

Rust

Le langage des 10 prochaines années

Salut !



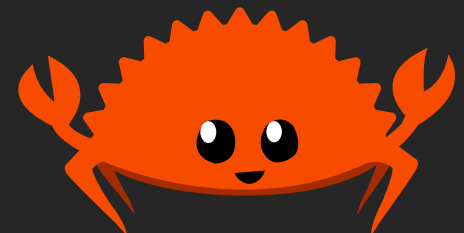
- **Architecte solution chez Owlnext**
- **Auparavant Head of software & sytem architecture chez iBanFirst**
- **Quelques réalisations**
 - Conception d'un système de chiffrement end-to-end : Heimdall
 - Création d'un système de téléphonie VoIP via WebRTC avec IVRs, Files d'attentes, redirections d'appels, connexions CRM, etc.
 - Refactoring d'un serveur de connexion aux salles de marchés internationale

Rust, c'est quoi ?



Rust

- **Remplaçant du C/C++**
 - Performances des langages bas-niveau
 - Concepts des langages haut-niveau
 - Sécurité
- **Utilisé principalement pour**
 - Des applications systèmes, des backends, des CLI
 - Du web avec WASM
 - Des services réseaux
 - De l'informatique embarqué
- **Créé en 2006 par Graydon Hoare, puis repris à partir de 2010 par la fondation Mozilla**
- **Version actuelle : 1.65.0 (2 Novembre 2022)**
- **Sa mascotte : Ferris**



Pourquoi choisir Rust ?

Il a :

- Un typage de données algébrique
- La validation de l'exécution au moment de la compilation
- De la métaprogrammation
- Un transpileur natif et intégré
- Une intégration des MONADS
- Une compatibilité native avec C

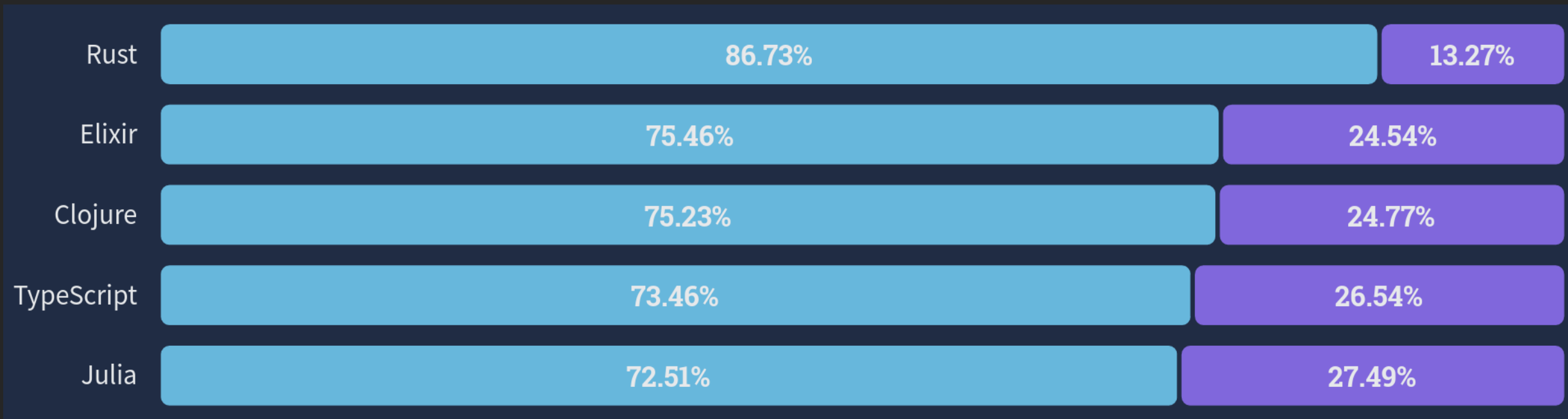
Il n'a pas :

- De surprises à l'exécution
- De classes
- D'environnement d'exécution ou de VM
- De transformation type « Bytecode »
- De compilation JIT
- De garbage collector
- De valeurs « NULL »

Les principales caractéristiques de Rust

- **Orientation bas-niveau**
 - Gestion de la mémoire via le mécanisme d'emprunt
 - Gestion des pointeurs et smart-pointers
- **Thread sécurisés intégrés directement au langage**
 - Validation des mémoires partagés à la compilation
- **Sécurisation & compilation**
 - Typage strict
 - Accès mémoires et emprunts validés à la compilation
 - Variables immutables par défaut
 - Pas de NULL
 - Inférence de type
- **Filtrage par motifs et Foncteurs**
 - `Option<T>`, `Result<T, E>`, `Futur<T>`, etc.
- **Généricité et métaprogrammation**

We ❤️ Rust



<https://survey.stackoverflow.co/2022/#technology-most-loved-dreaded-and-wanted>

On se lance ?



Installation



```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

L'IDE

- **Visual Studio Code est l'IDE de choix pour Rust**
 - Gratuit
 - Possède toute la suite d'outil pour travailler avec Rust
 - Configuration en 2 minutes



Hello world !



```
/// hello-world.rs  
fn main() {  
    println!("Hello, world! 🦀");  
}
```

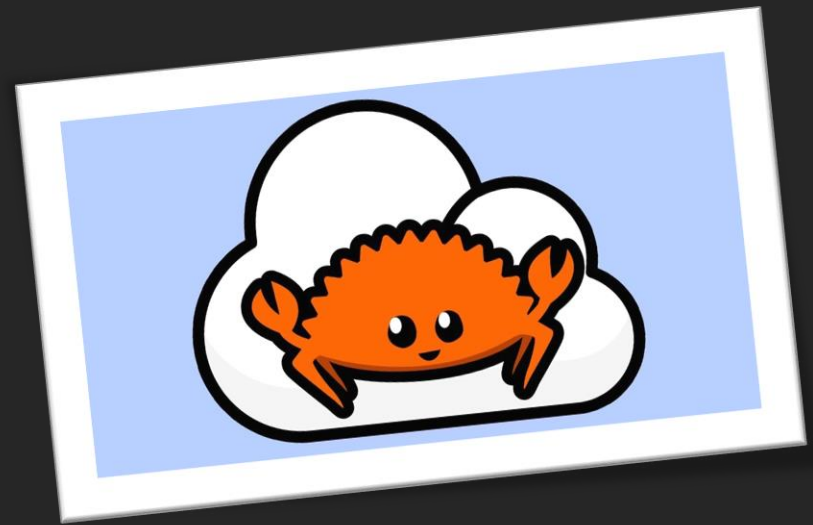


```
# compilation  
rustc ./hello-world.rs  
  
# lancement  
./hello-world
```



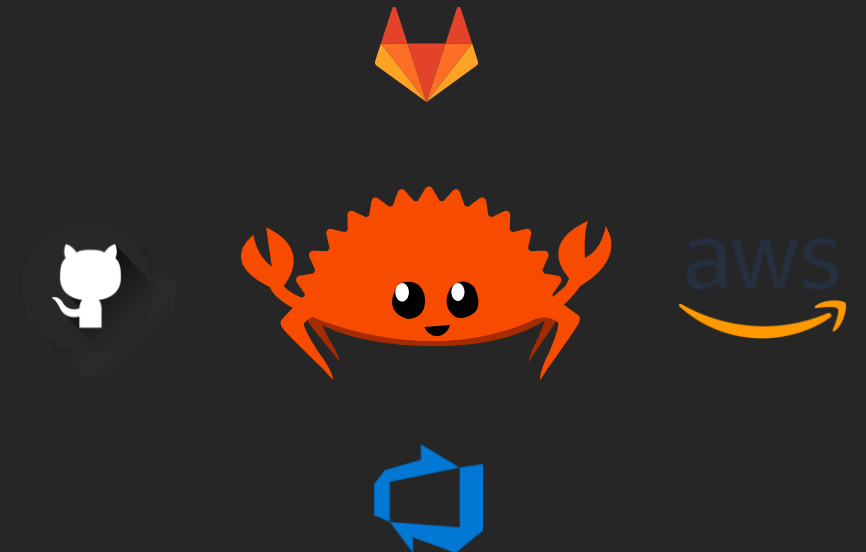
Hello, world! 🦀

Et après ?



On déploie comment ?

- Un seul binaire de généré pour toute votre application
- Lançable en une ligne de commande, conteneurisable en 10
- Tout les outils dont vous avez besoin pour vos CI/CD
 - Tests unitaires
 - Linting & formatting
 - Loggers & outils de monitoring
- Intégration direct avec les principaux outils de CI/CD



Cargo : le packager

- **Gestionnaire de paquet pour les projets Rust**
 - Utilise les paquets de crates.io
 - Paquets = crates
 - Actuellement +100k crates géré sur crates.io
- **Pas qu'un gestionnaire de paquet :**
 - Installe les outils additionnels (linters, etc.)
 - Lance les builds & compilations
 - Génère la documentation
 - Etc.



Quelques autres outils

- **RUSTFMT**
 - Le code-formatter
- **Clippy**
 - Le linter
- **CargoTest**
 - Pour les tests unitaires
- **CargoDoc**
 - Pour la génération de documentation
- **Tout ces outils sont intégrables automatiquement à VSCode**

Le playground

RUN ▶ ...

DEBUG ▾

STABLE ▾

...

SHARE

TOOLS ▾

⚙️ CONFIG ▾

?

```
1- fn main() {  
2-     println!("Hello world!");  
3- }  
4
```

Execution

Close

Standard Error

Compiling playground v0.0.1 (/playground)
Finished dev [unoptimized + debuginfo] target(s) in 0.59s
Running `target/debug/playground`

Standard Output

Hello world!


**Rust, on en fait
quoi ?**



Avec Rust, on peut « quasi » tout faire

- **On peut développer**
 - Des applications CLI/Serveur
 - Des ETL
 - Des applications haute-performances
 - Des applications web
 - Des micro-services
 - Du remplacement de code legacy C/C++
 - Des clients légers & hybrides
- **On ne peut pas – pour l’instant – développer**
 - Des clients lourds
 - Des applications mobiles

Un rapide tour : les CLI avec CLAP

 **crates.io**

[Browse All Crates](#) | [Log in with GitHub](#)

clap v4.0.23

A simple to use, efficient, and full-featured Command Line Argument Parser

[#cli](#) [#argument](#) [#arg](#) [#parse](#) [#parser](#)

[Readme](#) [314 Versions](#) [Dependencies](#) [Dependents](#)

clap

Command Line Argument Parser for Rust

crates.io

v4.0.23

downloads

85M

license

Apache 2.0

license

MIT

build

passing

coverage

89%

contributors

408

Dual-licensed under [Apache 2.0](#) or [MIT](#).


About


Create your command-line parser, with all of the bells and whistles, declaratively or procedurally.


For more details, see:

- [docs.rs](#)
- [examples](#)

Metadata

 1 day ago

 MIT OR Apache-2.0


 205 kB

Install


Add the following line to your Cargo.toml file:

```
clap = "4.0.23"
```


Documentation

 [docs.rs/clap/4.0.23](#)

Repository

 [github.com/clap-rs/clap](#)

Un rapide tour : Rocket, le framework web

 v0.5-rc


[overview](#) [guide](#) [api](#) [news](#) [code](#)

Meet Rocket.

Rocket is a web framework for Rust that makes it simple to write *fast*, secure web applications without sacrificing flexibility, usability, or type safety.

[Get Started](#) [Learn More](#) [Read FAQ](#)


Latest Release: 0.5.0-rc.2 (May 09, 2022)



Type Safe

From request to response Rocket ensures that your types mean something.


[Learn More](#)



Boilerplate Free

Spend your time writing code that really matters, and let Rocket generate the rest.


[See Examples](#)



Easy To Use

Simple, intuitive APIs make Rocket approachable, no matter your background.

[Get Started](#)



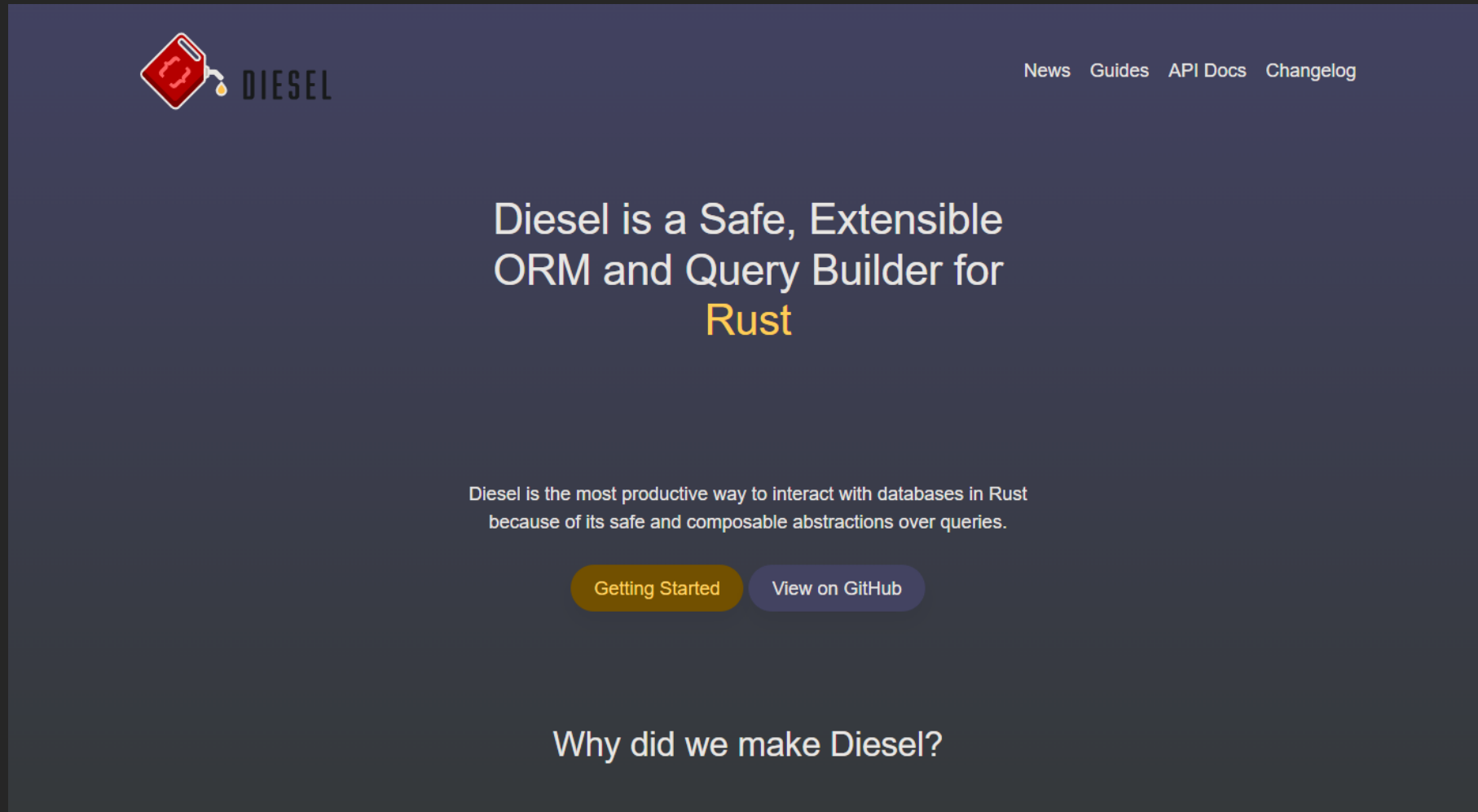
Extensible

Create your own first-class primitives that any Rocket application can use.

[See How](#)

Un rapide tour : ...avec un peu de SQL ?

L'ORM Diesel



Un rapide tour : Le frontend avec WASM



Rust

[Installer](#)

[Apprendre](#)

[Bac à sable](#)

[Outils](#)

[Gouvernance](#)

[Communauté](#)

[Blog](#)

Français (fr)



WebAssembly

Pourquoi Rust ?



Des performances prévisibles

Pas d'interruption imprévisible à cause d'un ramasse-miettes. Pas de chute de performance liée à un compilateur JIT. Juste un contrôle de bas-niveau et un haut niveau d'ergonomie.



Code succinct





Un exécutable plus léger permet des chargements de page plus rapides. Les `.wasm` générés par Rust n'incluent rien de superflu, comme un ramasse-miettes. Des optimisations avancées ainsi que du tree shaking suppriment le code mort.









Fonctionnalités modernes

Un écosystème vivant de bibliothèques pour vous aider à démarrer sur les chapeaux de roue. Des abstractions expressives à coût zéro. Et une communauté accueillante pour vous aider à apprendre.

Un rapide tour : ...ou remplacer React par Yew

 Yew 0.19.0  English  Docs  Tutorial

Community Blog Playground  API  GitHub  

Search  

 Yew

A framework for creating reliable and efficient web applications.

[Get Started](#) [Playground !\[\]\(279cd448dded2ade95bc3b6bc2ed243e_img.jpg\)](#)

Component Based

Features a component-based framework which makes it easy to create interactive UIs. Developers who have experience with frameworks like React and Elm should feel quite at home when using Yew.

[Learn more](#)

HTML macro

Features a macro for declaring interactive HTML with Rust expressions. Developers who have experience using JSX in React should feel quite at home when using Yew.

[Learn more](#)

Server Side Rendering

Features server side rendering for all the SEO and enhancements of server-rendered app while keeping the feel of an SPA

[Learn more](#)

Un rapide tour : créer une application client hybride

Tauri 1.2 and tauri-egui 0.1 have launched!

TAURI

Guides API References Blog Community About

Releases GitHub English Search

Announcing the release of

TAURI 1.2

Build an optimized, secure, and frontend-independent application for multi-platform deployment.

Bash PowerShell **Cargo** npm Yarn pnpm


```
$ cargo install create-tauri-app
$ cargo create-tauri-app
```

Quick Start

Brownfield

Compatibility with any front-end framework means you don't have to change your stack.


Learn More




Security

Front-of-mind for the Tauri Team driving our highest priorities and biggest innovations.


Learn More



Un rapide tour : ...ou écrire le code lourd de votre app mobile en Rust ?

 pub.dev

flutter_rust_bridge 1.49.1




Published 7...days ago •  cjycode.com Null safety

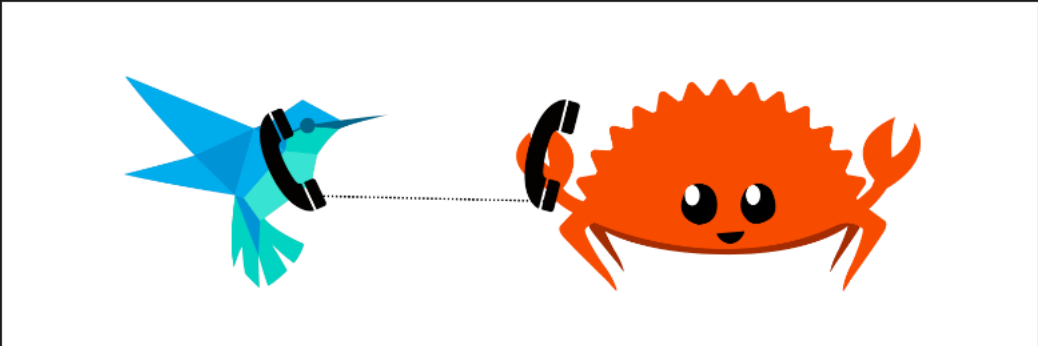
SDK DART FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

91

Readme Changelog Example Installing Versions Scores

flutter_rust_bridge: High-level memory-safe binding generator for Flutter/Dart <-> Rust

crates.io v1.49.1 pub v1.49.1 stars 1.9k  CI failing  Post-release failing  code quality A




91
LIKES

140
PUB POINTS

91%
POPULARITY

Publisher

 cjycode.com

Metadata


High-level memory-safe binding generator for Flutter/Dart <-> Rust

Repository (GitHub)

Documentation

API reference

License

 MIT (LICENSE)

Dependencies

Un rapide tour : et pourquoi pas développer des jeux-vidéo ?



BEVY Features

[Learn](#) [News](#) [Community](#) [Assets](#) [Examples](#)

[Donate](#) ❤️

[Get Started](#)



A refreshingly simple data-driven game engine built in Rust
Free and Open Source Forever!

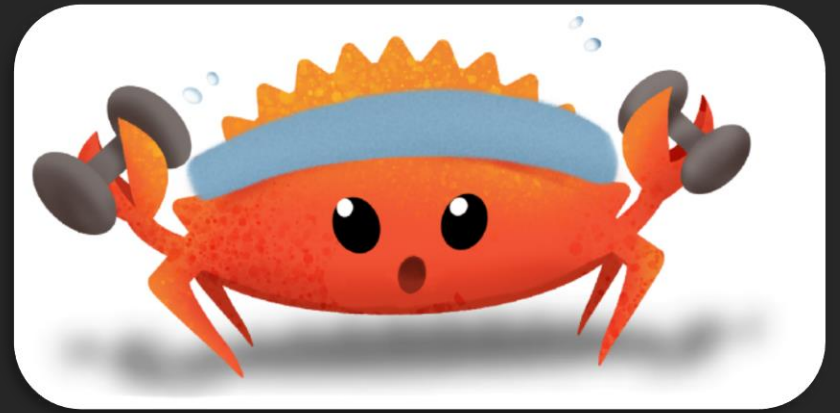
Data Driven

All engine and game logic uses Bevy ECS, a custom Entity Component System

- **Fast:** Massively Parallel and Cache-Friendly. The fastest ECS according to some benchmarks
- **Simple:** Components are Rust structs, Systems are Rust functions
- **Capable:** Queries, Global Resources, Local Resources, Change Detection, Lock-Free Parallel Scheduler

```
#[derive(Component)]  
struct Player;  
  
fn system(  
    q: Query<(Entity, &Player)>  
) {  
}
```

**Et la
performance
dans tout ça ?**

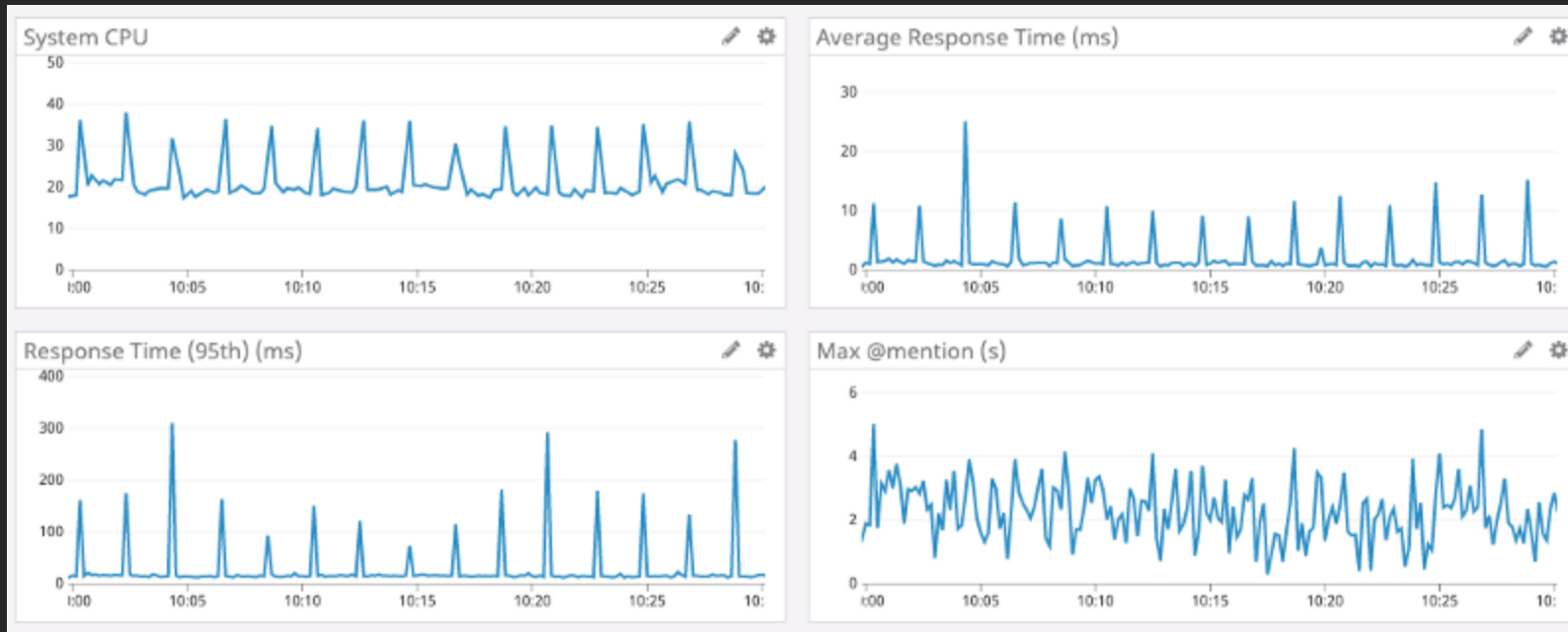


Discord : de Go à Rust



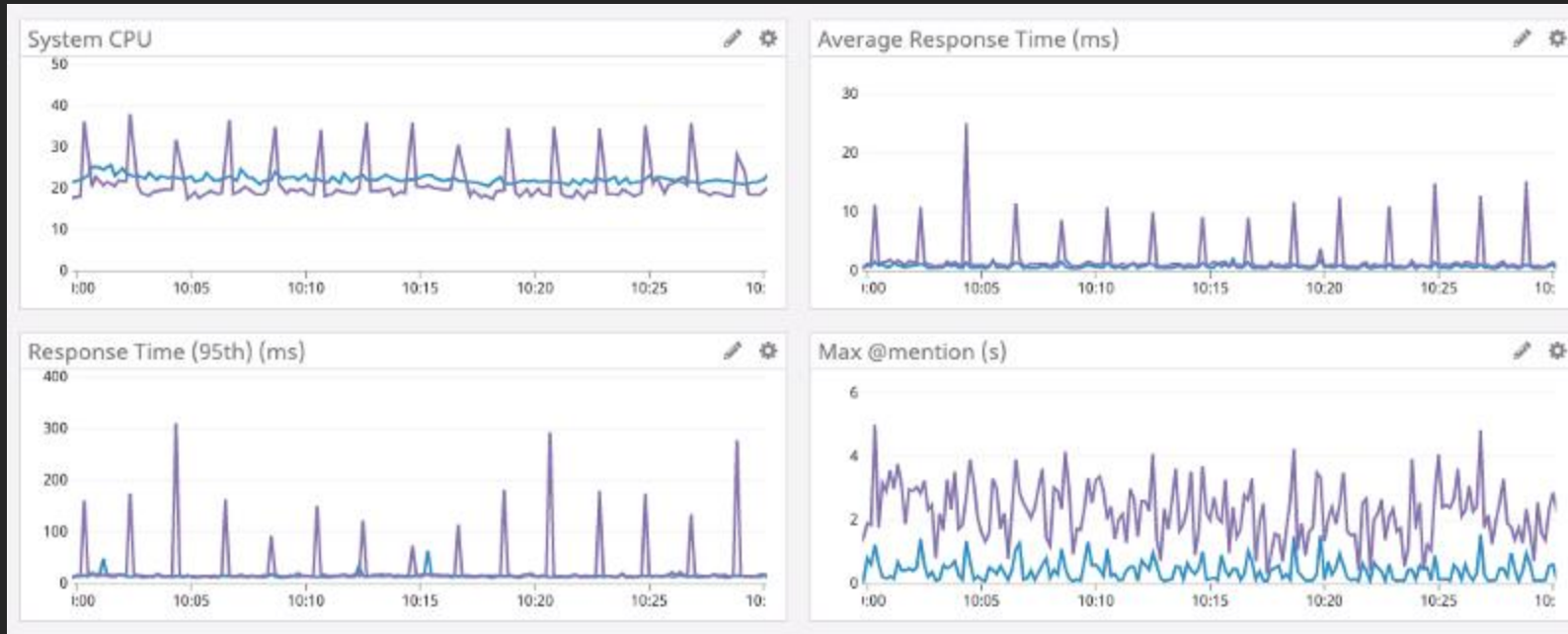
DISCORD

Discord : de Go à Rust



<https://discord.com/blog/why-discord-is-switching-from-go-to-rust>

Discord : de Go à Rust



En violet Go, en bleu Rust

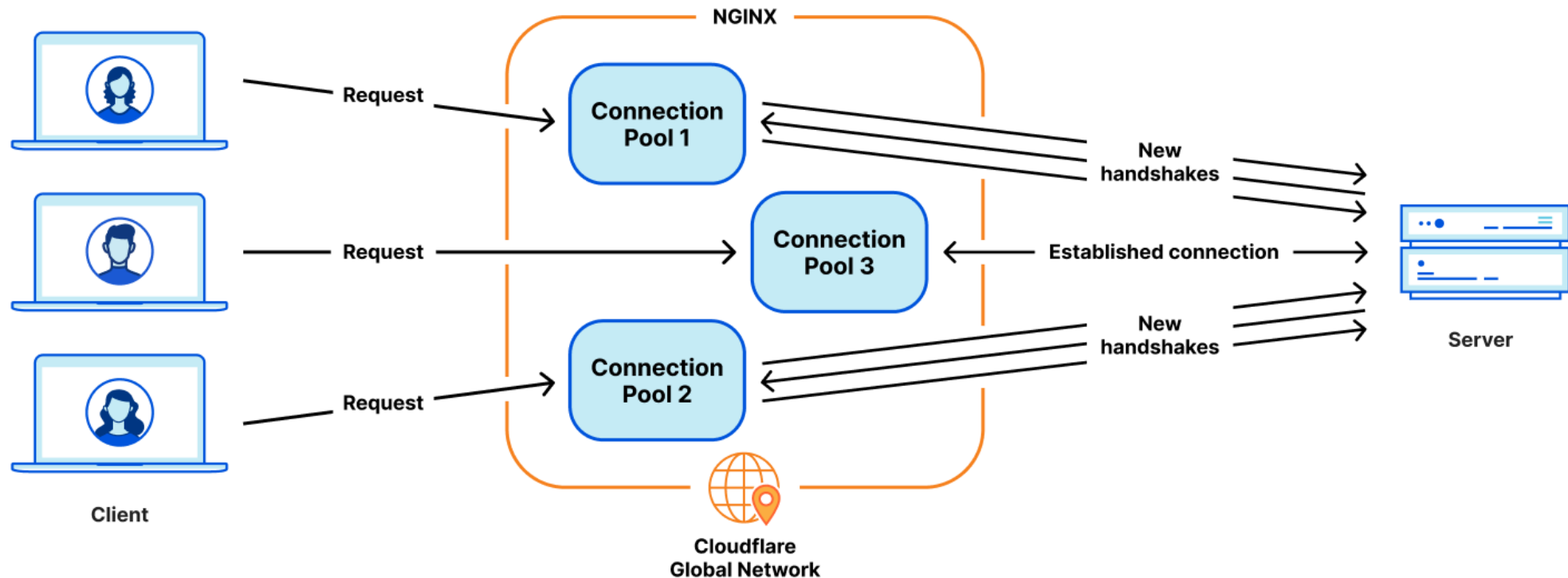
<https://discord.com/blog/why-discord-is-switching-from-go-to-rust>

Cloudflare : de Nginx à Pingora (Rust)

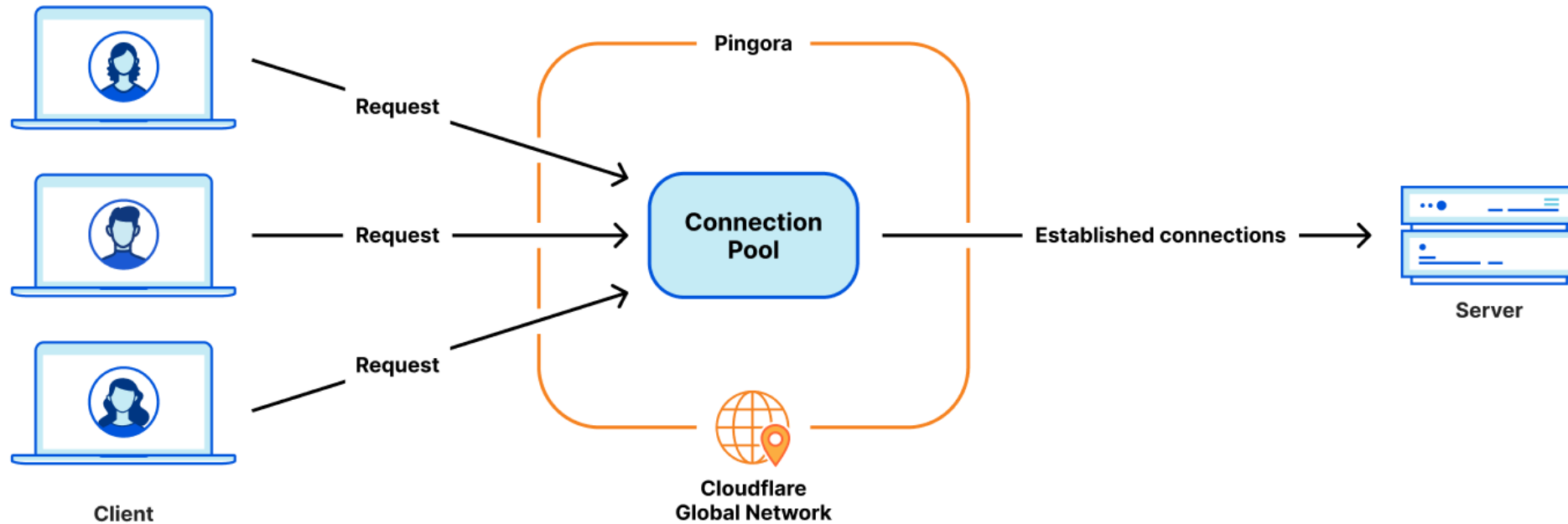


CLOUDFLARE®

Cloudflare : de Nginx à Pingora (Rust)



Cloudflare : de Nginx à Pingora (Rust)



<https://blog.cloudflare.com/how-we-built-pingora-the-proxy-that-connects-cloudflare-to-the-internet/>

Cloudflare : de Nginx à Pingora (Rust)

"In fact, Pingora crashes are so rare we usually find unrelated issues when we do encounter one.

Recently we discovered a kernel bug soon after our service started crashing."

- Cloudflare

<https://blog.cloudflare.com/how-we-built-pingora-the-proxy-that-connects-cloudflare-to-the-internet/>

Noyau linux : du Rust dans la ver6.1



Noyau linux : du Rust dans la ver6.1



C++ is a horrible language. It's made more horrible by the fact that a lot of substandard programmers use it, to the point where it's much much easier to generate total and utter crap with it.

- Linus Torvalds -

D'autres exemples concrets

- Le moteur de rendu de Figma est écrit en Rust et compilé via WASM
- Le moteur de rendu HTML Servo de Firefox est écrit en Rust
- Le compilateur de Deno, remplaçant de NodeJS est écrit en Rust
- Magic Pocket, le système de stockage distribué de Dropbox est écrit en Rust
- ...

En résumé



Bon alors, Rust ??

Les +

- Mêmes performances que le C
- Un code sûr en production
- On peut en faire -presque- ce que l'on veut
- Une communauté grandissante de jours en jours
- Le langage est en cours d'adoption par de grandes entreprises

Les -

- Le recrutement
- La courbe d'apprentissage
- Le temps de compilation

Conclusion



- Le repo Git

- Le contenu de la présentation
- Comment bien démarrer Rust
- Des ressources complémentaires
 - Extensions VSCode
 - Playlists et vidéos d'explications
 - Conteneurs docker prêt à l'emploi
- ...Et bien plus encore



<https://github.com/AdrienGras/presentation-rust-2022>