# The Ascendancy of Rust: A Definitive Analysis of its Adoption Across Global Industries

The global adoption of the Rust programming language is undergoing a strategic and rapid expansion, driven by its unique capacity to deliver both enterprise-grade security and disruptive performance. This dual appeal has established Rust as a foundational technology for modernizing legacy systems in public corporations and as a critical competitive advantage for innovative private companies, cementing its role as a definitive language for the next generation of critical software. An exhaustive analysis of companies across the globe reveals a clear pattern: Rust is being integrated into the core systems of a diverse array of industries, chosen for its unparalleled guarantees of memory safety, which eliminate entire classes of common security vulnerabilities, combined with its high performance, reliability, and advanced concurrency features. This dual adoption pattern—public giants fortifying their infrastructure and private innovators disrupting markets—is creating a powerful, self-reinforcing cycle of growth and maturation for the entire Rust ecosystem.

## The Dual Engines of Growth: Enterprise Fortification and Startup Disruption

The accelerating adoption of Rust is propelled by two distinct, yet complementary, strategic imperatives. For large, publicly traded corporations, Rust represents a powerful tool for fortification—a way to systematically de-risk critical infrastructure, modernize legacy C/C++ systems, and eliminate historical sources of security vulnerabilities at a massive scale. In contrast, for privately funded companies and agile startups, Rust is a tool of disruption—a competitive differentiator used to build products with superior performance, achieve greater resource efficiency, and attract elite engineering talent with a modern, high-performance language. This bifurcation in adoption drivers highlights the language's versatility and is fundamental to understanding its strategic importance in the current technology landscape.1

### Public Corporations: The Pursuit of Stability, Security, and Scale

Tech giants are strategically deploying Rust not merely for new projects, but as a primary weapon in the ongoing battle to secure and stabilize their most critical, large-scale infrastructure. This is not an experimental endeavor but a calculated response to the persistent and costly problem of memory safety vulnerabilities in systems-level code.

**Microsoft's War on Memory Bugs:** Microsoft's initiative to rewrite core components of the Windows kernel and Azure services in Rust is a direct and strategic response to a massive historical vulnerability vector. This decision is grounded in hard data from the Microsoft Security Response Center (MSRC), which found that approximately 70% of its assigned Common Vulnerabilities and Exposures (CVEs) were related to memory safety issues—a class of bugs that Rust's compiler is designed to prevent at compile time.1 By adopting Rust, Microsoft aims to eliminate entire categories of memory-related flaws that have plagued system software for decades, representing a fundamental shift toward building more secure and reliable products from the ground up.1 This move provides a powerful economic and security justification for a strategic technological shift. The discovery of the first Rust-based kernel flaw in August 2025 provides a nuanced perspective; while Rust eliminates entire bug classes like buffer overflows, implementation and logic flaws remain a challenge, reinforcing the need for robust security practices even with safer languages.4

**Google's Proactive Hardening of Android:** In a similar vein, Google is deeply integrating Rust into the Android operating system as a core part of its security strategy. In the Android 13 release alone, 1.5 million lines of new native code for critical components like Keystore2, the Ultra-wideband (UWB) stack, and the Android Virtualization Framework (AVF) were written in Rust.1 The motivation is singular: to reduce the platform's attack surface by eliminating memory safety vulnerabilities, which have historically been the leading cause of security issues in Android. The results have been stark and offer definitive proof of concept for Rust's value proposition. As of the Android 13 release, there have been zero memory safety vulnerabilities discovered in the new Rust code.1

**Amazon Web Services (AWS) and Secure Multi-tenancy:** AWS's use of Rust in foundational services like Firecracker and Bottlerocket is central to its serverless and container offerings, including AWS Lambda and AWS Fargate. Firecracker, a lightweight virtualization technology, can launch microVMs in as little as 125 milliseconds while consuming only about 5 MiB of memory, enabling thousands to run on a single host and handling trillions of requests monthly for AWS Lambda.1 Rust's compile-time guarantees of memory and thread safety provide the strong isolation and minimal attack surface necessary for a secure multi-tenant cloud environment—a non-negotiable requirement for a public cloud provider of AWS's scale.1

### Private Companies: The Quest for Performance and Competitive Edge

While public corporations use Rust for defense, privately funded companies and startups are using it for offense. For these organizations, Rust's extreme performance is not just an operational efficiency but a core product feature and a key to disrupting established markets.

**Discord's Elimination of Latency Spikes:** The well-documented migration of Discord's 'Read States' service from Go to Rust was driven by the need for predictable, low-latency performance, a critical feature for a real-time communication platform. The Go garbage collector was causing intermittent latency spikes that degraded the user experience. By switching to Rust, which has no garbage collector, Discord completely eliminated these spikes, achieving average response times in microseconds and gaining a direct competitive advantage in performance and reliability.1

**Figma's Real-Time Collaboration Engine:** The collaborative design platform Figma rewrote its high-performance, real-time multiplayer syncing engine in Rust. This core component, which powers the collaborative editing of all Figma documents, was originally written in TypeScript. However, as the user base scaled, the team encountered performance bottlenecks and issues related to the JavaScript garbage collector. The rewrite to Rust delivered superior performance and low resource usage, enabling the seamless, real-time experience that defines their product and supports a massive user base.1

**Vercel's Developer Tooling Revolution:** Vercel, the company behind the popular Next.js framework, is using Rust to build a new generation of high-performance developer tooling. Projects like Turbopack (an incremental bundler) and Turborepo (a build system migrated from Go) demonstrate a strategy of using Rust's performance to create a superior developer experience. Turbopack, positioned as the successor to Webpack, shows updates 700x faster, dramatically improving developer productivity and build times.1

The adoption patterns in public and private sectors are not isolated; they create a mutually beneficial feedback loop. The investment by public companies like AWS, Google, and Microsoft, who are founding members of the Rust Foundation, provides corporate backing, funds core development, and de-risks the language for broader enterprise use.1 This creates a stable foundation upon which private companies can innovate. These startups and private firms, in turn, push the boundaries of the ecosystem, creating new, high-performance libraries in domains like AI and databases. These libraries are often open-sourced, enriching the ecosystem for all users, including the large enterprises that provided the initial stability. This symbiotic relationship accelerates the maturity of the entire ecosystem far faster than if either sector were acting alone. Furthermore, for startups, choosing a modern, highly-regarded language like Rust serves as a strategic tool for attracting elite engineers who are motivated by challenging technical problems and cutting-edge technologies, providing a crucial advantage in the competitive market for talent.

**Table 1: Comparative Analysis of Rust Adoption Drivers in Public vs. Private Companies**

| Company Name | Public/Private Status | Primary Driver | Key Projects | Strategic Outcome |
| --- | --- | --- | --- | --- |
| **Microsoft** | Public Company | Security Fortification | Rewriting Windows kernel components; Azure SDK | Strategic reduction in memory-safety CVEs across core products. |
| **Google** | Public Company | Security Fortification | Android OS components (Keystore2, UWB stack) | Zero memory-safety vulnerabilities in new Android Rust code. |
| **Amazon Web Services (AWS)** | Public Company | Security & Performance | Firecracker (VMM for Lambda), Bottlerocket (Container OS) | Secure, high-performance isolation for multi-tenant cloud services. |
| **Cloudflare** | Public Company | Performance & Robustness | Pingora (HTTP proxy), quiche (HTTP/3 implementation) | Higher throughput with significantly lower CPU/memory footprint. |
| **Discord** | Private Company | Performance Differentiation | 'Read States' service migration from Go to Rust | Elimination of GC-related latency spikes for real-time communication. |
| **Figma** | Private Company | Performance Differentiation | Multiplayer syncing engine rewrite from TypeScript | Best-in-class speed and low resource usage for seamless collaboration. |
| **1Password** | Private Company | Security & Consistency | Unified 'Core' logic for all clients (desktop, mobile, web) | Cross-platform consistency and a single, verifiable secure codebase. |
| **Vercel** | Private Company | Performance Differentiation | Turbopack (bundler), Turborepo (build system) | Order-of-magnitude faster build times and developer experience. |

Source: Synthesized from Influential Case Studies and company data.1

## Re-Plumbing the Internet: Rust as the New Foundation for Cloud and Network Infrastructure

Rust's most mature and impactful domain of adoption is in the foundational software that powers the internet and cloud services. An analysis of leading technology, security, and observability companies reveals a clear trend: Rust is becoming the default choice for new, high-throughput, and security-critical infrastructure. Its combination of C++-level performance with compile-time memory safety guarantees allows engineers to build systems that are both exceptionally fast and fundamentally more secure, addressing the core challenges of internet-scale services.1

### Replacing the Old Guard with a Safer, Faster Alternative

A significant indicator of Rust's maturity is its use in replacing decades-old, battle-tested infrastructure written in C and C++. These are not trivial decisions; they represent a strategic belief that Rust's benefits outweigh the substantial cost of a rewrite.

The most prominent example is **Cloudflare's Pingora**, an in-house HTTP proxy framework built entirely in Rust to replace their long-standing NGINX infrastructure.1 The results of this migration are profound: Pingora handles over one trillion requests per day while requiring only a third of the CPU and memory resources of the previous NGINX-based system. This is not merely a technical achievement but a significant economic one, translating directly into millions of dollars in operational savings at Cloudflare's scale. The decision to replace a technology as deeply entrenched as NGINX underscores a powerful business case for Rust, where performance gains lead to tangible financial benefits. The recent discovery and rapid mitigation of a request smuggling vulnerability (CVE-2025-4366) in the open-source version of Pingora also serves as a case study in the maturity of the Rust security ecosystem, demonstrating a robust process of community-driven discovery, responsible disclosure, and swift resolution.30

### Building for Extreme Throughput and Observability

The performance and efficiency of Rust are particularly critical for data-intensive observability and security platforms that must process massive volumes of data in real time with minimal overhead.

**Datadog**, a leader in the observability space, maintains and uses **Vector**, a high-performance, open-source data pipeline built entirely in Rust.1 Vector is benchmarked as being up to 10 times faster than alternative solutions, with its largest user processing over 500TB of data daily. This demonstrates Rust's suitability for high-throughput data engineering at extreme scale. Datadog has further leveraged Rust to rebuild its AWS Lambda Extension, reducing cold start times by 82% and memory consumption by 40%. In a recent project, Datadog also migrated its static analyzer from Java to Rust, achieving a 3x performance increase and a 10x reduction in memory usage, enabling them to embed the analyzer directly into IDEs for a tighter developer feedback loop.38

Similarly, **Sentry's** critical edge ingestion service, **Relay**, is built in Rust to handle hundreds of thousands of events per second globally.1 Relay acts as a proxy that pre-processes all error and performance data from Sentry SDKs. Its architecture, which makes extensive use of the actor model for concurrency, showcases how Rust's language features can be used to build robust, highly scalable data ingestion pipelines that are resilient by design.39

### The "Single Secure Core" Architectural Paradigm

Rust, particularly when combined with its mature WebAssembly compilation target, is enabling a new architectural pattern for building secure, cross-platform applications. This "single secure core" model involves writing the most sensitive logic—such as cryptography, business rules, and data handling—once in Rust and then deploying that same compiled library natively on all target platforms, including desktop, mobile, and the web.

**1Password** is the quintessential example of this approach. The company rebuilt its entire application logic around a single 'Core' component written in Rust, which now powers every 1Password client.1 This ensures that critical operations like autofilling logic and cryptography are identical and equally secure everywhere, from a Windows desktop application to an iOS app to a browser extension running via WebAssembly. This architecture drastically reduces development overhead and, more importantly, eliminates entire classes of bugs and security vulnerabilities that arise from inconsistencies between platform-specific implementations. It represents a fundamental shift in how secure, multi-platform applications can be built, offering a powerful and repeatable model for other companies in the security space and beyond.45

**Table 2: Rust Adoption in Core Technology, Cloud, and Hosting Infrastructure**

| Company | Public/Private Status | HQ Country | Rust Usage Details | Notable Outcomes |
| --- | --- | --- | --- | --- |
| **Amazon Web Services (AWS)** | Public (Subsidiary) | USA | Critical infrastructure services (Firecracker, Bottlerocket), components of S3, EC2, CloudFront, Nitro System. | Firecracker enables microVMs to launch in 125ms with 5 MiB overhead; enhances security and resource efficiency. |
| **Microsoft** | Public | USA | Rewriting core components of Windows kernel; official Azure SDK for Rust for cloud applications. | Strategic reduction in memory safety vulnerabilities (~70% of historical CVEs). |
| **Google** | Public | USA | Heavy integration into Android OS (UWB stack, Keystore2); Chromium browser libraries. | Zero memory safety vulnerabilities discovered in Android's new Rust code as of Android 13. |
| **Cloudflare** | Public | USA | Core proxy framework (Pingora), QUIC/HTTP/3 implementation (quiche), Cloudflare Workers. | Pingora handles >1 trillion requests/day with 1/3 the CPU/memory of NGINX; 37% faster web loading with HTTP/3. |
| **Dropbox** | Public | USA | Core desktop sync engine (Nucleus), parts of Magic Pocket storage system, Dropbox Capture library. | Improved screen recording quality (to 4K), faster sharing, and reduced application footprint. |
| **Discord** | Private | USA | 'Read States' service (migrated from Go), Elixir NIFs, client-side video encoding. | Solved GC latency spikes; scaled backend to over 11 million concurrent users. |
| **Vercel** | Private | USA | High-performance developer tooling: Turbopack (bundler), Turborepo (build system), SWC (compiler). | Turbopack is 700x faster than Webpack; SWC is 17x faster than Babel, improving developer productivity. |
| **Shopify** | Public | Canada | YJIT (Just-In-Time compiler for Ruby), high-performance network servers, Wasm extensions. | YJIT delivered a 14.1% average latency improvement, handling 80M requests/min during peak events. |
| **Figma** | Private | USA | Real-time multiplayer syncing engine (rewritten from TypeScript). | Significant performance improvements and low resource usage, enabling seamless real-time collaboration. |
| **Canonical** | Private | UK | Integration into core Ubuntu OS components; support for Rust-based Linux kernel modules. | Promotes the creation of safer and more reliable system-level software on a premier Linux platform. |
| **Mozilla** | Private (Non-profit) | USA | Foundational use in the Firefox browser, integrating components from the Servo engine. | Significantly improved Firefox's performance, memory safety, and multi-core processor utilization. |
| **Qumulo** | Private | USA | Core distributed file system transitioned from C to Rust for product development. | Enhances the reliability and security of its cloud-native file system for data-intensive industries. |

Source: Synthesized from Tech, Cloud, and Hosting Companies data.1

**Table 3: Rust in Security, Networking, and Observability Platforms**

| Company | Public/Private Status | HQ Country | Product Modules Using Rust |
| --- | --- | --- | --- |
| **1Password** | Private | Canada | The entire 'Core' backend and application logic for all clients, CLI, and cryptographic modules. |
| **Datadog** | Public | USA | Real-time timeseries storage engine, AWS Lambda Extension, and the Vector observability data pipeline. |
| **Kentik** | Private | USA | High-performance host/sensor agents (ksynth), zero-copy protocol decoding, and backend data storage formats. |
| **Sentry** | Private | USA | 'Relay' service for event ingestion and forwarding, handling schema validation, PII scrubbing, and normalization. |
| **Oso** | Private | USA | Core authorization policy engine ('Polar') and official Rust language bindings for embedding. |
| **ThreatX** | Private | USA | Backend services and modifications to its NGINX-based sensor for sub-millisecond threat detection. |
| **AppSignal** | Private | Netherlands | The core of its lightweight and stable monitoring agent, ensuring a small footprint in customer applications. |
| **Red Sift** | Private | UK | eBPF tooling (RedBPF library) and ingraind, a metric collector agent for security monitoring. |
| **BlueCat Networks** | Private | North America | Critical network infrastructure components, including an open-source DHCPv4 server (dora). |
| **Imperva** | Private | USA | RASP core analysis engine and bot detection/mitigation platform where low latency and security are key. |
| **Starry Network** | Private | USA | Critical networking functions in its starry\_node project, including P2P services and crypto authentication. |
| **Yomura Fiber** | Private | USA | GPON provisioning and statistic gathering systems, core to its fiber network operations. |

Source: Synthesized from Security, Networking, and Observability Companies data.1

## Powering the Future of Data and Intelligence

Rust's influence is rapidly expanding into the performance-critical layers of the AI, machine learning, and data science ecosystems. While Python remains the dominant language for high-level research, experimentation, and orchestration, Rust is becoming the essential language for the underlying infrastructure that demands maximum performance, memory efficiency, and concurrency. This emerging architectural pattern, a symbiotic Python/Rust duality, leverages the strengths of both languages to build the next generation of data and AI platforms.1

### The Performance Backbone of the AI Ecosystem

Leading AI platforms and research organizations are relying on Rust to build their core, speed-sensitive components, where even small performance gains can translate into significant reductions in training time and inference latency.

**OpenAI**, a leader in artificial intelligence research, uses Rust for its performance-critical tooling. A key example is **tiktoken**, a high-speed Byte Pair Encoding (BPE) tokenizer for its language models.1 This library is reported to be 3 to 6 times faster than comparable open-source tokenizers. This is not a trivial improvement; tokenization is a fundamental and frequently executed step in processing data for LLMs. Faster tokenization directly translates to faster data preparation, which in turn accelerates training and inference pipelines. OpenAI also rewrote its Codex CLI tool from TypeScript to Rust to achieve faster execution, a smaller binary footprint, and enhanced security sandboxing.48

**Hugging Face**, the central hub of the open-source AI community, has built its foundational libraries in Rust to provide the performance backbone for its platform. This includes its widely used **tokenizers** library, which can process a gigabyte of text in under 20 seconds, the **safetensors** format for secure and fast model sharing, and the **candle** minimalist ML framework. These Rust-based tools enable the high-speed data processing necessary for training and serving modern machine learning models at scale.1

### Rewriting Data Platforms for a New Level of Performance

A growing number of data-intensive companies have undertaken full rewrites of their core platforms in Rust, a significant engineering investment made to achieve order-of-magnitude performance gains and solve long-standing architectural challenges.

**InfluxData** re-engineered its next-generation time-series database, **InfluxDB 3.0**, moving from Go to Rust and building on top of Apache Arrow and DataFusion.1 This strategic decision yielded dramatic results: up to

**45x better write throughput** and **5-25x faster queries** compared to the previous version. This architectural shift allowed InfluxDB to finally solve the long-standing industry problem of handling high-cardinality data efficiently, unlocking new use cases in observability and IoT that were previously impractical.54

**Deepgram** migrated its entire AI speech recognition platform, including its neural speech engine for inference, from Python and Go to Rust. This move was driven by the need for superior performance and control. The switch resulted in a **30-80% gain in speed**, lower latencies, higher throughput, and enhanced request concurrency, directly improving the quality and efficiency of its core product offering.1

This pattern reveals a powerful architectural trend in modern AI stacks: a Python front-end for ease of use, rapid experimentation, and high-level orchestration, coupled with a Rust core for raw computational performance. This is not a competition between languages but a symbiotic relationship. Rust handles the heavy lifting—such as tokenization, data parsing, and inference calculations—which is then exposed to the vast Python ecosystem via robust bindings. This allows data scientists and ML engineers to work in their preferred, productive environment without sacrificing the performance and efficiency required for production-grade systems. This model leverages the best of both worlds and is becoming a defining characteristic of successful AI infrastructure.

**Table 4: Rust's Role in AI, Machine Learning, and Data Science Companies**

| Company | Public/Private Status | HQ Country | Rust Usage Details | Product Impact |
| --- | --- | --- | --- | --- |
| **OpenAI** | Private | USA | Performance-critical tooling: tiktoken (BPE tokenizer) and a rewrite of its Codex CLI tool. | tiktoken is 3-6x faster than comparable tokenizers; rewritten CLI is faster and more secure. |
| **Hugging Face** | Private | USA | Core ecosystem libraries: tokenizers, safetensors (secure model sharing), candle (ML framework). | tokenizers library processes 1GB of text in <20 seconds, enabling extremely fast data processing. |
| **Deepgram** | Private | USA | Entire AI speech recognition platform, including its neural speech engine for inference. | 30-80% gain in speed, lower latencies, higher throughput, and enhanced request concurrency. |
| **Rerun.io** | Private | USA | Core data stack and visualization tools for multimodal data, including a 2D/3D visualizer. | Provides a high-performance toolkit for robotics and spatial computing, used by labs like Meta and Deepmind. |
| **Materialize** | Private | USA | Core execution engine of its real-time data platform, based on Timely/Differential Dataflow. | Powers the ability to process and analyze rapidly changing data streams in real-time using standard SQL. |
| **Polars** | Private | Netherlands | The core query engine of its high-performance DataFrame library, built on Apache Arrow. | Central to Polars' exceptional speed in data manipulation, making it a popular high-performance choice. |
| **InfluxData** | Private | USA | Next-generation time-series database engine, InfluxDB 3.0, built on Apache Arrow and DataFusion. | Up to 45x better write throughput and 5-25x faster queries compared to the previous Go-based version. |
| **SurrealDB** | Private | UK | The entire SurrealDB multi-model database is built in Rust for performance, safety, and concurrency. | Enables a scalable, high-performance, and reliable database platform for modern, real-time web apps. |
| **Mezmo** | Private | USA | Logging agents (improving performance and adding ARM64 support) and use of the Vector tool. | Enhances the performance, platform support, and efficiency of its data collection agents. |
| **Datalust** | Private | Australia | Re-engineered core storage engine, Flare, for its Seq structured logging tool. | Resulted in a 25% overall performance improvement for the Seq platform and enhanced data durability. |

Source: Synthesized from AI, ML, and Data Science Companies data.1

## The Language of Trust: Rust in Safety-Critical and High-Integrity Domains

Beyond raw performance, Rust's most defining feature is its set of compile-time guarantees against common memory errors. This focus on safety and correctness is making Rust an indispensable tool in industries where software failure can have catastrophic consequences. From the complex electronic systems in modern vehicles to the secure handling of financial assets, Rust is emerging as the language of trust for building the next generation of safety-critical and high-integrity systems.1

### Automotive and Aerospace: Engineering for a Zero-Failure World

The automotive and aerospace industries are undergoing a profound transformation into software-defined domains. As vehicles and spacecraft become more autonomous and connected, the complexity and criticality of their software are increasing exponentially, making memory safety a paramount concern.

**Renault Group**, through its EV and software company **Ampere**, has officially adopted a "Rust-by-default" strategy for its next generation of Software-Defined Vehicles (SDV).1 The company is using Rust to secure on-board communication systems and other critical components, viewing the language as a way to address new cybersecurity concerns, simplify architecture, and improve time-to-market by preventing common programming errors in safety-critical systems.62

**Volvo Cars** has also officially confirmed that Rust code has been running in production vehicles, specifically the EX90 and Polestar 3 models, since October 2024.1 Their use cases include software for Electronic Control Units (ECUs) responsible for low-power management. Like Renault, Volvo is expanding its use of the language and aims for a "Rust-by-default" policy for new high-reliability projects.

In the aerospace sector, **Firefly Aerospace** has publicly detailed rewriting its entire flight control stack in Rust, a move that resulted in zero crash reports during operation.1 Furthermore, its Blue Ghost 1 lunar lander mission was powered in part by open-source Rust libraries, demonstrating Rust's viability in the demanding environment of space exploration.

**Toyota Motor Corporation** is also making a clear strategic move towards adoption. While initial usage was anecdotal, its subsidiary, **Woven by Toyota**, has officially joined the Safety-Critical Rust Consortium and is actively hiring Rust engineers for its vehicle software platforms, signaling a formal commitment to using Rust in safety-critical systems.1

### Embedded Systems and IoT: Reliability in Resource-Constrained Environments

Rust's ability to produce small, efficient binaries with no garbage collector makes it an ideal choice for embedded systems, IoT devices, and robotics, where resources are often constrained and reliability is essential.

**Espressif Systems**, a major manufacturer of Wi-Fi and Bluetooth microcontrollers like the ESP32, provides extensive and official support for Rust on its hardware.1 The company dedicates developer time to maintaining hardware abstraction layers (

esp-hal) and supports a thriving community ecosystem, making Rust a first-class citizen for embedded development on its popular chips.

In consumer robotics, **Matic Robots** has built approximately 95% of its autonomous floor cleaning robot's intelligence codebase in Rust.1 This includes critical on-device processing for real-time SLAM (Simultaneous Localization and Mapping) and 3D mapping, where Rust's performance, memory safety, and robust error handling are crucial for reliable autonomous operation.

Further demonstrating its suitability for embedded systems, Google developed **KataOS**, a secure operating system for embedded devices running on RISC-V chips, written entirely in Rust to create a verifiably secure foundation.1

### Financial Services and Cryptocurrency: Securing the Digital Vault

In the high-stakes world of financial services and cryptocurrency, the integrity and security of software are non-negotiable. Rust's strong safety guarantees and performance make it a compelling choice for building trading platforms, payment processors, and custodial solutions.

**Kraken**, a major cryptocurrency exchange, uses Rust extensively to modernize its core backend services, including distributed data storage, cryptography, and application services.1 Operating in a highly regulated environment, Kraken leverages Rust to build secure and performant infrastructure capable of handling the demands of a global trading platform.

**Hyperswitch**, an open-source financial switch built entirely in Rust, provides a single API for businesses to connect to multiple payment processors.1 The platform is PCI DSS 3.2.1 Level 1 certified, the highest level of security certification for handling cardholder data, demonstrating Rust's suitability for building systems that meet the most stringent financial security standards.

**Table 5: Adoption of Rust in Automotive, Aerospace, and Transportation**

| Company | Public/Private Status | HQ Country | Application Areas | Adoption Classification |
| --- | --- | --- | --- | --- |
| **Renault (via Ampere)** | Public | France | SDV platform, in-vehicle software, ADAS, on-board communications | Official ('Rust-by-default' strategy announced). |
| **Volvo Cars** | Public | Sweden | In-vehicle software, ECU software for low-power management | Official (in production vehicles since Oct 2024). |
| **Firefly Aerospace** | Private | USA | Flight control systems, spacecraft software | Official (entire flight control stack rewritten in Rust). |
| **Toyota Motor Corporation** | Public | Japan | Vehicle OS (Arene OS), automated driving, safety-critical systems | Transitioning to Official (subsidiary joined consortium, active hiring). |
| **Starship Technologies** | Private | USA | Autonomous delivery robots (Level 4 autonomy) | Official (listed on official Rust user forum). |
| **Volkswagen (VW) Group** | Public | Germany | Infotainment software | Unofficial (evidence from third-party software firm collaboration). |
| **Honeywell** | Public | USA | Embedded technologies, high-performance secure software | Unofficial (evidence from third-party partner reports). |
| **Hove** | Private | Denmark | Transit data management and enrichment | Official (verifiable through open-source contributions). |
| **SpiderOak** | Private | USA | Space-based communications security | Official (demonstrated OrbitSecure module rewritten in Rust). |

Source: Synthesized from Automotive, Aerospace, and Transportation Companies data.1

**Table 6: Rust Implementations in Embedded Systems, IoT, and Robotics**

| Company | Public/Private Status | HQ Country | Device Verticals | Rust Usage Details |
| --- | --- | --- | --- | --- |
| **Espressif Systems** | Public | China | Wi-Fi/Bluetooth SoCs & MCUs | Provides extensive official support for Rust on its microcontrollers (ESP32), including HALs and training. |
| **System76** | Private | USA | Laptops and desktops | Heavily utilizes Rust in its firmware (system76-firmware) and host tooling for its Pop!\_OS. |
| **Matic Robots** | Private | USA | Consumer home robotics | ~95% of its autonomous robot's intelligence codebase is in Rust, including on-device SLAM and 3D mapping. |
| **UpdateHub** | Private | Brazil | IoT and industrial Linux devices | Firmware Over-The-Air (FOTA) agent was rewritten from Go to Rust for better memory predictability. |
| **STABL Energy** | Private | Germany | Energy storage systems | Uses Rust in production firmware on ESP32 microcontrollers for cloud connectivity and monitoring. |
| **SmartThings** | Private (Subsidiary) | South Korea | Smart home hubs and IoT devices | Utilized in firmware for SmartThings hubs and their Rule Engine for low-latency local automations. |
| **Google** | Public | USA | Embedded Operating Systems | Developed KataOS, a secure OS for embedded systems on RISC-V, written entirely in Rust. |
| **Kinetic** | Private | USA | Wearable technology | Develops the firmware for its wearable devices, designed to reduce workplace injuries, using Rust. |

Source: Synthesized from Embedded, IoT, and Robotics Companies data.1

**Table 7: Rust in Financial Services, FinTech, and Cryptocurrency**

| Company | Public/Private Status | HQ Country | Rust Usage Details | Regulatory Context |
| --- | --- | --- | --- | --- |
| **Kraken** | Private | USA | Modernizing core backend services (storage, crypto, apps); primary language for Derivatives team. | First crypto exchange to receive a US bank charter; operates in a highly regulated environment. |
| **Hyperswitch** | Private | India | Entire open-source financial switch is built in Rust, providing a single API to multiple payment processors. | PCI DSS 3.2.1 Level 1 certified, demonstrating suitability for high-security financial systems. |
| **Braintree** | Subsidiary of Public | USA | Targeted use for performance-critical tasks like accelerating batch processing and for small CLI utilities. | As part of PayPal, fully compliant with PCI DSS standards. |
| **TenX** | Private | Singapore | Building a secure and scalable distributed payments system; monolith rewritten in Rust. | Became PCI-DSS compliant during its re-platforming process to Rust. |
| **ANIXE** | Acquired | Poland | Built its next-generation travel services trading platform in Rust (first production service in 2016). | Handles financial transactions within the travel industry, where memory safety is a key benefit. |
| **Spoqa** | Private | South Korea | Point of Sale (POS) integration SDK is written entirely in Rust. | Operates in a space where transaction integrity and system reliability are crucial. |

Source: Synthesized from Financial Services, FinTech, and Crypto Companies data.1

## From Milliseconds to Market Share: Translating Performance into Product Superiority

In consumer-facing industries like e-commerce, media, and gaming, performance is not just a technical metric—it is a core product feature that directly impacts user experience, engagement, and ultimately, market share. Companies in these sectors are adopting Rust to gain a competitive edge by building faster, more responsive, and more reliable applications. The analysis across these verticals demonstrates how Rust's ability to deliver low-latency, high-throughput performance translates directly into tangible business outcomes.1

### E-commerce and Logistics: Optimizing for Peak Demand

For e-commerce and logistics platforms, handling massive traffic spikes and optimizing complex real-time calculations are critical for success.

**Shopify**, a leading e-commerce platform, leverages Rust for its most performance-critical systems.1 Its most notable use is in

**YJIT**, a new Just-In-Time compiler for Ruby, which was ported to Rust to enhance safety and maintainability. YJIT delivered an average of **14.1% faster latency** on Shopify's core Storefront Renderer, a crucial improvement for a platform that handles over **80 million requests per minute** during peak events like the Black Friday Cyber Monday weekend. This demonstrates how Rust can be used to augment and significantly accelerate ecosystems built on other languages, providing substantial performance gains for existing, large-scale applications without a full rewrite.66

**Deliveroo**, a major online food delivery company, migrated its critical Tier 1 Dispatcher service from Ruby to Rust.1 This service, responsible for the optimal assignment of delivery orders to riders, saw a significant performance boost. In one large operational zone, the dispatch processing time dropped from approximately 4 seconds to just 0.8 seconds. This improvement in a core logistics algorithm directly impacts the efficiency and speed of their delivery network.

### Consumer Media and Gaming: Delivering a Flawless Real-Time Experience

In media and gaming, where real-time interaction and low latency are paramount, Rust's lack of a garbage collector and its focus on performance provide a decisive advantage.

**Brave**, the company behind the privacy-focused web browser, wrote its core ad-blocking engine, adblock-rust, entirely in Rust.1 This resulted in a transformative

**69x average performance improvement**, reducing the time to classify network requests to just 5.7 microseconds. This not only speeds up browsing for users but also lowers CPU usage on their devices, directly enhancing the product's core value proposition.

**Netflix**, while not using Rust in its core video streaming pipeline, has demonstrated its use in specific, high-performance applications. A senior software engineer presented on using the tracing crate in Rust for low-overhead observability in servers that handle a very high volume of requests per second, a critical capability for maintaining reliability at Netflix's scale.1

### Productivity and Healthtech: Ensuring Correctness and Performance

In productivity tools, where responsiveness is key, and in healthtech, where correctness is vital, Rust provides a powerful combination of speed and safety.

**Dropbox** uses Rust in multiple performance-critical areas. Its core file synchronization engine, 'Nucleus', was rewritten in Rust to improve correctness by encoding complex system invariants directly into the type system.1 Additionally, a custom Rust library was developed for the

**Dropbox Capture** product, which enabled higher-quality screen recording (up to 4K), faster sharing, and a smaller application footprint by removing 17MB of Swift libraries on macOS.71

In the healthtech sector, **CancerIQ** strategically rewrote its complex cancer risk prediction models in Rust, creating a standalone JSON API.1 This decision was driven by the need to leverage Rust's strong guarantees of memory safety, thread safety, and robust error handling for critical calculations where correctness is of the utmost importance.

**Table 8: Rust Usage in Consumer, Media, and Gaming Applications**

| Company | Public/Private Status | HQ Country | Rust Usage Details | Impact Narrative |
| --- | --- | --- | --- | --- |
| **Discord** | Private | USA | Backend 'Read States' service, Elixir NIFs, client-side video encoding ('Go Live'). | Eliminated GC latency spikes, enabling scaling to >11M concurrent users with consistent performance. |
| **Brave** | Private | USA | Core ad-blocking engine (adblock-rust) integrated directly into the browser. | 69x average performance improvement, reducing request classification time to 5.7 microseconds. |
| **Figma** | Private | USA | High-performance, real-time multiplayer syncing engine rewritten from TypeScript. | Significant performance improvements and low resource usage, enabling seamless real-time collaboration. |
| **Dropbox** | Public | USA | Custom library for Dropbox Capture (screen recording) and core file sync engine ('Nucleus'). | Enabled 4K screen recording, faster sharing, and a smaller app footprint (saved 17MB of libraries). |
| **Amedia** | Private | Norway | Content platform (Innholdsplattform) and backend services for user preferences. | Aimed at building robust, stable, and easily maintainable solutions for over two million daily readers. |
| **Netflix** | Public | USA | Used in specific, non-core applications, such as using the tracing crate for observability. | Enables low-overhead performance monitoring for servers handling very high traffic volumes. |
| **Matchday** | Private | Spain | Confirmed usage in the gaming startup's technology stack. | Specific impact details are not publicly available. |

Source: Synthesized from Consumer, Media, and Gaming Companies data.1

**Table 9: Performance-Driven Rust Adoption in E-commerce and Logistics**

| Company | Public/Private Status | HQ Country | Rust Components | Benefits |
| --- | --- | --- | --- | --- |
| **Shopify** | Public | Canada | YJIT (JIT compiler for Ruby), high-performance network servers, Wasm extensions. | 14.1% average latency improvement on Storefront Renderer; crucial for handling 80M requests/min. |
| **Deliveroo** | Public | UK | Critical Tier 1 Dispatcher service (migrated from Ruby), bindings for OSRM. | Significant performance boost; dispatch processing time dropped from ~4s to 0.8s in a large zone. |
| **OneSignal** | Private | USA | Core push notification delivery system (OnePush), Kafka consumers for analytics. | 24x increase in overall delivery rate and 14x increase in max burst rate; data-race freedom. |
| **Hyperswitch** | Private | India | The entire open-source global payments orchestrator is built with Rust at its core. | High performance, reliability, and safety for a critical payment processing system. |
| **Symless** | Private | UK | Initially used for background service of Synergy 3; still used for internal tools. | Initial motivation was memory safety for a long-running process; provides a counterpoint on adoption challenges. |

Source: Synthesized from E-commerce and Logistics Companies data.1

**Table 10: Rust Applications in Productivity, Healthtech, and Education**

| Company | Public/Private Status | HQ Country | Sector | Product Areas Influenced by Rust |
| --- | --- | --- | --- | --- |
| **10x Genomics** | Public | United States | Biotech | Genomic analysis software (Cell Ranger, Space Ranger) for computationally intensive algorithms. |
| **CancerIQ** | Private | United States | Healthtech | Complex cancer risk prediction models rewritten into a standalone JSON API for safety and robustness. |
| **Fulcrum Genomics** | Private | United States | Biotech | Specialized bioinformatics toolkit (fqtk) for processing FASTQ genomic data files. |
| **Benchling** | Private | United States | Biotech | Provides an official Rust client library (benchling-rs) to interact with its R&D cloud platform. |
| **Coursera** | Public | United States | Education | Critical security components within its programming assignments infrastructure for grading student code. |
| **Figma** | Private | United States | Productivity | Real-time multiplayer syncing server rewritten for best-in-class speed and low resource usage. |
| **Dropbox** | Public | United States | Productivity | Core file sync engine ('Nucleus') and a custom library for Dropbox Capture for 4K screen recording. |
| **ToEverything** | Private | Singapore | Productivity | OctoBase, an open-source, local-first collaborative data engine for its knowledge management app. |
| **Discord** | Private | United States | Productivity | 'Read States' service migrated from Go to solve latency spikes; video encoding and Elixir backend scaling. |
| **Vercel** | Private | United States | Productivity | High-performance developer tooling (Turborepo, Turbopack, SWC) to speed up the web dev lifecycle. |
| **Zed.dev** | Private | United States | Productivity | The Zed code editor is built almost entirely in Rust for a high-performance, collaborative experience. |

Source: Synthesized from Productivity, Healthtech, and Education Companies data.1

## The Global Ecosystem: A Landscape of Expertise, Support, and Regional Momentum

The maturation of a programming language is marked not only by its adoption in production systems but also by the growth of a robust ecosystem that supports its continued use and expansion. The global landscape for Rust now includes a growing number of specialized consulting and training firms, alongside strong regional adoption hubs. This indicates that Rust has transitioned from a niche language, championed by early adopters, to a mainstream technology with the commercial and community infrastructure necessary for widespread enterprise deployment.1

### A Mature Consulting and Training Ecosystem

The emergence of specialized consulting firms is a key indicator of enterprise readiness. These organizations provide the expert training, architectural guidance, and implementation support that large companies require when adopting a new technology. The Rust ecosystem now boasts a healthy number of such firms across North America and Europe.

Companies like **Ferrous Systems** (Germany), creators of Ferrocene, a qualified Rust toolchain for safety-critical applications, and **Tweede golf** (Netherlands), which is developing core internet infrastructure components like sudo-rs, provide deep expertise in systems-level and embedded Rust.1 In the United States, firms like

**Integer 32** and **Ardan Labs** offer custom training and team augmentation, while **Immunant, Inc.** specializes in the complex task of migrating legacy C/C++ codebases to safer Rust code.1 The existence of these firms lowers the barrier to entry for large organizations, providing a clear path for upskilling existing teams and successfully executing complex Rust projects.

### Adoption in Telecommunications and Regional Hubs

Rust's performance and reliability are also making it a strong candidate for the telecommunications sector, where uptime and low latency are critical. ISPs like **Yomura Fiber** (USA) are using Rust to power core network operations, including their GPON provisioning and statistics gathering systems.1

**Starry Network** (USA) uses Rust for peer-to-peer networking services in its starry\_node project. Furthermore, **Huawei** (China), a global telecommunications giant and a founding member of the Rust Foundation, has officially migrated parts of its codebase for telecommunications equipment to Rust, valuing its performance and security.1

Analysis of regional adoption patterns reveals significant momentum in South Asia, particularly **India**. The Indian market shows remarkably strong, diverse, and widespread adoption. The language is in high demand across multiple key sectors, including Blockchain/Web3, backend systems, embedded systems (especially automotive), AI/ML, and FinTech. Large multinational corporations like Infosys and Bosch, specialized tech startups, and IT service firms are all actively hiring for Rust roles, indicating a mature and rapidly growing market that contrasts with other regions where adoption may be more concentrated in a single sector.1

**Table 11: Global Directory of Rust Consulting and Training Providers**

| Company | Public/Private Status | HQ Country | Services Offered |
| --- | --- | --- | --- |
| **Ferrous Systems** | Private | Germany | Specialized Rust development, training, and support; creators of the Ferrocene qualified toolchain. |
| **Tweede golf** | Private | Netherlands | Rust engineering, consultancy, and training for systems, web, and embedded platforms. |
| **Integer 32** | Private | United States | Custom Rust training, prototype development, and team augmentation. |
| **Immunant, Inc.** | Private | United States | Specializes in migrating legacy C/C++ codebases to Rust to improve safety and security. |
| **Red Iron (OCamlPro)** | Private | France | Robust Rust development, advising, code reviews, performance audits, and training programs. |
| **Rustunit** | Private | Netherlands | Consulting for Rust, game development, and large-scale distributed backend services. |
| **Corrode** | Private | Germany | Training, consulting, mentoring, and team augmentation, specializing in backend systems and APIs. |
| **Fledgio** | Private | United Kingdom | Hands-on Rust bootcamps, online courses, and tailored corporate training. |
| **Ardan Labs** | Private | United States | Staff augmentation, strategic consulting, practical training, and development services. |
| **Mainmatter** | Private | Germany | Helps teams build APIs, data pipelines, and web solutions with Rust; organizes EuroRust conference. |

Source: Synthesized from Consulting and Training Providers data.1

**Table 12: Rust Adoption in Telecommunications and ISP Infrastructure**

| Company | Public/Private Status | HQ Country | Evidence of Usage | Confidence Score |
| --- | --- | --- | --- | --- |
| **Huawei** | Private | China | As a Rust Foundation member, has officially migrated parts of its codebase for telecom equipment. | High |
| **Yomura Fiber** | Private | United States | Officially quoted stating, "Rust powers our GPON provisioning and statistic gathering." | High |
| **Starry Network** | Private | United States | Maintains a public GitHub repository ('starry\_node') written in Rust for P2P networking services. | High |
| **AT&T** | Public | United States | Engineering blog post discusses Rust for real-time backend tasks and specific microservices. | Medium |
| **Openreach (BT Group)** | Public | United Kingdom | An official-looking (though now 'yanked') Rust crate suggests past or experimental development. | Medium |
| **Calyptech** | Private | Australia | Third-party resources list usage for high-performance embedded system components. | Medium |
| **NTT Communications** | Public | Japan | Engineer's blog post advocates for Rust in serverless, confirming expertise and interest. | Low |

Source: Synthesized from Telecom and ISP Companies data.1

## Conclusion

The evidence presents a clear and compelling narrative: Rust has transcended its origins as a promising systems language to become a strategic technology asset for a broad and growing spectrum of global industries. Its adoption is not a monolithic trend but a multifaceted movement driven by two powerful, concurrent forces.

On one hand, the world's largest technology corporations are systematically integrating Rust into their most critical infrastructure as a means of fortification. For these public giants, Rust is a long-term investment in stability, security, and operational efficiency, providing a robust solution to the decades-old problem of memory safety vulnerabilities in foundational software. The measurable reduction in security flaws at companies like Google and Microsoft provides a powerful testament to Rust's effectiveness in this role.

On the other hand, a vibrant ecosystem of private companies and startups is leveraging Rust as a tool of disruption. For these agile innovators, Rust's unparalleled performance and efficiency are not just technical benefits but direct competitive advantages. By building faster, more reliable, and more resource-efficient products—from real-time communication platforms and collaborative design tools to next-generation databases and AI infrastructure—these companies are using Rust to challenge incumbents and redefine what is possible in their respective markets.

This dual adoption pattern creates a virtuous cycle that is rapidly accelerating the maturity of the Rust ecosystem. The stability and investment provided by large corporations create a reliable foundation, while the rapid innovation of startups pushes the boundaries of the language and its libraries. The emergence of a global network of specialized consulting firms further solidifies Rust's position as an enterprise-ready technology.

Ultimately, the ascendancy of Rust is a story of a language uniquely suited to the demands of modern software development. It resolves the long-standing trade-off between high-level safety and low-level performance, offering a path forward for building the secure, efficient, and reliable systems that will power the next wave of technological innovation. For strategic technology leaders, the conclusion is clear: Rust is no longer an emerging technology to watch, but a foundational one to be actively considered for any project where performance, reliability, and security are paramount.

## Appendix: Research Framework

### Unofficial Usage Validation

* **Company Name:** Volkswagen, Toyota, and Honeywell
* **Conclusion:** The nature of Rust adoption varies significantly among these three companies. Volkswagen's usage is now considered official and strategic, driven by its software subsidiary CARIAD. Toyota's adoption is also official and strategic, confirmed through its subsidiaries Woven by Toyota and Toyota Connected, which are actively hiring for Rust roles and participating in safety-critical consortia. In contrast, Honeywell's usage remains unconfirmed and anecdotal, with no official statements, job postings, or internal projects to support claims of corporate adoption.
* **Detailed Findings:** For Volkswagen, evidence includes a public webinar hosted by its subsidiary CARIAD on 'Rust: A game changer in automotive software development' and presentations at the Oxidize Conference 2024. A CARIAD developer also publicly confirmed Rust usage in an interview. For Toyota, evidence includes its subsidiary Woven by Toyota joining the Safety-Critical Rust Consortium, official job postings for Rust engineers, and Toyota Connected's membership in the Rust Foundation. For Honeywell, the only link found was a third-party, community-maintained Rust package for one of its APIs, with no official company involvement. Searches for official job postings or engineering blogs from Honeywell yielded no results related to the Rust programming language.
* **Public Status:** All three parent companies—Volkswagen AG, Toyota Motor Corporation, and Honeywell International Inc.—are publicly traded.
* **Confidence Rating:** High confidence for the conclusions on Volkswagen and Toyota, based on multiple official sources from their subsidiaries. Low confidence for any official usage at Honeywell, as the evidence is non-existent or purely anecdotal.

### Rust Adoption Methodology

* **Inclusion Criteria:** A company is considered to be 'using Rust' if there is credible evidence of its application in one or more of the following scopes: 1) Production Services, where Rust is used in live, customer-facing services or critical backend infrastructure; 2) Core Tooling, where Rust is used to build internal development tools, build systems, or other essential operational software; 3) Open-Source Software (OSS) Powering Products, where the company actively develops or maintains significant open-source Rust projects that are integral to its commercial offerings; or 4) Official Job Postings, where the company has verifiable job listings that explicitly require Rust programming skills for a specific role.
* **Evidence Hierarchy and Scoring:** To evaluate the quality of evidence, a framework adapted from the NATO Admiralty Code is used, assigning a two-part score for Source Reliability (A-F) and Information Credibility (1-6). This results in a Confidence Score (e.g., A1, C3). The hierarchy of evidence, from most to least reliable, is as follows: 1) Official Company Engineering Blogs/Documentation (Highest Confidence, e.g., A1); 2) Source Code in Official Company GitHub Organizations (High Confidence, e.g., A1); 3) Conference Talks by Company Employees at reputable events (High Confidence, e.g., B2); 4) Official Job Postings on company career pages (Medium Confidence, e.g., B2); and 5) Third-Party Reports, including news articles and community discussions, which require careful vetting and corroboration (Variable Confidence, e.g., C3 to D4).
* **Funding Status Classification:** Companies are classified into one of the following categories based on their ownership and funding structure: 1) Public Company: A company whose shares are traded on a public stock exchange, verified using SEC filings (like the CIK Lookup tool) and official exchange directories, with the exchange and ticker symbol recorded. 2) Private Company: A business without public ownership, sub-classified as either VC-Funded (verified via press releases, Crunchbase) or Bootstrapped (verified via company history or founder interviews). 3) Subsidiary of a Public Company: A firm controlled by a publicly-traded parent, verified via the parent's SEC filings (Exhibit 21). 4) Nonprofit-Owned: An organization operated by a non-profit entity, verified via official registries.
* **Data Model Fields:** To ensure comprehensive and structured information, each entry in the final dataset includes the following fields: company\_name (normalized official name), sector (industry classification), hq\_country (headquarters country), evidence\_link (URL to the source), evidence\_quote (specific text confirming usage), evidence\_date (publication date of evidence), usage\_scope (e.g., 'production service', 'core tooling'), funding\_status (e.g., 'public', 'private\_VC\_funded'), ticker (stock symbol if public), funding\_notes (details on investors or parent company), last\_verified\_date (date of last accuracy check), and confidence\_score (the assigned Admiralty Code rating).

#### Works cited

1. trun\_4122b840faa84ad7a24867ec3a76c16a (copy).txt
2. Rust Archives - Microsoft Open Source Blog, accessed on August 16, 2025, <https://opensource.microsoft.com/blog/programming-language/rust/>
3. Microsoft Azure security evolution: Embrace secure multitenancy, Confidential Compute, and Rust, accessed on August 16, 2025, <https://azure.microsoft.com/en-us/blog/microsoft-azure-security-evolution-embrace-secure-multitenancy-confidential-compute-and-rust/>
4. Check Point Research Uncovers Six New Microsoft Vulnerabilities, accessed on August 16, 2025, <https://enterpriseitworldmea.com/check-point-research-uncovers-six-new-microsoft-vulnerabilities-including-first-rust-based-kernel-flaw/>
5. Six New Windows Vulnerabilities Found, Including First Rust-Based Kernel Flaw, accessed on August 16, 2025, <https://informationsecuritybuzz.com/six-new-windows-vulnerabilities-found-including-first-rust-based-kernel-flaw/>
6. Android Rust introduction | Android Open Source Project, accessed on August 16, 2025, <https://source.android.com/docs/setup/build/rust/building-rust-modules/overview>
7. Google Security blog: Scaling Rust Adoption Through Training - Rust Users Forum, accessed on August 16, 2025, <https://users.rust-lang.org/t/google-security-blog-scaling-rust-adoption-through-training/100454>
8. Google's Shift to Rust Programming Cuts Android Memory Vulnerabilities by 68% - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/programming/comments/1iub0rk/googles_shift_to_rust_programming_cuts_android/>
9. Two Android Engineers Explain How They Extended Rust In Android's Firmware - Slashdot, accessed on August 16, 2025, <https://developers.slashdot.org/story/24/09/08/0455238/two-android-engineers-explain-how-they-extended-rust-in-androids-firmware>
10. Android's Rusty new code shakes off huge number of memory-safe ..., accessed on August 16, 2025, <https://www.threatdown.com/blog/androids-rusty-new-code-shakes-off-huge-number-of-memory-safe-vulnerabilities/>
11. Rust | AWS Open Source Blog, accessed on August 16, 2025, <https://aws.amazon.com/blogs/opensource/tag/rust/>
12. Firecracker, accessed on August 16, 2025, <https://firecracker-microvm.github.io/>
13. Announcing the Firecracker Open Source Technology: Secure and Fast microVM for Serverless Computing - AWS, accessed on August 16, 2025, <https://aws.amazon.com/blogs/opensource/firecracker-open-source-secure-fast-microvm-serverless/>
14. Firecracker – Lightweight Virtualization for Serverless Computing | AWS News Blog, accessed on August 16, 2025, <https://aws.amazon.com/blogs/aws/firecracker-lightweight-virtualization-for-serverless-computing/>
15. Discord infra engineer here -- this blog post needs an update! Since then we've - Hacker News, accessed on August 16, 2025, <https://news.ycombinator.com/item?id=19240040>
16. Engineering - Discord Blog - Medium, accessed on August 16, 2025, <https://medium.com/discord-engineering/tagged/engineering>
17. Engineering & Developers - Discord Blog, accessed on August 16, 2025, <https://discord.com/category/engineering>
18. “We rewrote it in Rust” articles with good before/after comparisons? - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/rust/comments/nnwkkj/we_rewrote_it_in_rust_articles_with_good/>
19. Rust in production at Figma : r/rust - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/rust/comments/8gic2a/rust_in_production_at_figma/>
20. Rust in production at Figma | Hacker News, accessed on August 16, 2025, <https://news.ycombinator.com/item?id=16977932>
21. accessed on January 1, 1970, <https://www.figma.com/blog/rust-in-production-at-figma/>
22. The Turbopack vision - Vercel, accessed on August 16, 2025, <https://vercel.com/blog/the-turbopack-vision>
23. Turbopack: High-performance bundler for React Server ... - Vercel, accessed on August 16, 2025, <https://vercel.com/blog/turbopack>
24. The creator of Webpack introduces Turbopack, a Rust-based successor that's 700x faster - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/rust/comments/ydddvq/the_creator_of_webpack_introduces_turbopack_a/>
25. Turbopack, new Hotness or promising Alpha - Valor Software, accessed on August 16, 2025, <https://valor-software.com/articles/turbopack-new-hotness-or-promising-alpha>
26. vercel/turborepo: Build system optimized for JavaScript and TypeScript, written in Rust, accessed on August 16, 2025, <https://github.com/vercel/turborepo>
27. API Reference: Turbopack | Next.js, accessed on August 16, 2025, <https://nextjs.org/docs/app/api-reference/turbopack>
28. Turbopack build feedback · vercel next.js · Discussion #77721 - GitHub, accessed on August 16, 2025, <https://github.com/vercel/next.js/discussions/77721>
29. Why Turborepo is migrating from Go to Rust - Vercel, accessed on August 16, 2025, <https://vercel.com/blog/turborepo-migration-go-rust>
30. Pingora - The Cloudflare Blog, accessed on August 16, 2025, <https://blog.cloudflare.com/tag/pingora/>
31. Resolving a request smuggling vulnerability in Pingora - The Cloudflare Blog, accessed on August 16, 2025, <https://blog.cloudflare.com/resolving-a-request-smuggling-vulnerability-in-pingora/>
32. CVE - Noise, accessed on August 16, 2025, <https://noise.getoto.net/tag/cve/>
33. Pingora has a Request Smuggling Vulnerability · CVE-2025-4366 - GitHub, accessed on August 16, 2025, <https://github.com/advisories/GHSA-93c7-7xqw-w357>
34. Request Smuggling Vulnerability in Pingora · Advisory - GitHub, accessed on August 16, 2025, <https://github.com/cloudflare/pingora/security/advisories/GHSA-93c7-7xqw-w357>
35. Cloudflare Closes Security Gap That Could Leak Visitor URLs - Cybersecurity News | Real-Time Breach Alerts & Vulnerability Intelligence | Infopercept, accessed on August 16, 2025, <https://www.infopercept.com/news/cloudflare-fixes-cve-2025-4366-in-pingora-oss>
36. Resolving a request smuggling vulnerability in Pingora - SSL / TLS - Cloudflare Community, accessed on August 16, 2025, <https://community.cloudflare.com/t/resolving-a-request-smuggling-vulnerability-in-pingora/801137>
37. Cloudflare CVEs and Security Vulnerabilities - OpenCVE, accessed on August 16, 2025, <https://www.opencve.io/cve?vendor=cloudflare>
38. How we migrated our static analyzer from Java to Rust | Datadog, accessed on August 16, 2025, <https://www.datadoghq.com/blog/engineering/how-we-migrated-our-static-analyzer-from-java-to-rust/>
39. Relay as a Lambda Extension Using the Actor Model in Rust ..., accessed on August 16, 2025, <https://blog.sentry.io/relay-as-a-lambda-extension-using-the-actor-model-in-rust/>
40. relay - Rust, accessed on August 16, 2025, <https://getsentry.github.io/relay/>
41. rust - Sentry Engineering, accessed on August 16, 2025, <https://sentry.engineering/tags/rust>
42. Relay Best Practices - Sentry Developer Documentation, accessed on August 16, 2025, <https://develop.sentry.dev/ingestion/relay/relay-best-practices/>
43. getsentry/relay: Sentry event forwarding and ingestion service. - GitHub, accessed on August 16, 2025, <https://github.com/getsentry/relay>
44. Operating Guidelines - Relay - Sentry Docs, accessed on August 16, 2025, <https://docs.sentry.io/product/relay/operating-guidelines/>
45. 1Password with Andrew Burkhart - Rust in Production Podcast, accessed on August 16, 2025, <https://corrode.dev/podcast/s04e06-1password/>
46. 1Password X 1.17: New brain, new menu, and even more accessible, accessed on August 16, 2025, <https://blog.1password.com/1passwordx-december-2019-release/>
47. Blog | 1Password, accessed on August 16, 2025, <https://blog.1password.com/>
48. Reflections on OpenAI - Calvin French-Owen, accessed on August 16, 2025, <https://calv.info/openai-reflections>
49. The Rust Renaissance: Why OpenAI and the Tech World are Shifting from Node.js for High-Performance Tools - Medium, accessed on August 16, 2025, <https://medium.com/@Fredtaylor1/the-rust-renaissance-why-openai-and-the-tech-world-are-shifting-from-node-js-d252efe79009>
50. openai/tiktoken: tiktoken is a fast BPE tokeniser for use with ... - GitHub, accessed on August 16, 2025, <https://github.com/openai/tiktoken>
51. zurawiki/tiktoken-rs: Ready-made tokenizer library for working with GPT and tiktoken, accessed on August 16, 2025, <https://github.com/zurawiki/tiktoken-rs>
52. darknight/tiktoken-rust: tiktoken is a fast BPE tokeniser for use with OpenAI's models. - GitHub, accessed on August 16, 2025, <https://github.com/darknight/tiktoken-rust>
53. An efficient Rust implementation of tiktoken, enhanced with the capability to calculate OpenAI token costs. Crafted for performance, accuracy, and seamless integration into your projects. - GitHub, accessed on August 16, 2025, <https://github.com/goliajp/tiktoken>
54. InfluxDB 3 Open Source Now in Public Alpha Under MIT/Apache 2 ..., accessed on August 16, 2025, <https://www.influxdata.com/blog/influxdb3-open-source-public-alpha/>
55. What InfluxDB Got Wrong | TigerData, accessed on August 16, 2025, <https://www.tigerdata.com/blog/what-influxdb-got-wrong>
56. Building InfluxDB 3.0 with Apache Arrow, DataFusion, Flight and Parquet - YouTube, accessed on August 16, 2025, <https://m.youtube.com/watch?v=I-Z7kFGsYRI>
57. InfluxDB v3: Why Rust Beat Go for Time Series Database - The New Stack, accessed on August 16, 2025, <https://thenewstack.io/influxdb-v3-why-rust-beat-go-for-time-series-database/>
58. Performance Benchmarking: InfluxDB 3 vs. InfluxDB Open Source | InfluxData, accessed on August 16, 2025, <https://www.influxdata.com/resources/influxdb-3-0-vs-oss/>
59. Performance Benchmarks for Time Series Database Platforms - InfluxDB, accessed on August 16, 2025, <https://www.influxdata.com/benchmarks/>
60. InfluxDB 3 Core: a complete rewrite designed for speed and simplicity | Grafana Labs, accessed on August 16, 2025, <https://grafana.com/blog/2025/07/03/influxdb-3-core-a-complete-rewrite-designed-for-speed-and-simplicity/>
61. InfluxDB 3.0 vs OSS Benchmarks Long Document, accessed on August 16, 2025, <https://get.influxdata.com/rs/972-GDU-533/images/InfluxDB-3-0-vs-OSS-Benchmarks.pdf>
62. Embedded Rust Adoption Tracking - The Embedded Rustacean, accessed on August 16, 2025, <https://www.theembeddedrustacean.com/p/embedded-rust-adoption-tracking>
63. Automotive Rust (are we there yet?) - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/rust/comments/1hx7lp3/automotive_rust_are_we_there_yet/>
64. Frédéric Ameye - Renault want to sell cars with rust! - YouTube, accessed on August 16, 2025, <https://www.youtube.com/watch?v=Z1xMvm3eS4k>
65. Qualcomm and Renault Group intend to extend their strategic cooperation to Renault's new electric and software company, Ampere, to co-develop a centralized platform architecture for software-defined electric vehicles powered by the Snapdragon Digital Chassis - Site media global de Renault Group, accessed on August 16, 2025, <https://media.renaultgroup.com/qualcomm-and-renault-group-intend-to-extend-their-strategic-cooperation-to-renaults-new-electric-and-software-company-ampere-to-co-develop-a-centralized-platform-architecture-for-software-defined-ele/>
66. Ruby 3.2's YJIT is Production-Ready - Shopify Engineering, accessed on August 16, 2025, <https://shopify.engineering/ruby-yjit-is-production-ready>
67. Our Experience Porting the YJIT Ruby Compiler to Rust | Hacker News, accessed on August 16, 2025, <https://news.ycombinator.com/item?id=31344065>
68. Ten minutes with YJIT - HashNotAdam, accessed on August 16, 2025, <https://www.hashnotadam.com/posts/2023/06/ten-minutes-with-yjit/>
69. Our Experience Porting the YJIT Ruby Compiler to Rust - Shopify Engineering, accessed on August 16, 2025, <https://shopify.engineering/porting-yjit-ruby-compiler-to-rust>
70. Shopify Embraces Rust for Systems Programming : r/ruby - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/ruby/comments/zm516o/shopify_embraces_rust_for_systems_programming/>
71. The Epic Story of Dropbox's Exodus From the Amazon Cloud Empire (feat. Rust) - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/rust/comments/4adabk/the_epic_story_of_dropboxs_exodus_from_the_amazon/>
72. Rewriting the Heart of our Sync Engine (in Rust) - Reddit, accessed on August 16, 2025, <https://www.reddit.com/r/rust/comments/fjt4q3/rewriting_the_heart_of_our_sync_engine_in_rust/>
73. Rewriting the heart of our sync engine - Dropbox Tech Blog, accessed on August 16, 2025, <https://dropbox.tech/infrastructure/rewriting-the-heart-of-our-sync-engine>
74. Rewriting the heart of our sync engine - Hacker News, accessed on August 16, 2025, <https://news.ycombinator.com/item?id=22595782>
75. Dropbox Tech Blog - Dropbox, accessed on August 16, 2025, <https://dropbox.tech/>