RED BADGER

WebAssembly

Stuart Harris, Red Badger

HbbTV

March 2025

Hi, I'm Stu

- Software engineer
- Founder and Chief Scientist at Red Badger
- @stuartharris

RED BADGER

About us

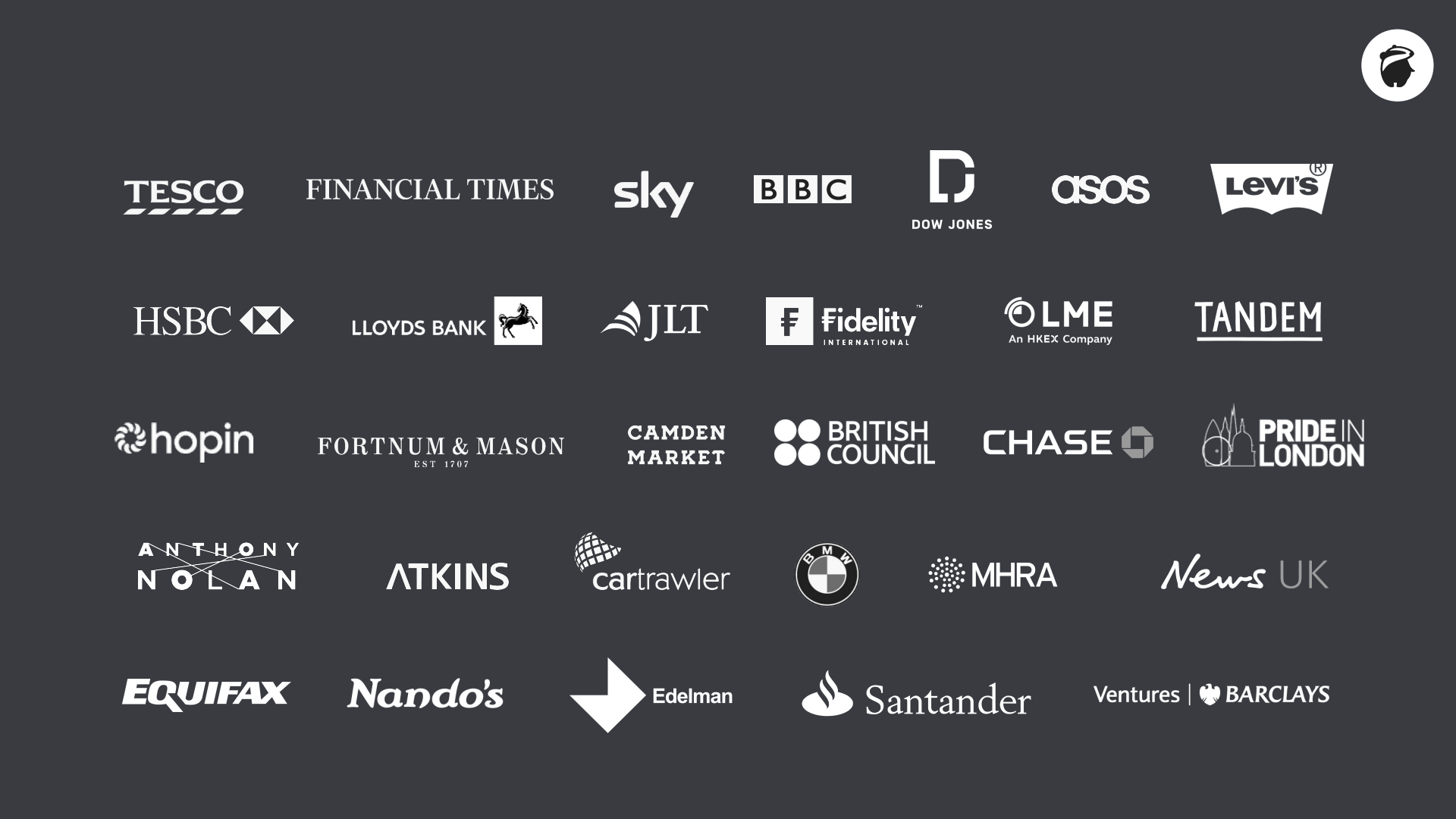
Red Badger is the Digital Product transformation consultancy

We help modern enterprises continuously evolve their products and services. We craft digital products customers love, build next generation platforms and embed new digital capabilities

15
Years old, founded 2010

c.100
People

c.90%+
Permanent, London team


TESCO
FINANCIAL TIMES
sky
BBC
DOW JONES
asos
Levi's
HSBC
LLOYDS BANK
JLT
Fidelity
INTERNATIONAL
LME
An HKEX Company
TANDEM
hopin
FORTNUM & MASON
EST 1707
CAMDEN MARKET
BRITISH COUNCIL
CHASE
PRIDE IN LONDON
ANTHONY NOLAN
ATKINS
cartrawler
MHRA
News UK
EQUIFAX
Nando's
Edelman
Santander
Ventures | BARCLAYS

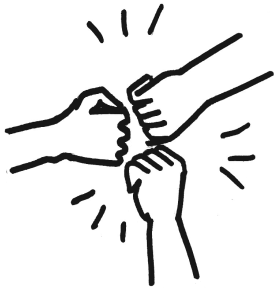
What is WebAssembly?

Well, it's not (only) Web and it's not Assembly!

It *is* a bytecode (like Java bytecode or the Common Intermediate Language of .Net).

More formally — it's a binary instruction format for a stack-based virtual machine.

How is it different?



Simple

Possibly the simplest virtual machine we have. Only has **4 types** (i32/64 and f32/64). No baked-in OOP concepts (like JVM). No coupling to APIs, the DOM, or screen-space (like Java applets).

Secure

Designed to run untrusted code in the browser. **Deny-by-default** sandbox ensures code cannot, itself, run *any* side effects.

Speedy

Designed to run code at **native** speeds. Lightweight. No runtime or garbage collection (although there is a spec for that). Streaming compilation.

A brief History

2011

C/C++ to JS with **emscripten**

2013

asm.js, a strict subset of JS

2014-2015

Nov 2017
all major browsers

WebAssembly 1.0

2019

W3C: *4th language of the web alongside HTML, JS, CSS*

2026?

WebAssembly 2.0 (draft)

WebAssembly Interface Types (**WIT**)

2022

WebAssembly **Component Model**

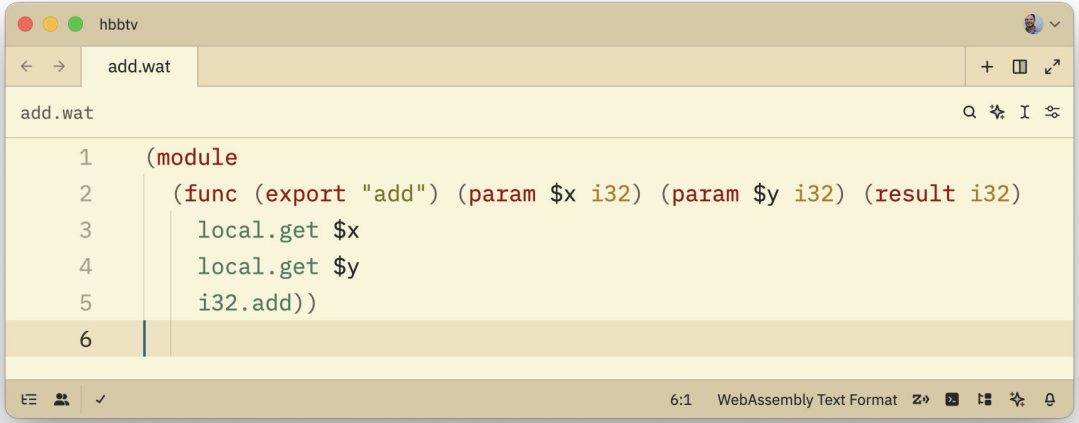
2024

WebAssembly System Interface (**WASI**)

Demo

What is the simplest WebAssembly module we could create?

WebAssembly Text

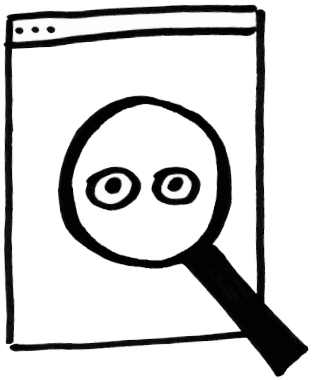


WebAssembly Modules

Demo

Let's use a *real programming language* to create a WebAssembly module on MacOS.

... and then let's run that *same binary* on a Linux machine!



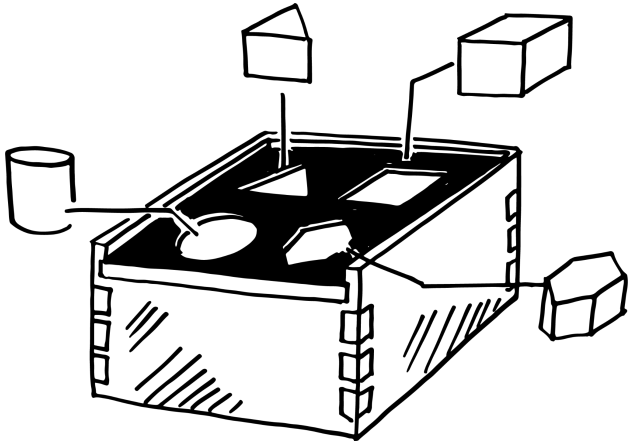
WebAssembly Module in Rust

WebAssembly Components

WebAssembly Interface Types

WASI preview 1 - modules

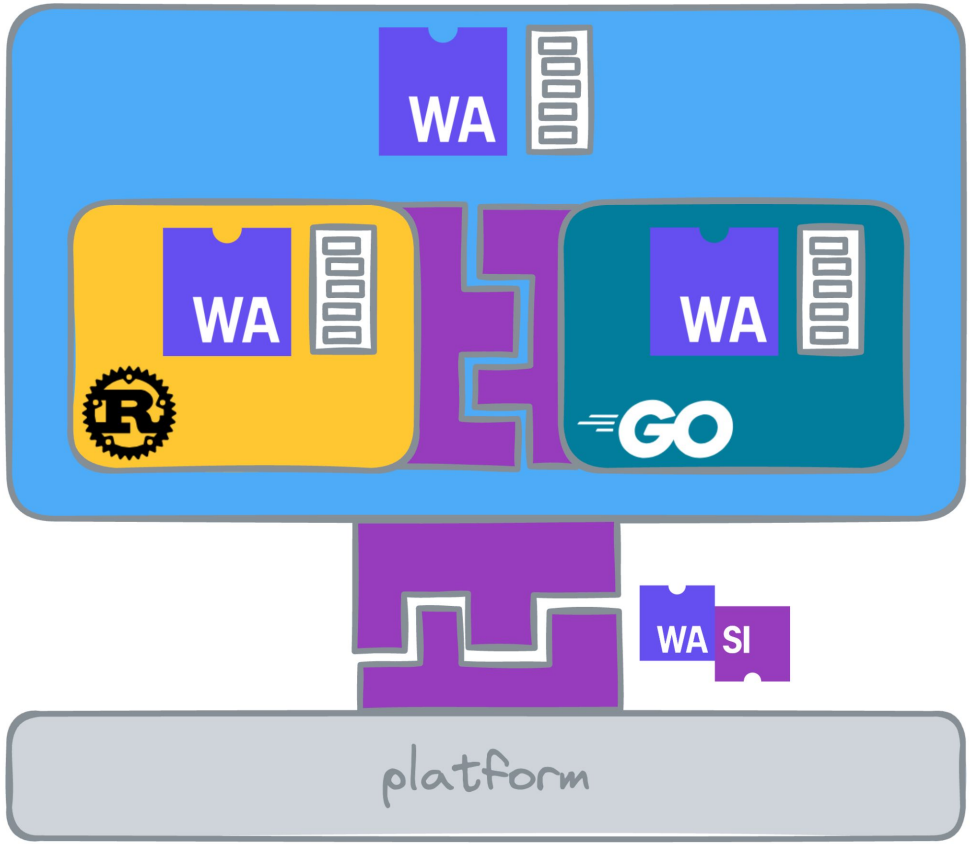
WASI preview 2 - components



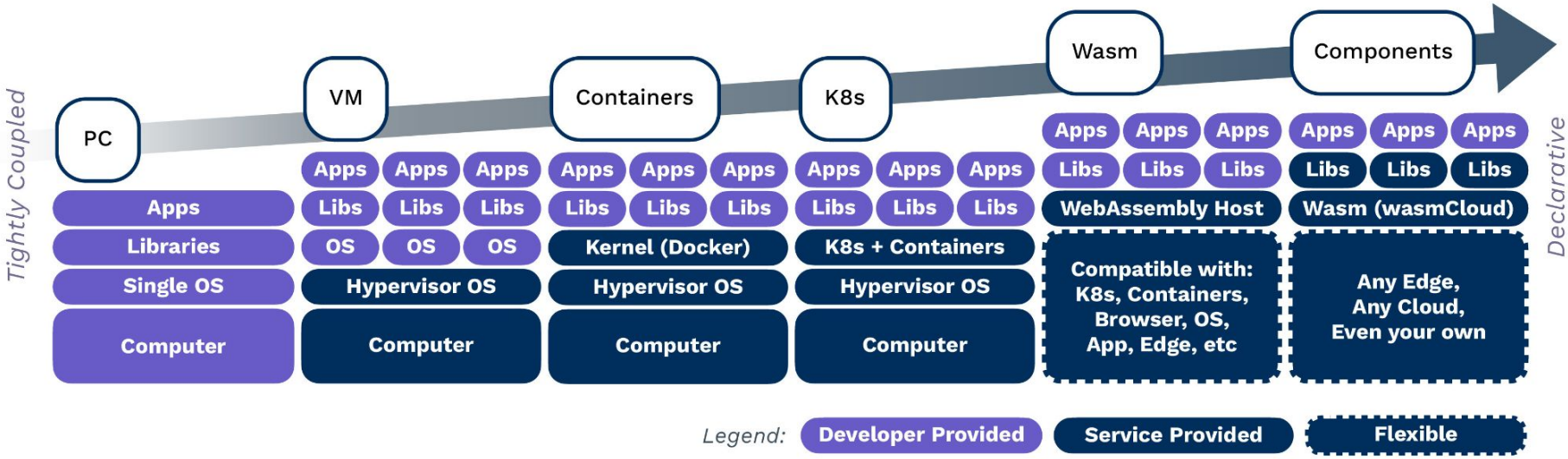
- Polyglot
- Canonical ABI, with static and dynamic linking
- Capability based security
- Bindings generation — e.g. wasm-bindgen
- Interface virtualisation — a component can’t tell if the other side is another component or the host
- Shared nothing architecture, with resources

WebAssembly Components

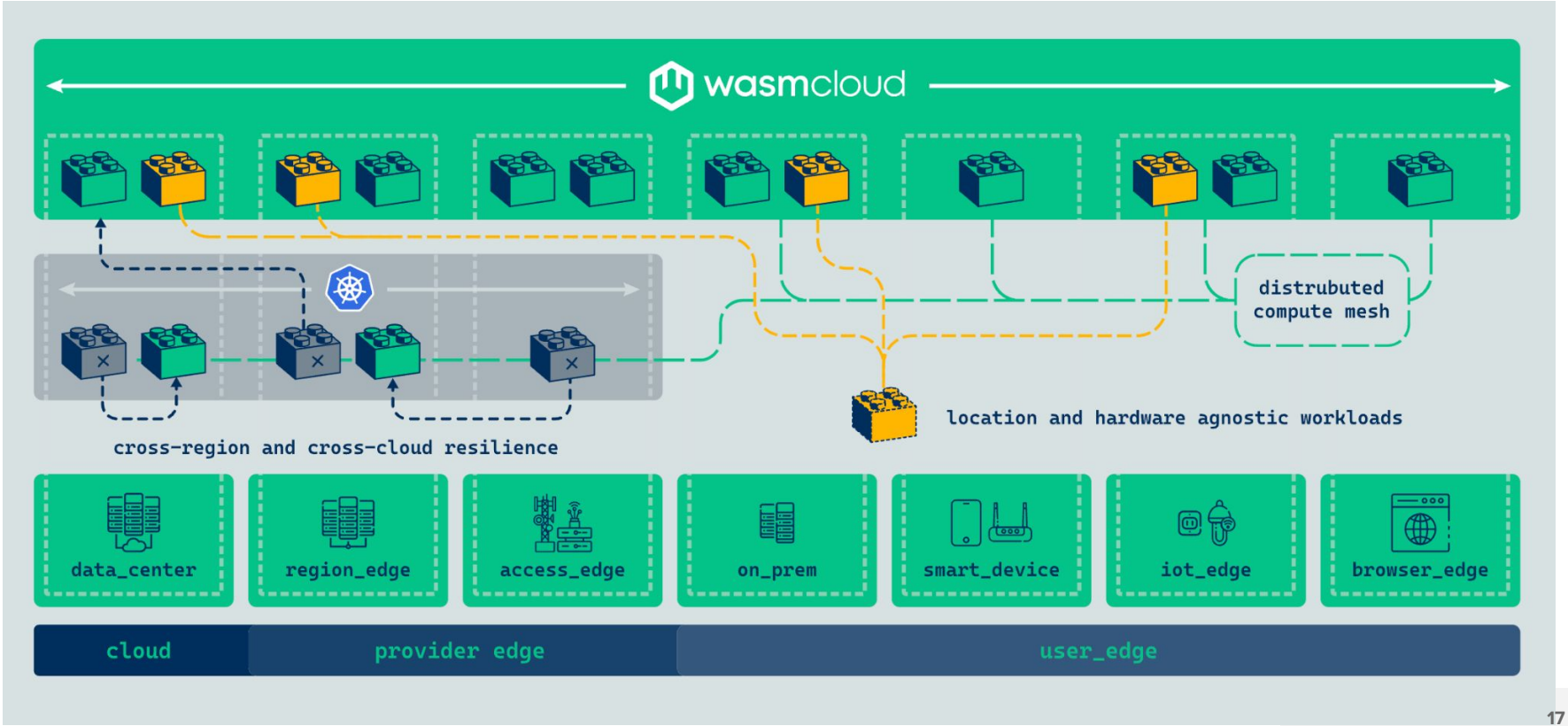
WebAssembly Components



Platform evolution



The wasmCloud platform

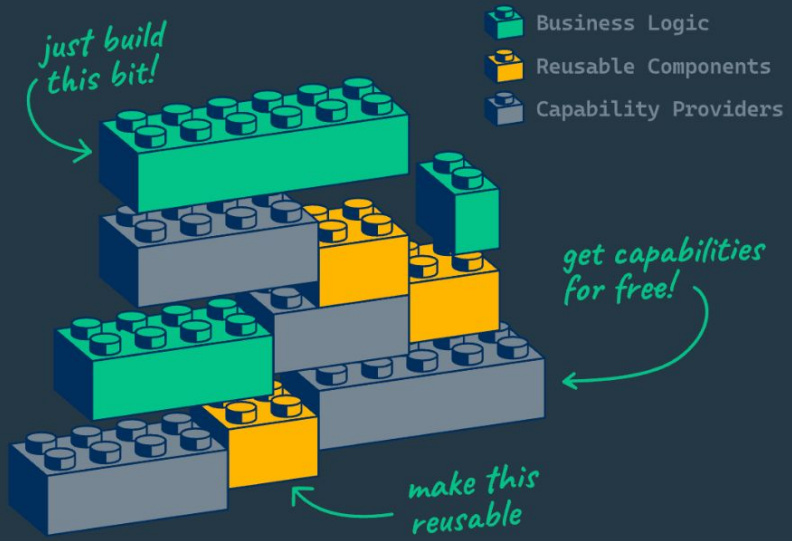


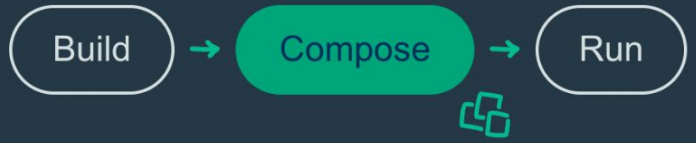
Build

Faster Development Cycles
Leverage reusable, polyglot, Wasm components on a reliable, distributed platform.

Centrally Maintainable Apps
Reusable, version-controlled components empower platform teams to maintain thousands of diverse apps centrally.

Integrate with Existing Stacks
wasmCloud has first-tier support for Kubernetes, AWS, Azure, GCP, Jenkins, Github Actions, ArgoCD, Backstage, Chainguard, Databases, Messaging, and more.





Compose

Development Without Lock-In

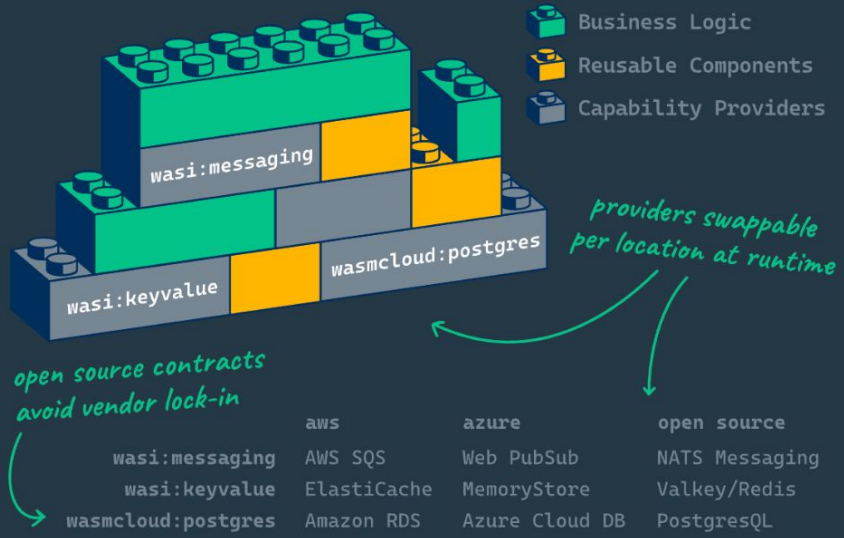
Define application dependencies at runtime via contract driven interfaces leveraging different vendors across deployments, dev, QA, or prod.

Truly Portable Apps

Run the same Wasm application across operating systems and architectures—no new builds required. Linux, MacOS X, Windows, ARM, x86, and more.

Custom Capabilities

Easily extend the secure wasmCloud host at runtime to support custom dependencies, hardware, or business contracts.



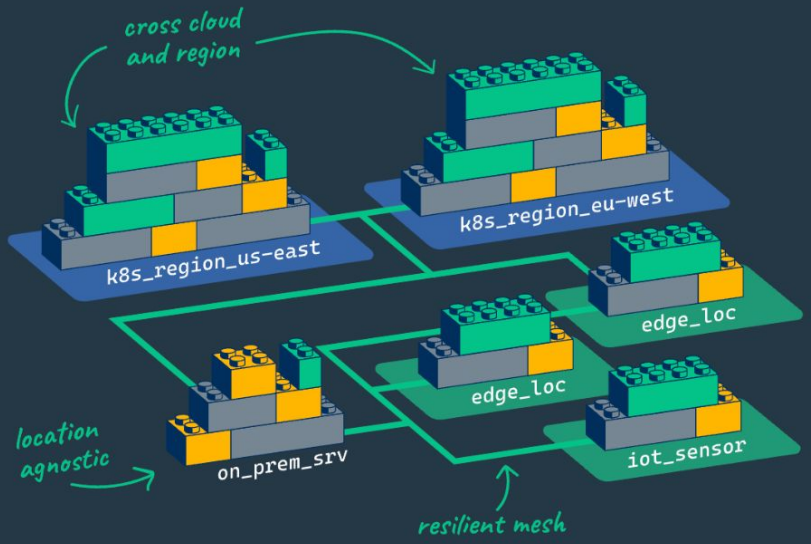
Build → Compose → Run

Run

Scale-to-Zero with Zero Cold Starts
Sub-millisecond start times and vertical autoscaling means workloads scale to the demand.

Reliable, Fault-Tolerant Apps
Horizontal scaling with automated fail-over gives apps capability-level resiliency, reliability, and scalability.

Deploy Across Clouds
Close to your users, with local-first routing and at-most-once delivery, wasmCloud delivers cross-region, cross-cloud, and cross-edge capability-level resiliency to every deployment



WebAssembly

“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.”

Performance

Simple stack-based virtual machine for executing code written in *any* language at near native speeds, with almost no overhead.

Safety

Designed for running untrusted code in the browser, Wasm’s sandbox is *essential* for running enterprise applications that are composed from open source software.

Portability

Portable across all machine architectures and operating systems, Wasm binaries are small and can start up instantly. Components are a *standard shape* and portable across platforms and clouds.

Thank you

red-badger.com

hello@red-badger.com