Bringing BSD Applications on Linux container platforms with urunc

Charalampos (Babis) Mainas

FOSDEM -BSD, illumos, bhyve, OpenZFS devroom 2026

About us

- SME (inc. 2020) involved in Research/Commercial and Open Source projects
- Focus on systems software
  - Hypervisors and container runtimes
  - Hardware acceleration
  - Bring cloud-native concepts to Edge / Far-Edge devices

NUBIS

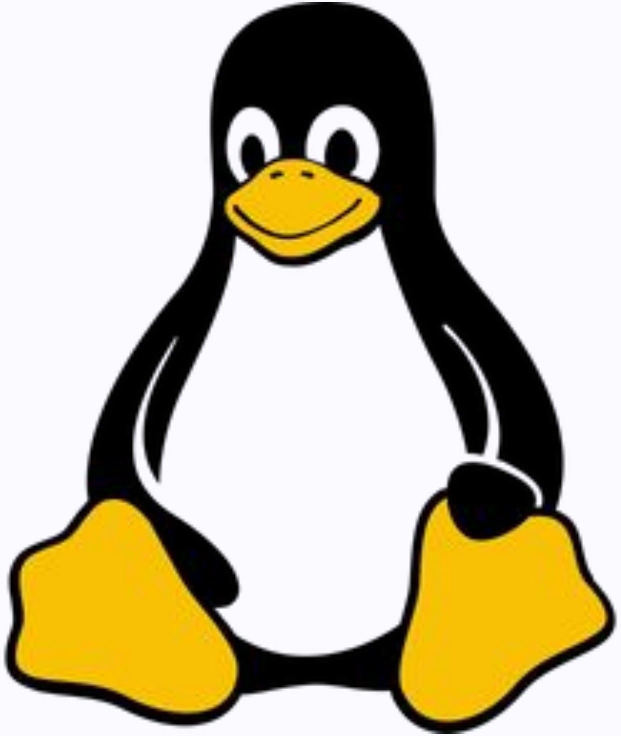
BUNNY

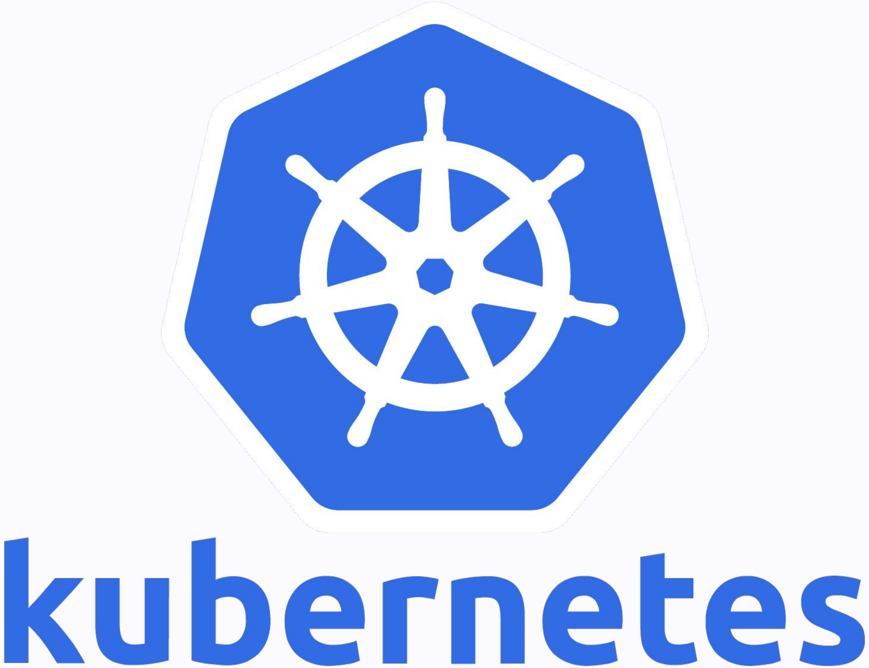
urunc

vaccel

Trigger warning

Let’s talk about

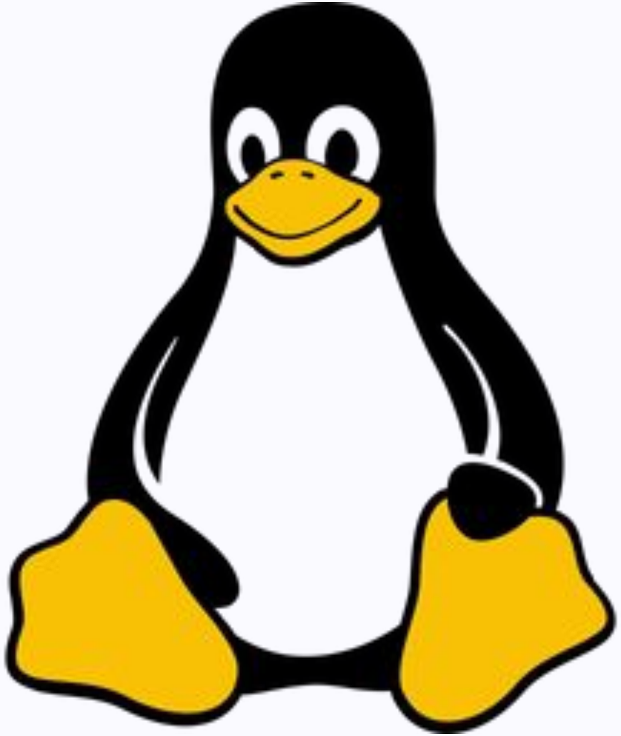


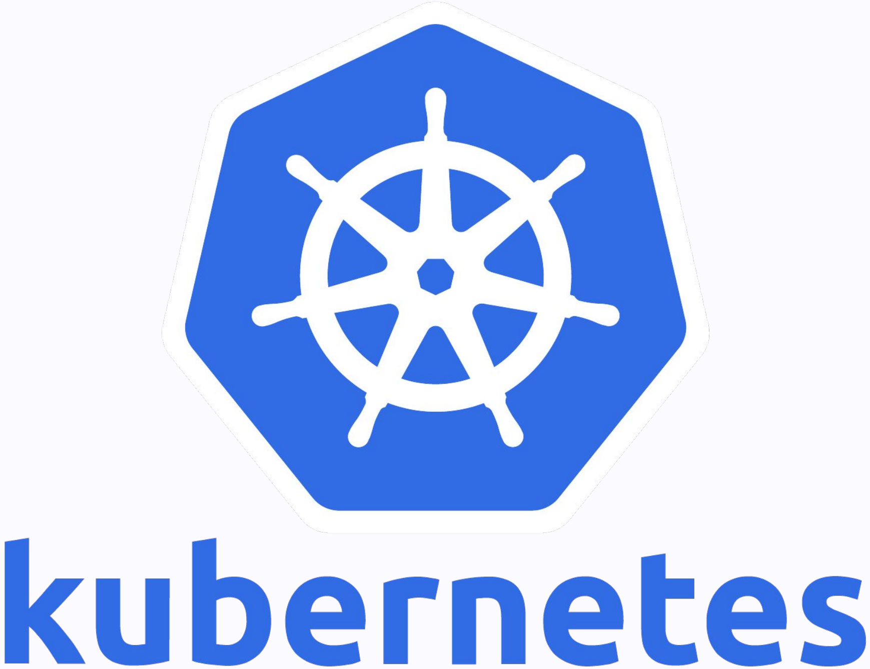


Let’s talk about BSD

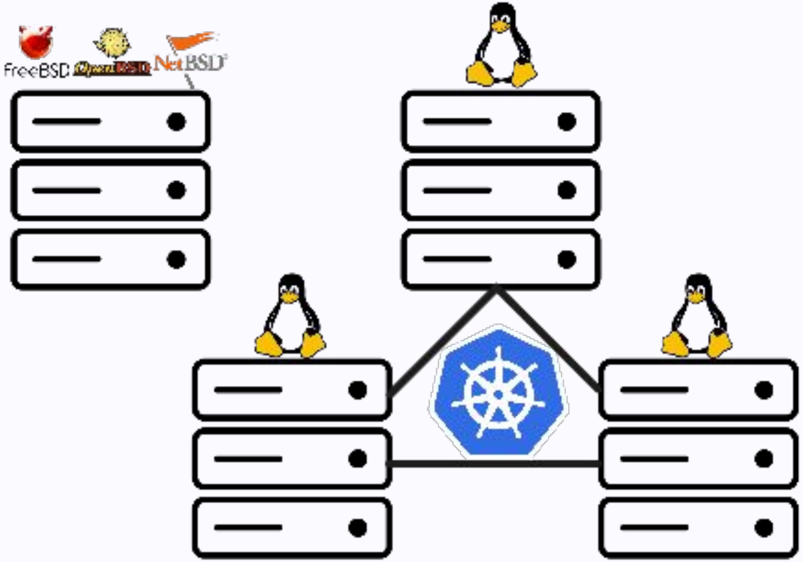
- BSD OSes are widely known for
  - stability
  - high security standards
  - network performance
- Use cases
  - Firewalls, routers
  - Storage controllers and data management systems
  - Load balancers

How can we fit BSD here?

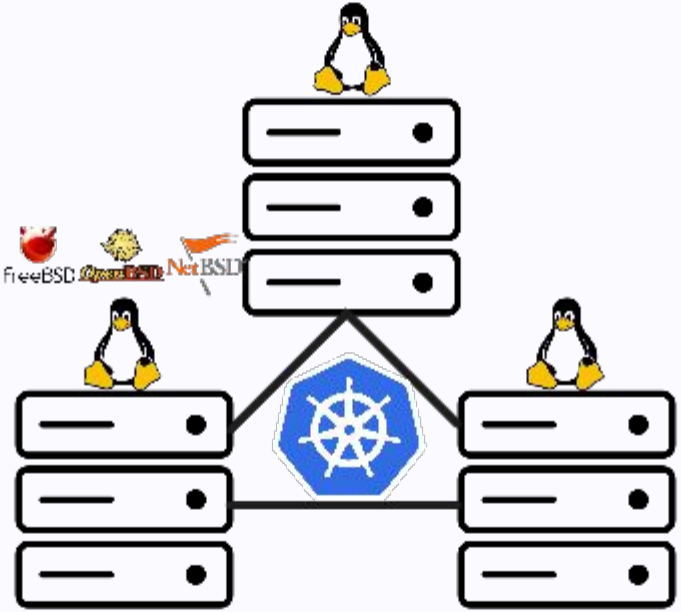




BSD deployments

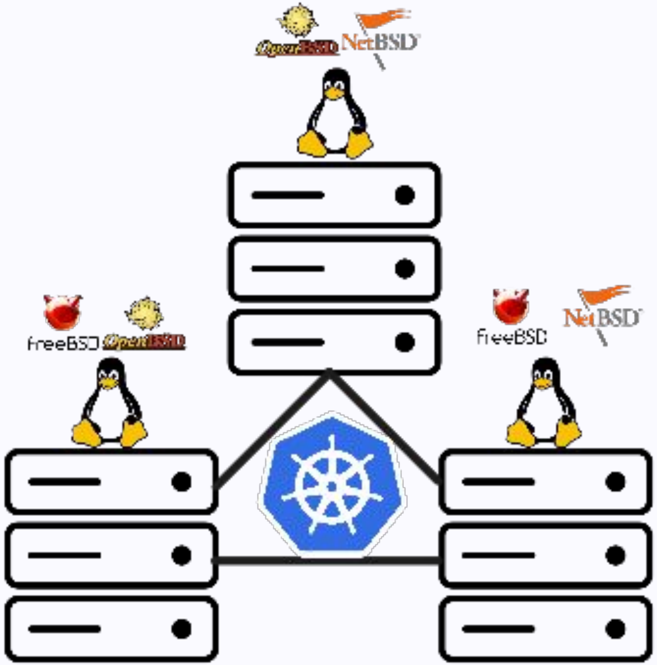


Dedicated server



Virtual Machine (VM)

BSD deployment as a mircoservice



Microservices

Embracing the microservices architecture

Full VM

- Fully featured BSD OS
- Unnecessary services
- Require a lot of resources
- Full distribution maintenance

Embracing the microservices architecture

Full VM

- Fully featured BSD OS
- Unnecessary services
- Require a lot of resources
- Full distribution maintenance

Single app kernel

- Configure kernel for a specific purpose
- Single service
- Lower resource usage, less noise
- Kernel and dependencies maintenance

Embracing the microservices architecture

Full VM

- Fully featured BSD OS
- Unnecessary services
- Require a lot of resources
- Full distribution maintenance

Single app kernel

- Configure kernel for a specific purpose
- Single service
- Lower resource usage, less noise
- Kernel and dependencies maintenance

Unikernel

- Specialized kernel and linked directly with the service
- Single address space
- Improved performance, less resources consumption
- Not really user friendly

How do we build and deploy these things?

How do we build and deploy these things?

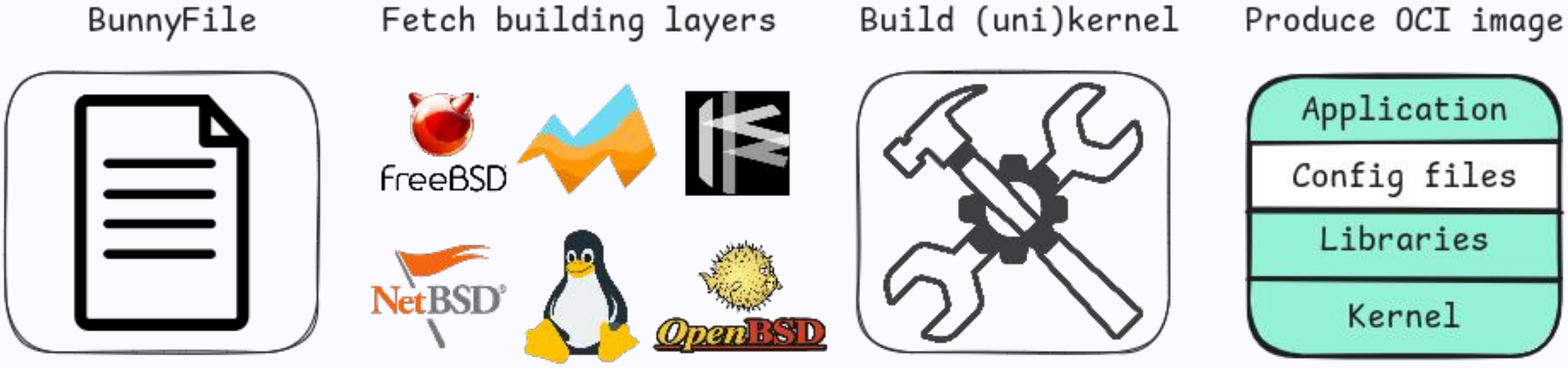
Build them like containers

Bunny: build (uni)kernels like containers

- A container-like experience
  - Same workflow with containers building
- 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
GRAMINE
OpenBSD
FreeBSD
NetBSD
OSv

Bunny: build (uni)kernels like containers



BunnyFile

Fetch building layers

Build (uni)kernel

Produce OCI image

Application

Config files

Libraries

Kernel

Demo: Building

How do we build and deploy these things?

How do we build and deploy these things?

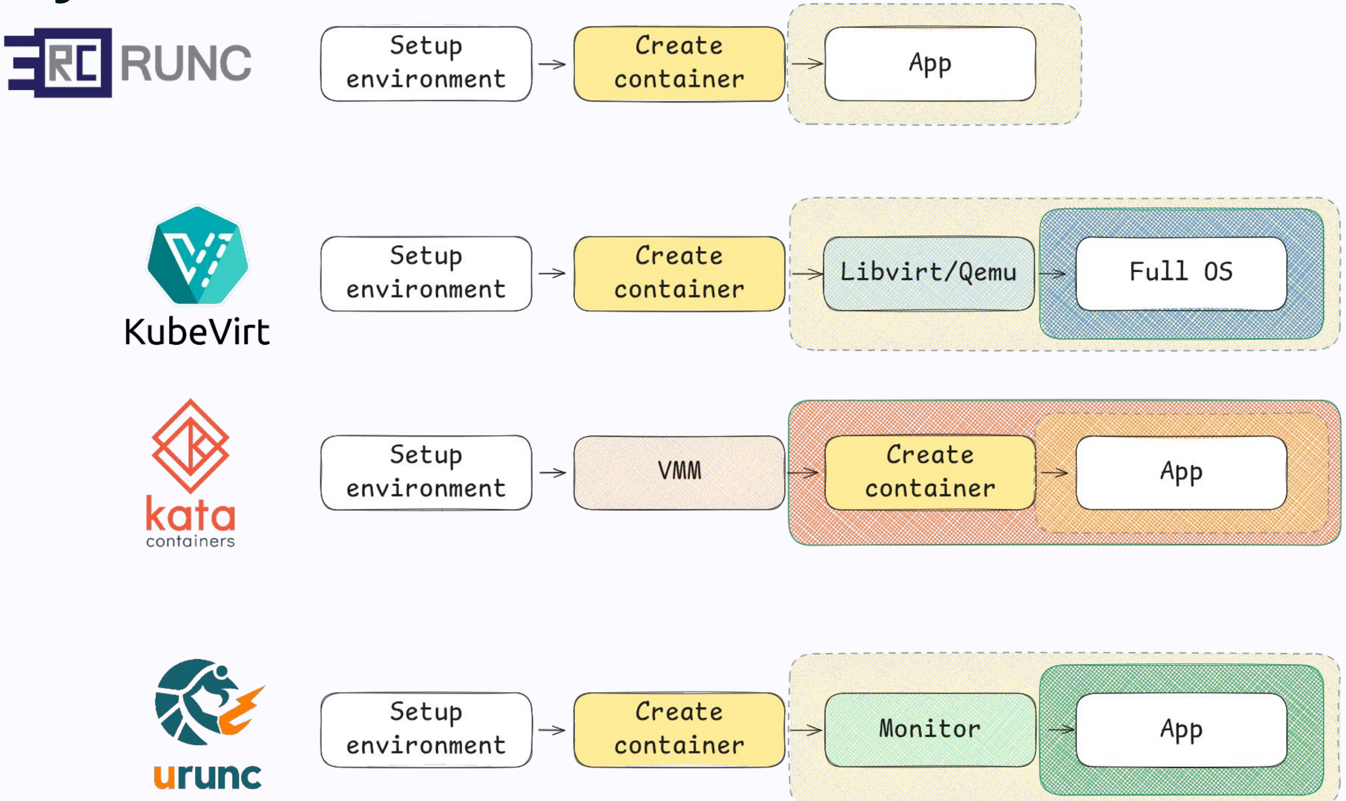
Deploy them like containers

urunc: The runc of unikernels & single app kernels

- CNCF Sandbox project
- CRI-compatible runtime written from scratch
- Support both SW-based and HW-assisted sandboxes
- Extensible and customizable
  - Easy to add support for new monitors and guests
  - No modifications required

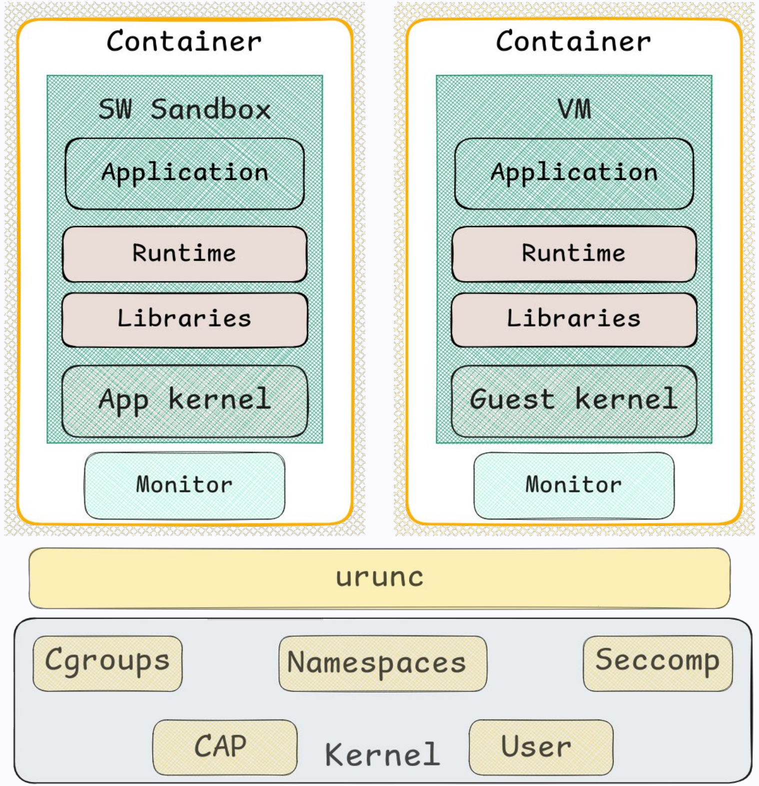
urunc

urunc: Key differences



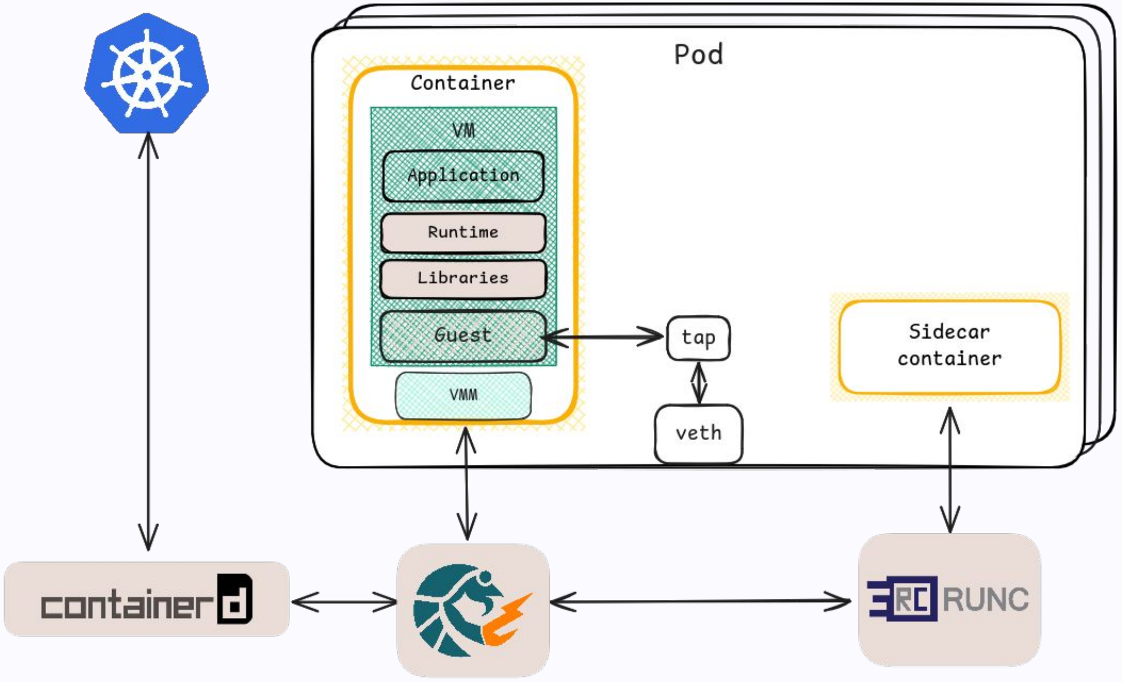
BSD over urunc

- Small rootfs containing only the application and packaged with a BSD kernel (unikernel/single app kernel)
- Urunc creates a Linux container for the VMM
- Start VMM and run a simple init to configure network/block
- Application runs as init



BSD in Kubernetes

- Create pod
- Spawn sidecar containers as typical Linux containers
- Spawn BSD microVM
- Separate user container from rest of containers in pod



Demo: Deploying

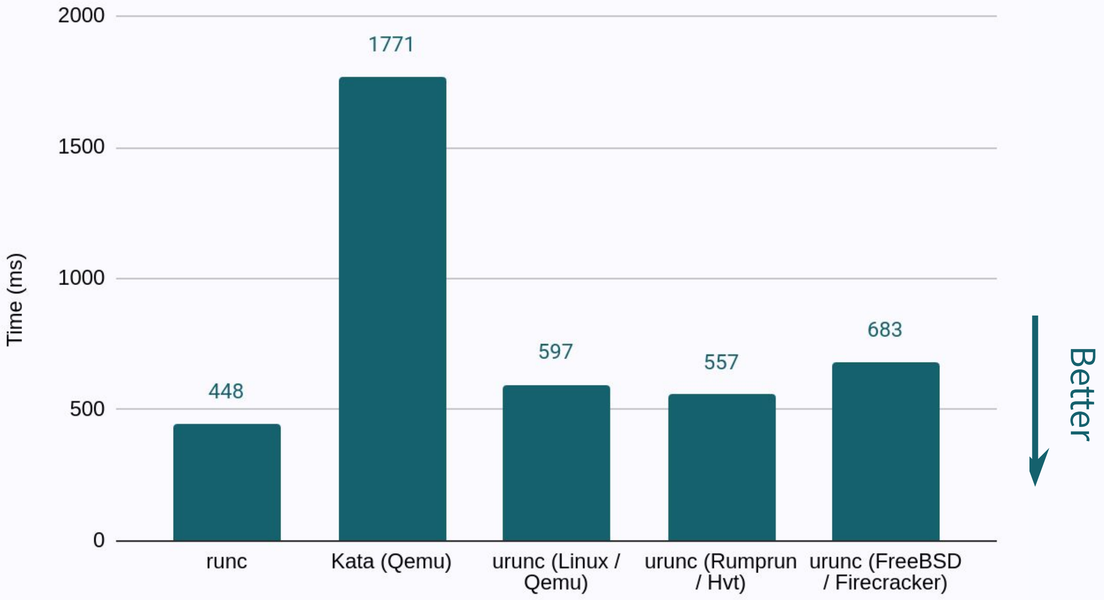
Use cases

- Deploy BSD-based microservices in Kubernetes
    - Manage them as any other service / pod
    - Seamless integration with the Kubernetes cluster
- BSD development in Linux
    - Spawn BSD environments in Linux
    - Docker build BSD (?)

Early evaluation

Start up time of service

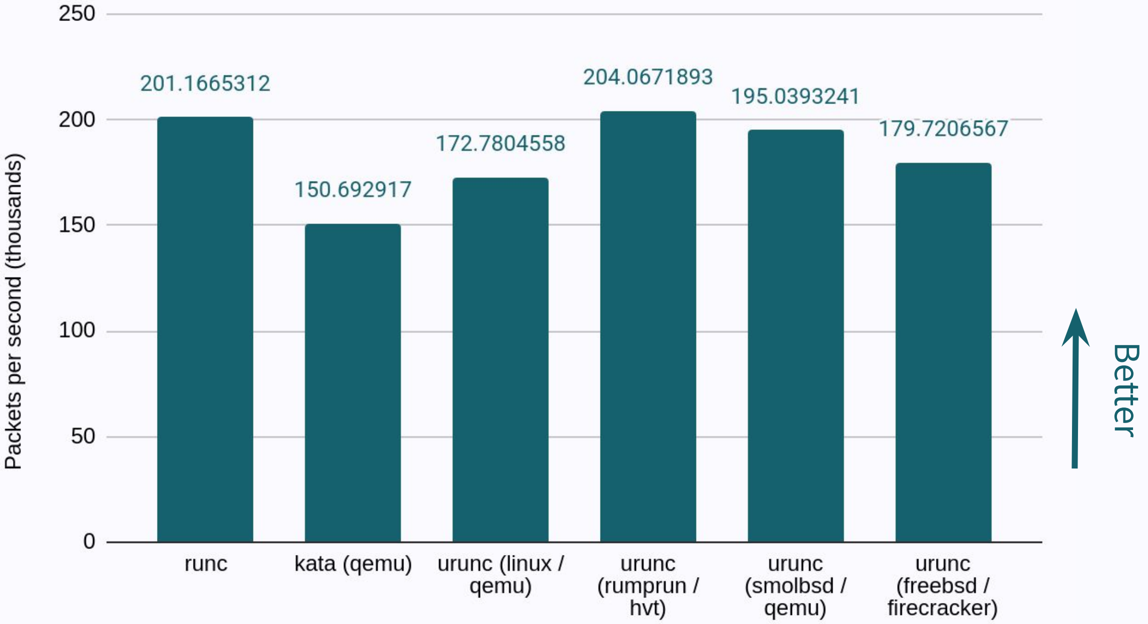
- Microbenchmark:
  - Server keeps timestamps
  - Client sends request to server and exits
  - Timestamp: deployment
  - Timestamp: 1st request
- Specs:
  - Single-node cluster
  - 4-core CPU
  - Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
  - 15 GB of RAM



FOSDEM -BSD, illumos, bhyve, OpenZFS devroom 2026

Start up time of service

- Microbenchmark:
  - Iperf3 server
  - Iperf3 client
- Specs:
  - Single-node cluster
  - 4-core CPU
  - Intel(R) Core(TM) i5-5300U CPU @ 2.30GHz
  - 15 GB of RAM



Future plan/ideas

- Further strip down the kernel and rootfs
  - Identify the parts that slowed down userland
- Integration with FreeBSD OCI images
  - Use the the image’s rootfs as the rootfs for VM and the process configuration
- Find ways to pass specific information inside the VM
  - Configure the execution environment inside the VM
- Docker build to create BSD rootfs (ufz/zfs) and kernels
  - Build BSD kernels and rootfs as containers
- Explore volumes integration with BSD-based filesystems
  - Declare and mount a BSD-based volume in a BSD microservice
- Rumprun maintenance
  - Vast majority is already maintained

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

Summary

- BSD has significant benefits for certain workloads
- Cloud infrastructure is hostile against BSD deployments
- Package BSD apps as unikernels or single application kernels
- Deploy BSD apps over urunc and manage them as any other Linux container

Summary

- BSD has significant benefits for certain workloads
- Cloud infrastructure is hostile against BSD deployments
- Package BSD apps as unikernels or single application kernels
- Deploy BSD apps over urunc and manage them as any other Linux container

- Github repository: https://github.com/urunc-dev/urunc
- Website & documentation: https://urunc.io
- Join urunc's channel in CNCF's slack workspace