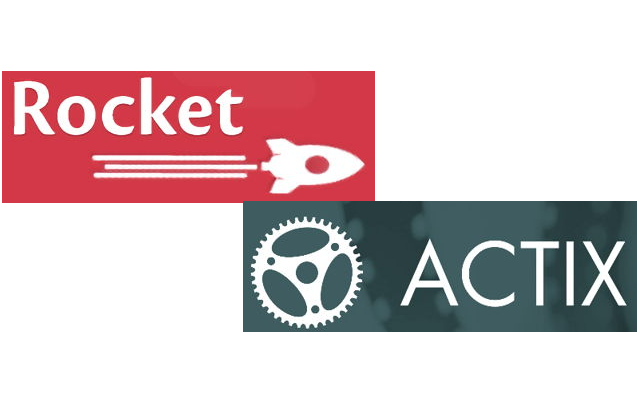
### **Actix or Rocket? Comparing Two Powerful Rust Web Frameworks**

****

### **Firstly, what are web frameworks?**

Before delving in to the comparison of two of Rust’s amazing frameworks, let us first clear our heads regarding web frameworks in general.

Before creating web applications like web services, web resources, and web APIs, we need a framework that would support the development of these applications. Frameworks are a combination of tools, helpers, and libraries that provide a way to efficiently build, test, and run applications. By using frameworks, we can have a standard set of rules for the creation and deployment of web apps. and automate the overhead associated with common activities performed in web development.

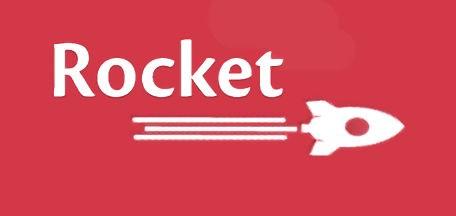
#### **Now, lets jump onto Rust web frameworks.**

Rust supports several frameworks for process of web development, which are as follows:

1. Rocket
2. Actix
3. Warp
4. Yew
5. Nickel
6. Gotham
7. Iron

All of these have their own set of pros and cons, however, Rocket and Actix have turned out to be two of the most popular ones due to their completeness. Let’s discuss the pros and cons of Rocket and Actix respectively:

### **Rocket**

****

Rocket is one of the most mature framework available for Rust. By using Rocket, you can write fast and secure web applications without compromising on speed, usability and flexibility.

More can be learnt about Rocket using its [documentation](https://rocket.rs/v0.4/guide/).

#### **PROS:**

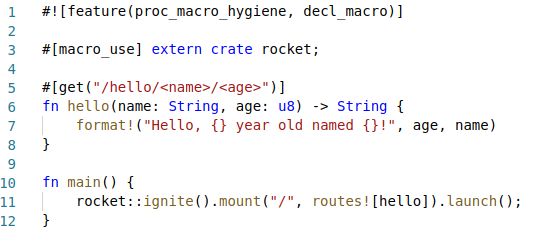
* Easy to use — Rust’s code generation tools are extensively used to provide a clean API
* Query Strings — Handling query strings and parameters is a breeze using Rocket
* Streams — Size isn’t a concern as Rocket streams all incoming and outgoing data
* Templating — Rocket has a built-in template support
* Extensible — you can create your own primitives easily, so that any Rocket app can use them
* Type Safe —It type checks route URLs, i.e it ensures that type errors are kept to a minimum
* Boilerplate Free — No need for a boiler plate code, a clean API can be easily provided using Rust’s code generation tools
* Testing Library — Using the built-in testing library, it runs unit tests on your applications with ease
* Config Environments — You can configure your application your own way for development, staging, and production
* Cookies — Hassle free viewing, adding, and removal of cookies, with or without encryption
* API Calls — Out of the box JSON support. By deriving Deserialize or Serialize you can receive or return JSON, respectively.
* Form Handling — Simplistic form handling through which bad form requests are filtered so your code doesn’t crash. By deriving FromForm for your structure, you can let Rocket know which parameter to use. It will then parse and validate the form request, create the structure, and then call your function

#### **CONS:**

* Nightly — Its one and only con is that it only works on Nightly version of Rust.

#### **Code Example:**

Below is a simple Hello, World! code to let you get started using Rocket:



If we go to<http://localhost:8000/hello/Sidra/26> you will receive the following output:

Hello, 26 year old named Sidra!

### **Actix**

****

Actix is a server-rendered framework, which means that the process of both building and serving an application will be from the server when a page is requested by the user. The architecture is based on [Rust’s very powerful actor system](https://docs.rs/axiom/0.2.1/axiom/#:~:text=Getting%20Started,actors%20for%20all%20processing%20activities.&text=An%20actor%20can%20be%20interacted,process%20a%20message%20only%20once.). It is well suited for writing services that have a logic with a higher level of difficulty.

To help you start out, it has a boiler plate code which could either help you get started quickly or would be somewhat overwhelming for building a simple application. It is a great framework with really good [documentation](https://actix.rs/docs/), approachable for even the most novice of developers.

#### **PROS:**

* Type safe —Just like Rocket, Actix provides type safety and ensures that type errors are minimal
* Blazingly Fast — Due to its powerful actor system, it works at amazingly fast speeds
* Feature Rich — Features like WebSockets, HTTP/2, pipelining, logging, etc. are provided out of the box.
* Extensible — you can create your own libraries that any Actix application can use

#### **CONS:**

It doesn’t have any con till date.

#### **Code Example:**

Below is a simple Hello, World! code to let you get started using Actix:



By going to<http://localhost:8000/> you will see the following output:

Hello World!

And by going to<http://localhost:8000/>name you will see:

Hello name!

### **Conclusion:**

Through this article, you may not have reached to a conclusion about which of these two Rust web frameworks is better, However, you now know about what these two frameworks aim to achieve and how they try to achieve it. Using Rocket or Actix is completely up to your own preferences and what kind of application you plan on building.