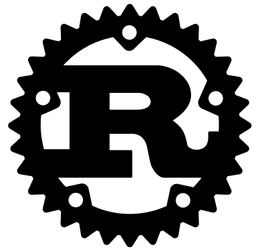
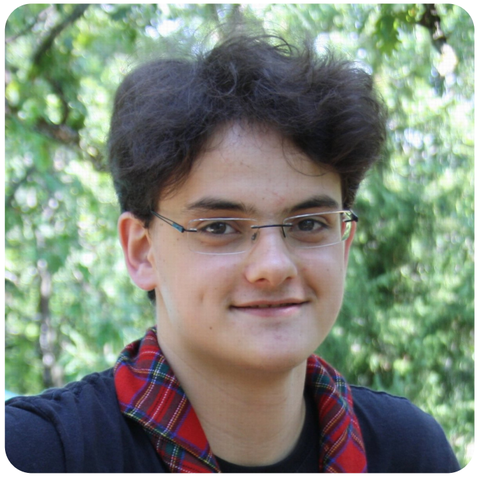


Shipping a stable compiler every six weeks

RustFest Barcelona, November 10th 2019



Pietro Albini

Rust Infrastructure team co-lead

Rust Release and crates.io teams member

Rust Security Response WG member

github.com/pietroalbini

www.pietroalbini.org

Rust 1.39.0 is out!

Released on November 7th, 2019.

Rust 1.38.0

Released on September 26th, 2019.

114,458 lines added and 91,886 lines removed.

5 regressions reported after the release (2 of them broke valid code).

Rust 1.37.0

Released on August 15th, 2019.

83,009 lines added, and 56,658 lines removed.

3 regressions reported after the release (all of them broke valid code).

Rust 1.36.0

Released on July 4th, 2019.

69,881 lines added, and 66,425 lines removed.

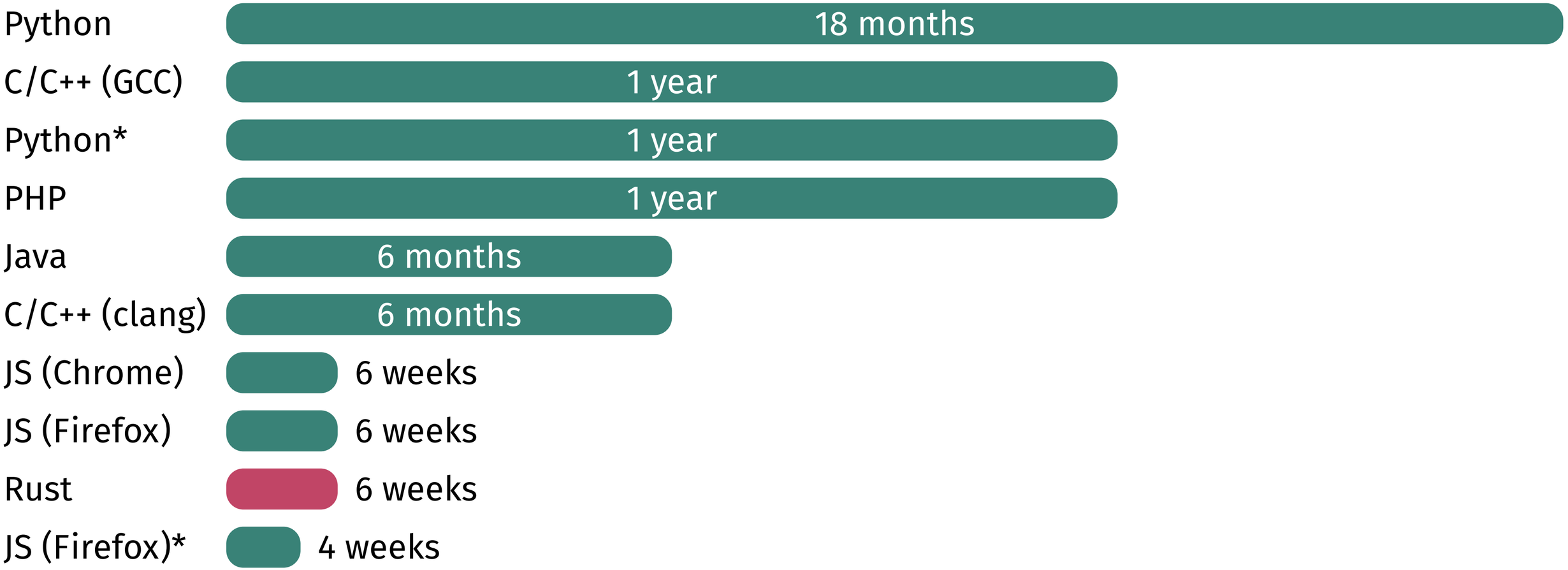
4 regressions reported after the release (2 of them broke valid code).

Why do we have this schedule?

How do we prevent regressions?

Why do we have this schedule?

It's unusual in the compiler world.



*: new schedule, planned to be used in the near future

No pressure to ship.

pietroalbini on Aug 13 Member

Rust 1.38 is scheduled to branch off today, but this PR didn't land yet and there are a bunch of blockers still open. If we still want to stabilize `async_await` in 1.38 we need to land this stabilization PR *right now* and then backport the blockers.

I'd prefer to stabilize `async_await` in Rust 1.39 though, to give the compiler and language team more time to properly iron the feature out. I'm aware lots of people would like to use `async_await` as soon as possible, but I'm not happy rushing things out.

cc **@rust-lang/lang @rust-lang/release**

👍 65 😕 6 ❤️ 18 🚀 7

Long release cycles don't work for us.

aturon (Aaron Turon) on Sep 5, 2018 • edited ▾ Member +😀 ✎ •••

EDIT: see this important update to the proposal.

We're getting close to the cut-off point for stabilizations for the Edition, and of course this feature is one of the most important ones remaining to get nailed down.

Based on the commentary in this tracking issue, the previous tracking issue, and various forum posts, I think it's fair to say that **the strong majority of people who have tried *any* variant of 2018 modules prefer it to 2015 modules**. And rustfix migration seems to be working well to limit the amount of manual churn required. So from a high level, I think we're in good shape to stabilize *some* variant for Rust 2018.

Given the limited time we have, I want to propose that we take a conservative route. We would ship the anchored paths variant, but with *future-proofing* that would make it possible to move to uniform paths later. The future proofing is simple: if `foo` is *both* an external crate name and a local item name, then a `use` statement must either say `use ::foo` or `use self::foo`, just as it would in the uniform paths variant.

aturon commented on Sep 6, 2018 Member •••

Given the concerns **@joshtriplett** raised -- the most important of which is that the future-proofing *is not currently implemented* and may take some time to land -- I want to revise the proposal slightly:

- For the upcoming release candidate 1, which will serve as a beta for the edition, we **stabilize anchored paths as-is**, but also allow you to opt in to uniform paths (on the beta channel) so that we can continue testing it and especially testing the ambiguity code.

- Over the next release cycle, we work to address remaining "false ambiguities" and to fully future-proof anchored paths for the final Edition release. (Or, potentially, we reach a firm consensus on one or the other path variants and just ship it directly, rather than the conservative version).

[beta] resolve: Implement edition hygiene for imports and absolute paths #56053

Edit

Merged **bors** merged 6 commits into `rust-lang:beta` from `petrochenkov:absedihyg` on Nov 26, 2018

Conversation 79 | Commits 6 | Checks 0 | Files changed 48 | +644 −541

petrochenkov (Vadim Petrochenkov) on Nov 19, 2018 • edited ▾ Member

The changes in behavior of imports and absolute paths are the most significant breaking changes of 2018 edition.
However, these changes are not covered by edition hygiene, so macros defined by 2015 edition crates expanded in 2018 edition crates are still interpreted in the 2018 edition way when they contain imports or absolute paths.
This means the promise of seamless integration of crates built with different editions, including use of macros, doesn't hold fully.
This PR fixes that and implements edition hygiene for imports and absolute paths, so they behave according to the edition in which they were written, even in macros.

Reviewers


nikomatsakis ✓


Centril ✓

Assignees


nikomatsakis

Labels

S-waiting-on-author

Thankfully it ended well.

Congrats to everyone involved in Rust 2018!

How do we prevent regressions?

The compiler's test suite.

Using the compiler
in the compiler itself.

Bug reports from users.

We can't ask people
to manually test beta.

Idea!

Let's test our users' code ourselves.

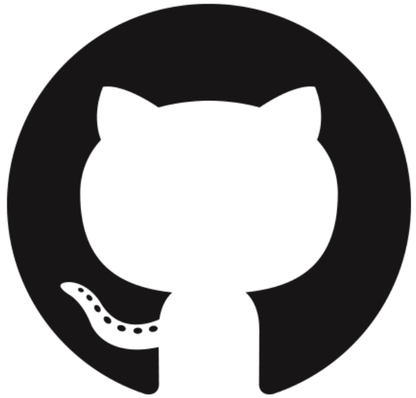
Crater

Crater

Queue Agents

Name	Assigned to	Reqs	Mode	Priority	Status
beta-1.40-1	distributed	linux	cargo test	10	Running (2%)
beta-1.40-rustdoc-1	-	linux	cargo doc	5	Queued

+



More than **75,000** projects tested, from crates.io and GitHub

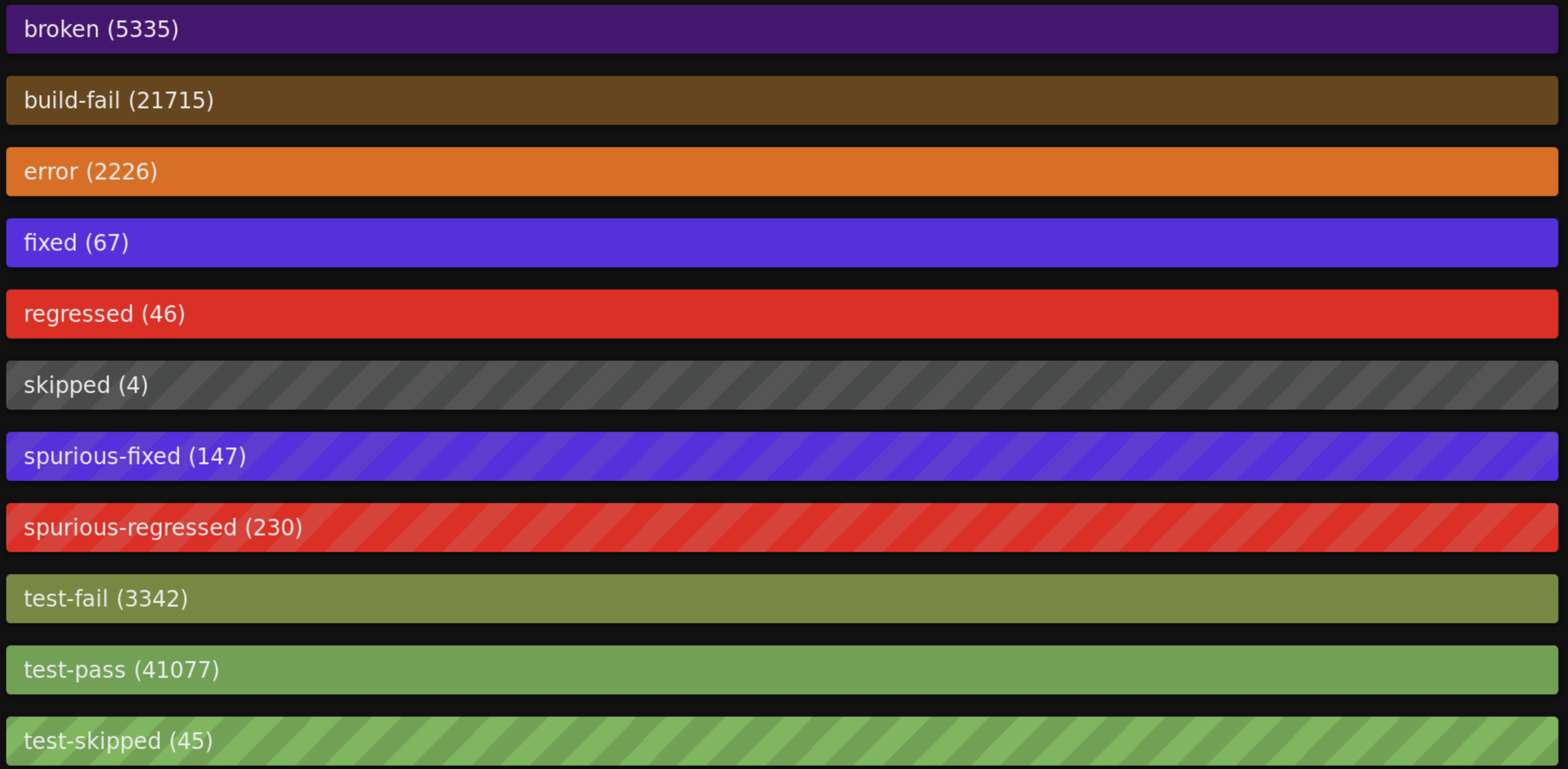
Run `cargo test` on every project
with two compiler builds.

Crater report for **beta-1.39-1**

Summary Full report Downloads

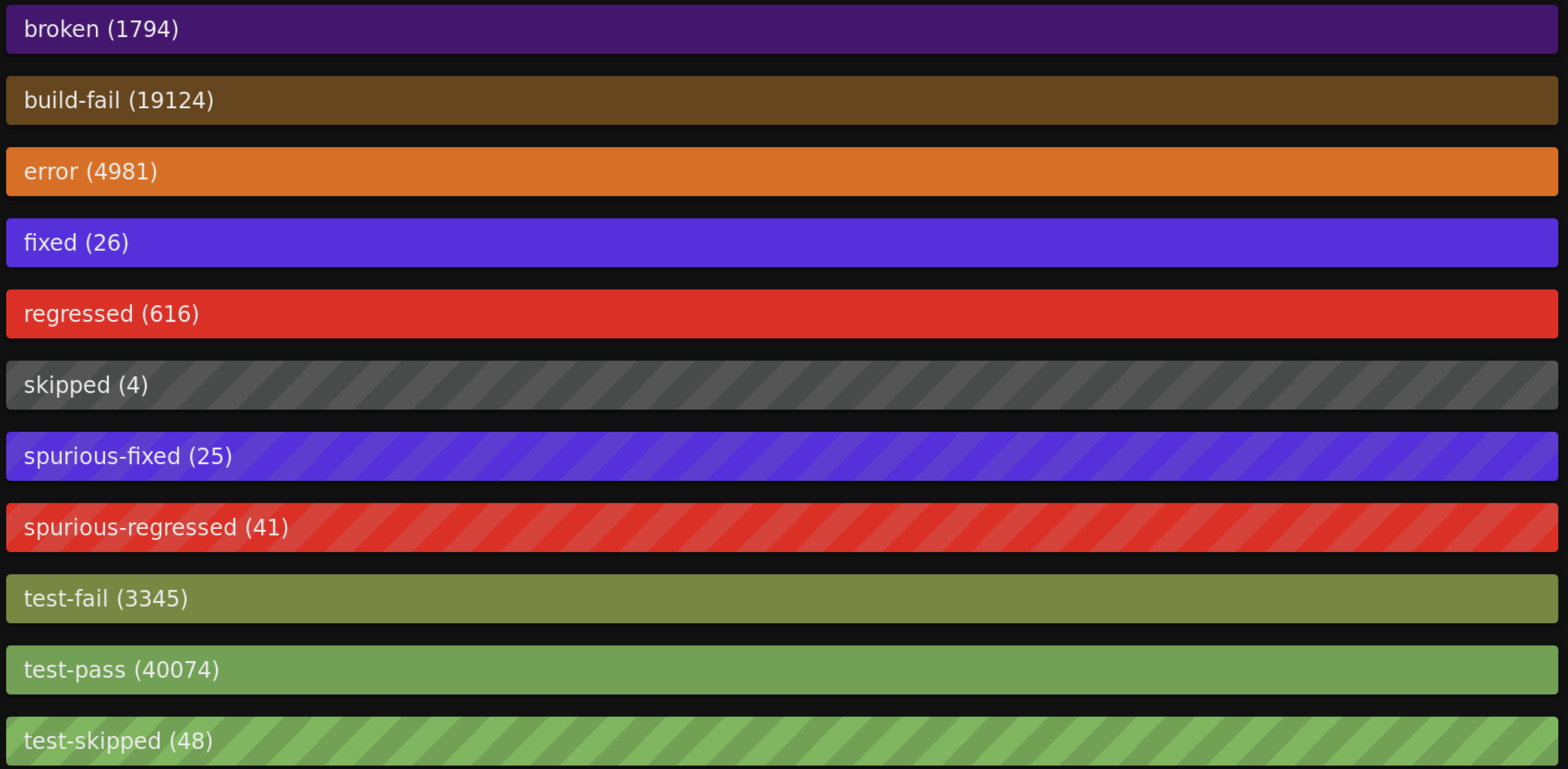
74234 crates tested

1.38.0 beta-2019-09-28



regressed (46)		
eraserhd.parinfer-rust.d9e7a2917ebfaccf93133b0009a4df135c38c19a	test passed	test failed
h3nnn4n.Strange-Attractor-Explorer.cac4cb37380f343bf20819f34dde9929c51098e7	test passed	test failed
hiroshiyui.libwonderarray.dd9b18d88105863b58027a94951ebfedf9af5133	test passed	build failed
mantal.expert_system.873622fa400d5bd721e592f22561e98860f69536	test passed	test failed
slazicoicr.bam_histogram_qc.76f1995e64ae232319aa05d407466f6ce0d927f0	test passed	build failed
slp.qsd.4538983da0f5c8dfca2e92ea01b41d2146db60cb	test passed	build failed
theaaf.decklink-rs.a9862c2764457485d3b5e858696986d25ff47519	test passed	test failed
adhesion-0.5.0	test passed	test failed
async-core-0.1.0	test passed	build failed
cmark-gfm-sys-0.29.0	test failed	build failed
cmark-gfm-0.1.1	test passed	build failed
croaring-sys-0.4.1	test passed	build failed
croaring-0.4.1	test passed	build failed
derive_less-0.2.0	test passed	build failed
duktape_ffi_raw-2.30.0	test passed	build failed

1.37.0 beta-2019-08-13



Crater is not perfect…

Crater is not perfect…

…today it works great though! 🎉

Let's recap!

Fast release cycles allow us not to worry about deadlines.

Crater is the tool allowing us to do that without breaking the world.

Thanks!