

Forum:[Hack Forums](#)

Thread: [Has anyone tried using Rust for Backend Development?](#)

happygonutty - May 7, 2024, 11:50 am

Title says it all. I am just curious if someone of you have already did a project (not an example or an exercise) that involves backend development using frameworks like Warp, Actix or maybe Rocket in Rust programming language? Do you guys mind sharing your experience with it and what are your comments with the framework and the language itself? ?

MavenGradle - May 7, 2024, 11:54 am

Yes, of course. I have some Rust backend projects in production today. Some heavy duty calculation stuff and honestly? It's been great! I mainly use Actix and Actix-web, because it's stable. At this point in time I cannot and will not utilize Axum for production simply because its still in development and open to big changes, which I try to save my clients from. The problem with Rocket is that a) Actix exists and is way more powerful than Rocket and b) Rocket is mostly maintained by a single developer in his spare time. Once again, very hard to convince clients to choose Rocket over Actix. I know the noobs like to recommend Axum because it's being developed by the same guys that are maintaining Tokio, but there are REAL reasons why you would use Actix over Axum even though they are both using Tokio. One of those reasons being stability and another being breaking changes. You should not use Axum for clients at this point in time unless they are fully aware of what you're doing and they're open and willing to rewrite big portions of their code, possibly 6 months from now or less. Using Axum for your own hobby projects is absolutely fine though.

But, in terms of why I would use Rust... We've noticed a reduced cost for our cloud deployment. In one case I was able to reduce memory usage from an old Java Spring application down from 250MB RAM usage to 10MB, and keep it stable, while being able to calculate more requests and handle them faster. At some points in the mornings it handles over 100.000 requests in less than an hour and still does not exceed 10MB of RAM usage. Couldn't be happier with it.

happygonutty - May 7, 2024, 11:58 am

(May 7, 2024 11:54 AM)MavenGradle Wrote: [\[CITE\]](#) Yes, of course. I have some Rust backend projects in production today. Some heavy duty calculation stuff and honestly? It's been great! I mainly use Actix and Actix-web, because it's stable. At this point in time I cannot and will not utilize Axum for production simply because its still in development and open to big changes, which I try to save my clients from.

But, in terms of why I would use Rust... We've noticed a reduced cost for our cloud deployment. In one case I was able to reduce memory usage from an old Java Spring application down from 250MB RAM usage to 10MB, and keep it stable, while being able to calculate more requests and handle them faster.

Couldn't be happier with it.

We have the same opinion with both of those frameworks that you have mentioned (Actix and Actix Web). I primarily use it for cheap ass clients who want heavy duty applications running on machines (offline server or VPS) with minimal specifications. And also, I do not know why but whenever I dive into chat rooms or maybe other forums, they always say that Axum is much more stable than the two. But like you, I can see that it also lacks of many features and needs more room for improvement.

Have you tried using docker containerization with it, then having them all pass through an nginx gateway? If you have an alternative for nginx, do you mind sharing? Please do not say Apache. Lol. ?

MavenGradle - May 7, 2024, 12:09 pm

(May 7, 2024 11:58 AM)happygonutty Wrote: [\[CITE\]](#) Have you tried using docker containerization with it, then having them all pass through an nginx gateway? If you have an alternative for nginx, do you mind sharing? Please do not say Apache. Lol. ?

Yes! Setting up CI/CD for Rust projects is an absolute must. It makes your life easier, it makes the client's lives easier. Especially in this day and age where everything is a microservice and everything is generally speaking horizontally scalable. I agree, it's sad how we have the power of Rust but the limiting factor of nginx or Traefik.. but Actix web is still faster than any python framework or any Java framework on the market today, so really if nginx is your bottleneck, that means you're in the land of serious cashflow. In addition nginx is written in C++ and its mostly fine. You're probably not going to run into its limits and if you do, you will need to step into the land of hardware load balancers (or first something like Cloudflare load balancing - DNS based load balancing). It costs a pretty dime though, and chances are that you're not working for Netflix or Amazon and will rarely need to set up infrastructure like this. But now you know what to do if you ever did ?.

<https://www.cloudflare.com/application-services/products/load-balancing/> (goes to say the kind where they don't put the prices online and you have to contact sales to find out how much you're gonna be paying - e.g. if you have to ask you cannot afford it)

Another FYI is that nginx (plus) is fast enough to be a load balancer itself and I'm generally speaking talking outside the scope of DDoS attacks and how to deal with those because that is an entirely different beast to tackle with many different solutions.

happygonutty - May 8, 2024, 2:53 am

(May 7, 2024 12:09 PM)MavenGradle Wrote: [\[CITE\]](#) Yes! Setting up CI/CD for Rust projects is an absolute must. It makes your life easier, it makes the client's lives easier. Especially in this day and age where everything is a microservice and everything is generally speaking horizontally scalable. I agree, it's sad how we have the power of Rust but the limiting factor of nginx or Traefik.. but Actix web is still faster than any python framework or any Java framework on the market today, so really if nginx is your bottleneck, that means you're in the land of serious cashflow. In addition nginx is written in C++ and its mostly fine. You're probably not going to run into its limits and if you do, you will need to step into the land of hardware load balancers (or first something like Cloudflare load balancing - DNS based load balancing). It costs a pretty dime though, and chances are that you're not working for Netflix or Amazon and will rarely need to set up infrastructure like this. But now you know what to do if you ever did ?.

<https://www.cloudflare.com/application-services/products/load-balancing/> (goes to say the kind where they don't put the prices online and you have to contact sales to find out how much you're gonna be paying - e.g. if you have to ask you cannot afford it)

Another FYI is that nginx (plus) is fast enough to be a load balancer itself and I'm generally speaking talking outside the scope of DDoS attacks and how to deal with those because that is an entirely different beast to tackle with many different solutions.

Indeed especially if you work with a team or maybe the customer's team. Yes most systems nowadays are composed of microservices, most monolithic systems are either obsolete or just used in schools just to make a one hell of a system. Actually most schools are already teaching microservices and web services. Yes, I vouch Actix web for being the fastest framework but you know for simple developers who aren't that hardcore, kinda hard for them. In addition to that, you

are right, the only problems you would encounter is hardware and load balancers.

Less worries with technical problems but more worries with the pay-as-you-go kind of approach. Hahahaha. ?

COCKBOT - May 8, 2024, 6:31 am

Yes. I wrote a game matchmaking service using Rust/Axum.

I should have just used Go.

MavenGradle - May 8, 2024, 9:33 am

(May 8, 2024 06:31 AM)COCKBOT Wrote: [\[CITE\]](#) Yes. I wrote a game matchmaking service using Rust/Axum.

I should have just used Go.

Why in the world would you switch from Rust to Go? Again, if it's because you used a pre-release version of a product (AXUM)... It's your own damn fault for choosing a framework that doesn't even have a v1.0 release.

Dirty - May 8, 2024, 9:44 am

(May 8, 2024 09:33 AM)MavenGradle Wrote: [\[CITE\]](#) Why in the world would you switch from Rust to Go? Again, if it's because you used a pre-release version of a product (AXUM)... It's your own damn fault for choosing a framework that doesn't even have a v1.0 release.

I think Go has it's own merits enough to warrant the switch back from Rust to Go, as a fan of both languages and running a start-up with code running on customer infrastructure. We have plenty of both and I don't think I would want to replace our Go code with Rust code.

COCKBOT - May 8, 2024, 9:44 am

(May 8, 2024 09:33 AM)MavenGradle Wrote: [\[CITE\]](#) Why in the world would you switch from Rust to Go? Again, if it's because you used a pre-release version of a product (AXUM)... It's your own damn fault for choosing a framework that doesn't even have a v1.0 release.

Because it's the better tool for the job.

Time to develop is extremely important and the tradeoff in performance is incredibly negligible in my use-case.

Extremely large platforms (Instagram) use things like Django which is built with Python and perform perfectly fine, even with having to deal with the GIL.

Rust just... isn't worth the time spent for most projects.

MavenGradle - May 8, 2024, 9:52 am

(May 8, 2024 09:44 AM)Dirtybox Wrote: [\[CITE\]](#) I think Go has it's own merits enough to warrant the switch back from Rust to Go, as a fan of both languages and running a start-up with code running on customer infrastructure. We have plenty of both and I don't think I would want to replace our Go code with Rust code.

Why? And also would you do the opposite; replace your Rust code with Go? If so, why? I'd like to know the reasons. if it's because of a lack of Rust developers or financial reasons, please point them out for the sake of curiosity.

(May 8, 2024 09:44 AM)COCKBOT Wrote: [\[CITE\]](#) Because it's the better tool for the job.

Time to develop is extremely important and the tradeoff in performance is incredibly negligible in my

use-case.

Extremely large platforms (Instagram) use things like Django which is built with Python and perform perfectly fine, even with having to deal with the GIL.

Rust just... isn't worth the time spent for most projects.

Says... You? You're using a pre-release framework in Rust, and you're complaining about the development time which is entirely dependent on the programmer. This is a personal and highly subjective reason, not a technical one. Someone else might be more efficient writing Rust versus Go. Rust might not be worth it to you but then again some companies are using Rust to save on cloud infrastructure costs, which can easily run up to 100k a year for even a simple startup company. If Rust can decrease that cost to 40k, that's a lot of money saved. Yes, this has happened and will continue to happen. When you're claiming something is the better tool for the job, I'm here to point out to you and everyone who reads such statements as true, that this is extremely subjective given the circumstances, personal feelings, efficacy etc. Same goes for frameworks like Spring, React, Next, Node, etc and languages like Java, C#, JavaScript, Python etc..

COCKBOT - May 8, 2024, 10:00 am

(May 8, 2024 09:52 AM)MavenGradle Wrote: [\[CITE\]](#) Why? And also would you do the opposite; replace your Rust code with Go? If so, why? I'd like to know the reasons. if it's because of a lack of Rust developers or financial reasons, please point them out for the sake of curiosity.

Says... You? You're using a pre-release framework in Rust, and you're complaining about the development time which is entirely dependent on the programmer. This is a personal and highly subjective reason, not a technical one. Someone else might be more efficient writing Rust versus Go. Rust might not be worth it to you but then again some companies are using Rust to save on cloud infrastructure costs, which can easily run up to 100k a year for even a simple startup company. If Rust can decrease that cost to 40k, that's a lot of money saved. Yes, this has happened and will continue to happen. When you're claiming something is the better tool for the job, I'm here to point out to you and everyone who reads such statements as true, that this is extremely subjective given the circumstances, personal feelings, efficacy etc. Same goes for frameworks like Spring, React, Next, Node, etc and languages like Java, C#, JavaScript, Python etc..
Yes, says me. It's my opinion.

"Development time is entirely dependent on the programmer"

Rust, on average, takes much longer to ship than any of its contemporary counterparts.

MavenGradle - May 8, 2024, 11:00 am

(May 8, 2024 10:00 AM)COCKBOT Wrote: [\[CITE\]](#) Rust, on average, takes much longer to ship than any of its contemporary counterparts.

What are you basing this on? Do you have a source or links for studies for this statement to back this up? Specifically the average part? Rust and ASM are comparable, but it takes me less time to write something in Rust than it does in ASM.

Perhaps your next thought of train is that Rust is more verbose than other languages, but this statement fails to take into account that verbosity of a language does not equal longer development times or offer a causality between verbosity and development time. Again, Rust could be considered less verbose than ASM. Verbosity is not necessarily bad and it is not necessarily the cause of longer development times when comparing programming languages.

COCKBOT - May 8, 2024, 11:08 am

(May 8, 2024 11:00 AM)MavenGradle Wrote: [\[CITE\]](#) What are you basing this on? Do you have a source or links for studies for this statement to back this up? Rust and ASM are comparable, but it takes me less time to write something in Rust than it does in ASM.

Facts. It's a language that inherently asks more of you than any other. I understand the reasons behind why it does this, and I at times even appreciate it. But it doesn't change the fact that in many projects, you simply don't need the speed. I referenced Instagram's success with a notoriously slow framework built atop a notoriously slow language in an earlier post.

The fact of the matter is, users care more about features than they care about endpoint load times. Other languages are simply better fit for rapid development than Rust. It is not what Rust was built to do, and as such, it loses out in this regard.

MavenGradle - May 8, 2024, 11:12 am

(May 8, 2024 11:08 AM)COCKBOT Wrote: [\[CITE\]](#) Other languages are simply better fit for rapid development than Rust. It is not what Rust was built to do, and as such, it loses out in this regard.

(May 8, 2024 11:08 AM)COCKBOT Wrote: [\[CITE\]](#) The fact of the matter is, users care more about features than they care about endpoint load times.

I feel like you're not grasping what I'm saying when you make statements like this. This is subjective and not factual at all. It depends on many factors including the developer, the hardware on which something is being compiled, the Operating System, the developers experience, the developers tools like a Copilot or AI derivative, just to name yet another couple of examples as to why this is just.. not true.

In terms of what users want. That too depends on the users and the project that you're working on.. You cannot generalize what users want when you're not even giving a domain or use-case for these statements. Don't you think the users of Photoshop web care about the speed of the web-version of the Photoshop tool? Wouldn't that be exactly why Adobe created parts of the Photoshop application using Rust and WASM (and yes, there is a web version of Photoshop that you run in the browser)?

I've given counter examples to your statements time and time again but I can't keep beating the same horse over and over again.

COCKBOT - May 8, 2024, 11:21 am

(May 8, 2024 11:12 AM)MavenGradle Wrote: [\[CITE\]](#) I feel like you're not grasping what I'm saying when you make statements like this. This is subjective and not factual at all. It depends on many factors including the developer, the hardware on which something is being compiled, the Operating System, the developers experience, the developers tools like a Copilot or AI derivative, just to name yet another couple of examples as to why this is just.. not true.

In terms of what users want. That too depends on the users and the project that you're working on.. You cannot generalize what users want when you're not even giving a domain or use-case for these statements. Don't you think the users of Photoshop web care about the speed of the web-version of the Photoshop tool? Wouldn't that be exactly why Adobe created parts of the Photoshop application using Rust and WASM (and yes, there is a web version of Photoshop that you run in the browser)?

I've given counter examples to your statements time and time again but I can't keep beating the same horse over and over again.

You can generalize what users want. If you couldn't, focus groups wouldn't exist.

MavenGradle - May 8, 2024, 11:24 am

(May 8, 2024 11:21 AM)COCKBOT Wrote: [\[CITE\]](#) You can generalize what users want. If you couldn't, focus groups wouldn't exist.

Focus groups are called groups because they focus on something. Asking something as generic as 'what do you want?' is such a broad and unfathomable statement that it makes no sense at all to research the global topic of what users want.

Some users want world peace, some users want to sleep the entire day. Some users want to eat, shit, sleep and repeat. Some users want a slow and unresponsive Instagram. Some users want to be left alone. Some users want to be millionaires. Some users want a combination of all of the above...

Are you trolling?

COCKBOT - May 8, 2024, 11:39 am

(May 8, 2024 11:24 AM)MavenGradle Wrote: [\[CITE\]](#) Some users want world peace, some users want to sleep the entire day. Some users want to eat, shit, sleep and repeat. Some users want a slow and unresponsive Instagram. Some users want to be left alone. Some users want to be millionaires. Some users want a combination of all of the above...

And if you have a group of 100, and 60 of the 100 fall into one group, you then focus on bettering the user-experience for that group, because they are the majority in your group and likely the majority of your users.

I'm not trolling, I just fundamentally disagree with what you're saying. I've yet to say anything controversial in my posts.

happygonutty - May 8, 2024, 3:52 pm

How I wish I could've logged in 4 hours ago and brought some popcorn! Good job everyone! Hahahaha.

MavenGradle - May 8, 2024, 4:17 pm

(May 8, 2024 11:39 AM)COCKBOT Wrote: [\[CITE\]](#) And if you have a group of 100, and 60 of the 100 fall into one group, you then focus on bettering the user-experience for that group, because they are the majority in your group and likely the majority of your users.

But here you're not asking the global populace what they want. Here you're asking a subset of YOUR users of YOUR product what they want in YOUR product, yes? In the true form of a focus group related to YOUR product and not a focus group of the entire world (which makes NO SENSE) which is what you were saying earlier with a global statement like 'this is not what users want' without further context or explanation.

That's entirely different from broad and global statements to everyone (all users of the world, registered to your product or not)..

You see how this logic is not making any sense when you're making global statements like 'this is not what users want'? It is ENTIRELY linked to YOUR product and not something that can be extrapolated to every other service or product on the market, yes?

COCKBOT - May 8, 2024, 4:20 pm

(May 8, 2024 04:17 PM)MavenGradle Wrote: [\[CITE\]](#) But here you're not asking the global populace what they want. Here you're asking a subset of YOUR users of YOUR product what they want in YOUR product, yes?

That's entirely different from broad and global statements to everyone (all users of the world, registered to your product or not)..

You see how this logic is not making any sense when you're making global statements like 'this is not what users want'? It is ENTIRELY linked to YOUR product and not something that can be extrapolated to every other service or product on the market, yes?

When I'm building something for other people to use, typically I'm thinking only of their user-experience while I'm building it.

I'm making these statements because they're, in my experience, true. My posts in this thread are also very far from an unpopular stance, for what that's worth.

MavenGradle - May 8, 2024, 4:23 pm

(May 8, 2024 04:20 PM)COCKBOT Wrote: [\[CITE\]](#) When I'm building something for other people to use, typically I'm thinking only of their user-experience while I'm building it.

Thanks. But then why are you making such bold claims, like 'Rust is not what users want' in a global context without further explanation? It makes no sense.

YOUR users don't want it. That doesn't make it true for products like Photoshop. That doesn't make it true for products like PopOS, etc.

Which is what I've been trying to say. Your experience in this matter means very little when you're dealing with the global world and all the different tastes and flavours and products AND users that are out there, making what 'users want' linked to a product or service dependant on the users for one and the product or service.

Extrapolating the focus group of product a to product b, which is in an entirely different field, is generally speaking no indication of anything nor does it hold any marketable value. This is common sense and actually measurable. Yes?

In other words, asking a subset of users of Google what they want in a search engine when you're DuckDuckGo makes sense, yes, but asking Google users what they want in a search engine makes no sense when you're building a blackjack casino. Basing your product around such a focus group would not result in favorable outcomes, yes? You would market a 'blackjack' machine capable of finding the cheapest pregnancy tests when asked, but no way of playing the actual blackjack machine, making your users angry. Now you're in a situation where you have a product that is a blackjack machine that will not play Blackjack. You have the best search engine available in your gambling device, that no one is ever going to use to gamble or make you money, because they were looking for a device they could actually gamble on, and not a search engine.

happygonutty - May 8, 2024, 4:26 pm

You know guys, there are users who really don't effing care what tech we are using especially those who don't give a damn what a programming language is. Everyone is free to use what they want or whatsoever regardless if Rust is inherently superior over the other. I guess it is a matter of preference that is and just marketing.

MavenGradle - May 8, 2024, 4:49 pm

(May 8, 2024 04:26 PM)happygonutty Wrote: [\[CITE\]](#) You know guys, there are users who really don't effing care what tech we are using especially those who don't give a damn what a programming language is. Everyone is free to use what they want or whatsoever regardless if Rust is inherently superior over the other. I guess it is a matter of preference that is and just marketing.

Exactly.. You seem to understand the subject material. Your conclusion is spot on: SUBJECTIVITY. Same goes for Python, C#, Django or Node, etc.

Claiming something is the best is just total and complete nonsense without context. The other way around, claiming something is the worst, is equally nonsensical without context. You need context,

you need use-cases, you need data, to make such statements for specific scenarios.

To give an example that involves physics and the real (ew, real) world: you can easily claim that using aluminum casing is the best without further context. But what does that mean? Yes, using Aluminum casing for space ships has advantages, when you're looking at how it decreases fuel consumption when compared to using iron casing when launching from earth, because aluminum is less heavy. But does aluminum offer the best radiation protection in space? Is Aluminum casing the best at preventing death when you're comparing car crashes between cars having aluminum casing versus cars having steel casing? These are rhetorical questions.

I guess humans will be humans and continue to make nonsensical statements because of confirmation bias and other psychological phenomena. Maybe its pointless to point this all out and I'm writing all this having had no impact at all on the thinking of the global populace. Tomorrow someone will come along and claim that Zig is the best, Python is the worst, php is evil and the earth is flat. How great indeed, to be young (?) and / or clueless. Global opinions on what you should or should not use or do without context are logically speaking worthless pieces of clouded information including my own biased opinions for not wanting to rewrite Rust code to Go 'as a first thought' but in my own defence I ask for reasons why to understand them and grow as a person.

happygonutty - May 8, 2024, 11:02 pm

(May 8, 2024 04:49 PM)MavenGradle Wrote: [\[CITE\]](#) Exactly.. You seem to understand the subject material. Your conclusion is spot on: SUBJECTIVITY. Same goes for Python, C#, Django or Node, etc.

Claiming something is the best is just total and complete nonsense without context. The other way around, claiming something is the worst, is equally nonsensical without context. You need context, you need use-cases, you need data, to make such statements for specific scenarios.

During my younger years, I was stubborn as fuck just like this. I mean always comparing things, and ending up being stuck in an infinite loop where I could not finish a single thing because of believing that one thing being superior over the other and keep on trying to figure things out by reinventing the wheel over and over until I realize that damn, I have spent over weeks repeating each process.

It's a total waste of time just like what you said. The best solution for issues like this is just pick any language or framework you are best comfortable with then just don't stop and keep on moving forward unless you want to start over again and fuck everything up.

I have been in tight places because of this stupid OCD of having a perfect code to the point I almost lost a project until I realize my own mistakes and having self-control of being an over perfectionist. ?

Dirty - May 9, 2024, 10:08 am

(May 8, 2024 09:52 AM)MavenGradle Wrote: [\[CITE\]](#) Why? And also would you do the opposite; replace your Rust code with Go? If so, why? I'd like to know the reasons. if it's because of a lack of Rust developers or financial reasons, please point them out for the sake of curiosity.

Anyway finally found the time to respond to this, firstly this is all in the context of me working on my start-up so there is some considerations to consider here, which also is linked to both my and my co-founders current abilities and state of the market we find ourselves in in our region.

For our start-up at first we wrote our front-end and back-end in simple JavaScript, and this was mostly due to the fact that if you ignore quality, and performance. But optimizing for time to market, we used minimal abstractions and anything else (even useful abstractions). This allowed us to get

our product's MVP in front of many people who were interested in what we were building and getting critical feedback off of them.

Also my co-founder is a lot less well versed in Rust than I am, and this would have forced a lot more of the coding to fall on me as he would be significantly slower.

For us personally writing this product in Go or Rust would have taken a lot longer and maybe even put us out of cash to continue.

Also a reason we focus most of our time on Go is that I have personally found it easier to find developers with reasonable skill writing Go + TS/JS, than Rust + TS/JS.

Also most Rust developers we talked to had a problem with writing JavaScript all together (which I have a lot of problems with, but that could be an other discussion).

It mostly came down to a runway + available talent + personal abilities.

I'm personally (as of today) still a lot faster writing Go code than Rust code, mostly due to my personal skill. But since we are creating a performant critical piece of software (related to databases) we are writing it in Rust now, as we have gotten lots of feedback, proved we can do it, and proved the idea has merit.

Wizdominance - May 9, 2024, 10:28 am

IMHO, when Rust is being used in the arena, the only two downsides are its lack of libraries which Go has an advantage of since it is more widely used AND also the compilation time before you can completely use it.

For example, most microservices are containerized in a docker right? So, if I were to put a binary file written in Go, it would just take less than a second to compile it then copy it to the container correct? But if we were to do this in Rust, it would take minutes before we can achieve this.

This is just my personal point of view. I hope you all don't mind.

MavenGradle - May 9, 2024, 1:46 pm

(May 9, 2024 10:08 AM)Dirtybox Wrote: [\[CITE\]](#) -snip-

That's perfectly reasonable and understandable. Good luck with your journey and thanks for sharing!

(May 9, 2024 10:28 AM)Wizdominance Wrote: [\[CITE\]](#) IMHO, when Rust is being used in the arena, the only two downsides are its lack of libraries which Go has an advantage of since it is more widely used AND also the compilation time before you can completely use it.

For example, most microservices are containerized in a docker right? So, if I were to put a binary file written in Go, it would just take less than a second to compile it then copy it to the container correct? But if we were to do this in Rust, it would take minutes before we can achieve this.

This is just my personal point of view. I hope you all don't mind.

Relatable but on the other hand how often do you have to compile your applications and build and deploy the actual container yourself for all the different platforms? Usually there's some form of CI/CD step that does this for you, in some form or another, allowing you to completely get rid of such a bottleneck if applicable. When you're not doing this, you might indeed run into situations where you're forced to wait until you've compiled your code. That's why you usually only 'cargo build' when absolutely required and use code analysis tools like a good 'LSP checker' (such as rust-analyzer) or even a cargo check which will essentially compile the packages without performing the final step of code generation, which is must faster. You could add a linter like clippy to the mix which runs in the

background during your coding. Your IDE can take a part in this too. I'm not a shill for JetBrains but their RustRover IDE might be just what you're looking for. Along with having its own analysis and linting and suggestion tools, it integrates nicely with clippy.

Anyway, if you want, I can share a GitHub Actions workflow with you to set up or point you in the right direction for utilizing the CI/CD automation part. If you're against GitHub Actions, do keep in mind that a simple N100 SFF server (or even a raspberry pi 5 - but this is armv8) can run a self-hosted GitHub Actions runner for you which will cost you 0 runtime minutes on GitHub itself.

Code name: Test

permissions: read-all

on:

pull_request:

branches:

- master
- development

jobs:

test:

strategy:

matrix:

os: [ubuntu-latest, macos-latest, windows-latest]

runs-on: \${{ matrix.os }}

steps:

- name: Checkout
uses: actions/checkout@v4
- name: Install latest Rust toolchain
uses: dtolnay/rust-toolchain@stable
- name: Rust cache
uses: swatinem/rust-cache@v2
- name: Check code formatting
run: cargo fmt --all -- --check
- name: Check code
run: cargo check --workspace --verbose
- name: Test code
run: cargo test --verbose --all

To deploy to a docker registry (in this case just the docker.io registry) and/or GitHub releases, you could something like this:

Code name: Release

on:

push:

tags:

- 'v*'

permissions:
 contents: write

jobs:
 docker-publish:
 if: startsWith(github.ref, 'refs/tags/')
 runs-on: ubuntu-latest

steps:

- name: Checkout
 uses: actions/checkout@v4
- name: Set up Docker Buildx
 uses: docker/setup-buildx-action@v3
- name: Login to Docker Hub
 uses: docker/login-action@v3
 with:
 username: \${{ secrets.DOCKERHUB_USERNAME }}
 password: \${{ secrets.DOCKERHUB_TOKEN }}
- name: Publish
 uses: docker/build-push-action@v5
 with:
 context: .
 push: true
 tags: \${{ vars.DOCKERHUB_REPO }}:latest

gh-release:
 if: startsWith(github.ref, 'refs/tags/')
 strategy:
 matrix:
 os: [ubuntu-latest, macos-latest, windows-latest]
 runs-on: \${{ matrix.os }}

steps:

- name: Checkout
 uses: actions/checkout@v4
- name: Install latest Rust toolchain
 uses: dtolnay/rust-toolchain@stable
- name: Rust cache
 uses: swatinem/rust-cache@v2
- name: Build (release)
 run: cargo build --release
- name: Rename (macOS)
 if: \${{ matrix.os == 'macos-latest' }}
 run: mv target/release/your-service target/release/your-service-macos

- name: Rename (Linux)
if: \${{ matrix.os == 'ubuntu-latest' }}
run: mv target/release/your-service target/release/your-service-linux

- name: Publish
uses: softprops/action-gh-release@v2
with:
 prerelease: false
 files: |
 target/release/your-service-linux
 target/release/your-service.exe
 target/release/your-service-macos
env:
 GITHUB_TOKEN: \${{ github.token }}

The inclusion of the rust cache step will greatly reduce your runner minutes as well.