A Love Letter to Isolation

Kelly Shortridge @shortridge@hachyderm.io (mastodon) | @swagitda_ (twitter)

Cloud Native Wasm Day 2023

Isolation is the unsung hero of computing.

shortridge@hachyderm.io | @swagitda_

Isolation is a core design element for how we sustain resilience against failures of all kinds

shortridge@hachyderm.io | @swagitda_

But we don’t love isolation the way we should

shortridge@hachyderm.io | @swagitda_

This is a love letter to isolation as we take the next big step in our relationship: Wasm

shortridge@hachyderm.io | @swagitda_


Isolation in
Complex Systems

Isolation is literally the basis of life.

shortridge@hachyderm.io | @swagitda_

“The necessity of thermodynamically isolating a subsystem is an irreducible condition of life.”

– Harold Morowitz

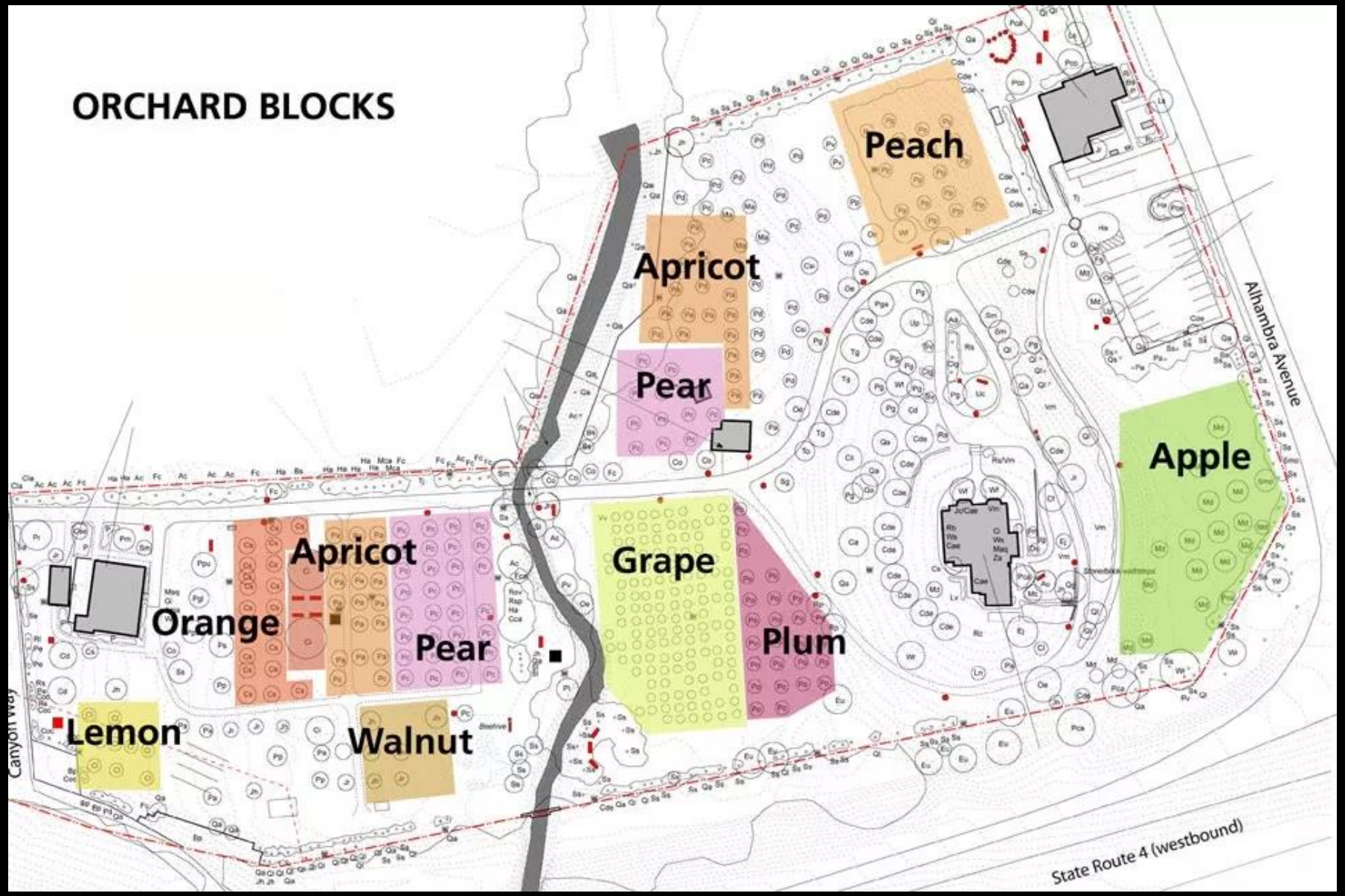
shortridge@hachyderm.io | @swagitda_

Nature was the original architect of nested isolation, long before we made the transistor.

shortridge@hachyderm.io | @swagitda_

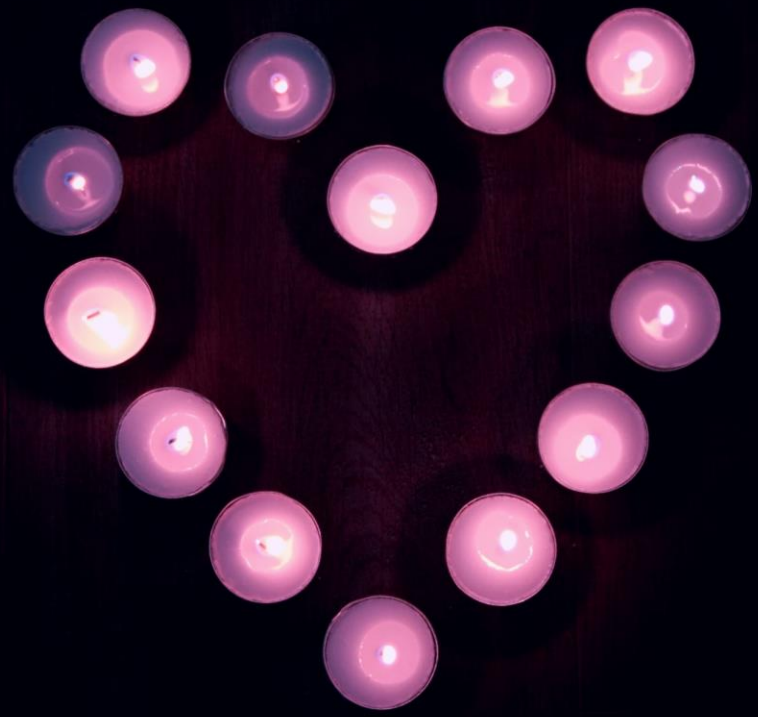
Proteins and genes, cells, organs, ecosystems...

shortridge@hachyderm.io | @swagitda_



shortridge@hachyderm.io | @swagitda_

Modularity



Isolation is the modularity superpower

shortridge@hachyderm.io | @swagitda_

Modularity is a system property reflecting the degree to which components can be decoupled

shortridge@hachyderm.io | @swagitda_

Modularity: allows structurally or functionally distinct parts to retain autonomy during periods of stress & allows for easier recovery from loss

shortridge@hachyderm.io | @swagitda_

Modularity directly enhances evolvability

shortridge@hachyderm.io | @swagitda_

Unless components can fail independently, you don't have modularity in the resilience sense.

shortridge@hachyderm.io | @swagitda_

Modularity enables iterative evolution; the system can “save its work” and still change

shortridge@hachyderm.io | @swagitda_

Modularity allows for basic encapsulation and separation of concerns, which enables isolation

shortridge@hachyderm.io | @swagitda_

Modularity minimizes incident impact

shortridge@hachyderm.io | @swagitda_

Aesop's Fable: "The Oak & the Reeds"

shortridge@hachyderm.io | @swagitda_

Where real world systems bend, software systems tend to break

shortridge@hachyderm.io | @swagitda_

Software isn't forgiving to crashes, exceptions, and attacks – which is why isolation is our hero

shortridge@hachyderm.io | @swagitda_

In software, we're lucky that we can isolate failure to handle unexpected interactions

shortridge@hachyderm.io | @swagitda_

Software Isolation

Isolation makes computers usable

shortridge@hachyderm.io | @swagitda_

Isolation allows us to wrap specific software components in a membrane

shortridge@hachyderm.io | @swagitda_

But adding each layer or dimension of isolation is a hard-fought battle, a great upheaval

shortridge@hachyderm.io | @swagitda_

Process isolation

shortridge@hachyderm.io | @swagitda_

User isolation

shortridge@hachyderm.io | @swagitda_

App / site isolation

shortridge@hachyderm.io | @swagitda_

Network isolation

shortridge@hachyderm.io | @swagitda_

Browser isolation

shortridge@hachyderm.io | @swagitda_

Cryptography

shortridge@hachyderm.io | @swagitda_

All of this innovation supports modularity; we should embrace isolation and deepen it.

shortridge@hachyderm.io | @swagitda_

WebAssembly & the Component Model

Wasm is the latest in a long history of trying to provide convenient and cheap isolation

shortridge@hachyderm.io | @swagitda_

Wasm's existing isolation model lets you build services and programs that are completely isolated from the underlying system

shortridge@hachyderm.io | @swagitda_

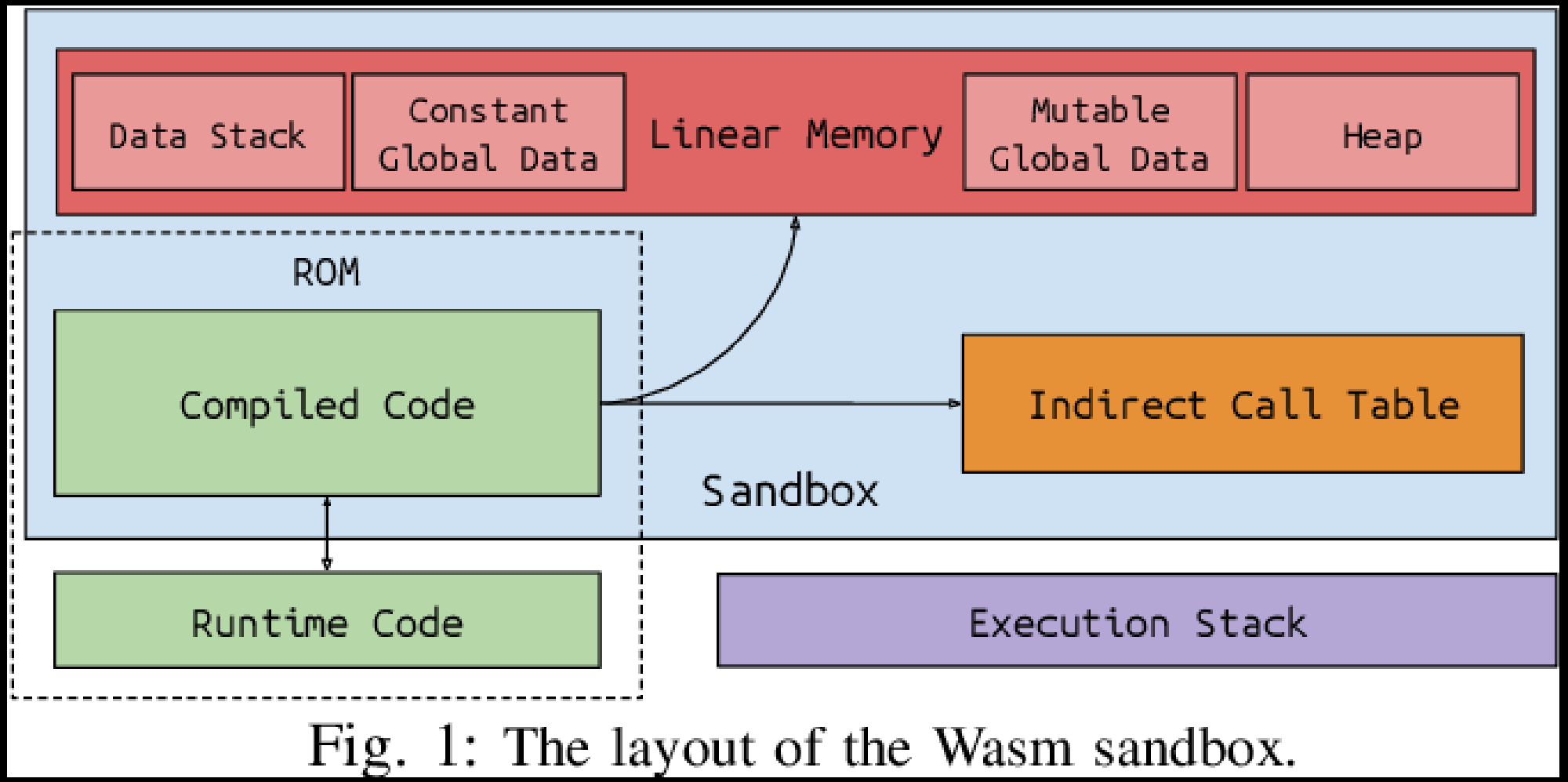


Fig. 1: The layout of the Wasm sandbox.

shortridge@hachyderm.io | @swagitda_

This is an improvement – but hazards remain, and it isn't the easiest thing to set up, either

shortridge@hachyderm.io | @swagitda_

We can do more to solve for a “strong isolation by default” world: the Component Model

shortridge@hachyderm.io | @swagitda_

Devs don’t want to think about isolation; they want some layer underneath to handle it

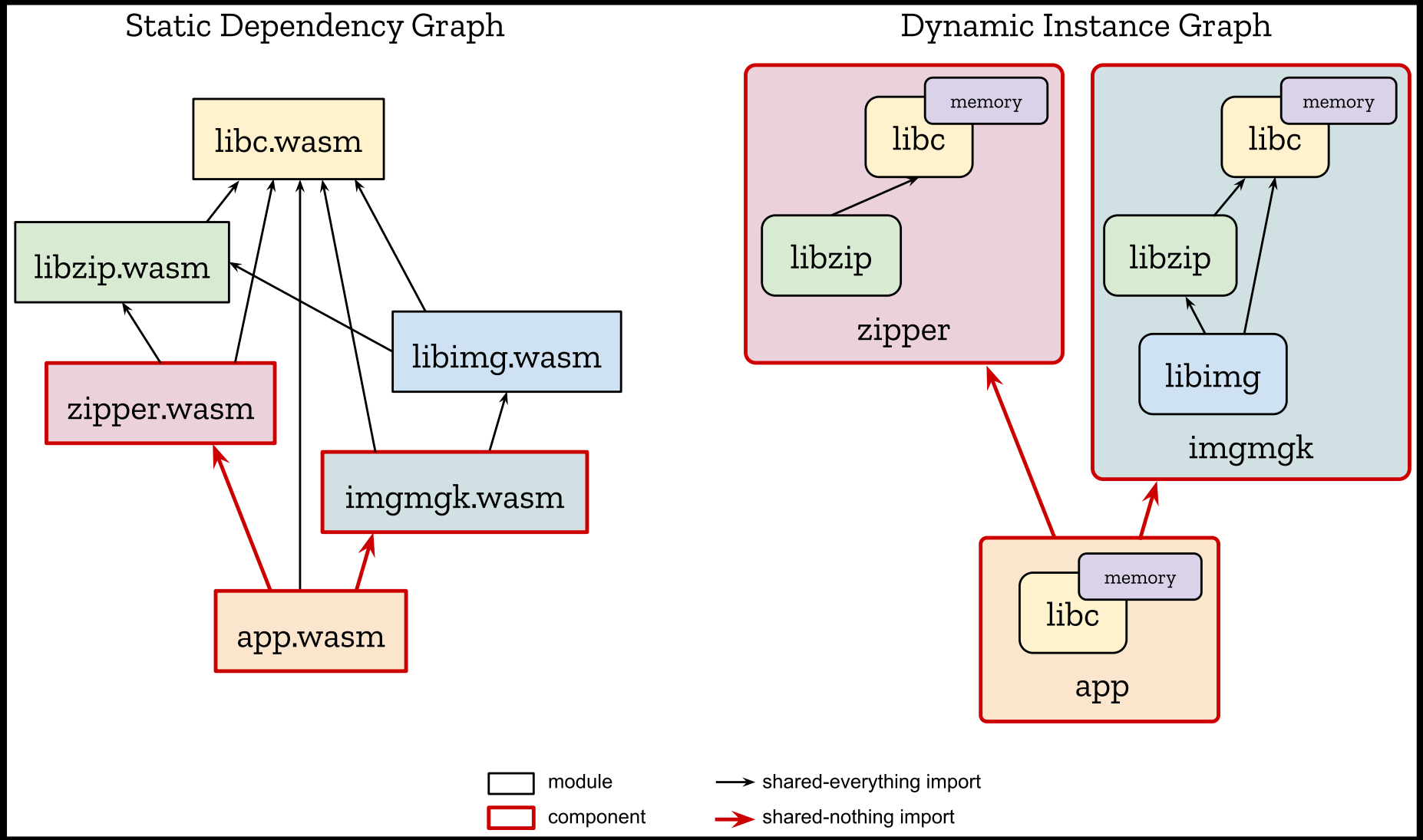
shortridge@hachyderm.io | @swagitda_

The Component Model adds the same safety benefits *between* components (like libraries)

shortridge@hachyderm.io | @swagitda_

Decompose apps into a modular architecture to get loose coupling with near-native speeds

shortridge@hachyderm.io | @swagitda_



shortridge@hachyderm.io | @swagitda_

These boundaries enforce “shared-nothing”

shortridge@hachyderm.io | @swagitda_

WCM gives us the ability to stitch isolated units together in a transparent, performant way

shortridge@hachyderm.io | @swagitda_

WCM is “modularity without microservices”

shortridge@hachyderm.io | @swagitda_

We gain isolation through a simple FFI call

shortridge@hachyderm.io | @swagitda_

Carrot: using libraries from other ecosystems via automatic binding generation

shortridge@hachyderm.io | @swagitda_

The Wasm CM will be the more productive and safe way to build and assemble software

shortridge@hachyderm.io | @swagitda_

The Wasm Component Model offers some of the most powerful security outcomes I've seen

shortridge@hachyderm.io | @swagitda_

It gets us closer to nature's nested isolation

shortridge@hachyderm.io | @swagitda_

Being able to split components arbitrarily and recursively to an infinite depth is basically:

shortridge@hachyderm.io | @swagitda_

shortridge@hachyderm.io | @swagitda_

CM means Wasm runtimes can implement stronger *temporal* isolation, not just logical

shortridge@hachyderm.io | @swagitda_

Create component instances on-demand, destroying them once they complete their task

shortridge@hachyderm.io | @swagitda_

This is HUGE in solving “supply chain security”

shortridge@hachyderm.io | @swagitda_

Can now isolate failures between dependencies

shortridge@hachyderm.io | @swagitda_

Attackers can compromise the dependency's instance state but can't access anything else

shortridge@hachyderm.io | @swagitda_

Reduce “attack surface” even more by defining each dependency-as-components capabilities

shortridge@hachyderm.io | @swagitda_

These properties benefit resilience to all sorts of failures, not just cyberattacks

shortridge@hachyderm.io | @swagitda_

Instead of “turtles all the way down,” with WMC it’s perhaps “isolation all the way down”

shortridge@hachyderm.io | @swagitda_

We can densely pack services together — even malicious ones — into a single process, safely

shortridge@hachyderm.io | @swagitda_

We could implement the “airlock approach” – yeeting a failing component out the airlock

shortridge@hachyderm.io | @swagitda_

Platforms could offer full observability between cross-component messages, too

shortridge@hachyderm.io | @swagitda_

Or migrate components around and scale them independently in respond to demand

shortridge@hachyderm.io | @swagitda_

All of this gets us closer to the ancient dream of software integrated circuits (SICs)

shortridge@hachyderm.io | @swagitda_

The Wasm Component model can revolutionize what resilience means in software

shortridge@hachyderm.io | @swagitda_

But to become *the* future of software dev & delivery, devs must actually use it...

shortridge@hachyderm.io | @swagitda_

DX is a core problem area Wasm must solve

shortridge@hachyderm.io | @swagitda_


Riding off into the sunset

Isolation is the key to modern life, and life itself

shortridge@hachyderm.io | @swagitda_

Software is usable because of isolation

The Wasm Component Model is the next upheaval – a software resilience revolution

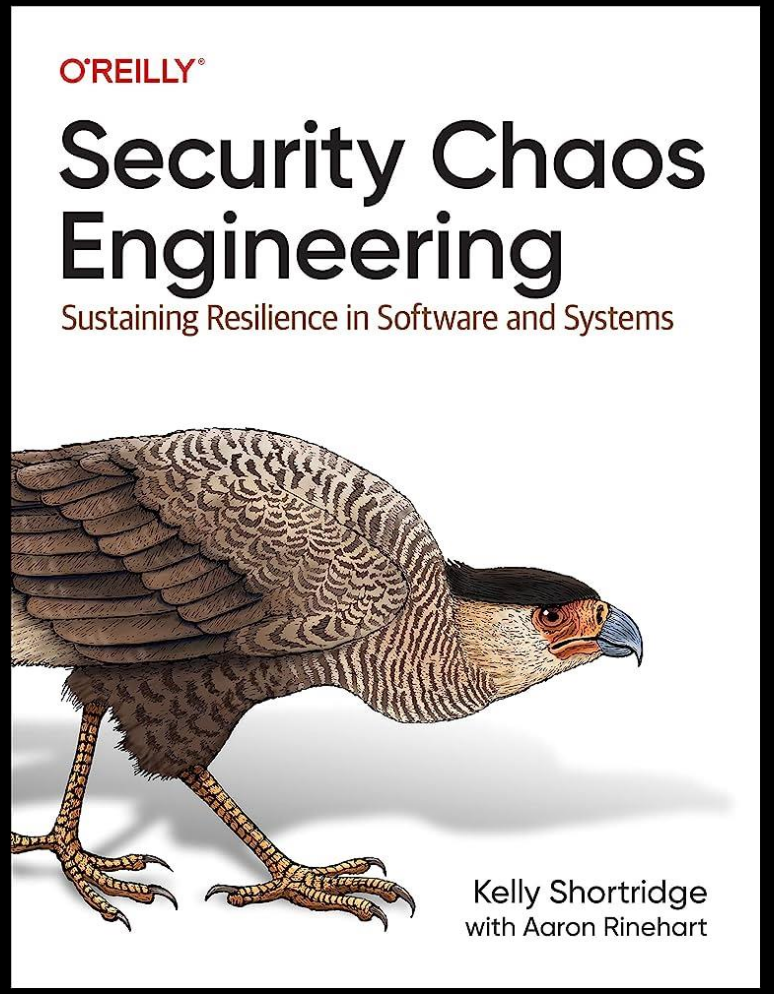
shortridge@hachyderm.io | @swagitda_

Order the book today:

Amazon

Bookshop

& other major retailers



shortridge@hachyderm.io | @swagitda_

/in/kellyshortridge

@swagitda_

shortridge@hachyderm.io

@shortridge.bsky.social

chat@shortridge.io