KubeCon CloudNativeCon Europe 2026

Streamlining WebAssembly builds with Bunny

Charalampos Mainas, Anastassions Nanos

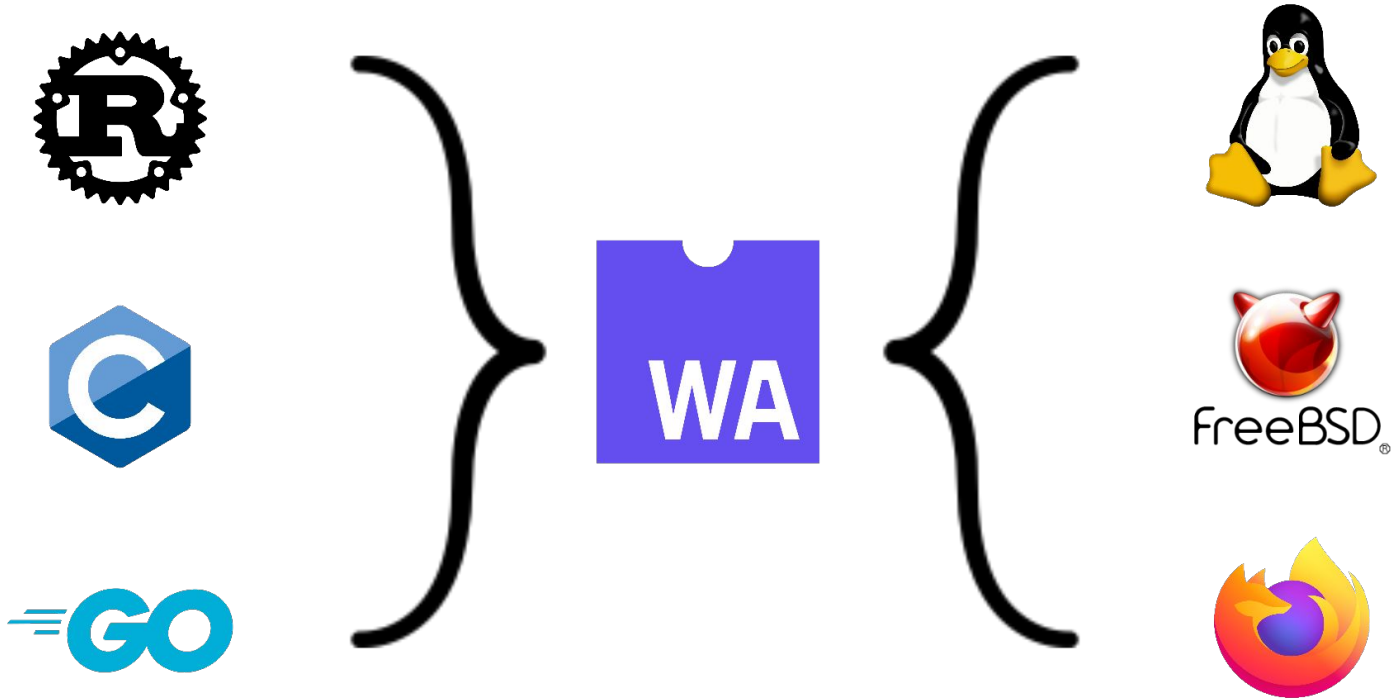
About Us

- Young SME (inc. 2020) doing research in virtualization systems
- Involved in Research/Commercial and Open Source projects
- Focus on systems software
  - Hypervisors and container runtimes
  - Optimize application execution
  - Bring cloud-native concepts to Edge / Far-Edge devices

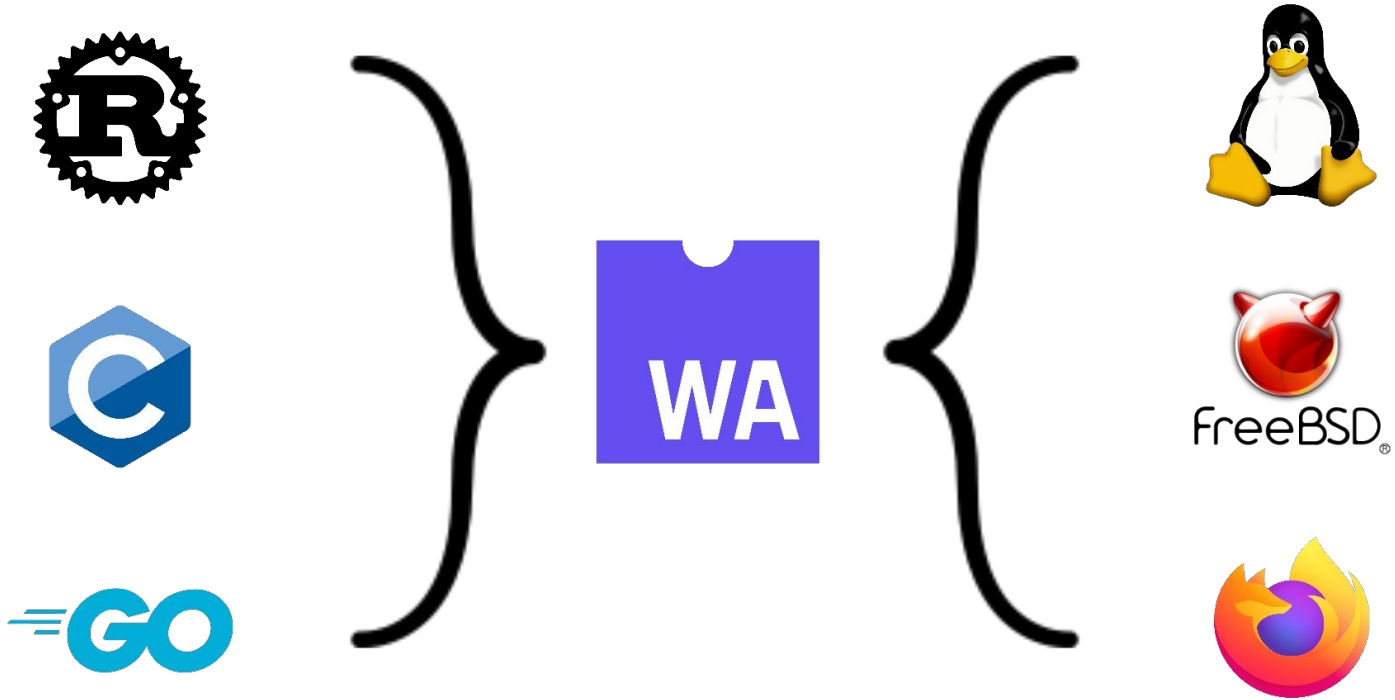




The Art of WebAssembly

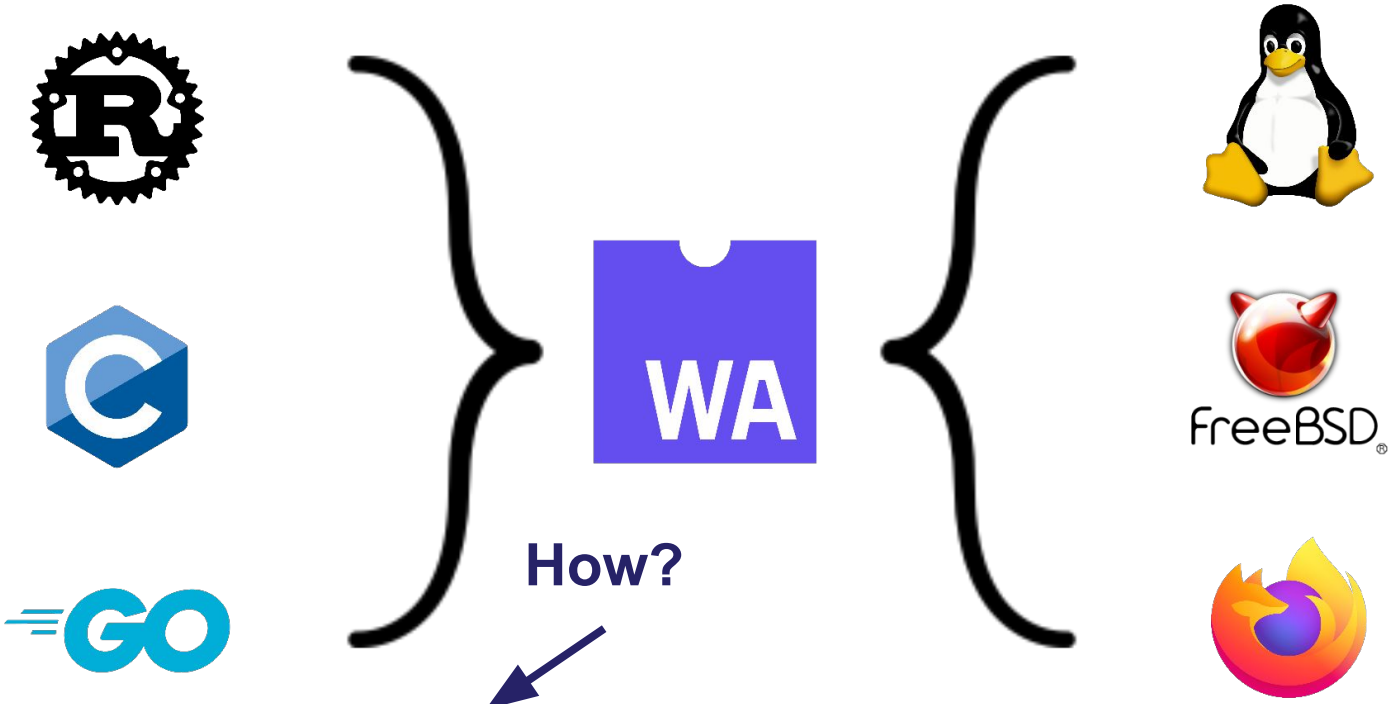


The Art of WebAssembly



Compile once, run everywhere

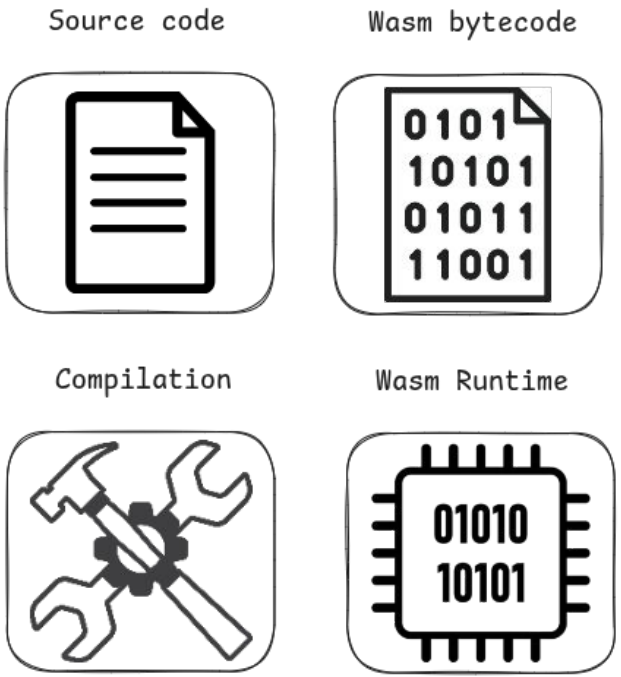
The Art of WebAssembly



A closer look at Wasm

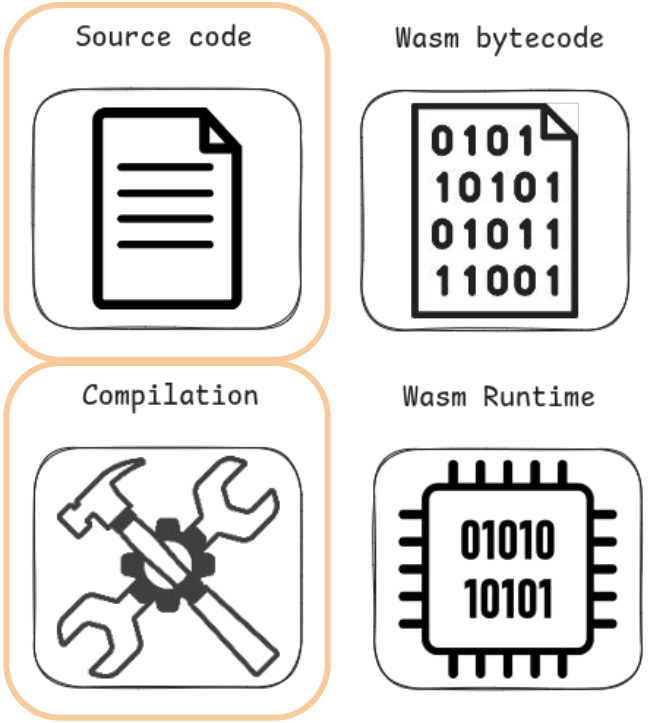
- WebAssembly is a binary instruction format
  - A low-level representation of source code
  - AOT/JIT compilation

- WebAssembly runs on top of a runtime
  - Implementation of Wasm's VM
  - Enforcement of WebAssembly semantics



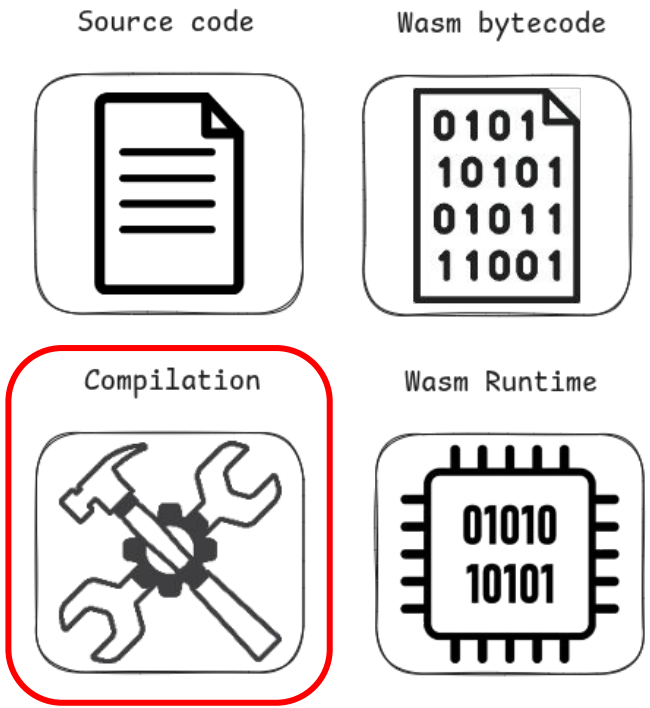
The Wasm ecosystem

- Every language has a different toolchain
  - Each language has different maturity level



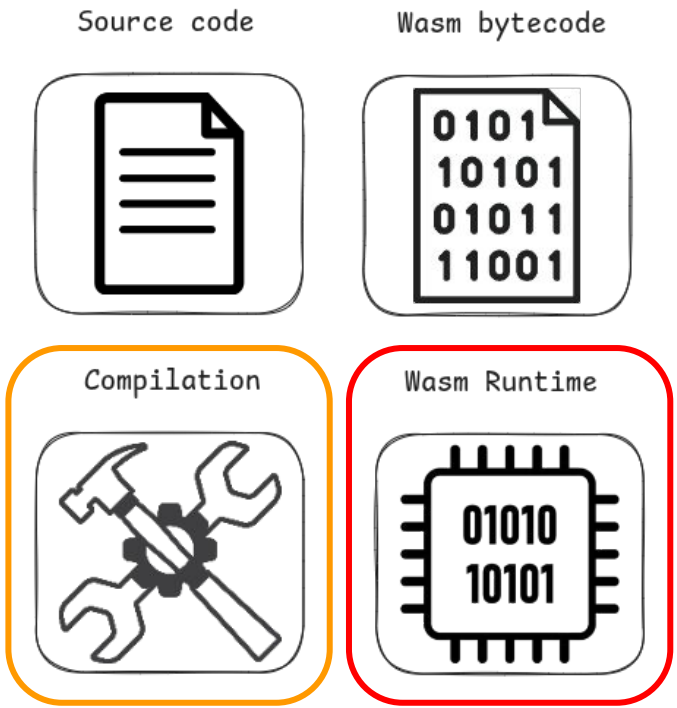
The Wasm ecosystem

- Every language has a different toolchain
  - Each language has different maturity level
- Multiple toolchains for a single language
  - Toolchains for different targets (runtimes)



The Wasm ecosystem

- Every language has a different toolchain
  - Each language has different maturity level
- Multiple toolchains for a single language
  - Toolchains for different targets (runtimes)
- Every runtime has different feature support
  - And specific extentions



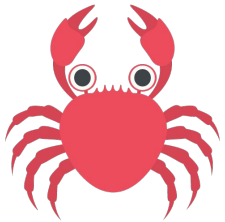
KubeCon CloudNativeCon Europe 2026

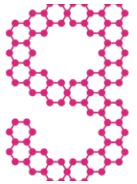
Memories…

Library Operating Systems

- Each OS component as a standalone independent library
  - Define dependencies between components
- Modular, creation of tailored OS images for a single purpose
  - Choose only the components an application requires
- Application compiles and links against libOS
  - Cross compilation of app with libOS tools
  - Static linking of app and libOS objects



Occlum

GRAMINE

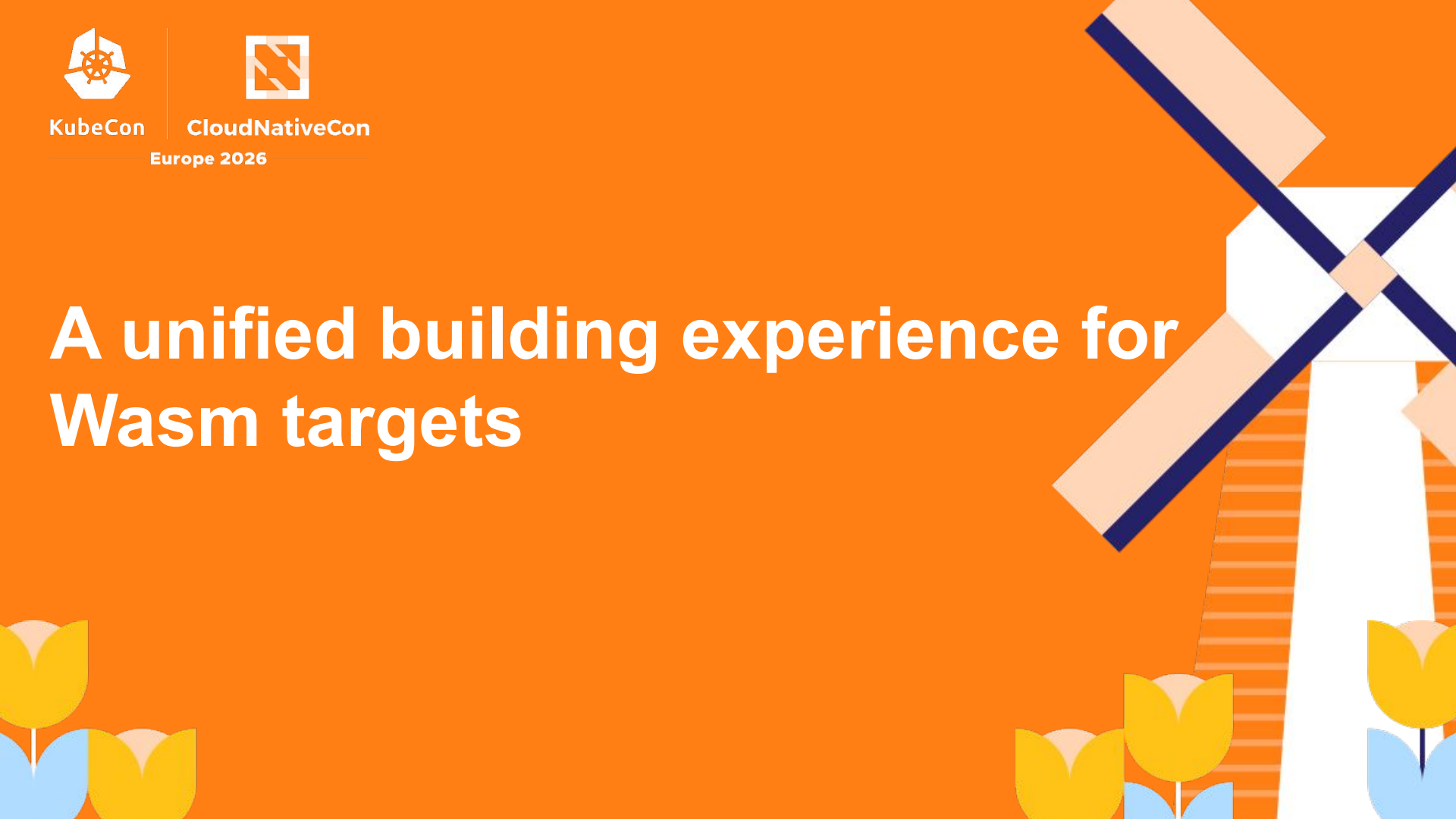
Shared Challenges in Building

LibOS

- Cross compilation required
- Custom toolchains
- Specific build processes

Wasm

- Cross compilation required
- Custom toolchains
- Specific build processes



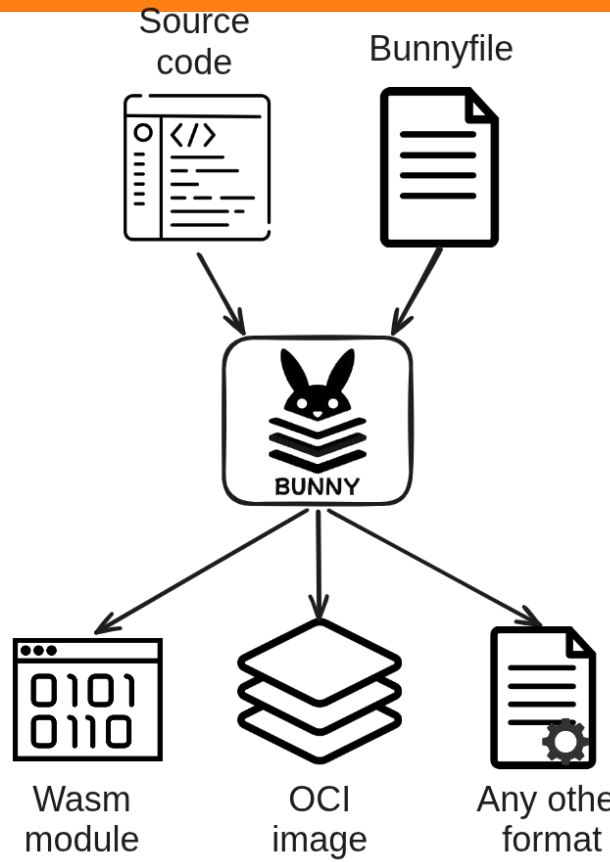
A unified building experience for Wasm targets

Bunny: Building Wasm with ease

- Configure once build against multiple toolchains and runtimes
  - Unified interface for all toolchains and target runtimes
- Simplify the process of building an app with WASM
  - Abstract away the diversity and complexity of each toolstack
- No dependency hell
  - Automatically identify and handle dependencies

Bunny: Building Wasm with ease

- A container-like experience
  - Similar workflow with containers building
- Generate various outputs
  - Wasm bytecode, OCI images, or other formats
- A layered building process
  - Reuse previously built components



Bunny: Interface

- A Dockerfile-like experience tailored to Wasm
  - Yaml file with the building information
- Target platforms
  - Runtime, toolchains, versions, etc.
- Application building information
  - Source code, dependencies, etc.

```
platforms:
  monitor: wasmtime
  framework: clang

app:
  name: hello-world
  source:local
  language: C
  libraries: ["math"]
  flags: -DDEBUG
```

Bunny: Building steps

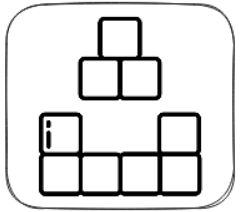
- Analyze user input
  - Identify framework, dependencies, app language

Bunny: Building steps

KubeCon CloudNativeCon Europe 2026

- Analyze user input
  - Identify framework, dependencies, app language

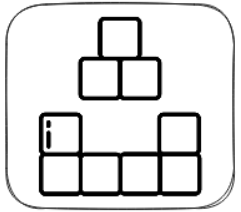
- Fetch building layers
  - Building tools and dependency layers (framework, libs)



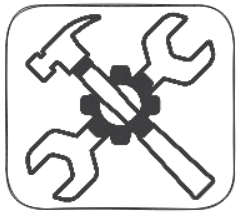
Bunny: Building steps

- Analyze user input
  - Identify framework, dependencies, app language

- Fetch building layers
  - Building tools and dependency layers (framework, libs)



- Build Wasm
  - Build user code and produce artifacts



Bunny: Layers

Tooling Layers

Dedicated environments for **toolchains** and **libraries**. Essential for the build process.

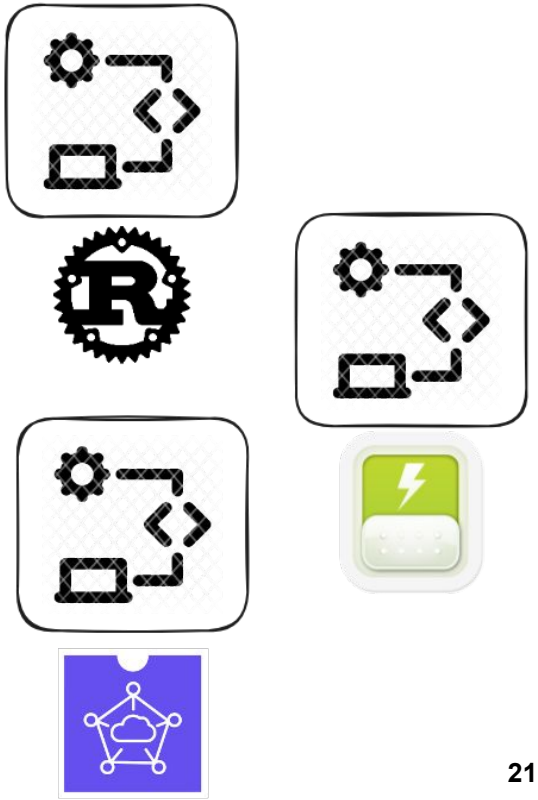
Component Layers

Modular building blocks for **existing pre-built components**. Enabling rapid assembly.

Bunny: Tooling layers

- Tooling layers contain the necessary tools to build
  - One tooling layer per toolchain
  - Compiler, libraries, etc.

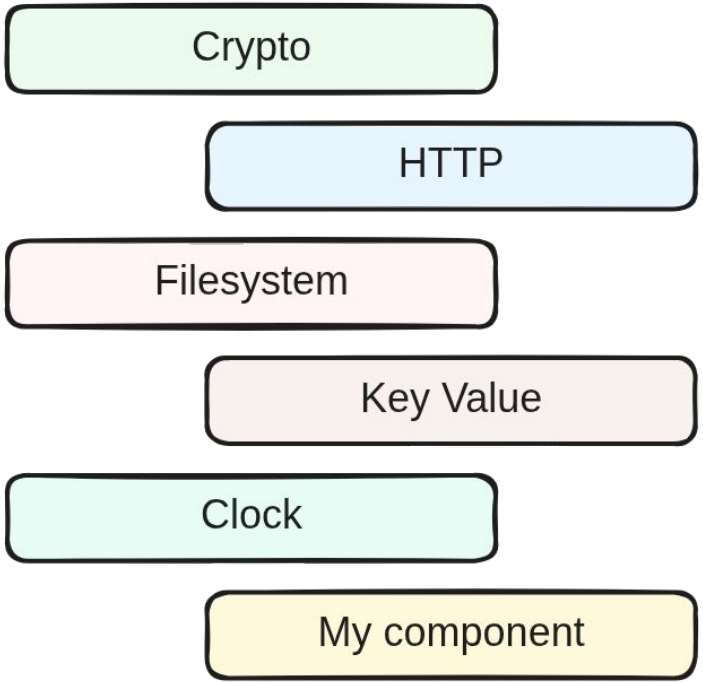
- Building process takes place inside a tooling layer
  - Compilation of user's app
  - Creation of Wasm artifact



Bunny: Component Layers

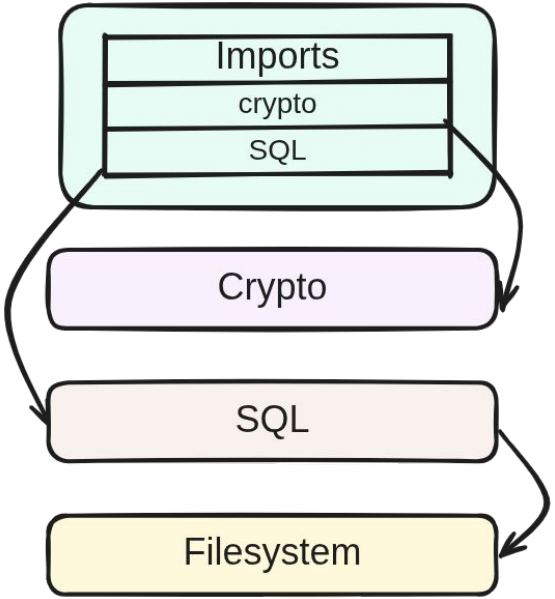
- Component layers contain
  - WIT interface defintions
  - The component binary

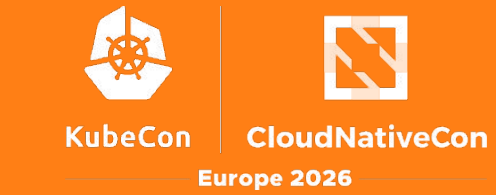
- Leverage OCI artifacts
  - Easily reference and distribute components
  - Integration with existing ecosystem solutions



Bunny: Dependency handling

- Create a dependency list from component's imports
  - Read and identify the imports of a component to use
- Collect all components that satisfy an implementation
  - Let the user choose the necessary component





Demo: Building and running a Wasm unikernel

Summary

- Wasm is an emerging and promising technology
- Wasm ecosystem is fragmented
- Diverse tools and building processes
- Bunny abstracts away the complexity of each toolchain
- A new layered approach on building Wasm modules

Summary

- Wasm is an emerging and promising technology
- Wasm ecosystem is fragmented
- Diverse tools and building processes
- Bunny abstracts away the complexity of each toolchain
- A new layered approach on building Wasm modules

Get in touch:

cmainas@nubificus.co.uk
ananos@nubificus.co.uk

https://github.com/nubificus/bunny