Bienvenidos a Rust Argentina

Agenda 📆:

“Bienvenida & Comunidad” ~ Hernán Gonzalez

“Cargo Lambda - Creando un Authorizer” ~ Nicolás Antinori

“Status de Rust Enterprise” ~ Alejandro Leiton

https://rust-lang.ar - 03/04/2025

Gracias a nuestros hosts 👏

https://rust-lang.ar - 03/04/2025

2024 Edition

Tenemos nuevo mayor release!

- **Async** closures
- **`gen`** keyword is reserved.
- Lifetime capture rules for **impl Trait** in returns.
- **FromIterator** and **Extend** for tuples.
- Cambios en **std**, **rustfmt** y más en https://blog.rust-lang.org.

- RPIT lifetime capture rules — Changes the default capturing of parameters by `impl Trait` types when `use<..>` is not present.
- `if let` temporary scope — Changes the scope of temporaries for `if let` expressions.
- Tail expression temporary scope — Changes the scope of temporaries for the tail expression in a block.
- Match ergonomics reservations — Disallow some pattern combinations to avoid confusion and allow for future improvements.
- Unsafe `extern` blocks — `extern` blocks now require the `unsafe` keyword.
- Unsafe attributes — The `export_name`, `link_section`, and `no_mangle` attributes must now be marked as `unsafe`.
- `unsafe_op_in_unsafe_fn` warning — The `unsafe_op_in_unsafe_fn` lint now warns by default, requiring explicit `unsafe {}` blocks in `unsafe` functions.
- Disallow references to `static mut` — References to `static mut` items now generate a deny-by-default error.
- Never type fallback change — Changes to how the never type `!` coerces, and changes the `never_type_fallback_flowing_into_unsafe` lint level to "deny".
- Macro fragment specifiers — The `expr` macro fragment specifier in `macro_rules!` macros now also matches `const` and `_` expressions.
- Missing macro fragment specifiers — The `missing_fragment_specifier` lint is now a hard error, rejecting macro meta variables without a fragment specifier kind.
- `gen` keyword — Reserves the `gen` keyword in anticipation of adding generator blocks in the future.
- Reserved syntax — Reserves `#"foo"#` style strings and `##` tokens in anticipation of changing how guarded string literals may be parsed in the future.

Rust 1.86.0

... recién anunciado

- **Trait** upcasting
- **HashMaps** and slices now support indexing multiple elements mutably
- **#[target_feature]** in safe functions.
- **i586-pc-windows-msvc** se nos va 👋🏽

```
trait Trait: Supertrait {}
trait Supertrait {}

fn upcast(x: &dyn Trait) -> &dyn Supertrait {
    x
}
```

Cargo Lambda

con Nicolás Antinori

Ingeniero de Software interesado en la programación de bajo nivel y la criptografía.

Cargo Lambda

Lambdas en Rust de forma fácil

- Es una herramienta CLI que nos facilita la creación, desarrollo y deployment de lambdas.
- Fácilmente instalaste en los sistemas operativos mayores (Windows, MacOS y Linux).
- Recomendada por SDK de Rust oficial de AWS.

Explorando Cargo Lambda

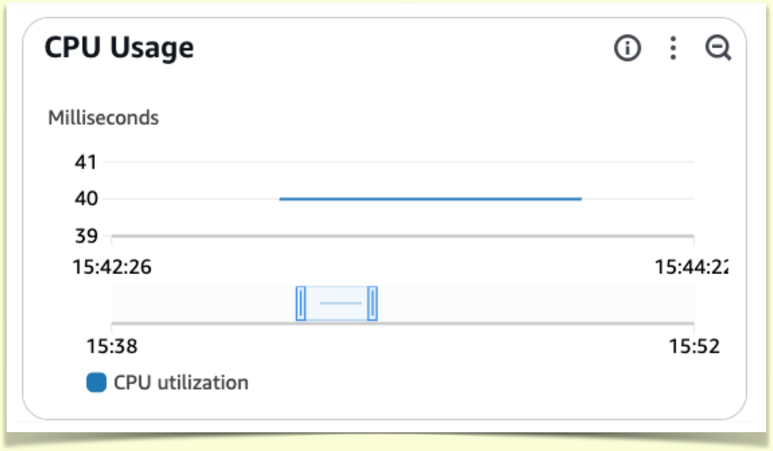
Principales comandos

- `cargo lambda new`: crea una nueva lambda.
- `cargo lambda build`: construye el proyecto.
- `cargo lambda watch`:
  invoca el servidor local de desarrollo.
- `cargo lambda invoke <lambda>`:
  invoca una lambda.
- `cargo lambda deploy <lambda>`:
  deploya una lambda en AWS.

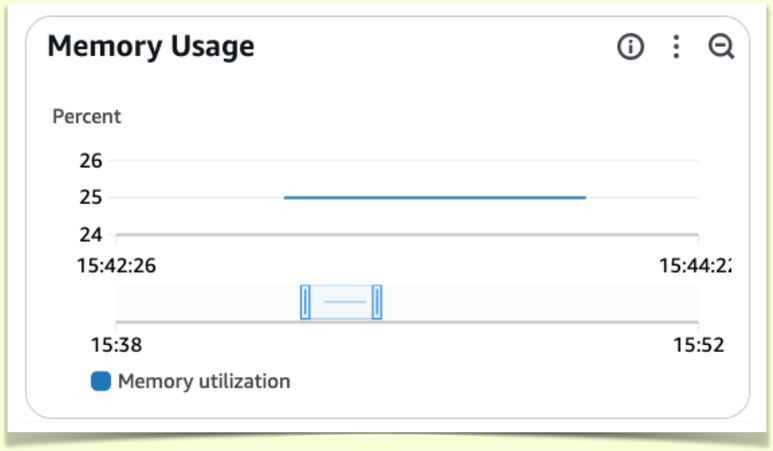
Creando un Lambda Authorizer en Rust con Cargo Lambda

Lambda Authorizer

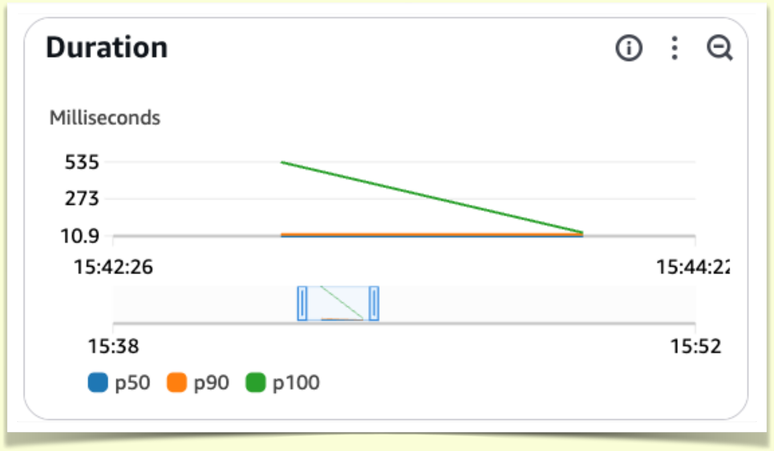
Performance



40 ms de uso de CPU



25 MB de uso
de memoria **constante**



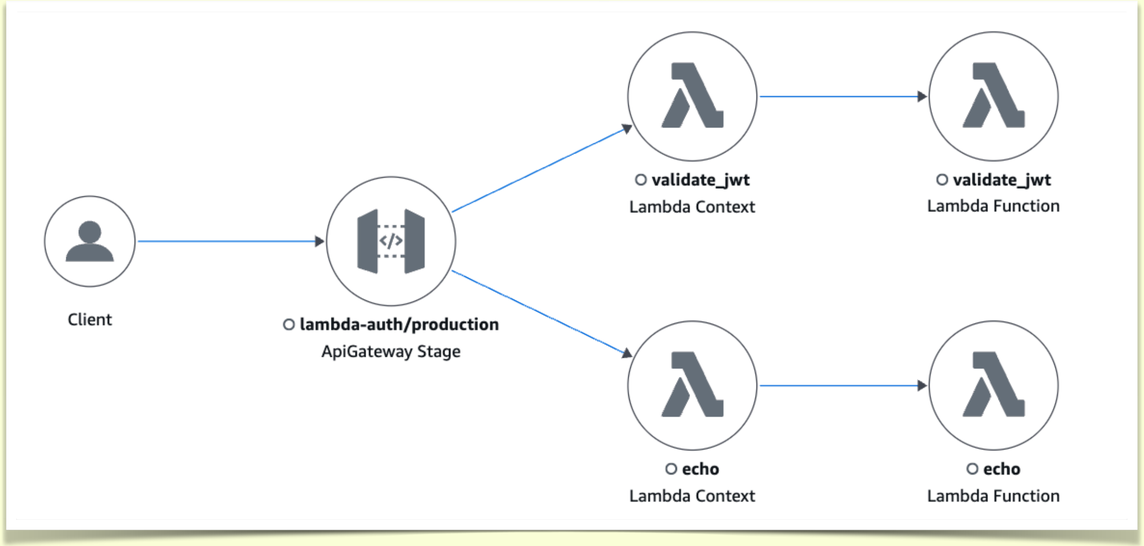
~500ms en **cold** start
~21ms en **warm** start

Lambda Authorizer

Tracing

Segments Timeline Info

Name	Segment status	Response code	Duration	Hosted in	0.0ms 20ms 40ms 60ms
lambda-auth/production AWS::ApiGateway::Stage					
lambda-auth/production	OK	200	56ms		POST https://lwmyw...
Lambda	OK	200	18ms		Invoke: echo
validate_jwt AWS::Lambda					
validate_jwt	OK	200	27ms		
validate_jwt AWS::Lambda::Function					
validate_jwt	OK	-	19ms		
Invocation	OK	-	18ms		
Overhead	OK	-	0ms		
echo AWS::Lambda					
echo	OK	200	11ms		
echo AWS::Lambda::Function					
echo	OK	-	2ms		
Invocation	OK	-	1ms		
Overhead	OK	-	1ms		



Trazabilidad de punta a punta utilizando el estándar de **OpenTelemetry**

Creando un Lambda Authorizer en Rust con Cargo Lambda

Podes encontrar el código en:

https://github.com/rustlatam/oxidar-lambdas

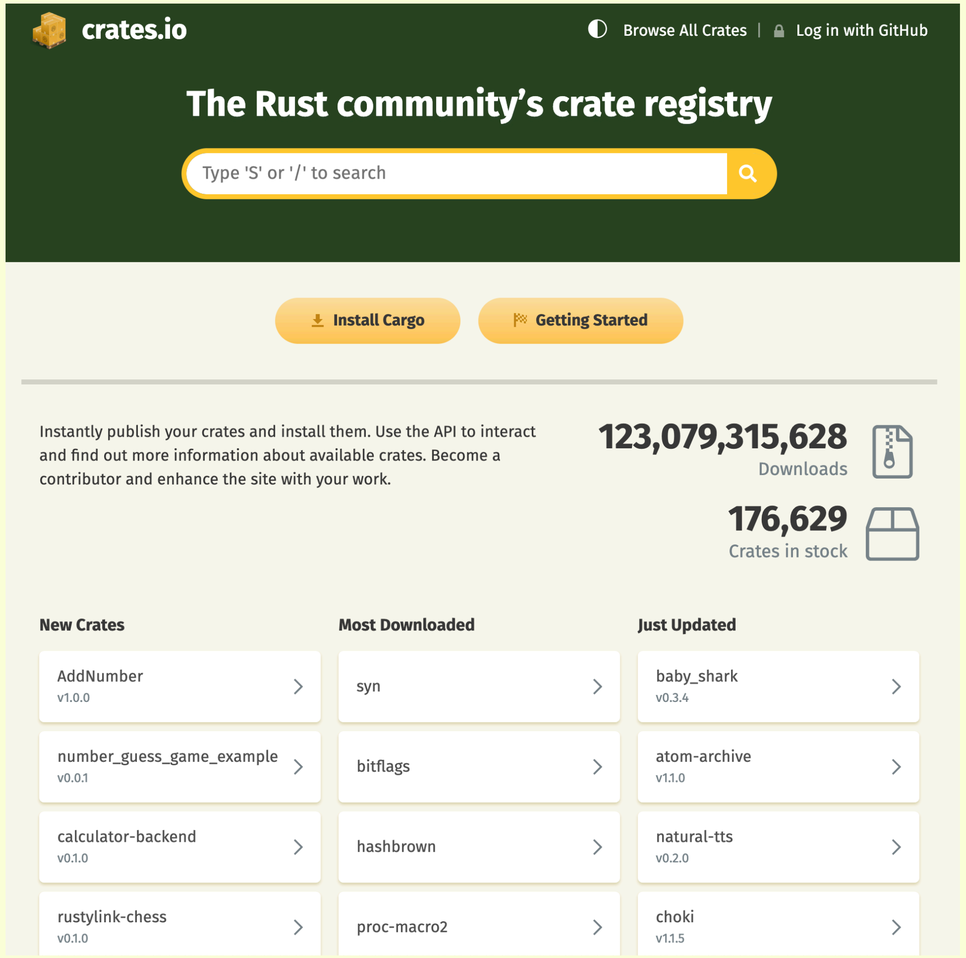
Alejandro Leiton

Correctness, Adoption & Community

- Desarrollador de software, trabajo actualmente en VAIRIX para clientes de USA.
- Entusiasta de Rust
- https://www.linkedin.com/in/alejandro-leiton/

Tooling

Crates & The Dependency Ecosystem



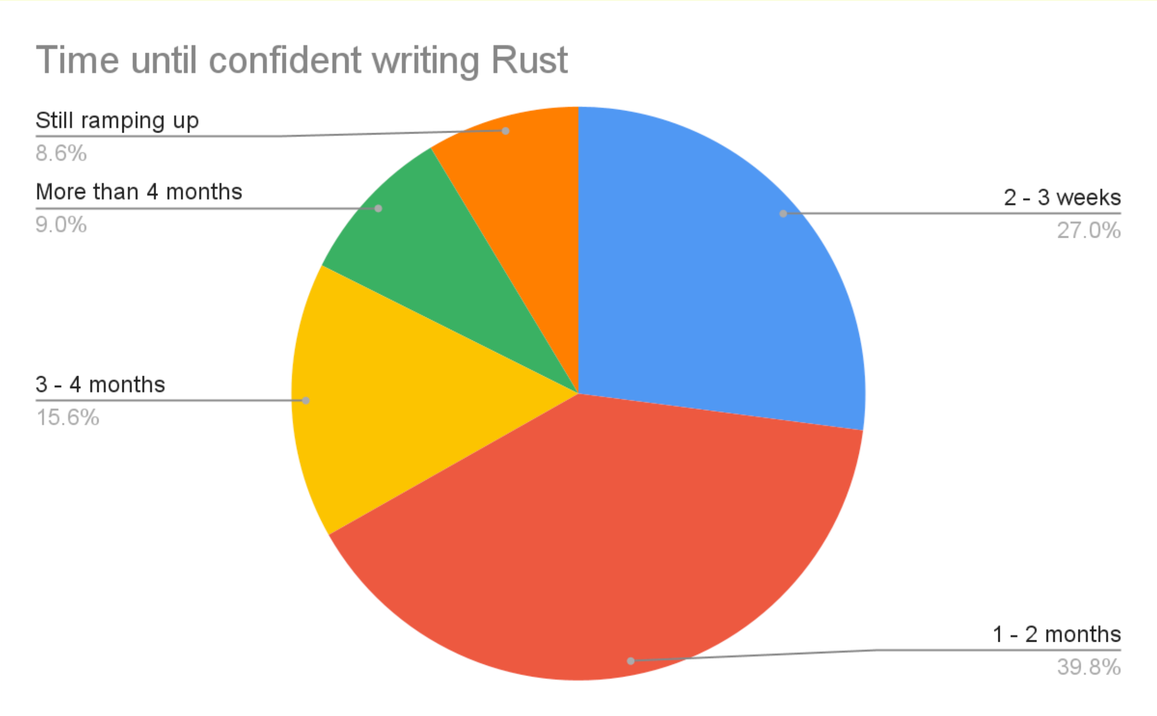
- Registro de paquetes oficial de Rust (similar a npm para JavaScript o pip para Python, etc.
- Respaldado por Rust Foundation oficialmente
  - Provee infraestructura
  - Monitoreos de seguridad
  - Coordina con el ecosistema de Rust
- Algunos crates empresariales
  - **Tokio** - Dev async por Rust Foundation
  - **Axum** - web framework por Rust Foundation
  - **MongoDB Rust Driver** - MongoDB para Rust
  - **AWS** - SDK de Amazon para sus servicios
  - **Pingora** - API gateway de CloudFront

aws Google HUAWEI Meta Microsoft

Adoption

Rust Adoption by **Google**

- Google: Rust es un lenguaje de primera clase para Android y Chromium.
- Google throws $1M at Rust Foundation to build C++ bridges
- **Lars Bergston**: Director of Engineering at Google, working on Android Platform Tools & Libraries
- Lars Bergstrom - Beyond Safety and Speed: How Rust Fuels Team Productivity (quien es (jefe de ingeniería compilador Android)
- Hicieron A/B test en Google:
  - Productividad = Go y 2x C++
  - Menos uso de Memoria
  - Menos defect rate (mejor correctness)
  - Toma lo mismo entrenar un dev en Rust que en Java (devs seniors)
  - En menos de 4 meses un dev se tiene productivo igual que en su antiguo lenguaje **If it compiles it works**
  - 85% de los devs se sienten mas confiados que su código en Rust es correcto comparado con otros lenguajes



85% confidence in 4 months

Adoption

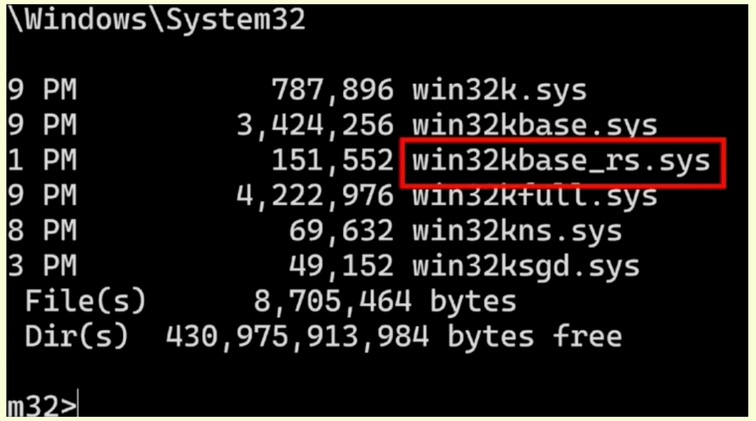
Rust Adoption by **Microsoft**

- Microsoft is Getting Rusty: A Review of Successes and Challenges - Mark Russinovich
- 70% de las vulnerabilidades que tuvieron en los últimos 10 años son de memory safety
- $1M Microsoft Donation to Fund Key Rust Foundation & Project Priorities
- Microsoft viene invirtiendo $10 millones en Rust tooling

Adoption

Rust Adoption by **Microsoft**

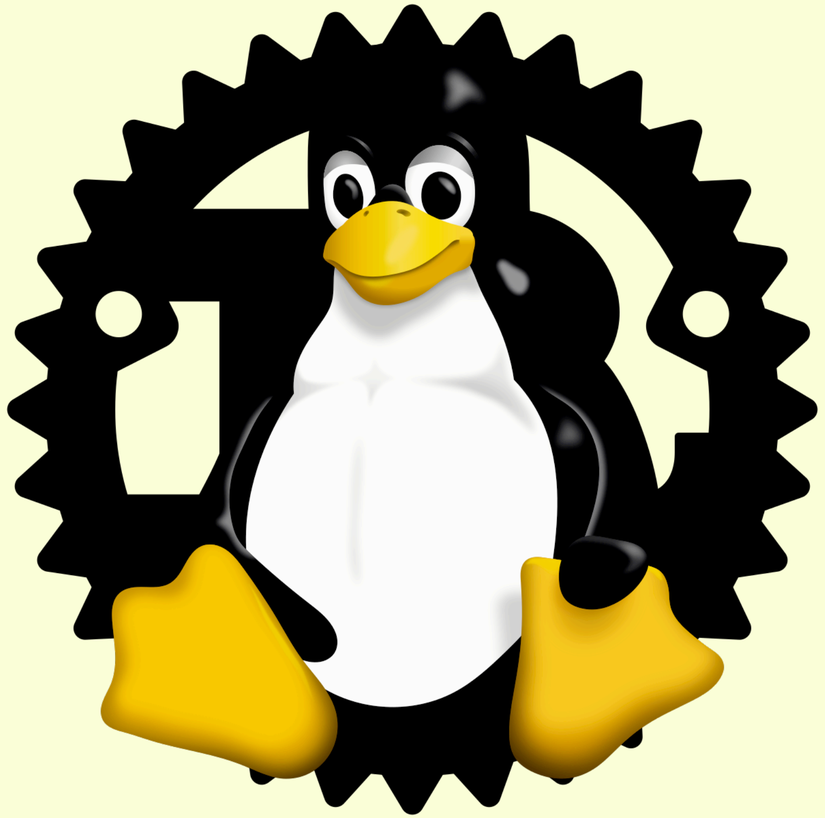
- Rust en Windows:
  - Soporte a UEFI dev con Project Mu
  - Portaron DirectWrite Core con 2 devs en 6 meses (154KLOC) con 5% ~ 15% mejora en performance
  - Win32K GDI Region con 2 devs en 3 meses (6KLOC sin los tests)
  - rustls-symcript
  - Azure utiliza Rust hace varios años, algunos ejemplos:
    - Caliptra - Harware Root of Trust
    - Azure Integrated HSM (new in-house security chip)
    - Azure Boost agents
    - Hyper-V (virtualization stack)
    - Open VMM (Virtual Machine Monitor)
    - HiperLight (Hyperlight: Achieving 0.0009-second micro-VM execution time)
    - Azure Data Explorer (maneja PB y millones de queries por día)
    - GraphRAG Migrar proyectos enteros



Adoption

Rust Adoption by **Ubuntu/Linux**

- Rust avanza en Linux desde el 2020 y en algunos distros:
  - Rust in Linux: Where we are and where we're going next
  - Ubuntu 25.10 plans to swap GNU coreutils for Rust



Adoption

Rust Adoption by other **Major Companies**

- **AWS (Amazon)**: Muchos servicios como Firecracker (VMs) y Bottlerocket (OS) están en Rust.
- **Meta (Facebook)**: Usa Rust para servicios de back-end services y criptografía.
- **Discord & Cloudflare**: Usan Rust para sistemas de baja latencia y alta performance.
  - Why Discord is switching from Go to Rust
- **Figma:** Se migraron partes de su backend para lograr un rendimiento de colaboración en tiempo real y un procesamiento de gráficos vectoriales complejos.
  - Rust in production at Figma (2018)

Community

Rust Community & Support

- Conferencias:
  - RustWeek
  - RustConf
  - EuroRust
  - RustNation UK
  - meetups locales
- Rust Lang Community
- r/Rust
- Rust forum
- Zulip real-time support.
- Guía para usar el Zulip de Rust

Hernán G. Gonzalez

http://hernan.rs

- 🗓️ 20 Engineering
- 🤖 Swift, Rust, C/C++/x86, Java, Python, JS,
- 🤝 Delivery, Architecture, Strategy & Mobile SME.
- 📈 Finance, Travel, Hospitality, Dating, Gaming, Media, Social Media & Public Sector.

MUCHAS GRACIAS!

rust-lang.ar

Que te pareció el encuentro?
Déjanos tus comentarios :)