WASM meets unikernels: Secure and Efficient Cloud-Native Deployments

Charalampos Mainas, Georgios Ntoutsos, Anastassios Nanos
{**cmainas**,gntouts,ananos}@nubificus.co.uk

urunc

BUNNY

NUBIS

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

NUBIS

BUNNY

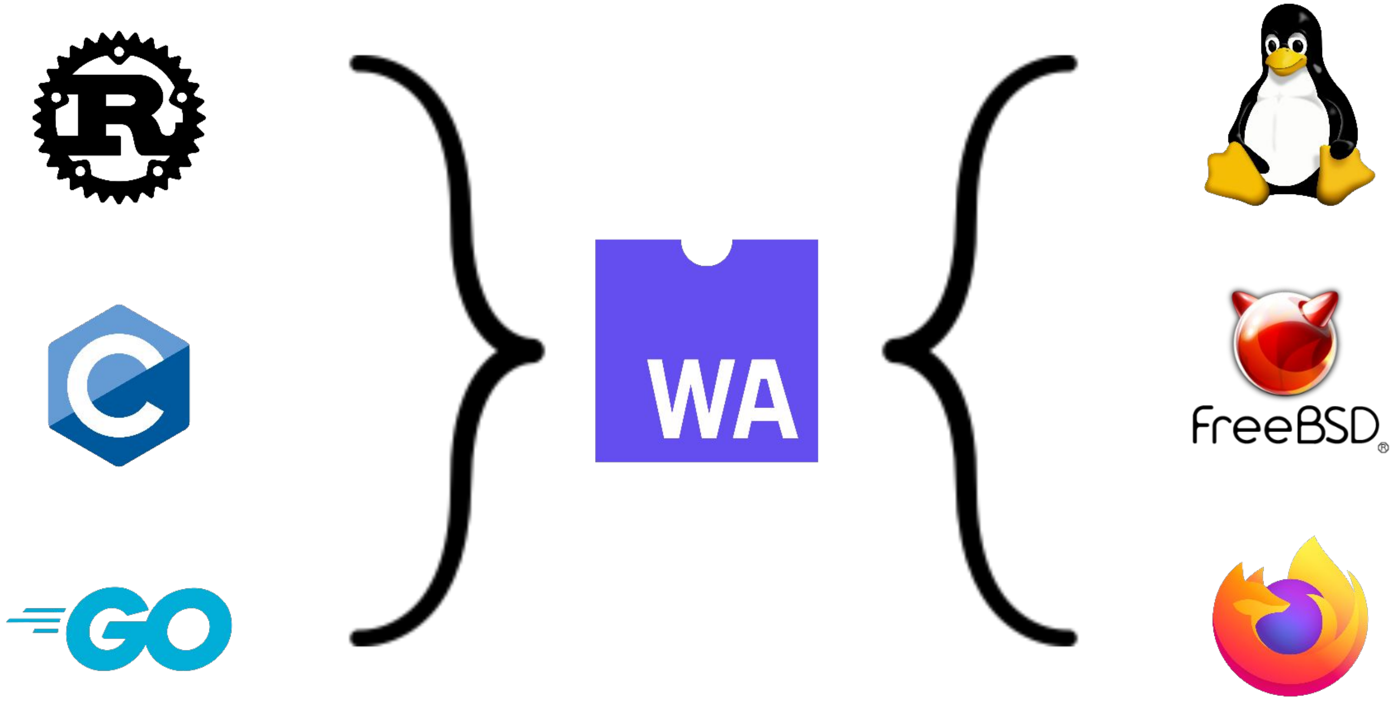
vaccel

urunc

Overview

- WASM
- A closer look on WASM's runtime environment
- The security concerns of WASM
- Providing isolation for WASM
- WASM and unikernels
- Build Wasm unikernels at ease
- Deploy Wasm unikernels at ease

The Art of WebAssembly



The Art of WebAssembly

Portability

Fast spawn

Sandbox

The Art of WebAssembly

Portability

Fast spawn

Sandbox

The Art of WebAssembly

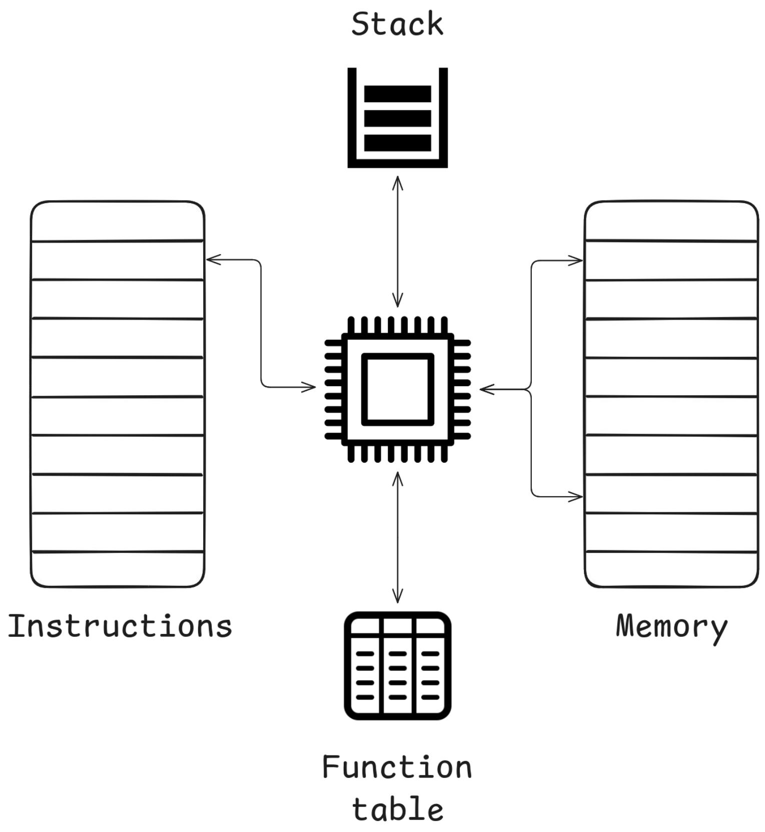
Portability

Fast spawn

Sandbox

A closer look at WebAssembly

- 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



WebAssembly and the outside world

- WASI
  - Standard interface
  - Interact with outside world
  - ABI and Inter-Component communication
- Capability-based system
  - Fine-grained control of system resources
  - Explicit declaration of resources

WebAssembly is not a panacea

- WebAssembly and traditional bugs/attacks [1]
  - Buffer overflow
  - Use after free
  - Double free
- Exploits of the runtime [2,3]
  - Escape sandbox – CVE-2023-26489
  - Data leak between instances– CVE-2022-39393
- User misconfiguration

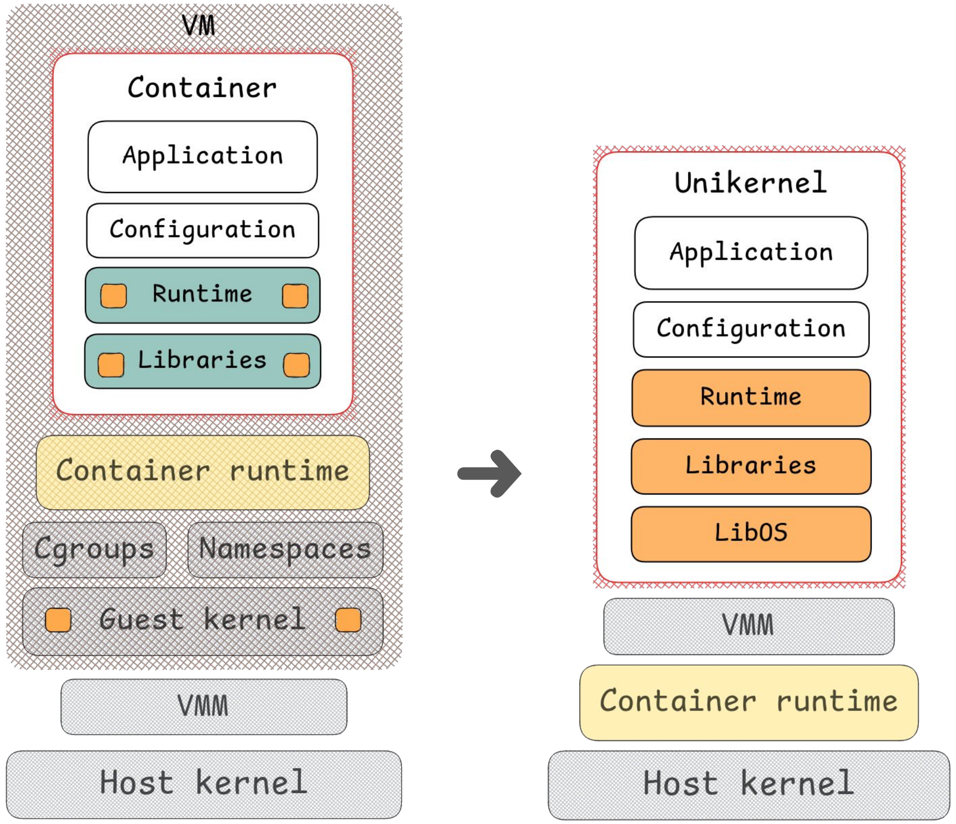
1. https://www.usenix.org/conference/usenixsecurity20/presentation/lehmann
2. https://i.blackhat.com/USA-22/Wednesday/US-22-Hai-Is-WebAssembly-Really-Safe-wp.pdf
3. https://ieeexplore.ieee.org/document/10179357

Sandboxing WebAssembly runtimes

Virtual Machines	Containers
Pros: ● Strong isolation ● Widely-used **Cons:** ● Slow boot time ● Inflexible resource management	**Pros:** ● Faster boot times ● Flexible resource management ● WIdely used **Cons:** ● Not that great isolation

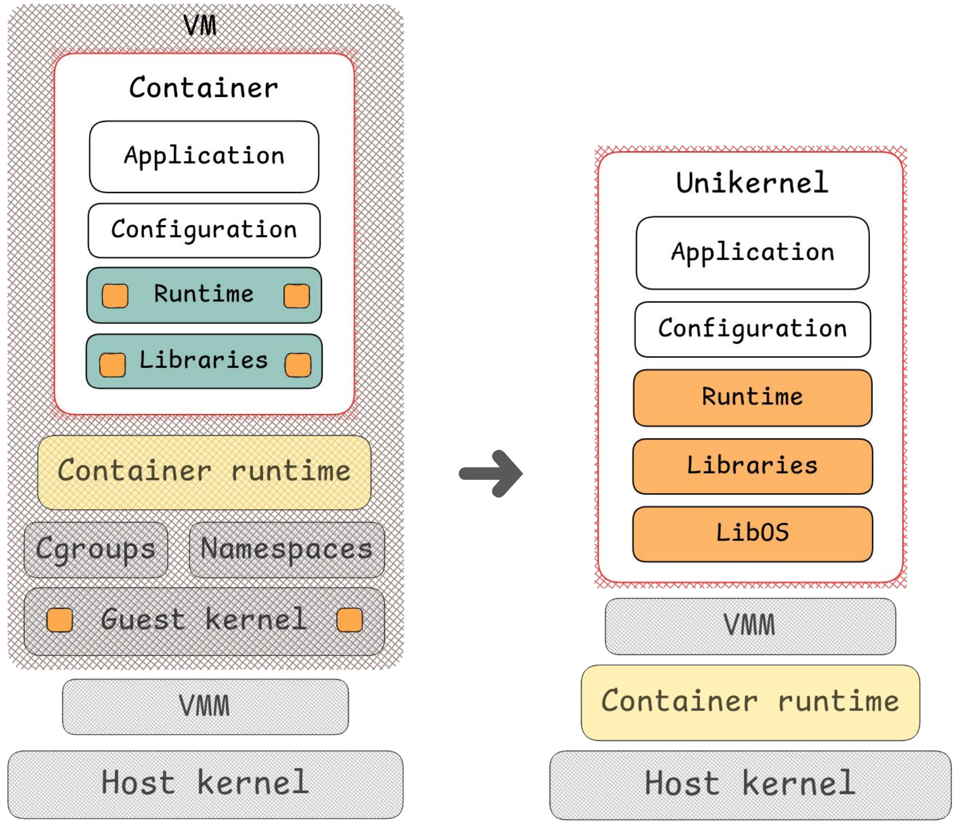
Unikernels

- A unikernel is:
  - specialized
  - single address space
  - constructed using a LibOS



Unikernels

- Benefits:
  - Fast boot times
  - Reduced attack surface
  - Truly isolated
  - Small memory/disk footprint
- Drawbacks:
  - Portability
  - Ease of use and integration with existing tools and practises



A perfect match?

Unikernel

- Low level setup
- Memory management
- I/O
- Strong isolation
- Small overhead/ footprint

A perfect match?

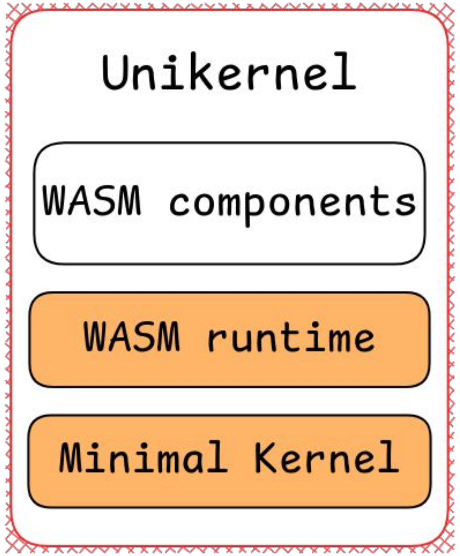
Unikernel
● Low level setup ● Memory management ● I/O ● Strong isolation ● Small overhead/ footprint

WebAssembly
● Process/Thread separation and management ● Standard application Interface ● No dependencies

A perfect match?

Unikernel

- Low level setup
- Memory management
- I/O
- Strong isolation
- Small overhead/ footprint



WebAssembly

- Process/Thread separation and management
- Standard application Interface
- No dependencies

WebAssembly in current unikernels

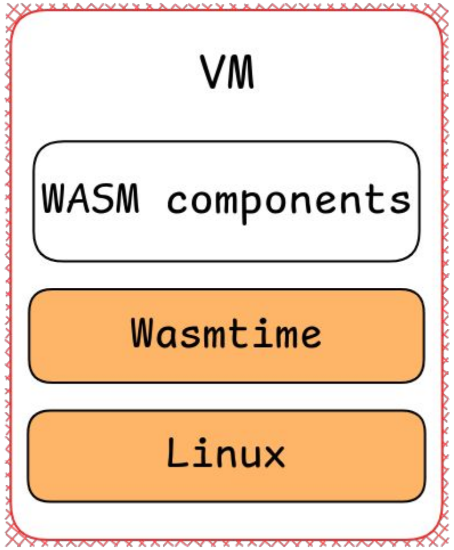
- Mewz
  - Written from scratch in Zig
  - Partial implementation of WASI preview 1 with sockets
- Unikraft:
  - Linux binary compatible unikernel
  - WAMR
- OSv:
  - Linux binary compatible unikernel
  - Wasmer
- Hermit-wasm:
  - Based on RustyHermit
  - Wasmi

More WASM-focused kernels

- k23
  - Microkernel-based
  - https://github.com/JonasKruckenberg/k23
- wasmlinux
  - Linux-based
  - https://github.com/okuoku/wasmlinux-project
- Redshirt
  - Redshirt
  - https://github.com/tomaka/redshirt/tree/main
- wasm-kernel
  - Minimal kernel to run WASM
  - https://github.com/michaelmelanson/wasm-kernel
- kwast
  - Microkernel-based
  - https://github.com/kwast-os/kwast

Cheating: A tiny Linux to run a WebAssembly runtime

- Why Linux:
  - Many WASM runtimes support
  - WIdely used
  - Highly customizable
- We are cheating:
  - Userspace / Kernelspace separation
  - Include unnecessary things
  - Multi-process



But…

- Wasm toolchains are already stiff
  - Unikernels are even more stiff
  - Unikernels are notorious of being user unfriendly
- Wasm can already get deployed as containers
  - Docker support
  - Various runtimes supported by crun
  - Runwasi from containerd
  - SUpport for Kubernetes

But…

- Wasm toolchains are already stiff
  - Unikernels are even more stiff
  - Unikernels are notorious of being user unfriendly
- Wasm can already get deployed as containers
  - Docker support
  - Various runtimes supported by crun
  - Runwasi from containerd
  - SUpport for Kubernetes

How on earth can I do the same with unikernels?

Building and deploying unikernels such as containers

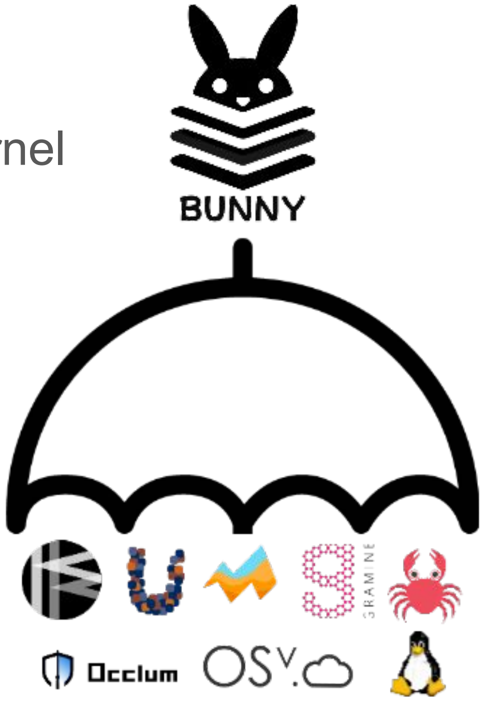
- Bunny: building WASM unikernels like containers
  - A buildkit frontend able to build code as wasm and build it as unikernel
- Urunc: The unikernel container runtime
  - Manage the execution of unikernels as containers

Building and deploying unikernels such as containers

- **Bunny: building WASM unikernels like containers**
  - A buildkit frontend able to build code as wasm and build it as unikernel
- Urunc: The unikernel container runtime
  - Manage the execution of unikernels as containers

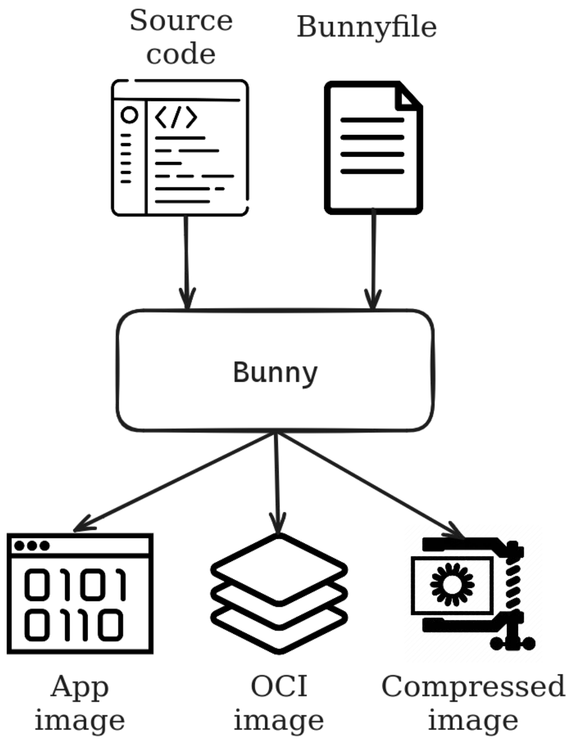
Bunny: build frameworks like containers

- Configure once build against multiple libOSes/kernels
  - Unified interface for libOSes/kernels
- Simplify the process of building an app with a libOS/kernel
  - Abstract away the diversity and complexity of each toolstack
- No dependency hell
  - Bunny takes care of resolving framework dependencies



Bunny: build libOSes/kernels like containers

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



Bunny: demo

- Building a WASM app as a unikernel targeting:
  - Unikraft
  - Mewz
  - Linux

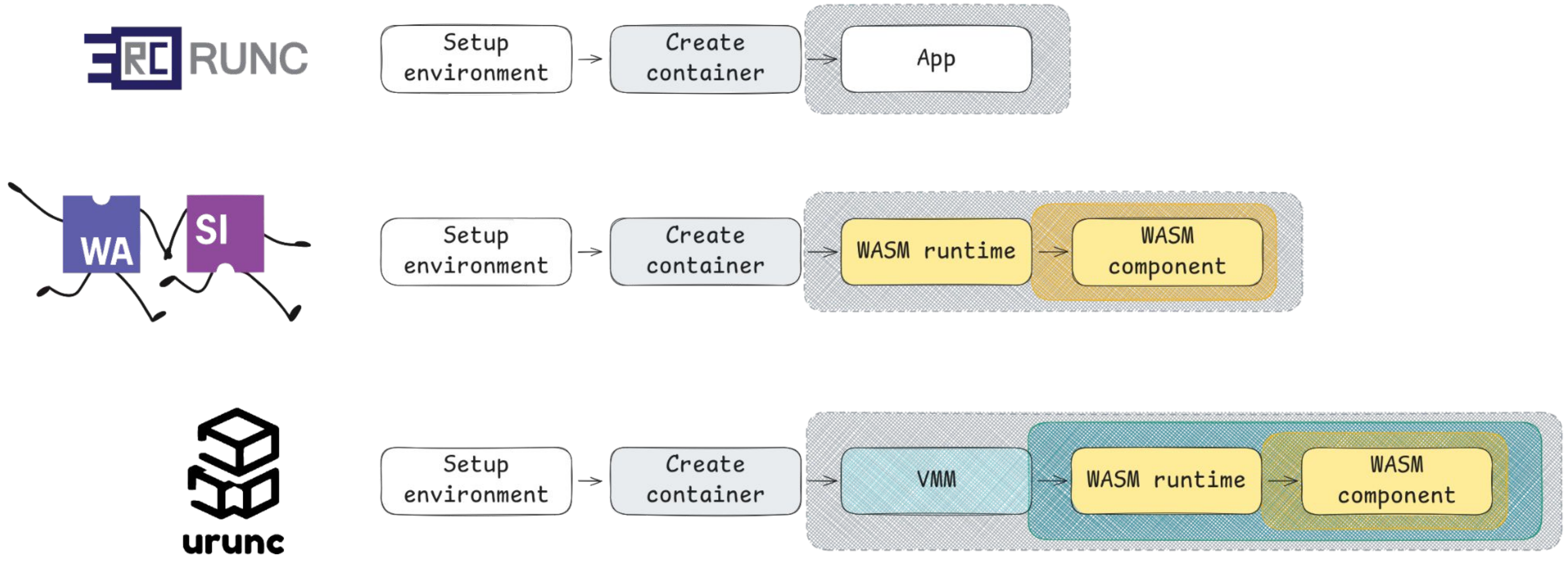
Building and deploying unikernels such as containers

- Bunny: building WASM unikernels like containers
  - A buildkit frontend able to build code as wasm and build it as unikernel
- **Urunc: The unikernel container runtime**
  - Manage the execution of unikernels as containers

Urunc: The runc of unikernels

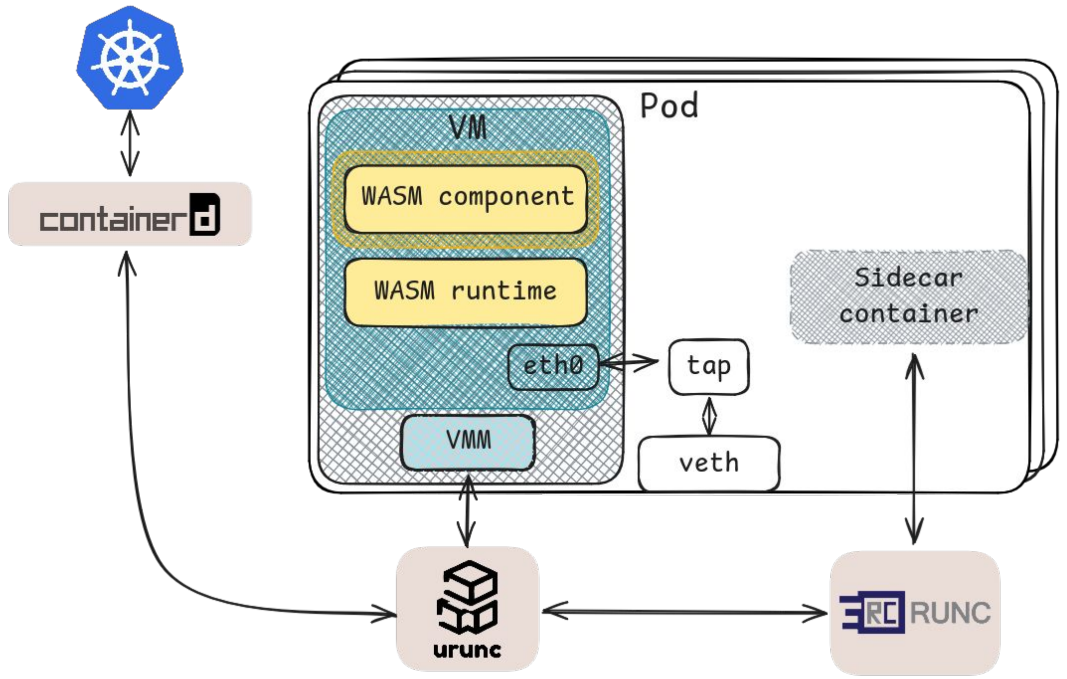
- **CRI-compatible** runtime written in Go
- **Extensible**, **easy** to add support, **without modifications** for unikernel frameworks & hypervisors
- **Hides complexity** of unikernel framework-specific and hypervisor command line options
- Key differences
  - Spawns **app** directly **inside** the **VM**
  - Treats **VMs as processes**
  - One **VM per container**

Urunc: The runc of unikernels



Urunc: Integration with k8s

- Networking
  - TC mirror tap-veth
- Storage
  - Initramfs
  - Block devices
  - Shared-fs
- Sidecar containers
  - Forward to generic container runtime



Urunc: demo

- Running the previously built WASM unikernels with urunc

This work is partially funded through Horizon Europe actions, MLSysOps (GA: 101092912) and DESIRE6G (GA: 101096466)

Summary

- WASM is an emerging and promising technology
- Security in-depth is necessary
- WASM can benefit from unikernels for isolation and unikernels for portability
- Bunny automates the build process of WASM unikernels
- Urunc enables the deployment and management of WASM unikernels as containers

Summary

- WASM is an emerging and promising technology
- Security in-depth is necessary
- WASM can benefit from unikernels for isolation and unikernels for portability
- Bunny automates the build process of WASM unikernels
- Urunc enables the deployment and management of WASM unikernels as containers

Check out the code on github:

- https://github.com/nubificus/urunc
- https://github.com/nubificus/bunny