

Has WebAssembly been lost in the AI hype?

Could it be the key
to sustainable computing?



RED BADGER

Stuart Harris, 15th Oct 2024

CIO
NAVIGATE LOCAL
LONDON 2024

A path to more sustainable computing

- WebAssembly
- WebAssembly Components
- Vertical scaling and scale-to-zero
- Next generation platforms
- Carbon, cost, and sustainability
- How do enterprises get there?

Hi, I'm Stu

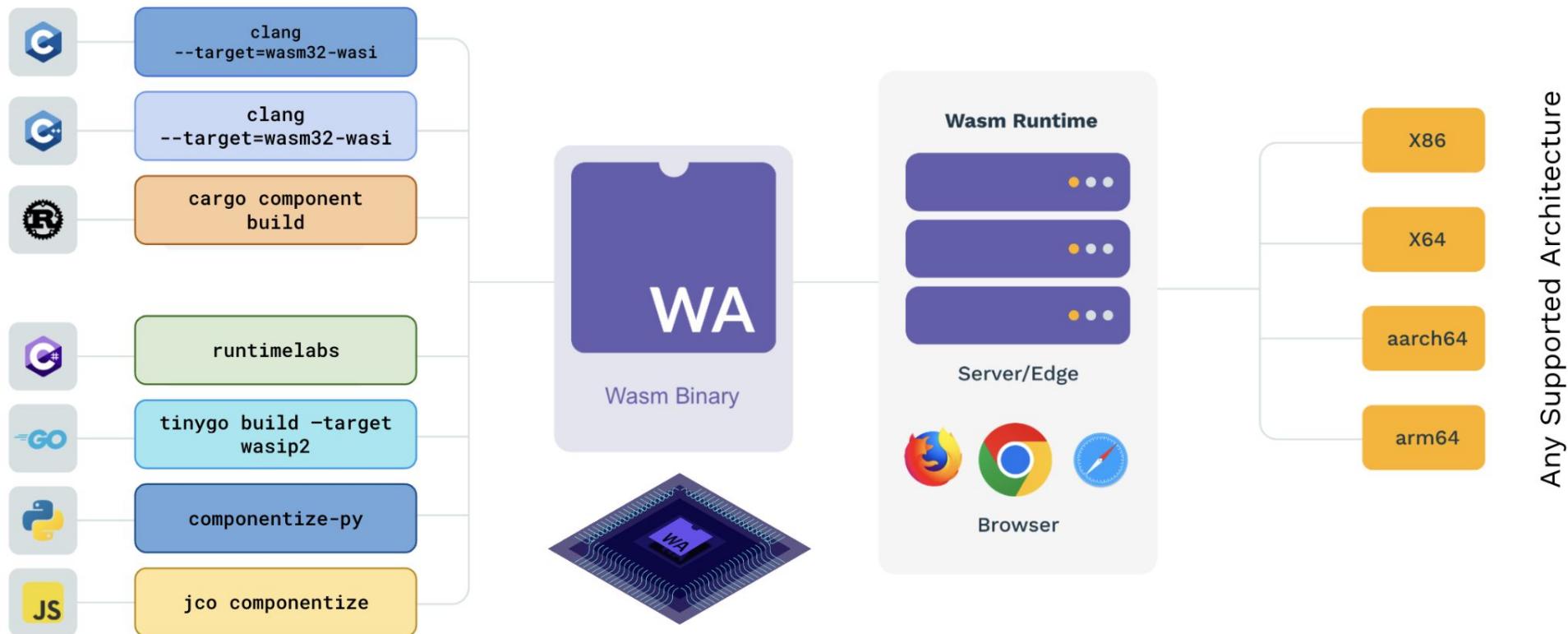
- Software engineer
- Founder and Chief Scientist
- @stuartharris



WebAssembly (Wasm)



WebAssembly



```
apple stuartharris ~ /wasm ⟳ ① 14:34
```

```
→ cargo new hello-world
```

```
Creating binary (application) `hello-world` package
```

```
note: see more `Cargo.toml` keys and their definitions at https://doc.rust-lang.org/cargo/reference/manifest.html
```

```
apple stuartharris ~ /wasm ⟳ ① 14:35
```

```
→ cd hello-world/
```

```
apple stuartharris ... /hello-world ⟳ ② master ? ⟳ v1.81.0 ⟳ ① 14:35
```

```
→ bat ./src/main.rs
```

```
File: ./src/main.rs
```

```
1 fn main() {  
2     println!("Hello, world!");  
3 }
```

```
apple stuartharris ... /hello-world ⟳ ② master ? ⟳ v1.81.0 ⟳ ① 14:35
```

```
→ cargo build --target wasm32-wasip1 --release
```

```
Compiling hello-world v0.1.0 (/Users/stuartharris/wasm/hello-world)
```

```
Finished `release` profile [optimized] target(s) in 0.60s
```

```
apple stuartharris ... /hello-world ⟳ ② master ? ⟳ v1.81.0 ⟳ ① 14:35
```

```
→ eza -la ./target/wasm32-wasip1/release/hello-world.wasm
```

```
.rwxr-xr-x 64k stuartharris 14 Oct 14:35 ./target/wasm32-wasip1/release/hello-world.wasm
```

```
apple stuartharris ... /hello-world ⟳ ② master ? ⟳ v1.81.0 ⟳ ① 14:35
```

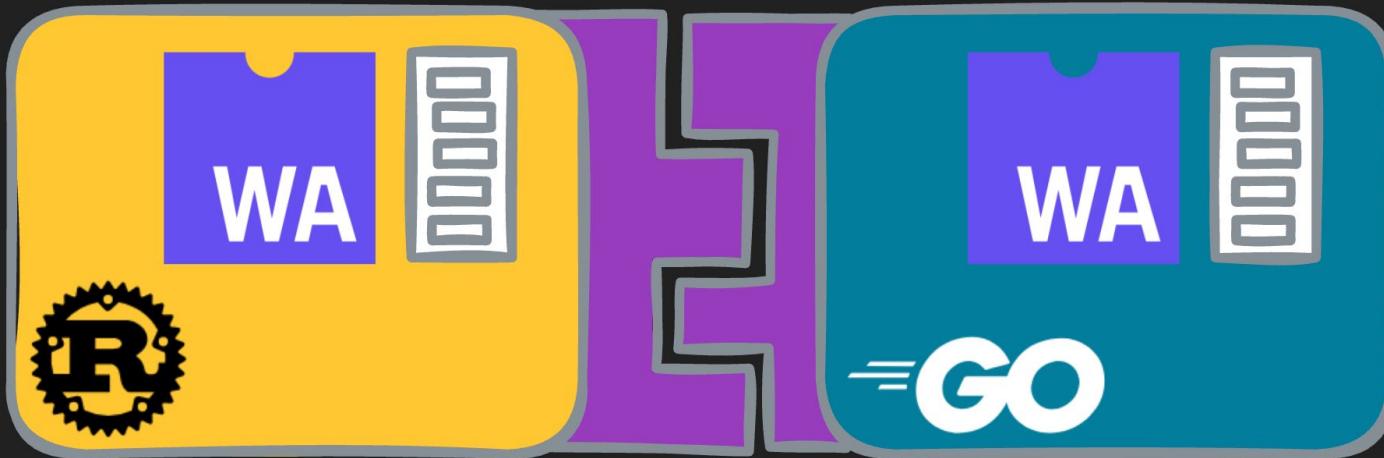
```
→ wasmtime run ./target/wasm32-wasip1/release/hello-world.wasm
```

```
Hello, world!
```

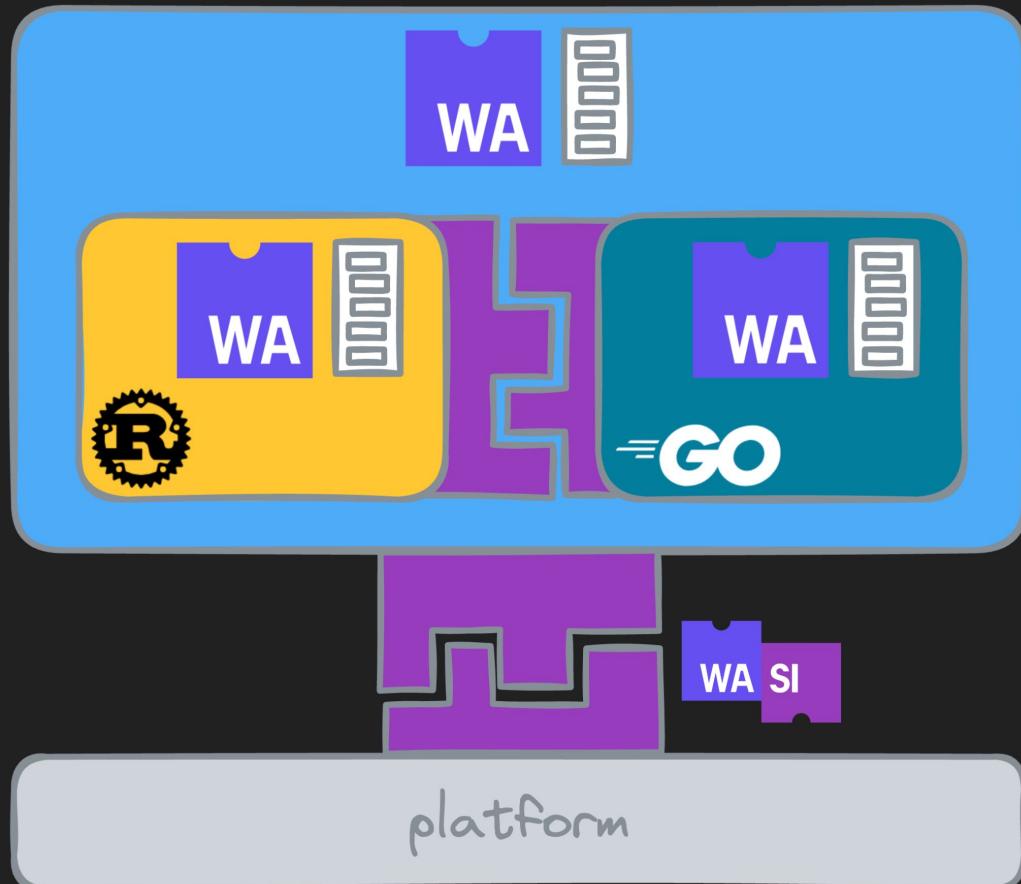
```
apple stuartharris ... /hello-world ⟳ ② master ? ⟳ v1.81.0 ⟳ ① 14:35
```

```
→
```

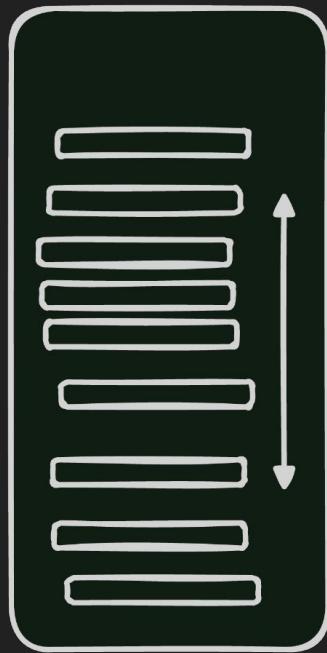
WebAssembly components



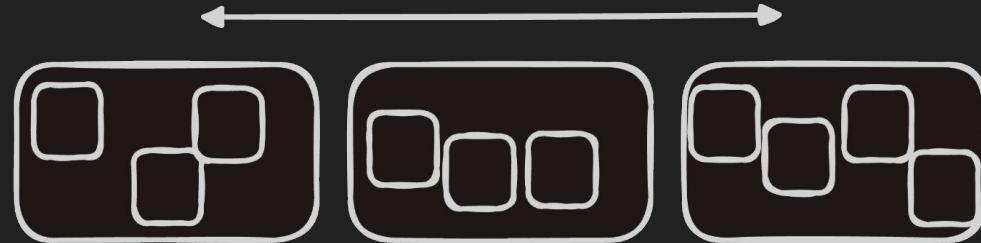
WebAssembly components



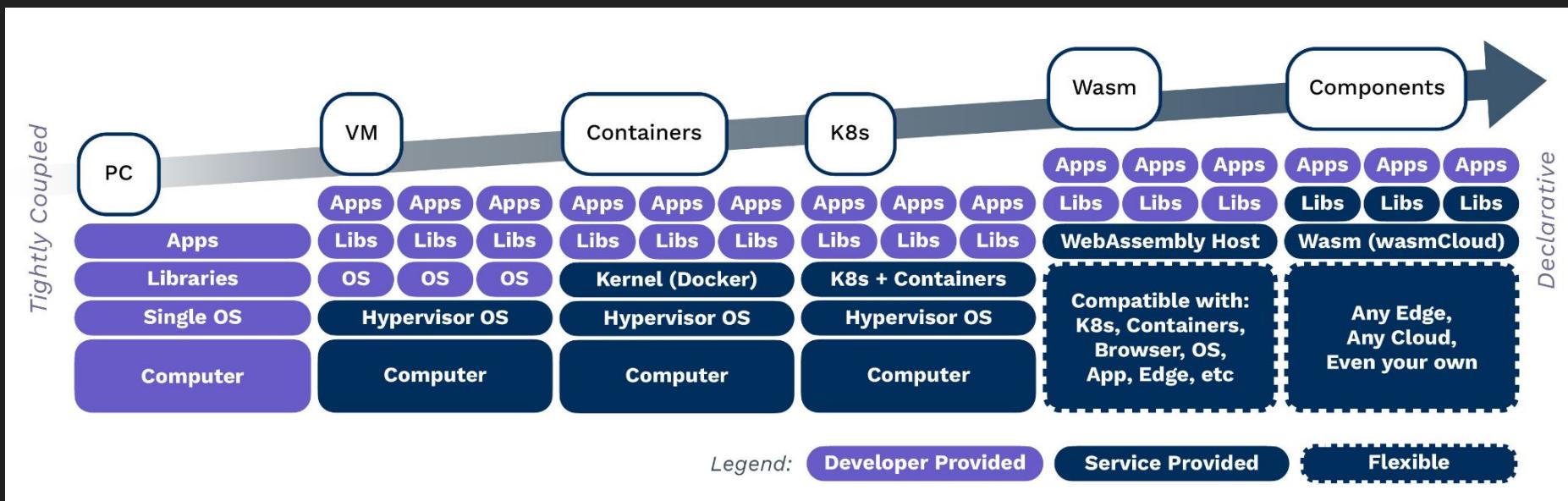
Scaling Vertically and Scale-to-zero



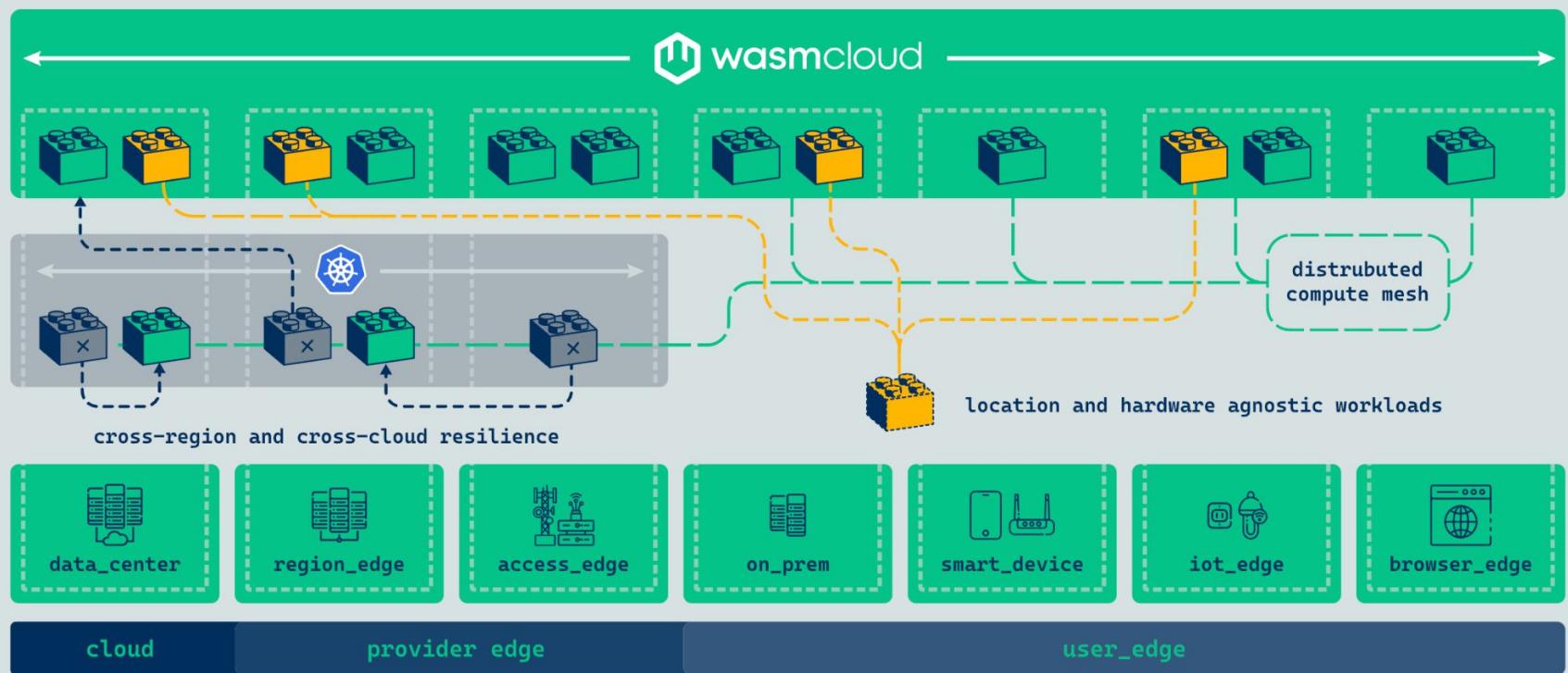
vs



Platform evolution



wasmCloud





Build

Faster Development Cycles

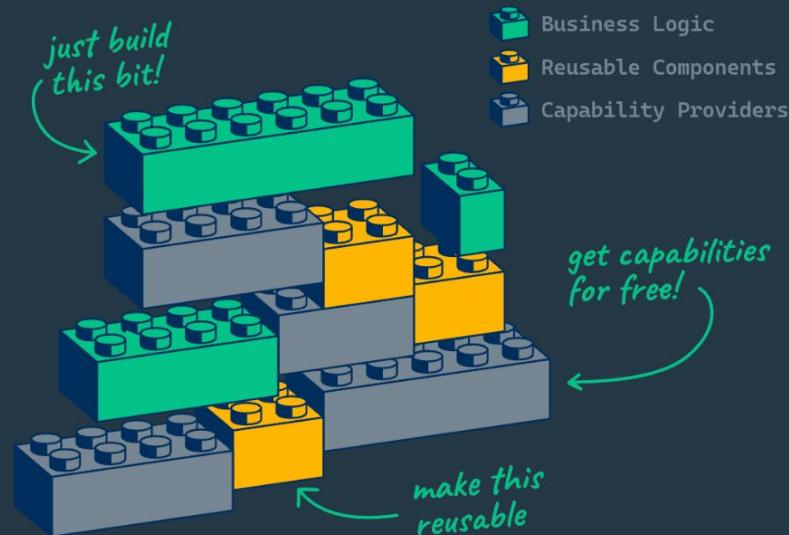
Leverage reusable, polyglot, Wasm components on a reliable, distributed platform.

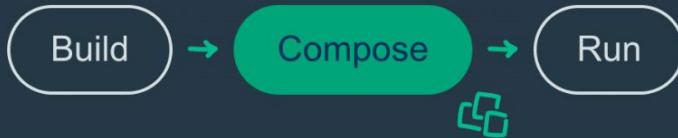
Centrally Maintainable Apps

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

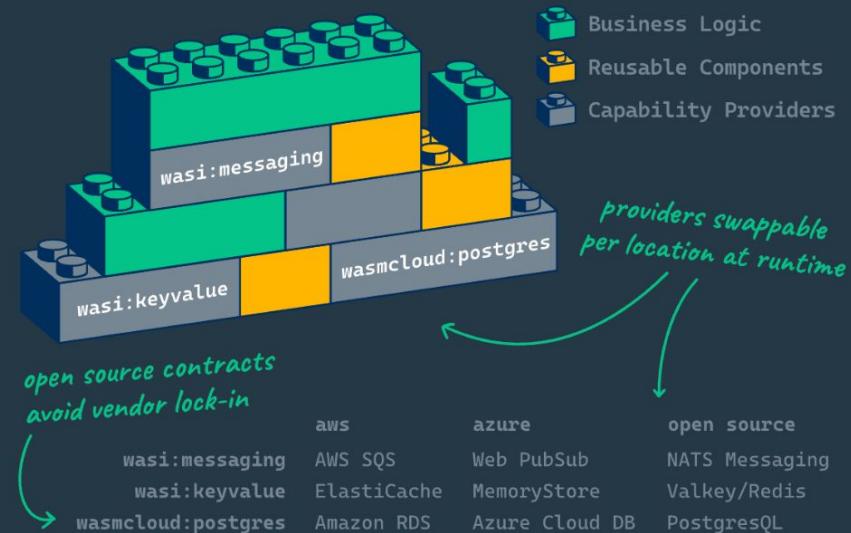
Define application dependencies at runtime via contract driven interfaces leveraging different vendors across deployments, dev, QA, or prod.

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.





Run

Scale-to-Zero with Zero Cold Starts

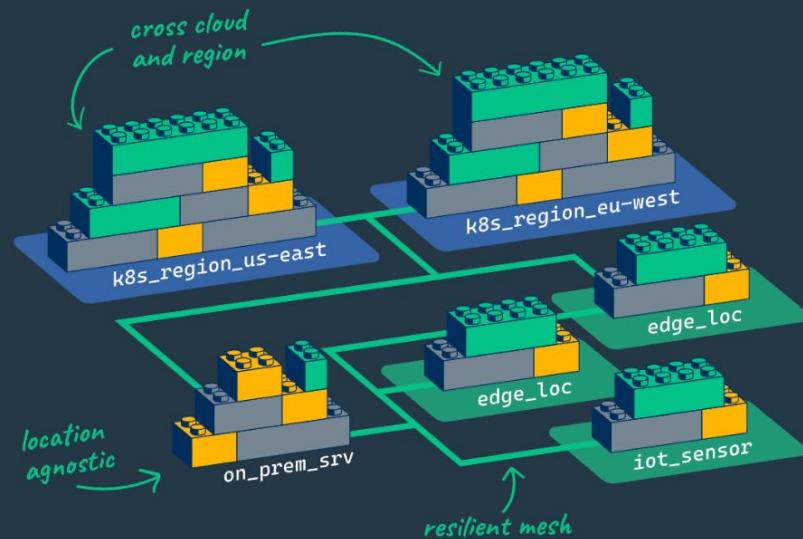
Sub-millisecond start times and vertical autoscaling means workloads scale to the demand.

Reliable, Fault-Tolerant Apps

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

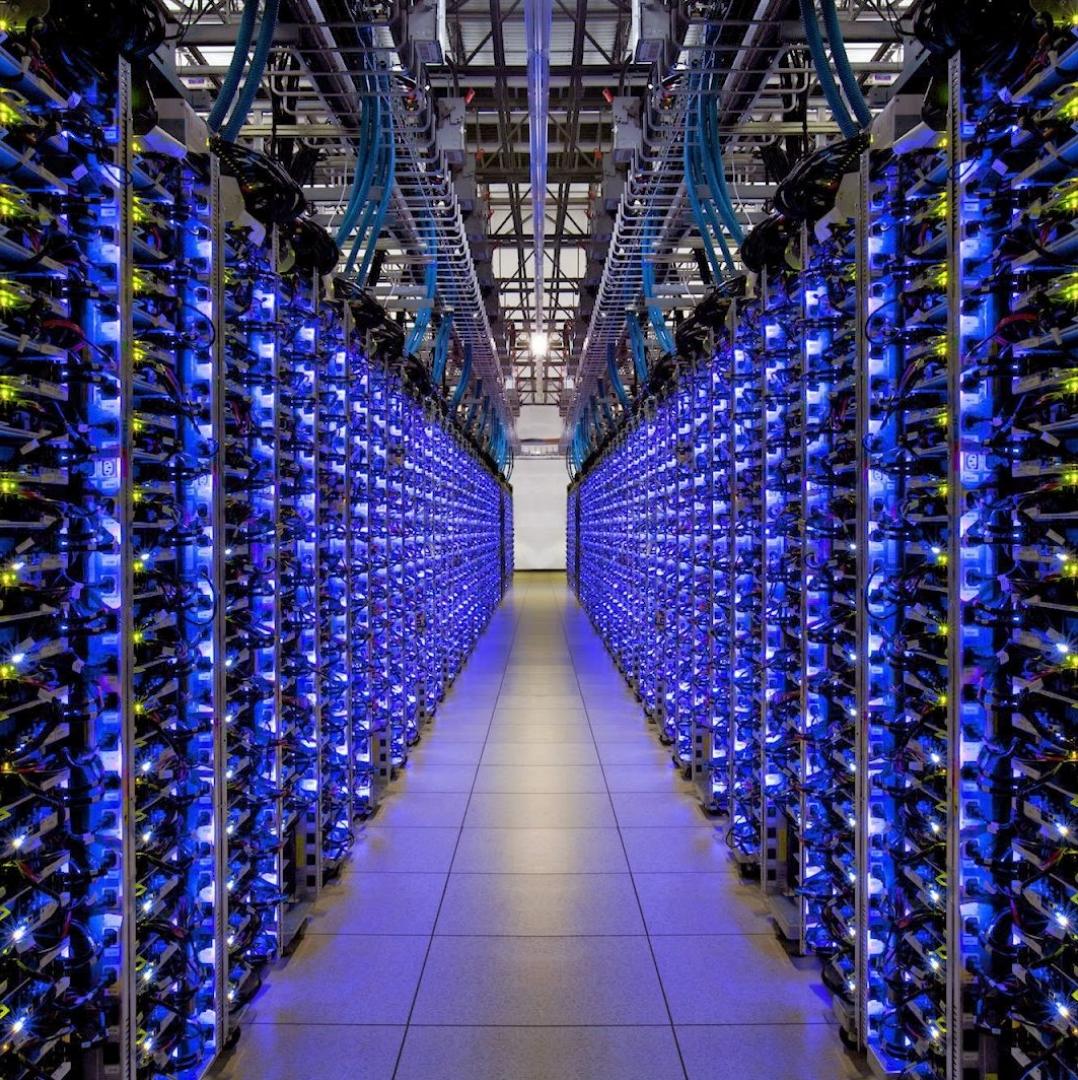
Deploy Across Clouds

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



Data Centres in Ireland

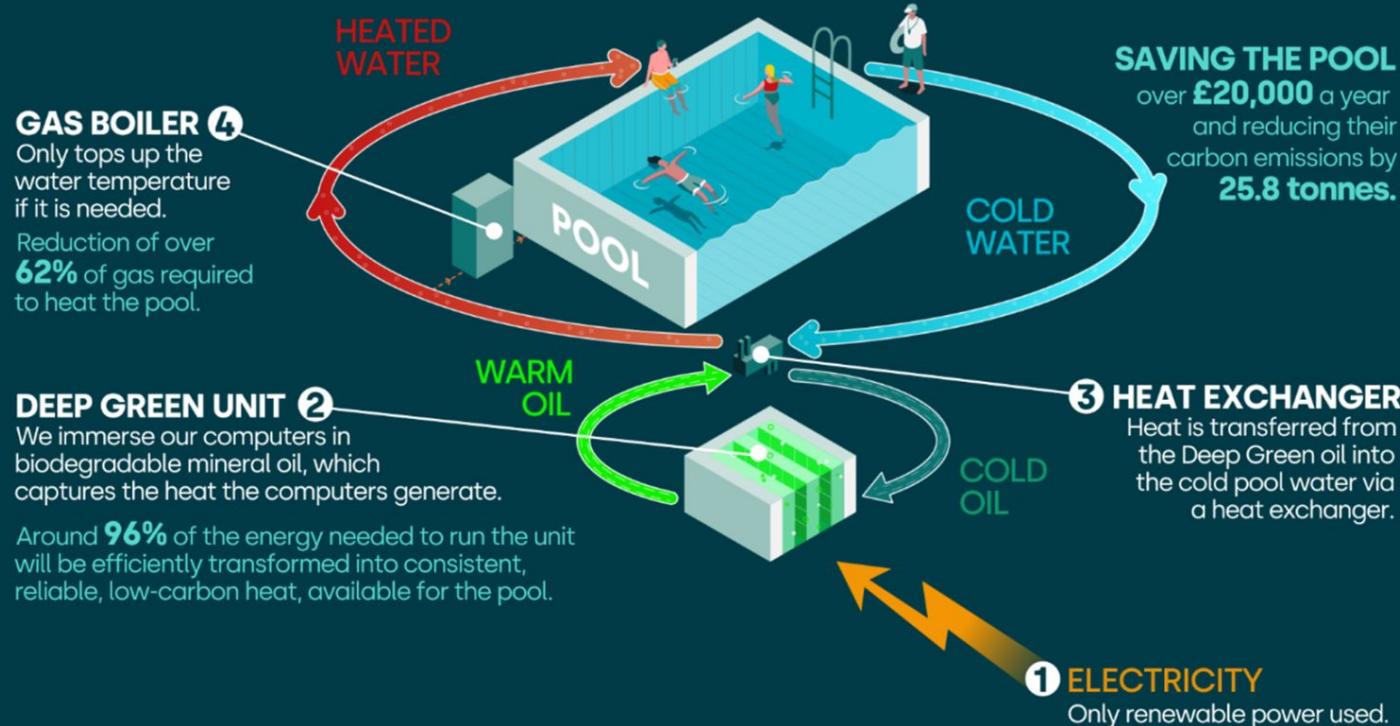
- **82** operational
14 under construction
52 planned [ref](#)
- Consume **21%** of the nation's electricity, that's more than all urban homes (18%), and projected to rise to **33%** within **3** years [ref](#)
- **50.7%** fossil fuels in 2023, including coal, peat and oil, and not including imports [ref](#)





DEEP
GREEN

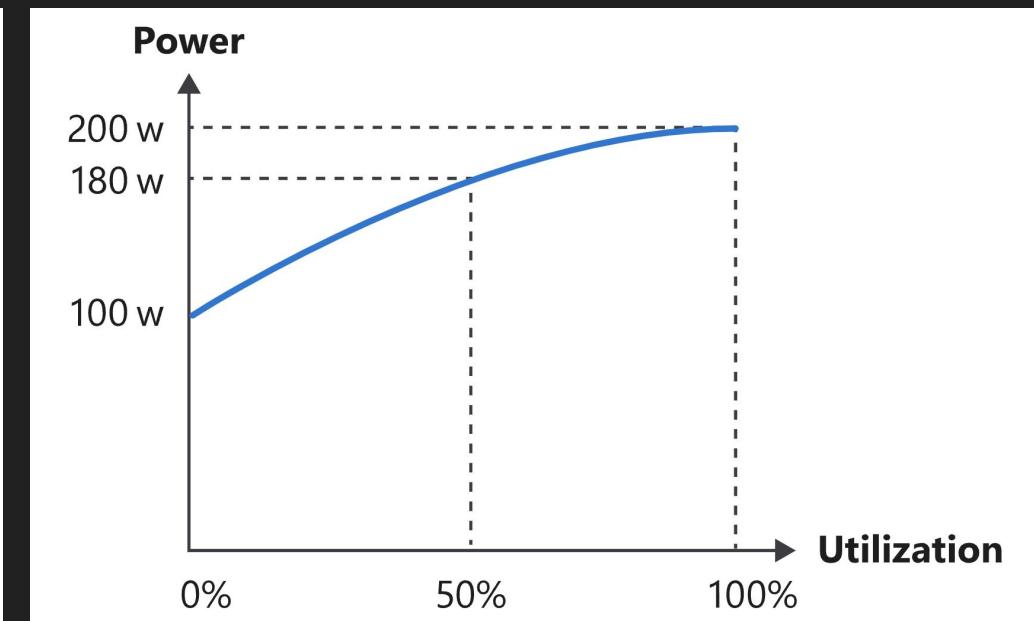
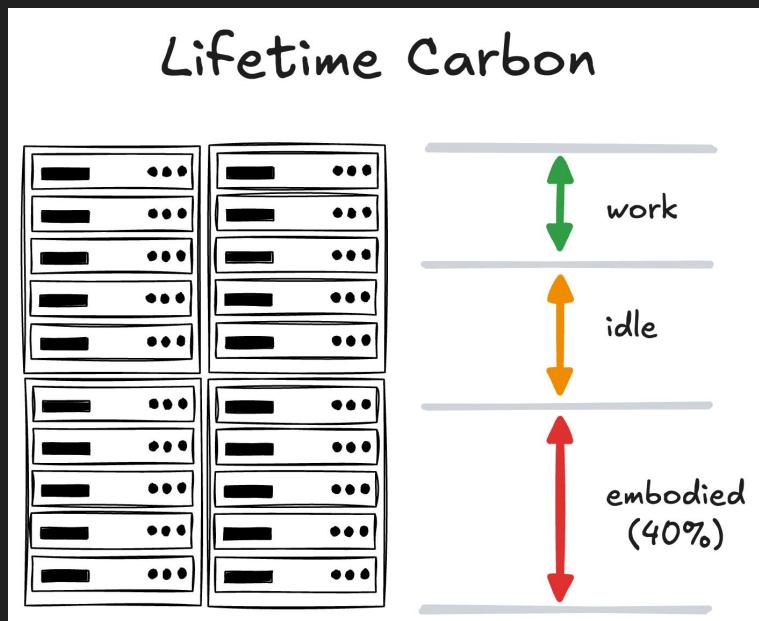
HOW DEEP GREEN HEATS A POOL



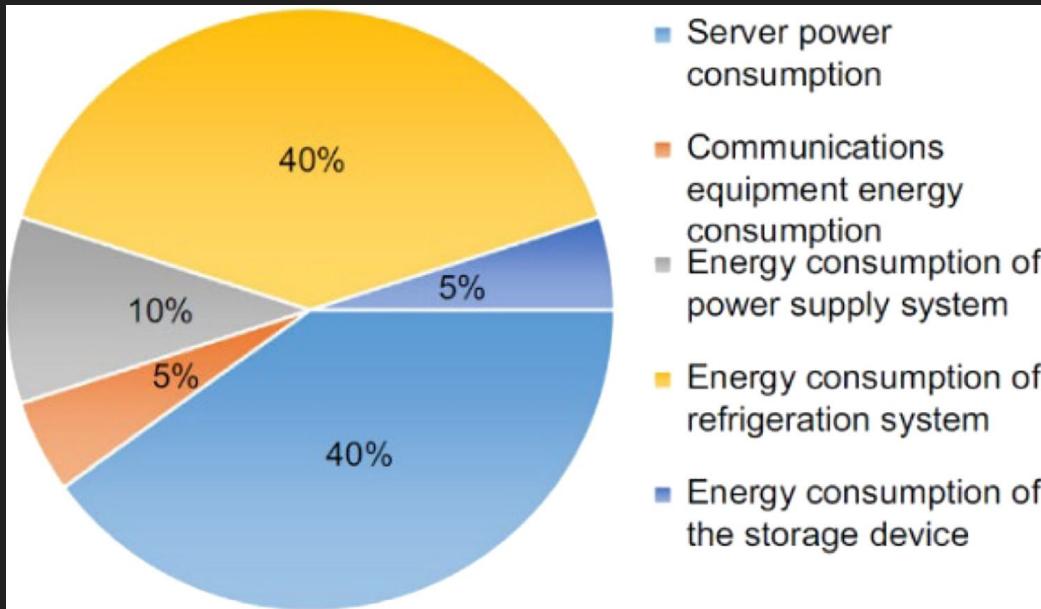
Equinix PA10



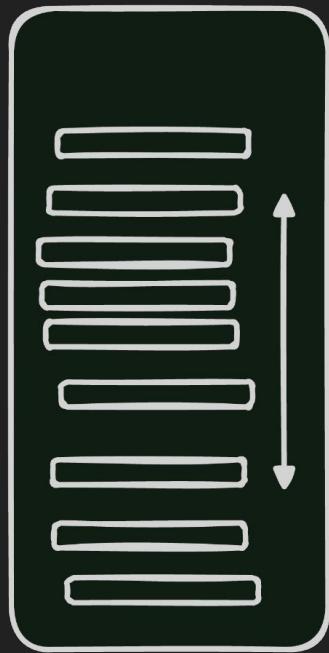
Carbon



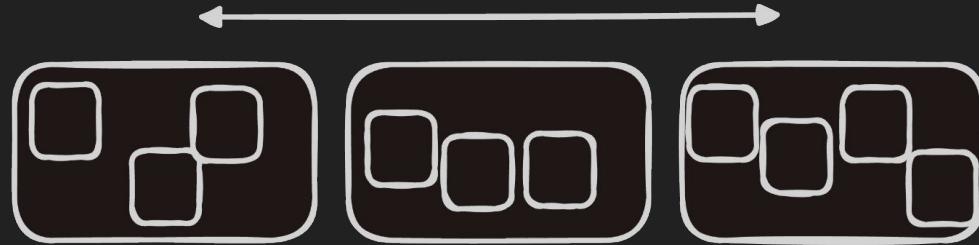
Cooling



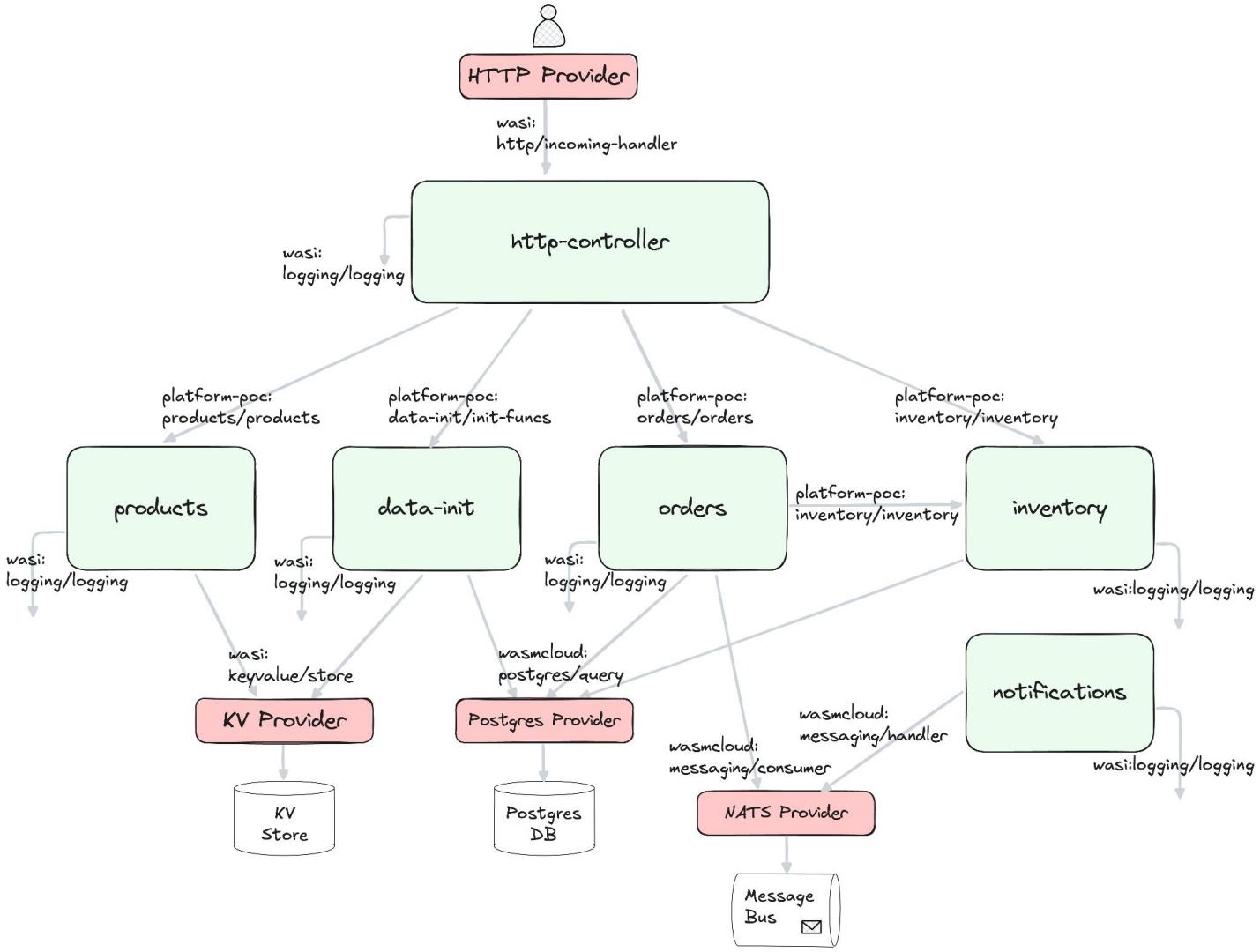
Scaling Vertically and Scale-to-zero



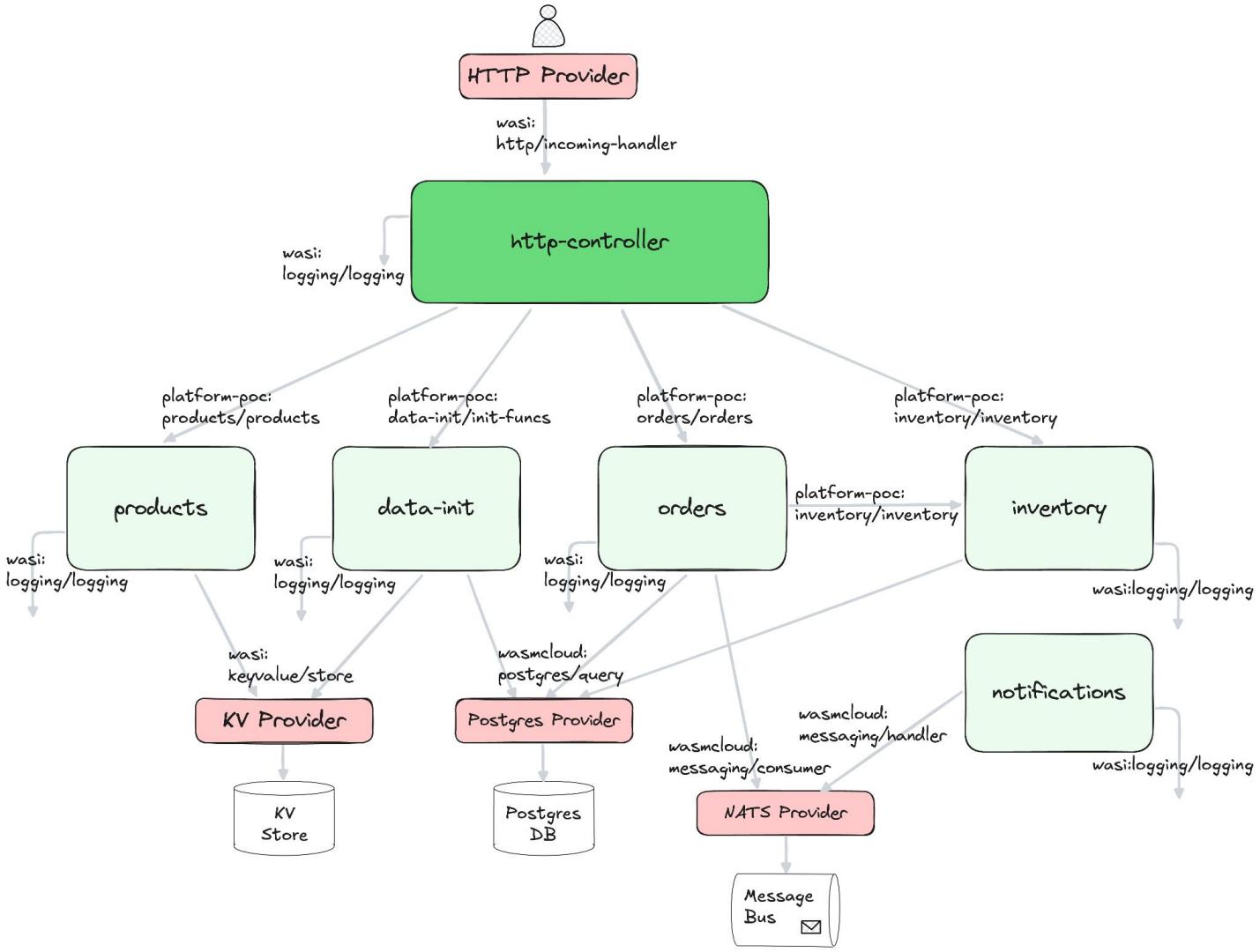
vs



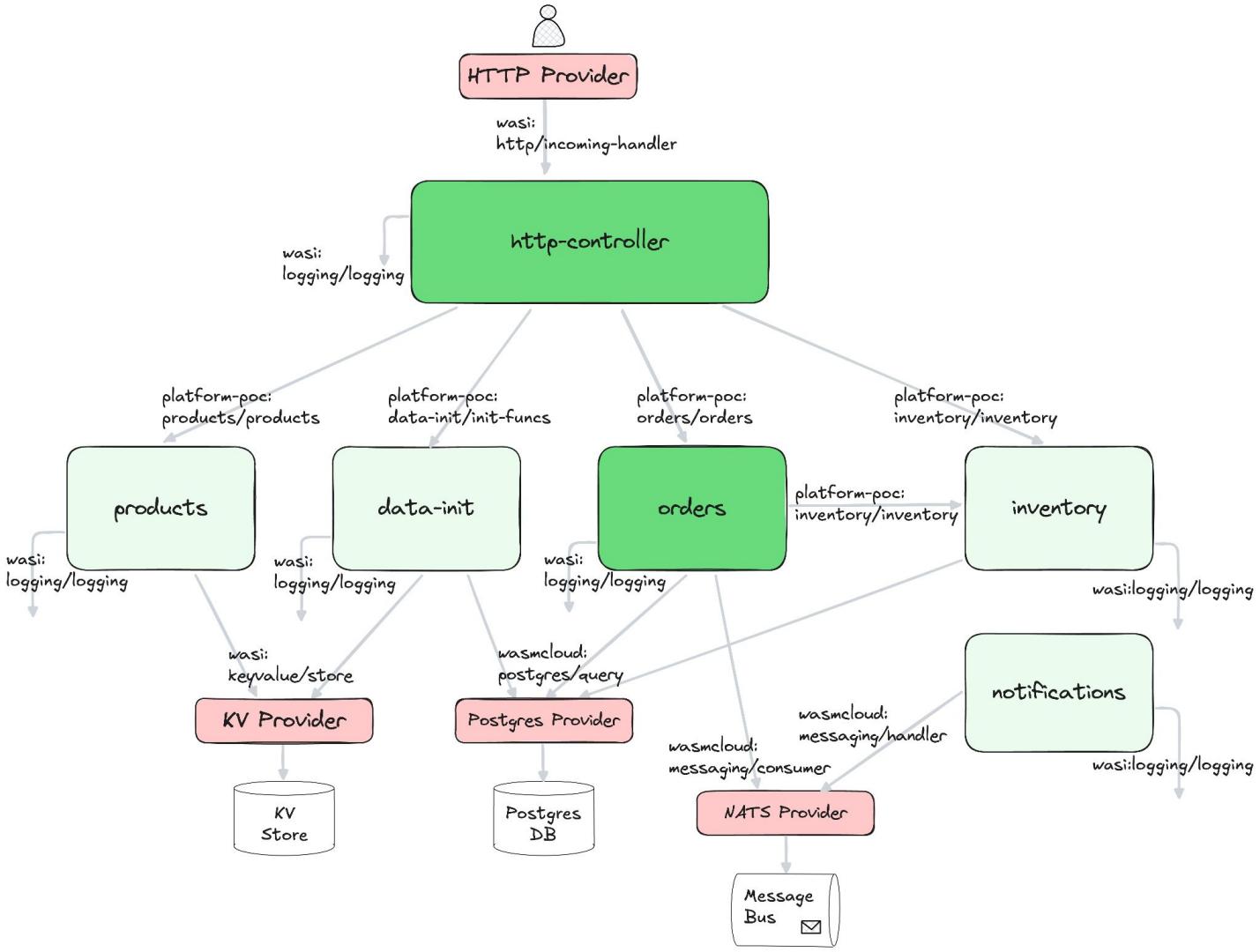
Example



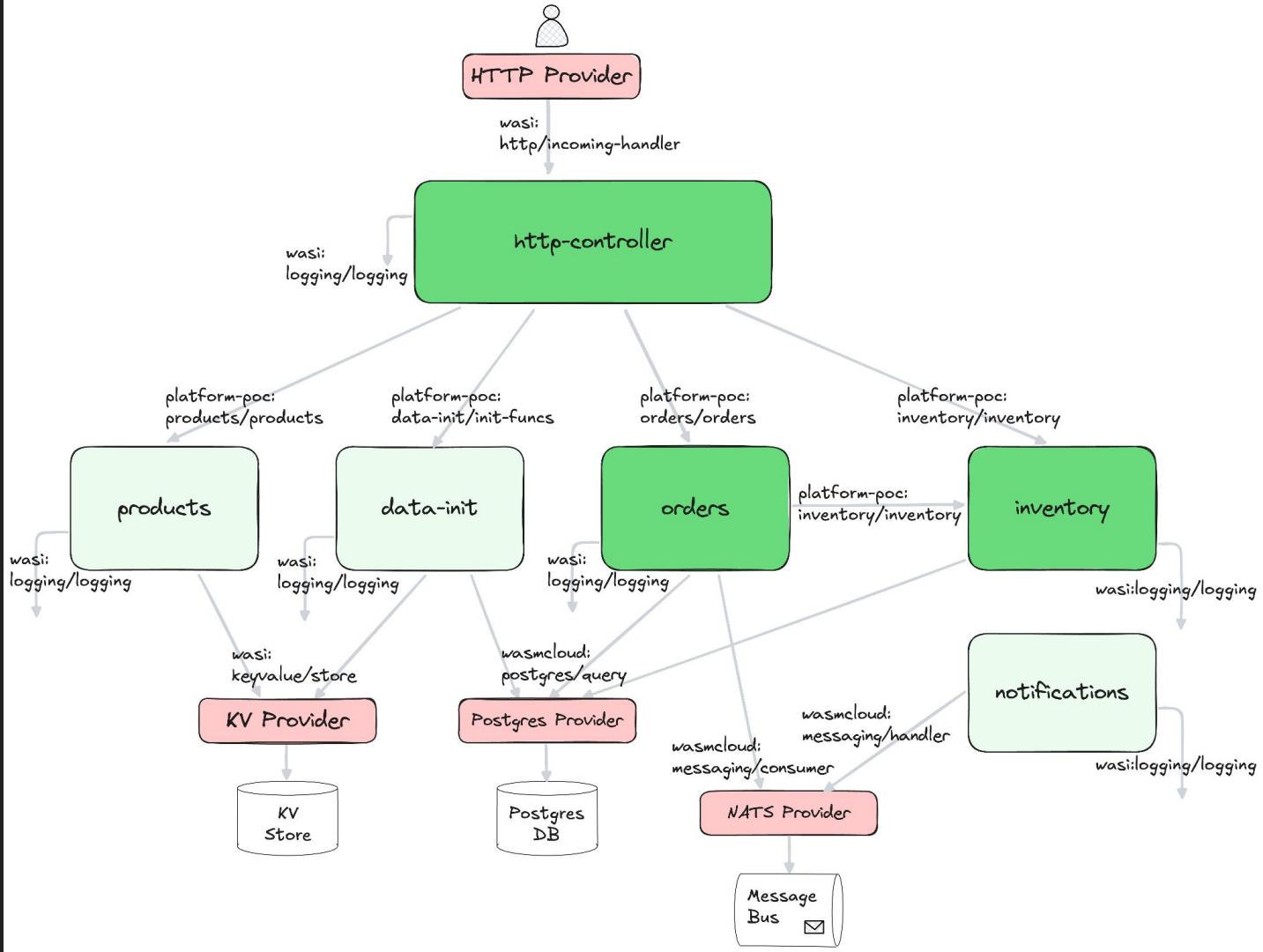
Example



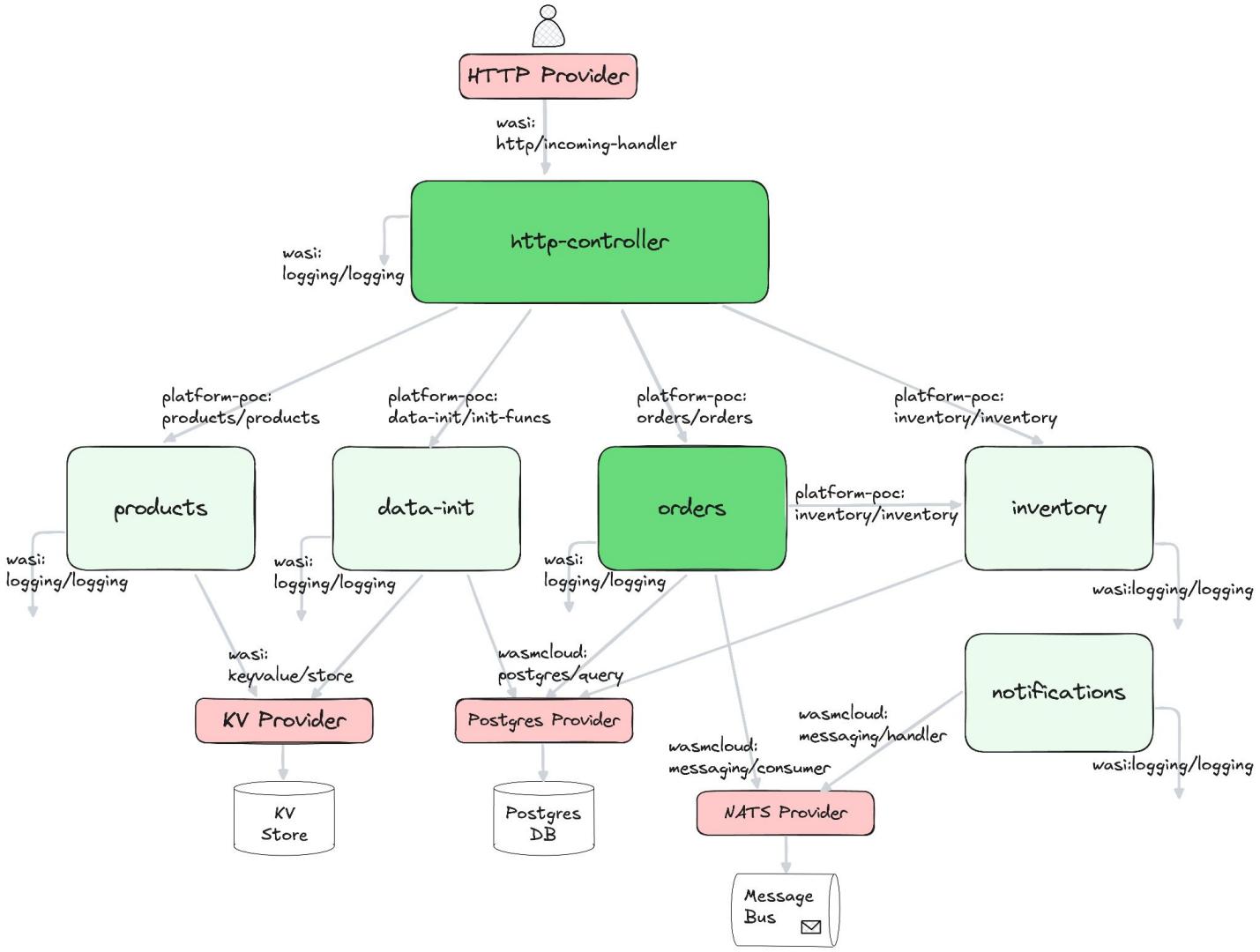
Example



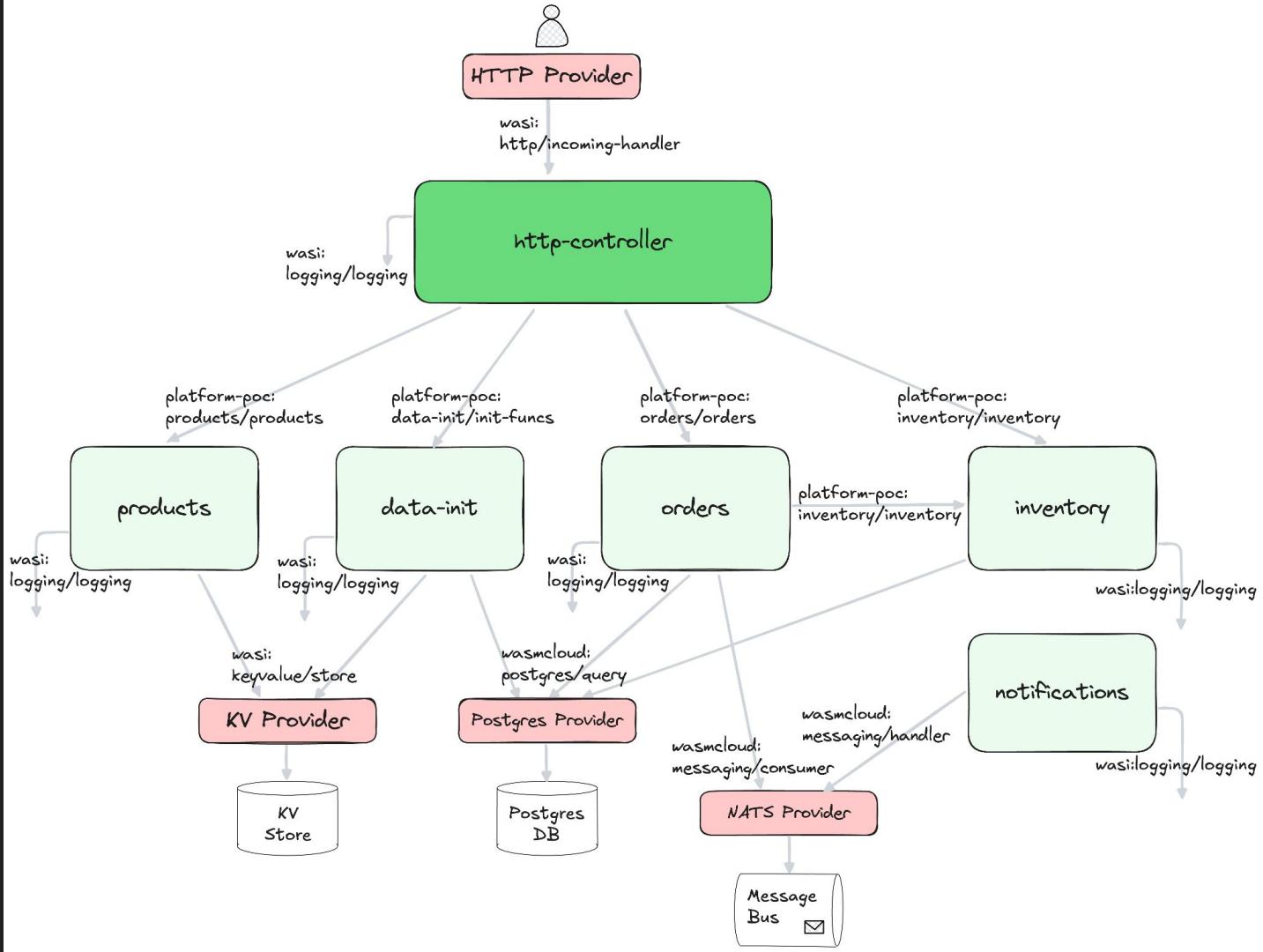
Example



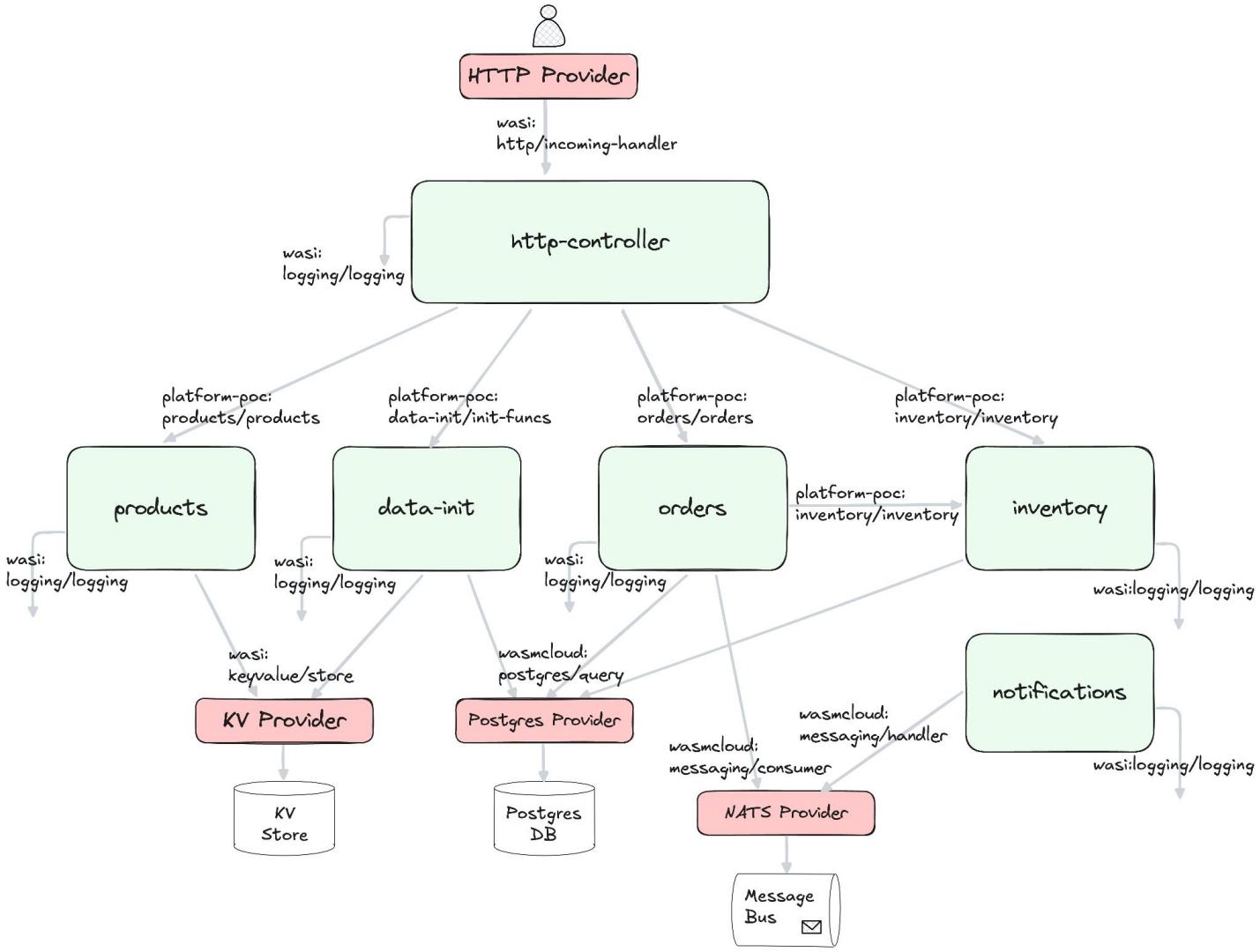
Example



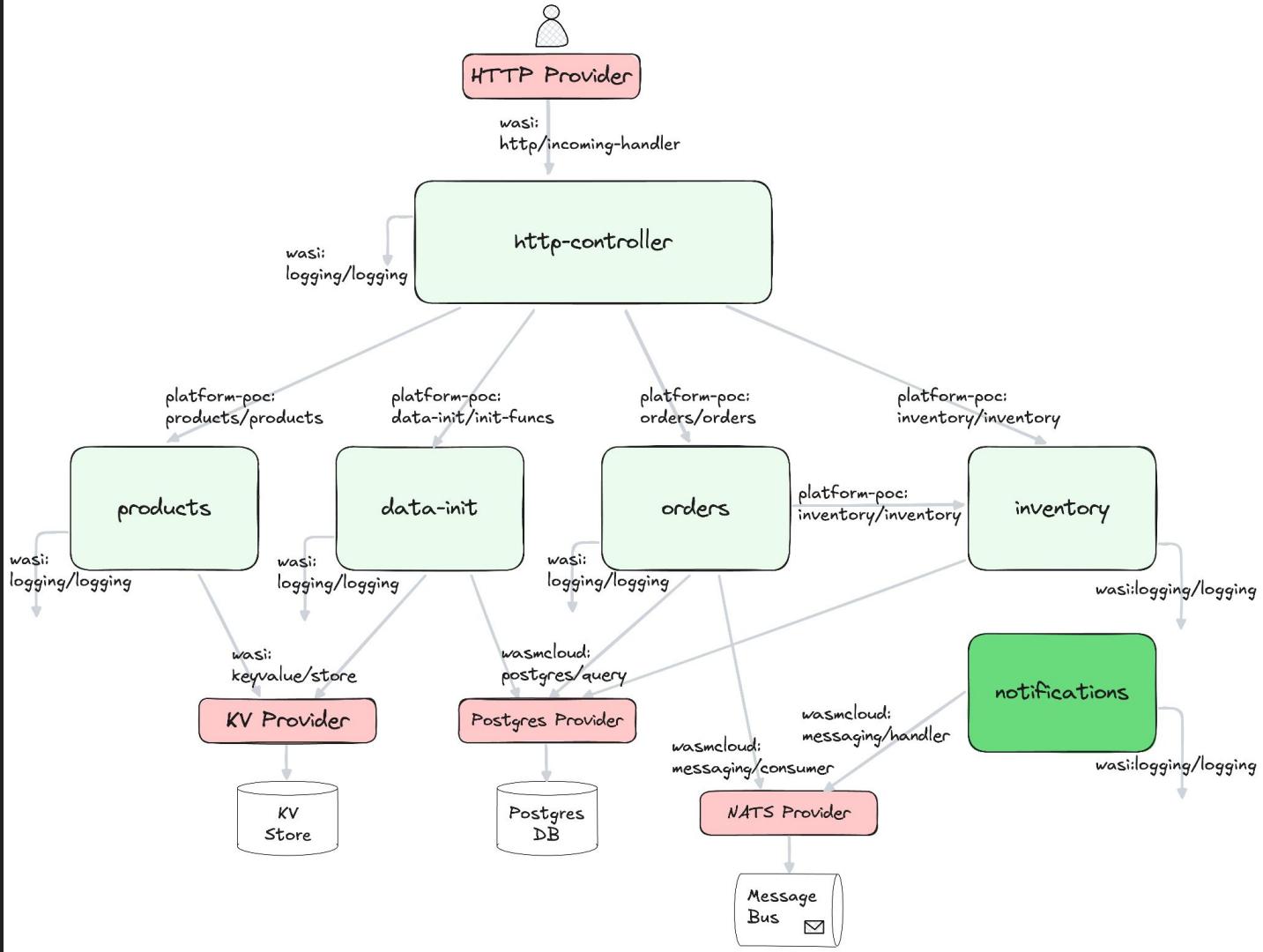
Example



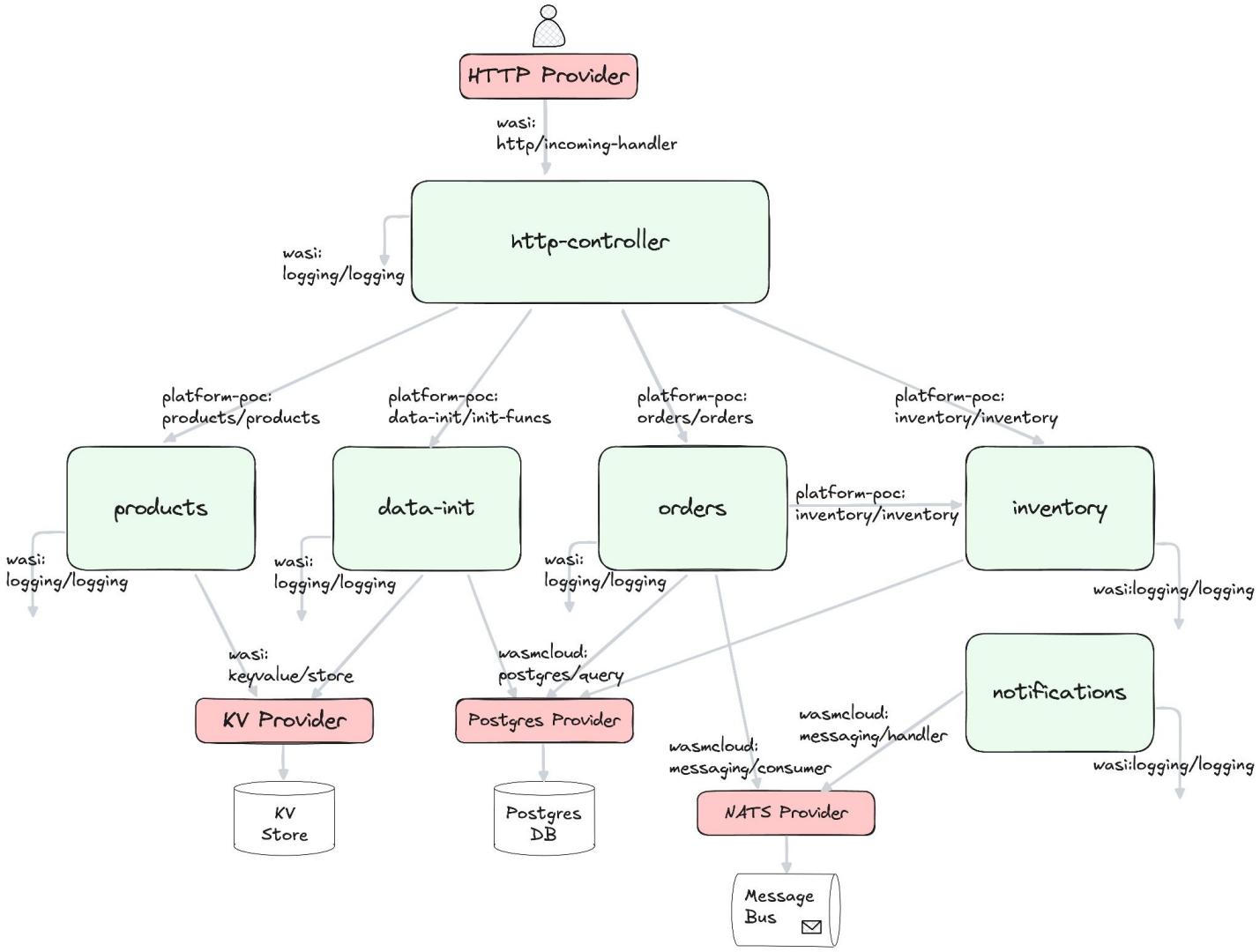
Example



Example



Example



Placing 10K orders

```
→ oha 'http://localhost:8080/orders' \
      -m POST \
      -d '[{"sku": "ENG-V8-500", "price": 1000, "quantity": 1}]' \
      -c 20 \
      -n 10000
```

Summary:

Success rate: 100.00%

Total: 4.3168 secs

Slowest: 0.0484 secs

Fastest: 0.0037 secs

Average: 0.0086 secs

Requests/sec: 2316.5356

Wasm binaries are small

About 1000 times smaller than their equivalent Java Spring Boot container image.

```
92k data_init_s.wasm
543k http_controller_s.wasm
70k inventory_service_s.wasm
89k notification_service_s.wasm
105k orders_service_s.wasm
100k products_service_s.wasm
```

Fermyon.com on Spin

“A fresh page load of Fermyon.com may result in starting, executing, and shutting down 30+ concurrent WebAssembly processes. Yet the result is a page that loads so fast that it scores a 99% on Google’s page speed ranking.”

“Not long ago, we had a very busy day. Our traffic jumped to ten times its normal number of page requests. At a few moments, we were seeing over 2,500 requests coming in at a time (90 concurrent users each fetching around 30 resources). In our model, we run three workers, each on a small-sized VM with 3 CPUs and 300Mb of memory per worker.”

Rust

Safety and speed.

Increased productivity, really.

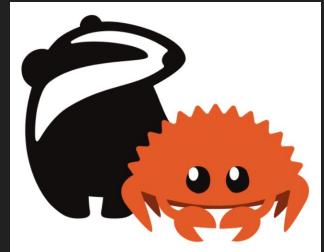
Shift left on quality.

Rust loves WebAssembly:

```
→ cargo build --target wasm32-wasip1
```

Build WebAssembly components on nightly:

```
→ cargo +nightly build --target wasm32-wasip2
```



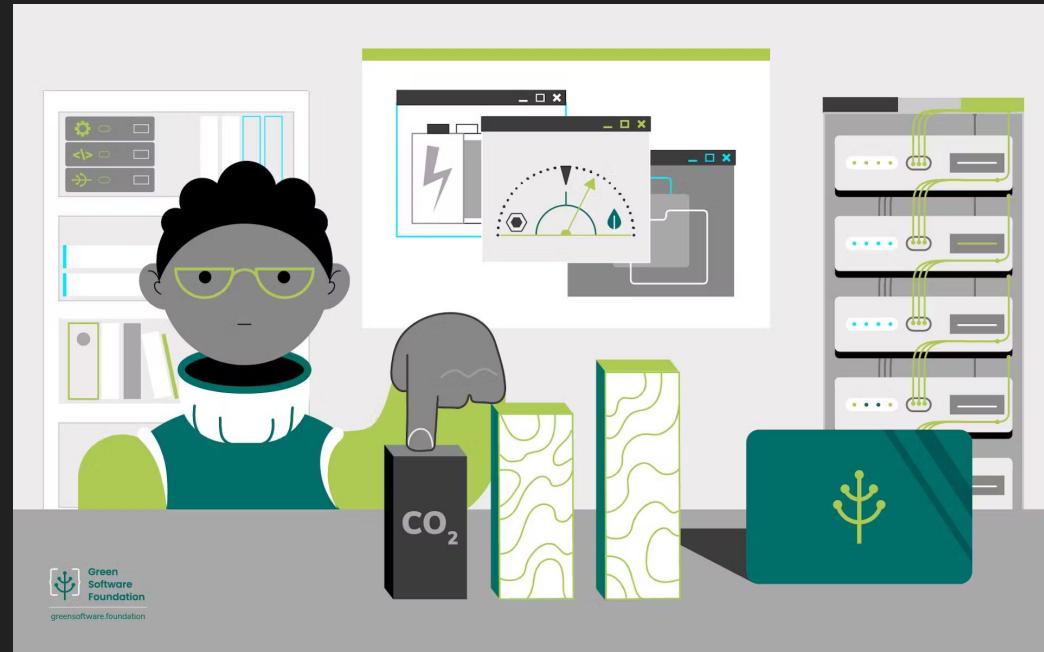
Roadmap

- Production ready — build greenfield scale-to-zero applications today
- Create a hybrid Kubernetes cluster using
 - ◆ wasmCloud Operator
 - ◆ SpinKube
- Gradually migrate, service by service, to new architecture
 - ◆ Remove all side effects, leaving just business logic

Let a decade of containers give way to a decade of WebAssembly components!

Let's build eco-friendly enterprise applications

- The Green Software Foundation [ref](#)
- The Principles of Sustainable Software Engineering [ref](#)



Thank you! 🙏

[slides](#)

Stuart Harris
@stuartharris
stuart.harris@red-badger.com



<https://github.com/rebadger/platform-poc>