

# ***Server-side WebAssembly***

The Post-Container Revolution is Here!

Ramón Huidobro

@hola\_soy\_milk

# ***Server-side WebAssembly***

The Post-Container\* Revolution is Here!

Ramón Huidobro

@hola\_soy\_milk

# ***Server-side WebAssembly***

The Post-Container\* Revolution is Here\*!

Ramón Huidobro

@hola\_soy\_milk

**Containers?** 🙋

**WebAssembly?** 🙋

@hola\_soy\_milk



I'm Ramón.

(he/him)

Developer Advocate @ Suborbital

DevRel Consultant

Developer Educator

egghead instructor

Ruby, JS, Rust

Mozilla tech speaker

Tech Career Mentor

Live Streamer

***Let's learn all about  
WebAssembly and why its  
future on the server is so  
exciting!***

# ***A few questions...***

- What?
  - Why?
  - How?
  - When?
  - Where?
-



# What is WebAssembly (Wasm)?



WA



## WEBASSEMBLY

[Overview](#)[Getting Started](#)[Specs](#)[Future features](#)[Community](#)[FAQ](#)

WebAssembly 1.0 has shipped in 4 major browser engines.

[Learn more](#)

WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.

<https://webassembly.org/>

What?

@hola\_soy\_milk

**Wasm is not a programming  
language**

What?

@hola\_soy\_milk

**Wasm is not a front-end  
framework**

What?

@hola\_soy\_milk

**Wasm is standard for  
low-level bytecode**

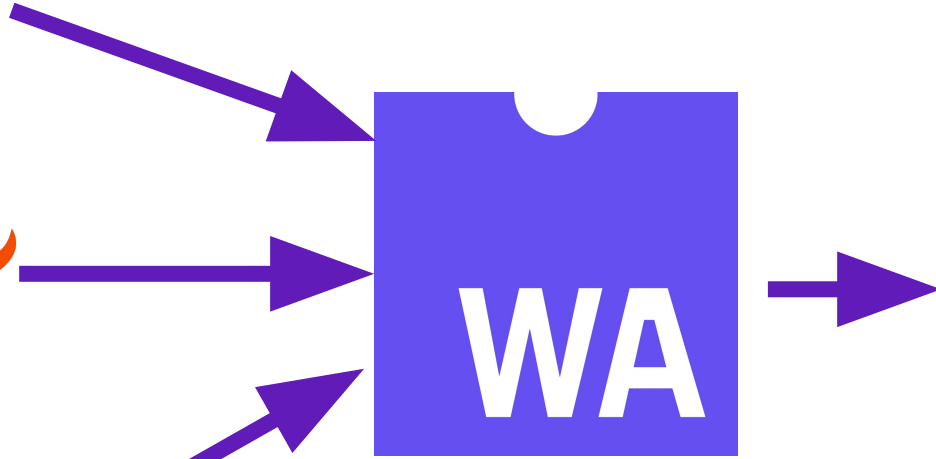
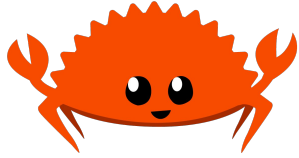
What?

@hola\_soy\_milk

***Wasm is a compilation target***

What?

@hola\_soy\_milk





# Prince of Persia

by Jul 29, 2014

Publication date 1990



Click here for the manual.

### Also For

Amiga, Amstrad CPC, Apple II, Atari ST, FM Towns, Game Boy, Game Gear, Game Boy Color, Genesis, iPad, iPhone, Macintosh, NES, Nintendo 3DS, PC-98, SAM Coupé, SEGA CD, Sharp X68000, SEGA Master System, SNES, TurboGrafx CD, Wii

Favorite Share Flag

2,035,543 Views

2,005 Favorites

50 Reviews

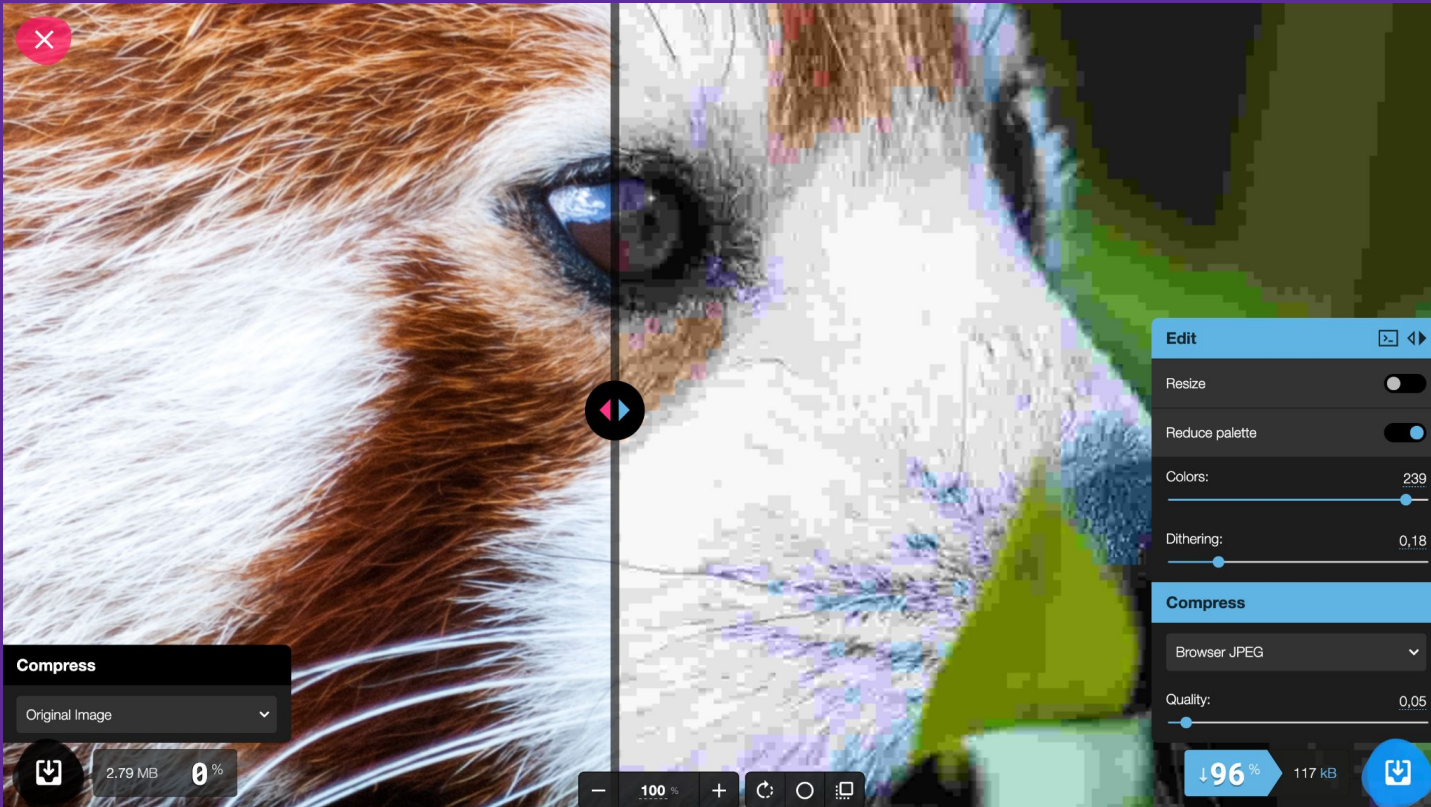
STREAM ONLY

[https://archive.org/details/msdos\\_Prince\\_of\\_Persia\\_1990](https://archive.org/details/msdos_Prince_of_Persia_1990)



What?

@hola\_soy\_milk



<https://squoosh.app/>

The image shows a code editor window for a Next.js project. The left sidebar displays the project structure with folders for 'pages' and 'styles', and files for 'package-lock.json', 'package.json', and 'README.md'. The main editor area shows the 'README.md' file with the following content:

```
1 This is a [Next.js](https://nextjs.org/) project bootstrapped with [create-next-app](https://github.com/vercel/next.js/tree/canary/packages/create-next-app).
2
3 ## Getting Started
4
5 First, run the development server:
6
7 ```bash
8 npm run dev
9 # or
10 yarn dev
11 ```
12
13 Open [http://localhost:3000](http://localhost:3000) with your browser to see the result.
14
15 You can start editing the page by modifying `pages/index.js`. The page auto-updates as you edit the file.
16
17 [API routes](https://nextjs.org/docs/api-routes/introduction) can be accessed on [http://localhost:3000/api/hello](http://localhost:3000/api/hello). This endpoint can be edited in `pages/api/hello.js`.
18
19 The `pages/api` directory is mapped to `/api/*`. Files in this directory are treated as [API routes](https://nextjs.org/docs/api-routes/introduction) instead of React pages.
20
21 ## Learn More
22
23 To learn more about Next.js, take a look at the following resources:
```

Below the README, a terminal window shows the following output:

```
~/projects/nextjs-pdofw
> npm install && npx next dev
warn preinstall No description field
warn preinstall No repository field
warn preinstall No license field
[1/4] * Resolving dependencies
└─ Loading ideal dependency tree
```

On the right side of the editor, there is a lightbulb icon and a progress indicator for 'Installing dependencies'. The progress bar shows three steps: 'Booting WebContainer' (checked), 'Installing dependencies' (checked), and 'Running start command' (unchecked).

The screenshot shows a web browser window at [lab.allotropia.de/wasm/](http://lab.allotropia.de/wasm/) displaying a LibreOffice Writer document. The document has two pages. The left page is titled "What is LibreOffice?" and contains introductory text and a table. The right page is titled "DrawShapes" and "Textframe" and contains diagrams illustrating drawing and text frame capabilities.

**What is LibreOffice?**

**Do more - easily, quickly**

LibreOffice is a powerful office suite; its clean interface and powerful tools let you unleash your creativity and grow your productivity. LibreOffice embeds several applications that make it the most powerful Free & Open Source Office suite on the market: Writer, the word processor, Calc, the spreadsheet application, Impress, the presentation engine, Draw, our drawing and flowcharting application, Base, our database and database frontend, and Math for editing mathematics.

**Finally, documents that look good**

Your documents will look professional and clean, regardless of their purpose: a letter, a master thesis, a brochure, financial reports, marketing presentations, technical drawings and diagrams.

**Use documents of all kinds**

LibreOffice is compatible with many document formats such as Microsoft® Word, Excel, PowerPoint and Publisher. But LibreOffice goes further by enabling you to use a modern open standard, the OpenDocument Format (ODF).

**Free as in Freedom, now and forever**

LibreOffice is Free and Open Source Software. Its development is open to new talent and new ideas. Our software is tested and used daily by a large and devoted user community; you, too, can [get involved](#) and influence its future development.

**Table**

A	B	C	D	E	F
I					
II					
III					
IV					

**DrawShapes**

Diagram illustrating drawing shapes. A blue rounded rectangle labeled "Text in Shape" is shown with a green arrow pointing to a pink star. A dimension line indicates a width of 36.83 mm. A green arrow points from the rectangle to the star, and a red arrow points from the star back to the rectangle.

**Textframe**

Diagram illustrating a text frame. A white rectangular frame contains the text: "He heard quiet steps behind him. That didn't bode well. Who could be following him this late at night and in this deadbeat part of town? And at this particular moment, just after he pulled off the big time and was making off with the greenbacks. Was there another crook who'd had the same idea, and was now watching him and waiting for a chance to grab the fruit of his labor? Or did the steps behind him mean that one of many law officers in town was on to him and just waiting to pounce and snap those cuffs on his wrists?"

**3DShape**

Diagram illustrating a 3D shape. A yellow 3D arrow pointing to the right is shown.

**Textbox**

Diagram illustrating a text box. A rounded rectangular box contains the text: "The quick brown Fox jumps over the lazy Dog".

***Ok but that's all on the  
browser side.***

***What's this about  
server-side Wasm?***

***Surely Wasm can't just  
run anywhere...***

What?

@hola\_soy\_milk

***That's where WASI comes in.***



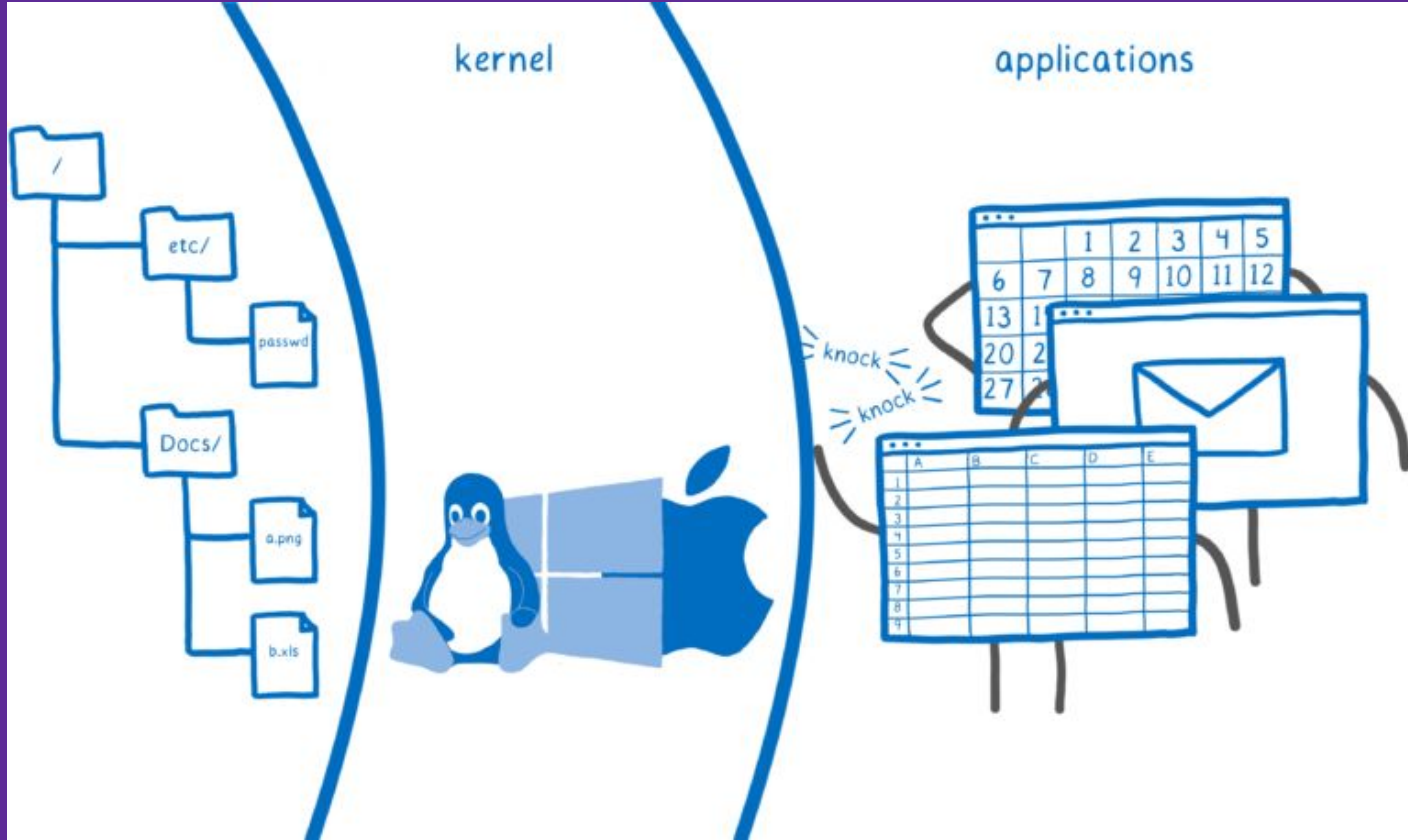
***That's where WASI comes in.***

WebAssembly,  
now with a  
System Interface!



What?

@hola\_soy\_milk



<https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/>

What?

@hola\_soy\_milk

***WASI is analogous to  
these system calls***



What?

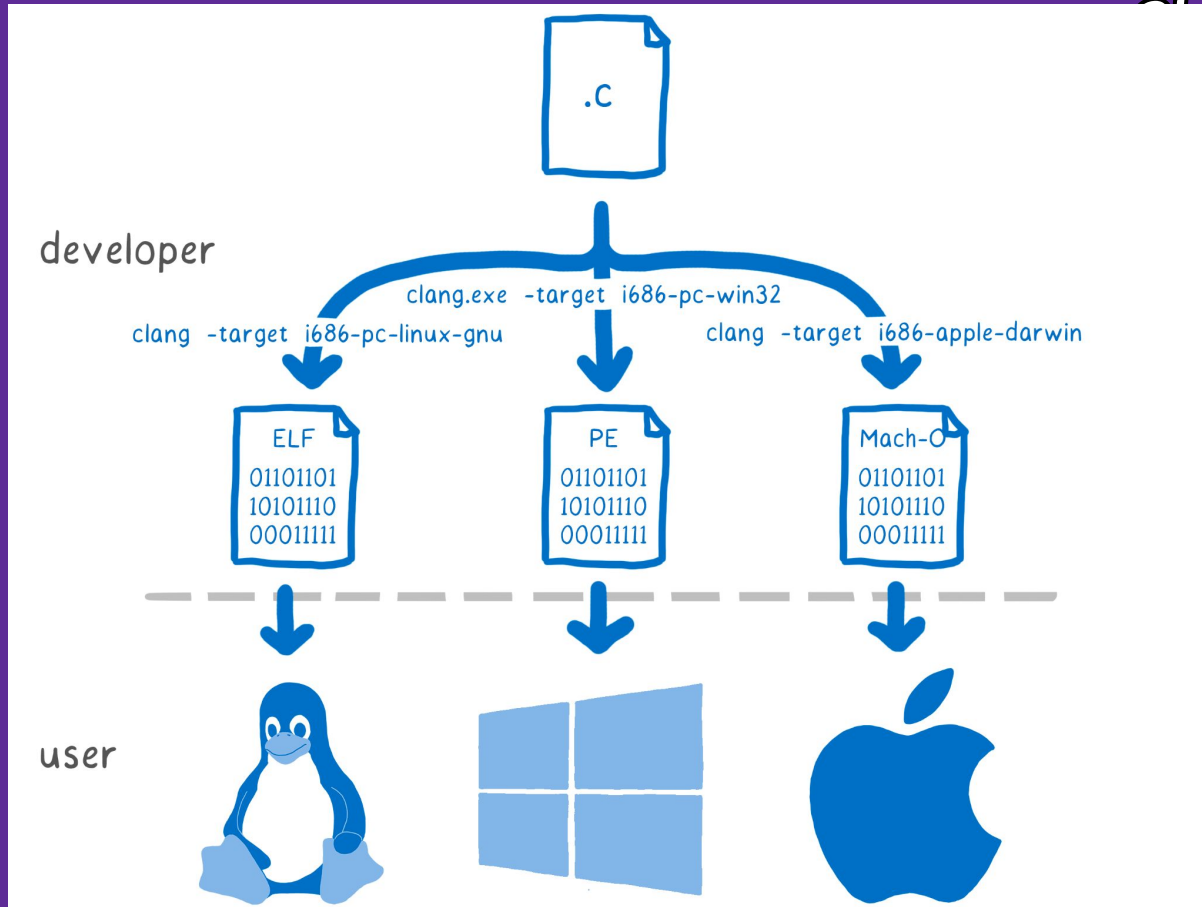
@hola\_soy\_milk

***It is not an OS  
replacement***

*It's an API designed by the Wasmtime project that provides access to several operating-system-like features, including files and filesystems, Berkeley sockets, clocks, and random numbers...*

What?

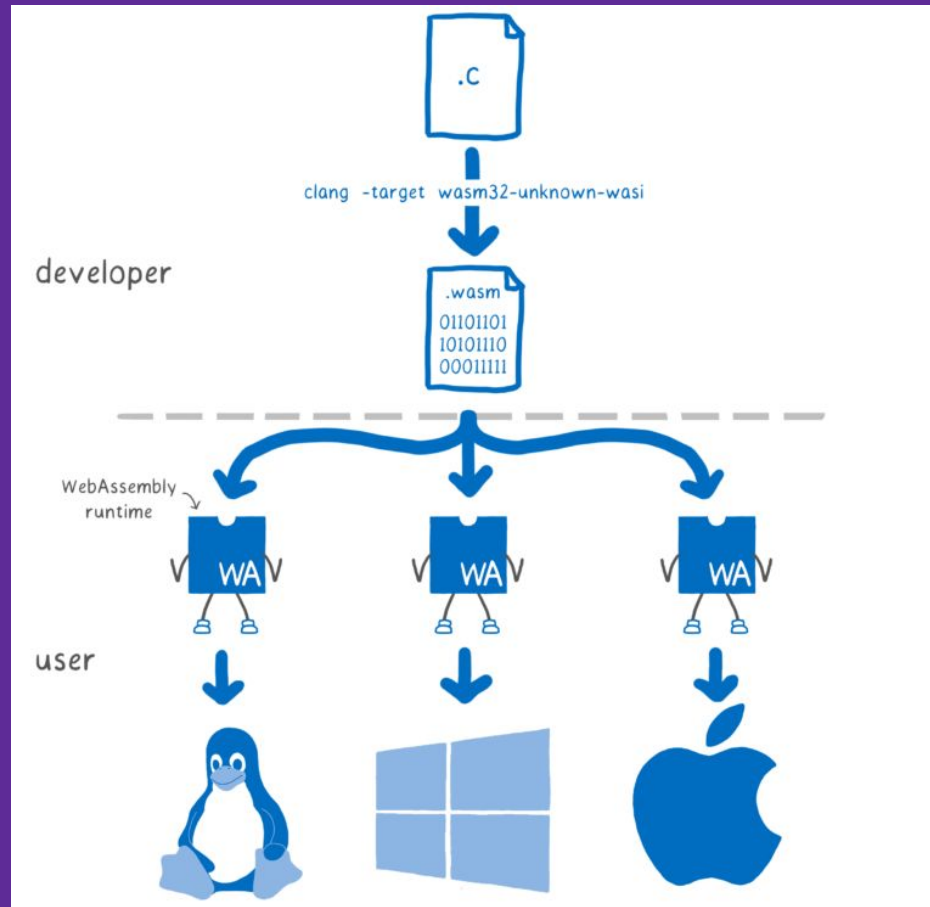
Stola\_soy\_milk



<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>

What?

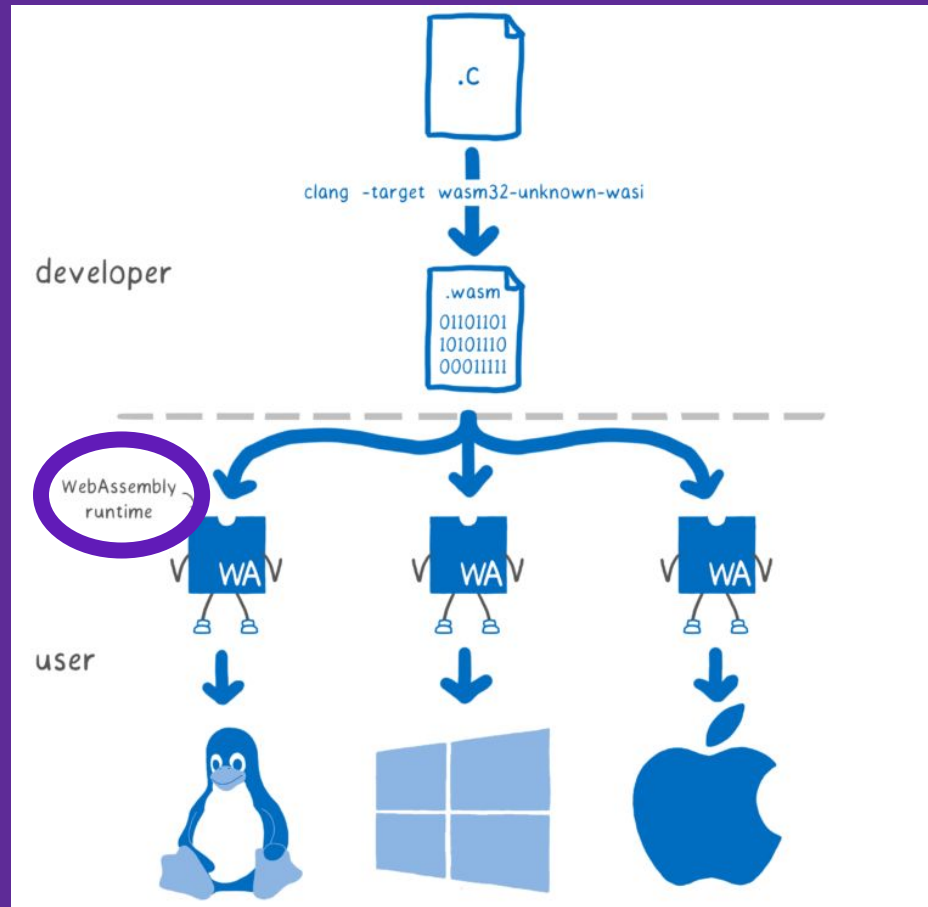
@hola\_soy\_milk



<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>

What?

@hola\_soy\_milk



<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>



## Introduction

Wasmtime is a [Bytecode Alliance](#) project that is a standalone wasm-only optimizing runtime for [WebAssembly](#) and [WASI](#). It runs WebAssembly code [outside of the Web](#), and can be used both as a command-line utility or as a library embedded in a larger application.

Wasmtime strives to be a highly configurable and embeddable runtime to run on any scale of application. Many features are still under development so if you have a question don't hesitate to [file an issue](#).

This guide is intended to serve a number of purposes and within you'll find:

- [How to create simple wasm modules](#)
- [How to use Wasmtime from a number of languages](#)
- [How to install and use the `wasmtime` CLI](#)
- Information about [stability](#) and [security](#) in Wasmtime.



... and more! The source for this guide [lives on GitHub](#) and contributions are welcome!

## Wasm3

**Wasm3** is the fastest WebAssembly interpreter, and the most universal runtime.

It's packaged into a `WebAssembly` package, so you can finally run `WebAssembly` on `WebAssembly` 😏



# wazero: the zero dependency WebAssembly runtime for Go developers

WebAssembly is a way to safely run code compiled in other languages. Runtimes execute WebAssembly Modules (Wasm), which are most often binaries with a `.wasm` extension.

wazero is the only zero dependency WebAssembly runtime written in Go.

<https://wazero.io/>



## wasmi - WebAssembly (Wasm) Interpreter

---

`wasmi` is an efficient WebAssembly interpreter with low-overhead and support for embedded environment such as WebAssembly itself.

At Parity we are using `wasmi` in [Substrate](#) as the execution engine for our WebAssembly based smart contracts. Furthermore we run `wasmi` within the Substrate runtime which is a WebAssembly environment itself and driven via [Wasmtime](#) at the time of this writing. As such `wasmi`'s implementation requires a high degree of correctness and Wasm specification conformance.

Since `wasmi` is relatively lightweight compared to other Wasm virtual machines such as Wasmtime it is also a decent option for initial prototyping.

Welcome to the Wasmer Documentation! 🙌

Wasmer is an open-source runtime for executing WebAssembly on the Server.



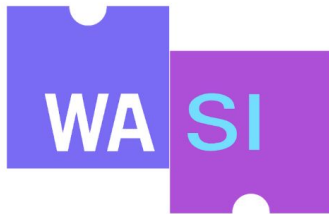
# Wasmer

Wasmer mission is make all software universally available

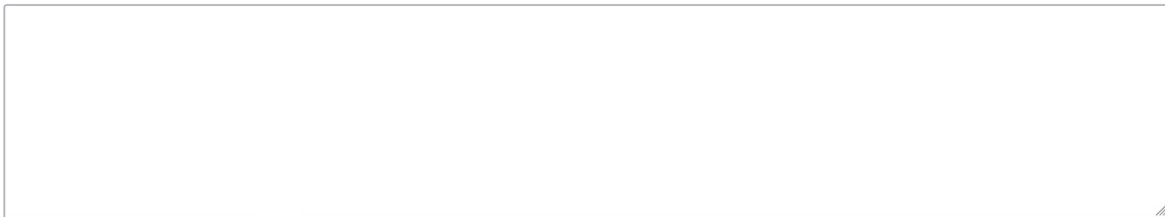
For an overview of WebAssembly, and what WebAssembly is, [take a look here](#).

i You can find the source code of the docs here: [github.com/wasmerio/docs.wasmer.io](https://github.com/wasmerio/docs.wasmer.io)

Any page can be easily edited, just by clicking on the **Edit on Github** link at the top right



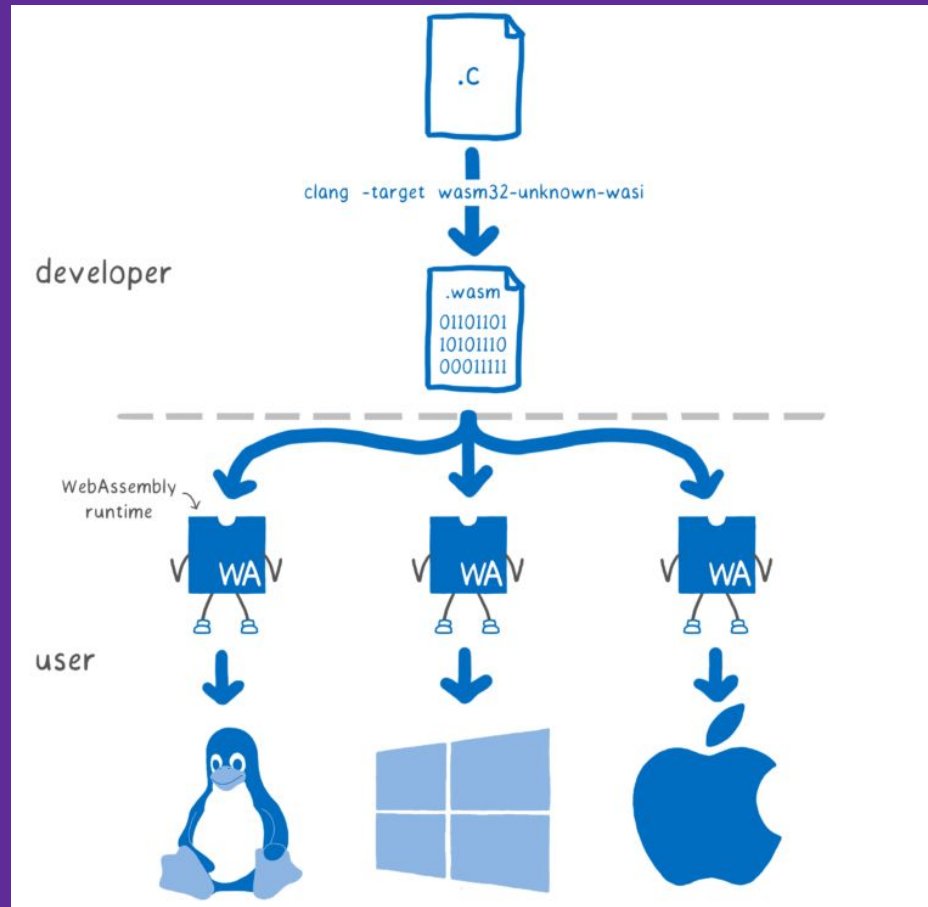
Examinar... Ningún archivo seleccionado.



This is a simple Web polyfill demo of [WASI](https://wasi.dev), a portable system interface for WebAssembly, allowing simple WASI programs that print to stout to be run in a browser. See [wasi.dev](https://wasi.dev) for more information on using WASI.

What?

@hola\_soy\_milk



<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>



What?

@hola\_soy\_milk

**Or even... in a  
container 🤯**

## Further Reading/Watching

- <https://developer.mozilla.org/en-US/docs/WebAssembly>
- <https://hacks.mozilla.org/2017/02/a-cartoon-intro-to-webassembly/>
- <https://github.com/bytecodealliance/wasmtime/blob/main/docs/WASI-overview.md>
- <https://hacks.mozilla.org/2019/03/standardizing-wasi-a-webassembly-system-interface/>

# ***A few questions...***

- What is Wasm?
  - Why?
  - How?
  - When?
  - Where?
-



What?

@hola\_soy\_milk

# Why run Wasm on the server?



WA

What?

@hola\_soy\_milk



Solomon Hykes

@solomonstre



If WASM+WASI existed in 2008, we wouldn't have needed to create Docker. That's how important it is. WebAssembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task!

[Traducir Tweet](#)



**Lin Clark**



@linclark · 27 mar. 2019

WebAssembly running outside the web has a huge future. And that future gets one giant leap closer today with...



Announcing WASI: A system interface for running WebAssembly outside the web (and inside it too)

[hacks.mozilla.org/2019/03/standa...](https://hacks.mozilla.org/2019/03/standa...)

[Mostrar este hilo](#)

9:39 p. m. · 27 mar. 2019 · Twitter Web Client

834 Retweets

164 Tweets citados

2.183 Me gusta

<https://twitter.com/solomonstre/status/1111004913222324225>

What?

@hola\_soy\_milk

***It's designed with  
capability-based  
security!***

What?

@hola\_soy\_milk

***It's polyglot by nature!***

What?

@hola\_soy\_milk

***Modules are typed,  
small, provisionable!***

What?

@hola\_soy\_milk

***It's got the speed!***

# First: why are folks putting wasm in production?

obvious reasons:

- Language-independence
- Open, formally-defined, portable standard
- Strong sandbox-based security



- Running the same CLI tools the same way
- Running the same containers the same way



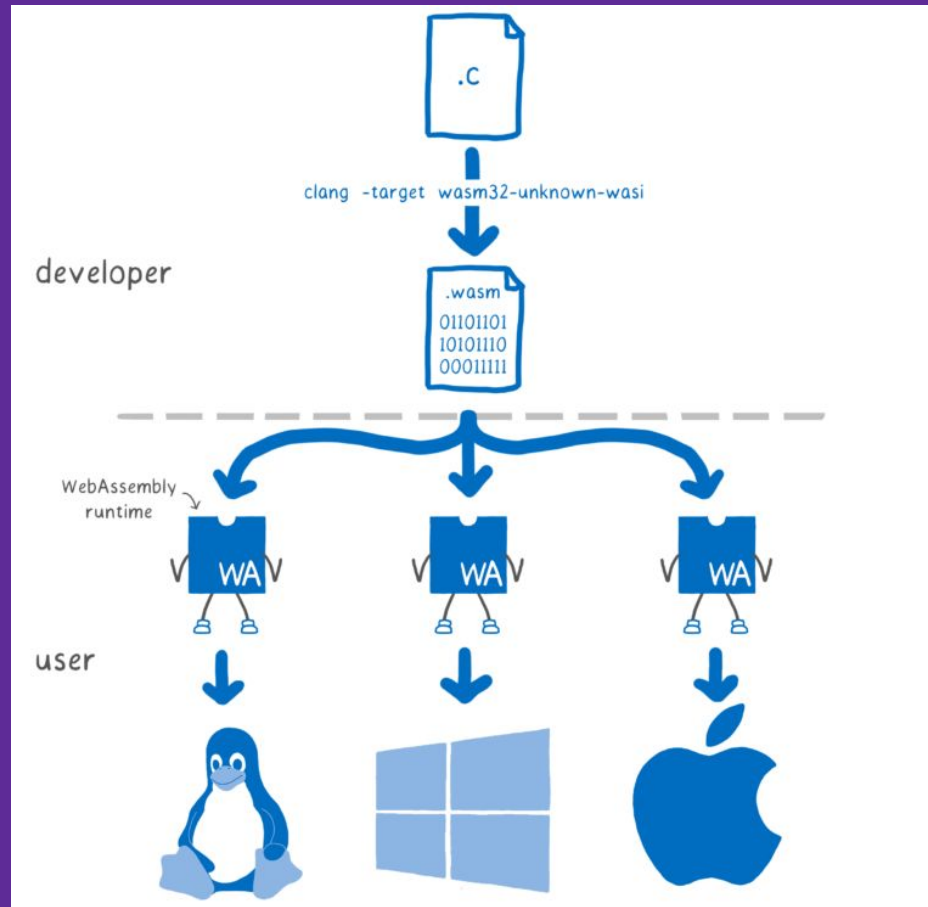
- Next step "smaller" in the progression
- "Serverless" execution: ultra-fast cold start, ephemeral



CLOUD NATIVE  
Wasm DAY  
NORTH AMERICA

What?

@hola\_soy\_milk

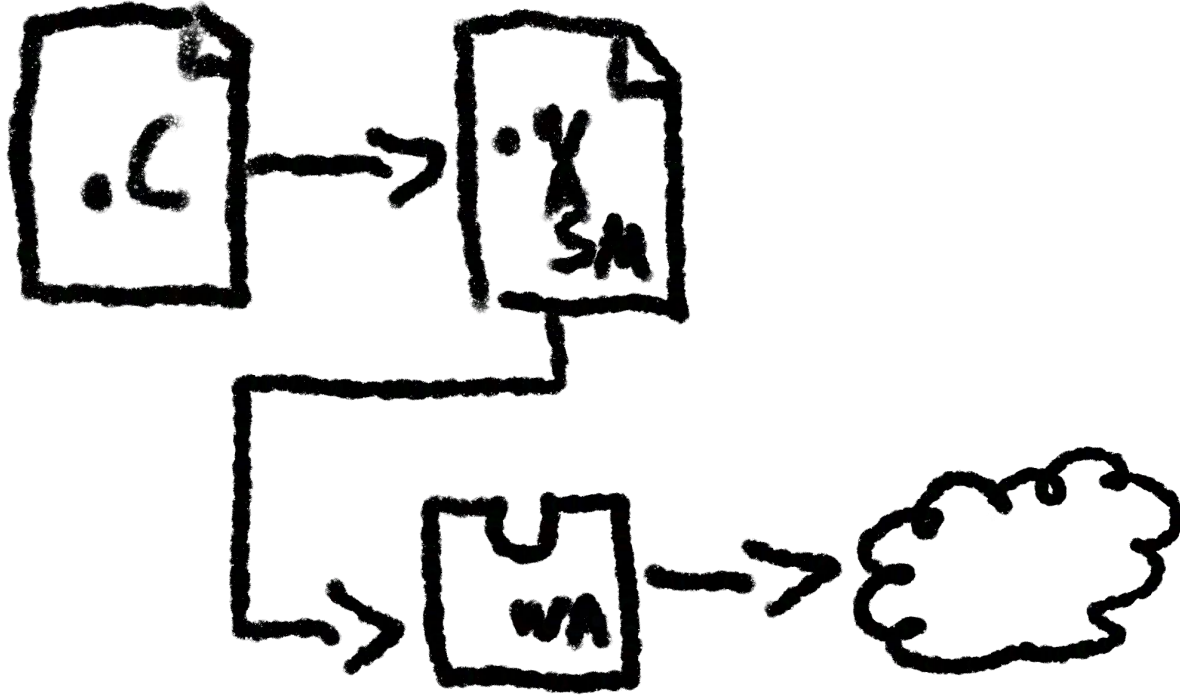


<https://hacks.mozilla.org/2017/02/creating-and-working-with-webassembly-modules/>



What?

@hola\_soy\_milk



***Code attaches to  
system resources at  
startup.***

***Code attaches to  
system resources at  
startup.***



What?

@hola\_soy\_milk

***But what does that mean for the industry?***

What?

@hola\_soy\_milk

***We can write code without  
worrying about***

**Server Setup**

***We can write code without  
worrying about***

***Startup/Shutdown times***

What?

@hola\_soy\_milk

***We can write code without  
worrying about***

**Scaling**

***We can write code without  
worrying about***

***Common security vectors***



## **Further Reading/Watching**

- <https://www.secondstate.io/articles/why-webassembly-server/>
- [https://wasmedge.org/book/en/use\\_cases/server\\_side\\_render.html](https://wasmedge.org/book/en/use_cases/server_side_render.html)
- <https://www.wasm.builders/thomastaylor312/why-webassembly-belongs-outside-the-browser-331a>

# ***A few questions...***

- What is Wasm?
  - Why run Wasm on the server?
  - How?
  - When?
  - Where?
-

**How is Wasm currently  
being used on the  
server?**

A blue square logo with a white outline, featuring the letters 'WA' in white. The logo is positioned in the bottom right corner of the slide.

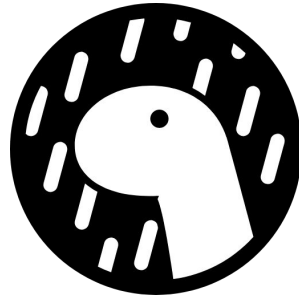
**WA**

How?

@hola\_soy\_milk

# Functions-as-a-Service (FaaS)

**fastly**



How?

@hola\_soy\_milk

# Edge Computing / Microservices



FERMYON



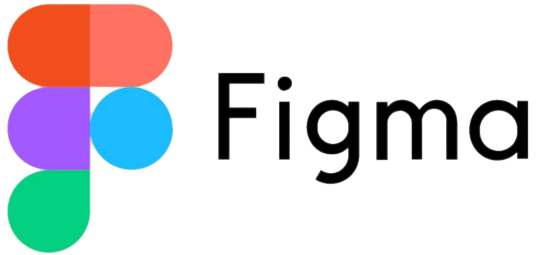
**CLOUD NATIVE**  
COMPUTING FOUNDATION



How?

@hola\_soy\_milk

# *Extensibility*



How?

@hola\_soy\_milk

# Blockchain



How?

@hola\_soy\_milk

# Embedded



Akri



## Further Reading/Watching

- <https://shopify.engineering/shopify-webassembly>
- <https://blog.suborbital.dev/webassembly-extensibility-today-and-tomorrow>
- <https://blog.suborbital.dev/webassembly-extensibility-today-and-tomorrow>

# **A few questions...**

- What is Wasm?
  - Why run Wasm on the server?
  - How is Wasm being run on the server?
  - When?
  - Where?
-

**When can we replace  
containers with  
Wasm?**



**WA**

When?



**Solomon Hykes**

@solomonstre

... @hola\_soy\_milk

“So will wasm replace Docker?” No, but imagine a future where Docker runs linux containers, windows containers and wasm containers side by side. Over time wasm might become the most popular container type. Docker will love them all equally, and run it all :)

[Traducir Tweet](#)



**Solomon Hykes** @solomonstre · 27 mar. 2019

If WASM+WASI existed in 2008, we wouldn't have needed to created Docker. That's how important it is. Webassembly on the server is the future of computing. A standardized system interface was the missing link. Let's hope WASI is up to the task! [twitter.com/linclark/statu...](https://twitter.com/linclark/status/111113329647325185)

[Mostrar este hilo](#)

4:50 a. m. · 28 mar. 2019 · Twitter Web App

56 Retweets   5 Tweets citados   165 Me gusta

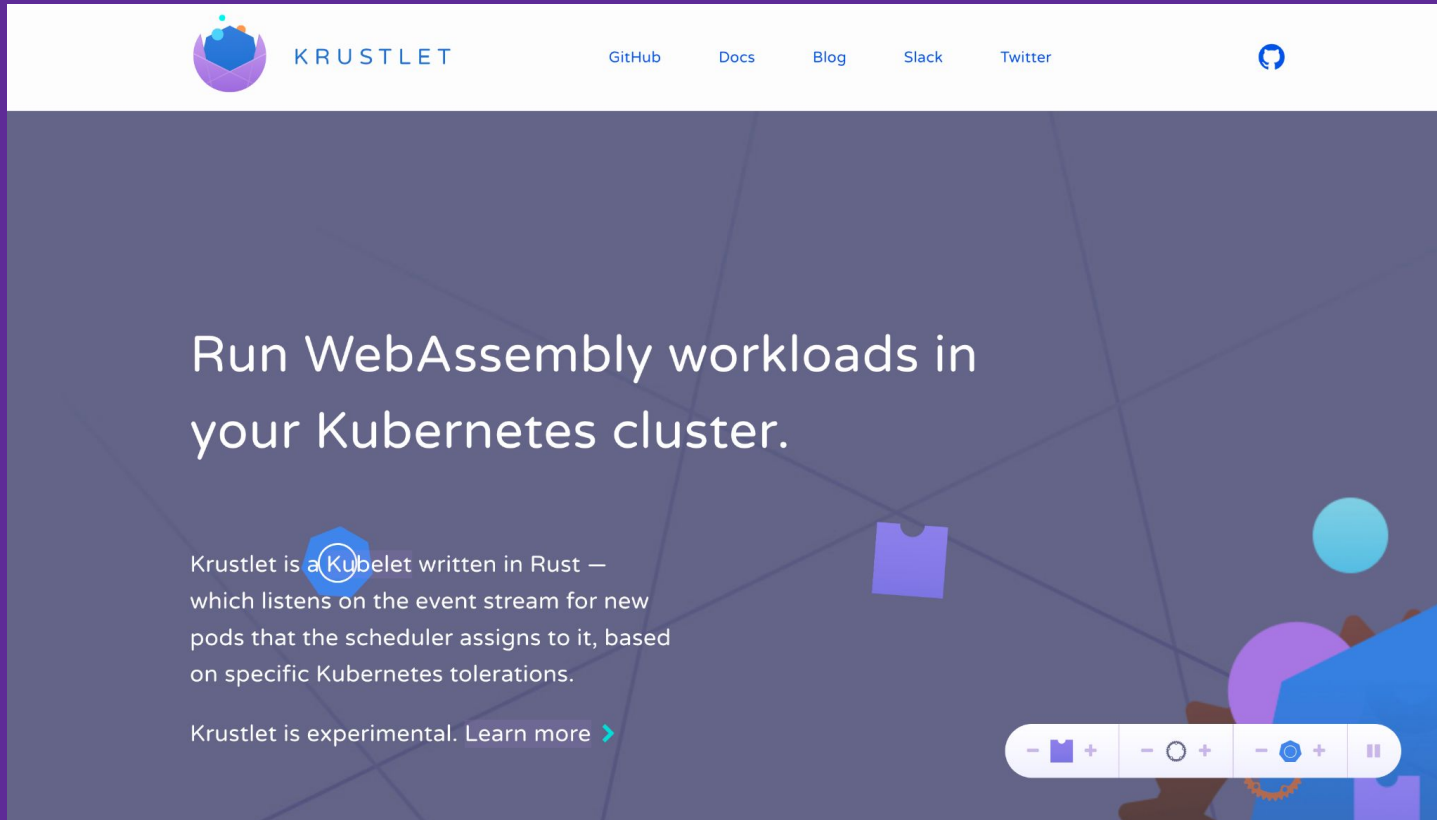
<https://twitter.com/solomonstre/status/111113329647325185>

# **Containers did not replace VMs wholesale...**

... They replaced VMs where VMs were used but not ideal

# ***Wasm will not replace containers wholesale...***

**... They'll replace containers where containers are  
used but not ideal**



The screenshot shows the homepage of the Krustlet project. At the top, there is a white navigation bar with the Krustlet logo on the left, which consists of a purple circle containing a blue globe with colorful dots. To the right of the logo is the word "KRUSTLET" in a bold, sans-serif font. Further right are links for "GitHub", "Docs", "Blog", "Slack", and "Twitter". On the far right of the navigation bar is a blue circular icon with a white speech bubble. The main content area has a dark grey background with a faint grid pattern. The primary headline is "Run WebAssembly workloads in your Kubernetes cluster." in a large, white, sans-serif font. Below this, a paragraph explains that Krustlet is a Kubelet written in Rust, which listens to the event stream for new pods. A link "Krustlet is experimental. Learn more >" is provided. At the bottom right of the page, there is a white video player control bar with icons for play, volume, and full screen.

KRUSTLET

GitHub Docs Blog Slack Twitter

# Run WebAssembly workloads in your Kubernetes cluster.


Krustlet is a Kubelet written in Rust — which listens on the event stream for new pods that the scheduler assigns to it, based on specific Kubernetes tolerations.

Krustlet is experimental. [Learn more >](#)

When?


@hola\_soy\_milk

NEW RELEASE **GRAIN V0.5 DURUM**—FASTER TO BUILD, FASTER TO RUN

 Grain

[Guide](#) [Documentation](#) [Community](#) [Blog](#) [Try](#)

[GitHub](#) [Twitter](#) [Discord](#)



## A modern web staple.

Grain is a new language that puts academic language features to work.

[LET'S GO](#)

<https://grain-lang.org/>



***But wait there's more:  
- Component model***

***But wait there's more:***

- Component model***
- wasi-nn***

***But wait there's more:***

- Component model***
- wasi-nn***
- Garbage Collection***

***But wait there's more:***

- Component model***
- wasi-nn***
- Garbage Collection***
- Multi-threading***

When?

@hola\_soy\_milk

**Who wants  
Docker+Wasm?**



docker docs Search the docs Home Guides Manuals Reference Samples Contribute

Home / Manuals / Docker Desktop / Wasm (Beta)

- Extensions SDK (Beta)
- Containerd Image Store (Beta)
- Wasm (Beta)**
- FAQs
- Give feedback
- Release notes
- Previous versions

Docker Engine

Docker Build

Docker Compose

Docker Hub

Docker subscription

Administration

Security

Contributor

Open-source projects

## Docker+Wasm (Beta)

Estimated reading time: 5 minutes

Wasm (short for WebAssembly) is a faster, lighter alternative to the Linux & Windows containers you're using in Docker today (with [some tradeoffs](#)).

This page provides information about the new ability to run Wasm applications alongside your Linux containers in Docker. To learn more about the launch and how the preview works, read [the launch blog post here](#).

**Beta**

The Docker+Wasm feature is currently in [Beta](#). We recommend that you do not use this feature in production environments as this feature may change or be removed from future releases.

### Enable the Docker+Wasm integration

The Docker+Wasm integration currently requires a technical preview build of Docker Desktop.

**Warning**

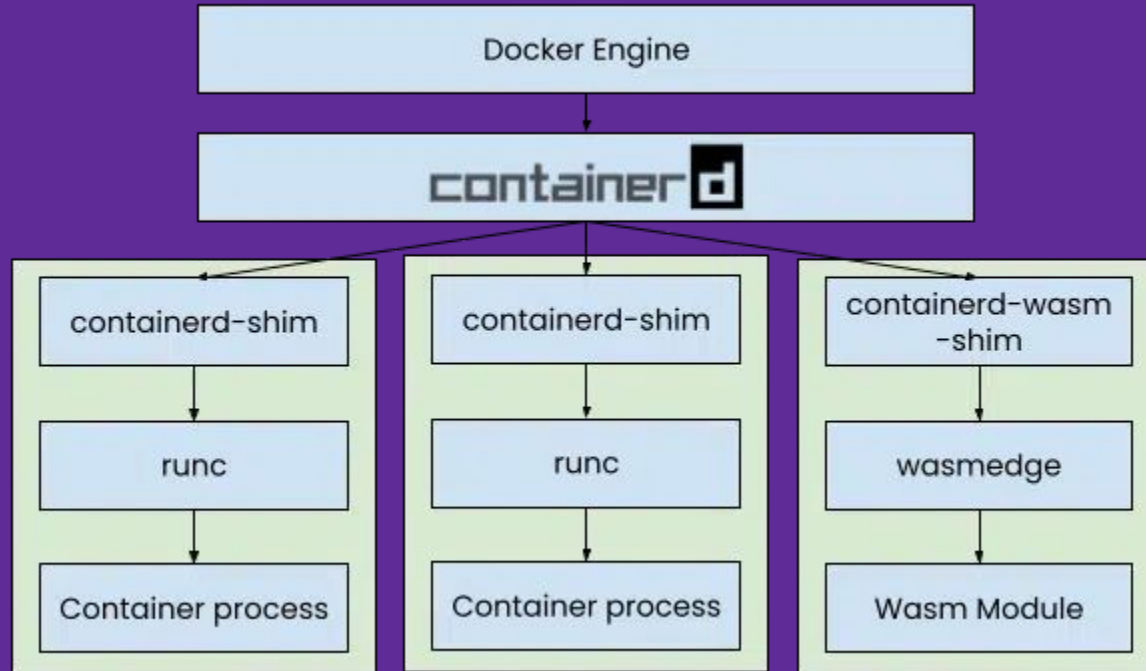
With the technical preview build of Docker Desktop, things might not work as expected. Be sure to back up your containers and images before proceeding.

Contents:

- Enable the Docker+Wasm integration
- Usage examples
  - Running a Wasm application with docker run
  - Running a Wasm application with Docker Compose
  - Running a multi-service application with Wasm
  - Building and pushing a Wasm module
- Docker+Wasm Release Notes
  - New
  - Known issues
- Feedback
  - Edit this page
  - Request changes

When?

@hola\_soy\_milk



<https://www.docker.com/blog/docker-wasm-technical-preview/>

## **Further Reading/Watching**

- <https://www.fermyon.com/blog/webassembly-vs-containers>
- <https://www.youtube.com/watch?v=phodPLY8zNE>



# **A few questions...**

- What is Wasm?
  - Why run Wasm on the server?
  - How is Wasm being run on the server?
  - When can we replace containers with Wasm?
  - Where?
-

**Where are people  
collaborating  
on/working on Wasm?**

**WA**

WEBASSEMBLY Search... Log in Create account

**Wasm Builders** is a community of 1,477 amazing builders  
Create account Log in

- Home
- Contact
- Events
- Learn
- Meet the admins
- Partners

**Other**

- Code of Conduct
- Privacy Policy
- Terms of Use

Popular Tags

Relevant Latest Top

**Welcome to Wasm Builders!**  
#welcome #introduction  
100 reactions 59 comments 3 min read

**WebAssembly @ KubeCon NA 2022**  
Scott Johnston CEO, Docker  
Kelsey Hightower Engineer, Google  
cosmonic vertex us

**WebAssembly with Kelsey Hightower & Docker CEO Scott Johnston**  
#opensource #cloudnative #docker  
Add Comment 1 min read

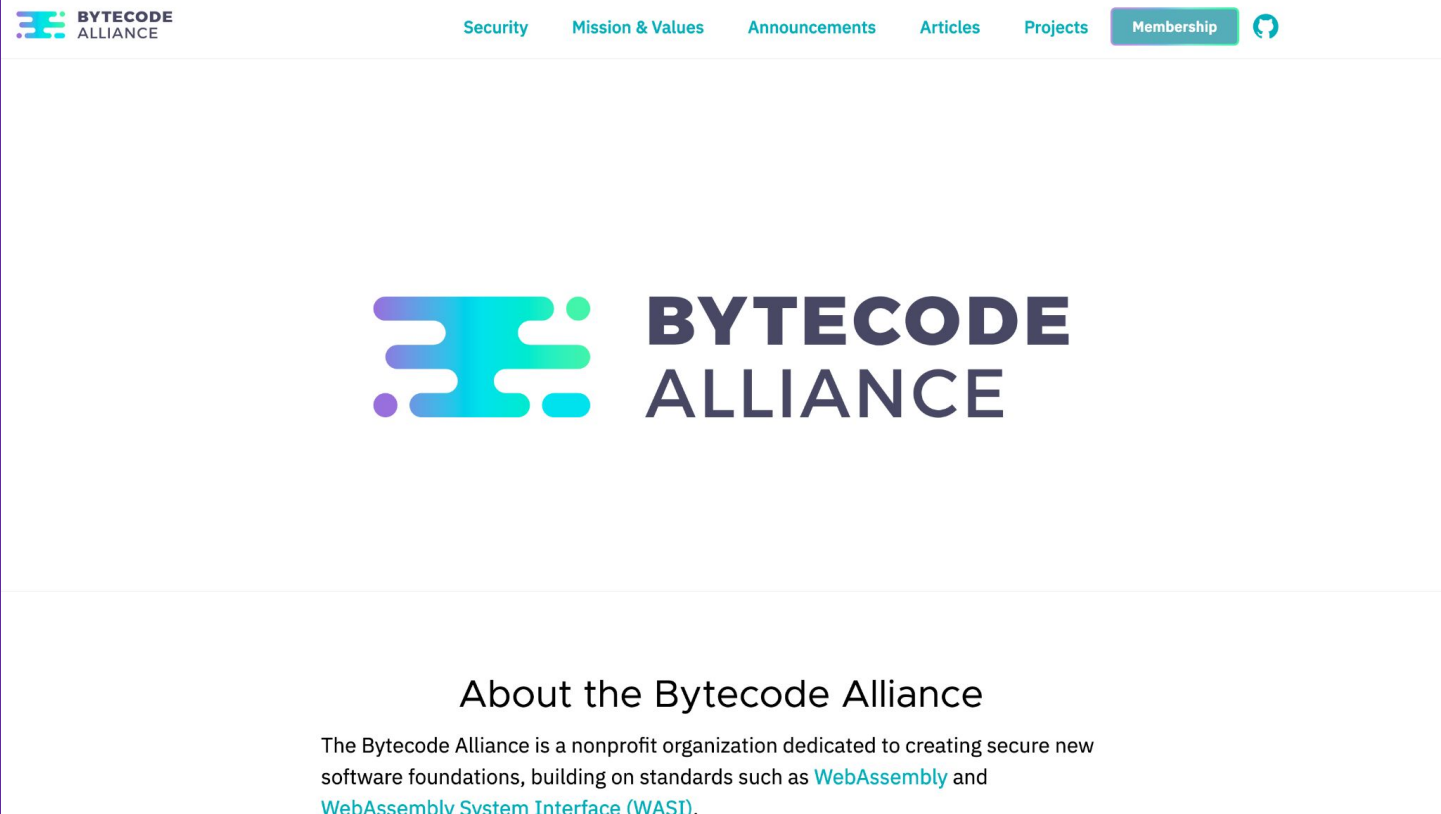
**#showcase**

- Bring your apps into MS Teams with the help of Uno Platform and WASM **New**
- Containerize React and WebAssembly (C++) with Docker **New**
- Serious Sam in the Browser 1 comment
- Running .Net where it has never run before RISC-V, FreeBSD and beyond... **New**
- Multi-Threaded, SIMD-Enabled, WASM Micro-Service Provider **New**

**#welcome**

- Welcome to Wasm Builders! 59 comments

**#browser**



<https://bytecodealliance.org/>

Members





The screenshot shows the GitHub profile for the WebAssembly organization. At the top, there is a navigation bar with the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The profile header includes the WebAssembly logo (a blue square with 'WA' in white), the name 'WebAssembly', a description 'Development of WebAssembly and associated infrastructure', and statistics: 752 followers, 'The Web!' location, and the website 'https://webassembly.org'. A 'Follow' button is on the right. Below the header are tabs for Overview (selected), Repositories (101), Projects (1), Packages, and People (29). The 'Pinned' section contains six repository cards: 'meetings' (Public, HTML, 360 stars, 117 forks), 'design' (Public, 11k stars, 711 forks), 'spec' (Public, WebAssembly, 2.7k stars, 417 forks), 'proposals' (Public, 752 stars, 44 forks), 'gc' (Public, WebAssembly, 598 stars, 47 forks), and 'WASI' (Public, Rust, 3.2k stars, 202 forks). On the right, the 'People' section shows a grid of 15 profile pictures and a 'View all' link. Below that is the 'Top languages' section with a legend for WebAssembly (blue), Rust (orange), C++ (red), HTML (red), and JavaScript (yellow). The 'Most used topics' section is partially visible at the bottom.

***So friends,  
we've learned...***

- That's what Wasm is!
  - That's why Wasm is great for the server!
  - That's how folks are using it in production!
  - That's when it'll get more widely adopted!
  - That's where folks are going to collab on it!
-



<https://ramonh.dev/server-side-wasm.pdf>

***Thanks a lot!*** 

Ramón Huidobro

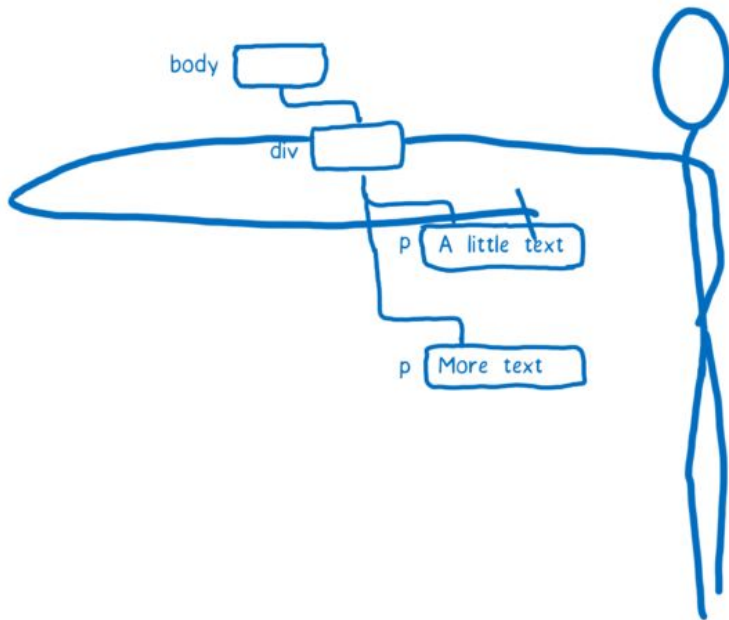
@hola\_soy\_milk

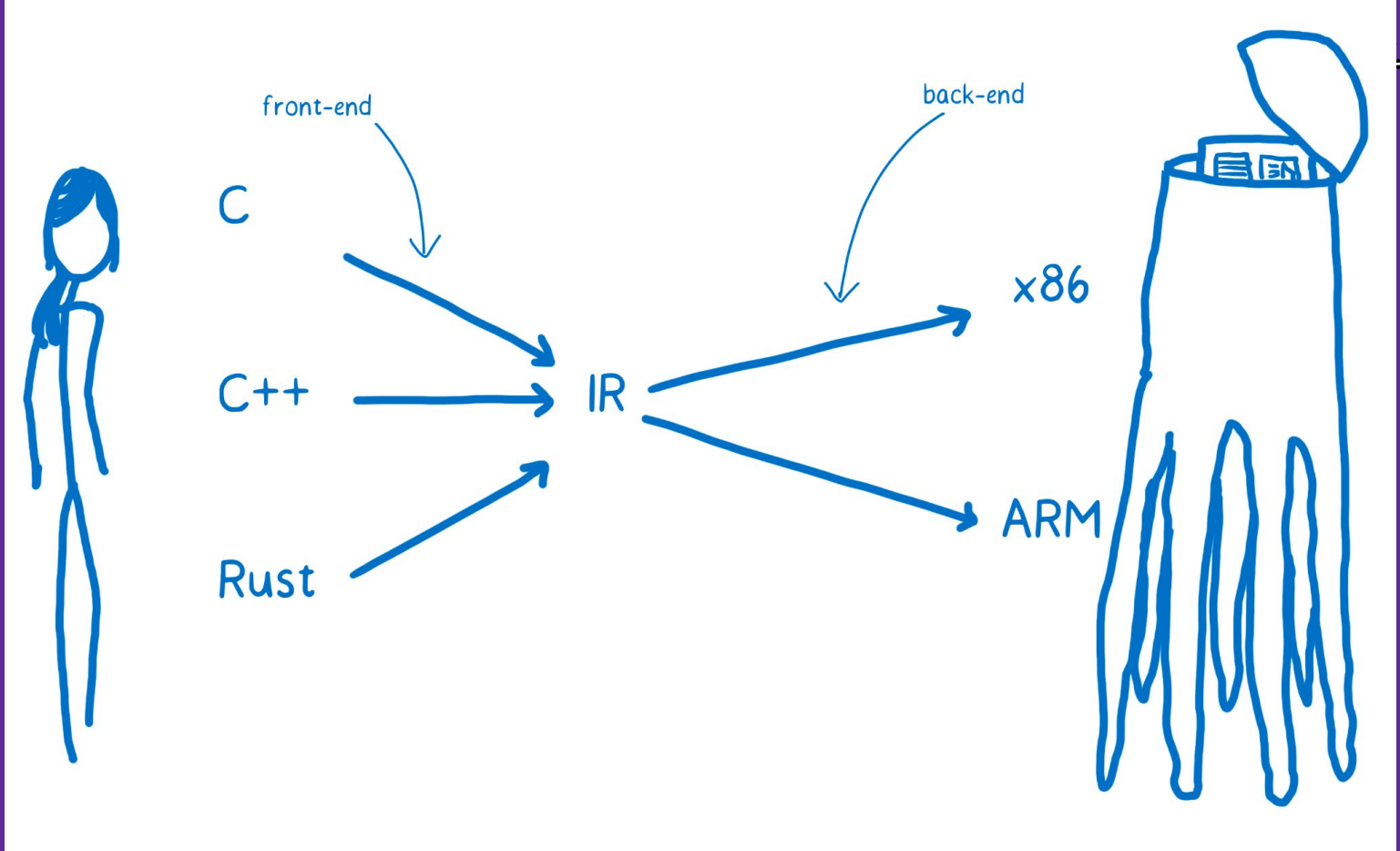
***Random slides in case I  
need them***

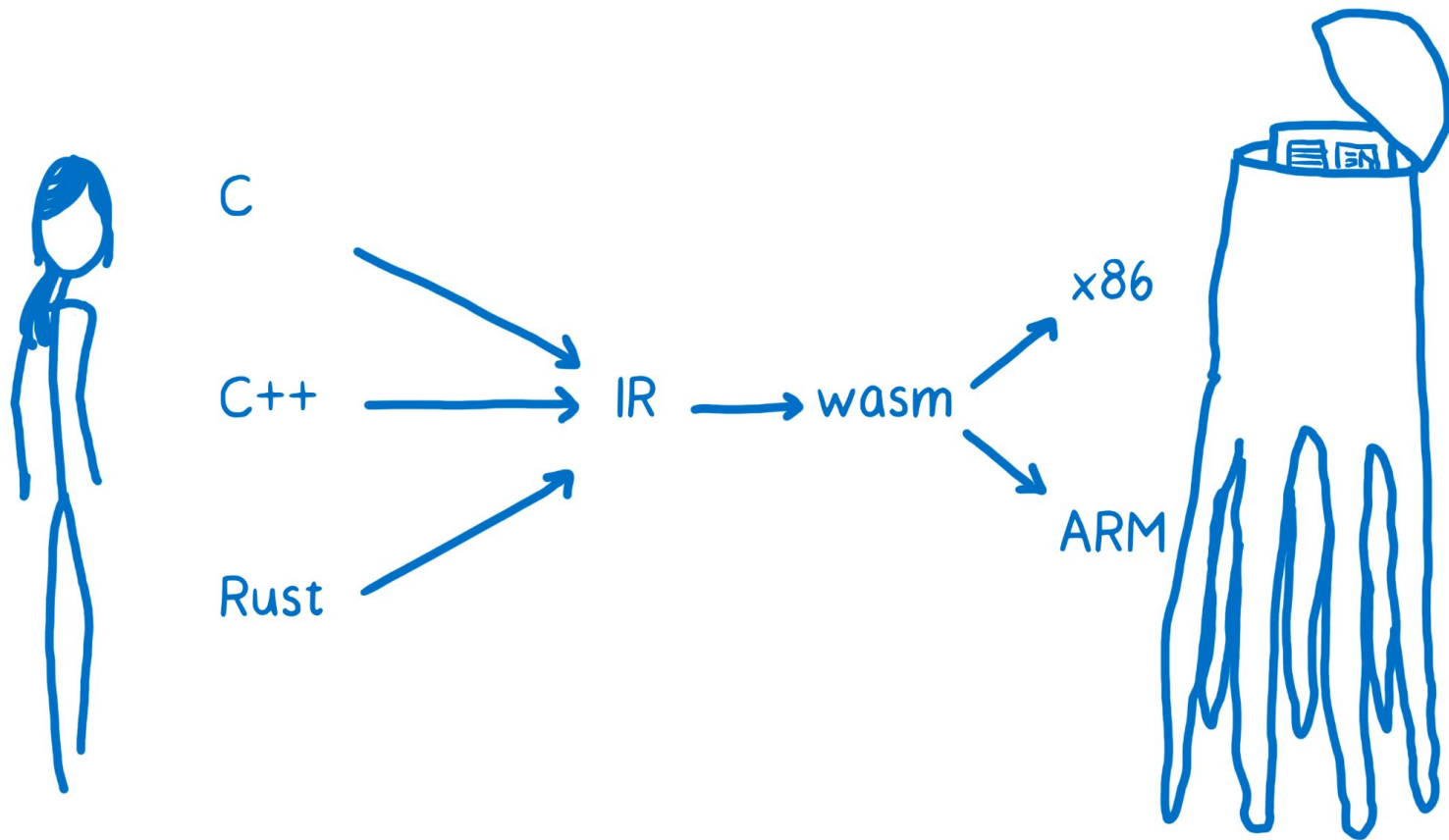
JS

DOM

WebAssembly







# Compiler toolchain

