Using WebAssembly Now: It's Easier Than You Think

Brooks Townsend and Taylor Thomas, Cosmonic
Who are we?

Brooks Townsend
- Lead Software Engineer at Cosmonic
- wasmCloud maintainer
- Serial open source contributor
- Brewer of Elixir, Wasm enjoyer, Rustacean
- Demo enthusiast

Taylor Thomas
- Director of Engineering at Cosmonic
- Rustacean
- Co-creator of Krustlet and Bindle
- Open Source Maintainer
- Emeritus Helm Maintainer
Agenda

- What is a Wasm?
- Where does it fit in?
- Demos galore ✨
- What can I do now?
- How to get involved
Neither Web, nor Assembly

- **Open W3C Standard**
  Open and widely supported standard

- **Safe & Secure**
  Deny by default secure sandbox, featuring capability driven permissions

- **Efficient and fast**
  Small size and near-native execution speed

- **Polyglot**
  Choice of deployment language means ability to reuse existing libraries

- **Portable**
  WebAssembly runs in all major browsers
**Good vs Bad**

- General server side computing
  - Microservices
  - Functions
- Runtime optimization
- Constrained devices (“the edge”)
- Plugins
- Libraries
- Browser

- Networking is still difficult
- Things are now in the “fast moving” stage
- Slower than native code
- Toolchains aren’t there for some languages
  - No lift and shift with Wasm
- Some domains aren’t really a good fit (yet)
  - Performance tuned applications
  - Nginx, Redis, MySQL, etc.
Why wasmCloud?

- If Wasm is so good, what do we need wasmCloud for?
- Runtime vs application runtime
WebAssembly Host Runtime

- Portable
- Secure
- Small
- Fast
- Language agnostic
wasmCloud Application Runtime

- Run and orchestrate Wasm
- Secure access to capabilities
- Horizontally and vertically scalable, stateless actors
- Manage networking, failover, request routing
Actors

- Implement only your business logic
- Stateless and reactive
- Tiny footprint, portable & scalable
- Easy to develop & low boilerplate
Capabilities

- Contract driven design
- Choose implementation at runtime, hot swap
- Decoupled libraries from business logic
Lattice Network

- Flattened topology, enables flexible dynamic deployments
- Seamlessly connected, “it just works”
- NATS (not NATs)
## Application Development Stacks

### FORMAT

- **PC**
  - **EXECUTION**:
    - Image (Datacenter)
  - **Dev Responsibility**: Full
  - **Abstraction**: -
  - **Compatibility**: All
  - **Size**: Large
  - **Portability**: -
  - **Security**: System
  - **Location**: On Prem & Co-location

- **VM**
  - **EXECUTION**:
    - VM (Public Cloud)
  - **Dev Responsibility**: OS, App, Lib
  - **Abstraction**: CPU
  - **Compatibility**: Most
  - **Size**: Med
  - **Portability**: Low
  - **Security**: OS
  - **Location**: Proprietary Cloud & Edge

- **CONTAINER**
  - **EXECUTION**:
    - Container (Docker)
  - **Dev Responsibility**: App, Lib
  - **Abstraction**: Linux Kernel
  - **Compatibility**: Most
  - **Size**: Small
  - **Portability**: Med (CPU, Linux)
  - **Security**: Process Boundary
  - **Location**: Dev, Edge, Cloud, K8s

- **K8S**
  - **EXECUTION**:
    - Containers (K8s / Cloud)
  - **Dev Responsibility**: App, Lib
  - **Abstraction**: K8s
  - **Compatibility**: Most
  - **Size**: Small
  - **Portability**: Med (CPU, Linux)
  - **Security**: Process Boundary
  - **Location**: Dev, Edge, Cloud, K8s

- **WASM**
  - **EXECUTION**:
    - WASM (Everywhere)
  - **Dev Responsibility**: Wasm
  - **Abstraction**: Secure Sandbox
  - **Compatibility**: Most
  - **Size**: Tiny
  - **Portability**: High
  - **Security**: Capability
  - **Location**: Dev, App, Edge, Cloud, K8s, Browser, Devices

- **WASM CLOUD**
  - **EXECUTION**:
    - Distributed WASM (Everywhere)
  - **Dev Responsibility**: Business Logic
  - **Abstraction**: Sandbox + Capabilities
  - **Compatibility**: Most
  - **Size**: Minuscule
  - **Portability**: Highest
  - **Security**: Actor
  - **Location**: Dev, App, Edge, Cloud, K8s, Browser, Devices

### Legend:
- **Developer Provided**
- **Service Provided**
- **Flexible**
DEMO TIME
KVCounter

Key:
- Actor
- Capability Provider
- External Service

Web HTTP Request

HTTP Server

wasmcloud:httpserver

KVCounter

wasmcloud: keyValue

KeyValue Store
Components and the future

**Current**

- Lattice
- wasmbus-rpc
- kvcounter.wasm
- wasmbus-rpc

- Lattice

- wasmCloud Host

**Future**

- Lattice
  - httpserver.wasm
  - kvcounter.wasm
  - value.wasm
  - wasmbus_sender.wasm
  - compiled.wasm

- Lattice

- wasmCloud Host

- User Managed Code
TODO App

Key:

Actor

Capability Provider

External Service

Link

TODO App

Web HTTP Request

HTTP Server

wasmcloud:httpserver

TODO API

wasmcloud:messaging

NATS Messaging

wasmcloud:messaging

Dist-KV

wasmcloud:KeyValue

KeyValue Store
TODO App Architecture
What could you do now?

1. Basic data/image/etc. processing
2. One small part of a service
   - Runs smaller and cheaper
3. A full stateful application
   - Use the various providers to connect to the data sources you need
References

- https://slack.wasmcloud.com/
- https://github.com/wasmCloud/wasmCloud

Additional resources
- https://github.com/wasmCloud/capability-providers
- https://github.com/wasmCloud/interfaces
Join our community Slack and check out our GitHub!

https://slack.wasmcloud.com

https://github.com/wasmCloud/wasmCloud