Rust 1.39.0 Release and Beyond

By *Roy Schestowitz*
Created *07/11/2019 - 4:33pm*
Submitted by Roy Schestowitz on Thursday 7th of November 2019 04:33:21 PM Filed under Development [1]

- **Announcing Rust 1.39.0 [2]**

  The Rust team is happy to announce a new version of Rust, 1.39.0. Rust is a programming language that is empowering everyone to build reliable and efficient software.

  [...]

  The highlights of Rust 1.39.0 include async/.await, shared references to by-move bindings in match guards, and attributes on function parameters. Also, see the detailed release notes for additional information.

- **Rust 1.39.0 released [3]**

  Version 1.39.0 of the Rust language is available. The biggest new feature appears to be the async/await mechanism, which is described in this blog post: "So, what is async await? Async-await is a way to write functions that can 'pause', return control to the runtime, and then pick up from where they left off. Typically those pauses are to wait for I/O, but there can be any number of uses."

- **Async-await on stable Rust! [4]**

  On this coming Thursday, November 7, async-await syntax hits stable Rust, as part of the 1.39.0 release. This work has been a long time in development -- the key ideas for zero-cost futures, for example, were first proposed by Aaron Turon and Alex Crichton in 2016! -- and we are very proud of the end result. We believe that Async I/O is going to be an increasingly

important part of Rust's story.

While this first release of "async-await" is a momentous event, it's also only the beginning. The current support for async-await marks a kind of "Minimum Viable Product" (MVP). We expect to be polishing, improving, and extending it for some time.

Already, in the time since async-await hit beta, we've made a lot of great progress, including making some key diagnostic improvements that help to make async-await errors far more approachable. To get involved in that work, check out the Async Foundations Working Group; if nothing else, you can help us by filing bugs about polish issues or by nominating those bugs that are bothering you the most, to help direct our efforts.

- **Support lifecycle for Clang/LLVM, Go, and Rust in Red Hat Enterprise Linux 8** **[5]**

  The Go and Rust languages continue to evolve and add new features with each compiler update, which is why so many users are interested in getting the latest versions of the compilers. At the same time, these compilers are designed to remain compatible with older code. So, even as we advance to newer versions of Go and Rust within the RHEL 8 application streams, you should not need to update your codebase to keep it compilable. Once you?ve compiled your valid code using the Go or Rust application stream, you can make the assumption that it will continue to compile with that stream for the full life of RHEL 8.

  We are excited to continue to bring you the latest and greatest in new compiler technologies. Stay tuned to the Red Hat Developer blog to learn more about what you can do with LLVM, Go, and Rust.

Development

Links:
[1] http://www.tuxmachines.org/taxonomy/term/145
[2] https://blog.rust-lang.org/2019/11/07/Rust-1.39.0.html
[3] https://lwn.net/Articles/804066/rss
[4] https://blog.rust-lang.org/2019/11/07/Async-await-stable.html
[5] https://developers.redhat.com/blog/2019/11/07/support-lifecycle-for-clang-llvm-go-and-rust-in-red-hat-enterprise-linux-8/