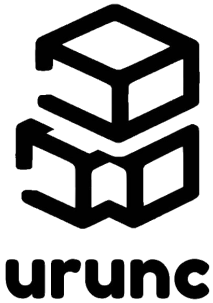
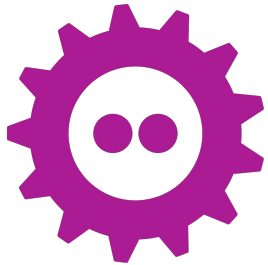
Less overhead, strong isolation: Running containers in minimal specialized Linux VMs

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



NUBIS

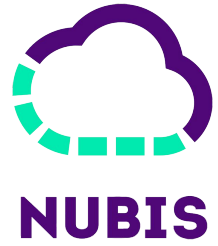


About us

- 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







About us

- Team of Researchers & engineers
- Based in the UK & GR

Apostolos Giannousas
Software Engineer
agian@nubificus.co.uk

Maria-Rafaella Gkeka
Research Engineer
mgkeka@nubificus.co.uk

Maria Goutha
Software Engineer
mgouth@nubificus.co.uk

Ilias Lagomatis
Systems Software Engineer
ilago@nubificus.co.uk

Charalampos (Babis) Mainas
Research Engineer
cmainas@nubificus.co.uk

Panos Mavrikos
Software Engineer / K8s
pmavrikos@nubificus.co.uk

Anastassios (Tassos) Nanos
Systems Researcher
ananos@nubis-pc.eu

Georgios Ntoutsos
Software & Systems Engineer
gntouts@nubis-pc.eu

Kostis Papazafeiropoulos
Systems Researcher
papazof@nubis-pc.eu

Overview

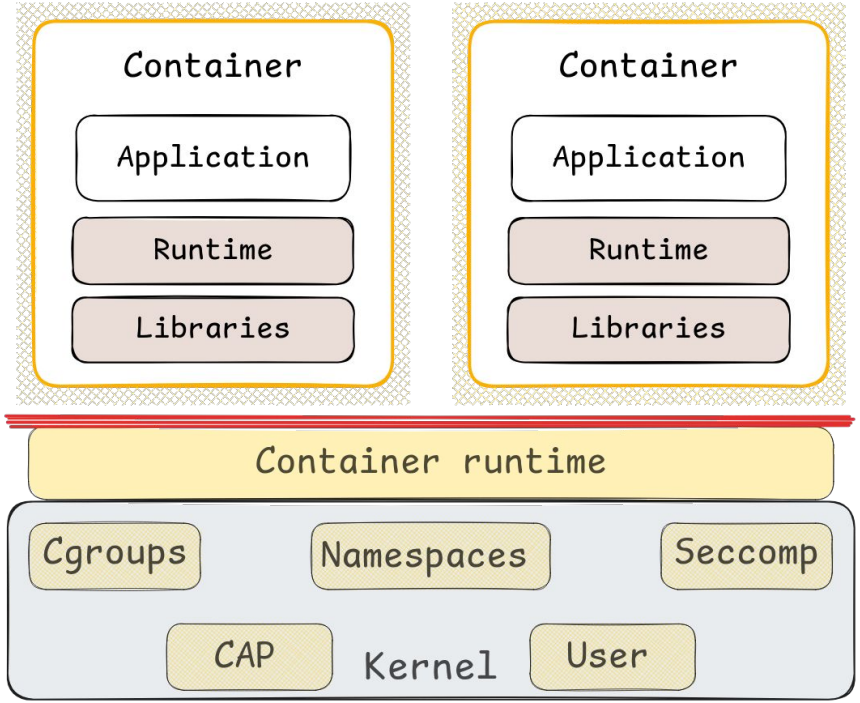
- Containers isolation
- Kata-containers
- Reverse the isolation boundaries
- Stripping down the Linux kernel
- Run containers over minimal Linux VMs
- Early evaluation

Containers: packaging and isolation

- The de-facto application packaging/deployment:
  - Lightweight
  - Fast spawn times
  - Easy to use
  - Rich ecosystem

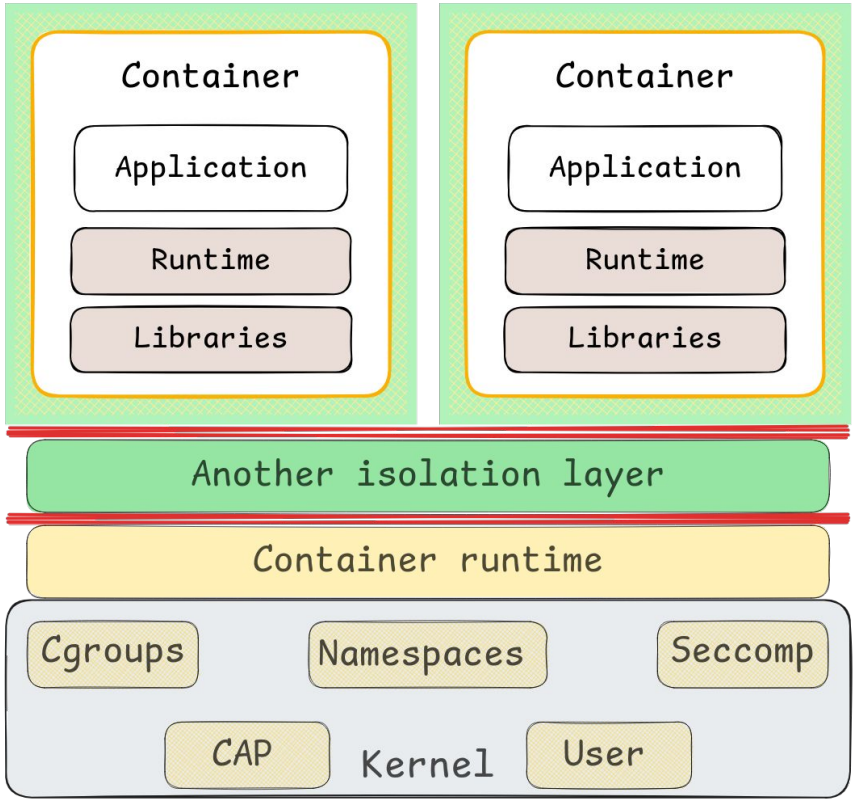
Containers: packaging and isolation

- The de-facto application packaging/deployment:
  - Lightweight
  - Fast spawn times
  - Easy to use
  - Rich ecosystem
- Main isolation mechanisms:
  - Process-level isolation
  - Based on kernel mechanisms
  - Shared kernel



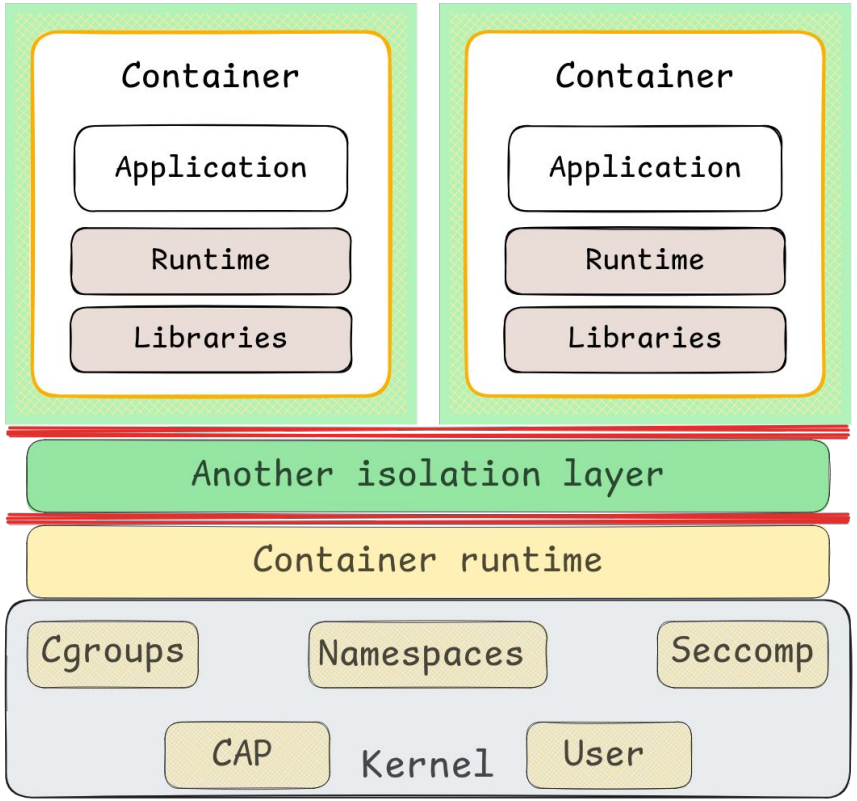
Sandboxing containers

- Provide an extra level of isolation
  - Container runs inside a sandbox
  - Interact with intermediate layer
  - Restricted access to host kernel
- Mechanisms:
  - Software-based
  - Hardware-assisted

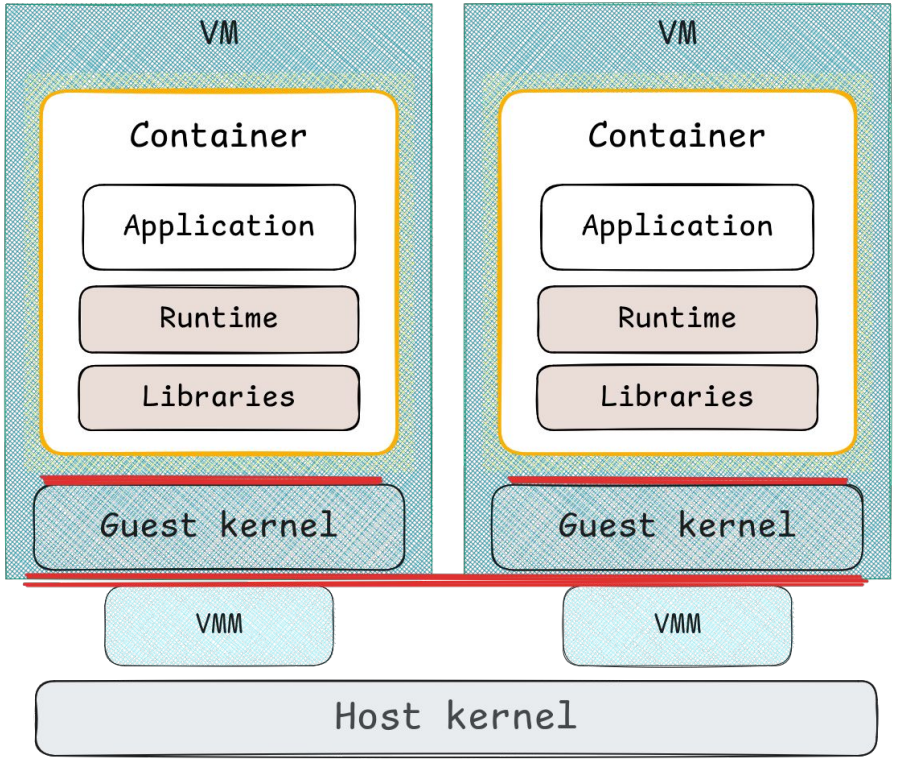


Sandboxing containers

- Provide an extra level of isolation
  - Container runs inside a sandbox
  - Interact with intermediate layer
  - Restricted access to host kernel
- Mechanisms:
  - Software-based
  - **Hardware-assisted**

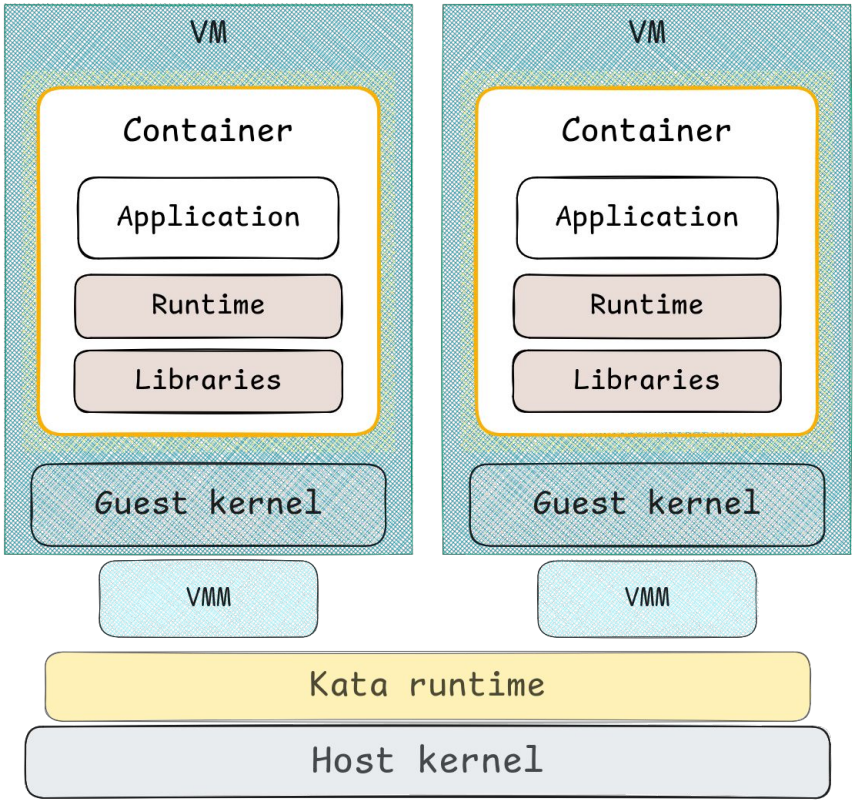


HW-assisted sandbox



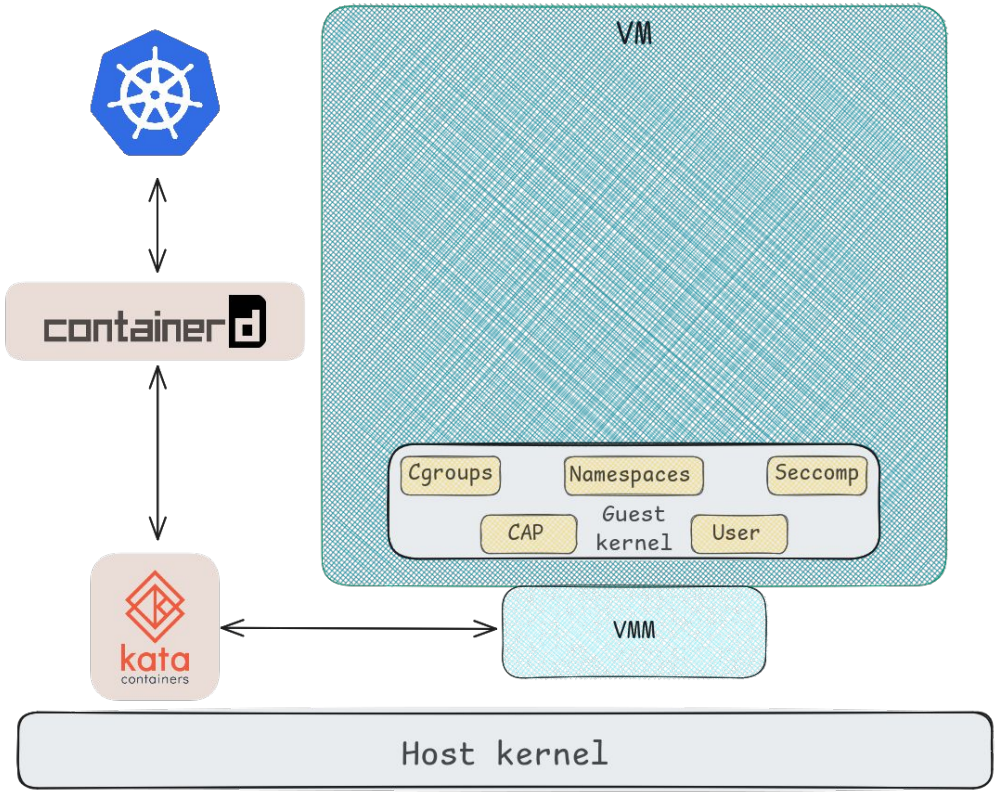
The case of Kata-containers

- Use of VMs as the additional layer
  - One VM per pod
  - Seamless integration
- Side effects
  - Spawn overhead
  - Storage / Memory footprint
  - Complicated stack



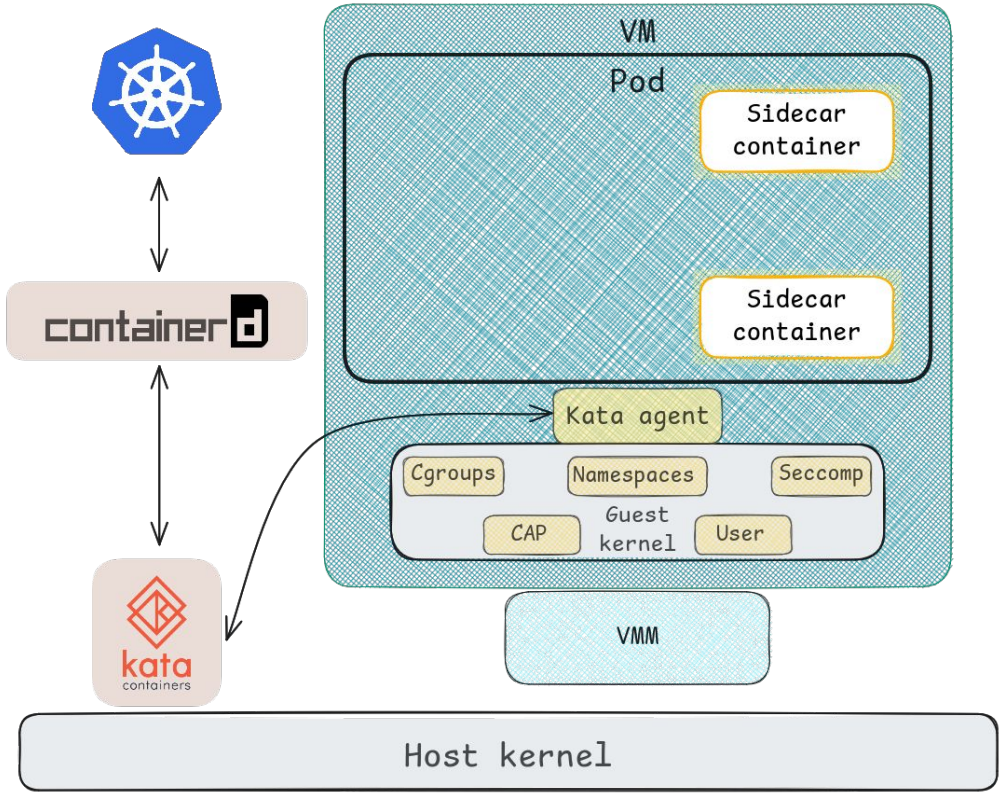
A closer look to Kata-containers

- Create a VM with:
  - Kata Linux kernel
  - Kata rootfs



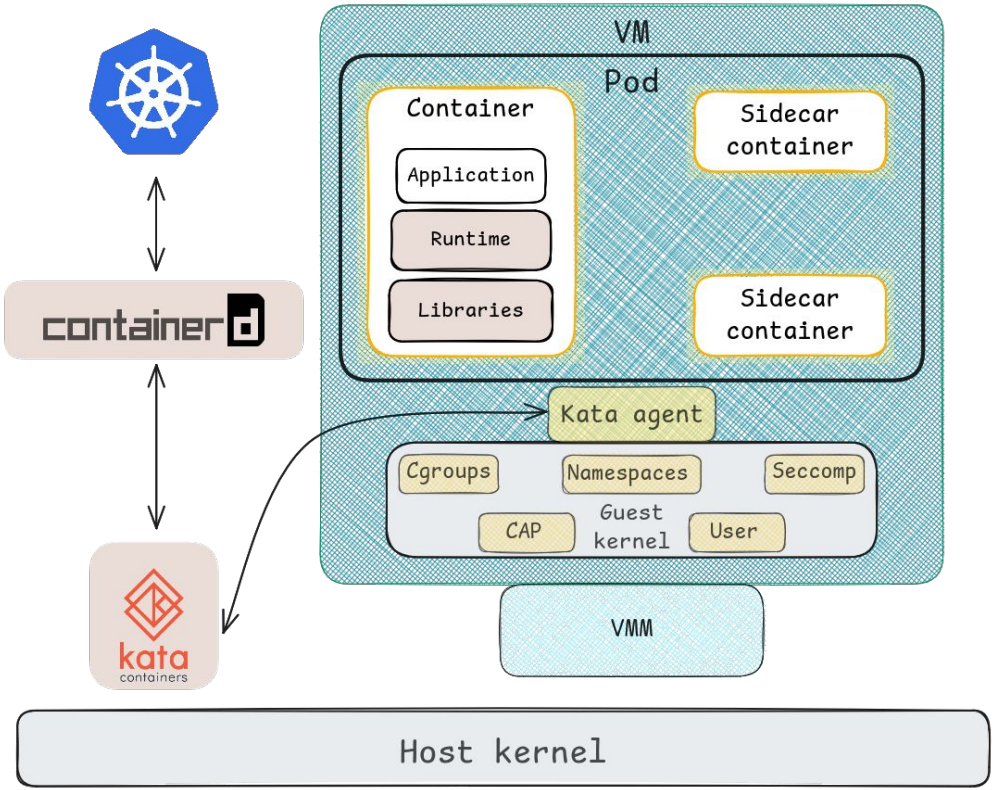
A closer look to Kata-containers

- Create a VM with:
  - Kata Linux kernel
  - Kata rootfs
- Create Pod with sidecar containers



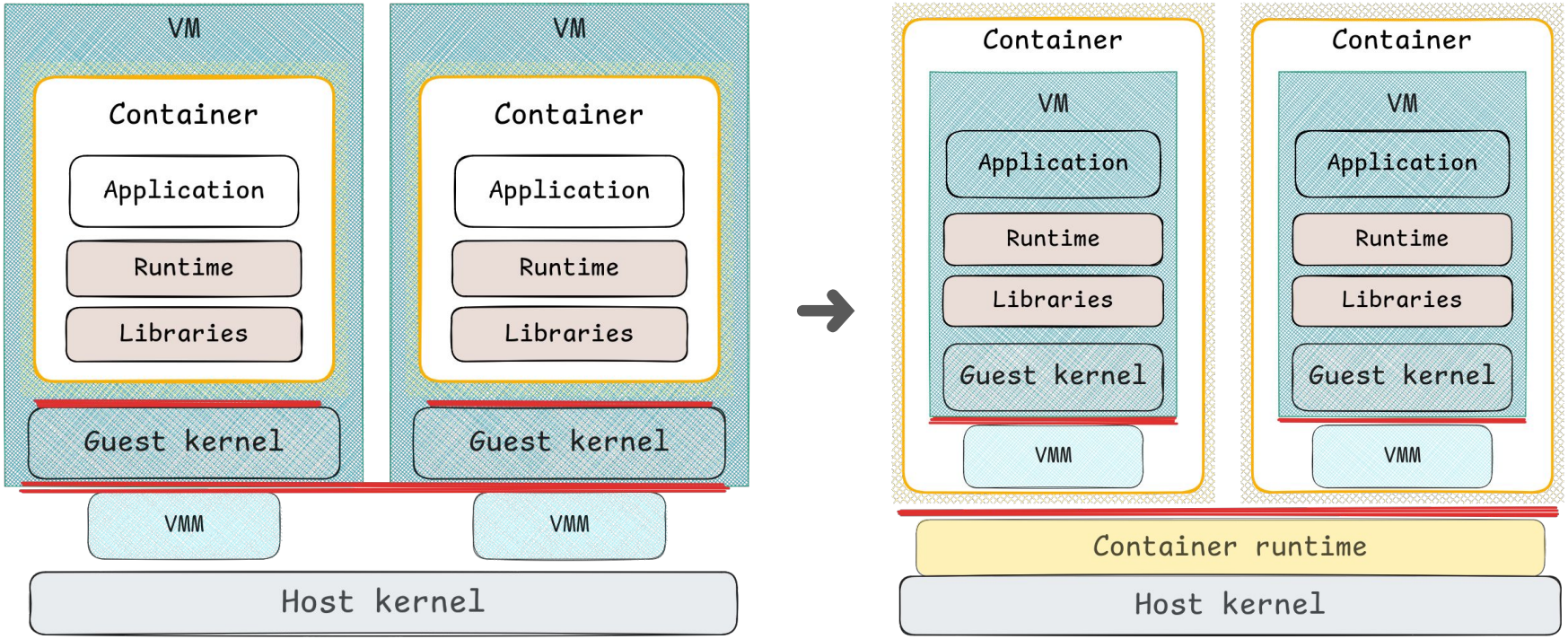
A closer look to Kata-containers

- Create a VM with:
  - Kata Linux kernel
  - Kata rootfs
- Create Pod with sidecar containers
- Create user container



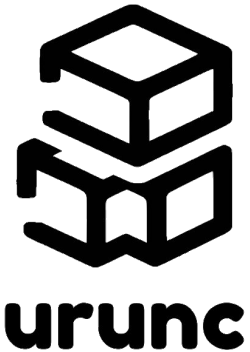
What if we do the opposite?

Redrawing the isolation boundaries

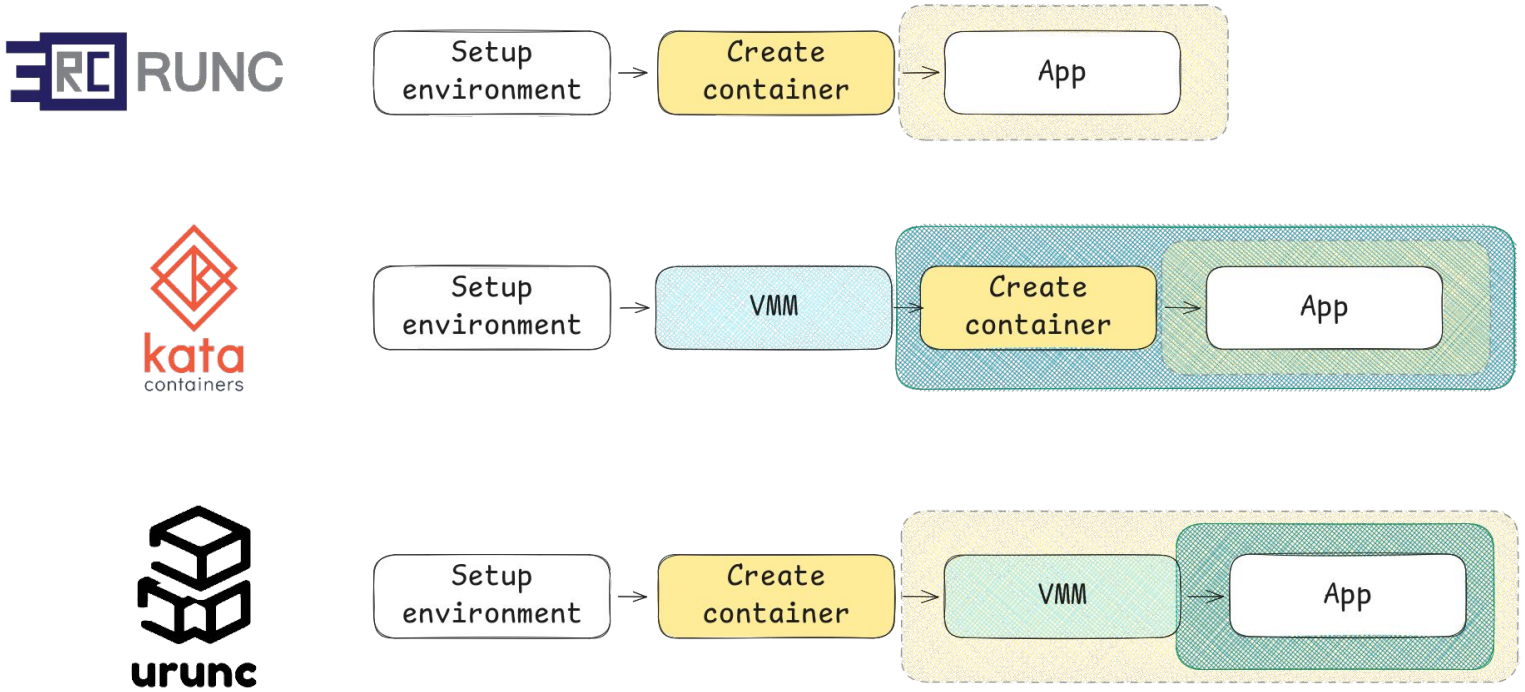


Urunc: The runc of unikernels*

- **CRI-compatible** runtime written in Go
- **Extensible**, **easy** to add support, **without modifications** for guest kernels & hypervisors
- Key differences
  - Spawns **app** directly **inside** the **VM**
  - Treats **VMs as processes**
  - One **VM per container**

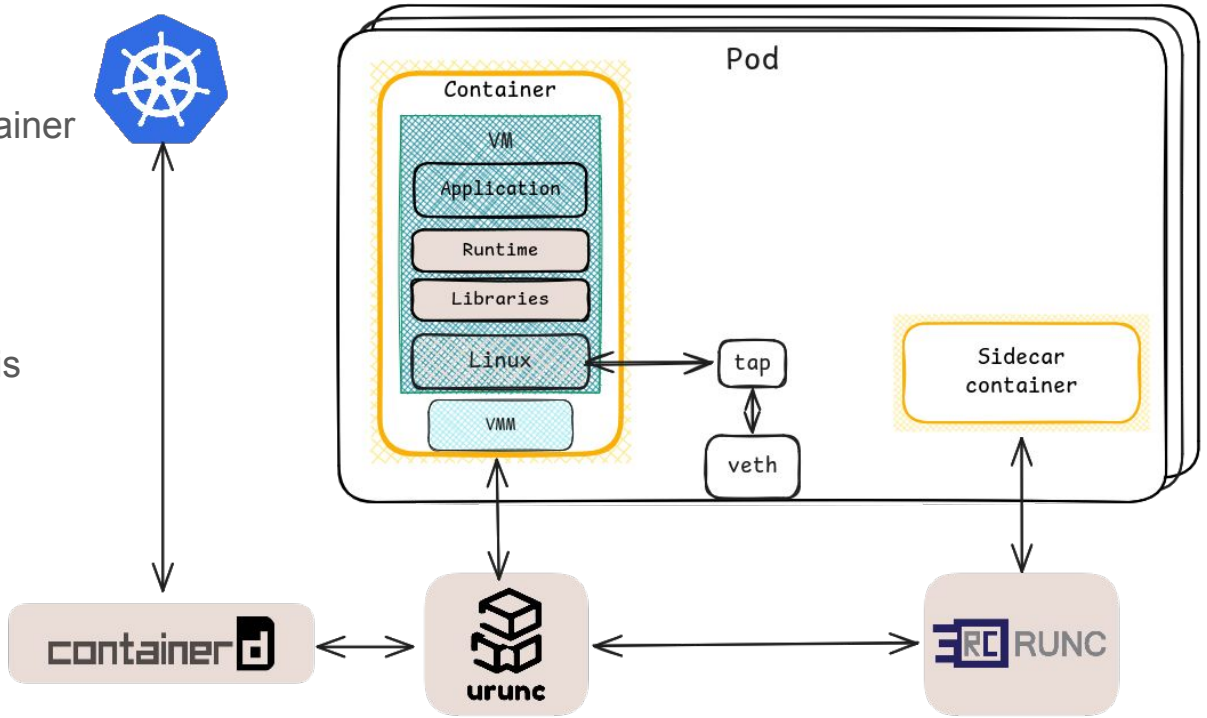


Runc: Comparison with other container runtimes



Urunc: Integration with k8s

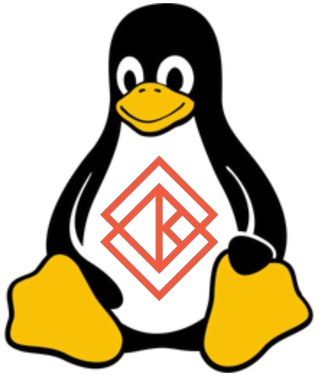
- Sidecar containers
  - Forward to generic container runtime
- Sandbox only the user container
  - Untrusted, rest of stack is trusted



Demo: Containers over a Linux VM

Kata-containers’ Linux kernel

- Default configuration
  - Cgroups
  - Namespaces
  - User management
  - Seccomp
  - Device hotplug
  - ACPI
  - Xtables
  - And many more..
- Specialization is vital for
  - Overhead/footprint
  - Performance
  - Security



Stripping down to the essentials

Kata Linux

- Generic LInux configuration
- Compatible with all applications
- Quite big

A lot of features are not required for an app

Stripping down to the essentials

Kata Linux

- Generic LInux configuration
- Compatible with all applications
- Quite big

A lot of features are not required for an app

Unikernels

- Specialized for one application
- Very small
- Hard to port

To provide compatibility, we end up with a Linux clone

Stripping down to the essentials

Kata Linux

- Generic LInux configuration
- Compatible with all applications
- Quite big

A lot of features are not required for an app

Not reinventing the wheel

- Linux is highly customizable
- Create a config specialized for an app
- Compatibility

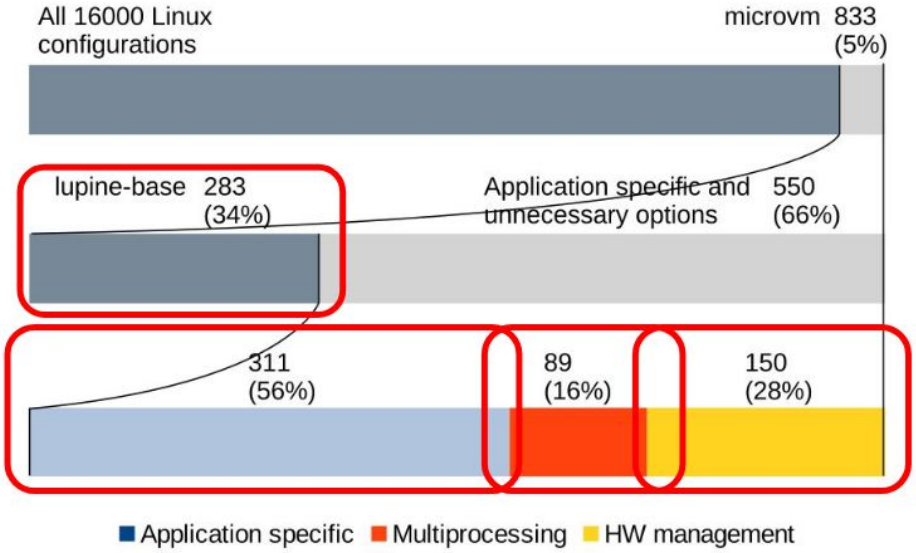
Unikernels

- Specialized for one application
- Very small
- Hard to port

To provide compatibility, we end up with a Linux clone

Lupine: Linux in unikernel clothing

- Convert Linux to a unikernel
  - Strip-down Linux using configuration options
  - Patches to run app in kernel space
- Key findings
  - Small Linux kernels
  - Boot as fast as unikernels
  - System call overhead:
    i. 4% macro benchmarks
    ii. 40% microbenchmarks

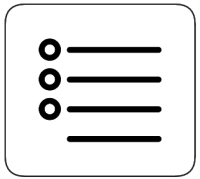


Source: https://dl.acm.org/doi/10.1145/3342195.3387526

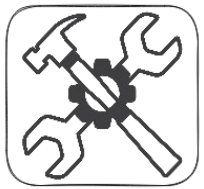
Create Linux kernels for a container

- Analyze user container
  - Identify requirements from the Linux kernel

- Create app-specific configuration
  - Starting from the bare minimum and adding up



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



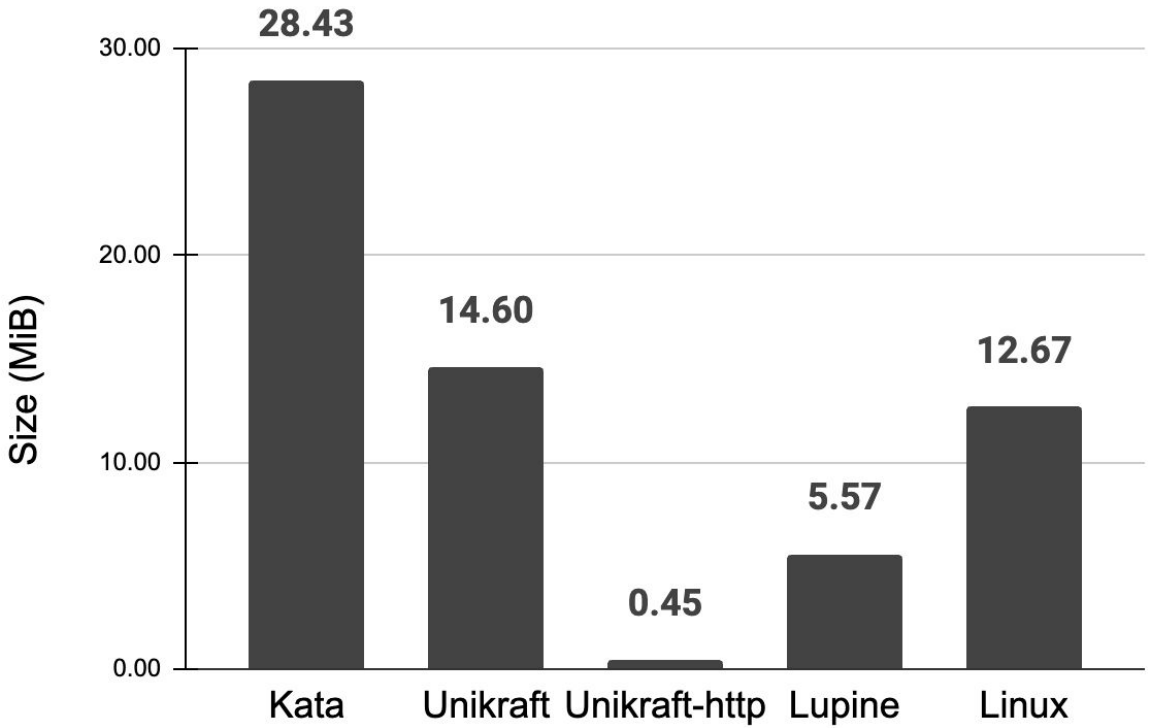
Demo: Containers over Minimal Linux VMs

Early evaluation

- Measure size of kernel image
- Measure start time of container for each runtime
  - Combined with various guests for urunc
- Testbed:
  - CPU: Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz
  - RAM: 376G

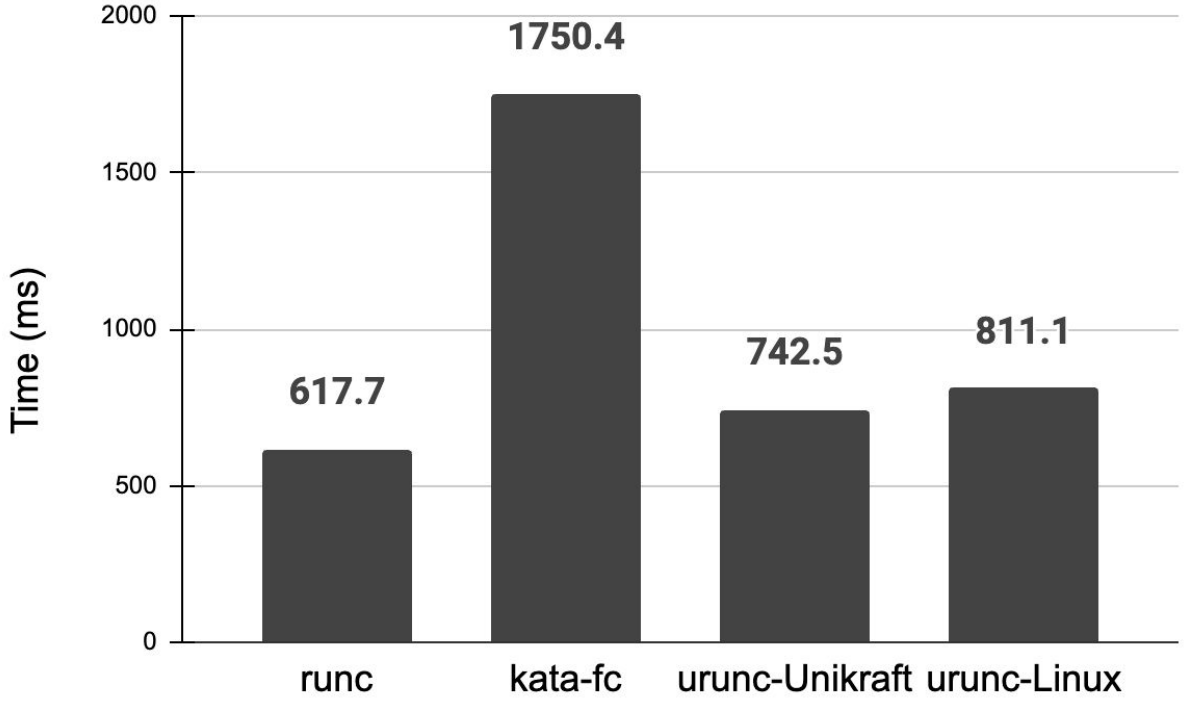
Kernel image size

- Targeting Nginx
  - Sizes in MiB
  - **Lower is better**
- Kernels:
  - Kata Linux v6.12.8
  - Unikraft v0.18.0
  - Unikraft v0.18.0 HTTP
  - Lupine Linux v4.0.0-rc1
  - Linux v6.12.8



Container start time

- Using Nginx
  - Start time in **ms**
  - **Lower is better**
- Runtimes/Guests
  - Runc
  - Kata
  - Urunc Unikraft
  - Urunc LInux



This work is partially funded through Horizon Europe Research and Innovation actions, MLSysOps (GA: 101092912) and EMPYREAN (GA: 101136024)

Summary

- Containers are great, but lack isolation
- Sandbox execution for stronger isolation
- Alternative sandbox mechanisms enforce overhead
- Reduce runtime overhead with urunc
- Strip down the Linux kernel for even lower overhead and attack surface

Summary

- Containers are great, but lack isolation
- Sandbox execution for stronger isolation
- Alternative sandbox mechanisms enforce overhead
- Reduce runtime overhead with urunc
- Strip down the Linux kernel for even lower overhead and attack surface

Check out the code on github:

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

Check out our WASM talk:
WASM meets unikernels: Secure and Efficient Cloud-Native Deployments