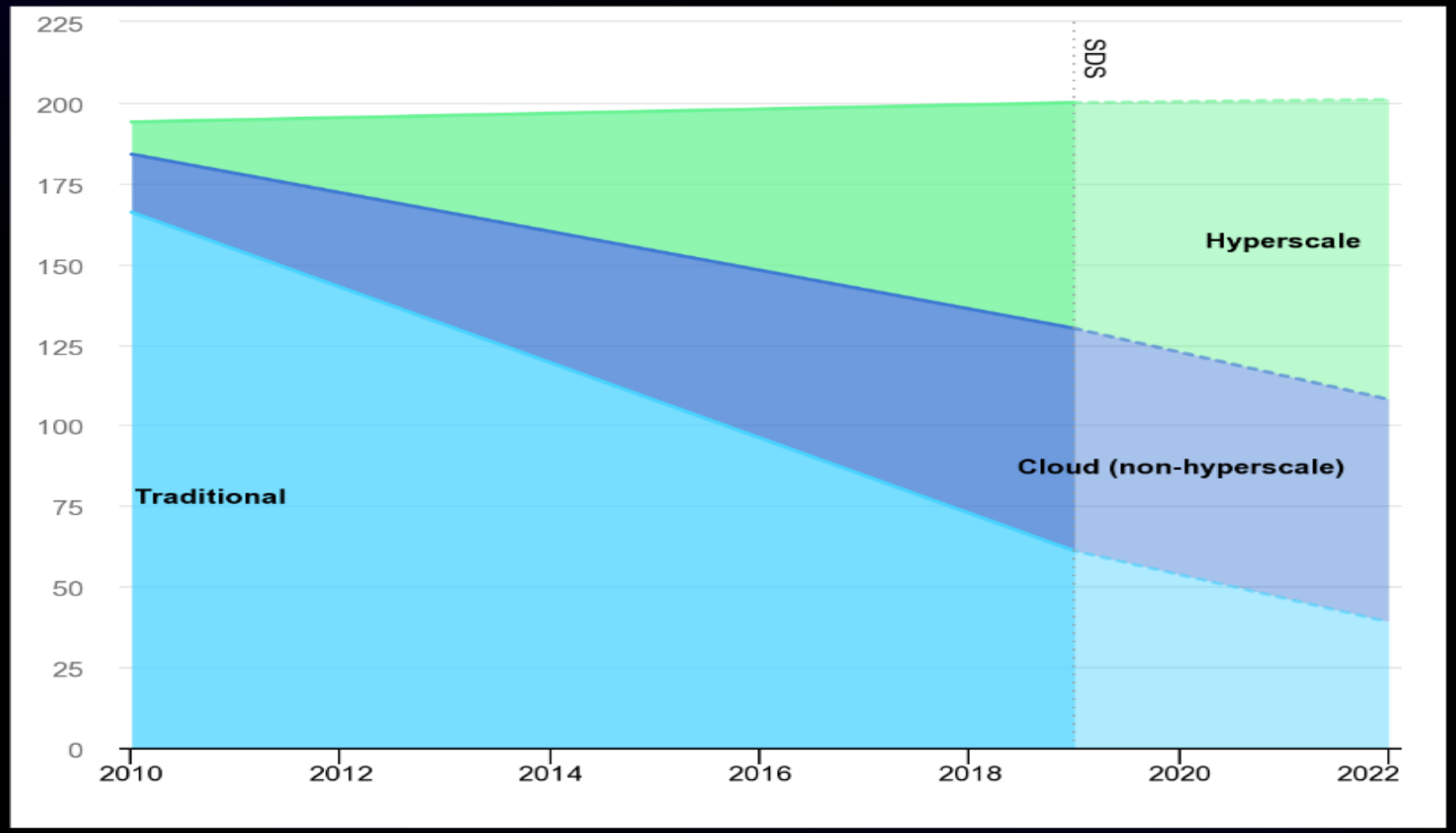
Rust secret sauce

Getting started with Rust

What’s next for Rust

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data center energy use



Source: IEA (2021), **Global data centre energy demand by data centre type, 2010-2022,** https://www.iea.org/data-and-statistics/charts/global-data-centre-energy-demand-by-data-centre-type-2010-2022. All rights reserved.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data center efficiency improvements

Computation reductions

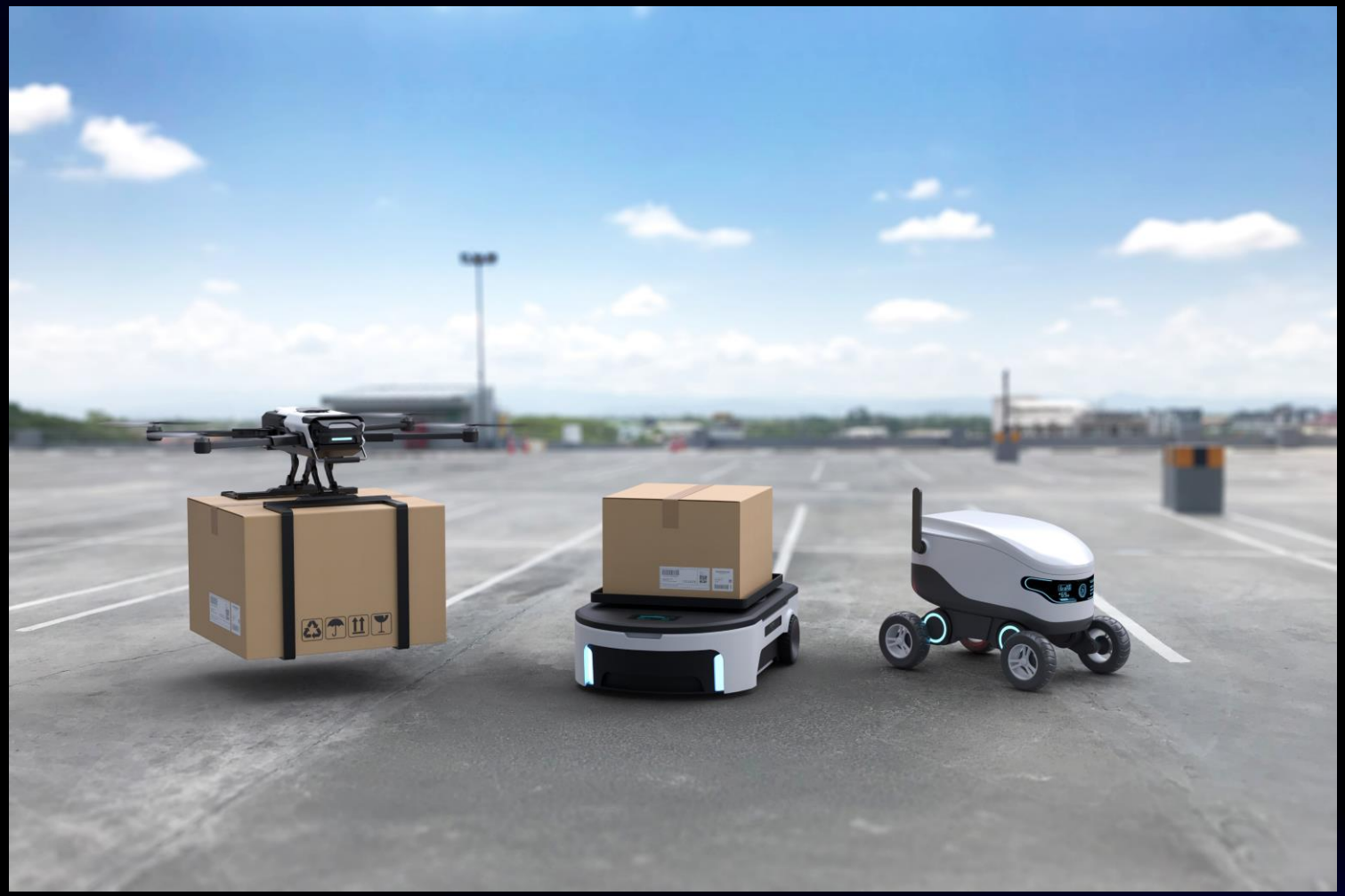
Storage reductions

Server growth reductions

Data center power use reductions

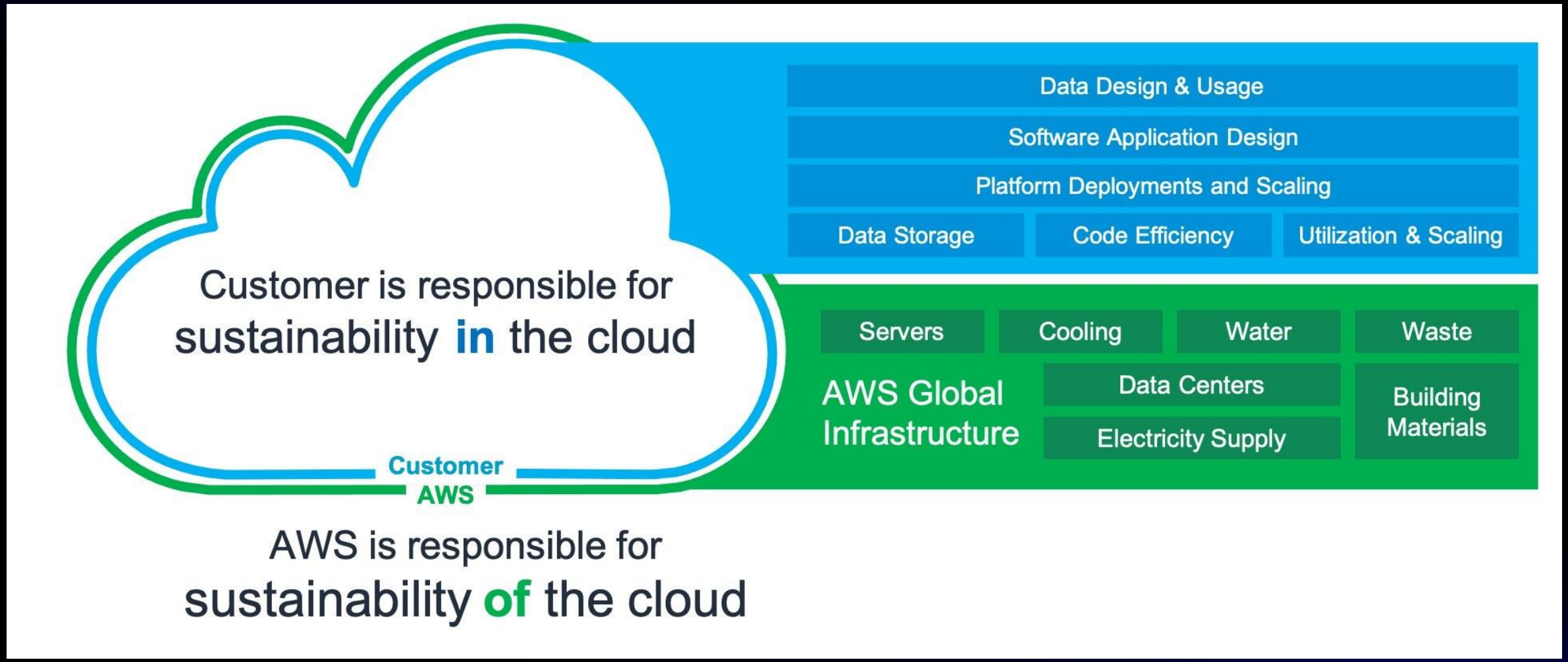
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Forecasting the future



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Sustainability is a shared responsibility



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

100% renewable energy



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“The greenest energy is the energy we don’t use.”

Peter DeSantis

SVP, Amazon Web Services

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Sustainability as a design principle

Relax SLAs for non-critical functions

Allow for longer device upgrade cycles when possible

Use caching

Classify data and implement automated lifecycle policies

Choose efficient algorithms for cryptography, compression, etc.

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Implement software in energy-efficient programming languages

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Energy efficiency across programming languages

10 benchmark problems from The Computer Language Benchmarks Game (CLBG)

27 programming languages

Measuring execution time, energy consumption, and peak memory usage

https://greenlab.di.uminho.pt/wp-content/uploads/2017/10/sleFinal.pdf

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

C and Rust for energy efficiency

	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“[C’s subtle type interactions] are not always logical [and] are pitfalls for pretty much anybody.”

Linus Torvalds

Creator of Linux

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Safety with Rust

More than 70% of the high-severity CVEs that occur in C/C++ would be prevented by implementing those same solutions in Rust

The Internet Security Research Group (ISRG)’s goal is to move the internet’s security-sensitive software infrastructure to memory safe code (Rust)

memorysafety.org

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Performance with Rust

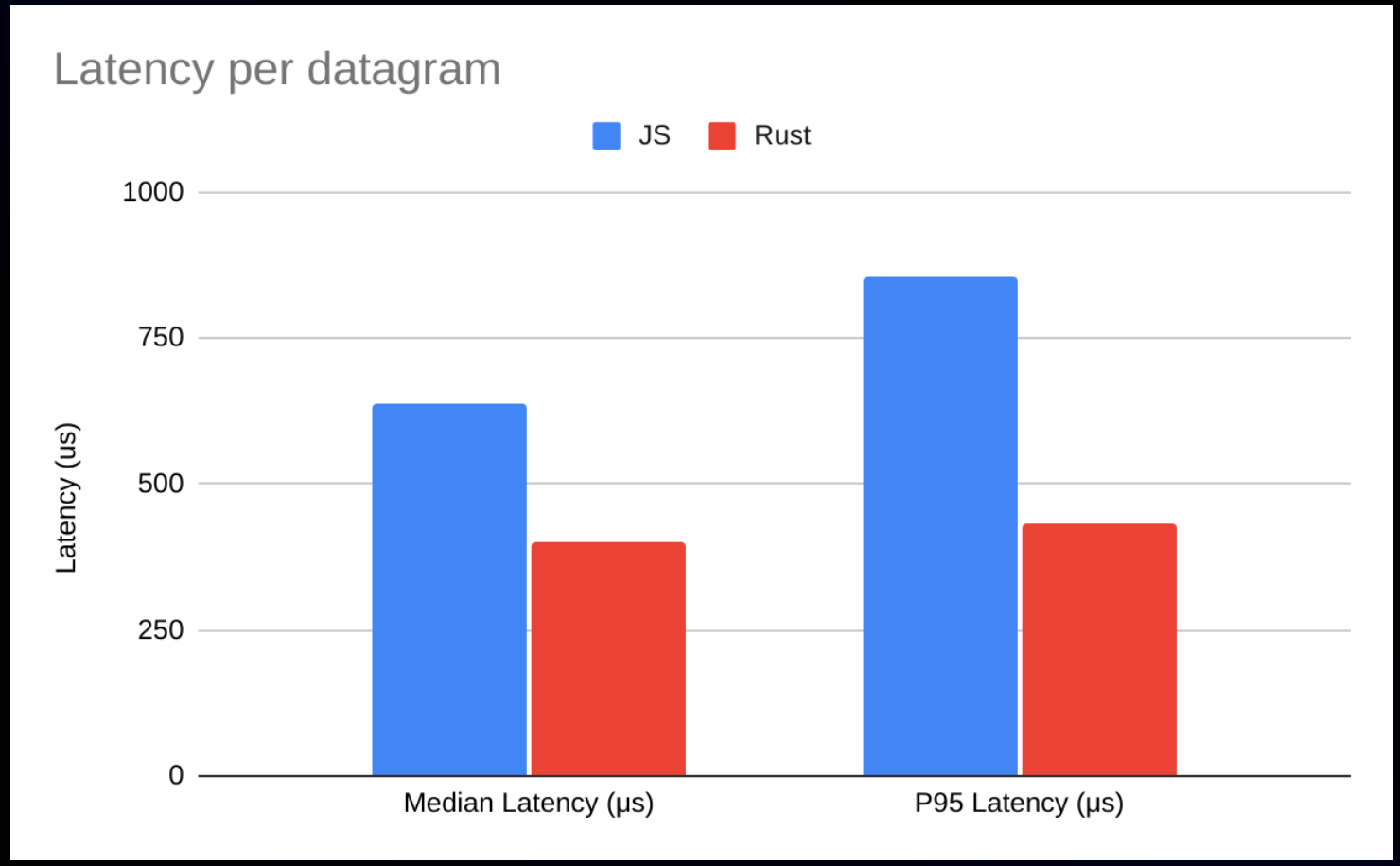
	Energy
(c) C	1.00
(c) Rust	1.03
(c) C++	1.34
(c) Ada	1.70
(v) Java	1.98
(c) Pascal	2.14
(c) Chapel	2.18
(v) Lisp	2.27
(c) Ocaml	2.40
(c) Fortran	2.52
(c) Swift	2.79
(c) Haskell	3.10
(v) C#	3.14
(c) Go	3.23
(i) Dart	3.83
(v) F#	4.13
(i) JavaScript	4.45
(v) Racket	7.91
(i) TypeScript	21.50
(i) Hack	24.02
(i) PHP	29.30
(v) Erlang	42.23
(i) Lua	45.98
(i) Jruby	46.54
(i) Ruby	69.91
(i) Python	75.88
(i) Perl	79.58

	Time
(c) C	1.00
(c) Rust	1.04
(c) C++	1.56
(c) Ada	1.85
(v) Java	1.89
(c) Chapel	2.14
(c) Go	2.83
(c) Pascal	3.02
(c) Ocaml	3.09
(v) C#	3.14
(v) Lisp	3.40
(c) Haskell	3.55
(c) Swift	4.20
(c) Fortran	4.20
(v) F#	6.30
(i) JavaScript	6.52
(i) Dart	6.67
(v) Racket	11.27
(i) Hack	26.99
(i) PHP	27.64
(v) Erlang	36.71
(i) Jruby	43.44
(i) TypeScript	46.20
(i) Ruby	59.34
(i) Perl	65.79
(i) Python	71.90
(i) Lua	82.91

	Mb
(c) Pascal	1.00
(c) Go	1.05
(c) C	1.17
(c) Fortran	1.24
(c) C++	1.34
(c) Ada	1.47
(c) Rust	1.54
(v) Lisp	1.92
(c) Haskell	2.45
(i) PHP	2.57
(c) Swift	2.71
(i) Python	2.80
(c) Ocaml	2.82
(v) C#	2.85
(i) Hack	3.34
(v) Racket	3.52
(i) Ruby	3.97
(c) Chapel	4.00
(v) F#	4.25
(i) JavaScript	4.59
(i) TypeScript	4.69
(v) Java	6.01
(i) Perl	6.62
(i) Lua	6.72
(v) Erlang	7.20
(i) Dart	8.64
(i) Jruby	19.84

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tenable performance improvements



https://medium.com/tenable-techblog/optimizing-700-cpus-away-with-rust-dc7a000dbdb2

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tenable infrastructure savings



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Using Rust at AWS

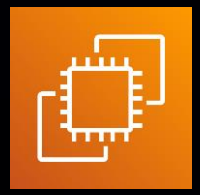
Firecracker



Amazon S3



Amazon EC2



Amazon CloudFront



Bottlerocket



aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Contributing to Rust at AWS

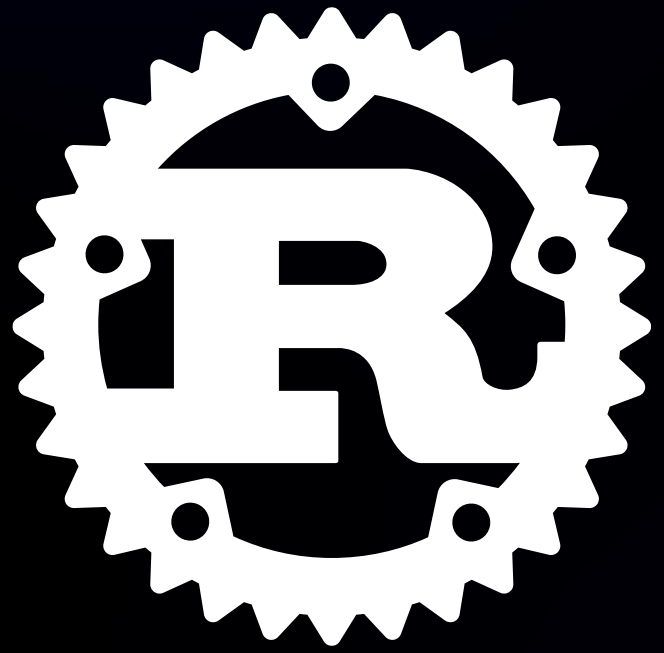
At AWS, we believe, “Leaders create more than they consume and always leave things better than how they found them”

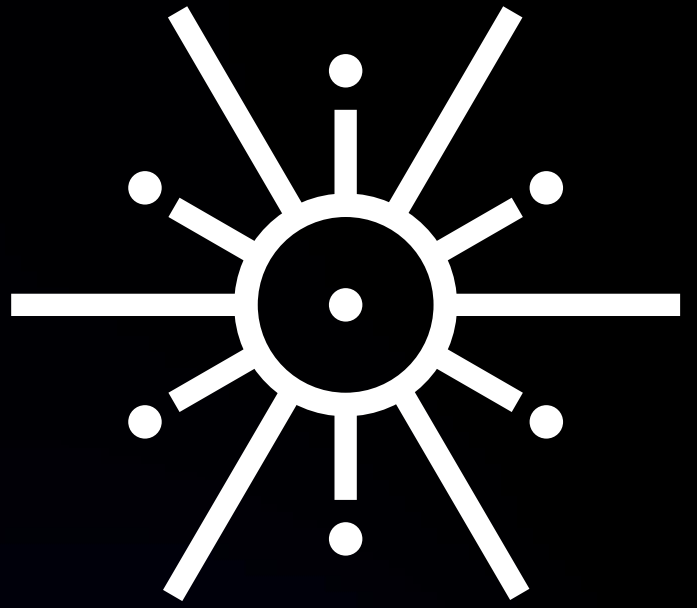
Proud sponsor of the Rust project since 2019

Started hiring Rust maintainers and contributors in 2020

Founding member of the Rust Foundation in 2021

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.





aws

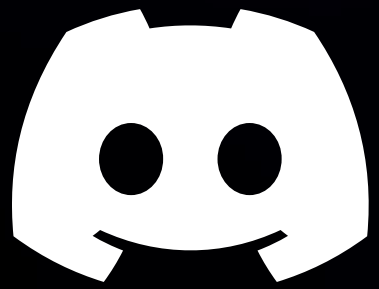
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

1% of all energy

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why is Rust used today?

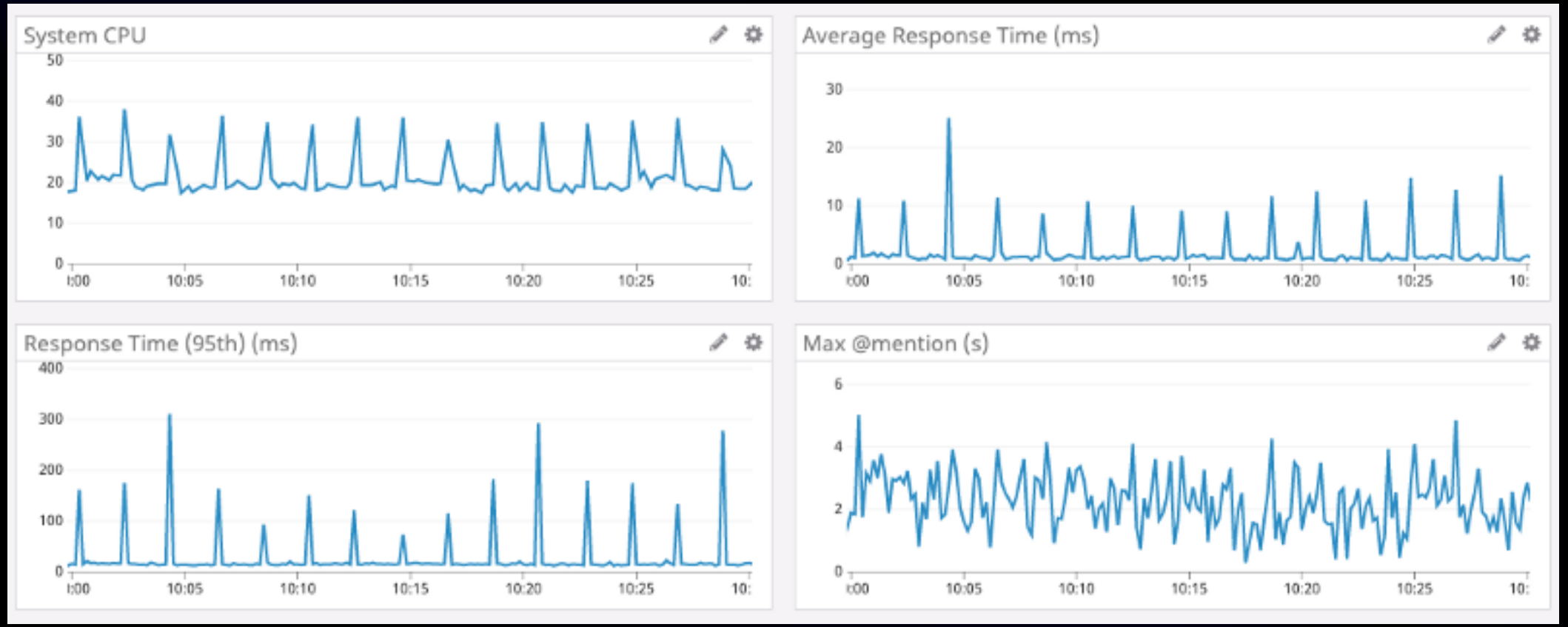
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Discord

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Discord



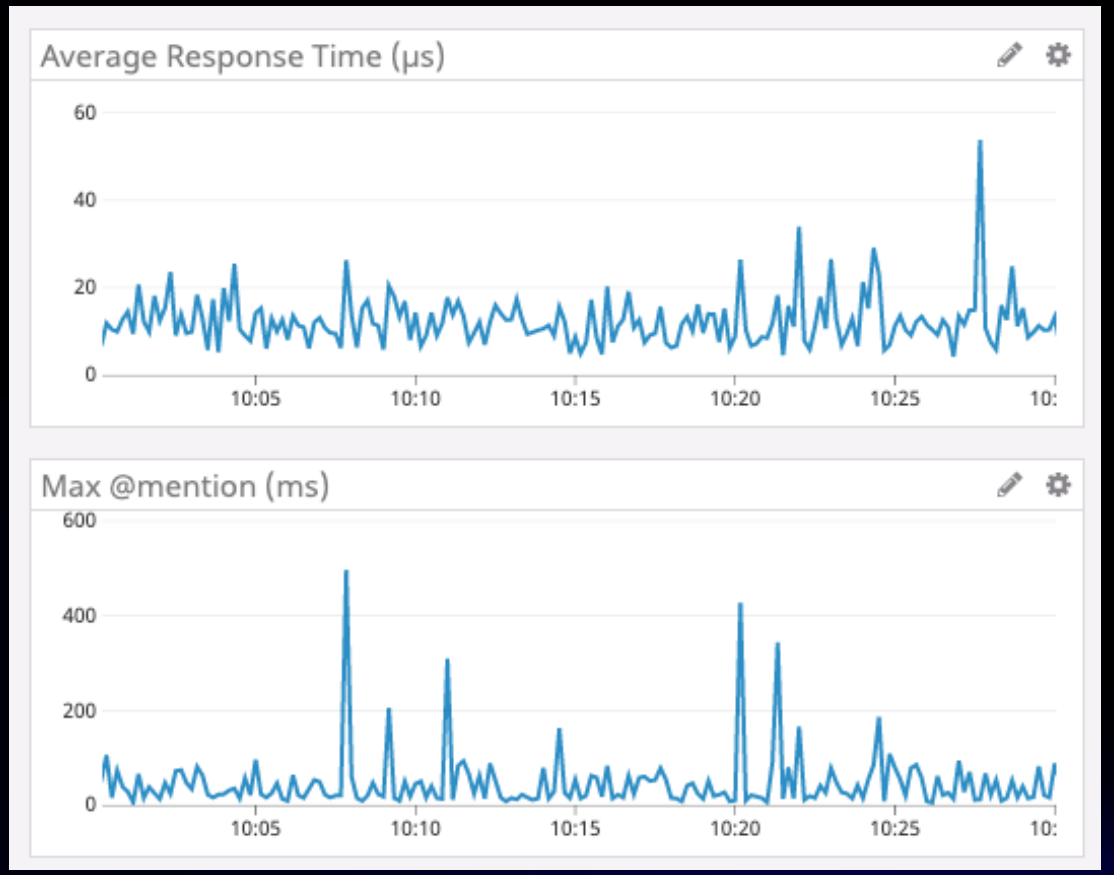
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Discord

Go



Rust



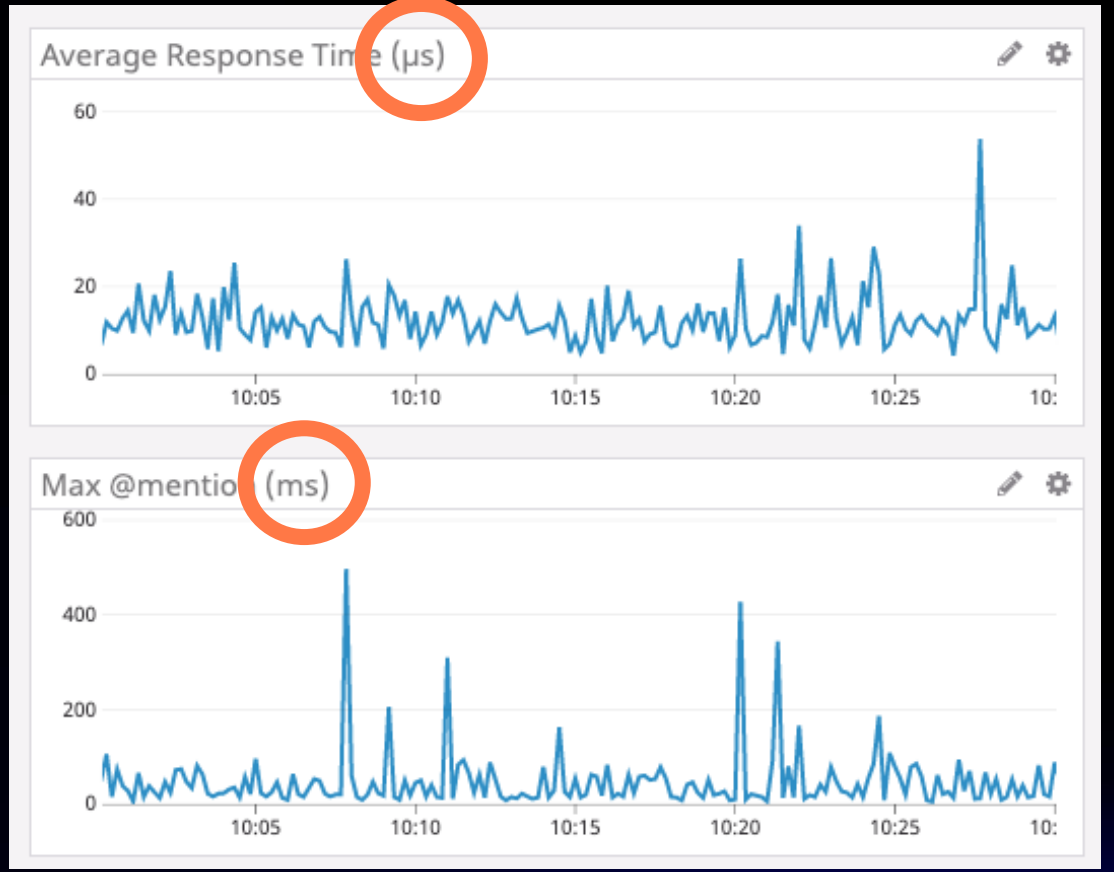
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Discord

Go



Rust



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ownership and borrow checking

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message passing with Go

```
gift := Gift { .. }
channel <- gift;
gift.open(); // oops
```

```
// The other goroutine
gift := <- channel
gift.open()
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Message passing with Rust

```
let gift = Gift::new();
channel.send(gift);

// ERROR: gift was already moved
gift.open();
```

```
let gift = channel.recv();
gift.open();
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Dropping

```
let gift = Gift::new();
println!("Look at my gift: {}", gift);
// Gift automatically freed
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Affine types

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Borrow checking

```
fn admire(gift: Gift) -> Gift {
    println!("wow, this {} looks nice!", gift);
    gift
}

let gift = Gift::new();
let gift = admire(gift);
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Borrow checking

```
fn admire(gift: &Gift) {
    println!("wow, this {} looks nice!", gift);
}

let gift = Gift::new();
admire(&gift);
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Borrow checking

Rust

```
let mut vec = vec![1];
let entry = &vec[0];

// ERROR: cannot borrow `vec`
vec.push(2);

println!("{}", *entry);
```

C++

```
std::vector<int> my_vec = { 1 };
int* entry = &my_vec[0];

my_vec.push_back(2);

// Outputs 0… sometimes
std::cout << *entry;
```

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fearless concurrency



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Fearless concurrency

Synchronous

```
fn sum_odd(nums: &[u32]) -> u32 {
    nums.iter()
        .filter(|i| *i % 2 == 0)
        .sum()
}
```

Concurrent

```
use rayon::prelude::*;

fn sum_odd(nums: &[u32]) -> u32 {
    nums.par_iter()
        .filter(|i| *i % 2 == 0)
        .sum()
}
```

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Getting started

Learn: rust-lang.org/learn

Forum: users.rust-lang.org

Discord: discord.gg/rust-lang-community

Crust of Rust: youtube.com/c/JonGjengset

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tokio

Learn: https://tokio.rs

Examples: github.com/tokio-rs/tokio

Mini Redis: github.com/tokio-rs/mini-redis

Discord: discord.gg/Tokio

OPN304: Async Rust: Hurry up and .await!

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Rust: Good for everything?

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reduce energy usage by 50%

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Learning curve

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“Rust can be difficult and frustrating to learn.”

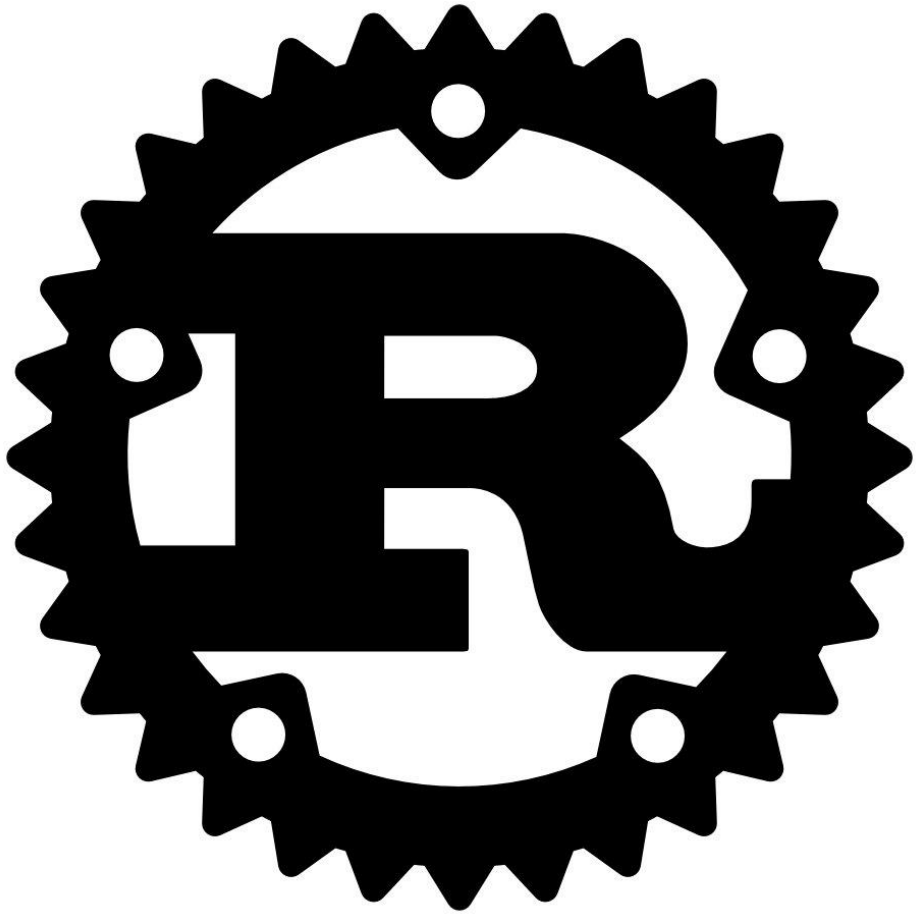
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“I am a beginner and am getting frustrated with Rust’s learning curve.”

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The cost of adopting Rust is still high

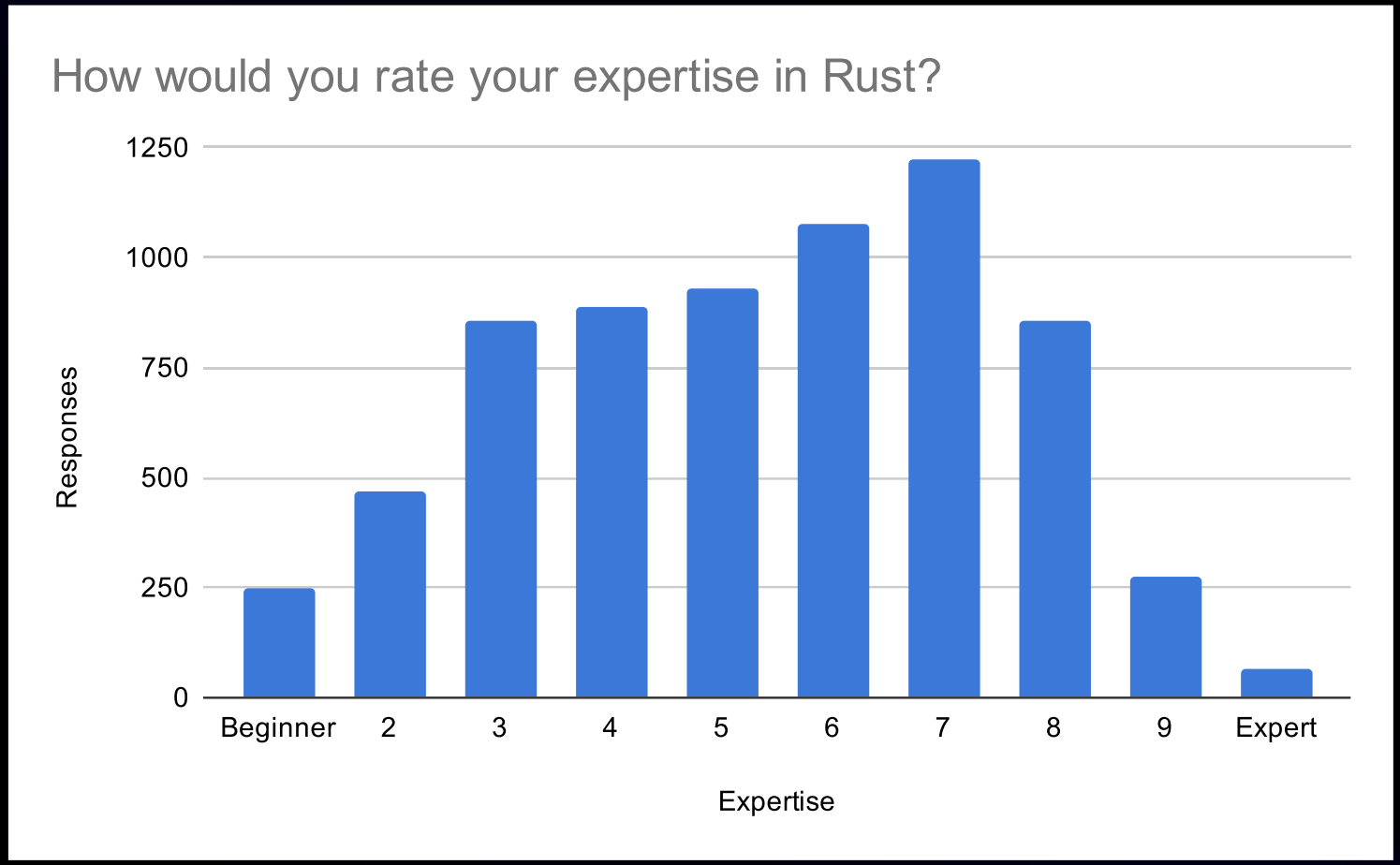
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.



The Rust Programming Language

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Productivity



https://blog.rust-lang.org/2020/12/16/rust-survey-2020.html

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Productivity is important to sustainability



SlashData, State of the Developer Nation Q3 2021

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Which developer’s experience?



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

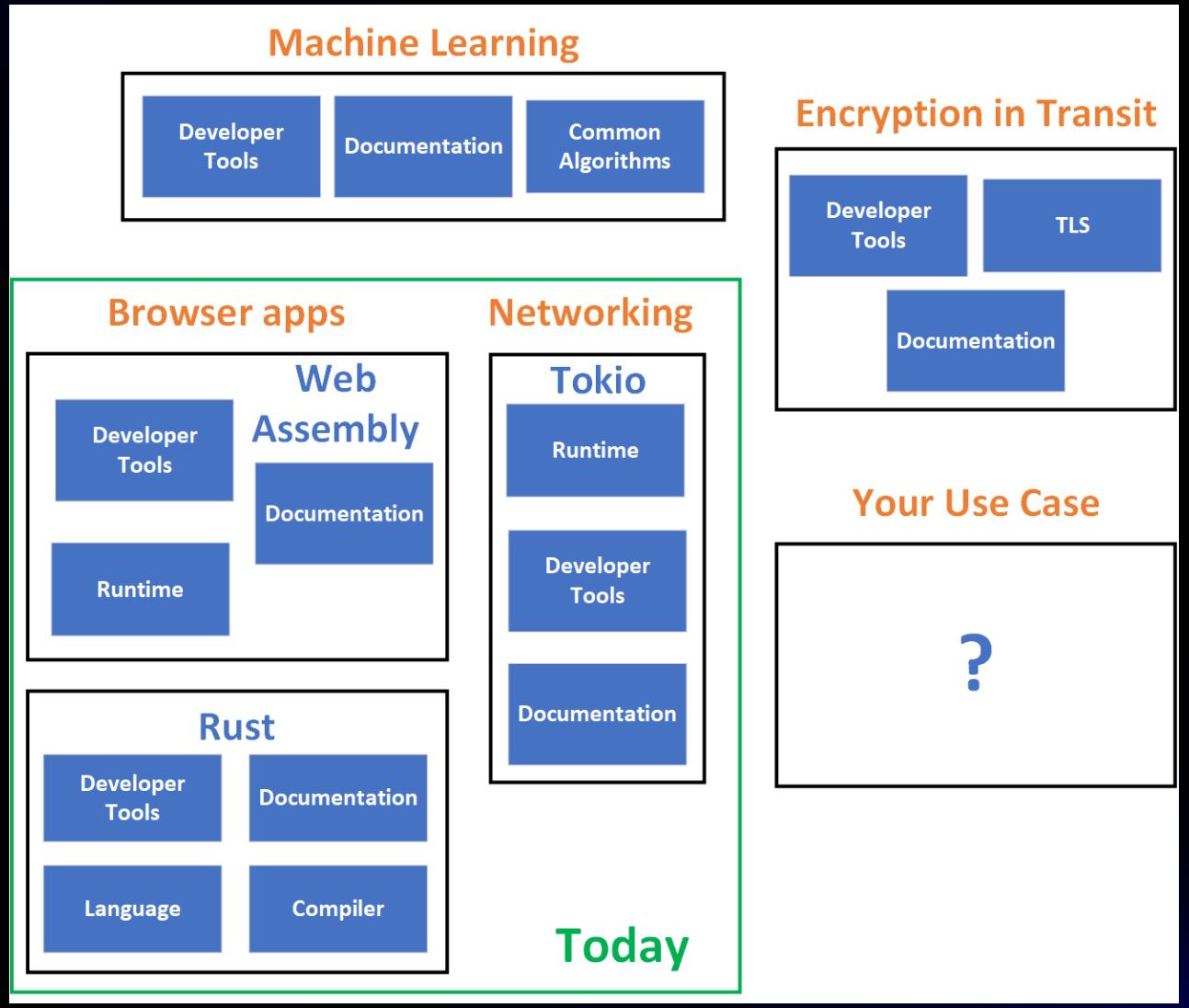
Personas and principles



https://www.freepik.com/vectors/icons Icons vector created by gstudioimagen

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The ecosystem for everyone



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The moral of the story

Rust is an amazing technology to secure and sustain our industry

You can start using Rust and make a difference **today**

Rust has a lot of work to do before everyone can use it

The Rust Foundation is working to create the platform for people across our industry to effectively collaborate on Rust

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Shane Miller

Twitter: @SkippersWif

Carl Lerche

Twitter: @carllerche



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.