



Ramón Huidobro
@hola_soy_milk

WebAssembly en el Servidor

¡Ha llegado la época post-contenedor!





ONLINE CONFERENCE
SPANISH EDITION

Ramón Huidobro
@hola_soy_milk

WebAssembly en el Servidor

¡Ha llegado la época post-contenedor*!





Ramón Huidobro
@hola_soy_milk

WebAssembly en el Servidor

¡Ha llegado* la época post-contenedor*!



¿Contenedores? 🙋

¿WebAssembly? 🙋

¿Yo? 🙋



Soy Ramón.

(él)

De Chile, en Austria

Developer Advocate @ Suborbital

Consultante DevRel

Instructor de Desarrollo

Mozilla tech speaker

Mentor de Carrera Tech

Live Streamer

Ponente de keynote

***¡A aprender qué es
WebAssembly y por qué su
futuro en el servidor es
emocionante!***

Unas preguntitas...

- ¿Qué?
 - ¿Por qué?
 - ¿Cómo?
 - ¿Cuándo?
 - ¿Dónde?
-

¿Qué es 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/>

¿Qué?

@hola_soy_milk

Wasm no es un lenguaje de programación

Wasm no es un framework de front-end

**Wasm es un formato de
código binario (bytecode)**

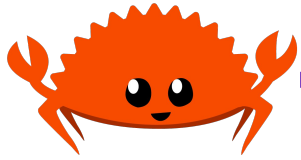
¿Qué?

@hola_soy_milk

**Wasm es un objeto de
compilador**

¿Qué?

@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

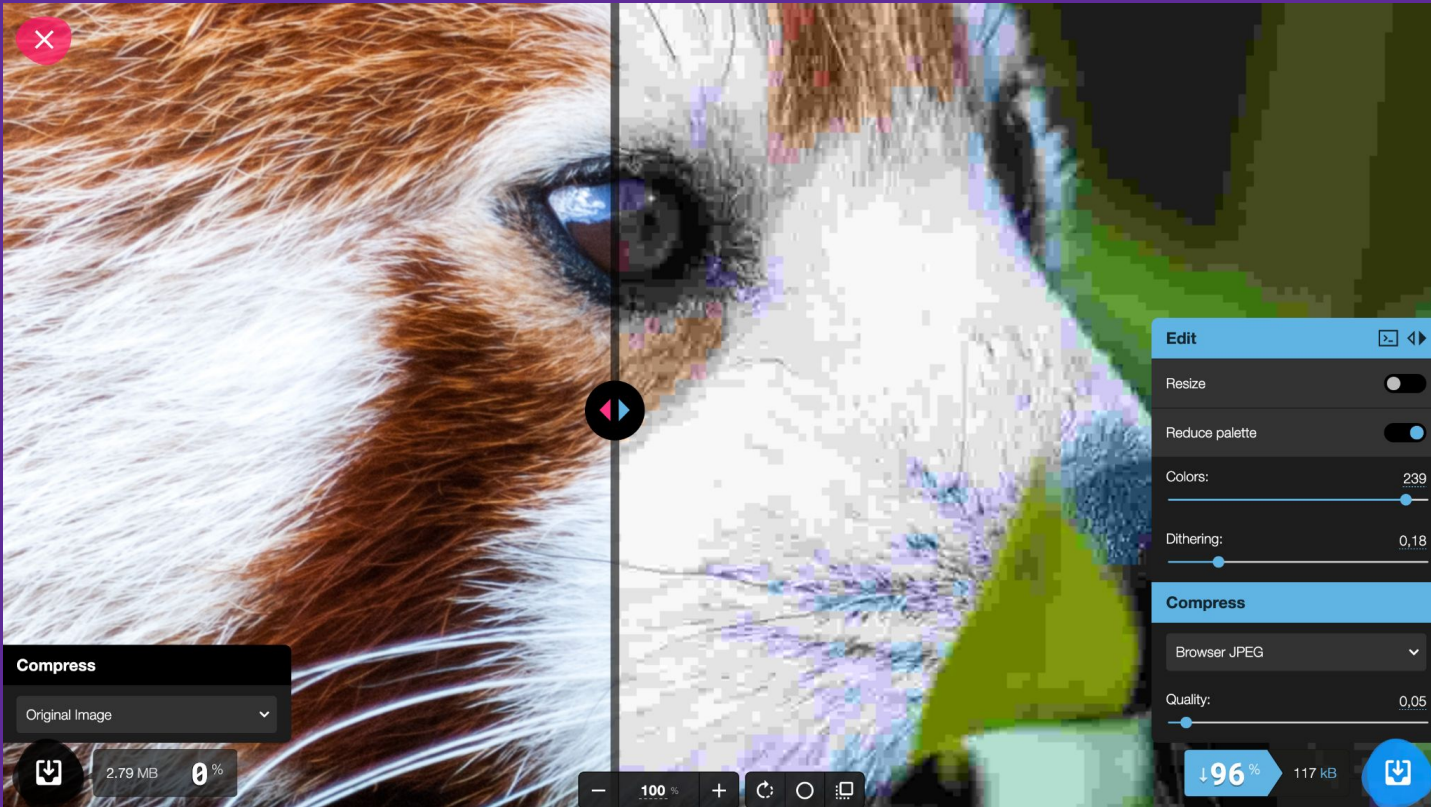
50 Reviews

STREAM ONLY

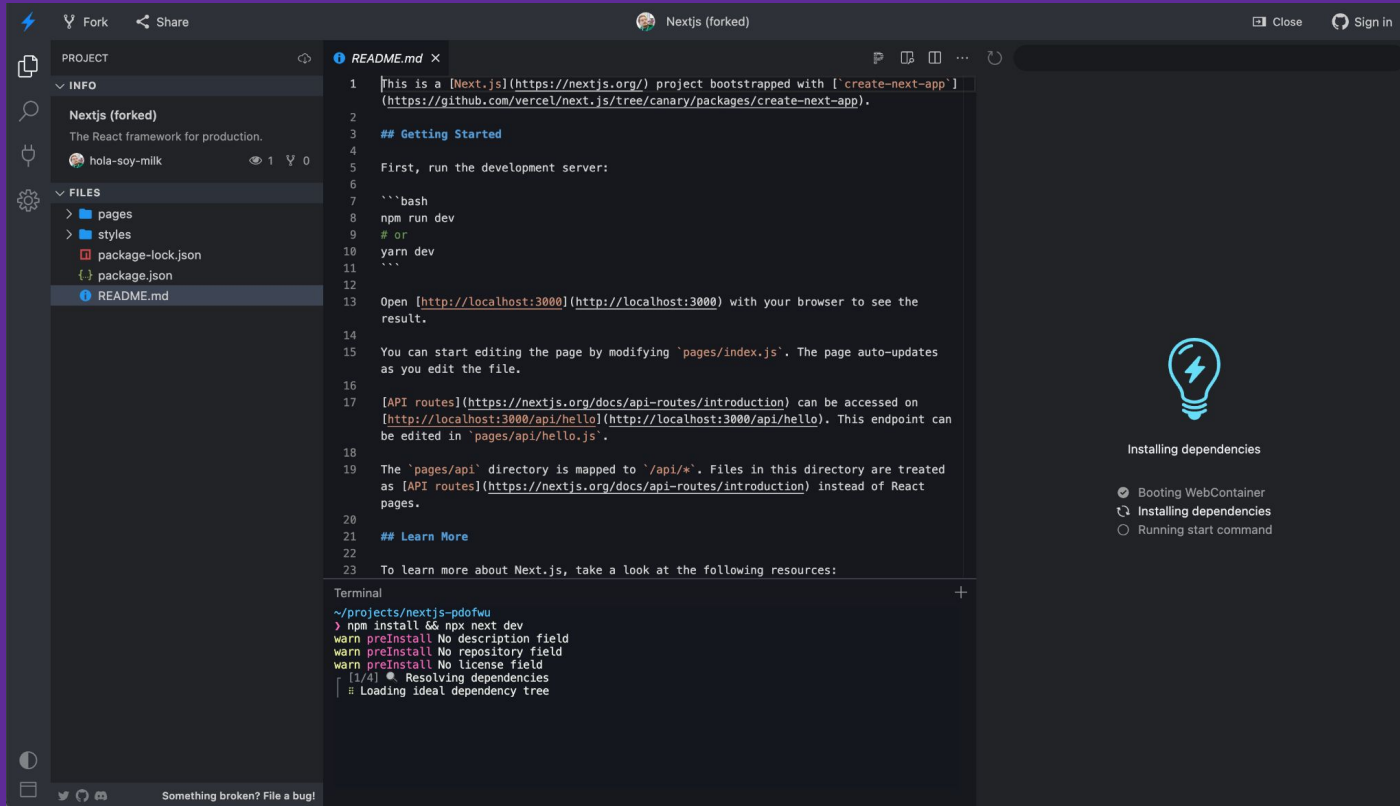
https://archive.org/details/msdos_Prince_of_Persia_1990

¿Qué?

@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 files like `pages`, `styles`, `package-lock.json`, `package.json`, and `README.md`. The main editor area shows the `README.md` content, which includes instructions on how to run the development server and access API routes. A terminal window at the bottom shows the command `npm install && npx next dev` and its output, including warnings about missing fields and the start of dependency resolution.

```
1 [This is a [Next.js](https://nextjs.org/) project bootstrapped with [create-next-app]
2 (https://github.com/vercel/next.js/tree/canary/packages/create-next-app).
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
14 result.
15
16 You can start editing the page by modifying `pages/index.js`. The page auto-updates
17 as you edit the file.
18
19 [API routes](https://nextjs.org/docs/api-routes/introduction) can be accessed on
20 [http://localhost:3000/api/hello](http://localhost:3000/api/hello). This endpoint can
21 be edited in `pages/api/hello.js`.
22
23 The `pages/api` directory is mapped to `/api/*`. Files in this directory are treated
24 as [API routes](https://nextjs.org/docs/api-routes/introduction) instead of React
25 pages.
26
27 ## Learn More
28
29 To learn more about Next.js, take a look at the following resources:
```

```
Terminal
~/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
```



Installing dependencies

- Booting WebContainer
- Installing dependencies
- Running start command


lab.allotropia.de/wasm/

example_larger.odt - LibreOfficeDev Writer 7.4 [d9b2f82b09d89ea918e6d1555d860426fd153c71]

File Edit View Insert Format Styles Table Form Tools Window Help

Title Droid Sans 28 pt

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).

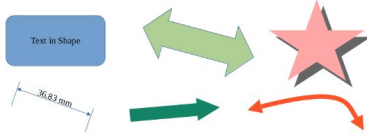
Free as in Freedom, now and forever

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



Textframe

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



Textbox

The quick brown Fox jumps over the lazy Dog

Ya bueno pero esto solo se trata del navegador.

¿Que tiene que ver con Wasm en el servidor?

¿Wasm no anda en cualquier parte, ¿o sí?

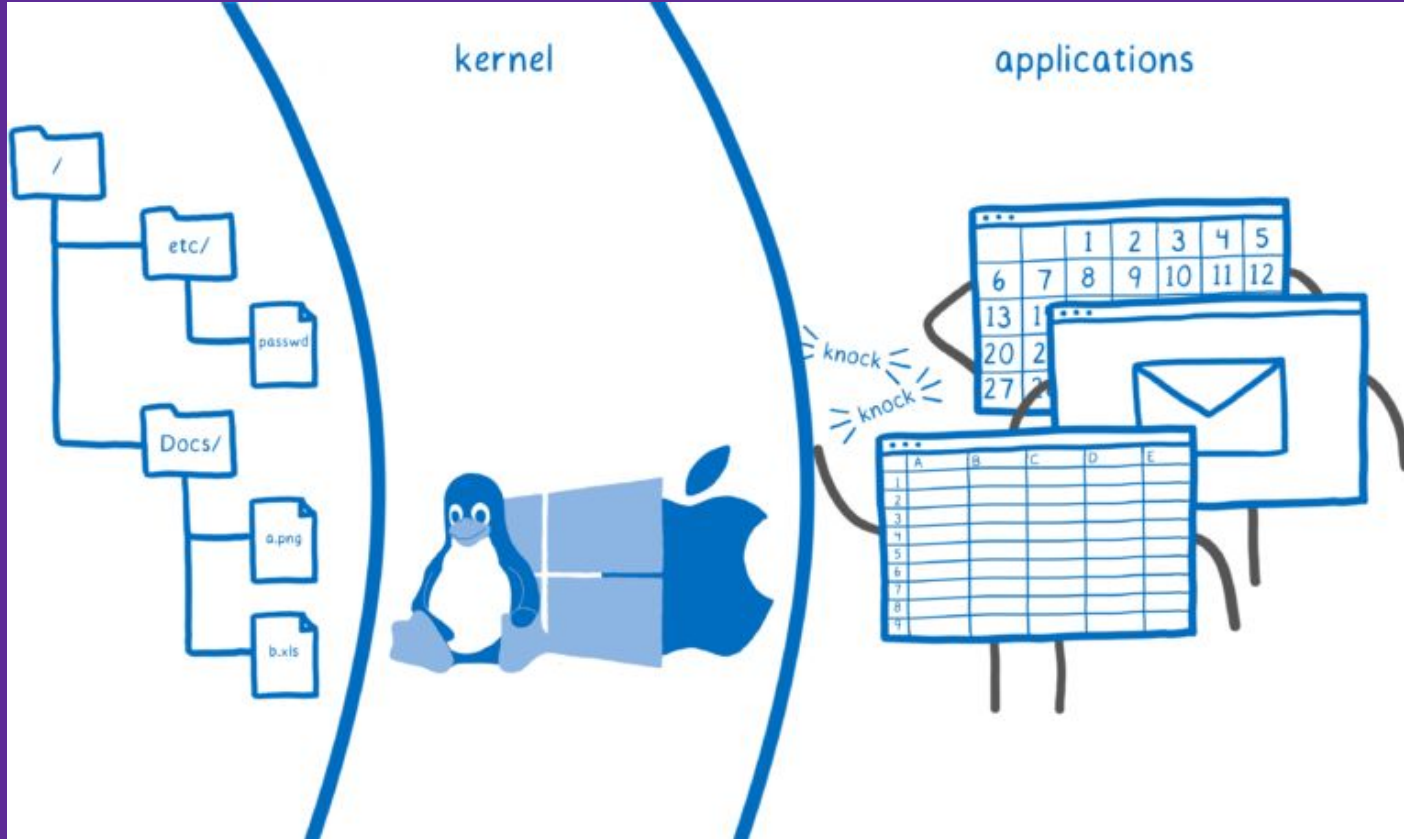
Es aquí donde entra WASI



Es aquí donde entra WASI

WebAssembly,
ahora con
Interfaz de Sistema!





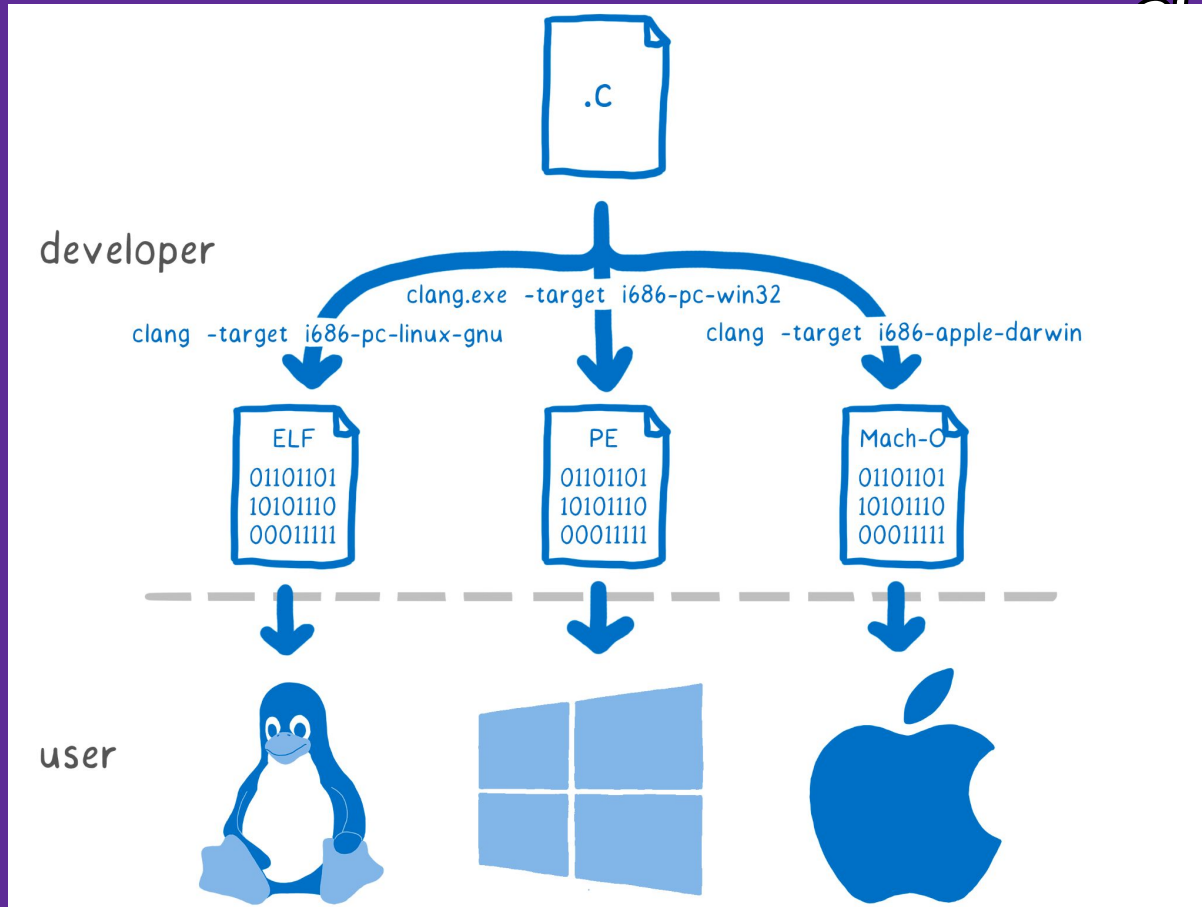
***WASI es similar a este
concepto de un kernel***

***Sin embargo, no
reemplaza el sistema
operativo***

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...

¿Qué?

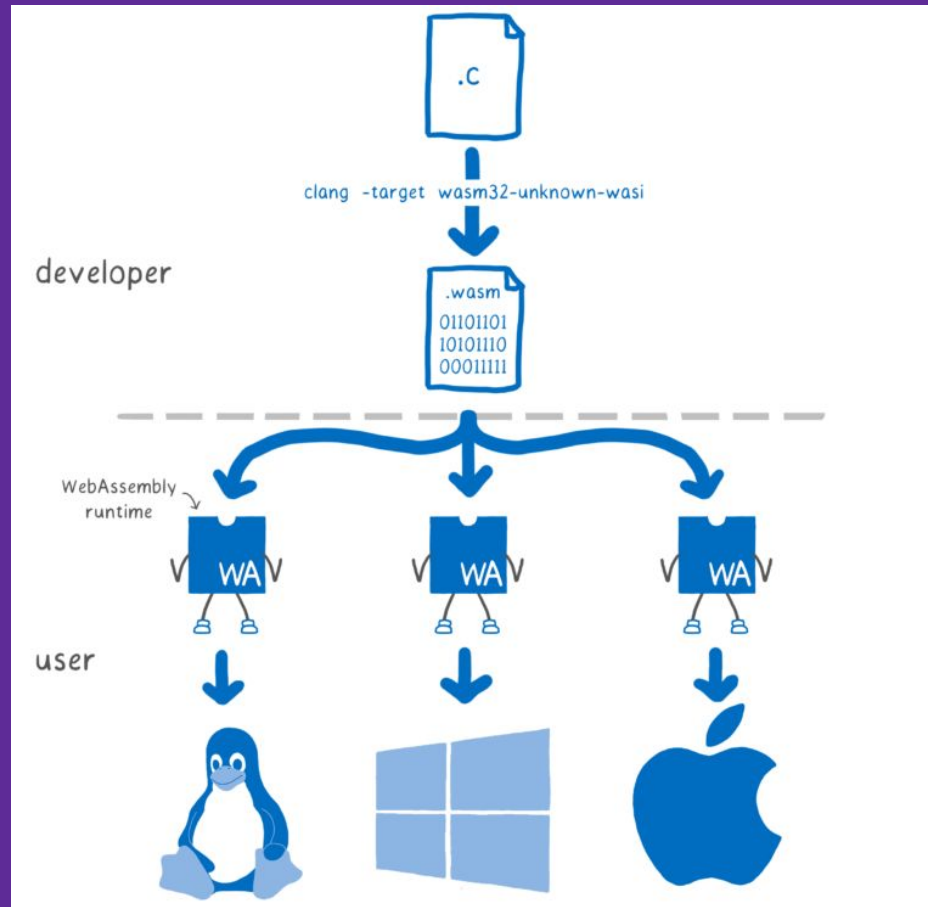
Stola_soy_milk



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

¿Qué?

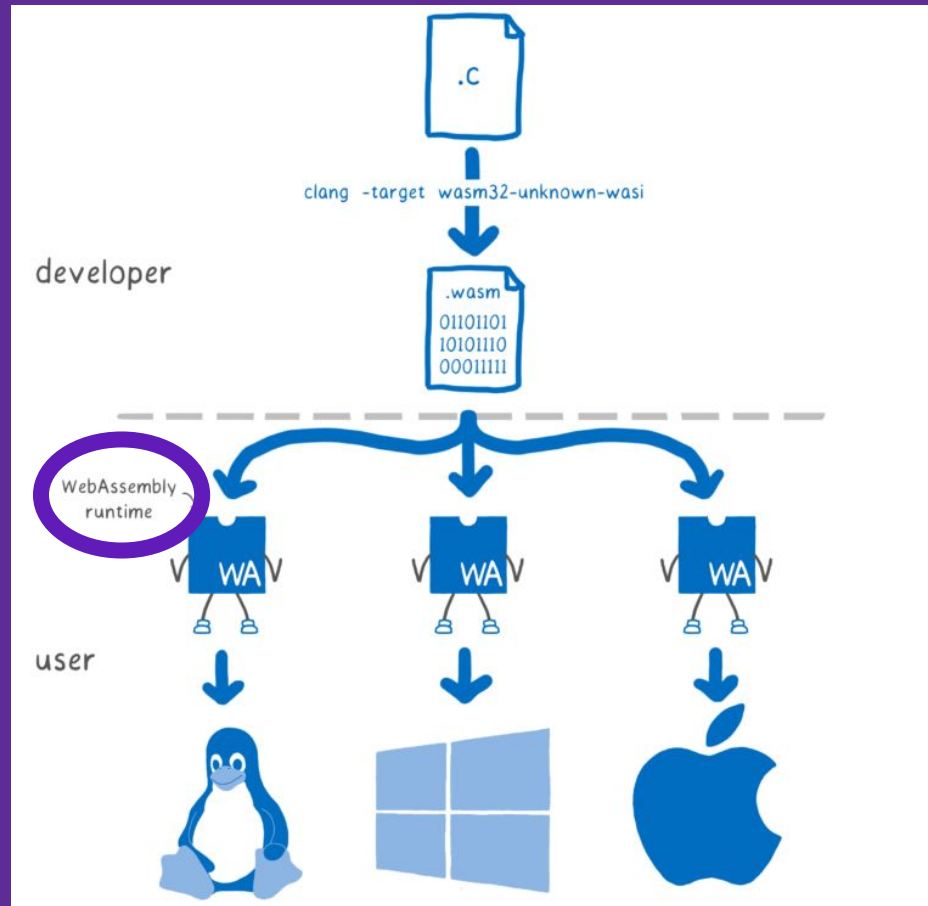
@hola_soy_milk



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

¿Qué?

@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

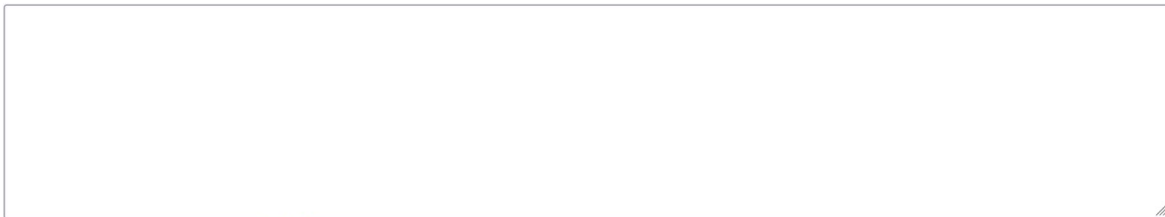
For an overview of WebAssembly, and what WebAssembly is, [take a look here](#).

i You can find the source code of the docs here: github.com/wasmerio/docs.wasmer.io

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



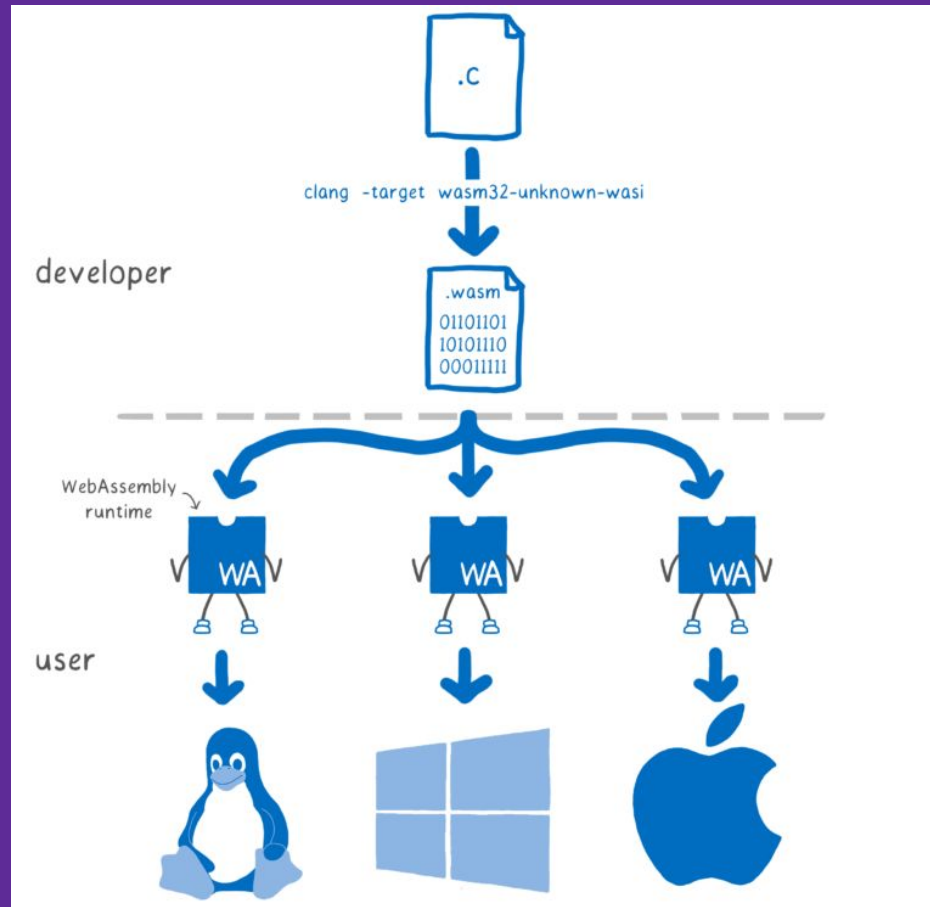
Examinar... Ningún archivo seleccionado.



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 for more information on using WASI.

¿Qué?

@hola_soy_milk



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

WasmEdge

Bring the cloud-native and serverless application paradigms to Edge Computing.

[Get Started](#)[Github](#)

```

(module
  (type (;0;) (func (param i32) (result i32)))
  (func $fib (type 0) (param $n i32) (result i32)
    local.get $n
    i32.const 2
    i32.lt_s
    if ;; label = @1
      i32.const 1
      return
    end
    local.get $n
    i32.const 2
    i32.sub
    call $fib
    local.get $n
    i32.const 1
    i32.sub
    call $fib
  )
)

```

```

(module
  (type (;0;) (func (param i32) (result i32)))
  (func $fib (type 0) (param $n i32) (result i32)
    local.get $n
    i32.const 2
    i32.lt_s
    if ;; label = @1
      i32.const 1
      return
    end
    local.get $n
    i32.const 2
    i32.sub
    call $fib
    local.get $n
    i32.const 1
    i32.sub
    call $fib
    i32.add
    return
  )
  (export "fib" (func $fib))
)

```

```

wasm
func fib(n i32) returns (i32) {
  local.get n
  i32.const 2
  i32.lt_s
  if ;; label = @1
    i32.const 1
    return
  end
  local.get n
  i32.const 2
  i32.sub
  call $fib
  local.get n
  i32.const 1
  i32.sub
  call $fib
  i32.add
  return
}
export "fib" (func $fib)

```

```

[{"name": "fib", "type": "func", "start": 0, "end": 1000000}]]

```



WasmEdge is a lightweight, high-performance, and extensible WebAssembly runtime for cloud native, edge, and decentralized applications.

<https://wasmedge.org/>

¿Qué?

@hola_soy_milk

**o hasta en un
contenedor** 🤯

Lectura adicional (inglés)

- <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/>

Unas preguntitas...

- ¿Qué es Wasm?
 - ¿Por qué?
 - ¿Cómo?
 - ¿Cuándo?
 - ¿Dónde?
-

¿Por qué?

@hola_soy_milk

**¿Por qué ejecutar
Wasm en el servidor?**



WA

¿Por qué?

 **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...

[Mostrar este hilo](#)

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

834 Retweets **164** Tweets citados **2.183** Me gusta

@hola_soy_milk

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

**¡Trae un diseño
“capacity-based
security”!**

¿Por qué?

@hola_soy_milk

¡Es políglota!

***Los componentes se
compilan con tipos,
son pequeños y
gestionables!***

¿Por qué?

@hola_soy_milk

¡Super alta velocidad!

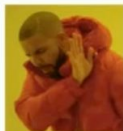
First: why are folks putting wasm in production?

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



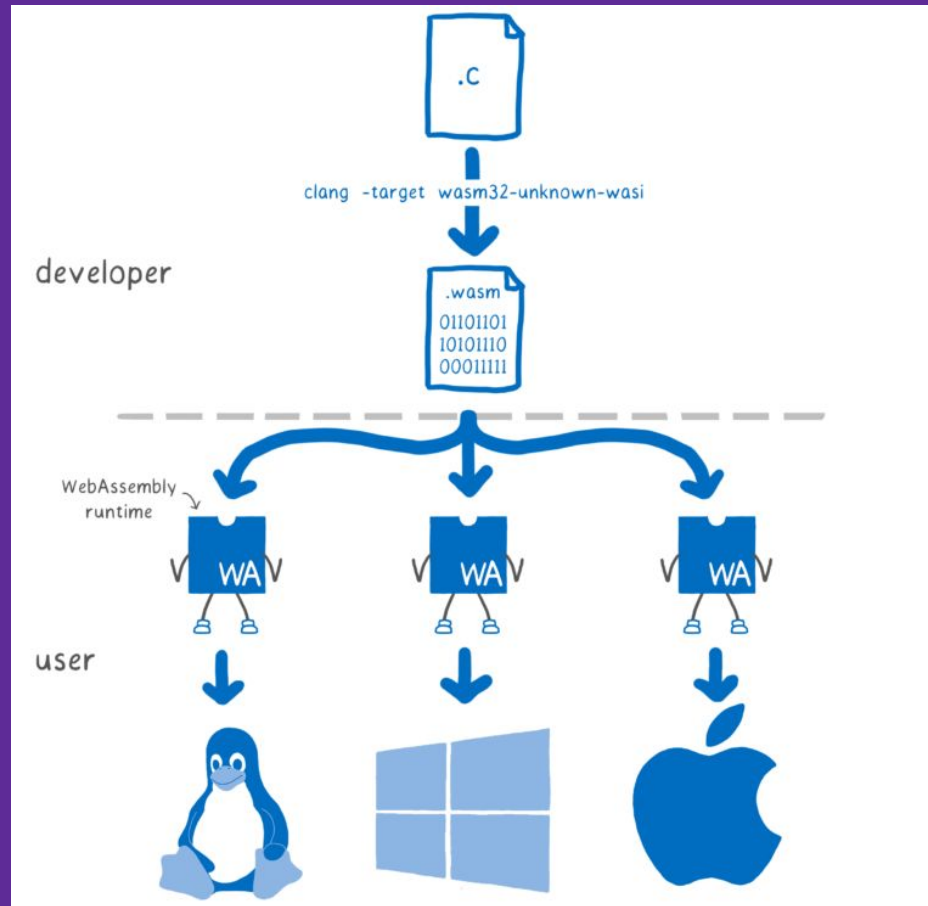
"Serverless" execution: ultra-fast cold start, ephemeral



CLOUD NATIVE
Wasm DAY
NORTH AMERICA

¿Por qué?

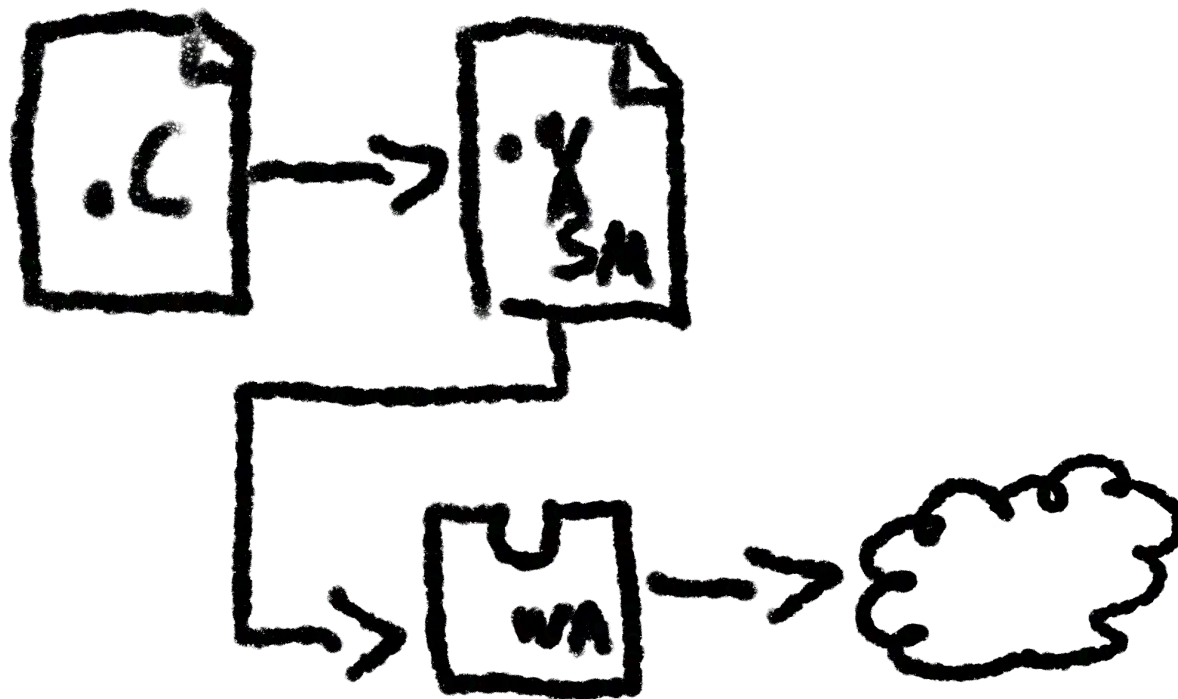
@hola_soy_milk



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

¿Por qué?

@hola_soy_milk



**Los componentes se
adjuntan a los recursos
del sistema al echarse
a andar**

¿Qué implica esto para el desarrollo de software?

Podemos codear sin preocuparnos de...

Gestionar servidores

Podemos codear sin preocuparnos de...

Tiempo de arranque y cierre

¿Por qué?

@hola_soy_milk

Podemos codear sin preocuparnos de...

Escalabilidad

Podemos codear sin preocuparnos de...

Debilidades de seguridad

Lectura Adicional

- <https://www.secondstate.io/articles/why-webassembly-server/>
- https://wasmedge.org/book/en/use_cases/server_side_render.html
- <https://www.wasm.builders/thomastaylor312/why-webassembly-belongs-outside-the-browser-331a>

Unas preguntitas...

- ¿Qué es Wasm?
 - ¿Por qué ejecutar Wasm?
 - ¿Cómo?
 - ¿Cuándo?
 - ¿Dónde?
-

**¿Cómo se está usando
Wasm en el servidor?**



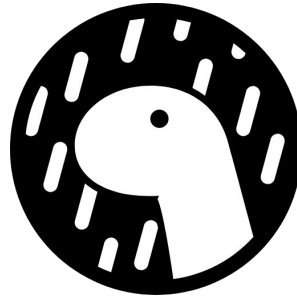
WA

¿Cómo?

@hola_soy_milk

Functions-as-a-Service (FaaS)

fastly



¿Cómo?

@hola_soy_milk

Edge Computing / Microservices



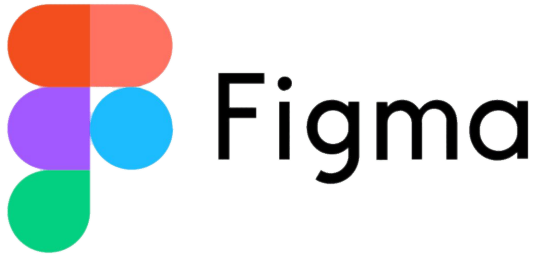
FERMYON



¿Cómo?

@hola_soy_milk

Extensibilidad



¿Cómo?

@hola_soy_milk

Blockchain



¿Cómo?

@hola_soy_milk

Embedido



Akri

Lectura adicional (inglés)

- <https://shopify.engineering/shopify-webassembly>
- <https://blog.suborbital.dev/webassembly-extensibility-today-and-tomorrow>
- <https://www.wasm.builders/aryank21/why-wasm-is-the-perfect-runtime-for-server-side-applications-1b9p>

Unas preguntitas...

- ¿Qué es Wasm?
 - ¿Por qué ejecutar Wasm?
 - ¿Cómo se está usando Wasm?
 - ¿Cuándo?
 - ¿Dónde?
-

**¿Cuándo podemos
reemplazar los
contenedores con
Wasm?**



WA

¿Cuándo?

... @hola_soy_milk



Solomon Hykes

@solomonstre

“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

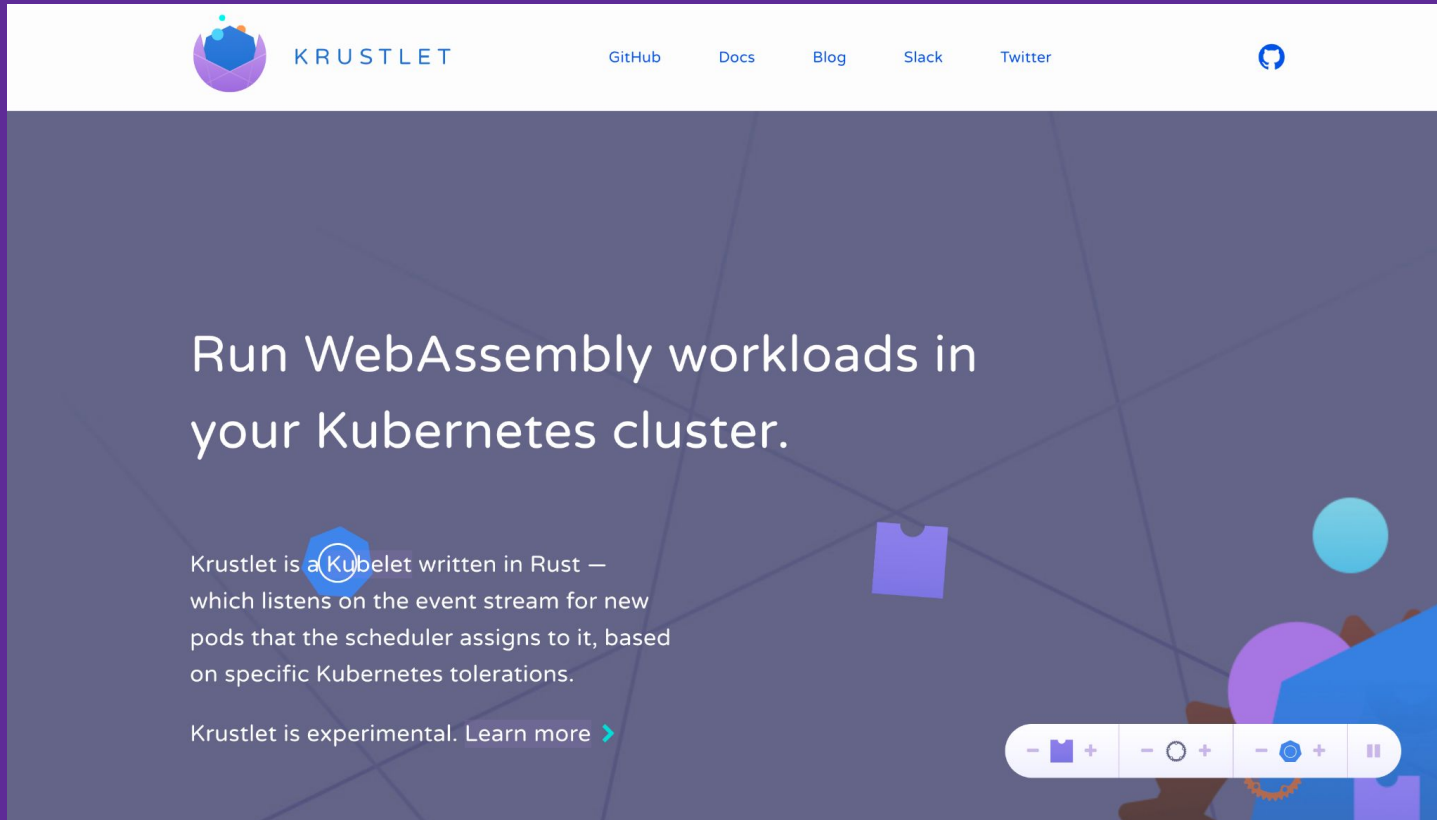
<https://twitter.com/solomonstre/status/111113329647325185>

Los contenedores no reemplazaron completamente los VMs...

... Reemplazaron los VMs en situaciones no ideales donde se usaban VMs

Wasm no reemplazará los contenedores...

... Reemplazarán los contenedores en situaciones no ideales donde se usan contenedores



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, followed by the text 'KRUSTLET'. To the right of the logo are links for 'GitHub', 'Docs', 'Blog', 'Slack', and 'Twitter'. Further right is a GitHub icon. The main content area has a dark blue background with a grid pattern and abstract shapes. The primary heading reads 'Run WebAssembly workloads in your Kubernetes cluster.' Below this, a paragraph explains that Krustlet is a Kubelet written in Rust that listens to the event stream for new pods based on specific tolerations. A link 'Krustlet is experimental. Learn more >' is provided. At the bottom right of the page, there is a video player control bar with icons for zooming in/out, full screen, and pausing.

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 >](#)

¿Cuándo?


@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/>

Oye y hay más:

- Component Model**

Oye y hay más:

- Component Model**
- wasi-*nn***

Oye y hay más:

- Component Model**
- wasi-nn**
- Garbage Collection**

Oye y hay más:

- **Component Model**
- **wasi-nn**
- **Garbage Collection**
- **Multihilo**

¿Cuándo?

@hola_soy_milk

**¿Quién quiere Wasm
con Docker?** 🙋

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

Docker Scout

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

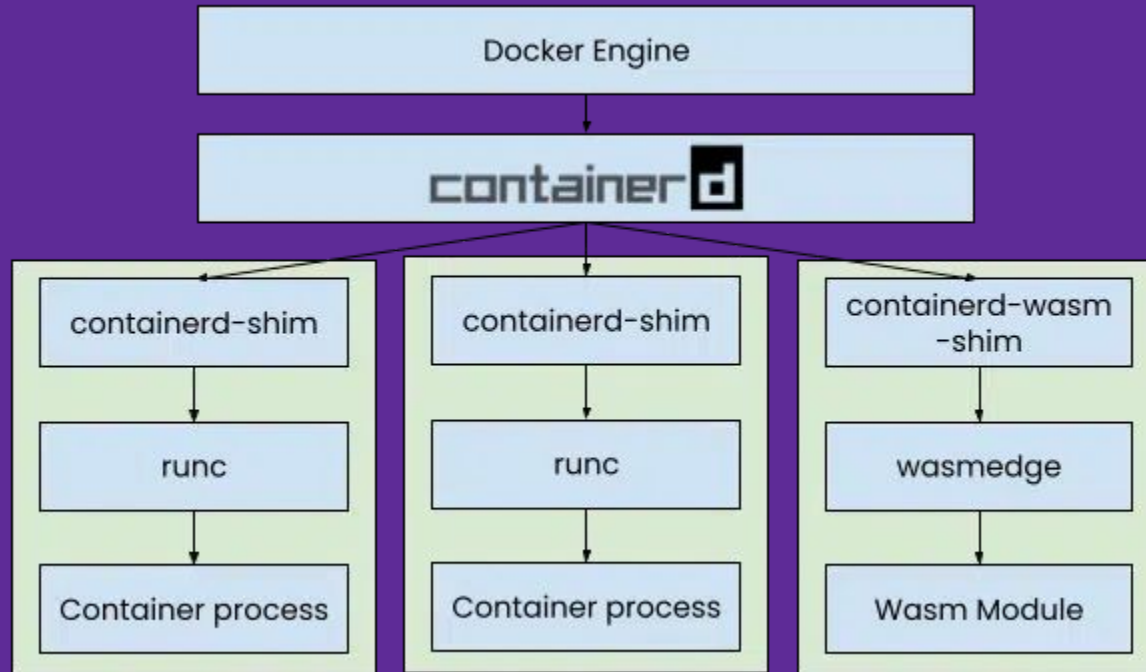
The Docker+Wasm integration currently requires a technical preview build of Docker Desktop.

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



***Mucho de lo que les estoy
mostrando es nuevo y
WIP...***

¡El futuro se ve brillante!

Lectura adicional (inglés)

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

Unas preguntitas...

- ¿Qué es Wasm?
 - ¿Por qué ejecutar Wasm?
 - ¿Cómo se está usando Wasm?
 - ¿Cuándo se viene?
 - ¿Dónde?
-

**¿Dónde encontramos
a la gente
colaborando en
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

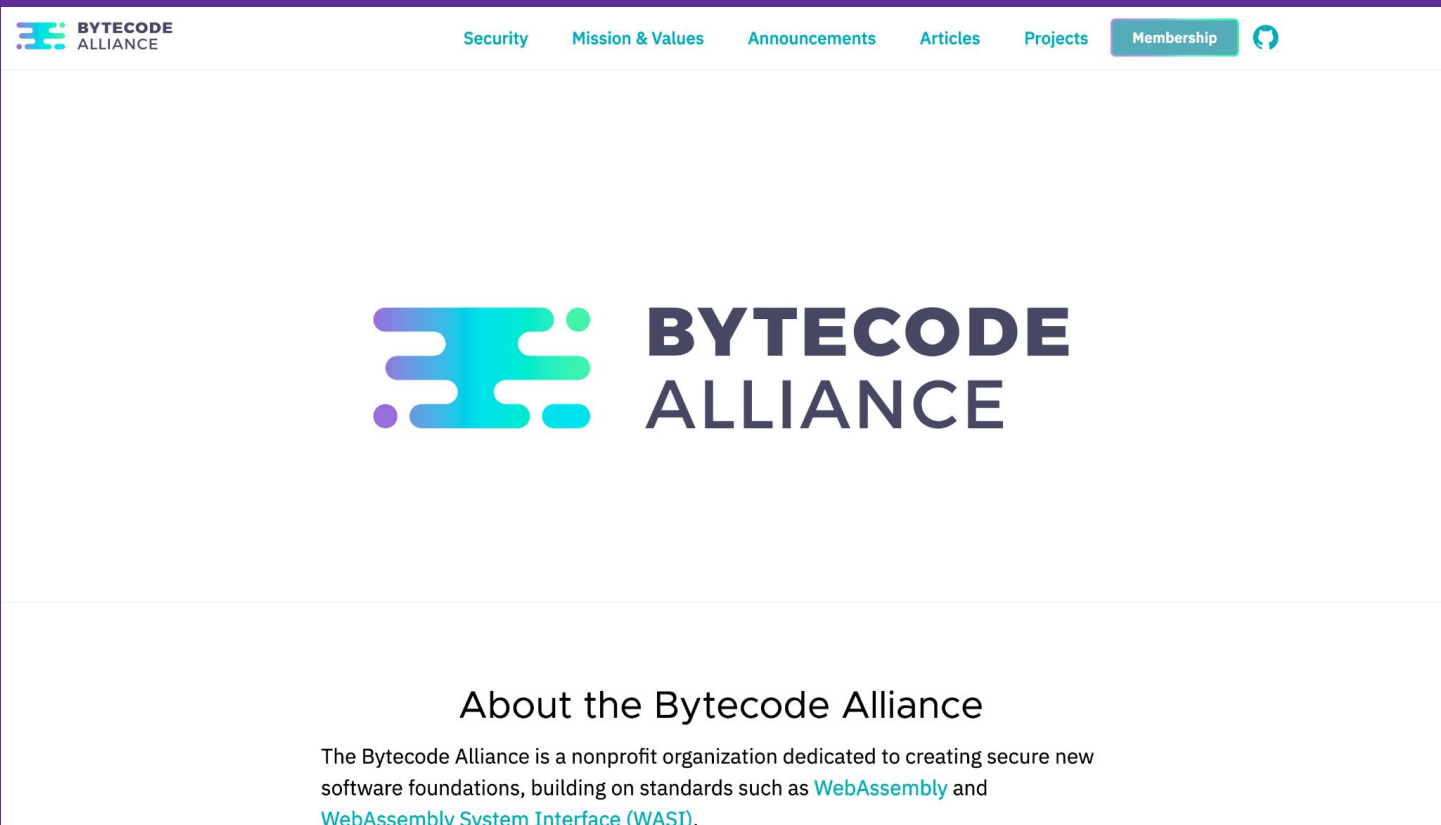
#showcase

- 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



Home / WebAssembly Community Group

WEBASSEMBLY COMMUNITY GROUP

The mission of this group is to promote early-stage cross-browser collaboration on a new, portable, size- and load-time-efficient format suitable for compilation to the web.

[webassembly/spec](#)

Group's public email, repo and wiki activity over time

Year	J	F	M	A	M	J	J	A	S	O	N	D
2015												
2016												
2017												
2018												
2019												
2020												
2021												
2022												

Note: Community Groups are proposed and run by the community. Although W3C hosts these conversations, the groups do not necessarily represent the views of the W3C Membership or staff.

No Reports Yet Published

Chairs, when logged in, may publish draft and final reports. Please see [report requirements](#).

PUBLISH REPORTS

Tools for this group

- Mailing List
- IRC
- Github repository
- RSS
- Contact This Group

Resources

- Charter
- Code of Conduct

Get involved

Anyone may join this Community Group. All participants in this group have signed the W3C Community Contributor License Agreement.

JOIN OR LEAVE THIS GROUP

Derek Schuff *Chairs*

GitHub navigation: Search or jump to... Pull requests Issues Marketplace Explore

WebAssembly Development of WebAssembly and associated infrastructure
 752 followers The Web! https://webassembly.org

Overview **Repositories 101** Projects 1 Packages People 29

Pinned

- meetings** (Public) WebAssembly meetings (VC or in-person), agendas, and notes
HTML 360 117
- design** (Public) WebAssembly Design Documents
11k 711
- spec** (Public) WebAssembly specification, reference interpreter, and test suite.
WebAssembly 2.7k 417
- proposals** (Public) Tracking WebAssembly proposals
752 44
- gc** (Public) Branch of the spec repo scoped to discussion of GC integration in WebAssembly
WebAssembly 598 47
- WASI** (Public) WebAssembly System Interface
Rust 3.2k 202

People

View all

Top languages

- WebAssembly
- Rust
- C++
- HTML
- JavaScript

Most used topics

<https://github.com/WebAssembly>

Dale, hemos aprendido...

- Lo que es Wasm
 - Por qué usar Wasm
 - Cómo se está usando Wasm
 - Cuándo se viene
 - Dónde se está avanzando
-

<https://ramonh.dev/wasm-en-el-servidor.pdf>

¡Mil Gracias!



¡Tengo stickers!

Ramón Huidobro

@hola_soy_milk